Data Science and Digital Systems: The 3Ds of Machine Learning Systems Design

Neil D. Lawrence

March 28, 2019

Abstract

Machine learning solutions, in particular those based on deep learning methods, form an underpinning of the current revolution in "artificial intelligence" that has dominated popular press headlines and is having a significant influence on the wider tech agenda. Here we give an overview of the 3Ds of ML systems design: Data, Design and Deployment. By considering the 3Ds we can move towards *data first* design.

1 Introduction

There is a lot of talk about the fourth industrial revolution centered around AI. If we are at the start of the fourth industrial we also have the unusual honour of being the first to name our revolution before it's occurred.

The technology that has driven the revolution in AI is machine learning. And when it comes to capitalising on the new generation of deployed machine learning solutions there are practical difficulties we must address.

In 1987 the economist Robert Solow quipped "You can see the computer age everywehere but in the productivity statistics". Thirty years later, we could equally apply that quip to the era of artificial intelligence.

From my perspective, the current era is merely the continuation of the information revolution. A revolution that was triggered by the wide availability of the silicon chip. But whether we are in the midst of a new revolution, or this is just the continuation of an existing revolution, it feels important to characterize the challenges of deploying our innovation and consider what the solutions may be.

There is no doubt that new technologies based around machine learning have opened opportunities to create new businesses. When home computers were introduced there were also new opportunities in software publishing, computer games and a magazine industry around it. The Solow paradox arose because despite this visible activity these innovations take time to percolate through to *existing* businesses.

1.1 Brownfield and Greenfield Innovation

Understanding how to make the best use of new technology takes time. I call this approach, *brownfield innovation*. In the construction industry, a brownfield site is land with pre-existing infrastructure on it, whereas a *greenfield* site is where construction can start afresh.

The term brownfield innovation arises by analogy. Brownfield innovation is when you are innovating in a space where there is pre-existing infrastructure. This is the challenge of introducing artificial intelligence to existing businesses. Greenfield innovation, is innovating in areas where no pre-existing infrastructure exists.

One way we can make it easier to benefit from machine learning in both greenfield and brownfield innovation is to better characterise the steps of machine learning systems design. Just as software systems design required new thinking, so does machine learning systems design.

In this paper we characterise the process for machine learning systems design, convering some of the issues we face, with the 3D process: Decomposition[1], Data and Deployment.

We will consider each component in turn, although there is interplace between components. Particularly between the task decomposition and the data availability. We will first outline the *decomposition* challenge.

One of the most succesful machine learning approaches has been supervised learning. So we will mainly focus on *supervised learning* because this is also, arguably, the technology that is best understood within machine learning.

[1]In earlier versions of the Three D process I've referred to this as the design stage, but decomposition feels more appropriate for what the stage involves and that preserves the word design for the overall process of machine learning systems design.

2 Decomposition

Machine learning is not magical pixie dust, we cannot simply automate all decisions through data. We are constrained by our data (see below) and the models we use. Machine learning models are relatively simple function mappings that include characteristics such as smoothness. With some famous exceptions, e.g. speech and image data, inputs are constrained in the form of vectors and the model consists of a mathematically well behaved function. This means that some careful thought has to be put in to the right sub-process to automate with machine learning. This is the challenge of *decomposition* of the machine learning system.

Any repetitive task is a candidate for automation, but many of the repetitive tasks we perform as humans are more complex than any individual algorithm can replace. The selection of which task to automate becomes critical and has downstream effects on our overall system design.

2.1 Pigeonholing

The machine learning systems design process calls for separating a complex task into decomposable separate entities. A process we can think of as pigeonholing.

Some aspects to take into account are

1. Can we refine the decision we need to a set of repetitive tasks where input information and output decision/value is well defined?
2. Can we represent each sub-task we've defined with a mathematical mapping?

The representation necessary for the second aspect may involve massaging of the problem: feature selection or adaptation. It may also involve filtering out exception cases (perhaps through a pre-classification).

All else being equal, we'd like to keep our models simple and interpretable. If we can convert a complex mapping to a linear mapping through clever selection of sub-tasks and features this is a big win.

For example, Facebook have *feature engineers*, individuals whose main role is to design features they think might be useful for one of their tasks (e.g. newsfeed ranking, or ad matching). Facebook have a training/testing pipeline called FBLearner. Facebook have predefined the sub-tasks they are interested in, and they are tightly connected to their business model.

It is easier for Facebook to do this because their business model is heavily focused on user interaction. A challenge for companies that have a more diversified portfolio of activities driving their business is the identification of the most appropriate sub-task. A potential solution to feature and model selection is known as *AutoML* [Feurer et al., 2015]. Or we can think of it as using Machine Learning to assist Machine Learning. It's also called meta-learning. Learning about learning. The input to the ML algorithm is a machine learning task, the output is a proposed model to solve the task.

One trap that is easy to fall in is too much emphasis on the type of model we have deployed rather than the appropriateness of the task decomposition we have chosen.

Recommendation: Conditioned on task decomposition, we should automate the process of model improvement. Model updates should not be discussed in management meetings, they should be deployed and updated as a matter of course. Further details below on model deployment, but model updating needs to be considered at design time. This is the domain of AutoML.

The answer to the question which comes first, the chicken or the egg is simple, they co-evolve [Popper, 1963]. Similarly, when we place components together in a complex machine learning system, they will tend to co-evolve and compensate for one another.

To form modern decision making systems, many components are interlinked. We decompose our complex decision making into individual tasks, but the performance of each component is dependent on those upstream of it.

This naturally leads to co-evolution of systems, upstream errors can be compensated by downstream corrections.

To embrace this characteristic, end-to-end training could be considered. Why produce the best forecast by metrics when we can just produce the best forecast for our systems? End to end training can lead to improvements in performance, but it would also damage our systems decomposability and its interpretability, and perhaps its adaptability.

The less human interpretable our systems are, the harder they are to adapt to different circumstances or diagnose when there's a challenge. The trade-off between interpretability and performance is a constant tension which we should always retain in our minds when performing our system design.

3 Data

It is difficult to overstate the importance of data. It is half of the equation for machine learning, but is often utterly neglected. We can speculate that there are two reasons for this. Firstly, data cleaning is perceived as tedious. It doesn't seem to consist of the same intellectual challenges that are inherent in constructing complex mathematical models and implementing them in code. Secondly, data cleaning is highly complex, it requires a deep understanding of how machine learning systems operate and good intuitions about the data itself, the domain from which data is drawn (e.g. Supply Chain) and what downstream problems might be caused by poor data quality.

A consequence of these two reasons, data cleaning seems difficult to formulate into a readily teachable set of principles. As a result it is heavily neglected in courses on machine learning and data science. Despite data being half the equation, most University courses spend little to no time on its challenges.

3.1 The Data Crisis

Anecdotally, talking to data modelling scientists. Most say they spend 80% of their time acquiring and cleaning data. This is precipitating what I refer to as the "data crisis". This is an analogy with software. The "software crisis" was the phenomenon of inability to deliver software solutions due to increasing complexity of implementation. There was no single shot solution for the software crisis, it involved better practice (scrum, test orientated development, sprints, code review), improved programming paradigms (object orientated, functional) and better tools (CVS, then SVN, then git).

However, these challenges aren't new, they are merely taking a different form. From the computer's perspective software *is* data. The first wave of the data crisis was known as the *software crisis*.

3.1.1 The Software Crisis

In the late sixties early software programmers made note of the increasing costs of software development and termed the challenges associated with it as the "Software Crisis". Edsger Dijkstra referred to the crisis in his 1972 Turing Award winner's address [Dijkstra, 1972].

> The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.
>
> Edsger Dijkstra, The Humble Programmer

We can update Dijkstra's quote for the modern era.

> The major cause of *the data crisis* is that machines have become more interconnected than ever before. Data access is therefore cheap, but data quality is often poor. What we need is cheap high quality data. That implies that we develop processes for improving and verifying data quality that are efficient.
>
> There would seem to be two ways for improving efficiency. Firstly, we should not duplicate work. Secondly, where possible we should automate work.

What I term "The Data Crisis" is the modern equivalent of this problem. The quantity of modern data, and the lack of attention paid to data as it is initially "laid down" and the costs of data cleaning are bringing about a crisis in data-driven decision making. This crisis is at the core of the challenge of *technical debt* in machine learning [Sculley et al., 2015].

Just as with software, the crisis is most correctly addressed by 'scaling' the manner in which we process our data. Duplication of work occurs because the value of data cleaning is not correctly recognised in management decision making processes. Automation of work is increasingly possible through techniques in artificial intelligence, but this will also require better management of the data science pipeline so that data about data science (meta-data science) can be correctly assimilated and processed. The Alan Turing institute has a program focussed on this area, AI for Data Analytics.

Data is the new software, and the data crisis is already upon us. It is driven by the cost of cleaning data, the paucity of tools for monitoring and maintaining our deployments, the provenance of our models (e.g. with respect to the data they're trained on).

Three principal changes need to occur in response. They are cultural and infrastructural.

3.2 The Data First Paradigm

First of all, to excel in data driven decision making we need to move from a *software first* paradigm to a *data first* paradigm. That means refocusing on data as the product. Software is the intermediary to producing the data, and its quality standards must be maintained, but not at the expense of the data we are producing. Data cleaning and maintenance need to be prized as highly as software debugging and maintenance. Instead of *software* as a service, we should refocus around *data* as a service. This first change is a cultural change in which our teams think about their outputs in terms of data. Instead of decomposing our systems around the software components, we need to decompose them around the data generating and consuming components.[2] Software first is only an intermediate step on the way to be coming *data first*. It is a necessary, but not a sufficient condition for efficient machine learning systems design and deployment. We must move from *software orientated architecture* to a *data orientated architecture*.

3.3 Data Quality

Secondly, we need to improve our language around data quality. We cannot assess the costs of improving data quality unless we generate a language around what data quality means.

Data Readiness Levels [Lawrence, 2017] are an attempt to develop a language around data quality that can bridge the gap between technical solutions and decision makers such as managers and project planners. The are inspired by Technology Readiness Levels which attempt to quantify the readiness of technologies for deployment. They contain three grades of data readinees. Grade C data is *hearsay* data. Data that is purported to exist, but has not been electronically loaded into a computer system (e.g. an analysis package such as R or made available via a REST API). Moving data from Grade C to Grade B also involves legal and ownership issues. Once data can be loaded in it becomes Grade B. Grade B is available electronically, but undergoes a process of validation. Aspects like missing values, outlier representation, duplicate records are dealt with in Grade B. Grade B also has some of the characteristics of *exploratory data analysis* [Tukey, 1977]. Grade A is then data in context. Finally, at Grade A we consider the appropriateness of data to answer a particular question. In historical statistics Grade A data might be data that is ready for confirmatory data analysis. Many statisticians and machine learning researchers are used to only dealing with data at Grade A. Either because they work mainly with benchmark data or because data was actively collected with a particular question in mind. The major change for the era of data science is that so much data is available by *happenstance*.

3.4 Move Beyond Software Engineering to Data Engineering

Thirdly, we need to improve our mental model of the separation of data science from applied science. A common trap in current thinking around data is to see data science (and data engineering, data preparation) as a sub-set of the software engineer's or applied scientist's skill set. As a result we recruit and deploy the wrong type of resource. Data preparation and question formulation is superficially similar to both because of the need for programming skills, but the day to day problems faced are very different.
Recommendation: Build a shared understanding of the language of data readiness levels for use in planning documents, the costing of data cleaning, and the benefits of reusing cleaned data.

3.5 Combining Data and Systems Design

One analogy I find helpful for understanding the depth of change we need is the following. Imagine as an engineer, you find a USB stick on the ground. And for some reason you *know* that on that USB stick is a particular API call that will enable you to make a significant positive difference on a business problem. However, you also know on that USB stick there is potentially malicious code. The most secure thing to do would be to *not* introduce this code into your production system. But what if your manager told you to do so, how would you go about incorporating this code base?

The answer is *very* carefully. You would have to engage in a process more akin to debugging than regular software engineering. As you understood the code base, for your work to be reproducible, you should be documenting it, not just what you discovered, but how you discovered it. In the end, you typically find a single API call that is the one that most benefits your system. But more thought has been placed into this line of code than any line of code you have written before.

Even then, when your API code is introduced into your production system, it needs to be deployed in an environment that monitors it. We cannot rely on an individual's decision making to ensure the quality of all our systems. We need to create an environment that includes quality controls, checks and bounds, tests, all designed to ensure that assumptions made about this foreign code base are remaining valid.

This situation is akin to what we are doing when we incorporate data in our production systems. When we are consuming data from others, we cannot assume that it has been produced in alignment with our goals for our own systems. Worst case, it may have been adversarialy produced. A further challenge is that data is dynamic. So, in effect, the code on the USB stick is evolving over time.

Anecdotally, resolving a machine learning challenge requires 80% of the resource to be focused on the data and perhaps 20% to be focused on the model. But many companies are too keen to employ machine learning engineers who focus on the models, not the data.

[2]This is related to challenges of machine learning and technical debt [Sculley et al., 2015], although we are trying to frame the solution here rather than the problem.

A reservoir of data has more value if the data is consumable. The data crisis can only be addressed if we focus on outputs rather than inputs.

For a data first architecture we need to clean our data at source, rather than individually cleaning data for each task. This involves a shift of focus from our inputs to our outputs. We should provide data streams that are consumable by many teams without purification.

Recommendation: We need to share best practice around data deployment across our teams. We should make best use of our processes where applicable, but we need to develop them to become *data first* organizations. Data needs to be cleaned at *output* not at *input*.

4 Deployment

4.1 Continuous Deployment

Once the decomposition is understood, the data is sourced and the models are created, the model code needs to be deployed.

To extend our USB stick analogy further, how would we deploy that code if we thought it was likely to evolve in production? This is what datadoes. We cannot assume that the conditions under which we trained our model will be retained as we move forward, indeed the only constant we have is change.

This means that when any data dependent model is deployed into production, it requires *continuous monitoring* to ensure the assumptions of design have not been invalidated. Software changes are qualified through testing, in particular a regression test ensures that existing functionality is not broken by change. Since data is continually evolving, machine learning systems require 'continual regression testing': oversight by systems that ensure their existing functionality has not been broken as the world evolves around them. An approach we refer to as *progression testing* [Diethe et al., 2019]. Unfortunately, standards around ML model deployment yet been developed. The modern world of continuous deployment does rely on testing, but it does not recognize the continuous evolution of the world around us.

Recommendation: We establish best practice around model deployment. We need to shift our culture from standing up a software service, to standing up a *data as a service*. Data as a Service would involve continual monitoring of our deployed models in production. This would be regulated by 'hypervisor' systems[3] that understand the context in which models are deployed and recognize when circumstance has changed and models need retraining or restructuring.

Recommendation: We should consider a major re-architecting of systems around our services. In particular we should scope the use of a *streaming architecture* (such as Apache Kafka) that ensures data persistence and enables asynchronous operation of our systems.[4] This would enable the provision of QC streams, and real time dash boards as well as hypervisors.

Importantly a streaming architecture implies the services we build are *stateless*, internal state is deployed on streams alongside external state. This allows for rapid assessment of other services' data.

5 Conclusion

Machine learning has risen to prominence as an approach to *scaling* our activities. For us to continue to automate in the manner we have over the last two decades, we need to make more use of computer-based automation. Machine learning is allowing us to automate processes that were out of reach before.

We operate in a technologically evolving environment. Machine learning is becoming a key component in our decision making capabilities. But the evolving nature of data driven systems means that new approaches to model deployment are necessary. We have characterized three parts of the machine learning systems design process. *Decomposition* of the problem into separate tasks that are addressable with a machine learning solution. Collection and curation of appropriate *data*. Verificaction of data quality through data readiness levels. Using *progression* testing in our *deployments*. Continuously updating models as appropriate to ensure performance and quality is maintained.

[3]Emulation, or surrogate modelling, is one very promising approach to forming such a hypervisor. Emulators are models we fit to other models, often simulations, but the could also be other machine learning modles. These models operate at the meta-level, not on the systems directly. This means they can be used to model how the sub-systems interact. As well as emulators we shoulc consider real time dash boards, anomaly detection, mutlivariate analysis, data visualization and classical statistical approaches for hypervision of our deployed systems.

[4]These approaches are one area of focus for my own team's reasearch. A data first architecture is a prerequisite for efficient deployment of machine learning systems.

References

Tom Diethe, Tom Borchert, Eno Thereska, Borja Balle, and Neil D. Lawrence. Continual learning in practice. Technical report, arXiv, 3 2019. Presented at the NeurIPS 2018 workshop on Continual Learning.

Edsger W. Dijkstra. The humble programmer. *Communications of the ACM*, 15(10):859–866, Oct 1972.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In Corinna Cortes, Neil D. Lawrence, Daniel Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, Cambridge, MA, 2015.

Neil D. Lawrence. Data readiness levels. Technical report, arXiv, 5 2017.

Karl R. Popper. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Routledge, London, 1963. ISBN 0-415-04318-2.

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2503–2511. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf.

John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977. ISBN 0-201-07616-0.

Data Readiness Levels

Neil D. Lawrence
Amazon Research Cambridge and University of Sheffield

6th April 2017

Abstract

Application of models to data is fraught. Data-generating collaborators often only have a very basic understanding of the complications of collating, processing and curating data. Challenges include: poor data collection practices, missing values, inconvenient storage mechanisms, intellectual property, security and privacy. All these aspects obstruct the sharing and interconnection of data, and the eventual interpretation of data through machine learning or other approaches. In project reporting, a major challenge is in encapsulating these problems and enabling goals to be built around the processing of data. Project overruns can occur due to failure to account for the amount of time required to curate and collate. But to understand these failures we need to have a common language for assessing the readiness of a particular data set. This position paper proposes the use of data readiness levels: it gives a rough outline of three stages of data preparedness and speculates on how formalisation of these levels into a common language for data readiness could facilitate project management.

1 Introduction

The universal formula of machine learning is

$$\text{data} + \text{model} \rightarrow \text{prediction}$$

with a high quality of prediction being dependent on both good models and high quality data. Much of the focus on machine learning in the academic world of machine learning is on the quality of the model. This focus arises because data is normally from benchmarks or publicly available data sets, so performance in a task is improved by exercising control over model.

The greater interconnectedness of modern society and apparent ease with which we digitise or record data places us in a particular position. We need to develop a lot more control over the quality of our data. Whether it be in the manner in which we choose to collect, or how we choose to annotate. Currently the academic or empirical study of modeling (e.g. support vector machines, neural networks, Gaussian processes) is prominent in the education of the graduates we produce, but approaches for understanding the quality of data are less widely used.

2 Data Readiness Levels

In this position paper we introduce the idea of *data readiness levels*. Data readiness levels are designed to deal with a challenge for human cognitive information processing. It's difficult for us to reason about concepts when we haven't developed a language to describe them. The idea of data readiness levels is to correct this issue and make it easier for us to reason about the state of our data.

The challenges of data quality arise before modeling even starts. Both questions and data are badly characterized. This is particularly true in the era of Big Data, where one gains the impression that the depth of data-discussion in many decision making forums is of the form "We have a Big Data problem, do you have a Big Data solution?", "Yes, I have a Big Data solution." Of course in practice it also turns out to be a solution that requires Big Money to pay for because no one bothered to scope the nature of the problem, the data, or the solution.

Data scientists and statisticians are often treated like magicians who are expected to wave a model across a disparate and carelessly collated set of data and with a cry of 'sortitouticus' a magical conclusion is drawn. It is apt to think of it as ocean of data, to paraphrase "water, water everywhere and not a drop to drink", we have "data, data everywhere and not a set to process". Just as extracting drinkable water from the real ocean requires the expensive process of desalination, extracting usable data from the data-ocean requires a significant amount of processing.

For any data analyst, when embarking on a project, a particular challenge is assessing the quality of the available data, how much processing is required? This difficulty can be compounded when project partners do not themselves have a deep understanding of the process of data analysis. If partners are not data-savvy they may not understand just how much good practice needs to be placed in the curation of data to ensure that conclusions are robust and representative. Just as water quality should be measured before consumption, so should data quality.

When scoping a project, in most proposal documents, very scant attention is paid to these obstacles, meaning in practice the process of improving data quality is under appreciated and under resourced.

One difficulty is that the concept of "data", for many people, is somehow abstract and diffuse. This seems to mean that it is challenging for us to reason about. Psychologists refer to the idea of *vivid* information as information that is weighted more heavily in reasoning than non-vivid or *pallid information*. In this sense data seems to be rendered *vivid* to be properly accounted for in planning. This may relate to availability heuristics[1].

This abstract nature is also true of other terms, for example for many people the idea of "technology" is also similarly diffuse, it is pallid information (although I've never heard anyone remark "we have a big technology problem, we need a big technology solution"). Perhaps to deal with this challenge, in large scale projects, when deploying technology, we are nowadays guided to consider its *readiness stage*. Technology readiness levels arose in NASA [2]. The readiness of the technology is made manifest through a set of numbers which describe its characteristics:[1] is it lab tested only? Is it ready for commercialization? Is it merely conceptual?

No doubt there are pros and cons of such readiness levels, but one of the pros is that the manifestation of the technological readiness pipeline ensures that some thought is given to that process. The technology is rendered more vivid even through a shared characterization.

It would therefore seem very useful to have a scale to make *data readiness* manifest. This idea would allow analysts to encourage better consideration of the data collection/production and consolidation, with a set of simple questions, "And what will the data readiness level be at that point?". Or "How will that have progressed the data readiness?". Or to make statements, "we'll be unable to deliver on that integration unless the data readiness level is at least B3.."[2]

This paper aims to trigger a discussion in statistics and data science communities by proposing an initial set of descriptors for data readiness.

The initial proposal is that data readiness should be split into three *bands*. Each band being represented by a letter, A, B and C. These bands reflect stages of data readiness which would each likely have some sub-levels, so the best data would be A1 and the worst data might be C4. The aim here is to avoid being fine-grained too early. We therefore begin the discussion by focussing on three bands of data readiness.

[1] See appendix for examples of technology readiness level descriptions.

[2] The nanotechnology community has also looked at data readiness levels in this discussion document from the nanotechnology community in 2013. However, their scope doesn't seem to be general enough to deal with the challenges of data processing in domains beyond nanotechnology.

2.1 Band C

Band C is about the accessibility of a data set. The lowest sub-level of Band C (let's label it as C4) would represent a belief that the data may exist, but its existence isn't even verified. Signs that data is C4 might include statements like "The sales department should have a record of that." Or "The data should be available because we stipulated it in the software requirements." We might think of it as *hearsay* data. Data that you've heard about so you say it's there. Problems with hearsay data might include

- whether it really is being recorded
- the format in which it's being recorded (e.g. handwritten log book, stored in PDF format or old machine formats)
- privacy or legal constraints on the accessibility of the recorded data, have ethical constraints been alleviated?
- limitations on access due to topology (e.g. the data's distributed across a number of devices)

So when we are first told a data set exists, when we have hearsay data, then it is at band C4. For data to arrive at C1, then it would have all these considerations dealt with.

When data arrives at C1 it's ready to be loaded into analysis software, or it can be made available for others to access (e.g. via a data repository such as OpenML [3]). It is machine readable and ethical procedures for data handling have been addressed. Bringing data to C1 is often a significant effort itself involving many lines of bespoke software and human understanding of systems, ethics and the law.

Some parts of Band C are sometimes referred to as "data munging" or "data wrangling", but those aren't the only components of this band, there are additional challenges such as ethical and legal that need to be resolved.

2.2 Band B

Band B is about the faithfulness and representation of the data. Now that it's loaded into the software, is what is recorded matching what is purported to be recorded? How are missing values handled, what is their encoding? What is the noise characterization (for sensors) or for manual data are there data entry errors? Are any scientific units correctly formulated?

Tukey's approach of "Exploratory Data Analysis" also fits within Band B. Visualizations of the data should be carried out to help render the data vivid and to ensure decision makers, who may not be data aware, can become involved in the analysis process. Decision makers (e.g. project managers, or the client) should also begin to get a sense of the limitations of their data set through appropriate visualisation.

As part of Band B the characteristics of the collection process should also be verified, was data collection randomized, is it biased in any particular way?

Other things to watch for at this stage include:

1. If the data has been agglomerated at some point (for example, for privacy) how were missing values dealt with before agglomeration? If they weren't dealt with then that entire section of the data may be invalidated
2. If the data has been through a spreadsheet software, can you confirm that no common spreadsheet analysis errors were made? For example, was a column or columns accidentally perturbed (e.g. through a sort operation that missed one or more columns)? Or was a gene name accidentally converted to a date?

By the end of Band B, when data is B1, a broad idea of limitations in the data should be present in the expert's mind. Data at C4 was hearsay data, someone heard the data existed and they said what they thought it might be good for. At B1 the analyst knows how faithful the data is to that description. This is the significant challenge for a data scientist. What people believe they have in their data versus what's actually there. Only at the end of Stage B would the analyst begin to have an intuition about what my really be possible with the data set. Getting to this point is often the most expensive part of the project, but we do not yet have good methods to guage progress, or share the status of a particular data set.

2.3 Band A

Band A is about data in context. It is at Band A that we consider the appropriateness of a given data set to answer a particular question or to be subject to a particular analysis.

The context must be defined. For example OpenML [3] defines *tasks* associated with data sets. A data set can only be considered in Band A once a task is defined. A task could be "Use the data to predict a user preference" or "Use this data to prove the efficacy of a drug" or "Use this data verify the functioning of our rocket engine".

Once data has been considered alongside a task and any remedial steps have been taken, then the data is in A1 condition. It is ready to be deployed in the context given and it can be used to make predictions with the data.

Because A1 is about data in context, it is possible for a data set to be A1 for one question (e.g. predicting customer churn) but only B1 for a different question (e.g. predicting customer susceptability to a particular special offer). So the definition of the context or task is an important pre-requisite for this band.

To bring a data set up to A1, there may be a need for significant annotation of the data by human expert. There may be a realisation that new data needs to be actively collected to get the answers required. In that sense, Band A has some characteristics of a classical statistical analysis where the question would normally preceed the data collection. It is in Band A where you should be carefully thinking about the statistical design because it is only when you have the question you wish to answer that you can really unpick how your data may be biased or what information is missing.

3 The Analysis Pipeline

A common mistake in data analysis is to not acknowledge the different processes above. They have different pre-requisites and require different skill sets to carry out. However, this path cannot be completely disconnected. Anyone performing an analysis at Band A also needs to be intimately familiar with the collection process so that any biases in data collection can be understood. Sharing information about decisions taken at Bands B and C will also be critical to achieving a good result.

What happens if we bring two data sets together to form a new data set? Some assesment of data readiness would still need to be performed on the new data set, even if the two other data sets had already been assessed. Why? Well, one example is that there may be ethical issues with combining data sets. For example, medical data can be easily deanonymized if it is combined with other related data sets. That would mean that some assesment would need to be made at Band C to see if it is ethically responsible to combine the two data sets in the same analysis package. Futher assesments would also need to be done at Band B to understand differences in collection protocols that might make the two data sets hard to combine.

3.1 Potential Results

The idea of these levels is to increase the accountability of the process and allow the nature of the data to be manifest. With data readiness levels in place you can now imagine conversations that would include statements like the following:

> Be careful, that department claims to have made 10,000 data sets available, but we estimate that only 25% of those data sets are available at C1 readiness.

The cost of bringing the data to C1 would be prohibitive for this study alone, but the company-wide data audit is targeting this data to be C1 by Q3 2017 which means we can go ahead and recruit the statisticians we need.

The project failed because we over recruited statistical expertise and then deployed them on bringing the data set to C1 readiness, a job that would have been better done by building up our software engineering resource.

What's the data readiness level? My team will be ineffectual until it's B1 and at the moment I see no provision in the plan for resource to bring it there.

We estimate that it's a $100,000 dollar cost to bring that data to B1, but we can amortize that cost across four further studies that also need this data.

I gave them the data readiness levels to go through and they realized they hadn't yet got the necessary ethics approval for sharing the zip codes. We'll revisit when they've got through that and can assure us they can share a C1 set.

While their knowledge of the latest methodologies wasn't as good as I'd hoped, the candidate had a lot of experience of bringing data from C1 to B1, and that's a skill set that we're in dire need of.

The project came in under budget because they found a team with experience of getting a closely related data set to A1. Many of the associated challenges were the same and they could even reuse some of that team's statistical models.

4 Case Studies

How useful would data readiness levels be? That's difficult to give a quantitative answer to, large scale analysis of the scale of this problem requires meta data science, i.e. the study of data science itself through acquiring data about data science. That would be a very worthwhile endeavour. In the absence of quantitative information, we provide two anecdotal case studies below. The first is on the cahllenge of extracting data from a popular machine learning conference proceedings.

4.1 Proceedings of Machine Learning Research

The Proceedings of Machine Learning Research[3] (PMLR) were begun in 2006 (as JMLR Workshop and Conference Proceedings) to provide a convenient way to publish machine learning conference proceedings without the overhead of a conventional publisher.

There are now 69 volumes of proceedings published and planned. They contain over 3,198 papers. The original website was manually curated to mimic the JMLR website, but since Volume 26 an automated proceedings production process which relies on editors providing a zip file of PDFs and a bibtex bibliography reference file (referred to below as 'bib files') specifying author names, abstracts and titles has been used.

In early 2017, as part of a rebranding process, the original website was moved to github and a new process for receiving proceedings and publishing was set up. For the rebranding the old web-sites needed to be scraped, converted to bib files.

A key aim in the rebanding was to make abstracts and titles easily available for analysis with an initial target language of Python.

This entire process can be seen as taking the data readiness of website data from C4 to C1. In other words, get to the point where the data could be loaded into an analysis software, our target here was to load it into the pandas framework within Python.

[3] http://proceedings.mlr.press

The *original* plan was to complete the work in some idle hours at the weekend. The actual work took much longer than projected. The major github commits of code along with a description of the effort involved are listed below.

1. An initial two day effort to create bibliography files for the first 26 volumes which were published before the publication process was first automated. Github commit: https://github.com/mlresearch/papersite/commit/daa51a
2. A three day effort to convert the proceedings to a new format website. Much of this work was on the website presentation, but part was on data curation. Github commit: https://github.com/mlresearch/papersite/commit/81
3. A three evening effort to tidy the resulting data so it could load into Python pandas via web download. Github commit: https://github.com/mlresearch/papersite/commit/81d7a0556948281d11d8f0652ecdca61005c4318

The result is now available via github as a short jupyter notebook: https://github.com/sods/ods/blob/master/notebooks/pods Data loading is now possible with a single library call.

```
import pods
data = pods.datasets.pmlr()
```

where the pods library is available from https://github.com/sods/ods/.

This data has therefore successfully transferred from the start of C (e.g. C4, "Hearsay data") to the end of C (e.g. C1, data loaded into analysis software).

The data set only contains 3,198 data points, it was all available in electronic form. There were no issues around privacy or intellectual property but work still took approximately six working days. Much of that work was laborious, but it still involved well qualified understanding of computer software, e.g. regular expressions, scripting languages, libraries for downloading from the internet, github etc.. Alongside that work a new format web page was also provided (http://proceedings.mlr.press). Indeed that could be argued as a consequence of the data tidy up. Naturally the hope is that future volumes will be more cleanly added to this data set.

4.2 Data Wrangling Snafus

Some snafus that occurred in the data wrangling of the Proceedings of Machine Learning Research.

Each bibliography file is provided by the editors of the relevant volume. Because they each used different approaches to generate the bib files, there were different issues with each bib file.

- Many of these bib files will have been produced from conference management software (e.g. CMT or Easychair). That means the original information source is often likely to be author provided. Many authors seemed to paste their abstracts from the PDFs of their papers into this software. This meant that abstracts contained ligatures. A ligature is a single typeset unit such as 'fi', which comes about when the previous letter runs into the second, the German 'Sz', ß, is an example of a ligature that in the end became a letter. Unfortunately the ligatures did not paste as unicode, but as escaped characters which the python yaml library was unable to read when presented as data.
- Papers are mostly written in Latex, so many of the abstracts contain Latex commands. But these commands in the abstracts or title are intepreted as escaped characters in yaml files. Additional 'escaping' was needed for these commands.
- Author names containing accents were oftentimes corrupted, perhaps due to differences in representation in the original files.

Each of the snafus above can be resolved by robust coding, but there is normally a trade off between producing the data (just getting it done quick) and producing the most robust code (ensuring the result is high quality and reusable). The right operating point for the trade off is driven by the scale of the data set. It also requires experience to judge and is dependent on the coding skills of the data engineer.

This is the work at the pit face of data mining, it's difficult to estimate the time it will take, and yet it is normally a critical dependency in any machine learning project. For each of these snafus, better planning at early stage analysis could have saved time in manual correction of the data later. But doing such planning well requires experience and an understanding of best practice. We are not associating enough value with this experience, and therefore we continue to be tripped by snafus when preparing data.

4.2.1 How Data Readiness Levels could have helped

This is one of the first projects I undertook after conceiving of data readiness levels, but in reflection I think I still did not take enough of the ideas seriously enough. Because I underestimated the time to be spent on manual curation after the initial scrape of the data, I did not put enough effort into robust code for dealing with ligatures and other unusual character codes.

There were several moments where, in retrospect, I should have refactored my processing code. But since I was driven by the desire to complete the project, particularly given I'd severely underestimated how long it would take, I chose to push forward. One interesting question is how ideas from software engineering, such as agile development philosophies, could have helped in making myself more aware of these errors.

4.3 Disease Monitoring in Uganda

The second case study refers to work in collaboration with the University of Makerere and UN Global Pulse in Kampala, Uganda in the prediction of disease outbreak [4]. Specifically, our original interest was understanding the spatial correlations of malarial disease in Uganda, and their interactions with other measures such as NDVI, rainfall, altitude etc. Our effort was a countrywide effort, but simultaneous to our work was an international effort in the Malaria Atlas Project.[4]

We constructed spatial models based on Gaussian processes that were designed to detect and use the correlations between the different covariates. The work was done through two PhD students' thesis, Martin from Kampala, and Ricardo from the UK. The work in Kampala mainly drove the data collection and collation. In particular, interpretation of satellite images as NDVI, alignment of the rainfall and altitude maps. The data collation in itself took probably around 70-80% of the work on the project. Martin working on it full time, and Ricardo assisting and working on both modeling and data munging. Ricardo visited Uganda twice, initially for a scoping visit and later for a longer collaboration visit in the processing of the data during his PhD. These visits were to the University of Makerere where Martin was working.

The malaria incidence data was drawn from Health Management Information System (HMIS) data. For privacy purposes, original data was not available to us, but rather it had been aggregated across certain regions.

One key challenge in the data munching is that the administrative areas (in Uganda known as districts) change. One administrative area can divide into two. When this happens the history of the two districts needs to be divided across the two areas. This presented a missing data problem that Ricardo also spent a deal of time addressing during his thesis.

By the time of submission of the thesis, Ricardo had recovered a negative result, he couldn't show spatial correlation among the districts. This was disappointing as it had been one of the main motivations of that thesis work. Nevertheless, even without the spatial correlation there was interest in deploying the system by the Ugandan ministry of health for disease outbreak prediction.

Ricardo's expertise was such that he was able to then spend 3 months in Africa, this time at UN Global Pulse, where Martin and his supervisor were now working, rather than Makerere. This was to help with the implementation of early warning systems based on the model. These early warning systems did not require the spatial correlation that hadn't worked during the thesis.

[4] http://www.map.ox.ac.uk/

During the period at the UN, Ricardo and Martin were able to work directly with disaggregated data. The UN had permission to see the raw health center results, whereas the University of Makerere did not. As a result, they found that aggregation we had previously been working on had been performed across data containing missing values. Ricardo and Martin reworked the model to deal with missing values and recovered the expected spatial correlation.

In other words, due to a data processing error at Band B, a negative result was obtained at Band A. A lack of documentation, or a lack of asking the right questions, led to an oversight on this error.

The models are now deployed for disease outbreak prediction across Uganda, where they are combined with knowledge of population movement from mobile phone data to trigger interventions.

The lessons learned were the following.

1. In data where we had direct access the amount of work required to align, and the number of design decisions we made in the summarization dominated the project.
2. There was still data that fell out of our control due to confidentiality reasons. Despite the work we'd done on our own data, which would have allowed us to infer that other design decisions would have been made by those that were aggregating the data, we failed to ask the right questions of the data providers. Indeed, those data providers may not have even known the answers. There was no substitute for directly looking at the data.
3. Although we began the project with one set of goals in mind: understanding the correlation between different factors in malaria, our outcome: disease prediction for early intervention, was quite different. Happenstance outcomes need to be accommodated in project goals. They arise particularly around Band B of the process.

4.3.1 How Data Readiness Levels could have helped

Being aware of data readiness levels would have done three things. Firstly, we would have estimated better how much time that Martin would require in data munging from satellite images etc. Secondly, we would have questioned the process by which the HMIS data was being presented to us, and what stage of readiness it was at, and how it got there. These are broadly questions that sit in Band C and B. Thirdly, awareness of the transition from Band B to Band A (data moving to its context) would have made us realize that the question may well evolve and be more responsive to that outcome. In the end this move was driven by the shift in collaboration from Makerere to the UN.

Ricardo has continued to work in this area, and one focus of his new UCSF Global Health research group has been to clean up existing data and make it available as geospatial layers for other groups to use.

5 Conclusion

Machine learning researchers probably didn't enter the field to do project management, but it may be that many failings on large data projects are associated with a failure to provision resource for the challenges involved in preparing our data, rather than a failing in the algorithmics of the system. Costs of data curation are often underestimated and those who do the work in Band C and Band B are very often undervalued.

Data readiness levels highlight the different skill sets required in each stage of analysis, from software engineer, to data-munger, to data scientist to machine learning scientist.

Some consensus about such levels would help organizations (and their managers, financial controllers) quantify the value associated with data and allocate resource correctly to developing data sets that are robust and representative. A well conducted data analysis will lead to a good customer experience, but by the same token badly waste resources and give a poor customer experience.

This paper has outlined the basis for a partial solution to these problem, but to deploy in practice we would need consensus around data readiness levels and how they should be deployed. The emerging field of data science is the ideal domain to explore the utility of these ideas, evolve their specification and begin to properly account for the value of well curated data.

For convenient reference below we show examples of *technology* readiness levels from the Department of Defense (sourced from Wikipedia. For *Data* Readiness we are proposing to start with the three bands described above. Technology readiness relies entirely on a numbered system.

[1] T. Gilovich, D. Griffin, and D. Kahneman, Eds., *Heuristics and biases: The psychology of intuitive judgment*. Cambridge University Press, 2002.

[2] J. Banke, "Technology readiness levels demystified," NASA, 2010.

[3] J. Vanschoren, J. N. Rijn, and B. Bischl, "Taking machine learning research online with OpenML," in *Proceedings of the 4th international workshop on big data, streams and heterogeneous source mining: Algorithms, systems, programming models and applications*, 2015, vol. 41, pp. 1–4.

[4] R. Andrade-Pacheco, M. Mubangizi, J. Quinn, and N. Lawrence, "Monitoring short term changes of infectious diseases in uganda with gaussian processes," in *Advanced analysis and learning on temporal data: First ecml pkdd workshop, aaltd 2015, porto, portugal, september 11, 2015, revised selected papers*, A. Douzal-Chouakria, J. A. Vilar, and P.-F. Marteau, Eds. Cham: Springer International Publishing, 2016, pp. 95–110.

A Exemplar Technology Readiness Levels

1. Basic principles observed and reported

   Lowest level of technology readiness. Scientific research begins to be translated into applied research and development (R&D). Examples might include paper studies of a technology's basic properties.

2. Technology concept and/or application formulated

   Invention begins. Once basic principles are observed, practical applications can be invented. Applications are speculative, and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies.

3. Analytical and experimental critical function and/or characteristic proof of concept

   Active R&D is initiated. This includes analytical studies and laboratory studies to physically validate the analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative.

4. Component and/or breadboard validation in laboratory environment

   Basic technological components are integrated to establish that they will work together. This is relatively "low fidelity" compared with the eventual system. Examples include integration of "ad hoc" hardware in the laboratory.

5. Component and/or breadboard validation in relevant environment

   Fidelity of breadboard technology increases significantly. The basic technological components are integrated with reasonably realistic supporting elements so they can be tested in a simulated environment. Examples include "high-fidelity" laboratory integration of components.

6. System/subsystem model or prototype demonstration in a relevant environment

   Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness. Examples include testing a prototype in a high-fidelity laboratory environment or in a simulated operational environment.

7. System prototype demonstration in an operational environment.

   Prototype near or at planned operational system. Represents a major step up from TRL 6 by requiring demonstration of an actual system prototype in an operational environment (e.g., in an aircraft, in a vehicle, or in space).

8. Actual system completed and qualified through test and demonstration.

   Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of true system development. Examples include developmental test and evaluation (DT&E) of the system in its intended weapon system to determine if it meets design specifications.

9. Actual system proven through successful mission operations.

   Actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation (OT&E). Examples include using the system under operational mission conditions.

Defense Science Board

SUMMER STUDY ON AUTONOMY

JUNE 2016

Report of the

Defense Science Board

Summer Study on

Autonomy

June 2016

Office of the Under Secretary of Defense

for Acquisition, Technology and Logistics

Washington, D.C. 20301-3140

This report is a product of the Defense Science Board (DSB).

The DSB is a Federal Advisory Committee established to provide independent advice to the Secretary of Defense. Statements, opinions, conclusions, and recommendations in this report do not necessarily represent the official position of the Department of Defense (DoD). The DSB Summer Study on Autonomy completed its information gathering in August 2015. The report was cleared for open publication by the DoD Office of Security Review on June 1, 2016

This report is unclassified and cleared for public release.

OFFICE OF THE SECRETARY OF DEFENSE
3140 DEFENSE PENTAGON
WASHINGTON, DC 20301–3140

DEFENSE SCIENCE BOARD

June 10, 2016

MEMORANDUM FOR UNDER SECRETARY OF DEFENSE FOR ACQUISITION, TECHNOLOGY & LOGISTICS

SUBJECT: Final Report of the Defense Science Board (DSB) Summer Study on Autonomy

I am pleased to forward the final report of the DSB Summer Study on Autonomy. This report offers important recommendations to identify the science, engineering, and policy problems that must be solved to permit greater operational use of autonomy across all warfighting domains.

The study focused on three areas: institutional and enterprise strategies to widen the use of autonomy; approaches to strengthening the operational pull for autonomous systems; and an approach accelerate the advancement of the technology for autonomy applications and capabilities. The study concluded that action is needed in all three areas to build trust and enable the most effective use of autonomy for the defense of the nation.

This report provides focused recommendations to improve the future adoption and use of autonomous systems. Recommendations also include 10 example projects intended to demonstrate the range of benefits of autonomy for the warfighter. The study also provides thoughts on how to expand the available technology for the use of autonomy for defense through several innovative technology stretch problem challenges.

I fully endorse all of the recommendations contained in this report and urge their careful consideration and soonest adoption.

Craig Fields
Chairman

DEFENSE SCIENCE BOARD

OFFICE OF THE SECRETARY OF DEFENSE
3140 DEFENSE PENTAGON
WASHINGTON, DC 20301–3140

June 9, 2016

MEMORANDUM FOR CHAIRMAN, DEFENSE SCIENCE BOARD

SUBJECT: Final Report of the Defense Science Board (DSB) Summer Study on Autonomy

The final report of the Defense Science Board 2014 Summer Study on Autonomy is attached. In accordance with its terms of reference, the study reviewed the applicability of autonomy across a broad array of DoD missions and concluded that there are both substantial operational benefits and potential perils associated with its use.

The study was informed by briefings describing a sampling of related DoD programs spanning the spectrum from deployed capabilities to research investments; relevant efforts in the commercial sector; and international activities. While evident that the DoD is moving forward in the employment of autonomous functionality, it is equally evident that the pull from diverse global markets is accelerating the underlying tech base and delivering high-value capabilities at a much more rapid pace.

The study provides recommendations aligned with three over-arching vectors:

- Accelerating DoD's adoption of autonomous capabilities
- Strengthening the operational pull for autonomy
- Expanding the envelope of technologies available for use on DoD missions

The first vector focuses on enterprise-wide recommendations that target barriers to increased operational use of autonomy. In providing recommendations the study focused on issues including the need to build trust in autonomous systems while also improving the trustworthiness of autonomous capabilities, and identified a number of enablers to align RDT&E processes to more rapidly deliver autonomous capabilities to DoD missions. The study concluded that action on this set of interdependent enterprise-wide recommendations is of far greater importance—and urgency—than the implementation of any single program of record.

The study observed that autonomy can deliver value by mitigating operational challenges including:

OFFICE OF THE SECRETARY OF DEFENSE
3140 DEFENSE PENTAGON
WASHINGTON, DC 20301–3140

DEFENSE SCIENCE BOARD

- Rapid decision-making
- High heterogeneity and/or volume of data
- Intermittent communications
- High complexity of coordinated action
- Danger of mission
- High persistence and endurance

Given the current budget environment, the study opted not to recommend major new programs. Instead, to strengthen the operational pull for autonomy, the study recommends a set of experiments/prototypes that would demonstrate clear operational value across these operational challenges. The recommended projects are intended also to serve as pilots to help refine and institutionalize the enterprise-wide recommendations.

Finally, given that commercial market drivers are spawning rapid advances in the underlying tech base, the report recommends that DoD take steps to engage non-traditional R&D communities in novel ways to both speed DoD's access to emerging research results and identify areas in which additional DoD investment is needed to fully address DoD missions.

While difficult to quantify, the study concluded that autonomy—fueled by advances in artificial intelligence—has attained a 'tipping point' in value. Autonomous capabilities are increasingly ubiquitous and are readily available to allies and adversaries alike. The study therefore concluded that DoD must take immediate action to accelerate its exploitation of autonomy while also preparing to counter autonomy employed by adversaries.

Dr. Ruth David
Co-Chairman

Dr. Paul Nielsen
Co-Chairman

TABLE OF CONTENTS

List of Tables, Figures, and Boxes in the Report

Acronyms and Abbreviations

3D	three-dimensional
A2/AD	anti-access and area denial
AAR	air-to-air refueling
ACO	airspace control order
ACRS	autonomous cyber resilient systems
AFRL	Air Force Research Laboratory
AEODRS	Advanced EOD Robotics System
AFOSR	Air Force Office of Scientific Research
AI	artificial intelligence
ALLOREQ	allotment request
ALLOT	allotment
AOD	air operations directive
APU	auxiliary power unit
ARCIC	Army Capabilities Integration Center
ARGUS-IS	Autonomous Real-Time Ground Ubiquitous Surveillance Imaging System
ASA(ALT)	Assistant Secretary of the Army for Acquisition, Logistics, and Technology
ASD(R&E)	Assistant Secretary of Defense for Research & Engineering
ATO	Air Tasking Order
ATR	automatic target recognition
BGP	border gateway protocol
C2	command and control
CASCOM	Combined Arms Support Command
CCTV	closed circuit television
CERDEC	Communications-Electronics Research, Development and Engineering Center
CIA	Central Intelligence Agency
COI	community of interest
CONOPs	concepts of operation
CRASH	Clean-Slate Design of Resilient, Adaptive, Secure Hosts
DARPA	Defense Advanced Research and Projects Agency
DCA	defensive counter air
DIA	Defense Intelligence Agency
DISA	Defense Information Systems Agency
DLA	Defense Logistics Agency
DNS	domain name service
DoD	Department of Defense
DOMEX	document and media exploitation
DOT&E	Director of Operational Test and Evaluation
DOTMLPF	Doctrine, organization, training, materiel, leadership and education, personnel, and facilities
DRFM	digital radio frequency memory
DSB	Defense Science Board
DTCWC	Dynamic Time Critical Warfighting Capability
DTE	Office of Developmental Test and Evaluation
EMBERS	Early Model Based Event Recognition using Surrogates

EO	electro-optical
EW	electronic warfare
EXCOM	executive committee
FAA	Federal Aviation Administration
FARP	forward arming and refueling point
FBI	Federal Bureau of Investigation
FFRDC	federally funded research and development center
GPS	global positioning system
IARPA	Intelligence Advanced Research and Projects Agency
IC	intelligence community
ICEWS	Integrated Conflict Early Warning System
IED	improvised explosive device
IHMC	Institute for Human and Machine Cognition
IoT	Internet of Things
IP	Internet protocol
IPA	Intergovernmental Personnel Act
ISR	intelligence, surveillance, reconnaissance
JAOP	joint air operations plan
JIDA	Joint Improvised-Threat Defeat Agency
JIEDDO	Joint Improvised Explosive Device Defeat Organization
JIPTL	joint integrated prioritized target list
JTCB	joint targeting coordination board
KI/CAS	killbox interdiction/close air support
LO/CLO	Low Observable and Counter Low Observable
M&S	modeling and simulation
MAAP	Master Air Attack Plan
MCCDC	Marine Corps Combat Development Command
MCM	mine countermeasures
MDAR	Mobile Detection Assessment and Response system
MRO	maintenance repair overhaul
MTRS	Mobile Tactical Robotic System
NAVSUP	Navy Supply Systems Command
NIPRNet	Non-classified Internet Protocol (IP) Router Network
NSA	National Security Agency
OASD(R&E)	Office of the Assistant Secretary of Defense for Research and Engineering
OCA	offensive counter air
ONR	Office of Naval Research
OSI	open source indicators
OTI	Office of Technical Intelligence
PCPAD-X	Planning & Direction, Collection, Processing & Exploitation, Analysis & Production, and Dissemination Experimentation
PEO-LCS	Program Executive Office Littoral Combat Ships
PNT	positioning, navigation, and timing
RAM	random access memory
R&D	research & development
REMUS	Remote Environmental Monitoring UnitS
RF	radio frequency

RFID	RF identification
ROE	rules of engagement
SEAD	suppression of enemy air defenses
SIGINT	signals intelligence
SMART	self-monitoring, analysis, and reporting technology
SPAWAR	Space and Naval Warfare Systems Command
STIX	Structured Threat Information Expression
TACDOMEX	tactical document and media exploitation
TACE	Test Automation Center of Excellence
TAXII	Trusted Automated eXchange of Indicator Information
T&E	test and evaluation
TRADOC	United States Army Training and Doctrine Command
TRL	technical readiness level
TRMC	Test Resource Management Center
TTPs	tactics, techniques, and procedures
UARC	University Affiliated Research Center
UAS	unmanned aircraft system
UA	unmanned aircraft
UGV	unmanned ground vehicle
UOES	user operational evaluation system
USCYBERCOM	United States Cyber Command
USD(AT&L)	Under Secretary of Defense (Acquisition, Technology, and Logistics)
USD(I)	Under Secretary of Defense for Intelligence
USD(P&R)	Under Secretary of Defense for Personnel & Readiness
USSOCOM	United States Special Operations Command
USV	unmanned surface vehicle
UUV	unmanned undersea vehicle
V&V	verification and validation

Executive Summary

At the request of the Under Secretary of Defense for Acquisition, Technology, and Logistics (USD(AT&L)), the Defense Science Board (DSB) conducted a study on the applicability of autonomy to Department of Defense (DoD) missions. The study concluded that there are both substantial operational benefits and potential perils associated with the use of autonomy.

Autonomy delivers significant military value, including opportunities to reduce the number of warfighters in harm's way, increase the quality and speed of decisions in time-critical operations, and enable new missions that would otherwise be impossible. Autonomy is by no means new to the DoD. Fielded capabilities demonstrate ongoing progress in embedding autonomous functionality into systems, and many development programs already underway include an increasingly sophisticated use of autonomy.

Autonomy also delivers significant value across a diverse array of global markets. Both enabling technologies and commercial applications are advancing rapidly in response to market opportunities. Autonomy is becoming a ubiquitous enabling capability for products spanning a spectrum from expert advisory systems to autonomous vehicles. Commercial market forces are accelerating progress, providing opportunities for DoD to leverage the investments of others, while also providing substantial capabilities to potential adversaries.

This study concluded that DoD must accelerate its exploitation of autonomy—both to realize the potential military value and to remain ahead of adversaries who also will exploit its operational benefits.

Major recommendations

The issue of trust is core to DoD's success in broader adoption of autonomy. On the one hand, an autonomous system must be designed to operate in a trustworthy fashion with respect to the missions for which it was designed. On the other hand, an autonomous system must be designed so that humans (and/or machines) can straightforwardly determine whether, once it has been deployed, it is operating reliably and within its envelope of competence — and, if not, that appropriate action can be taken. Establishing trustworthiness at design time and providing adequate capabilities so that inevitable variations in operational trustworthiness can be assessed and dealt with at run time is essential, not only for operators and commanders, but also for designers, testers, policymakers, lawmakers, and the American public. The broad topic of trust shaped many of the recommendations that follow.

The first set of recommendations focuses on accelerating DoD's adoption of autonomous capabilities and includes:

- Tackling the engineering, design, and acquisition challenges
- Mitigating cyber issues introduced by increasingly autonomous and networked systems
- Creating new test and evaluation and modeling and simulation paradigms
- Integrating technology insertion, doctrine, and concepts of operations

- Developing an autonomy-literate workforce
- Improving technology discovery
- Improving DoD governance for autonomous systems
- Countering adversary use of autonomy

These interdependent recommendations are intended to build trust in autonomous systems, while at the same time accelerating DoD's progress.

The next set of recommendations focuses on strengthening the operational pull for autonomy, both by better understanding how others may use autonomy against the U.S., and by equipping our forces to counter such capabilities. These recommendations take the form of a series of demonstrations and experiments that demonstrate near-term military value while also building warfighter trust:

- Autonomous agents to improve cyber-attack indicators and warnings
- Onboard autonomy for sensing
- Time-critical intelligence from seized media
- Dynamic spectrum management for protection missions
- Unmanned undersea vehicles (UUVs) to autonomously conduct sea mine counter-measures missions
- Automated cyber response
- Cascaded UUVs for offensive maritime mining
- Organic tactical unmanned aircraft (UA) to support ground forces
- Predictive logistics and adaptive planning
- Adaptive logistics for rapid deployment

The final set of recommendations is intended to expand the envelope of technologies available for use on DoD missions. "Stretch problems" are proposed as a means to both strengthen the operational pull and mature the underlying technologies, such that they would be trusted for application on DoD missions:

- Early warning system for understanding global social movements
- Autonomous swarms that exploit large quantities of low-cost assets
- Intrusion detection on the Internet of things
- Autonomous cyber resilience for military vehicle systems
- Autonomous air operations planning

The recommendations of the study are briefly outlined in Table 4 at the end of the report, on page 102. Details for each recommendation are included in the relevant chapter sections and provide additional information to assist with their implementation.

Summary comments

Autonomy has many definitions and interpretations. For this reason, the report begins with an introductory section that defines the term and its context for the purposes of this study. This section also includes examples drawn from the commercial sector that illustrate diverse applications with operational relevance to the DoD.

The second major section addresses the issue of trust, and highlights both similarities and differences between military applications and commercial uses of autonomy. The study argues that an integrated approach—one that spans the entire lifecycle of a system—is needed to establish, maintain, and act upon current evaluations of trustworthiness in autonomous systems.

The remaining three sections motivate and elaborate on the major recommendations summarized above.

In summary, the study concluded that autonomy will deliver substantial operational value across an increasingly diverse array of DoD missions, but the DoD must move more rapidly to realize this value. Allies and adversaries alike also have access to rapid technological advances occurring globally. In short, speed matters—in two distinct dimensions. First, autonomy can increase decision speed, enabling the U.S. to act inside an adversary's operations cycle. Secondly, ongoing rapid transition of autonomy into warfighting capabilities is vital if the U.S. is to sustain military advantage.

1 Introduction

In November 2014, the Under Secretary of Defense for Acquisition, Technology, and Logistics directed the Defense Science Board to conduct a study to identify the science, engineering, and policy problems that must be addressed to facilitate greater operational use of autonomy across all warfighting domains. The study team identified opportunities for DoD to enhance mission efficiency, shrink life-cycle costs, reduce loss of life, and perform new missions—in both physical and virtual domains. The team concluded that there are both substantial operational benefits and potential perils associated with the use of autonomy.

Imagine if....

We could covertly deploy networks of smart mines and UUVs to blockade and deny the sea surface, differentiating between fishing vessels and fighting ships
and not put U.S. Service personnel or high-value assets at risk.

We had an autonomous system to control rapid-fire exchange of cyber weapons and defenses, including the real-time discovery and exploitation of never-seen-before zero day exploits
enabling us to operate inside the "turning radius" of our adversaries.

We had large numbers of small autonomous systems that could covertly enter and persist in denied areas to collect information or disrupt enemy operations
a "sleeper presence" on call.

We had large numbers of low-cost autonomous unmanned aircraft capable of adaptively jamming and disrupting enemy PNT capabilities
destroying their ability to coordinate operations.

We had autonomous high performance computing engines capable of not only searching "big data" for indicators of WMD proliferation, but of deciding what databases to search
to provide early warning and enable action

And imagine if we are unprepared to counter such capabilities in the hands of our adversaries.

About autonomy

Autonomy results from delegation of a decision to an authorized entity to take action within specific boundaries. An important distinction is that systems governed by prescriptive rules that permit no deviations are *automated*, but they are not *autonomous*. To be autonomous, a system must have the capability to independently compose and select among different courses of action to accomplish goals based on its knowledge and understanding of the world, itself, and the situation.[1]

[1] Definitions for intelligent system, autonomy, automation, robots, and agents can be found in L.G. Shattuck, *Transitioning to Autonomy: A human systems integration perspective, p. 5.* Presentation at *Transitioning to Autonomy: Changes in the role of humans in air transportation* [March 11, 2015]. Available at human-factors.arc.nasa.gov/workshop/autonomy/download/presentations/Shaddock%20.pdf (Accessed June 2016.)

Recognizing that no machine—and no person—is truly autonomous in the strict sense of the word, we will sometimes speak of autonomous *capabilities* rather than autonomous *systems*[2]

Figure 1 DoD is increasingly employing autonomous capabilities across a diverse array of systems.

The primary intellectual foundation for autonomy stems from artificial intelligence (AI), the capability of computer systems to perform tasks that normally require human intelligence (*e.g.*, perception, conversation, decision-making). Advances in AI are making it possible to cede to machines many tasks long regarded as impossible for machines to perform.

Intelligent systems aim to apply AI to a particular problem or domain—the implication being that the system is programmed or trained to operate within the bounds of a defined knowledge base. Autonomous function is at a system level rather than a component level. The study considered two categories of intelligent systems: those employing *autonomy at rest* and those employing *autonomy in motion*. In broad terms, systems incorporating *autonomy at rest* operate virtually, in software, and include planning and expert advisory systems, whereas systems incorporating *autonomy in motion* have a presence in the physical world and include robotics and autonomous vehicles. As illustrated in Figure 1, many DoD and commercial systems are already operating with varying kinds of autonomous capability.

Robotics typically adds additional kinds of sensors, actuators, and mobility to intelligent systems. While early robots were largely *automated*, recent advances in AI are enabling increases in *autonomous* functionality.

One of the less well-known ways that autonomy is changing the world is in applications that include data compilation, data analysis, web search, recommendation engines, and forecasting. Given the limitations of human abilities to rapidly process the vast amounts of data available today, autonomous systems are now required to find trends and analyze patterns. There is no need to solve the long-term AI problem of general intelligence in order to build high-value applications that

2 See J.M. Bradshaw, R.R. Hoffman, M. Johnson, and D.D. Woods, "The Seven Deadly Myths of 'Autonomous Systems," *IEEE Intelligent Systems 28*, no. 3. [May/June 2013], pp. 54-61. Available at jeffreymbradshaw.net/publications/IS-28-03-HCC_1.pdf (Accessed April 2016.)

exploit limited-scope autonomous capabilities dedicated to specific purposes. DoD's nascent Memex program is one of many examples in this category.[3]

Rapid global market expansion for robotics and other intelligent systems to address consumer and industrial applications is stimulating increasing commercial investment and delivering a diverse array of products. At the same time, autonomy is being embedded in a growing array of software systems to enhance speed and consistency of decision-making, among other benefits. Likewise, governmental entities, motivated by economic development opportunities in addition to security missions and other public sector applications, are investing in related basic and applied research. Applications include commercial endeavors, such as IBM's Watson, the use of robotics in ports and mines worldwide, autonomous vehicles (from autopilot drones to self-driving cars), automated logistics and supply chain management, and many more. Japanese and U.S. companies invested more than $2 billion in autonomous systems in 2014, led by Apple, Facebook, Google, Hitachi, IBM, Intel, LinkedIn, NEC, Yahoo, and Twitter. [4]

A vibrant startup ecosystem is spawning advances in response to commercial market opportunities; innovations are occurring globally, as illustrated in Figure 2 (top). Startups are targeting opportunities that drive advances in critical underlying technologies. As illustrated in Figure 2 (bottom), machine learning—both application-specific and general purpose—is of high interest. The market-pull for machine learning stems from a diverse array of applications across an equally diverse spectrum of industries, as illustrated in Figure 3.

Autonomy accelerates enterprise performance

Commercial enterprises are enhancing performance through the use of autonomy to exploit advances in processing power, big data analytics, and networked systems that leverage diverse and distributed sensor arrays. Opportunities exist for DoD to enhance mission performance by employing autonomy at rest and autonomy in motion, both supporting human-machine collaboration. The commercial sector is a lucrative source of both basic capability and best practices relevant to many such opportunities.

Over the past several years, autonomous sensing and decision-support techniques have been demonstrated by the commercial sector. The following are only a few examples:

Footage from the estimated 52,000 government-operated closed circuit television (CCTV) cameras in the United Kingdom, along with the 1.85 million total cameras across the country, is used in as many as 75 percent of the 3.9 million criminal cases annually.[5,6]

3 W. Shen, *Memex*. Available at www.darpa.mil/program/memex (Accessed June 2016.)

4 Quid, referenced in *Robot Revolution–Global Robot and AI Primer.* [Bank of America Merrill Lynch, December 16, 2015].

5 *The Price of Privacy: How local authorities spent £515m on CCTV in four years*. [Big Brother Watch, February 2012]. Available at www.bigbrotherwatch.org.uk/files/priceofprivacy/Price_of_privacy_2012.pdf. (Accessed January 2016.)

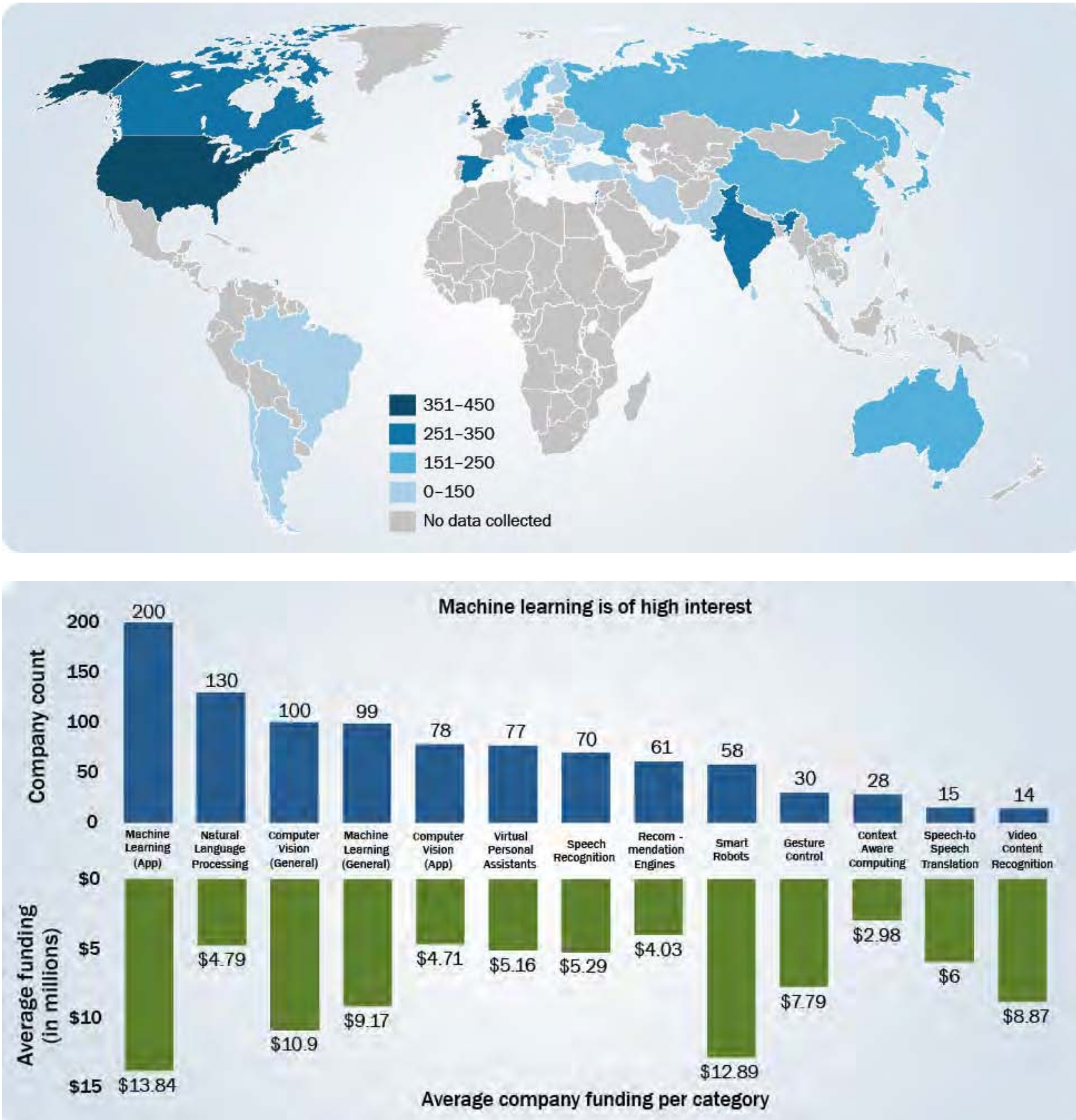


Figure 2 Global autonomy startups are mapped (top); startup opportunity targets are categorized

- Automated video content analysis software searches long videos for key events and can incorporate human facial and gait recognition.

[6] P. Taylor and S. Bond, eds., *Crimes detected in England and Wales 2011/12.* [July 2012]. Available at www.gov.uk/government/uploads/system/uploads/attachment_data/file/116435/hosb0812.pdf. (Accessed January 2016.)

Figure 3 The machine intelligence ecosystem comprises a diverse array of applications and industries.

SOURCE: www.shivonzilis.com/machineintelligence

- The U.S. credit, debit, and prepaid card industry monitors more than 1,200 transactions per second with automated tools that identify fraudulent transactions within 10 milliseconds on a customer base of more than 1.3 billion cards.[7]
- IBM's Watson for Oncology system has ingested over 12 million pages of medical content, including over 200 medical textbooks and 290 medical journals, and provides oncologists with recommended courses of treatment within 30 seconds, largely from unstructured records.[8]
- Google machine-learning password classifiers authenticate users via multiple signals, such as Internet protocol (IP) address, geolocation, and login time, resulting in 99.5 percent reduction in accounts compromised by spammers.[9]
- The mining industry has integrated autonomy into many systems. Both excavating and hauling vehicles are equipped with vehicle controllers, high precision global positioning system (GPS) sensors, and obstacle detection. These features allow safe operation through a complex load, haul, and dump cycle, and the ability to integrate with other vehicles, people, and obstacles that are also part of the autonomous system.[10]

---

7 The Nilson Report, *U.S. General Purpose Cards - Midyear 2015,* Issue 1069. [August 2015].

8 *IBM Watson for Oncology*. Available at www.ibm.com/smarterplanet/us/en/ibmwatson/watson-oncology.html. (Accessed January 2016.)

9 Google, *An update on our war against account hijackers*. [February 19, 2013]. .Available at googleblog.blogspot.com/2013/02/an-update-on-our-war-against-account.html. (Accessed January 2016.)

10 Warner Norcross & Judd, *Trucking, Mining Industries Blazing a Path to Vehicle Autonomy*. [January 14, 2015]. Available at www.wnj.com/Publications/Trucking-Mining-Industries-Blazing-a-Path-to-Vehic (Accessed June 2016.)

Lucrative global markets are attracting ongoing investment, advancing basic technologies while also targeting delivery of capabilities relevant to DoD missions. The following are only a few examples:

- Vehicles that drive themselves in limited circumstances (*e.g.*, on the freeway and in traffic jams in good weather) will begin to enter the market in 2016 and may be on the road in large numbers by 2017. "The global market for autonomous vehicles is projected to grow from $42 billion in 2025 to $77 billion by 2035," and Japan and Western Europe are likely to be early adopters of the technology.[11]
- The advanced autonomous waterborne applications initiative was funded by Tekes (Finnish Funding Agency for Technology and Innovation) in 2015 and is led by Rolls-Royce. It will bring together "universities, ship designers, equipment manufacturers, and classification societies to explore the economic, social, legal, regulatory and technological factors which need to be addressed to make autonomous ships a reality." [12]
- A prototype hybrid drone, capable of flying as a traditional helicopter as well as a fixed-wing aircraft, was demonstrated at a conference in Denmark in 2015. The drone, which stems from a project led by the Technical University of Denmark, can maneuver with an accuracy of five centimeters and is intended to enable public and private enterprises to collect data faster and more accurately than previously possible.[13]

Enterprises are already achieving accelerated performance through the integration of available autonomy-enabling technologies. At the same time, their successes are stimulating increased investment across a broad array of underlying technologies that together will spawn new types of autonomous systems. See Box 1 on page 10 for an example of how this strategy evolved at Amazon.

In this study, four broad categories are used to characterize underlying technologies critical to the development of autonomous systems: *Sense; Think/Decide; Act*; *Team*. The relative importance of each category varies by application, as do the drivers that spawn new capabilities. Advances to *Sense* are driven by a diverse array of applications (including, but not limited to, autonomous systems) that share a common need to reduce sensor size, weight, and power requirements. Artificial intelligence, which enables the *Think/Decide* functionality in autonomous systems, is benefiting from advances in computational power as well as availability of vast data sets. Demand for productivity growth via automation was an early driver of advances in actuators and mobility that *Act*; and that demand is growing as robotics become more intelligent and new applications are emerging. A growing number of applications require human-machine teaming and collaboration—letting each do what it does best, but also imposing new requirements on the underlying *Team* technologies. Table 1 summarizes

[11] Boston Consulting Group, *The Autonomous Vehicle: The Car of the Future*. Available at on.bcg.com/1HNAHKH (Accessed June 2016.)

[12] Rolls-Royce, *Rolls-Royce to Lead Autonomous Ship Research Project*. [July 2, 2015]. Available at www.rolls-royce.com/media/press-releases/yr-2015/pr-02-07-15-rolls-royce-to-lead-autonomous-ship-research-project.aspx. (Accessed January 2016.)

[13] Homeland Defense & Security Information and Analysis Center, *Denmark Creates Hybrid Smart UAV.* [August 3, 2015] Available at www.hdiac.org/node/2069 (Accessed January 2016.)

Box 1: Amazon: A Commercial Enterprise Example

Commercial enterprises are now leveraging autonomy as a means to create entirely new business models. A well-known example is Amazon, which began as an online bookseller and has expanded to offer online retail, computing services, consumer electronics, and digital content, as well as other local services such as daily deals and groceries. While autonomy has not been the only factor in Amazon's success, Amazon has employed autonomy effectively in areas where its business model does not allow human activity or where autonomy can break a bottleneck.

Amazon began as a dot-com company where its primary advantages were perceived to be elimination of bricks and mortar costs and the broad reach of the early Internet. It could sell and deliver physical books at costs lower than competitors.

The early differentiator for physical bookstores was their ability to know the tastes and interests of frequent customers, which added to loyalty and sales. Amazon moved quickly to develop a recommendation engine that captured and assessed large amounts of data about its customers. It also sought, aggregated, and published customer reviews, along with customer-generated questions and answers about its wide range of products. As Amazon expanded into other retail areas, it has expanded and refined this use of autonomy at rest.

Amazon also employs autonomy in motion by custom-packaging books for shipment using a fully autonomous line. When Amazon expanded into a broader range of retail items, warehouse operations became a bottleneck. In 2012, Amazon purchased Kiva Systems, a manufacturer of mobile robotic fulfillment systems.

The figure below displays the revenue performance of Amazon over the past twenty years. It is evident that Amazon leadership embraces the benefits of autonomy as part of that picture.

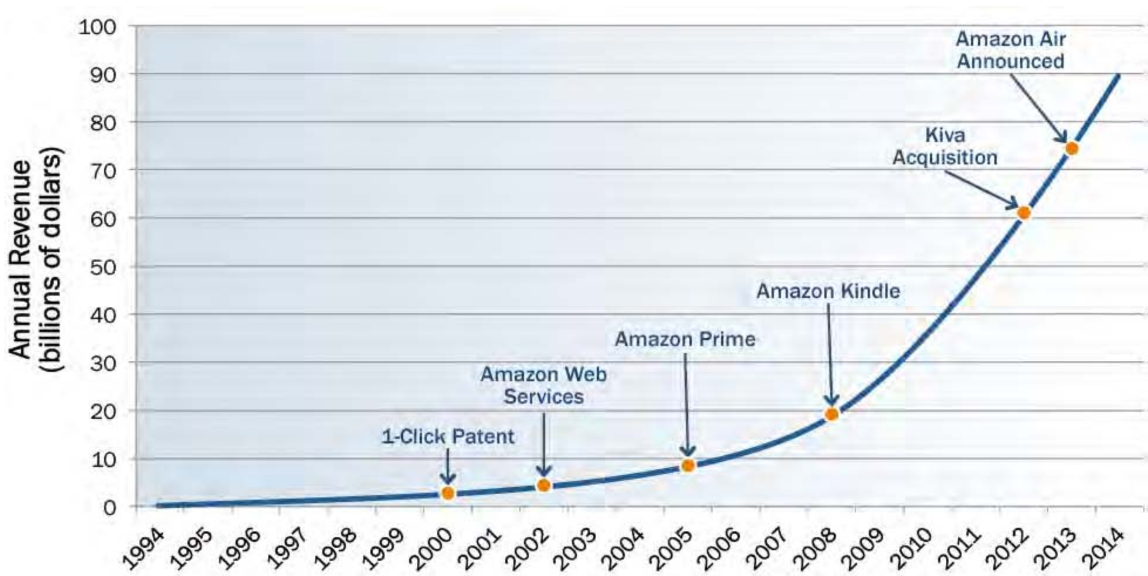


Table 1 Projected capabilities for autonomous systems

SENSE: Sensors, Perception, Fusion

- *Available today*: Full-spectrum sensing (EM, bits, vibration, chemical…); Object recognition
- *Likely available near term*: Human senses (sight, smell…); Integration of perception with motor skills
- *May be available long term*: High-fidelity touch; Scene understanding

THINK/DECIDE: Analysis, Reasoning, Learning

- *Available today*: High-volume computational throughput and data architectures; Algorithm variety and complexity; Task-specific, rule-based decision; Rules; Learning from training data, sentiment analysis
- *Likely available near term*: Explicit and testable knowledge representation; Anomaly recognition; Option generation, pruning; Social and behavioral models; Culturally informed, values-based reasoning; Transparent decision logic; C2 for many nodes; Learning by doing, watching
- *May be available long term*: Goal definition; Abstraction, Skills transfer; Inference; Empathy; General purpose, idea-based reasoning; Judgment, intuition

ACT: Motion, Manipulation

- *Available today*: Navigation (routing); Strength, endurance
- *Likely available near term*: Navigation (obstacle avoidance); Agility, dexterity
- *May be available long term*: Navigation (dense, dynamic domains); High degree of freedom actuator control

TEAM: Human/machine, Machine/machine, Info exchange

- *Available today*: High man:machine ratio; Rule-based coordination of multiple platforms; High-volume communications and data transfer
- *Likely available near term*: Observability and directability; Provably correct emergent behavior; Trustworthiness and trust calibration under defined conditions; Natural language processing
- *May be available long term*: Shared "mental models," mutual predictability; Understanding intent; Fully adaptive coordination; Implicit communication

the study team's assessment of the availability of critical underlying technologies in each of these four categories.

Military value and current DoD uses

The DoD has strategically increased its adoption of robotics and unmanned vehicle systems in the last decade, but the vast majority of the systems are remotely operated rather than autonomous. Recent programs show a progression from pre-programming and remote control to autonomous functionality, but progress has been slow. The Department is engaged in R&D across many aspects of autonomy, but has not yet addressed the R&D needed to overcome the systemic challenges to the widespread use of autonomy.

The growth in robotics and unmanned systems was largely driven by perceived improvements in performance and cost. The actual advantages are more complex. Safety improves by reducing the lethality of warfare and the ability to adopt riskier tactics because a system is unmanned. Accuracy also improves, with more endurance, range, and speed in comparison to manned vehicles. Systems are also more flexible and more mobile. Autonomy also enables the execution of new missions—particularly in domains such as cyber and electronic warfare, in which decision speed is critical to success. Figure 4 summarizes the relative value of autonomy against key mission parameters.

Figure 4 Autonomy derives operational value across a diverse array of vital DoD missions.

While more difficult to quantify for the DoD, commercial enterprises are increasingly demonstrating substantial cost savings through the adoption of autonomy. From a DoD perspective, UA are believed to result in cost reductions over time; other applications may deliver similar benefits—particularly in areas such as logistics in which DoD can leverage advances driven by large commercial markets. But, as indicated in Figure 4, the value of autonomy to DoD missions extends well beyond cost reductions.

U.S. defense spending on UA has grown to almost $3 billion in 2016.[14] The number of unmanned aircraft has grown to more than 11,000, or 40 percent of all aircraft.[15] In the U.S., the Federal Aviation Administration (FAA) introduced a national framework for registering UA at the end of 2015, and over 181,000 drones were registered in the first two weeks, dwarfing the DoD population.[16] The U.S. is not alone; ninety countries around the world operate UA, with thirty armed UA programs established or in development.[17]

While commercial market opportunities will advance many underlying technologies that are critical to DoD applications, there are areas in which DoD cannot rely on the commercial industrial base to develop needed capabilities, including the following:

---

[14] U.S. Department of Defense, March 2016.

[15] J. Gertler, *U.S. Unmanned Aerial Systems.* [Congressional Research Service CRS R42136, January 3, 2012] and *How many UAVs for DoD?* [CRS IN10317, August 27, 2015].

[16] R. Marsh and H. Kelly, "181,000 drones registered with FAA in two weeks," *CNN Money* [January 6, 2016]. Available at money.cnn.com/2016/01/06/technology/faa-drone-registration/index.html (Accessed June 2016.)

[17] K. Sayler, *A World of Proliferated Drones: A Technology Primer* [Center for a New American Security, June 2015]. Available at www.cnas.org/world-of-proliferated-drones-technology-primer#.V1L4myGEBqM (Accessed June 2016.)

- The commercial sector can often structure the environment to simplify decision-making for an autonomous system. In many DoD missions, the operation will be conducted in an environment as encountered, with no opportunity to structure the environment or map it to make the problem easier for the autonomous system.
- The commercial sector can decompose a problem and first tackle the easy parts using autonomous systems; hard parts of the problem can still be solved using humans when this is more cost-effective (or until the required technology matures). In some DoD applications—for example, operating in contested environments—using humans to address the limitations of autonomous systems may prove challenging (or impossible).
- Commercial applications rarely deal with intelligent adversaries that try to actively defeat the technology in use, although cyberspace is one notorious counter example. Many DoD applications will need autonomy capabilities that are robust enough to cope with the deployment of deception, assault, and counter-autonomy technologies by adversaries.

In summary, the study concluded that autonomy has the potential to deliver substantial operational value across a diverse array of vital DoD missions, and that the DoD should speed its adoption to realize the potential benefits across a diverse array of missions.

Study approach

Throughout the course of the study, the team grappled with difficult questions, including the following:

- How can systems be "future-proofed" to enable autonomy technology advances to gracefully—and rapidly—upgrade?
- How can military advantages be sustained, given the rapidly advancing commercial global technology base?
- How can autonomous systems benefit from test and evaluation (T&E) when every time they are used they change themselves?... and how can we verify that systems are learning the right things?
- How can the U.S. compensate for adversaries who may have very different rules of engagement to employ lethal autonomous systems?
- How can autonomous systems be made more "cyber-proof"—given that every new capability introduces a vulnerability?
- Will we increase trust in autonomous systems if each includes a "black box" audit trail that can explain why they did what they did?
- How can we counter an adversary's autonomous systems? And how can we protect our autonomous systems from an adversary's interference?
- Can autonomous systems save lives of U.S. Service personnel as well as those of innocent non-combatants? ...and could the U.S., therefore, be less deterred from pursuit of foreign policy objectives?

2 Trustworthiness and Trust in Autonomous Systems

Most commercial applications of autonomous systems are designed for operation in largely benign environments, performing well-understood, safe, and repetitive tasks, such as routing packages in a fulfillment center warehouse. Design for commercial systems rarely considers the possibility of high-regret outcomes in complex, unpredictable, and contested environments. In military operations, these can include an adversary whose goal is to neutralize the use and effectiveness of such systems, either through deception, direct force, or increased potential for collateral damage or fratricide. Although commercial applications are gradually expanding beyond these controlled environments, *e.g.*, self-driving cars, delivery drones, and medical advisory systems, fielded autonomous systems do not yet face a motivated adversary attempting to defeat normal operations.

Trust is complex and multidimensional.[18] The individual making the decision to deploy a system on a given mission must trust the system; the same is true for all stakeholders that affect many other decision processes. Establishing trustworthiness of the system at design time and providing adequate indicator capabilities so that inevitable context-based variations in operational trustworthiness can be assessed and dealt with at run-time is essential, not only for the operator and the Commander, but also for designers, testers, policy and lawmakers, and the American public.

Key issues and barriers to trust

Methods for ensuring trustworthiness include careful design and implementation of systems to assure key attributes including high levels of competence, reliability, and integrity. Of course, designers are expected to embed these attributes in development and manufacturing of autonomous weapons systems. However, such attributes may be undercut by characteristics associated with hybrid, multi-party human-machine teams, including:

Lack of human-analog sensing and thinking by the machine. Because an autonomous system may have different sensors and data sources than any of its human teammates, it may be operating on different contextual assumptions of the operational environment. In addition, for some specific algorithm choices—such as neuromorphic pattern recognition for image processing, optimization algorithms for decision-making, deep neural networks for learning, and so on—the "reasoning" employed by the machine may take a strikingly different path than that of a human decision-maker.

Lack of self- or environmental awareness by the machine. Self-awareness can be as simple as understanding its own system health, such as battery level, or more subtle, such as knowing when it is operating outside of its original design boundaries or assumptions. Environmental awareness includes conventional sensing of the environment; such as icing on a wing or jammed communications, as well as more subtle effects such as GPS spoofing. Of course, it is not sufficient

[18] R. R. Hoffman, Matthew Johnson, J.M. Bradshaw, and Al Underbrink. "Trust in Automation." *IEEE Intelligent Systems,* Vol. 28, Issue 1 [January/February 2013], pp. 84-88.

for a machine to be aware of changes in itself and its operating environment, it must also be able to adapt to those changes flexibly and effectively.

Low observability, predictability, directability, and auditability. Autonomous systems not only need to operate reliably and within their envelope of competence in dynamically varying and complex operational contexts, but also to be able to make relevant information *observable* to human and machine teammates. Moreover, even if machines are competently designed to enable observation of *current* state and effects, they may not incorporate sufficient *anticipatory* indicators to allow other human and machine teammates to ensure *predictability*. In addition, when something goes wrong, as it will sooner or later, autonomous systems must allow other machine or human teammates to intervene, correct, or terminate actions in a timely and appropriate manner, ensuring *directability*. [19] Finally, the machine must be *auditable*—in other words, be able to preserve and communicate an immutable, comprehensible record of the reasoning behind its decisions and actions after the fact.[20]

Low mutual understanding of common goals. If humans and autonomous machines are to work effectively together, they need common goals and a mutual knowledge of those common goals. Many of the commercial aircraft accidents in the 1990s associated with automation occurred when the flight crew had one goal (*e.g.*, staying on the glide slope during an approach) and the flight management computer had another (*e.g.*, executing a go-around). Improved training of personnel can help to address such issues. Dealing with future autonomous systems, however, may demand more than a one-sided approach, and may include increasing the machine's awareness of what the operator is trying to achieve.

Ineffective interfaces. Conventional computer interfaces, such as mouse point-and-click, can slow communications between humans and machines and inhibit the coordination and cooperation needed in time-sensitive or high-risk situations. Better interfaces, such as natural-language processing or more effective visualizations of complex information and situations can help mitigate such issues.

Systems that learn. Machines are being developed with experience that change their capabilities and limitations and adapt to their use and environment. Such systems will outgrow their initial verification and validation and will require more dynamic methods to perform effectively throughout their lifecycle.

Autonomy in support of command and control

One of the most contentious applications of autonomy is for command and control in military operations or warfighting, but the potential benefits are real. The time for concepts of operations

---

[19] On observability, predictability, and directability, see M. Johnson, M., J.M. Bradshaw, P. J. Feltovich, C. M. Jonker, M. B. van Riemsdijk, and M. Sierhuis, "Coactive Design: Designing Support for Interdependence in Joint Activity, *Journal of Human-Robot Interaction*, Vol. 3, No. 1. [2014], pp. 43-69.

[20] M.R. Endsley, "Measurement of Situation Awareness in Dynamic Systems," *Human Factors,* 37(1) [March 1995], pp. 65-84; and M.R. Endsley, "Building Resilient Systems: Incorporating Strong Human-system Integration," *Defense AT&L Magazine* [January–February 2016, 2015]. Available at dau.dodlive.mil/2015/12/28/building-resilient-systems-via-strong-human-systems-integration (Accessed March 2016.)

(CONOPs) development, target selection, and mission assignments can be significantly reduced, and, during combat operations, commanders could be better equipped to respond to changing situations and redirect forces. While commanders understand they could benefit from better, organized, more current, and more accurate information enabled by application of autonomy to warfighting, they also voice significant concerns.

Implementation of autonomous capabilities will require significant changes in command and control concepts. A previous DSB study acknowledged the importance of addressing command and control of autonomous systems, but found that it was an unsolved problem and did not address it further.[21]

Whether mediated by man or machine, all acts, but especially acts related to warfighting, must be executed in accordance with policy and so, in some sense, there is no completely autonomous behavior. Any use of autonomy must conform to a substantive command and control regime laying out objectives, methods and express limitations to ensure that autonomous behavior meets mission objectives while conforming to policy.[22]

In fact, most autonomous combat systems will and should act under the guidance and instructions of a field commander who will exercise direct oversight. Initial use of autonomous systems for combat will likely assist commanders and their staffs with developing situational awareness and planning missions. The volume and velocity of the data used by the underlying system will change the pace of operations. For example, current air operations planning often involves a several-day process, beginning with identification of objectives; moving to general target selection; to intelligence support identifying particular targets; to determination of final plans by the Commander balancing risks and potential value in achieving objectives; coordination of air, land, and sea assets; and, finally, mission execution and battle damage assessment. Given human limitations, each stage results in static point in time, and planning is generally a linear and time-consuming process. Because planning often needs to respond to new information, autonomous systems will greatly accelerate the pace of information update and can suggest significant plan changes far more quickly.

Commanders will not only need to develop models to calibrate understanding of machine generated information, but will need better automated tools to adjust to the pace of update. This requires careful design of autonomous systems so they can explain and justify recommendations in principled terms that build trust between commanders, staff, and machine-generated output as well as develop a concrete understanding of mission outcomes that depend critically on the data and methodology employed by the autonomous systems.

---

21 Defense Science Board, *Report of the Task Force on the Role of Autonomy in DoD Systems.* [2012], p 16. Available at www.acq.osd.mil/dsb/reports/AutonomyReport.pdf (Accessed March 2016.)

22 For one approach to this problem, see A. Uszok, J.M. Bradshaw, J. Lott, M. Johnson, M. Breedy, M. Vignati, K, Whittaker, K. Jakubowski, and J. Bowcock, "Toward a Flexible Ontology-Based Policy Approach for Network Operations Using the KAoS Framework," *Proceedings of the 2011 Military Communications Conference (MILCOM 2011)*. [New York City, NY: IEEE Press, November 2011], pp. 1108-1114.

Commanders and their staff need to both exercise oversight of fielded autonomous systems, and have access to efficient mechanisms to develop control datasets that teach autonomous systems so they can react in well-understood ways to unexpected and subtle changes. This requires new approaches to human factors that are informed by warfighting practice and tradition. It also requires development of commander and staff training.

Human-machine collaboration and combat teaming

Applications of autonomy engage both human and machine throughout the system lifecycle. Certain roles will remain the purview of the human, others will be shared, and some tasks will be implemented solely by machines. The specific roles of humans will vary by mission and over time.

The overall initial operational design will be done primarily by humans. This includes tasks such as defining the envelope and boundaries of potential autonomous behavior, identifying general behavioral parameters, and establishing the range of rules of engagement within which the system will be designed to operate.

Development of the system may be shared. Humans would define behavioral parameters for a range of missions and train the system for that range of missions; machines will learn behaviors in both standard and unplanned situations while adhering to the rules of engagement.

During deployment, the machine might continue to learn while executing the mission within established bounds. Alternately, the machine might be operationally limited to execution of its trained behaviors. Humans in the loop will set mission parameters and may continue to refine the rules of engagement by verifying operations across varied operational conditions. Some operational modes may dynamically specify what responsibilities humans and machines will perform and share throughout the mission. Only humans would retain the rights to change mission parameters or change the rules of engagement.

Effective human-machine collaboration requires that team members share common goals. This requires mutual understanding of the common goals, even though the goals may be expressed in different frameworks and semantics. It is critically important for the human operator to have a good understanding of the machine's goals, otherwise human-machine team failures are bound to occur. It may be equally important to provide a means for a machine to understand team goals. Significant autonomy capabilities will derive from a machine's ability to infer the commander's intent and to act adaptively in a non-pre-programmed fashion, and in doing so, being able to deal with unanticipated situations not foreseen by either the designer or the operator.

In addition, because teamwork between humans and autonomous machines may require various levels of communications, system architectures must support a variety of machine-to-machine and machine-to-human communications links, with the latter focused on improving the link through emerging technologies (*e.g.*, neurolinguistics programming). Finally, humans and systems must train together (as shown in **Figure 5**) to develop CONOPs, as well as to achieve skilled human-machine team performance across a wide range of missions, threats, environments, and operators.

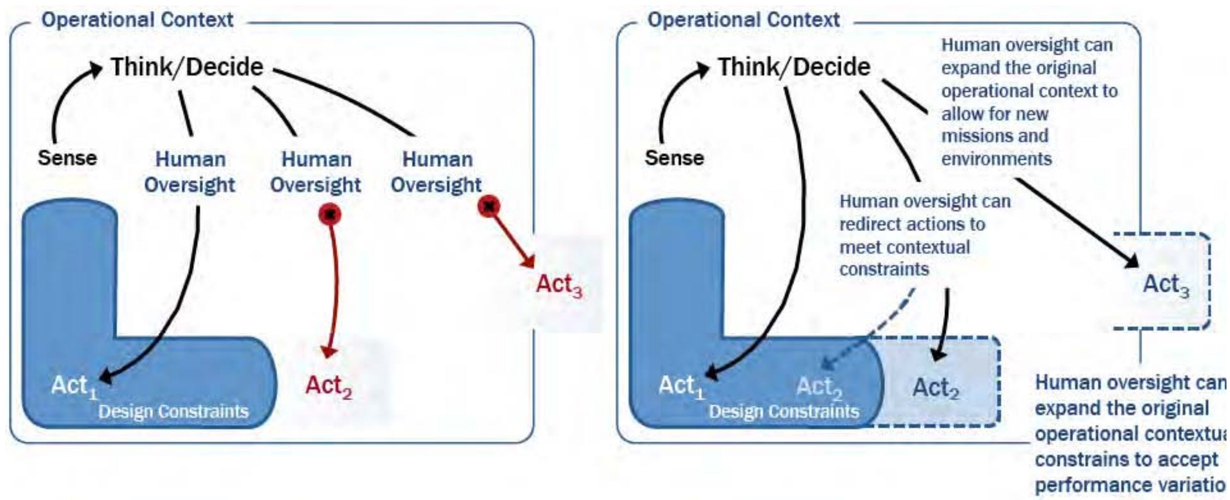
It is important for the human operator to understand the basic competence of the machine, and, during operations, when it may be operating outside of its design assumptions or operational boundaries. A direct approach would simply encode the operational parameters; a more comprehensive approach would rely on in-line modeling and simulation to instruct behavior, including nominal and out-of-envelope behaviors. Models, tools, and datasets must be developed to deal with the fact that operational boundaries are situational, may evolve, and may violate the original system design assumptions.

Figure 5 Combat veterans regularly refresh their unmanned aircraft skills.
SOURCE: News, February 12, 2013, www.army.mil

Because many autonomous system behaviors will change over time due to learning, discrepancies may occur between actual system performance and operator expectations, possibly leading to teammate surprise during operations. One approach to ameliorate this situation is to provide any human teammate with a training history of the autonomous system's experience or competencies, analogous to current military practice based on a human teammate's service history and rank. Alternatively, a human and machine team rehearsing missions together could provide each teammate with implicit expectations of behavior based on current capabilities.

In addition, if a human or machine changes capabilities and limitations—learns—over time, more dynamic and adaptive methods are needed to reliably verify and validate system behavior for future engagements. This will necessitate a significant change in today's relatively static T&E practices.

A wide range of design approaches can be implemented to provide human and machine teammates with insight and foresight with respect to a system's integrity. The autonomous system design needs to support self-awareness (including an assessment of internal system health and external environmental and contextual factors) and anticipation of the future to support its own reasoning and adaptation processes. It also needs to provide this information in response to commands or requests from other machine or human teammates. This information, in conjunction with an understanding of the system's own design performance limitations or constraints, can be used for both self- and external assessment of a system's operation within its design envelope;



Human oversight is <u>in the loop</u> to

- confirm actions (Act1)
- deny actions outside designed constraints (Act2)
- deny actions outside the operational context (Act3)

Human oversight is <u>on the loop</u> as needed to

- allow actions outside designed constraints (Act2)
- allow actions outside the operational context (Act3) – and take advantage of evolving opportunities.

Figure 6 Oversight "on-the-loop" provides additional opportunities for human-machine partnership.

violations or anticipated violations of this envelope can be communicated to other teammates, human or machine, for appropriate mitigation. In addition, designing for transparency and traceability of a system's situational awareness and decision-making can help to explain decisions taken, both operationally and forensically, given that adequate communications and semantic links are included. Figure 6 illustrates how the range of operational options might be dynamically expanded through the implementation of "human-on-the-loop" intervention.

Because of the potential for high-regret outcomes in complex scenarios, especially via adversary attacks against system vulnerabilities, designs need to be red teamed at all phases of concept evaluation, development, acquisition, and employment. One approach is to work with an in-line suite of consistent and related models and simulations employed and verified at all stages of the acquisition and deployment cycle.

By red-teaming early in design and engineering, concepts can be de-selected or augmented. With suitable modeling and simulation (M&S) representation, the effectiveness of conventional adversary actions (*e.g.*, kinetic weapons, electronic warfare) can be evaluated, as well as any potential vulnerabilities to more subtle attacks such as sensor spoofing, communications intercepts, or embedded cyber-attacks. As development progresses, the level of the M&S representation and tools should be made more sophisticated, providing greater fidelity of not only capabilities, but also of potential vulnerabilities, setting the stage for another round of red teaming, at a higher level of resolution. At

> *A red team is an independent group that challenges the organization—its doctrine, CONOPs, and its systems—with the mindset of an adversary to improve the process or the product.*

a sufficient level of maturity, ideally soon after concept development, live and constructive simulations can be used to introduce humans in-the-loop and on-the-loop to help develop CONOPs and tactics for the human-machine teams. Later in the development cycle, more sophisticated M&S facilities can be used for both training and dealing with vulnerabilities, to provide additional opportunities for red teaming.

Cultural, policy, and legal issues

The overwhelming majority of potential military applications for autonomy are non-lethal and offer the potential for improved efficiencies or entirely new capabilities. Skepticism about the employment of autonomy in military operations is almost wholly focused on the use of autonomous weapons systems with potential for lethality. For this reason, any new autonomous capability may meet with resistance unless DoD makes clear its policies and actions across the spectrum of applications.

DoD recently undertook a comprehensive review of policy and procedures associated with autonomy in weapon systems, and produced Directive 3000.09 on "Autonomy in Weapon Systems" in November 2012. As stated in the purpose statement of the directive:

> *DoDI 3000.09 establishes DoD policy and assigns responsibilities for the development and use of autonomous and semi-autonomous functions in weapon systems, including manned and unmanned platforms; establishes guidelines designed to minimize the probability and consequences of failures in autonomous and semi-autonomous weapon systems that could lead to unintended engagements.*

The Directive provides comprehensive guidance to all of the stakeholders concerned with deployment of autonomous systems. The most important policy points to be made from the Directive that are relevant to public concerns are that there are no proscriptions for the development of lethal autonomous weapon systems, but their development would require a much more rigorous review and approval process. Emphasis is placed on assurance that the system will perform as intended and be as immune as possible to unintended loss of control, capture, or compromise by the adversary. Moreover, appropriate use of human judgment over the use of force is required and use must be in accordance with all applicable domestic and international law, in particular, the law of war. [23]

Distinctions are drawn among three types of weapons systems:

- Semi-autonomous weapon systems, which require human operator selection and authorization to engage specific targets (*e.g.*, human in-the-loop control)
- Human-supervised autonomous weapon systems, which allow human intervention and, if needed, termination of the engagement, with the exception of time-critical attacks on platforms or installations (*e.g.*, human on-the-loop control)

[23] Office of General Counsel, *Department of Defense Law of War Manual*. [June 12, 2015], p. 329. Available at www.dod.mil/dodgc/images/law_war_manual15.pdf (Accessed June 2016.)

> *The law of war does not specifically prohibit or restrict the use of autonomy to aid in the operation of weapons. In fact, in many cases, the use of autonomy could enhance the way law of war principles are implemented in military operations. For example, some munitions have homing functions that enable the user to strike military objectives with greater discrimination and less risk of incidental harm. As another example, some munitions have mechanisms to self-deactivate or to self-destruct, which helps reduce the risk they may pose generally to the civilian population or after the munitions have served their military purpose.*
>
> *DoD Law of War Manual*
> *Section 6.5.9.2. No Law of War Prohibition on the Use of Autonomy in Weapon Systems.*

- Autonomous weapon systems, which upon activation can select and engage targets without human intervention

Extensive verifications, validation, test, and evaluation are required before fielding autonomous weapons systems. The use of automated regression testing of system software is also recommended, acknowledging the difficulty of full path regression testing in learning systems. The principles of military necessity, distinction, discrimination, and proportionality apply, according to the law of war.

In spite of the clarity provided in official documents, they alone are not sufficient for allaying public concerns about the use of autonomous weapons. Recent statements by prominent scientists and technologists are attempting to promulgate the notion of dire consequences due to the rapid adoption of artificial intelligence and autonomous robots.[24,25,26] Some of the published statements are careful to use the qualifier fully autonomous weapons, but the distinction can be easily lost in communications. The potential for a backlash against the introduction of non-lethal and semi-autonomous systems for military use could grow.

An integrated approach is needed

Establishing—and maintaining—trust in autonomous systems requires a broad view of the entire lifecycle of the system. This principle is idealized in Figure 7 as a continuous process (purple arrows), which begins with experimentation and development of doctrine and CONOPs, followed by specification of operational requirements, and proceeds to system design, development, testing, training, operations, and maintenance. While all these functions are part of any normal process to

[24] R. Cellan-Jones, *Stephen Hawking warns artificial intelligence could end mankind*, BBC News. [December 2, 2014]. Available at www.bbc.com/news/technology-30290540 (Accessed June 2016.)

[25] Future of Life Institute, *Autonomous weapons: An open letter from AI & robotics researchers* [July 28, 2015]. Available at futureoflife.org/AI/open_letter_autonomous_weapons (Accessed June 2016.)

[26] International Human Rights Program at Harvard Law School, *Advancing the Debate on Killer Robots: 12 Key Arguments for a Preemptive Ban on Fully Autonomous Weapons* [May 2014]. Available at hrp.law.harvard.edu/wp-content/uploads/2014/05/Advancing-the-Debate_final.pdf (Accessed June 2016.)

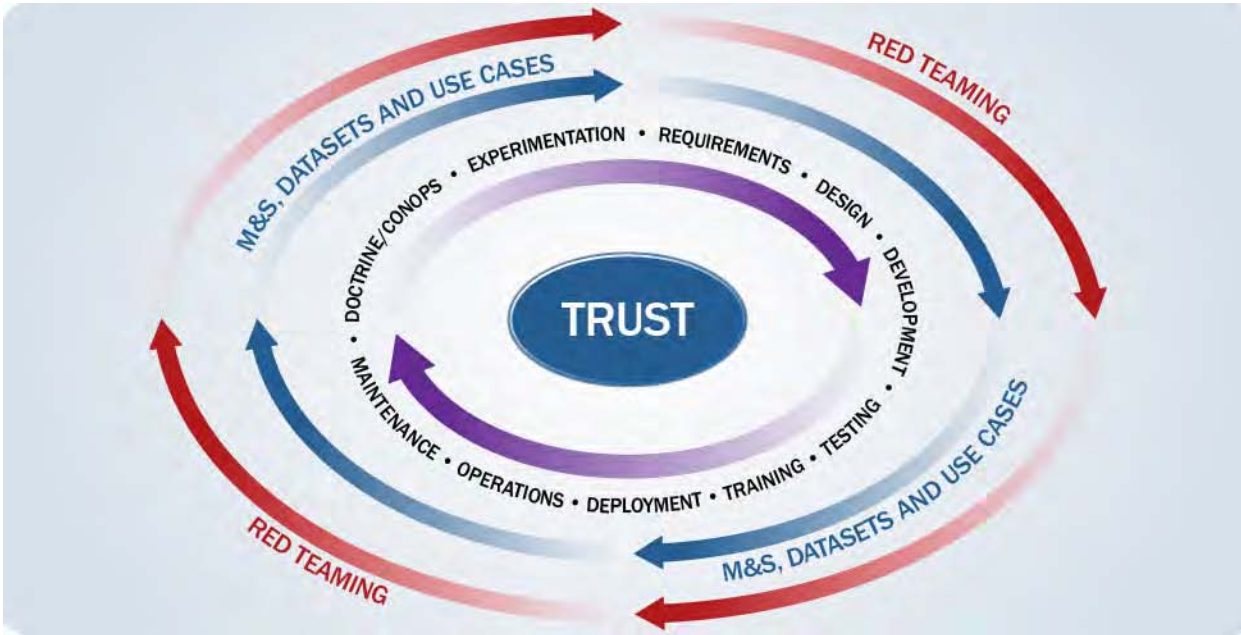


Figure 7 **The lifecycle of the systems needs to establish an appropriate calibration of trust in autonomous systems.**

field a new system, the distinction for an autonomous system is that the sequence is not linear, but a continuous process spanning the entire system lifecycle.

Because autonomous systems are sensor- and software-intensive, rapid upgrading of subsystems will be vital for maintaining military advantage. The full lifecycle process, as conceptualized in Figure 7, efficiently integrates upgraded subsystems into operational systems. Moreover, the learning that is inherent in autonomous system development and employment will couple into evolution of doctrine and CONOPs, modifications to the system, and additional testing. There may be many sub-loops in the realization of this process, but the concept of the continuous loop throughout the system lifecycle is of primary importance.

The surrounding circle of M&S (blue arrows) depicts the need for M&S integration throughout the cycle, from initial concept to operational test and evaluation and through operator training. A continuing and pervasive thread of M&S activities can support rapid evolution of system design and performance. This will require a transformation of the conventional model of developmental T&E and operational T&E from discrete segments of the acquisition cycle to an ongoing evaluation and evolution of the technology and concepts within the operational community. Military operators at all levels must become more familiar with and begin to employ T&E concepts and techniques within routine training operations. In addition, for any given development cycle, autonomous systems that learn can take advantage of datasets generated during previous development and training cycles to build on past successes—and learn from failures.

The outer circle for red teaming (red arrows) depicts the need for this in all stages of the cycle, from early concept development, to developmental and operational test and engineering, fielding, and training. The rapidly advancing state-of-the-art in autonomy, together with the software-intensive and embedded processing characteristics of any autonomous system, whose functionality may evolve over time, introduce many more attack surfaces that will not all be discovered prior to deployment.

In conclusion, assuring appropriate calibration of trust assessments in the implementation of autonomy in DoD systems is clearly central to their adoption. The decision for DoD to deploy autonomous systems must be based both on trust that they will perform effectively in their intended use and that such use will not result in high-regret, unintended consequences. Without such trust, autonomous systems will not be adopted except in extreme cases such as missions that cannot otherwise be performed. Further, inappropriate calibration of trust assessments—whether over-trust or under-trust—during design, development, or operations will lead to misapplication of these systems. It is therefore important for DoD to focus on critical trust issues and the assurance of appropriate levels of trust.

3 Accelerating the Adoption of Autonomous Capabilities

Enabling transition to accelerate innovation encompasses many cross-cutting issues for the entire DoD enterprise. The study proposes an interdependent set of enterprise-wide recommendations that focus on critical enablers essential to accelerating innovation in the use of autonomy.

Tackling engineering, design, and acquisition challenges

Autonomous systems can be cyber-physical or totally cyber-dominated. In any case, these systems will be dominated by a software architecture and integrated software modules. The DoD historically has had difficulty in specifying, developing, testing, and evaluating software-dominated systems.

Autonomous systems present a new level of complexity for the acquisition community because of the potential for an evolving role of the human operator as part of a human-machine system. Additionally, the fact that some autonomous systems will be adaptive and have the capacity to learn from the environment in which they operate will require much higher levels of trust demanded by operational users, policy makers, and the public.

Prior to and during the requirements development process, a set of key decisions should be made as early as possible that will be critical to control cost, schedule, risk and vulnerabilities for the proposed autonomous system. Those decisions will require early definition of the functional modularity of the system that will allow for subsystem iteration based on technology advances and potential system learning. The functional architecture of the system must define the role of the human but also allow for an evolution of the functions performed by the human and those performed autonomously. Establishing the functional architecture early while allowing for system evolution need not be inconsistent objectives if the functional architecture is established at the appropriate level and the development process is flexible and can allow for the rapid insertion of new modules that allow the various subsystems of the overall autonomous system to be iterated independently.

Because autonomous systems are dominated by software and adaptive software architectures, cyber-security becomes even more imperative. Technologies that support cyber-security must be architected within the system from the outset, and they must be flexibly capable of being updated during the development process as new cyber-attack surfaces are detected. Cyber-security must also be integrated into the system as new capabilities are added since every new capability brings with it a new vulnerability.

Good systems performance seldom occurs in the absence of good, informed design. For autonomous systems, designers must anticipate a substantially broader range of operational circumstances. They must reconcile the fact that the system may be required to learn and adapt, so its specification is, to a degree, non-stationary. Experience has shown that the more constrained the design space, the more successful the acquisition will be (but the operational utility and system lifecycle may be limited). Conversely, the more fluid the requirements, the more likely it is that the acquisition will be both less successful and costlier; poorly designed systems will be more expensive in a changing environment.

A consequential decision is how the requirement is parsed into subsystems. A modular system holds the promise that the overall system performance can be improved by changing out a module or two with little adverse effect on the other component modules, and without the need for full regression testing of the entire system. Ideally, the functional modules are sufficiently granular to allow system upgrades that represent significant advances in performance. A modular architecture can allow for slower-changing core functions and faster-changing peripheral functions. This describes good engineering systems design processes.

Autonomous systems allocate and share functions among humans and machines; this will influence functional modularity and decisions about system design. Users who generate requirements will often want functional allocation between humans and machines to be fluid with the expectation of enhanced autonomous performance as the system evolves. At the early stages of system requirements definition, the fundamental allocation and sharing of human-machine functions must be established, and the modular system design must allow for increased autonomous performance. These are critical design issues, which may increase initial system schedule and cost—while at the same time enhancing lifecycle effectiveness and reducing lifecycle costs. [An extreme—and costly—outcome might be to design the system to have the capability to be either fully human controlled or completely autonomous.] The use of modeling and simulation during the requirements development process will allow system designers to establish the functional modularity of the system and the evolution towards extended autonomous functionality while staying within cost and schedule.

Additionally, as early as possible, a comprehensive modeling and simulation capability, system use cases and the necessary data sets to allow for system design, modular architecture development, validation and verification of the system, and a continuous test and evaluation process must be defined, budgeted for, and implemented. Throughout the definition, requirements development, development, validation and verification, testing and evaluation, and operational fielding of autonomous systems, the capability for red teaming must be provided to address system vulnerabilities and establish system performance boundaries.

Engineering effective human-machine collaboration

Human-machine interaction is a highly specialized discipline with major branches for interacting with software ("autonomy at rest") and with software-augmented hardware ("autonomy in motion"). It provides a framework for analyzing and assigning functions and tasks across teams of humans and machines, as well as providing guidelines for designing future systems with complex, adaptive, and dynamic modes of interaction with individual or human teams. The community includes theorists and practitioners from a number of areas, including human factors engineers, cognitive scientists, artificial intelligence researchers, and controls theorists, among others.

A major problem throughout the history of automation has been the human out-of-the-loop control problem, where a task or process encounters a problem and a human who is doing some other task has to suddenly take control. The inability of a human to suddenly change mental gears and successfully diagnose a complex problem currently being addressed by the automation is well documented.

Alternatives to this are being developed, especially when exerting human control on an autonomous system is not an option. These include providing an explanation capability, both for the system to "explain" what it is doing and why, as well as a means for the human to "explain" what it means when a command may be vague or ambiguous, or when a situation may have changed from what had been anticipated. Complementing this explicit explanation capability is an implicit understanding of each other's mental models (of the system by the human, and vice versa), in terms of goals held, situational awareness, decision plans, and so forth.

In addition, system self-awareness can provide for additional system robustness, through awareness of self-health and of the environment, and an understanding of where the system is operating with respect to its "envelope of competence" (or operating envelope). If the system includes learning, interactions with human operators would be facilitated if the system came with a design and training pedigree to help human teammates and supervisors anticipate novel system behaviors as the system evolves in reaction to past experiences and training.

Defending blue autonomy by red teaming

In the current context, red teaming is the relentless search and extirpation of vulnerabilities in one's own systems. This single-mindedness is essential to the resilience of our autonomous systems. As an added benefit, red teaming may provide significant insight into vulnerabilities in the opposition's autonomous systems, possibly unlocking an adversary's autonomous system without direct access to the article.

The ingredients for successful red teaming include domain knowledge—an understanding of the mission and functions of the system, as well as the larger military purpose it was designed to serve. This means that a red team is intimately related, though still independent in thought, to the acquisition program or the operational element. It also means that red teaming expertise is distributed and often compartmented.

Because red teaming, in practice, is necessarily distributed, there is a benefit in establishing a community of interest (COI) to grow the art and practice while still acknowledging the need to know specific topics. Red teaming will be essential to all stages of the development of autonomous systems.

Recommendation 1.

USD(AT&L) should require that the following practices be developed and applied to all software dominated systems and, in particular, autonomous systems:

- Software designed to best engineering practices, and for incremental upgrades that can be implemented without full system regression testing
- Iterative development of subsystems

- Early decision on functional modularity to allow the system and the role of the human to evolve
- A framework for analyzing and assigning functions and tasks across teams of humans and machines
- Planning and budgeting for reliable datasets, use cases, modeling and simulation, validation and verification, testing and evaluation, user engagement, and self-monitoring at the earliest stages of autonomous system development
- Red teaming at all stages of autonomous system development

Mitigating cyber issues

Autonomous functionality—basically decision-making—in a system resides in software replete with branching logic and tables of variables and parameters which, together, model the mission to be accomplished, the environment in which it must be executed, and the conditions that pertain. The more complex the mission and the more diverse the environment, the more extensive and complex is the software. Typically, too, autonomous systems will have organic sensors, a considerable corpus of stored information, and optional communication for some supervisory functions, along with a capability to receive and implement over-the-air updates. Insofar as they are mobile, they will implement precision, navigation, and timing (PNT) and collision avoidance. Additionally, there may be self-diagnostics and contingency fail-safe provisions.

Cyber intruders view such systems as target rich, rife with capabilities that also introduce vulnerabilities. Cyber defenders will worry about the extent of the cyber-attack surface, which might include "one touch" access at a number of stages in the lifecycle from the drawing board to the battlefield, remote network access, or entry points via the onboard sensor suite.

Exacerbating concerns about cyber complexity and vulnerability is the premier characteristic of autonomous systems: they have a wide range of consequential actions up to and, perhaps, including lethality, and may decide upon these actions with little or no human supervision. There may be provisions made for the machine to wrest control from the human when, as designed, it believes it knows best; is acting to protect the health and safety of humans or machines; or is given ambiguous, contradictory, or paradoxical instructions by its human supervisor. Curiously, when the machine assumes control under such circumstances, we say it is "out of control"—literally correct, but emotionally loaded.

In general, adversaries seek, and defenders worry about three rather different attack objectives—confidentiality, integrity, and availability. The first objective, breaching confidentiality or stealing secrets is a lesser concern here, although the autonomous system will admittedly be loaded with items of intelligence value: commander's intent, mission orders, and target parameters, as well as elements of doctrine, CONOPs, and rules of engagement (ROE). The second and third objectives, availability and integrity, while less revealing, are nevertheless the essence of mission assurance—will the system be available when you need it and perform as designed and directed?

Corruption of these aspects can result in bad judgments, missed opportunities, inappropriate targeting, self-destruction, system abort and return to base, and so on.

The vulnerabilities that an adversary might exploit could result from poor design or implementation. They could have been introduced by an adversary who had even momentary access— physical or virtual—to the system. They also could stem from the design requirements: for example, the desire for fail-safe and return-to-base modes subject to compromise by an adversary.

Autonomous systems of the class under consideration may also have the capability to adapt and learn from experience—that is, adjust certain decision parameters like the estimated prior probabilities and presumed costs and values of potential outcomes. And, of course, machines will be expected to modify their decisions and actions in accordance with dynamic behavioral guidance. Such capabilities open the door to adversary manipulation. By presenting a misleading set of circumstances they might "mis-train" the system.

The bottom line is that capabilities almost always entail vulnerabilities, and autonomous systems will be quite capable. In addition, their software—indeed, their logic—will be quite complex, making it difficult to validate and verify actions and to root out induced vulnerabilities should the adversary have gained access.

The Department of Defense is working hard to respond to incessant attacks on its enterprise systems and those of its contractors (the defense industrial base), as well as to moderate the attendant publicity. Now it must consider how to ensure the viability of the autonomous systems it will field, both to guarantee mission success and to avoid further loss of public confidence.

There are important differences, to be sure, between the two situations. First, the enterprise systems attacks have breached confidentiality and secrets have been stolen. While regrettable, the damage seldom rises to that which would ensue if key autonomous military systems did not operate when and as directed. Second, the enterprise systems are heavily indebted to commercial off-the-shelf software and, to date, the marketplace has rewarded capabilities at the expense of vulnerabilities. Autonomous military systems are more likely to be under DoD control, even if commercial components are included, which provides a better chance to build in more cyber resiliency. Third, erosion of public confidence in autonomous systems—all the more critical if lethality could be involved—may seriously derail an otherwise desirable move toward autonomous systems.

These differences provide a challenge and an opportunity for DoD and the defense industrial base. A number of steps have been proposed to protect the integrity of autonomous systems throughout the entire lifecycle—drawing board to battlefield—including:

- Examining the requirements to eliminate unnecessary vulnerabilities
- Designing and building to the requirements using best practices
- Running an aggressive counterintelligence program to counter insider threats
- Ensuring adequate time, resources, and incentives for validation and verification
- Red teaming early and often
- Securing the supply chain

Operational considerations must anticipate the adversary and include:

- Prioritizing critical communication channels and integrity for C2 and over-the-air updates
- Planning for changing sources of reliable PNT
- Including sensors and processing to overcome spoofing
- Examining sensor inputs and providing reality checks

Both the National Security Agency (NSA) and the U.S. Cyber Command (USCYBERCOM) can play a pivotal role in finding and eliminating vulnerabilities, largely cyber, in the design and application of autonomous military systems. NSA has the preponderance of experience and firsthand intelligence access. The NSA Director adjudicates cyber offense-defense and, as Commander of USCYBERCOM, executes cyber operations. This experience base provides the necessary fundamentals for cyber red teaming.

Recommendation 2.

USD(AT&L) should address the special issues associated with cyber resiliency in autonomous systems, which include:

- Ensuring DoD-wide practices for cyber hygiene throughout the development process
- Mitigating initial introduction of vulnerabilities through requirements, design, and supply chain best practices
- Requiring system self-monitoring, redundancy, and baseline comparatives to identify spoofing, unauthorized access, or intrusion that alerts the operator
- Incorporating cyber issues into overall red teaming efforts
- Working with NSA and USCYBERCOM to harvest, integrate, and disseminate the science, distributed knowledge, expertise, and best practices of red teaming in the cyber domain

Creating new test, evaluation, modeling, and simulation paradigms

Autonomous systems present a number of challenges with regard to T&E, M&S, and related analysis. Most systems are continuously monitored by humans who can note deviations from desired performance and correct the behavior of the system. Autonomous systems may have periods of time with limited or no communication capability; during those periods the system must reliably behave in known ways to the full range of stimuli that the system is designed for. Insuring that the system will respond appropriately to all of the possible inputs will exceed the capability of conventional testing. It will require using a combination of modelling and simulation to explore thousands of test cases, statistically measuring system performance against the desired standard, then doing real world testing of the system to ensure that the modelled and real world behavior match for corner cases that span the range of system performance. This approach has been successfully used in commercial systems; the fidelity of the modelling and simulation must meet the requirements of the program and the modelling and simulation parameters must match the actual parameters of the system.

Maintaining accurate model parameters requires that attention be given to the process by which the parameters are chosen and changed. The Department also needs to ensure that it will have the full sources of all of the models and data available for its use. In some current DoD programs, models are generated in order to size the system and explore alternatives, but these models are not kept current with the system as design changes are made during development and are also not kept current throughout the life of the system. Using M&S and live testing early to evaluate the performance of the system with respect to the requirements allows for rapid cycles of model, build, test, and modify, to advance the design and performance of the system. The testing should include the expected and predicted adversary capability that may be used to thwart the system.

The act of modelling the system and field-testing will generate significant databases. This data provides deep insight into the capabilities of the system, and must be appropriately safeguarded. The data will be important in enhancing and upgrading the system, as well as in designing new systems. This valuable data must not be allowed to fall into adversary hands.

Test and evaluation for software that learns and adapts

DoD's current testing methods and processes are inadequate for testing software that learns and adapts. Because such software exhibits different behavior as it incorporates more data about its task, and learns to provide better results partly based on experience, such software cannot be exhaustively tested. There is no single, known correct answer. With experience, the software may discover unexpected, useful patterns that lead to better, different answers. Such software should be routinely evaluated as its behavior changes throughout its lifecycle. Consequently, testing should be conducted throughout the development and deployment processes. See Box 2 on page 31 for an example of this concept.

Commercial software developers have developed methods that permit software to be developed and tested in increments. After core functionality is developed, the software can be tested and put in the hands of users with increments of function tested and released incrementally. The lifecycle encompasses a long-lived sequence of incremental upgrades interspersed with test. This development approach delivers the software very early to users to gain early user feedback. As discussed earlier, the DoD has a rigid testing process—both for development test and operational test–which is in direct conflict with more advanced commercial development practices that are better suited for testing adaptive software. DoD's strong separation between developmental testing and operational testing is in conflict with the best known methods for managing the development of such software. Operators will have to change their mindset from expecting weapon systems that "just work" out of the box to systems that require their time and effort into shaping their ever-evolving instantiation, but will ultimately be better customized to their mission, style, and behaviors.

Box 2. Logistics as a Testbed for Software that Learns

Logistics planning and execution is a particularly good candidate application for T&E to experiment with new test methods for learning, adaptive software because the behavior of logistic software can be evaluated against crisply known metrics. Given inventory available at multiple locations to fill a customer requisition, there may be multiple logistics plans that are acceptable. Each of those plans may be more or less attractive considering dimensions beyond simply filling the customer's requisition, *e.g.*, pallet packaging, container packing plans, urgency, and transportation constraints.

Core functionality of logistics software can be delivered as soon as it is developed and additional function can be delivered in later releases. An example later function would be anticipatory order filling based on knowledge of the customer's mission goals, priorities, and preferences or incorporating near real-time knowledge about the customer's situation, *e.g.*, impending bad weather a week hence. A logistics system could then anticipate customer needs rather than simply filling requisitions.

A useful testbed will allow continual testing during development and then through the software system's lifecycle. Operators can be involved early to allow their evaluation of the logistics software, even with only core functionality, made known to the developers. The user interface is particularly important with adaptive software because the user needs to not only interact with the software, but to understand what it is doing, and occasionally to guide it or select among options that it proffers. Early user involvement will yield better interfaces, and inevitably, better functionality.

This method of testing that intersperses testing and code releases requires disciplined configuration control to permit frequent software updates, with routine, controlled rollback as needed when system behavior is not acceptable to the users. Testing can be aligned to match upgrade cycles to enable frequent incremental releases.

Such testing requires a realistic, synthetic environment that incorporates models of the environment, and even of human frailty when planning and executing logistics missions. The environment needs to be able to present scenarios to the software under test, and to evaluate the output of that software for each scenario. Classic regression testing might be used early, but the adaptive nature of the software under test requires that the testing infrastructure become increasingly knowledgeable as more functionality is added. An effective synthetic testing environment will collect accuracy and quality metrics and automatically track improvement (or lack thereof) over time.

Logistics is a good application for early experimentation because the behavior of the logistics system can, to a great extent, be evaluated rapidly using known and straightforward metrics. Likewise, given knowledge of inventory at hand, a requisition, and the plan produced by the software, expert logicians can readily judge the quality of that plan produced by the software.

With a successful logistics demonstration, lessons learned can be evaluated so these new test methods for software that learns can be applied to other applications that are more difficult to test and evaluate.

DoD will increasingly utilize software that learns and adapts for diverse applications. Such software incrementally enriches its database to describe relevant context, environment, threats, user inputs, and mission objectives. It records new input, then integrates and generalizes past experience to make decisions partly based on the accumulated data and experience. The software may record its reasoning and can report out how it derived intermediate decisions. With machine learning algorithms, more data is usually better, and the learning algorithms are expert at finding that data which is useful, and ignoring the irrelevant. For these reasons, it can be useful to transfer data from one instantiation of software application to another so that there is a broader base of data from which the software can learn. Additionally, reusing building blocks of previously evaluated and validated software modules will reduce the required time and resources for follow-on implementations of autonomous solutions.

Recommendation 3.

The Director of Operational Test and Evaluation (DOT&E), in conjunction with the Office of Developmental Test and Evaluation (DT&E), should establish a new T&E paradigm for testing software that learns and adapts. Considerations include:

- Opportunities to adopt or adapt commercial best practices in T&E of learning systems
- Experimentation and development of new methods and means for testing software that learns, such that the "correct answer" changes with context, experience, and new data

A more continuous and iterative approach to address validation and verification

As noted in previous sections, systems that learn or adapt present a significant challenge to the current T&E process because system behavior will depend upon the sequence of stimuli that the system receives. Research is required to develop approaches to test and evaluate the readiness of learning systems to be used by the warfighter. Humans serve as good models of learning systems that we have trust in. A similar approach to qualifying non-human autonomous systems may be used as a starting point for the research.

In non-learning autonomous systems, emergent behavior complicates the test and evaluation of the system. The emergent behavior will be manifested when groups of autonomous devices are presented with similar stimuli, causing them to act in a desired coordinated manner. Insuring that emergent behavior occurs in the desired ways and cannot be induced to occur in ways that would be detrimental to the system performance presents a significant T&E challenge. Research is required to understand the best ways to test for emergent behavior characteristics.

Recommendation 4.

The DoD test and evaluation community should establish a new paradigm for T&E of autonomous systems that encompasses the entire system lifecycle. Considerations include:

- Make extensive use of live and synthetic environments for evaluating and qualifying transition from development to fielded systems
- Establish standards and guidelines for continuous verification and validation (V&V) for autonomous systems
- Start testing early and iterate in development, operational testing, and fielding: "build, test, change, modify, test, change…."
- Develop datasets for assessing autonomous functionality, and expand based on live test validation results (experimentation can add to the datasets)
- Include expected adversary-induced environments, *e.g.*, cyber, electronic warfare, etc.
- Plan for involvement throughout the lifecycle of the system—the system will learn and self-modify, or be intentionally modified

Expanding the roles for modeling and simulation

Modeling and simulation are frequently used to explore the design space of a system, to predict the performance of the system, and to understand the limitations of design alternatives. Models typically are updated during the development process to better represent the system that will be built. These digital experiments are useful in understanding the design trade space of the system.

M&S is also used for experimentation and refinement of system requirements. For more sophisticated autonomous systems, M&S is needed to augment test and evaluation by simulating the testing environment and creating, running, and measuring thousands of random but realistic scenarios. By providing statistical evaluation criteria and behavior measurement, M&S has been shown to be effective in exploring the large scenario space in which a system is designed to operate. Thousands of test cases can be generated and simulated to increase confidence that the system will satisfy design requirements. Israeli Aerospace Industries has used the Gazebo simulation tool, which is built on the robot operating system (ROS), to perform such simulations and has shown that it is an effective means of performing a large number of digital experiments to gain a statistical measure of system behavior.

During the recent Robotics Challenge organized by the Defense Advanced Research and Projects Agency (DARPA), the use of M&S was further extended. By running the Gazebo simulator at the user workstation, the operator could watch a simulation of the robot's performance that would predict the robot's action when the communication channel was not able to provide real time updates, thus enabling operators to more effectively intervene when needed.

A simulator can also be run on the robot itself. This allows the robot to have an on-board monitor that can ensure that the robot stays within a set of operating rules. Incoming command sequences can first be run on the simulator to ensure that the outcome meets an established rule-set

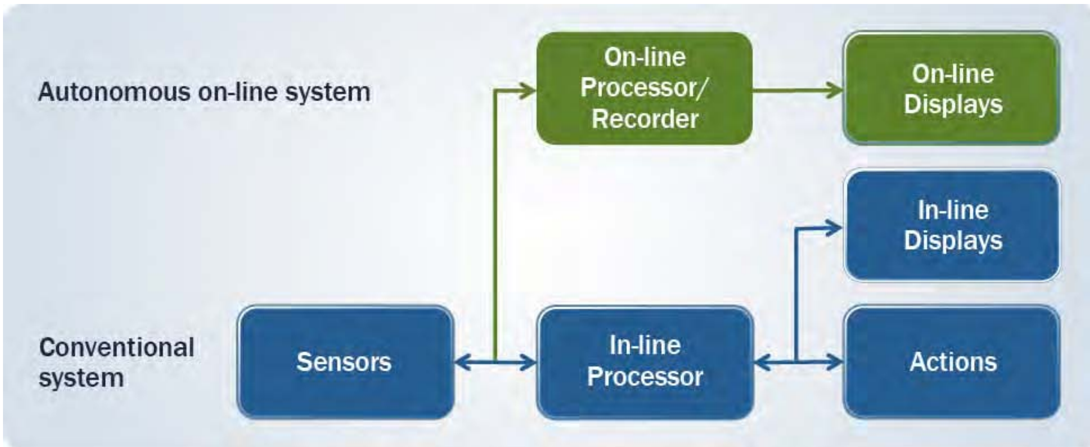


Figure 8 On-line processors implement new functions for system V&V and performance enhancement.

prior to execution by the robot. In addition to making sure that the robot does not perform an action that would cause undesired damage, this can help detect malicious commands. Finally, if the simulator determines that the robot will be forced into a situation where the machine will be compromised, it can force the erasure of key data and programs or take other actions to prevent the system from being reverse-engineered.

In a conventional system, the in-line control processor provides input to the in-line displays and system actions. For some systems, such as the Aegis Combat system, an additional processor has been added, termed an "on-line processor or recorder," as shown in Figure 8. The on-line processors implement new functions and allow the operator to see the effect of the new function prior to it being implemented in the in-line system.

The approach of adding a second processor allows implementation of a number of new functions that are useful over the life of the system. In addition to demonstrating new functions, it can enable command validity checking and facilitate mission rehearsal. It can also provide a black box recording function to capture data for subsequent analysis of the inputs and decision sequence that the system implemented.

The secondary processor may also be useful in allowing explanatory dialogue of the robot actions and allow for new collaborative and cooperative behavior. Additional on-line functions may include advanced autonomy algorithms to provide local diagnostics, alternative courses of action, and explanatory dialogue.

Recommendation 5.

USD(AT&L) should require the acquisition community to establish and implement a consistent and comprehensive M&S strategy throughout the lifecycle of the system, including:

- Alternative system approaches and implementations
- Design and prediction of system performance
- Performance of test and evaluation
- Support for manufacturing
- Support for training
- Support for operations and sustainment
- Exposure of vulnerabilities at all stages from requirements development
- Ensure the harmonization and coordination of existing models and simulations across the acquisition and training communities

Integrating technology insertion, doctrine, and concepts of operation

During recent conflicts, unexpected threats on the battlefield spurred efforts to pursue rapid-equipping initiatives. The goal was to deliver relevant technology to the battlefield as soon as practical. Among initiatives launched to expedite solutions was the creation of Joint IED Defeat Organization (JIEDDO), charged with finding solutions to serious improvised weapon threats.[27] These accelerated acquisitions often followed the "try a little, buy a little" development and proof-testing model. Some efforts were more fully integrated, with developers and users working together during all phases of the expedited procurement—from design and modeling, through simulation, prototyping, testing, CONOPs and TTPs (tactics, techniques, and procedures) development, and issuance to the field—all performed in an overlapping, parallel joint atmosphere. Ongoing operational feedback improved effectiveness as adversaries adjusted their own attack TTPs.

The Rapid Equipping Force Initiative provides solutions to unanticipated needs and is a useful example for expediting the development and fielding of military semi-autonomous and autonomous systems. As the DoD moves to peacetime acquisition processes, there is a need to create and institutionalize an accelerated acquisition system to cope with threats that are developing very quickly using globally available technology to conceive and build capabilities that are often inexpensive, broadly available, and potentially lethal.

Today, many more tools are available to develop, test, and assess progress, delivering constantly improving capability in modeling and simulation. The goal is to create an environment where all the developmental and operational players are involved from the start rather than sequentially scheduled into a linear program evaluation and review process. With this change, the developer sees better ways and means to deliver the stated requirement, and the user develops early CONOPs and TTPs, while seeing new potential capabilities and uses not previously conceived in

[27] JIEDDO is currently called the Joint Improvised-Threat Defeat Agency (JIDA).

the requirements document. This synergistic cooperation continues through prototyping and field test where new possible use cases will continue to emerge.

This presents opportunities throughout the conceptual design and development process to make tradeoffs in capability, performance, cost, and schedule. In the standard linear model, such opportunities are not generally visible until much later in the acquisition process. This parallel approach allows the operator or user to develop draft CONOPs and draft TTPs alongside product development processes.

This approach—joint development and operational conceptualization—argues that time from concept to initial fielding should be the primary performance metric, recognizing that the global market is rapidly introducing new dual use capability with countless applications, recreational as well as potentially nefarious.

Recommendation 6.

Military Service Chiefs should integrate technology insertion, doctrine, and CONOPs by:

- Ensuring early experimentation for autonomous systems includes multiple methods:
  - Modeling, simulation, and operational gaming
  - Prototype testing employing hardware and software modifications to existing systems to allow autonomy investigations
  - Developing unique prototypes that explore new capabilities not possible in systems designed to be controlled by humans
- Employing alternative sources throughout the experimentation process:
  - Use available commercial systems to allow early experimentation
  - Strongly consider Military Service Laboratories, federally funded research and development centers' (FFRDCs), and university affiliated research centers' (UARCs) participation in prototype development and red teaming
- Ensuring that field experimentation with developmental autonomous hardware and software informs employment doctrine and CONOPs:
  - Designing and fielding early test hardware examines difficult trades early in the systems effort
  - Accelerating system maturity validates technical and operational expectations

Developing an autonomy-literate workforce

American military forces, formerly equipped with largely electro-mechanical platforms, are now fielding systems that are dependent on software for combat effectiveness. This technology shift has placed a huge demand on education and training to provide qualified people across all aspects of the economy—a demand that is far from satisfied and is growing.

Skilled technical people are essential to develop and measure accelerated joint technical and operational development and later fielding. Such an undertaking is data-heavy in all phases, from design, through modeling, simulation, validation, verification, tech insertion, and operational concepts and tactics, techniques, and procedures.

With commercial development the technology leader, it is also a more effective competitor for talent. The military can train its own but is at a serious disadvantage to retain experience—talented operators, maintainers, supervisors, and technology leaders.

Programs need to be developed that formalize broad exchanges between government, military, and commercial enterprises for extended periods—closer to months rather than days—so that both government and commercial personnel can learn and understand emerging technologies and capabilities as well as the range of user concepts and applications.

Sustaining an autonomy literate workforce challenges both the commercial enterprise and the government. The demand for such talent spans the U.S. economy, across a growing number of industries. In addition to finding new and innovative ways to establish broad personnel exchanges across the commercial, government, and military divides, the military needs to identify and functionally manage the currently small but essential cohort of personnel who are experienced and knowledgeable about autonomy, both technical and operational.

Such necessary measures may include creating a military service career identifier, insuring their continued assignment in the autonomy field, categorizing autonomy trained personnel in the highest pro pay category, and offering significant re-enlistment bonuses and officer retention bonuses. These steps are necessary but insufficient measures that need early implementation.

It is also worth assessing all four Military Service recruiting screening processes, testing, and selection for identifying the right inductees with the requisite talent and capability for assignment to the autonomy function or related cyber career fields.

New approaches also need to be considered for the training of reserve forces, both National Guard and the Reserve, who are currently assigned to units accomplishing autonomy-related missions. Some already work in commercially related enterprises as their day job. Why not assign them annual training to commercial enterprises in the autonomy domain? Or, take advantage of Reserve personnel who work in the intelligent system industry? Such measures could create a sustaining link that benefits both parties—the developer and the user—and provides focused quality training to the Reserve component.

Finally, the DoD should develop a formal program to regularly draw key members of relevant commercial autonomy enterprises to serve in Defense Agency and Military Service postings, especially at the operational level where they should be maximally exposed to field operations and exercises in order to understand military employment concepts and future needs. All available incentives should be used to attract such personnel, to include broad use of the Intergovernmental Personnel Act (IPA) process to compensate at rates as competitive as possible.

Recommendation 7.

The Undersecretary of Defense for Personnel and Readiness (USD(P&R)), working with USD(AT&L) and Military Service Chiefs, should develop an autonomy-literate workforce by:

- Establishing standing relationships with the commercial sector to be technologically aware:
  - Assign DoD civilian and military personnel to work within commercial sector on temporary rotating assignments
  - Ensure some National Guard and Reserve personnel do their training within commercial industry
  - Ensure National Guard and Reserve personnel who work in the intelligent systems industries can use their experience during military assignments
  - Immerse commercial technical personnel in operational field environments and exercises
- Establishing Military Service career identifiers for key specialists and operational employment experts:
  - Align recruiting tests and qualifications with finding the best candidates to employ autonomous systems
  - Implement performance excellence recognition (pro pay, etc.) programs with persuasive retention incentives

Improving technology discovery

As covered in the introduction to this study, autonomous systems and enabling technologies comprise a growing, global commercial enterprise. Furthermore, R&D to advance the state of the art is also occurring on a global scale. As a result, DoD will need to be exquisitely aware of capabilities in the commercial sector and the possibilities of how they might be used outright, or modified or adapted for military use.

Part of the approach for achieving and maintaining global awareness is already well established in DoD. Traditional "tech watch" programs, such as the Office of Technical Intelligence (OTI) within the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)), and the Military Services' 6.1 programs, such as the Office of Naval Research (ONR) Global and Air Force Office of Scientific Research (AFOSR) International, have been part of the Department's R&D community for many years. In addition, improved techniques for horizon scanning to improve the ability to anticipate advances is being pursued, such as efforts in data analytics and cluster analysis. The current level of effort in these organizations related to autonomy may or may not be sufficient to support a Department push to introduce new capabilities into the forces.

While tech watch programs are necessary, they will not be sufficient for timely understanding of the potential impact of advances, whether to more rapidly field capabilities or that might pose a serious threat to our mission success. The fast pace at which autonomy is moving, both technically and commercially, coupled with limited governance on the use of autonomous systems, is leading to many surprising and novel applications, as well as honing operator skills outside of DoD.

One effective way to compete in this rapidly evolving environment is for DoD to create and nurture a community charged to build on the findings of technology watch programs to prototype and experiment with commercially available technologies and systems in a military operational context. Such a community would not only explore new uses for autonomy, counter-autonomy, and countering potential adversary autonomy, but also more realistically inform what the tactical advantages and vulnerabilities would be to both the U.S. and adversaries in adopting or adapting commercially available technology. Such a program could also create options for insertion into current capabilities that might initially be too risky or too disruptive. Some of the best sources for participation in this effort are the government laboratories, and independent, not-for-profit laboratories, including the FFRDCs and UARCs, because of both the technical proficiencies of their workforces and their working knowledge of national security missions. Some of these individuals and organizations may have bridges to the commercial and academic sectors that can aid government programs.

Another way to expand beyond tech watch, and more importantly, to advance the quality and skills of the workforce is to arrange for sabbaticals for both civilian and uniformed scientists and engineers in commercial and laboratory organizations worldwide that are engaged in autonomy-related R&D. An important corollary effort should be the aggressive recruitment of IPAs from a wide array of outside R&D organizations to perform R&D in DoD.

A final ingredient for success in harvesting the fruits of a robust awareness program is a partnership with the intelligence community (IC). The team's results can provide cues for the IC on what to look for, while enhanced technical collection by the IC can better focus efforts.

Recommendation 8.

The Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) should improve global autonomy technology discovery by:

- Enabling civilian and uniformed scientists and technologists to do sabbaticals in R&D organizations in the U.S. and abroad to understand and practice the state of the art in autonomy
- Attracting more IPAs to perform autonomy R&D in DoD
- Engaging government lab, FFRDC, and UARC personnel to establish a robust program of autonomy technology demonstrators using globally available technologies to:
  - Explore new uses for autonomy, counter-autonomy, and countering potential adversary autonomy
  - Investigate the tactical advantage and potential vulnerabilities of both blue and red
- Coordinating with the IC to expand technical intelligence regarding global advances in autonomy with military relevance.

Improving DoD governance for autonomous systems

This section explores the techniques for the management of autonomous systems with the desire to establish whether management changes would assist with the acceptance of systems employing this rapidly changing technology. Explored here are different and nonstandard techniques successfully employed in analogous situations.

Experience indicates that nonstandard techniques should be employed only when the imperative of operational exploitation outweighs the disadvantages of establishing additional organizational structures. Exemplars such as the exploitation of low observables technology, cruise missiles, remotely piloted vehicles, and, more recently, electronic warfare demonstrate the imperative for use of nonstandard techniques. The discussion of imperatives for nonstandard governance for autonomous systems includes:

- Acceleration of development and fielding of autonomous systems—including complex systems that include embedded autonomous functionality
- Stay current with the speed, diversity, and global nature of relevant technology development
- Countering potential adversary use of autonomy

Over recent history alternative methods of acquisition have been employed that could be helpful in accelerating deployment of autonomous systems to the operating force. One management strategy was employed by JIEDDO to counter enemy use of improvised weapons in Iraq and Afghanistan. This approach employed a large permanent workforce with intensive management to ensure dedicated focus on the required developments. The centralized office controls finance and directs development and fielding.

This approach has also ensured the rapid introduction of cruise missiles and remotely piloted vehicles. The management of autonomous systems could similarly be assigned to an existing DoD organization, such as the JIEDDO, or could serve as the mission for a new organization. Generally, such direction is exercised through Military Service mechanisms to ensure transition to requisite Military Services at the appropriate time in the development process. While this process provides the best assurance of acceleration, successful transition to Military Service management has proven difficult.

The recommended approach is to establish an executive committee (EXCOM). This is a coordination mechanism comprised of Defense Agency and Military Service principals acting in their official capacities. The advantage of this approach is that the standard acquisition approach is accelerated by the intensive focus provided by the principles working through the EXCOM. Concentration of senior management on resolving financial, acquisition, and policy issues is an important benefit of this approach. This approach was successfully applied to the development and exploration of stealth and counter stealth, and is currently being employed for the electronic warfare community.

Recommendation 9.

The Deputy Secretary of Defense should establish departmental governance of autonomy by:

- Creating an EXCOM with the responsibility to oversee and ensure the development and fielding of autonomous systems:
  - Ensure funding is available to Military Services for near-term autonomous systems demonstrations and system acquisitions
  - Ensure an integrated approach to building trust (recommended by this study)
- Tasking Military Services to:
  - Establish advocates (*e.g.*, Deputy Under Secretary of the Navy for Unmanned Systems) for resourcing autonomous programs and move them into development and acquisition cycles
  - Charter program managers for each new system to implement all aspects of the recommended processes, *e.g.*, coordinated M&S, red teaming, use cases
  - Ensure the operators cover all of doctrine, organization, training, materiel, leadership and education, personnel, and facilities (DOTMLPF) for rapid adoption of autonomous systems

One of the most important aspects of accelerating the adoption of autonomy is shifting the underlying policy, legal, and cultural framework. An effective governance structure must coordinate efforts to implement these changes.

Recommendation 10.

USD(AT&L), the Under Secretary of Defense for Policy, and the Assistant Secretary of Defense for Public Affairs should take a proactive, two-pronged approach to anticipate cultural objections to the use of autonomy. In particular, they should instruct and require that each autonomous program:

- Establish and refine a communications plan that provides transparency into the trust building measures (*e.g.*, safety and security systems, information assurance, anti-tamper, audit trails) undertaken from the start of the development of every autonomous system
- Routinely engage the public to build confidence that the Department is acting in accordance with applicable treaties and the Department's policies

As a starting point for effective public engagement, promulgate the lessons learned from the Army's programs on chemical demilitarization and assembled chemical weapons alternatives. Together the two programs offer numerous guidelines for both what works and what does not in engaging the public interest and understanding.

Countering adversary use of autonomy

As has become clear in the course of the study, the technology to enable autonomy is largely available anywhere in the world and can—both at rest and in motion—provide significant advantage in many areas of military operations. Thus, it should not be a surprise when adversaries employ autonomy against U.S. forces. Preparing now for this inevitable adversary use of autonomy is imperative.

This situation is similar to the potential adversary use of cyber and electronic warfare. For years, it has been clear that certain countries could, and most likely would, develop the technology and expertise to use cyber and electronic warfare against U.S. forces. Yet most of the U.S. effort focused on developing offensive cyber capabilities without commensurate attention to hardening U.S. systems against attacks from others.[28] Unfortunately, in both domains, that neglect has resulted in DoD spending large sums of money today to "patch" systems against potential attacks. The U.S. must heed the lessons from these two experiences and deal with adversary use of autonomy now.

While many policy and political issues surround U.S. use of autonomy, it is certainly likely that many potential adversaries will have less restrictive policies and CONOPs governing their own use of autonomy, particularly in the employment of lethal autonomy. Thus, expecting a mirror image of U.S. employment of autonomy will not fully capture the adversary potential.

The potential exploitations the U.S. could face include low observability throughout the entire spectrum from sound to visual light, the ability to swarm with large numbers of low-cost vehicles to overwhelm sensors and exhaust the supply of effectors, and maintaining both endurance and persistence through autonomous or remotely piloted vehicles.

Autonomy also inherently provides a greater surface of vulnerabilities and opportunities that may enable countering these advantages. Using deception to confound rules-based logic, overloading the processing capabilities embedded in a vehicle swarm, or disrupting the adversary's supply chain all provide opportunities to limit or defeat the use of autonomy against U.S. forces.

Despite understanding that autonomy used against U.S. forces provides both a threat and an opportunity, DoD capabilities and knowledge in this area are fragmented, often compartmented, and provide little opportunity to benefit from both offensive and defensive technologies, techniques and programs. What needs to be done is better integrate these activities; share the knowledge gained from both sides of the offense–defense paradigm; and build a "ladder" for red teaming, with each rung informed by what has been learned on one side prior to the exercise, thus providing new knowledge and capabilities for the other side on the next rung of learning. Integration of both red and blue use of autonomy will thus help shape both U.S. offensive and defensive initiatives and responses.

[28] Defense Science Board, *21st Century Military Operations in a Complex Electromagnetic Spectrum* [2014]. Available at www.acq.osd.mil/dsb/reports/DSB_SS13--EW_Study.pdf (Accessed June 2016.)

Figure 9 Both inexpensive systems, such as the Flight Red Dragon Quadcopter (left), and more expensive ones, such as the Haiyan UUV (right), are becoming more capable and more available.

The U. S. will face a wide spectrum of threats with varying kinds of autonomous capabilities across every physical domain—land, sea, undersea, air, and space—and in the virtual domain of cyberspace as well.

Figure 9 (photo on left) is a small rotary-wing drone sold on the Alibaba web site for $400.[29] The drone is made of carbon fiber; uses both GPS and inertial navigation; has autonomous flight control; and provides full motion video, a thermal sensor, and sonar ranging. It is advertised to carry a 1 kg payload with 18 minutes endurance.

Figure 9 (photo on right) shows a much higher end application of autonomy, a UUV currently being used by China. Named the Haiyan, in its current configuration it can carry a multiple sensor payload, cruise up to 7 kilometers per hour (4 knots), range to 1,000 kilometers, reach a depth of 1,000 meters, and endure for 30 days.[30] Undersea testing was initiated in mid-2014. The unit can carry multiple sensors and be outfitted to serve a wide variety of missions, from anti-submarine surveillance, to anti-surface warfare, underwater patrol, and mine sweeping. The combat potential and applications are clear.

Figure 10 shows a variety of small UA characterized by their gross takeoff weight and the weight of the payloads they can carry. They lie on a line close to a 45 degree slope, meaning that a vehicle of x pounds can carry a payload of an equal weight. The Airborg H6-1500 (pictured) follows this trend, shown in the highlighted triangle. This is a more robust 1500mm hex rotor UA.[31] This vehicle has 6 26" carbon fiber propellers; an estimated flying time of 2 hours, at a maximum velocity of 40 mph, with a maximum payload of 9 kg (20 lbs); a maximum range of 160 km (100 miles); and can operate in wind/gust conditions up to 35 mph.

[29] iFlight Red DragonFly Quadcopter. Available at www.alibaba.com/product-detail/iFlight-Red-DragonFly-FPV-Quadcopter-Quadrocopter_60020379201.html (Accessed January 2016.)

[30] *China tests long-range unmanned mini sub* [June 29, 2014]. Available at www.china.org.cn/china/2014-06/29/content_32804788.htm (Accessed June 2016.)

[31] Top Flight Technologies, Malden MA. Available at www.tflighttech.com/products.html (Accessed June 2016.)

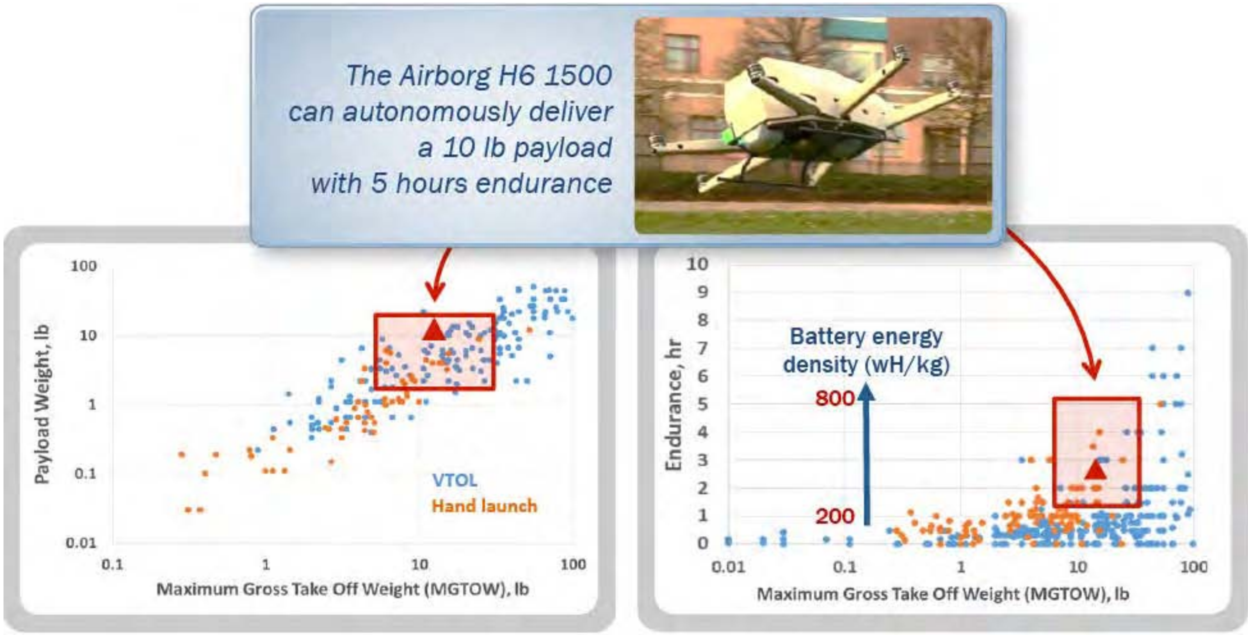


Figure 10 The Airborg (center top) capabilities are shown in the red boxes. The maximum gross take off weight of UA is compared with payload (left) and endurance (right).

Recommendation 11.

The Deputy Secretary of Defense should take immediate action to counter adversary autonomy, including:

- Direct USD(AT&L) to establish a counter autonomous systems community to develop and test counter-autonomy technologies, surrogates, and solutions:
  - Conceive, fund, develop, experiment, and demonstrate approaches
  - Develop adversary autonomous systems as "test targets"
  - Include full-spectrum expertise from cyber to directed energy to electronic warfare, cross-Service participation, and standing participation from the intelligence community
  - Create a standing countering adversary autonomy red team[32]
- Direct the Undersecretary of Defense for Intelligence (USD(I)) to raise the priority of collection and analysis of foreign autonomous systems:
  - Technologies important to advancing autonomy capability are global in nature and commercially available
  - Actively maintaining global awareness of emerging advances in technical capability and field application must be understood and taken into account
- Direct the Military Service Chiefs to:
  - Equip Military Service opposition forces in training exercises with autonomous systems and counter-autonomy capabilities
  - Use lessons learned to support CONOPs, doctrine, and training

[32] A useful model for this is the Low Observable and Counter Low Observable (LO/CLO) Executive Committee.

4 Strengthening Operational Pull for Autonomy

Commercial interest is exploding for autonomy, from widespread commercial development of UA for civilian applications, to self-driving and driver-assist applications for automotive applications, to dynamic spectrum management for cell phones, to IBM's Watson technology providing decision support to human operators in a wide range of big data applications.

Autonomy delivers operational value across a diverse array of vital DoD missions. The study established a categorization for the ways that autonomy can benefit DoD missions:

- Required decision speed – More autonomy is valuable when decisions must be made quickly (*e.g.*, cyber operations and missile defense).
- Heterogeneity and volume of data – More autonomy is valuable with high volume data and variety of data types (*e.g.*, imagery; intelligence data analysis; intelligence, surveillance, reconnaissance (ISR) data integration).
- Quality of data links – More autonomy is valuable when communication is intermittent (*e.g.*, times of contested communications, unmanned undersea operations).
- Complexity of action – More autonomy is valuable when activity is multimodal (*e.g.*, an air operations center, multi-mission operations).
- Danger of mission – More autonomy can reduce the number of warfighters in harm's way (*e.g.*, in contested operations; chemical, biological, radiological, or nuclear attack cleanup).
- Persistence and endurance – More autonomy can increase mission duration (*e.g.*, enabling unmanned vehicles, persistent surveillance).

As has been the case in a number of other technologies, most notably information technology and the Internet, where the DoD was at one point the driving force behind technology development, much of the leading research in autonomy is happening outside the Department and, in some cases, outside the United States. A key objective of this study was to identify opportunities for DoD to more rapidly exploit ongoing advances. By selecting several demonstrations of autonomous systems with near-term benefits, the study hopes to both demonstrate the operational value of autonomy, **while simultaneously strengthening the enterprise business practices recommended in this report that will make or break the transition of new technologies from the lab to the battlefield**.

Because the DoD mission is so broad, it was beyond the scope of this study to conduct an exhaustive review of, and search for, all of the beneficial roles for autonomy. Rather, the study chose to select representative system and mission applications to illustrate the potential value of autonomy. The study investigated four areas in depth–protection, battlespace awareness, force application, and logistics. These are joint capability areas that could immediately adopt existing autonomous technologies.

Within these capability areas, the study evaluated potential experiments against the relative benefits of autonomy. Descriptions of ten representative demonstration projects are interspersed in the following sections to show where focused prototyping and experimentation can validate advanced CONOPs and demonstrate the benefits associated with more aggressively adopting autonomy technology. In Figure 11, these 10 projects are shown plotted against the six benefit areas listed above. These specific efforts have the potential to either present significant challenges to an adversary—by costing more to counter than it costs the U.S. to deploy—or to negate an adversary's transformative capability.

Each of the following 10 projects could be started immediately and is predicted to yield wide-ranging impacts. While these specific demonstrations are strongly recommended, the list is not intended to be exhaustive. Other demonstrations could deliver the same—or additional—benefits.

Autonomy for battlespace awareness

The evolving national security landscape places an increasing premium on the Department's ability to develop and sustain situational awareness. Across the globe, threats are increasingly diffuse and growing, decision cycles—especially for cyber-threats—are dramatically shortened, and resource constraints will continue to limit the ability to cover every scenario with equal vigor.

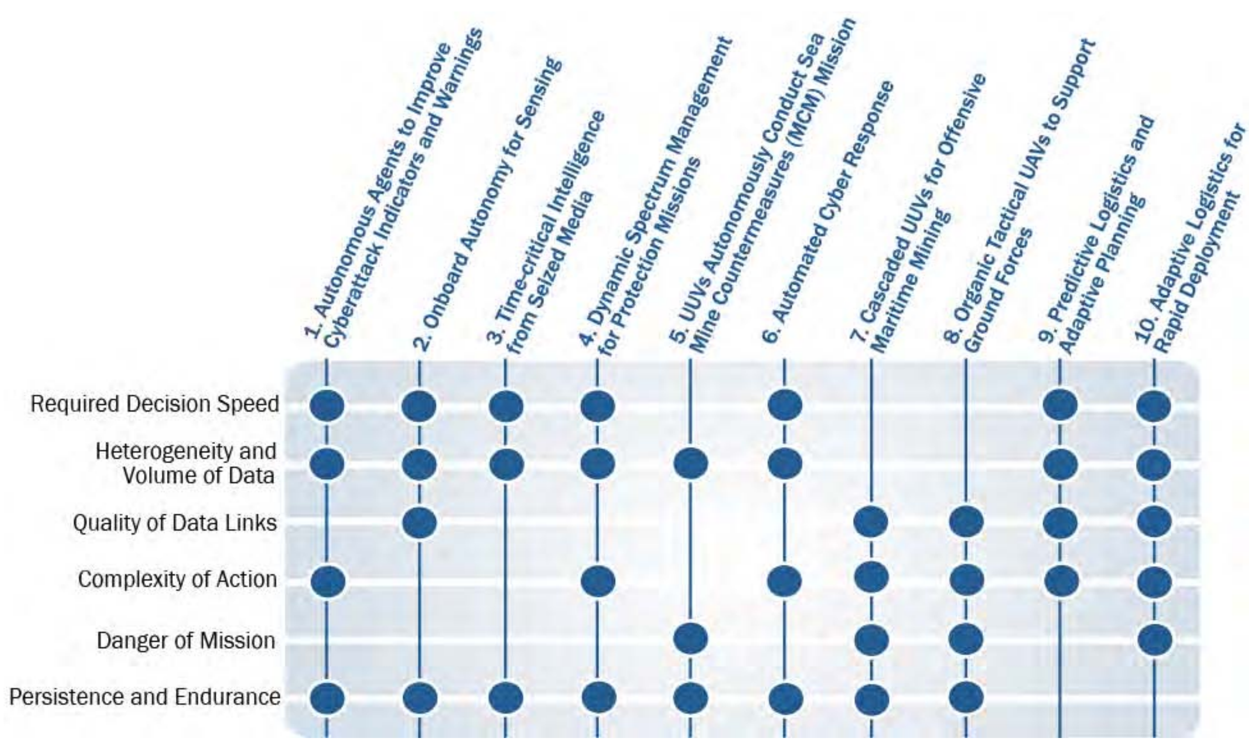


Figure 11 The study evaluated many candidate projects and selected those that encompassed the range of benefits of autonomy.

Providing battlespace awareness to the warfighter in potentially far-flung and congested battlefields is an increasingly complex problem, with a solution that is increasingly multifaceted. It will require better sensors and better organization, as well as more robust communications capabilities and data security. It will perhaps require a new tool: that of the capability for increased autonomy. Autonomy has the potential to enhance the ability to do more, at speed, at the sensor platform, which would then facilitate rapid re-direction of sensors; reduced transmission of data; enhanced sensor functionality; and continuing mission relevance in the absence of ground communications or a communications denied environment.

Autonomy can allow the commander to understand the real-time battlespace using data and techniques in ways that humans cannot by addressing volume, complexity, speed, and continuity. It can also assist in the merging of open source data with classified data sources.

Autonomous tools can help commanders integrate sparse, disparate, and unformatted data sources. Autonomy allows the human to detect and resolve inconsistencies deriving from any source by making information harder to hide and harder to spoof. At the same time, a typical adversary will be motivated to modify situational awareness capability to their advantage. Autonomy will enable intelligent sensing by helping sensors utilize their own data to collect better data through reduction of uncertainty, energy use, and communication bandwidth, as well as to address dynamic sensor configuration to support mission-relevant collection that is tied to commander's critical information requirements.

Battlespace awareness has typically been associated with the sensing part of the sense-think-decide-act process, but the rapid advance of technology may begin to blur these distinctions as battlespace awareness platforms must necessarily become more intelligent and more mobile. Over centuries of warfare, technology has inexorably expanded to permeate the various steps of sensing-thinking and deciding-acting, with emphasis first on the actuation of systems in battle. With the rise of electronics, sensing has extended far beyond the capability of human operators. Today, computers and networks are increasingly integrated with sensors, creating complex systems of systems of battlespace awareness capabilities.

Evolving support for conflict

Over the past decade, a generation of new sensors and decision support tools has been added to the Department's arsenal to address the growing complexity, ambiguity, and velocity of conflict. The ISR and related counter-IED systems employed in Iraq and Afghanistan, such as Constant Hawk, Gorgon Stare, and Argus provide wide area persistent surveillance. Similarly, a new class of sensors operates in cyber-space to enable the Department to assimilate threat data from forward deployed intelligence systems. These feed boundary defense devices that detect and counter cyber-threats at scope and scale in real- to near-real-time.

While these advances are impressive, these systems continue to require a vast logistics pipeline and a significant footprint of human talent to collate, cross-walk, and groom the sensor data in order to feed the downstream systems that generate responsive courses of action. Current systems continue to strain to fully leverage the data volume, data velocity, data variety, and data veracity, which continue to outstrip human capacity to process and understand. The ability to extend the autonomy of these tools and techniques for the DoD could have a significant impact when applied to battlespace awareness in physical space as well as cyber-space.

The study concluded that appropriate employment of autonomy across the DoD military enterprise will yield a wide range of improvements in performance. Many commercial companies have developed, either for their own use or for sale, products that provide situational awareness and that incorporate various levels of autonomy. The Department is capitalizing on these where its operations are similar to those of commercial enterprises, but additional opportunities exist.

Project #1: Autonomous agents to improve cyber-attack indicators and warnings

Cyber-threat actors still enjoy a significant advantage over defenders in their ability to mount and sustain attacks using the natural camouflage that derives from the fact that any one defender can only observe a small component of the overall stream of adversary actions. This is especially acute as defenders attempt to detect threat streams that cross multiple networks, jurisdictions, and areas of responsibility.

The DoD very often cannot share information in a timely way because doing so would compromise classified sources and methods. At the same time, DoD frequently fails to capture insights available in open sources. While there is broad recognition of the imperative to share information across sectors and organizations to "connect the dots" and more easily reveal these threat actors, a particularly difficult challenge remains in the marriage of classified and unclassified data sources to feed a common operational picture. Much threat relevant data is held by the private sector, while the U.S. government generates and holds unique and valuable information in the form of insights gained from classified sources on both threat actors and activities.

Recent successes with cutting-edge tools for data analytics, such as conditional random fields for scene classification, and the ability of data analytics to synthesize and derive near real-time insight from large and complex volumes of data, offers the opportunity to employ these same techniques to the extraction of insights, tips, and queues from disparate data across classified and unclassified sources.

DoD alone enjoys permissive access to classified sources while at the same time enjoying access to increasingly robust open source data obtained through its own experience and authorities, or through collaborative or customer-provider relationships with the private sector. A demonstration is proposed to show the usefulness of this approach without compromising classified material.

An autonomous agent could examine open source data in multiple formats based on a cue from a classified product. Autonomy will allow the agent to aggregate information from multiple sources while obfuscating the search. The result will be sharable information that will not point back to classified sources and methods.

A successful program would add value to existing data from all sources. Most importantly, this process could enable cooperation among groups with different classification accesses.

Recommendation 12.

NSA, in partnership with DARPA and the Intelligence Advanced Research and Projects Agency (IARPA), should fully develop the means to tip and cue the Defense Information Systems Agency (DISA) and the defense industrial base to defend the DoD information infrastructure, extending to U.S. government and private sector support as appropriate. The study recommends an allocation of $50 million over three years to pursue an aggressive goal to develop a working prototype comprised of the following:

- Take lessons learned from nascent efforts underway since 2007 and consider the threat-paced time urgency for fully mature defensive systems[33]
- Leverage commercial and intelligence community tools and datasets to develop a continuous assessment of network conditions, threat vectors, and anomalous behaviors, and a rapidly configurable toolkit to provide fine-grained intelligence in support of threat identification, attribution, and tipping of operational cyber-defense forces
- Extract and integrate information from multiple and dynamic sources, and obfuscate search using dummy queries and meta-queries
- Rapidly sanitize information for sharing and dissemination to supported customers, the private sector, and allies

Project #2: Onboard autonomy for sensing

Counter-terrorism, time-critical targeting, and urban operations, among other missions, require wider field-of-view sensing with higher resolution and frame rates. A new class of such sensors is reaching the battlefield, including Constant Hawk, Gorgon Stare, and Argus. As users gain access to high-definition full-motion video, they are becoming dissatisfied with low-resolution images. However, comprehensive transmission of complete high-resolution collection, even in uncontested environments, is not feasible on the foreseeable communications infrastructure. Communication networks are already overburdened and are a key vulnerability in contested environments.

Sensor technology for ISR is rapidly expanding in terms of both resolution and coverage area. The pace of this growth has greatly outpaced our ability to communicate raw sensor data back to ground stations for processing and analysis, and even is outpacing processor capability growth as widely characterized by Moore's Law (Figure 12). This is in part due to enhancements in focal plane array technology.

Operationally, video analysts still manually review ISR video where the relevant information content to data ratio is quite low. Data fusion and analysis software is maturing. Dynamic time

[33] Such a system is Tutelage, which tips and queues SIGINT and collateral information on cyber-threats to sensors protecting the Defense Industrial base and DoD-managed networks.

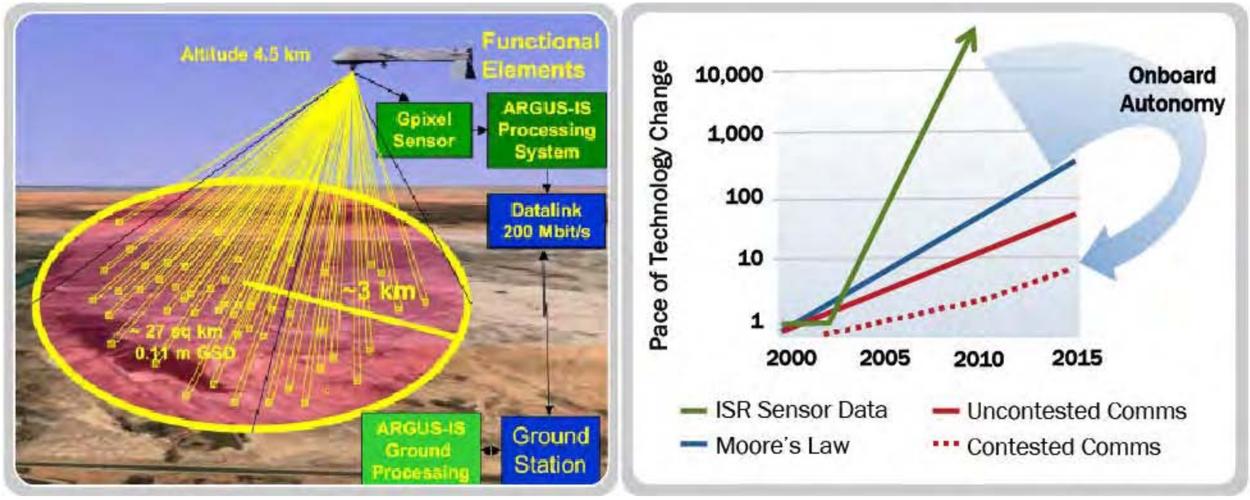


Figure 12 Elements of the ARGUS-IS Wide Area sensor are shown (left), along with the pace of technology change in sensor capabilities that can enable onboard autonomy (right).

critical warfighting capability has been developed by the Air Force and reached technology readiness level (TRL) 7 in 2009, with a focus on ground-based multi-source fusion for time critical targets. This capability was able to demonstrate greater than 90 percent success in automatically detecting moving humans and vehicles.[34] The Air Force Research Laboratory (AFRL) has an ongoing program called Planning & Direction, Collection, Processing & Exploitation, Analysis & Production, and Dissemination Experimentation (PCPAD-X) for ground-based fusion as well. DARPA is finishing the Insight program with a similar focus on ground-based multi-source ISR sensor fusion.

Some unattended ground sensors and undersea systems already use onboard, autonomous sensor processing. Technology advances in high-throughput, embedded processing, and machine learning offer the promise of onboard processing of high-resolution, multi-source airborne ISR sensor data. Autonomous sensor processing and high-level information generation would greatly reduce the required communications bandwidth and reduce the burden on human analysts. It also provides higher quality, improved persistence, better resilience, better tasking, and higher reliability. Perhaps the greatest operational benefit is agility: a single platform can adjust collection in real time based on observation, and many such platforms can coordinate to achieve better theater coverage. This would dramatically improve targeting information.

The application of machine learning enables much of this autonomous behavior. While the data needed for real-time analysis and cueing may be greatly reduced in this scenario, it is important to capture and retain the datasets to calibrate and retrain autonomous systems. Autonomous sensor data screening and fusion software will have increasing levels of complexity and will require different models for learning. They can be categorized as follows:

[34] A. J. Newman and G.E. Mitzel, *Upstream Data Fusion: History, Technical Overview, and Applications to Critical Challenges* [DARPA MTO Industry Day, 2015].

- Level 1–Object Refinement: A consolidated estimate of the observed and observable objects (*e.g.*, vehicles, facilities, persons) in the battlespace, and their kinematic state, representing the combined information from all sensors and information sources
- Level 2–Situation Refinement: An estimate of the militarily relevant entities (*e.g.*, units, groups, events) in the battlespace, their status, and relationships among the entities and between entities and observable objects
- Level 3–Threat Refinement: An estimate of the threat posed by entities in the estimated situation, including intent relevant to blue force plans, projected range of actions, and potential impact on those plans

Recommendation 13.

DARPA, working with AFRL and the 711th Human Performance Wing, should initiate a new program to adapt existing ISR data screening and fusion tools, such as the Air Force's Dynamic Time Critical Warfighting Capability (DTCWC) or PCPAD-X, or DARPA's Insight for autonomous, real-time use. The estimated cost for this effort is $80 million over three years. Some suggested implementation steps include:

- Integrate software into an embedded processing payload on a BizJet-class UA platform to autonomously prioritize and process extracted sensor data and transmit mission-relevant information over communication channel capacities that would be available in a contested military environment
- Demonstrate a likelihood greater than 90 percent of including target information in the real-time communications stream for the autonomous screening and processing algorithms as needed to achieve operational relevance
- Participate for demonstration purposes in a Red Flag exercise of time-critical detection and tracking of surface-to-air missiles in a denied environment

Project #3: Time-critical intelligence from seized media

Special operations forces and Military Service tactical document and media exploitation (TACDOMEX) teams routinely seize massive quantities (terabytes) of diverse data types on digital media (*e.g.*, computers, tablets, smart phones) from adversaries. For time-critical, counter-terrorism operations, these media can provide valuable intelligence on people, places, and organizational structures— if exploited on operationally relevant timelines, meaning hours or days rather than weeks or months. Current document and media exploitation (DOMEX) operations focus on extracting information only from human-searchable files such as text and metadata. However, images, video, and audio can provide additional valuable information that can automatically be extracted by commercial tools like image analysis, translation, summarization systems, email network analysis, scanning and word recognition, and speech analysis. This in turn can be used as input to a new tool that constructs, for example, a social network graph and with node annotations, *e.g.*,

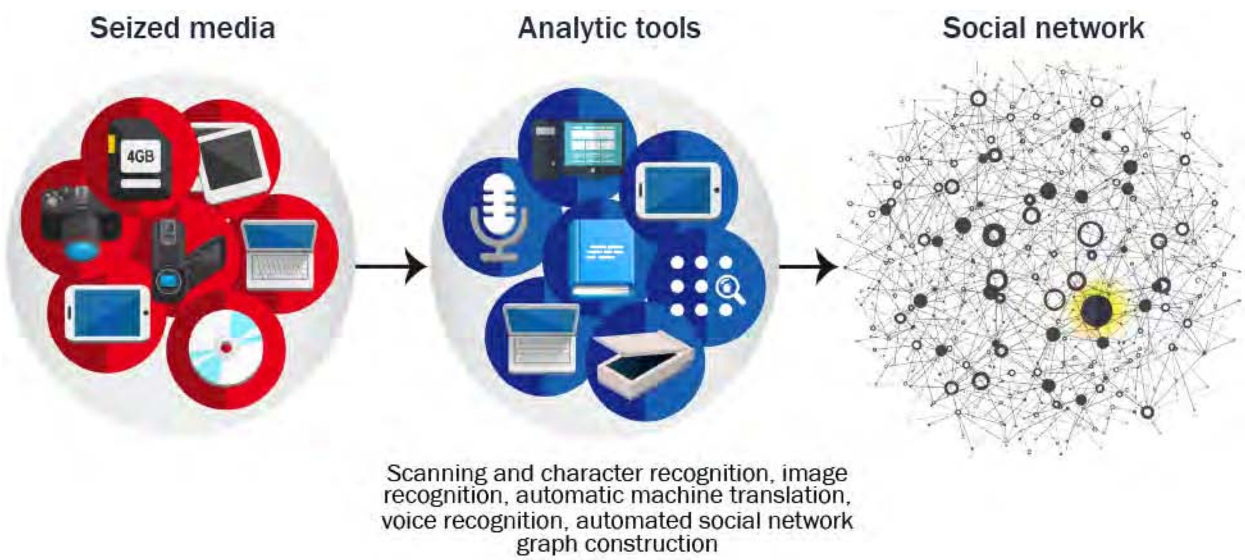


Figure 13 Examples of seized media are shown (left), along with tools that can make sense of the stored information in real time (center). The resulting social network can reveal a real-time threat (right).

approximating a human terrain map that unifies disparate data sets to reveal adversary actions, aspirations, capabilities and future plans (Figure 13).

High fidelity, real-time, situational awareness derived from this "digested information" can drive ongoing operations intended to optimize blue force defenses and sustain tactical, offensive advantage over adversaries already disrupted by the initial phase of an operation. When appropriate, significant leverage will be derived from the real-time marriage of these time-sensitive assessments of captured information, and the on-going restrike of adversary capabilities and initiatives.

Commercially available computing hardware and open source deep learning software will be an important foundation of this work, assisting in the production of finished intelligence in real time, revealing adversary networks, communications, plans, operational methods, and insight. Outputs from these tools can be opportunistically communicated to human analysts for further insight so that benefits accrue to both ongoing operations and the intelligence operations intended to inform them and future plans. The result will be more timely and actionable information in the field, significant improvements in ongoing operational leverage, faster and higher fidelity cues to human analysis, and deeper analysis derived from holistic assessments beyond the scope of current tools.

Recommendation 14.

The Defense Intelligence Agency (DIA) and the Special Operations Command (USSOCOM) should integrate commercial components and build a new machine-learning analysis tool, and prototype the resulting system using existing historical data, seized media, and commercial (collateral) sources. The cost is estimated at $20 to 30 million over 2 to 3 years:

- USSOCOM should develop the capability to digitize any non-digital information and transmit this and all collected digital material in real time to analysis cells

- Rapidly sanitize information for sharing and dissemination to supported customers, the private sector, and allies

Autonomy for protection

The ability to prevent or mitigate the adverse effects of attacks on personnel, both combatants and non-combatants, and the physical assets of the United States, allies, and friends is an important DoD mission.[35] It includes both kinetic and non-kinetic attacks and other adversary offensive capabilities from all domains—surface, air, undersea, space, cyber and electronic warfare. It also addresses fixed facilities and locations, such as bases, borders, and air and missile defense sites, as well as mobile assets, including aircraft, ships, satellites, and personnel.

As adversaries make use of more sophisticated technology and weaponry, maintaining vigilance against potential attacks and responding rapidly to threats will require increasing use of autonomy-enabled capabilities. Because missions are largely defensive, fewer policy issues arise in comparison to those emerging around the offensive use of lethal autonomous systems.

Current and potential uses of autonomy in protection

Two of the six major drivers to stimulate the adoption of autonomy technology are behind many of the current uses in the protection missions. The two drivers are the required decision speed and the need for persistence and endurance. When both drivers are in play, there may be no alternative to autonomy for missions such as missile, space, and cyber-defense.

For these applications, collaboration with the human operator is often limited to system initiation and oversight. For example, in many air and missile defense systems there are two modes– manual and automatic. In the manual mode, the operator collaborates with the system to identify and validate adversary targets, and then launches an autonomous interceptor. The system then switches to automatic mode, where the operator monitors the system as it detects and engages targets autonomously unless the operator chooses to call off the engagement.

Protection missions, such as defending against an incoming missile salvo or cyber-attack, are also driven by both the need for persistence and rapid decisions. They are enabled by extensive signal processing of large volumes of sensor data coupled with the use of autonomous agents for rapid decision-making and actuation of control systems. The cyber-defense mission is almost completely analogous to the protection of civil and commercial information technology systems, and, consequently, DoD can benefit greatly from collaboration with the commercial sector to benefit from developments in that market.

When persistence is the key driver for autonomy, the autonomous system of choice has often been a UA that can protect large geographic areas over long times. These UA are generally remotely piloted today, with only a few functions delegated to autonomy, such as station-keeping or sensor

[35] Joint Capabilities Assessment, 2010 [Refinement approved April 8, 2011].

management. Systems may also have automatic target recognition, either alone or as a cueing aid for a human operator.

To protect limited geographic areas with reasonably predictable terrain features, unmanned ground vehicles (UGVs) have often been selected. The Mobile Detection Assessment and Response System (MDARS) is such an application that has numerous commercial analogs. Unmanned ground vehicles have also been deployed to minimize human exposure to hazards in the IED land mine clearance mission. The QnetiQ Talon (Mobile Tactical Robotic System, MTRS Mk1) and iRobot Packbot (MTRS Mk2) systems were deployed extensively in this role during the recent wars in Iraq and Afghanistan. The next generation of this capability is under development in the Advanced EOD Robotics System (AEODRS) program. As is true for UA, most UGVs are remotely operated and use autonomy technology almost exclusively for navigation and obstacle avoidance.

The study concluded more opportunities will emerge to delegate cognitive functions to an intelligent system in such areas as vehicle health monitoring or situational awareness. Such applications would exploit mature capabilities already in commercial use.

Commercial technologies are also outpacing military technology in autonomy in unmanned undersea applications. Developing and employing autonomous undersea systems has long been the purview of the U.S. Navy, but in recent years the commercial undersea survey and oil exploration industry and the scientific oceanography sector have taken the lead in deploying autonomous, often low-cost, platforms. While the Navy has kept pace in conducting foundational research and developing prototype systems in this area, there is significant value yet to be realized in operationalizing military systems. Currently deployed counter-mine applications use UUVs for persistence and protecting humans from danger, but rely on human operators at a command center to process data for target classification. This is followed by a separate mission to neutralize any mines detected. Autonomy can reduce both the time to neutralize the threat and the danger to the personnel assigned to the task.

Finally, the use of autonomy in the mission to counter chemical, biological, radiological, and nuclear threats is motivated by both the need for persistence and endurance and to protect humans from danger. Several current and potential programs aimed at using autonomy to reduce risk and improve protection of U.S. assets were identified.

Protection is an area where the benefits of autonomy have been well demonstrated, because it requires persistence and endurance that are often limited by human factors. Further, in many situations, protection requires speed of response or exposure to hazardous environments that may be better addressed by an autonomous system with an appropriate level of supervision by human operators.

The study reviewed the impact of a broad sampling of current and developmental uses of autonomy to improve protection capabilities. Many opportunities were identified to exploit the benefits of autonomy technology. Demonstration and early successes with deployed systems could accelerate adoption across the protection capability area.

Project #4: Dynamic spectrum management for protection missions

Today, the military use of the radio frequency (RF) spectrum is manual and largely pre-planned. Because of the complexity and dynamic nature of the environment, this approach can neither maximize use by U.S. forces nor deny adversary use. If not addressed, this situation will only worsen.

The opportunity presented by automating sensor, communications, and jamming coordination within the environment is to protect the ability to achieve information dominance while imposing high "cost" and disruption on adversaries.

Recommendation 15.

The study recommends two simultaneous and complementary programs that inform each other to achieve dynamic spectrum management:

- The U.S. Army Communications-Electronics Research, Development and Engineering Center (CERDEC), AFRL (Rome, New York), and the Space and Naval Warfare Systems Command (SPAWAR) should develop Military Service prototypes for local, agile spectrum deconfliction and control among a few systems. One demonstration per Military Service should be coordinated through semi-annual collaboration conferences. The estimated cost for this program is $400 million over 5 years.
  - Each prototype demonstration should involve at least two non-collocated systems carrying out different but contemporaneous missions. A first demonstration is suggested for a locally shared electromagnetic spectrum common operating picture to demonstrate negotiation techniques that include agile adaptation to the electromagnetic environment. It should demonstrate the first implementation of centralized rules and policies, rather than centralized spectrum assignments and should be carried out in coordination with the single, joint program on spectrum management. The results should inform the evolution of that program.
- DARPA should develop an architectural framework and algorithms for near-real time, theater-level spectrum deconfliction and control for a full ensemble of joint, coalition systems. The estimated cost for this program is $180 million over 3 years.
  - The recommended program would identify and develop the enabling framework and technologies for dynamic spectrum management over a large area involving thousands of systems, both friendly and hostile. The program would identify the protocols and algorithms required for distributed negotiation for mitigating inference among blue forces and maximizing electronic attack on red forces.

Project #5: UUVs autonomously conduct sea mine countermeasures mission

Sea mines have been laid during various conflicts since the Civil War and threaten both military and civilian maritime operations. The sea mine countermeasures (MCM) mission, localizing

Figure 14 Current Mine countermeasure capabilities utilize two separate vehicles – an autonomous UUV for search and detection (left) and a vehicle remotely operated by a manned ship in the mine field for disposal (right).

and neutralizing mines, is critical to ensure the safety of waterways for civilian traffic and to ensure military access to areas of importance.

Current manned and unmanned MCM platforms all require personnel in the minefield. MCM-1 class ships can detect, classify, and neutralize all known types of mines, but are manned by a crew of over 80 individuals. Increased utilization of autonomy-enabled UUVs can significantly reduce personnel risk during MCM operations by allowing uniformed personnel to supervise MCM operations remotely rather than entering the minefield. The MK-18 Mod 2, shown in Figure 14, program has demonstrated significant progress in utilizing Remote Environmental Monitoring UnitS (REMUS) UUVs for MCM. Further gains are possible. Current MK-18 operations, for example, continue to require long tactical timelines with intensive operator involvement, including a manned platform entering the minefield during the neutralization stage. Increased autonomy could reduce the demand for manning and personnel risk, and decrease the tactical timeline.

The study acknowledges that some of these recommended actions are already being explored and stresses the importance of continued progress in this area to further reduce personnel risk and decrease the time needed to conduct mission-critical MCM operations. For example, in the Single Sortie Detect to Engage program, ONR is developing enabling technologies to support integrating these two elements of the mine countermeasure mission into a single activity.

Additionally, while significant progress has been made in the basic research domain, there is a need for further research and development to harden and make robust the aforementioned capabilities. Continued focus in parallel R&D, particularly in improved automatic target recognition (ATR) capability and autonomous launch and recovery of UUVs, will ultimately enable delivery of an MCM mission package by larger unmanned platforms.

Recommendation 16.

The Navy Program Executive Office Littoral Combat Ships (PEO-LCS) should conduct a user operational evaluation system (UOES) program run by PEO-LCS in partnership with ONR. The estimated cost is $60 million per year for three years. Suggested implementation actions for this demonstration project are the following:

- Equip an existing UUV platform, such as the MK 18 Mod 2, with embedded ATR capability to enable autonomous detection, classification, localization, and identification. Embedded ATR algorithms will utilize sonar and optical sensors to locate and identify mine-like contacts utilizing viable communication channels to query a remote operator to confirm identification. This will reduce the time for mine detection, classification, and identification, currently conducted over two sorties with intensive operator interaction, to a single-sortie task with streamlined operator interaction.[36]
- Update an existing mine disposal platform, such as Seafox, with contact reacquisition and neutralization capability. Seafox is a wire-guided mine neutralizer that uses a shaped kinetic charge. Providing communications from the mine disposal platform to operators, along with higher-level autonomous control functions, will retain operator control of neutralization and will remove the need for personnel to enter the minefield to execute fly-by-wire operations.
- Expand delivery of the MCM sensing and neutralization nodes to an unmanned surface vehicle (USV) or large UUV to enable the detection, identification, and neutralization to be accomplished in a single sortie. As a stretch goal, delivery of this mission package by a large UUV could enhance mission covertness when required, for example, for MCM in support of an amphibious assault operation. The delivery platform would facilitate communications and command and control functionality as required.

Utilizing this acquisition model, fleet operators would work with developers during the course of the program to experiment with the system to rapidly evolve CONOPs, and design and characterize system strengths and limitations. After four years, the program would transition the enhanced MCM package to the Navy's 5th Fleet.

Project #6: Automated cyber-response

Despite increased awareness of cyber-threats created by the daily revelation of yet another audacious hack into systems containing sensitive and personally identifiable information, threat actors still seem to be winning in the daily contest with defenders across both the public and private sector. DoD classified and unclassified networks, the financial sector, industrial control systems, key elements of the DoD industrial base, elements of the power grid, cryptography firms, commercial companies, gaming industry, and even the entertainment industry have all been affected. The unfortunate truth is that even a greater number of incidents are not made public, and in a still greater

[36] D. Scheidt and G. Pollitt, *Hybrid Control Algorithms for Cooperating Vehicles Final Report, Appendix B: Benefits of Autonomous Operations (AO) Beyond Undersea Cooperative Cueing and Intervention* [September 2009].

number of incidents, the victim is unaware of the compromise.[37] In the known incidents, the security posture of the target was clearly not up to the challenge.

In many cases, the target failed to apply patches needed to address published vulnerabilities or inadequately trained staff made elementary mistakes. The aftermath of these incidents is unpleasant, embarrassing, costly to fix, and damaging to reputation and attendant investor and customer confidence. Worse still, the *modus operandi* for defending these systems is still largely focused on solutions that address adversary attacks based on previously observed behavior and subsequent mitigation based on detecting a recurrence. The systems employed to address individual adversary tactics are often not networked to achieve a comprehensive, let alone real-time, sense of adversary behaviors that cut across the otherwise stove-piped and unconnected defensive sensors. The present practice of bringing in security experts to remove the malicious software/hardware, restore the system, and determine attack attribution cedes initiative and advantage to adversaries who pick the time and place of their attacks and overwhelm defensive tactics with increasingly sophisticated campaigns.

Even after addressing the obvious shortfalls of not patching and inadequate training, systems will remain vulnerable to more sophisticated attacks that take place much faster than human decision cycles can address. The laudatory goal of perfectly secure systems is an impossible one, a reality deriving from their complexity, the constant changes they undergo as a result of system and infrastructure upgrades, and the reality of unpredictable human behavior on the part of both authorized and adversarial humans. As a result, security doctrine across all sectors is moving from one of creating and maintaining secure systems (impervious to attacks) to one of creating defensible systems that are well defended and supported by a diverse array of tools, authorities, and intelligence.

The emerging strategy represents a fundamental shift from a focus on inherent properties of systems, which remain important, to a dynamic understanding of the behavior of systems and actors and the active management or interdiction of them over time. Given the speed at which these changes take place and the complex nature of the systems themselves, we need greater levels of automation and autonomy to increase our ability to protect these systems within the timeframe of the attack. The foundation—Tutelage—for such a system already exists within the intelligence community. Today, this system provides real-time protection of the Non-classified Internet Protocol (IP) Router Network (NIPRNet), inspecting and analyzing more than 3 million packets per second for threats. Over the past five years Tutelage has prevented hundreds of millions of attacks. Additionally, this system is designed to provide actionable intelligence to DoD partners to help protect the nation. The proposed investments will significantly build upon these current capabilities.

A comprehensive network of sensors needs to be designed and deployed within blue space (systems we own), grey space (systems being used by the attacker that are owned by an unwitting third party), and even deeply buried within red space (systems being used to support the attack and owned or residing within the domain of the attacker). Clearly, there are technological, operational, and policy challenges with such an architecture.

---

[37] Verizon, *2015 Data Breach Investigation Report*. Available at www.verizonenterprise.com/DBIR/2015/ (Accessed January 2016.)

This sensor network that is fed into advanced autonomous systems will in real-time develop options to thwart the attack in the timeframe required to protect the target. Because many of these options could impact U.S. law, authorities, and policy, an option development engine must ingest these rule boundaries and then be able to rack and stack options that stop the attack while operating consistent with agreed-on rules of engagement. In the event that all options infringe upon at least one of the rules, the option and rule infringement should be highlighted, and a workaround identified or a waiver requested. While search results may not be able to stop an imminent attack, they could make systems ready for the next one.

In cases where the execution of the option is within the legal, policy, and authority bounds, additional infrastructure must be developed or existing infrastructure augmented to autonomously engage the cyber-weapon and block or stop the weapon's effect on the target. In different cases, the response option may target the weapon, the weapon infrastructure, the weapon command and control, or the effectiveness of the operator. Some examples of industry efforts for autonomous detection and mitigation include:

- Better analysis and detection can be facilitated by safe browsers with access to endpoint data. This has been used to analyze the Great Cannon attack on Github.[38]
- Google Project Shield is an initiative to explore new ways of using Google's attack mitigation technology to offer news sites free protection from distributed denial of service attacks.[39]
- Malware sharing forums, Internet protocol (IP) attack compilation statistics, industry cyber-attack analysis (IBM, Symantec, McAfee, Intel, FireEye, Verizon). Most sharing is done pairwise based on trust.
- The Structured Threat Information Expression (STIX) is a collaborative community-driven effort to define and develop a standardized language to represent structured cyber-threat information. The STIX language intends to convey the full range of potential cyber-threat information and strives to be fully expressive, flexible, extensible, automatable, and as human-readable as possible. All interested parties are welcome to participate in evolving STIX as part of its open, collaborative community. Trusted Automated eXchange of Indicator Information (TAXII) is the main transport mechanism for cyber-threat information represented as STIX. Through the use of TAXII services, organizations can share cyber-threat information in a secure and automated manner.[40]

---

38 Ars Technica, *Meet "Great Cannon," the man-in-the-middle weapon China used on GitHub* [April 10, 2015]. Available at arstechnica.com/security/2015/04/meet-great-cannon-the-man-in-the-middle-weapon-china-used-on-github (Accessed June 2016.)

39 Google Project Shield. *Protecting free expression from DDoS*. Available at projectshield.withgoogle.com/public/about (Accessed June 2016.)

40 STIX/TAXII Standards Transition – Frequently Asked Questions. Available at stixproject.github.io/oasis-faq.pdf (Accessed June 2016.)

Recommendation 17.

The U.S. Cyber Command should take the lead to develop an automated cyber-response, in partnership with the Central Intelligence Agency (CIA), the Federal Bureau of Investigation (FBI), NSA, DARPA (Plan X), key cyber-security industry players, and DISA. The team should implement a limited demonstration within one year and full operational capability within three years. The estimated cost is $50 million per year for two to three years. Some suggested implementation actions include:

- Build upon existing success of the DoD's current active defense system (Tutelage), leveraging private sector innovation and attendant rapid advances being achieved there:
  - Clarify authority and accountability within the Department for the overall design and implementation of the system that will connect and leverage sensors and analytics across DISA, the services, USCYBERCOM, and the combatant commands.
  - Design a family of sensors that can be introduced unilaterally and in partnership in blue, grey, and red domains.
  - Compile the legal authority and policy constraints in a form that can be ingested into the option development engine. Engage legal and policy experts early to identify obstacles.
  - Develop a global clandestine infrastructure that will enable the deployment of the defensive option to thwart an attack.
  - Ensure that the sensors, tools, options, and infrastructure used to support this enhanced defensive mission architecture do not compromise capabilities that support our various other missions that must remain much less visible.
  - Once designed, benchmark the effectiveness of the system against the plethora of historical attacks on U.S. targets.
  - Crawl, walk, and run in lock step with legal, policy, and partners to demonstrate system effectiveness with acceptable levels of unintended consequences—and in doing so, develop confidence and trust.

Autonomy for force application

Force application is the ability to integrate the use of maneuver and engagement in all environments to create the effects necessary to achieve mission objectives. Plans may include maneuver to insert, to influence, or to secure a location. Engagement may be through kinetic or non-kinetic means, using both lethal and non-lethal weapons. Force application planning assesses the most appropriate capability to achieve the objective.

Autonomy can improve the speed and accuracy, and by extension, the effectiveness of all aspects of force application. Anti-access and area denial (A2/AD) is a primary example of a mission that could be enhanced by autonomous systems. Autonomously operating UA could assume several functions now performed by manned aircraft in areas that are difficult to access (*e.g.*, aerial refueling, airborne early warning, intelligence, surveillance, reconnaissance, anti-ship warfare, and command). Additionally, large UA could be designed to dispense small UA that could operate autonomously to facilitate both offensive strike (via electronic warfare, communications jamming, or decoys), as well as defensive measures as decoys, sensors and emitters, target emulators, and so on—to confuse, deceive, and attrite adversary attacks. These small swarms could be deployed as perimeter and close-in defensive actions with payloads tailored to the situation.

In this study, an unmanned aircraft (UA) refers to a single asset. An unmanned aircraft system (UAS), however, refers to a system or systems of aircraft and payloads, command and control systems, communications architecture, ground stations, and CONOPs, which together comprise an entire capability that is greater than the sum of the individual parts. Because the individual UA are heterogeneous, the UAS provides an integrated capability far beyond that offered by a single UA.

These concepts could be readily applied to other missions. For undersea missions, acoustic and RF decoy payloads would likely be much smaller than sea mines, and thus could be more easily deployed in quantity from existing commercial UUVs. While today's electromagnetic maneuver warfare capabilities are limited, UUVs could provide a means to significantly extend capabilities and enable a covert option with a small observable footprint until electronic warfare (EW) operations are initiated. Typical communications systems could be emulated in the size, weight, and power available onboard 12-inch commercial vehicles.

Potential adversaries to the U.S. are creating systems (*e.g.*, very quiet submarines) and capabilities (*e.g.*, sophisticated sensors) that threaten U.S. forces as well as the undersea infrastructure. Moreover, with the current reliance on exquisite platforms, such as nuclear submarines, the U.S. runs the risk of being asymmetrically disrupted. To mitigate the risks, the U.S. must be more proactive and complement their submarine force with other capabilities, such as powerful new autonomous UUVs and sensor networks.

The Navy and DARPA have performed foundational work in many undersea areas, but there is no lack of additional possibilities to explore. Autonomous UUVs, in particular, hold great promise. Having been used by both DoD and the commercial sector (*e.g.*, in the oil industry), there are several UUV platforms that can provide a basis for rapid prototyping and experimentation.

Project #7: Cascaded UUVs for offensive maritime mining

One area, in particular, that could be leveraged more effectively is cascaded use of autonomous UUVs. With cascaded operations, extra-large autonomous UUVs (that may be close to 100 feet in length with large carrying capacities) would deploy smaller UUVs with targeted payloads, as illustrated in Figure 15. The extra-large UUV, acting as a submerged delivery vehicle, could be

Figure 15 The cascading unmanned undersea vehicle concept deploys smaller UUVs with targeted payloads.

launched from shores far from the area of operations or from surface ships, and then traverse autonomously to desired locations. Once close enough to the area of operations, the extra-large UUV would deploy UUVs with specialized payloads that could make their way to desired points for final action. An additional benefit of this model is that the extra-large UUVs would have the capacity to collect, process, shape, and then pass higher fidelity real-time contextual information to the smaller UUVs at the time of the smaller UUV's deployment.

Cascaded operations could enable many missions, including offensive mining, sea mine countermeasures, chokepoint monitoring and control, decoy delivery, and others in which it would be difficult to send surface platforms or submarines. While the concept is generalizable, there are many details to be worked and questions to be answered that would be influenced by the mission being executed, such as how critical recovery of significant deployed assets really is.

To illustrate the concept of cascaded UUV operations, consider offensive maritime mining. Today's offensive sea mining capabilities are limited, but UUVs could provide a means to significantly extend capabilities by increasing the influence range via mobility. Extra-large UUVs could be deployed from one or more shore sites or surface ships, and autonomously travel to an area of operations. Once they arrive, they could deploy a number of smaller UUVs or variants of modular torpedoes that have both automated target recognition capabilities and enough explosive material to disrupt or disable (or possibly even destroy) surface vessels. The UUV modular torpedoes would essentially serve as intelligent mines that are able to maneuver in an area and disrupt or disable adversary ships upon target verification. This would enable friendly forces to restrict adversarial freedom of movement and control access to key maritime areas, such as chokepoints in harbors. The UUVs also could be used to stop adversary ships from returning to their ports, thus precluding replenishment.

Recommendation 18.

U.S. Navy and DARPA should collaborate to conduct an experiment in which assets are deployed to create a minefield of autonomous lethal UUVs. The cost for this effort is estimated to be $60 million over three years:

- The funding would cover the cost of leasing an extra-large UUV, purchasing or leasing USVs, purchasing or leasing UUV modular torpedoes, adapting the UUV modular torpedoes to work together, modifying and developing needed software, performing trade studies, establishing range and range requirements, and conducting the experiment.
- The experiment would culminate with an extra-large UUV traveling to the area of operations and deploying UUV modular torpedoes. The large UUV could also serve as an intermediate communications point for limited minefield control. The UUV modular torpedoes would be equipped with ATR capabilities and would draw upon a database of acoustic and other sensor signatures, including RF intercepts, to monitor ships in the area, some of which would be targets. Ideally, the UUV modular torpedoes would autonomously heal holes in the minefield caused by loss of mines due to, for example, vessel destruction. It should be noted, however, that this self-healing capability would be difficult to achieve, especially if being covert was critical and communications and control were highly constrained.

The experiment could be conducted in phases to address key technology areas in an incremental manner. The first phase of the experiment would use USVs to demonstrate the ability to detect and home on surface combatants using RF emissions and acoustic signatures. It would also demonstrate the cooperative behaviors needed to initiate a kill and maintain control of a chokepoint.

A second phase, if warranted, would be similar to the first, but would be performed with a field of UUV modular torpedoes. A key requirement of the second phase would be to demonstrate the needed underwater command and control. Before this phase of the experiment, a trade study should be performed to assess existing commercial UUVs as well as the state of development of the modular torpedo. Typically, the terminal approach to intercept a ship is difficult and requires a speed advantage. Existing commercial UUVs generally do not have the propulsion capability needed for effective terminal approach and may also be limited in payload capacity for a warhead capable of a hard kill. As a point of comparison, the MK62 Quickstrike mine carries 200 pounds of high explosives and the MK65 carries 1350 pounds of high explosives. Thus, use of existing commercial UUVs may result in reduced influence range, effectively requiring a ship to pass over the UUV in a tripwire rather than allowing the UUV to extend the influence range by introducing mobility to intercept the ship. Payload limitations (*i.e.*, less than a few hundred pounds of high explosives) may also require consideration of mission kill rather than hard kill. Thus, the modular torpedo that is currently under development may ultimately be a better option for addressing this mission, but it may not be ready in time for this experiment.

A final phase would use an extra-large commercial UUV to autonomously deploy the field of UUV modular torpedoes. The extra-large UUV could be launched from a surface ship at sea or from a shore. It would autonomously navigate to an undersea test range area. Once in the range area, the extra-large UUV would deploy UUV modular torpedoes. These UUV modular torpedoes would either loiter in the area or execute a search pattern to seek adversary targets. At some point in the deployment a message would be sent to either the extra-large UUV before releasing the UUVs or modular torpedoes or directly to the UUV modular torpedoes to let them know it was time to go active.

For covert communications, messages could potentially be hidden in acoustic masks designed to appear like sounds in the local area. These masks could, for example, be sounds of local maritime traffic or aquatic animals. In a sense, this can be seen as the use of steganographic techniques in the acoustic domain. If the message only needs to go to the extra-large UUV, then the UUV modular torpedoes would be active as soon as released. If, on the other hand, the UUV modular torpedoes could receive messages, then they could stay passive while loitering or searching and go active when they received direction to do so. The final act would be the navigation of the UUV modular torpedo towards the adversarial target, once one was recognized, and the detonation of surrogate explosive when the UUV modular torpedo was in close-enough proximity.

The combined set of experiment phases would require commercial unmanned surface vehicles, an extra-large commercial UUV, and several commercial UUV modular torpedoes. The extra-large UUV would need autonomous navigation capabilities to get to a particular area of operations. It would also need to be modified to carry and deploy smaller UUV modular torpedoes. These UUV modular torpedoes need to be equipped with sensors and ATR software. In addition, they would have to be equipped with surrogate charges that could be virtually detonated when the UUV modular torpedo was in close-enough proximity to a target ship.

Project #8: Organic tactical unmanned aircraft to support ground forces

To achieve the U.S. defense strategy's mandate to project power and win decisively, U.S. ground forces must be able to enter foreign territory in the presence of armed opposition as well as an advanced A2/AD environment (*e.g.*, air and missile defense, jammed communications, and so on).

Currently, tactical ground units engaged in asymmetric and near-peer conflicts are under constant threat, operating in an environment that is complex, constantly changing, and unpredictable. The speed at which ground units discover, assess, and react to battle-space change is vital to tactical success. A unit's agility, or the ability to rapidly respond to unexpected change, is known to be an important characteristic of highly capable units.[41] Recognizing the relationship between decision-making speed and mission success, the U.S. Army promotes agility by instructing

41 D.S. Alberts and R.E. Hayes, *Power to the Edge* [CCRP Press, 2003]. Available at www.dodccrp.org/files/Alberts_Power.pdf (Accessed June 2016.)

soldiers to unilaterally take decisive action when necessary as long as that action falls within their commander's intent.

Unmanned aircraft system (UAS) support of an agile force requires that UAS capabilities be made available to platoon and squad-level units on an as needed basis. UASs carrying sensing, communications, jamming, and strike packages are capable of providing ISR, EW, overwatch, and tactical strike support that represent useful—potentially decisive—advantages for small tactical units. To be useful, UAS support must be timely. Since unit agility is often in response to unanticipated changes in conditions, the supporting UAS must be pre-positioned in advance of specific unit request. Because an operating theater may contain large numbers of small units operating over a large area, and the exact nature, timing, and location of unit need is unpredictable, a UAS may be required to provide simultaneous cover over a large area. In addition, the UAS manning requirement must be low so as to avoid placing an undue burden on front-line or supporting units.

Currently fielded UASs rarely factor into small unit ground combat because the systems and operating procedures used in their deployment cannot support the rapidly evolving needs of an agile force. Small organic UAS, such as the RQ-11 Raven, are difficult to use in unexpected engagements because the engaged unit is required to dedicate personnel to prepare and launch the Raven, which takes two soldiers to launch, distracting them from other tasks. Larger, centrally controlled vehicles, which include the Boeing Scan Eagle, RQ-7 Shadow, and MQ-9 Reaper, are difficult to use for small tactical forces. First, the cost and manpower requirements make dedicating a UAS for each small tactical unit infeasible; second, the time required to request for UAS support and subsequently vector support to an engagement is too long to be useful in many engagements. Thus, UAS capabilities are rarely used to support small unit actions.

The future battlefield environment will necessitate that ground forces achieve a high degree of autonomy within their indigenous weapons systems. This new reality will require blending of the attributes of autonomy with multiple UA—resulting in an effective force multiplier.

The value proposition of this blended approach is that autonomous systems:

- Are capable of operating in denied environments—supporting the rapid shaping of an engagement by denying adversaries effective communications and sensing while maintaining effective communications/sensing for blue forces
- Enable pervasive, persistent perimeters to be maintained—keeping the enemy at arm's length from capital assets. The "perimeter" can also be used to rapidly strike targets of opportunity (because they are pre-positioned).
- Permit persistent tripping of adversary forces' sensor networks—thereby complicating the adversary's ability to effectively determine and then target the critical elements of our attacks. This will produce new opportunities for assured access to previously denied areas.

Autonomously operated UA with various payloads for battlespace awareness, strike, or jamming, will provide small tactical ground units with the ability to protect themselves and facilitate offensive action. This operational concept substantially increases the operational tempo of unmanned vehicle capabilities supplied to front-line squads including:

- Supporting rapid, on-demand tactical strike
- Providing immediate alerts and battlefield intelligence (to threatened squads)
- Providing adaptive, continually reforming communications and navigation infrastructure
- Providing cover via coordinated deception and electronic warfare

Teams of heterogeneous autonomous UA could promote needed agility within small tactical units by providing an organic ability to use UAS support to anticipate threats, provide protection, and facilitate offensive action, as shown in Figure 16. Autonomous UAS support will provide immediate response to unit ISR, EW, and strike needs. In addition to these core capabilities, heterogeneous autonomous UA can further improve unit effectiveness by providing blue force communications, PNT, and blue force tracking.

Launched and recovered from a central base or ship, 10 to 40 medium-sized heterogeneous, autonomous UA could provide a persistent cover to small units operating over large areas. The UAS squadron would provide services to line units on patrol or located in forward outposts at speeds unattainable by human-piloted aircraft by accepting tasking from and providing services directly to the front-line user. Because the individual UA are heterogeneous, the UAS provides an integrated capability far beyond that offered by a single UA. Services provided by the UAS include electro-optic, infrared, and acoustic surveillance; jamming and EW spoofing; PNT, communications; and strike options. Presented with a diverse set of tactical needs from multiple units, the UAS squadron will self-organize, autonomously coordinating to satisfy the needs of all units.

Figure 16 The concept for organic tactical ground vehicle support with an unmanned aircraft system would enable small units to anticipate threats, provide protection, and facilitate offensive action.

The autonomous UAS squadron will be resilient and robust, and capable of operating in denied environments and accepting losses without compromising performance. Each UAS operates independently and is capable of using its own sensors, payloads, and autonomy software to perform mission objectives provided to it by small units. Distributed artificial intelligence is used by *ad hoc* teams of one or more UA to self-organize, divide up tasking, and accomplish the mission objectives provided by small unit users. It should be equipped to support deception tactics and thwart deceptive tactics used by adversaries. Using EW payloads that include jammers, spoofing transmitters, and digital radio frequency memory (DRFM) transceivers, UAS teams will synchronize to provide misleading contacts, obfuscate blue force signatures, identify false targets employed by adversaries, maintain needed communications between small units and command, and limit adversary communications. Communications between a UAS and a peer UAS, end users, and senior command improve mission performance when available, but are required only when lethal force is a factor. UAS resilience is facilitated by the use of an *ad hoc*, delay and disruption tolerant network that allows each UAS to work with local cliques of peers and users when end-to-end connectivity is unavailable, and to use temporary local communications links to coordinate asynchronously. Resilient operations require that common operating pictures are also shared in an asynchronous, *ad hoc* manner that requires decentralized sensor fusion and delay tolerant information exchange. Heterogeneous autonomous UAS squadrons that are capable of performing missions without reliable communications will enable missions in denied environments or when stealth requires a communications blackout.

Small tactical units, particularly in an unexpected engagement, do not have the excess manpower or equipment required to direct UAS operations. For autonomous UASs to be useful, the interface between the frontline unit and the UAS support team must be minimal, consisting of an app on an existing handheld device or audio interactions over an ear bud and microphone, similar to the Apple Siri app. Use of the UAS team is managed by objective; users are not required to provide explicit waypoint directions or to monitor UAS performance.

Recommendation 19.

The U.S. Marine Corps, DARPA, ONR Code 30, and an FFRDC or UARC develop and experiment with a prototype heterogeneous, autonomous UAS support team that includes ten or more UA. The estimated cost for the effort over three years is $40 million.

This experiment should be part of a Marine Corps training exercise conducted at 29 Palms or Camp Lejeune. The exercise should be conducted within three years with a stretch goal of quickly transitioning to a fieldable initial operating capability. In parallel with preparing for and conducting the exercise, the Marine Corps Combat Development Command (MCCDC) should conduct a design competition for a suitable production platform. The prototype UAS should be based upon commercially available hardware, including commercial airframes, payloads, radios, and ground stations. The baseline UAS should be capable of carrying between 2 and 12 kg of payload with a flight durance of at least twelve hours. The following tasks will be necessary in preparing for and conducting the exercise:

- Develop and assess CONOPs for local tactical employment of a small swarm UA fleet in a combat environment (MCCDC, FFRDC, or UARC)

- Integrate low cost UAS fleet of at least ten vehicles with distributed ISR, EW, communications, and strike payloads. The initial strike payload should be a non-lethal proxy that is suitable for experimentation purposes. (DARPA, ONR)
- Develop and integrate autonomy applique for UA fleet. The applique should support autonomous ISR, EW, communications, and strike missions. (FFRDC or UARC)
- Develop and integrate an *ad hoc*, delay or disruption tolerant communications infrastructure for UAS and small unit coordination (FFRDC or UARC)
- Develop and integrate an *ad hoc*, delay or disruption tolerant information management system capable of multi-sensor fusion and sharing common operating pictures (FFRDC or UARC)
- Develop and prototype a lightweight, mission-focused user interface that supports the CONOPs with audio and visual user-UAS dialogues on equipment with a mass of less than 1kg (FFRDC or UARC)
- Develop and refine a launch and recovery process in which a squad of no more than three is capable of recovering, refueling, and re-launching a UA in under 15 minutes (MCCDC)
- Conduct simulation-based testing to validate the CONOPs and prepare for hardware in-the-loop testing (MCCDC, FFRDC or UARC)
- Acquire a non-invasive test process capable of providing UAS safety assurance during experimentation in cooperation with the Test Resource Management Center (TRMC) and the Test Automation Center of Excellence (TACE)

To demonstrate that the heterogeneous, autonomous UAS concept is not only technically and logistically feasible, but economically feasible, MCCDC should conduct an unmanned air vehicle design competition. The competition should provide for an unmanned air vehicle design that includes:

- Non-proprietary avionics bus that supports modular payloads and a secondary processor suitable for hosting autonomy and data fusion software
- Autopilot—a guidance and control sensor suite capable of stable, level flight and waypoint navigation
- Standardized, non-proprietary communications payload bays
- Non-proprietary remotely piloted command and control system
- Integrated software switch allowing the remotely piloted command and control system to switch between automatic and autonomous operating modes
- Minimum 12-hour flight endurance
- Minimum 100 knot speed
- Minimum 2 kg payload capacity
- Modular attachment points to support a gimbaled payload and fixed wing and fuselage payloads

Autonomy for logistics

In any military operation, ensuring protection and timeliness of the U.S. supply chain while disrupting the supply chain of the adversary is arguably the most critical element of a successful military strategy. History validates this with numerous examples, such as the defeat of the British in the Revolutionary War and the defeat of Germany in the African theater in World War II. More recently, the victory over Iraq in Operation Desert Storm again proved the value of sound logistics strategy and execution. In a speech to Congress following the war, President Bush stated, "In a very real sense, this victory belongs to them—to the privates and the pilots, to the sergeants and the supply officers, to the men and women in the machines and to the logisticians who made them work."[42]

Logistics is the management of the flow of things between the point of origin and the point of consumption in order to meet requirements of consumers. The resources managed in logistics include physical items—personnel, equipment, weapons, ammunition, repair parts, food, water, fuel, medical supplies, and so on. It also includes the integrated flow of critical supply chain information to allow the warfighter to plan accordingly. Said another way, logistics is getting the right stuff to the right place at the right time—the what, where, and when of the resource equation.

Many commercial advances in logistics have been adopted by the Defense Logistics Agency (DLA) and the Military Services, resulting in significant gains over recent years through changes to historical distribution, maintenance, inventory management, and procurement practices. For example, the use of SAP supply chain software makes essentially autonomous many basic logistics functions such as customer order processing, contract solicitation, and contract award. Through these improvements, DLA increased sales to the military services from $17 to $46 billion over the past 15 years with no increase in logistics staffing and achieving record levels of readiness.

Employment of logistics autonomy can also be proactively used against an adversary. For example, speeding logistics helps get inside an adversary's decision cycle. Dynamically distributing logistics operations can increase resilience and help thwart attacks against a single, fixed center of gravity. But as dependence on autonomy increases, adversaries will attempt to attack U.S. logistics autonomy, and so logisticians must learn how to deter, pre-empt, and defend against attacks. At the same time, the U.S. must develop methods and effects to counter adversaries' employment of logistics autonomy.

Autonomy in commercial logistics

The study considered some of the rapid advances in the use of automation and robotics in the private sector. The competitive drive to reduce labor costs; improve efficiency in storage, distribution, and energy; capital investment utilization; and rapid responsiveness to market demands has driven commercial logistics to widespread automation and increasing autonomy. While DoD has increased logistics efficiency, commercial logistics leadership provides an opportunity for DoD to selectively adopt or adapt key advances. One critical area is the sensing, reasoning, deciding, and

[42] George H.W. Bush, *Address before a Joint Session of Congress on the End of the Gulf War* [March 6, 1991]. Available at millercenter.org/president/bush/speeches/speech-3430 (Accessed June 2016.)

acting in an anticipatory manner about logistics, both with respect to low-cost consumables and high-value parts and repair.

Consumables: Real time situational awareness and anticipation

Commercial and retail sales in organizations have become highly automated and are increasingly incorporating autonomy to be better informed, anticipatory, and predictive. Low-cost sensors (*e.g.*, RF identification (RFID) location tracking), as well as taggants (*e.g.*, infrared, watermarking, DNA), enable product tracking as well as integrity monitoring. For example, DLA uses vegetable DNA to tag microchips.[43] In terms of sales, commercial providers track what consumers browse and purchase, and provide personalized recommendations for relevant items. Analytics leverage big data from purchasing, inventory systems, social media, weather models, and other sources to forecast demand more accurately in order to improve delivery time scales from days down to—eventually—hours.

Amazon recommender systems are well known for building models of a consumer's purchasing behavior using previous orders, product searches, wish lists, returns, or shopping cart data, and predicting what a customer would likely buy next.[44] Two-day shipping of these items is commonplace now, and the proposed Amazon Air promises even speedier delivery of products up to 5 pounds (86 percent of inventory) within a few years.[45]

Amazon's acquisition of Kiva Technologies further increased the autonomy in fulfillment centers to include not only force multiplication (about four times) of human inventory pickers through robotic movement of materials in warehouses, but also dynamic reconfiguration (and wholesale movement) of inventories within warehouses based on anticipated delivery. Amazon's recent anticipatory package shipping patent extends this dynamic and anticipatory reconfiguration to enroute delivery. The method adds a complete destination address to a package after it has already shipped from the warehouse based on anticipated need reasoning about both a shipping model and a forecasting model. The potential is now emerging for 3D printing in trucks to print customers' product needs enroute, thus reducing storage and transportation costs, delivery time, and carbon footprints.

Walmart has similarly employed enterprise inventory management for predictive supply chain management. Analytics at Walmart Labs gather information from sources, including online purchase transactions, the long-term online shopping records or customer lifecycles of online consumers, and information on industry trends in e-commerce. Walmart recently purchased OneOps, an e-commerce,

---

43 S. Freedberg, *DLA Demands Chip Makers Tag Products with Plant DNA: A War on Counterfeiters* [October 08, 2012]. Available at breakingdefense.com/2012/10/dla-demands-chip-makers-tag-products-with-plant-dna-a-war-on-co (Accessed June 2016.)

44 P. Kopalle, "Why Amazon's Anticipatory Shipping is Pure Genius," *Forbes, On Marketing* [January 28, 2014]. Available at www.forbes.com/sites/onmarketing/2014/01/28/why-amazons-anticipatory-shipping-is-pure-genius (Accessed June 2016.)

45 D. Guarini, "Amazon Reveals It Wants To Deploy Delivery Drones. No Joke," *The Huffington Post* [December 1, 2013]. Available at www.huffingtonpost.com/2013/12/01/amazon-prime-air-delivery-drones_n_4369685.html (Accessed June 2016.)

cloud computing development company, as well as Tasty Labs, which specializes in developing ways for retailers to connect with consumers through social media, and Inkiru, which analyzes "big data" to predict likely conversion rates and fraudulent transaction rates for particular promotions and marketing campaigns.[46]

Other retailers anticipate supply chain disruptions. For example, Home Depot employs severe weather prediction and maintains several key locations outside but close to likely impact areas, then strategically places products so they can get goods to impacted areas quickly. The night prior to a storm, trucks loaded with goods proceed to the projected impact area. This is augmented with a post storm upturn in delivery for six weeks or more.[47] Ace Hardware employs a similar approach, stocking retail support centers with core items such as batteries, flashlights, generators, chain saws, and pumps, as well as clean-up items such as rakes, gloves, and garbage bags. Ace has also invested in a cloud-based transportation management system with a supplier portal, which enables them to interact with suppliers to determine when shipments are ready and send that information to carriers.

Parts and repair: Self-monitoring and anticipatory diagnostics

In addition to forecasting human purchasing behavior, commercial vendors forecast failures of critical parts to prevent failures. Vendors employ built-in, real-time self-monitoring and diagnostics on a variety of high value items. For example, a study by Google found that disc drives with increased heat, noise, and read/write errors detected by self-monitoring, analysis, and reporting technology (SMART) were 39 times more likely to fail, enabling active countermeasures, such as isolating failing sectors to prevent data loss or alerting maintainers to back up the disk.[48]

A broad range of militarily relevant products such as engines, steam turbines, compressors, fans, generators, pumps, heating, ventilation, and cooling can benefit from monitoring and predictive maintenance. Monitoring methods include thermography, tribology (lube oil and wear particle analysis), ultrasonics, visual inspection, and digital testing and analysis.[49] A range of actions can be taken to generate maintenance alarms, initiate work orders, or recommend operator actions. Potential

---

46 P. Demer, *Wal-Mart buys a 'predictive analytics' firm*, *Internet Retailer* [June 10, 2013]. Available at www.internetretailer.com/2013/06/10/wal-mart-buys-predictive-analytics-firm (Accessed June 2016.) See also L. Rao, *Walmart Labs buys data analytics and predictive intelligence startup Inkiru*. [June 10, 2013]. Available at techcrunch.com/2013/06/10/walmart-labs-buys-data-analytics-and-predictive-intelligence-startup-inkiru (Accessed June 2016.)

47 J. Brown, *Forecasting the Unexpected: Home Improvement Retailers and Emergency Response*, Inbound Logistics. [July 2014]. Available at www.inboundlogistics.com/cms/article/forecasting-the-unexpected-home-improvement-retailers-and-emergency-response (Accessed June 2016.)

48 E. Pinheiro, W. Weber, and L. Barroso, *Failure Trends in a Large Disk Drive Population*. Appears in the Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07) [February 2007].

49 K. Mobley, *An Introduction to Predictive Maintenance*, 2nd Edition [2002].

benefits of anticipatory maintenance include system reliability improvement, operational readiness improvement, life extension, failure prevention, spare reduction, and maintenance facility reduction.[50]

In applications where full autonomy can be employed without adding operator and maintenance burden, this would reduce personnel that could result in a reduction in logistics demands. For example, Amazon's employment of Kiva robots reduces by about a quarter the number of human pickers in a warehouse. An additional military benefit of an autonomous warehouse that required fewer personnel in a deployed area would be a reduction of personnel in harm's way.

Because of the high value of many of these benefits, the aviation industry provides many relevant examples. Customers use advanced diagnostics and engine management to help plan engine maintenance to keep costs down and availability up. Mature predictive techniques in Honeywell's predictive trend monitoring and diagnostics tools estimate how many hours are left before major repairs are necessary to its air-transport auxiliary power units (APUs) and to offer troubleshooting tips, including estimates of the probability of each tip's success. Deeper understanding can result in improved designs that extend lifespan, such as the incorporation of 3D printed fuel nozzles designed to stay on the wings 8 to 10 years before their first major overhaul.

Aircraft maintenance analysis today constantly monitors health and transmits faults or warning messages to ground control for customers. Such tools offer rapid access to maintenance documents and troubleshooting steps prioritized by likelihood of success. Real-time health monitoring systems in technical aircraft-on-ground maintenance control centers can give real-time troubleshooting assistance, guide spare provisioning, and monitor system health to anticipate failure. Regional aircraft manufacturers have also created tools that consolidate aircraft data from onboard systems to monitor and recommend maintenance.

Project #9: Predictive Logistics and Adaptive Planning

Data analytics are critical to state of the art commercial supply chain management.[51,52] Predictive analytics, data mining, and decision support allow companies to be more agile, more effective, and more efficient; they can help identify opportunities, quickly address crises and support long-term planning of all activities related to logistics. For example, these systems can exploit the availability of historical data to anticipate opportunities and identify necessary actions (*e.g.*, prepositioning blizzard supplies in advance of weather, changing product lines based on buying trends at particular stores, reorganizing shelves to encourage additional purchases).

---

50 H. Canaday, "New Predictive Maintenance Repair Overhaul (MRO) Tools Cut Costs: Gathering aircraft systems data is often easier than analyzing and determining how to act on it," *Aviation Week & Space Technology* [Feb 11, 2013].

51 J. Baljko, *Betting on Analytics as Supply Chain's Next Big Thing* [May 3, 2013]. Available at www.ebnonline.com/author.asp?section_id=1061&doc_id=262988 (Accessed June 2016.)

52 M.A. Waller and S.E. Fawcett, "Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management," *Journal of Business Logistics* 34, no. 2 [2013], 77-84.

The DoD has even more compelling reasons for collecting and exploiting data about logistics. The operational situations are highly diverse. For example, expeditionary forces may need to bring everything with them, with some uncertainty about what "everything" is. Routine operations are much more predictable in their needs and in the modes for acquisition and transportation.

Geopolitical situations introduce constraints in availability of materiel, options for transportation, variability in information available and needed, sensitivity to culture and laws, and local rules of operation. As staff turns over, knowledge can be lost—both knowledge about how the systems function as well as knowledge about what decisions and options have been most successful in the past.

Logistics for the military means deciding on what resources, such as parts, fuel, materials, information, hardware, software, medical facilities, and so on, are needed by whom, at what times, and through what means, such as sources as well as transportation and handling required. The current logistics system (SAP) includes some capabilities for collecting and using historical data that may be helpful in developing predictive analytics and adaptive planning capabilities.

These capabilities require collaboration with operators to collect, exploit, and act on information. Such interaction may require the software to have a model of what the operator is trying to do and how they can be helped in doing so. A project at Carnegie Mellon University has incorporated plan recognition into a proactive agent for assisting planning for applications such as emergency response and peace-keeping missions.[53] The "Advisable Planning" project at SRI supported mixed-initiative planning by making the internal representations and reasoning accessible to operators.[54] In addition, the "Task Assistant" project at SRI enables organizations to capture and exploit knowledge about plans.[55]

Such enhancements will reduce response time to user requests and reduce the cost of operation by impacting inventory, suitability of supplies, and transportation costs. In a larger sense, improved predictive logistics will decrease the time required to generate new plans.

The military has compelling reasons for collecting and exploiting historical data, considering the turnover in staff and knowledge lost as people move on, the high variance in the situations in which the logistics planning is needed, and the variability in the constraints governing different geographic regions. However, DoD currently develops point solutions rather than implementing a systems approach to address supply chain issues.

---

53 J. Oh, F. Meneguzzi, and K. Sycara, "Probabilistic plan recognition for proactive assistant agents," In *Plan, Activity, and Intent Recognition: Theory and Practice*, edited by G. Sukthankar, R. P. Goldman, C. Geib, D. V. Pynadath, and H. H. Bui, Elsevier [2014].

54 K. Myers and T. Lee, *Generating Qualitatively Different Plans through Metatheoretic Biases*, in Proceedings of the Sixteenth National Conference on Artificial Intelligence [AAAI Press, 1999].

55 B. Peintner, *Task Assistant* [2015]. Available at www.sri.com/sites/default/files/publications/1756.pdf (Accessed June 2016.)

Recommendation 20.

Naval Supply Systems Command (NAVSUP) should demonstrate the use of modern intelligent adaptive planning in conjunction with SAP. The cost is estimated to be $10 million over two years. The key characteristics are to:

- Capture richer historical data, such as constraints that dictated the prior plan, metrics on costs and effectiveness, and logistician's comments and notes
- Use analytics to make recommendations for new missions based on prior missions
- Autonomously make decisions such as critical supply prepositioning or dynamic plan adjustments

The use of analytics to make starting recommendations for new missions based on prior missions is a goal of the demonstration. The difficulty of resupply for the Navy makes them an excellent first adopter because they are already exploring opportunities such as additive manufacturing for creating parts on demand.

Project #10: Adaptive Logistics for Rapid Deployment

Deploying logistics warehouses is slow, costly, and problematic. Leveraging advances in commercial warehousing and logistics planning could reinvent military logistics deployments. For example, algorithms can use delivery deadlines to select the best combination of long-haul and short-haul shipping to meet all deadlines while minimizing shipping costs for the total flow of material. Machine learning and knowledge of upcoming sales can be used to preposition inventory geographically to anticipate demand while limiting over-ordering. As items become obsolete or as vendors change packaging, automation detects these issues, stops ordering obsolete product, and disambiguates between individual items, packages of items, or cases of packages of items. This keeps inventory accurate, and the right kind of number of items accurate within orders.

Entire warehouses using modular robotic components have been moved from one site to another over 48 hours.[56] Shelves were shrink-wrapped and shipped as is. Robot stations and chargers were unbolted from the floor and packed for shipping. Reestablishing the warehouse at the new site involved laying stickers on the floor, bolting down robot stations and chargers, then letting the robots autonomously store the shelves on the new floor. This was done within 48 hours in a commercial setting. Similar operations for infantry units took 144 hours. The time for an initial warehouse installation can be three to six months, significantly shortened from 18 months. Robotic sites can be designed, constructed, and brought online much more quickly than conveyance-based systems that require detailed design for every join and split point in material flow, and significant interconnection of components during construction.

[56] J. Dineen, "Meet the Robot Armies that are Transforming Amazon's Warehouses," *Forbes CITVoice*. [2015]. Available at www.forbes.com/sites/cit/2015/03/20/meet-the-robot-armies-that-are-transforming-amazons-warehouses/#781a0db723ee (Accessed March 2016.)

Recommendation 21.

Combined Arms Support Command (CASCOM), Ft Lee 10th Mountain Division, and the Ft Drum Joint Readiness Training Center should develop and deploy adaptive logistics decision support for a relocatable robotic warehouse and trained personnel in preparation for rapid deployment to unstable regions. The estimated cost for this effort is estimated to be $30 million over three years. The implement steps for this recommendation should include:

- Create an adaptive planning process for rapid deployment
- Develop a relocatable robotic warehouse at Ft Lee
- Train a core group of logisticians to adapt plans
- Assign new logisticians to a unit
- Deploy to Ft Drum Joint Readiness Training Center with new capabilities
- Prepare for rapid deployment

CASCOM will need to baseline current operations and recommend level of adoption of robotic warehouse technologies and adaptive planning.

5 Expanding the Envelope

The previous chapter described a set of representative demonstrations that are "ready now," based on the commercial and military advances of recent years. Autonomous systems offer the promise of even greater capability in the future, especially as commercial markets continue to drive the advance of underlying technologies—see, for examples, the "Imagine If…" possibilities provided in the Introduction to this report. However, converting commercial advances into military capability such as these requires two additional elements: operational pull and technology maturation in aspects unique to military needs. "Stretch problems," as described here, are proposed as a mechanism to drive both elements, thereby expanding the envelope of technology available to support military goals. Here, each stretch problem is recommended in support of its own "Imagine If…" possibility. In practice, the recommended stretch problems also provide additional venues for practicing the first 11 recommendations of this study, those designed to "accelerate the adoption of autonomous systems." Execution of stretch problems, such as those recommended, confronts the issues of trust and cultural barriers by building familiarity and by increasing transparency of autonomous "reasoning." The problems involve varying types of human-machine teaming, can provide valuable insight into adversaries' possible use of autonomy, tackle cyber vulnerabilities, emphasize the use of M&S, and offer opportunities for the T&E community to engage with learning systems.

A stretch problem is a goal that is "hard-but-not-too-hard," and its purpose is to accelerate the process of bringing a new capability into widespread application. The most successful stretch problems are ones that largely leverage existing technology, with additional technology development as the "glue" necessary for integrating an end-to-end solution.

Stretch problems have been used successfully in a variety of implementations. For example, DARPA Grand Challenges (Mojave and Urban) offered cash prizes for successful demonstrations of autonomous navigation of unmanned ground vehicles, initially off-road and subsequently on-road in traffic. These successes galvanized today's investment and progress in autonomous automobiles. More recently, the DARPA Robotics Challenge stimulated significant progress in controlling humanoid robots to support disaster relief missions. The Ansari X PRIZE awarded $10 million to the first team to "build a reliable, reusable, privately financed, manned spaceship capable of carrying three people to 100 kilometers above the Earth's surface twice within two weeks." Other stretch problems are structured around a game construct: for example, RoboCup and *FIRST*® are both designed to allow teams to create their own autonomous players, with each team competing against others in structured events.

These various implementations of stretch problems have some essential commonalities: they accelerate progress by generating excitement, spurring creative approaches, and exploiting dynamic tension among competitors. In formulating these stretch problems, the tenets began with a crisp definition of a mission-relevant goal (or goals). The goal should be bold enough to capture the imagination and attract participants from the full range of technology providers: the fast-moving commercial industry, whether large commercial companies that can operate at scale or small start-up companies that drive some of the best innovations; and the academic community, including both students and professors. Broad participation accelerates technology maturation as necessary to real the stretch goals.

These examples provide additional lessons. They typically involve repeated trials, some wildly unsuccessful at first but with later efforts building on the success of prior activities. Success metrics should be simple and clearly defined; they should describe "what" defines success without needlessly restricting "how" the goal is reached. Financial prizes should be awarded when the goals are met. Supporting infrastructure is required—competition or training ranges, starter-kits or basic platforms, simulation capabilities, data sets. The stretch problems that follow are structured around repeated competitions, and include a brief description of some of the supporting infrastructure anticipated to be necessary.

Another essential aspect of the proposed stretch problems is the participation of the full range of DoD stakeholders throughout their execution. Getting operators involved in these stretch problems will give them hands-on exposure to better understand autonomy's value in military missions, creating "operational pull" and shaping the requirements for future procurements. The acquisition establishment must participate, to ensure the right capabilities are developed and fielded and to better assess the make/buy (or adopt/adapt) trades. The stretch problems also provide a venue for developing the testing methodology appropriate to autonomous systems. In addition, because autonomy is expected to provide the greatest value by enabling new missions (rather than in simply substituting machines for humans), it is essential that this hands-on experimentation explicitly consider the CONOPs, doctrine, and policy implications for new ways to use new systems. This requires up-front involvement from those communities, as well. Thus, a diverse set of DoD stakeholders must participate simultaneously as the operational pull is being created, so that all relevant equities are considered and iterated together.

This approach is in contrast to traditional DoD processes used for acquisition in which both operational needs and technology enablers are well understood. Because the types of autonomous capabilities described here are so unfamiliar, and the CONOPs for their use so undefined, the traditional, more sequential approach to stakeholder involvement would be fraught with opportunities for failure and delay. The reasons could be whether operational and policy conflicts were only identified after procurement, the development community did not make use of fast-moving commercial developments, testing methods were not suited to inherent system characteristics, or because of myriad other misalignments that could occur.

For these reasons, the study recommends that each stretch problem include active engagement by the full range of DoD stakeholders, as shown in **Table 2**. This aspect sets our recommendations for stretch problems somewhat apart from prior DoD experience with challenges and competitions. It may be most akin to the Army Warfighting Assessment at the U.S. Army Training and Doctrine Command (TRADOC), an annual wargame planned to start in 2017 and intended to bring together operators and industry to explore ideas for using new

Table 2 Value in participating in stretch problems

DoD Stakeholder	**Insight gained by participating in stretch problems:**
Operators Doctrine writers	Potential uses, limitations, and vulnerabilities of autonomous systems
Policy makers	Implications for policies related to autonomous systems (*e.g.*, rules of engagement, etc.)
Testers	Testing methodologies suited to complex, software-intensive, and learning systems
Requirements community	Requirements for autonomy and counter autonomy systems
Acquisition community	Identification of new, high-payoff programs Limitations of commercially available technology, to clarify the adopt/adapt/develop acquisition strategy
DoD S&T community	Identification of priority focus areas for aligning technology investments
Non-DoD Providers	**Attractiveness of participating in stretch problems**
Academia	Maturing technology to enable solutions to important, hard problems
Start-up companies	Potential new markets where they have essential differentiation
Commercial industry	Potential new markets they can evaluate without normal burden of government contracting
Defense industrial base	Understanding system integration opportunities for future programs of record

technologies, as well as understanding their impact on tactics and CONOPs.[57] However, the study recommendation goes a step further in also advocating the involvement of policy makers and the testing community—the latter being involved not to impose traditional test methods, but to learn how to create and use test methods suited to software-centric, adaptive, and learning systems. The full spectrum of participants is a way to build trust, as described in Chapter 2.

Each stretch problem that follows is motivated by a vision for an important new military capability. Current technology enablers and shortfalls are identified. The essential role of autonomy in developing the capability is articulated and each stretch problem is outlined to show how it can accelerate the development and use of the envisioned capability.

Finally, it is important to emphasize what the stretch problems are not. They are not traditional Programs of Record that result in the procurement of materiel vetted for all the "-ilities"

[57] S.J. Freedberg, Jr., "AWA is not NIE: Army tries to buy weapons that work," *Breaking Defense* [2015]. Available at **breakingdefense.com/2015/04/awa-is-not-nie-army-wrestles-with-requirements-reform** (Accessed June 2016.)

necessary for fielded systems. Rather, they are carefully constructed opportunities for the Department to engage with the full range of technology providers in a way that purposely accelerates technology maturation as they stimulate and clarify operational pull.

Generating future loop options

Imagine if national leaders had sufficient time to act in emerging regional hotspots to safeguard U.S. interests using interpretation of massive data including social media and rapidly generated strategic options.

Accurate and timely understanding of global social movements is critical for protecting U.S. interests. How many lives might have been saved with a timely anticipation of Arab Spring, or a clear understanding of situations unfolding around our embassies? Providing our national leaders with a well-considered slate of strategic options—diplomatic, information, military, economic—requires improved early recognition of emerging geopolitical events as well as an understanding of event drivers and repercussions, causal linkages, and possible non-kinetic solutions.

Such a capability may soon be achievable. Massive datasets are increasingly abundant and could contain predictive clues—especially social media and open-source intelligence. The U.S. uniquely enjoys access to open-source data as well as the full cadre of DoD's multiple intelligence sources. Recent advances in data science, ever-improving computational resources, and new insights into social dynamics offer the possibility that we could leverage these massive data sources to make actionable predictions about future world events.

An autonomous early awareness system could:

- Ingest and event-code the wide array of data sources available today, including multiple intelligence sources, open source data, and social media, all in real-time with minimal human intervention to…
- Identify causal linkages between actions and outcomes globally to…
- Use these linkages to identify possible future outcomes, assess "what-if" scenarios, and analyze candidate U.S. courses of action.

The purpose of such a system would be to better understand possible future trajectories of unfolding events, and to help decision-makers assess various shaping options by estimating their likely impacts and repercussions.

Some essential first steps have recently been taken along the path towards creating such a system. For example, military and commercial systems have demonstrated the ability to forecast sentiment, threats, and disease outbreaks. DoD's program on Integrated Conflict Early Warning System (ICEWS) uses static models and various raw data sources, news text, and econometric models to generate monthly forecasts for individual countries. IARPA's Open Source Indicators (OSI) program, including Virginia Tech's Early Model Based Event Recognition using Surrogates (EMBERS) efforts, have used high-velocity ingest of open source and social media data to demonstrate seven-day lead time for civil protest and three-week lead time for disease outbreaks, as compared to the World Health Organization assessments. These and other efforts demonstrate it is possible to forecast future levels of instability—sometimes even insurgencies and rebellions—with remarkable accuracy, months into the future.

While impressive, these systems have limitations. They are based on correlations between situations and outcomes, which limits their ability to elucidate underlying causes. By analogy to weather forecasting, ICEWS can make predictions such as "40% chance of rain on Monday" but does not provide a clear explanation for why it will rain, or how the weather will unfold from Monday to Tuesday. OSI provides leading indicators of unrest (*e.g.*, indications of the arrival of a storm front), but not a predicted event trajectory or insight into what could influence the trajectory.

Creating the type of system envisioned here requires moving from todays' correlation-based "forecasting" models to models that identify the causal linkages that underlie emerging social movements. Such models would illuminate the interconnected drivers of observed behavior, and thus provide a basis for enumerating possible future event trajectories and assessing the impact of various courses of action. Continuing the weather analogy, causal linkages would allow us to go beyond simply forecasting a storm to understanding the possibilities for how storm might evolve over time, and to assessing the impact of options under our control, such as moving civilians to safer locations, releasing water from a dam, and so forth. But the envisioned system must also account for the fact that, unlike weather-related events, human behavior changes in response to our actions, and correctly capturing this feedback is essential. For DoD applications, such a system would require algorithms that sense the state of the world and build an internal representation of the underlying causal linkages. These algorithms would use statistically based extrapolation to identify possible future event trajectories and their likelihoods; algorithms for planning and analysis that generate large numbers of possible courses of actions and assess likely outcomes; and algorithms to assess the impact of an intervention to learn how to have the desired effect in the target country. These capabilities build upon but go beyond what is available today.

Causal models have a further advantage over correlation-based models in their ability to handle rare events. Correlation-based models require large training sets, so they have difficulty forecasting events that occur infrequently, such as *coups d'état*. Causal models replace the requirement for many *coup* examples in the training set with a requirement to clearly understand the precursors for a *coup* to take place.

Weather is a helpful analogy, but it is much simpler to predict the weather and its consequences than it is to unwind the myriad, interrelated elements that impact the trajectory of social movements. Nonetheless, recent advances suggest the creation of such causal models is becoming possible. An explosion of techniques in machine learning and deep learning; cognitive and social science modeling of populations, groups and individuals; and our growing understanding of the role and impact of social media on society and culture are key enablers.

A major difficulty in creating an autonomous system for identifying causal relationships is the absence of ground truth regarding predictions for future events. The recommended approach deals with that problem by relying on ground truth where available (*i.e.*, for historical events) and by emphasizing transparency about model "reasoning" and evidentiary support for predictions regarding possible future trajectories.

Autonomy is essential in the envisioned system, first and foremost, because of the scope, variety, and complexity of data that must be continuously and quickly analyzed. In addition, the number of branching paths for future event trajectories and their various probabilities would be far beyond the ability of humans to track manually. Finally, autonomy will be necessary for both creating the underlying models and in generating their training data by event-coding various data sets. Today, each aspect requires significant human involvement, which represents a stifling bottleneck that must be overcome before it will be possible to operate at the scale and complexity envisioned in this application. Instead of manually creating each model and coding individual events, humans will use their expertise to create the causal schema (universe of candidate causal architectures) from which the machines can learn and select the appropriate causal models.

Recommendation 22.

DARPA should initiate a stretch problem designed to create a system that autonomously, globally, and in real-time identifies the causal linkages behind emerging social movements, and helps leaders understand the impact of possible courses of action along various possible future event trajectories. It is estimated this will take four years and cost approximately $75 million in total.

- A critical requirement of this project is the construction of a digital test range as a scale model of society, analogous to the National Cyber Range, the scale model of the Internet used to carry out cyber wargames. DoD should build out the test range and equip it with huge volumes of unclassified data about recent historical and ongoing "events" from a wide variety of open sources, including social media. The test range must also include models that provide the best available simulation of societies and their interactions at multiple scales (*i.e.*, groups, regions, countries, and ideologies). The range should be staffed by government subject matter experts with backgrounds in social sciences, who operate the range; find, acquire, and curate the data; and procure and validate simulations. They facilitate and manage community involvement in the competitions.
- The competition should be structured in two parts: an ongoing "qualification" phase and a "prize" phase consisting of regularly scheduled competitions. The "qualification" phase is focused on historical data, where ground truth is available. In this stage the full complement of technology participants (*i.e.*, government, FFRDC, defense industrial base, commercial companies, academia, startups) are encouraged to use the test range to develop and calibrate their causal models, with the goal of each developing a system that accurately captures the dynamics that drive large-scale social, societal, or government movements. The systems ingest historical data and attempt to predict likely future trajectories for the movements. Participants qualify for the next round whenever they can accurately "predict" the outcome of a particular social or societal movement that has been withheld from the

training set and for which, during the qualifying test, they are provided data that stops short of the actual event to be predicted.[58]

- The "prize" phase is available to all participants who successfully qualify as above, and this phase is repeated on a regular basis. Its focus is on predictions about the unfolding of events underway at the time of each competition, so absolute ground truth is elusive. In this phase, the predictive systems compete head-to-head, and prizes are awarded for best performance in two distinct categories. The first category focuses on the persuasiveness of a system's predictions. Here, competitors' models are judged against each other for their clarity and succinctness in identifying the causal relationships that drive the unfolding events that are the focus of the current challenge, and in providing the evidentiary support for those relationships. This focus on transparency of model "reasoning" helps create trust in the model predictions, and allows users to better recognize when the model is appropriate or not. The second category of award focuses on completeness in enumerating future event paths. Here, competitors' models are judged against each other for their ability to identify a full range of possible future outcomes. More weight can be given to future outcomes that are strongly supported by evidence, as identified in the first type of competition. Prizes are awarded for the best performers in each category during every round of competition. The government user community should play a key role in judging and selecting the winners.
- Although beyond the scope of the stretch problem described here, it is expected that the most compelling models will attract further interest from the user judges. This could lead to an expansion of the test range to include classified data, so that the users and (appropriately cleared) competitors could further test and tune the most successful models based on the full data sets available to operational users.

Some critical elements of this program may include:

- A strong emphasis on the development of autonomous capabilities for both model building and event-coding, which are missing enablers for the overall system and missing from current programs in this area.
- Model development that emphasizes the generation of human-understandable explanation of the causal linkages discovered by models, rather than allowing the "explanations" to be implicit and hidden within proprietary code.
- Access to large volumes of high-quality training data. Just as access to training data ushered in an explosion in computer vision technologies, so will the creation, curation, and maintenance of a global event database serve to foster innovative approaches to building and testing the types of models required for this capability–and for other government needs.[59]

[58] For example, all data associated with Crimea/Ukraine may be withheld from the training set, and the start of hostilities may be the event to be predicted. During qualification, "predict" means the system identifies the "actual" outcome as one of the possible likely outcomes generated by the system's underlying causal models.

[59] The study notes that the government often pays multiple times for the same data. For example, one program manager suspected that the government must own 100+ separate contracts for Twitter data. In creating the training database for this program, the Department should determine how to host the data (data.gov? dataverse?) to make it available for other appropriate uses.

- Person-to-person interaction as an important contributor to innovation suggests the testbed be housed in Silicon Valley; Cambridge, Massachusetts; or other innovation hub in order to maximize in-person involvement by both commercial and academic participants, increase the exchange of ideas, and speed innovation.
- While meeting these goals will build on recent commercial and military accomplishments, success will nonetheless require substantial, purpose-built technology development.

Enabling autonomous swarms

Imagine if commanders could deny the enemy's ability to disrupt through quantity by launching overwhelming numbers of low-cost assets that cooperate to defeat the threat.

In military applications, swarming is a convergent attack from many directions. As described in *Swarming & The Future of Conflict*, "Swarming is seemingly amorphous, but it is a deliberately structured, coordinated, strategic way to strike from all directions, by means of a sustainable pulsing of force and/or fire…".[60] Many scholars have considered the role that swarming might play in future warfare, including the role for autonomous weapons systems as part of a swarm (or constituting a swarm in its entirety). For instance, *Robots on the Battlefield II: The Coming Swarm* provides a recent, comprehensive discussion on the topic.[61] As robotic platforms become increasingly capable, some role for autonomous swarms seems highly feasible–although not yet proven.

Robotic swarms could be constituted over a wide range of characteristics, as shown in **Error! Reference source not found.**. The Department has a number of "swarm" efforts underway, and these are making important progress in understanding how robotic platforms can cooperate with each other and with humans in the performance of military missions. However, the study observes that the current efforts tend to align towards the *left* end of the possible range for the attributes itemized in Table 4. This alignment is remarkable for two reasons. First, the left end of the range is the more difficult to implement in autonomous systems. Second, the emphasis of current work is at odds with the admiration that military proponents of swarms often express when observing nature's examples of swarms, such as hive insects, which are aligned with the *right* end of the table's range of attributes. Hives consists of massive numbers of genetically identical organisms that obey very simple rule sets and have limited (or no) direct, peer-to-peer communication. Nonetheless, they can demonstrate sufficient coordination to accomplish tasks far beyond the capability of any individual, and even allow adaptation to changing environments.[62] For example, fire ants use a few simple rules to build bridges out of their own bodies so that the colony can float or cross bodies of water, as shown in Figure 17.

60 J. Arquilla and D. Ronfeldt, *Swarming and the Future of Conflict*, RAND/D8-311-OSD [RAND CORP Santa Monica CA, 2000].

61 P. Scharre, *Robotics on the Battlefield Part II: The Coming Swarm*, The Center for New American Research [2014].

62 While the organisms are identical, the functions they perform for the hive vary.

The simple self-construction methods result in proliferated air pockets that promote buoyancy and allow ants on the bottom layer to breathe.[63]

Figure 17 Raft built entirely of fire ants, where the building follows a few simple rules and results in a buoyant structure that allows ants to survive until they reach dry land.

SOURCE: National Park Service, available at www.nps.gov/akr/photosmultimedia/photogallery.htm?id=385E5498-1DD8-B71C-073997EB3E9682E1

Such significant accomplishments by organisms as simple as insects offer an intriguing possibility: that very large numbers of platforms with limited individual capability might be able to accomplish meaningful military missions. The possibility is attractive for several reasons. As limited-capability platforms, they could be cheap enough individually to be affordable for proliferation in overwhelming numbers; this would allow truly attritable assets, enabling exploration of missions and tactics in which "quantity has a quality all its own". Limited communication or direct, peer-to-peer coordination between platforms offers a built-in robustness to the jamming and contested electromagnetic spectrum that is expected to be a constant on future battlefields. Finally, the simple rules followed by individual insects give rise to emergent behaviors, *i.e.*, the collective behavior is different than that exhibited by the individuals. The collective behavior of an emergent system can depend strongly on the environmental conditions, even when the basic rule set followed by individual members is essentially constant. In principle, emergent behavior could lead to highly adaptive military systems. However, predicting collective behaviors from the rules followed by individual entities is difficult, and today it would be difficult to know *a priori* if the collective's adaptive responses would be beneficial or detrimental to a military mission.

For the remainder of this discussion, the term "swarm" is used to mean a massive collection of hundreds or thousands of simple autonomous systems with characteristics described by the *right* side of **Error! Reference source not found.**. This is not to suggest that other uses of the term "swarm" are inappropriate or to question the value of missions they might carry out. The purpose of this definition is simply to clarify the type of swarm that is the focus here.

[63] N. Mlot, C. Tovey, and D. Hu, *Dynamics and Shape of Large Fire Ant Rafts*. Commun. Integr. Biol. 5 [2012], p. 590-597.

Table 3 Forms of Robotic Swarms

Attribute	Harder to implement *Most DoD "swarm" efforts*	Easier to implement *Natural examples*
Diversity	Heterogeneous *e.g.*, mixed ground and air platforms	Homogeneous *e.g.*, standard platform, perhaps with modular payloads
"Intelligence"	High *e.g.*, complex reasoning	Minimal *e.g.*, simple, pre-defined rule sets
C2/decision making	Complex *e.g.*, highly interactive decisions	Minimal *e.g.*, implicit C2
Communications bandwidth	High *e.g.*, to provide detailed intra- (or extra-) swarm updates	Low *e.g.*, stigmergy (environmental marking)
Complexity of human interaction	High *e.g.*, could require advanced human-machine interface	Minimal *e.g.*, limited to human giving "Go" command

Massive swarms of this type might be used for various offensive or defensive missions (examples are provided below). Indeed, the most effective counter to massive swarms may be other massive swarms. The possibility of having to face massive swarms is a good reason for the Department to accelerate its understanding of these systems.

One type of mission in which an overwhelming number of simple platforms might be effective is in disrupting operations at a forward arming and refueling point (FARP). Exposed fuel, munitions, and runways (or landing areas) all represent points of vulnerability that could be disrupted with relatively small explosive payloads. These payloads could be delivered by a large number of small fixed- or rotor-wing aircraft using crude targeting methods, relying on random delivery within a defined area to statistically ensure coverage of the FARP, rather than pinpoint targeting or between-platform coordination and deconfliction. While these threats could be simple to target individually, doing so would distract from executing the FARP's mission and therefore degrade operations. And in sufficiently large numbers, they could overwhelm the FARP's defensive capacity by depleting magazine depth, or presenting more targets than could be prosecuted at once. In this case, one element of countering such a swarm might be another swarm, consisting of platforms that are at least as agile as the offensive swarm and that intercept and detonate the intruders before they can reach the FARP's sensitive points.

Another mission for a massive number of simple platforms could be in agile mines. An adversary could use mobile mines to continuously self-replenish a mined area, or to mine an area previously determined to be mine-free by U.S. forces. Such mines could use very simple rules and coordination. For example, they might detect whether they are close to any other mines; if so they could move in some predetermined manner, and if not they could remain in place. While we have the ability to deal with mines, an apparently endless self-replenishment capability increases their disruptive capacity. Again, there might be a role for a counter-mine-swarm to augment our existing counter-mine capability.

As a final example, swarms may be well suited for roles in the radio frequency domain, where having a large number of platforms allows geographic diversity to be exploited in new ways. Swarms could enable a new and fatal form of communications "jamming", in which a large number of quadcopter-borne RF sensors are pre-positioned to blanket a contested area. When a sensor detects adversary communications coming online in its vicinity (*e.g.*, simply by detecting an increase in signal intensity in the appropriate communication bands), the quadcopter could fly into the communications emitter and self-detonate. If the emitter stops transmitting before the quadcopter self-detonates, the quadcopter could assume another got there first and it could re-settle until needed again. Thus, peer-to-peer coordination could be unnecessary. A similar technique might be effective to counter a swarm of low-power, proliferated barrage jammers that are interfering with U.S. or allied communications and that are difficult to defeat using traditional methods. More advanced versions of swarms that use DRFM techniques could exploit geographical diversity in additional ways. For example, swarms could use DRFMs across multiple, mobile, blinking, and cooperating emitters to spoof U.S. radars or screen high-value targets. Counter-swarms having their own mobility and operating within the area of interest could help unwind ground truth from false data, by resolving and geolocating the full variety of emitters.

Recent advances driving the feasibility of such swarms include the proliferation of unmanned platforms–ground, air, and sea. For example, sales of quadcopters are exploding worldwide. Precise figures are difficult to determine, but estimates are that consumer drone sales have grown to over a million a year recently.[64] Autonomous navigation and control systems are readily available, as are small, high-quality cameras and other electronic components. Buzz, an open-source programming language specifically designed for understanding and predicting swarm behavior, may also speed innovation in this area.[65]

The swarms contemplated here are autonomous, by assumption. They are intended to have the simplest possible interaction with humans, along the lines of receiving a "go" command from their human operators. Even without this restriction, the hundreds-to-thousands of individual platforms would be beyond the ability of humans to control directly.

---

64 A. Amato, *Drone Sales Numbers: Nobody knows, so we venture a guess*. [April 16, 2015.] Available at dronelife.com/2015/04/16/drone-sales-numbers-nobody-knows-so-we-venture-a-guess (Accessed June 2016.) See also R. Lever, *Drones swoop into electronic show as interest surges*, YahooTech. [January 7, 2015.] Available at www.yahoo.com/tech/s/drones-swoop-electronics-show-interest-surges-061549575.html (Accessed June 2016.)

65 C. Pinciroli, A. Lee-Brown, and G. Beltrame, *Buzz: A novel programming language for heterogeneous robot swarms*. Available at robohub.org/buzz-a-novel-programming-language-for-heterogeneous-robot-swarms (Accessed May 2016.)

Recommendation 23.

The Assistant Secretary of the Army for Acquisition, Logistics, and Technology (ASA(ALT)), with close participation by the Army Capabilities Integration Center (ARCIC), should establish an annual "swarm games" challenge. This is expected to cost about $25 million per year, including funding for facilities, participants, and equipment.

- This stretch problem is envisioned as an annual game to encourage open exploration of a variety of concepts. Each event should define specific mission goals that are appropriate for massive swarms, and participants may field teams to either accomplish or defeat those mission goals ("offense" and "defense"). This asymmetry makes these swarm games different from other robotic games: the two sides have different purposes, they do not have to play by the same rules, and they need not field identical teams. To further enhance the realism of these games, the organizer may choose to impose different rules of engagement on the two sides.
- The games can be structured as a series of head-to-head competitions, with prizes given for the best offense, best defense, and the overall winner. The game organizer is responsible for clearly defining the metrics for each category, as appropriate for each mission. Candidate missions include disrupting operations at a fuel depot, attacking a fixed facility, disrupting ground-force maneuvers, jamming communications, and spoofing and decoying sensors.
- There are several critical considerations for these games. First, it requires an outdoor test arena suitable for (simulated) kinetic operations; the facility must include instrumentation to measure impact on operations, score the games, and play back events. Second, testing should include persistent RF jamming, to enforce the strong limitations on intra- and inter-swarm communications necessary for operations in RF-denied environments. Third, the government should offer to furnish participants with basic platforms and payloads to encourage focus on the development of algorithms and CONOPs. Finally, the government should make available a common simulation environment for use by participants and use simulation results to determine which participants may progress to the live games.

Intrusion detection on the Internet of Things

Imagine if commanders could defeat adversary intrusions in the vast network of commercial sensors and devices by autonomously discovering subtle indicators of compromise hidden within a flood of ordinary traffic.

The Internet of Things (IoT) is the set of IP-addressable devices that interact with the physical environment. IoT devices typically contain elements for sensing, communications, computational processing, and actuation. They span a range of complexity and physical size—from thermostats to traffic lights to televisions, from mini-drones to full-size vehicles. Applications include media targeting, data capture, environmental monitoring, infrastructure management, manufacturing, energy management, medical and healthcare systems, building and home automation, and

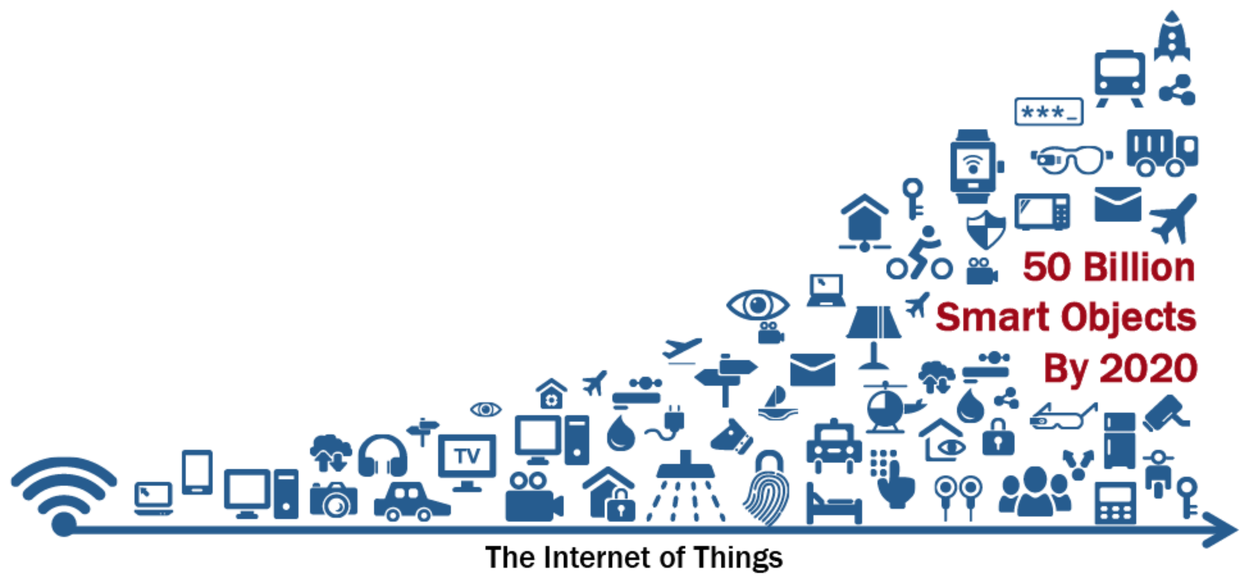


Figure 17 The Internet of Things is increasing rapidly in both numbers and types of smart objects.

transportation. In principle, the IoT can incorporate almost any device or function imaginable—and countless not yet imagined.

The IoT is vast and growing rapidly, as shown in Figure 18. According to one estimate, more than 50 billion IoT devices will generate more than $3 trillion in spending by 2020.[66] The Department of Defense is already a major presence on the IoT as a large purchaser of many kinds of innovative commercial devices, and it is likely to develop or exploit many more IoT devices in the future. Additionally, the myriad devices of more than three million service members and employees also reside in this globe-spanning network.

The seemingly limitless opportunities afforded by the Internet of Things also bring deeply embedded risks. Here we focus on just a few aspects of those risks: vast scale, limited configurability, and already demonstrated security flaws.

With the shift to the IPv6 addressing standard, the number of devices that can be networked is many orders of magnitude larger than the current Internet. IPv6 uses 128-bit addresses, meaning that it can, in principle, handle up to 340 undecillion unique devices (340 with 36 zeroes after it); this compares to fewer than five billion devices under the outgoing IPv4 standard. This immense, sparsely populated space of interconnected devices could serve as a globe-spanning, multi-sensing surveillance system or as a platform for massively proliferated, distributed cyber-attacks—or as an immense test range for real-world, non-permissive testing of large-scale autonomous systems and swarms.

By their nature, the IoT's small, networked devices are designed for simple operation. The devices' inner workings are hidden to the degree possible, and configuration options are limited. This makes it very difficult for end users to mitigate risks or to detect when devices or data have

66 *Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015.* Available at www.gartner.com/newsroom/id/3165317 (Accessed June 2016.)

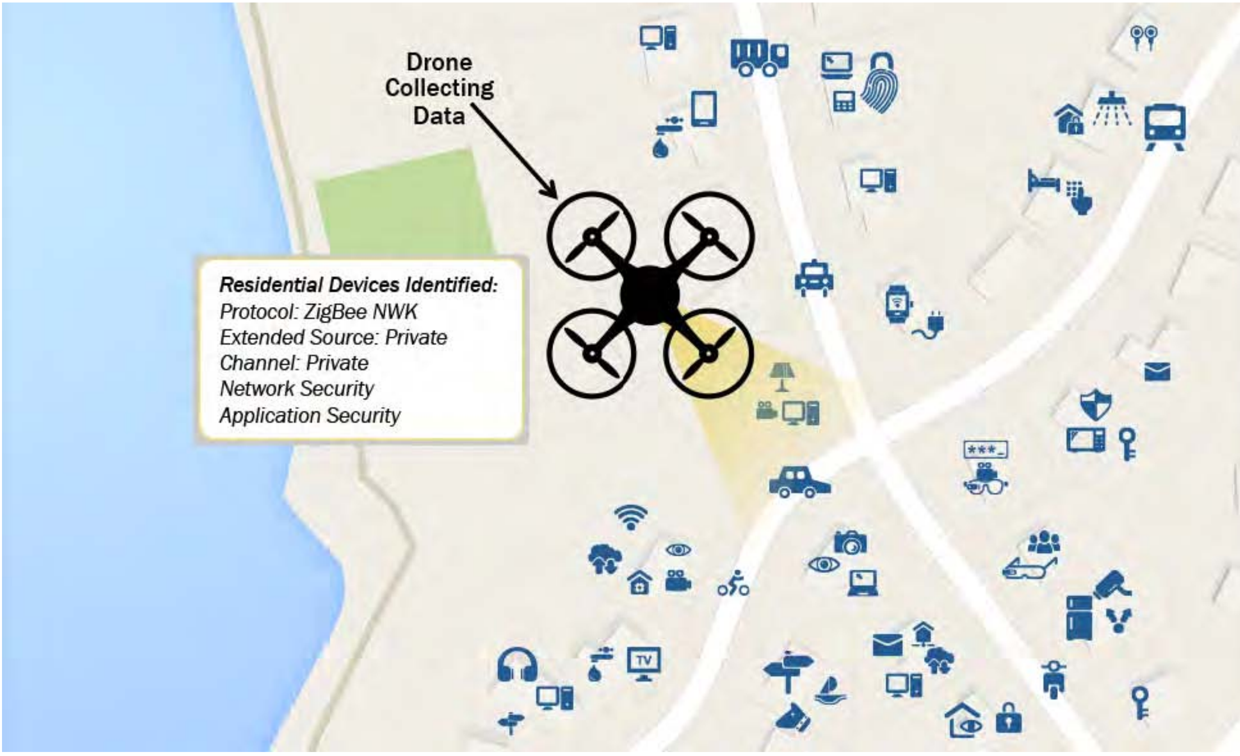


Figure 18 This schematic of a drone collecting data from the Internet of Things in a typical neighborhood shows how IoT devices can wirelessly identified and mapped.

SOURCE: M. Kumar, How Drones Can Find and Hack Internet-of-Things Devices From the Sky. thehackernews.com/2015/08/hacking-internet-of-things-drone.html [2015].

been compromised. For example, a flying drone with a custom tracking tool has demonstrated how certain IoT devices can be wirelessly identified and mapped with no one the wiser, as shown in Figure 19.[67] Without the knowledge of their owners, the high-quality microphones of other IoT devices have been hijacked to eavesdrop on conversations in the room. Chillingly, the small print in the documentation of Samsung's "Smart TV" states it "will not only capture your private conversations, but also pass them onto third parties".[68]

Today's IoT devices typically have little built-in security. According to a recent study by Hewlett-Packard, 70 percent of the most commonly used IoT devices contain certain exploitable vulnerabilities. Each device studied had approximately 25 security holes.[69] Imagine the consequences

[67] M. Kumar, *How Drones Can Find and Hack Internet-of-Things Devices From the Sky*, The Hacker News [August 7, 2015]. Available at thehackernews.com/2015/08/hacking-internet-of-things-drone.html (Accessed June 2016.)

[68] C. Matyszczyk, *Samsung's warning: Our smart TVs record your living room chatter* [February 8, 2015]. www.cnet.com/news/samsungs-warning-our-smart-tvs-record-your-living-room-chatter (Accessed June 2016.)

[69] K. Nelson, *70 Percent of Internet of Things Devices Are Vulnerable to Hacking, Study Says* [August 2, 2014]. mashable.com/2014/08/02/internet-of-things-hacking-study (Accessed June 2016.)

if the effects of a recent demonstration where attackers took over a Jeep's control systems were replicated across hundreds of vehicles at a time.[70]

The widespread and growing adoption of the IoT, along with its inherent and largely unaddressed security issues, mean that it represents a threat that could soon dwarf that of the internet. Vint Cerf, one of the fathers of the Internet, was recently quoted about his views on the IoT. While acknowledging its potential benefits he also admitted, "Sometimes I'm terrified by it. It's a combination of appliances and software, and I'm always nervous about software — software has bugs."[71]

For the nascent opportunities in the IoT to be fully realized, the public must be convinced that what has been called "the greatest mass surveillance infrastructure ever conceived" will not share information gleaned from our devices unwittingly with "the original manufacturer, the information services we subscribe to, national security agencies, contractors, cloud computing services, and anyone else who has broken into, or been allowed into, the data stream".[72] This is especially important for DoD, given its large and growing exposure to the IoT and its special attractiveness as a target.

As a first step, the Department should develop a capability to detect large-scale intrusions on the IoT without having direct access to the individual devices. This is different from attempting to detect compromise of individual devices. Instead, it should focus on the characteristics and signatures associated with the remote activation of massive and/or coordinated intrusions of IoT devices. Developing this capability would require new algorithms and techniques to detect the changes in operating behaviors of large numbers of IoT devices as seen from various observation probes.

The feasibility of such a detection scheme is based on large-scale techniques already in use today by ISPs, multinational companies, and private threat intelligence companies to detect large botnets, worm outbreaks, and other major Internet events. A number of techniques, such as border gateway protocol (BGP) event monitoring, unused IP address space monitoring, passive domain name service (DNS) analysis, netflow changes, and information sharing clearing houses, all play a role in large-scale event detection on today's Internet. New approaches can build upon these proven techniques to help achieve the IoT system envisioned.

The scale and speed of IoT attacks will overwhelm human-in-the-loop defenses. Autonomous systems will be necessary to deal with the massive amount of data to be processed, as well as the speed necessary to defend and act within the difficult and diverse ecosystems of the IoT.

---

70 A. Greenberg, "Hackers Remotely Kill a Jeep on the Highway—With Me in It," *Wired*, [July 21, 2015]. Available at www.wired.com/2015/07/hackers-remotely-kill-jeep-highway (Accessed June 2016.)

71 K. Noyes, "Sometimes I'm terrified of the Internet of Things, says father of the Internet," *IoT Council: A Thinktank for the Internet of Things* [August 26, 2015]. Available at www.theinternetofthings.eu/katherine-noyes-'sometimes-im-terrified-internet-things-says-father-internet (Accessed June 2016.)

72 See, for example, J. Powles, "Internet of Things: the greatest mass surveillance infrastructure ever?" *The Guardian.* Available at www.theguardian.com/technology/2015/jul/15/internet-of-things-mass-surveillance (Accessed June 2016.)

Recommendation 24.

DARPA should develop autonomous systems that detect large-scale intrusions on the IoT, by passively and remotely monitoring bulk network traffic, and identifying aggregate indicators of compromise hidden within the flood of ordinary traffic. The program should include a series of competitions over a period of five years at an estimated cost of $80 million, which includes funds for a testbed, real and synthetic data sets, prizes, and block grants.

- This stretch problem is envisioned as a series of competitions to meet increasingly difficult intrusion challenges. Participants win prizes for outstanding performance as either intruders or detectors, with prizes for successful detection being ten times greater than for successful intrusion. This difference is because intrusion is inherently easier, and the prize differential will encourage innovation in detection approaches while still rewarding those who help elucidate new threat (intrusion) pathways.
- The periodic challenges would become progressively more difficult. For example, the first challenge may only require the participants to be able to detect a "noisy" and overt attack on the IoT test range using a large number of passive sensors. The next challenge may require participants to build upon the previous success and detect a "quieter" and stealthier intrusion using fewer sensors. Subsequent challenges would increase the complexity of the detection problem while reducing the quantity or quality of the sensing data relative to the size of the test range.
- To encourage a diverse range of participants, refreshed over time, the program should establish a regularly scheduled series of head-to-head competitions and, at the start of each new challenge, provide a round of block grants to support the development of the most promising approaches.[73] Block grants, rather than contracts, are recommended to maximize the likelihood of engaging commercial, start-up, and academic groups. In order to maximize innovation, each competition event should be open to any qualified participant regardless of whether they had received a grant that round or whether they had been involved in earlier competition cycles.
- An essential aspect of this challenge is a test range that simulates the scale and heterogeneity of full IoT. This range must be purpose built (perhaps built out from an existing cyber range) and include a large, diverse set of IoT devices within an instrumented network that can ingest, store, and process extremely large data sets generated by the devices and their network traffic. In addition, synthetic IoT data is likely to be required to enhance the realism of the test environment by increasing its scale virtually. This facility will be used to conduct the periodic competitions, and in between these events it should be made available to qualified competitors to allow them to test out their approaches under the most realistic conditions possible.

[73] This approach has been proposed in DARPA's Cyber Grand Challenge, a DARPA program planned to launch at DEFCON in August 2016.

Building autonomous cyber-resilient military vehicle systems

Imagine if commanders could trust that their platforms are resilient to cyber-attack through autonomous system integrity validation and recovery.

The vulnerability of networks to cyber-attacks is increasingly understood, and new methods are being developed to handle cyber-threats. Today the most robust solutions rely on large quantities of data collected from globally distributed, cooperative sensors, and advanced analytical methods carried out using high-performance computing. The best techniques not only carry out real-time cyber-defense, they also extract useful information about the attacks and generate signatures that help predict and defeat future attacks across the entire network. They are powerful, resource-intensive and reliant on high-bandwidth network access. Said differently, they exploit network resources to protect networks.

Autonomous and semi-autonomous vehicles, like networks today, also face cyber-threats and, like networks a decade ago, they often have limited defenses. For example, in 2011 the Predator and Reaper UA cockpits at Creech Air Force Base were infected with malware that proved very difficult to remove.[74] As the degree of autonomy increases in U.S. platforms, the cyber-vulnerability of subsystems will have increasing impact. Increased autonomy is inevitable—even today, humans cannot operate some high-performance vehicles without autonomous subsystems to maintain platform stability; the loss of these subsystems would force the vehicle to operate in a degraded state, if it could operate at all. Autonomous (unmanned) platforms lack a human operator to take over in the event of subsystem compromise. And further, fully or partially autonomous platforms may have to operate in communications-denied environments, limiting the value of defensive measures that are off-board or reliant on networking.

Thus, the network-centric cyber defenses that provide the best defense for networks are not well suited for providing cyber protection to autonomous platforms (or autonomous subsystems). Protecting autonomous platforms requires a different paradigm.

Rather than focusing on *robustness*, as is traditional in cyber defense of networks, the cyber-protection of autonomous (or semiautonomous) platforms should focus on *resilience*. Robustness seeks to ensure resistance to an attack, whereas resilience emphasizes rebound from attack and/or operating through the attack with as much mission performance as possible.[75]

An emphasis on resilience opens new options for protection approaches. Because a system under attack would not seek to fully understand or decisively defeat the attack, it could take a more limited approach to defending itself. It would need to detect the fact–and unfolding–of the attack, for example, by run-time integrity validation. It would need to recognize which subsystems were corrupted, and be able to autonomously determine the criticality of each affected system for the

74 N. Shachman, *Computer Virus Hits U.S. Drone Fleet* [October 7, 2011]. Available at www.wired.com/2011/10/virus-hits-drone-fleet (Accessed June 2016.)

75 D.D. Woods, "Four concepts for resilience and the implications for the future of resilience engineering," *Reliability Engineering & System Safety*, 141 [September 2015], pp. 5-9.

current mission set. It would need the capacity to restore essential subsystems from known good images, and to isolate or shut down non-essential systems as appropriate.

Some aspects of this type of system have already been demonstrated. For instance, Volexity advertises a run-time system that continuously validates the integrity of a computer's operating system via analytics carried out in random access memory (RAM). DARPA's Clean-Slate Design of Resilient, Adaptive, Secure Hosts (CRASH) is creating new computing architectures that focuses on system security, processing/memory segmentation, and resilience.

Further work remains in optimizing methods for hardware- and software-based integrity validation, autonomous assessment of subsystem compromise, and autonomous adaptation, including the restoration or shutdown of subsystems. It may be useful to develop so-called trusted "sidecar" modules that can easily integrate with various vehicle platforms under meaningful size, weight, and power constraints. These modules could execute out-of-band system-integrity assessments as well as host and restore the known good subsystem images. Such sidecars could also hold slight variations in subsystem images, to increase the likelihood of resistance to any specific attack. As well, a sidecar architecture could facilitate between-mission updates.

Autonomous systems, especially those unable to communicate with humans, require the ability to defend themselves autonomously. Even for autonomous subsystems that are components of larger systems with humans in the loop, the timescale required to respond to cyber-attack can be far too short to allow human involvement.

Recommendation 25.

DARPA should implement a stretch problem to demonstrate autonomous cyber-resilient systems (ACRS) for autonomous military vehicles. A competition should be run annually for six years at an estimated cost of $60 million.

- This stretch problem should be structured as a series of competitions to meet increasingly difficult cyber-attacks. Participants are awarded prizes for outstanding performance as either cyber-attackers or defenders, where defenders create resilient systems that enable autonomous vehicles to operate through the attack. Prizes for resilience are ten times greater than prizes for successful attacks. This difference is because cyber-attack is inherently easier than autonomous cyber defense, and the prize differential will encourage innovation in defensive approaches while still rewarding those who help elucidate new vulnerabilities.
- The annual competitions would be progressively more difficult. For example, the first competition may only require a candidate ACRS to detect and recover from an obvious disruption-style attack that is attempting to disable critical platform subsystems. Subsequent competitions might require candidate ACRS' to detect and recover from increasingly stealthy compromises that attempt to subvert system functions versus simply trying to disable subsystems. Later competitions could also increase the attack rates and sources to further validate the ACRS' ability to deal with a highly contested cyber-

environment, possibly with multiple adversaries using different attack techniques and approaches.

- To encourage a diverse range of participants that is refreshed over time, the program should create a regularly scheduled series of head-to-head competitions and, at the start of each new challenge, provide a round of block grants to support the development of the most promising approaches. (Block grants, rather than contracts, are recommended to maximize the likelihood of engaging commercial, start-up, and academic performers). In order to maximize innovation, each competition event should be open to any qualified participant regardless of whether they had received a grant that round or whether they had been involved in earlier competition cycles.
- Participants must be provided limited access to military vehicle operating systems and hardware architectures. The block-grant approach also provides an opportunity for screening the suitability of potential participants, independent of whether they receive government funding. In addition, the program requires a range for the autonomous operation of military ground and air vehicles, which is instrumented to capture ground truth of mission effectiveness. The test vehicles must be allowed to be subjected to cyber-attack. The range must also be suited to cyber-attack, including appropriate instrumentation to assess cyber-"health" of the platforms. Between competitions, the range should be made available periodically to qualified participants for ongoing development and testing.

Planning autonomous air operations

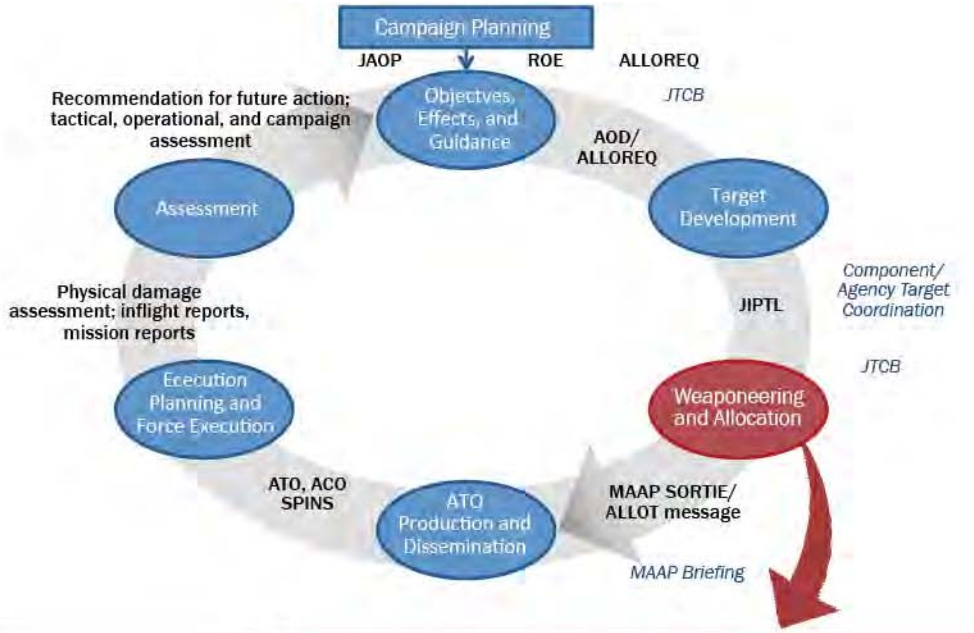
Imagine if commanders could operate inside adversary timelines by continuously planning and replanning tactical operations using autonomous ISR analysis, interpretation, option generation, and resource allocation.

The Joint Air Tasking Cycle is illustrated in Figure 20, which highlights the centrality of the Master Air Attack Plan (MAAP). Decisions made during the MAAP become the core of the daily Air Tasking Order (ATO). The MAAP process currently takes 12 hours and must be completed 24 hours prior to execution to allow time for subsequent ATO generation and dissemination to units for detailed mission execution planning.

The range of functions carried out in the MAAP generation process is shown in the lower right of Figure 20. Today, the process is heavily manual, with as many as 40-50 people required for generating the master plan for a large operation. Human planners do mission and resource planning, aided by stand-alone (non-integrated) models for individual platforms and effects.

The current timelines to complete the MAAP/ATO process are too long to effectively counter an adaptive adversary. In fact, there is ample evidence that the timeline has been too slow even for the relatively modest threats faces over the past several decades. [76] For example, by the end of Operation Enduring Freedom the time between target identification and target destruction had

[76] P. Winkler, *The Evolution of the Joint ATO Cycle, Joint Advanced Warfighting School, Joint Forces Staff College* [2006]. Available at www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA451239 (Accessed June 2016.)



Acronyms

ACO: airspace control order
ALLOREQ: allocation request
AOD: air operations directive
ATO: air tasking order
JAOP: joint air operation plan
JIPTL: joint integrated prioritized target list
JTCB: joint targeting coordination board
MAAP: master air attack plan
ALLOT: allotment
ROE: rules of engagement
SPINS: special instructions

MAAP Planning Function	Description
Force Allocation	Sortie flows, air-to-air refueling (AAR) missions and packages
Air-to-Air Plan	Missions and packages for Defensive Counter Air (DCA)
Electronic Warfare (EW)/ Suppression of Enemy Air Defenses (SEAD)	Combat air patrol to suppress enemy air defenses
Killbox Interdiction/ Close Air Support (KI/CAS)	Actions to interdict targets identified after mission begins
Force Application	Strike missions and packages for JIPTL (Joint Integrated Prioritized Target List) including Offensive Counter Air (OCA)/SEAD needed

Figure 19 The range of functions carried out by the MAAP team within the Joint Air Tasking Cycle is a manual, slow process.

shrunk to under 20 minutes. This acceleration was the basis for the formalization of the concept of kill box interdiction and close air support (KI/CAS) during Operation Iraqi Freedom, which enabled aircraft to be launched without preplanned targets in order to handle dynamic targeting. A subsequent analysis showed that almost 80 percent of the targets struck were of this class, *i.e.*, they were outside of the formal ATO process (although they did rely heavily on the MAAP to assign a geographic kill box to platforms).

Experience proves we have adapted our planning process well to defeat the regional powers and non-state threats of the past several decades. As we prepare, however, to meet a near-peer threat we must expect the planning problem to become more complex, with more platforms of different types (including a mix of manned and unmanned assets), potentially executing multiple missions or operating in tight collaboration with other platforms, all within a congested and contested electromagnetic spectrum. Planning complexity will increase, and at the same time, a near-peer can be expected to use tactics specifically designed to defeat a multi-day planning cycle such as we use today. A clever adversary will generate multiple plans, initiating one as a foil for the U.S. planning function and then switching to another one after it is too late for the U.S. to adapt. We must prepare to counter such tactics.

Autonomy could enable a much faster planning cycle through the use of integrated tools and models that can handle the allocation of large numbers of resources in complex sequences with high interdependencies. It will be essential in managing and optimizing the complexity of branching scenarios associated with what-if analyses and pre-planning for contingencies. And, during execution, it enables a running comparison of "plan" to "actual," to assess whether large-scale re-planning is merited as events unfold. In this sense, planning is no longer a phase that happens before execution. Instead it becomes a continuous, background process that assists the commander in redirecting assets when needed during execution.

The tools to shorten the MAAP/ATO process by an order of magnitude do not yet exist. However, there is evidence of enabling progress in many underlying capabilities. For example, DARPA's Distributed Battle Management program is developing automated decision aids for managing air-to-air and air-to-ground combat.

Autonomy is required to handle the scale and complexity of the planning at speed. The study notes that a better planning tool on its own is not sufficient to shorten the planning timeline. That will also require changes in workflow, staffing, and other factors.

Recommendation 26.

The AFRL/RIS office should undertake a stretch problem to generate a new MAAP/ATO within one hour of target development. The cost is estimated at $25 million per year, with tests every 18 to 24 months until the objective is achieved.

- This stretch problem differs from some others in that it is tied to a specific operational process. Thus, its form appears most like a standard program—but with some essential features to drive innovation from the commercial world. One critical element is the necessity for an open architecture, to enable participation by a variety of performers. An open architecture also supports the development of modular mission capabilities, so that the mission-planning scope can be expanded over the course of the program. The program manager should spend the first year of execution defining the open architecture to be used within the program.
- Another essential feature is a program structured around a series of milestones that assign increasingly challenging time goals for ATO generation, *e.g.*, 12 hours, six hours, one hour. The program manager should apportion an expanding scope of mission capabilities to each milestone. Once the architecture is defined, the government should open a competition to fund at least two competing systems integrators, each leading an innovative team of subcontractors that bring diverse and leading technology expertise. Periodically (*e.g.*, every 18 months) the sponsor should run a head-to-head competition of performance against the current milestone goals. The winning team should be assigned a prize, possibly in the form of an award fee. Then each prime should be allowed to reselect a new team of subcontractors, based on performance to date and emerging technologies.
- The program should carry out the assessments at interim milestones using a combination of live, virtual, and constructive facilities. The graduation exercise should be carried out at Red

Flag, to enforce performance assessment in a realistic combat environment that mimics the fully complexity of modern operations—fighter interdiction, attack/strike, air superiority, enemy air defense suppression, airlift, air refueling, and reconnaissance missions.

Summary

The study concluded that autonomy will deliver substantial operational value—in multiple dimensions—across an increasingly broad spectrum of DoD missions, but the DoD must move more rapidly to realize this value. Allies and adversaries alike also have access to increasingly rapid technological advances occurring globally, which are driven by commercial market forces stemming from a diverse array of commercial markets. While difficult to quantify, the study concluded that autonomy, fueled by advances in artificial intelligence, has attained a "tipping point" in value, and that autonomous capabilities are increasingly ubiquitous.

Over-arching themes that emerged from the study included:

- The need to build trust in autonomous systems while also improving the trustworthiness of autonomous capabilities
- The need to accelerate adoption of autonomous capabilities through DoD enterprise-wide enablers
- The need to strengthen the operational pull for autonomy by demonstrating operational value across a broad range of missions
- The need to expand the technology envelope to help the U.S. sustain military advantage through the increasing use of autonomy

While DoD is already embracing the value of autonomous capabilities, in both fielded systems and developmental programs, it has not yet adapted its enterprise processes to effectively support the rapid and widespread adoption warranted by the potential benefits—and made imperative by the potential perils of autonomy in the hands of adversaries. The study therefore concluded that action on the enterprise-level recommendations is of far greater importance—and urgency—than the implementation of any single program. These interdependent recommendations focus on the enablers needed to accelerate adoption of autonomous capabilities.

An important objective of this study was to identify opportunities for DoD to more rapidly exploit ongoing technological advances. By selecting several demonstrations of autonomous systems with near-term benefits, the study intends to illustrate the operational value across a diverse array of missions, thereby strengthening the operational pull for autonomous capabilities. It should be noted, however, that the full value of such demonstration programs will be realized only if they are conducted in concert with—and used to refine and mature—the recommendations focusing on the enterprise enablers.

The study also observed that DoD has research efforts underway that will, over time, expand its envelope of technological options. The study's recommendations, which focus on a set of stretch problems, are intended as a supplement—not a replacement—for such ongoing research. While focusing primarily on expanding the technology envelope, execution of the stretch problems as designed yields broader benefits. By engaging a broad array of non-DoD providers, together with the diverse spectrum of DoD stakeholders, the stretch problems are intended to foster relationships that not only accelerate innovation but also accelerate DoD's ability to exploit that innovation.

At its core, autonomy is about decision-making. The working definition used during this study was "autonomy results from delegation of a decision to an entity that is authorized to take action within specific boundaries." As used in this report, the "entity" to which decision authority is delegated is a software algorithm. A key benefit is that the use of autonomy can increase decision speed—enabling the U.S. to act inside an adversary's operations cycle.

But speed is equally important in a second dimension—rapid transition of autonomy into warfighting capabilities in order for the U.S. to sustain military advantage. And this dimension requires a DoD enterprise that is both ready and eager to realize the benefits of autonomy across its entire mission set.

Table 4 summarizes the recommendations in the report. Details for implementation of each recommendation are found throughout the report on the pages listed in the table.

Table 4: Summary of Recommendations

No.		Page
	Accelerating Adoption of Autonomous Capabilities	
1	USD(AT&L) should require that best practices be developed and applied to all software dominated systems and, in particular, autonomous systems.	28
2	USD(AT&L) should address the special issues associated with cyber resiliency in autonomous systems.	30
3	DOT&E in conjunction with DT&E should establish a new T&E paradigm for testing software that learns and adapts.	34
4	The DoD test and evaluation community should establish a new paradigm for T&E of autonomous systems that encompasses the entire system lifecycle.	34
5	USD(AT&L) should require the acquisition community to establish and implement a consistent and comprehensive M&S strategy throughout the lifecycle of the system.	37
6	Military Service Chiefs should integrate technology insertion, doctrine, and CONOPs by ensuring early experimentation that uses alternative sources and informs employment doctrine.	38
7	USD(P&R), working with USD(AT&L) and Military Service Chiefs, should develop an autonomy-literate workforce.	40
8	ASD(R&E) should improve global autonomy technology discovery by encouraging personnel exchanges and coordinating partner organization efforts in FFRDCs, UARCS, and the IC.	41
9	The Deputy Secretary of Defense should establish departmental governance of autonomy by creating an EXCOM for oversight and establishing advocates in the Military Services.	43
10	USD(AT&L), USD(P), and ASD(PA) should take a proactive, two-pronged approach to anticipate cultural objections to the use of autonomy.	43

11	The Deputy Secretary of Defense should take immediate action to counter adversary autonomy.	46
	Strengthen Operational Pull for Autonomy	
12	NSA, in partnership with DARPA and IARPA, should fully develop the means to tip and cue DISA and the defense industrial base to defend the DoD information infrastructure, extending to U.S. government and private sector support as appropriate.	52
13	DARPA, working with AFRL and the 711th Human Performance Wing, should initiate a new program to adapt existing ISR data screening and fusion tools, such as the Air Force's Dynamic Time Critical Warfighting Capability (DTCWC) or PCPAD-X, or DARPA's Insight for autonomous, real-time use.	54
14	DIA and USSOCOM should integrate commercial components and build a new machine-learning analysis tool, and prototype the resulting system using existing historical data, seized media, and commercial (collateral) sources.	55
15	CERDEC, AFRL, and SPAWAR should develop Military Service prototypes for local, agile spectrum deconfliction and control among a few systems; concurrently DARPA should develop an architectural framework and algorithms for near-real time, theater-level spectrum deconfliction and control for a full ensemble of joint, coalition systems.	58
16	The Navy PEO for Littoral Combat Ships should conduct a user operational evaluation system program run by PEO-LCS in partnership with ONR.	60
17	USCYBERCOM should take the lead to develop an automated cyber-response, in partnership with CIA, FBI, NSA, DARPA (Plan X), key cyber-security industry players, and DISA.	63
18	U.S. Navy and DARPA should collaborate to conduct an experiment in which assets are deployed to create a minefield of autonomous lethal UUVs.	66
19	The U.S. Marine Corps, DARPA, ONR Code 30, and an FFRDC or UARC develop and experiment with a prototype heterogeneous, autonomous UAS support team that includes ten or more UA.	70
20	NAVSUP should demonstrate the use of modern intelligent adaptive planning in conjunction with SAP.	77
21	CASCOM, Ft Lee 10th Mountain Division, and the Ft Drum Joint Readiness Training Center should develop and deploy adaptive logistics decision support for a relocatable robotic warehouse and trained personnel in preparation for rapid deployment to unstable regions.	78
	Expand Technology Envelope for Autonomous Systems	
22	DARPA should initiate a stretch problem designed to create a system that autonomously, globally, and in real-time identifies the causal linkages behind emerging social movements, and helps leaders understand the impact of possible courses of action along various possible future event trajectories.	84
23	ASA(ALT), with close participation by ARCIC, should establish an annual "swarm games" challenge.	90
24	DARPA should develop autonomous systems that detect large-scale intrusions on the IoT, by passively and remotely monitoring bulk network traffic, and identifying aggregate indicators of compromise hidden within the flood of ordinary traffic.	94

25	DARPA should implement a stretch problem to demonstrate autonomous cyber-resilient systems (ACRS) for autonomous military vehicles.	96
26	The AFRL/RIS office should undertake a stretch problem to generate a new MAAP/ATO within one hour of target development.	99

Terms of Reference

THE UNDER SECRETARY OF DEFENSE
3010 DEFENSE PENTAGON
WASHINGTON, DC 20301-3010

ACQUISITION, TECHNOLOGY AND LOGISTICS

NOV 17 2014

MEMORANDUM FOR CHAIRMAN, DEFENSE SCIENCE BOARD

SUBJECT: Terms of Reference – Defense Science Board 2015 Summer Study on Autonomy

The technology of autonomy is rapidly advancing and finding widespread private sector and public sector application. Relevant capabilities span the spectrum from autonomy, i.e. the brains, to autonomous systems (e.g. robots, drones, etc.) which integrate autonomy into physical systems. Applications include IBM's Watson, the use of robotics and automation in ports and mines, autonomous vehicles (from UAVs to Google's self-driving car), automated logistics and supply chain management, and many more.

The purpose of this study is to identify the science, engineering, and policy problems that must be solved to permit greater operational use of autonomy across all warfighting domains. The study will assess opportunities for DoD to enhance mission efficiency, shrink life-cycle costs, and reduce loss of life through the use of autonomy. Emphasis will be given to exploration of the bounds—both technological and social—that limit the use of autonomy across a wide range of military operations. The study will ask questions such as: What activities cannot today be performed autonomously? When is human intervention required? What limits the use of autonomy? How might we overcome those limits and expand the use of autonomy in the near term as well as over the next 2 decades?

Applications to be considered include decision aids, planning systems, logistics, surveillance, and war-fighting capabilities. The study will also identify cost-imposing opportunities such as the use of autonomy to spoof adversaries, creating confusion and consuming their resources; and will also consider potential threats stemming from the use of autonomy by adversaries.

The study will examine the international landscape, identifying key players (both commercial and government), relevant applications, and investment trends. Considerations will include "baked-in" security, scalability, and variable cooperation between autonomous algorithms/systems and humans.

The study will consider opportunities such as: the use of large numbers of simple, low-cost (i.e. "disposable") objects vs. small numbers of complex (multi-functional) objects; use of "downloadable" functionality (e.g. apps) to repurpose basic platforms; and an ability to vary the degree of autonomy vs. human supervision/control for specific missions rather than developing mission-specific platforms.

The study will deliver a plan that identifies the barriers to increased operational use of autonomy and ways to reduce or eliminate those barriers. The study report should include: recommendations to reduce or eliminate the barrier, an assessment of risks to successful implementation of the recommendation, and an estimate of resources required to implement the recommendation.

I will sponsor the study. Dr. Ruth A. David and Dr. Paul D. Nielsen will serve as Co-chairmen of the study. Dr. Jonathan Bornstein, US Army Research Laboratory, will serve as Executive Secretary. Lt Col Michael Harvey, USAF, will serve as the DSB Secretariat Representative.

The study will operate in accordance with the provisions of P.L. 92-463, the "Federal Advisory Committee Act" and DoD Directive 5105.04, the DoD Federal Advisory Committee Management Program." It is not anticipated that this study will need to go into any "particular matters" within the meaning of title 18, United States Code, section 208, nor will it cause any member to be placed in the position of action as a procurement official.

Frank Kendall

Members of the Study

Study Chairs

Dr. Ruth David	Private Consultant
Maj Gen Paul Nielsen, USAF (Ret.)	Software Engineering Institute

Executive Secretaries

Dr. Jonathan Bornstein	U.S. Army Research Lab
Lt Col Scott Heritsch	Joint Staff J8

Members

Mr. James R. Allard	Amazon
Dr. Amy E. Alving	Private Consultant
Dr. Michael R. Anastasio	Los Alamos National Laboratory
Dr. Brent Appleby	Draper Laboratory
Dr. Wanda M. Austin	Aerospace Corporation
Mr. Michael Bayer	Private Consultant
Dr. Jeffrey M. Bradshaw	Institute for Human and Machine Cognition
Mr. Frank J. Cappuccio	Private Consultant
Mr. James F. Carlini	Private Consultant
Gen Michael P. C. Carns,USAF (Ret.)	Private Consultant
Dr. Arup K. Chakraborty	Massachusetts Institute of Technology
Dr. Mark Chevillet	Applied Physics Laboratory
Hon. David S.C. Chu	Institute for Defense Analyses
Ms. VIctoria Coleman	Technicolor
Mr. Christopher W. Day	Packet Forensics
Dr. Eric D. Evans	MIT Lincoln Laboratory
ADM William J. Fallon, US Navy (Ret.)	Countertack, Inc.
Dr. Craig Fields	Private Consultant
CAPT James R. Gosler, US Navy (Ret.)	Applied Physics Laboratory
Mr. Alfred Grasso	MITRE Corporation
Dr. S.K. Gupta	University of Maryland
CAPT Mark R. Hagerott, US Navy (Ret.)	U.S. Naval Academy
Mr. Paul J. (Page) Hoeper	Onpoint Technologies, Inc.
Dr. Adele Howe	Colorado State University
Brig Gen Chris Inglis, USAF (Ret.)	U.S. Naval Academy
Dr. Miriam E. John	Lawrence Livermore National Laboratory
Hon. Anita K. Jones	University of Virginia
Hon. Paul G. Kaminski	Technovation, Inc.
Dr. Ronald L. Kerber	Private Consultant
Gen Paul Kern, US Army (Ret.)	Cohen Group

Dr. Amy A. Kruse	Cubic Global Defense, Inc.
Hon. Zachary J. Lemnios	IBM
Mr. Ashley J. Llorens	Applied Physics Laboratory
Dr. Robert Mandelbaum	Lockheed
Dr. John L. Manferdelli	Google
Dr. Joseph Markowitz	Private Consultant
Dr. Mark T. Maybury	MITRE
Gen James P. McCarthy, USAF (Ret.)	U.S. Air Force Academy
Hon. Judith Miller	Private Consultant
Hon. James N. Miller, Jr.	Applied Physics Laboratory
Dr. Robin R. Murphy	Texas A&M University
Mr. Robert F. Nesbit	Private Consultant
Mr. David H. Scheidt	Applied Physics Laboratory
Hon. William Schneider, Jr.	International Planning Services, Inc.
Dr. Ralph D. Semmel	Applied Physics Laboratory
Dr. Ross D. Shachter	Stanford University
Mr. James D. Shields	Private Consultant
Mr. Robert M. Stein	Private Consultant
VADM Edward M. Straw, US Navy (Ret.)	Private Consultant
Dr. James A. Tegnelia	University of New Mexico
Dr. Anthony J. Tether	Sequoia Group
Mr. David M. Van Buren	L-3
Mr. Vincent Vitto	Private Consultant
Mr. Lewis Von Thaer	DynCorp International
Dr. David A. Whelan	Boeing
Dr. Robert L. Wisnieff	IBM
Dr. Greg L. Zacharias	U.S. Air Force

Defense Science Board

Mr. David Jakubek	Executive Director
LtCol Michael Harvey	Deputy for Operations, US Air Force
Mr. Robert Ramsay	
Mr. Tom Simms	

Government Advisers

Mr. Maynard Holliday	ODUSD(AT&L)
Dr. Greg Hudas	U.S. Army
Mr. Scott Littlefield	Tactical Technologies Office, DARPA
Dr. James Overholt	Air Force Research Laboratory
Dr. Brian Sadler	Army Research Laboratory
Dr. Frederick E. (Fritz) Schultz	OASD(R&E)
Mr. Donald Sofge	Naval Research Laboratory

Dr. Marc Steinberg	Office of Naval Research

Staff

Ms. Rosemary Battles	Strategic Analysis, Inc.
Ms. Amy Cauffman	Strategic Analysis, Inc.
Ms. Erin Erickson	Strategic Analysis, Inc.
Ms. Meghan Fitch	Strategic Analysis, Inc.
Ms. Hannah Freeman	Strategic Analysis, Inc.
Ms. Ashlee Gilligan	Strategic Analysis, Inc.
Ms. Sarena Harvey	Strategic Analysis, Inc.
Mr. Marcus Hawkins	Strategic Analysis, Inc.
Mr. Leland Lambert	Strategic Analysis, Inc.
Dr. Toni Maréchaux	Strategic Analysis, Inc.
Ms. Diane O'Neill	Strategic Analysis, Inc.
Ms. Jen Schimmenti	Strategic Analysis, Inc.
Ms. Jeray Simms	Strategic Analysis, Inc.
Ms. Stephanie Simonich	Strategic Analysis, Inc.
Ms. Melissa Smittle	Strategic Analysis, Inc.
Mr. Ted Stump	Strategic Analysis, Inc.

Briefers to the Study

February 25–26, 2015

Dr. Lynn Parker	National Science Foundation
CPT Matt Pregmon	National Defense University
Mr. Paul Scharre	Center for a New American Security
Dr. Brad Tousley	Tactical Technology Office, DARPA
Dr. Richard Voyles	White House Office of Science and Technology

March 17–18, 2015

Dr. Paul Cohen	Information Innovation Office, DARPA
Mr. Jim Geurts	U.S. Special Forces
Mr. Bob Work	Deputy Secretary of Defense
Mr. Eric Sundberg	Aerospace Corporation
LTC Matt Dooley	U.S. Army Capabilities Integration Center
Gen Mike Hostage	USAF Air Combat Command (retired)
Mr. Frank Kendall	Under Secretary of Defense (AT&L)

April 29–30, 2015

Dr. Guru Banavar	IBM
Dr. Steve Chien	NASA Jet Propulsion Laboratory
Dr. George Ka'iliwai	U.S. Pacific Command

May 18–19, 2015

Mr. Jean-Charles Ledé	Tactical Technologies Office, DARPA

June 16–17, 2015

Mr. Carl Johnson	Northrop Grumman Electronic Systems
Dr. Gill Pratt	Defense Sciences Office, DARPA
Mr. Stephen Randich	FINRA
Dr. Peter W. Singer	New America Foundation

July 28–29, 2015

Mr. Michael Gilmore	DoD Operational Test and Evaluation
Mr. Jehezkel Grizim	Israel Aerospace Industries
Mr. Oskar Levander	Rolls Royce Marine
LTG H.R. McMaster	U.S. Army Capabilities Integration Center
Mr. Brian Pierce	Tactical Technologies Office, DARPA

THE AIM INITIATIVE

A STRATEGY FOR AUGMENTING INTELLIGENCE USING MACHINES

Contents

This page intentionally left blank.

THE AIM INITIATIVE

Augmenting Intelligence using Machines Increasing insight
and knowledge through Artificial Intelligence, Automation, and Augmentation

Foreword

FROM THE DIRECTOR OF NATIONAL INTELLIGENCE:

Closing the gap between decisions and data collection is a top priority for the Intelligence Community (IC). The pace at which data are generated and collected is increasing exponentially—and the IC workforce available to analyze and interpret this all-source, cross-domain data is not. Leveraging artificial intelligence, automation, and augmentation technologies to amplify the effectiveness of our workforce will advance mission capability and enhance the IC's ability to provide needed data interpretation to decision makers. The Augmenting Intelligence using Machines (AIM) Strategy provides the framework for the incorporation of AIM technologies to accelerate mission capability development across the IC. I challenge the IC workforce, based on the principles outlined in the AIM Strategy, to establish and implement an IC-wide AIM framework, inclusive of mission partners—be big but be practical—to provide real capability to close the gap between decisions being made and data collection.

Dan Coats
Director of National Intelligence

FROM THE PRINCIPAL DEPUTY DIRECTOR OF NATIONAL INTELLIGENCE:

To meet its vision of ensuring intelligence advantage, the IC must adapt to the rapid global technological democratization in sensing, communications, computing, and machine analysis of data. These trends threaten to erode what were previously unique USIC capabilities and advantages; going forward, we must improve our ability to analyze and draw conclusions from IC-wide data collections at scale. I have identified AIM technologies as key transformative elements that will enable our analytic workforce to effectively leverage the increasing data volume for decision advantage. This document provides the overarching strategy and objectives for effective incorporation of AIM into the IC baseline. I welcome your feedback on this document.

Susan Gordon
Principal Deputy Director of National Intelligence

Executive Summary

It is the job of the IC to analyze data, connect disparate data sets, apply context to data, infer meaning from data, and ultimately make analytic judgments based on all available data. The pace at which data are generated, whether by collection or publically available information (PAI), is increasing exponentially and long ago exceeded our collective ability to understand it or to find the most relevant data with which to make analytic judgments. AIM AAA technologies (Artificial intelligence, process Automation, and IC officer Augmentation) as key transformative elements are crucial for future mission success and efficiency.

This document outlines how the IC will incorporate AIM capabilities in a manner that resolves key IC legal, policy, cultural, technical, and structural challenges while producing optimally effective analytic and operational contributions to the intelligence mission.

Artificial intelligence (AI), especially its sub-discipline machine learning (ML), has shown dramatic advances in autonomous systems, computer vision, natural language processing, and game playing. These AI systems can perform tasks significantly beyond what was possible only recently (e.g., autonomous systems) and in some cases even beyond what humans can achieve (e.g., chess and Go). In light of these recent advances, the IC is carefully considering methods for fully automating well-defined processes and augmenting human expertise with analytics or planning capabilities for their potential benefit. The IC is also monitoring these same technologies with respect to their vulnerabilities in development and adoption. Accordingly, AIM seeks to determine how the IC can best manage uncertainty by achieving acceptable risk suited to the demonstrable analytic and operational advantages in AIM-enabled solutions and tradecraft.

Due to the widespread commercial application of these AI technologies, the private sector is making considerable investments in related infrastructure and people. Therefore, we must carefully monitor and leverage private investment, focus our efforts on areas of unique mission need, and rethink how we attract and retain human expertise. This strategic imperative exists because our adversaries, notably Russia and China, also recognize the potential for AI to transform military and intelligence operations and are investing aggressively to make that advantage a reality.

Individual components of the IC have already recognized the value of AAA technologies. It is the goal of the AIM initiative to bring those disparate efforts together in order to maximize impact and accelerate development. Increases in data volume and velocity are putting pressure on existing workflows, and our adversaries are putting significant effort into AI technologies that can blind or deceive the IC. By adopting AIM, the IC will be able to meet those challenges. This initiative leverages lessons learned from current and past AI efforts; strengthens the collaboration between the IC and industry, research agencies, and academic talent; and grows the talent pool of expertise for the IC. We will continue to expand our interagency approach to AIM development to ensure that the implementation plan we deliver is the IC's plan as opposed to the Office of the Director of National Intelligence's (ODNI) AIM plan for the IC.

The AIM initiative will enable the IC to fundamentally change the way it produces intelligence. We will achieve superiority by adopting the best available commercial AI applications and combining them with IC-unique algorithms and data holdings to augment the reasoning capabilities of our analysts. Simply stated, our goal is the following:

> **"If it is knowable, and it is important, then we know it."** – Sue Gordon

The AIM initiative is an IC-wide strategy for three reasons:

- First, there is intense competition in the private sector for AI and especially ML talent. The IC needs to establish new incentive and hiring models and stop competing internally for the same scarce resources.
- Second, AI and ML systems require large high-quality tagged data sets that must be shared with IC partners to the maximum extent allowable. Rule sets, algorithms, and expert knowledge bases that capture the tacit knowledge of intelligence domain experts must be available to all appropriate and relevant mission areas.
- Third, to rapidly accelerate AI adoption, the IC must have a solid digital foundation. This means leveraging the investment we have already made in the IC Information Technology Enterprise (IC ITE) and continuing to invest in and improve the IC ITE infrastructure.

The AIM initiative has four primary investment objectives:

Objective 1 – Immediate and ongoing – Digital Foundation, Data, and Science and Technical Intelligence (S&TI): AI activities are not a substitute for an enduring, secure, standardized, and measurable IC-wide digital infrastructure and data ecosystem; they are dependent on that foundation. In addition, the IC must improve foundational understanding of many aspects of AAA, to include a deeper understanding of the commercial supply chain, identification of ongoing developmental programs within the federal government that can be leveraged for a wider audience, and identification of adversarial uses of AI.

Objective 2 – Short term – Adopt Commercial and Open Source Narrow AI Solutions: The IC must leverage the existing private sector and government investments by rapidly transitioning the best available commercial and open source Narrow AI capabilities.

Objective 3 – Medium term – Invest in the Gaps (AI Assurance and Multimodal AI): To create and maintain strategic advantage, the IC must develop both the capability and capacity to take advantage of available data across all INTs and open source, and develop AI solutions that process and relate information from multiple modalities. To facilitate this, the IC must continue to implement policies to break down traditional INT stovepipes.

Objective 4 – Long term – Invest in Basic Research Focused on Sense-Making: It is not enough to simply fuse information from multiple modalities together in response to a single, narrow task. The construction of shared models is needed to provide the basis for trust between human and machine teams. This level of understanding demands basic research advances in representing knowledge; goals and intent; entity extraction from incomplete, multimodal data; and discourse generation.

Inclusive of all four objectives, it is critical for the IC to address issues of AI assurance, transparency, and reliability as well as potential adversarial uses of AI. The AIM initiative must include a continuous effort to both understand how AI algorithms may fail.

The AIM initiative is about much more than technology. Implementing the strategy will entail addressing workforce challenges and understanding and shaping the policies and authorities governing how the IC deploys and uses AI. The global nature of the challenge and the rapidly evolving technological and societal frameworks dictate that the IC must have strong partnerships with other government agencies, the private sector, foreign partners, national laboratories, Federally Funded Research and Development Centers (FFRDC), University Affiliated Research Centers (UARC), and academic institutions. Lastly, the AIM initiative includes a robust communication and outreach plan for the workforce, Congress, members of the Executive Branch, industry and foreign partners, and the American people.

This page intentionally left blank.

UNCLASSIFIED

Mission Imperative

The business of the IC, both in its raw material and its product, is intelligence, which comes from data. It is the job of the IC to analyze, connect, apply context, infer meaning, and ultimately make analytic and operational judgments based on all available data. The pace at which data are generated is increasing exponentially and is stressing our collective abilities. Some examples:

- By 2021, it is estimated that the data generated by global web traffic will reach 3.3ZB/year (up from 1.2ZB/year in 2016); this corresponds to 3.5 networked devices per global capita.[1]
- The Director of NGA has publically estimated that at the current, accelerating pace of collection, we would need over 8 million imagery analysts by 2037 to process all imagery data.[2]

One particular area of concern for the IC is related to AI mission assurance, especially in light of recent commercial efforts that utilize AI to generate high-quality, affordable forgeries of audio and video media. This could lead to widespread difficulties separating truth from fiction. Adding to this challenge is the problem that AI expertise is scarce, distributed around the world, and very limited in the IC.

Overview

In addition to the vision and guiding principles, this document provides guidance on:

- Investments
- Partnering with industry, academia, research agencies, and national laboratories
- Creating a new policy framework and tradecraft expectation that enable AI and ML while simultaneously promoting safe use and mitigating risk
- Reshaping the IC workforce
- Engagement with the Department of Defense (DoD) and international partners
- A strategy for creating a classified activity to generate strategic advantage for years to come
- Research and development
- Governance and IC collaboration models
- A robust communications strategy for all of our constituents including Congress, the work force, our industry and international partners, and the American people
- Establishing consistent classification/declassification processes through AI that promote secure information sharing and facilitate appropriate transparency to the public

1 "The Zettabyte Era: Trends and Analysis", June 2017. Cisco Report. https://www.cisco.com/c/en/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html
2 Remarks from Director Robert Cardillo at 31st Annual Small Sats Conference, Utah, 8/7/2017. http://www.nga.mil/MediaRoom/speechesremarks/Pages/Small-Satellites/Small-Satellites---Big-Data.aspx

Vision

The AIM initiative seeks to secure and maintain a strategic competitive information advantage for the IC through focused development and rapid adoption of AAA technologies.

The leading private sector companies, both in market capitalization and prospects for growth, all recognize the importance of digital infrastructure and have made massive and ongoing investments in related technologies such as cloud services and big data. Since each new generation of technology builds on the previous one, it is critical that the IC continue to invest in its digital foundations. This initiative will guide the IC to accelerate the adoption of digital and analytics transformation, identify mission use cases, build a coherent data ecosystem, acquire the appropriate AI tools, reshape the workforce, adapt new workflow processes, and change the culture.

The IC can and must do this. Our IC ITE investments in cloud technology and data services have paved the road to harness the power of unique data collections and insights to provide decision advantage at machine speed.

Guiding Principles

The following guiding principles define the set of unwavering precepts that influence and guide the direction of the AIM strategy to facilitate cultural, political, and legal adoption across the IC.

The opportunity is great; the threat is real; the approach must be bold: Recognizing that strategic advantage is fleeting and fragile, the IC must be willing to rethink or abandon processes and mechanisms designed for an earlier era, establish disciplined engineering and operations practices, and maintain an absolute focus on assuring advantage in an intensely competitive global adversarial environment.

ML models are IC assets: Building on the IC ITE principle that "Data is an IC asset," machine learned models are also IC assets (as opposed to agency or INT-specific assets).

- Training and validation data sets: Most ML methods require large, high-quality, tagged data sets. These data sets are important IC assets and must be shared with IC partners to the maximum extent allowable.
- Rule sets, algorithms, and expert knowledge bases that capture the tacit knowledge of intelligence domain experts are also IC assets that must be shared with all appropriate and relevant mission areas.
- This community approach must also recognize and act on the need for INT-specific improvements, as they will be the main drivers for transformative capabilities in the near term.
- While improving INT-specific technologies, the community approach must take into account the need for correlated cross-INT data sets.
- Even when the actual training data cannot be shared, sharing the models derived from these training sets, along with the lessons learned from them, increases the value of these assets.

AI can be a powerful tool, but we must recognize challenges:

- ML classifiers are only as good as the data that is used to train them. For example, an image classifier that is trained with ground-based imagery may fail to classify images collected from overhead.
- Even state-of-the-art AI models are vulnerable to adversarial exploitation.
- AI and ML models are subject to "concept drift," i.e., the notion that in the real world data often arrives in streams and evolves over time in non-obvious ways. Therefore, the models must continually adapt to changes in the data environment so that opportunities to improve their accuracy are not missed.
- The IC should be aware of popular trends in AI but should stay focused on how we can best use the technology. When the media hype dies down, the IC must be ready to perform the long-term and difficult task of creating lasting operational value from these technologies.
- Many ML models do not include a description of their decision-making process in their standard output, and thus their results can be misunderstood by the casual user.

AI assurance models and adversarial AI must be addressed in parallel with AI systems: The level of effort to fool an AI algorithm is considerably lower than to develop them. Therefore:

- Intelligence systems must account for failure modes. For example, image classifiers may be fooled by very small changes in the input data, reinforcing the need for recurring human involvement in AI activities.[3]
- The IC must understand and anticipate how foreign entities may use AI and develop techniques and tactics to deny and disrupt those activities.
- The IC must develop intelligence systems that can demonstrate the underlying rationale behind decisions and responses to both users and overseers. For intelligence systems that make critical decisions regarding classification, dissemination, or life-critical decisions, such decisions and responses must be able to evince some degree of proof of correctness in addition to transparency.
- Recent developments in computer vision have resulted in approaches that can generate fake (altered or fabricated) images and audio recordings that are difficult to distinguish from unaltered digital media. The IC needs to develop ways of countering this capability.

AI is not a substitute for developing a solid digital foundation; it requires that foundation:

- The IC must continue to invest in and improve the IC ITE infrastructure and develop strategies for shared state-of-the-art hardware and/or other High Performance Computing (HPC) systems.
- The IC must accelerate activities that make data widely shareable. Need-to-know requirements and operational sensitivities will be honored but must not be used as an excuse to unnecessarily limit data sharing.
- The IC must create a sustained program of investment in the creation of high-quality training sets for the most important intelligence priorities.

3 Su, J., Vargas, D., Sakurai, K., "One Pixel Attack for Fooling Deep Neural Networks", arXiv 1710.08864, Feb. 2018

Despite the perceived investment gap, the IC has opportunities:

- U.S. Government (USG) investment in AI is dwarfed by investment of the private sector, and the IC investment is a fraction of what Department of Defense (DoD) is investing.
- The IC must not only leverage the investment of the DoD and private sector, but we must also be prepared to invest in areas of unique interest to the IC.
- The IC should invest in areas critical to the IC mission where the private sector has few incentives to invest, such as low-shot learning and adversarial AI/AI assurance.

Common services are a priority, however, there is still a need for specialization:

- The IC must create common services for common capabilities in computer vision, human language technology, identity intelligence, process modeling, analytic discovery, automated planning, and other areas, while encouraging principled approaches to mission-specific specialization where appropriate.

Investments in the workforce must be made: The IC must develop a more technologically sophisticated and enterprise aware workforce. We must:

- Embrace strategic workforce planning and workforce analytics to address AAA workforce requirements and skill gaps.
- Invest in programs for training and retooling the existing workforce in skills essential to working in an AI-augmented environment.
- Redefine recruitment, compensation, and retention strategies to attract talent with high-demand skills.
- Develop and continually expand partnership programs with industry, including internship and externship programs, to increase the number of cleared individuals with relevant skills both in and out of government.
- Leverage the IC Joint Duty (JD) Program more strategically to share expertise across the IC in a seamless manner.
- Understand and maximize human capital authorities, policies, and programs to augment the AAA workforce.

Engagement with partners is essential:

- A successful AI strategy requires engagement USG-wide, with the private sector, educational institutions, FFRDCs, national laboratories, and international partners (particularly Five Eye [FVEY] Partners).

Maintaining an understanding of the foreign threat is an intelligence priority:

- The IC must place an emphasis on S&TI integrated with operations and focused on AI in order to maintain strategic advantage, effectively counter these threats, and develop appropriate intelligence policies.

Investment Strategy

Worldwide private sector investment in AI, ML, and related technologies is growing rapidly. Estimates of global private sector investment in 2016 range from $26B to $39B (McKinsey).[4] This investment strategy acknowledges the significant private sector investment and prioritizes investments that 1) allow the IC to rapidly adopt the best commercial and open source capabilities, and 2) accelerate research in those areas unique to the IC and where the private sector is not currently focused. A successful investment strategy also recognizes we must maintain momentum on foundational infrastructure gains, such as completing the IC's HPC architecture as well as accelerate data conditioning, storage, and sharing activities. This four-part investment plan, illustrated in Figure 1, addresses each aspect of basic research, applied R&D, and development and adoption.

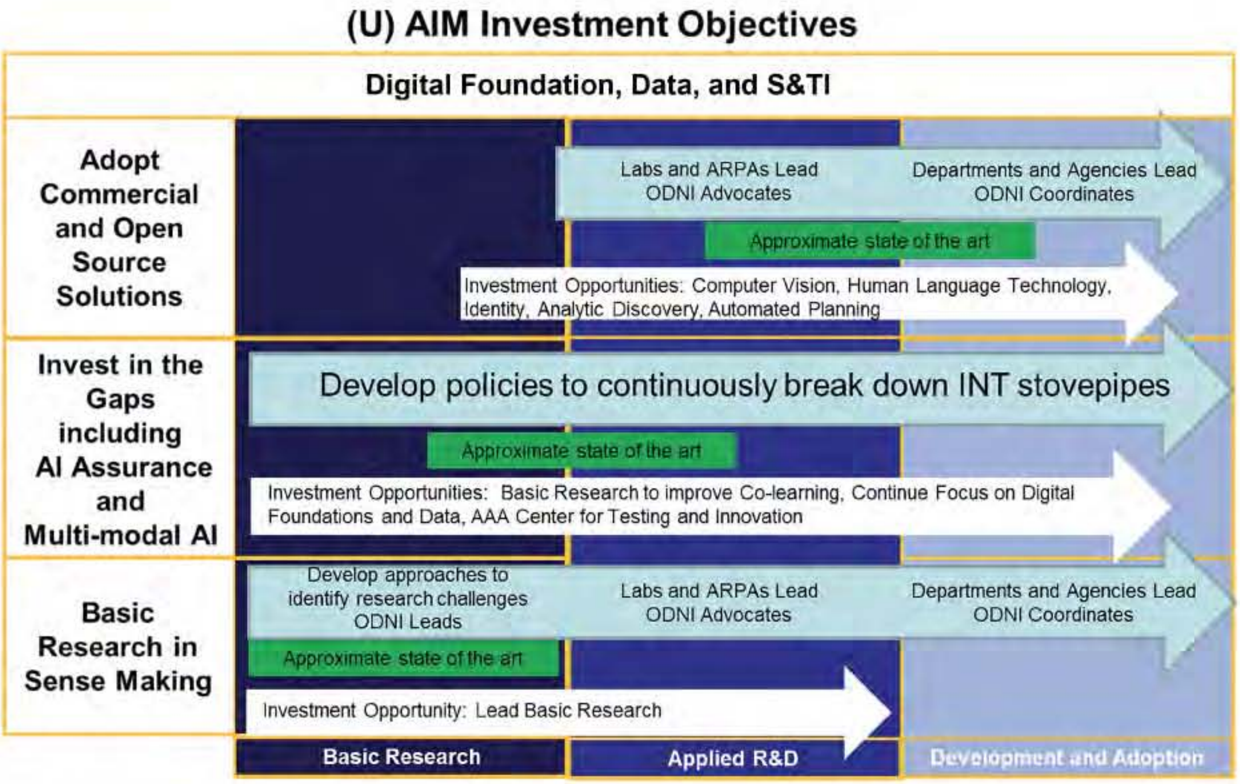


Figure 1: AIM Investment Strategy

4 "Artificial Intelligence – The Next Digital Frontier", June 2017, McKinsey Global Institute

Objective 1 – Immediate and ongoing – Digital Foundation, Data, and S&TI: AI activities are not a substitute for an enduring, secure, standardized, and measurable IC-wide digital infrastructure and data ecosystem. The IC will:

- Make data accessible to a wide variety of analytic platforms and models.
- Establish and maintain relevant training data across all INTs and disciplines.
- Adapt policies and tradecraft to enable more automated methods of assembling and vetting training data.
- Seek to future-proof data. Establish standards for data labeling and metrics for evaluation.
- Undertake a program of continuous growth in computational resources to ensure sufficient numbers of current generation hardware are available to IC AI practitioners.
- Improve foundational S&TI for AI, including adversarial uses of AI.
- Research co-learning and AI assurance models, especially vulnerabilities and standards.

Objective 2 – Short term – Adopt Commercial and Open Source AAA Solutions: The IC must leverage the massive private sector investment by rapidly transitioning the best available commercial and open source AAA capabilities. This will be accomplished as follows:

- **Aggressively pursue shovel-ready opportunities across IC Agencies**
- **Establish an IC AIM Center:** To foster innovation and rapidly prototype transformative solutions, the IC will establish an AIM Center staffed with AI and ML talent from across the IC, augmented by experts from industry.
- **Collaborate with key partners to identify opportunities:** Strengthen partnerships with the Intelligence Advanced Research Projects Activity (IARPA), the Defense Advanced Research Projects Agency, In-Q-Tel, the national laboratories, Defense Innovation Unit-Experimental, and industry. Advocate for those activities that address gaps with a minimum amount of duplicative effort, which will facilitate rapid transition of appropriate AAA capabilities to operations.

Objective 3 – Medium term – AI Assurance and Multimodal AI: In order to create and maintain strategic advantage, the IC must develop AI solutions that process and relate information from multiple modalities. To facilitate this, the IC must continue to implement policies that break down traditional INT stovepipes.

Objective 4 – Long term – Invest in Basic Research Focused on Sense-Making: The IC must understand multimodal information in context and look for ways that substantially augment the activities of IC Officers.

The massive investments to date as well as those currently planned (both public and private) will not be sufficient to meet the unique, foundational, and all-source sense-making needs of the IC. Therefore, basic research will focus on those areas and facilitate collaboration across the public and private sectors.

Policy and Authorities

As part of the AIM strategy, the IC will examine the current tradecraft landscape and address emerging policy issues with appropriate efforts internal to the USG and, where needed, international venues. Policies to codify AIM activities (e.g., acquisition, enterprise management, classification, and analytic integrity) will be developed with consultation from appropriate general counsel, civil liberties, privacy, and policy personnel. ODNI will provide a dedicated, integrated policy and legal effort to break down barriers to information sharing, particularly INT-specific data, so that we do not inadvertently slow the pace of technological progress.

Workforce Strategy

The IC must develop a more technologically sophisticated and enterprise-aware workforce. We must:

Embrace strategic workforce planning and analytics: Workforce planning will aid in accurately identifying current and future skill gaps, and will also enhance the IC's ability to determine the most appropriate mitigation strategies (e.g., training, compensation, etc.).

Invest in programs for training and equipping the workforce in essential AI skills: This does not mean everyone in the workforce needs to become an expert in deep learning or Python coding, but everyone does need to understand how AIM fits into the new workflow and how they can contribute. Specific actions include:

- Leadership – must understand the implications on the intelligence process, have a sophisticated understanding of the threat environment and foster an environment that enables an open and collaborative culture while reskilling the workforce to operate in an AI accelerated environment.

Build on the IC 2025 workforce transformation to attract talent with high-demand AI skills: The role descriptions for people with these skills have gone by many different terms in recent years. Therefore, individuals with these skills may be available but under different keywords. These alternative terms include analytics, data science, data wrangling, statistics, ML, deep learning, and modeling. These cover both the researchers who propose and test new methods, as well as model builders who use these algorithms to create and validate models.

Develop partnership programs with industry and academia to increase the pool of people inside the IC with awareness of best practices and available tools in this fast moving area, and to encourage individuals outside of the IC to build capabilities that meet the needs of the IC. Specific actions include:

- Recruit talent before graduation, and before competition with industry salaries, through service-for-education agreements ("ROTC"), expansion of IC postdocs, and internship/externship programs.
- Support temporary non-government to government (internship, externship) rotations.
- Expand sabbaticals, part-time industry Intergovernmental Personnel Act positions, and consultancies that grant clearances to faculty to increase the available technical skill available to the IC.

- Investigate changes in policy or funding to improve retention and attraction of U.S. national and foreign-born graduates in technical fields, including staff roles that do not require a clearance, and "fast-track" hiring that allows experts to perform productive work before obtaining a clearance.
- Expand use of open challenge problems (e.g., IARPA) and develop data and proxy problems that focus external communities on IC regions of interest.
- Identify unclassified equivalent domains for researchers to pursue. While the IC represents a unique environment, often similar domains give uncleared researchers an opportunity to develop and test algorithms on data that has many of the same qualities as IC data. This also fosters an interest in public service.

Leverage the IC JD Program: As competition for talent continues to increase outside of the IC, the community must leverage the IC Civilian JD program to share and retain talent across the IC and provide the workforce opportunities in other IC missions. We should:

- Identify related positions in each agency that will benefit from the JD program.
- Track JD opportunities for professionals and the AI community's use of the JD program.
- Ensure that the return on investment of personnel participating in the JD program meets AI objectives and is sustainable through the sense-making investment stage.

Understand and maximize human capital authorities and policies to augment the AI workforce: IC elements and the DNI have certain authorities at their disposal to assist in the management of the IC employment lifecycle. In order to ensure the most effective use of these authorities, we should:

- Identify and implement authorities that will create efficiencies in recruiting, hiring, compensation, training, and retention of AI professionals.
- Ensure that human capital policies enable IC elements to support the employment of AI personnel and do not erect barriers that may disengage the AI workforce.

Leverage current human capital programs and monitor implementation and user feedback: AI managers must continually collaborate with human capital professionals to take advantage of programs that enable the IC workforce to meet mission objectives. Examples include:

- Scholarships and other educational financial aid (e.g., Stokes Scholarships).
- Well-rounded recruiting programs that include outreach to diverse schools (e.g., Adopt-A-School, IC Wounded Warrior Program, STEM Outreach).
- IC Heritage Community Liaison Council, which is a forum that supports IC workforce development objectives, including outreach and recruitment.
- Recruiting efforts such as the IC Virtual Career Fair and IC Centers for Academic Excellence.

Industry Partnership Strategy

Since the bulk of the nation's AI resources reside in the private sector, partnership is essential to the IC. Yet the barriers working with government often require considerable effort to clear. This requires a more flexible acquisition paradigm. This includes cooperative agreements that may trade data for algorithms or "Analysis-as-a-Service," as well as public prize challenges to solve IC problems. With the bulk of development occurring outside the IC, we must collectively prioritize Certification and Accreditation of new software so that code can more quickly be deployed on secure networks. ODNI, in collaboration with the IC elements, will develop an industry partnership plan for AIM capabilities. Elements of the plan will include:

- Industry access to USG data for algorithm development
- Enabling human resource strategies to simplify the development and sharing of AI skills between government and industry to include new approaches to security
- ODNI advocacy for AI basic research funding
- Creating AI services of common concern or specific capability contracts
- Update intelligence and industry data- and capability-sharing policies and oversight

Roles for USG Agencies, National Labs, FFRDC, UARC, Commercial and Academic Institutions

To capitalize on the combined capabilities of the USG, national laboratories, private industry, and academic institutions, the ODNI must facilitate partner integration. Therefore, partner roles include:

IC:

- Promote communications between AIM partners
- Promote development of shared analytic services where feasible
- Share datasets and computing
- Capture and share expert knowledge from IC systems and analytics
- Capture and share mission data for future training datasets and simulations
- Develop defensive and offensive techniques for adversarial/counter AI
- Coordinate R&D activities
- Modernize multi-agency data sharing practices
- Improve S&TI on foreign AIM capabilities and intentions

Whole-of-USG:

- Share datasets across labs, private industry, and academic institutions
- Acquire and retain experts on immigration policy, IPAs, or service-for-education agreements ("ROTC")
- Coordinate DoD and IC R&D, computing and data purchases, and data-labeling efforts
- Synchronize funding for basic and applied research efforts

National Labs/FFRDC/UARC

- Provide expert advisors to USG
- Verify and validate algorithms and data sets, testing and evaluation (T&E), and AI methodology
- Conduct mission-focused research
- Develop AIM-related algorithm and systems prototyping
- Support talent pipeline development

Industry

- Provide commercial tools accessible through USG acquisitions and/or investment
- Conduct mission-focused, AIM-related research and development
- Provide expert advisors to USG
- Appropriately share datasets through a legal, supportable business model

Academic Institutions

- Perform the research needed to develop long-term scientific breakthroughs
- Provide expert advisors to USG
- Train the next generation to be a highly skilled workforce equipped to develop AAA tools and develop skills to utilize AAA systems

Five Eye Foreign Partner Engagement

Allied and partner nations can enhance our joint development of intelligence products. Expanding international partnerships will provide opportunities to increase collection access and reliability, improve the quality and quantity of partner data and analysis, align strategic capabilities and emerging technologies, and promote compatibility across digital architectures and analytic tradecraft.

AI Assurance – Secure and Maintain Competitive Advantage

The unique data and tools that the IC creates using those data are important IC assets that provide competitive advantage for USG missions. That advantage is fleeting and fragile, requiring disciplined engineering and operations practices, and an absolute focus on assuring advantage in an intensely competitive global adversarial environment. Commercial and USG needs differ in important ways but largely overlap with the concomitant requirement for continuous investment in data, tradecraft, tools, T&E, security, and S&TI.

AI technologies have clearly demonstrated that they can provide powerful capabilities. They have also demonstrated their brittleness and vulnerabilities. There are some principles and best practices that can be used today.

- **Data:** ML systems are only as good as the data used to train them. Acquiring those data in volume from the intended operational environments is a critical advantage. These data must be continuously monitored and reacquired as necessary. This is an engineering tradecraft best practice, akin to standard software test suite discipline.
- **Software:** The leading edge AI/ML software suites were written to support science, not national security operations. There is no notion of cyber security. USG needs are not aligned exactly with those of industry and universities; we need to differentiate in the state-of-the-evolving-art tools in a robust, sustainable way.
- **Systems:** Continuous evaluation of performance is required. There is very little theory to inform us as to when ML systems fail, or even whether they will work as expected[5]. This situation is not acceptable for any safety-critical or national security system. We must always incorporate performance monitoring, and we should support theory development.
- **Test and Evaluation:** Too many AI/ML projects launch without metrics to allow the IC to understand whether the investment is on track to succeed or fail. Create the discipline to define metrics up front and establish rigorous testing regimes and schedules.

Concept Drift must be addressed. "Concept Drift" is the idea that all computer tools are built with specific assumptions about the real world and that the basis for these assumptions generally changes over time, requiring the tools be monitored and updated. Best practices in established disciplines such as control systems theory can help structure how this challenge is attacked; we must detect issues and— when possible—automatically correct.

Adversarial AI techniques represent opportunities and risks. We have highly sophisticated adversaries with access to the same tools, their own data, and experts trained in the same universities as our own people. AI is merely one of the new battlegrounds for a technology-based arms race.

S&TI is a priority. We must develop a better understanding of foreign adversary tactics, techniques, and procedures.

5 Papernot, N., McDaniel, P., Jha, S., Fredriskson, M., Celik, Z., Swami, A., "The Limitations of Deep Learning in Adversarial Settings", IEEE European Symposium on Security and Privacy, IEEE 2016, Saarbrucken, Germany

Understanding when AAA techniques fail is critical. The technical literature is replete with examples of how to deceive AAA systems.[6] We need to know how and where adversarial systems are in use against our assets.

Outreach / Communications Strategy

A key factor in the success of transformation efforts like the AIM initiative comes through awareness and education of all of the varied constituents of the enterprise. Therefore, the ODNI will establish and maintain a robust communications engagement strategy for each of the following audiences:

- The IC, including leadership and the workforce
- The DoD and other government agencies
- Congress and the White House
- The private sector
- The national laboratories and academia
- The American people

Governance

Following the example of private sector firms that are successfully implementing AI and recognizing that strong executive leadership goes hand in hand with stronger AI adoption, the PDDNI will, along with the IC Deputy Executive Committee (DEXCOM), serve as the executive sponsors for the strategy.

Conclusion

AIM technologies will have a transformative effect on how the IC operates. Increases in data volumes and velocity require the IC to dramatically rethink how we perform our mission. Additionally, our adversaries have recognized the importance of AIM methods and are putting significant effort into these technologies. The principles and strategies laid out here will allow us to meet those challenges. Most notably, those strategies will build on and leverage lessons learned from current and successful AIM efforts; strengthen the collaboration between the IC and industry, research agencies, and academic talent; and grow the talent pool of AIM technology expertise for the IC. Our goal in all of this is to meet our IC objective now and into the future. "If it is knowable, and it is important, then we know it." – Sue Gordon

6 Goodfellow, Ian, "Attacking Machine Learning with Adversarial Examples", https://blog.openai.com/adversarial-example-research/

Appendix A: Background on AI

Artificial Intelligence (AI): The IC defines AI as "the branch of computer science focused on programming machines to perform tasks that replicate or augment aspects of human cognition," a term coined in the 1950s.[7] At that time, scientists began to harness nascent computer capabilities to perform advanced information manipulations much more rapidly. In particular, it was realized that computers could be used not only to perform calculations on numbers, but also to perform inference on other types of information such as symbols, data, and text. This popularized the idea of a "thinking machine" that could, if filled with all the right knowledge and rules for access and retrieval, simulate a human response.[8]

Technologies and research areas generally considered to be sub-domains to AI:

- Automated Planning and Scheduling
- Computer Vision
- Decision Support, Predictive Analytics, and Analytic Discovery
- Distributed Artificial Intelligence/Agent-based Systems
- Human Language Technologies
- Identity Intelligence
- ML
- Process Modeling
- Robotics/Autonomous Systems

Ideal AI System: A machine capable of ideal human intelligence with a computer's speed, capacity, and precision.[9]

Adversarial AI: A subset of AI focused on understanding how AI systems behave in the presence of a malicious adversary.

Artificial Narrow Intelligence (ANI): Also known as "Narrow AI" or "weak" AI, this is an AI system that is specialized for a single purpose and cannot be generalized. All current applications are ANIs.

Artificial General Intelligence (AGI): Also known as "General AI" or "strong" AI, this is an AI system that can handle any human intellectual task—memory, learning, abstraction, and creativity. There are no AGI systems in existence, although building an AGI has been the goal of the field since it was founded in the 1950s.

The AIM INITIATIVE—Augmenting Intelligence using Machines

Narrow AI and Multimodal AI: Nearly all current commercial applications of AI are narrow solutions in that they solve a single problem with a single kind of data. Image classification, face recognition, and human language translation are all examples of narrow AI solutions. The IC must bring together

7 National Intelligence Council Sense of the Community Memorandum SOCM 2016-039C, 24 June 2016
8 Fiscal Years 2019-2030 Major Issue Study Final Report, Advanced Analytics, Deep Learning, and Artificial Intelligence.
9 Yost, Kirk, "Threats Posed by Advances in Artificial Intelligence", MITRE Technical Report sponsored by OSD Office of Net Assessment, 2016

data from multiple INTs to provide context and meaning to analysts over a variety of different data. Multimodal AI presents a whole new group of challenges in a number of areas that the IC must overcome. The challenges include:

- Representation - Presenting and summarizing multimodal data in a way that exploits its complementarity and redundancy. For example, development of representations that allow simultaneous analysis of audio derived from SIGINT with imagery and video.
- Translation – Learning how to translate or map one mode to another while recognizing that the relationship between modalities is often subjective. For example, there are any number of correct ways to describe an image with words, but a perfect translation from image to text may not exist.
- Alignment – Understanding how to identify direct relationships between elements and sub-elements to derive meaning from multiple modalities. For example, aligning a verbal description of an event with sequences in a video requires measuring similarity between modalities and understanding long-range dependencies and ambiguity.
- Fusion – Understanding how to join information from multiple modalities, which may have different predictive power and noise characteristics. For example, in audio-visual speech recognition, the visual description of the lip motion is fused with the speech signal to predict spoken words.
- Co-learning – Exploring how knowledge gained learning from one modality can help computational models trained on a different modality.

Automated Planning: A branch of AI focused on generating strategies or action sequences necessary to achieve a goal.

Automation: Computational systems designed to perform repetitive tasks.

Autonomous Systems: Systems that carry out tasks without human intervention. In AIM we are especially focused on computational systems that perform complex reasoning tasks.

Catastrophic Forgetting: A learning problem which occurs when performance learned in earlier tasks in a series is entirely or mostly lost after being given examples of later tasks.

Co-learning: A sub area of machine learning focused on either understanding how multiple agents can simultaneously learn, or how a single agent can use learning from one modality to improve computational models trained on a different modality.

Computer Vision: A field of study that aims to analyze, extract, and understand objects and relationships from within single or multiple images.

Concept Drift: The notion in ML that the concept being learned will change over time. Differentiating between different types of malware, for example, is a classification task that changes as new malware is produced.

Deep Learning: "Representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned." (LeCun, Y., Bengio, Y., and Hinton, G., "Deep Learning", Nature, Vol 521, 2015.)

Graphical Processing Unit (GPU): Specialized electronics designed to perform rapid mathematical functions to render images, animations, and videos.

Human or Intelligence Augmentation: Use of information technology to augment human intelligence in the performance of some task. Unlike autonomous systems which aim to replace human activity, augmentation is designed with humans as central.

Knowledge Discovery: A process of discovering useful knowledge from a collection of data.

Low-shot Learning: An object recognition, ML classification task where learning must take place despite having only one, or a few, example images for training.

Machine Learning: The field of study interested in building computational systems that can improve their own performance of some task.

Machine Learning Classifier: A ML model designed to assign given examples into known discrete categories (i.e., classification).

Machine Learning Model: An explicit summary of data which is useful for performing some task. The product of ML systems like decision tree algorithms or neural networks are generically known as models.

Multimodal AI: A subset of AI focused on methods that emphasize the integration of linguistic, acoustic, and visual data in the completion of some task.

Natural Language Processing: A field of study that aims to analyze and understand human language communications both spoken and textual. Can include analysis and generation of language.

Sense-making: A process of creating understanding in situations of high complexity.

Technical Debt: Complications accumulated during the construction and use of software or ML models that make maintenance of these models difficult (e.g., hidden feedback sources, undeclared consumers, data dependencies, and changes in the external world).

Testing data: A collection of examples used to evaluate the performance of a ML Model.

Training data: A collection of examples used to inform the construction of a ML model.

Appendix B: Acronym List

- AAA Artificial Intelligence, Automation, and Augmentation
- AGI Artificial General Intelligence
- AI Artificial Intelligence
- AIM Augmenting Intelligence using Machines
- ANI Artificial Narrow Intelligence
- ASIC Application-Specific Integrated Circuit
- CIA Central Intelligence Agency
- DEXCOM Deputy Executive Committee
- DIA Defense Intelligence Agency
- DNI Director of National Intelligence
- DoD Department of Defense
- FBI Federal Bureau of Investigation
- FFRDC Federally Funded Research and Development Center
- FVEY Five Eye
- GPU Graphical Processing Unit
- HPC High Performance Computing
- IAA Identity, Authentication, and Authorization
- IARPA Advanced Research Projects Activity
- IC Intelligence Community
- IC ITE IC Information Technology Enterprise
- ICAIP Intelligence Community AAA Implementation Plan
- JD Joint Duty
- ML Machine Learning
- NGA National Geospatial-Intelligence Agency
- NRO National Reconnaissance Office
- NSA National Security Agency
- ODNI Office of the Director of National Intelligence

- PAI Publically Available Information
- PDDNI Principal Deputy Director of National Intelligence
- R&D Research and Development
- S&TI Science and Technical Intelligence
- SEI Software Engineering Institute
- STEM Science Technology Engineering Math
- T&E Testing and Evaluation
- UARC University Affiliated Research Center
- USG US Government
- ZB Zettabyte

Acknowledgements

This document was created by the AIM Implementation Team with representatives from:

- ODNI
- CIA
- DIA
- FBI
- NGA
- NRO
- NSA
- USD(I)

Software Is Never Done

Refactoring the Acquisition Code for Competitive Advantage

Defense Innovation Board

May 3, 2019

Cover design by the Office of Secretary of Defense, Graphics and Presentations Division

Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage

Defense Innovation Board, 3 May 2019

J. Michael McQuade and Richard M. Murray (co-chairs)
Gilman Louie, Milo Medin, Jennifer Pahlka, Trae' Stephens

5 April 2019

I am pleased to forward the final report of the Software Acquisition and Practices (SWAP) study conducted by the Defense Innovation Board (DIB). The study, co-chaired by Dr. Richard Murray and Dr. Michael McQuade, was executed pursuant to Section 872 of the 2018 National Defense Authorization Act (NDAA). It makes 10 primary recommendations and 16 additional recommendations to address the most critical statutory, regulatory, and cultural hurdles DoD faces in modernizing its approach to software. In a year replete with reports on artificial intelligence, quantum computing, and blockchain, it may seem mundane to write about software, but software is the foundation of all things digital, and the Department does it exceedingly poorly. It is scarcely possible to imagine how the Department can achieve the modernization objectives of the National Defense Strategy without overhauling its approach to software, which is what motivated us to take on this task.

I have questioned the usefulness of the myriad studies and reports on technology authorized by Congress and DoD over the years. These mandates are often executed in a vacuum and the recommendations rarely seen through to implementation. Consequently, we resolved early on to conduct the SWAP study differently, with emphasis on pragmatism, accessibility, and candor. The iterative and inclusive approach the SWAP study team opted to take gave Congress, industry partners, and Department stakeholders at all levels unprecedented opportunity to participate in and contribute to the vision for the future of DoD software. Most importantly, each one of these communities provided critical feedback to ensure the recommendations contained in this report are timely, actionable, and contextualized by the ultimate mission -- swiftly delivering capability to the warfighter. Fittingly, it seems the recipe for delivering value in reports is the same as in modern software development: user feedback. Ultimately, we produced a report that changed my mind about the usefulness of such reports, and I hope you agree.

I hope the draft implementation plans at Appendix A are particularly valuable; we intended to provide sufficient detail to make them immediately implementable, but not so much as to be inflexible. They should provide the Department and Congress a starting point from which to act on the recommendations and instill critical accountability across the organization and in industry.

I recommend that this report be distributed to the offices within OSD and the Services that deal with software technology and acquisition directly, and to the appropriate industries and organizations that support them, as well as to numerous organizations that have seen their mission transformed by software, however reluctant they may appear to be to acknowledge it. With regard to industry and private sector partners, we deliberately wrote for the defense industrial base as well as for those companies in the national security innovation base that may not identify themselves as defense suppliers, but whose support will be essential for the Department's future technological relevance. Pay mind to the report's first theme: software *truly* is ubiquitous; and a modern acquisition and development approach should be architected at the enterprise-level with broad support and understanding from the "users."

Finally, I fully support the DIB's further involvement, at the discretion of relevant Department leadership, as advisors in the implementation of the recommendations contained in this report. We consider it our privilege to have engaged with such talented and devoted uniformed personnel, civilians, and contractors over the course of the SWAP study and we welcome a continued dialogue about how to make the ideas contained in this report a reality.

Sincerely,

Dr. Eric Schmidt
Chairman
Defense Innovation Board

DIB | DEFENSE INNOVATION BOARD

MEMORANDUM TO THE CHAIRMAN, DEFENSE INNOVATION BOARD

SUBJECT: Final Report of the Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) Study

Attached is the final report from the DIB's SWAP study, documenting our efforts, analysis and conclusions working with Congress and DoD on how to develop, procure, assure, deploy, and continuously improve software for use in the Department. In developing this report over the past eighteen months, we have had substantial conversations with congressional staffers, DoD leadership at many levels, program offices, contractors, (government and private sector) software developers, and a variety of other representatives from government, industry, academia, and the public. There is broad consensus on the goal of delivering high quality software to DoD users in a manner that is timely, secure, and cost effective. Our study details the external and self-inflicted barriers DoD faces in implementing modern software practices and lays out steps to address current gaps. We hope that our recommendations will serve as a basis for the implementation phase of this work. We are happy and ready to support that effort.

We would like to thank the study members, Gilman Louie, Milo Medin, Jennifer Pahlka and Trae' Stephens for their contributions to the report. Bess Dopkeen served as the initial study director and established an outstanding structure for the study and the support staff. Jeff Boleng supported the study throughout its initial phase and stepped in as study director after Bess. We would also like to acknowledge the outstanding help provided by Courtney Barno, Kevin Garrison, Nick Guertin, Devon Hardy, Sandra O'Dea, Forrest Shull, and Craig Ulsh. A longer list of the many people who have helped on the study is included in the Acknowledgements section of the main report and in Appendix J.

J. Michael McQuade

Richard M. Murray

Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage

Defense Innovation Board, 3 May 2019

J. Michael McQuade and Richard M. Murray (co-chairs)
Gilman Louie, Milo Medin, Jennifer Pahlka, Trae' Stephens

Extended Abstract

U.S. national security increasingly relies on software to execute missions, integrate and collaborate with allies, and manage the defense enterprise. The ability to develop, procure, assure, deploy, and continuously improve software is thus central to national defense. At the same time, the threats that the United States faces are changing at an ever-increasing pace, and the Department of Defense's (DoD's) ability to adapt and respond is now determined by its ability to develop and deploy software to the field rapidly. The current approach to software development is broken and is a leading source of risk to DoD: it takes too long, is too expensive, and exposes warfighters to unacceptable risk by delaying their access to tools they need to ensure mission success. Instead, software should enable a more effective joint force, strengthen our ability to work with allies, and improve the business processes of the DoD enterprise.

Countless past studies have recognized the deficiencies in software acquisition and practices within DoD, but little seems to be changing. Rather than simply reprint the 1987 Defense Science Board (DSB) study on military software that pretty much said it all, the Defense Innovation Board's (DIB's) congressionally mandated study[1] on Software Acquisition and Practices (SWAP) has taken a different approach. By engaging Congress, DoD, Federally Funded Research and Development Centers (FFRDCs), contractors, and the public in an active and iterative conversation about how DoD can take advantage of the strength of the U.S. commercial software ecosystem, we hope to move past the myriad reports and recommendations that have so far resulted in little progress. Past experience suggests we should not anticipate that this report will miraculously result in solutions to every obstacle we have found, but we hope that the two-year conversation around it will provide the impetus for figuring out how to make the changes for which everyone is clamoring.

In this report, we emphasize three fundamental themes:

1. **Speed and cycle time are the most important metrics for managing software**. To maintain advantage, DoD needs to procure, deploy, and update software that works for its users at the speed of mission need, executing more quickly than our adversaries. Statutes, regulations, and cultural norms that get in the way of deploying software to the field quickly weaken our national security and expose our nation to risk.
2. **Software is made by people and for people, so digital talent matters**. DoD's current personnel processes and culture will not allow its military and civilian software capabilities to grow nearly fast or deep enough to meet its mission needs. New mechanisms are needed for attracting, educating, retaining, and promoting digital talent and for supporting the workforce to follow modern practices, including developing software hand in hand with users.

[1] 2018 National Defense Authorization Act (NDAA), Sec. 872. Defense Innovation Board analysis of software acquisition regulations.

3. **Software is different than hardware (and not all software is the same)**. Hardware can be developed, procured, and maintained in a linear fashion. Software is an enduring capability that must be supported and continuously improved throughout its life cycle. DoD must streamline its acquisition process and transform its culture to enable effective delivery and oversight of multiple types of software-enabled systems, at scale, and at the speed of relevance.

To take advantage of the power of software, we advocate four main lines of effort:

A. **Congress and DoD should refactor statutes, regulations, and processes for software**, enabling rapid deployment and continuous improvement of software to the field and providing increased insight to reduce the risk of slow, costly, and overgrown programs.

B. **The Office of the Secretary of Defense (OSD) and the Services should create and maintain cross-program/cross-Service digital infrastructure** that enables rapid deployment, scaling, testing, and optimization of software as an enduring capability; manage them using modern development methods; and eliminate the existing hardware-centric regulations and other barriers.

C. **The Services and OSD will need to create new paths for digital talent (especially *internal* talent)** by establishing software development as a high-visibility, high-priority career track and increasing the level of understanding of modern software within the acquisition workforce.

D. **DoD and industry must change the practice of how software is procured and developed** by adopting modern software development approaches, prioritizing speed as the critical metric, ensuring cybersecurity is an integrated element of the entire software life cycle, and purchasing existing commercial software whenever possible.

Report structure. The main report provides an assessment of the current and desired states for software acquisition and practices, as well as a review of previous reports and an assessment of why little has changed in the way DoD acquires software, with emphasis on three fundamental themes. The report's recommendations are broken into four lines of effort, with a set of primary recommendations provided for each (bold), along with additional recommendations that can provide further improvements. Each recommendation is accompanied by a draft implementation plan and potential legislative language.

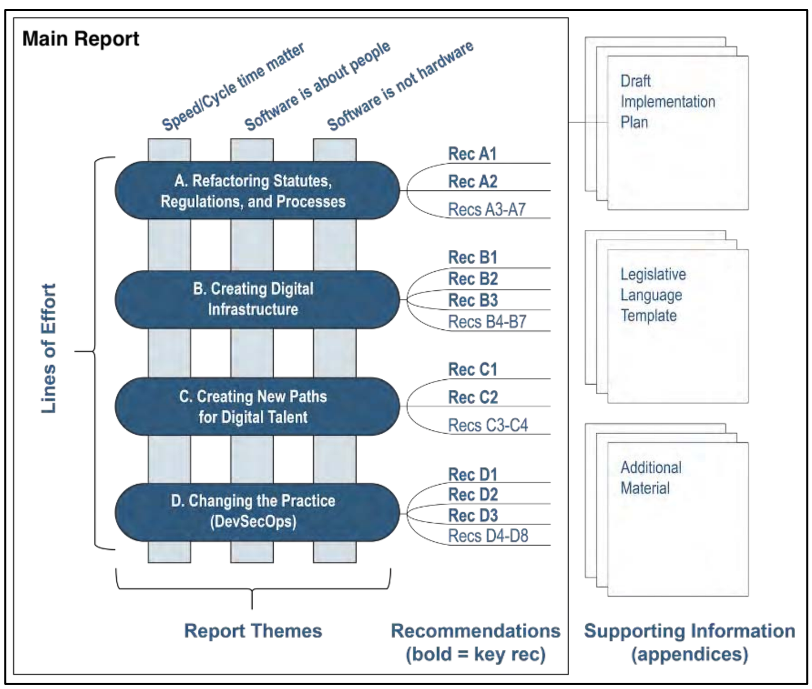


Table of Contents

Supporting Information

Chapter 0. README (Executive Summary)

In 2011, Marc Andreessen claimed in an op-ed for *The Wall Street Journal* that "Software Is Eating the World."[2] He argued that *every* industry (not just those considered to be "information technology") would be transformed by software—bytes rather than atoms. Eight years later, it is clear he was right.

This transformation is happening in defense, and we are not prepared for it. Software is leveling the playing field with our rivals, eroding the advantages we have spent many decades accruing. Software is the focal point of many important advances in national security technology, including data analytics, artificial intelligence (AI), machine learning (ML), and autonomy. Software is ubiquitous. It is part of everything the Department of Defense (DoD) does, from logistics to management to weapon systems. U.S. national security is critically dependent on the capabilities of DoD's software.

DoD must be able to develop, procure, assure, deploy, and continuously improve software faster than our adversaries. Unfortunately, DoD still treats software much like hardware, and often misunderstands the relationship between speed and security. As a result, a large amount of DoD's software takes too long, costs too much, and is too brittle to be competitive in the long run. If DoD does not take steps to modernize its software acquisition and development practices, we will no longer have the best military in the world, no matter how much we invest or how talented and dedicated our armed forces may be.

The good news is that there are organizations within DoD that have already acknowledged the risks of falling further behind in software and are leveraging more modern acquisition and development practices with notable success. The Defense Digital Service (DDS), the Defense Innovation Unit (DIU), the Joint Improvised-Threat Defeat Organization (JIDO), and the Air Force's Kessel Run are examples that demonstrate that DoD has the ability to ship world-class software. The challenge remains doing this at scale.

DoD needs to build on these foundations to create an ecosystem and standard operating procedures that enable the practices of great software without requiring employees to "hack the system." To do that, we must address the prioritization, planning, and acquisition processes and policies that create the worst bottlenecks for deploying capability to the field at the speed of relevance. Further, we must address all the practices that not only put the U.S. Armed Forces at risk and reduce the efficiency of DoD's operations, but also drive away the very people who are most needed to develop this critical capability.

Our adversaries are already doing this. China actively leverages its private industry to develop national security software (particularly in AI), recruits top students under the age of 18 to work on "intelligent weapons design,"[3] and poaches U.S. software talent directly from the United States. In Russia, Vladimir Putin has told students, that "artificial intelligence is the future, not only for Russia, but for all humankind.... Whoever becomes the leader in this sphere will become the ruler

---

[2] Marc Andreessen, "Why Software Is Eating the World," *The Wall Street Journal*, August 20, 2011, 1.
[3] Stephen Chen, "China's Brightest Children Are Being Recruited to Develop AI 'Killer Bots,'" *South China Morning Post*, November 8, 2018.

of the world."[4] We can and must outcompete with software and the people who make it, not only to maintain U.S. military superiority but also to ensure that the power that software represents is used in accordance with American values.

What this report is about. This report summarizes the assessment of the Defense Innovation Board's (DIB's) Software Acquisition and Practices (SWAP) study. Congress charged[5] the DIB to recommend changes to statutes, regulations, processes, and culture to enable the better use of software in DoD. We took an iterative approach, mirroring the way modern software is successfully done, releasing a sequence of concept papers describing our preliminary observations and insights. (The latest versions of these are included in Appendix E.) We used those papers to encourage dialogue with a wide variety of individuals and groups to gain insights into the current barriers to implementing modern software effectively and efficiently. This document captures key insights from these discussions in an easy-to-read format that highlights the elements that we consider critical for DoD's success and serves as a starting point for continued discussions required to implement the changes that we recommend here.

This report is organized as follows:

- **Extended Abstract:** A two-page summary of the key takeaways from the report.
- **README** (this document)**:** A more detailed executive summary of the report. (A README file is used by the open source software community to provide essential information about a software package.) If your boss heard about the report or read the extended abstract, thought it was intriguing, and asked you to read the entire report and provide a short summary, cut and paste this chapter into your reply and you should be good to go.
- **Recommendations Cheat Sheet:** A list of the main lines of effort and primary recommendations, so you can pretty much stop at that point—or better yet, stop after suggesting to your boss they adopt them all.
- **Chapters 1–4:** Short descriptions of key areas and topics. If you attach the extended abstract to any one of these as a preface, it should be comprehensible.
- **Chapter 5:** A more detailed description of the recommendations and our rationale.
- **Supporting Information:** To ensure that the executive summary and the main body of the report satisfy the takeoff test[6] and the staple test,[7] we put most of the additional information generated during the study into a set of appendices. These provide a wealth of examples and

---

[4] James Vincent, "Putin Says the Nation that Leads in AI 'will be the ruler of the world,'" *The Verge*, September 4, 2017: https://www.theverge.com/2017/9/4/16251226/russia-ai-putin-rule-the-world.
[5] Section 872 of the FY18 NDAA directed the Secretary of Defense to "direct the Defense Innovation Board to undertake a study on streamlining software development and acquisition regulations." The DIB-SWAP members were charged to "review the acquisitions regulations applicable to, and organizational structures within, the Department of Defense…; review ongoing software development and acquisition programs…; produce specific and detailed recommendations…; and produce such additional recommendations for legislation." See Section 872 of the FY18 NDAA at https://www.congress.gov/115/plaws/publ91/PLAW-115publ91.pdf or Appendix J of this report.
[6] Reports should be short enough to read during takeoff, before the movies start and drinks are served.
[7] Any report that is going to be read should be thin enough to be stapled with a regular office stapler.

evidence, but we took care to put our essential arguments up front for less wonky types. Some highlights:

- **Draft implementation** (Appendix A)**:** For each recommendation, a summary of the background, desired state, stakeholders, role of Congress, and actions to be taken.
- **Legislative language** (Appendix B)**:** In response to 2016 NDAA Section 805, template legislative language for a new acquisition pathway and appropriation category for software, aligned with our recommendations.
- **An alternative to P-Forms and R-Forms** (Appendix C)**:** A different mechanism for budget submissions for software programs.
- **FAQs** (frequently asked questions, Appendix D)**:** A list of the most common questions that we get about the study and our attempt to answer them. (Question 1: Hasn't all of this been recommended before? A: Yes…).

Note: If you are reading any portion of the report in paper form, a navigable version is available at http://innovation.defense.gov/software.

Overarching themes. The rise of electronics, computing, and networking has forever transformed the way we live: software is a part of almost everything with which we interact in our daily lives, either directly through embedded computation in the objects around us or indirectly through the use of information technology through all stages of design, development, deployment, and operations. Our military advantage, coordination with allies and partners, operational security, and many other aspects of DoD activities are all contingent upon our software edge, and any lack thereof presents serious consequences. Software drives our military advantage: what makes weapon systems sophisticated is the software, not (just) the hardware.

Commercial trends show what is possible with software, from the use of open source tools to agile development techniques to global-scale cloud computing. Because of these changes, software can be developed, deployed, and updated much more quickly, which means systems need to be in place to support this speed. But modern software development requires a new set of skills and methodologies (e.g., generalist software engineers, specialized product management, DevOps and DevSecOps, agile development). Hence, the policies and systems surrounding software must be transformed to support software, not Cold-War-era weapon manufacturing.

The incoming generation of military and civilian personnel began life digitally plugged-in, with an innate reliance on software-based systems. They will demand new concepts of operations, tactics, and strategies to maintain the edge they need. If DoD can refactor its acquisition processes and transform its culture and personnel policies before it is too late, this software-savvy generation can still set the Department on the right course.

As we studied the methods that the private sector has used to enable software to transform its operations and considered how to best apply those practices to the defense enterprise, three overarching themes emerged as the basis for our recommendations:

1. Speed and cycle time are the most important metrics for software.

2. Software is made by people and for people, so digital talent matters.
3. Software is different than hardware (and not all software is the same).

Speed and cycle time are the most important metrics for software. Most DoD software projects are currently managed using "waterfall" development processes, which involve spending years on developing requirements, taking bids and selecting contractors, and then executing programs that must meet the listed requirements before they are "done." This results in software that takes so long to reach the field that it is often not well matched to the current needs of the user or tactics of our adversaries, which have often changed significantly while the software was being written, tested, and accepted. Being able to develop and deploy faster than our adversaries means that we can provide more advanced capabilities, respond to our adversaries' moves, and be more responsive to our end users. Faster reduces risk because it demands focus on the critical functionality rather than over-specification or bloated requirements. It also means we can identify trouble earlier and take faster corrective action, which reduces cost, time, and risk. Faster leads to increased reliability: the more quickly software/code is in the hands of users, the more quickly feedback can focus on efforts to deploy greater capability. Faster gives us a tactical advantage on the battlefield by allowing operation and response inside our adversaries' observe–orient–decide–act (OODA) loops. Faster is more secure. Faster is possible.

Software is made by people and for people, so digital talent matters. Current DoD human resource policies are not conducive to attracting, retaining, and promoting digital talent. Talented software developers and acquisition personnel with software experience are often put in jobs that do not allow them to make use of those talents, particularly in the military where rotating job assignments may not recognize and reward the importance of software development experience. As Steve Jobs observed,[8] one of the major differences between hardware and software is that for hardware the "dynamic range" (ratio between the best in class and average performance) is, at most, 2:1. But, the difference between the best software developer and an average software developer can be 50:1, or even 100:1, and putting great developers on a team with other great developers amplifies this effect. Today, in DoD and the industrial base that supports it, the people with the necessary skills exist, but instead of taking advantage of their skills we put them in environments where it is difficult for them to be effective. DoD does not take advantage of already existing military and civilian personnel expertise by offering pay bonuses, career paths that provide the ability to stay in their specialization, or access to early promotions. Skilled software engineers and the related specialties that are part of the overall software ecosystem need to be treated as a special force; the United States must harness their talent for the great benefits that it can provide.

Software is different than hardware (and not all software is the same). Over the years, Congress and DoD have established a sophisticated set of statutes, regulations, and instructions that govern the development, procurement, and sustainment of defense systems. This process evolved in the context of the Cold War, where major powers designed and built aircraft carriers, nuclear weapons, fighter jets, and submarines that were extremely expensive, lasted a very long time, and required tremendous access to capital and natural resources. Software, on the other hand,

[8] Steve Jobs, "Steve Jobs: The Lost Interview," interview by Robert X. Cringely for the 1995 PBS documentary, *Triumph of the Nerds*, released to limited theaters in 2012, video.

is something that can be mastered by a ragtag bunch of teenagers with very little money—and can be used to quickly destabilize world powers. Currently most parts of DoD develop, procure, and manage software like hardware, assuming that it is developed based on a fixed set of specifications, procured after it has been shown to comply with those specifications, "maintained" by block upgrades, and upgraded by replaying this entire procurement process linearly. But software development is fundamentally different than hardware development, and software should be developed, deployed, and continuously improved using much different cycle times, support infrastructure, and maintenance strategies. Testing and validation of software is also much different than for hardware, both in terms of the ability to automate but also in the potential vulnerabilities found in software that is not kept up to date. Software is never "done" and must be managed as an enduring capability that is treated differently than hardware.

Main lines of effort. DoD's current approach to software is a major driver of cost and schedule overruns for Major Defense Acquisition Programs (MDAPs). Congress and DoD need to come together to fix the acquisition system for software because it is a primary source of its acquisition headaches.

Bringing about the type of change that is required to give DoD the software capabilities it needs is going to take a significant amount of work. While it is possible to use the current acquisition system and DoD processes to develop, procure, assure, deploy, and continuously improve DoD software, the statutes, regulations, processes, and culture are debilitating. The current approach to acquisition was defined in a different era, for different purposes, and only works for software projects through enormous effort and creativity. Congress, the Office of the Secretary of Defense (OSD), the Armed Services, defense contractors, and the myriad government and industry organizations involved in getting software out the door need to make major changes (together).

To better organize our specific recommendations, we identified broad lines of effort that bring together different parts of the defense ecosystem as stakeholders. Here are the four main lines of effort that we recommend they undertake:

A. **(Congress and DoD) Refactor statutes, regulations, and processes for software,** enabling rapid deployment and continuous improvement of software to the field and providing increased insight to reduce the risk of slow, costly, and overgrown programs. The management and oversight of software development and acquisition must focus on different measures and adopt a quicker cadence.

B. **(OSD and the Services) Create and maintain cross-program/cross-Service digital infrastructure** that enables rapid deployment, scaling, testing, and optimization of software as an enduring capability; manage it using modern development methods; and eliminate the existing hardware-centric regulations and other barriers.

C. **(The Services and OSD) Create new paths for digital talent (especially internal talent)** by establishing software development as a high-visibility, high-priority career track—with specialized recruiting, education, promotion, organization, incentives, and salary—and increasing the level of understanding of modern software within the acquisition workforce.

D. **(DoD and industry) Change the practice of how software is procured and developed** by adopting modern software development approaches, prioritizing speed as the critical metric, ensuring cyber protection is an integrated element of the entire software life cycle, and purchasing existing commercial software whenever possible.

None of these can be done by a single organization within the government. They will require a bunch of hard-working, well-meaning people to work together to craft a set of statutes, regulations, processes, and (most importantly) a culture that recognizes the importance of software, the need for speed and agility (theme 1), the critical role that smart people have to play in the process (theme 2), and the impact of inefficiencies of the current approach (theme 3). In many ways this mission is as challenging as any combat mission: while participants' lives may not be directly at risk in defining, implementing, and communicating the needed changes to policy and culture, the lives of those who defend our nation ultimately depend on DoD's ability to redefine its approach to delivering combat-critical software to the field.

Refactor statutes, regulations, and processes, streamlined for software. Congress has created many workarounds to allow DoD to be agile in its development of new weapon systems, and DoD has used many of these to good effect. But the default statutes, regulations, and processes that are used for software too often rely on the traditional hardware mentality (repeat: software is different than hardware), and those practices do not take advantage of what is possible (or, frankly, necessary, given the threat environment) with modern software. We think that a combination of top-down and bottom-up pressure can break us out of the current state of affairs, and creating a new acquisition pathway that is tuned for software (of various types) will make a big difference. To this end, Congress and DoD should prototype and, after proving success, create mechanisms for ideation, appropriation, and deployment of software-driven solutions that take advantage of the unique features of software (versus hardware) development (start small, iterate quickly, terminate early) and provide purpose-fit methods of oversight. As an important aside, note that throughout this study our recommendations adhere to this guiding axiom—start small, iterate quickly—the same axiom that characterizes the best of modern software innovation cycles (see the "DIB Ten Commandments of Software" in Appendix E for more information about the DIB's guiding principles for software acquisition).

Create and maintain cross-program/cross-Service digital infrastructure. Current practice in DoD programs is that each individual program builds its own infrastructure for computing, development, testing, and deployment, and there is little ability to build richer development and testing capabilities that are possible by making use of common infrastructure. Instead, we need to create, scale, and optimize an enterprise-level architecture and supporting infrastructure that enables creation and initial fielding of software within six months and continuous delivery of improvements on a three-month cycle. This "digital infrastructure," common in commercial IT, is critical to enable rapid deployment at the speed (and scale) of relevance. In order to implement this recommendation, Congress and DoD leadership must figure out ways to incentivize the Services and defense contractors to build on a common set of tools (instead of inventing their own) *without* just requiring that everyone uses one DoD-wide (or even Service-wide) platform. Similarly, OSD will have to define non-exceptions-based alternatives to (or at least pathways through) Joint Capabilities Integration and Development System (JCIDS), Planning, Programing, Budget and Execution

(PPB&E), and Defense Federal Acquisition Regulation Supplement (DFARS)[9] that are optimized for software. The Director, Operational Test and Evaluation (DOT&E) will need new methods for OT&E that match the software's speed of relevance, and Cost Assessment and Program Evaluation (CAPE) will have to capture better data and leverage AI/ML as a tool for cost assessment and performance evaluation. Finally, the Services will need to identify, champion, and measure platform-based, software-intensive projects that increase software effectiveness, simplify interconnectivity among allies, and reform business practices. Subsequent chapters in our report provide specific recommendations on each of these areas.

Create new paths for digital talent (especially internal talent). The biggest enabler for great software is providing great people with the means to contribute to the national security mission. While the previous recommendations speak to providing the tools and infrastructure DoD technologists need to succeed, it is equally important that the Department's human capital strategies allow them to even do this work consistently in the first place. Driving the cultural transformation to support modern, cloud-based technology requires new types of skills and competencies, changing ratios of program managers to software engineers, moving from waterfall development to DevSecOps[10] development, and dealing with all of the change management that comes with it. This is not an easy task, but arguably one of the most important. While compensation is a major driver in attracting competitive talent, DoD must also make changes in the roles, methodologies, cultures, and other aspects of the transformation that industry is already undergoing and that the government must undergo as well.

Increasing developer talent is not the only workforce challenge. DoD must also change how the government manages its programs and contractors, which goes beyond just moving to DevSecOps development. The government must have experts well steeped in the software development process and architecture design to adequately manage both organic activities and contracted programs. They must have the skills to detect when contractors are going down the wrong path, choosing a bad implementation approach, or otherwise wasting government resources. This is perhaps the best argument for ensuring we have software development experience natively in the government, rather than relying primarily on external vendors; unless there are software-knowledgeable members on the core team, it is impossible to effectively monitor and manage outsourced projects. This is especially true with the movement to DevSecOps.

In implementing this change in the workforce, it is particularly important to provide new career paths for digital talent and enable the infrastructure and environment required to allow them to succeed. The current General Schedule (GS) system favors time in grade over talent. This simply will not work for software. The military promotion system has the same problem. As with sports, great teams make a huge difference and, in software, we need to make sure those teams have the tools they need to succeed and reward them appropriately—through recognition, opportunities for impact, career advancement, and pay. Advanced expertise in procurement, project management, evaluation and testing, and risk mitigation strategies will also be needed to create the types of elite teams that are necessary. A key element of success is finding ways to keep talented

[9] Common DoD acronyms are defined in Appendix I (Acronyms and Glossary).
[10] An iterative software development methodology that combines development, security, and operations as key elements in delivering useful capability to the user of the software. See Section 2.1 for details.

people in their roles (rather than transferring them out because it is the end of their assignment), and promoting people based on their abilities, not based on their years of service.

Change the practice of how software is procured and developed. The items above are where we think Congress and the Department should focus in terms of statutory, regulatory, and process changes. But a major element is also the need to change the *culture* around software within Congress, DoD, and the defense industrial base. We use the term “DevSecOps” as our label for the type of culture that is needed: iterative development that deploys secure applications and software into operations in a continuing (and continuous) fashion.

Numerous projects and groups have demonstrated the ability to implement DevSecOps within the existing acquisition system. But the organizations we previously mentioned—DDS, JIDO, DIU, and Kessel Run—are the exception rather than the rule, and the amount of effort required to initiate and sustain their activities is enormous. Instead, DoD should make legacy programs that use outdated techniques for developing software fight for existence (and in most cases replace them with new activities that embrace a DevSecOps approach).

Getting started now. The types of changes we are talking about will take years to bring to complete fruition. But it would be a mistake to spend two years figuring out what the answer should look like, spend another two years prototyping the solutions to make sure we are right, and then spend two to four more years implementing the changes in statutes, regulations, processes, and culture that are actually required. Let’s call that approach the “hardware” approach. Software is different than hardware, and therefore the approach to implementing change for software should be different as well.

Indeed, most (if not all) of the changes we are recommending are not new and not impossible to make. The 1987 DSB Task Force on Military Software,[11] chaired by legendary computer scientist Fred Brooks, wrote an outstanding report that already articulated much of what we are saying here. And the software industry has already implemented and demonstrated the utility of the types of changes we envision. The problem appears to be in getting the military enterprise to adopt a software mindset and implement a DevSecOps approach in a system that was intended to make sure that things would not move too quickly.

DoD could address many of our issues by adopting existing best practices of the private sector for agile development, including making use of software as a service; taking advantage of modern (cloud) infrastructure, tools, computing, and shared libraries; and employing modern software logistics and support delivery systems for software maintenance, development, and updating (patching). We do not need to study these; we need to get going and implement them. Here is a proposed timeline for implementing the primary recommendations of this report, starting *now*:

- (Immediately): Define, within 60 days after delivery of this report to Congress, a detailed implementation plan and assign owners to begin each of the top recommendations.

[11] Defense Science Board Task Force, *Military Software* (Washington, DC: Office of the Under Secretary of Defense for Acquisition, September 1987), https://apps.dtic.mil/dtic/tr/fulltext/u2/a188561.pdf.

- FY19 (create): High-level endorsement of the vision we articulate here, and support for activities that are consistent with the desired end state (i.e., DevSecOps and enterprise-level architecture and infrastructure). Identify and launch programs to move out on the priority recommendations (start small, iterate quickly). If you are reading this and are in a position of leadership in your organization, pass this on to others with your seal of approval and a request for your team to develop two or three plans of action for how it can be applied in your domain. If someone comes to you with a proposal that aligns with the objectives we have outlined here, find a way to be on the front line of changing DoD to a "culture of yes."
- FY20 (deploy): Initial deployment of authorities, budgets, and processes for software acquisition and practices reform. Execute representative programs according to the lines of effort and recommendations in this report, implement now, measure results, and modify approaches. Implement this report in the way we implement modern software.
- FY21 (scale): Streamlined authorities, budgets, and processes enabling software acquisition and practices reform at scale. In this time frame, we need a new methodology to estimate as well as determine the value of software capability delivered (and not based on lines of code).
- FY22 (optimize): Conditions established so that all DoD software development projects transition (by choice) to software- enabled processes, with the talent and ecosystem in place for effective management and insight.

In the remainder of this report, we provide a rationale for the approach that we are advocating. Chapter 1 makes the case for why software is important to DoD, including a taxonomy of the different types of software that need to be considered (not all software is the same). In Chapter 2, we describe how software is developed in the private sector and what is required in terms of workforce, infrastructure, and culture. Chapter 3 is an attempt to summarize what has already been said by other studies and groups, why the situation has not changed, and how we think this study can potentially lead to a different outcome. Chapters 4 and 5 contain our recommendations for how to move forward. In Chapter 4, we present three alternative paths to consider: doing the best we can with the current system; streamlining statutes, regulations, and processes so that they are optimized for software (instead of hardware); and making more radical changes that create entirely new appropriation categories and acquisition pathways. Finally, Chapter 5 describes the path that we recommend be taken, broken out along the lines of effort described above, and with a set of 10 primary recommendations followed by 16 additional recommendations (a detailed draft implementation plan for implementing each is included in Appendix A).

A two-page summary ("cheat sheet") of the lines of effort and recommendations follows.

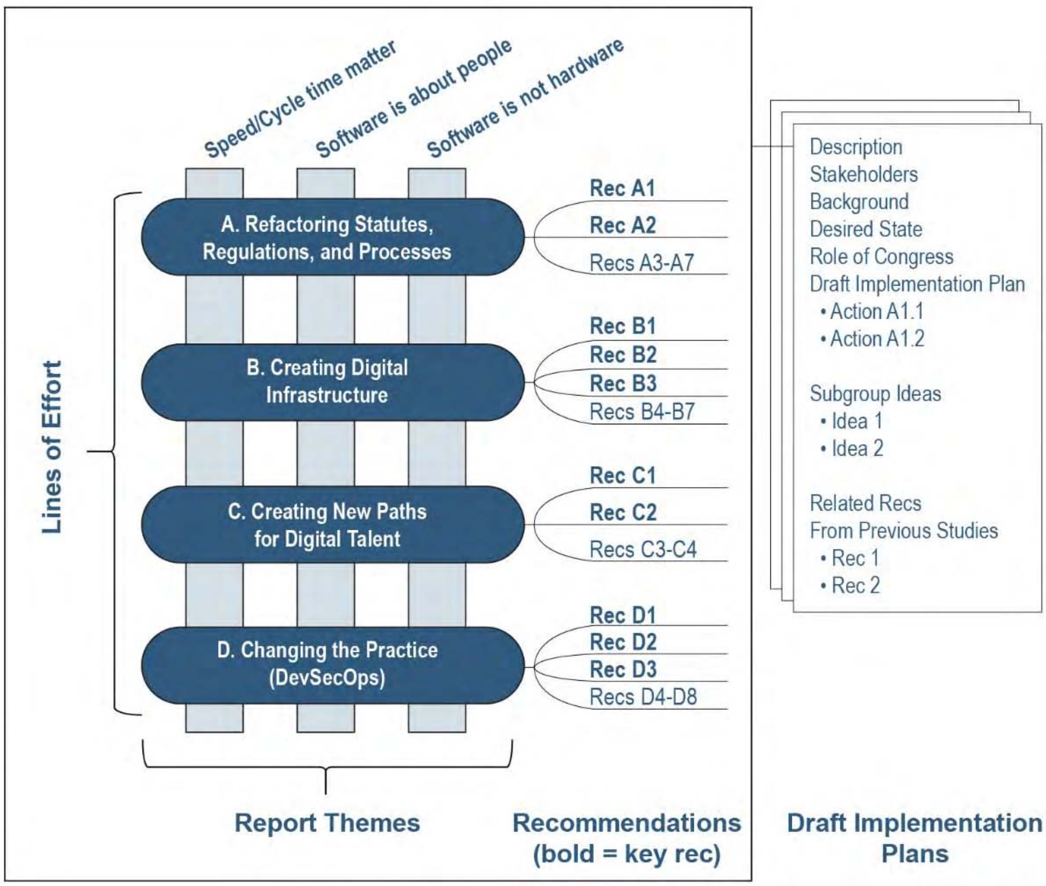
DIB SWAP Study

Recommendations "Cheat Sheet"

This sheet contains a list of the recommendations from the Defense Innovation Board's (DIB's) Software Acquisition and Practices (SWAP) study. The recommendations below include input from the following sources:

- DIB Guides for Software (Appendix E)
- SWAP working group reports (Appendix F)
- Previous software acquisition reform studies (starting with the 1987 DSB study)

The recommendations are organized according to four major lines of effort and each recommendation contains background information, a proposed owner for implementing the recommendation, as well as a more detailed draft implementation plan, a list of other offices that are affected, and additional details. The following diagram documents this structure:



For each recommendation, a draft implementation plan can be found in Appendix A that gives more detail on the rationale, supporting information, similar recommendations, specific action items, and notes on implementation. Potential legislative language to implement selected recommendations is included in Appendix B.

The Ten Most Important Things to Do (Starting Now!)

Line of Effort A (Congress and OSD): Refactor statutes, regulations, and processes for software	
A1	Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics
A2	Create a new appropriation category for software capability delivery that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&E, production, and sustainment
Line of Effort B (OSD and Services): Create and maintain cross-program/cross-Service digital infrastructure	
B1	Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors
B2	Create, implement, support, and use fully automatable approaches to testing and evaluation (T&E), including security, that allow high-confidence distribution of software to the field on an iterative basis
B3	Create a mechanism for Authorization to Operate (ATO) reciprocity within and between programs, Services, and other DoD agencies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapon) systems, and Services
Line of Effort C (Services and OSD): Create new paths for digital talent (especially *internal* talent)	
C1	Create software development units in each Service consisting of military and civilian personnel who develop and deploy software to the field using DevSecOps practices
C2	Expand the use of (specialized) training programs for CIOs, SAEs, PEOs, and PMs that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software
Line of Effort D (DoD and industry): Change the practice of how software is procured and developed	
D1	Require access to source code, software frameworks, and development toolchains—with appropriate IP rights—for DoD-specific code, enabling full security testing and rebuilding of binaries from source
D2	Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough
D3	Shift from the use of rigid lists of requirements for software programs to a list of desired features and required interfaces/characteristics to avoid requirements creep, overly ambitious requirements, and program delays

Chapter 5 provides additional context and Appendix A contains draft implementation plans.

Chapter 1. Who Cares: Why Does Software Matter for DoD?

The future battlespace is constructed of not only ships, tanks, missiles, and satellites, but also algorithms, networks, and sensor grids. Like no other time in history, future wars will be fought on civilian and military infrastructures of satellite systems, electric power grids, communications networks, and transportation systems, and within human networks. Both of these battlefields—electronic and human—are susceptible to manipulation by adversary algorithms.

— Cortney Weinbaum and Lt Gen John N.T. "Jack" Shanahan, "Intelligence in a Data-Driven Age," (Joint Force Quarterly 90, 2018), 5

This chapter provides a high-level vision of why software is critical for national security and the types of software we will have to build in the future. We also provide a description of different types of software, where they are used, and why a one-size-fits-all approach will not work.

1.1 Where Are We Coming From, Where Are We Going?

While software development has always been a challenge for the Department of Defense (DoD), today these challenges greatly affect our ability to deploy and maintain mission-critical systems to meet current and future threats. In the past, software simply served as an enabler of hardware systems and weapons platforms. Today, software defines our mission-critical capabilities and our ability to sense, share, integrate, coordinate, and act.

Software is everywhere and is in almost everything that the Department operates and uses. Software drives our weapon systems; command, control, and communications systems; intelligence systems; logistics; and infrastructure, and it drives much of the backroom enterprise processes that make the Department function. If cyber is the new domain in which we are fighting, then our ability to maintain situational awareness and our ability to fight, defend, and counter threats will be based on the capabilities of our software. In this new domain, software is both an enabler as well as a target of the fight.

As our military systems become increasingly networked and automated, as autonomy becomes more prevalent, and as we become more dependent on machine learning (ML) and artificial intelligence (AI), our ability to maintain superiority will be directly linked to our ability to field and maintain software that is better, smarter, and more capable than our adversaries' software. Even our ability to defend against new physical and kinetic threats such as hypersonics, energetics, and biological weapons will be based on software capabilities. We need to identify and respond to these new threats as they happen in near real time. Our ability to identify and respond to these new threats will be based on our ability to develop and push new software-defined capabilities to meet those threats on time scales that greatly outpace our adversaries' ability to do so.

The need to meet future threats requires us to rethink how we develop, procure, assure, deploy, and continuously improve software. DoD's current procurement processes treat software programs like hardware programs, but DoD can no longer take years to develop software for its major systems. Software cannot be an afterthought to hardware, and it cannot be acquired, developed, and managed like hardware. DoD's acquisition and development approaches are increasingly antiquated and do not meet the timely demands of its missions. Fixing the

Department's software approach involves more than just making sure that we get control over cost and budget; it concerns our ability to maintain our fighting readiness and our ability to win the fight and counter any threat regardless of domain and regardless of adversary.

1.2 Weapons and Software and Systems, Oh My! A Taxonomy for DoD

Not all software systems are the same, and therefore it is important to optimize development processes and oversight mechanisms to the different types of software DoD uses. We distinguish here between two different aspects of software: *operational function* (use) and *implementation platform*. To a large extent, a given operational function can be implemented on many different computational platforms depending on whether it is a mission support function (where high-bandwidth connectivity to the cloud is highly likely) or a field-forward software application (where connectivity many be compromised and/or undesirable).

We define three broad operational categories:

- *Enterprise systems*: very large-scale software systems intended to manage a large collection of users, interface with many other systems, and generally used at the DoD level or equivalent. These systems should always run in the cloud and should use architectures that allow interoperability, expandability, and reliability. In most cases the software should be commercial software purchased (or licensed) without modification to the underlying code, but with DoD-specific configuration. Examples include e-mail systems, accounting systems, travel systems, and human resources (HR) databases.

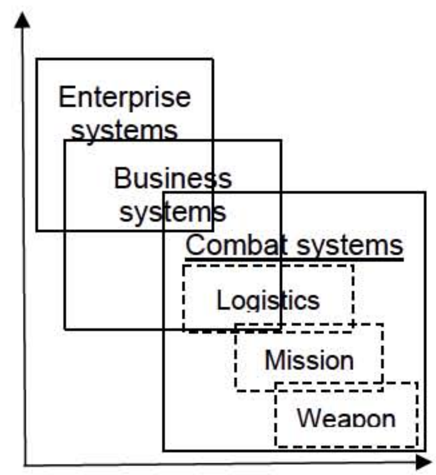


Figure 1.1. Different types of software.

- *Business systems*: essentially the same as enterprise systems, but operating at a slightly smaller scale (e.g., for one of the Services). Like enterprise systems, they are interoperable, expandable, reliable, and probably based on commercial offerings. Similar functions may be customized differently by individual Services, though they should all interoperate with DoD-wide enterprise systems. Depending on their use, these systems may run in the cloud, in local data centers, or on desktop computers. Examples include software development environments and Service-specific HR, financial, and logistics systems.

- *Combat systems*: software applications that are unique to the national security space and used as part of combat operations. Combat systems may require some level of customization that may be unique to DoD, not the least of which will be specialized cybersecurity considerations to enable them to continue to function during an adversarial attack. (Note that since modern DoD enterprise and business systems depend on software, cyber attacks to disrupt the operations of these systems have the potential to be just as crippling as those aimed at combat systems.)

We further break down combat systems into subcategories:

- *Logistics systems*: any system used to keep track of materials, supplies, and transport as part of operational use (versus Service-scale logistics systems, with which they should interoperate). While used actively during operations, logistics systems are likely to run on commercial hardware and operating systems, allowing them to build on commercial off-the-shelf (COTS) technologies. Platform-based architectures enable integration of new capabilities and functions over time (probably on a months-long or annual time scale). Operation in the cloud or based on servers is likely.

- *Mission systems*: any system used to plan and monitor ongoing operations. Similar to logistics systems, this software will typically use commercial hardware and operating systems and may be run in the cloud, on local services, or via a combination of the two (including fallback modes). Even if run locally (such as in an air operations center), they will heavily leverage cloud technologies, at least in terms of critical functions. These systems should be able to incorporate new functionality at a rate that is set by the speed at which the operational environment changes (days to months).

- *Weapon systems*: any system capable of delivering lethal force, as well as any direct support systems used as part of the operation of the weapon. Note that our definition differs from the standard DoD definition[1] of a weapon system, which also includes any related equipment, materials, services, personnel, and means of delivery and deployment (if applicable) required for self-sufficiency. The DoD definition would most likely include the mission and logistics functions, which we find useful to break out separately. Software on weapon systems is traditionally closely tied to hardware, but as we move toward greater reliability of software-defined systems and distributed intelligence, weapon systems software is becoming increasingly hardware independent (similar to operating systems for mobile devices, which run across many different hardware platforms).

We also define several different types of computing platforms on which the operational functions above might be implemented:

- *Cloud computing*: computing that is typically provided in a manner such that the specific location of the compute hardware is not relevant (and may change over time). These systems typically run on commercial hardware and use commercial operating systems, and the applications running on them run even as the underlying hardware changes. The important point here is that the hardware and operating systems are generally transparent to the application and its users (see figure 1.2).

[1] The Department of Defense, *DoD Dictionary of Military and Associated Terms* (Washington, DC: Department of Defense, as of February 2019), 252.

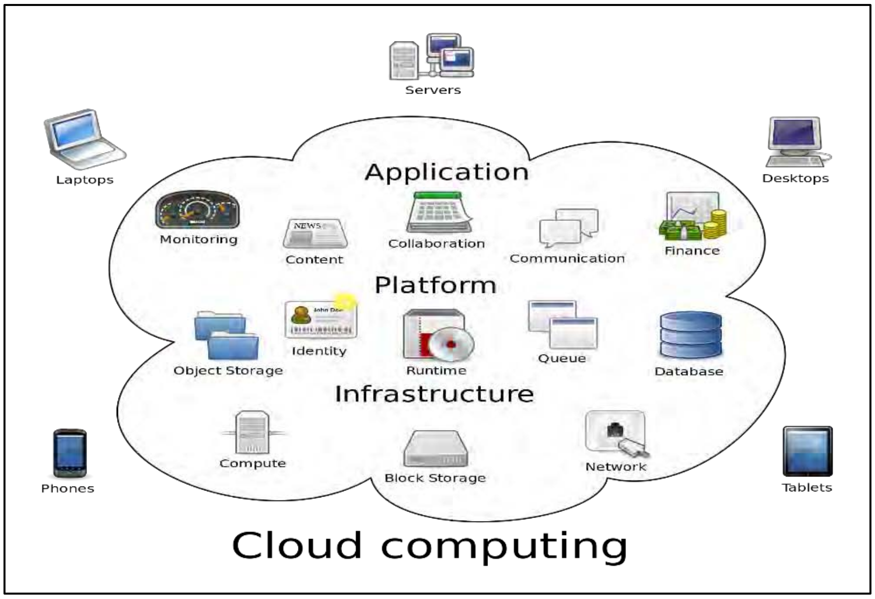


Figure 1.2. Cloud computing environment.
[Image by Sam Johnston is licensed under CC BY-SA 3.0]

- *Client/server computing*: computing provided by a combination of hardware resources available in a computing center (servers) as well as local computing (client). These systems usually run on commercial hardware and use commercial operating systems.

- *Desktop/laptop/computing*: computing that is carried out on a single system, often by interacting with data sources across a network. These systems usually run on commercial hardware and use commercial operating systems.

- *Mobile computing:* computing that is carried out on a mobile device, usually connected to the network via wireless communications. These systems usually run on commercial operating systems using commodity chipsets.

- *Embedded computing*: computing that is tied to a physical, often-customized hardware platform and that has special features that require careful integration between software and hardware (see figure 1.3).

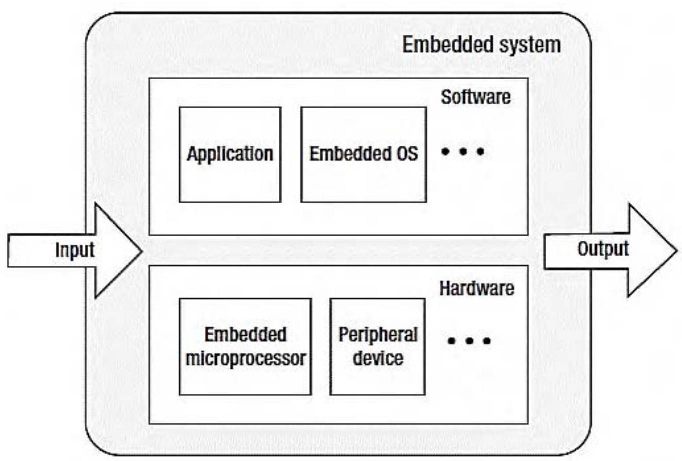


Figure 1.3. Embedded system architecture.
[Image from Ebrary.net]

A single software system may have multiple components or functions that span several of these definitions, and components of an integrated system likely have elements that do the same. The key point is that each type of software system has different requirements in terms of how quickly it can/should be updated, the level of information assurance required, and the organizations that will participate in development, testing, customization, and use of the software. Different statutes, regulations, and processes may be required for different types of software (and these would differ greatly from those used for hardware).

Having defined systems that deliver effects and the kinds of computing platforms on which software is hosted, we now distinguish between four primary types of software. We use these terms throughout the rest of the report to differentiate the acquisition and deployment approaches needed for different types of software:

- **Type A (Commercial Off-the-Shelf [COTS] applications):** The first class of software consists of applications that are available from commercial suppliers. Business processes, financial management, HR, software development, collaboration tools, accounting software, and other "enterprise" applications in DoD are generally not more complicated nor significantly larger in scale than those in the private sector. Unmodified commercial software should be deployed in nearly all circumstances. Where DoD processes are not amenable to this approach, the Department should modify its processes, not the software.

- **Type B (Customized Software):** The second class of software constitutes those applications that consist of commercially available software that is customized for DoD-specific usage. Customization can include the use of configuration files, parameter values, or scripted functions tailored for DoD missions. These applications generally require (ongoing) configuration by DoD personnel, contractors, or vendors.

- **Type C (COTS Hardware/Operating Systems):** The third class of software applications is those that are highly specialized for DoD operations but run on commercial hardware and standard operating systems (e.g., Linux or Windows). These applications will generally be able to take advantage of commercial processes for software development and deployment, including the use of open source code and tools. This class of software includes applications written by DoD personnel as well as those that are developed by contractors.

- **Type D (Custom Software/Hardware):** This class of software focuses on applications involving real-time, mission-critical, embedded software whose design is highly coupled to its customized hardware. Examples include primary avionics or engine control, or target tracking in shipboard radar systems. Requirements such as safety, target discrimination, and fundamental timing considerations demand that extensive formal analysis, test, validation, and verification activities be carried out in virtual and "iron bird" environments before deployment to active systems. These considerations also warrant care in the way application programming interfaces (APIs) are potentially presented to third parties.

We note that these classes of software are closely related to those described in the 1987 Defense Science Board (DSB) study on military software, which categorized software as "standard" (roughly capturing types A and B), "extended" (type C), "embedded" (type D), and "advanced" (which the study categorized as "advanced and exploratory systems," which are not so relevant here).

1.3 What Kind of Software Practices Will We Have to Enable?

The competitor that can realize software-defined military capability the fastest is at an advantage in future conflicts. We must shorten our development cycles from years to months so that we can react and respond within the observe–orient–decide–act (OODA) loop of the threats we face. Agile methodologies such as DevSecOps enable this rapid cycle approach (see "Detecting Agile BS" in Appendix E for more information about agile methodologies), and in addition to development we will need to test and validate software in real time as part of the integrated approach that DevSecOps demands. Quality assurance must be a continuous and fully integrated process throughout every phase of the software cycle. We need to build software pipelines that are able to develop and deploy software and provide updates as quickly as modern-day commercial companies so that we can respond to new threats (especially when the target will be our software). We must treat software as a continuous service rather than as block deliverables. It is important to have the agility in our procurement approach that will allow program managers to change priorities based on the needs and timing of the end users.

In the near future, DoD's acquisition and use of business systems should closely mirror industry and the private sector. DoD should modify its processes to mimic industry's best practices rather than try to contract for and maintain customized software. Figure 1.4 illustrates how this looks at Facebook (see also Section 2.1 for examples of best practices in industry).

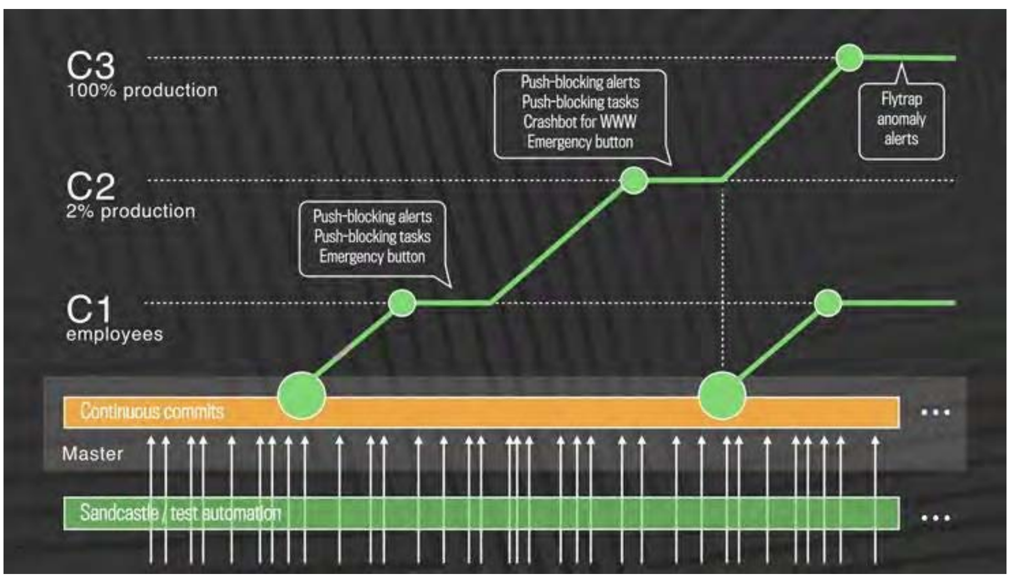


Figure 1.4. Facebook's continuous delivery process. Code updates that have passed a series of automated internal tests (bottom) land in the master development branch and are pushed out to employees (C1). In this stage, push-blocking alerts are generated if there are problems, and an emergency stop button keeps the release from going any further. If everything is OK, changes are pushed to 2 percent of production (C2), where signal and monitor alerts are again collected, especially for edge cases that testing or employee use may not have picked up. Finally, changes are rolled out to 100 percent of production (C3), where the "Flytrap" tool aggregates user reports and provides alerts on any anomalies. The cycle time between updates can be as short as a few hours. [Diagram and caption adapted from Facebook Engineering Blog, 31 Aug 2017 post on "Rapid release at massive scale"]

DoD should also adopt commercial logistics and mission planning software (COTS) wherever possible and reduce its reliance on government off-the-shelf (GOTS) solutions. Good logistics and mission software reduces process complexity, improves situational awareness, reduces costs, and simplifies planning while improving speed of delivery and streamlining performance.

For software that is closely tied to hardware, software-defined systems should be easier to develop, maintain, and upgrade than classic embedded systems. A well-designed system would allow new capabilities to be delivered directly to the edges of the network from the cloud in the same way new capabilities are delivered to consumer mobile devices.

DoD should manage software by measuring value delivered to the user rather than by monitoring compliance with requirements. Accountability should be based on delivering value to the user and solving user needs, not on complying with obsolete contracts or requirements documents.

Program managers must identify potential problems earlier (ideally, within months) and take corrective action quickly. Troubled programs must fail quickly, and the Department needs to learn from them. As we witnessed throughout our work on this study, many software programs are too big, are too complex, and take too long to deliver any value to users. Development must be staged and follow the best practice of smaller deliverables faster, with higher frequency of updates and new features. Initially, program development should focus on developing the "minimum viable product" (MVP) and getting it delivered to the customer more quickly than traditionally run programs. (The MVP for a software program represents the first point at which the code can start doing useful work and also at which feedback can be gathered that supports refinement of features.)

Software developers within the defense community need the same modern tools, systems, environments, and collaboration resources that commercial industry has adopted as standard. Without these, the Department undermines the effectiveness of its software developer base, and its ability to attract and retain our software human capital, both within DoD and among its suppliers. With the introduction of new technologies like ML and AI and the ever-increasing interdependence among networked heterogeneous systems, software complexity will continue to increase logarithmically. DoD needs to continuously invest in new development tools and environments including simulation environments, modeling, automated testing, and validation tools. DoD must invest in research and development (R&D) into new technologies and methodologies for software development to help the Department keep up with the ever-growing complexity of defense systems.

1.4 What Challenges Do We Face (and Consequences of Inaction)?

The world is changing. The United States used to be the dominant supplier of software and the world leader in software innovation. That is no longer the case. Due to the global digital revolution driven by the consumer and commercial markets, countries are building their own indigenous software capabilities and their own technology clusters. Countries like China are making huge investments in AI and cyber. China's 2030 plan envisions a $1 trillion AI industry in China.[2] China wants to become a cyber superpower and is investing in its capital markets, universities, research centers, defense industry, and commercial software companies to reach that goal.[3]

The potential long-term consequences of inaction are that our adversaries' software capabilities could catch and surpass those of the United States. If that happens, our adversaries would be able to develop new capabilities and potentially iterate faster than we can. They could respond to our defense systems faster than we can respond to theirs. If their algorithms and AI become superior to ours, they could hold a decisive advantage when any of our systems go up against any of theirs. And if their cyber capability becomes superior to ours, they could shut us down, cause chaos, continue to steal our secrets as they choose and without repercussions—especially if we could not attribute those attacks. Our adversaries' software capabilities are growing rapidly. If we do not keep pace, we could lose our defense technology advantage within a decade or much sooner.

[2] Vikram Barhat, "China Is Determined to Steal A.I. Crown from US and Nothing, Not Even a Trade War, Will Stop It," CNBC, May 4, 2018, https://www.cnbc.com/2018/05/04/china-aims-to-steal-us-a-i-crown-and-not-even-trade-war-will-stop-it.html.

[3] "China Is Seeking to Become a Cyber Superpower," *The Economist*, March 20, 2018, https://www.economist.com/graphic-detail/2018/03/20/china-is-seeking-to-become-a-cyber-superpower; and Rogier Creemers, Paul Triolo, and Graham Webster, "Translation: Xi Jinping's April 20 Speech at the National Cybersecurity and Informatization Work Conference," New America Blog Post, April 30, 2018, https://www.newamerica.org/cybersecurity-initiative/digichina/blog/translation-xi-jinpings-april-20-speech-national-cybersecurity-and-informatization-work-conference/.

Chapter 2. What Does It Look Like to Do Software Right?

Deliver performance at the speed of relevance. Success no longer goes to the country that develops a new technology first, but rather to the one that better integrates it and adapts its way of fighting. Current processes are not responsive to need; the Department is over-optimized for exceptional performance at the expense of providing timely decisions, policies, and capabilities to the warfighter. Our response will be to prioritize speed of delivery, continuous adaptation, and frequent modular upgrades. We must not accept cumbersome approval chains, wasteful applications of resources in uncompetitive space, or overly risk-averse thinking that impedes change. Delivering performance means we will shed outdated management practices and structures while integrating insights from business innovation.

— U.S. Department of Defense, "Summary of the 2018 National Defense Strategy of the United States of America: Sharpening the American Military's Competitive Edge," (Washington, DC: U.S. Department of Defense, 2018), 10

In many cases, the software acquisition approaches and practices in place within DoD today look strange and perplexing to those familiar with commercial software practices. While the mission-, security-, and safety-critical nature of DoD's software in the context of embedded weapons will have an impact on practices, the extreme degree of divergence from contemporary commercial practice has been an area of our focus. Our case studies, site visits, and other study activities allowed a closer look into the reasons for divergence and whether the absence of many commercial best practices is justified.

2.1 How It Works in Industry (and Can/Should Work in DoD): DevSecOps

Modern software companies must develop and deliver software quickly and efficiently in order to survive in a hyper-competitive environment. While it is difficult to characterize the entire software sector, in this section we outline a set of practices—based on documented approaches in industry[4]—that are representative of commercial environments where the delivery of software capability determines the success or failure of the company. These practices generally hold true in other industries where companies have unexpectedly found themselves in the software business due to an increasing reliance on software to provide their key offerings, such as automotive, banking, healthcare, and many others. In any

Figure 2.1 A former U.S. Marine Corps sergeant, now a Microsoft field engineer, works with an IT support specialist with the Navy as part of his job to travel to commercial companies and military bases across the country and train IT staff about a systems management product. [Photo by Sgt. Shellie Hall]

[4] Fergus Henderson, "Software Engineering at Google" (arXiv:1702.01715 [cs.SE], January 31, 2017).

environment, software engineering practices must be matched with the recruitment and retention of talented software expertise. These practices must be honed over time and adapted to lessons learned.

At a high level, DoD must move from waterfall and spiral development methods to more modern software development practices such as Agile, DevOps, and DevSecOps. "DevOps" represents the integration of software development and software operations, along with the tools and culture that support rapid prototyping and deployment, early engagement with the end user, automation and monitoring of software, and psychological safety (e.g., blameless reviews). "DevSecOps" (as depicted in figure 2.2) adds the integration of security at all stages of development and deployment, which is essential for DoD applications. DoD should adopt these techniques, with appropriate tuning of approaches used by the Agile/DevSecOps community for mission-critical, national security applications. DoD should use open source software when possible to speed development and deployment and leverage the work of others.

Figure 2.2. Continuous integration of development, security, and deployment (DevSecOps). [Adapted from an image by Kharnagy, licensed under CC BY-SA 4.0]

Generally, successful software companies have developed best practices in three categories:

Software development. These are software engineering practices that include source code management, software build, code review, testing, bug tracking, release, launch, and postmortems. Key best practices applicable to DoD software programs include the following:

- All source code is maintained in a single repository that is available to all software engineers. There are control mechanisms to manage additions to the repository, but in some cases all engineers are culturally encouraged to fix problems, independent of program boundaries.
- Developers are strongly encouraged to avoid "forking" source code (creating independent development branches) and focus work on the main branch of the software development.
- Code review tools are reliable and easy to use. Changes to the main source code typically require review by at least one other engineer, and code review discussions are open and collaborative.
- Unit test is ubiquitous, fully automated, and integrated into the software review process. Integration, regression, and load testing are also widely used, and these activities should be an integrated, automated part of daily workflow.
- Releases are frequent—often weekly. There is an incremental staging process over several days, particularly for high-traffic, high-reliability services.

- Postmortems are conducted after system outages. The focus of the postmortem is on how to avoid problems in the future and not on affixing blame.

Project management. Software projects must contribute to the overall aim of the business, and efforts must be aligned to that end goal.

- Individuals and teams set goals, usually quarterly and annually. Progress against those goals is tracked, reported, and shared across the organization. Goals are mechanisms to encourage high performance but can be decoupled from performance appraisal or compensation.
- The project approval process is organic. Significant latitude to initiate projects is given at all levels, with oversight responsibility given to managers and executives to allocate resources or cancel projects.

People management. Given the scarce number of skilled software engineers, successful software companies know how to encourage and reward good talent. Examples include the following:

- Engineering and management roles are clearly separated, with advancement paths for both. Technical career progression (e.g., for advanced and senior developers, fellows and senior fellows) parallels management career ladders; technical professionals receive similar compensation and accrue comparable respect within the organization. Similar distinctions are made between technical management and people management. The ratio of software engineers to product managers and program managers ranges from 4:1 to 30:1.
- Mobility throughout the organization is encouraged. This allows for the spread of technology, knowledge, and culture throughout the company.

In addition to these specific software development practices, another common approach to managing programs in industry is to move away from the specifications and requirements approach towards a feature management approach. This approach allows program managers to make agile decisions based on evolving needs and capabilities. Using a feature management approach, a program manager has a list of features and capabilities ranked by need, risk, cost, resources, and time. This list of capabilities is two to three times larger than what generally can be accomplished within a given time frame, a given budget, and a set of resources. Program managers make decisions about the feature mix, match investments to needs, and balance risk against performance. Capabilities are tested and delivered on a continuous basis, and maximum automation is leveraged for testing.

In industry, software programs initially start as an MVP. An MVP has just enough features to meet basic minimum functionality. It provides the foundational capabilities upon which improvements can be made. MVPs have significantly shorter development cycles than traditional waterfall approaches. The goal of MVPs is to get basic capabilities into users' hands for evaluation and feedback. Program managers use the evaluation and feedback results to rebalance and re-prioritize the software capability portfolio.

Portfolio success is measured based on performance of the *delivery* of capabilities as measured against user needs and strategic objectives within an investment cycle. Value is determined by output measurements rather than process measurements. Portfolio value is the aggregate of the total value of all of the capabilities delivered divided by total cost invested within a period of time. Blending higher risk/higher reward capabilities with lower risk/lower reward capabilities is the art of good portfolio management. Within a given period of time, program managers use diversification to spread risk and rewards. Good program managers identify troubled projects early and are encouraged either to quickly correct the problems or to quickly abandon failing efforts so that remaining resources can be husbanded and then reallocated to other priorities.

Software budgets are driven by time, talent, compute resources, development environment, and testing capabilities required to deliver capabilities. The capability and cost of talent vary greatly between software engineers, designers, programmers, and managers. The quality of engineering talent is the single largest variable that determines cost, risk, and duration of a software project. Good portfolio managers must take inventory of the range of software talent within a program and carefully allocate that talent across the portfolio of capabilities development.

2.2 Empowering the Workforce: Building Talent Inside and Out

One of the biggest barriers to realizing the software capabilities the Department so desperately needs is the way the Department manages the people necessary to build that capability. DoD cannot compete and dominate in defense software without a technical and design workforce within the Department that can both build software natively and effectively manage vendors to do the same, using the proven principles and practices described above. Some of the Department's human capital practices actively work against this critical goal.

If the Department wants to be good at software, it must be good at recruiting, retaining, leveraging, managing, and developing the people who make it. When we look at private-sector organizations and institutions that effectively use software to fulfill their mission, they

- understand the software professionals that they have, understand their workforce needs at a high level, and understand the gap between the two. (We say "at a high level" because we believe the gap is large enough that it is much more important to begin closing the gap than it is to measure the gap with too much precision.)
- have a strategy to recruit the people and skills they need to fulfill their mission, understanding what they uniquely have to offer in a competitive market.
- clearly understand the competencies required by software professionals in their organizations and the expectations of these professionals at each level in the organization.
- define career ladders for technical professionals that map software competencies and expectations from entry level to senior technical leadership and management.
- offer opportunities for learning and mentorship from more senior engineering and design leaders.
- count engineering and design leaders among their most senior leadership, with the ability to advocate across silos for the needs of the software and software acquisition workforce and support other senior leaders in understanding how to work with both.

- support a cadre of leadership able and empowered to create a culture of software management and promote common approaches, practices, platforms, and tools, while retaining the ability to use judgement about when to deviate from those common approaches and tools.
- reward software professionals based on merit and demonstrated contribution rather than time in grade.

Unfortunately, these are not the common descriptors for the software workforce practices in today's DoD.

DoD has long recognized that medicine and law require specialized skills, continuing education, and support and made it not only possible but desirable and rewarding to have a career as a doctor or lawyer in the armed forces. In contrast, software developers, designers, and managers in the Services must practice their skills intermittently and often without support as they endure frequent rotations into other roles. DoD does not expect a trained physician to constantly rotate into deployments focused on aviation maintenance, nor does it interrupt the training of a lawyer to teach him or her HR skills. Who would be comfortable being treated by a physician who worked in an institution that lacked common standards of care and provided no continuing education? And though software is often a matter of life and death, DoD's current human capital practices include all of these counterproductive features.

Figure 2.3. Airmen participate in Kessel Run's pair programming. [U.S. Air Force photo by Rick Berry]

The process to retool human capital practices to meet the challenge of software competency in DoD must start with the people the Department already has who have software skills or who are interested in acquiring them. Unlike medicine, software skills can be acquired through self-directed and even informal training resources such as on-demand, online webinars and coding boot camps, etc., and the Department has military and civilian individuals who have taken it upon themselves to gain technical skills outside of or in addition to formal DoD training. This kind of initiative and aptitude, especially when it results in real contribution to the mission, should be rewarded with appropriate opportunities for career advancement in this highly sought-after specialty. As we have witnessed during site visits for this study, there are also many individuals with more formally recognized software skills who are working with determination and even courage to try to deliver great software in service of the mission, but whose efforts to practice modern software techniques are poorly supported, and often actively blocked. Changes to policy that make clear the Department's support for these practices will help, but they must be married with support for the individuals to stay and grow within their chosen field. DoD could leverage several possible human capital pathways:

- Core military occupational series (MOS) and civilian occupational series for software development that include subcategories to address the various duties found in modern software development (e.g., developers/engineers, product owners, and designers).
- A secondary specialty series/designator for military members for software development. Experts come from various backgrounds, and a special secondary designator or occupational series for Service Members would be invaluable to tapping into their expertise even if they are not part of the core "Information Technology" profession.
- A Special Experience Identifier or other Endorsement for military and civilian acquisition professionals that indicates they have the necessary experience and training to serve on a software acquisition team. This Identifier or Endorsement should be a requirement to lead an acquisition team for a software procurement. Furthermore, this Identifier or Endorsement needs to be expanded to the broader team working the software procurement to include legal counsel, contract specialists, and financial analysts.

2.3 Getting It Right: Better Oversight AND Superior National Security

Getting software right in the Department requires more than changing development practices; oversight (and budgeting and finance) must also change. Those responsible for oversight of DoD software projects will need to learn to ask different questions and require different kinds of information on different tempos, but their reward will be more clarity, greater satisfaction with military software investments, and, ultimately, stronger national security.

Rules of thumb for those in appropriations and oversight roles over DevSecOps projects include the following:

Expect value to the user earlier. Oversight of monolithic, waterfall projects has generally focused on whether the team hit pre-determined milestones that may or may not represent actual value or even working code, and on figuring out what to do when they do not. When evaluating and appropriating funds to DevSecOps projects, it is more suitable to judge the project on the speed by which it delivers working code and actual value to users. In a waterfall project, changes to the plan generally reflect the team falling behind and are a cause for concern. In a project that is agile and takes advantage of the other approaches this study recommends (including software reuse), the plan is intended to be flexible because the team should be learning what works as they code and test.

Ask for meaningful metrics. Successful projects will develop metrics that measure value to the user, which involves close, ongoing communication with users. Source lines of code (SLOC) is not a measure of value and should not be used to evaluate projects in any case, as its use creates perverse incentives.

Assign a leader and hold him or her accountable. Part of the role of oversight is to ensure that there is a single leader who is qualified to lead in a DevSecOps framework and has the authority and responsibility to make the decisions necessary for the project to succeed. That person should have the authority to assign tasks and work elements; make business, product, and technical decisions; and manage the feature and bug backlogs. This person is ultimately responsible for how well the software meets the needs of its users, which is how the project should be evaluated.

Clarity and quality of leadership has long been tied to successful defense programs. Consider Kelly Johnson with the U-2, F-104, and SR-71. Paul Kaminski with stealth technology. Admiral Hyman Rickover with the nuclear Navy. Harry Hillaker with the F-16; and Bennie Schriever with the intercontinental ballistic missile. The list goes on. The United States Digital Service recognized this with Play 6 of the *Digital Services Playbook*—Assign One Leader and Hold That Person Accountable.[5] DoD would do well to remember this part of its history and work this practice into its oversight plan.

Speed increases security. Conventional wisdom in DoD says that programs must move slowly because moving quickly would threaten security. Often, the opposite is true. As we have learned from the cyber world, when we are facing active threats, our ability to achieve faster detection, response, and mitigation reduces the consequences of an attack or breach. In the digital domain, where attacks can be launched at machine speeds, where AI and ML can probe and exploit vulnerabilities in near real time, our current ability to detect, respond, and mitigate against digital threat leaves our systems completely vulnerable to our adversaries.

> The Department of Defense (DoD) faces mounting challenges in protecting its weapon systems from increasingly sophisticated cyber threats. This state is due to the computerized nature of weapon systems; DoD's late start in prioritizing weapon systems cybersecurity; and DoD's nascent understanding of how to develop more secure weapon systems. DoD weapon systems are more software dependent and more networked than ever before…. Potential adversaries have developed advanced cyber-espionage and cyber-attack capabilities that target DoD systems. (U.S. Government Accountability Office, Weapon Systems Cybersecurity: DoD Just Beginning to Grapple with Scale of Vulnerabilities [Washington, DC: U.S. Government Accountability Office, Oct 9, 2018], 2)

DoD must operate within its adversaries' digital OODA loop. Much like today's consumer electronic companies, the Department needs the ability to identify and mitigate evolving software and digital threats and to push continuous updates to fielded systems in near-real time.

DoD must be able to deploy software faster without sacrificing its abilities to test and validate software. To accomplish this, the Department needs to reimagine the software development cycle as a continuous flow rather than discrete software block upgrades. It should not only modernize to use a DevSecOps approach to software development but should also modernize its entire suite of development and testing tools and environments. DoD needs to be able to instrument its fielded systems so that we can build accurate synthetic models that can be used in development and test. The Department needs to be able to patch, update, enhance, and add new capabilities faster than our adversaries' abilities to exploit vulnerabilities.

Colors of money doom software projects. The foundational reasons for specific Congressional guidance on how money is to be spent make sense. But because software is in continuous

[5] "Digital Services Playbook," U.S. Digital Service, https://playbook.cio.gov/#plays_index_anchor.

development (it is never “done”—see Windows, for example), colors of money tend to doom programs. We need to create pathways for “bleaching” funds to smooth this process for long-term programs.

Do not pay for the factory every time you need a car. Appropriators must realize that DoD desperately needs common infrastructure if it is to increase the speed and quality of the software it produces. Today, it is as if the Department were buying cars but paying for the entire factory to build each car separately. Appropriators should fund the smart development of common infrastructure and reward its use in individual programs and projects. Evaluators should be wary of programs and projects that fail to articulate how they are taking advantage of common infrastructure and reusable components.

Standard is better than custom. In the same vein as the above, appropriators and evaluators should understand the benefits of using standards from the software development industry. Standards enable quality, speed, adoption, cost control, sustainability, and interoperability.

Technical debt is normal, and it is worth investing to pay it down. “Technical debt” refers to the cost incurred by implementing a software solution that is expedient rather than choosing a better approach that would take longer. Appropriators and evaluators should understandably expect to see progress in terms of features on a regular basis. The exceptions are when software teams must pay down technical debt or refactor code for greater performance. (This often results in fewer lines of code but higher performance, which is why it is a mistake to judge a software project based on the number of lines of code.) These periodic investments are to be expected on a DevSecOps project and are necessary to ensure the overall quality and stability of the project.

Use data as a compass, not a grade. Too often, evaluators and appropriators receive data about a program that suggests it is failing, but by the time they receive it, there is not much to be done about it. Data is collected manually, then processed and presented, and by the time it is being discussed, it is out of date. Mostly what happens at this point is that the project is given a poor grade, which makes the teams increasingly risk averse and demoralized. Instead, projects should be instrumented—equipped with built-in ways of seeing how and where they are going—so that the data is available both to the teams and to evaluators in time to make adjustments. In this model, the data is more like a compass, helping all parties make small corrections quickly to avoid the poor grade. An effective oversight function will help steer projects and hold them accountable, rather than punish poor performance.

2.4 Eye on the Prize: What Is the R&D Strategy for Our Investment?

The nature of software development may radically change in the near future. It is essential that the DoD adequately fund R&D programs to advance the fields of computer science, including computer programming, AI and ML, autonomy, quantum computing, networks and complex systems, man–machine interfaces, and cybersecurity.

Today, computers are controlled by programs that are comprised of sets of instructions and rules written by human programmers. AI and ML change how humans teach computers. Instead of providing computers with programmed instructions, humans will train or supervise the learning algorithm being executed on the computer. Training is inherently different than programming. Data becomes more important than code. Training errors are very different than programming errors. Hacking AI is very different than hacking code. The use of synthetic environments and "digital twins" (simulation-based emulators of physical components) may also become increasingly important tools to train a computer. The impact of AI and ML on software development will be profound and necessitates entirely new approaches and methods of developing software.

New computing technologies are also on the horizon. Experts may agree that we are many years away from developing a universal quantum computer (UQC), a generally programmable computer combining both classical and quantum computing elements. Nevertheless, the United States cannot afford to come in second in the race to develop the first UQC. The challenge is not only confined to development of the UQC hardware, but includes developing quantum computing programming languages and software. We also need to continue to invest in new quantum-resistant technologies such as cryptography and algorithms and apply those technologies as soon as possible to protect today's data and information from tomorrow's UQC attacks.

The field of computer science continues to advance with the discovery and development of new computer architectures and designs. We have already seen the impact of new architectures such as cloud computing, GPUs (graphics processing units), low-power electronics, and Internet of Things (IoT) on computing. New architectures are being studied and developed by both industry and academia. DoD should not only continue to invest in the development of new architectures but also to invest in new methods for quicker adoption of these technologies.

Given today's challenge of cybersecurity and software assurance, R&D must continue developing more trusted computing to thwart future cyber attacks and creating abilities to execute software with assurance on untrusted networks and hardware.

DoD should invest in new approaches to software development (beyond Agile), including the use of computer-assisted programming and project management. While agile development is currently a best practice in industry, managing the software cycle is still more art form than science. New analytical approaches and next-generation management tools could significantly improve software performance and schedule predictability. The Department should fund ongoing research as well as support academic, commercial, and development community efforts to innovate the software process.

Chapter 3. Been There, ~~Done~~ Said That: Why Hasn't This Already Happened?

Probably the most dangerous phrase you could ever use in any computer installation is that dreadful one: "but we've always done it that way." That's a forbidden phrase in my office.

— *Rear Admiral Grace Hopper (1906-1992), computer programmer, presentation at MIT Lincoln Laboratory on 25 April 1985, 23m41s*

DoD and Congress have a rich history of asking experts to assess the state of DoD software capabilities and recommend how to improve them. A DoD joint task force chaired by Duffel in 1982 started its report by saying,

> Computer software has become an important component of modern weapon systems. It integrates and controls many of the hardware components and provides much of the functional capability of a weapon system. Software has been elevated to this prominent role because of its flexibility to change and relatively low replication cost when compared to hardware. It is the preferred means of adding capability to weapon systems and of reacting quickly to new enemy threats. (Report of the DoD Joint Service Task Force on Software Problems, 1982)

Indeed, this largely echoes our own views, although the scope of software has now moved well beyond weapon systems, the importance of software has increased even further, and the rate of change for software is many orders of magnitude faster, at least in the commercial world.

Five years later, a task force chaired by Fred Brooks began its executive summary as follows:

> Many previous studies have provided an abundance of valid conclusions and detailed recommendations. Most remain unimplemented. … [T]he Task Force is convinced that today's major problems with military software development are not technical problems, but management problems. (Report of the Task Force on Military Software, Defense Science Board, 1987)

This particular assessment, from over 30 years ago, referenced over 30 previous studies and is largely aligned with the assessments of more recent studies, including this one.

And finally, in its 2000 study on DoD software, Defense Science Board (DSB) Chair Craig Fields commented that,

> Numerous prior studies contain valid recommendations that could significantly and positively impact DoD software development programs. However the majority of these recommendations have not been implemented. Every effort should be made to understand the inhibitors that prevented previous recommendations. (Defense Science Board Task Force on Defense Software, 2000)

So to a large extent the problem is not that we do not know what to do, but that we simply are not doing it. In this chapter we briefly summarize some of the many reports that have come before ours and attempt to provide some understanding of why the current state of affairs in defense software is still so problematic. Using these insights, we attempt to provide some level of confidence that our recommendations might be handled differently (remembering that "hope is not a strategy").

3.1 37 Years of Prior Reports on DoD Software

The following table lists previous reports focused on improving software acquisition and practices within DoD.

Date	Org	Short title / Summary of contents
Jul'82	DoD	Joint Service Task Force on Software Problems 37 pp + 192 pp Supporting Information (SI); 4 major recommendations The opportunities and problems posed by computer software embedded in DoD weapon systems were investigated by a joint Service task force. The task force members with software experience combined existing studies with the observations of DoD project managers. The task force concluded that software represents an important opportunity in regard to the military mission. Further, it was concluded that technological excellence in software is an important factor in maintaining U.S. military superiority, but that many problems facing DoD in software endangers this superiority.
Sep'87	DSB	Task Force on Military Software 41 pp + 36 pp SI; 38 recommendations The task force reviewed current DoD initiatives in software technology and methodology, including the Ada effort, the STARS program, DARPA's Strategic Computing Initiative, the Software Engineering Institute (SEI), and a planned program in the Strategic Defense Initiative. The five initiatives were found to be uncoordinated, and the task force recommended that the Undersecretary of Defense (Acquisition) establish a formal program coordination mechanism for them. In spite of the substantial technical development needed in requirements setting, metrics and measures, tools, etc., the Task Force was convinced that the major problems with military software development were not technical problems, but management problems. The report called for no new initiatives in the development of the technology, some modest shift of focus in the technology efforts underway, but major re-examination and change of attitudes, policies, and practices concerning software acquisition.
Dec'00	DSB	Task Force on Defense Software 36 pp + 10 pp SI; 6 major recommendations The Task Force determined that the majority of problems associated with DoD software development programs are a result of undisciplined execution. Accordingly the Task Force's recommendations emphasized a back-to-the-basics approach. The Task Force also noted that numerous prior studies contain valid recommendations that could significantly and positively impact DoD software development programs. The fact that the majority of these recommendations have not been implemented should lead to efforts designed to understand the inhibitors preventing these recommendations from being enacted.
2004	RAND	Attracting the Best: How the Military Competes for Information Technology Personnel 149 pp; no explicit recommendations Burgeoning private-sector demand for IT workers, escalating private-sector pay in IT, growing military dependence on IT, and faltering military recruiting all led to a concern that military capability was vulnerable to a large shortfall in IT personnel. This report examined the supply of IT personnel compared to the military's projected future manpower requirements. It concluded that IT training and experience, augmented by enlistment bonuses and educational benefits as needed, seemed sufficient to ensure an adequate flow of new recruits into IT. However, sharp increases in military IT requirements had the potential to create difficulties.
Feb'08	NCMA	Generational Inertia: An Impediment to Innovation? 7 pp; no explicit recommendations This article cites data to the effect that approximately 50 percent of the acquisition workforce is within 5 years of retirement. Rather than being a problem, the article feels that retirement of senior contracting specialists could effectively lead to acquisition reform: "Senior contracting specialists' resistance to change and indifference to professional development is the elephant

		in the room that acquisition reformers are unwilling to acknowledge."
Mar'09	DSB	Task Force on Department of Defense Policies and Procedures for the Acquisition of Information Technology 68 pp + 2 pp dissent + 15 pp SI; 4 major recommendations with 13 subrecommendations The primary conclusion of the task force is that the conventional DoD acquisition process is too long and too cumbersome to fit the needs of the many IT systems that require continuous changes and upgrades. The task force recommended a unique acquisition system for information technology.
2010a	NRC	Achieving Effective Acquisition of Information Technology in the Department of Defense 164 pp + 16 major recommendations This study board was asked to assess the efficacy of DoD's acquisition and test and evaluation (T&E) processes as applied to IT. The study concluded that DoD is hampered by "a culture and acquisition-related practices that favor large programs, high-level oversight, and a very deliberate, serial approach to development and testing (the waterfall model)." This was contrasted with commercial firms, which have adopted agile approaches that focus on delivering smaller increments rapidly and aggregating them over time to meet capability objectives. Other approaches that run counter to commercial, agile acquisition practices include "the DoD's process-bound, high-level oversight [that] seems to make demands that cause developers to focus more on process than on product, and end-user participation often is too little and too late."
2010b	NRC	Critical Code: Software Producibility for Defense 148 pp + 15 major recommendations This study was charged to examine the nature of the national investment in software research and ways to revitalize the knowledge base needed to design, produce, and employ software-intensive systems for tomorrow's defense needs. The study notes the continued reliance by DoD on software capabilities in achieving its mission and notes that there are important areas where DoD must push the envelope beyond mainstream capability. In other areas, however, DoD benefits by adjusting its practices to conform to government and industry conventions, enabling it to exploit a broader array of more mature market offerings.
Jul'16	CRS	The Department of Defense Acquisition Workforce: Background, Analysis, and Questions for Congress 14 pp; no explicit recommendations The increase in the size of the acquisition workforce has not kept pace with increased acquisition spending, which has signified an increase not only in the workload but also in the complexity of contracting work. This report summarized four Congressional efforts aimed at enhancing the training, recruitment, and retention of acquisition personnel.
Dec'16	CNA	Independent Study of Implementation of Defense Acquisition Workforce Improvement Efforts 147 pp + 30 pp SI; 21 major recommendations This report examines the strategic planning of the Department of Defense regarding the acquisition workforce (AWF). The study found significant improvements in several areas that "not only reversed the decline in AWF capacity from the 1990s, but also reshaped the AWF by increasing the number of early and mid-career personnel."
Feb'17	SEI	DoD's Software Sustainment Study Phase I: DoD's Software Sustainment Ecosystem 101 pp; 5 major recommendations Since the time in the early 1980s when software began to be recognized as important to DoD, software sustainment has been considered a maintenance function. After almost four decades, DoD is also at a tipping point where it needs to deal with the reality that software sustainment is not about maintenance, but rather it is about continuous systems and software engineering for the life cycle to evolve the software product baseline. This report recommends

		changing that paradigm to enable the innovation needed to address a rapidly changing technology environment, specifically through investments in human capital, better performance measurement of software sustainment, and better visibility for the software portfolio.
Mar'17	BPC	Building a F.A.S.T. Force: A Flexible Personnel System for a Modern Military 82 pp + 15 pp SI; 4 major themes with 39 recommendations This study describes today's DoD personnel system as out of step with contemporary needs and issues: "the current system is typically poorly coordinated, lacks accountability, is unable to quickly obtain specialized talent, and fosters a groupthink mentality within the force." It concludes that an effective personnel system has to build a force that is adaptable to new threats as they arise and technically proficient (among other characteristics).
Feb'18	DSB	Design and Acquisition of Software for Defense Systems 28 pp + 22 pp SI; 7 (high-level) recommendations + ~32 subrecommendations The Task Force assessed best practices from commercial industry as well as successes within DoD. Commercial embrace of iterative development has benefited bottom lines and cost, schedule, and testing performance, while the Department and its defense industrial base partners are hampered by bureaucratic practices and an existing government-imposed reward system. The Task Force concluded that the Department needs to change its internal practices to encourage and incentivize new practices in its contractor base. The assessment of the Task Force is that the Department can leverage best practices of iterative development even in its mission-critical software systems.
2018	2016 NDAA	Section 809 Panel - Streamlining and Codifying Acquisition 1,275 pp; 93 recommendations The Section 809 Panel was established by Congress in the FY 2016 NDAA to address issues with the way DoD buys what it needs to equip its warfighters. The panel published an Interim Report and a three-volume Final Report, containing a total of 93 recommendations aimed at changing the overall structure and operations of defense acquisition both strategically and tactically. Some changes hold potential for immediate effect, such as those that remove unnecessary layers of approval in the many steps contracting officers and program managers must take and those that remove unnecessary and redundant reporting requirements. Other changes require a large shift in how the system operates, such as buying readily available products and services in a manner similar to the private sector and managing capabilities from a portfolio, rather than program, perspective.
Apr'19	DIB	Software Is Never Done; Refactoring the Acquisition Code for Competitive Advantage (this document) 78 pp + 207 pp SI; 4 main lines of effort, 10 primary and 0x10 additional recommendations In this report, we focus on three overarching themes: (1) speed and cycle time are the most important metrics for managing software; (2) software is made by people and for people, so digital talent matters; and (3) software is different than hardware (and not all software is the same). We provide a set of major recommendations that focus on four main lines of effort: (A) refactoring statutes, regulations, and processes specifically for software—including acquisition, development, assurance, deployment, and maintenance—to remove hardware-centric bottlenecks while providing more insight and better oversight; (B) creating and maintaining interoperable (cross-program/cross-Service) digital infrastructure to enable continuous and rapid deployment, scaling, testing, and optimization of software as an enduring capability; (C) creating new paths for digital talent and increasing the level of understanding of modern software within the acquisition workforce; and (D) changing the practice of how software is procured and developed by adopting modern software development approaches.

As the table shows, studies dating back to at least 1982 have identified software as a particular area of growing importance to DoD—and software acquisition as requiring improvement—and the frequency and urgency of such studies identifying software acquisition as a major issue requiring reform has increased markedly since 2010. Notable recent examples include the 2010 studies by the National Research Council on *Achieving Effective Acquisition of Information Technology in the Department of Defense* and *Critical Code: Software Producibility for Defense*, the 2017 study conducted by the Carnegie Mellon University Software Engineering Institute (SEI) on DoD's Software Sustainment Ecosystem, and the 2018 DSB study on *Design and Acquisition of Software for Defense Systems*.

The properties of software that contribute to its unique and growing importance to DoD are summarized in this quote from the 2010 *Critical Code* study:

> Software is uniquely unbounded and flexible, having relatively few intrinsic limits on the degree to which it can be scaled in complexity and capability. Software is an abstract and purely synthetic medium that, for the most part, lacks fundamental physical limits and natural constraints. For example, unlike physical hardware, software can be delivered and up-graded electronically and remotely, greatly facilitating rapid adaptation to changes in adversary threats, mission priorities, technology, and other aspects of the operating environment. The principal constraint is the human intellectual capacity to understand systems, to build tools to manage them, and to provide assurance—all at ever-greater levels of complexity. (*Critical Code: Software Producibility for Defense*, NRC, 2010)

Prior studies have observed that much of DoD software acquisition policy is systems- and hardware-oriented and largely does not take these unique properties into account.[6]

The lack of action on most of the software recommendations from these studies has also been a subject of perennial comment. The DSB's 2000 study noted this phenomenon:

> [Prior] studies contained 134 recommendations, of which only a very few have been implemented. Most all of the recommendations remain valid today and many could significantly and positively impact DoD software development capability. The DoD's failure to implement these recommendations is most disturbing and is perhaps the most relevant finding of the Task Force. Clearly, there are inhibitors within the DoD to adopting the recommended changes. (Task Force on Defense Software, Defense Science Board, 2000)

The situation has not changed significantly since then despite additional studies and significant numbers of new recommendations. There is little to suggest that the inhibitors to good software practice have changed since 2000, and it is likely that the pace of technological change and addition of new capabilities provided by software have only increased since then.

Major categories of prior recommendations. The SWAP study team conducted a literature review of prior work on DoD software acquisition and extracted the specific recommendations that had been made, binning them according to major topics. The focus of the effort was on recent studies, with the bulk of the work since 2010, resulting in 139 recommendations that were extracted and categorized.

[6] For example, "DoD's Software Sustainment Study Phase I: DoD's Software Sustainment Ecosystem," SEI, 2017.

A few prevailing themes stood out from this body of work, representing issues that were commented upon in multiple studies:

- Contracts: contracts should be modular and flexible.
- Test and evaluation: test and evaluation (T&E) should be incorporated throughout the software process with close user engagement.
- Workforce: software acquisition requires specific skills and knowledge along with user interaction and senior leadership support.
- Requirements: requirements should be reasonable and prioritized; X (the focus of each report) should advocate for the need to move from compliance-based, overly prescriptive requirements to more iterative approaches.
- Acquisition strategy/oversight: DoD should encourage agencies to pursue business process innovations.
- Software process: the Department should adopt spiral/agile development approaches to reduce cost, risk, and time.

The three areas that were dealt with most often in the prior studies were acquisition oversight, contracting, and workforce. These three topics alone accounted for 60 percent of all of the recommendations we compiled. We summarize the major recurring prior recommendations in each of those areas as follows:

Recommendations from recent work in acquisition oversight:

- Ensure non-interruption of funding of programs that are successfully executing to objective (rather than budget), while insulating programs from unfunded mandates.
- Ensure that durations be reasonably short and meaningful and allow for discrete progress measurement.
- Design the overall technology maturity assessment strategy for the program or project.
- Encourage program managers to share bad news, and encourage collaboration and communication.
- Require program managers to stay with a project to its end.
- Empower program managers to make decisions on the direction of the program and to resolve problems and implement solutions.
- Follow an evolutionary path toward meeting mission needs rather than attempting to satisfy all needs in a single step.

Recommendations from recent work in contracting:

- Requests for proposals (RFPs) for acquisition programs entering risk reduction and full development should specify the basic elements of the software framework supporting the software factory, including code and document repositories, test infrastructure, software tools, check-in notes, code provenance, and reference and working documents informing development, test, and deployment.

- Establish a common list of source selection criteria for evaluating software factories for use throughout the Department.
- Contracting Officers (KOs) must function as strategic partners tightly integrated into the program office, rather than operate as a separate organization that simply processes the contract paperwork.
- Develop and maintain core competencies in diverse acquisition approaches and increase the use of venture capital–type acquisitions such as Small Business Innovative Research (SBIR), Advanced Concept Technology Development (ACTD), and Other Transaction Authority (OTA) as mechanisms to draw in nontraditional companies.

Recommendations from recent work on workforce issues:

- Service acquisition commands need to develop workforce competency and a deep familiarity with current software development techniques.
- The different acquisition phases require different types of leaders. The early phases call for visionary innovators who can explore the full opportunity space and engage in intuitive decision making. The development and production phases demand a more pragmatic orchestrator to execute the designs and strategies via collaboration and consensus decisions.
- U.S. Special Operations Command (USSOCOM) must develop a unique organizational culture that possesses the attributes of responsiveness, innovation, and problem solving necessary to convert strategic disadvantage into strategic advantage.
- Encourage employees to study statutes and regulations and explore innovative and alternative approaches that meet the statutory and regulatory intent.
- Rapid acquisition succeeds when senior leaders are involved in ensuring that programs are able to overcome the inevitable hurdles that arise during acquisition, and empower those responsible with achieving the right outcome with the authority to get the job done while minimizing the layers in between.

To help illustrate the continuity of the history of these issues and the lack of progress despite consistent, repeated similar findings, we consider the case of recommendations related to software capabilities of the acquisition workforce (areas where we are also recommending change).

Calls to improve DoD's ability to include software expertise in its workforce have a long history. DoD studies dating back to 1982 have raised concerns about the technical competencies and size of DoD's software workforce [DSB'82, DSB'87]. In 1993, the DoD Acquisition Management Board identified a need to review the DoD's software acquisition management education and training curricula. This study concluded that no existing DoD workforce functional management group was responsible for the software competencies needed in the workforce and that software acquisition competencies were needed in many different acquisition career fields. However, the Board asserted that no new career field was needed for Software Acquisition Managers.

In 2001, the same concerns regarding the software competencies of the DoD acquisition workforce once again surfaced. The DoD Software Intensive Systems Group conducted a software education and training survey of the acquisition workforce.[7] This survey demonstrated that less than 20 percent of the ACAT program staff had taken the basic Software Acquisition Management course (SAM 101) and that less than 20 percent of the ACAT program staff had degrees in computer science, software engineering, or information technology. The specific recommendations from this analysis included (1) instituting mandatory software-intensive systems training for the workforce; (2) developing a graduate-level program for software systems development and acquisition; and (3) requiring ACAT 1 programs to identify a chief software/ systems architect.

A year later, Congress mandated that the Secretary of each military department establish a program to improve the software acquisition processes of that military department.[8] Subsequently each Service established a strategic software improvement program (Army 2002, Air Force 2004, and Navy 2006). These Service initiatives have continued at some level. However, with the sunsetting of the Software Intensive Systems Group at the Office of the Secretary of Defense (OSD) level, the enterprise focus on software waned. During this same period, the Navy started the Software Process Improvement Initiative (SPII), which identified issues preventing software-intensive projects from meeting schedule, cost, and performance goals. This initiative highlighted the lack of adequately educated and trained software acquisition professionals and systems engineers.

In 2007, OSD issued guidance to create the Software Acquisition Training and Education Working Group (SATEWG) with a charter to affirm required software competencies, identify gaps in Defense Acquisition Workforce Improvement Act (DAWIA) career fields, and develop a plan to address those gaps. This group was composed of representatives from the Services, OSD, and other organizations, including the SEI. The group developed a software competency framework that identified four key knowledge areas and 29 competencies that could inform the different acquisition workforce managers about the software competencies to be integrated into their existing career field competency models. There has been no follow-on effort to evaluate the progress of the SATEWG or its outcomes.

Today, in the absence of a DoD-wide approach to describing, managing, and setting goals against a common understanding of needed software skills, each Service (as well as each software sustainment organization) has evolved its own approach or model for identifying software competencies for its workforce.

This historical context highlights two key points. First, DoD has long recognized the challenges of addressing the technical competencies and size of the software workforce across the life cycle. However, there is limited evidence of the outcomes from these different efforts. Second, this history clearly indicates that acquiring software human capital and equipping that workforce with the necessary competencies are persistent and dynamic challenges that demand a continuous enterprise strategy.

---

[7] Dennis Goldenson, & Matthew Fisher, *Improving the Acquisition of Software Intensive Systems* (CMU/SEI-2000-TR-003), (Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000), http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5171.
8 Public Law 107-314, Section 804, 2 December 2002, https://www.govinfo.gov/content/pkg/PLAW-107publ314/html/PLAW-107publ314.htm.

3.2 Breaking the Spell: Why Nothing Happened Before, but Why This Time Could Be Different

Given the long and profound history of inaction on past studies, we have attempted to create our own "Theory of (Non)Change." Why does the Department struggle to step up to rational, generally agreed-upon change? We offer the following three drivers:

The (Patriotic and Dutifully) Frozen Middle. Our process in executing this study has been to talk to anyone and everyone we could within various departments of DoD and the Services, to gather as many different perspectives as possible on what is needed, and to find out what is working and what needs to be stomped upon. As with many change management opportunities, we find significant top-down support for what we are trying to do, especially from those who see the immediate need for more, better, faster mission capability and those at the command level who are directly frustrated by the current processes that are just not working. At the other end, we see digital natives demanding change but with limited power to make it happen—people who are fully enmeshed in how the tech world works, people who have all the expectations that have been created by their private-sector lifestyle and economy. And then we have *the middle,* who are dutifully following the rules and have been trained and had success defined for a different world. For *the middle,* new methodologies and approaches introduce unknown risks, while the old acquisition and development approaches built the world's best military. We question neither the integrity nor the patriotism of this group. They are simply not incentivized to the way we believe modern software should be acquired and implemented, and the enormous inertia they represent is a profound barrier to change.

Unrequited Congress. Congress is responsible for approving and overseeing DoD's development programs. While it is clear that Congress takes its oversight role seriously, it does so knowing that to have oversight requires something to oversee, and it understands its fundamental responsibility is to enable the Department to execute its mission. But oversight matters, and recommendations for change that do not also provide insight into how new ways of doing things will allow Congress to perform its role are a very tough sell. In addition, there is a sense of unrequited return from past changes and legislation such as Other Transaction Authorities (OTAs), pilot programs, and special hiring authorities. In many cases, Congress believes it has already provided the tools and flexibilities for which DoD has asked. It is perhaps unreasonable to expect a positive response to ask for more when current opportunities have not been fully exploited.

Optimized Acquisition (for Something Else!).

> *Knowing was a barrier which prevented learning. — Frank Herbert*

While some may (justifiably) argue that the current acquisition system is not optimized for anything, it is the product of decades of rules upon rules, designed to speak to each and every edge case that might crop up in the delivery of decades-long hardware systems, holds risk elimination at a premium, and has a vast cadre of dedicated practitioners exquisitely trained to prosper within that system. This is a massive barrier to change and informs our recommendations that argue for major new ways of acquiring software and not just attempt to re-optimize to a different local maximum.

What we are trying to do that we think is different. Given the long history of DoD and Congressional reports that make recommendations that are not implemented, why do we think that this report will be any different? Our approach has been to focus not on the report and its recommendations *per se*, but rather on the series of discussions around the ideas in this report and the people we have interacted with inside the Pentagon and at program site visits. The recommendations in this report thus serve primarily as documentation of a sequence of iterative conversations, and the real work of the study is the engagements before and after the report is released.

We also believe that there are some ideas in the report that, while articulated in many places in different ways, are emphasized differently here. In particular, a key point of focus in this report is the use of speed and cycle time as the key drivers for must change and the need to optimize statutes, regulations, and processes to allow management and oversight of software. We believe that optimizing for the speed at which software can be utilized for competitive advantage will create an acquisition system that is much better able to provide security, insight, and scale.

Finally, we have tried to make this report shorter and pithier than previous reports, so we hope people will read it. It also is staged so that each reader, with his or her specific levels of authority and responsibility, can navigate an efficient path to reaching his or her own conclusions on how best to support what is contained here.

3.3 Consequences of Inaction: Increasing Our Attack Surface and Shifting Risk to the Warfighter

So what happens if history does, in fact, repeat itself and we again fail to step up to the changes that have been so clearly articulated for so long? Certainly by continuing to follow acquisition processes designed to limit risk for the hardware age, we will not reduce risk but instead will simply transfer that risk to the worst possible place—the warfighter who most needs the tools in her arsenal to deliver the missions we ask her to perform. But in addition, as we have continually stressed throughout this study, there are several real differences in today's world compared to the environment in which past efforts were made.

First, and most important, weapon systems, and the bulk of the operational structure on which DoD executes its mission, are now fundamentally software (or software-defined) systems, and as such, delays in implementing change amplify the capability gaps that slow, poor, or unsupportable software creates. Second, the astonishing growth of the tech sector has created a very different competitive environment for the talent most needed to meet DoD's needs. Decades ago, DoD was the leading edge of the world's coolest technology, and passionate, skilled software specialists jumped at the chance to be at that edge. That is simply not the case today, and while a commitment to national security is a strong motivator, if the changes recommended in this study are not implemented, the competitive war for talent, *within our country,* will be lost.

The modern software methodologies enumerated in this report—and the recommendations concerning culture, regulation and statute, and career trajectories that enable those methodologies—are the best path to providing secure, effective, and efficient software to users.

Cyber assurance, resilience, and relevance are all delivered much more effectively when done quickly and incrementally, using the tools and methods recommended in this study.

Finally we call attention back to Section 1.4 (What are the challenges that we face [and consequences of inaction]?). To summarize: “The long-term consequence of inaction is that our adversaries’ software capabilities can catch and surpass ours. … Our adversaries’ software capabilities are growing as ours are stagnating.”

Chapter 4. How Do We Get There from Here: Three Paths for Moving Forward

The history of technology is the story of man and tool-hand and mind-working together. If the hardware is faulty or if the software is deficient, the sounds that emerge will be discordant; but when man and machine work together, they can make some beautiful music.

— Melvin Kranzberg, Technology and History: Kranzberg's Laws, (Technology and Culture, 27[3]:1986), 558

The previous three chapters provided the rationale for why we need to *do* (not just say) something different about how DoD develops, procures, assures, deploys, and continuously improves software in support of defense systems. The private sector has figured out ways to use software to accelerate their businesses and DoD should accelerate its incorporation of those techniques to its own benefit, especially in ensuring that its warfighters have the tools they need in a timely fashion to execute their missions in today's hardware-enabled, software-defined environment. In this chapter, we lay out three different paths for moving forward, each under a different set of assumptions and objectives. A list of some representative, high-level steps is provided for each path, along with a short analysis of advantages and weaknesses.

4.1 Path 1: Make the Best of What We've Got

Congress has provided DoD with substantial authority and flexibility to implement the mission of the Department. Although difficult and often inefficient, it is possible to implement the recommendations outlined in this report making use of the existing authorities and, indeed, there are already examples of the types of activities that we envision taking place across OSD and the Services. In this section, we attempt to articulate a path that builds on these successes and does not require any change in the law nor major changes in regulatory structure. The primary steps required to implement this path should focus on changing the practices by which software is developed, procured, assured, and deployed as well as updating some of the regulations and processes to facilitate cultural and operational changes.

To embark on this first path, DoD should streamline its processes, allowing more rapid procurement, deployment, and updating of software. OSD and the Services should also work together to allow better cross-service and pre-certified Authorization to Operate (ATO), easier access to large-scale cloud computing, and use of modern toolchains that will benefit the entire software ecosystem. The acquisition workforce, both within OSD and the Services, should be provided with better training and insight on modern software development (one of the more frequent recommendations over the past 37 years) so that they can take advantage of the approaches that software allows that are different than hardware. Most importantly, government and industry must come together to implement a DevSecOps culture and approach to software, building on practices that are already known and used in industry.

The following list provides a summary of high-level steps that require changes to DoD culture and processes, but could be taken with no change in current law and relatively minor changes to existing regulations:

- Make use of existing authorities such as OTAs and mid-tier acquisition (Sec 804) to implement a DevSecOps approach to acquisition to the greatest extent possible under existing statutes, regulations, and processes.
- Require cost assessment and performance estimates for software programs (and software components of larger programs) to be based on metrics that track speed and cycle time, security, code quality, and useful capability delivered to end users.
- Create a mechanism for ATO reciprocity between Services and industrial base companies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapons) systems, and Services.
- Remove obstacles to DoD usage of cloud computing on commercial platforms, including Defense Information System Agency (DISA) cloud access point (CAP) limits, lack of ATO reciprocity, and access to modern software development tools.
- Expand the use of (specialized) training programs for chief information officers (CIOs), Service acquisition executives (SAEs), program executive officers (PEOs), and program managers (PMs) that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software.
- Increase the knowledge, expertise, and flexibility in program offices related to modern software development practices to improve the ability of program offices to take advantage of software-centric approaches to acquisition.
- Require access to source code, software frameworks, and development toolchains, with appropriate intellectual property (IP) rights, for all DoD-specific code, enabling full security testing and rebuilding of binaries from source.
- Create and use automatically generated, continuously available metrics that emphasize speed, cycle time, security, and code quality to assess, manage, and terminate software programs (and software components of hardware programs).
- Shift the approach for acquisition (and development) of software (and software-intensive components of larger programs) to an iterative approach: start small, be iterative, and build on success—or be terminated quickly.
- Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough.
- Shift from a list of requirements for software to a list of desired features and required interfaces/characteristics to avoid requirements creep or overly ambitious requirements.
- Maintain an active research portfolio into next-generation software methodologies and tools, including the integration of ML and AI into software development, cost estimation, security vulnerabilities, and related areas.
- Invest in transition of emerging approaches from academia and industry to creating, analysis, verification, and testing of software into DoD practice (via pilots, field tests, and other mechanisms).

- Automatically collect all data from DoD weapon systems and make the data available for machine learning (via federated, secured enclaves, not a centralized repository).
- Mandate a full program review within the first 6–12 months of development to determine if a program is on track, requires corrective action, or deserves cancellation.

This path has the advantage that the authorities required to undertake it are already in place and the expertise exists within the Department to begin moving forward. We believe that the there is strong support for these activities at the top and bottom of the system, and several groups (e.g., the Defense Digital Service [DDS], the Joint Improvised Threat Defeat Organization [JIDO], and Kessel Run) have demonstrated that the flexibilities exist within the current system to develop, procure, assure, deploy, and update software more quickly. The difficulty in this path is that it requires individuals to figure out how to go beyond the default approaches that are built into the current acquisition system. Current statutes, regulations, and processes are very complicated; there is a "culture of no" that must be overcome; and hence using the authorities that are available requires substantial time, effort, and risk (to one's career, if not successful). The risk in pursuing this path is that change occurs too slowly or not at scale, and we are left with old software that is vulnerable and cannot serve our needs. Our adversaries have the same opportunities that we do for taking advantage of software and may be able to move more quickly if the current system is left in place.

4.2 Path 2: Tune the Defense Acquisition System to Optimize for Software

While the first steps to refactoring the defense acquisition system can be taken without necessarily having to change regulations, the reality of the current situation is that Congress and DoD have created a massive "spaghetti code" of laws and regulations that are simply slowing things down. This might be OK for some types of long-development, long-duration hardware, but as we have articulated in the previous three chapters it is definitely not OK for (most types of) software.

This path takes a more active approach to modifying the acquisition system for software by identifying those statutes, regulations, and processes that are creating the worst bottlenecks and modifying them to allow for faster delivery of software to the field. We see this path as one of removing old pieces of code (statutory, regulatory, or process) that are no longer needed or that should not be applied to software, as well as increasing the expertise in how modern software development works so that software programs (and software-centric elements of larger programs) can be optimized for speed and cycle time.

The following list provides a set of high-level steps that require some additional changes to DoD culture and process, but also modest changes in current law and existing regulations. These steps build on the steps listed in path 1 above, although in some cases they can solve the problems that the previous actions were trying to work around.

- Refactor and simplify Title 10 and the defense acquisition system to remove all statutory, regulatory, and procedural requirements that generate delays for acquisition, development, and fielding of software while adding requirements for continuous (automated) reporting of cost, performance (against updated metrics), and schedule.

- Create streamlined authorization and appropriation processes for defense business systems (DBS) that use commercially available products with minimal (source code) modification.
- Plan, budget, fund, and manage software development as an enduring capability that crosses program elements and funding categories, removing cost and schedule triggers that force categorization into hardware-oriented regulations and processes.
- Replace the Joint Capabilities Integration and Development System (JCIDS), the Planning, Programming, Budgeting and Execution (PPB&E) process, and the Defense Federal Acquisition Regulation Supplement (DFARS) with a portfolio management approach to software programs, assigned to "PEO Digital" or an equivalent office in each Service that uses direct identification of warfighter needs to decide on allocation priorities.
- Create, implement, support, and require a fully automatable approach to T&E, including security, that allows high-confidence distribution of software to the field on an iterative basis (with frequency dependent on type of software, but targeting cycle times measured in weeks).
- Prioritize secure, iterative, collaborative development for selection and execution of all new software programs (and software components of hardware programs) (see DIB's Detecting Agile BS as an initial view of how to evaluate capability).
- For any software developed for DoD, require that software development be separated from hardware in a manner that allows new entrants to bid for software elements of the program on the basis of demonstrated capability.
- Shift from certification of executables, to certification of code, to certification of the development, integration, and deployment toolchain, with the goal of enabling rapid fielding of mission-critical code at high levels of information assurance.
- Require CIOs, SAEs, PEOs, PMs, and any other acquisition roles involving software development as part of the program to have prior experience in software development.
- Restructure the approach to recruiting software developers to assume that the average tenure of a talented engineer will be 2–4 years, and make better use of highly qualified experts (HQEs), intergovernmental personnel act employees (IPAs), reservists, and enlisted personnel to provide organic software development capability.
- Establish a Combat Digital Service (CDS) unit within each Combatant Command (COCOM) consisting of software development talent that can be used to manage Command-specific IT assets, at the discretion of the combatant commander. DDS, operating at the OSD level, is a good model for what a CDS can do for each COCOM.

Pursuing this path will allow faster updates to software and will improve security and oversight (via increased insight). In many cases, the Department is already executing some of the actions required to enable this path. The weakness in this path is that software would generally use the same basic approach to acquisition as hardware, with various carve-outs and exceptions. This approach runs the risk that software programs still move too slowly due to the large number of people who have to say yes and the need to train a very large acquisition force to understand how software is different than hardware (and not all software is the same).

4.3 Path 3: A New Acquisition Pathway and Appropriations Category for Software to Force Change in the Middle

The final path is the most difficult and will require dozens of independent groups to agree on a common direction, approach, and set of actions. At the end of this path lies a new defense acquisition system that is optimized for software-centric systems instead of hardware-centric systems and that prioritizes security, speed, and cycle time over cost, schedule, and (rigid) requirements.

To undertake this path, Congress and OSD must write new statutes and regulations for software, providing increased (and automation-enabled) insight to reduce the risk of slow, costly, and overgrown programs and enabling rapid deployment and continuous improvement of software to the field. Laws will have to be changed, and management and oversight will have to be reinvented, focusing on different measures and a quicker cadence. OSD and the Services will need to create and maintain interoperable (cross-program/cross-Service) digital infrastructure that enables rapid deployment, scaling, testing, and optimization of software as an enduring capability; manage it using modern development methods; and eliminate the existing hardware-centric regulations and other barriers for software (and software-intensive) programs. Finally, the Services will need to establish software development as a high-visibility, high-priority career track with specialized recruiting, education, promotion, organization, incentives, and salary.

The following list of high-level steps are required to pursue this path, builds on the steps listed in the previous paths:

- Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics.
- Create a new appropriations category that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&E, production, and sustainment.
- Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors.
- Plan and fund computing hardware (of all types) as consumable resources, with continuous refresh and upgrades to the most recent, most secure operating system and platform components.
- Create software development groups in each Service consisting of military and/or civilian personnel who write code that is used in the field, and track individuals who serve in these groups for future DoD leadership roles.

This path attempts to solve the longstanding issues with software by creating an acquisition pathway and an appropriations category that are fine-tuned for software. It will require a very large effort to get the regulations, processes, and people in place that are required to execute it effectively, and there will be missteps along the way that generate controversy and unwanted publicity. In addition, it will likely be opposed by those currently in control of selling or making software for DoD, since it will require that they retool their business to a very new approach that is not well defined at the outset. But if successful, this path has the potential to enable DoD to develop, procure, assure, deploy, and continuously improve software at a pace that is relevant for modern missions and builds on the substantial success of the U.S. private sector.

Chapter 5. What Would the DIB Do: Recommendations for Congress and DoD

It takes a lot of hard work to make something simple, to truly understand the underlying challenges and come up with elegant solutions.

— *Steve Jobs as quoted by Walter Isaacson, "How Steve Jobs' Love of Simplicity Fueled a Design Revolution," (Smithsonian Magazine, September 2012)*

In this final chapter we lay out our recommendations for what Congress and DoD should do to implement the type of software acquisition and practices reform that we believe is needed for the future. Our recommendations are organized according to four lines of effort, each of which bring together different parts of the defense ecosystem as stakeholders:

A. Congress and OSD should refactor statutes, regulations, and processes for software
B. OSD and the Services should create and maintain cross-program/cross-Service digital infrastructure
C. The Services and OSD should create new paths for digital talent (especially *internal* talent)
D. DoD and industry must change the practice of how software is procured and developed

For each of these lines of effort, we have identified the 2–3 most important recommendations that we believe Congress and DoD should undertake. These "Top Ten" primary recommendations were chosen not because they solve the entire problem but because they will make the biggest difference; without them, substantial change is not likely. In addition, we have identified 16 additional recommendations for consideration once the execution of the first 10 recommendations is successfully underway. For each recommendation, a draft implementation plan is provided in Appendix A that gives a list of actions that can be used to implement the recommendation, as well as more detail on the rationale, supporting information, and similar recommendations from other studies. Potential legislative and regulatory language to implement selected recommendations is included in Appendix B. While we have tried hard to provide specific actions, owners, and target dates that will drive an implementation plan for each recommendation, we recognize that in the end, owners will be decided by the Department's response to our study and owners will use our actions as a starting point to their own implementation plans.

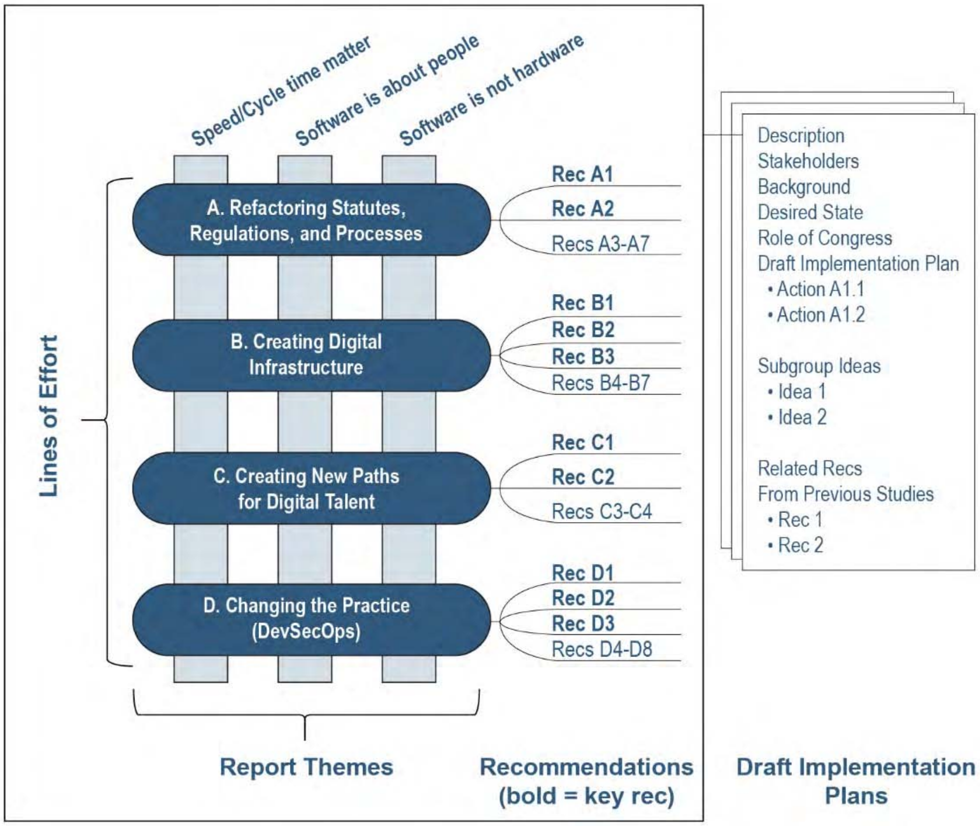


Figure 5.1 Recommendation structure. For each line of effort, a set of primary recommendations (bold) is provided, along with a set of additional recommendations for consideration. Each recommendation contains a draft implementation plan that includes background information on the rationale, vision, and stakeholders.

5.1 The Ten Most Important Things to Do (Starting Now!)

In this section we lay out what we believe are the most important steps for Congress and DoD to take to fully leverage the opportunities presented by software and the private sector's strength in modern development practices. Our commitment to these steps will directly impact the Department's ability to achieve the 2018 National Defense Strategy[9] goals of increased lethality, stronger alliances while positioning for new partnerships, and reformed business practices for better performance and affordability.

[9] U.S. Department of Defense, *Summary of the 2018 National Defense Strategy: Strengthening the American Military's Competitive Edge*, (Washington, DC: U.S. Department of Defense), https://dod.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf.

Line of Effort A. Congress and OSD should refactor statutes, regulations, and processes for software, providing increased insight to reduce the risk of slow, costly, and overgrown programs and enabling rapid deployment and continuous improvement of software to the field. Reinvent management and oversight, focusing on different measures and a quicker cadence.

Figure 5.2. The West Front of the U.S. Capitol. [Photo by Architect of the Capitol]

Recommendation A1. Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics

Current law, regulation, policy, and internal DoD processes make DevSecOps-based software development extremely difficult, requiring substantial and consistent senior leadership involvement. Consequently, DoD is challenged in its ability to scale DevSecOps software development practices to meet mission needs. The desired state is that programs have the ability to rapidly field and iterate new functionality in a secure manner, with continuous oversight based on automated reporting and analytics, and utilize IA-accredited commercial development tools.

Implementation of this recommendation could be accomplished by having USD(A&S), in coordination with USD(C) and Cost Assessment and Program Evaluation (CAPE), submit a legislative proposal using Sec 805 to propose new acquisition pathways for two or more classes of software (e.g., application, embedded), optimized for DevSecOps, for approval by the House and Senate Armed Services Committees. A draft of such language, in response to 2016 NDAA Section 805, is included in Appendix B. If approved, USD(A&S) could develop and issue a Directive-Type Memorandum (DTM) for new software acquisition pathways, and the SAEs could issue Service-level guidance for new acquisition pathways. USD(A&S), with SAEs, should select an initial set of programs that are using DevSecOps to convert to or utilize the new software acquisition pathways at the same time as developing and implementing training at Defense Acquisition University (DAU) on new software acquisition pathways for all acquisition communities (FM, Costing, PM, IT, SE, etc.). As the pathways become better understood, the DTM can be converted to a DoD Instruction (5000.SW?), incorporating lessons learned during initial program implementation.

This recommendation is supported by the ideas for change listed by the Acquisition & Strategy subgroup and is aligned with the recommendations of the 1987 and 2009 DSB studies.

Recommendation A2. Create a new appropriation category for software capability delivery that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&E, production, and sustainment

Current law, regulation, and policy treat software acquisition as a series of discrete sequential steps; accounting guidance treats software as a depreciating asset. These processes are at odds with software being continuously updated to add new functionality and create significant delays in fielding user-needed capability. The desired state is the establishment of a new appropriation (major force program category) so that programs are better able to prioritize how effort is spent on new capabilities versus fixing bugs/vulnerabilities, improving existing capabilities, etc. Such prioritization can be made based on warfighter/user needs, changing mission profiles, and other external drivers, not constrained by available sources of funding.

Implementation of this recommendation could be accomplished by having USD(A&S) submit a legislative proposal to create a new appropriations category for software and software-intensive programs for approval by the House and Senate Armed Services Committees and funding by the House and Senate Appropriations Committees. A draft of such language, linked to the acquisition pathway described in Recommendation A1, is included in Appendix B. The DoD Comptroller, working with CAPE, would need to make necessary modifications in supporting PPB&E systems to allow use and tracking of the new software appropriation. USD(A&S), in coordination with the SAEs, should select the initial programs that will use the new software appropriation from among those that are currently using DevSecOps-compatible development approaches. Budget exhibits for the new software appropriation, replacing the current P-Forms and R-Forms, should be prepared by USD(A&S) working with USD(C), CAPE, and the Appropriations Committees, and those programs selected to use the new appropriation category should begin using the exhibits upon selection into the category (see Appendix C). Finally, the Federal Accounting Standards Advisory Board in coordination with USD(A&S) and USD(C) will need to change the audit treatment of software for this category to : (1) create a separate category for software instead of characterizing software as property, plant, and equipment; (2) establish a default setting that software is an expense, not an investment; and (3) ensure that “sustainment” is an integrated part of the software life cycle.

This recommendation builds on the recommendations in the DIB’s Ten Commandments of Software (at Appendix E) and our Visit Observations and Recommendations that budgets for software (and software-intensive) programs should support the full, iterative life cycle of the software. In addition, the Acquisition & Strategy, Appropriations, Contracting, and Sustainment & Modernization subgroups all had recommendations that support this approach. The basic approach advocated here was also articulated in the 1987 DSB task force on military software and Government Accountability Office (GAO) studies in 2015 and 2017, and is consistent with the Portfolio Management Framework Recommendations 41 and 42 of the Section 809 Panel.

Line of Effort B. OSD and the Services should create and maintain cross-program/ cross-Service digital infrastructure that enables rapid deployment, scaling, and optimization of software as an enduring capability, managed using modern development methods in place of existing (hardware-centric) regulations and providing more insight (and hence better oversight) for software-intensive programs.

Figure 5.3. Soldiers review the Army's Command Post Computing Environment, a software system that consolidates tools, programs, and tasks into an integrated, interoperable, and cybersecure computing infrastructure framework. [U.S. Army photo by Dan Lafontaine, PEO C3T]

Recommendation B1. Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors

Currently, each DoD program develops its own development and test environments, which requires redundant definition and provisioning, replicated assurance (including cyber), and extended lead times to deploy capability. Small companies have difficulties providing software solutions to DoD because those software and development test environments are not available outside the incumbent contractor or they have to build (and certify) unique infrastructure from scratch. The desired state is that defense programs will have access to, and be stakeholders in, a cross-program, modern digital infrastructure that can benefit from centralized support and provisioning to lower overall costs and the burden for each program. Development infrastructure supporting continuous integration/continuous delivery (CI/CD) and DevSecOps is available as best-of-breed, and government off-the-shelf (GOTS) is provided so that contractors want to use it, though DoD programs or organizations that want or need to go outside that existing infrastructure can still do so.

Recommendation B2. Create, implement, support, and use fully automatable approaches to testing and evaluation (T&E), including security, that allow high-confidence distribution of software to the field on an iterative basis

To deliver software at speed, rigorous, automated testing processes and workflows are essential. Current DoD practices and procedures often see operational test and evaluation (OT&E) as a tailgate process, sequentially after development has been completed, slowing down delivery of useful software to the field and leaving existing (potentially poorly performing and/or vulnerable) software in place. The desired state is that development systems, infrastructure, and practices are focused on continuous, automated testing by developers (with users). To the maximum extent possible, system operational testing is integrated (and automated) as part of the development cycle using data, information, and test protocols delivered as part of the development environment. Testing and evaluation/certification of COTS components occurs once (if justified), and then ATO reciprocity (Rec B3) is applied to enable use in other programs, as appropriate.

Recommendation B3. Create a mechanism for Authorization to Operate (ATO) reciprocity within and between programs, Services, and other DoD agencies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapon) systems, and Services

Current software acquisition practice emphasizes the differences among programs: perceptions around different missions, different threats, and different levels of risk tolerance mean that components, tools, and infrastructure that have been given permission to be used in one context are rarely accepted for use in another. The lack of ATO reciprocity drives each program to create its own infrastructure, repeating time- and effort-intensive activities needed to certify elements as secure for their own specific context. The desired state is that modern software components, tools, and infrastructure, once accredited as secure within the DoD, can be used appropriately and cost-effectively by multiple programs. Programs can then spend a greater percentage of their budgets on developing software that adds value to the mission rather than spending time and effort on basic software infrastructure. COTS components are accredited once and then made available for use in other programs, as appropriate.

Line of Effort C. The Services and OSD should create new paths for digital talent (especially internal talent) by establishing software development as a high-visibility, high-priority career track and increasing the level of understanding of modern software within the acquisition workforce. Increased internal capability is necessary both to allow organic (internal) development and to enable the Department to best serve as a knowledgeable partner for software acquired from commercial sources.

Figure 5.4. Airmen assigned to the 707th Communications Squadron, which supports more than 5,700 personnel around the world, update software for Air Force networks. [U.S. Navy photo by Rick Naystatt/Released]

Recommendation C1. Create software development units in each Service consisting of military and civilian personnel who develop and deploy software to the field using DevSecOps practices

DoD's capacity to apply modern technology and software practices to meet its mission is required to remain relevant in increasingly technical fighting domains, especially against peer adversaries. While DoD has both military and civilian software engineers (often associated with maintenance activities), the IT career field suffers from a lack of visibility and support. The Department has not prioritized a viable recruiting strategy for technical positions, and has no comprehensive training or development program that prepares the technical and acquisition workforce to adequately deploy modern software development tools and methodologies. The desired state is that DoD recruits, trains, and retains internal capability for software development, including by Service Members, and maintains this as a separate career track (like DoD doctors, lawyers, and musicians). Each Service has organic development units that are able to create software for specific needs and that serve as an entry point for software development capability in military and civilian roles (complementing work done by contractors). The Department's workforce embraces commercial best practices for the rapid recruitment of talented professionals, including the ability to onboard quickly and provide modern tools and training in state-of-the-art training environments. Individuals in software development career paths are able to maintain their technical skills and take on DoD leadership roles.

Recommendation C2. Expand the use of (specialized) training programs for CIOs, SAEs, PEOs, and PMs that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software

Acquisition professionals have been trained and had success in the current model, which has produced the world's best military, but this model does not serve well for software. New methodologies and approaches introduce unknown risks, and acquisition professionals are often not incentivized to make use of the authorities available to implement modern software methods. At the same time, senior leaders in DoD need to be more knowledgeable about modern software development practices so they can recognize, encourage, and champion efforts to implement modern approaches to software program management. The desired state is that senior leaders, middle management, and organic and contractor-based software developers are aligned in their view of how modern software is procured and developed. Acquisition professionals are aware of all of the authorities available for software programs and use them to provide flexibility and rapid delivery of capability to the field. Program leaders are able to assess the status of software (and software-intensive) programs and spot problems early in the development process, as well as provide continuous insight to senior leadership and Congress. Highly specialized requirements are scrutinized to avoid developing custom software when commercial offerings are available that are less expensive and more capable.

Line of Effort D. DoD and industry must change the practice of how software is procured and developed by adopting modern software development approaches, prioritizing speed as the critical metric, ensuring cybersecurity is an integrated element of the entire software life cycle, and purchasing existing commercial software whenever possible.

Figure 5.5. Connected battle command suites. [U.S. Army photo]

Recommendation D1. Require access to source code, software frameworks, and development toolchains—with appropriate IP rights—for all DoD-specific code, enabling full security testing and rebuilding of binaries from source

Source code for many DoD systems is not available to DoD for inspection and testing, and DoD relies on suppliers to write code for new compute environments. As code ages, suppliers are not required to maintain codebases without an active development contract, and "legacy" code is not continuously migrated to the latest hardware and operating systems. The desired state is that DoD has access to source code for DoD-specific software systems that it operates and uses to perform detailed (and automated) evaluation of software correctness, security, and performance, enabling more rapid deployment of both initial software releases and (most important) upgrades (patches and enhancements). DoD is able to rebuild executables from scratch for all of its systems and has the rights and ability to modify (DoD-specific) code when new conditions and features arise. Code is routinely migrated to the latest computing hardware and operating systems, and routinely scanned against currently known vulnerabilities. Modern IP language is used to ensure that the government can use, scan, rebuild, and extend purpose-built code, but contractors are able to use licensing agreements that protect any IP that they have developed with their own resources. Industry trusts DoD with its code and has appropriate IP rights for internally developed code.

Recommendation D2. Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough

Current DoD systems often rely on security-at-the-perimeter as a means of protecting code from unauthorized access. If this perimeter is breached, then a large array of systems can be compromised. Multiple reports by the GAO, the Department of Defense Office of Inspector General (DoDIG), and other agencies have identified cybersecurity as a major issue in acquisition programs. The desired future state is that DoD systems use a zero-trust security model in which it is not assumed that anyone who can gain access to a given network or system should have access to anything within that system. DoD uses regular and automated penetration testing to track down vulnerabilities, and engages red teams to attempt to breach our systems before our adversaries do.

Recommendation D3. Shift from the use of rigid lists of requirements for software programs to a list of desired features and required interfaces/characteristics to avoid requirements creep, overly ambitious requirements, and program delays

Current DoD requirements processes significantly impede its ability to implement modern software development practices by forcing programs to spend years establishing requirements and insisting on satisfaction of requirements before a project is considered "done." This impedes rapid implementation of features that are of greatest value to the user. The desired state is that rather than a list of requirements for every feature, programs should establish a minimum set of requirements required for initial operation, security, and interoperability, and place all other desired features on a list that will be implemented in priority order, with the ability for DoD to redefine priorities on a regular basis.

5.2 The Next Most Important Things to Tackle

DoD must make a large number of changes to fully realize the vision that 37 years of studies have articulated. This study solicited input from a wide range of stakeholders in the defense software enterprise, including OSD and Service leaders, industry participants in our visits and roundtables, and FFRDC personnel who helped put together our report and identify the recommendations that we should make. The list of recommendations below are the next 0x10 (16) recommendations that we believe can be implemented after actions on the 10 above are solidly underway (like software, implementing recommendations is never "done"). We list these second not because they are dependent on the primary recommendations but simply to emphasize the urgency of the Top Ten.

ID	Recommendation
A3	Require cost assessment and performance estimates for software programs (and software components of larger programs) of appropriate type be based on metrics that track speed and cycle time, security, code quality, and functionality
A4	Refactor and simplify Title 10, DFARS, and DoDI 5000.02/5000.75 to remove statutory, regulatory, and procedural requirements that generate delays for acquisition, development, and fielding of software; while adding requirements for continuous (automated) reporting of cost, performance (against updated metrics), and schedule
A5	Create streamlined authorization and appropriation processes for defense business systems (DBS) that use commercially available products with minimal (source code) modification
A6	Plan, budget, fund, and manage software development as an enduring capability that crosses program elements and funding categories, removing cost and schedule triggers associated with hardware-focused regulations and processes
A7	Replace JCIDS, PPB&E, and DFARS with a portfolio management approach to software programs, assigned to "PEO Digital" or an equivalent office in each Service that uses direct identification of warfighter needs to determine allocation priorities for software capabilities

B4	Prioritize secure, iterative, collaborative development for selection and execution of new software development programs (and software components of hardware programs), especially those using commodity hardware and operating systems
B5	Remove obstacles to DoD usage of cloud computing on commercial platforms, including DISA CAP limits, lack of ATO reciprocity, and access to modern software development tools
B6	Shift from certification of executables for low- and medium-risk deployments to certification of code/architectures and certification of the development, integration, and deployment toolchain
B7	Plan and fund computing hardware (of all appropriate types) as consumable resources, with continuous refresh and upgrades to current, secure operating systems and platform components
C3	Increase the knowledge, expertise, and flexibility in program offices related to modern software development practices to improve the ability of program offices to take advantage of software-centric approaches to acquisition
C4	Restructure the approach to recruiting digital talent to assume that the average tenure of a talented engineer will be 2–4 years, and make better use of HQEs, IPAs, special hiring authorities, reservists, and enlisted personnel to provide organic software development capability, while at the same time incentivizing and rewarding internal talent
D4	Create and use automatically generated, continuously available metrics that emphasize speed, cycle time, security, user value, and code quality to assess, manage, and terminate software programs (and software components of hardware programs)
D5	Shift the approach for acquisition and development of software (and software-intensive components of larger programs) to an iterative approach: start small, be iterative, and build on success—or be terminated quickly
D6	Maintain an active research portfolio into next-generation software methodologies and tools, including the integration of ML and AI into software development, cost estimation, security vulnerabilities, and related areas
D7	Invest in transition of emerging tools and methods from academia and industry for creating, analyzing, verifying, and testing of software into DoD practice (via pilots, field tests, and other mechanisms)
D8	Automatically collect all data from DoD national security systems, networks, and sensor systems, and make the data available for machine learning (via federated, secured enclaves, not a centralized repository).

5.3 Monitoring and Oversight of the Implementation Plan

It would be naive to believe that just listing the recommendations above will somehow ensure they are quickly and easily implemented after 37 years of previous, largely consistent recommendations have had relatively minor impact. We believe that DoD should use these recommendations (and the ones that preceded them) to create an implementation plan for review by stakeholders (including the DIB, if there is interest). This implementation plan might use as its starting point the proposed implementation plans that we have articulated in Appendix A, with agreement by the Secretary of Defense, the Undersecretaries of Defense, the Service Chiefs, CAPE, and DOT&E to support the creation and execution of the next iteration of the implementation plan.

We propose the following timeline for implementing the recommendations proposed here:

- (Immediately): Define, within 60 days after delivery of this report to Congress, a detailed implementation plan and assign owners to begin each of the top recommendations.

- FY19 (create): High-level endorsement of the vision of this report, and support for activities that are consistent with the desired end state (i.e., DevSecOps and enterprise-level architecture and infrastructure). Identify and launch programs to move out on the priority recommendations (start small, iterate quickly).
- FY20 (deploy): Initial deployment of authorities, budgets, and processes for reform of software acquisition and practices. Execute representative programs according to the main lines of effort and primary recommendations in this report. Implement these recommendations in the way we implement modern software: implement now, measure results, and modify approaches.
- FY21 (scale): Streamlined authorities, budgets, and processes enabling reform of software acquisition and practices at scale. In this time frame, adopt a new methodology to estimate as well as determine the value of software capability delivered (and not based on lines of code).
- FY22 (optimize): Conditions established so that all DoD software development projects transition (by choice) to software-enabled processes, with the talent and ecosystem in place for effective management and insight.

5.4 Kicking the Can Down the Road: Things That We Could Not Figure Out How to Fix

Despite the fairly comprehensive view that we have attempted to take in this study regarding how to improve the defense software enterprise, there are a number of challenges remaining that we were not able to address. We summarize these here for the next study (or perhaps one 37 years from now) to consider as DoD continues this path forward.

Over-oversight. DoD's sprawling software enterprise has many oversight actors, spanning Congress, OSD, Service or Component leadership, and other executive branch actors like the GAO. These actors each take frequent oversight action in attempts to improve the software in specific programs and also make well-intentioned efforts to improve the health of the overall system. However, these oversight actions focus primarily on addressing the behavior of the people developing and maintaining the software, overlooking the fact that the oversight itself is equally part of DoD's software problem. Ultimately, we cannot fix software without fixing oversight.

There are at least two categories of problems when it comes to software oversight: structural and substantive.

From a structural perspective, there are too many actors involved in oversight. A program manager, tasked with leading a software development effort, may have as many as 17 other actors who can take some form of oversight action on the program. Most of these individuals do not possess the authority to cancel a program unilaterally, but all have the ability to delay progress or create uncertainty while seeking corrective action for their concerns. These oversight actors often have overlapping or unclear roles and authorities, as well as competing interests and incentives. This means that in addition to the necessary checks and balances required between organizations, there is debate and active competition inside each of the organizations with, for example, various offices in OSD arguing among themselves in addition to arguing with Congress and the Services. Further, there is significant personnel turnover within these positions, meaning that any consensus tends to be short lived.

Substantively, the various oversight actors often do not possess a shared understanding of what constitutes good practice for software or its oversight. Further, these actors may not share a common vision for what DoD's software enterprise should look like today or in the future. The majority of oversight attention and action is placed on individual programs than on considering portfolios in the aggregate or the performance of the system as a whole. This program oversight is highly subjective in nature, relying on reports and PowerPoint slides presenting narratives and custom-created data. Worse, this oversight operates primarily according to conventional wisdom associated with the oversight of hardware programs, using decades-old heuristics when considering cost, schedule, and performance.

Without understanding what good looks like, or the right questions to ask, oversight actors risk enacting poor fixes. These actions can also be at odds with stated policy. Oversight actions are always more powerful than written policy, meaning that disparities between the two create the risk of cognitive dissonance or a shadow policy environment. Disparities also put program leadership in the unfair position of having to resolve the competing priorities of others, with the knowledge that failure to do so will lead to more blame and action from above.

Structural and substantive problems lead to oversight that is inconsistent and confusing, making it essentially impossible to systematically identify symptoms, determine root causes, or implement scalable fixes. This, in turn, allows everyone involved in DoD software development and maintenance to feel aggrieved, blame everyone other than themselves for systemic issues, and continue their behavior without reflection or change, thus perpetuating the cycle.

The approach by oversight organizations both on the Hill and in DoD should be that policy is treated as the current hypothesis for how best to ship code that DoD's users need. Through the use of data-driven governance, each program should then be tested against that policy while also being a test of the policy. The hypothesis, and policy, must be continually updated based on standard data that is recognized by, and accessible to, all oversight actors. Implementing such an approach is within the power of the oversight community but would be challenging and appears unlikely given current culture and practices. Regardless, those involved in the oversight of DoD software should not expect meaningfully improved outcomes for that software until the oversight practices used to improve that software are themselves improved.

Promotion practices. Software is disproportionately talent driven. Access to strong engineering talent is one of the most important factors that determine the success or failure of software projects. All that our rivals have to do to surpass us in national security applications of software such as AI, autonomy, or data analytics is to leverage their most talented software engineers to work on those applications. And yet in DoD, as much as we struggle to attract those with technical talent, we also struggle to elevate the talent we have.

The companies and institutions that are winning the software game recognize the importance of identifying and cultivating talented software leaders (whether they are engineers, managers, or strategists working closely with contractors) and actively promote and reward employees based on merit and demonstrated contributions. In contrast, human capital practices in DoD, sometimes by design and sometimes by habit and culture, narrowly limit how technical talent can be evaluated and often prioritize time in grade. The Department needs to figure out how to recognize when civilians and Service Members show an aptitude for software and software management and be able to promote, reward, and retain these individuals outside of the current constraints.

Using commercial software whenever possible. DoD should not build something that it can buy. If there is an 80 percent commercial solution, it is better to buy it and adjust—either the requirements or the product—rather than build it from scratch. It is generally not a good idea to over-optimize for what we view as "exceptional performance," because counter-intuitively this may be the wrong thing to optimize for as the threat environment evolves over time. Similarly, DoD should take actions to ensure that both the letter and spirit of commercial preference laws (e.g., 10 USC 2377, which requires defense agencies to give strong preference to commercial and non-developmental products) are being followed.

There is a myth that the U.S. private sector—where much of the world's software talent is concentrated—is unwilling to work on national security software. The reality is that DoD has failed to award meaningful government contracts to commercial software companies, which has generally led to companies making a *business decision* to avoid it. DoD's existing efforts to target the commercial software sector are governed by a "spray and pray" strategy, rather than by making concentrated investments.[10] DoD seems to love the idea of innovation, but does not love taking sizeable bets on new entrants or capabilities. It is interesting that Palantir and SpaceX are the only two examples since the end of the Cold War of venture-backed, DoD-focused businesses reaching multibillion dollar valuations. By contrast, China has minted around a dozen new multibillion dollar defense technology companies over the same time period. Some of these problems are purely cultural in nature and require no statutory/regulatory changes to address. Others likely will require the changes detailed in our recommendations.

That said, in many cases, there will not be an obvious "buy" option on the table. DoD and the Services should also work together to prioritize interoperable approaches to software and systems that enable rapid deployment, scaling, testing, and optimization of software as an enduring capability; manage them using modern development methods; and eliminate selected hardware-centric regulations and other particularly problematic barriers. The Services should find ways to better recognize software as a key area of expertise and provide specialized education and organizational structures that are better tuned for rapid insertion and continuous updates of software in the field and in the (back) office.

[10] While the overall funding commitments are large—$2 billion from DARPA for AI, for example—those commitments have resulted in few, if any, contracts for private companies other than traditional defense contractors. They have therefore failed to create significant incentives for the commercial tech sector to invest in government applications of AI.

Acknowledgments

The SWAP study members are indebted to a large number of individuals who helped provide valuable input, guidance, and support for the study and for the creation of this report.

We would first like to thank the SWAP study team, who coordinated the many activities associated with the study, including arranging for visits, briefings, and meetings; running the SWAP working group activities; and assisting with the production of the final report. Our initial study director, Bess Dopkeen, was detailed to the study from CAPE and provided outstanding leadership to the overall study. Her vision, energy, and knowledge of the Department were essential in establishing the interactive nature of this activity and helping us obtain insight into the many previously unknown aspects of DoD. She was succeeded by Jeff Boleng, the USD(A&S) Special Assistant for Software, who initially served as our liaison to A&S and took over as study director when Bess departed the Pentagon. Bess and Jeff were assisted by three outstanding members of the core SWAP team: Courtney Barno, Devon Hardy, and Sandra O'Dea. The study and the report could not have come together without the tireless (and patient!) efforts of Bess, Courtney, Devon, Sandy, and Jeff, who participated in every aspect of the report and helped us shape its content, style, and tone.

The SWAP study was also assisted by individuals from the Institute for Defense Analyses (IDA), SEI, and MITRE who served as our experts on the acquisition process and were invaluable in working through the detailed recommendations. Their knowledge of past studies, the acquisition regulations, the many novel approaches to acquisition reform, and the language of the acquisition community helped us better understand the challenges and opportunities for software acquisition and reform. We would particularly like to thank Kevin Garrison (IDA), Nick Guertin (SEI), Tamara Marshall-Keim (SEI), Forrest Shull (SEI), and Craig Ulsh (MITRE) for their help, encouragement, and constant advice.

A major element of the study was the participation of a large SWAP working group consisting of DoD employees who worked with Bess and the SWAP team to provide input to the study and to articulate pain points, ideas for changes, and proposed updates to legislation and regulations. A full list of individuals who participated in the working groups is listed in Appendix J, but we would particularly like to thank John Bergin, Ben FitzGerald, Bill Greenwalt, Amy Henninger, Paul Hullinger, Peter Levine, Melissa Naroski Merker, Jane Rathbun, Ed Wolski, and Philomena Zimmerman.

The Defense Innovation Board (DIB) staff were tightly linked to the SWAP study, which took place under the auspices of the Science and Technology (S&T) Committee. Josh Marcuse was instrumental in initiating the study (including identifying and hiring Bess) and providing keen insights into the report contents and recommendations. Mike Gable, Janet Boehnlein, and Christopher "Bruno" Brunett served as our designated federal officers (DFOs), accompanying us on trips, visits, and meetings and helping us uphold the Federal Advisory Committee Act (FACA) guidelines in a manner that enabled us to interact in a transparent and interactive way with members of the public, the Department, and Congress.

Many high-ranking officials within the Pentagon took the time to meet with us and provide their input, views, and encouragement for our efforts. Chief among these was Ellen Lord, Under Secretary of Defense for Acquisition & Sustainment, who provided input to our study and support for our meetings, while always being careful to help protect the independence of the study team in support of the charge from Congress. We would also like to thank Bob Daigle (CAPE), Dana Deasy (CIO), Bob Behler (DOT&E), Hondo Guerts (USN), and Will Roper (USAF) for their willingness to meet with us on multiple occasions.

Finally, we are indebted to the many individuals working on DoD programs with whom we met, both in industry and in government. On our many visits and in countless briefings, individuals who were working within the current system, and often pushing the boundaries of what is possible, gave us their honest insights and feedback. We are particularly grateful for the help we received from Tory Cuff, Leo Garciga, and CAPT Bryan Kroger, for their willingness to speak with us and help us understand what the future could look like.

SWAP Vignettes

To help illustrate some of the issues facing the Department in the area of software acquisition and practices, the SWAP study solicited a set of "vignettes" on different topics of relevance to the study. These vignettes represent "user stories" contributed by study team members and collaborators; the views expressed here do not necessarily reflect the views of the SWAP study (though they are consistent with the overarching themes contained in the report). The intent of these vignettes is to provide some additional points of view and insights that are more specific and, in some cases, more personal.

List of vignettes:

- Implementing Continuous Delivery: The JIDO Approach
- F22: DevOps on a Hardware Platform
- Making It Hard to Help: A Self-Denial of Service Attack for the SWAP Study
- DDS: Fighting the Hiring Process Instead of Our Adversaries
- Kessel Run: The Future of Defense Acquisitions Is #AgileAF
- JMS: Seven Signs Your Software (Program) Is in Trouble

Vignette 1 – Implementing Continuous Delivery: The JIDO Approach

Forrest Shull

One theme that emerges from the work in this study is that DoD certainly does have successes in terms of modern, continuous delivery of software capability; however, in too many cases, these successes are driven by heroic personalities and not supported by the surrounding acquisition ecosystem. In fact, in several cases the demands of the rest of the ecosystem cause friction that, at best, adds unnecessary overhead to the process and slows the delivery of capability. The Joint Improvised-Threat Defeat Organization (JIDO), within the Defense Threat Reduction Agency, is a compelling example.

JIDO describes itself as "the DoD's agile response mechanism, a Quick Reaction Capability (QRC) as a Service providing timely near-term solutions to the improvised threats endangering U.S. military personnel around the world."[11] As such, the speed of delivery is a key success criterion, and JIDO has made important improvements in this domain. Central to accomplishing these successes has been the adoption of a DevSecOps solution along with a continuous ATO process, which exploits the automation provided by DevSecOps to quickly assess security issues.

At least as important as the tooling are the tight connections that JIDO has enabled among the stakeholder groups that have to work together with speed to deliver capability. JIDO has personnel embedded in the user communities associated with different COCOMs, referred to as Capability Data Integrators (CDIs). These personnel are required to be familiar with the domain, familiar with the technology, and forward-leaning in terms of envisioning technical solutions to help warfighter operations. Almost all CDIs have prior military experience and are deployed in the field, moving from one group of users to another, helping to train them on the tools that are available, and at the same time understanding what they still need. CDIs have tight reachback to JIDO and are able to identify important available data that can be leveraged by software functionality and can be developed with speed through the DevSecOps pipeline.

JIDO has also focused on knocking down barriers among contractors and government personnel. JIDO finds value in relying on contractor labor that can flex and adapt as needed to the technical work, with effort spent on making sure that the mix of government personnel and multiple contractor organizations can work together as a truly integrated team. To accomplish this, JIDO has created an environment with a great deal of trust between government and contractors. There are responsibilities that are inherently governmental and tasks that can be delegated to the contractor. Finding the right mix requires experimentation, especially since finding the personnel with the right skillset on the government side is difficult.

Despite these successes at bringing together stakeholders within the JIDO team, stakeholders in the program management office (PMO) sometimes describe substantial difficulties in working with the rest of the acquisition ecosystem, since on many dimensions the Agile/DevSecOps approach does not work well with business as usual. For example, they describe instances where the Services or the Joint Chiefs push back on solutions that were created to address requirements from the field. Thanks to the CDIs, JIDO can create a technical solution that answers identified

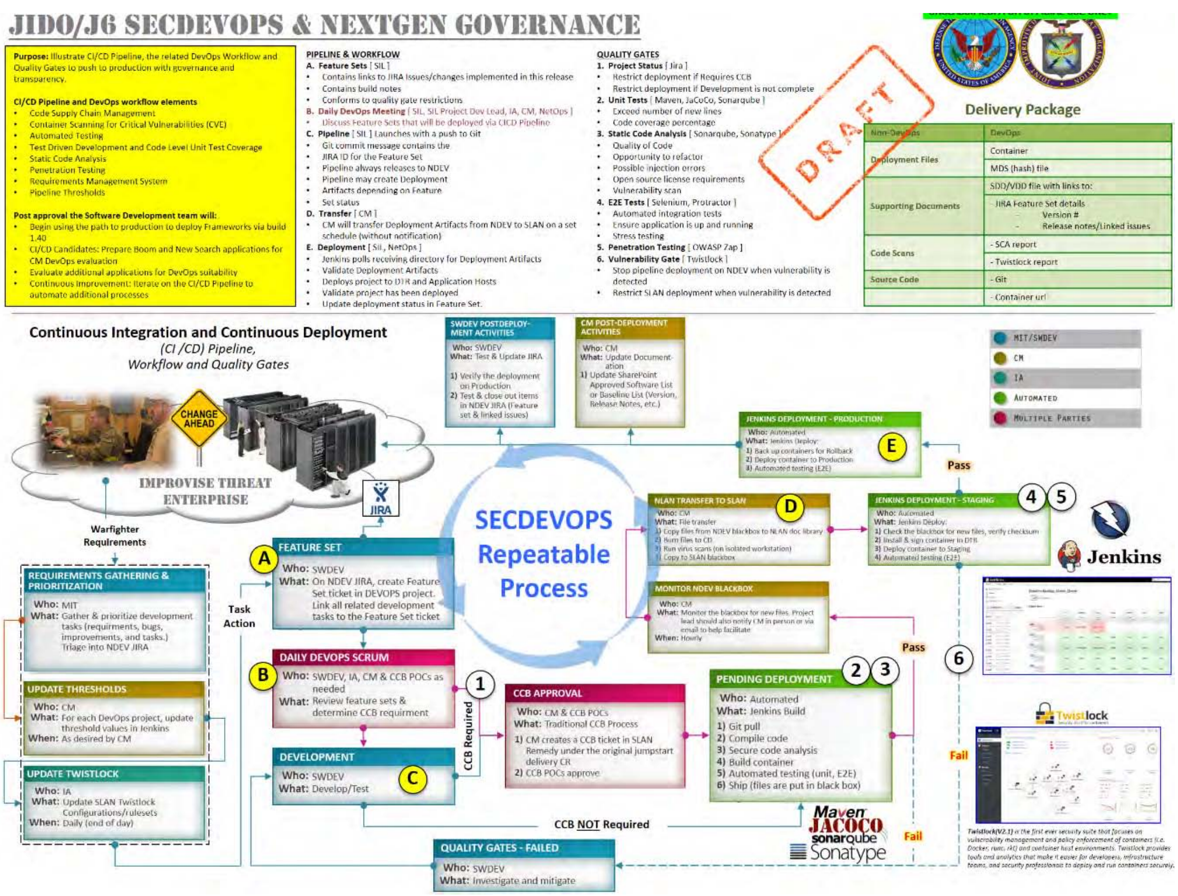
---

[11] JIDO SecDevOps Concept of Operations, v1.

requirements from warfighters in the field, but that does not mean it will get approval for deployment. There is a mismatch and potential for miscommunication when the organizations that control deployment don't own the requirements themselves.

Also, because JIDO operates in an agile paradigm in which requirements can emerge and get re-prioritized, it is difficult for the organization to justify budget requests upfront in the way that their command chain requires. JIDO addresses this today by creating notional, detailed mappings of functionality to release milestones. Since a basic principle of the approach is that capabilities being developed can be modified or re-prioritized with input from the warfighter, this predictive approach provides little or no value to the JIDO teams themselves. Even though JIDO refuses to map functionality in this way more than 2 years out, given that user needs can change significantly in that time, the program has had to add headcount just to pull these reports together.

JIDO has no problem showing value for the money spent. It is able to show numbers of users and, because it has personnel embedded with user communities, can discuss operational impact. As mentioned above, JIDO's primary performance metric is "response from the theater." Currently, JIDO faces a backlog of tasks representing additional demand for more of its services, as well as a demand for more CDIs. Despite these impactful successes, the surrounding ecosystem unfortunately provides little in the way of support and much that hinders the core mission. It is difficult to see how these practices can be replicated in other environments where they can provide positive impact, until these organizational mismatches can be resolved.



Slide image received from former DTRA-JIDO chief technology officer.

Vignette 2 – F22: DevOps on a Hardware Platform

Craig Ulsh and Maj Zachary McCarty

The F-22A Raptor program recognized a need for greater speed and agility and took action. In mid-2017, the F-22 Program Office realized the F-22A Raptor modernization efforts were not delivering at a speed that would keep pace with emerging threats. Program leadership secured the expertise of the Air Force Digital Service (AFDS). A joint team assessed the program and captured a series of observations and recommendations. The overarching assessment was:

> The Air Force must move faster, accept a greater amount of risk, and commit to radical change with how the F-22A modernization effort is managed and technology is implemented. Competitors are moving faster, and blaming poor vendor performance will not help the F-22A Raptor remain the dominant air superiority platform.

The F-22A Program Office realized that change was needed. The F-22 acquisition process, steeped in the traditional DoDI 5000 model, was slow and cumbersome, with initial retrofits taking at least 6 years to deliver. The program recognized the following symptoms:

- Requirements were static and rigidly defined.
- Capability was delivered in large, monolithic releases.
- Change was avoided and treated as a deviation from well-guarded baselines.
- The development team placed too much focus on intensive documentation.
- Separate programs with separate contracts drove inefficiencies and conflicting interests.
- Insufficient automation for incremental testing resulted in marathon test events. More specifically, the team identified a number of issues that are common among weapon systems:

Development practices. Development processes were matched to the traditional acquisition process. Large feature sets, multiple baselines, highly manual developer testing tools, and limited focus on continuous software infrastructure upgrades contributed to the slow capability delivery cycle. The team made several specific recommendations under the overarching recommendation for the software development teams to adopt modern software practices.

Planning. Several inefficiencies were identified in the planning process including lack of metrics for estimation of effort, inability to prioritize, and inefficient use of developer time. Again, the team proposed that the program adopt modern agile software processes.

Organization. Organizational gaps included poor collaboration across teams, lack of incentives for engineering talent, and competing priorities across multiple vendors.

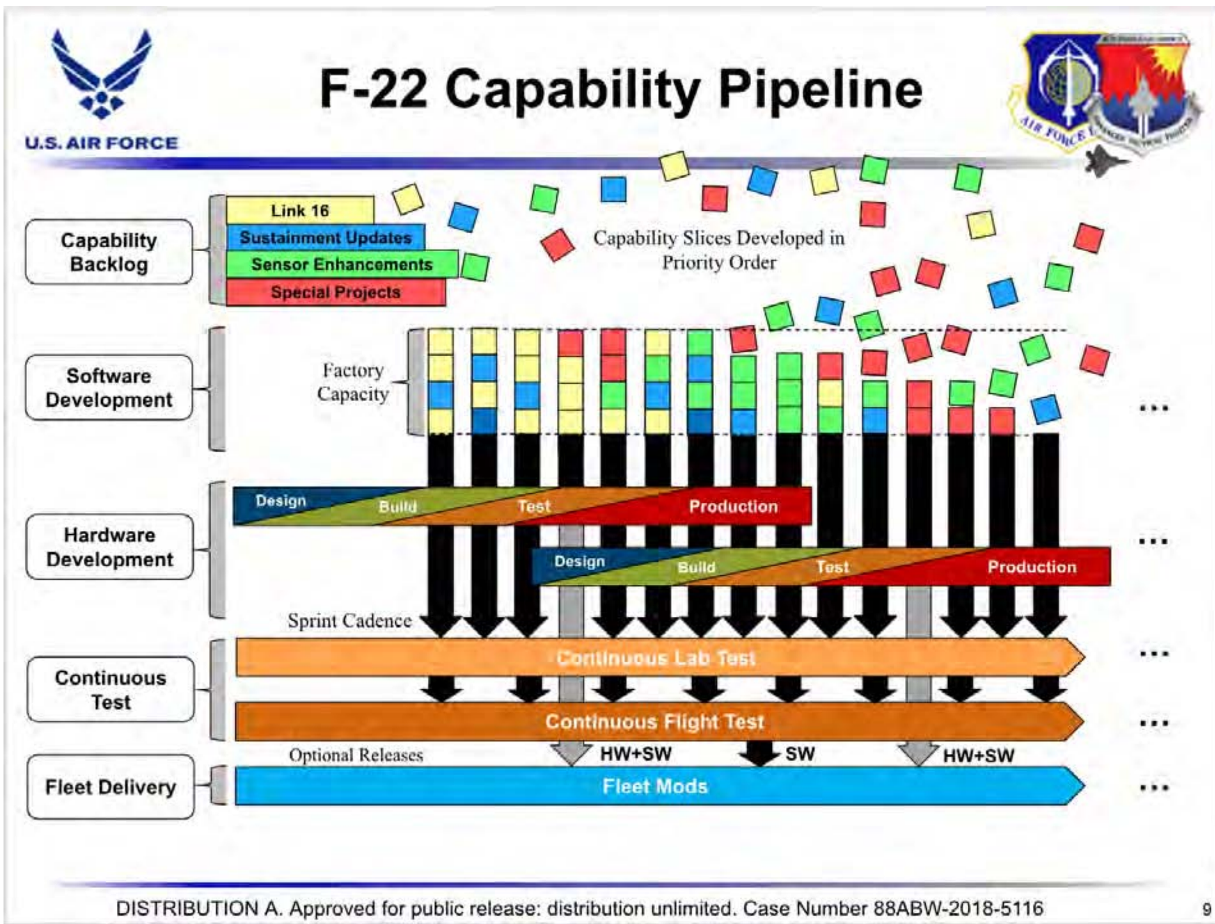
Contracts. The single most significant observation is the failure to prioritize.

In November 2017, the F-22 Program Office took several steps to accelerate the F-22A modernization efforts. In response to outdated development practices, the program office restructured TACLink 16 and TACMAN programs into a single agile development stream. To properly match the contractor effort with a new development approach, a “level of effort” for prime development labor was adopted. To address some of the planning concerns, steps were taken to adjust program alignments and authorities.

The F-22A Raptor program has made positive steps in adopting a more modern approach to both hardware and software acquisition. Perhaps the best example is a new contract structure that allows for quick reaction to emerging requirements and changing user priorities while incentivizing a long-time incumbent contractor for continuous improvement. The Program Office has learned lessons during the transition to more agile approaches, including:

- Culture change has been the biggest hurdle.
- The program must recognize and accept that things will go wrong.
- Security controls limit flexibility and communication.

The program is on the right track with a sound plan to accelerate delivery. But the program office also noted, in the immortal words of Mike Tyson, “Everyone has a plan until they get punched in the face.”



Slide image received for briefing from F22A Raptor Program Office.

Vignette 3 – Making It Hard to Help: A Self-Denial of Service Attack for the SWAP Study

Richard Murray

DoD makes use of advisory committees consisting of a mixture of government, industry, and academic experts, all trying to help. However, the Department can make it extremely difficult for these groups to function, an example of what we refer to on the Defense Innovation Board (DIB) as a "self-denial of service attack."[12] The DIB SWAP study is itself a case in point.

<rant>

The DIB Software Acquisition and Practices (SWAP) study clock started ticking when the 2018 NDAA was signed on 12 December 2017. We had our first SWAP discussion at the Pentagon on 16 January 2018, before we had officially been requested by the Under Secretary for Defense (Acquisition and Sustainment) to start, but knowing this was coming (and using the DIB Science & Technology [S&T] committee to ramp up quickly). We identified potential subcommittee members by 12 February, and we were officially charged to carry out the study on 5 April 2018. The one-year Congressionally-mandated end date was thus set as 5 April 2019. The DIB S&T subcommittee submitted the list of suggested subcommittee members. Then we started waiting…

On 24 May, after a DIB meeting, one of the SWAP co-chairs found out that there had been no movement on these positions. He sent a note to the DIB's Executive Director, expressing disappointment and reiterating the importance of getting these people on board early in the study. The Executive Director tried to use this note to push things along. More waiting…

The first activity in which any new member of the SWAP subgroup participated took place on 1 November 2018— a full 30 weeks after our 52-week countdown started and 9 months after we had identified the people whom we wanted to enlist in to help in our study. Even this took repeated interventions by the DIB staff and, in the end, only two of the four people who we hoped could help were able to participate in the study. The timing was such that we had already visited five of the six programs with which we met, written seven of the eight concept papers that we generated, and held three of the four public meetings that provided input for our report.

Why did things take so long? These people were ready to help, had served in government advisory roles in the past, and provided incredibly valuable input in the end (but only in the end). Maybe we need some sort of "FACA Pre ✓" that allows DoD to make use of people who are willing to help and all we need to do is ask.

Another example: the SWAP study decided to use Google's G Suite as the means for writing our report. It had some nice features for collaboration and several of us were familiar with using it. Setting up a G Suite site is fast and easy, and a member of the study had previously created a site in a matter of minutes and had a fully operational, two-factor authenticated set of accounts

---

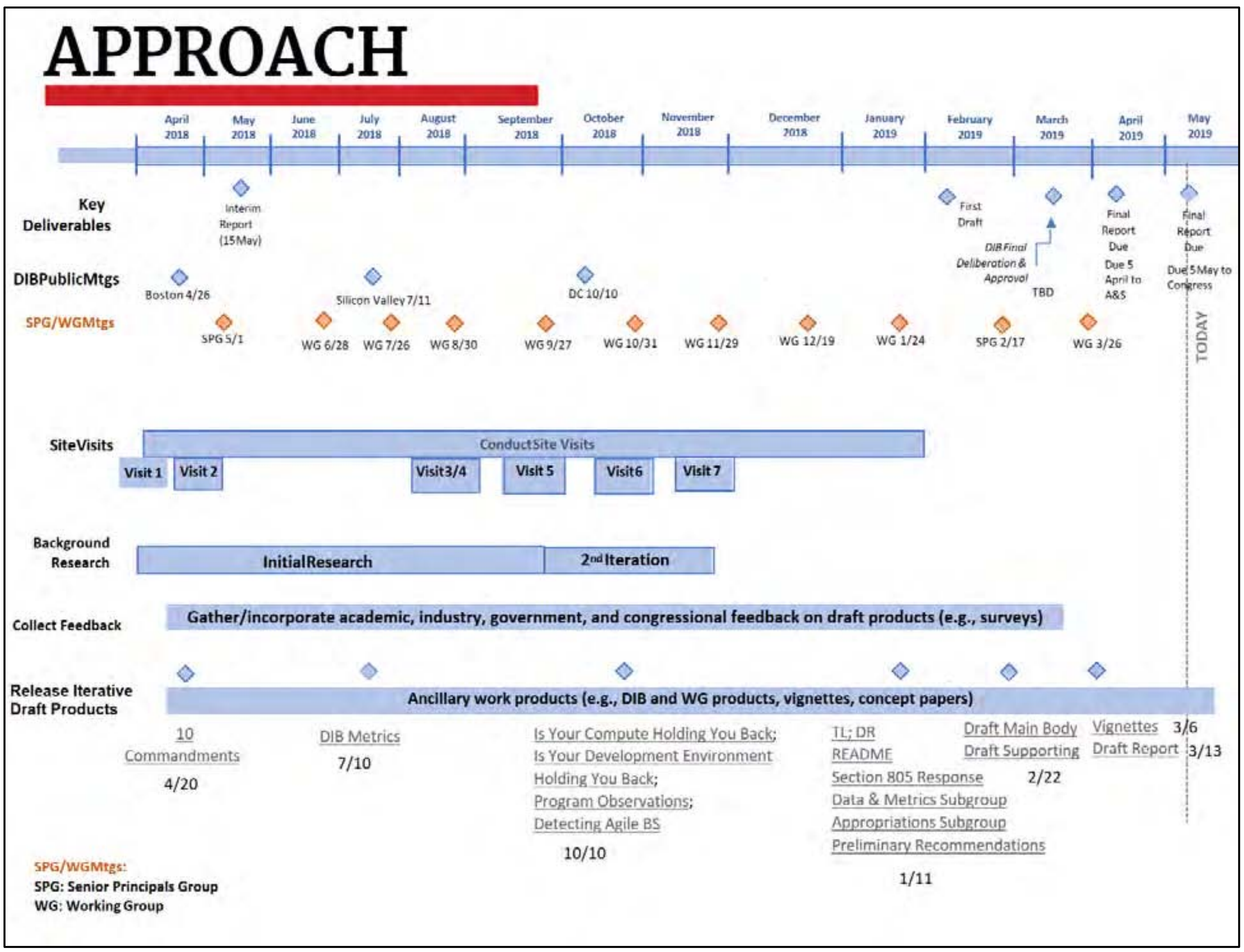
[12] The DIB first heard this term from one of the military instructors at the Air Force Academy and we now use it all the time.

up and running in less than a week. It turns out that the Department has the authority to create official G Suite sites and so we just needed to get permission to use it.

Our request went in ~10 April 2018. The site was created on 8 August 2018, 17 weeks after our request. As near as we can tell, the only thing that happened during the 4 months that it took to get the site working was that people said “no” and then other people had to spend time figuring out why they said no and either convincing them that this really was useful and a good solution for the study’s needs and/or going above their heads.

A major theme from the beginning of the SWAP study, and more generally in the DIB’s overall work, has been that DoD technology must move at the speed of (mission) need, faster than our adversaries and, certainly, not that much slower than what has proven possible and effective in the private sector. If the Department wants to take advantage of people who can help it be more effective in development and delivery of technology for improving national security, it should figure out how to *quickly* put together groups of people from inside and outside government, provide them with modern collaboration environments, and let them spend their time providing service to the Department instead of struggling with the bureaucracy.

</rant>



SWAP study schedule (used for briefings).

Vignette 4 – DDS: Fighting the Hiring Process Instead of Our Adversaries

Sean Brady, Kevin Carter, Justin Ellsworth

In novelist James Patterson and former President Bill Clinton's political thriller, *The President Is Missing*, a terrorist group threatens to unleash cyber-warfare on the Western World, bringing about the "Dark Ages." The President (in the story) must sneak away from the White House incognito, engage in shootouts, survive an ambush on Memorial Bridge, and assemble the best computer scientists from our government and military to take out the impending computer virus before it strikes.

At this point, the novel introduces a top "white hat hacker" who joins the President's team. She impresses the FBI with her hacking abilities and the Bureau hires her on the spot. In a sensational thriller that constantly demands suspended disbelief, this was by far the most unbelievable.

There's no way government hiring works that effectively or efficiently.

We know because we tried.

The Defense Digital Service (DDS) is an organization within the Pentagon tasked with driving a giant leap forward in the way DoD builds and deploys technology and digital services. One of DDS's most visible programs is Hack the Pentagon, the first bug bounty program in the history of the federal government. Bug bounties (also known as crowd-sourced hacking challenges) allow private citizens to harness their diverse range of talents to contribute and strengthen our nation's security posture in exchange for a monetary reward for finding security issues. Bug bounties are an integral part of private-sector security strategies at companies including Microsoft, Google, Twitter, and Facebook.

The winner of one of these Hack the Pentagon challenges was a 17-year-old high school student, who beat out 600 other invited hackers by reporting 30 unique vulnerabilities to the Department. After the challenge, he expressed interest in interning so he could help contribute to our nation's security outside of the challenges.

DDS staff spent the next 8 months and approximately 200 man hours trying to navigate the hiring process to bring the hacker onboard. DDS engaged with the Washington Headquarters Service, the Air Force internship program, and U.S. Army Cyber HR organizations to identify applicable hiring authorities and, more important, the HR specialists who could help drive the hiring actions for a non-traditional, but obviously qualified, candidate.

Unfortunately, what we found was a system ill-equipped to evaluate technical expertise (especially when demonstrated through experience or skill rather than certifications or education) and resistant to leveraging the full flexibilities and authorities provided.

Twice the hacker's resume was rejected as insufficient to qualify him at the necessary grade level for using direct hire authority. Ultimately, the candidate lengthened his resume to a total of five pages, which a classifier reviewed and determined would qualify him for the General Schedule (GS)-4 level, which equates to less than $16 per hour. (For what it's worth, the GS-5 only requires "experience that provided a knowledge of data processing ... gained in work such as a computer operator or assistant, [or] computer sales representative…" according to the OPM GS-2210: Information Technology Management Series General Schedule Qualification Standards). We like to point out that he would have qualified if he had worked a year at Best Buy.

Oh, and did we mention he landed on *TIME*'s List of the 25 Most Influential Teenagers of 2018? He is currently studying computer science at Stanford University.

We recognize that it is unreasonable to expect a classification specialist to understand and translate the experience listed in a resume into the education, demonstrated knowledge, and specialized experience requirements that must be met for each grade level in each job series.

The classification specialist may not have known how this particular candidate's listed experience developing *"mobile applications in IonicJS, mobile applications using Angular, and APIs using Node.js, MongoDB, npm, Express gulp, and Babel,"* met or did not meet the classification requirements of *"experience that demonstrated accomplishment of computer-project assignments that required a wide range of knowledge of computer requirements and techniques pertinent to the position to be filled.*"

This is why DDS provided a supporting memo to the classifier that identified where the candidate's resume and classification guide matched. However, the HR office refused to accept the supporting document despite OPM guidance that *"It is entirely appropriate (and encouraged!) to use Subject Matter Experts (SMEs) outside of HR to rate and rank applicants and determine the most highly qualified candidates for a position."*

Thankfully, our story, like *The President Is Missing*, has a happy ending. When it became clear that we would lose the hacker to a competing offer from the private sector, leaders at some of the highest levels of the Pentagon intervened and ordered their HR office to make the hire. With sufficient visibility and the right people assigned, the hacker's original (one-page) resume was reviewed and used to hire him at a reasonable but still below-market rate. We were ultimately able to hire him, but the process required escalation and is not scalable for more than a small number of hires.

The hacker, now 18, joined DDS as an employee during the summer of 2018 and during that time identified numerous vulnerabilities that threatened the security of information and potentially the safety of our nation.

His story was not isolated to one HR specialist or one service. As a Department, we made it as hard as possible for him to join (all while the private sector offered higher salaries and housing stipends). Hiring him did not require a new law or regulation; it required an understanding of his technical abilities, trust in those who evaluated him, and leadership that prioritizes people over process.

Vignette 5 – Kessel Run: The Future of Defense Acquisitions Is #AgileAF

Dan Ward

I've seen the future, and it's #agileAF.

That's the hashtag used by an Air Force software company known as Kessel Run—the "AF" stands for Air Force, by the way. And I did say "software company," which is how members of this military unit describe their organization. Kessel Run does not look like any other program office the Air Force has ever seen. That is its great strength. That is its great peril. And that is why it is the future.

Kessel Run's lab director welcomes new engineers. [U.S. Air Force photo by Todd Maki]

What's so great about Kessel Run? For starters, it delivers. As one example from many, in less than 130 days Kessel Run fielded an accredited Secret Internet Protocol Router (SIPR) cloud-native DevOps platform at Al Udeid Air Base, then replicated the instance at Shaw Air Force Base and fielded another DevOps platform at Osan Air Base in Japan. Don't worry if that last sentence sounded like technobabble—the point is they put stuff into the field quickly. In contrast, the previous program charged with addressing this need (which went by the catchy name "AOC 10.2") spent $430 million over 10 years before being terminated "without delivering any meaningful capability," to quote Senator John McCain. But while Kessel Run's ability to field operational software is noteworthy, its organizational achievement and the culture the team has built just might be the real breakthrough.

It turns out disruptive new technologies do not merely require cutting-edge tech. They also require new *organizational architectures*, to use Professor Rebecca Henderson's term, and very specific cultural features.

Easier said than done, of course. Building and sustaining these innovative structures inside a large legacy organization like the U.S. military requires replacing existing standards and norms. That's even harder than it sounds and is why so many large companies fail to make the switch.

Despite the difficulty, the Kessel Run team seems to have cracked the code and built a unique organization that operates at warp speed. The most visible difference between Kessel Run and business-as-usual military program offices is their location. Rather than spending all their time on the military base they are *technically* assigned to, Kessel Run personnel operate from a brightly lit We Work office in downtown Cambridge, MA. The conference rooms have Star Wars–themed names instead of Mil-Standard room numbers. The walls are covered in multi-colored sticky notes. The view of Boston is spectacular. You get the picture.

Only slightly less visible is Kessel Run's approach to contracting. Instead of handing the work over to a major defense contractor, team members built a collaborative partnership with a small-ish software company named Pivotal. Together they use DevOps methods like pair programming, where Air Force coders work side-by-side with Pivotal coders to produce software that runs on classified military systems and supports real-world military operations.

Where people sit and how they collaborate are just the tip of the iceberg. The Kessel Run culture is the product of hundreds of thoughtful design decisions that continually reinforce principles of learning, collaboration, critical thinking, and agility. The details of these decisions are beyond the scope of this short vignette, but the fact that Kessel Run continues to do the hard work of deliberately crafting and maintaining its culture is absolutely foundational to its success story.

That story is happening right now, so saying "the future is #agileAF" is actually an observation about the present. Kessel Run's approach is what right looks like *today*. Kessel Run is the new standard of military acquisition excellence, and already the other Services are starting to follow suit. Just last month the U.S. Naval Institute's blog had a post titled The Navy's Kessel Run. When your program office's name gets used in a headline like that, it's a sure sign you're doing something right.

Some skeptical commentators have expressed concern about the risks inherent in a high-speed operation like Kessel Run. In response, let's hear from the four-star commander of U.S. Strategic Command, General John Hyten. He's responsible for the nation's nuclear arsenal and is precisely the type of serious, thoughtful, risk-averse leader we want in charge of nuclear weapons. If anyone has a definitive professional opinion on Kessel Run's risk profile, it's General Hyten.

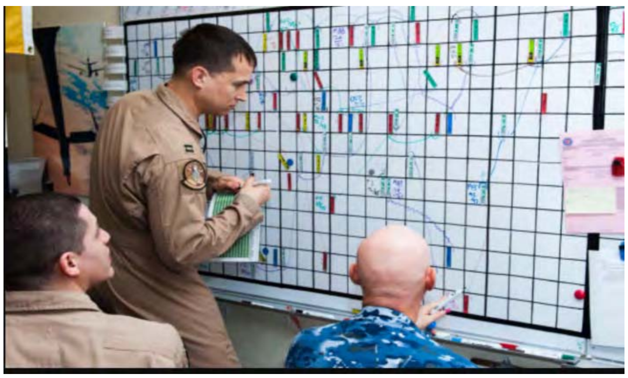
On several occasions General Hyten has stated that what keeps him up at night is the thought that the U.S. military's technology community has "lost the ability to go fast." This inability to move quickly increases the likelihood of operational shortfalls and degrades our nation's overall defense posture. In General Hyten's assessment, going too slow is far riskier than going too fast. He sounds quite comfortable with Kessel Run's pace.

In a similar vein, Secretary of the Air Force Heather Wilson submitted a report to Congress in October 2018 that described Kessel Run's achievements to date. She wrote "The use of Agile DevOps methodologies … is proving successful and we are able to rapidly deliver cloud native applications that increase operational utility. … *We believe we have demonstrated the ability to continuously deliver software that adds value to the warfighter.* " (emphasis added.)

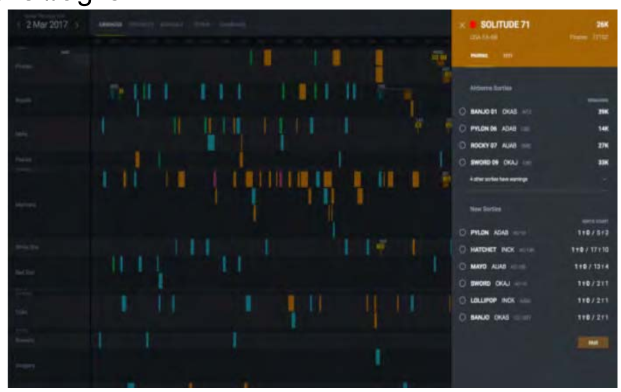
So the question is not whether the Kessel Run team delivers good results or addresses the needs of the operational community. It clearly does. Instead, the question is how long it will take the Department of Defense to adopt this organizational innovation on a larger scale. How long will DoD wait before making Kessel Run-style organizations and culture the default rather than the exception?

Replicating the Kessel Run culture requires more than giving all your conference rooms Star Wars-themed names and putting military personnel into civilian clothes. In fact, the best way to replicate the Kessel Run culture is to *not* replicate it exactly. The wisest imitators will use Kessel Run's example for illumination, not imitation. They will learn from Kessel Run's practices, not simply cut and paste them onto existing organizational structures. The wisest imitators will commit to having the difficult, ongoing conversations about values, attitudes, and beliefs that lead to genuine culture shifts. They will do the hard work of establishing and maintaining a healthy culture that unleashes people's talent and enables them to do their best work.

Kessel Run is not perfect, of course. It has collected a number of critics and skeptics alongside its fans and supporters. Interestingly, no critics see the project's shortcomings more clearly and pointedly than the Kessel Run members themselves. The team members are very aware they are still learning, still experimenting, still making mistakes and identifying opportunities for improvement. They are the first to tell you that Kessel Run has problems and struggles. They are quick to agree with some of their critics about ways the program can and should improve. That is the thing I admire most about this team. That just might be the most important practice for the rest of us to follow. And that is precisely why the future is #agileAF.



Whiteboard on which tanker refueling operations were planned. [Photo by U.S. Air Force]



The tanker refueling planning app that replaced the AOC's whiteboard. [Photo by U.S. Air Force]

Air Force Kessel Run Headquarters in Boston, MA. [U.S. Air Force photo by J.M. Eddins Jr.]

Vignette 6 – JMS: Seven Signs That Your Software (Program) Is in Trouble

Richard Murray

The DIB SWAP study visited the JMS (JSpOC [Joint Space Operations Center] Mission System) program in August 2018. The JMS team was open and cooperative, and the people working on the project were highly capable and well-intentioned. At the same time, our assessment of the program was that it was doomed to failure. Because the JMS program was restructured after our visit, we felt it was OK to spell out the problems as examples of what can go wrong.

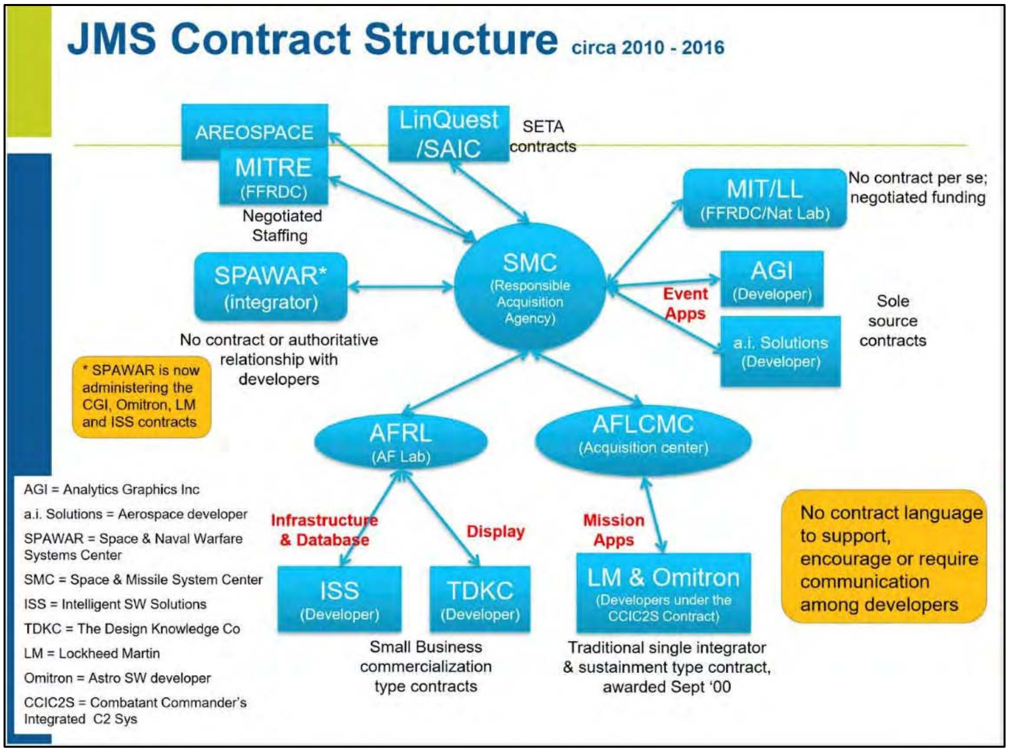
While there were many issues that led to the failure of the JMS program, the following seven are ones that are not a function of that program *per se*, but rather of the process that created it. We thus call these out as general things to look for as indications that your software (program) may be in trouble.

1. The problem is being made harder than it needs to be. JMS increment 2 had a budget of just under $1B. The basic function of the JMS system was to track objects in space. While there are engineering challenges to doing this with the proper precision, the basic problem is *not that hard*. Our sense was that the project could be converted to an "app" within AOC Pathfinder, or something equivalent. Assign 20–30 [50? 100?] programmers (+ 20% program management, administration) to work on it for 3 years at $10–20M/year, with first capability due in 6 months and increments every 2 weeks (based on user feedback). Interface to existing data sources (via software interfaces), run in the cloud, and use a scalable architecture that can get to 1M objects in the next year or two. Make sure that the app architecture can accept a commercial product if one is available that meets the needs of the user (there were some indications this might have already been happening). Target budget: $10–20M/year for first 5 years, $5–15M/year in perpetuity after that.

2. The requirements are outdated. Many of the requirements for JMS increment 2 appeared to trace back to its original inception circa 2000 and/or its restart in 2010. Any software program in which a set of software requirements was established more than 5 years ago should be shut down and restarted with a description of the desired end state (list of features with specifications) and a prioritization of features that should be targeted for simplest usable functionality.

3. The program organizational structure is designed to slow things down. Any software program with more than one layer of indirection between the prime contractor/integrator and the companies doing the engineering work should be shut down and restarted with a set of level-of-effort–style contracts that go directly from the system integrator to the companies delivering code. The system integrator should own the architecture, including the design specifications for the components that plug into that architecture.

4. The program contract structure is designed to slow things down even more. The program had at least a dozen contracts with all sorts of small companies and National Labs. It was apparently treated as a COTS integration problem with lots of pieces, but it was implemented in a way that seemed designed to ensure that nobody could make any progress.



JMS contract structure. [Photo courtesy of former JMS program office]

5. The program is implementing "waterfall with sprints" (otherwise known as Agile BS). The program was implementing "sprints" of ~6–9 months (Agile BS detector alert!). Sprints had hundreds of tasks spread across six development teams. Just coordinating was taking weeks. For a while the program had used 4-week sprints, but infrastructure was not available to support that cadence. Test happened after delivery of software, with very little automation.

6. The program management office is too big and does not know enough about software. We were told there were 200–260 FTEs in the program office. The overall program management should be limited to 10–20% of the size of the program so that resources are focused on the development team (including system architects, user interface designers, programmers, etc.), where the main work gets done. The program office must have expertise in software programs so that it is able to utilize contract and oversight structures that are designed for software (not hardware).

7. OT&E is done as a tailgate process. As an ACAT1 program, JMS was mandated to conduct operational test, a process that nominally required the program to freeze its baseline, do the tests, and then wait 120 days for report. The Operational User Evaluation conducted in early 2018 was terminated early by the Air Force due to poor performance of the system. The OT&E process being used by the program added information to support the termination decision, but it is important to note that had the program not been terminated the tailgate nature of the evaluation was one that would have added further delays.

The JMS program has since undergone major changes to address the issues above, so the criticisms here should be taken as an example of some of the signs that a program is in trouble.

Software Is Never Done:
Refactoring the Acquisition Code for Competitive Advantage

Defense Innovation Board, 3 May April 2019

J. Michael McQuade and Richard M. Murray (co-chairs)
Gilman Louie, Milo Medin, Jennifer Pahlka, Trae' Stephens

Supporting Information

This document contains the supporting information for the Defense Innovation Board's (DIB's) Software Acquisition and Practices (SWAP) study.

Contents

Appendix A: Draft Implementation Plan

The following pages contain summaries for each recommendation that give more detail on the rationale, supporting information, similar recommendations, specific action items, and notes on implementation. The beginning of each recommendation summary includes the recommendation statement, proposed owner, background information, description of the desired state, proposed role for Congress, and a short list of actions describing how the recommendation might be implemented. The remainder of the summary contains a list of recommendations from the DIB Guides (contained in Appendix E of the supporting information), a list of recommendations from the working group reports (Appendix F of the supporting information), and some related recommendations from previous reports.

The recommendations listed here are relatively decoupled, but there are some dependencies between them, as shown to the right. In figure A.1, an arrow leading from one recommendation toward a second recommendation means that the first implementation depends at least somewhat on the implementation of the second. Hence by choosing one recommendation and following the arrows, the list of all recommendations that should also be implemented can be obtained.

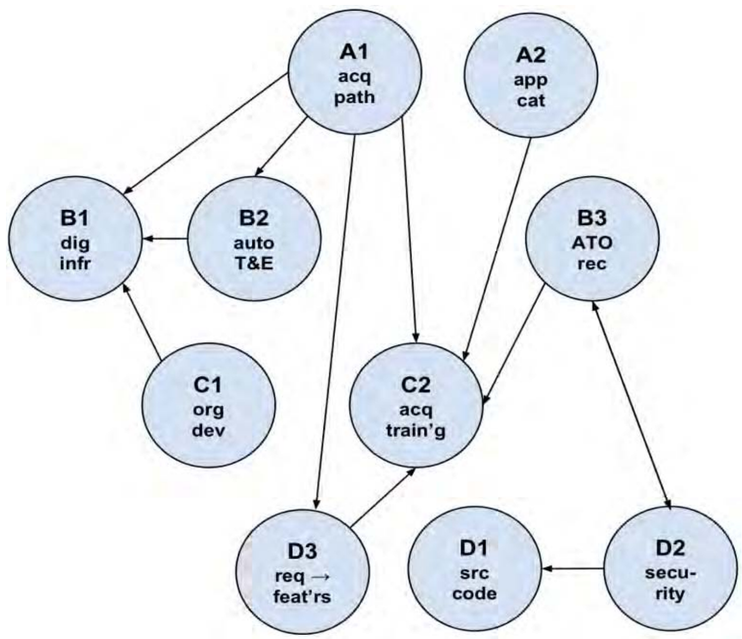


Figure A.1. Interdependency of recommendations.

The recommendations of the report are broken up into four primary lines of effort:

A. Refactor statutes, regulations, and processes for software

B. Create and maintain cross-program/cross-service digital infrastructure

C. Create new paths for digital talent (especially internal talent)

D. Change the practice of how software is procured and developed

For each of the lines of effort, we give a set of two or three primary recommendations (bold) and two to four additional recommendations (see Chapter 5 for insights).

Primary Recommendation A1
New Acquisition Pathway

Line of Effort	Refactor statutes, regulations, and processes for software.
Recommendation	**Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics.**
Stakeholders	A&S, HASC/SASC, USD(C), CAPE, DOT&E, R&E/DT, SAE, Service FM & PA&E, Joint Staff
Background	Current law, regulation, policy, and internal DoD processes make DevSecOps software development extremely difficult, requiring substantial and consistent senior leadership involvement. Consequently, DoD is challenged in its ability to scale DevSecOps software development practices to meet mission needs.
Desired State	Tailored, software-specific pathways that provide guidance to acquisition professionals for navigating the acquisition and requirements life cycle to rapidly deliver capabilities. Each pathway streamlines the processes, reviews, and documents based on the type of IT/SW capability. Programs choosing these pathways have the ability to rapidly field and iterate new functionality in a secure manner, with continuous oversight based on automated reporting and analytics, and utilizing IA-accredited commercial development tools. Rapid acquisition authority should be available for software already in use and accredited, especially when purchased as a capability delivery (as a service). Over time, this becomes the default choice for software and software-intensive programs/program elements.
Role of Congress	This acquisition pathway should become the primary pathway that DoD chooses to use for software and software-intensive programs and should provide Congress with the insight required to oversee software projects that move at a much faster pace than traditional HW programs, with traditional metrics and milestones replaced by more software-compatible measures of progress.

Draft Implementation Plan		Lead Stakeholder	Target Date
A1.1	(optional) Submit legislative proposal using Sec 805 to propose new acquisition pathways for two or more classes of software (e.g., application, embedded), optimized for DevSecOps.	USD(A&S), in coordination with USD(C) and CAPE	Q3 FY19
A1.2	Create new acquisition pathway(s) for two or more classes of software, optimized for DevSecOps (based on A2c.1 or Appendix B.1).	HASC, SASC	FY20 NDAA
A1.3	Develop and issue a Directive-Type Memorandum (DTM) for the new software acquisition pathway.	USD(A&S)	Q1 FY20
A1.4	Issue Service-level guidance for new acquisition pathway.	SAEs	Q2 FY20

A1.5	Select 5 initial programs using modern software development (DevSecOps) to convert to or use new software acquisition pathway.	USD(A&S), with SAEs	Q2 FY20
A1.6	Develop and implement training at Defense Acquisition University on new software acquisition pathway for all acquisition communities (FM, Costing, PM, IT, SE, etc.).	USD(A&S)	Q3 FY20
A1.7	Convert DTM to DoD Instruction (perhaps 5000.SW), incorporating lessons learned during initial program implementation.	USD(A&S)	Q4 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Define software as a critical national security capability under Section 805 of FY16 NDAA "Use of Alternative Acquisition Paths to Acquire Critical National Security Capabilities."
Acq	Create an acquisition policy framework that recognizes that software is ubiquitous and will be part of all acquisition policy models.
Acq	Create a clear, efficient acquisition path for acquiring non-embedded software capability. Deconflict supplemental policies.
Acq	Develop an Enterprise-level Strategic Technology Plan that reinforces the concept of software as a national security capability and recognizes how disruptive technologies will be introduced into the environment on an ongoing basis.
Acq	Additionally, take all actions associated with Rec A2a to refactor and simplify those parts of Title 10, DoD 5000 and other regulations and processes that are still in force for software-intensive programs.

Related recommendations from previous studies

DSB87	Rec 13: The Undersecretary of Defense (Acquisition) should adopt a four-category classification as the basis of acquisition policy [standard (COTS), extended (extensions of current systems, both DoD and commercial), embedded, and advanced (advanced and exploratory systems)].
DSB87	Rec 14: USD(A) should develop acquisition policy, procedures, and guidance for each category.
DSB09	The USD(AT&L) should lead an effort, in conjunction with the Vice Chairman, Joint Chiefs of Staff, to develop new, streamlined, and agile capabilities (requirements) development and acquisition processes and associated policies for information technology programs.

Primary Recommendation A2
New Appropriation Category

Line of Effort	Refactor statutes, regulations, and processes for software.
Recommendation	**Create a new appropriation category for software capability delivery that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&E, production, and sustainment.**
Stakeholders	A&S, HAC-D/SAC-D, HASC/SASC, USD(C), CAPE, SAE, Service FM & PA&E, FASAB, OMB
Background	Current law, regulation, and policy treat software acquisition as a series of discrete, sequential steps; accounting guidance treats software as a depreciating asset. These processes are at odds with software being continuously updated to add new functionality, and they create significant delays in fielding user-needed capability.
Desired State	Appropriations for software and software-intensive programs use a Major Force Program (MFP) category that provides a single budget to support full life cycle costs of software, including development, procurement, assurance, deployment, and continuous improvement. Programs are better able to prioritize how effort is spent on new capabilities versus fixing bugs/vulnerabilities, improving existing capabilities, etc. Such prioritization can be made based on warfighter/user needs, changing mission profiles, and other external drivers, not constrained by available sources of funding.
Role of Congress	This should become the primary pathway that Congress uses to fund software and software-intensive programs and should provide Congress with the insight required to oversee software projects that move at a much faster pace than traditional HW programs, with traditional metrics and milestones replaced by more software-compatible measures of progress.

	Draft Implementation Plan	Lead Stakeholder	Target Date
A2.1	(optional) Submit legislative proposal using Sec 805 to create a new appropriations category for software and software-intensive programs.	USD(A&S), with USD(C) and CAPE	Q3 FY19 for FY20 NDAA
A2.2	Create new appropriation category for software-intensive programs, with appropriate reporting and oversight for software (based on Action A2.1 or Appendix B.1).	HAC-D, SAC-D, with OSD, HASC, SASC	FY20 NDAA, FY20 budget
A2.3	Select initial programs using DevSecOps to convert to or use new SW Appropriation in FY20.	USD(A&S), with Service Acquisition Executives	Q4 FY19
A2.4	Define budget exhibits for new SW appropriation (replacement for P- and R-Forms; see Appendix C).	USD(A&S), with USD(C), CAPE, HAC-D, SAC-D	Q4 FY19

A2.5	Change audit treatment of software with these goals: (1) separate category for software instead of being characterized as property, plant, and equipment; (2) default setting that software is an expense, not an investment; and (3) "sustainment" is an integrated part of the software life cycle.	FASAB, with USD(A&S) and USD(C)	End FY20
A2.6	Make necessary modifications in supporting PPB&E systems to allow use and tracking of new software appropriation.	USD(C) and CAPE	Q1 FY21
A2.7	Ensure programs using new software appropriation submit budget exhibits in the approved format.	SAE with USD(C), CAPE	FY 22 POM

SWAP concept paper recommendations related to this recommendation

10C	Budgets should be constructed to support the full, iterative life cycle of the software being procured with amount proportional to the criticality and utility of the software.
Visits	Construct budget to support the full, iterative life cycle of the software.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Revise 10 USC 2214 to allow funding approved by Congress for acquisition of a specific software solution to be used for research and development, production, or sustainment of that software solution, under appropriate conditions.
App	A new multi-year appropriation for Digital Technology needs to be established for each Military Defense Department and the Fourth Estate.
App	Components will program, budget, and execute for information and technology capabilities from one appropriation throughout life cycle rather than using RDT&E, procurement, or O&M appropriations—often applied inconsistently and inaccurately—allowing for continuous engineering.
Con	Congress establishes new authority for contracting for SW development and IT modernization.
M&S	Revise 10 USC 2460 to replace the "software maintenance" with "software sustainment" and use a definition that is consistent with a continuous engineering approach across the life cycle.
M&S	A DoD Working Group should be established to leverage ongoing individual Service efforts and create a DoD contracting and acquisition guide for software and software sustainment patterned after the approach that led to creation of the DoD Open Systems Architecture Contracting Guide.
M&S	Acquisition Strategy, RFP/Evaluation Criteria, and Systems Engineering Plan should address software sustainability and transition to sustainment as an acquisition priority.
Con	Manage programs at budget levels, allow programs to allocate funds at project investment level.
Con	Work with appropriators to establish working capital funds so that there is not pressure to spend funds sooner than when you're ready (iterative contracts may produce more value with less money).

Related recommendations from previous studies

GAO15	When assigning resources to all activities, the schedule should reflect the resources (labor, materials, travel, facilities, equipment, and the like) needed to do the work, whether they will be available when needed, and any constraints on funding or time.
GAO17	Hold suppliers accountable for delivering high-quality parts for their products through activities including regular supplier audits and performance evaluations of quality and delivery.

GAO17	Prioritize investments so that projects can be fully funded and it is clear where projects stand in relation to the overall portfolio.
CSIS18	Performance Based Logistics (PBL) contracts should have a duration that allows for tuning and re-baselining with triggered options and rolling extensions.
Sec809	Rec. 41: Establish a sustainment program baseline, implement key enablers of sustainment, elevate sustainment to equal standing with development and procurement, and improve the defense materiel enterprise focus on weapon system readiness.
Sec809	Rec. 42: Reduce budgetary uncertainty, increase funding flexibility, and enhance the ability to effectively execute sustainment plans and address emergent sustainment requirements.

Additional Recommendation A3
Metrics for Cost Assessment and Performance Estimates

Line of Effort	Refactor statutes and regulations for software.
Recommendation	**Require cost assessment and performance estimates for software programs (and software components of larger programs) of appropriate type be based on metrics that track speed and cycle time, security, code quality, and functionality.**
Stakeholders	CAPE, CMO, USD(A&S), Service CMOs and SAEs
Background	Current software cost estimation and reporting processes and procedures in DoD have proven to be highly inaccurate and time consuming. New metrics are required that match the DevSecOps approach of continuous capability delivery and maintenance and provide continuous insight into program progress.
Desired State	Program oversight will re-focus on the value provided by the software as it is deployed to the warfighter/user and will rely more heavily on metrics that can be collected in a (semi-)automated fashion from instrumentation on the DevSecOps pipeline and other parts of the infrastructure. Specific metrics will depend on the type of software rather than a one-size-fits-all approach.
Role of Congress	Congress needs to emphasize the need for new software acquisition reporting that focuses on value provided for the investment in software and frequency of deployments to the warfighter/user. Congress needs to work with CAPE and USD(A&S) to provide feedback on meaningful content and level of detail in reporting.

Draft Implementation Plan		Lead Stakeholders	Target date
A3.1	Identify (or hire) a small team (3-4) programmers to implement software for automated collection and analysis of metrics and provide them with a modern development environment.	CAPE, DDS	Q4 FY19
A3.2	Identify low-level metrics that are already part of standard commercial development environments (see Appendix C for reporting approach and Appendix E.2 (DIB's "Metrics for Software") for initial lists).	CAPE, SAO	MVP[1] Q4 FY19, then quarterly
A3.2a	Speed and cycle time: launch → initial use, cycle time	Dev team, users	
A3.2b	Code quality: unit test coverage, bug burn-rate, bugs-in-test:bugs-in-field	Dev team, users	
A3.2c	Security: patch → field, OS upgrade → field, HW/OS age	Dev team, users	
A3.2d	Functionality: user satisfaction, number/type of features/cycle	Dev team, users	
A3.2e	Cost: head count, software license cost, compute costs	Dev team, users	
A3.3	Identify 3-5 ongoing programs that are collecting relevant metrics and that partner with CAPE to collect and use data.	CAPE, A&S, CMO, SAEs	In parallel with A6.2

[1] Minimum viable product (first useful iteration)

A3.4	Create a mechanism to transfer and process low-level metrics from development team to PMO on a continuous basis with selectable levels of resolution across the program.	CAPE, SAEs, PMO	MVP Q4 FY19, then quarterly
A3.5	Begin reporting metrics to Congress as part of annual reporting; iterate on content, level, format.	CAPE, Comp, A&S	FY2020
A3.6	Use initial results to establish expectations for new proposed software or software-intensive projects and integrate use of new cost and performance estimates into contract selection.	A&S, SAEs, CAPE	FY2020
A3.7	Establish ongoing capability within CAPE to update metrics on continuous basis, with input from users (of the data).	CAPE	FY2021
A3.8	Identify and eliminate remaining uses of ESLOC as metric for cost and schedule estimation of software/software-intensive programs.	CAPE, SAEs	FY2022

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Con	Revise estimation models - source lines of code are irrelevant to future development efforts, estimations should be based on the team size and investment focused (Cultural).

Related recommendations from previous studies

SEI01	Effort Estimation: • Utilize most likely effort estimates in proposals and status reports; • Find ways to promote the use of accurate effort estimation and productivity evaluation; • Lowest cost is not equivalent to best value. Question outliers.
OSD06	Adjust program estimates to reflect “high confidence”—defined as a program with an 80 percent chance of completing development at or below estimated cost—when programs are baselined in the Stable Program Funding Account.
SEI10	Don’t require PMO to adopt contractors’ estimate for the program—or else use the difference as PM “reserve.”
SEI10	Change from traditional 50% estimation confidence level to 80% level.
SEI10	DoD should consider use of Vickrey “second price” auction mechanism for acquisition proposal bidding.
SEI15	Use the government’s cost estimates (using perhaps an 80% confidence level) rather than contractors’ estimates as the basis for program budgets and place the difference (if the government’s estimate is larger) in a reserve fund available to program managers with sufficient justification. Contractors’ estimates should be acquired using mechanisms that promote accurate estimates, e.g., using Vickrey auctions, the Truth-Revealing Incentive Mechanism (TRIM), or more standard methods of review and acceptance by independent third parties.
DSB18	Rec 3b: The MDA with the Cost Assessment and Program Evaluation office (CAPE), the USD(R&E), the Service Cost Estimators, and others should modernize cost and schedule estimates and measurements.
DSB18	Rec 3b.1: [DoD] should evolve from a pure SLOC approach to historical comparables as a measurement, and should adopt the National Reconnaissance Office (NRO) approach (demonstrated in Box 5) of contracting with the defense industrial base for work breakdown schedule data to include, among others, staff, cost, and productivity.
DSB18	Rec 3c: The MDA should immediately require the PM to build a program-appropriate framework for status estimation.

Additional Recommendation A4
Simplify Laws and Policies

Line of Effort	Refactor statutes and regulations for software.
Recommendation	**Refactor and simplify Title 10, DFARS, and DoDI 5000.02/5000.75 to remove statutory, regulatory, and procedural requirements that generate delays for acquisition, development, and fielding of software while adding requirements for continuous (automated) reporting of cost, performance (against updated metrics), and schedule.**
Stakeholders	USD(C), CAPE, SAE, Service FM & PA&E, Joint Staff
Background	Current law, regulation, policy, and internal DoD processes make modern software development extremely difficult, requiring substantial and consistent senior leadership involvement. Consequently, DoD is challenged in its ability to scale modern software development practices to meet mission needs. Recommendation A1 (new acquisition pathway) provides a pathway that is optimized for software, but it is also possible to modify existing statutes, regulations, and processes to remove barriers for software.
Desired State	Programs have the ability to rapidly field and iterate new functionality in a secure manner, with continuous oversight based on automated reporting and analytics, and utilizing IA-accredited commercial development tools. Congress has better insight into the status of software programs through improved reporting of relevant metrics (see also Recommendations A3 and D4 on metrics).
Role of Congress	Work with DoD to review current statutes and evaluate their effectiveness for different types of software, removing barriers that add time and interfere with the continuous nature of modern software development. See Appendix F for a list of issues to consider.

Draft Implementation Plan		Lead Stakeholders	Target Date
A4.1	Submit legislative proposal(s) to simplify Title 10 for software (see also: Sec 809 Panel report).	USD(A&S)	Q3 FY19
A4.2	Convene working group with stakeholders and develop and issue a Directive-Type Memorandum (DTM) for the new simplified software acquisition process.	USD(A&S)	Q1 FY20
A4.3	Issue Service-level guidance for new simplified software acquisition process.	SAE	Q1 FY20
A4.4	Identify initial set of programs using modern software development methods to convert to or utilize new, simplified software acquisition process.	USD(A&S), with SAEs	Q1 FY20
A4.5	Convert DTM to DoD Instruction, incorporating lessons learned during initial program implementation.	USD(A&S)	Q1 FY20
A4.6	Develop and implement training at Defense Acquisition University on new, simplified software acquisition process for all acquisition communities (FM, Costing, PM, IT, SE, etc.).	USD(A&S)	Q1 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Ensure appropriate integration of a data strategy and the Department's Cloud Strategy. Examine a Steering Committee approach for management.
Acq	Examine the organizational structure with the intent of achieving a more responsive and flat organizational model that de-conflicts roles and responsibilities between the DoD CIO, the USD(A&S), and the CMO regarding software.
Acq	Re-focus the software acquisition workforce on teaming and collaboration, agility, improved role definition, career path advancement methods, continuing education and training opportunities, incentivization, and empowerment.
Acq	Increase flexibility and agility for software programs by eliminating mandated content for acquisition strategies and authorities in Section 821 of the FY16 NDAA, except for MDAPs.
Acq	Eliminate hardware-centric cost, fielding, and performance goals in 10 USC 2488 (established by Sec 807 of the FY17 NDAA) for software-intensive programs.
Acq	Eliminate Nunn-McCurdy breaches (10 USC 2433) for software-intensive programs and replace with continuous evaluation of software performance metrics.
Acq	Remove statutory definition of "major system" for software-intensive programs in 10 USC 2302 and 2302d to remove confusion, since most software in weapons systems inherently functions together to fulfill a mission need.
Acq	Develop language for 10 USC 2366 that allows exemption for software-intensive programs, where DOT&E must justify adding the program for oversight with the MDA and must streamline the process.
Acq	Only require DOT&E oversight for software-intensive programs when requested by the SAE, USD(A&S), or Congress, or if the program is an MDAP.
Acq	For the Fourth Estate, combine all three authorities for DBS under the DoD CMO. After one year, conduct assessment and make a determination if this should be applied to the Services as well.
Acq	Eliminate the separate annual funding certification process for defense business system from 10 USC 2222 or require that funding certification be merged in to the PPBE process.
Acq	Replace annual configuration steering board (CSBs) for software-intensive programs with board (or equivalent entities) established by the CAE, PEO, or PM [FY09 NDAA Sec 814; DoDI 5000.02].
Acq	Expand the FAR 39 (Acquisition of IT) to allow for one area to drive technology purchases. Unless otherwise stated, no other FAR rules would apply.
Acq	Rewrite FMR Volume 2A, Chapter 1, Section 010212(B) to [1] acknowledge that, for the purpose of modifying or enhancing software, there is no technically meaningful distinction between RDT&E, Procurement, and O&M; [2] eliminate the $250,000 barrier between expenses and investments (i.e., stop explicitly tying to a dollar threshold, the determination of whether software is an expense or an investment).
Acq	Revise or eliminate DoDI 8330.01 to eliminate the following elements for software-intensive programs: [1] NR KPP required; [2] DoD-specific architecture products in the DoDAF format that are labor intensive and of questionable value; [e] Interoperability Support Plans (ISPs) required, where DoD CIO can declare any ISP of "special interest"; [2] requirement of DT authority to provide assessments at MS C; [5] mandates JITC to do interoperability assessments for IT with "joint, multinational, and interagency interoperability requirements."

Acq	Revise PfM policy (DoDD 7045.20) to consider the role of data and metrics, as well as additional portfolios (like NC3), and determine authority for the policy.
Con	Separate Contract requirements (scope, PoP, and price) from technical requirements (backlog, roadmap, and stories).
Con	Use SOO vs. SOW to allow the vendor to solve the objectives how they are best suited.
Con	Establish clear and intuitive guidelines on how and when to apply existing clauses.
Con	Have standard clause applications for each of the above that must be excepted vs. accepted.
D&M	Congress could establish, via an NDAA provision, new data-driven methods for governance of software development, maintenance, and performance. The new approach should require on-demand access to standard (and perhaps real-time) data with reviews occurring on a standard calendar, rather than the current approach of manually developed, periodic reports.
M&S	Title 10 USC 2460 should be revised to replace the term “software maintenance” with the term “software sustainment” and use a definition that is consistent with a continuous engineering approach across the life cycle.
Req	The Joint Staff should consider revising JCIDS guidance to focus on user needs, bypassing the JCIDS process as needed to facilitate rapid software development. Guidance should specifically account for user communities (e.g., Tactical Action Officer (TAO), Maritime Operations Center (MOC) director) that do not have one specific PoR assigned to them, but use multiple systems and data from those systems to be effective.
Req	The Joint Staff should consider revising JCIDS guidance to separate functionality that needs high variability from the functionality that is deemed “more stable” (e.g., types of signals to analyze vs. allowable space for the antenna). Then implement a “software box” approach for each one in which the contours of the box are shaped by the functionality variability.
Req	The Joint Staff should consider revising JCIDS guidance to document stable concepts, not speculative ideas. Acknowledge that software requirement documents will iterate, iterate, iterate. JCIDS must change from a “one-pass” mentality to a “first of many” model that is inherently agile, delegating approval to the lowest possible level.

Related recommendations from previous studies

DSB87	Rec 21: DoD should examine and revise regulations to approach modern commercial practice insofar as practicable and appropriate.
NPS16a	Program offices spend far too much time generating paperwork and navigating the bureaucracy rather than thinking creatively about program risks, opportunities, and key elements of their strategies.
NDU17	Develop and maintain core competencies in diverse acquisition approaches and increase the use of venture-capital-type acquisitions, such as Small Business Innovative Research (SBIR), Advanced Concept Technology Development (ACTD), and Other Transaction Authority (OTA), as mechanisms to draw in non-traditional companies.
NDU17	Encourage employees to study statutes and regulations and explore innovative and alternative approaches that meet the statutory and regulatory intent.
Sec809	Rec. 62: Update the FAR and DFARS to reduce burdens on DoD’s commercial supply chain to decrease cost, prevent delays, remove barriers, and encourage innovation available to the

	Military Services.
Sec809	Rec. 74: Eliminate redundant documentation requirements or superfluous approvals when appropriate consideration is given and documented as part of acquisition planning.
Sec809	Rec. 75: Revise regulations, instructions, or directives to eliminate non-value-added documentation or approvals.
Sec809	Rec. 90: Reorganize Title 10 of the U.S. Code to place all of the acquisition provisions in a single part, and update and move acquisition-related note sections into the reorganized acquisition part of Title 10.

Additional Recommendation A5
Streamlined Processes for Business Systems

Line of Effort	Refactor statutes and regulations for software.
Recommendation	**Create streamlined authorization and appropriation processes for defense business systems (DBS) that use commercially available products with minimal (source code) modification.**
Stakeholders	CMO, USD(A&S), Service CMOs, SAEs, DoD CIO
Background	Current DoD business processes are minimally standardized due to a high number of legacy systems that inhibit business process reengineering. In addition, solicitation for new business systems often insists on customization because DoD is "different," resulting in hard-to-maintain systems that become obsolete (and possibly insecure) quickly.
Desired State	DoD uses standard commercial packages for enterprise and business services, changing its processes to match those of large industries, allowing its systems to be updated and modified on a much faster cadence. The only specialized defense business systems should be those for which there is no commercial equivalent (to include cases in which minor modifications would be required) and there is a funded internal capability to maintain and update the software at a near-commercial cadence.
Role of Congress	Congressional approval for new software development programs should be based on a clear assessment of the current state of commercial software and the need for DoD-specific customization. In many cases it should be possible to make use of commercial systems and modify the DoD process to be consistent with commercial practice rather than attempting to build and maintain specialized business systems. Support legislative change of 10 USC §2222, as needed.

	Draft Implementation Plan	**Lead Stakeholders**	**Target Date**
A5.1	Use a Net Promoter Score (NPS) assessment to identify 10 programs whose customers (soldiers, civilians, or others) believe the functionality could be better executed with commercial software.	CMO, with USD(A&S), Service counterparts	Q4 FY19
A5.2	Using the results of A5.1, select four projects for a more detailed assessment of possible savings and/or efficiency improvements.	CMO, with Service CMOs and business process owners	Q1 FY20
A5.3	Implement COTS opportunities, with contracts in place.	Services, with CMO oversight	Q1 FY21
A5.4	Submit legislative change proposal to modify Title 10 §2222 to reflect the lessons learned through process re-engineering to utilize commercially available system over DoD-specific solutions.	CMO, with USD(A&S) and Service counterparts	FY21

SWAP concept paper recommendations related to this recommendation

10C	Use commercial process and software to adopt and implement standard business practices within the Services.
D&D	For common functions, purchase existing software and change DoD processes to use existing apps.

Related recommendations from previous studies

DSB87	Rec 15: The USD(A) and the ASD(Comptroller) should direct Program Managers to assume that system software requirements can be met with off-the-shelf subsystem and components until it is proved that they are unique.
Sec809	Rec 16: Combine authority for requirements, resources, and acquisition in a single, empowered entity to govern DBS portfolios separate from the existing acquisition chain of command.
Sec809	Rec 18: Fund DBSs [defense business systems] in a way that allows for commonly accepted software development approaches.

Additional Recommendation A6
Enduring Capability

Line of Effort	Refactor statutes, regulations, and processes for software.
Recommendation	**Plan, budget, fund, and manage software development as an enduring capability that crosses program elements and funding categories, removing cost and schedule triggers associated with hardware-focused regulations and processes.**
Stakeholders	USD(A&S), USD(C), SAE, Service FM, HASC, SASC
Background	The current approach to acquiring software is based on projects that have a beginning and end. However, many missions are "enduring capabilities" and need software program and portfolio management that continually and perpetually deliver across the spectrum of new capability, incremental enhancements, and life cycle sustainment. The Department should pilot and then scale methods for appropriating software budgets for these enduring capability programs as an ongoing, regularly evaluated expense, with continuous oversight, rather than large, multi-year development contracts.
Desired State	The Department can manage software acquisition as an activity requiring continuous development, deployment, and sustainment, recognizing that software systems are long-lived and have a continuous need for a level of activity to evolve capabilities and address vulnerabilities. Assessment of progress will be maintained throughout the software lifespan by means of continual user engagement with working software, rather than at large-scale milestone gates that do not map well to the underlying technical activities.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholder	Target Date
A6.1	Modify FMR to implement this continuous funding approach.	USD(C)	Q4 FY19
A6.2	Select and launch five programs to be managed as enduring capability, two-year pilot projects.	USD(A&S) with SAE	Q4 FY19
A6.3	Work with FASAB to create an audit treatment of enduring capability software that has a category distinct from Property, Plant, and Equipment; defaults to treating software as an expense, not an investment; and does not distinguish between development and sustainment.	USD(A&S) with USD(C)	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Budgets should be constructed to support the full, iterative life cycle of the software being procured with amount proportional to the criticality and utility of the software.
D&D	Treat software development as a continuous activity, adding functionality continuously.

Additional Recommendation A7
Portfolio Management

Line of Effort	Refactor statutes, regulations, and processes for software.
Recommendation	**Replace JCIDS, PPB&E, and DFARS with a portfolio management approach to software programs, assigned to "PEO Digital" or an equivalent office in each Service that uses direct identification of warfighter needs to determine allocation priorities for software capabilities.**
Stakeholders	USD(A&S), CAPE, JCS, USD(C), SAE, Service FM & PAE
Background	The current requirements process often drives the development of exquisite requirements that tend to be overly rigid and specific and attempt to describe the properties of systems in dynamic environments years in advance. The speed of requirements development and analysis is out of sync with the pace of technology and mission changes. Most importantly, requirement documents that are developed are often disconnected with the end-user requirements.
Desired State	Software programs are managed using a portfolio approach, in which resources are available for reallocation across programs and funding categories based on the importance and opportunities of given elements of the portfolio. Relevant portfolios are defined based on the linkages between programs of similar function, as defined by OSD and/or Services.
Role of Congress	Congress should approve and monitor metrics of success defined within different portfolios and measure the progress against those metrics in determining allocations of funding to different portfolios (with the decisions within a portfolio made by the portfolio office, which would be held accountable for those decisions).

Draft Implementation Plan		Lead Stakeholders	Target Date
A7.2	Select initial capability areas in each Service to place under portfolio management by PEO Digital (or equivalent).	SAEs	Q3 FY19
A7.1	Issue guidance for management of software portfolios with a "PEO Digital" or similar office with OSD and/or the Services.	USD(A&S) SAE	Q4 FY19
A7.3	Stand up PEO Digital or equivalent office with necessary resources allocated and aligned.	SAE	Q1 FY20
A7.4	Implement new portfolio management methods for initial program capability areas.	PEO Digital	Q3 FY20
A7.5	Determine intermediate successes of, or required modifications to, portfolio management approach.	PEO Digital	Q1 FY21
A7.6	Establish portfolio management approach as standard work for software.	PEO Digital, SAE	FY22

SWAP working group inputs (reflected in Appendix F) related to this recommendation

App	Within each Component-unique Budget Activity (BA), Budget Line Items (BLINs) align by functional or operational portfolios. The BLINs may be further broken into specific projects to provide an even greater level of fidelity. These projects would represent key systems and supporting activities, such as mission engineering.
App	By taking a portfolio approach for obtaining software-intensive capabilities, the Components can better manage the range of requirements, balance priorities, and develop portfolio approaches to enable the transition of data to information in their own portfolios and data integration across portfolios to achieve mission effects, optimize the value of cloud technology, and leverage and transition to the concept of acquisition of whole data services versus individual systems.
App	This fund will be apportioned to each of the Military Departments and OSD for Fourth Estate execution.
App	Governance: management execution, performance assessment, and reporting would be aligned to the portfolio framework—BA, BLI, project.
Req	OSD and the Joint Staff should consider creating "umbrella" software programs around "roles" (e.g., USAF Kessel Run).

Related recommendations from previous studies

OSD06	Transform the Planning, Programming, and Budgeting, and Execution process and stabilize funding for major weapons systems development programs.
DSB09	The USD(AT&L) aggressively delegate milestone decision authority commensurate with program risk.
DSB09	The USD(AT&L) consider a more effective management and oversight mechanism to ensure joint program stability and improved program outcomes.
DSB09	Consolidate all acquisition oversight of information technology under the USD(AT&L) by moving into that organization those elements of the OASD (NII)/DOD CIO and Business Transformation Agency responsible for IT acquisition oversight. The remainder of OASD (NII)/DOD CIO is retained as it exists today, but should be strengthened as indicated in the previous recommendation.
Sec809	Rec 36: Transition from a program-centric execution model to a portfolio execution model.
Sec809	Rec 37: Implement a defense-wide capability portfolio framework that provides an enterprise view of existing and planned capability, to ensure delivery of integrated and innovative solutions to meet strategic objectives.
Sec809	Rec. 38: Implement best practices for portfolio management.
Sec809	Rec. 39: Leverage a portfolio structure for requirements.

Primary Recommendation B1
Digital Infrastructure

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors.**
Stakeholders	A&S, CIO, SAE, USD(C)
Background	Currently, DoD programs each develop their own development and test environments, which requires redundant definition and provisioning, replicated assurance (including cyber), and extended lead times to deploy capability. Small companies and other new entrants have difficulties providing software solutions to DoD because those environments are not available outside the incumbent contractor or because they have to build (and certify) unique infrastructure from scratch.
Desired State	Programs will have access to, and be stakeholders in, a cross-program, modern digital infrastructure that can benefit from centralized support and provisioning to lower overall costs and the burden for each program. Development infrastructure supporting CI/CD and DevSecOps is available as best of breed and GOTS provided so that contractors want to use it, though DoD programs or organizations that want or need to go outside of that existing infrastructure can still do so.
Role of Congress	Congress should track the availability, scale, use, and cost effectiveness of digital infrastructure, with the expectation that overall capacity will expand while unit costs decrease over time. Sufficient funding should be provided on an ongoing basis to maintain and upgrade digital infrastructure and to maintain best-of-breed capability that accelerates software development.

	Draft Implementation Plan	**Lead Stakeholder**	**Target Date**
B1.1	Designate organization(s) responsible for creating and maintaining the digital infrastructure for each Service's digital infrastructure. Explore the use of tiered approaches with infrastructure at Service or Program level, as appropriate.	DoD CIO, USD(C) and Services (SAE and Service CIO)	Q3 FY19
B1.2	Designate organization(s) responsible for creating and maintaining digital infrastructure(s) for DoD agencies and organizations, including joint digital infrastructure available to the Services.	USD(A&S), with CIO, CMO	Q3 FY19
B1.3	Provide resources for digital infrastructure, including cloud solutions, pre-approved "drop-ship" local compute capability, approved development environments (see DIB Compute Environment concept paper, Appendix I [Glossary]).	USD(A&S), SAE with CAPE, USD(C)	FY20 budget
B1.4	Define baseline digital infrastructure systems and implement procurement and deployment processes and capability.	Responsible organizations from B1.1, B1.2	Q2 FY20

B1.5	Implement digital infrastructure and provide access to ongoing and new programs.	Responsible organizations from B1.1, B1.2	Q3 FY20
B1.6	Identify acquisition programs to transition to digital infrastructure.	SAE	Q2 FY20
B1.7	Transition programs to digital infrastructure.	SAE, CIO, PEO, PM	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Make computing, storage, and bandwidth, and programmers abundant to DoD developers and users.
D&D	Use validated software development platforms that permit continuous integration & delivery evaluation (DevSecOps platform).
Visits	Separate development of mission-level software from development of IA-accredited platforms.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

T&E	Build the enterprise-level digital infrastructure needed to streamline software development and testing across the full DoD software portfolio.

Related recommendations from previous studies

DSB87	Rec 16: All methodological efforts, especially STARS, should look to see how commercially available software tools can be selected and standardized for DoD needs.
SEI01	Infrastructure: In distributed development activities, get high-quality, secure broadband communications between sites. It is an enabler, not a cost.

Primary Recommendation B2
Automated Testing and Evaluation

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Create, implement, support, and use fully automatable approaches to testing and evaluation (T&E), including security, that allow high-confidence distribution of software to the field on an iterative basis.**
Stakeholders	DOT&E, USD(A&S), DDR&E(AC), SAE, Service Test Agencies
Background	To deliver SW at speed, rigorous, automated testing processes and workflows are essential. Current DoD practices and procedures often see OT&E as a tailgate process, sequentially after development has completed, slowing down delivery of useful software to the field and leaving existing (potentially poorly performing and/or vulnerable) software in place.
Desired State	Development systems, infrastructure, and practices are focused on continuous, automated testing by developers (with users) with frequency dependent on type of software, but targets cycle times measured in weeks. To the maximum extent possible, system operational testing is integrated (and automated) as part of the development cycle using data, information, and test protocols delivered as part of the development environment. Embedded software in safety-critical systems is tested with high confidence in representative (physical and simulated) environments. Testing and evaluation/certification of COTS components is done once (if justified), and then ATO reciprocity (Rec B3) is applied to enable use in other programs, as appropriate. System-level testing using modeling and simulation ("digital twin") is routinely used.
Role of Congress	DOT&E should provide annual reports to Congress that describe the availability, scale, use, and effectiveness of automated T&E, with the expectation that level/depth of testing will increase at the same time as speed and cycle time are being improved.

Draft Implementation Plan		Lead Stakeholders	Target Date
B2.1	Establish procedures for fully automated testing on digital infrastructure (Rec B1), updating DoDI 5129.47 and Service equivalents as needed.	USD(A&S), DOT&E, with Service Testers	Q1 FY20
B2.2	Establish processes for automated and red-team-based security testing, including zero-trust assumptions, penetration testing, and vulnerability scanning.	USD(A&S), DOT&E, with Service Testers	Q1 FY20
B2.3	Identify initial programs to use tools and workflows.	SAE	Q1 FY20
B2.4	Implement minimum viable product (MVP) tools and workflows on digital infrastructure (Rec B1).	SAE, DOT&E, with PMOs	Q2 FY20
B2.5	Migrate initial programs to digital infrastructure using automated T&E.	PEO, with Responsible Organizations	Q3 FY20
B2.6	Use tools and workflows, identify lessons learned and improvements (using DevSecOps iterative approach).	Service Testers, with PEO/PM	Q4 FY20

B2.7	Modify tools and workflows; document procedures.	Responsible Organizations, Service Testers	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Automate testing of software to enable critical updates to be deployed in days to weeks, not months or years.
D&D	Create automated test environments to enable continuous (and secure) integration and deployment to shift testing and security left.
Visits	Automate testing of software to enable critical updates to be deployed in days to weeks, not months or years (also requires changes in testing organization).
Visits	Add testing as a service.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	DOT&E should use test data collected through existing test methodologies present in software-intensive programs and not recommend or prescribe additional independent, one-time test events.
Acq	One-time IOT&Es or cybersecurity test events should not be recommended for software-intensive systems except in specific circumstances if warranted.
T&E	Build the enterprise-level digital infrastructure needed to streamline software development and testing across the full DoD software portfolio.
T&E	DoD should expand DOT&E's current capability to obtain state-of-the-art cyber capabilities on a fee-for-service basis.

Related recommendations from previous studies

DSB87	Rec 27: Each Service should provide its software Using Commands with facilities to do comprehensive operational testing and life-cycle evaluation of extensions and changes.
SEI12	Merge agile and security best practices (e.g., integrate vulnerability scans into continuous integration process, leverage automated test cases for accreditation validation, adhere to secure coding standards).
SEI16	Employ concurrent testing and continuous integration.
USDS	When issuing a solicitation, it should explain the agile software development process. The solicitation should also describe the required testing of functional requirements and make it clear that testing should be integrated into each sprint cycle.
IDA18a	Analysis of planned operational test lengths indicates that the test scope is generally not long enough, demonstrate operational reliability with statistical confidence.

Primary Recommendation B3
ATO Reciprocity

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Create a mechanism for Authorization to Operate (ATO) reciprocity within and between programs, Services, and other DoD agencies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapon) systems, and Services.**
Stakeholders	DoD CIO, A&S, Service CIOs, DISA
Background	Current software acquisition practice emphasizes the differences among programs: perceptions around different missions, different threats, and different levels of risk tolerance mean that components, tools, and infrastructure that have been given permission to be used in one context are rarely accepted for use in another. The lack of ATO reciprocity drives each program to create their own infrastructure, repeating time- and effort-intensive activities needed to certify elements as secure for their own specific context.
Desired State	Modern software components, tools, and infrastructure, once accredited as secure within DoD, can be used appropriately and cost-effectively by multiple programs. Programs can spend a greater percentage of their budgets on developing software that adds value to the mission rather than spending time and effort on basic software infrastructure. Accreditation of COTS components is done once and then made available for use in other programs, as appropriate.
Role of Congress	N/A

	Draft Implementation Plan	**Lead Stakeholder**	**Target Date**
B3.1	Issue guidance making reciprocity the default practice in DoD with limited exceptions and update DoDI 8510.01 to reflect updated risk management framework. Exceptions should require signoff by the DoD CIO to discourage their use.	DoD CIO, with Service CIOs	Q3 FY19
B3.2	Establish DoD-wide repository for ATO artifacts with tools and access rules that enable Services to identify existing ATOs and utilize them when possible.	DoD CIO, with Service CIOs, DISA	Q4 FY19
B3.3	Implement procedures and access controls so that Authorizing Officials have visibility over other programs that are using compatible ATOs.	DoD CIO, with Service CIOs, DISA	Q2 FY20
B3.4	Implement mechanisms to allow FedRAMP and other non-DoD security certifications to be used for DoD ATO when appropriate based on intended use and environment.	DoD CIO, with FedRAMP	Q4 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Sec	As security is "baked in" to software during the development process, people must be educated about what that means as different tools look at different security aspects.
Sec	People must learn to appreciate that speed helps increase security. Security is improved when changes and updates can be made quickly to an application. Using automation, software can be reviewed quickly.
Sec	The AO must also be able to review documentation and make a risk decision quickly and make that decision on the process and not the product.

Related recommendations from previous studies

SEI12	Define criteria for reaccreditation early in the project.
SEI12	Leverage long accreditation approval wait time with frequent community previews.
SEI12	Don't apply all the information assurance controls blindly.

Additional Recommendation B4
Prioritize Modern Software Development Methods

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Prioritize secure, iterative, collaborative development for selection and execution of new software development programs (and software components of hardware programs), especially those using commodity hardware and operating systems.**
Stakeholders	USD(A&S), USD(C) DOT&E, SAE, Service Test Agencies
Background	Despite 37+ years of recommendations to stop using waterfall development for software programs, DoD continues to make use of hardware-centric approaches to development for software and software-intensive programs. While portions of the DoD 5000.02 Instructions apply to "Defense Unique Software Intensive" programs and "Incrementally Deployed Software Intensive" programs, these are still waterfall processes with years between the cycles of deployments (instead of weeks). These processes may be appropriate for some (though not all) embedded systems, but they are not the right approach for DoD-specific software running on commercial hardware and operating systems.
Desired State	DoD makes use of commercial software (without customization) whenever possible. When DoD-specific software development is required, contractors with demonstrated ability in the implementation of modern software development processes (e.g., Agile, DevOps, DevSecOps) are prioritized in the selection process and a contract structure is used that enables those methods to be successfully applied. For those applications for which hardware and software development are closely coupled, modern methods are still used as appropriate, especially in terms of information assurance testing.
Role of Congress	Congress should review metrics for performance on software (and software-intensive) programs with the expectation that modern methods of software are able to deliver software to the field quickly, provide rapid and continuous updates of capability, perform extensive automated testing, and track metrics for speed and cycle time, security, code quality, and useful capability.

Draft Implementation Plan		Lead Stakeholders	Target Date
B4.1	Establish metrics for evaluation of software development environments, following DSB 2018 recommendations on software factors and the DIB's "Development Environment" and "Agile BS Detector" concept papers.	USD(A&S)	Q3 FY19
B4.2	Issue Directive-Type Memorandum (DTM) to specify DoD's default software development approach is secure, iterative, modular, and collaborative.	USD(A&S)	Q3 FY19
B4.3	Create new DoD Instruction (DoDI) 5000.SW (or update DoDI 5000.02 and 5000.75) to specify DoD's default software development approach is secure, iterative, modular, and collaborative.	USD(A&S)	Q1 FY20

B4.4	Update courseware at Defense Acquisition University to specify DoD's default software development approach is secure, iterative, modular, and collaborative.	USD(A&S)	Q2 FY20

SWAP concept paper recommendations related to this recommendation

10C	Adopt a DevOps culture for software systems.
D&D	Require developers to meet with end users, then start small and iterate to quickly deliver useful code.
Visits	Adopt a DevOps culture: design, implement, test, deploy, evaluate, repeat.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Con	Use collaborative tools and libraries so that all content is available to all parties at all times.
Con	Use an agile process to manage structure and technical requirements.
Sec	As security is "baked in" to software during the development process, people must be educated about what that means as different tools look at different security aspects.
Wkf	Incentivize defense contractors to demonstrate their ability to leverage modern software methodologies.
Wkf	Contractor Reform. Adjust future NDAA's to add incentives for defense contractors to use modern development practices. (See FY18NDAA / §§873 & 874)

Related recommendations from previous studies

DSB87	Rec 12: Use evolutionary acquisition, including simulation and prototyping, as discussed elsewhere in this report, to reduce risk.
DSB87	Rec 17: DoD should devise increased productivity incentives for custom-built software contracts and make such incentivized contracts the standard practice.
DSB87	Rec 18: DoD should devise increased profit incentives on software quality.
DSB87	Rec 23: The USD(A) should update DoD Directive 5000.29, "Management of Computer Resources in Major Defense Systems," so that it mandates the iterative setting of specifications, the rapid prototyping of specified systems, and incremental development.
DSB87	Rec 24: DoD STD 2167 should be further revised to remove any remaining dependency on the assumptions of the "waterfall" model and to institutionalize rapid prototyping and incremental development.
DSB87	Rec 29: The USD(A) should develop economic incentives, to be incorporated into standard contracts, to allow contractors to profit from offering modules for reuse, even though built with DoD funds.
DSB87	Rec 30: The USD(A) should develop economic incentives, to be incorporated into all cost-plus standard contracts, to encourage contractors to buy modules and use them rather than build new ones.
DSB87	Rec 31: The USD(A) and ASD(Comptroller) should direct Program Managers to identify in their programs those systems, components, and perhaps even modules that may be expected to be acquired rather than built, and to reward such acquisition in the RFPs.
SEI12	Make sure Agile project teams understand the intent behind security requirements and organize the backlog accordingly.
SEI12	Ensure agile development processes produce and maintain "just enough" design documentation.

SEI12	Make sure there is at least one person with strong security analysis expertise on the Agile project team.
SEI12	Foster Agile project team and accrediting authority collaboration.
SEI12	Leverage unclassified environments for agile development and community previews.
SEI12	Agile and the information assurance community must join forces to continue improving information assurance processes.
GAO16a	Establish a department policy and process for the certification of major IT investments' adequate use of incremental development, in accordance with OMB's guidance on the implementation of FITARA.
NPS16a	Systems leveraging open architectures and incremental designs can focus on delivering initial capability quickly and then iterate improvements over time. The DoD can tailor acquisition processes for each major type of system to streamline each program's path through focused guidance.
SEI16	Ensure that the RFP contains language that allows the use of Agile. One promising approach that is consistent with Agile is to make sure the original contract is written with Agile in mind and contains sufficient flexibility to permit a wide scope of activity that could be modified as the situation develops. Agile program managers (PMs) could establish contract vehicles that allow for collaborative discussions to resolve and address dynamic developments over the life of the effort.
DSB18	Requests for proposals (RFPs) for acquisition programs entering risk reduction and full development should specify the basic elements of the software framework supporting the software factory, including code and document repositories, test infrastructure, software tools, check-in notes, code provenance, and reference and working documents informing development, test, and deployment.
DSB18	Rec 1: A key evaluation criterion in the source selection process should be the efficacy of the offeror's software factory.
DSB18	Rec 1a: Establish a common list of source selection criteria for evaluating software factories for use throughout the Department.
DSB18	Rec 1b: Competing contractors should have to demonstrate at least a pass-fail ability to construct a software factory.
DSB18	Rec 1c: Criteria for evaluating software factories should be reviewed and updated every five years.
DSB18	Rec 5e: Defense prime contractors must build internal competencies in modern software methodologies.
DSB18	Rec 2: The DoD and its defense industrial base partners should adopt continuous iterative development best practices for software, including through sustainment.
DSB18	Rec 2c: [DoD should] engage Congress to change statutes to transition Configuration Steering Boards (CSB) to support rapid iterative approaches (Fiscal Year (FY) 2009 National Defense Authorization Act (NDAA), Section 814).
DSB18	Rec 2d: [DoD] should require all programs entering Milestone B to implement these iterative processes for Acquisition Category (ACAT) I, II, and III programs.
DSB18	Rec 4a: For ongoing development programs, the USD(A&S) should immediately task the PMs with the PEOs for current programs to plan transition to a software factory and continuous iterative development.
DSB18	Rec 4c: Defense prime contractors should incorporate continuous iterative development into a long-term sustainment plan.
DSB18	Establish a common list of source selection criteria for evaluating software factories for use

	throughout the Department.
FCW18	Contractors would allow government to develop past performance reports with less documentation and less contractor opportunity to appeal their ratings.
USDS	Agile software development is the preferred methodology for software development contracts that contribute to the creation and maintenance of digital services, whether they are websites, mobile applications, or other digital channels.
USDS	Although Part 39 does not directly speak to agile software development practices, it endorses modular contracting principles where information technology systems are acquired in successive, interoperable increments to reduce overall risk and support rapid delivery of incremental new functionality.
USDS	With agile software development, requirements and priorities are captured in a high-level Product Vision, which establishes a high-level definition of the scope of the project, specifies expected outcomes, and produces high-level budgetary estimates.
USDS	Under agile software development, the Government retains the responsibility for making decisions and managing the process; it plays a critical role in the IPT as the Product Owner by approving the specific plans for each iteration, establishing the priorities, approving the overall plan revisions reflecting the experience from completed iterations, and approving deliverables.
USDS	OMB’s 2012 Contracting Guidance to Support Modular Development states that IDIQ contracts may be especially suitable for agile software development because they provide a high level of acquisition responsiveness, provide flexibility, and accommodate the full spectrum of the system life cycle that provides both development and operational products and services. BPAs may work with agile software development using modular contracting methods. Additionally, stand-alone contracts or single-award contracts may be used.
USDS	The Agile process works only if there are appropriate dedicated resources, as the process can be labor intensive. Agencies need to ensure adequate resources are applied to manage their contracts irrespective of the strategy used. Strong contract management ensures projects stay on course and helps prevent the agency from becoming overly reliant on contractors.

Additional Recommendation B5
Cloud Computing

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Remove obstacles to DoD usage of cloud computing on commercial platforms, including DISA CAP limits, lack of ATO reciprocity, and access to modern software development tools.**
Stakeholders	DoD CIO, Service CIOs, USD(A&S)
Background	Lack of ATO reciprocity and current DoD procedures for cloud are obstacles to leveraging modern infrastructure and tools.
Desired State	DoD developers and contractors are able to use modern cloud computing environments and commercial development tools quickly, with a single certification that is transferable to other groups using the same environment and tools.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholders	Target Date
B5.1	Rescind Cloud Access Point (CAP) policy and replace with policy that ensures security at scale (including end-to-end encryption).	DoD CIO	Q3 FY19
B5.2	In conjunction with primary Rec B3, allow transfer of ATOs for commercial platforms between programs and Services.	DoD CIO	Q3 FY19
B5.3	Create specifications and certification process for approval of standard development tools (w/ ATO reciprocity).	DoD CIO	Q4 FY19
B5.4	In conjunction with Rec B1, establish a common, enterprise ability to develop software solutions in the "easy-to-acquire-and-provision" cloud that is fully accredited by design of the process, tools, and pipeline.	USD(A&S)	Q1 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Include an approach for enterprise-level DevSecOps and other centralized infrastructure development and management, approach for shared services, and applications management.
Inf	Establish a DoD enterprise ability to procure, provision, pay for, and use cloud that is no different from the commercial entry points for cloud computing.
Inf	DoD should establish a common, enterprise ability to develop software solutions in the "easy-to-acquire-and-provision" cloud that is fully accredited by design of the process, tools, and pipeline.

Related recommendations from previous studies

Sec809	Rec. 43: Revise acquisition regulations to enable more flexible and effective procurement of consumption-based solutions.

Additional Recommendation B6
Certify Code/Toolchain

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Shift from certification of executables for low- and medium-risk deployments to certification of code/architectures and certification of the development, integration, and deployment toolchain.**
Stakeholders	USD(A&S), SAE, DoD CIO, Service CIO
Background	Today, the typical focus of security accreditation on programs is to certify each version of the code that is intended for release. This works against the goal of frequent updates because the more versions of software that are created, the more often the time and expense of the certification have to be borne by the program.
Desired State	The Department will accredit software infrastructures that are capable of producing quality code when used appropriately, enabling each version of the code produced on that infrastructure to be treated as certifiably secure (within appropriate limits, e.g., for versions that do not entail major architectural changes). With this change in certification, DoD will enable rapid fielding of mission-critical code at high levels of information assurance.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholders	Target Date
B6.1	Identify and use commercial certification procedures for security assessments and deployment mechanisms that can be used for DoD software programs.	CIO	Q4 FY19
B6.2	Identify three lead programs for initial implementation of certification procedures.	A&S, SAE	Q1 FY20
B6.3	Expand certification procedures to 10 additional sites, spanning all Services and multiple OSD offices; update procedures with each new certification to streamline process.	A&S, SAE with CIO	Q3 FY20
B6.4	Update DoDI 8501.01, Risk Management Framework for DoD Information Technology, to reflect revised certification procedures.	CIO with SAE, A&S	Q4 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Exempt the DoD from the Clinger Cohen Act, 40 U.S.C. 1401(3)
Inf	DoD should establish a common, enterprise ability to develop software solutions in the "easy-to-acquire-and-provision" cloud that is fully accredited by design of the process, tools, and pipeline.

Related recommendations from previous studies

SEI12	Use common operating environment (COE), software development toolkits (SDKs), and enterprise services to speed up accreditation time.

SEI12	Apply a risk-based, incremental approach to security architecture.
SEI12	Leverage design tactics such as layering and encapsulation to limit impact of change.
SEI13	For an SoS or for the more likely case of a system or component that participates in an existing SoS, an effective risk management approach should: • scale to size and complexity of systems of systems • incorporate dynamics • integrate across full life cycle: requirements to sustainment • focus on success as well as failure

Additional Recommendation B7
Hardware as a Consumable

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Plan and fund computing hardware (of all appropriate types) as consumable resources, with continuous refresh and upgrades to current, secure operating systems and platform components.**
Stakeholders	USD(A&S), SAE, DoD CIO, Service CIO, USD(C), CAPE
Background	Current information technology (IT) refreshes take 8-10 years from planning to implementation, which means that most of the time our systems are running on obsolete hardware that limits our ability to implement the algorithms required to provide the level of performance needed to stay ahead of our adversaries. Maintaining legacy code for different variants that have hardware capabilities ranging from 2 to 12 years old is an almost impossibly large spread of capability in computing, storage, and communications. From a contracting perspective, this change would require DoD to provide a stable annual budget that paid for new hardware and software capability (see Commandment #3), but this would very likely save money over the longer term.
Desired State	Whenever possible, applications are run in the cloud, so that algorithms can be run on the latest hardware and operating systems. For weapons systems, a continuous hardware refresh mentality is in place that enables software upgrades, crypto updates, and connectivity upgrades to be rapidly deployed across a fleet on an ongoing basis. The adoption rate of the latest hardware and operating system versions is tracked and targets are set for maintaining hardware and operating system "readiness." The paradigm for computing hardware from current Property, Plant, and Equipment categorization (as investments with depreciation schedules) is modified to treat hardware as an expense.
Role of Congress	Provide funding for ongoing replacement of computing hardware as a consumable with a 2–4-year lifetime. Track "readiness" of currently deployed software capability in part by measuring age of the hardware and operating systems on which software is being run.

Draft Implementation Plan		Lead Stakeholders	Target Date
B7.1	Establish funds for initial existing weapons platforms involving computing hardware to replace hardware every 2–4 years (like oil).	CIO with USD(C), SAE	Q1 FY20
B7.2	Establish draft guidance for determining when to update hardware and operating systems to balance cost with risk/capability.	CIO	Q2 FY20
B7.3	Work with FASAB to change audit treatment of software/IT with these goals: (1) Separate category for software instead of being characterized as Property, Plant, and Equipment; (2) Default setting that software is an expense, not	USD(A&S), in coordination with USD(C)	Q4 FY20

	an investment; and (3) there is no "sustainment" phase for software.		
B7.4	Modify DoD Financial Management Regulation (FMR) to capture changes in how hardware is purchased and retired from service.	USD(C)	Q1 FY21

SWAP concept paper recommendations related to this recommendation

10C	Move to a model of continuous hardware refresh in which computers are treated as a consumable with a 2-3 year lifetime.
Visits	Make use of platforms (hardware and software) that continuously evolve at the timescales of the commercial sector (3-5 years between HW/OS updates).

Related recommendations from previous studies

Sec809	Rec. 44: Exempt DoD from Clinger–Cohen Act Provisions in Title 40:
Sec809	Rec. 56: Use authority in Section 1077 of the FY 2018 NDAA to establish a revolving fund for information technology modernization projects and explore the feasibility of using revolving funds for other money-saving investments.

Primary Recommendation C1
Organic Development Groups

Line of Effort	Create new paths for digital talent (especially internal talent).
Recommendation	**Create software development units in each Service consisting of military and civilian personnel who develop and deploy software to the field using DevSecOps practices.**
Stakeholders	USD(A&S), USD(P&R), SAE, Service HR
Background	DoD's capacity to apply modern technology and software practices to meet its mission is required in order to remain relevant in increasingly technical fighting domains, especially against peer adversaries. While DoD has both military and civilian software engineers (often associated with maintenance activities), the IT career field suffers from a lack of visibility and support. The Department has not prioritized a viable recruiting strategy for technical positions, and there is no comprehensive training or development program that prepares the technical and acquisition workforce to adequately deploy modern software development tools and methodologies.
Desired State	DoD recruits, trains, and retains internal capability for software development, including by service members, and maintains this as a separate career track (like DoD doctors, lawyers, and musicians). Each Service has organic development units that are able to create software for specific needs and that serve as an entry point for software development capability in military and civilian roles (complementing work done by contractors). The Department's workforce embraces commercial best practices for the rapid recruitment of talented professionals, including the ability to onboard quickly and provide modern tools and training in state-of-the-art training environments. Individuals in software development career paths are able to maintain their technical skills and take on DoD leadership roles.
Role of Congress	Congress should receive regular "readiness" reports that include organic software development capability and provide budget required to maintain desired capability level and resources for modern software development.

	Draft Implementation Plan	**Lead Stakeholders**	**Target Date**
C1.1	Exercise existing acquisition and cybersecurity hiring authorities to increase the number of software developers in DoD programs with vacant positions.	SAE, PEO, with CIO (cyber excepted service ability)	Immediately
C1.2	Create new military occupational specialty (MOS) and core occupational series plus corresponding career tracks for each Service; use to grow digital talent for modern software development (e.g., Agile, DevSecOps).	J1 and comparable X1 for each Service with USD(P&R)	Q1 FY20
C1.3	Create regulations to allow standard identification, recruitment, and onboarding of experienced civilian software talent, especially on rotation from private sector roles.	USD(P&R)	Q1 FY20

C1.4	Create mechanism for tracking software development expertise and use as preferred experience for promotion into software engineer and acquisition roles.	A&S, CIO	Q2 FY20
C1.5	Obtain additional manpower authorizations for military and civilian SW developers.	USD(A&S), with USD(P&R), SAE	FY20, FY21
C1.6	Stand up one or more software factories within each Service, tied to field needs that can be satisfied through organic software development groups.	SAEs, with PEOs Digital	FY20 (create), FY21 (scale)

SWAP concept paper recommendations related to this recommendation

10C	Establish Computer Science as a DoD core competency.
D&D	Hire competent people with appropriate expertise in software to implement the desired state and give them the freedom to do so ("competence trumps process").

SWAP working group inputs (reflected in Appendix F) related to this recommendation

M&S	The definition of "core capabilities" in 10 USC 2464 should be revisited in light of warfighter dependence on software-intensive systems to determine the scope of DoD's core organic software engineering capability, and we should engage with Congress on the proposed revision to clarify the intent and extent of key terminology used in the current statute.
M&S	Revise industrial base policy to include software and DoD's organic software engineering capabilities and infrastructure. Start enterprise planning and investment to establish and modernize organic System Integration Labs (SILs), software engineering environments, and technical infrastructure; invest in R&D to advance organic software engineering infrastructure capabilities.
Wkf	Develop a core occupational series based on current core competencies and skills for software acquisition and engineering.
Wkf	Overhaul the recruiting and hiring process to use simple position descriptions, fully leverage hiring authorities, engage subject matter experts as reviewers, and streamline the onboarding process to take weeks instead of months.
Wkf	Embrace private-sector hiring methods to attract and onboard top talent from non-traditional backgrounds that may require special authorities to join the Department.
Wkf	Develop a strategic recruitment program that targets civilians, similar to the recruitment strategy for military members, [including] prioritizing experience and skills over cookie-cutter commercial certifications or educational attainment.
Wkf	The Department should incentivize and provide software practitioners access to modern engagement and collaboration platforms to connect, share their skills and knowledge, and develop solutions leveraging the full enterprise.
Wkf	Allow for greater private-public sector fluidity across the workforce while empowering the existing workforce to create a place where they want to work.
Wkf	Modify Title 10, §1596a to create a new Computer-language proficiency pay statute.
Wkf	Pilot a cyber-hiring team with the necessary authorities to execute report recommendations and that can serve as a Department-wide alternative to organization's traditional HR offices and will provide expedited hiring and a better candidate experience for top-tier cyber positions.

Related recommendations from previous studies

DSB87	Rec 26: Each Service should provide its software Product Development Division with the ability

	to do rapid prototyping in conjunction with users.
DSB87	Rec 36: Establish mechanisms for tracking personnel skills and projecting personnel needs.
DSB87	Rec 37: Structure some office careers to build a cadre of technical managers with deep technical mastery and broad operational overview.
SEI10	Improve compensation and advancement opportunities to increase tenure.

Primary Recommendation C2
Acquisition Workforce Training

Line of Effort	Create new paths for digital talent (especially internal talent).
Recommendation	**Expand the use of (specialized) training programs for CIOs, SAEs, PEOs, and PMs that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software.**
Stakeholders	USD(A&S), DoD CIO, SAE, Service CIO
Background	Acquisition professionals have been trained and had success in the current model, which has produced the world's best military, but this model is not serving well for software. New methodologies and approaches introduce unknown risks, and acquisition professionals are often not incentivized to make use of the authorities available to implement modern software methods. At the same time, senior leaders in DoD need to be more knowledgeable about modern software development practices so they can recognize, encourage, and champion efforts to implement modern approaches to software program management.
Desired State	Senior leaders, middle management, and organic and contractor-based software developers are aligned in their view of how modern software is procured and developed. Acquisition professionals are aware of all of the authorities available for software programs and use them to provide flexibility and rapid delivery of capability to the field. Program leaders are able to assess the status of software (and software-intensive) programs and spot problems early in the development process, as well as provide continuous insight to senior leadership and Congress. Highly specialized requirements are scrutinized to avoid developing custom software when commercial offerings are available that are less expensive and more capable.
Role of Congress	Prioritize experience with modern software development environments in approval of senior acquisition leaders.

Draft Implementation Plan		Lead Stakeholders	Target Date
C2.1	Leverage existing training venues to add content about modern software development practices.	USD(A&S), SAEs with DAU	Q4 FY19
C2.2	Create and provide training opportunities via boot camps and rotations for acquisition professionals to obtain hands-on experience in DevSecOps programs.	A&S with SAEs, USD(P&R)	FY20 (MVP)[2] FY21 (scale)
C2.3	Develop additional training opportunities for key leaders about modern software development practices.	USD(A&S), SAE, DAU	Q2 FY20
C2.4	Create software continuing education programs and requirements for CIOs, SAEs, PEOs, and PMs, modeled after MCLE (Minimum Continuing Legal Education) for lawyers.	A&S, DAU	Q3 FY20

[2] Minimum viable product (first useful iteration)

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Con	Provide training to KOs, PMs, and leadership to understand the value and methods associated with Agile and modular implementation.
Wkf	Create a software acquisition workforce fund (similar to the existing Defense Acquisition Workforce Development Fund (DAWDF)) ... to hire and train a cadre of modern software acquisition experts.
Wkf	Pilot development programs that provide comprehensive training for all software acquisition professionals, developers, and associated functions.
Con	Provide training to KOs, PMs, and leadership to understand the value and methods associated with Agile and modular implementation.
Con	Educate PMs and KOs on Open Source, proprietary, and government-funded code.

Related recommendations from previous studies

DSB09	All CIOs should approve IT acquisition program manager training and certification and advise the personnel selection process.
DSB09	The USD(AT&L) shall direct the Defense Acquisition University, in coordination with the Information Resources Management College, to integrate the new acquisition model into their curriculum.
DSB18	USD(A&S) should task the PMs of programs that have transitioned successfully to modern software development practices to brief best practices and lessons learned across the Services.
DSB18	Rec 5d: The USD(A&S) and the USD(R&E) should direct the Defense Acquisition University (DAU) to establish curricula addressing modern software practices leveraging expertise from the DDS, the FFRDCs, and the University Affiliated Research Centers (UARCs).
DSB18	Rec 5g: DoD career functional Integrated Product Team (IPT) leads should immediately establish a special software acquisition workforce fund modeled after the Defense Acquisition Workforce Development Fund (DAWDF), the purpose of which is to hire and train a cadre of modern software acquisition experts across the Services.
DSB18	Rec 5h: PMs should create an iterative development IPT with associated training. The Service Chiefs should delegate the role of Product Manager to these IPTs.
DSB18	Rec 5b: The Service Acquisition Career Managers should develop a training curriculum to create and train [a] cadre [of] software-informed PMs, sustainers, and software acquisition specialists.
Sec809	Rec 27: Improve resourcing, allocation, and management of the Defense Acquisition Workforce Development Fund (DAWDF).
Sec809	Rec. 59: Revise the Defense Acquisition Workforce Improvement Act to focus more on building professional qualifications.

Additional Recommendation C3
Increase PMO Experience

Line of Effort	Create new paths for digital talent (especially internal talent).
Recommendation	**Increase the knowledge, expertise, and flexibility in program offices related to modern software development practices to improve the ability of program offices to take advantage of software-centric approaches to acquisition.**
Stakeholders	USD(A&S), SAE, USD(P&R)
Background	Acquisition professionals do not always have experience and insights into modern software development environments, especially in the opportunities (and limitations) for continuous integration/continuous delivery (CI/CD), automated testing (including security testing), and modern cloud-computing architectures. New methodologies and approaches introduce unknown risks, while the old acquisition and development approaches built the world's best military. Program offices are not incentivized to adopt new approaches to acquisition and implementation of software, and inertia represents a barrier to change.
Desired State	Program management offices have staff available with experience in modern software development environments and who are able to make creative (but legal) use of available authorities for acquisition of software to fit the needs of modern software development solutions. Management of most types of software relies on (continuous) measurement of capability delivered to the field rather than being tied to satisfaction of objectives. Time and cost are used as constraints with schedule of delivery of features replanned at each iteration cycle based on warfighter/user feedback.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholders	Target Date
C3.1	Establish list of skills and experience needed by program office staff to be considered "fully staffed" for a software program.	A&S with SAEs, USD(P&R)	Q4 FY19
C3.2	Modify Position Descriptions for those in leadership positions in software acquisition programs to prioritize and reward prior experience in software development.	USD(A&S), SAE, Service HR	Q1 FY20
C3.3	Create and provide training opportunities via boot camps and rotations for acquisition professionals to obtain hands-on experience in DevSecOps programs.	A&S with SAEs, USD(P&R)	Q2 FY20 (MVP)[3] FY21 (scale)
C3.4	Modify PM training requirements to obtain DAU Level IIII certification to include hands-on experience with modern software development.	USD(A&S), DAU	Q3 FY20
C3.5	Evaluate readiness level of software (and software-intensive) program offices by comparing experience/skill sets available with the list of needed skills from C3.1	A&S with SAEs, USD(P&R)	Q4 FY20 (MVP) FY21 (scale)

[3] Minimum viable product (first useful iteration)

	(hint: consider tracking those skills sets; see Action C1.2).		

SWAP concept paper recommendations related to this recommendation

D&D	Hire competent people with appropriate expertise in software to implement the desired state and give them the freedom to do so ("competence trumps process").

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Lead tester from either DOT&E or JITC (preferably both, if JITC is being used as test org) must be a subject matter expert in the subject being tested, similar to how qualified test pilots run test flights (health records, financial systems, etc.).
Wkf	Empower a small cadre of Highly Qualified Experts (HQEs) and innovative Department employees to execute the changes from this report.
Wkf	Establish a software acquisition workforce fund, similar to the Defense Acquisition Workforce Development Fund (DAWDF), but the primary use will be for hiring and training a cadre of modern software acquisition experts.
Wkf	Provide Agile, Tech, and DevSecOps coaches in Program Offices to support transformations, adoption of modern software practices, and share lessons across the enterprise.
Wkf	Develop a core occupational series based on current core competencies and skills for software acquisition and engineering.
Wkf	Modify the existing language in 5 USC Part III, Subpart D, Chapter 53 to add a pilot training program for all software acquisition professionals, developers, and associated functions.
Wkf	Modify Title 10 §1746 to include authorities for the development of a modern academy under the Defense Acquisition University; the HQE cadre (see above) should lead its development. Note: Tied with FY18 NDAA §891 (training on agile and iterative development methods.)
Wkf	Modify Title 5, §§3371-3375 to expand the Inter-Government Personnel Act and allow more civil service employees to work with non-Federal Agencies and Educational Institutions. In addition, modify Title 10, §1599g to expand the Public-Private Talent Exchange Program and modify the language to reduce the "repayment" period from 1:2 to 1:1 ratio.

Related recommendations from previous studies

OSD06	Establish a consistent definition of the acquisition workforce with the Under Secretary of Defense for Acquisition Technology and Logistics, working with the Service Secretaries to include in that definition all acquisition-related budget and requirements personnel.
OSD06	Immediately increase the number of federal employees focused on critical skill areas, such as program management, system engineering, and contracting. The cost of this increase should be offset by reductions in funding for contractor support.
OSD06	Request that the White House Liaison Office create a pool of acquisition-qualified, White House pre-cleared, non-career senior executives and political appointees to fill executive positions, to provide leadership stability in the Acquisition System.
OSD06	Seek legislation to retain high-performance military personnel in the acquisition workforce to include allowing military personnel to remain in uniform past the limitations imposed by the Defense Officer Personnel Management Act and augment

	their pay to offset the “declining marginal return” associated with retired pay entitlement.
OSD06	Realign responsibility, authority, and accountability at the lowest practical level of authority by reintegrating the Services into the acquisition management structure.
OSD06	Fully implement the intent of the Packard Commission. Create a streamlined acquisition organization with accountability assigned and enforced at each level.
SEI10	Assign PMs, DPMs, and other key positions for the program’s duration and into deployment. Use civilians if military rotations are not amenable.
SEI10	Improve qualifications of acquisition staff, emphasizing software expertise.
CSIS15	Rapid acquisition succeeds when senior leaders are involved in ensuring that programs are able to overcome the inevitable hurdles that arise during acquisition and empower those responsible with achieving the right outcome with the authority to get the job done while minimizing the layers in between.
CSIS15	Rapid acquisition is fundamentally an ongoing dialogue between the acquisition and operational communities about what the real needs of the warfighter are and what the art of the possible is in addressing them.
SEI15	5. Government Personnel Experience. Government personnel with extensive experience in developing and managing acquisition strategy and technical architecture should be dedicated and available to a program throughout its duration.
NPS16a	The growth of rapid acquisition organizations gives acquisition executives new avenues to meet their top priority and rapid capability demands. However, these organizations may also have negative effects on traditional acquisition organizations. The DoD’s top talent will flock to the rapid acquisition organizations so that they can work on high-priority programs with minimal restrictions and likely achieve greater success.
NPS16a	Contracting Officers (COs) must function as strategic partners tightly integrated into the program office, rather than operate as a separate organization that simply processes the contract paperwork.
NPS16b	Culturally, the acquisition community needs to embrace the available tools as opportunities, while being selective with procurement methods and adaptive to the market environment.
GAO17	Empower program managers to make decisions on the direction of the program and to resolve problems and implement solutions.
GAO17	Hold program managers accountable for their choices.
GAO17	Require program managers to stay with a project to its end.
GAO17	Encourage program managers to share bad news, and encourage collaboration and communication.
DSB18	Rec 5a: The service acquisition commands (e.g., the LCMC, the NAVAIR, the U.S. Naval Sea Systems Command (NAVSEA), and the AMC) need to develop workforce competency and a deep familiarity of current software development techniques.
DSB18	Rec 5a.2: Services acquisition commands should use this cadre early in the acquisition process to formulate acquisition strategy, develop source selection criteria, and evaluate progress.
DSB18	Over the next two years, the service acquisition commands need to develop workforce competency and a deep familiarity of current software development techniques.

Sec809	Rec. 40: Professionalize the requirements management workforce.
Sec809	Rec. 46: Empower the acquisition community by delegating below-threshold reprogramming decision authority to portfolio acquisition executives.

Additional Recommendation C4
Recruiting (Transient) Digital Talent

Line of Effort	Create new paths for digital talent (especially internal talent).
Recommendation	**Restructure the approach to recruiting digital talent to assume that the average tenure of a talented engineer will be 2-4 years, and make better use of HQEs, IPAs, special hiring authorities, reservists, and enlisted personnel to provide organic software development capability, while at the same time incentivizing and rewarding internal talent.**
Stakeholders	USD(A&S), USD(P&R), SAE, A-1/G-1/N-1
Background	Current DoD personnel systems assume that military and government employees will "grow through the ranks" and that individuals will stay in government service for long periods of time. The attractions of the private sector create personnel-retention challenges that are not likely to be overcome, so a different approach is needed.
Desired State	DoD leverages all individuals who are willing to serve, whether for a long period or a short period, and amplifies the ability of individuals to make a contribution during their time in government. Internal talent is recognized and retained through merit-based systems of promotion and job assignment.
Role of Congress	Support and encourage the use of existing authorities to hire digital talent in creative ways that match the intent of Congress and solve the need for more flexible arrangements in which talented individuals move in and out of government service (without creating unnecessary barriers).

	Draft Implementation Plan	Lead Stakeholders	Target Date
C4.1	Exercise existing hiring authorities to increase the number of highly skilled software people in DoD program, such as the Cyber Excepted Workforce.	SAE, PEO, CIO	Starting now
C4.2	In conjunction with Recs C1, create a database of individuals in enlisted, officer, reserve, and civilian positions with software development skills and experience for internal recruiting use to software squadrons & PAOs.	USD(P&R) and Service equivalents	Q3 FY19
C4.3	Within organic software programs, create processes for maintaining release cadence under the assumption of up to 25% turnover per year.	PMOs	Q4 FY19
C4.4	Require software-intensive project proposals to include a plan for maintaining cadence-related metrics in the face of up to 25% turnover of staff.	SAEs	Q4 FY19
C4.5	Identify bottlenecks in providing security clearances for software developers and target granting of interim clearances within 1 month of start date.	DSS	Q1 FY20
C4.6	Revise GS and military promotion guidelines for software developers to allow rapid promotion of highly qualified individuals with appropriate skills, independent of "time in grade."	USD(P&R)	FY20 for FY21 NDAA

C4.7	Obtain additional funding for military, civilian SW developers, including existing personnel, HQEs, IPAs, reservists, and direct commissioning.	USD(A&S), USD(P&R), SAE	FY21

SWAP concept paper recommendations related to this recommendation

10C	Establish Computer Science as a DoD core competency.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Wkf	Develop a core occupational series based on current core competencies and skills for software acquisition and engineering.
Wkf	Overhaul the recruiting and hiring process to use simple position descriptions, fully leverage hiring authorities, engage subject matter experts as reviewers, and streamline the onboarding process to take weeks instead of months.
Wkf	Embrace private-sector hiring methods to attract and onboard top talent from non-traditional backgrounds that may require special authorities to join the Department.
Wkf	Develop a strategic recruitment program that targets civilians, similar to the recruitment strategy for military members, [including] prioritizing experience and skills over cookie-cutter commercial certifications or educational attainment.

Related recommendations from previous studies

DSB87	Rec 34: Do not believe that DoD can solve its skilled personnel shortage; plan how best to live with it, and how to ameliorate it.
SEI10	Divide large acquisition development efforts into multiple smaller, shorter duration programs.
Sec809	Rec. 45: Create a pilot program for contracting directly with information technology consultants through an online talent marketplace.

Primary Recommendation D1
Source Code Access

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Require access to source code, software frameworks, and development toolchains—with appropriate IP rights—for DoD-specific code, enabling full security testing and rebuilding of binaries from source.**
Stakeholders	USD(A&S), CIO, SAE
Background	For many DoD systems, source code is not available to DoD for inspection and testing, and DoD relies on suppliers to write code for new compute environments. As code ages, suppliers are not required to maintain codebases without an active development contract, and "legacy" code is not continuously migrated to the latest hardware and operating systems.
Desired State	DoD has access to source code for DoD-specific software systems that it operates and uses to perform detailed (and automated) evaluation of software correctness, security, and performance, enabling more rapid deployment of both initial software releases and (most importantly) upgrades (patches and enhancements). DoD is able to rebuild executables from scratch for all of its systems, and it has the rights and ability to modify (DoD-specific) code when new conditions and features arise. Code is routinely migrated to the latest computing hardware and operating systems and routinely scanned against currently known vulnerabilities. Modern IP language is used to ensure that the government can use, scan, rebuild, and extend purpose-built code, but contractors are able to use licensing agreements that protect any IP that they have developed with their own resources. Industry trusts DoD with its code and has appropriate IP rights for internally developed code.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholders	Target Date
D1.1	Work with industry to modernize policies for software code ownership, licensing, and purchase. See 2018 Army IP directive as an example.	USD(A&S)	Q3 FY19
D1.2	Modify FAR/DFARS guidance to require software source code deliverables for GOTS and for government-funded software development. Obtain rights for access to source code for COTS wherever possible (and useful).	USD(A&S)	Q3 FY20
D1.3	Modify DoDI 5000.02 and DoDI 5000.75 to make access to code and development environments the default.	USD(A&S)	Q3 FY20
D1.4	Develop a comprehensive source-code management plan for DoD including the safe and secure storage, access control, testing, and field of use rights.	USD(A&S), with CIO	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Every purpose-built DoD software system should include source code as a deliverable.

D&D	Require source code as a deliverable on all purpose-built DoD software contracts. Continuous development and integration, rather than sustainment, should be a part of all contracts. DoD personnel should be trained to extend the software through source code or API access.

Related recommendations from previous studies

DSB87	Rec 22: DoD should follow the concepts of the proposed FAR 27.4 for data rights for military software, rather than those of the proposed DoD 27.4, or it should adopt a new "Rights in Software" Clause as Recommended by Samuelson, Deasy, and Martin in Appendix A6.
DSB18	Rec 6b: Availability, cost, compatibility, and licensing restrictions of [the proposed software factory] framework elements to the U.S. Government and its contractors should be part of the selection criteria for contract award.
DSB18	Rec 6c: All documentation, test files, coding, application programming interfaces (APIs), design documents, results of fault, performance tests conducted using the framework, and tools developed during the development, as well as the software factory framework, should be delivered to the U.S. Government at each production milestone; OR escrowed and delivered at such times specified by the U.S. Government (i.e., end of production, contract reward).
DSB18	Rec 6d: Selection preference should be granted based on the ability of the United States to reconstitute the software framework and rebuild binaries, re-run tests, procedures, and tools against delivered software and documentation.

Primary Recommendation D2
Security Considerations

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough.**
Stakeholders	USD(A&S), CIO, DDS, SAE, DDR&E(AC), DOT&E
Background	Current DoD systems often rely on security-at-the-perimeter as a means of protecting code for unauthorized access. If this perimeter is breached, then a large array of systems can be compromised. Multiple GAO, DoDIG, and other reports have identified cybersecurity as a major issue in acquisition programs.
Desired State	DoD systems use a zero-trust security model in which it is not assumed that anyone who can gain access to a given network or system should have access to anything within that system. Regular and automated penetration testing is used to track down vulnerabilities, and red teams are engaged to attempt to breach our systems before our adversaries do.
Role of Congress	Review (classified) reporting of vulnerabilities identified in DoD systems and provide the resources required to ensure that hardware and operating systems are at current levels (see Recommendation B7, Hardware as a Consumable).

Draft Implementation Plan		Lead Stakeholders	Target Date
D2.1	Adopt standards for secure software development and testing that use a zero-trust security model.	CIO, with DDS	Q3 FY19
D2.2	Develop, deploy, and require the use of IA-accredited (commercial) development tools for DoD software development.	CIO, PEO Digital	Q4 FY19
D2.3	Establish automated and red-team based penetration testing as part of OT&E evaluation (integrated with program development).	DOT&E	Q1 FY20
D2.4	Establish a red team responsible for ongoing vulnerability testing against any defense software system.	CIO with DDS	Q2 FY20
D2.5	Establish security as part of the selection criteria for software programs.	A&S with CIO, SAEs	Q3 FY20

SWAP concept paper recommendations related to this recommendation

10C	Only run operating systems that are receiving (and utilizing) regular security updates for newly discovered security vulnerabilities.
10C	Data should always be encrypted unless it is part of an active computation.
D&D	Create automated test environments to enable continuous (and secure) integration and deployment to shift testing and security left.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Sec	People must learn to appreciate that speed helps increase security. Security is improved when

	changes and updates can be made quickly to an application. Using automation, software can be reviewed quickly.
Sec	The AO must also be able to review documentation and make a risk decision quickly and make that decision on the process and not the product.
T&E	Establish a statutory “Live Fire” requirement on software-intensive systems as there is on “Covered Systems” for protecting our warfighters from kinetic threats. “Shoot at it” before design is complete and certainly before it is put into the operational environment.
T&E	Establish a federation of state-of-the-art cyber testing capabilities from non-profit institutions to support trusted, survivable, and resilient defense systems and ensure the security of software and hardware developed, acquired, maintained, and used by the DoD.
T&E	Establish cybersecurity as the “4th leg” in measurement of Acquisition system/program performance: Cost, Schedule, Performance, Cybersecurity.
T&E	Develop mechanisms to enforce existing software and cybersecurity policies (from cradle-to-grave) that are not (now) being adequately enforced.
T&E	Ensure each DoD Component is responsible for representing its own forces and capabilities in a digital modeling environment (e.g., M&S and digital twin), making them available to all other DoD users, subject to a pre-defined architecture and supporting standards. DIA will represent threat forces and capabilities in a digital form consistent with this architecture/standards. Programs are required to use DIA-supplied threat models, unless sufficient justification is provided to use other.

Related recommendations from previous studies

DSB09	In the Services and agencies, the CIOs should also have strong authorities and responsibilities for system certification, compliance, applications development, and innovation.
DSB09	The DOD CIO, supported by CIOs in the Services and agencies, should be responsible for certifying that systems and capabilities added to the enterprise do not introduce avoidable vulnerabilities that can be exploited by adversaries.
Sec809	Rec. 77: Require role-based planning to prevent unnecessary application of security clearance and investigation requirements to contracts.

Primary Recommendation D3
Software Features

Line of Effort	Change the practice of how software is procured and developed.		
Recommendation	**Shift from the use of rigid lists of requirements for software programs to a list of desired features and required interfaces/characteristics to avoid requirements creep, overly ambitious requirements, and program delays.**		
Stakeholders	USD(A&S), Joint Staff, SAEs		
Background	Current DoD requirements processes significantly impede DoD's ability to implement modern SW development practices by spending years establishing program requirements and insisting on satisfaction of requirements before a project is considered "done." This impedes rapid implementation of features that are of the most use to the user.		
Desired state	Rather than a list of requirements for every feature, programs should establish a minimum set of requirements required for initial operation, security, and interoperability and place all other desired features on a list that will be implemented in priority order, with the ability for DoD to redefine priorities on a regular basis.		
Role of Congress	Modify relevant statutes to allow the use of evolving features over rigid requirements and develop alternative methods for obtaining information on program status (See Rec A2, Action A2.4).		
Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
D3.1	Modify requirements guidance by memo to shift from a list of requirements for software to a list of desired features and required interfaces/characteristics.	USD(A&S), with CMO	Q4 FY19
D3.2	Update CJCSI 3170.01H (JCIDS requirements process) to reflect contents of guidance memos.	Joint Staff	Q1 FY20
D3.3	Modify DoDI 5000.02 and DoDI 5000.75 (or integrate into new DoDI 5000.SW).	USD(A&S)	Q2 FY20
D3.4	Define and use new budget exhibits for software programs using evolving lists of features in place of requirements (see also Rec A2).	USD(A&S), with USD(C), CAPE, HAC-D, SAC-D	Q3 FY20

SWAP concept paper recommendations related to this recommendation

10C	Adopt a DevOps culture for software systems.
10C	All software procurement programs should start small, be iterative, and build on success—or be terminated quickly.
D&D	Accept 70% solutions in a short time (months) and add functionality in rapid iterations (weeks).

Related recommendations from previous studies

SEI01	Ensure that all critical functional and interoperability requirements are well specified in the contract (statement of work, Statement of Objectives).
SEI01	Handle requirements that have architectural consequences as systems engineering issues—up front.

SEI12	Ensure requirements prioritization of backlog considers business value and risk.
GAO17	Match requirements to resources—that is, time, money, technology, and people—before undertaking new development efforts.

Additional Recommendation D4
Continuous Metrics

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Create and use automatically generated, continuously available metrics that emphasize speed, cycle time, security, user value, and code quality to assess, manage, and terminate software programs (and software components of hardware programs).**
Stakeholders	USD(A&S), CAPE, USD(C), SAE, Service Cost Orgs
Background	Current program reporting requirements are largely manual and time consuming, and they provide limited insight into the SW health of a program. New metrics are required that match the DevSecOps approach of continuous capability delivery and maintenance and provide continuous insight into program progress.
Desired State	Program oversight will re-focus on the value provided by the software as it is deployed to the warfighter/user, and it will rely more heavily on metrics that can be collect in an automated fashion from instrumentation on the DevSecOps pipeline and other parts of the infrastructure. Specific metrics will depend on the type of software rather than a one-size-fits-all approach.
Role of Congress	N/A (but see Rec A3)

Draft Implementation Plan		Lead Stakeholder	Target Date
D4.1	Modify acquisition policy guidance to specify use of automatically generated, continuously available metrics that emphasize speed, cycle time, security, and useful functionality.	USD(A&S)	Q3 FY19
D4.2	Modify cost estimation policy guidance to specify use of automatically generated, continuously available metrics that emphasize speed, cycle time, security, and code.	CAPE	Q3 FY19
D4.3	Develop specific measure of software quality, value, and velocity and the tools to implement the automatic generation and reporting.	DDS, with CAPE, CIO, USD(C)	Q4 FY19
D4.4	Modify DoDI 5000.02, DoDI 5000.75, and DoDI 5105.84 to reflect use of updated methods and remove earned value management (EVM) for software programs.	A&S	Q1 FY20

SWAP working group inputs related to this recommendation

Acq	Revise DFARS Subpart 234.201, DoDI 5000.02 Table 8, and OMB Circular A-11 to remove EVM requirement.
Con	Allow for documentation and reporting substitutions to improve agility (agile reporting vs. EVM) (Cultural and EVM Policy).
Con	Establish a clear definition of done targets for software metrics for defense systems of different types (code coverage, defect rate, user acceptance).
D&M	Congress could establish, via an NDAA provision, new data-driven methods for governance of software development, maintenance, and performance. The new approach should require on-demand access to standard (and perhaps real-time) data with reviews occurring on a standard

	calendar, rather than the current approach of manually developed, periodic reports.
D&M	DoD must establish the data sources, methods, and metrics required for better analysis, insight, and subsequent management of software development activities. This action does not require Congressional action but will likely stall without external intervention and may require explicit and specific Congressional requirements to strategically collect, access, and share data for analysis and decision making.
T&E	Establish requirements for government-owned software to be instrumented such that critical monitoring functions (e.g., performance, security) can be automated as much as possible, persistently available, and such that authoritative data can be captured, stored, and reused in subsequent testing or other analytic efforts.

Related recommendations from previous studies

DSB87	Rec 19: DoD should develop metrics and measuring techniques for software quality and completeness and incorporate these routinely in contracts.
DSB87	Rec 20: DoD should develop metrics to measure implementation progress.
Sec809	Rec 19: Eliminate the Earned Value Management (EVM) mandate for software programs using agile methods.
MITRE18	Elevate Security as a Primary Metric in DoD Acquisition and Sustainment.

Additional Recommendation D5
Iterative Development

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Shift the approach for acquisition and development of software (and software-intensive components of larger programs) to an iterative approach: start small, be iterative, and build on success or be terminated quickly.**
Stakeholders	USD(A&S), CAPE, USD(C), USD(P&R), SAE, Service HR
Background	Current-language DoD acquisition guidance is largely based around a hardware-centric paradigm, with a well-defined start and end and sequential life cycle activities.
Desired State	Software acquisition in DoD follows an iterative approach, with frequent deployment of working software, supported by a DevSecOps infrastructure that enables speed through continuous integration/continuous delivery. Software projects are continuously evaluated by the quality of their deployed capability and are terminated early if they are found to be non-performant. Software is never "complete." Programs are viewed as an ongoing service rather than a discrete project.
Role of Congress	Authorize and track software programs that utilize iterative methods of development rather than milestone-based progress. Recognizing that the distinction between RTD&E, procurement, and sustainment is not appropriate for many types of software, identify new ways of providing oversight while enabling much more flexibility for programs.

Draft Implementation Plan		Lead Stakeholders	Target Date
D4.1	Issue guidance immediately changing the default for acquisition programs to use iterative software development methodologies (e.g., DevSecOps, agile development).	USD(A&S)	Q3 FY19
D4.2	Issue guidance changing the default for acquisition programs to be iterative software development methodologies.	SAE	Q3 FY19
D4.6	Select three software programs widely perceived to be in dire straits and go through a program termination exercise to identify new potential solutions and the blockers to more effectively terminating non-performing programs.	USD A&S	Q1 FY20
D4.3	Modify DoDI 5000.02 and 5000.75 (or DoDI 5000.SW) to reflect more iterative approaches for software development.	USD(A&S)	Q2 FY20
D4.4	Modify Service acquisition policy to reflect more iterative approaches for software development.	SAE	Q2 FY20
D4.5	Build a Congressional Reporting Dashboard that would be available to the four Defense Committees to show the progress of DoD and Services DevSecOps programs, including speed and cycle time, code quality, security, and user satisfaction.	USD(A&S)	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Adopt a DevOps culture for software systems.
10C	All software procurement programs should start small, be iterative, and build on success—or be terminated quickly.
D&D	Accept 70% solutions in a short time (months) and add functionality in rapid iterations (weeks).
D&D	Take advantage of the fact that software is essentially free to duplicate, distribute, and modify.
D&D	Treat software development as a continuous activity, adding functionality continuously across its life cycle.
Visits	Spend time upfront getting the architecture right: modular, automated, secure.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Con	Treat procurements as investments; “What would you pay for a possible initial capability?”
Con	Leverage incentives to make smaller purchases to take advantage of simplified acquisition procedures.
Con	Use modular contracting to allow for regular investment decisions based on perceived value.
Con	Streamline acquisition processes to allow for replacing poorly performing contractors.
T&E	Develop the enterprise knowledge management and data analytics capability for rapid analysis/ presentation of technical RDT&E data to support deployment decisions at each iterative cycle.

Related recommendations from previous studies

OSD06	Change DoD’s preferred acquisition strategy for developmental programs from delivering 100 percent performance to delivering useful military capability within a constrained period of time, no more than 6 years from Milestone A. This makes time a Key Performance Parameter.
OSD06	Direct changes to the DoD 5000 series to establish Time Certain Development as the preferred acquisition strategy for major weapons systems development programs.
GAO17	Follow an evolutionary path toward meeting mission needs rather than attempting to satisfy all needs in a single step.
GAO17	Ensure that critical technologies are proven to work as intended before programs begin. Assign more ambitious technology development efforts to research departments until they are ready to be added to future generations (or increments) of a product.
NDU17	Prioritize technical performance and project schedules over cost. Maintain aggressive focus on risk identification and management across all elements of the open system, and resolve technical problems as rapidly as possible.
DSB18	Rec 2a: [DoD programs should] develop a series of viable products (starting with MVP) followed by successive next viable products (NVPs).
DSB18	Rec 2b: [DoD programs should] establish MVP and the equivalent of a product manager for each program in its formal acquisition strategy and arrange for the warfighter to adopt the initial operational capability (IOC) as an MVP for evaluation and feedback.
DSB18	Rec 3a: The MDA (with the DAE, the SAE, the PEO, and the PM) should allow multiple vendors to begin work. A down-selection should happen after at least one vendor has proven they can do the work, and should retain several vendors through development to reduce risk, as feasible.

Additional Recommendation D6
Software Research Portfolio

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Maintain an active research portfolio into next-generation software methodologies and tools, including the integration of ML and AI into software development, cost estimation, security vulnerabilities, and related areas.**
Stakeholders	USD(R&E), USD(A&S)
Background	Software is essential to national security, and DoD needs to stay ahead of adversaries on emerging SW development practices.
Desired State	DoD benefits from a feedback loop between research and practice, in areas important to retaining the ability to be able to field innovations in software-enabled technologies. Mission needs and a practical understanding of the acquisition ecosystem inform research programs in emerging technologies. Results emerging from research impact the department's warfighting and other systems thanks to high-quality and modular software systems, a DevSecOps infrastructure capable of moving fast, and other enablers. Model-based engineering of software (including "digital twin" approaches) is routinely used to speed development and increase security.
Role of Congress	N/A

	Draft Implementation Plan	Lead Stakeholders	Target Date
D6.1	Designate a responsible person or organization to coordinate software research activities.	USD(R&E)	Q4 FY19
D6.2	Stand up a Chief Engineer for Software to direct the implementation of next-generation software methodologies and tools.	SAEs	Q4 FY19
D6.4	Direct the Principal Civilian Deputy to the SAE to implement the acquisition infrastructure for DevSecOps, allowing quick incorporation of new technologies into DoD systems, implemented by someone with software development experience.	SAEs	Q4 FY19
D6.6	Create a documented DoD Software strategy, perhaps patterned on the DoD cyber strategy,[4] with ties to other existing national and DoD research strategies, and with involvement of A&S and the Services.	USD(R&E)	Q4 FY19
D6.5	Make acquisition data collected continuously from DevSecOps infrastructure and tools available to researchers with appropriate clearances, as a testbed for AI, ML, or other technologies. (See Recs A3, D4)	USD(A&S)	Q4 FY20

Related recommendations from previous studies

DSB18	Rec 7a: Under the leadership and immediate direction of the USD(R&E), the Defense Advanced

[4] https://media.defense.gov/2018/Sep/18/2002041658/-1/-1/1/CYBER_STRATEGY_SUMMARY_FINAL.PDF

	Research Projects Agency (DARPA), the SEI FFRDC, and the DoD laboratories should establish research and experimentation programs around the practical use of machine learning in defense systems with efficient testing, independent verification and validation (IVV), and cybersecurity resiliency and hardening as the primary focus points.

Additional Recommendation D7
Transition Emerging Tools and Methods

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Invest in transition of emerging tools and methods from academia and industry for creating, analyzing, verifying, and testing of software into DoD practice (via pilots, field tests, and other mechanisms).**
Stakeholders	USD(A&S), USD(R&E), Service Digital PEOs
Background	Software is essential to national security, and DoD needs to stay ahead of adversaries in implementing emerging SW development practices. Research work at universities and in the private sector, along with best practice implementation from the private sector, can provide valuable tools and methods to be deployed across DoD.
Desired State	Development and test technology, tools, and methods that are being created and used in the private sector and academia and are known and visible to the PEOs Digital who enable transition into Service programs. DoD labs are investing internally and externally to mature software development and analysis tools.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholders	Target Date
D7.1	Create a community of practice, code repositories, and other mechanisms to keep all practitioners knowledgeable about the latest trends and capabilities in software development, testing, and deployment.	USD(A&S)	Q4 FY19
D7.2	Invest in and engage with academic and private sector efforts to transition tools to do software engineering: creating, analyzing, verifying, testing, and maintaining software.	Service Digital PEOs, USD(R&E)	FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Req	OSD should consider identifying automated software generation areas that can apply to specific domains.

Related recommendations from previous studies

OSD06	Direct the Deputy Director for Research and Engineering to coordinate service science and technology transition plans with the appropriate military service.
OSD06	Direct the Deputy Director for Research and Engineering to actively participate in the Joint Capabilities Acquisition and Divestment process to reemphasize technology push initiatives.

Additional Recommendation D8
Collect Data

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Automatically collect all data from DoD national security systems, networks, and sensor systems, and make the data available for machine learning (via federated, secured enclaves, not a centralized repository).**
Stakeholders	USD(A&S), USD(P&R), SAE, CMO, CAPE, DOT&E, DDR&E(AC)
Background	DoD discards or does not have access to significant amounts of data for its systems and has not established an infrastructure for storing data, mining data, or making data available for machine learning. Current analytical efforts are siloed and under-resourced in many cases.
Desired State	DoD has a modern architecture to collect, share, and analyze data that can be mined for patterns that humans cannot perceive. Data is being used to enable better decision-making in all facets of the Department, providing significant advantages that adversaries cannot anticipate. Data collection and analysis is done without compromising security, and DoD, with minimum exceptions, should have complete data rights for all systems (developed with industry).
Role of Congress	N/A

Draft Implementation Plan		Lead stakeholders	Target Date
D8.1	Develop comprehensive data strategy for DoD, taking into account future AI/ML requirements,	CDO with USD(A&S), SAE	Q1 FY20
D8.2	Implement a minimum viable product (MVP) that collects and analyzes the most critical data element for one or more programs.	CDO with USD(A&S), SAE	Q3 FY20
D8.3	Create digital data infrastructure to support collection, storage, and processing.	CDO with USD(A&S), SAE	Q1 FY21
D8.4	Require that all new major systems should specify a data collection and delivery plan.	A&S	Q2 FY21
D8.5	Implement data collection requirements for new sensor and weapon system acquisition.	A&S	FY21

SWAP concept paper recommendations related to this recommendation

10C	All data generated by DoD systems—in development and deployment—should be stored, mined, and made available for machine learning.

Related recommendations from previous studies

DSB18	Rec 7b: [USD(R&E)] should establish a machine learning and autonomy data repository and exchange along the lines of the U.S. Computer Emergency Readiness Team (US-CERT) to collect and share necessary data from and for the deployment of machine learning and autonomy.
DSB18	Rec 7c: [USD(R&E)] should create and promulgate a methodology and best practices for the construction, validation, and deployment of machine learning systems, including architectures and test harnesses.

Appendix B: Legislative Opportunities in Response to 2016 NDAA Section 805

(Template Language for Recommendations A1 and A2)

This appendix provides a template for the type of legislative language that could represent a new category/pathway to procure, develop, deploy and continuously improve software for DoD applications, aligned with Recommendations A1 and A2 in Chapter 5. This template is designed to serve as an example of how the types of changes we envision might be implemented and has not been reviewed or endorsed by the Department. It is written to be consistent with 2016 NDAA Section 805 (Use of alternative acquisition paths to acquire critical national security capabilities).

SEC. [???]. SPECIAL PATHWAYS FOR RAPID ACQUISITION OF SOFTWARE APPLICATIONS AND UPGRADES.

(a) GUIDANCE REQUIRED.—Not later than [90, 180, 270] days after the date of the enactment of this Act, the Secretary of Defense shall establish guidance authorizing the use of special pathways for the rapid acquisition of software applications and upgrades that are intended to be fielded within one year.

(b) SOFTWARE ACQUISITION PATHWAYS.—

(1) The guidance required by subsection (a) shall provide for the use of proven technologies and solutions to continuously engineer and deliver capabilities in software. The objective of an acquisition under this authority shall be to begin the engineering of new capabilities quickly, to demonstrate viability and effectiveness of those capabilities in operation, and continue updating and delivering new capabilities iteratively afterwards. An acquisition under this authority shall not be treated as an acquisition program for the purpose of section 2430 of title 10, United States Code or Department of Defense Directive 5000.01.

(2) Such guidance shall provide for two rapid acquisition pathways:

(A) APPLICATIONS.—The applications software acquisition pathway shall provide for the use of rapid development and implementation of applications and other software and software improvements running on commercial commodity hardware (including modified or ruggedized hardware) operated by the Department; and

(B) EMBEDDED SYSTEMS.—The embedded systems software acquisition pathway shall provide for the rapid development and insertion of upgrades and improvements for software embedded in weapon systems and other military-unique hardware systems.

(c) EXPEDITED PROCESS.--

(1) IN GENERAL.—The guidance required by subsection (a) shall provide for a streamlined and coordinated requirements, budget, and acquisition process that results in the rapid fielding of software applications and software upgrades to embedded systems in a period of

not more than [one year] from the time that the process is initiated. It shall also require the collection of data on the version fielded and continuous engagement with the users of that software, so as to enable engineering and delivery of additional versions in periods of not more than one year each.

(2) EXPEDITED SOFTWARE REQUIREMENTS PROCESS.—

(A) Software acquisitions conducted under the authority of this provision shall not be subject to the Joint Capabilities Integration and Development System Manual and Department of Defense Directive 5000.01, except to the extent specifically provided in the guidance required by subsection (a).

(B) The guidance required by subsection (a) shall provide that—

(1) Requirements for covered acquisitions are developed on an iterative basis through engagement with the user community, and utilization of user feedback in order to regularly define and prioritize the software requirements, as well as to evaluate the software capabilities acquired;

(2) The requirements process begins with the identification of 1) the warfighter or user need, 2) the rationale for how these software capabilities will support increased lethality and/or efficiency, and 3) the identification of a relevant user community;

(3) Initial contract requirements are stated in the form of a summary-level list of problems and shortcomings in existing software systems and desired features or capabilities of new or upgraded software systems;

(4) Contract requirements are continuously refined and prioritized in an evolutionary process through discussions with users that may continue throughout the development and implementation period;

(5) Issues related to life-cycle costs and systems interoperability are considered; and

(6) Issues of logistics support in cases where the software developer may stop supporting the software system are addressed.

(3) RAPID CONTRACTING MECHANISM.— The guidance required by subsection (a) shall authorize the use of a rapid contracting mechanism, pursuant to which—

(A) Aa contract may be awarded within a [90-day] period after proposals are solicited on the basis of statements of qualifications and past performance data submitted by contractors, supplemented by discussions with two or more contractors determined to be the most highly-qualified, without regard to price;

(B) a contract may be entered for a period of not more than one-year and a ceiling price of not more than [$50 million] and shall be treated as a contract for the acquisition of commercial services covered by the preference in section 2377 of title 10, United States Code;

(C) a contract shall identify the contractor team to be engaged for the work, and substitutions shall not be made during the base contract period without the advance written consent of the contracting officer;

(D) the contractor may be paid during the base contract period on a time and materials basis up to the ceiling price of the contract to review existing software in consultation with the user community and utilize user feedback to define and prioritize software requirements, and to design and implement new software and software upgrades, as appropriate;

(E) a contract may provide for a single one-year option to complete the implementation of one or more specified software upgrades or improvements identified during the period of the initial contract, with a price of not more than [$100 million] to be negotiated at the time that the option is awarded; and

(F) an option under the authority of this section may be entered on a time and materials basis and treated as an acquisition of commercial services or entered on a fixed price basis and treated as an acquisition of commercial products, as appropriate.

(4) EXECUTION OF RAPID ACQUISITIONS.--The Secretary shall ensure that —

(A) software acquisitions conducted under the authority of this provision are supported by an entity capable of regular automated testing of the code, which is authorized to buy storage, bandwidth, and computing capability as a service or utility if required for implementation;

(B) processes are in place to provide for collection of testing data automatically from [entity specified in (A)] and using those data to drive acquisition decisions and oversight reporting;

(C) the Director of Operational Test and Evaluation and the director of developmental test and evaluation participate with the acquisition team to design acceptance test cases that can be automated using the entity specified in (A) and regularly used to test the acceptability of the software as it is incrementally being engineered;

(D) acquisition progress is monitored through close and regular interaction between government and contractor personnel, sufficient to allow the government to understand progress and quality of the software with greater fidelity than provided by formal but infrequent milestone reviews;

(E) an independent, non-advocate cost estimate is developed in parallel with engineering of the software, and is based on an investment-focused alternative to current estimation models, which is not based on source lines of code;

(F) the performance of fielded versions of the software capabilities are demonstrated and evaluated in an operational environment; and

(G) software performance metrics addressing issues such as deployment rate and speed of delivery, response rate such as the speed of recovery from outages and cybersecurity vulnerabilities, and assessment and estimation of the size and complexity of software development effort are established that can be automatically generated on a [monthly, weekly, continuous] basis and made available throughout the Department of Defense and the congressional defense committees.

(5) ADMINISTRATION OF ACQUISITION PATHWAY.—The guidance for the acquisitions conducted under the authority of this section may provide for the use of any of the following streamlined procedures in appropriate circumstances:

(A) The service acquisition executive of the military department concerned shall appoint a project manager for such acquisition from among candidates from among civilian employees or members of the Armed Forces who have significant and relevant experience in modern software methods.

(B) The project manager for each large software acquisition as designated by the service acquisition executive shall report with respect to such acquisition directly, and without intervening review or approval, to the service acquisition executive of the military department concerned.

(C) The service acquisition executive of the military department concerned shall evaluate the job performance of such manager on an annual basis. In conducting an evaluation under this paragraph, a service acquisition executive shall consider the extent to which the manager has achieved the objectives of the acquisition for which the manager is responsible, including quality, timeliness, and cost objectives.

(D) The project manager shall be authorized staff positions for a technical staff, including experts in software engineering to enable the manager to manage the acquisition without the technical assistance of another organizational unit of an agency to the maximum extent practicable.

(E) The project manager shall be authorized, in coordination with the users of the equipment and capability to be acquired and the test community, to make trade-offs among life-cycle costs, requirements, and schedules to meet the goals of the acquisition.

(F) The service acquisition executive or the defense acquisition executive in cases of defense wide efforts, shall serve as the decision authority for the acquisition.

(G) The project manager of a defense streamlined acquisition shall be provided a process to expeditiously seek a waiver from Congress from any statutory or regulatory requirement that the project manager determines adds little or no value to the management of the acquisition.

(6) OTHER FLEXIBLE ACQUISITION METHODS.—The flexibilities provided for software acquisition pathways under this section do not preclude the use of acquisition flexibilities otherwise available for the acquisition of software. The Department may use other transactions authority, broad agency announcements, general solicitation competitive procedures authority under section 879 of the National Defense Authorization Act for Fiscal Year 2017, the challenge program authorized by section 2359b of title 10, United States Code, and other authorized procedures for the acquisition of software, as appropriate. Such authorities may be used either in lieu of or in conjunction with the authorities provided in this section.

(d) FUNDING MECHANISMS.—

(1) SOFTWARE FUND.—

(A) IN GENERAL.—The Secretary of Defense shall establish a fund to be known as the [‘‘Department of Defense Rapid Development of Effective Software Fund’’] to provide funds, in addition to other funds that may be available for acquisition under the rapid software development pathways established pursuant to this section. The Fund shall be managed by a senior official of the Department of Defense designated by the [Under Secretary of Defense for Acquisition and Sustainment]. The Fund shall consist of amounts appropriated to the Fund and amounts credited to the Fund pursuant to section [???] of this Act.

(B) TRANSFER AUTHORITY.—Amounts available in the Fund may be transferred to a military department for the purpose of starting an acquisition under the software acquisition pathway established pursuant to this section. These funds will be used to fund the first year of the software acquisition and provide the Department an opportunity to field software capabilities that address newly discovered needs. A decision to continue the acquisition on other funds will be made based upon the progress demonstrated after the first year. Any amount so transferred shall be credited to the account to which it is transferred. The transfer authority provided in this subsection is in addition to any other transfer authority available to the Department of Defense.

(C) CONGRESSIONAL NOTICE.—The senior official designated to manage the Fund shall notify the congressional defense committees of all transfers under paragraph (2). Each notification shall specify the amount transferred, the purpose of the transfer, and the total projected cost and funding based on the effort required each year to sustain the capability to which the funds were transferred. The senior official will also notify the congressional defense committees at the end of the one-year timeframe and report on the fielded capabilities that were achieved. A notice under this paragraph shall be sufficient to fulfill any requirement to provide notification to Congress for a new start.

(2) PILOT PROGRAM. The Secretary may conduct a pilot program under which funding is appropriated in a single two-year appropriation for life-cycle management of software-intensive and infrastructure technology capabilities conducted under the authority of this section. The objective of the appropriation software pilot program would be to provide 1) greater focus on managed services versus disaggregated development efforts, 2) additional accountability and transparency for information centric and enabling technology capabilities, and 3) flexibility to pursue the most effective solution available at the time of acquisition; 4) much greater insight into the nature of software expenditures across the DOD enterprise; 5) an improved ability to measure costs and program performance;

Appendix C: An Alternative to P-Forms and R-Forms: How to Track Software Programs

Background. DoD's Planning, Programming, and Budgeting System (PPBS) establishes the basis for the budget submission to Congress. Multiple statutes, instructions, and directives must be addressed in order to change the way the budget is put together, adjudicated, enacted and managed. Exhibits are prepared by OSD and DoD Components to support requests for appropriations from Congress and help justify the President's budget. These include a number of forms that are aligned with the existing appropriations process:

- P-Form: Procurement
- R-Form: Research, Development, Test, and Evaluation (RDT&E)
- O-Form: Operations and Maintenance
- M-Form: Military Personnel
- C-Form: Military Construction

As described by the Section 809 panel, the competing objectives of the acquisition system make it very difficult for Congress and the Department to effectively budget and manage defense projects, as illustrated in the following diagram (from the Section 809 panel, Volume 3):

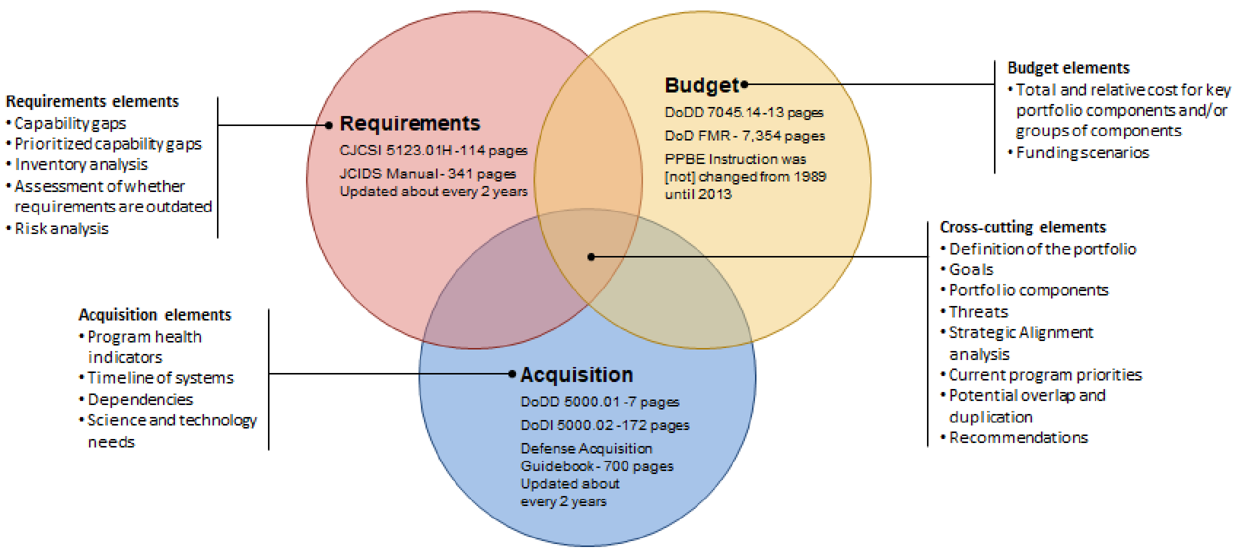


Figure C.1. Multi-layered DoD budget environment.

In this appendix, we describe a different type of mechanism for budget management for software programs, one that is tuned to the nature of software development. We envision this design to reflect and be interweaved with our primary recommendations—in particular A1 (new acquisition pathway for software) and A2 (new appropriations category for software). It could be also be used for software programs that are making use of other pathways (e.g., traditional DoD 5000.02, mid-tier [Sec 804] acquisition, other transaction authority [OTA] based pathways, or operations and maintenance [O&M]).

Key Characteristics. It is useful to list some of the properties that the new process should satisfy before presenting a specific approach for new methods of managing the budget for software programs. The characteristics that we believe are most important are that the process be:

- *Iterative*: In proposing a new approach for approval and oversight of software programs, we envision a process very similar to the way that software itself is developed: Congress and DoD should articulate what their needs are for oversight and approval of software programs, then try out different ways to gain transparency in proposing and monitoring of software programs. Oversight processes can evolve iteratively, ultimately achieving better oversight

- *Efficient:* The current budget process requires the separate creation of standalone forms and documents that are not a part of the regular information that is maintained and tracked as part of the planning and execution of the software program. Instead, we emphasize the use of automated and machine-readable budget information that is interoperable with financial management tools (with translation to human-readable form when useful).

- *Insightful.* The process should provide insights to both DoD and Congress about the planned and current capabilities of the program and opportunities for portfolio optimization. This includes making use of metrics that are appropriate for software (cycle time, rollback time, automated test coverage, etc.), extracting those metrics in an automated fashion wherever possible, and treating software as an enduring capability.

- *Electronic.* Consistent with the nature of software and software development, the budget artifacts used by Congress and DoD should be largely electronic in nature. By "electronic" we do not mean electronic forms that are "printable" (e.g., PDF and Word files), but rather information that is available in electronic form and requires no further processing to be ingested into analysis systems.

Budget Information for Ongoing Software Programs. Since software is never done, the most important budget artifacts will be those for ongoing programs. The information that is required depends on the type of software, so we briefly describe here our advice for what information should be most relevant in evaluating and renewing the budget of an ongoing program.

- *Type A (commercial off-the-shelf apps):* By its nature, ongoing expenses for COTS apps will be based on the commercial price of the software or service. Existing mechanisms for budgeting materials, supplies, and consumables for DoD functions should be used: usage, spend rate, attainment of (volume) price discounts, etc. It is also important to track resources (money and people) needed to perform upgrades made mandatory by vendor version updates and obsolescence.

- *Type B (customized software)* and *Type C (COTS hardware/operating system):* These classes of software represents custom software that is developed, assured, deployed, and maintained by either organic developers or a contractor/vendor for DoD-specific purposes. Type B software will require primarily configuration management and customization, whereas Type C software will involve customized coding. These types of software are perhaps the least well-suited to the traditional spiral development/hardware-focused acquisition and budgeting process, since they often represent an enduring capability in which new features are continuously added.

The diagram below shows the expected cost profile of a software program of this type, in which the annual cost starts small (and may terminate, if not successful), rises as the software is scaled to its full extent, and then falls as it is optimized and continuously improved.

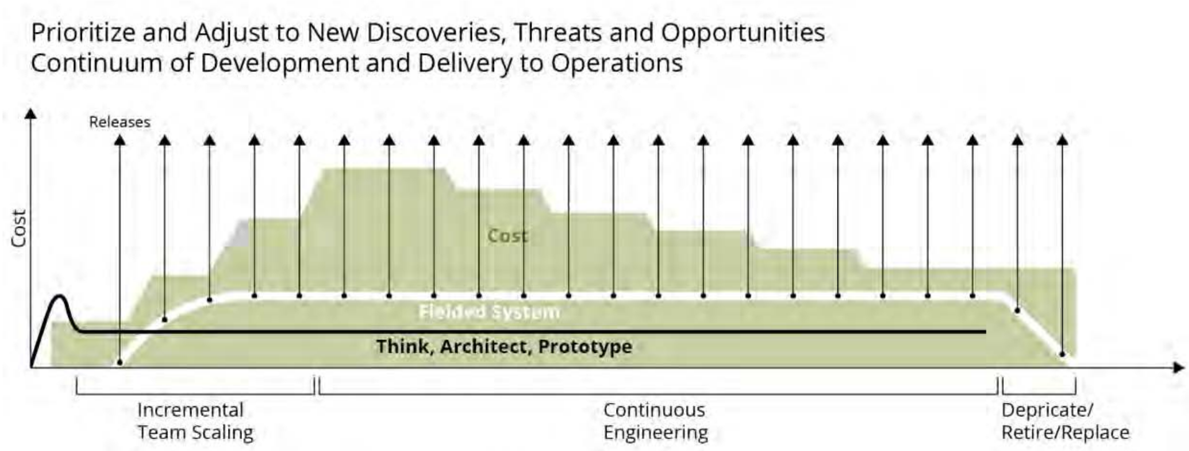


Figure C.2. DevSecOps life cycle cost profile.

The information available as part of the budget process should reflect the following data on the current and desired state of the program:

○ List of features implemented and those planned for future releases
○ Number of active users and level of satisfaction of the user base
○ Time required to field high priority functions (specifications → operations) or fix newly found security holes (discovery → operations)
○ Time from code committed to code in use
○ Time required for full regression test and cybersecurity audit/penetration testing (and the percentage of such testing that is automated)
○ Time required to restore service after outage
○ Percentage test coverage of specs/code, including percentage of tests that are automated
○ Number of bugs caught in testing versus in field use
○ Change failure rate (rollback deployed code)
○ Percentage code available to DoD for inspection/rebuild

The cost data associated with the program should include the following information:

○ The size and annual cost of the development team, along with the percentage of programmers, designers, user interface engineers, system architects and other key development categories.
○ The size and annual cost of the program management team, including both government and contractor program management (if applicable).
○ Software licensing fees
○ Computing costs (including cloud services)
○ Other costs associated with the program

These metrics should be tracked over time, with reports of the past three years of data as well as targets for the coming two years. Annual budget submissions should compare the projected metrics and costs of the program from the past fiscal year with the actual metrics and costs for that period, as well as rolling updating the time horizons to drop the oldest year of tracking data and add the newest year of projected data.

- *Type D (custom hardware and software, including embedded systems):* Embedded systems associated with custom hardware that is still in the development phase is most likely to be reported as part of the hardware development program (using traditional budget items). However, once the software/hardware platform and form factor has been designed then the continued development of the software should be reported in a manner similar to Type C (COTS hardware/operating systems).

Budget Information for New Software Programs. Creating new software programs involves estimation of the cost of the software over at least the initial procurement and deployment phases. Such programs should start small, be iterative, and build on success—or be terminated quickly. Whenever possible, new software programs should have small budgets, require early demonstration of results, and then be turned into ongoing programs (with budget justification as described above). We remark briefly on specific considerations based on the type of software.

- *Type A (commercial off-the-shelf apps)* and *Type B (customized software).* For commercial software of these two types, the most relevant information is the features to be provided by the software, the number of instances of the software expected over time, and the cost of that software (either as purchase cost or licensing costs). For Type B software, additional information should be provided regarding the staffing needs for software configuration, in a manner that is similar to customized software (Type C), though with less intensive development costs.

- *Type C (COTS hardware/operating system).* For custom software running on commodity hardware and operating systems, there are two primary questions that must be addressed: (a) is the software functionality available in commercial products that meets the (primary) needs of the Department and, if not, (b) how large should the initial development effort be in order to create a minimum viable product (MVP) and then begin to scale the initial deployment if successful.

  For comparing customized software to commercially available software, the following information should be provided:

  - A list of features that are desired and an indication of which of those features are available in commercial packages versus those that are DoD-specific.
  - A list of commercial software packages providing similar functionality and the cost of purchasing or licensing that software for initial and full-scale deployment.
  - A justification for why DoD processes cannot be adopted to the development and operations practices of standard commercial approaches and/or why a smaller software development program focused on interfacing DoD specific cases to commercial packages cannot be accomplished.

The goal of providing this information is to ensure that commercial processes/software can be adopted and implemented as standard business practices within DoD. If a DoD-customized software is needed, this information also serves as a good comparison point on the rough costs that are available for related commercial software (when it exists).

- *Type D (custom hardware and software, including embedded systems):* The initial phases of development for custom hardware and software are likely to track hardware development, although in some cases it may be possible to begin software development using emulation and simulation. Care should be taken that embedded software truly requires custom solutions: the trend in commercial software is to establish a layer between hardware and software that allows software to be hardware agnostic (converting Type D into Type C). This approach is quite prevalent in consumer electronics (smart phones and other mobile devices) and transportation systems (automobiles, aircraft).

Software Program Budget Exhibits. Since software programs will be integrated into larger programs and elements of larger programs will have software component, it will be necessary to provide budget exhibits that are compatible with other budget processes used by Congress and DoD. As described above, we believe that the primary information used for tracking ongoing programs should be electronic in nature, and that it should be pulled from existing databases and systems rather than compiled specifically for the budget process.

Following the format used by R-docs, we believe that software programs budget exhibit can be broken down into 5 levels, as shown in the diagram to the right. Each of the exhibits should reflect the information described above (depending on the type of software program) and should exist primarily as electronic databases whose information can be presented in a form consistent with the information that Congress desires.

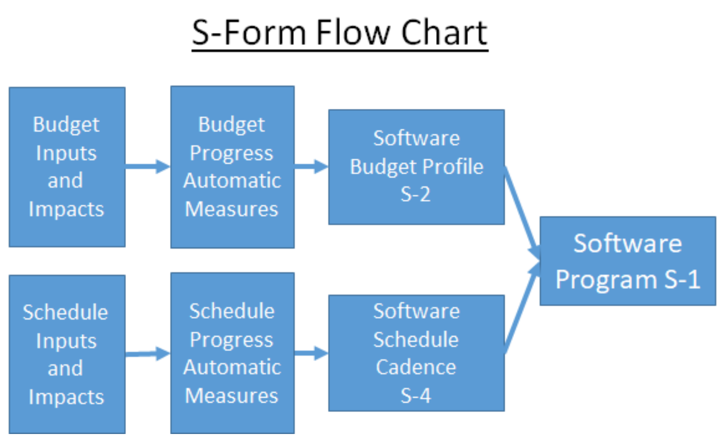


Figure C.3. S-Form inputs.

The individual exhibits are as follows:

- S-1 Exhibits: the basic document for presenting DoD's software program information. The S-1 is prepared at the OSD-level, with one exhibit for each separate software appropriation account/portfolio. Because the S-1 is a summary document, all other software exhibits submitted for a program element must reconcile to the numbers shown on the S-1. The S-1 form should be automatically generated from information maintained by the Component headquarters based on information provided (electronically) be individual software program elements.

- S-2 Exhibits: feeds into the S-1 and are automatically populated to provide summary funding information, program description, metrics, and budget justification for each software program element.

- S-4 Exhibits: generate a display of major program releases. This exhibit is required for each project. If a program element consists of only one project, then the S-4 is prepared for the entire program element.

Multi-Element Program Budgets. For the purpose of establishing a new funding authority that will address the continuous improvement nature of software, a coordinated set of budget exhibits must be put in place. Capability elements that are solely software are relatively rare. The hardware platform that the software must run on will either be provided by a different program under a platform-as-a-service (PaaS), or involve computing hardware that is necessarily coincident to a military vehicle (carried in a ship, aircraft, ground or space). When physical space, power, weight and cooling needs for the computer services have to be managed at the vehicle level, a coordination of the design and implementation of the hardware/software environment must be established and managed over a long period—several epochs of lifespans for computer equipment on which the continuously changing software must run. This is a fundamentally different environment than hardware and must be accommodated in a new software budget exhibit, at the right time of development, while managing within the appropriate form-factor.

Fortunately, the PPBS environment has a mechanism for managing this—the multi-Program Element Project. The coordination of research (R-Form), Procurement (P-Form) and Operations (O-Form) with software program information (electronically generated S-Form) can be accommodated in a single project or set of projects in the PPBS. The most limiting case is the one that requires the greatest level of coordination in software-intensive and embedded products. The figure below shows a parallel timeline for the ideation, creation, scaling and implementation phases of software with the spiral nature of hardware for research, engineering/manufacturing development, procurement, operations, sustainment and disposal.

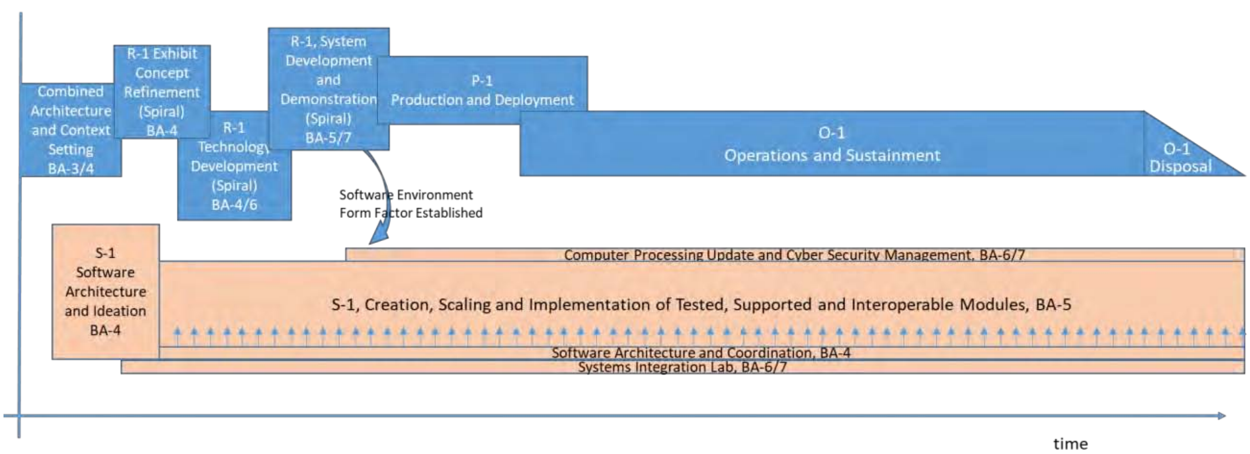


Figure C.4. Budget exhibits by program phase.

Sample Budget Exhibits. To illustrate the type of information that could be presented to Congress as part of the budgeting process, we provide below a sample of some "S-Forms" that might be used to describe a hypothetical software program. For the purposes of illustration, we focus here on a Type C (custom software on commercial hardware/operating system). Other types of software could make use of similar exhibits. We again emphasize that the desired state is that these documents are automatically populated based on electronic databases used within program offices and maintained as part of ongoing development activities.

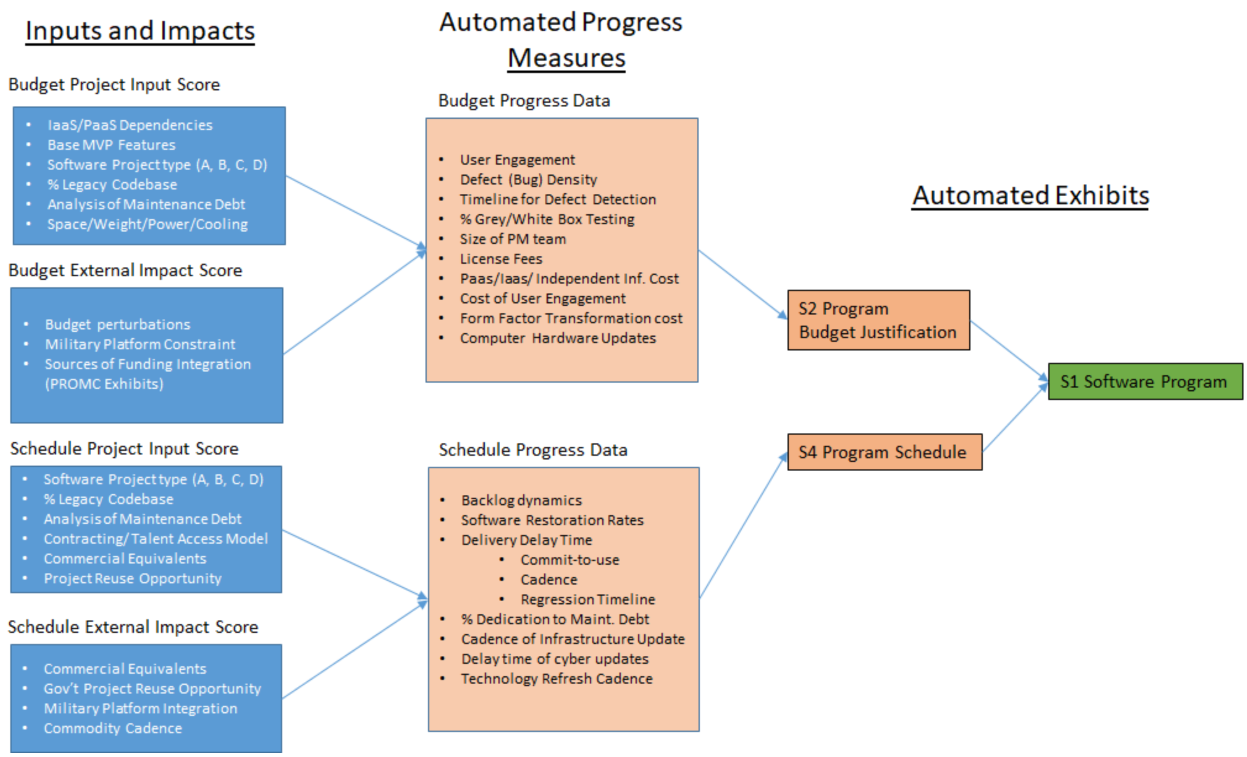


Figure C.5. Software progress metrics and budget exhibit crosswalk

Appendix D: Frequently Asked Questions (FAQ)

This document captures some of the common questions and comments that we have received as we discussed the report with various groups.

1. **Haven't all of these ideas already been recommended in previous studies? Why is this study/report any different?**

   Yes, the vision for how to do software right has existed for decades and most of the best practices that we and others have recommended are common practice in industry today. Chapter 3 (Been There, ~~Done~~ Said That) summarizes previous work and provides our assessment of why things haven't changed. Here are the parts we think are new and different:

   - The recommendations in this report serve primarily as documentation of a sequence of iterative conversations and the real work of the report is the engagements before and after the report is released.
   - Our engagements in the process, and the iterative ways we have worked on this study (just like good software!) have created a willing group of advocates (inside the Department) ready to move forward. If we permit them, we believe change will occur.
   - We focus on speed and cycle time as the key drivers for what needs to change and recommend optimizing statutes, regulations, and processes to allow management and oversight of speed at scale. This won't fix everything, but if you optimize for speed then many other things will improve as well (including oversight).
   - This report is shorter and pithier than previous reports, so we hope people will read it.

2. **Shouldn't Congress just get out of the way and let DoD run things the way they want?**

   This is not the way that the Constitution works. The Legislative branch is an equal branch of government and has a responsibility to see that the Executive branch performs its duties well and properly uses taxpayer resources. This makes implementation of many of the ideas in this report a challenge, but we believe that oversight of software is actually *easier* than oversight of hardware, and Congress can and should take advantage of the insights provided by optimizing speed and cycle time to perform oversight of defense software.

3. **Military software is different than commercial software since lives and national security are at stake, so we can't just do things like they do in industry.**

   Not all (defense) software is the same. Some software requires different consideration in DoD compared with industry, but some software is very much equivalent. Foreign governments perform espionage against U.S. companies and those companies should be protecting themselves in the same way as the U.S. government should (and in many cases, companies are doing better at protecting their code than the government, in our experience).

   And even for those types of software that are very different from what we would find in the commercial world, the broad themes of modern software development are the same: software is never done, speed and cycle time are critical measures, software is by people and for people, and software is different from hardware. In all cases we believe that the acquisition of software must recognize these broad themes to take advantage of the opportunities provided by modern software development practices.

While certainly agreeing that the role of military is different, there are many areas of the private sector in which health, economic well-being, and life safety are critically dependent on software - aircraft, hospitals, traffic management, etc.

4. **Embedded software (in weapons systems) is different than commercial software since it is closely tied to hardware, so we can't just do things like they do in industry.**

   Not all software is the same, and embedded systems have different requirements for testing and verification that may not be present in other types of systems. The broad themes of modern software development also hold for embedded systems: software is never done, speed and cycle time are critical measures, software is by people and for people, and software is different from hardware. The issue of cycle time is the one that usually raises the most concern, but we note that embedded software can also have bugs and vulnerabilities and figuring out how to deploy patches and updates quickly is a valuable feature (think about hardware-coupled features in a mobile device or a Tesla as examples of where this is already being done in industry).

5. **For military systems, training is an essential element and we can't change the software quickly because we can't retrain people to use the new version.**

   Not all software is the same and many types of software have functions that are not directly evident to the user. Indeed, there are some types of software where you might want to update things more slowly to avoid creating confusion for a human operating under stress and having to rely on their training to avoid doing something wrong. For those systems, it will be important to figure out how to couple software updates with training so that warfighters have access to the latest version of the software that provides the functionality and security required to carry out their mission. It is also important to continuously evolve our training regimes to take advantage of what may be increased flexibility and adaptability of "digital natives."

6. **Providing source code to the government is a non-starter for industry. How will they make money if they have to give the government their code?**

   It is critical that DoD have access to source code for purpose-build software: it is required in order to do security scans to identify and fix vulnerabilities, and only with access to the source code and build environment can the government maintain code over time. However, providing source code is different than handing over the rights to do anything they want with that code. Modern intellectual property (IP) language should be used to ensure that the government can use, scan, rebuild, and extend purpose-built code, but contractors should be able to use licensing agreements that protect any IP that they have developed with their own resources.

8. **Won't Congress simply reject modern continuous, incremental software programs believing that "software is never done" is just an open invitation to make programs last forever?**

"Software is never done" specifically highlights that certain capabilities will be enduring, e.g., DoD will always need the capability to ingest data from overhead assets, process that data, and disseminate it and the information it contains. In this situation sensors will change, new analyses will be developed and new products will be required by decision makers. In the traditional DoD software world, a highly defined requirement would be defined, a program would be launched and years later a (likely) out-of-date capability would be delivered, followed immediately by a new, large scale, highly definable requirement, blah, blah, blah. In a world where this need will endure, a continuously funded, incrementally managed software program works better. We must be comfortable that we will spend a certain amount of money each year, we let the program use modern tools for delivering value to real end users incrementally, and we measure success by real-time metrics delivered by the development infrastructure and through direct feedback from the user community. This is the best way to provide Congress with the oversight it deserves.

9. **Have you read a P-Form and an R-Form?**

We have! To us, these do not seem to be able to provide the type of insight into a software (or software-intensive) program that would be required to make a sound judgement about whether a program is in trouble. In addition, they appear to require substantial manual effort to generate and that effort has relatively little added value, they are missing key metrics that are important to understand whether a software program is on track (speed, cycle time, bugs found in test versus in the field, etc.), and the information they contain is updated to infrequently.

In Appendix C of our report we describe a different type of mechanism for budget submissions for software programs, one that is tuned to the nature of software development. We believe that it is possible to implement a mechanism for managing software program that makes use of digitally generated information that is part of the ongoing data that are used in the software development process and that provides improved insight into how well that program is delivering value to the end user.

10. **Government will never hire software developers that are as good as industry.**

While it is certainly true that the vast majority of the highest capability software developers are in the private sector, it is also true that we found extremely capable and dedicated people in the Department—just not nearly enough of them. Actions as consistently detailed in our study can help to address this gap. First, the government should continue to partner with industry and to make use of contractors as a mechanism for obtaining the talent that it needs to develop software that meets its needs. For those cases where it makes sense to use organic (government) software development, the government should make use of existing or new hiring authorities to offer salaries that are as competitive as possible. It is highly unlikely that these will match commercial salaries, but it will show that DoD values software development expertise and that it recognizes that this expertise is in high demand and short supply. On top of this, DoD should anticipate that they will not be able to attract software developers for their entire career. Instead, DoD should have a plan and a set of mechanisms that allow it to hire people for shorter periods of time (e.g., 2-4 years), a period which we believe individuals who are interested in serving their country will be willing to devote. Recommendation C4 (Recruiting (Transient) Digital Talent) provides some ideas for how this might be implemented.

11. What is the purpose of the use of commercial services guidance in the new acquisition pathway that you propose (Recommendation A1 and Appendix B)?

Commercial item procurement was established in 1994 by Congress as a way of encouraging new entrants into the industrial base. While the law was directed at Silicon Valley it also included the vast majority of other types of commercial products at the time — eventually to expand into a greater number of services. Procedures were established (under FAR Part 12) to exempt these types of fixed price contracts from a significant portion of defense-unique acquisition requirements. A preference was also established for the government to buy commercial products and solutions where they existed over defense unique solutions.

The rapid contracting mechanism in Appendix B would essentially treat all purchases through this mechanism as a commercial item covered under FAR Part 12 to limit DoD from applying unique accounting and oversight procedures applicable to traditional defense contracts. Defining these purchases as commercial item purchases triggers two things: (1) a purchasing preference and (2) relief from regulatory burdens, including government-unique contract clauses and data requirements. The purpose of this language is to ensure this favorable treatment for the alternative acquisition pathway without requiring the contractor to make any proof that is a “commercial” vendor.

12. Would the use of the proposed acquisition pathway (Recommendation A1) and/or proposed appropriation category (Recommendation A2) be required for all software programs?

No. We envision this as becoming the *preferred* pathway for software because it is optimized for software. However, traditional acquisition pathways would still be available.

Appendix E: DIB Guides for Software

As a mechanism for obtaining feedback as it carried out its work, the SWAP study wrote a sequence of short "concept papers" that provided a view on what software acquisition and practice should look like. These documents were released on the DIB website (http://innovation.defense.gov/software) and discussed in DIB public meetings. Feedback from the DIB and other stakeholders was used to iterate on the concept papers. The current snapshot of these papers is provided in this appendix.

List of concept papers:

1. Ten Commandments of Software
2. Metrics for Software Development
3. Do's and Don'ts for Software
4. Detecting Agile BS
5. Is Your Development Environment Holding You Back?
6. Is Your Compute Environment Holding You Back?
7. Site Visit Observations and Recommendations
8. How To Justify Your Agile Budget

The copies of the concept papers in this appendix reflect versions in place as of the approval of this report. We anticipate updating and augmenting these reports as the study continues into the implementation phase. The most up-to-date versions of the concept papers can be found at http://innovation.defense.gove/software.

Defense Innovation Board
Ten Commandments of Software

Executive Summary

The Department of Defense (DoD) must be able to develop and deploy software as fast or faster than its adversaries are able to change tactics, building on commercially available tools and technologies. Recognizing that "software" can range from off-the-shelf, non-customized software to highly-specialized, embedded software running on custom hardware, it is critical that the right tools and methods be applied for each type. In this context we offer the following ten "commandments" of software acquisition for the DoD:

1. Make computing, storage, and bandwidth abundant to DoD developers and users.
2. All software procurement programs should start small, be iterative, and build on success – or be terminated quickly.
3. The acquisition process for software must support the full, iterative life cycle of software.
4. Adopt a DevSecOps culture for software systems.
5. Automate testing of software to enable critical updates to be deployed in days to weeks, not months or years.
6. Every purpose-built DoD software system should include source code as a deliverable.
7. Every DoD system that includes software should have a local team of DoD software experts who are capable of modifying or extending the software through source code or API access.
8. Only run operating systems that are receiving (and utilizing) regular security updates for newly discovered security vulnerabilities.
9. Security should be a first-order consideration in design and deployment of software, and data should always be encrypted unless it is part of an active computation.
10. All data generated by DoD systems - in development and deployment - should be stored, mined, and made available for machine learning.

Motivation and Scope

The latest industry best practices for developing, fielding, and sustaining software applications and information technology (IT) systems are substantially outpacing the US government's industrial-era planning, programming, budgeting, and execution system (PPBES) methods. In the commercial software industry, there is no clear delineation between development, procurement, and sustainment; rather it is a continuous cycle that changes rapidly. New functionality is made available and deployed to users in months to weeks (and even days, for truly critical updates). Existing government appropriation structures make it difficult to implement this approach in the DoD.

Currently available commercial technology for rapidly deploying new advances in software, electronics, networking, and other areas means that our adversaries can rapidly develop new tactics that will be used against us. The only defense is to get inside our adversaries' observe, orient, decide, and act (OODA) loop, which requires the ability to rapidly develop and deploy software into operational environments. For software that is used as part of operations, whether it is run in the Pentagon or in the field, this will require new methods for (automated) testing and rapid deployment of updates, patches, and new functionality.

In this document, we provide ten "commandments" (principles) for DoD software that provide an approach to development that builds on the lessons learned in the software industry and enables rapid deployment of software into military operations. These principles are not universal and may not apply in all situations, but they provide a framework for improving the use of software in DoD operations going forward that we believe will provide substantial improvements compared to the current state of practice.

Software Types

Not all software is alike and different types of software require different approaches for procurement and sustainment. It is important to avoid a "one size fits all" approach to weapons systems. Acquisition practices for hardware are almost never right for software: they are too slow, too expensive, and too focused on enterprise-wide uniformity instead of local customization. Similarly, the process for obtaining software to manage travel is different than what is required to manage the software on an F-35. We suggest a taxonomy with four types of software requiring four different approaches:

- A: commercial ("off-the-shelf") software with no DoD-specific customization required;
- B: commercial software with DoD-specific customization needed;
- C: custom software running on commodity hardware (in data centers or in the field);
- D: custom software running on custom hardware (e.g., embedded software).

While many of the principles below apply to all DoD software, some are relevant only for specific types, as we indicate at the end of each description.

To amplify at the extremes of this continuum of software types, we note especially the tendency of large organizations to believe their needs are unique when it comes to software of Type A. Business processes, financial, human resources, accounting and other "enterprise" applications in DoD are generally not more complicated nor significantly larger in scale than those in the private sector. Commercial software, unmodified, can be deployed in nearly all circumstances. At the opposite end of the spectrum we recognize the highly coupled nature of real-time, mission-critical, embedded software with its customized hardware, denoted in Type D. Examples here include primary avionics or engine control, or target tracking in shipboard radar systems, where requirements such as safety, target discrimination and fundamental timing considerations demand that extensive formal analysis, test, validation and verification activities be carried out.

The DIB's Ten Commandments of Software

Commandment #1. Make computing, storage, and bandwidth abundant to DoD developers and users, especially in operational systems. Effective use of software requires that sufficient resources are available for computing, storage, and communications. The DoD should adopt a strategy for rapidly transitioning DoD IT to current industry standards such as cloud computing, distributed databases, ubiquitous access to modernized wireless systems (leveraging commercial standards), abundant computing power and bandwidth that is made available as a platform, integration of mobile technologies, and the development of a DoD platform for downloading applications. Unit cost of IT infrastructure and services should be used as a metric in track improvements. An important metric of abundance must be the ability to actually deliver code, and DoD must be able to count the number of programmers within an organization and make sure that the balance of coders to managers is correct [All types]

Commandment #2. All software procurement programs should start small, be iterative, and build on success – or be terminated quickly. Good software development provides value to the customer quickly, based on working with users starting on day one and defining success based on customer value, not creation of code. Large software programs are doomed to fail because of the rigidity, process, competition protests, and bureaucracy that accompany them (typically starting with the Joint Capabilities Integration Development System (JCIDS) process). The separation of software development into research, development, test and evaluation (RDT&E), procurement, and operations & maintenance (O&M) appropriations (colors of money) – and the use of cost-based triggers within each acquisition category (ACAT) – causes delays and places artificial limitations on the program management office's (PMO's) ability to quickly meet the changing needs, resulting in increased lifetime cost of software and slower deployment. Modern ("agile") approaches used in commercial software development will result in faster deployment *and* significant cost savings. [All types, especially B and C]

Commandment #3. The acquisition process for software must support the full, iterative life cycle of software. Software does not age well. It must be constantly maintained and updated, ideally in an automated fashion. The PPBES process is nominally a two (2) year timeline to request and receive funding, with initial planning occurring five (5) years prior to actual receipt, and funding must be requested by intent of use (RDT&E, procurement, and O&M). But this fiscal separation does not match the process of software development, where all creation of code is "development," whether it falls within the fiscal law definition or not. As an alternative, the DoD should make use of "level of effort" (or capacity) constructs to allow continuous development and testing. Assume that low criticality software that is routinely used will require 10% of the development cost to maintain (per year) and more critical software will likely require more resources. This funding must be planned for at the time of initial development, not as an annual allocation that could be interrupted. Enhanced software capability should never be considered "ahead of need." [All types]

Commandment #4. Adopt a DevSecOps culture for software systems. "DevOps" represents the integration of software development and software operations, along with the tools and culture that support rapid prototyping and deployment, early engagement with the end user, automation and monitoring of software, and psychological safety (e.g., blameless reviews). "DevSecOps" adds the integration of security at all stages of development and deployment, which is essential for DoD applications. These techniques should be adopted by the DoD, with appropriate tuning of approaches used by the community for mission-critical, national security applications. Open source software should be used when possible to speed development and deployment, and leverage the work of others. Waterfall development approaches (e.g., DoD-STD-2167A) should be banned and replaced with true, commercial agile processes. Thinking of software "procurement" and "sustainment" separately is also a problem: software is never "finished" but must be constantly updated to maintain capability, address ongoing security issues and potentially add or increase performance (see Commandment #3). [Type C; Type D when tied to iterative hardware development and deployment methodologies]

Commandment #5. Automate configuration, testing, and deployment of software to enable critical updates to be deployed in days to weeks, not months or years. While operational test and evaluation (OT&E) is useful, it must not be the pacing item for deployment of software, especially upgrades to existing software. Automated configuration management, unit testing, software/hardware-in-the-loop (SIL/HIL) testing, continuous integration, A/B testing, usage and issues tracking, and other modern tools of software development should be used to provide high confidence in software correctness and enable rapid, push deployment of patches, upgrades, and apps. Make use of modern software development tool sets that support these processes (and other types of development stack automation and software instrumentation) to enable code optimization and refactoring. [All types]

Commandment #6. Every purpose-built DoD software system should include source code as a deliverable. DoD should have the rights to and be able to modify (DoD-specific) code when new conditions and features arise. Providing source code will also allow the DoD to perform detailed (and automated) evaluation of software correctness, security, and performance, enabling more rapid deployment of both initial software releases and (most importantly) upgrades (patches and enhancements). [Types C, D]

Commandment #7. Every DoD system that includes software should have a local team of DoD software experts who are able to modify or extend the software through source code or API access. Modern weapons systems are software-driven and utilization of those systems in a rapidly changing environment will require that the system (software) be customizable by the user. In order to do this, all fielded DoD systems that include software must also have a local team (responsible to the operational unit) that has the skills and permission to modify and extend the software through either source code (Commandment #6) or application programming interface (API) access. Local experts should have "reachback" capabilities to larger team and the ability to pull new features into their code (and generate pull requests for features that they add which should go back into the main codebase [repository]). [Types B, C, sometimes D]

Commandment #8. Only run modern operating systems that are receiving (and utilizing) regular security updates for newly discovered security vulnerabilities. Outdated operating systems are a major vulnerability and the DoD should assume that any computer running such a system will eventually be compromised. Standard practice in industry is that security patches should be applied within 48 hours of release, though this is probably too big a window for defense systems. Treat software vulnerabilities like perimeter defense vulnerabilities: if there is a hole in your perimeter and people are getting in, you need to patch the hole quickly and effectively. [Types A, B, C]

Commandment #9. Security should be a first-order consideration in design and deployment of software, and data should always be encrypted unless it is part of an active computation. All data should be encrypted, whether in motion (across a network) or at rest (memory, disk, cloud, etc). A possible exception is real-time data that is part of an embedded control system and is being sent across an internal bus/network that is not accessible from outside that network. [Types A, B, C and D when possible]

Commandment #10. All data generated by DoD systems – in development and operations – should be stored, mined, and made available for machine learning. Create a new architecture to collect, share, and analyze data that can be mined for patterns that humans cannot perceive. Utilize data to enable better decision-making in all facets of the Department, providing significant advantages that adversaries cannot anticipate. Forge culture of data collection/analysis to meet the demands of a software-centric combat environment. Such data collection and analysis can be done without compromising security: in fact, a comprehensive understanding of the data the DoD collects can improve security. [All types]

Supporting Thoughts and Recommendations

In addition to the ten principles given above, we offer the following thoughts and recommendations for how the DoD can best take advantage of software as a force multiplier. While not directly related to software, they are enablers for adopting the principles required for rapid development and deployment of software.

Establish Computer Science as a DoD core competency. Do not rely solely on contractors as the only source of coding capability for DoD systems. Instead, the DoD should recruit, train, and retain internal capability for software development, including by service members, and maintain this as a separate career track (like DoD doctors, lawyers, and musicians). This should complement work done by civilians and contractors. Create new and expand existing programs to attract promising civilian and military science, technology, engineering and math (STEM) talent. Reach into new demographic pools of people who are interested in the work DoD does but otherwise would not be aware of DoD opportunities. Be able to count the number of programmers within an organization and make sure that the balance of developers to managers is correct

Use commercial process and software to adopt and implement standard business practices within the Services. Modern enterprise-scale software has been optimized to allow business to operate efficiently. The DoD should take advantage of these systems by adopting its internal (non-warfighter specific) business processes to match industry standards, which are implemented in cost-efficient, user-friendly software and software as a service [SaaS] tools. Rather than adopt a single approach across the entire DoD, the individual Services should be allowed to implement complementary approaches (with appropriate interoperability).

Move to a model of continuous hardware refresh in which computers are treated as a consumable with a 2-3 year lifetime. The current approach — in which technology refreshes take 8-10 years from planning to implementation — means that most of the time our systems are running on obsolete hardware that limits our ability to implement the algorithms required to provide the level of performance required to stay ahead of our adversaries. Moving to the cloud provides a solution to this issue for enterprise and other software systems that do not operate on local or specialized hardware. However for weapons systems, a continuous hardware refresh mentality would enable software upgrades, crypto updates, and connectivity upgrades to be rapidly deployed across a fleet, rather than maintaining legacy code for different variants that have hardware capabilities ranging from 2 to 12 years old (an almost impossibly large spread of capability in computing, storage, and communications). From a contracting perspective, this change would require DoD to provide a stable annual budget that paid for new hardware and software capability (see Commandment #3), but this would very likely save money over the longer term.

Defense Innovation Board
Metrics for Software Development

Software is increasingly critical to the mission of the Department of Defense (DoD), but DoD software is plagued by poor quality and slow delivery. The current state of practice within DoD is that software complexity is often estimated based on number of source lines of code (SLOC), and rate of progress is measured in terms of programmer productivity. While both of these quantities are easily measured, they are not necessarily predictive of cost, schedule, or performance. They are especially suspect as measurements of program success, defined broadly as delivering needed functionality and value to users. Measuring the health of software development activities within DoD programs using these obsolete metrics is irrelevant at best and, at worst, could be misleading. As an alternative, we believe the following measures are useful for DoD to track performance for software programs and drive improvement in cost, schedule, and performance.

#	Metric	Target value (by software type)[5]				Typical DoD values for SW
		COTS apps	**Custom-ized SW**	**COTS HW/OS**	**Real-time HW/SW**	
1	Time from program launch to deployment of simplest useful functionality	<1 mo	<3 mo	<6 mo	<1 yr	3-5 yrs
2	Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops)	N/A <1 wk	<1 mo <1 wk	<3 mo <1 wk	<3 mo <1 wk	1-5 yrs 1-18 m
3	Time from code committed to code in use	<1 wk	<1 hr	<1 da	<1 mo	1-18 m
4	Time req'd for full regression test (automat'd) and cybersecurity audit/penetration testing	N/A <1 mo	<1 da <1 mo	<1 da <1 mo	<1 wk <3 mo	2 yrs 2 yrs
5	Time required to restore service after outage	<1 hr	<6 hr	<1 day	N/A	?
6	Automated test coverage of specs/code	N/A	>90%	>90%	100%	?
7	Number of bugs caught in testing vs field use	N/A	>75%	>75%	>90%	?
8	Change failure rate (rollback deployed code)	<1%	<5%	<10%	<1%	?
9	% code avail to DoD for inspection/rebuild	N/A	100%	100%	100%	?
10	Number/percentage of functions implemented	80%	90%	70%	95%	100%
11	Usage and user satisfaction	TBD	TBD	TBD	TBD	?

[5] Target values are notional; different types of software (SW) as defined in DIB Ten Commandments.

Acronyms defined: Commercial off-the-shelf (COTS), apps is short for applications, specs is short for specifications, hardware/operating system (HW/OS), hardware/software (HW/SW)

12	Complexity metrics	#/type of specs # programmers structure of code #/skill level of teams #/type of platforms #/type deployments				Partial/ manual tracking
13	Development plan/environment metrics					
14	"Nunn-McCurdy" threshold (for any metric)	1.1X	1.25X	1.5X	1.5X each effort	1.25X Total $

Supporting Information

The information below provides additional details and rationale for the proposed metrics. The different types of software considered in the document are described here in greater depth, followed by comments on the proposed metrics, grouped into four categories: (a) deployment rate metrics, (b) response rate metrics, (c) code quality metrics, and (d) program management, assessment and estimation metrics.

Software Types (from DIB Ten Commandments)

Not all software is alike, and different types of software require different approaches for development, deployment, and life-cycle management. It is important to avoid a "one size fits all" approach to weapons systems. Acquisition practices for hardware are almost never right for software: they are too slow, too expensive, and too focused on enterprise-wide uniformity instead of local customization. Similarly, the process for obtaining software to manage travel is different than what is required to manage the software on an F-35. We suggest a taxonomy with four types of software requiring four different approaches:

- A: commercial ("off-the-shelf") software with no DoD-specific customization required;
- B: commercial software with DoD-specific customization needed;
- C: custom software running on commodity hardware (in data centers or in the field);
- D: custom software running on custom hardware (e.g., embedded software).

Type A (COTS apps): The first class of software consists of applications that are available from commercial suppliers. Business processes, financial management, human resources, accounting and other "enterprise" applications in DoD are generally not more complicated nor significantly larger in scale than those in the private sector. Unmodified commercial software should be deployed in nearly all circumstances. Where DoD processes are not amenable to this approach, those processes should be modified, not the software.

Type B (Customized SW): The second class of software constitutes those applications that consist of commercially available software that is customized for DoD-specific usage. Customizations can include the use of configuration files, parameter values, or scripted functions that are tailored for DoD missions. These applications will generally require configuration by DoD personnel, contractors, or vendors.

Type C (COTS HW/OS): The third class of software applications is those that are highly specialized for DoD operations but can run on commercial hardware and standard operating systems (e.g., Linux or Windows). These applications will generally be able to take advantage of commercial processes for software development and deployment, including the use of open source code and tools. This class of software includes applications that are written by DoD personnel as well as those that are developed by contractors.

Type D (Custom SW/HW): This class of software focuses on applications involving real-time, mission-critical, embedded software whose design is highly coupled to its customized hardware. Examples include primary avionics or engine control, or target tracking in shipboard radar systems. Requirements such as safety, target discrimination, and fundamental timing considerations demand that extensive formal analysis, test, validation, and verification activities be carried out in virtual and "iron bird" environments before deployment to active systems. These considerations also warrant care in the way application programming interfaces (APIs) are potentially presented to third parties.

Types of Software Metrics

Deployment Rate Metrics

Overview: Consistent with previous Defense Innovation Board (DIB) commentary, and software industry best practices, an organizational mentality that prioritizes speed is the ultimate determinant of success in delivering value to end users. An approach to software development that privileges speed over other factors drives efficient decision-making processes; forces the use of increased automation of development and deployment processes; encourages the use of code that is machine-generated as well as code that is correct-by-construction; relies heavily on automated unit and system level testing; and enables the iterative, deliver-value-now mentality of a modern software environment. Thus we list these metrics first.

#	Metric	Target value (by software type)				Typical DoD values for SW
		COTS apps	Customized SW	COTS HW/OS	Real-time HW/SW	
1	Time from program launch to deployment of simplest useful functionality	<1 mo	<3 mo	<6 mo	<1 yr	3-5 yrs
2	Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops)	N/A <1 wk	<1 mo <1 wk	<3 mo <1 wk	<3 mo <1 wk	1-5 yrs 1-18 m
3	Time from code committed to code in use	<1 wk	<1 hr	<1 da	<1 mo	1-18 m

Background: These measures capture the rate at which new functions and changes to a software application can be put into operation (in the field):

1. The time from program launch to deployment of the "simplest useful functionality" is an important metric because it determines the first point at which the code can start doing useful work and also at which feedback can be gathered that supports refinement of the features. There is a tendency in DoD to deliver code only once it has met all of the specifications, but this can lead to significant delays in providing useful code to the user. We instead advocate

getting code in the hands of the user quickly, even if it only solves a subset of the full functionality. Something is better than nothing, and user feedback often reveals omissions in the specifications and can refine the initial requirements. As code becomes more customized, this interval of time might extend due to the need to run more complex tests to ensure that all configurations operate as expected, and that complex timing and other safety/mission-critical specifications are satisfied. It is important to note that this metric is not just about coding time. It also measures the time required to process and adjudicate the changes (including release approval), often the most time-consuming part of providing new or upgraded functionality.

2. Once the code is deployed, it is possible to measure the amount of time that it takes to make incremental changes that either implement new functions or fix issues that have been identified. The importance of the functionality or severity of the error will determine how quickly these changes should be made, but it should be possible to deploy high priority code updates much more quickly and in much smaller increments than typical DoD "block" upgrades. A similar measure to the time it takes to deploy code to the field is deployment frequency. Deployment frequency can be on-demand (multiple per day), once per hour, once per day, once per week, etc. Faster deployment frequency often correlates with smaller batch sizes.

3. The time from which code is committed to a repository until it is available for use in the field is referred to as "lead time," and good performance on this metric is a necessary condition for rapid evolution of delivered software functionality. Shorter product delivery times demand faster feedback, which enables tighter coupling to user needs. For commercially available applications, the lead time will be based on vendor deployment processes and may be slower than what is needed for customized code, be it for commercial hardware/operating systems or custom hardware. However, we believe that in the selection of commercial software, emphasis should be given to the vendor's iteration cycles and lead time performance. Embedded code will often require much more extensive testing before it is deployed, and therefore its lead time may be longer.

Response Rate Metrics

Overview: Our philosophy is that delivering a partial solution to the user quickly is almost always better than delivering a complete or perfect solution at the end of a contract, on the first attempt. Consistent with that, mistakes will occur. No software is bug-free, and so it is unrealistic and unnecessary to insist on that, except where certain safety matters are concerned.[6] Code that does most things right will still be useful while a patch is being identified and fielded. How gracefully software fails, how many errors are caught and resolved in testing, and how rapidly developers patch bugs are excellent measures of software development prowess.

---

[6] The Department and its suppliers (due to the requirements of the contracts to which they are bound) often resort to blanket pronouncements about safety and security, which often lead to applying the most extreme measures even when not needed; this risk-averse approach to treating everything as a grave risk to cyber security or safety has been labeled by the DIB as a "self-denial of service attack." While cybersecurity is clearly critical for software systems, the Department needs to rely on product managers who use judgment to make subtle, nuanced, and risk-based judgments about trade-offs during the software development process.

#	Metric	Target value (by software type)				Typical DoD values for SW
		COTS apps	Custom ized SW	COTS HW/OS	Real-time HW/SW	
4	Time required for full regression test (automated) and cybersecurity audit/penetration testing[7]	N/A <1 mo	<1 da <1 mo	<1 da <1 mo	<1 wk <3 mo	2 yrs 2 yrs
5	Time required to restore service after outage	<1 hr	<6 hr	<1 day	N/A[8]	?

Background: These two metrics are intended for "generic" software programs with moderate complexity and criticality. Their purpose is to:

4. Measure the ability to conduct more complete functional tests of the full software suite (e.g., regression tests) in a timely fashion, to identify problems in deployed software that can be quickly corrected, and to restore service after an incident such as an unplanned outage or service impairment, occurs (also called "mean time to repair," (MTR)).

5. Track the time required to resolve an interruption to service, including a bad deployment.

Code Quality Metrics

Overview: These metrics are intended to be used as a measure of the quality of the code and to focus on identifying errors in the code, either at the time of development (e.g., via unit tests) or in the field.

#	Metric	Target value (by software type)				Typical DoD values for SW
		COTS apps	Custom ized SW	COTS HW/OS	Real-time HW/SW	
6	Automated test coverage of specs/code	N/A	>90%	>90%	100%	?
7	Number of bugs caught in testing vs field use	N/A	>75%	>75%	>90%	?
8	Change failure rate (rollback deployed code)	<1%	<5%	<10%	<1%	?
9	% code avail to DoD for inspection/rebuild	N/A	100%	100%	100%	0%

[7] The two different response rate metrics for different types of software reflect the level of complexity of the software, the likely resources available to identify and fix problems, and the level of integration of the hardware and software.

[8] We note that for embedded systems, which must be running at all times and which are updated much less frequently, the notion of "restoring" service is not directly applicable.

Background:

6. Automated developmental tests provide a means of ensuring that updates to the code do not break previous functionality and that new functionality works as expected. Ideally, for each function that is implemented, a set of automated tests will be constructed that cover both the specification for what the performance should achieve as well as the code that is used to implement that function.

7. The percentage of specifications tested by the automated test suite provides rapid confidence that a software change has not caused some specification to fail, as well as confidence that the software does what it is supposed to do. Test coverage of the code is a common metric for software test quality and one that most software development environments can compute automatically (e.g., in a continuous integration (CI) workflow, each commit and/or pull request to a repository would run all the automated developmental tests and compute the percentage covered). For customized software and applications that run on commercial hardware and operating systems, 90% unit test coverage is a good target. Embedded code should strive for 100% coverage (i.e., no "dark" code) since it is often safety- or mission-critical.[9] The focus of these metrics is on developmental tests, as operational testing is important, but expensive, so it is far less expensive to find and fix defects through developmental testing.

8. Developmental tests do not cover every conceivable situation in which an application might be used, so errors will be discovered in the field. The percent of bugs caught in testing (via unit tests or regression tests) versus those caught in the field provide a measure of the both the quality of the code and the thoroughness of the testing environment. Bugs discovered late in the development cycle or after deployment can "cost" an order of magnitude more than early bugs (in terms of time to fix and impact to a program), and a software system that is mature finds many more bugs during testing and few in the field. Late bugs are particularly expensive when fixing those bugs can require hardware changes, and so code running on custom hardware should be tested more strenuously. Bugs should be prioritized by severity and the trends over time for serious bugs should be monitored and used to drive changes in the test environment and software development process.

9. When bugs do occur, it may be necessary to roll back the deployed code and return to an earlier version. Change fail percentage is the percentage of changes to production that fail, including software releases and infrastructure changes. This should include changes that result in degraded service or subsequently require remediation, such as those that lead to service impairment or outage, or require a hotfix, rollback, fix-forward, or patch. For COTS applications, this should occur rarely because the amount of testing done by the vendor, including test deployments to beta users, will typically be very high. There may be a higher change failure rate as the application becomes more customized—because it can be difficult to test for issues where there is a variety of hardware configurations operating in the field, for example—but for embedded code, the change failure rate should be small, due to the more safety-critical nature of that code leading to more emphasis on up-front testing.

Functionality metrics

Overview: These metrics are intended to capture how useful the software program is in terms of delivering value to the field. We envision that a software program will have a number of desired

[9] Safety- or mission-critical software often strives for more rigorous test coverage metrics, such as high branch coverage or in some cases high modified condition/decision coverage (MC/DC).

features that define its functionality. Software should be instrumented so that the use of those features is measured and, when appropriate, users of the software should be monitored or surveyed to determine their use of/satisfaction with the software.

#	Metric	Target value (by software type)				Typical DoD values for SW
		COTS apps	Custom -ized SW	COTS HW/OS	Real-time HW/SW	
10	Number/percentage of functions implemented	80%	90%	70%	95%	100%
11	Usage and user satisfaction	TBD	TBD	TBD	TBD	?

Background:

10. An ongoing software program will have some number of functions that it performs and a list of additional functions that are to be added over time. These new functions could be feature requires from users or desired features generated by the program office that are on the list for consideration to be implemented next. Keeping track of these features and the rate at which they are implemented provides a measure of the delivery of functionality to the user. This specific way in which functionality is measured will be dependent on the type of software being developed.

11. For software that is used by a person, the ultimate metric is whether the software is helping that person get useful work done. Keeping track of the usage of the software (and different parts of the software) can be done by instrumenting the code and keeping track of the data it generates. To determine whether or not the software is providing good value to the person who is using it, surveying the user may be the most direct mechanism (similar to rating software that you use on a computer or smart phone).

Program Management, Assessment, and Estimation Metrics

Overview: The final set of metrics are intended for management of software programs, including cost assessment and performance estimation. These metrics describe a list of “features” (performance metrics, contract terms, project plans, activity descriptions) that should be required as part of future software projects to provide better tools for monitoring and predicting time, cost, and quality. In its public deliberations regarding software acquisition and practices, the DIB has described how metrics of this type might be used to estimate the cost, schedule, and performance of software programs.

#	Metric	Target value (by software type)				Typical DoD values for SW
		COTS apps	Custom ized SW	COTS HW/OS	Real-time HW/SW	
12	Complexity metrics	#/type of specs # programmers structure of code #/skill level of teams #/type of platforms #/type deployments				Partial/ manual tracking
13	Development plan/environment metrics					

14	"Nunn-McCurdy" threshold (for any metric)	1.1X	1.25X	1.5X	1.5X each effort	1.25X Total $

Background:

12. Structure of specifications, code, and development and execution platforms.

To measure the complexity of a software program, and therefore assess the cost, schedule and performance of that program, a number of features must be measured that capture the underlying "structure" of the application. The use of the term "structure" is intentionally flexible, but generally includes properties such as size, type, and layering. Examples of features that can be captured that related to underlying complexity include:

- *Structure of specifications:* Modern specification environments (e.g., application life-cycle management [ALM] tools) provide structured ways of representing specifications, from program level requirements to derived specifications for sub-systems, or individual teams. The structure represented in these tools can be used as a measure of the difficulty of the application that is being designed.
- *Level and type of user engagement during application development:* How much time do developers spend with users, especially early in the program? How many developers are "on site" (in the same organization and/or geographic location as the end user)?
- *Structure of the code base (software architecture):* Modern software development environments allow structured partitioning of the code into functions, libraries/frameworks, and services. The structure of this partitioning (number of modules, number of layers, and amount of coupling between modules and layers) can provide a measure of the complexity of the underlying code.
- *The amount of reuse of existing code, including open source code:* In many situations there are well-maintained code bases that can be used to quickly create and scale applications without rewriting software from scratch. These libraries and code frameworks are particularly useful when using commodity hardware and operating systems, since the packages will often be maintained and expanded by others, leveraging external effort.
- *Structure of the development platform/environment:* This includes the software development environments that are being used, the types of programming methodologies (e.g., XP, agile, waterfall, spiral) that are employed, and the level of maturity of the programming organization (ISO, CMMI, SPICE).
- *Structure of the execution platform/environment:* The execution environment can have an impact on the ability to emulate the execution environment within the development environment, as well as the portability of applications between different execution environments. Possible platforms include various cloud computing environments as well as platform-as-a-service (PaaS) environments that support multiple cloud computing vendors.

13. Structure and type of development and operational environment.

To predict and monitor the level of effort required to implement and run a software application, measurement of the development and operation environments is critical. These measurements include the structure of those environments (e.g., waterfall versus spiral versus agile, use of continuous integration tools, integrated tools for issue tracking/resolution, code review mechanisms), the tempo of development and delivery, and use of the functionality provided by the application. Example of features that can be captured that relate to the structure and type of development and operation use:

- Number and skill level of programmers on the development team
- Number of development platforms used across the project
- Number of subcontractors or outside vendors used for application components
- Number and type of user operating environments (execution platforms) supported
- Rate at which major functions (included in specifications) are delivered and updated
- Rate at which the operational environment must be updated (e.g., hardware refresh rate)
- Rate at which the mission environment changes (driving changes to the code)
- Number (seats or sites) and skill level of the users of the software

14. Tracking software program progress

To properly manage the continuous development and deployment of software, DoD should be able to track the metrics above with minimal additional effort from the programmers because this information should be gathered and transmitted automatically through the development, deployment, and execution environments, using automated tools. Some examples of the types of metrics that are readily available include commit activity data (number and rate of commits), team size, number of commenters (on pull requests), number of pull request mergers, average and standard deviation of the months in which there was development activity, average and standard deviation of the number of commits per month.

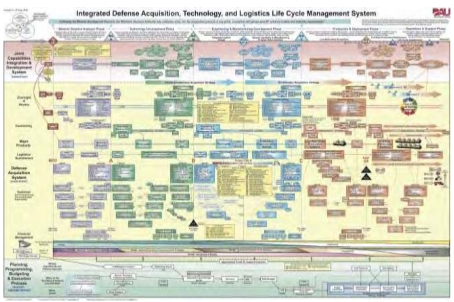
Thresholds should be established to determine when management attention is required, but also when a program is so far off its initial plan that it should be re-evaluated. Today's "Nunn-McCurdys" or "Critical Changes" refer to breaches in cost or schedule thresholds. The current 25% unit cost growth and 50% total program cost growth thresholds often will not make sense for continuously developed software programs.

An alternative to cost-based thresholds is to establish thresholds based on the metrics listed above, with different thresholds for different types of software. A notion of a "Nunn-McCurdy type breach" for software programs based on some of the above performance metrics recorded at lower levels of effort or on specific applications could serve as better means of identifying major issues earlier in a program. Commercially available software, with or without customization, should be the easiest type for which to establish accurate metrics, since it already exists and should be straightforward to purchase and deploy. Metrics for customized software running on either commercial or DoD-specific hardware is likely to be more difficult to predict, so a higher threshold can be used in those circumstances.

Defense Innovation Board Do's and Don'ts for Software

This document provides a summary of the Defense Innovation Board's (DIB's) observations on software practices in the DoD and a set of recommendations for a more modern set of acquisition and development principles. These recommendations build on the DIB's "Ten Commandments of Software." In addition, we indicate some of the specific statutory, regulatory, and policy obstacles to implementing modern software practices that need to be changed.

Executive Summary

Observed practice (Don'ts)	**Desired state (Do's)**	**Obstacles**
 Defense Acquisition University, June 2010	 https://commons.wikimedia.org/wiki/File:Devops-toolchain.svg (modifications licensed CC-BY-SA)	10 U.S.C. §2334 10 U.S.C. §2399 10 U.S.C. §2430 10 U.S.C §2433a 10 U.S.C. §2460 10 U.S.C. §2464 DODI 5000.02, par 5.c.(2) and 5.c.(3)(c)-(d)
Spend 2 years on excessively detailed requirements development	Require developers to meet with end users, then start small and iterate to quickly deliver useful code	DODI 5000.02, par 5.c.(2) CJCSI 3170.01I App A.1.b
Define success as 100% compliance with requirements	Accept 70% solutions[10] in a short time (months) and add functionality in rapid iterations (weeks)	10 U.S.C. §2399 OMB Cir A-11 pp 42-43
Require OT&E to certify compliance after development and before approval to deploy	Create automated test environments to enable continuous (and secure) integration and deployment to shift testing left	10 U.S.C. §139b/d 10 U.S.C. §2399 Cultural
Apply hardware life-cycle management processes to software	Take advantage of the fact that software is essentially free to duplicate, distribute, and modify	10 U.S.C. §2334 10 U.S.C. §2399 10 U.S.C. §2430 48 CFR 207.106 DODI 5000.02
Require customized software solutions to match DoD practices	For common functions, purchase existing software and change DoD processes to use existing apps	Culture

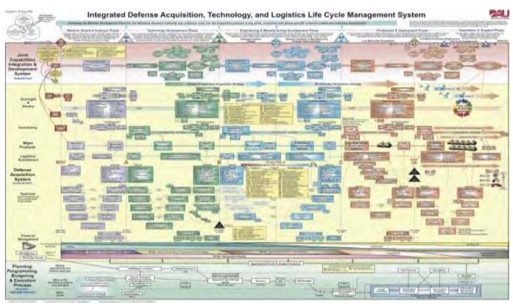
[10] 70% is notional. The point is to deliver the simplest, most useful functionality to the warfighter quickly.

Use legacy languages and operating systems that are hard to support and insecure	Use modern software languages and operating systems (with all patches up-to-date)	10 U.S.C. §2334 DoDI 5000.02, Enclosure 11 Culture
Evaluate cyber security after the systems have been completed, separately from OT&E	Use validated software development platforms that permit continuous integration & evaluation (DevSecOps)	DOT&E Memos Culture
Consider development and sustainment of software as entirely separate phases of acquisition	Treat software development as a continuous activity, adding functionality across its life cycle	10 U.S.C. §2399 10 U.S.C. §2430 10 U.S.C. §2460 10 U.S.C. §2464 DODI 5000.02, par 5.c.(2) and 5.c.(3)(c)-(d)
Depend almost entirely on outside vendors for all product development and sustainment	Require source code as a deliverable on all purpose-built DoD software contracts. Continuous development and integration, rather than sustainment, should be a part of all contracts. DoD personnel should be trained to extend the software through source code or API access[11]	Culture (no apparent statutory obstacle) FAR/DFARS technical data rights
Turn documents like this into a process and enforce compliance	Hire competent people with appropriate expertise in software to implement the desired state and give them the freedom to do so ("competence trumps process")	Culture

[11] As noted in the DIB's 10 Commandments of Software

Supporting Information

The information below, broken out by entry in the executive summary table (see table E.8 above), provides additional information and a rationale for each desired state.

Don't	Do
 Defense Acquisition University, June 2010	 https://commons.wikimedia.org/wiki/File:Devops-toolchain.svg

The DoD 5000 process, depicted on the left in figure E.1, provides a detailed DoD process for setting requirements for complex systems and ensuring that delivered systems are compliant with those requirements. The DoD's "one size fits all" approach to acquisition has attempted to apply this model to software systems, where it is wholly inappropriate. Software is different than hardware. Modern software methods make use of a much more iterative process, often referred to as "DevOps," in which development and deployment (operations) are a continuous process, as depicted on the right. A key aspect of DevOps is continuous delivery of improved functionality through interaction with the end user.

Why this is hard to do, but also worth doing:[12]

- DoD 5000 is designed to give OSD, the Services, and Congress some level of visibility and oversight into the development, acquisition, and sustainment of large weapons systems. While this directive may be useful for weapons systems with multi-billion dollar unit costs, it does not make sense for most software systems.
- While having one consistent procurement process is desirable in many cases, the cost of using that same process on software is that software is delivered late to need, costs substantially more than the proposed estimates, and cannot easily be continuously updated and optimized.
- Moving to a software development approach will enable the DoD to move from a *specify, develop, acquire, sustain* mentality to a more modern (and more useful) *create, scale, optimize* (DevOps/DevSecOps) mentality. Enabling rapid iteration will create a system in which the United States can update software at least as fast as our adversaries can change tactics, allowing us to get inside their OODA loop.

[12] These comments and the similar ones that follow for other area were obtained by soliciting feedback on this document from people familiar with government acquisition processes and modern software development environments.

Acronyms defined: Office of the Secretary of Defense (OSD), OODA is short for the decision cycle of Observe, Orient, Decide, and Act.

Don't	Do
Spend 2 years on excessively detailed requirements development	Require developers to meet with end users, then start small and iterate to quickly deliver useful code
Define success as 100% compliance to requirements	Accept 70% solutions in a short time (months) and add functionality in rapid iterations (weeks)

Developing major weapons systems is costly and time consuming, so it is important that the delivered system meets the needs of the user. The DoD attempts to meet these needs with a lengthy process in which a series of requirements are established, and a successful program is one that meets those requirements (ideally close to the program's cost and schedule estimates). Software, however, is different. When done right, it is easy to quickly deploy new software that improves functionality and, when necessary, rapidly rollback deployed code. It is more useful to get something simple working quickly (time-constrained execution) and then exploit the ability to iterate rapidly in order to get the remaining desired functionality (which will often change in any case, either in response to user needs or adversarial tactics).

Why this is hard to do, but also why it is worth doing:

- Global deployment of software on systems which are not always network-connected (e.g., an aircraft carrier or submarine underway) introduces very real problems around version management, training, and wisely managing changes to mission-critical systems.
- In the world of non-military, consumer Internet applications, it is easy to glibly talk about continuous deployment and delivery. In these environments, it is easy to execute and the consequences for messing up (such as making something incredibly confusing or hard to find) are minor. The same is not always true for DoD systems—and DoD software projects rarely offer scalable and applicable solutions to address the need for continuous development.
- Creating an approach (and the supporting platforms) that enables the DoD to achieve continuous deployment is a non-trivial task and will have different challenges than the process for a consumer Internet application. The DoD must lay out strategies for mitigating these challenges. Fortunately, there are tools that can be build upon: many solutions have already been developed in consumer industries that require failsafe applications with security complexities.
- Continuous deployment depends on the entire ecosystem, not just the front-end software development.
- Make sure to focus on product design and *product* management, which prioritizes delivery of capability to meet the changing needs of users, rather than program/project management, which focus on execution against a pre-approved plan. This shift is key to user engagement, research, and design.

Don’t	Do
Require OT&E to certify compliance after development and before approval to deploy	Create automated test environments to enable continuous (and secure) integration and deployment to shift testing left
Evaluate cyber security after the system has been completed, separately from OT&E	Use validated software development platforms that permit continuous integration and evaluation

Why this is hard to do, but also worth doing:

- The DoD typically performs a cyber evaluation on software only after delivery of the initial product. Modern software approaches have not always explicitly addressed cyber security (though this is changing with “DevSecOps”). This omission has given DoD decision-makers an easy “out” for dismissing recommendations (or setting up roadblocks) for DevOps strategies like continuous deployment. Cyber security concerns must be addressed head on, and in a manner that demonstrates better security in realistic circumstances. Until then, change is unlikely.
- More dynamic approaches to address the cyber security concerns must be developed and implemented through some amount of logic and a fair bit of data. Case studies of red teaming also help: *Hack the Pentagon* should be able to provide some true examples that generate concern. It may be necessary to obtain access to some additional good data that goes beyond what corporations are willing to share publicly.
- To succeed, it will be important not to assume that it will be clear how these recommendations solve for all cyber security concerns. Recommendations should make explicit statements about what can be accomplished, taking away the reasons to say “no.”

Don’t	Do
Apply hardware life-cycle management processes to software	Take advantage of the fact that software is essentially free to duplicate, distribute, and modify
Consider development and sustainment of software as entirely separate phases of acquisition	Treat software development as a continuous activity, adding functionality across its life cycle

Why this is hard to do, but also worth doing:

- Program of record funding is specifically broken out into development and sustainment. These distinct categories of appropriations lead program managers and acquisition professionals to the conclusion that new functionality can only be added within development contracts and that money allocated for sustainment cannot be used to add new features. Vendor evaluation for development and sustainment contracts are different; vendors on sustainment contracts often do not have the same development competencies and frequently are not the people who built the original system. To create an environment that will support a DevOps/DevSecOps approach, DoD Commands and Services should jointly own the development and maintenance of software with contractors who provide more specialized capabilities. Contracts for software should focus on developing and deploying software (to operations) over the long term, rather than the typical, sequential approach - "acquiring" software followed by "sustaining" that software.

Don't	**Do**
Require customized software solutions to match DoD practices	For common functions, purchase existing software and change DoD processes to use existing apps

Business processes, financial, human resources, accounting and other "enterprise" applications in the DoD are generally not more complicated nor significantly larger in scale than those in the private sector. Commercial software, unmodified, should be deployed in nearly all circumstances. Where DoD processes are not amenable to this approach, those processes should be modified, not the software. Doing so allows the DoD to take advantage of the much larger commercial base for common functions (e.g., Concur has 25M active users for its travel software).

Don't	**Do**
Use legacy languages and operating systems that are hard to support and insecure	Use modern software languages and operating systems (with all patches up-to-date)

Modern programming languages and software development environments have been optimized to help eliminate bugs and security vulnerabilities that were often left to programmers to avoid (an almost impossible endeavor). Additionally, outdated operating systems are a major security vulnerability and the DoD should assume that any computer running such a system will eventually be compromised.[13] Standard practice in industry is to apply security patches within 48 hours of release, though even this is probably too big a window for defense systems. Treat software vulnerabilities like perimeter defense vulnerabilities: if there is a hole in your perimeter and people are getting in, you need to patch the hole quickly and effectively.

Why this is hard to do, but also worth doing:

- DoD looks at the cost of upgrading hardware as a major cost that is tied to "modernization." But hardware should be thought of as a consumable like any other, such as fuel and parts, that must be continually replaced for a weapon system to maintain operational capability. This change would require DoD to provide a stable annual budget that paid for new hardware and software capability.
- The advantage of using modern hardware and operating systems on DoD systems are manifold: better security, better functionality, reduced (unit) costs, and lower overall maintenance costs.

[13] See the DIB 10 Commandments of Software supporting thoughts and recommendations. "*Move to a model of continuous hardware refresh in which computers are treated as a consumable with a 2-3 year lifetime.*"

Don't	Do
Turn documents like this into a process and enforce compliance	Hire competent people with appropriate expertise in software to implement the desire state and give them the freedom to do so ("competence trumps process")

Why this is hard to do, but also why it is worth considering doing it:

- Good engineers want to build things, not just write and evaluate contracts. If their jobs are mainly contracting or monitoring, their software skills will quickly become outdated. This can be solved in the short term by a rotational program: do not allow programmers to stay in contracting for more than 4 years, so their technical capabilities are current.
- The government must team with commercial companies to ensure that it has access to the collection of talent required to develop modern software systems, as well as develop internal talent. The DoD should increase its use of contractors whose aim is not just to provide software, but to increase the software development capabilities and competency of the department. By making use of enlisted personnel, reservists, contractors, and other resources, it is possible to create and maintain highly effective teams who contribute to national security through software development.

Additional Obstacles

In addition to the specific obstacles listed above, we capture here a collection of statutes, regulations, processes and cultural norms that are impediments to implementing a modern set of software acquisition and development principles.

Statutes

The statutes below provide examples of impediments to the implementation of modern software development practices in DoD systems.

Acquisition strategy (10 U.S.C §2431a): 2431a(d) establishes the review process for major defense acquisition programs and is written around the framework of waterfall development for long timescale, hardware-centric programs. In particular, this statute establishes decision-gates at Milestone A (entry into technology maturation and risk reduction), Milestone B (entry into system development and demonstration), and entry into full-rate production. For many software programs this set of terms and approach does not make sense and is incompatible with the ability to deliver capability to the field in a rapid fashion.

Critical cost growth in major defense acquisition programs (10 U.S.C. §2433a [Nunn-McCurdy]): 2433 establishes the conditions under which Congress reviews a major program that has undergone critical cost growth and determines with it should continue. By the time a software program hits a Nunn-McCurdy breach it has already gone well past the point where the program should have been terminated and restarted using a different approach. All software procurement programs should start small, be iterative, and build on success – or be terminated quickly.

Independent cost estimation and cost analysis (10 U.S.C. §2334)

Working capital funds (10 U.S.C. §2208(r)):

- 2+ year lead times from plan to budget does not allow for continuous engineering
- Differentiating software development workload as Research, Development, Test and Engineering (RDT&E), Procurement, or Operations and Maintenance (O&M) is meaningless as there should be no final fielding or sustainment element to continuous engineering.
- System-defined program elements hinder the ability to deliver holistic capabilities and enable real-time resource, requirements, performance and schedule trades across systems without significant work.

Operational Test and Evaluation (10 U.S.C. §139b/d, 10 U.S.C. §2399): 139 establishes the position of the Director of Operational Test and Evaluation (DOT&E) and requires that person to carry out field tests, under realistic combat conditions, of weapon systems for the purpose of determining the effectiveness and suitability of those systems in combat by typical military users. 2399(a) states that a major defense acquisition program "may not proceed beyond low-rate initial production until initial operational test and evaluation of the program, subprogram, or element is completed." 2399(b)(4) further states that the program many not proceed "until the Director [of Operational Test and Evaluation] has submitted to the Secretary of Defense the report with respect to that program under paragraph (2) and the congressional defense committees have received that report." These are obstacles for DevSecOps implementation of software, where changes should be deployed to the field quickly as part of the (continuous) development process. They are an example of a "tailgate" process for OT&E that impedes our ability to deploy software quickly and drives a set of processes in which OT&E impedes rather than enhances the software development process. Instead of this process, Congress should allow independent OT&E of software to occur in parallel with deployment and also require that OT&E cycles for software match development cycles through the use of automated workflows and test harnesses wherever possible.

Additional issues:

- Testing and evaluation (T&E) must be integrated into the development life cycle to facilitate DevSecOps, and reduce operations and sustainment (O&S) costs. T&E should be present from requirements setting to O&S
- Programs need persistent and realistic environments that permit continuous, agile testing of all systems (embedded, networked, etc.) in a representative SoS environment
- Software environments should be part of the contract deliverables and accessible to T&E, including source code, build tools, test scripts, data

Definition of a major acquisition program (10 U.S.C. §2430): The designation of a program as a major acquisition program triggers a set of procedures that are designed for acquisition of hardware. This includes triggering of the DoD Instruction 5000.02, which is currently tuned for hardware systems. An alternative instruction, DoD Instruction 5000.75, is better tuned for software, but can only used for defense *business* systems; it is not valid for "weapons systems."

Depot level maintenance and repair; core logistics (10 U.S.C. §2460, 10 U.S.C. §2464): The definitions of maintenance, repair, and logistics are based on an acquisition model that is appropriate for hardware but not well aligned with the operation of modern software. For example, §2464 says that Services will “maintain and repair the weapon systems.” But software is not maintained, it is *optimized* (with better performance and new functionality) on a *continuous* basis. §2460(b)(1) further states that depot level maintenance and repair “does not include the procurement of major modifications or upgrades of weapon systems that are designed to improve program performance.”

Additional issues:

- DoD’s challenge in shifting from applying a Hardware (HW) maintenance mindset to Software (SW) hinders DoD’s ability to better leverage DoD’s organic SW engineering infrastructure to deliver greater capability to the warfighter.
- DoD’s acquisition process is not emphasizing an upfront focus on design for software sustainment and a seamless transition to organic software engineering sustainment to reduce the life-cycle cost of software and to speed delivery of capability over the life cycle. Such upfront emphasis is critical given the scope, complexity, and mix of the growing software sustainment demand, in the face of persistent affordability concerns.
- DoD’s organic software engineering capabilities and infrastructure are critical to national security, but there is limited enterprise visibility of this infrastructure, its capabilities, workload, and resources to leverage it at the enterprise level to deliver greater capability more affordably to the warfighter.

Regulations

The regulations are the mechanism by which the DoD implements the statutes that govern its operations. They provide additional examples of impediments to the implementation of modern software development practices in DoD systems.

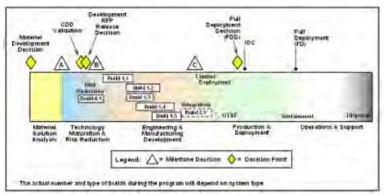
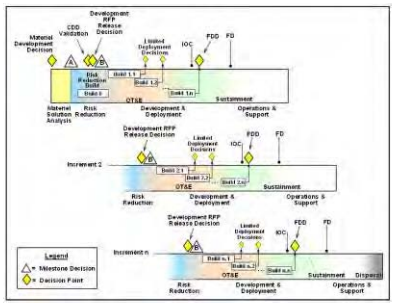
Cost estimating system requirements (48 CFR 252.215-7002) : These regulations set out the expectations for estimation of costs of a program against a set of system requirements. While perhaps appropriate for a hardware-oriented system, they do not take into account the type of continuous development cycle that is required to implement modern software.

Additional requirements for major systems (48 CFR 207.106): These regulations set out procedures for competition of contracts and are written in a manner that separates out the initial deployment of a system with the operation and sustainment of that system. This doesn’t make sense for software.

Processes (Instructions)

The detailed processes used to implement the regulations are laid out in Department of Defense Instructions. We illustrate here some of the specific instructions that are obstacles to implementation of modern software development practices.

Major acquisition program development process (DODI 5000.02, par 5.c.(2) and 5.c.(3)(c)-(d)): These portions of the DoD Instructions apply to "Defense Unique Software Intensive" programs and "Incrementally Deployed Software Intensive" programs. While well-intentioned, they are still waterfall processes with years between the cycles of deployments (instead of weeks). These processes may be appropriate for some embedded systems, but are not the right approach for DoD-specific software running on commercial hardware and operating systems, as the diagrams below illustrate:

Definitely not this:	**Better, but still not right:**	**What we need:**
Specify, design, deploy, sustain  DODI 5000.02, Figure 4. Model 2: Defense Unique Software Intensive Program	 DODI 5000.02, Figure 5. Model 3: Incrementally Deployed Software Intensive Program	Implement, scale, optimize  https://commons.wikimedia.org/wiki/File:Devops-toolchain.svg (modifications licensed CC-BY-SA)
Waterfall development	Waterfall development with overlapping builds	Continuous integration and deployment (DevSecOps)

Requirements for programs containing information technology (DoDI 5000.02, Enclosure 11): This enclosure attempts to define the requirements for ensuring information security. It is written under the assumption that the standard waterfall process is being used.

Preparation, Submission, and Execution of the Budget - Acceptance (OMB Cir A-11, II.10): This document is the primary document that instructs agencies how to prepare and submit budget requests for OMB review and approval. Section II.10 describes the conditions for acceptance of an acquired item by the government, and requires that the asset meets the requirements of the contract. The impact of this procedure is that it establishes a "100% compliance" mentality in order for the government to accept a software "asset."

Culture

In this final section we catalog a list of culture items that do not necessarily require changes in statutes, regulations, or instructions, but rather a change in the way that DoD personnel interpret implement their processes. Changing the culture of DoD is a complex process, depending in large part on incentivizing the behaviors that will lead to the desired state.

Data and metrics

- Multiple, competing, and sometimes conflicting types of data and metrics used, or not used, for assessing software in DOD
- Inability to collect meaningful data about software development and performance in a low cost manner, at scale
- Inability to turn data into meaningful analysis and inability to implement decisions or changes to software activities (L/R/C)

Contracts

- Individual contracts are subject to review processes designed for large programs (of which they are likely enabling). This limits the agility of individual contract actions, even when modular contracting approaches are applied. In addition, the acquisition process is rigid and revolves around templates, boards, and checklists thus limiting the ability for innovation and streamlining execution.
- Contracts focus on technical requirements instead of contractual process requirements. The contract should address overall scope, PoP, and price. The technical execution requirements should be separate and managed by the product owner or other technical lead.
- Intellectual Property (IP) rights are often generically incorporated without considering the layers of technology often applied to a solution. A single solution might include open source, proprietary SW, and government custom code. The IP clauses should reflect all of the technology that is used.

Security Accreditation

- Although developing and operating software securely is a primary concern, the means to achieve and demonstrate security is overly complex and hampered by inconsistent and outdated/misapplied policy and implementation practices (e.g., overlaying historical DoD Information Assurance Certification and Accreditation Process (DIACAP) over risk management framework (RMF) controls for individual pieces of software versus system accreditation). The sense is that the certification and accreditation process is primarily a "check- the-box" documentary process, adds little value to the overall security of the system, and is likely to overlook flaws in the design, implementation, and the environment in which the software operates.
- The DoD needs to be able to calculate the true and component costs for implementing the RMF and certification and accreditation (C&A) in order to identify inefficiencies, duplicative capabilities, and redundant or overlapping security products and services that are being acquired or developed. Absent a set of metrics it is difficult to prioritize risk areas, investments, and evaluating risk reduction and return on investment.
- The DoD needs to ensure that each Joint Capability Area (JCA) flow-down its strategy, best practices, and implementation requirements/guidance for security and accreditation to allow the Component responsible for implementing the software to appropriately tailor RMF and plan the development, accreditation, and operation of the software.
- The DoD needs to provide automated tools and services needed to integrate continuous monitoring with the development life cycle, enable continuous assessment and accreditation, and delegate decision making at the lowest level possible. The DoD should embrace DevSecOps (not just DevOps) and provide policy supported processes, certified libraries, tools, and a toolchain reference implementation to produce "born secure" software

Testing and Evaluation

- The DoD lacks the realistic test environments needed to support test at the pace of modern software methods.
- The DoD lacks the modern software intellectual property (IP) regime needed to support test and evaluation at the pace of modern software methods
- The DoD lack the enterprise knowledge management/data analytics capability needed to support evaluation of test data at the pace of modern software methods

Workforce

- No defined requirements for software developers
- Antiquated policies (talent management, software development)
- Culture and knowledge (DoD, societal, defense contractors)

Appropriations/Funding

- 2+ year lead times from plan to budget does not allow for continuous engineering
- Differentiating software development workload as Research, Development, Test and Engineering (RDT&E), Procurement, or Operations and Maintenance (O&M) is meaningless as there should be no final fielding or sustainment element to continuous engineering.
- System defined program elements hinder the ability to deliver holistic capabilities and enable real-time resource, requirements, performance and schedule trades across systems without significant work.

Infrastructure

- Creating software: The DoD lacks availability of vetted, secure, reusable components, either as source code, or other digital artifacts (think hardened Docker containers or virtual machines (VMs) here). A repository of discoverable, well indexed, vetted, secure, and reusable components could go a long way. This also emphasizes the point that an awful lot of software now-a-days is software by construction with minimal “glue” code applied.
- Building/managing/testing software: There is a general lack of available tools to build software, especially automated tools (testing/scanning/fuzzing etc.) integrated into a secure pipeline supporting rapid agile development. There is also a significant need to have a common, government owned and managed code repository that all programs could/should/must use (e.g., government-furnished GitHub).
- Running/hosting software: The DoD needs to continually push the level of abstraction up as much as possible for programs. Traditionally programs, even cloud-based solutions, tend to start at Infrastructure as a Service (IaaS) and build their own rest of the stack. We need secure and available Platform as a Service (PaaS) and Function as a Service (FaaS) so that programs only need to focus on core business logic and not on securing a database or message bus over and over again.
- Operating/updating securely: Once developed and instantiated on a secure and available platform, we need to continually monitor, red team (automated?), and evolve the software. This requires proper instrumentation, logging, and monitoring of the platform, supporting libraries/components, and the core program code. A standard/common way to provide instrumentation and monitoring of the running services built into the infrastructure would be very helpful.

Requirements

- A byproduct of top-level requirement flow down is rigidity and over specificity at the derived requirements level that greatly hinders agile s/w design.
- Too often exquisite requirements are levied on a system that in turn drive extensive complex software requirements and design, affecting development, integration, and system test.
- Data sets are siloed within programs: a common “law of requirements” is that programs of record try to avoid dependencies with other programs of record. This is problematic for software-based capabilities because data is often siloed within single programs of record. We have network programs to “pass” data, but the promise of artificial intelligence (AI), including machine learning (ML), is that software algorithms can leverage pools of data from disparate sources (data lakes).
- By tying software to a program of record, it becomes harder to transfer that code across systems and data environments. As a result, DoD limits code reuse within and across Services.

Modernization and sustainment

- DoD’s challenge in shifting from applying a hardware maintenance mindset to software hinders DoD’s ability to better leverage DoD’s organic software engineering infrastructure to deliver greater capability to the warfighter.
- DoD’s acquisition process is not emphasizing an upfront focus on design for software sustainment and a seamless transition to organic software engineering sustainment to reduce the life-cycle cost of software and to speed delivery of capability over the life cycle. Such upfront emphasis is critical given the scope, complexity, and mix of the growing software sustainment demand, in the face of persistent affordability concerns.
- DoD’s organic software engineering capabilities and infrastructure are critical to national security, but there is limited enterprise visibility of this infrastructure, its capabilities, workload, and resources to leverage it at the enterprise level to deliver greater capability more affordably to the warfighter.

Acquisition Strategy

- Acquisition policy framework: Create a cohesive acquisition policy architecture within which effective, efficient software acquisition policy has a home.
- Acquisition management and governance: Flip the concept of an oversight model on its head.

DIB Guide: Detecting Agile BS

Agile is a buzzword of software development, and so all DoD software development projects are, almost by default, now declared to be "agile." The purpose of this document is to provide guidance to DoD program executives and acquisition professionals on how to detect software projects that are really using agile development versus those that are simply waterfall or spiral development in agile clothing ("agile-scrum-fall").

Principles, Values, and Tools

Experts and devotees profess certain key "values" to characterize the culture and approach of agile development. In its work, the DIB has developed its own guiding maxims that roughly map to these true agile values:

Agile value	**DIB maxim**
Individuals and interactions over processes and tools	"Competence trumps process"
Working software over comprehensive documentation	"Minimize time from program launch to deployment of simplest useful functionality"
Customer collaboration over contract negotiation	"Adopt a DevSecOps culture for software systems"
Responding to change over following a plan	"Software programs should start small, be iterative, and build on success – or be terminated quickly"

Key flags that a project is not really agile:

- Nobody on the software development team is talking with and observing the users of the software in action; we mean the *actual* users of the *actual* code.[14] (The PEO does not count as an actual user, nor does the commanding officer, unless she uses the code.)
- Continuous feedback from *users* to the development team (bug reports, users assessments) is not available. Talking once at the beginning of a program to verify requirements doesn't count!
- Meeting requirements is treated as more important than getting something useful into the field as quickly as possible.
- Stakeholders (dev, test, ops, security, contracting, contractors, end-users, etc.)[15] are acting more-or-less autonomously (e.g., 'it's not my job.')
- End users of the software are missing-in-action throughout development; at a minimum they should be present during Release Planning and User Acceptance Testing.

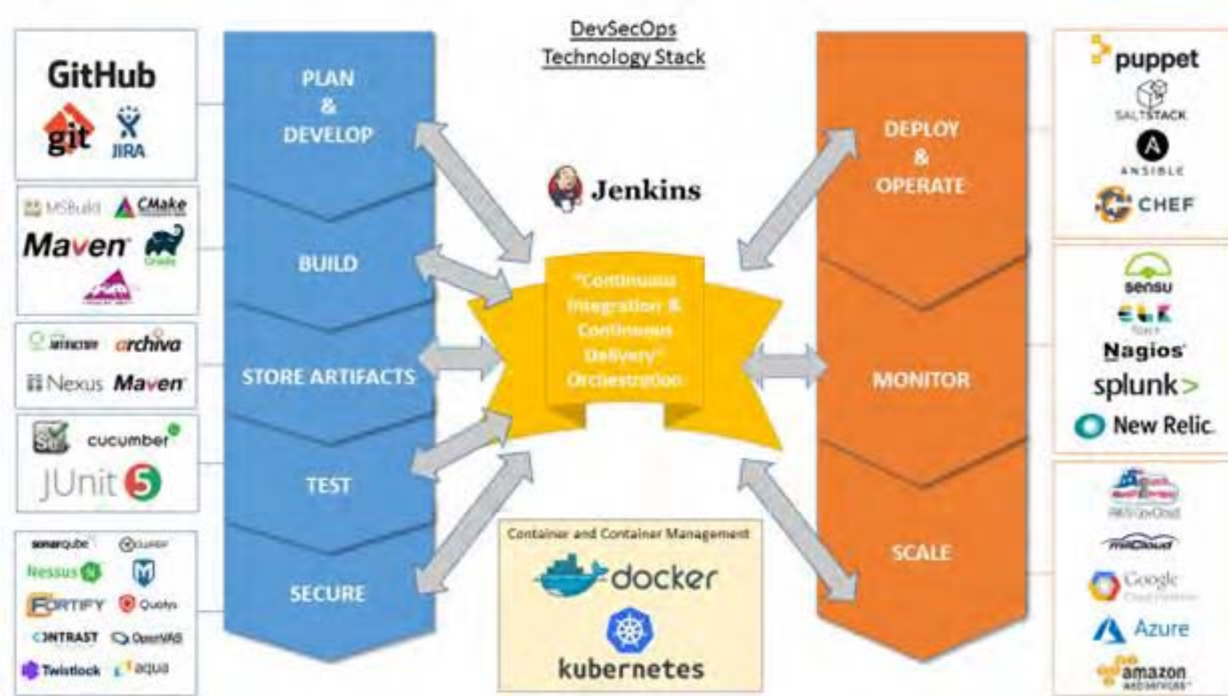
[14] Acceptable substitutes for talking to users: Observing users working, putting prototypes in front of them for feedback, and other aspects of user research that involve less talking.
[15] Dev is short for development, ops is short for operations

- DevSecOps culture is lacking if manual processes are tolerated when such processes can and should be automated (e.g., automated testing, continuous integration, continuous delivery).

Some current, common tools in use by teams using agile development (these will change as better tools become available):[16]

- Git, ClearCase, or Subversion - version control system for tracking changes to source code. Git is the *de facto* open source standard for modern software development.
- BitBucket or GitHub - Repository hosting sites. Also provide issues tracking, continuous integration "apps" and other productivity tools. Widely used by the open source community.
- Jenkins, Circle CI or Travis CI - continuous integration service used to build and test BitBucket and GitHub software projects
- Chef, Ansible, or Puppet - software for writing system configuration "recipes" and streamlining the task of configuring and maintaining a collection of servers
- Docker - computer program that performs operating-system-level virtualization, also known as "containerization"
- Kubernetes or Docker Swarm for Container orchestration
- Jira or Pivotal Tracker - issues reporting, tracking, and management

Graphical version:



Questions to Ask Programming Teams

---

[16] Tools listed/shown here are for illustration only: no endorsement implied.

- How do you test your code? (Wrong answers: “we have a testing organization,” “OT&E is responsible for testing”)
    - Advanced version: what tool suite are you using for unit tests, regression testing, functional tests, security scans, and deployment certification?
- How automated are your development, testing, security, and deployment pipelines?
    - Advanced version: what tool suite are you using for continuous integration (CI), continuous deployment (CD), regression testing, program documentation; is your infrastructure defined by code?
- Who are your users and how are you interacting with them?
    - Advanced version: what mechanisms are you using to get direct feedback from your users? What tool suite are you using for issue reporting and tracking? How do you allocate issues to programming teams? How to you inform users that their issues are being addressed and/or have been resolved?
- What is your (current and future) cycle time for releases to your users?
    - Advanced version: what software platforms to you support? Are you using containers? What configuration management tools do you use?

Questions for Program Management

- How many programmers are part of the organizations that owns the budget and milestones for the program? (Wrong answers: “we don’t know,” “zero,” “it depends on how you define a programmer”)
- What are your management metrics for development and operations; how are they used to inform priorities, detect problems; how often are they accessed and used by leadership?
- What have you learned in your past three sprint cycles and what did you do about it? (Wrong answers: “what’s a sprint cycle?,” “we are waiting to get approval from management”)
- Who are the users that you deliver value to each sprint cycle? Can we talk to them? (Wrong answers: “we don’t directly deploy our code to users”)

Questions for Customers and Users

- How do you communicate with the developers? Did they observe your relevant teams working and ask questions that indicated a deep understanding of your needs? When is the last time they sat with you and talked about features you would like to see implemented?
- How do you send in suggestions for new features or report issues or bugs in the code? What type of feedback do you get to your requests/reports? Are you ever asked to try prototypes of new software features and observed using them?
- What is the time it takes for a requested feature to show up in the application?

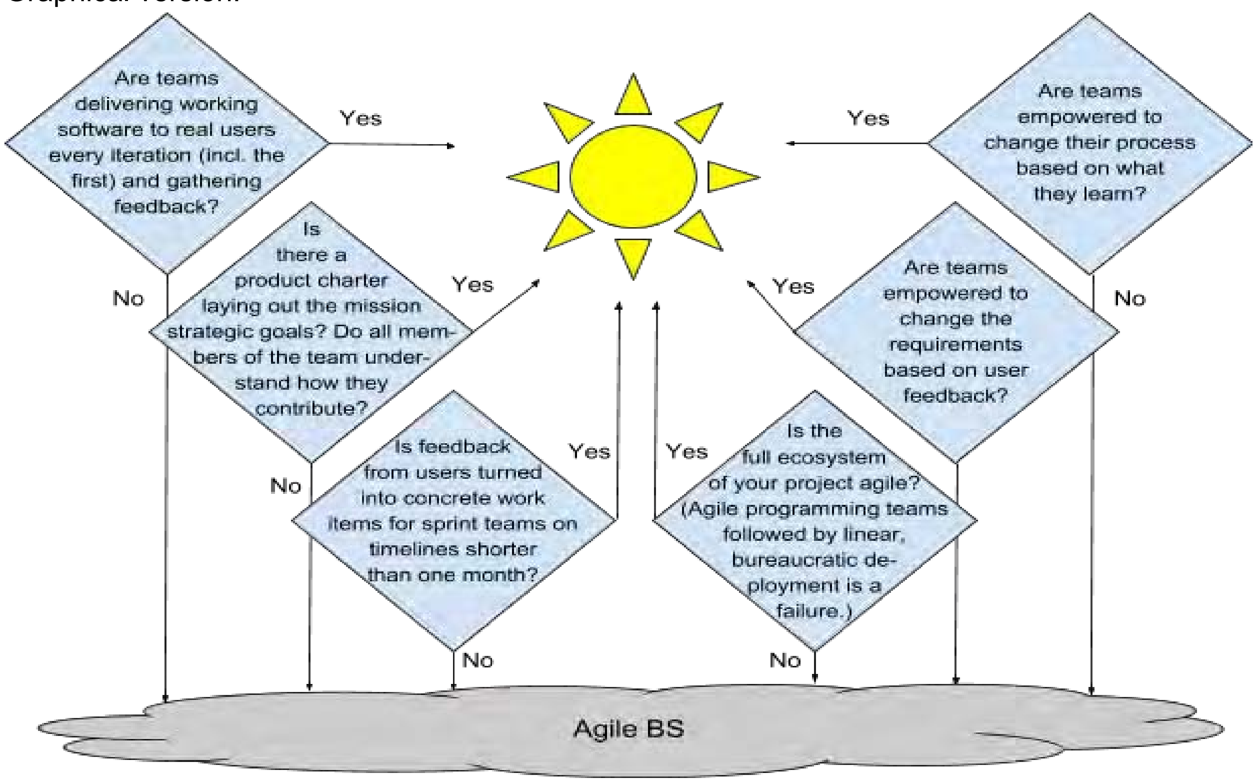
Questions for Program Leadership

- Are teams delivering working software to at least some subset of real users every iteration (including the first) and gathering feedback? (alt: every two weeks)

- Is there a product charter that lays out the mission and strategic goals? Do all members of the team understand both, and are they able to see how their work contributes to both?
- Is feedback from users turned into concrete work items for sprint teams on timelines shorter than one month?
- Are teams empowered to change the requirements based on user feedback?
- Are teams empowered to change their process based on what they learn?
- Is the full ecosystem of your project agile? (Agile programming teams followed by linear, bureaucratic deployment is a failure.)

For a team working on agile, the answer to all of these questions should be “yes.”

Graphical version:



More information on some of the features of DoD software programs are included in the DIB’s “Ten Commandments of Software,” the DIB’s “Metrics for Software Development,” and the DIB’s “Do’s and Don’ts of Software.”

Is Your Development Environment Holding You Back?
A DIB Guide for the Acquisition Community

A strong software development team is marked by some common attributes, including the use of practices, processes, and various tools.

An effective team starts with clear goals. The entire software team should have a clear sense of the project's goals and the value they seek to provide "the client." The goals should be translated into specific objectives, which may be measured in terms of agreed-upon key performance indicators (KPIs) or other frameworks. An effective development environment is one designed to deliver value toward those goals. (This KPI-driven paradigm should *not* be seen as an invitation to reprise an extended debate about requirements.)

Technical practices and processes that enable a development environment to deliver value toward those goals include:

- Organization of activities through discrete "user stories" that can be broken down into smaller components and continually prioritized by the product owner
- Relatively short "sprints" (often two weeks), each ending in a retrospective, that enable measurement and learning throughout the process
- Blameless postmortems that allow for maximum learning and speedy recovery from failures
- Automated testing, security, and deployment
- Testing (including user testing) and security should be shifted to the left and be part of the day-to-day operations within the development teams
- Continuous integration, in which developers integrate code into a shared repository several times a day, and check-ins are then verified by an automated build for early problem detection
- Continuous delivery or continuous deployment, in which the software is seamlessly deployed into staging and production environments
- Trunk-based development, in which team members work in small batches and develop off of trunk or master, rather than long-lived feature branches
- Version control for all production artifacts including open source and third party libraries
- Infrastructure as code: version control for all configuration, networking requirements, container orchestration files, continuous integration/continuous delivery (CI/CD) pipeline files
- Ability to execute A/B testing and canary deployments
- Ability to get rapid and continuous user feedback and to test new features with users throughout the development process

Effective teams will practice continuous delivery, in which teams deploy software in short cycles, ensuring that the software can be reliably released at any time. Continuous deployment can be measured by a team's ability to achieve the following outcomes:

- Teams can deploy on-demand to production or to end users throughout the software delivery life cycle.

- Fast feedback on the quality and deployability of the system is available to everyone on the team, and people make acting on this feedback their highest priority.

Specific measures that will help you gauge if your development environment is working as it should include development frequency; lead time for changes; time to restore service after outage; and change failure rate (rollback deployed code). These questions and data, borrowed from the 2017 State of DevOps Report from DORA, can help assess where your teams stand:

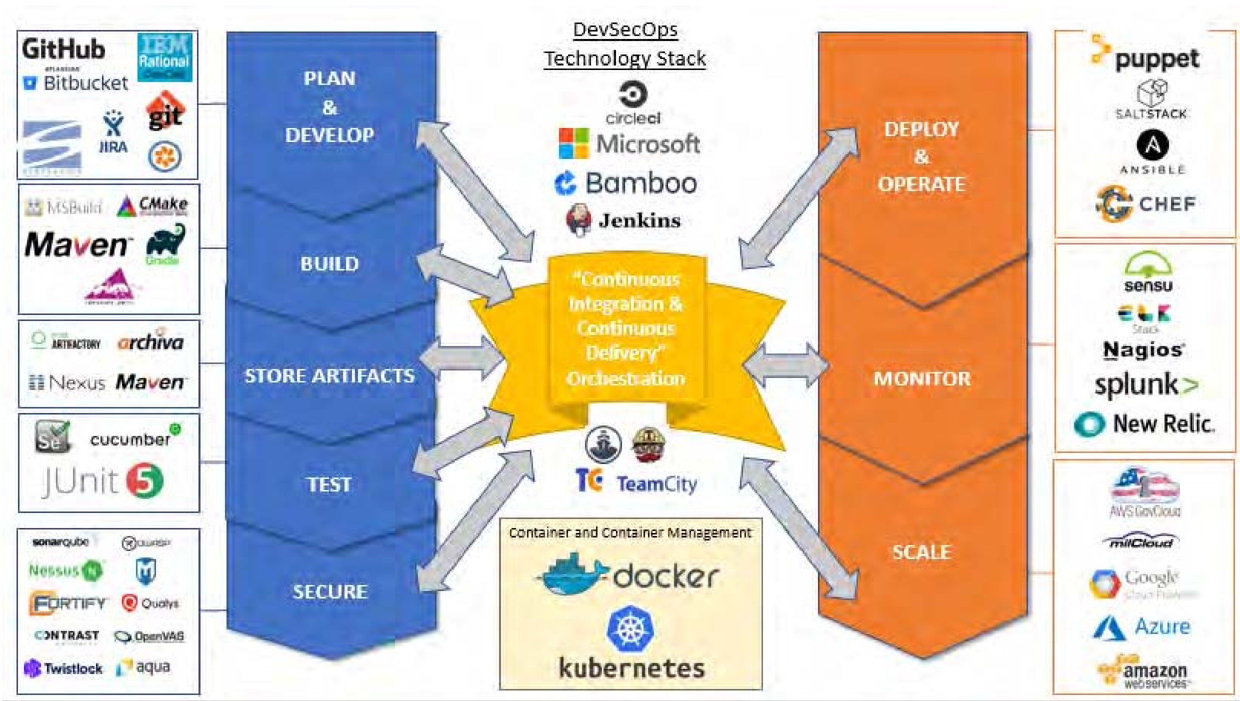
	High performance	**Medium performance**	**Low performance**
Deployment frequency How often does your organization deploy code?	On demand (multiple deploys per day)	Between once per week and once per month	Between once per week and once per month
Lead time for changes What is your lead time for changes (i.e., how long does it take to go from code-commit to code successfully running in production)?	Less than one hour	Between one week and one month	Between one week and one month*
Mean time to recover (MTTR) How long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?	Less than one hour	Less than one day	Between one week and one day
Change failure rate What percentage of changes results either in degraded service or subsequently requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, fix forward, patch)?	0-15%	0-15%	31-45%

* Low performers were lower on average (at a statistically significant level), but had the same median as the medium performers (2017 DevOps Report)

There is ***no exact set of tools*** that indicate that your development environment is working as it should, but the use of some tools will often indicate that the practices and processes above are in place. You commonly see effective software teams using:

- An issue tracker, like Jira or Pivotal Tracker
- Continuous integration and/or continuous integration/continuous delivery (CI/CD) tools, like Jenkins, Circle CI, or Travis CI
- Automated build tools, like Maven, Grable, Cmake, and Apache Ant
- Automated testing tools, like Selenium, Cucumber, J-Unit

- A centralized artifacts repository, like Nexus, Artifactory, or Maven
- Automated security tools for static and dynamic code analysis and container security, like Sonarqube, OWASP ZAP, Fortify, Nessus, Twistlock, Aqua, and more.
- Automation tools, like Chef, Ansible, or Puppet
- Automated code review tools, like Code Climate
- Automated monitoring tools, like Nagios, Splunk, New Relic, and ELK
- Container and container orchestration tools like Docker, Docker Swarm, Kubernetes, and more



Warning signs that you may have screwed up your development environment include:

- If teams cannot effectively track progress toward defined goals and objectives roughly every two weeks
- If teams cannot rapidly deploy various environments that mirror production to test their code such as in development, QA, and staging
- If teams cannot have real-time feedback regarding their code building, passing tests, and passing security scans
- If it takes months for end users to be able to see changes and provide feedback
- If teams cannot rapidly roll-back to previous versions or perform rolling-update to new versions without downtime
- If recovering from incidents results in significant drama or the assignment of blame
- If having code ready to deploy is a big event (it should happen routinely and without drama)
- If changes to the software frequently result in breaking it

If developers are not empowered to change the code or build new functionality based on user feedback, or to change their process based on what they learn.

Is Your Compute Environment Holding You Back? A DIB Guide for the Acquisition Community

To enable software to provide a competitive advantage to the warfighter, DoD must adopt a strategy for rapidly transitioning DoD IT to current industry standards. This modernization agenda should include providing distributed databases and abundant computing power; making bandwidth available as a platform; integrating mobile technologies; and developing DoD platforms for downloading applications. This document outlines compute and infrastructure capabilities that should be available to DoD programmers (and contractors) who are developing software for national defense. The capabilities include:

1. **Scalable compute.** Access to computing resources should never be a limiting factor when developing code. Modern cloud environments provide mechanisms to provide any developer with a powerful computing environment that can easily scale with the needs of an individual programmer, a product development team, or an entire enterprise.

2. **Containerization.** Container technology provides sandbox environments in which to test new software without exposing the larger system to the new code. It “packages up” an application with all of the operating system services required for executing the application and allowing that application to run in a virtualized environment. Containers allow isolation of components (that communicate with each other through well-defined channels) and provide a way to “freeze” a software configuration of an application without freezing the underlying physical hardware and operating system.

3. **Continuous integration/continuous delivery (CI/CD) pipeline (DevSecOps platform).** A platform that provides the CI/CD pipeline is used for automated testing, security, and deployment. This includes license access for security tools and a centralized artifacts repository with tools, databases, and a base operating system (OS) with an existing authorization to operate (ATO).

4. **Infrastructure as code: automated configuration, updating, distribution, and recovery management.** Manual configuration management of operating systems and middleware platforms leads to inconsistencies in fielded systems and drives up the operating costs due to the labor hours required for systems administration. Modern software processes avoid this by implementing “infrastructure as code,” which replaces manual processes for provisioning infrastructure with automated processes that use machine-readable definition files to manage and provision containers, virtual machines, networking, and other components. Adopting infrastructure as code and software distribution tools in a standardized way streamlines uniformity of deployment and testing of changes, which are both vital to realizing the benefits of agile development processes.

5. **Federated identity management and authentication backend with common log file management and analysis.** Common identity management across military, government, and contractors greatly simplifies the assignment of permissions for accessing information across multiple systems and allows rapid and accurate auditing of

code. The ability to audit access to information across multiple systems enables the detection of inappropriate access to information, and can be used to develop the patterns of life that are essential for proper threat analysis. Common identity management can ease the integration of multi-factor authentication across servers, desktops, and mobile devices. Along with public key infrastructure (PKI) integration, it allows verification of both the service being accessed by the user and the user accessing information from the service.

6. **Firewall configuration and network access control lists.** Having a common set of OS and application configurations allows network access control not just through network equipment, but at the server itself. Pruning unnecessary services and forcing information transfer only through intentional interfaces reduce the attack surface and make servers more resilient against penetration. Server-to-server communication can be encrypted to protect from network interception and authenticated so that software services can only communicate with authorized software elements.

7. **Client software.** Remote login through remote desktop access is common throughout DoD. This greatly increases the difficulty of integrating mobile platforms and of permitting embedded devices to access vital information, especially from the field. It also complicates uniform identity management and multi-factor verification, which is key to securing information. By moving to web client access mobile integration - and development - is greatly eased. It also becomes possible to leverage industry innovation, as this is where the commercial sector is heading for all interactions.

8. **Common information assurance (IA) profiles.** Information assurance (IA) for DoD systems is complex, difficult, and not yet well-architected. Test, certification, and IA are almost always linear “tailgate” processes instead of being integrated into a continuous delivery cycle. Common IA profiles integrated into the development environment and part of the development system architecture are less likely to have bugs than customized and add-on solutions.

Desired State with Examples

Effective use of software requires sufficient resources for computing, storage, and communications. Software development teams must be provided with abundant compute, storage, and bandwidth to enable rapid creation, scaling, and optimization of software products.

Modern cloud computing services provide such environments and are widely available for government use. In its visits to DoD programs, the DIB Software Acquisition and Practices (SWAP) team has observed many programs that are regenerating computing infrastructure on their own—often in a highly non-optimal way—and typically due to constraints (or perceived constraints) created by government statutes, regulations, and culture. This approach results in situations where compute capability does not scale with needs; operating systems cannot be upgraded without upgrading applications; applications cannot be upgraded without updating the operating systems; and any change requires a complete information assurance recertification.

Compute platforms are thus "frozen" at a point established early in the program life cycle, and development teams are unable to take advantage of new tools and new approaches as they become available. The DIB SWAP team has noted a general lack of good tools for profiling code, maintaining access and change logs, and providing uniform identity management, even though the DoD has system-wide credentials through Common Access Control (CAC) cards.

It would be highly beneficial to create common frameworks and/or a common set of platforms that provide developers with a streamlined or pre-approved Authority to Operate (ATO). Use of these pre-approved platforms should not be mandated, but they create cost and time incentives by enabling more consolidated platforms. DoD could make use of emerging government cloud computing platforms or achieve similar consolidation within a DoD-owned data center (hybrid cloud). DoD should move swiftly from a legacy data center approach to a cloud-based model, while taking into account the lessons learned and tools and services available from commercial industry, with assumed hardware and operating system updates every 3-5 years.

Warning signs

Some indicators that you may have screwed up your compute environment include:

- Your programmers are using tools that are less effective than what they used in school
- The headcount needed to support the system grows linearly with the number of servers or instances
- You need system managers deployed with hardware at field locations because it is impossible to configure new instances without high skill local support
- You have older than current versions of operating systems or vendor software because it is too hard to test or validate changes
- Unit costs for compute, network transport and storage are not declining, or are not measurable to be determined
- Logging in via remote desktop is the normal way to access an information service
- You depend on network firewalls to secure your compute resource from unauthorized access
- You depend on hardware encryptors to keep your data safe from interception
- You have to purge data on a regular basis to avoid running out of storage
- Compute tasks are taking the same or longer time to run than they did when the system was first fielded
- Equipment or software is in use that has been "end of lifed" by the vendor and no longer has mainstream support
- It takes significant work to find out who accessed a given set of files or resources over a reasonable period of time
- No one knows what part of the system is consuming the most resources or what code should be refactored for optimization
- Multifactor authentication is not being used
- You cannot execute a disaster recovery exercise where a current backup up of a system cannot be brought online on different hardware in less than a day

Getting It Right

These capabilities should be available to all DoD programmers and contractors developing software for national defense:

Scalable compute

- Modern compute architectures
- Environments that make transitions across cloud and local services easy
- Graphics Processing Unit (GPU)- and ML-optimized compute nodes available for specialized tasks
- Standardized storage elements and ability to expand volumes and distribute them based on performance needs
- Standardized network switching options with centralized image control
- Property management tagging—no equipment can be placed in a data center without being tagged for inventory and tracked for End of Life support from vendors
- Supply chain tracking for all compute elements

Containerization

- Software deployment against standard profile OS image
- Containers can be moved from physical to cloud-based infrastructure and vice versa
- Applications and services run in containers and expand or contract as needed
- OS updates separated from application container updates
- Centralized OS patch validation and testing
- Containers can be scaled massively horizontally
- Containers are stateless and can be restarted without impact
- Configuration management for deployment and audit

Continuous integration/continuous delivery (CI/CD) pipeline (DevSecOps platform)

- Select, certify, and package best of breed development tools and services
- Can be leveraged across DoD Services as a turnkey solution
- Develop standard suite of configurable and interoperable cybersecurity capabilities
- Provide onboarding and support for adoption of Agile and DevSecOps
- Develop best-practices, training, and support for pathfinding and related activities
- Build capability to deliver a Software Platform to the Defense Enterprise Cloud Environment
- Self-service portal to selectively configure and deliver software toolkit with pre-configured cybersecurity capabilities

Infrastructure as code: automated configuration, updating, distribution and recovery management

- Ability to test changes against dev environments
- Standardized profiling tools for performance measurement
- Centralized push of patches and updates with ability for rapid rollback
- Auditing and revision control framework to ensure proper code is deployed and running
- Ability to inject faults and test for failover in standardized ways

- Disaster recovery testing and failover evaluation
- Utilization tracking and performance management utilities to predict resource crunches
- Standardized OS patch and distribution repositories
- Validation tools to detect manual changes to OS or application containers with alerting and reporting

Federated identity management and authentication backend with common log file management and analysis

- Common identity management across all DoD and contractors
- Common multifactor backends for authentication of all users along with integration of LDAP/Radius/DNS or active directory services
- Integrated PKI services and tools for automated certificate installation and updating
- Common DRM modules that span domains between DoD/contractors and vendor facilities that can protect, audit and control documents, files, and key information. All encrypted at rest, even for plain text files.
- Useful for debugging and postmortem analysis
- Develop patterns of life to flag unusual activity by users or processes
- Automated escalation to defensive cyber teams

Firewall configuration and network access control lists

- Default configuration for containers is no access
- Profiles for minimal amounts of ports and services being open/run
- All network communications are encrypted and authenticated, even on the same server/container

Client software

- Web-based access the norm, from desktops/laptops as well as mobile devices
- Remote login used as a last resort - not as the default
- Security technical implementation guides (STIGs) for browsers and plugins, as well as common identity management at the browser interface (browsers authenticate to servers as well as servers authenticating to browsers)
- Minimal state kept on local hardware - purged at end of session

Common information assurance (IA) profiles

- Enforces data encrypted in flight and at rest
- Software versions across DoD with automated testing
- Application lockdowns at the system level so only authorized applications can run on configured systems
- “Makefile” to build configurations from scratch from base images in standardized approved configurations
- Use of audit tools to detect spillage and aid in remediation (assisted via DRM

SWAP Program Visits: Questions and Observations

Programs Reviewed

Reviewed 6 programs to date:

- Next Generation fighter jet
- Next Generation ground system
- Kessel Run—AOC Pathfinder
- Space tracking system
- Naval radar system
- Cross-service business system

What we hope to understand:

- Why is the software the way it is?
- How have you gone about developing and deploying it?
- What constraints/obligations have you been under and what would be your recommendations to change those?

Standard Questions

- What is the coding environment and what languages/SW tools do you use?
- What do the software and system architectures look like?
- What is the computational environment (processing, comms, storage)?
- How is software deployed and how often are updates delivered to the field?
- What determines the cycle time for updates?
- How does software development incorporate user feedback? What is the developer-user interface? How quickly are user issues addressed and fixed?
- How long does it take to compile the code from scratch?
- How much access does the DoD have to the source code?
- How is testing done? What tool suites are used? How much is automated? How long does it take to do a full regression test?
- How is cybersecurity testing done? How are programs/updates certified?
- What does the workforce look like (headcounts, skill sets)? How many programmers? How much software expertise is there in the program office?
- What is the structure of the contract with the government? How are changes, new features, and new ideas integrated into the development process?

Preliminary Observations

- Software is being delivered to the field 2-10X slower than it could be due to outdated requirements, test requirements, and lack of trust in SW
- Many systems are using legacy hardware and outdated architectures that make it much harder to exploit advances in computing and communications

- Program requirements were often formulated 5+ years ago (when the threat environment + available technologies were very different => wasted effort)
- New capabilities and features are added in multi-year (multi-decade?) development "blocks" instead of continuously and iteratively
- Most program offices don't have enough expertise in modern SW methods
- Most SW teams are attempting to implement DevOps and "agile" approaches, but in most cases the capabilities are still nascent (and hence fragile)
- Transition to DevOps is often hindered by a gov't support structure focused on technical performance in a waterfall setting ("waterfall with sprints")
- Information assurance (IA) is complex, difficult, and not yet well architected
- Test, certification and IA are almost always linear "tailgate" processes instead of being integrated into a continuous delivery cycle.

What should be done differently in future programs?

- Spend time upfront getting the architecture right: modular, automated, secure
- Make use of platforms (hardware and software) that continuously evolve at the timescales of the commercial sector (3-5 years between HW/OS updates)
- Start small, be iterative, and build on success – or terminate quickly
- Construct budget to support the full, iterative life cycle of the software
- Adopt a DevOps culture: design, implement, test, deploy, evaluate, repeat
- Automate testing of software to enable critical updates to be deployed in days to weeks, not months or years (also requires changes in testing organization)
- Have a local team of DoD software experts who are capable of modifying or extending the software through source code or API access
- Separate development of mission level software from development of IA-accredited platforms

How to Justify Your Budget When Doing DevSecOps

As we transition software development from big spiral programs into DevSecOps, program managers will have to wrestle with using new practices of budget estimation and justification, while potentially being held to old standards that should no longer apply. In addition to all of the regular challenges of retaining a budget allocation (budget reviews, audits, potential reductions and realignment actions, all many times a year), defending a budget for a DevSecOps acquisition requires additional explanation and justification because those charged with oversight—whether inside the Department or in Congress—have come to expect specific information on a tempo that doesn't make sense for DevSecOps projects. Program managers leading DevSecOps projects therefore must not only do the hard work of leading agile teams toward successful outcomes, but also create the conditions that allow those teams to succeed by convincing cost assessors and performance evaluators to evaluate the work differently. Fortunately, commercial industry already has best practices for budget estimation and justification for DevSecOps and that DoD should follow industry approaches rather than create new ones

This DIB Guide is intended to help with this challenge. It seeks to provide guidelines and approaches to help program managers of DevSecOps projects[17] interact with those cost assessors and performance evaluators through the many layers of review and approval authorities while carrying out their vital oversight role. This guide should help with projects where the development processes is optimized for software rather than hardware and where most key stakeholders are aligned around the goal of providing needed capability to the warfighter without undue delay.

Questions that we attempt to answer in this concept paper:

1. What does a well-managed software program look like and how much should it cost?
2. What are the types of metrics that should be provided for assessing the cost of a proposed software program and the performance of an ongoing software program?
3. How can a program defend its budget if the requirements aren't fixed or are changing?
4. How do we estimate costs for "sustainment" when we are adding new features?
5. Why is ESLOC (effective source lines of code) a bad metric to use for cost assessment (besides the obvious answer that it is not very accurate)?

What does a well-managed DevSecOps program look like and how much should it cost?

The primary focus for DevSecOps programs is about regular and repeatable, sustainable delivery of innovative results on a time-box pattern, not on specifications and requirements without bounding time (Figure 1). The fixed-requirements spiral-development spending model has created program budgets that approach infinity. DevSecOps projects, on the other hand will be focused on different activities at different stages of maturity. In a DevSecOps project, management should be tracking services and measuring the results of working software as the product evolves, rather than inspecting end items when the effort is done, as would be expected

[17] Not all software is the same; we focus here only on software programs using or transitioning to DevSecOps.

in a legacy model. Software is never done and not all software is the same, but generally the work should look like a steady and sustainable continuum of useful capability delivery.

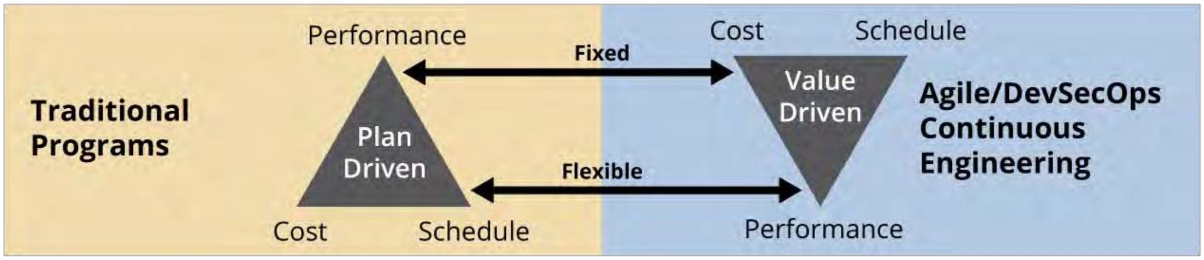


Figure 1. Value Driven Iron Triangle (Carnegie Mellon University, Software Engineering Institute).

- During the creation phase, program managers will most likely decide to adopt Agile based on criteria that fits their design challenge (e.g., software dependent). They would also be motivated to build their products on top of widely used software platforms that are appropriate for the technical domain at hand (e.g., embedded vs. web applications). During this phase team also establishes base capability and what they consider a minimum viable product (MVP).[18] This is where all programs start and many should end. Starting small and incrementing is not only the right way to do software, but it is also a great way to limit financial exposure. A key tenet of agile development is learning early and being ready to shift focus to increase the likelihood for success.
- During the scaling phase, the entire team (industry and government) commit and learn how to transition to appropriate agile activities that are optimizing for implementing DevSecOps for the project. This should focus the team on transitioning to a larger user base with improved mechanisms for automated testing (including penetration testing), red team attacks, and continuous user feedback. A key management practice in agile development is to keep software projects to a manageable size. If the project requires more scope, divide the effort into modular, easily connected chunks that can be managed using agile methods and weave the pieces together in implementation.
- Once into implementation, a well-managed program should have a regular release cadence (e.g., for IT projects every 2-3 weeks, while safety-critical products could run a bit longer, 3-4 weeks). Each of these releases delivers small increments of software that are as intuitive to use as possible and directly deployable to actual users. DevSecOps programs move from small successes into larger impacts.

With allowances made for different sizes of project, DevSecOps should share certain characteristics, including:

- An observer should easily find an engaged program office, as well as development teams that are small (5-11 people), and well connected to one another through structured meetings and events (a.k.a. "ceremonies").

[18] The MVP should not be overspecified since the main goal is getting the MVP into the hands of users for feedback.

- A set of agile teams work on cross-functional capabilities of the system and include a planning team and a system architecture team.
- The teams should have frequent interaction with subject matter experts and users from the field or empowered product owners. Active user engagement is a vital element of an Agile approach, but getting actual users (*not* just user representatives) to participate also needs to be a managed cost that the program needs to plan for.
- The project should have a development environment that supports transparency of the activities of the development teams to the customer. Maximal automation of reporting is the norm for commercial development and should be for DoD programs as well.
- The program should include engaged test and certification communities who are deeply involved in the early stages (i.e., who have "shifted left") and throughout the development process. Not just checkers at the end of that process. They would help design and validate the use of automation and computer-assisted testing/validation tools whenever possible as well.
- Capability should also be delivered in small pieces on a continuing basis—as frequently as every two weeks for many types of software (see the DIB's Guide to Agile BS).

The cost of a program always depends on the scale of the solution being pursued, but in an agile DevSecOps project, the cost should track to units of 5–11-person cross-functional team (team leader, developers, testers, product owners, etc.) with approximately 6–11 teams making up a project. If the problem is bigger than that, the overall project could be divided up into related groups of teams. A reliance on direct interaction between people is another central element of Agile and DevSecOps; the communication overhead means that this approach loses effectiveness with too many people in a team (typically 5–11 cross-functional members). Also, groups of teams have difficulty scaling interactions when the number of teams gets too large (less than twelve). A team-of-teams approach will allow scaling to fit the overall scope. Organizing the teams is also a valuable strategy where higher level development strategies and system architectures get worked out and the lower level teams are organized around cross-domain capabilities to be delivered. Cost incentives for utilizing enterprise software platform assets should be so attractive, and the quality of that environment so valuable, that no program manager would reasonably decide to have his/her contractor build their own.

Here are some general guidelines for project costs when pursuing a DevSecOps approach:

- Create: deliver initial useful capability *to the field* within 3-6 months (the use of commodity hardware and rapid delivery to deployment). If this cannot be achieved, it should be made clear that the project is at risk of not delivering and is subject to being canceled. Outcomes and indicators need to be examined for systematic issues and opportunities to correct problems. Initial investment should be limited in two ways: 1) in size to limit financial exposure and 2) in time to no more than 1 year.

- Scale: deliver increased functionality across an expanding user base at decreasing unit cost with increased speed. Investment should be based on the rate limiting factors of time and talent, not cost. Given a delivery cycle and the available talent, the program should project only spending to the staffing level within a cycle.

- Good agile management is not about money, it is about regular and repeated deliver. That is to say, it is about time boxing everything. Releases, staffing, budget, etc. Nick, strongly recommend that you rework this to reflect time boxing as the most important aspect of "defending your agile budget.

- Optimize: deliver increased functionality fixed or decreasing unit cost (for a roughly constant user base). Investment limit should be less than 3 project team sets[19].

What are the types of metrics that should be provided for assessing the cost of a proposed software program and the performance of an ongoing software program?

Assessing the cost of a proposed software program has always been difficult, but can be accomplished by starting one or more set of project teams at a modest budget (1-6 sets of teams) and then adjusting the scaling of additional teams (and therefore the budget) based on the value those teams provide to the end user. It may be necessary to identify the size of the initial team required to deliver the desired functions at a reasonable pace and then price the program as the number of teams scales up. The DIB recommends that program managers start small, iterate quickly, and terminate early. The supervisors of program managers (e.g., PEOs) should also reward aggressive early action to shift away from efforts that are not panning out into new initiatives that are likely to deliver higher value. Justifying a small budget and getting something delivered quickly is the best way to provide value (and the easiest way to get and stay funded).

The primary metric for an *ongoing* program should be user satisfaction and operational impact. This can be different for every program and heavily depends on the context. The challenge, and therefore the responsibility of the PM then is to define mission relevant metrics to determine achieved and delivered value. Examples could include, personnel hours saved, number of objects tracked or targeted, accuracy of the targeting solution, time to first viable targeting solution, number of sorties generated per time increment, number of ISR sensors integrated, etc. Other key metrics that are often advocated by agile programs (inside and outside of DoD) include:

- *deployment frequency* (Is the program getting increments of functionality out into operations?),
- *lead time* (how quickly can the program get code into operation?),
- *mean time to recover* (how quickly can the program roll back to a working version, if problems are found in operation?), and
- *change fail rate* (rate of failures in delivered code).

These four break down into two process metrics (release cadence and time from code-commit to release candidate, and two are quality metrics (change fail rate and time to roll back). In addition, each project should also have 3-5 key value metrics that are topical to the solution space being addressed. Metrics must be available both to the teams and the customer so they can see how their progress compares to the projected completion rate for delivering useful functionality. A key reason for Government access to those metrics is for supporting the real-time tracking of progress and prediction of new activities in the future. The biggest difference between a DevSecOps

[19] Average of 8 people per team with an average of 8 teams per project.

program and the classic spiral approach is that the cadence of information transparency between the developers and the customer is, at slowest, weekly, but if properly automated, should be instantly and continuously available.. Quality metrics and discovery timelines (such as defects identified early in development versus bugs identified in the field) can also be used to evaluate the maturity of a program. This kind of oversight enables fast and effective feedback before the teams end up in extremis, or set up unrealistic expectations.

Software projects should be thought of as a fixed cadence of useful capability delivery where the "backlog" of activities are managed to fit the "velocity" of development teams as they respond to evolving user needs. Data collected on developers inside of the software development infrastructure can be provided continuously, instead of packaged into deliverables that cannot be directly analyzed for concerns and risks.

The DIB's "Metrics for Software Development" provide a set of metrics for monitoring performance:

1. Time from program launch to deployment of simplest useful functionality.
2. Time to field high priority functions (spec → ops) or fix newly found security holes
3. Time from code committed to code in use
4. Time required for regression tests (automated) and cybersecurity audit/penetration tests
5. Time required to restore service after outage
6. Automated test coverage of specs/code
7. Number of bugs caught in testing vs field use
8. Change failure rate (rollback deployed code)
9. Percentage of code available to DOD for inspection/rebuild
10. Complexity metrics
11. Development plan/environment metrics

These data provide management flexibility since data about implementation of capability can be made *during* development—instead of at a major milestone review or after "final" delivery, when changing direction comes at a much higher cost and schedule impact. So data collection and delivery must be continuous as well. Another note, these metrics are recommendations and not intended to be prescriptive. Use what fits your program. Not all of these may be required.

An additional pair of overarching key metrics are headcount and expert talent available. If the project headcount is growing, but delays are increasing,, aggressive management attention is called for. The lack of expert talent also increases risks of failure.

How can a program defend its budget if the requirements are not fixed years in advance, or are constantly changing?

It is relatively easy to defend changing capability by making changes to the software of existing systems, as compared to starting up a new acquisition. Software must evolve with the evolving needs of the customers. This is often the most cost effective and rapid way to respond to new requirements and a changing threat landscape. A new approach to funding the natural activities of continuous engineering and DevSecOps requires a system that can prioritize new features and manage these activities as dependent and tightly aligned in time (see Figure 1). A continuous deployment approach is needed for delivering on the evolving needs culled from user involvement combining R&D, O&M, Procurement, and Sustainment actions within weeks of each other, not years (see Figure 2). Great software development is an iterative process between developers and users that see the results of the interaction in new capability that is rapidly put in their hands for operational use.

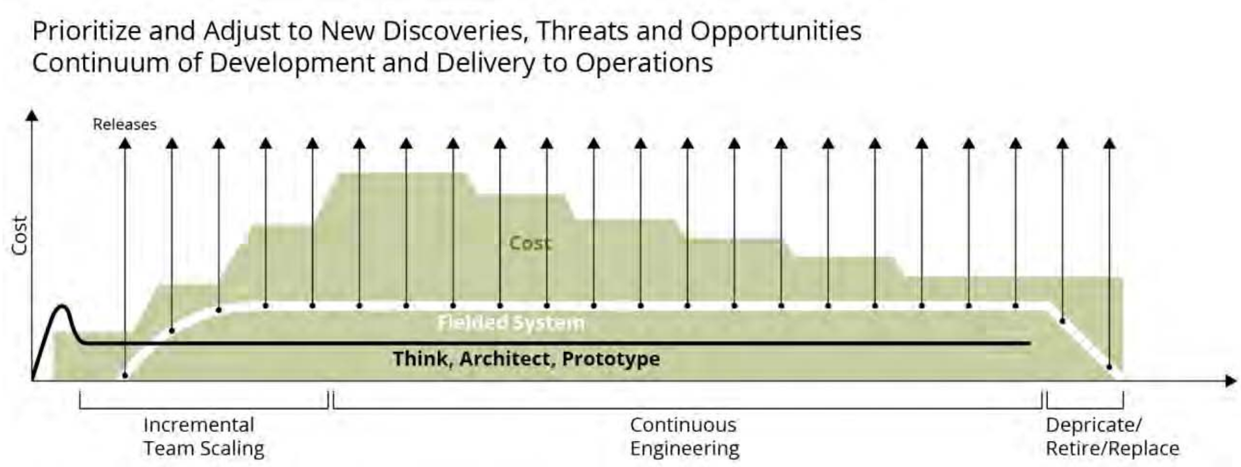


Figure 2. Continuous Delivery of Modular Changes to Working Software (Carnegie Mellon University, Software Engineering Institute).

Elements to address include in budget justification and management materials:

- DevSecOps programs have to be at least as valuable and urgent to fund as a classic DoD spiral program in the hyper-competitive budget environment. Over time, DoD will realize that the DevSecOps approach is inherently more valuable. However, time is of the essence. It must be acknowledged that the current waterfall approach is no longer serving us well in the area of software. The mainstream software industry has already made the move to agile ten years ago and the methods are rigorously practices and proven valuable.
- The classic approach of doing cost estimates of designs based on fixed requirements has always been wrong, even when accounting for intended capability growth because the smart adversaries get a continuous vote on the threat environment. Accurate prediction of a rapidly changing technology environment and solution methods only exacerbate the unknowns of product development outcomes.
- DevSecOps programs have requirements, but start out at a higher level and use a disciplined approach to continuously change and deliver greater value.
- DIB's "Ten Commandments of Software" calls for the use of shared infrastructure and continuous delivery, which will reduce the cost of infrastructure and overhead, thus freeing up capital to advance unique military capability.
- Data available above the program manager's level has been insufficient for cost and program evaluation communities to assess software projects. However, the reporting of metrics that are a natural consequence of using DevSecOps approaches should be automated to provide transparency and rapid feedback.

The benefits of this approach are manifold. It allows for thoughtful rigor up front and early and the rapid abandonment of marginal or failure-prone approaches early in the design cycle before large investments are sunk. Details are allowed to evolve. More stable chunks of capability are defined at the "epic" level and a stable cadence of engineering and design pervades the life cycle. Under this operational concept, testing is performed early, during the architecture definition stage and continuously as new small deployments of functionality are delivered to the user. The identification of budget is redistributed as value is provided and validated for warfighting impact. A closer alignment of flexible requirements and budget allocation/ appropriation will be necessary in order to ensure that the national defense needs and financial constraints are continuously managed.

Continuous access to design and delivery metrics will illuminate developer effectiveness, user delight, and the pace of delivery for working code to include analytical data for in-stride oversight and user/programmatic involvement This will replace the standard practice of document-based deliverables and time-late data packages that take months to develop and are not current when provided.

The way that DoD has classically managed these activities is to break them up into different "colors of money" associated with hardware-centric phases (see Figure 3). This places an artificial burden on excellence in software. Rapid and continuous delivery of working code requires addressing these different types of requirements within shorter time-horizons than is natural for the existing federal budgeting process.

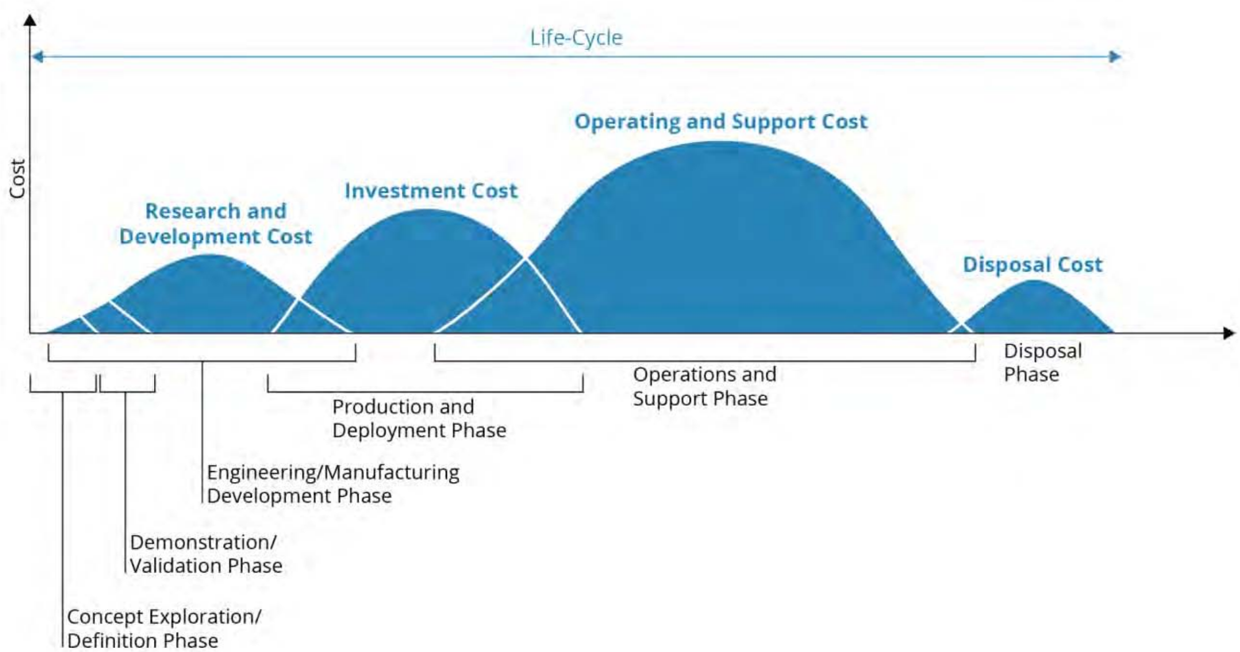


Figure 3. Notional DoD Weapon System Cost Profile (Defense Acquisition University).

In addition, the classic approach of developing detailed technical requirements far in advance of performing product design needs to be replaced. The new paradigm must begin with an architecture that will support the requirements and scale associated with needs for future compatibility (e.g., modularity security, or interoperability). Also, using an agile approach, a program can incorporate the best available technologies and methods throughout the entire life cycle and avoid a development cycle is longer than the useful life of the technology it is built on. Getting these things wrong is not recoverable. Establishing detailed requirements over a period of years before beginning, to be followed by long development efforts punctuated by major design reviews (i.e., Software Requirements Review, Preliminary Design Review, Critical Design Review, Test Readiness Review, Production Readiness Review) that require a span of years between events are inherently problematic for software projects for at least two reasons. First, these review events are designed around hardware development spirals that are time-late and provide little in the way of in-stride knowledge of software coding activities that can be used to aid in real-time decision making. Second, development teams are in frequent contact with users and adjusting requirements as they go, which up-ends the value of major design reviews that are out of cadence with the development teams. DevSecOps implementation methods such as feature demonstrations and cycle planning events provide much more frequent and valuable information on which program offices can engage to make sure the best value is being created.

Defending a budget has to be done in terms of providing value. Different programs value different things—increasing performance, reducing cost, minimizing the number of humans-in-the- loop—so there is no one size fits all measure. But in an agile environment, knowing what to measure to show value is possible because of the tight connection to the user/warfighter. Those users are able to see the value they need because they are able to evaluate and have an impact on the working software. This highlights the need to collect and share the measures that show improvement against a baseline in smaller increments.

How do we do cost for "sustainment" when we are adding new features?

The first step is to eliminate the concept of sustaining a fixed base of performance. Software can no longer be thought of as a fixed hardware product like a radar, a bomb, or a tank. That leads to orphaned deployments that need unique sustainment and a growth of spending that does not deliver new functionality (see Figure 4).

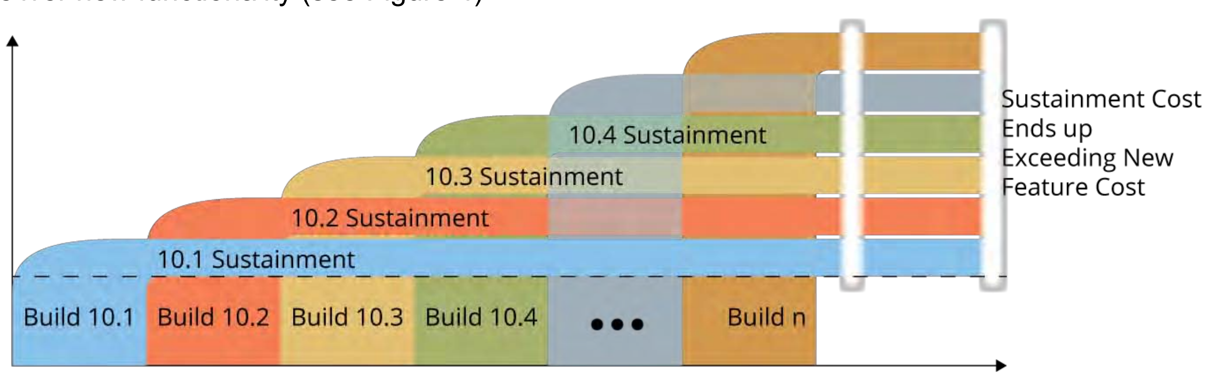


Figure 4. Layers of Sustainment to Manage Unique Deployments

Software can continue to evolve and be redeployed for comparatively little cost (see Figure 2). Users continue to need and demand greater performance and improved features, if for no other reason than to retain parity with warfighting threats. Also internal vulnerabilities and environmental updates must be continuously deployed to support ever improving cyber protections. The most secure software is the one that is most recently updated. Lastly, new capabilities for improved warfighting advantage are most often affordably delivered through changes to fielded products.

Software development is a very different way of delivering military capability. It should be considered more like a service of evolving performance. When new features are needed, they get put in the backlog, prioritized, and scheduled for a release cycle (see Figure 5). If the program is closer to providing satisfactory overall performance, then the program can dial down to the minimum level needed to satisfy the users and keep the environment and applications cyber-secure. It can be thought of as recursive decisions on how many (software) “squadrons” are required for our current mission set and then fund those teams at the needed staffing level to create, scale, or optimize the software (depending on the stage of continuous development). Because these patterns can be scaled up and down by need in a well-orchestrated way, new contracting models are available that might not have been used in the past. For example, fixed price contracts for a development program was strongly discouraged, but under this model, where schedule and team sizes are managed and capability is grown according to a rigorous plan (Figure 1), a wider array of business, contracting and remuneration models can be explored.

Figure 5. Release Cycle With New Opportunities, Discoveries and Response to Threats (Carnegie Mellon University, Software Engineering Institute).

Two financial protections built into acquisition laws and regulations need to be reexamined in the light of software being continuously engineered, vice sustained: Nunn-McCurdy and the Anti-Deficiency Act. The continuous engineering pipeline will continue to push out improved capability until the code base is retired. While Nunn-McCurdy is a valid constraint for large hardware acquisitions, it does not apply to software efforts. In a similar vein, software should also never trigger the Anti-Deficiency Act - just like keeping a ship full of fuel, or paying for air-traffic controllers; we know we are going to be doing these things for a long time. To build a ship that will need fuel for 40 years does not invoke the ADA. Therefore, starting a software project that will incrementally deliver new functionality for the foreseeable future should not do so either.

Why is ESLOC a bad metric to use for cost assessment?

The thing we really want to estimate and then measure is the effort required to develop, integrate, and test the warfighting capability that is delivered by software. SLOC might have been a used as a surrogate for estimating the effort required, but it has never been accurate. Not all software is the same, not all developers are the same, and not all development challenges use the same approaches to reduce problems into solutions. For example, in a project there may things like detailed algorithms that require deep expertise and detailed study to properly implement small amounts of code, running alongside large volumes of automatically generated code of relatively trivial complexity. Many different levels of effort are needed to create a line of code that will deliver military capability, and estimations of source code volume is an inherently problematic and error-filled approach to describing the capability thus produced. That's why DevSecOps efforts use measures of relative effort like story points to communicate across a particular set of teams how much effort it will take to turn a requirement into working software that meets an agreed upon definition of done within a set cadence of activity. Because these story points are particular to a specific team, they do not accurately transition to generally prescribable measures of cost.

Estimating by projecting the lines of code starts the effort from the end and works backwards. SLOC is an output metric (something to know when the job is done—akin to predicting what size clothing your child will wear as an adult). It does not capture the human scale of effort. Traditional models like COCOMO or SEER attempt to use a variety of parameters in their models to capture things like formality, volatility, team capabilities, maturity and others. However, these surrogates for effort have well documented error sources and have failed time and again to accurately capture the cost of executing a software program. There are also inherent assumptions built into these models that are obviated by performing agile development of capability models running on a software platform.

In the beginning stages of DoD's transformation to DevSecOps methods, the development and operations community will need to work closely with the cost community to derive new ways of predicting how fast capability can be achieved. For example, estimating how many teams worth of effort will be needed to invest in a given period of time to get the functionality needed. As they do this, it needs to be with the understanding that the methods are constantly changing and the estimation methods will have to evolve too. New parameters are needed, and more will be discovered and evolve over time.

Appendix F: SWAP Working Group Reports

The information in this appendix was developed based on feedback and analysis performed by members of a working group that included subject matter experts (SMEs) within the Department who provided input for consideration to the SWAP study. The working group was asked to: (1) distill the feedback received from case studies, interviews, literature reviews, and feedback from the Board members into main issue points; (2) as SMEs identify the statutory, regulatory, and cultural obstacles to achieving the Board's vision for a desired end state; and (3) provide suggested language to remove the barriers.

The following reports were generated by 10 subgroups:

- Acquisition Strategy
- Appropriations
- Contracting
- Data and Metrics
- Infrastructure
- Requirements
- Security Accreditation/Certification
- Sustainment and Maintenance
- Test and Evaluation
- Workforce

These reports describe input to the SWAP study and the specific views and ideas for change in the reports do not necessarily reflect the final views of the SWAP study. These reports have been lightly edited by the study for consistency with the terminology of this report and are included to provide context and insight into our final themes, lines of effort, and recommendations.

Appendix F.1: Acquisition Strategy Subgroup Report

Contributing authors: Melissa Naroski Merker (lead), Jeff Boleng, Nicolas Chaillan, Ben FitzGerald, Jonathan Mostowski, Don Johnson, COL Harry Culclasure (809 Panel), Gabe Nelson (809 Panel), Larry Asch (809 Panel), Nick Tsiopanas (809 Panel), Nick Kosmidis.
Additional advice / assistance from MITRE, IDA, and DAU

This appendix examines pain points, obstacles, change ideas, and future vision for the Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) Study in the area of Acquisition Strategy and Oversight (i.e., *Acquisition Environment*). In 2017 the Office of the DASD(C3CB) under the ASD(A) commissioned an IT acquisition study with Deloitte. The study recommended the following attributes of an effective and efficient IT acquisition structure:

- *Fast* to incorporate current technology and make efficient use of Agency resources
- *Flexible* and adaptable to support rapid changes in technology and input from stakeholders about capability needs
- *Collaborative* to seek stakeholder involvement and input to be incorporated throughout

In a previous study completed in September 2016, Deloitte also provided key findings on commercial IT practices. Findings were taken into consideration when forming the proposals following in this appendix. The team recognizes that DoD is falling short of the preferred attributes outlined above with the current IT acquisition structure, in addition to multiple statutory, regulatory, and cultural issues that currently hinder an effective and efficient DoD acquisition environment that would benefit from reform.

Pain Points

Acquisition Policy Environment. DoD lacks a cohesive acquisition policy architecture and robust policy for software acquisition. Existing policies, to include tangential or supplemental policies that are integral to the operation of the defense acquisition system, do not fit well together and result in discrepancies, conflicts, and gaps. The defense acquisition system is monolithic, compiled in pieces as needs arose instead of as an integrated and evolving environment. It has proven unable to keep up with or remain ahead of the pace of change and technological advancements that require speed and agility. While it has regularly been revised, the changes tend to be conservative and incremental, requiring the agreement of too many parties protecting narrow interests and who are reluctant to relinquish authority or evolve. The system remains focused on oversight and situational control rather than insight and trust. The policies, practices, and documents become quickly entrenched and manifest themselves in the form of the Department's culture, leading to additional bureaucracy and decreased levels of organizational trust, that are difficult to rapidly reverse. Furthermore, the environment is risk averse, seeking out what is perceived to be the "safest" route to get things done, stifling the innovation and risk- taking that's required to maintain an advantage over adversaries.

As an example, one DoD weapons system program, which is implementing a DevSecOps pipeline to enable agile capability releases, informed us it took 18 months to get approval of a Test and

Evaluation Master Plan (TEMP). The process within the TEMP drove them into sequential developmental and operational test—which is antithetical to continuous delivery under the DevSecOps concept.

Governance and Management. The Department lacks a strategic approach that recognizes software's criticality as the backbone and nervous system of the Department's mission and operations, often leading to widespread duplication of capabilities that could be consolidated and scaled at an enterprise level (whether Service-enterprise or OSD-enterprise). This absence of any strategy, compounded by a long-standing lack of organizational trust in the Department, is exemplified by various situations in the software environment. For example, the lack of reciprocity on matters such as security standards, architecture, and compliance methods—my way is "better" (insert "less expensive," "more efficient," "more effective") than your way, or, "our requirements/processes are unique," regardless of validity. Further, DoD issues separate policies on matters such as cloud, architecture, and risk management, with no unified approach at the strategic level. Management and governance of these matters takes the form of prolific numbers of senior working groups (or equivalent) that make few decisions but have frequent meetings. DoD's lack of an overarching strategic plan for key technologies, with a robust decision making framework that pushes responsibility and authority down to the lowest executable level, creates inefficiency, duplication, and waste.

Organization and Culture. DoD lacks an organizational structure with clear responsibility and authority for software acquisition and management; there are confusing roles and responsibilities between DoD CIO, USD(A&S), and the DoD CMO. This state of ambiguity leads to overlap, inefficiency, and unnecessary bureaucracy; and it is replicated at the Service level. The result is a slow, rigid, siloed organization unable to adapt in the present and plan for the future in order to maintain competitive advantage. DoD is not a change-ready environment and the acquisition system was not designed for rapid change. DoD employees tend to receive change mandates rather than participating in them. A case in point is that when DoD issues a policy, the Services will implement their own supporting version or "supplemental guidance," which expands the policy and introduces multiple layers of bureaucracy, eliminating any semblance of flexibility that was intended by the original policy issued. For example, the Department issued DoD Instruction 5000.75 in February 2018, a tailored requirements and acquisition approach for business systems. Subsequently, the Army produced accompanying implementation guidance—91 pages—which introduces additional forms, templates, processes, and time constraints.

Desired (end) state An acquisition system that enables rapid delivery of cost-efficient, relevant software capability through the application of creative compliance and fact-based critical thinking under a logical and minimal policy framework. The Department treats software as a national security capability and continuously retrains the workforce to be able to adapt to an ever-changing technology environment, embraces continuous collaboration between user and developers, embraces changing requirements, accepts and take risks, and deliver adversary- countering capabilities to the warfighter. Executing the approach requires an end state with an efficient contracting environment; a culture that rewards informed risk-taking and fast failures; the use of limits or guardrails instead of prescriptive requirements that limit creativity; outcome-based metrics that focus on value vs. execution against a plan; and a move away from traditional funding models and compliance-driven management.

Obstacles The Department operates with a general lack of urgency regarding its software—it is not recognized or treated as a national security capability. There is an aversion to informed risk-taking regarding new and innovative approaches to doing business and adopting emerging (or even simply relevant) technologies, even though it's risky, or riskier, to continue using outdated technologies that are not secure or facing obsolescence in the face of evolving threats. Dramatic changes in policy or process are viewed as risky yet our current ways of operation are not despite a known degradation in strategic advantage previously enjoyed over adversaries. The inability to evolve and support rapid changes in technology and input from stakeholders about capability needs is bred through organizational silos and stovepipes that stifle the collaboration necessary to develop and operationalize software. Further, stakeholder involvement is limited by following restrictive controls, timelines, and processes in a sequential manner that impedes progress and results in a lower state of readiness. The duplication of authorities and responsibilities among organizations both horizontally and vertically, within the defense acquisition system only exacerbates an already complex environment where a protectionist culture is ingrained and the workforce is not incentivized to change. In its endeavors to improve the status quo, "help" from Congress over the past decades translates into entrenched policies, processes, and procedures—"cultural norms" that are difficult to reverse.

Ideas for Change

*Acquisition Policy Environment***.** Define software as a critical national security capability under Section 805 of FY16 NDAA "Use of Alternative Acquisition Paths to Acquire Critical National Security Capabilities." Create an acquisition policy framework that recognizes that software is ubiquitous and will be part of all acquisition policy models. Recommend the creation of a clear, efficient acquisition path for acquiring non-embedded software capability. Reconcile and resolve discrepancies among supplemental policies that lead to conflicts. Consider the following tenets in development of a reformed software acquisition policy:

- Emphasis on quickly delivering working software
- Encourage projects and pilot efforts that serve to reduce risk and complexity - fail fast
- Reimagine program structures and program offices—i.e., accommodate move to "as-a-service" capabilities, agile, microservices, and micro-applications
- Iterative, incremental development practices based on agile methods
- Rapid adoption of emerging technologies through piloting or prototyping
- Elimination of traditional A, B, C milestones; replaced by more sprint-centric decision points
- Elimination of arbitrary phases or merge phases to reflect rapid, agile development methods
- Tailor in requirements (statutory, regulatory—i.e., documentation) rather than tailor out;

start with a minimum set

- No big-bang testing with sequential DT/OT; move to fully integrated test approaches driven by automated testing as well as regular, automated cybersecurity scanning
- Use a "guardrail-based" (upper/lower limit) approach for software requirements rather than defining every requirement up front
- Track value-driven outcome metrics which can be easily and continuously generated rather than measuring execution against a plan

Governance and Management - Software as an Asset. Develop an enterprise-level Strategic Technology Plan that reinforces the concept of software as a national security capability. Include an approach for enterprise-level DevSecOps and other centralized infrastructure development and management, an approach for shared services, and applications management. The plan should recognize how disruptive technologies will be introduced into the environment on an ongoing basis. Ensure appropriate integration of a data strategy and the Department's Cloud Strategy. Examine a Steering Committee approach for management.

Organization and Culture Reform. Examine roles and responsibilities with the intent to streamline reconcile, and resolve discrepancies for software acquisition and management among the DoD CIO, the USD(A&S) and the CMO. Re-focus the software acquisition workforce on teaming and collaboration, agility, improved role definition, career path advancement methods, continuing education and training opportunities, incentivization, and empowerment. Involve them in the change process.

Appendix F.2: Appropriations Subgroup Report

Contributing author: Jane Rathbun (lead)

The Department's current Planning, Programming, Budgeting and Execution (PPBE) system framework and process using defined Program Elements (PEs), is categorized by life-cycle–phased appropriations, and requires two years or more in lead time from plan to start of execution. This approach was designed and structured for traditional waterfall acquisition used to deliver monolithic platforms such as aircraft, ships, and vehicles. The PPBE framework and process is challenging when leveraging agile and iterative acquisition methodologies to deliver software-intensive, information-enabling capabilities through a continuous delivery process. The current process limits the ability to quickly adapt systems against rapidly changing threats and increases the barriers for integrating advancements in digital technology in a timely and effective manner.

Pain Points and Obstacles

Appropriation methods intended for hardware systems and platforms are not consistent with the speed and technology pace of modern software and how it is successfully acquired and deployed. DoD continues to acquire and fund information-centric systems using processes designed for hardware-centric platforms. Current funding decision processes and data structures do not effectively support leading software development practices. As a result, DoD is not effective in leveraging and adapting at the pace of innovation seen in industry. Differentiating continuous iteration and continuous delivery of software workload into hardware-defined phases (Research, Development, Test & Evaluation (RDT&E), Procurement, or Operations and Maintenance (O&M)) is meaningless in a world view where software is never done - not because planned work isn't accomplished but because modern methods allow a project to continuously improve, adapt to evolving threats, and take advantage of rapid technology advances. There should be no final fielding or sustainment element in continuous engineering. System defined program elements hinder the ability to deliver holistic capabilities and services and do not enable real-time resource, requirements, performance, and schedule trades across systems without significant work.

Establishing a culture of experimentation, adaptation and risk-taking is difficult. The Department requires a process that supports early adoption of the most modern information-centric technologies and enables continuous process and capability improvement. The Deputy Secretary of Defense directed aggressive steps "…to ensure we are employing emerging technologies to meet warfighter needs; and to increase speed and agility in technology development and procurement." The current cycle of planning, budgeting, and executing across appropriation categories slows acquisition, development, and execution to a pace that is not sustainable for mission success.

Desired state. The desired state for the Department would be one in which continuous capability deployment throughout a software program's life cycle is possible, and the lengthy two-plus year lead times for programming and budgeting is removed. This would provide flexibility to execute desired features with the speed and agility necessary to meet the rapid changes in threats, information technologies, processes, and services. The single appropriation across the life cycle of a capability will enable continuous development, security, and operations (DevSecOps); allow for minimum viable product delivery at a relevant speed; support the use of managed (or cross PoR/enterprise) services; provide for greater transparency for information-centric capabilities; and provide the flexibility to pursue the most effective solution available at the time of acquisition without current restrictions of appropriations.

Ideas for change. *A new multi-year appropriation for Digital Technology needs to be established for each Military Defense Department and the Fourth Estate.* This appropriation fund would provide a single two-year appropriation for the life-cycle management of software-intensive and infrastructure-technology capabilities. This could be a stand-alone appropriation, or fall under the umbrella of an already established appropriation, with the appropriate caveats that allow it to behave as the single source of funding across the life cycle. The Department would seek to couple this new appropriation with the movement to a capability or service portfolio management construct. A project framework within each capability PE (i.e., logistics or intelligence) would represent the systems and key investments supporting the delivery of information-centric capabilities such as data conditioning and process reengineering. Capability portfolio management would better enable agile/iterative force development and management decisions to include realignment of resources from one system to another system or process reengineering effort within the portfolio to increase the velocity of minimum viable product output and overall capability delivery. PPBE decision making would be adjusted to allow for less detail in the programming process and greater specificity in the budgeting process—as close to execution as possible—to realize the benefits of agile/iterative development.

- The Components should program, budget, and execute for information and technology capabilities from one appropriation throughout the life cycle rather than using RDT&E, procurement, or O&M appropriations, which are often applied inconsistently and inaccurately. This will allow for continuous engineering.
- Within each Component-unique Budget Activity (BA), Budget Line Items (BLINs) align by functional or operational portfolios. The BLINs may be further broken into specific projects to provide an even greater level of fidelity. These projects would represent key systems and supporting activities, such as mission engineering.
- By taking a portfolio approach for obtaining software intensive capabilities, the Components can better manage the range of requirements, balance priorities, and develop portfolio approaches to enable the transition of data to information in their own portfolios and data integration across portfolios to achieve mission effects, optimize the value of cloud technology, and leverage and transition to the concept of acquisition of whole data services vice individual systems.
- This fund will be apportioned to each of the Military Departments and OSD for Fourth Estate execution.
- Governance: management execution, performance assessment, and reporting would be aligned to the portfolio framework—BA, BLI, project.

Appendix F.3: Contracting Subgroup Report

Contributing author: Jonathan Mostowski (lead)

The contacting challenges faced by DoD today are almost entirely cultural. This premise is asserted by instances of excellence throughout the Department where effective contracting methods have been executed (DDS, DIU, Kessel Run).

That said, rather than attempting to battle each cultural challenge as they arise, it is easier to create a new modern acquisition platform from which to execute contracts that starts from a point of “how should it be done” as a product of “what should we be buying.”

The historical acquisition system was created to prevent fraud. The new priority is to establish technical superiority over our adversaries. While the prevention of fraud continues to be, and always will be, important, as a singular priority it serves to undermine the current identified need of speed and efficiency, which results in technical excellence for the Department.

Pain Points

Individual contracts are subject to review processes designed for large programs (of which they are likely enabling). This limits the agility of individual contract actions, even when modular contracting approaches are applied. In addition, the acquisition process is rigid and revolves around templates, boards, and checklists thus limiting the ability for innovation and streamlining execution.

Contracts focus on technical requirements instead of contractual process requirements. The contract should address overall scope (required capability), Period of Performance and price. The technical execution requirements should be separate and managed by the product owner or other technical lead.

Intellectual Property (IP) rights are often genetically incorporated without considering the layers of technology often applied to a solution. A single solution might include open source, proprietary software, and government custom code. The IP clauses should reflect all of the technology used.

Desired State

The desired state is an acquisition model that is liberated from the decades of policy and regulations that singularly focus on fraud prevention and provides for efficiency allowing DoD to keep pace with the private sector and adversaries. This can be accomplished through a new authority Congress establishes a separate *new* authority for contracting for software development and IT modernization.

Obstacles

- Requires act of Congress ⇒ work with Armed Service Committees Staffers
- There is no infrastructure to support this ⇒ establish policy for guidance
- There are no Contracting Officers with specific certifications ⇒ Leverage current certifications

- Could cause confusion on implementation (what applies, what doesn't) ⇒ A&S issues guidance

Ideas for Change

Congress establishes a separate *new* authority for contracting for software development and IT modernization

To address "Individual contracts being subject to review processes designed for large programs":

- Treat procurements as investments "what would you pay for a possible initial capability" (cultural).
- Manage programs at budget levels, allow programs to allocate funds at a project investment level (policy).
- Work with appropriators to establish working capital funds so that there is not pressure to spend funds quicker then you're ready (iterative contracts may produce more value with less money) (statute).
- Leverage incentives to make smaller purchases to take advantage of simplified acquisition procedures (cultural).
- Revise estimation models - source lines of code are irrelevant to future development efforts, estimations should be based on the team size, capability delivered, and investment focused (cultural).
- Allow for documentation and reporting substitutions to improve agility (agile reporting vs EVM) (cultural and EVM policy).
- Provide training to contracting officers, program managers, and leadership to understand the value and methods associated with agile and modular implementation (cultural).

To address "*Contracts focus on technical requirements instead of contractual process requirements*":

- Separate contract requirements (scope, PoP, and price) from technical requirements (backlog, roadmap, and stories) (cultural).
- Use statement of objectives (SOO) vs statement of work (SOW) to allow the vendor to solve the objectives how they are best suited (cultural).
- Use collaborative tools and libraries so that all content is available to all parties at all times (cultural).
- Use an agile process to manage structure and technical requirements (cultural).

- Establish a clear definition of done for the end of a sprint (code coverage, defect rate, user acceptance) (cultural).
- Use modular contracting to allow for regular investment decisions based on realized value (cultural).
- Streamline acquisition processes to allow for replacing poor performing contractors (cultural).
- Provide training to contracting officers, program managers, and leadership to understand the value and methods associated with agile and modular implementation (cultural).

To address "*Intellectual Property (IP) rights which are often genetically incorporated without considering the layers of technology often applied to a solution*":

- Establish clear and intuitive guidelines on how and when to apply existing clauses (cultural).
- Educate program managers and contracting officers on open source, proprietary, and government funded code (cultural).
- Have standard clause applications for each of the above that must be excepted vs accepted (cultural).

Appendix F.4: Data and Metrics Subgroup Report

Contributing authors: Ben FitzGerald (lead) and Matthias Maier

The Department of Defense (DoD) has long standing methods for capturing data, developing metrics, and reporting program progress, however these practices do more to obfuscate than provide insight when it comes to software and stand in the way of more effective methods. DoD's approach to data and metrics is fundamentally intertwined with its governance and compliance culture, which centers around reporting on individual programs to inform specific decisions by senior leaders and the Congress. Attempts to change DoD's data and metrics methods must therefore also address this culture and, critically, link with other reform efforts including policy, tools for the software development environment, and overall approaches to governance and investment.

Note: in the context of this appendix, data refers to` information associated with the development, maintenance, enhancement, and performance of software systems, not the substantive data that they process or generate.

Pain Points

Multiple, competing, and sometimes conflicting types of acquisition/management data and metrics are used for divergent purposes in the assessment of software in DoD. DoD has long standing practices to collect data on programs: primarily cost, schedule, and performance. These data are imperfect and do not necessarily reflect the health of software in any way but are important, particularly for satisfying existing reporting requirements. These data must be improved and linked with data and metrics focused on assessing the health of software activities. Doing so will potentially cause bureaucratic confusion and competition.

Challenges collecting meaningful data, in a low cost manner, at scale. To the extent that DoD currently collects data on its software activities, it does so through the manual entry of reporting data in separate and disparate reporting/management systems. This approach is prone to errors and incredibly time-consuming and burdensome to program offices. DoD components responsible for developing and maintaining the systems reporting information have few incentives to share such data, as they are often used against them, meaning that the data are hard to capture, include mistakes, and no constituency wants to invest in systems to automate data collection.

Inability to turn data into meaningful analysis and inability to implement decisions or changes to software activities. Even if DoD had clarity on its use of data and the ability to collect those data passively and at scale, it may not be able to meaningfully change the outcomes of its software activities and could become caught in a Cassandra predicament. The culture of decision making, acquisition policy, contracting, formality of requirements, appropriations rules and oversight mean that data driven insights do not naturally translate into improved decision making on DoD software activities.

Desired State

An operational system and culture that makes policy, investment, and program decisions based on insight and analysis developed in a transparent manner from standardized data collected automatically from software development tools.

Obstacles

The Department, in most but not all instances, does not possess the tools or analysts to achieve the desired state. Those are addressable challenges. The bigger obstacle is the culture of high level reporting, driven from Congress and OSD, on individual programs on a period basis, for example congressionally mandated annual Selected Acquisition Reports (SARs), and in turn, Defense Acquisition Executive Summary (DAES) reports that inform OSD quarterly of the same information. This approach means that data are not strategically collected at the level that allows for real insight and longitudinal analysis, instead they are developed at a summary level to minimally meet requirements and avoid further scrutiny. Most importantly, they do not provide the real-time tools to enable a software program manager to manage her program.

While there are few legislative barriers to implementing the desired state, Congressional action may be required to create the right incentives for DoD to generate, capture, and use data in useful ways. Congress should also address its own oversight culture, which can sometimes drive much of the behavior the Congress dislikes.

Ideas for Change

- Congress could establish, via an NDAA provision, new data-driven methods for governance of software development, maintenance, and performance.[20] The new approach should require on demand access to standard data with reviews occurring on a standard calendar, rather than the current approach of manually developed, periodic reports.

- DoD must establish the data sources, methods, and metrics required for better analysis, insight, and subsequent management of software development activities. This action does not require Congressional action but will likely stall without external intervention and may require explicit and specific Congressional requirements to strategically collect, access, and share data for analysis and decision making.

- Key steps for implementation:

    - Identification of existing definitive data sources (e.g., DAVE, FPDS[21]);

    - Establishment of robust data crosswalks to analyze data across systems and use cases;

---

[20] Congress could build on Secs 911-913 of FY2018 NDAA

[21] Defense Acquisition Visibility Environment (DAVE) https://dave.acq.osd.mil/; Federal Procurement Data System (FPDS) https://www.fpds.gov/

- Identification and mitigation of any significant gaps in existing data, with priority placed on building out functionality from existing applications where possible;
- Establishment of mechanisms to ensure data sharing and transparency (i.e., require all components to share their data);
- Disambiguation of roles and responsibilities, e.g., OSD = policy/governance ≠ program review. Components = execution;
- Linking data and metrics to governance and policy analysis and decision making.

Appendix F.5: Infrastructure Working Group Report

Contributing author: Jeff Boleng (lead)

Despite several years of effort to "move DoD to the cloud," significant friction still exists for DoD to easily leverage the required compute, storage, and bandwidth infrastructure that the commercial world so readily enjoys. The major obstacle is not at all technical, but is broadly one of accessibility: the ability to specify, contract for, pay for, connect to, secure, and continuously monitor sufficient modern computing infrastructure. Modern computing infrastructure refers primarily to cloud-based computing technologies and stacks. "Cloud-based" does not necessarily presuppose commercial cloud, but could also be on premises or hybrid cloud solutions. Similarly, "computing technologies and stacks" can run the full spectrum from infrastructure, to platform, to function, to software as a Service (IaaS, PaaS, FaaS, SaaS).

Pain Points and Obstacles

How much cloud do I need? Countless developers and IT professionals have wrestled with this question, and often the answer is to "dive in," move some apps, see what is needed, and then scale and tweak from there. The Department's culture hampers our ability to even take a "leap of faith" like this. We must be able to precisely size and cost our cloud requirements before ever starting to experiment or prototype. It should become more clear why this analysis paralysis exists as the below pain points are outlined and considered.

How do I buy cloud? Oh, just head on over to FedRAMP, pick an approved provider, sign up and you're on your way… FedRAMP? Is that a cloud? What about GovCloud, cloud.gov (not the same thing by the way), and MilCloud (is that version 1.0 or 2.0?)? What's the difference between AWS GovCloud and Azure Government? Can I just sign up with a credit card like a normal private citizen and start hosting my compute and data in the cloud? Sadly, the answer is a definitive and resounding NO! Even if you know which "government-approved" cloud you're moving to, it's just not easy to contract for it or buy it.

There is not space here to answer all these rhetorical questions. For a good description of the difficulty of buying cloud, please refer to the DoD Cloud Acquisition Guidebook at https://www.dau.mil/tools/t/DoD-Cloud-Acquisition-Guidebook. Here the Defense Acquisition University (DAU) outlines the multiple activities that need to be accomplished to contract for cloud services. Starting with the dreaded IT Business Case Analysis (BCA), moving on to applying the DoD Cloud Security Requirements Guide (SRG - more on this soon), to getting an Authorization to Operate (ATO), ensuring DISA approves of your Boundary Cloud Access Point (BCAP) and your Cyber Security Service Provider (CCSSP), and lastly to applying the DFARS supplementary rule to your cloud contact. No friction here right?

How do I know my cloud is secure? Easy. FedRAMP pre-evaluates and approves Cloud Service Providers (CSSPs) for Information Impact Levels (IILs) 2, 4, 5, and 6 (don't ask about levels 1 and 3; apparently we over specified and they aren't necessary any longer). Whew, now things are making sense… Not so fast, the FedRAMP IILs are for US Government cloud use, but not DoD![22]

[22] Don't ask… we know DoD is part of the US Government.

We need FedRAMP+ for DoD use, and DISA doesn't evaluate Cloud Service Providers (CSPs), only Cloud Service Offerings (CSOs). Huh? Be sure to go through the DoD Cloud Computing SRG, ensure those extra security controls are in place for FedRAMP+, and you're on your way. Again, not so fast Program Manager (or small business owner)! How are you and your customers going to access the fancy new cloud you just finally got on contract?

How do I access my cloud? The cloud, sort of by definition, implies ease of access, right? The National Institute of Standards and Technology (NIST) definition in SP 800-145 defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." Well, if you're a DoD user, you need to ensure you've got a BCAP in place between your application/service and your users. It's OK and accurate to immediately envision bottleneck and single point of failure here.[23] Mis-configuring and under-provisioning BCAPs is the norm rather than the exception, so even with all that compute and storage in the cloud that you somehow ran the contracting gauntlet to get, you're going to severely lack adequate bandwidth and likely suffer from significant latency. Friction++.

How do I pay for cloud? The best part of cloud computing is that I can only pay for what I use. A true consumption-based cost model. Just like a utility. Not so for Government and DoD though. The Anti-Deficiency Act doesn't allow us to pay for cloud computing like a utility. A common way around this is to pay a third party contractor to buy the cloud service for us. This results in a situation where we estimate the highest charges we could ever incur in a year, add a bit of padding to that (say 20-30%), pay the third party, and we've paid for our cloud. What happens if we don't use it all up by the end of the year? Nothing (i.e., no refunds). Money spent. The third party contractor makes (quite?) a bit of extra profit for "taking the risk off the government." So much for consumption-based payments.

Desired State

The ability to provision, pay for, consume, access, and monitor cloud computing (compute, storage, and bandwidth) the same way any commercial organization does. It is understood that there are unique DoD security requirements, but that should only affect cloud pricing (say 1.5 to 2 times commercial, worst case), and not any of the other procedures to easily access cloud computing technologies and resources.

Obstacles

Significant obstacles remain to easily leverage commercially equivalent compute, storage, and bandwidth infrastructure. Contracting, security procedures (not necessarily requirements), network access (i.e., a modern technological approach to BCAP), and billing all loom large. The most important of these is the DoD's inability to contract and pay for cloud computing on a consumption basis.

---

[23] There are better ways to do this, like zero trust networks. The commercial world has some really good examples and architectures that don't require this man-in-the-middle attack called a BCAP which actually breaks end-to-end encryption by design…

Ideas for Change

Establish a DoD enterprise ability to procure, provision, pay for, and use cloud that is no different from the commercial entry points for cloud computing. The Joint Enterprise Defense Infrastructure (JEDI) Cloud initiative is a bold attempt at this solution and should be awarded. Cloud.gov (which is ironically hosted in GovCloud) is another promising program that is already very straightforward to provision and buy, but is limited to IIL 2 data and applications. The objective cloud procurement and billing contract must include the ability to truly pay for consumption of cloud services and not be artificially limited by the Anti-Deficiency Act. Modern software demands the ability to consume and pay for cloud services just as we do any other utility.

In addition to this, DoD should establish a common, enterprise ability to develop software solutions in the “easy-to-acquire-and-provision” cloud that is fully accredited by design of the process, tools, and pipeline. Said another way, DoD should stop the security accreditation of individual applications, but *should instead invest in accrediting the ability to produce software*. The pipeline, automated tooling, procedures, and operational monitoring and auditing of software should be the focus and target of security accreditation, not each individual application and version of an operating system or application.

Another essential and necessary, though not sufficient, change that must occur is to adopt modern commercial approaches to software and system security in the cloud that does NOT involve BCAPs, Internet Access (choke) Points (IAPs), or CSSPs that cannot be performed entirely by trusted commercial entities. DoD must adopt modern cloud security approaches such as zero trust networks[24], micro-segmentation, and eliminate the perimeter approach to network security and trust that is based on assigned IP address or network connection point. Perimeter-based security cannot scale to accommodate the bandwidth, traffic, and latency demands of modern cloud access, applications, and services. Furthermore, it is a failed architectural practice that has proven to be readily exploitable by adversaries and is especially vulnerable to insider threats.

[24] https://www.oreilly.com/library/view/zero-trust-networks/9781491962183/ch01.html

Appendix F.6: Requirements Subgroup Report

Contributing authors: Fred Gregory (lead), Philomena Zimmerman, Jeff Boleng, Margaret Palmieri, Jennifer Edgin, Owen Seely, Victoria Cuff, and Donald Johnson

The Department of Defense (DoD) in 2003 institutionalized the identification and validation of requirements via the Joint Capability Integration and Development System (JCIDS). Created to support the statutory responsibility of the Joint Requirements Oversight Council (JROC), it is one of three processes (Acquisition, Requirements, and Funding) that support the Defense Acquisition System (DAS). Considered revolutionary in its design, moving DoD from a threat-based to a capability-based model, it has begun to show its age in today's era of software-intensive systems intending to leverage agile software practices. These evolving agile practices upend traditional industrial-age process attempts to credibly and accurately predict a future 15-20 years away, necessitating unimaginable precision and foresight upfront in support to capability development. The requirement process, writ large, must adapt to support delivering capabilities at the speed of relevance; processes, cultures, and expectations of the Service and Joint Force requirement communities.

Pain Points

A byproduct of top-level requirement flow down is rigidity and over specificity at the derived requirements level, that greatly hinders agile software design. Capability validated by the JROC does not proscribe requirement allocation to either hardware or software solutions. However, the resulting flowdown of derived requirements incorporated into the source selection/contract award and the subsequent allocation of these between hardware and software by the prime can ultimately discourage software design flexibility. The decisions, often made years before software coding even begins, locks the prime and the government into a proscribed path that often does not produce the desired warfighter capability within the needed time frame. Preserving software design flexibility must be a key component throughout the requirements validation process. "Requirers" will need to learn to settle for "less" not "more" at capability need inception.

Too often exquisite requirements, intended to be 100 percent correct, are levied on a system that in turn drives extensive complex software requirements and design, affecting development, integration, and system test. Today's requirements process more closely mimics the "big-bang" theory often vilified by industry, government, and Congress. As the warfighting community loses faith in the acquisition community's ability to meet their commitments through timely incremental improvements, the temptation to "gold-plate" a requirement becomes more prevalent. Likewise, as the acquisition community is forced to defend shifting warfighter priorities in budget deliberations and Congressional engagements, the temptation to "lock requirements down early" permeates acquisition strategies. With both of these choices in play, exquisite requirements must be described perfectly at capability inception in order to maintain a low-risk acquisition program - obviously an impossible outcome.

Data sets are siloed within programs - a common Law of Requirements is that programs of record (PoR) try to avoid dependencies with other PoRs. By tying SW to a PoR, it becomes *nearly impossible to transfer that code across systems and data environments*. Data “lakes,” “pools,” and “ponds” will be the foundation for future weapon system data repositories, and the requirements process must be flexible enough to accommodate this new archetype. Breaking from the past mold of tying software code to a program of record and a specific data environment frees the program manager from the arduous task of integrating seams across multiple PORs.

Example. The Navy operates forward at sea and on-shore at maritime operations centers (MOCs). Command and control between sea and shore is a key aspect of how they fight—they need shared battlespace awareness at aligned actions across distributed units at best. However, the systems afloat and ashore are not always the same because ships need systems that are hardened for combat at sea. If a new algorithm can help manage supply and logistics on the cloud ashore, it may not run the same at sea because different system exists afloat. Extrapolating across Services, the USAF writes an algorithm to optimize F-16 maintenance, however it is highly unlikely that the Navy can pick it up and apply it to F-18s. This depends on the vertical integration of the algorithm, data, and system (PoR).

Desired state. Go from Sailor (Airman, Rifleman, etc)-stated need to software delivery in their hands within days to support future conflicts. This necessitates a process for concept/requirements determination/setting that takes advantage of the agility in software development and software products to increase the agility and modifiability in our systems. Requirements flow down must also maintain a broad-based approach into the lowest levels of design. We also note that one of the overarching agile principles is that “increments are small.” Fast requirements, fast deployments and fast test cycles for usefulness are tough to accomplish with huge, monolithic software projects. Start small, stay small! Finally, recognizing that documenting and contracting for a moving target is not easy but must be done.

Obstacles. Breaking the tyranny of siloed PoRs will require a concerted effort across the Department, Combat Support Agencies, and will require Congressional engagement and support. Considerable cultural barriers must also be overcome as the algorithms themselves become capability, and the methods used to document, validate, and maintain currency enter the mainstream. Complexity and dependencies among multiple elements prevent widespread usage of Family-of-Systems (FoS) and System-of-Systems (SoS) requirement documents. Government requirements and acquisition communities take on extra oversight burden when they take a FoS or SoS approach because they have to manage all the pieces coming together effectively. Lastly, current statutory guidance does not promote, encourage, or reward the use of agile software development practices or environments.

Ideas for Change

- The Joint Staff should consider revising JCIDS guidance to separate functionality that needs high variability from the functionality that deemed “more stable” (e.g., types of signals to analyze vs. allowable space for the antenna). Then implement a “software box” approach for each, one in which the contours of the box are shaped by the functionality variability

- OSD should consider identifying automated software generation areas that can apply to specific domains
- The Joint Staff should consider revising JCIDS guidance to document stable concepts, not speculative ideas.
    - Specifying needed capabilities is important up front, however it must be acknowledged that initial software requirements need to be "just barely good enough" for the situation at hand or, in other words, "document late"
    - Acknowledge that software requirement documents will iterate, iterate, iterate. JCIDS must change from a "one-pass" mentality to a "first of many" model that is inherently agile delegating approval to the lowest possible level
- DoD should consider instituting a distributed model-based approach to requirements development extended across the enterprise
    - The model should be used to develop result-based metrics for requirement evaluation
- The Joint Staff should consider revising JCIDS guidance to focus on user needs, bypassing the JCIDS process as needed to facilitate rapid software development. Guidance should specifically account for user communities (e.g., Tactical Action Officer (TAO), Maritime Operations Center (MOC) director) that do not have one specific PoR assigned to them, but use multiple systems and data from those systems to be effective
- OSD and the Joint Staff should consider creating "umbrella" software programs around "roles" (e.g., USAF Kessel Run)

Appendix F.7: Security Accreditation/Certification Subgroup Report

Contributing authors: Leo Garciga (lead), Tom Morton, and Ana Kreiensieck

The Department's current Security Certification and Accreditation (C&A) process is a complicated and time-consuming process that is measured in months and years. The process is typically seen as a serial process that occurs after development with a checklist mentality. While this fits with a waterfall approach to development, the Department is changing to an agile, DevSecOps approach. The overall security paradigm must change from one where updates to software happen optimistically on a yearly basis to one where software is updated weekly or daily in response to emerging threats and this is recognized as more secure than the slow, static process. Additionally, we must strive to accredit the process, tools, and platforms to allow and enable Continuous ATO when software changes meet the required thresholds.

Pain Points

Complex, time-consuming, and misapplied process. Although developing and operating software securely is a primary concern, the means to achieve and demonstrate security is overly complex and hampered by inconsistent and outdated/misapplied policy and implementation practices (e.g., overlaying historical DoD Information Assurance Certification and Accreditation Process (DIACAP) process over Risk Management Framework (RMF) controls for individual pieces of software versus system accreditation). The sense is that the Certification and Accreditation (C&A) process is primarily a "check-the-box" documentary process, adds little value to the overall security of the system, and is likely to overlook flaws in the design, implementation, and the environment in which the software operates.

No way to calculate total costs of C&A process. The Department needs to be able to calculate the true and component costs for implementing the RMF and C&A in order to identify inefficiencies, duplicative capabilities, and redundant or overlapping security products and services that are being acquired or developed. Absent a set of metrics it is difficult to prioritize risk areas, investments, and evaluating risk reduction and return on investment.

Lack of top-down security requirements. The Department has not decomposed security requirements from an enterprise level to a mission level to a functional implementation level. Programs waste resources implementing security controls that should be inherited.

Lack of automation. The C&A process is predominantly a manual process which makes it a very low process. Programs must plan in terms of months and years to get a product through the security accreditation process. This slow process does not provide the warfighter the timely, modern solutions that are needed.

Desired State

Accredit the process, not the product. Done correctly, security is applied from the beginning of software development using automated tools. Before transitioning into operations, an Authorizing Official (AO) reviews the process under which the software was developed and accepts the risk as determined from various scans and tests. The AO signs a Continuous ATO so that as long as the process remains intact and is continuously operationally monitored, the subsequent software releases are accredited.

Obstacles

Two primary obstacles are culture change and workforce skills. The current security culture is that security is a checkbox activity at the end of the development process. As RMF is implemented, this is beginning to change the culture of security from compliance to continuous risk assessment. However, the process is still very manual. The culture change needs to include using automation to speed up risk assessment and continuous risk monitoring of operational software.

The other obstacle is the security and accreditation workforce skill set. While tools can provide reports and speed up security activities like scans and code analysis, it takes a particular skill set to understand those inputs and recommend or make at-risk decisions. The current security workforce must be trained in these new skills.

Ideas for Change

Embrace DevSecOps. The Department should embrace DevSecOps (not just DevOps) and provide the necessary resources to develop the common software components and automation to assemble, test, accredit, and operate software systems. DevSecOps also includes policy-supported processes, certified libraries, tools, and an operational platform (with appropriately instrumented run-time software), and a toolchain reference to implementation to produce “born secure” software.

Automate, Automate, Automate! The Department needs to provide automated tools and services needed to integrated continuous monitoring with the development life cycle, enable continuous assessment and accreditation, and delegate decision making at the lowest level possible. Examples of automation are using static code analysis during the “build” stage, running automated unit tests, functional test, regression tests, integration tests, and resiliency/performance tests during the “test” stage, using dynamic code analysis, fuzzing scans, running container security scans, STIG compliance scans, and 508 compliance scans during the “secure” stage, and running continuous monitoring tools and ensuring logs are being pushed to the appropriate entity during the “monitoring” and “operational” stages.

Define top-down implementation requirements. The Department needs to ensure that each Joint Capability Area (JCA) flows-down its strategy, best practices, and implementation requirements/guidance for security and accreditation to allow the Component responsible for implementing the software to appropriately tailor RMF and plan the development, accreditation, and operation of the software. Furthermore, each JCA should endeavor to clearly state its risk profile and tolerance so that the RMF can be applied effectively and appropriately mitigate identified risks.

Education is necessary at all levels. As security is “baked in” to software during the development process, people must be educated about what that means as different tools look at different security aspects. They must also be educated in what it means to bring different security reports together and make a risk decision, both during development, and continuously during operations.

Culturally, people must learn to appreciate that speed helps increase security. Security is improved when changes and updates can be made quickly to an application. Using automation, software can be reviewed and updated quickly. The AO must also be able to review documentation and make a risk decision quickly and make that decision on the process and not the product and document it in a Continuous ATO.

Appendix F.8: Sustainment and Modernization Subgroup Report

Contributing authors: Kenneth Watson (lead), Stephen Michaluk, and Bernard Reger. Additional advice / assistance from SEI

Improving the materiel readiness of our fielded weapon systems and equipment is an imperative across the Department in accordance with the new National Defense Strategy.[25] The time is now to shift from our traditional, hardware-centric focus and identify what core[26] means for software intensive weapon systems and associated software engineering capabilities. Software is a foundational building material for the engineering of systems, enabling almost 100 percent of the integrated functionality of cyber-physical systems, especially mission- and safety-critical software-reliant systems. More simply, these systems cannot function without software.

For fielded weapon systems and military equipment, software life-cycle activities follow somewhat predictable cycles of corrective, perfective, adaptive, and preventative modifications while major modifications drive new periods of development. Software development activities, even those following agile methods, encounter a phase where the program transitions from adding new features to supporting and sustaining day-to-day use and operations. At that point, development changes and signals a move to "sustainers" within the organic industrial base. Therefore, sustainment may be defined as the sum of all actions and activities necessary to support a weapon system or military equipment after it has been fielded.

Prioritizing the transition to software sustainment during requirements and engineering development is critical to timely, effective, and affordable sustainment, regardless of how software engineering organizations are structured and resourced. Software sustainment organizations must be engaged and embedded at the earliest design stages to ensure we can keep pace with new capabilities as systems become operational. Lastly, access to software source code, emphasizing an early focus on designing for sustainment, and investment into establishing and modernizing system integration laboratories, are just a few of the challenges faced by the DoD software enterprise.

Pain Points

Applying a hardware maintenance mindset to software hinders DoD's ability to better leverage the organic software engineering infrastructure. DoD maintenance policies and maintenance-related Congressional statutes have traditionally been optimized for hardware and are difficult to change due to long standing policies, practices, inertia, and incentives. The goal of hardware maintenance is to repair and restore form, fit, and function. This mindset does not align well with the ever evolving nature of software. The scope of software engineering for sustainment mitigates defects and vulnerabilities, fact-of-life interface changes, and add new enhancements. Software is never done and any time it is "touched," it triggers the software engineering development life

---

[25] "Summary of the 2018 National Defense Strategy" (Washington, DC: Department of Defense, 2018), https://dod.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf.

[26] As defined in 10 USC 2464, *Core logistics capabilities.*

cycle which produces a new configuration. Therefore, any system that is dependent on software to remain operational, is always in a state of continuous engineering during sustainment (or O&S phase of the life cycle).

DoD's acquisition process is not emphasizing an upfront focus on design for software sustainment and a seamless transition to organic sustainment. It is critical that software be designed to be more affordably sustained with high assurance and the ability to integrate changes and enhancements more rapidly to provide a continual operational capability to the warfighter. Moreover, software must be decoupled from hardware to the greatest extent possible in order to enable leveraging rapid and continuous hardware improvements. We need to place increased emphasis in acquisition on designing in software sustainability with a consistent emphasis on how DoD contracts for software as well as the span of requirements, architecture, design, development, and test. Additionally, this includes making provisions for timely access to the necessary range of software technical data to enable timely and effective organic software engineering and rapid re-hosting. It is essential that DoD and industry work collaboratively to meet the increasing software sustainment demand.

Public Private Partnerships (PPPs) provide one means to leverage DoD and industry capabilities as a team to deliver warfighter capability. However, PPPs and other options are not being considered up front and leveraged across DoD as an inherent element of the acquisition and engineering strategy of programs. This team strategy may facilitate mutual access to the technical data inherent in executing the software development life cycle.

Limited visibility of DoD organic software engineering infrastructure, capabilities, workload, and resources. Title 10 USC 2464 establishes a key imperative for DoD to establish core Government Owned Government Operated (GOGO) capabilities as a ready and controlled source of technical competence and resources for national security. DoD's focus has traditionally been on hardware and therefore there has seen significant Service and DoD enterprise focus on hardware GOGO capabilities and infrastructure for core. However, there has been significantly less upfront acquisition focus and visibility on what core means for software intensive systems and the associated GOGO software engineering capability. For the traditional DoD hardware-centric model, core capability is based on individual weapon systems or platforms at the depot level. All systems operate interdependently in a net-centric environment, where force structure and execution of mission capabilities are products of a system-of-systems capability. In a software intensive environment "Go to War" analysis of what core means as it relates to software requires more strategic thinking about core than just focusing on individual weapon systems or platforms (aircraft, ship, tank, etc.) as hardware. The hardware-centric focus on weapon systems likely underestimates the scope and magnitude of what should be considered a core requirement in a software intensive systems operational environment.

Desired State. Require government integrated software sustainment participation from the very beginning of development activities.

Ideas for Change

- Title 10 USC 2460 should be revised to replace the term software maintenance with the term software sustainment and a definition that is consistent with a continuous engineering approach across the life cycle.
- DoD should establish a capability for visibility into the size and composition of DoD's software sustainment portfolio, demographics, and infrastructure to better inform enterprise investment and program decisions.
- A DoD working group should be established to leverage on-going individual Service efforts and create a DoD contracting and acquisition guide for software and software sustainment patterned after the approach that led to the creation of the DoD Open Systems Architecture Contracting Guide.
- Acquisition Strategy, RFP/Evaluation Criteria, and Systems Engineering Plan should address software sustainability, re-hosting, and transition to sustainment as an acquisition priority. The engineering strategy and plan should engage software sustainment engineers upfront and co-locates government software sustainment engineers on the contractor software development teams to enable effectively and timely transition to an organic sustainment capability.
- The definition of "core capabilities" in 10 USC 2464 should be revisited in light of warfighter dependence on software intensive systems to determine the scope of DoD's core organic software engineering capability, and we should engage with Congress on the proposed revision to clarify the intent and extent of key terminology used in the current statute.
- DoD should revise industrial base policy to include software and DoD's organic software engineering capabilities and infrastructure. Start enterprise planning and investment to establish and modernize organic System Integration Labs (SILs), software engineering environments, and technical infrastructure; invest in R&D to advance organic software engineering infrastructure capabilities.

Appendix F.9: Test and Evaluation Subgroup Report

Contributing authors: Amy Henninger and Greg Zacharias

The fundamental purpose of DoD test and evaluation (T&E) is to provide knowledge that helps decision makers manage the risk involved in developing, producing, operating, and sustaining systems and capabilities. While colloquially referred to as a single construct, T&E is composed of two distinct functions: obtaining the data and assessing the data. This distinction is important because the T&E community will report "pain points" in both functions. There are also two major types of test: Developmental Test (DT) and Operational Test (OT). DT, by nature, is "experimental," performed on behalf of the Program Management Office (PMO), supporting a formative evaluation and identifying design elements that will drive mission-critical capability to inform the evolution of component and system design. OT is "evaluative," performed by and on behalf of the warfighter, supporting a summative evaluation of system capabilities to support warfighting missions across the operational envelope.

Because T&E has historically occurred toward the end of, often, a long and costly acquisition process (e.g., requirements, design, and development), it can be perceived as simply adding time and cost to an already late and over-budget effort; PMOs therefore can view this "last step" T&E as simply making the situation worse. And if T&E finds a system substantially defective, necessitating expensive re-engineering of the design late in developing, it adds to the perception that T&E simply adds cost and time to project execution. A continuous iterative T&E model is clearly called for, occurring alongside design and development, where T&E can both; catch defects early so they can be solved quickly and cheaply and inform/shape system requirements based on early feedback from the warfighter. Experience shows that active, early involvement by independent testers—combined with a PMO who responds to the independent testers' advice—makes a positive difference to program outcomes. We have seen this in modern iterative approaches, such as agile development, applied effectively in DoD, especially in Major Automated Information Systems (MAIS).[27] Taken together, these observations point to the need to move away from what can be a linear waterfall process segregated by siloes, to a more iterative and collaborative model that fuses all development, test, processes, tools, and information to enable the continuous delivery of tested capability. T&E can then be viewed as saving time/cost in development, instead of adding time/cost.

Pain Points and Obstacles

DoD lacks the enterprise digital infrastructure needed to test the broad spectrum of software types and across the span of T&E to support developmental efficiency (in DT) and operational effectiveness (in OT). Digital models of test articles (e.g., "Digital Twins") are not always available and not built to common standards. T&E environments, including threat surrogates or models, are often program-focused and funded, with short-term development goals and narrowly-scoped capabilities defined by the program. Building (and re-building) representative T&E environments is time and cost prohibitive for individual programs and results in duplicative infrastructure investments across DoD. Moreover, current T&E practices in the Services, including those

[27] FY16 DOT&E Annual Report.

focused on software-intensive systems, do not adequately test systems in Joint and Coalition environments, nor do they consistently use appropriate risk-based, mission-focused testing.

DoD lacks the enterprise data management and analytics capability needed to support the evaluation of test data in accordance with the pace of modern iterative software methods. As data required to make informed acquisition decisions continues to grow due to higher resolution measurements, higher acquisition rates, and other additional requirements for software intensive systems (e.g., interdependency, need to operate in system-of-systems, family-of-systems, Joint, and Coalition environments), the need for a T&E infrastructure to collect, aggregate, and analyze this data must likewise evolve to keep pace. More timely data fusion will require improvements in data management techniques, access speeds, data access policies, data verification techniques, and the availability of more intelligent and agile tools. Without this infrastructure, and within the current paradigm, we are failing to adequately gather and analyze these highly diverse and complex datasets, which leads to invalid assessments of acquisition program progress and system performance, undercuts mission readiness, and places warfighters at risk. This gap becomes an even more prominent choke point in an iterative cycle. Thus, even if we mitigate the first pain point with modernized realistic test environments, and had the capability to collect the appropriate mix/quantity of data in testing, we would still not have the analytics horsepower to turn around an assessment to support the pace of an Agile/DevSecOps iterative cycle.

DoD lacks the resources needed to adequately emulate advanced cyber adversaries, to support fielding of trusted, survivable, and resilient software-intensive defense systems. Various oversight entities (e.g., NDAAs and GAO Reports) have acknowledged this gap, and past DOT&E Annual Reports have documented a significant number of adverse cyber findings in OT that should not require an operational environment to discover. While the gap exists now (in the absence of modern software methods), it will become an even more prominent choke point in a rapid development and operational fielding paradigm. We do not have the advanced cyber test resources (manpower, methods, and environment) to support a true Agile/DevSecOps approach to developing, testing, and fielding the broad range of software-intensive systems needed by DoD now and in the future, in an environment increasingly populated by advanced cyber adversaries.

DoD lacks a modern software intellectual property (IP) strategy to support T&E in a rapid software development and fielding environment. Overcoming this pain point is critical to overcoming all of the three previously described pain points. Specifically, none of the previously described pain points is fully achievable without sufficient access to necessary technical data associated with the software deliverables. Software acquisition processes are and will continue to be suboptimal (with respect to time and risk) without access to relevant technical data and this gap will become an even more prominent choke point in an Agile/DevSecOps-based paradigm without that access. A modern software IP strategy must include access to software environments (e.g., source code, build tools, test scripts, and cybersecurity artifacts/risk assessments) so tests are repeatable, extendable, and reusable. This strategy will also have to strike a balance with the IP rights of the innovator (usually industry) to ensure continued engagement of DoD with leading-edge technology organizations.

A modern software IP strategy would support the three previously described pain points via:

- Enhance our ability to operationalize the concept of "digital twins," with sufficient access to the source code of a given system (balancing DoD and innovator IP rights), so as to be able to adequately represent that system.
- Support the instrumentation of software-intensive systems as needed during testing.
- Support cyber vulnerability assessments and the assignment of risks to residual vulnerabilities, via access to system data (e.g., code and technical data).

Desired State

While DoD does a fair amount of "integrated testing" now (across DT and OT), that is not the same as "integrating T&E with the Voice of the End User continuously and alongside software development."[28] T&E must strive for continuous software testing, automated and integrated into the development cycle to the fullest extent possible, across the entirety of DoD's software portfolio. The qualifier, "fullest extent possible" is important, as many experts have acknowledged that no single "one size fits all" approach will work best across the entire DoD software portfolio all of the time.[29,30] In this envisioned state, independent testers would work alongside developers and operators to help software development programs succeed and deliver capability at the speed of need. T&E would no longer be perceived as "slowing things down" or "costing money post-development" because it occurs toward the end of a highly linear and inefficient process, but would instead be associated with saving time and money during development. This vision, applied across the entire DoD software portfolio (i.e., beyond just IT or MAIS) requires the right kinds of tools, architectures and standards (see first three pain points), access to the right kind of data (see second and fourth pain points), and an ability to partner with and work alongside the developer, while yet maintaining independence and objectivity in our assessments.

Ideas for Change

Build the enterprise-level digital infrastructure needed to streamline software development and testing across the full DoD software portfolio. Beyond the DevSecOps platform (or Digital Technology concept), DoD requires a digital *engineering* infrastructure to streamline integration and testing. This suggests that the DevSecOps platform must be made available to all DoD software developers and:

- Integrated with (systems-level) model-based/digital engineering infrastructure, including digital twin(s),
- Integrated with existing T&E infrastructure (e.g., open-air ranges, labs, and other test facilities),
- Integrated with comprehensive tactical/mission-level infrastructure, and
- Available to others who could benefit (e.g., analysis, training, and planning).

---

[28] Steven Hutchison, "Test and Evaluation for Agile Information Technologies," *ITEA Journal* 31(2010): 461.
[29] 2018 Defense Science Board Task Force on Design and Acquisition of Software for Defense Systems.
[30] Boehm and Turner, 2009. Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley. Boston, MA.

Even with this kind of complete testing infrastructure providing the capability to collect the appropriate mix/quantity of data in testing, we would still not have the analytics horsepower to turn around an assessment sufficiently rapidly to support the pace of an Agile/DevSecOps iterative cycle. We must develop the enterprise knowledge management and data analytics capability for rapid analysis/presentation of technical data to support deployment decisions at each iterative cycle.

Finally, to advance our cyber test resources such that we can achieve overmatch to our most capable adversaries while yet supporting the pace of the modern software development, DoD should expand DOT&E's current capability to obtain state-of-the-art cyber capabilities on a fee-for-service basis. This provides a straightforward way to acquire skilled cyber personnel from leading institutions (e.g., academia, university affiliated or federally funded research and development centers), to help the DoD to keep pace with advanced cyber adversaries.

Appendix F.10: Workforce Subgroup Report

Contributing authors: Maj Justin Ellsworth (lead), Sean Brady, and Kevin Carter

DoD's workforce (civilian, military, and supporting contractor personnel) is our most valuable resource. The workforce's capacity to apply modern technology and software practices to meet the mission is the only way we can remain relevant in increasingly technical fighting domains, especially against our sophisticated peers, Russia and China.

Improved management of the Department's software acquisition talent will also drive success across the other subgroups and sections of this report. Policies, processes, and bureaucratic practices are never a sufficient substitute for competence.

The Department's challenges are well documented and well known by the software acquisition and engineering professionals who suffer most from the accrued technology, cultural, and leadership debt. The Workforce Subgroup identified prevalent pain points, but focused on providing concrete and actionable solutions for improving the recruitment, retention, development, and engagement of the workforce.

Pain Points

The Department's reputation as an employer is a weakness rather than a strength. Candidates base their employment decision on a variety of factors, but the organization's reputation and day-to-day work are chief among their considerations. The demand, and competition with the private sector, for an experienced and qualified workforce, is increasing as threats to our data security become more sophisticated. DoD has a reputation as an antiquated employer that rewards time in grade rather than competence and most often outsources its technical execution. Technical employees often serve as oversight or move away from "hands-on-keyboard" as they advance in their careers; no longer contributing to creative or innovative execution.

The Department does not adequately understand which competencies and skill sets are possessed and needed within its software acquisition and engineering workforce. Without the ability to distinguish the workforce, DoD cannot effectively drive human capital initiatives. Furthermore, there is no enterprise-wide talent management system to manage the workforce (e.g., geographically or by skills), which leads to bureaucratic silos and the inability to leverage the Total Force.

The Department has not prioritized a comprehensive recruiting strategy or campaign targeting civilians (90 percent of the acquisition workforce) for technical positions. When candidates do apply, they face an "overly complex and lengthy hiring process (that) frequently results in the Government losing potential employees to private sector organizations with more streamlined hiring processes," according to the President's Management Agenda.[31]

---

[31] "President's Management Agenda: Modernizing Government for the 21st Century," (Washington, DC: Office of Management and Budget, April 2018), 20, https://www.whitehouse.gov/omb/management/pma/.

There is no comprehensive training or development program that prepares the software acquisition and technical workforce to adequately deploy modern development tools and methodologies within our dynamic environments. Hiring top technical talent into the Department will never be a silver bullet. The Department also needs to consider how to equip, reward, promote, and empower its existing workforce.

The Department is unable to leverage modern tools that are common in the private sector and our personal lives (e.g., cloud storage and collaborative software) due to bureaucratic barriers. Top talent expects access to these tools to meet mission demands, and their absence may discourage qualified candidates from applying or staying. Although the Department has pockets of innovation and entrepreneurship within rapid fielding offices across the Services, this culture has not scaled to the larger acquisition programs and offices. Long-cycle times, bureaucratic silos, and information-hoarding prevail.

Desired State

The Department requires a workforce capable of acquiring, building, and delivering software and technology in real time, as threats and demands emerge. This workforce should resemble successful technology companies that must move quickly to meet market challenges. They do so by promoting an agile culture, celebrating innovation, learning from calculated failures, and valuing people over process.

The Department's workforce embraced commercial best practices for the rapid recruitment of talented professionals. Once onboarded quickly, they will use modern tools and continuously learn in state-of-the-art training environments, bringing in the best from industry and academia, while pursuing private-public exchange programs to broaden their skill sets.

Obstacles

The bureaucratic culture of the Department creates significant barriers compared to a commercial sector ecosystem that moves at the speed of relevance. These barriers are now ingrained within the institution, perpetuating a risk-averse environment that represents the most significant obstacle to reform. While there are minor legislative solutions to achieving the desired state, we believe that the Department has the necessary authorities and flexibilities, but has shown lack of impetus to move to the modern era of talent management.

While small pockets of expertise and progress exist, the Department as a whole lacks sufficient understanding of current software development practices and talent management models that support them. Studies on the workforce dating back 35 years that show "limited evidence these different efforts had any lasting impact or resulted in meaningful outcomes."[32]

---

[32] McLendon, Michael H.; Shull, Forrest; Miller, Christopher, "DoD's Software Sustainment Ecosystem: Needed Skill Sets," (Naval Postgraduate School, Monterey, California, April 30, 2018).

Ideas for Change

Foundational. Taking into account history and the significant challenges with changing the culture in a bureaucracy, the Department should *empower a small cadre of Highly Qualified Experts and innovative Department employees to execute changes* from this report. This cadre is empowered with the authority to create, eliminate, and change policies within the Department for organizations beyond themselves. If needed, create a software acquisition workforce fund similar to the existing Defense Acquisition Workforce Development Fund (DAWDF). As called out by the Defense Science Board, the purpose of this fund will be to hire and train a cadre of modern software acquisition experts. This fund should also be used to provide Agile, Tech, and DevSecOps coaches in Program Offices to support transformations, adoption of modern software practice and sharing lessons across the enterprise.[33]

Workforce Foundations. The Department must develop a core occupational series based on current core competencies and skills for software acquisition and engineering. This occupational series should encompass all workforce roles required for modern software development and acquisition - engineers, designers, product managers, etc. Additionally, the Department should create a unique identifier or endorsement of qualified (experience & training) individuals who are capable of serving on an acquisition for software. This includes the development of a modern talent marketplace (and associated knowledge and skill tags/badges) to track these individuals. The competencies for this series should be flexible enough to evolve alongside technology, something that has constrained the 2110 IT Series.

Contractor Reforms. Defense contractors develop the majority of software in the Department. The Department should incentivize defense contractors that demonstrate modern software methodologies; this may take the form of software factory demonstrations and rapid software delivery challenges when evaluating proposals. Additional consideration should be given to contractors with demonstrated excellence creating commercially successful software.

Recruitment and Hiring. The Department must overhaul its recruiting and hiring process to use simple position titles and descriptions, educate hiring managers to leverage all hiring authorities, engage subject-matter experts as reviewers, and streamline the onboarding process to take weeks instead of months. The Department needs to embrace private-sector hiring methods to attract and onboard top talent from non-traditional backgrounds (e.g., hackers and entrepreneurs). Too often, these types of candidates are passed over or require special authorities to join the Department, due to lack of education or regular pay stubs. Furthermore, the Department must develop a strategic recruitment program that targets civilians, similar to its recruitment strategy for military members. This includes prioritizing experience and skills over cookie-cutter commercial certifications or educational credentials.

Development, Advancement, Engagement, and Retention. The Department must pilot development programs that provide comprehensive training for all software acquisition professionals, developers, and associated functions. Programs should be built in partnership with

---

[33] Design and Acquisition of Software for Defense Systems," Defense Science Board, Feb. 2018, https://www.acq.osd.mil/dsb/reports.htm.

academia and industry, leveraging commercial training solutions rather than custom and expensive Federal solutions. This will include continuing education courses to help the workforce stay current and ensure technical literacy across the acquisition workforce. The Department must emphasize promoting and rewarding those that have proven both commitment and technical competence. Continually looking outside the Department is demoralizing and insulting to existing professionals that demonstrate innovation, excellence, and the ability to deliver already. The Department should incentivize and provide software practitioners access to modern engagement and collaboration platforms to connect, share their skills and knowledge, and develop solutions leveraging the full enterprise.

Finally, the Department should encourage greater private-public sector fluidity within its workforce. Federal employees who come from the private sector bring with them best practices, modern methodologies, and exposure to new technologies. Federal employees who leave bring their understanding of our unique mission and constraints, helping the private sector develop offerings and services that meet our needs.

Appendix G: Analysis the Old-Fashioned Way: A Look at Past DoD Software Projects

The Department has been building and buying software for decades. The study's initial idea was to take a cutting edge machine learning tool, hook it up to the Department's databases, and do an analysis across all of the plentiful software data collected over the years.

Unfortunately, initial attempts at analysis quickly led to the realization that the Department had never strategically collected data on its software. The data that have been collected cover only a subset of the systems the Department acquires and are typically collected by hand, with all the potential for erroneous or missing values that that implies. The granularity at which data are collected also does not typically support insight into specific questions of acquisition performance. Without massive data calls, enormous amounts of PDF scanning, and an impossible number of non-disclosure agreements, a comprehensive analysis would not be possible.

Instead, the SWAP members broke the analysis into two main efforts:

1. Analysis of the available data in order to test the board's hypotheses as they evolve. Subject Matter Experts who are familiar with the existing data and its constraints explored the available data in search of insights that would confirm or refute the board's hypotheses about DoD software acquisition performance. These results are described in this appendix.

2. Application of cutting edge machine learning and other modern analytical techniques to datasets from outside of DoD, to support reasoning about the type of insights that could be gained and reported, if the Department had access to more comprehensive data about its software. These results are described in Appendix D.

G.1 Data Used in This Analysis

The focus of this study is on software-intensive programs—and the specific software scope within these programs—presenting top-level insights into software acquisition performance. We focused our analysis on a few major data sources collected by the Department, which can provide insight on these issues.

The data in our first source are known as Software Resources Data Reports (SRDRs). The SRDR data were selected for use because they are specifically focused on the software activities of DoD acquisition programs. The SRDR is a contract data deliverable that formalizes the reporting of software metrics data and is the primary source of data on software projects and their performance. The SRDR reports are provided at the project level or subsystem level, not at the DoD Acquisition Program level. The data points included in the analyses reported here are representative of software builds, increments, or releases. In many cases, there are multiple data points in the set that represent different subsystems or projects from the same program.

The SRDR applies to all major contracts and subcontracts, regardless of contract type, for contractors developing or producing software elements that meet specific criteria[34] and with a projected software effort greater than $20M.

SRDR reports are designed to record both the estimates and actual results of new software development efforts or upgrades, with the goal of supporting cost estimation. The reports collect many characteristics about software activities in both structured and unstructured formats. The primary data analyzed in our work were size, effort, and schedule. Notably absent from the SRDRs are any data about quality. Defect data have been optional until recently and hence were not reported.

Other data sources used to explore some of the assumptions and recommendations of the DIB are the IPMR (Integrated Program Management Report) and SAR (Selected Acquisition Report) datasets. Programs in these datasets fall into the category of Major Defense Acquisition Programs (MDAPs). These datasets include:

1. Software development effort measured in labor hours, software size, and development activity duration metrics delivered as mandated respective to contractual agreements.

2. Software development performance as identified within each contract report. However, each contract contained common elements supporting both software and non-software activity on contracts. These were treated in proportion to the weight of software activity cost on contract. These reports contain data for measuring contractor's cost compared to budget baselines on Department acquisition contracts as well as projections of cost at completion.

3. Planned and executed schedule milestone dates reported to the Department at the aggregate program level as required by acquisition policy. This information is included as a part of a comprehensive summary of total program cost, schedule, and unit cost breach information.

These software development effort metrics, contract performance, and program level schedule data represent the best source of product development, contract cost, and schedule performance information available on various projects throughout DoD. In addition, these datasets are also independently validated by agencies within the Department and subject to audits that require maximum fidelity to accounting standards.

It is worth noting that these datasets provide the best available information on DoD software acquisition, but are mainly limited to contract cost and budget performance (versus technical functionality performance) and were collected by hand. This scenario seems to address larger structural and cultural problems:

---

[34] Specifically, "within acquisition category (ACAT) I and IA programs and pre-MDAP and pre-MAIS programs, subsequent to milestone A approval."

- The Department has no real acquisition data system that holds anything more than top-level data on our largest programs.
- There is no automated collection of acquisition data, despite the fact that software tools and infrastructures, from which data can be automatically extracted, are integral parts of the state of the practice in the software industry.
- For much of the limited software-specific data that we do have (for example, source lines of code, or SLOC), this study has argued that they do not provide meaningful technical insight. Metrics like SLOC are not what the private sector would use to assess and manage programs.
- Leadership often relies on experience and trusted advisors because timely, authoritative data are not available for real analysis.

G.2 Software Development Project Analysis

One area of analysis focused on the SRDR data to describe, at an enterprise- or portfolio-level, what the Department is able to say about its software based on the software-specific data. As described above, SRDR data are more project- or subcomponent-focused versus program- or contract-focused; indeed, it is not easy and perhaps not possible to create a program-level understanding of software activities from the SRDR data.

The results reported here address 3 three questions:

1. How well do software projects perform in terms of effort and schedule?
2. Is there a difference in project performance related to the size of the project and the use of agile development?
3. How long do software projects take to reach completion?

The source of the data was the May 2018 compilation file published by members of the Software Resources Data Report Working Group. This file contains 3993 submissions that yielded 475 initial reports of planning estimates, 598 reports of final actual values, and 295 pairs of initial and final reports. Upon further investigation, 131 pairs contained full life-cycle information and therefore serve as a better dataset for studying effort and schedule growth. Thus, while we base our conclusions in this section on the best available data for software, it is important to keep in mind the data represent only a small subset of the Department's software.

The results presented below were primarily based on common statistical methods. Although a variety of additional explorations were conducted, the results were not found to be stable or to have achieved high confidence. These included dynamic simulation modeling, causal learning, and analysis with repetitive partitioning and regression trees.

Software Project Effort and Schedule Performance

In the current DoD acquisition life cycle, substantial effort goes into defining requirements upfront in extensive detail, and projecting the cost and schedule for achieving the capabilities so described. Despite that, it is often said that the Department has problems acquiring the software capabilities it needs within budget and schedule. This analysis explored whether there was support for this conventional wisdom.

DoD projects in the dataset generally do indeed experience substantial effort growth. As seen in the following figure, the median number of estimated hours is 22,250 while the median number of actual hours is 30,120. (Note that the vast majority of points lie above the green line, indicating that actual values were greater than estimated.) The median rate of growth is 25%. However, there are some projects that expend less than their estimated effort, sometimes by a substantial amount as reflected by the points within the red circle. Unfortunately, based on the data reported we cannot discern whether they delivered the full committed functionality or not.

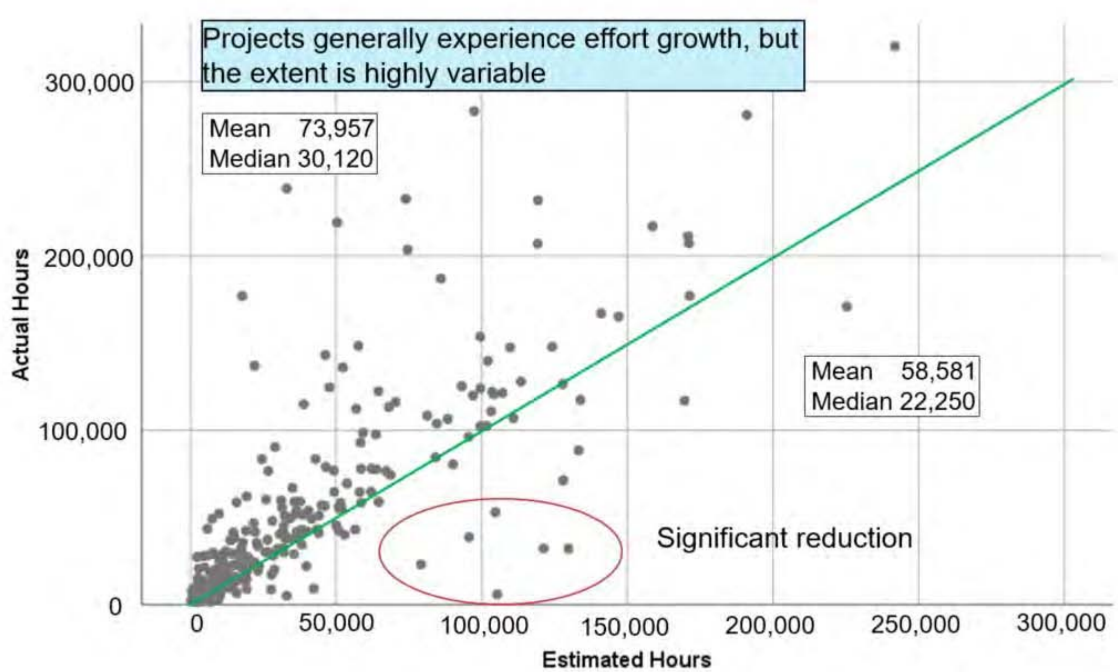


Figure G.1. Estimated and actual project hours for project with less than 300,000 estimated hours.

The growth in project duration is generally not as large as the growth in effort. The median planned duration is 28 months and the actual duration is 34.9 months. The median growth in duration is 12%.

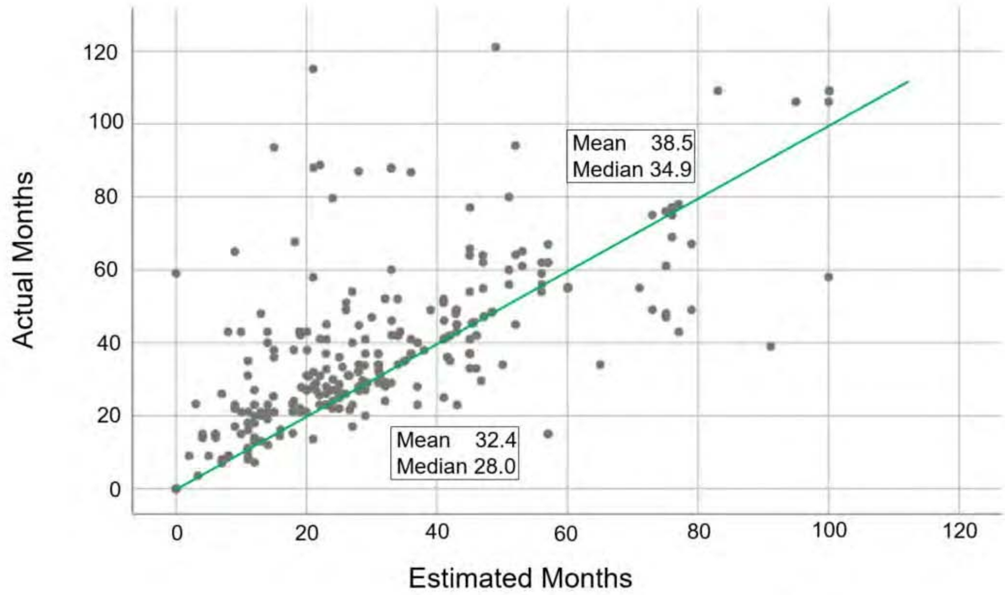


Figure G.2. Estimated and actual project duration.

Interestingly, effort and duration growth are only weakly correlated and the highly skewed nature of their distributions means that averages create a more negative impression of performance than may be warranted. That is, the average exaggerates the degree of growth across the portfolio of projects. Nonetheless, in the data we have available, overruns of effort and duration are the norm.

Does Project Size Affect Performance?

The DIB has recommended that software programs should start small. The next analysis examined the historical data available to test whether small programs performed better than large ones, at least in terms of delivering capabilities on time and within budget.

To perform this analysis, projects were categorized in terms of their estimated equivalent source lines of code (ESLOC)[35] and effort. ESLOC is not collected but computed from the detailed SLOC measures that are collected: ESLOC combines the different sources of lines of code, new, modified, reused, and autogenerated, into a single count. Projects that were in the lower and upper quartiles on both effort and ESLOC measures were labelled as small and large projects respectively. This yielded 53 small and 55 large projects. An analysis of variance was conducted for growth in effort and duration.

The results found that small projects do not outperform large projects. Large projects do have less effort growth on a percentage basis but more growth in terms of raw hours. Surprisingly,

[35] Elsewhere in this report, we reflect on the problems inherent with using SLOC as a measure. However, this is a key measure that has been collected historically by the department and so represents the best available data for this analysis.

schedule growth is very similar. Variation in performance overwhelms any apparent difference and the results do not achieve statistical significance.

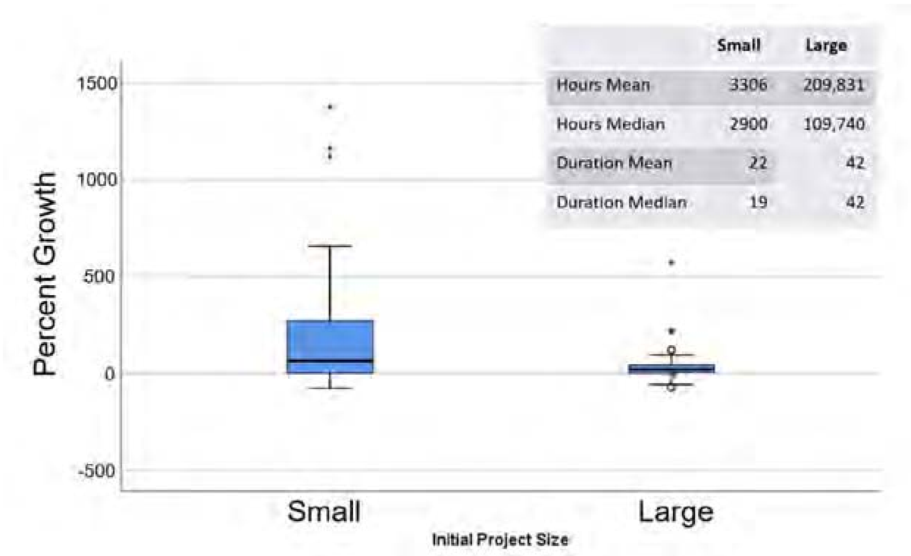


Figure G.3. Effort growth by project size.

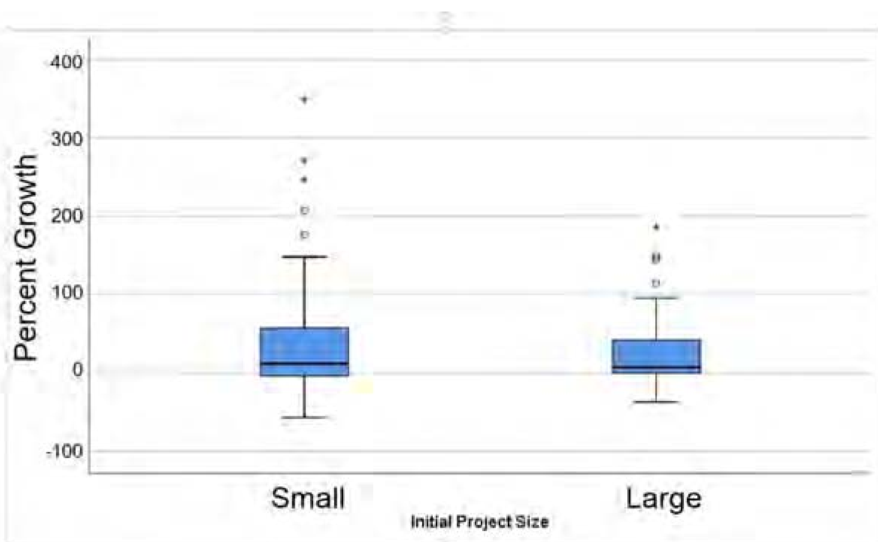


Figure G.4. Duration growth by project size.

The fact that small projects still experience the same growth as large projects does not negate the advice that projects should start small, iterate often, and be terminated early if unsuccessful, since this can still result in significant savings in costs for projects that are not performing well.

Do Development Approaches Affect Performance?

There is much interest in the software development community and DoD in the use of agile methods. While the most recently updated SRDR form explicitly calls out measures for agile projects, this has not been the case for the historical SRDR data upon which these analyses rely.

Furthermore, the identification of the development approach is captured in an open text field. This necessitated interpretation and grouping of the entries in order to perform this analysis. A significant number of projects reported using “Waterfall,” “Incremental,” “Spiral,” or “Iterative” approaches. The remainder suggest use of a customized or hybrid approach. For the analysis here, “Waterfall” is compared to “Incremental,” “Spiral,” and “Iterative” projects.

Again, using ANOVA, the results indicate that effort growth does not significantly vary by development approach. However, duration growth is significantly less for projects using incremental development approaches as compared to waterfall (28% v 70% on average).

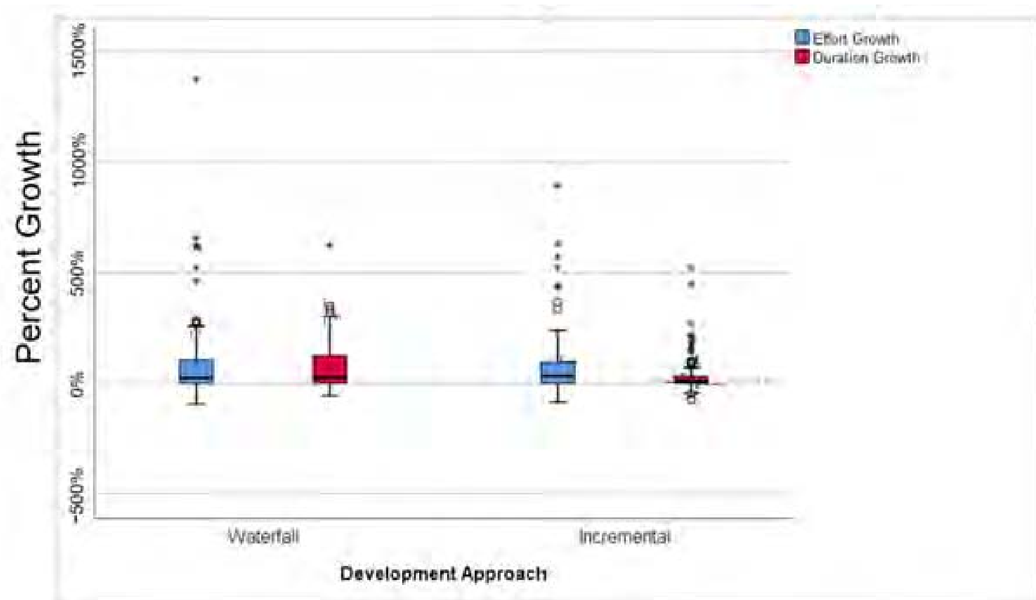


Figure G.5. Effort and duration growth by development approach.

How Long Does It Currently Take to Complete a Project/Deliver Software?

As can be seen in the following figure, it is very rare for a project to complete in 12 months or less. Out of 371 projects used for this analysis, only 21 (6%) completed in this timeframe.

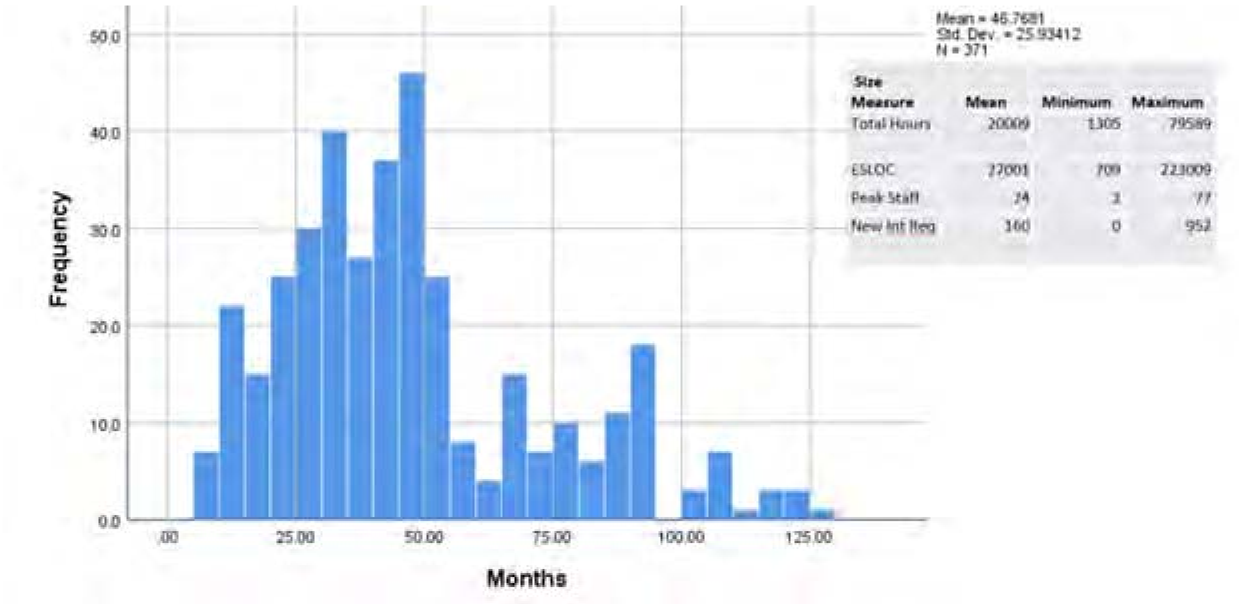


Figure G.6. Actual duration for 371 AIS, Engineering, and Real-time projects.

Additional Insights from the SRDR Data

The preceding analyses were guided by the recommendations and proposed measures in DIB authored documents. In the course of performing those analyses, other questions and issues were posed and investigated. Briefly, these findings are:

1. Extreme variability in project performance confounds the identification of statistically significant results. This was noted above and is most likely actually due to performance and reporting inconsistencies.

2. Planned values can be useful for establishing expectations regarding reported actual effort and duration. That is, planned and actual values tend to be highly correlated with each other.

3. Planning for reuse is associated with significantly more schedule growth as compared to projects that do not plan for reuse.

The last one deserves more explanation as it is a somewhat counterintuitive result. Based on 275 projects that reported either no plan for code reuse or did plan for code reuse, the growth analysis showed no statistically significant differences in effort growth, but a significant difference in the amount of duration growth. Projects planning for code reuse had 52% duration growth as compared to only 20% for those that did not plan for code reuse. This phenomenon has been noted before and attributed to over-optimism about the amount and ease of code reuse. As the ability to reuse code falls short, unplanned effort and time go into producing new or modified code to compensate for the unrealized code reuse. Why effort growth is not significantly different is but likely at least partially related to the extreme variability in the performance measures.

Opportunities for Improving SRDR Data for Use

Issues regarding the data quality of SRDR data used here hampered the analyses. As is noted earlier, there is a substantial reduction from the number of submissions in the system to the number of usable records. At its most extreme there are 131 high quality pairs (262 records) out of the 3993 submissions included in the compilation dataset. That is, roughly 93% of the data is discarded.

The following opportunities are available for improving SRDR data for use in addition to supporting the needs of the DOD cost community. Briefly, they are:

1. Leverage data collection and reporting from automation within the software environments (software factory). Minimize the need for manual entry and transformation.

2. Capture information about the quality of the delivered system.

3. Make the data more broadly available and encourage analyses into DoD software challenges (DIB Recommendation A3).

4. Identify the information needs of the stakeholders and intended users of the data beyond the cost community.

G.3 Software Development Data Analyses

A second investigation focused on cost and schedule performance data reported on recently completed and ongoing software development efforts within DoD. As these data provided insights *within* programs (and allowed understanding how values changed over time), we expected that this analysis would allow for deeper dives that could better explain how software acquisition occurs in programs.

This information was extracted from IPMRs, which are deliverables required by most contracts. The team also reviewed SARs for the large ACAT I programs to gain perspective on programs as they evolve over time.

Poor Data Quality and Inconsistent Data Reporting

There are approximately 130 ACAT I programs reporting research and development (R&D) contract performance over the past 10 years. We discarded from our analysis:

- Contracts for which the first IPMR report showed 65% (or about two-thirds) completed in work scope, reasoning that too much of the work had occurred before data collection began;

- Contracts for which the latest IPMR reported work that was less than 70% complete, reasoning that we would not have the ability to evaluate a significant portion of work completed.

146 contracts (35%) did not meet these data quality criteria out of the total of the 413 ACAT I program development contracts for which we have data (Figure 7). The fact that more than one-third of contracts do not meet this criterion implies that DoD would benefit from improving the quality and consistency of software development performance reporting. DoD cannot comprehensively assess the performance and value of the billions of dollars in investment without insight into a third of the complete portfolio.

Additionally, there are many data that are of limited utility due inconsistencies related to reporting. These have to do with problems with filing the mandated regular reports, and a lack of contextual data (i.e., metadata) being collected in a readily analyzable form. The DIB Software Metrics Recommendations contain recommended best practices on data collection and metrics definitions to not only capture data, but to establish standards meant to enhance software development performance.

Cost and Schedule Data

The resulting list of contracts was prioritized based on the budget assigned to the software-specific development efforts, and the top 46 contracts with the largest budgets were included in this study. These 46 contracts covered roughly half of the total dollar scope for all development programs in our dataset, and thus provided a reasonable sample size for our analysis. In addition, 35 contracts for smaller ACAT II and ACAT III software intensive Command and Control (C2) and Automated Information System (AIS) programs were included in this analysis. This resulted in the study capturing 81 total contracts valued at $17.9B in software development cost over the past 10 years (2008-2018). This study did not attempt to qualify or quantify the reasons for cost and schedule growth, recognizing that growth is not always indicative of poor performance by the program and/or contractor.

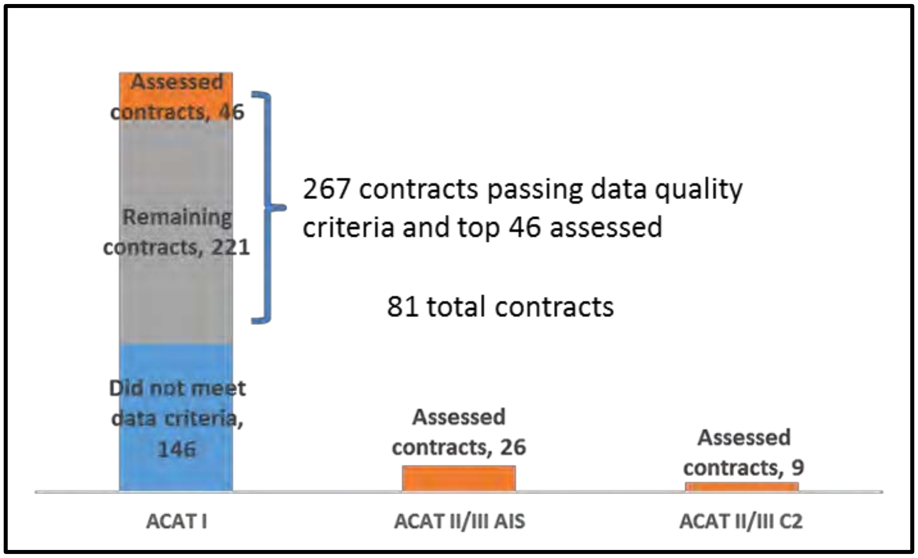


Figure G.7. Results of contract selection process.

The 81 total contracts included in this analysis covered the portfolio of DoD programs, including software intensive C2 and AIS programs as well as aircraft, radars, land vehicles, and missile weapon systems, as shown in Figure 8.

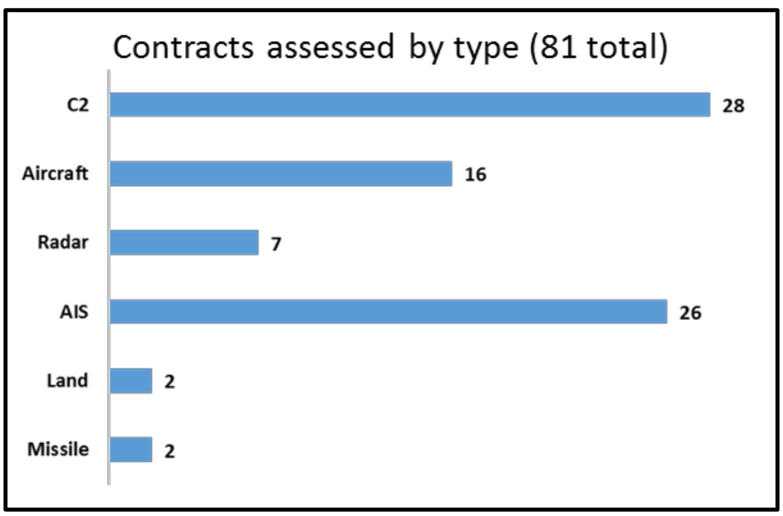


Figure G.8. Contracts analyzed by weapon system type.

Large Software Cost Growth

The analysis of IPMR data found that on average, the contracts experienced 138% cost growth. The total combined value of the software development budgets within these contracts was $7.6B at the time of initial reporting. By the time these contracts reported the latest (or in some cases, final) performance baseline, the software development budget total grew by $10.4B. Based on the analysis completed, significant software development cost growth was experienced across all platform and program types, resulting in a second observation: In general, the DoD struggles to minimize software development cost growth across the complete portfolio of projects. Figure 9 provides a summary of the 81 contracts evaluated, organized by project and by platform type. Note that the cost growth of “C2 Program A05” was truncated in the figure as it was an outlier in the analysis.

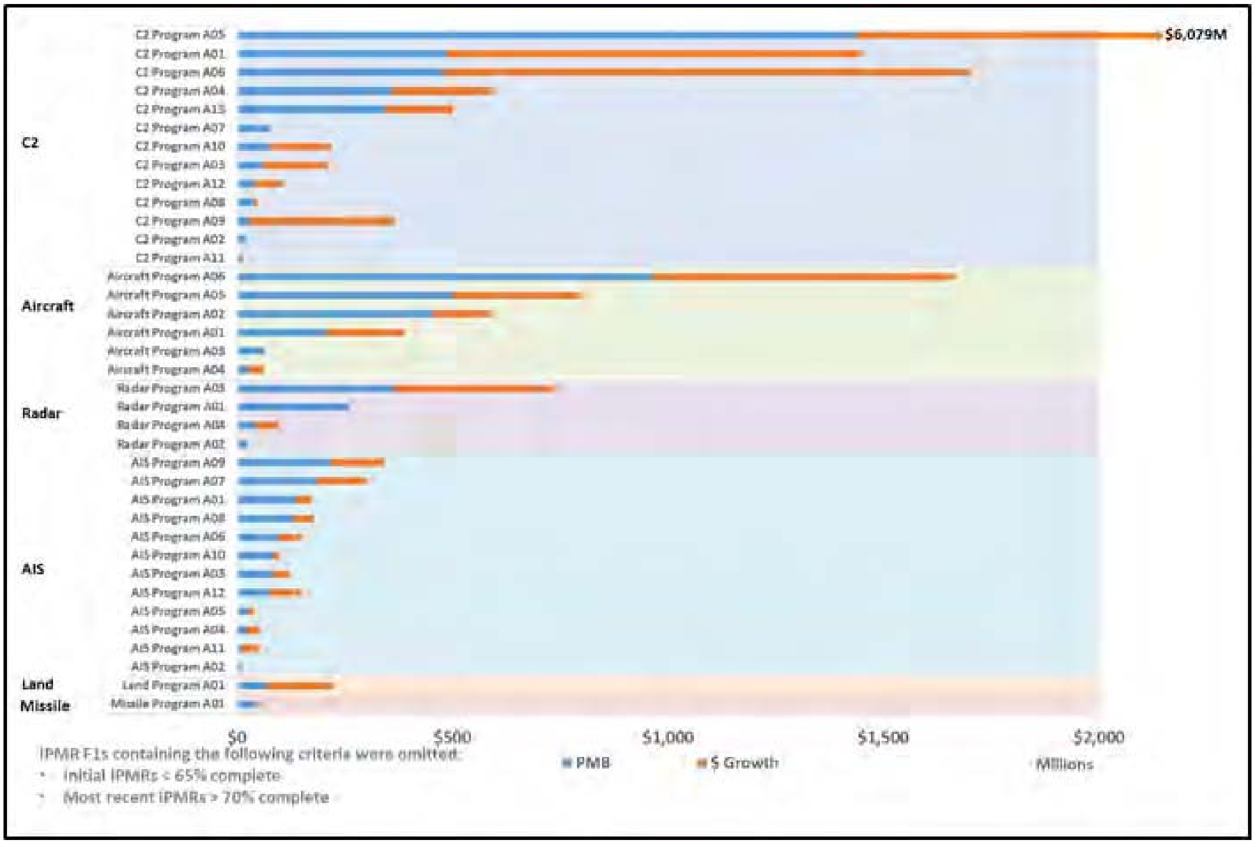


Figure G.9. Contract software development cost growth by program and by platform.

The study team used information provided by SARs and other relevant acquisition documentation to calculate project schedule growth. Figure 10 illustrates both dimensions of cost and schedule performance and identifies programs for which actual performance exceeds more than twice the baseline cost and schedule. Two programs, “AIS Program A01” and “C2 Program A02,” experienced cost or schedule growth so extreme that the bounds of the diagram axis plots were exceeded. This figure also supports the second observation that recent software development programs experience significant cost growth. The DIB SW Commandment 3 addresses cost growth by advocating that software budgets be planned upfront to support the full life cycle versus the current funding life cycle, defined around Planning, Programming, Budgeting, and Execution (PPB&E).

Figure G.10. Software development cost growth vs. program schedule growth

Long Planned Durations and Frequent Re-baselining

The third study observation results from a deeper look into programs with high cost growth. This research found that in numerous instances, program baselines shifted (re-baselined) during the contract period of performance. The contracts with what appear to be significant “re-rebaselining” (i.e., multiple recurring increases to the expected cost) were analyzed in further detail.

SAR program milestones and available open source data were evaluated to provide a scale of time and functionality. It is observed that the software development effort crosses the same percent complete, as defined by the Earned Value Management (EVM) metric as the ratio of Budgeted Cost of Work Performed (BCWP) to Budget at Completion (BAC), multiple times. This represents an incremental method of adding cost, which is presumably associated with the addition of technical scope and requirements, which can result in a doubling or tripling of the total original budgeted value of the software development effort.

Figure 11 provides an example of this behavior, showing the “C2 Program A01” program effort that appears to re-baseline several times. The software development effort crosses the same percent complete point multiple times.

DIB Software Commandment 2 provides the recommendation that software development should begin small, be iterative and build on success; otherwise, be terminated quickly. DoD programs that take this approach are likely to see an improvement in performance once scope and requirements can be delimited through successful iteration. The behavior demonstrated in Figure 11 seems to indicate that to some extent, at least some programs are already behaving in an iterative way that better suits the technical work of software evolution. Unfortunately, our reporting mechanisms are not suited to reflect this reality, and in fact cannot differentiate a reasonable approach to incremental development from problematic cost or schedule growth. Looking just at the top-line numbers, these instances could be interpreted as excessive cost growth on the program, representing a problem from the Department's point of view since the predictability of performance against cost and schedule baselines are normally taken as indicators of success. What this scenario seems to point to is a need to improve our metrics collection to better reflect the underlying technical reality of software, where good performance often leads to a demand for new capabilities and new scope, as well as better educating our decision makers about how to interpret the results.

Thus this example provides more information about associated reporting issues tied to observation 5, that budgets should be contracted to support the full, iterative life cycle of the software being procured with amounts definitized proportionally to the criticality and utility of the software.

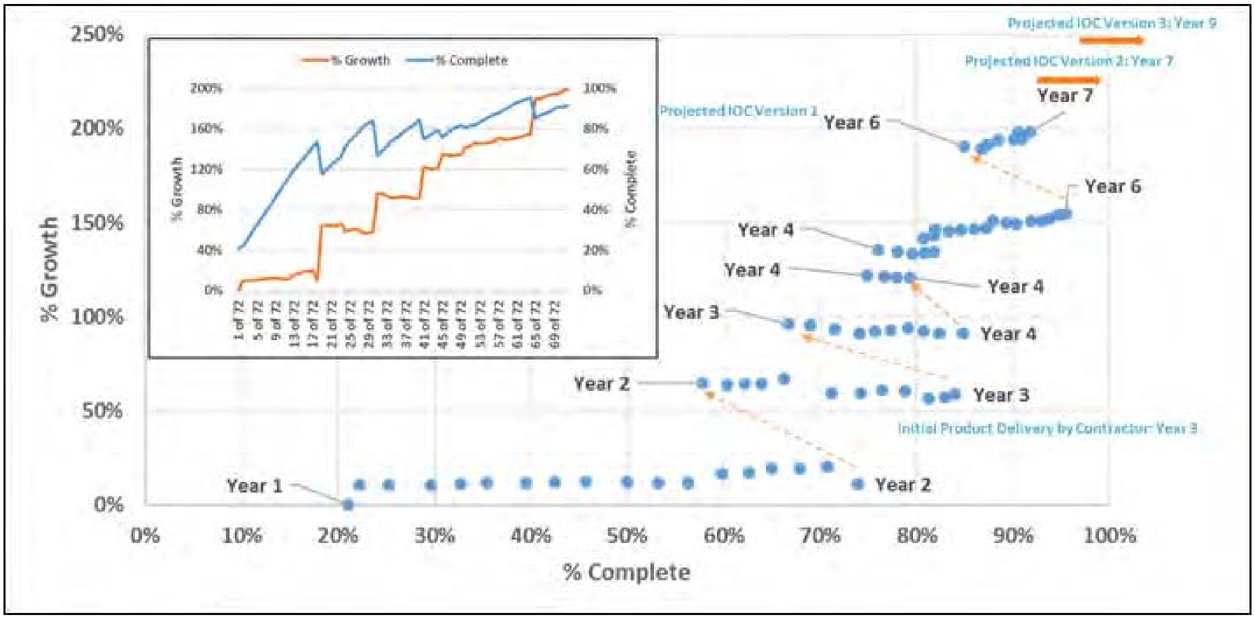


Figure G.11. C2 Program A01 performance measurement re-baselining.

Agile Software Development Can Improve Program Performance

This study researched the performance of agile development methods that are implemented in existing programs. IPMRs do not explicitly state the type of development effort being used (incremental, agile, etc.). However, an article published in the journal Defense Acquisition provided an instance where agile development was applied and considered a success story. Although this article did not name the program, we were able to identify the most likely candidate, "Aircraft Program A05," by matching the timeline presented in the article against the timeline of contracts that we could see in the program data.

The IPMR data for this program are shown in Figure 12. The contract work completed using an agile approach are shown in blue and represent a 21% cost reduction when compared to the initial budgeted value. This is in contrast to the contracts that seem to adopt a waterfall development methodology, i.e., contracts with planned long durations, which are shown in shades of orange and represent a 129% cost growth compared to the initial budgeted cost.

This analysis supports the fourth study observation that agile development may reduce cost growth compared to more traditional waterfall approaches. The DIB SW Commandment 2 also advocates that agile approaches seen in commercial development result in faster deployment of functionality and cost savings which we observe in this instance.

Though a comparison of cost is one facet of performance, more research is required to increase the certainty that better overall performance and results were achieved with agile methods.

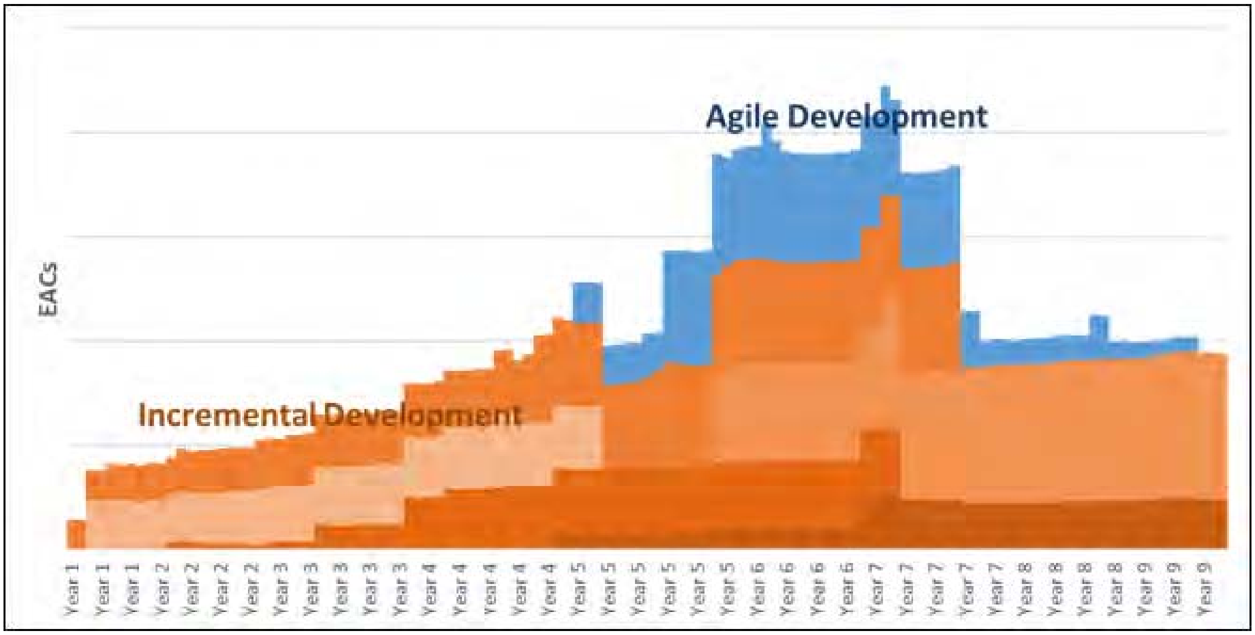


Figure G.12. Aircraft program A05: incremental vs. agile development efforts

Cost and Schedule Analysis Summary

In important ways, this analysis was typical of other efforts that aim to use Department data to examine the performance of acquisition. Due to the limited nature of the data available, our best analyses typically take months to create, with substantial time needed to find the data, to collect them, and to compile them into a structured format from multiple siloed and restricted systems.

The observations taken from data analysis of DoD program cost and schedule performance support the supposition that the current state of software acquisition is highly problematic and unsustainable relative to affordability and functionality. The DIB SW Commandments 2, 3, and 4 provide recommended measures to contain growth and increase the opportunity for cost savings by detaching software development from a hardware manufacturing industrial model and integrating software development and operations to quickly provide functionality to users and meet changing needs dictated by a dynamic global environment.

The preceding sections have described specific conclusions from the analyses our team conducted. Equally important, however, are the types of analyses we were *unable* to conduct given the data that were available.

A notable omission is that the Department is unable to address questions of *how much* software it has. Not in terms of software size but in terms of an index of how many important software systems have been acquired or are being sustained by the Department: There is no DoD or Service framework for describing the types of software intensive systems, or any inventory/ catalogue of the software in use. As a result, it is challenging to comprehend the scope and magnitude of the DoD software enterprise, and to design appropriate solutions for issues such as infrastructure or workforce that can meet the magnitude of the problem. Although done at a smaller scale, NASA's software inventory is an example of such an inventory model that is used to make strategic decisions for a federal agency.[36]

There is a large and growing body of work on software analytics, the automated or tool-assisted analysis of data about software systems (usually collected automatically) in order to make decisions. Conferences such as Mining Software Repositories[37] and Automated Software Engineering[38] annually showcase the best of the new research in these areas, and these methods are having a practical impact in commercial and government environments as well. A summary of software analytic applications lists several important questions that can be explored in this way: to name just a few, "using process data to predict overall project effort, using software process models to learn effective project changes, … using execution traces to learn normal interface usage patterns, … using bug databases to learn defect predictors that guide inspections teams to where code is most likely to fail."[39] Without access to its own software data, DoD is missing the opportunity to exploit another area of research that could provide practical benefit for improving acquisition.

In a later section of this report (Appendix H), we provide the results of a small study that was undertaken to demonstrate potential practical impacts that could be achieved if software data access could be possible in the future.

---

[36] NASA Engineering Handbook (https://swehb.nasa.gov/display/7150/SWE-006+-+Agency+Software+Inventory#_tabs-6).

[37] https://2018.msrconf.org/

[38] http://ase-conferences.org/

[39] T. Menzies and T. Zimmermann, "Software Analytics: So What?," in *IEEE Software*, vol. 30, no. 4, pp. 31-37, July-Aug. 2013. DOI: 10.1109/MS.2013.86

Appendix H: ~~Replacing~~ Augmenting CAPE with AI/ML

Linda Harrell, John Piorkoski, Phil Koshute, Erhan Guven, Marc Johnson (JHU/APL)
Vladimir Filkov, Farhana Sarkar, Guowei Yang, Anze Wang (UC Davis)
Steven Lee (Rotunda Solutions)

H.1 Introduction

The Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) study chartered an exploratory study to explore the use of modern tools in data analytics and Machine Learning (ML) to provide insights into cost, time, and quality of Department of Defense (DoD) software projects. The data analytics and ML effort were performed by a team from academia (University of California Davis (UC-Davis)), a university affiliated research center (The Johns Hopkins University Applied Physics Laboratory (JHU/APL)) and industry (Rotunda Solutions). Since a suitable DoD data set was not available, the three teams leveraged existing data sets that were readily available to perform ML experiments and quickly get results.

ML models were created to predict the cost, time, and other aspects of software projects and gain a deeper understanding of the potential impact of project characteristics on overall project budget and effort. The models were trained with different data sets and were constructed to predict different performance metrics throughout the software development life cycle.

The JHU/APL team developed ML models to predict software project duration and effort using the commercially available International Software Benchmarking Standards Group (ISBSG) Development and Enhancement (D&E) Repository of completed software projects. The UC-Davis team developed ML models to forecast software project duration, effort, and popularity using the publicly available GitHub repository of open-source projects. Finally, Rotunda Solutions created a defect density ML model to capture the code complexity and predict potential risk of code modules using a publicly available NASA dataset.

Additionally, the Rotunda Solutions team identified a number of opportunities for harnessing ML and Artificial Intelligence (AI) to improve the software acquisition process during different phases of the procurement cycle. This research effort is referred to as the Opportunities for Analytic Intervention. Rotunda Solutions also started development of a conceptual mock-up to explore some of these opportunities.

Overall, the three ML model development approaches demonstrated promising results aimed at improving predictions of software cost, time, and quality during different life-cycle phases.

- The JHU/APL team identified features (software metrics) that can support predictions of duration and effort at the project onset and shows that ML models have very good accuracy even with as few as 5 to 15 important features, most of which can be easily collected. It also shows how the prediction accuracy increases slightly by also including the effort expended in different life-cycle phases (e.g., planning, specification, design, build, test, and implementation). Since this analysis addresses the whole software life cycle, the APL effort is referred to as the Software Life-Cycle Prediction Model.

- The UC-Davis team shows how monitoring of software development activities over time via automated tools that capture metrics (such as the number of lines of code, the number of commits, and team size) can support accurate forecasts of duration, software effort (SWE), and software popularity. Additionally, the UC-Davis analysis showed that the ML models could obtain very good forecasting accuracy only 6 months after code development has started. Hence the UC-Davis ML model can serve as an early warning indicator. Since this analysis leveraged data obtained during software development activities to forecast future outcomes, it is referred to as the Software Development Forecasting Model.
- The Rotunda Solutions defect density model automatically processed code files and output code complexity metrics to aid efficient resource allocations and risk mitigation.

Interestingly, despite the differences in the approaches taken by JHU/APL and UC-Davis, the teams shared similar conclusions. For instance, both teams identified the team size and the project timing as being important features for the predictions.

Section H.2 of this document describes the methodology applied to the APL Software Life-Cycle Prediction Model and the UC-Davis Software Development Forecasting Model. Section H.3 summarizes the major findings of all three analyses. Section H.4 offers implications of these study results for DoD programs.

H.2 Methodology

The approaches taken for the APL Software Life-Cycle Prediction Model and the UC-Davis Software Development Forecasting Model were complementary. Table H.1 summarizes key aspects of the two approaches. These aspects include:

ML Techniques. Both studies leveraged readily available commercial or open- source ML techniques. This enabled the teams to meet the task's quick reaction turn-around timeline and also ensures that DoD government personnel and contractors can apply a similar approach when they develop their own prediction models for software projects. Although the teams developed several types of ML models, this report focuses on those with the best results: the APL Random Forest (RF) and the UC-Davis Neural Network (NN) models.

Data Sets. The APL team leveraged the 2018 International Software Benchmarking Standards Group (ISBSG) Development and Enhancement (D&E) Repository of completed software projects. This diverse database contains thousands of software projects that are described by a rich set of features that span the whole software life cycle, but most of these projects have less than one year in duration or less than two years of effort. The UC-Davis team mined the GitHub collaborative project development and repository site, which contains historical trace data captured from millions of open-source software projects. The resulting database includes hundreds of thousands projects of various sizes. Its feature set is not as rich as in the ISBSG database, but it automatically tracks development metrics including commits, discussions, and other activities.

Target Variables. The APL team focuses on predicting software project duration and effort, two of the three metrics of greatest interest to the DIB. On the other hand, the UC-Davis team aims to predict the project duration (via its proxy months committed), the number of software commits (which is an incomplete proxy for software effort), and the number of stars (which is an indicator of the popularity of a project in GitHub).

Project Tiers and Boundaries. Large differences between proposal estimates and actual outcomes for software development duration and effort cause the biggest challenges for DoD; small deviations are much more manageable. To reflect this perspective, both studies gathered their target variables into discrete tiers with boundaries shown in Figure H.1.

Performance Metrics. Both studies assessed the performance of their models with confusion matrices (which shows the distribution of predictions in terms of predicted and actual tiers) and overall accuracy.

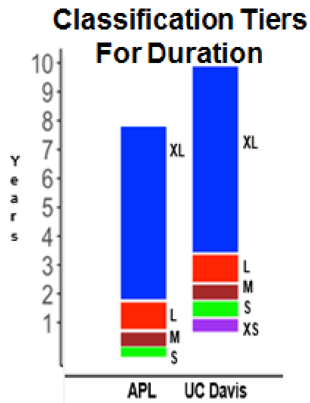
Table H.1. Key aspects of APL and UC-Davis studies

Parameter		**APL Software Life-Cycle Prediction Model**	**UC-Davis Software Development Forecast Model**
Data Set		2018 ISBSG D&E Repository	2018 GitHub Repository
Number of Projects (after preprocessing)		2,818	Approx. 127,000
Number of Features (after reduction)		176	36
Target Variables for …	Duration	Project Duration	Months Committed
	Effort	Effort	Total Number of Commits
	Popularity	N/A	Number of Stars
ML Techniques		Off-the-shelf (NB, SVM, RF)	Off-the-shelf (MR, NB, RF, NN)
Results: Overall Accuracy; Confusion Matrices		Overall accuracy: Yes Confusion Matrix: 4 tier	Overall accuracy: Yes Confusion Matrix: 5 tier
Prediction Snapshots		Early concept development and procurement; Software development in process	After 6 months of software development ; Most recent software development
Feature Reduction		Yes	Yes

Definitions: NB = Naive Bayes, SVM = Support Vector Machines, MR = Multivariate Regression, NN = Neural Networks

APL Classification Tiers

Project Size	Software Effort
Small	2 months to 0.999 year
Medium	1 year to 1.999 years
Large	2 years to 7.999 years
Extra Large	>= 8 years



UC Davis Classification Tiers

Project Size	Number of Commits	Number of Stars
Extra Small	30 - 90	0 - 2
Small	91 - 140	3 - 7
Medium	141 - 217	8 - 22
Large	218 - 384	23 - 90
Extra Large	385 - 3469	91 - 31168

Figure H.1. Classification tier boundaries.

Prediction/Forecasting Snapshots. APL made predictions at two project phases (snapshots). The first snapshot is at onset, which includes features that are available or can be estimated during the concept, proposal, and procurement stage. The second is after software development has been underway; it can include additional features as they become available. UC-Davis made predictions at three snapshots, corresponding to the time elapsed for each project: 6 months from first commit, 12 months from first commit, and most recent snapshot (1/1/2018). The most recent snapshot is taken to be the actual outcome (even if the project is still under development). For simplicity, the results with the 12-month snapshot are not discussed herein.

Feature Importance Ranking and Reduction. The APL RF and UC-Davis NN models both determined feature importance by evaluating the importance of each feature to the overall accuracy prediction and developed corresponding models with only the top ranked features.

Pre-Processing and Feature Selection. The pre-processing actions taken by the APL and UC-Davis are discussed in separate reports.

Project Context (Cluster) Creation. To fine-tune their predictive models, UC-Davis used an Autoencoder NN to group projects into four similarity clusters (i.e., contexts). A separate model NN was trained for each cluster. This technique allows for greater accuracy when project context is known early on, by, for example, tracking project metrics from the start.

H.3 Key Results and Findings

APL Software Life-Cycle Prediction Model

Table H.2 shows the performance of the APL models that predict software project duration and effort with all features included. Even with minimal data cleaning, model tweaking, or sensitivity studies, and using a very sparse and unevenly distributed data set, the ML models predict a project's size tier with an overall accuracy ranging from 57% to 74%. These are impressive results for a quick-turnaround exploratory analysis.

As expected, the prediction estimates once development is underway are better than the predictions at program onset. This is because additional features, such as the effort expended in various life-cycle phases, help to improve predictions. However, with the features included in this analysis, the improvement was slight.

Even when the ML model does not correctly predict the size of the software project, the prediction is most often in adjacent tiers rather than significantly further away. This is evident in the confusion matrix in Table H.3 and the additional confusion matrices provided in separate reports. This is important because it indicates that incorrect predictions still tend to be fairly close (e.g., an extra-large project predicted as large or vice versa).

Table H.2. Performance summary for APL prediction models (with all features)

Model	Overall Accuracy
Predicting Duration at Project Onset	57%
Predicting Duration after the Project is Underway	58%
Predicting Effort at Project Onset	68%
Predicting Effort after the Project is Underway	74%

Table H.3. APL confusion matrix for predicting effort as project is underway (with all features)

Accuracy values are shown as a percent of all projects of a given class		Predicted Class			
		S	**M**	**L**	**XL**
Actual Class	**Small (S)**	80	18	2	0.1
	Medium (M)	23	59	18	0.6
	Large (L)	2	20	73	5
	Extra Large (XL)	0.1	0.8	14	85

Table H.4 identifies the most important features that influence the predictions. Naturally, the ranking of importance for each feature varies slightly for the predictions of duration and effort and for the two different phases (at project onset versus while the software development is underway), but the discrepancies are generally slight. Encouragingly, the features in this table are generally easy to obtain or estimate: function point standards, team size, software type, project implementation date, scope, programming language. The only feature category that is time consuming to gather is the functional size estimate. Each of the features in these tables is further described in the APL report.

Table H.4 Most important features for ML accuracy predictions

Category of Feature	Most Important Features	Project Phase
Software Size	Functional Size, Relative Size, Adjusted Function Points	Project Onset
Standards for Function Point Estimates	Function Point Standards, Count Approach	Project Onset
Team	Maximum Team Size, Team Size	Project Onset
Type of Software	Industry Sector, Organization Type, Application Type, Business Area	Project Onset
Timing	Year of Project, Implementation Date	Project Onset
Scope	Project Activities, Development Type	Project Onset
Programming Language	Primary Programming Language, Language Type, Development Platform	Project Onset
Incremental Effort	Effort in the Planning Phase, Effort in Specify Phase, Effort in Design Phase, Effort in Build Phase, Effort for Implementation, Effort in Test Phase	When the Project is Underway
Cost	Total Project Cost	When the Project is Underway

Figure H.2 depicts the accuracy prediction with small subsets of the most important features, and shows how the accuracy increases as additional features are added. This figure shows that although the database includes 176 features, very good predictions can be obtained using only as few as 5 to 15 features. These features are captured in Table H.4.

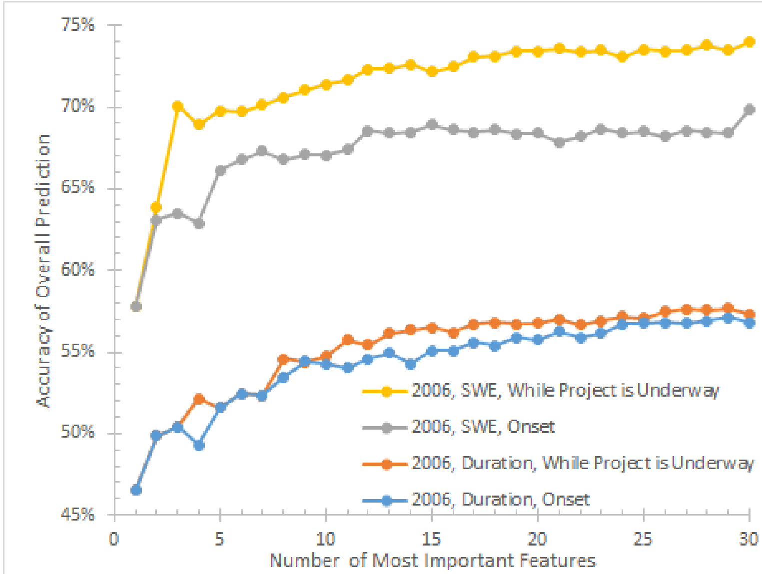


Figure H.2. Accuracy of APL's software project duration and software effort (with reduced, prioritized feature set).

The APL Software Life-Cycle Prediction model results clearly show that ML models can quickly be developed and trained using only a relatively small number of projects, a very small number of features, and a large amount of missing data. Furthermore, the resulting predictions for a software project's duration and total effort can be reasonably accurate at the project onset, and can then improve slightly over time by tracking the effort that is expended over the life cycle. Only about 5 to 15 features are required to achieve reasonable predictions. The most important features for the predictions were identified; most of them are easy to obtain or estimate.

UC-Davis Software Development Forecasting Model

UC-Davis developed models that predict project duration, number of commits, and popularity using all available historical data of completed projects in the January 2018 snapshot, starting from the first commit of software. Table 3.4 shows the best-case overall prediction accuracies that can be obtained with these models and all of this data. The best-case overall accuracy of the prediction estimate for project duration is 84% and the best-case overall accuracy of the prediction estimate for the number of commits is 72%. Predictions for popularity were less accurate. These results indicate that the features in the GitHub database will be very useful for predicting software project duration and to a lesser extent the predictions for the number of commits. It appears that additional features will be necessary to improve the predictions for software popularity.

Additionally, Table H.5 also shows that the best-case overall accuracy results for these models vary for different context clusters of similar projects. For instance, the accuracy values for each target variable increase within certain clusters; accuracy is greater in Cluster 1 by 16% for project duration and by 24% for number of commits and in Cluster 4 by 13% for popularity. These increases suggest that clustering projects based on similar context can increase the best-case prediction accuracy and that different models may be necessary to best predict different project contexts. The descriptions of these different clusters are not available at this time, but it would be valuable to investigate this further in order to understand the project characteristics that distinguish the clusters.

Table H.6 shows the best-case overall accuracy of the UC-Davis models that use only the 9 most important features from the full project lifetime. These results are very close to those of the models that use all available features, indicating that the reduced feature set is sufficient for accurate predictions.

Table H.5. Full lifetime (best-case) prediction accuracy

Target Variable	All Projects	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Number of Projects	126,799	21,462	31,918	55,065	18,354
Project Duration (months committed)	84%	99.5%	83%	80%	78%
Number of Commits	72%	96%	70%	62%	69%
Popularity (number of stars)	49%	46%	48%	42%	62%

Table H.6. Full lifetime (best-case) prediction accuracy with reduced feature set

Target Variable	**All Features** (All Clusters)	**9 Most Important Features** (All Clusters)
Project Duration (months committed)	84%	84%
Number of Commits	72%	74%
Popularity (number of stars)	49%	48%

Table H.7 shows the accuracy results of the forecasting models, which predict the target variable in the final snapshot using features from a snapshot taken 6 months after project starts. These results are averaged over each of the 4 clusters (i.e., include 126,799 projects). These forecasting results show that data from only the first 6 months into a project can predict future outcomes, reaching accuracies of approximately 50% for both project duration and number of commits.

Table H.8 identifies the most important features that influenced the UC-Davis predictions and forecasting. This table shows that features related to teams and commit activity are the most important for the UC-Davis models.

Table H.7. Forecasting accuracy (averaged over all clusters)

Target Variable	Prediction of target variable at last snapshot given 6 month snapshot	Prediction of target variable at last snapshot given all data
Project Duration (months committed)	53%	84%
Number of Commits	50%	72%
Popularity (number of stars)	41%	49%

Table H.8. Most important features for the UC-Davis predictions and forecasting

Feature Category	Most Important Features
Commit Activity Data	First Commit Date, Months Committed
Team Member Data	Team Size, Number of Commenters, Number of Pull Request Mergers, Average Months Active, Standard Deviation (SD) Months Active, Average Commits per Month, SD Commits per Month

In summary, the UC-Davis analysis shows excellent results for being able to forecast project duration and the number of commits only 6 months into a project. Only 9 features are required to achieve these forecasts. The most important features for the predictions were identified; all of them easily obtained with automation tools that track software development activities. Additionally, UC-Davis uncovered clusters of projects that if better understood could lead to improved models and accuracy predictions.

Rotunda Solutions Investigation of Opportunities for Analytic Intervention

The Rotunda Solutions effort focused on identifying strategic opportunities to leverage ML and AI at key points in the overall DoD procurement process. It extended academic research and state-of-the-art quality management principles to identify opportunities to improve the likelihood of successful software development outcomes. It also developed initial conceptual mock-ups to explore potential applications, including a defect prediction platform.

Rotunda Solutions adopted a basic stage-gate model to represent the general structure and stages of a DoD procurement and project development effort. Multiple opportunities are identified in each stage where analytics, ML, and other modern techniques can assist project managers. First, analytics can provide metrics and insights to support the project manager's yes/no/hold decision for whether the project should move to the next development stage. Second, analytics and ML can facilitate the search and interpretation of DoD procurement and development data sets so that decision makers have better access to historical data. Third, analytics can be run on this historical data to provide insights that can inform future projects. The application of modern techniques within a basic stage-gate model for a typical DoD procurement and development project can be envisioned as follows.

Stage 1: Idea Generation/Need Analysis. Analyze the internal unstructured documents from the program office and communications between suppliers and procurement officials. Then apply problem identification analytics to define the problem to be solved, considering the following 5 major groups/factors: need spotting, solution spotting, mental invention, market research, and trend. The literature shows a clear trend in savings of time and resources during the development process by maximizing the effectiveness of the idea generation stage.

Stages 2 and 3: Proposal Development and Response. Analyze internal unstructured documents from the program office and communications as they relate to proposal development and response. Use qualitative techniques such as focus groups, in-depth interviews, and surveys to determine factors associated with development success and failure. Additionally, use natural language processing (NLP) techniques to prepare the documents for further analysis. Both methods can identify key mechanisms and characteristics of software development success.

Stage 4: Contract and Award. Identify keywords through analysis of prior software contracts. Use NLP and topic extraction on legal documents surrounding the final selection of the supplier, contract vehicles, set-asides, and all stipulations to determine content. This can increase the ease of detecting associations between numerous demographic and supplier characteristics and software development performance. It also provides the ability to build a grading system and general profile of contractors and their performance on projects.

Stage 5: Software Development. Gather representative data regarding project management metrics, code base, and development metrics, and compile a list of metrics that can help identify the likelihood of success of a DoD software development project. This helps DoD in two ways: first by identifying projects that are likely to succeed or fail in each stage; and second by informing cost and time estimates for future software acquisition projects. Alternatively, analyze code to inform the development of ML tools to assist project managers and developers understand the state of their code. Potential benefits of this analysis include tools that can rapidly identify errors and increase efficiency for automation, audits, process checkpoints, and standardization.

Stage 6: Implementation. Harness available information on users, development, delivery personnel, and performance metrics of the software system. Measure the efficacy of the deployed or implemented software systems through metrics such as dependability, system performance, extensibility, and cross-platform functionality. This provides a postmortem analysis of the efficiency and effectiveness of the software and the development process, allowing DoD to learn from past experience and increase the likelihood of future development success.

Conceptual Mock-Ups

Rotunda Solutions aims to help DoD in four ways: (1) understand the potential impact of variables, decisions, and project characteristics on project budget and effort, based on historical data of similar projects; (2) make data-informed project decisions pertaining to the adjustment of project structure, methods, and other details; (3) create and explore what-if scenarios to promote better planning; and (4) encourage transparency and traceability of factors and decision-points affecting project performance. To this end, a number of concepts offer potential for further development and exploration. For instance, the concept of an "intelligent" burn-down chart is especially intriguing. Given sufficient sprint data and historical trend data, effort estimation tools and ML algorithms can be leveraged to make real-time predictions and issue alerts when estimates of team effort needs a closer review. Also, a defect prediction algorithm may be able to support risk mitigation activities and improve resource allocations.

Focus Area: Defect Prediction Platform

Software defect prevention is an essential part of the quality improvement process; timely identification of defects is important for efficient resource allocation, increased productivity, and risk mitigation, yet complete testing of an entire system is generally not feasible due to budget and time constraints. Studies show that the majority of software bugs are often contained within a small number of modules. To more rapidly identify these modules, Rotunda Solutions developed a system to automatically process code files and output code complexity metrics. They built off extensive industry research and tested representative NASA software modules using NN, SVM, Gaussian mixtures, and ensembles of ML techniques. The NN model performed best and was selected for production.

The NN model consists of 8 hidden layers, each layer becoming smaller until converging on a single probability to represent the existence of defects in the file. This model learns to assign importance weights to each of the 17 features and to combine these features in non-linear ways to identify any potential defects. The NN can then be used to give a probability of defects for future files. This could help the management team in three ways: (1) to recognize the likeliest modules to have defects and allocate corrective resources effectively; (2) to provide an overview of the riskiest code modules to identify opportunities to re-architect the application; and (3) to understand the risk of deployment in production by an automated code complexity review.

Caveats and Limitations

It is important to note that there are significant differences between the software repositories used in this work and important classes of software acquired by DoD. For example, embedded software used in DoD weapons platforms is typically marked by high complexity, with low tolerance for reliability, availability, safety, and security issues. Although the testbeds on which the ML approaches were applied do contain some NASA software, only a small subset at best of the systems providing data are expected to have similar characteristics. As a result, it is important to view these results as showing a potential method that would be applicable to DoD programs and could learn characteristics of interest within that environment. While the method may be of interest, the specific results summarized may not directly carry over to some types of software present in the DoD environment.

Conclusions

The Rotunda Solutions exploration outlined the potential benefits of harnessing ML/AI throughout the DoD software acquisition life cycle. These benefits include increased accuracy of budget predictions, comprehensive planning, mitigation of expensive defects, and transparency. Rotunda Solutions also identified many opportunities and applications that may improve DoD software development and estimation practices.

H.4 Implications of the Study Results for DoD

This ML study demonstrated promising results by creating models with publicly available software project data. It uncovered a promising approach (the APL Life-Cycle Prediction Model) that can be used to develop good predictions of software duration and effort in the early stages of software procurement and development. The study also uncovered another approach (the UC-Davis Forecasting Model) that can further improve project estimates once software development has been underway for 6 months or more. Finally, the Rotunda Solutions defect density model can highlight modules requiring additional resources and risk mitigation efforts.

The generalizability of these models to DoD software projects requires validation. For instance, a pilot study could be conducted with a small subset of DoD projects. Ultimately, strategies can be developed to enable DoD leadership to effectively leverage ML models.

One strategy could entail a strong centralized mandate for DoD software development teams to provide project data to DoD oversight personnel for evaluation with the APL and UC-Davis models.

A second, more streamlined and evolutionary strategy is to provide these models as tools for DoD software development teams to use as part of best practices to guide their development plans. This strategy would alleviate the exchange of data and would allow a more collaborative community effort to refine the models and resulting software development performance over time.

Appendix I: Acronyms and Glossary of Terms

Acronyms

- ACAT - acquisition category
- ACTD - advanced concept technology demonstration
- AI - artificial intelligence
- ATO - authority (or authorization) to operate
- CAPE - Cost Assessment and Performance Evaluation
- CFR - Code of Federal Regulations
- CI/CD - continuous integration/continuous delivery
- CIO - Chief Information Officer
- COCOM - combatant command
- COTS - commercial off-the-shelf
- DAU - Defense Acquisition University
- DDS - Defense Digital Service
- DFARS - Defense Federal Acquisition Regulation Supplement
- DIB - Defense Innovation Board
- DoD - Department of Defense
- DoDI - Department of Defense Instruction
- DoDIG - Department of Defense Office of Inspector General
- DOT&E - Director, Operational Test & Evaluation
- DSB - Defense Science Board
- DSS - Defense Security Service
- FACA - Federal Advisory Committee Act
- FAR - Federal Acquisition Regulation
- FARs - Federal Acquisition Regulations
- FFRDC - Federally-Funded Research and Development Center
- FFRDCs - Federally Funded Research and Development Centers
- FM - financial management
- FTEs - full-time equivalents
- GAO - General Accounting Office
- GOTS - government off-the-shelf
- GPU - graphics processing unit
- IT - information technology
- JCIDS - Joint Capabilities Integration and Development System
- JIDO - Joint Improvised-Threat Defeat Organization
- KO - contracting officer
- ML - machine learning
- MOS - military occupational specialty
- MVP - minimum viable product
- O&M - operations and maintenance
- OODA - Observe, Orient, Decide, and Act
- OSD - Office of the Secretary of Defense
- OT&E - Operation, Test & Evaluation

- OTA - Other Transaction Authority
- PAO - program acquisition office
- PM - program management
- PMO - program management office
- PPB&E - Planning, Programing, Budgeting and Execution
- R&D - research and development
- RDT&E - research, development, test, and evaluation
- RFP - request for proposals
- SAE - Service Acquisition Executive
- SE - systems engineering
- SWAP - software acquisition and practices 9study)
- T&E - Testing & Evaluation
- USC - United States Code
- USD(A&S) - Under Secretary for Defense (Acquisition and Sustainment)
- USD(C) - Under Secretary of Defense (Comptroller)
- USD(R&E) - Under Secretary of Defense (Research and Engineering)

Glossary

In this subsection we provide a short glossary of some of the terms that we use throughout the report. For each term we provide a short definition of that term, including references if it is a term used elsewhere, and then provide some context and motivation for the use of the term in this report.

Agile development [DSB00]**.** Agile development, also called "iterative" development, begins with the creation of a software factory. Development and testing sprints—a set period of time during which specific work is completed—allow a team to do rapid iterations of development, obtain user feedback, and adjust goals for the next increment. This framework allows for continuous development throughout the life of the product.

ATO (authorization to operate). Formal declaration by a Designated Approving Authority (DAA) that authorizes operation of an IT system and explicitly accepts the risk to agency operations. Obtaining an ATO is required under the Federal Information Security Management Act (FISMA) of 2002 and regulated by Federal Government and DoD guidance that specifies the minimum security requirements necessary to protect Information Technology (IT) assets.

Business systems [Sec 1.2]**.** Essentially the same as enterprise systems, but operating at a slightly smaller scale (e.g., for one of the Services). Like enterprise systems, they are interoperable, expandable, reliable, and probably based on commercial offerings. Similar functions may be customized differently by individual Services, though they should all interoperate with DoD-wide enterprise systems. Depending on their use, these systems may run in the cloud, in local data centers, or on desktop computers. Examples include software development environments and Service-specific HR, financial, and logistics systems.

CI/CD (continuous integration/continuous delivery). Continuous integration (CI) is the practice of merging all software developer working copies of code to a shared master development branch on a continuous basis. Continuous delivery is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. The combination of continuous integration and continuous delivery is a common feature of DevOps (and DevSecOps) development environments.

Cloud computing [Sec 1.2]. Computing that is typically provided in a manner such that the specific location of the compute hardware is not relevant (and may change over time). These systems will typically be running on commercial hardware and using commercial operating systems, and the applications running on them will run even as the underlying hardware changes. The important point here is that the hardware and operating systems are generally transparent to the application and its user.

Client/server computing [Sec 1.2]. Computing provided by a combination of hardware resources available in a computing center (servers) as well as local computing (client). These systems will usually be running on commercial hardware and using commercial operating systems.

Combat systems [Sec 1.2]. Software applications that are unique to the national security space and used as part of combat operations. Combat systems may require some level of customization that may be unique to DoD, not the least of which will be specialized cybersecurity considerations to enable them to continue to function during an adversarial attack. (Note that since modern DoD enterprise and business systems depend on software, cyber attacks to disrupt operations have the potential to be just as crippling as those aimed at combat systems.)

Desktop/laptop/tablet computing [Sec 1.2]. Computing that is carried out on a single system, often by interacting with data sources across a network. These systems will usually be running on commercial hardware and using commercial operating systems.

DevSecOps. "DevOps" represents the integration of software development and software operations, along with the tools and culture that support rapid prototyping and deployment, early engagement with the end user, and automation and monitoring of software, and psychological safety (e.g., blameless reviews). "DevSecOps" is a more recent term that reflects the importance of integrating security into the DevOps cycle (and not bolting on security at the end). DevOps development is closely related to agile development and the two are often used interchangeably. The term DevSecOps places more focus on security as a critical element. More information: https://tech.gsa.gov/guides/understanding_differences_agile_devsecops/.

Figure I.1. Continuous integration of development, security, and deployment (DevSecOps). [Adapted from an image by Kharnagy, licensed under CC BY-SA 4.0]

DevSecOps techniques should be adopted by DoD, with appropriate tuning of approaches used by the Agile/DevOps community for mission-critical, national security applications. Open source software should be used when possible to speed development and deployment, and leverage the work of others. Waterfall development approaches (e.g., DOD-STD-2167A) should be banned and replaced with true, commercial agile processes. Thinking of software "procurement" and "sustainment" separately is also a problem: software is never "finished" but must be constantly updated to maintain capability, address ongoing security issues and potentially add or increase performance.

Moving to a DevSecOps software development approach will enable DoD to move from a specify, develop, acquire, sustain mentality to a more modern (and more useful) create, deploy, scale, optimize mentality. Enabling rapid iteration will create a system in which the US can update software at least as fast as our adversaries can change tactics, allowing us to get inside their OODA loop.

Digital Infrastructure. Enterprise-scale computing hardware and software platforms that enable rapid creation and fielding of software. Critical elements include:

- *Scalable compute*: elastic mechanisms to provide any developer with a powerful computing environment that can easily scale with the needs of an individual programmer, a product development team, or an entire organization enterprise.

- *Containerization*: sandbox environments that "package up" an application or microservices with all of the operating system services required for executing the application and allowing that application to run in a virtualized segmented environment.

- *Continuous integration/continuous delivery (CI/CD) pipeline*: platform for automated testing, security, and deployment of software, including licenses access for security tools and a centralized artifacts repository of containers with tools, databases, and operating system images.

- *Automated configuration, updating, distribution, and recovery management:* automated processes that use machine-readable definition files (stored in the same source code repository as your software source code) to manage and provision environments, containers, virtual machines, load balancing, networking, access rules, and other components.

- *Federated identity management and authentication*: common identity management for accessing information across multiple systems and allows rapid and accurate auditing of code.

- *Firewall configuration and network access control lists*: forces information transfer only through intentional interfaces to reduce the attack surface and make system servers more resilient against penetration.

- *Common information assurance (IA) profiles*: Common IA profiles integrated into the development environment and part of the development system architecture are less likely to have bugs than customized and add-on solutions.

- *Modeling and simulation capability:* The use of high fidelity simulations and digital models, enables software developers to develop and validate software more quickly with greater reliability (see also *digital twin*).

Currently, DoD programs each develop their own development and test environments, which requires redundant definition and provisioning, replicated assurance, including cyber, and extended lead times to deploy capability. Digital infrastructure, common in commercial IT, is critical to enable rapid deployment at the speed (and scale) of relevance. The Services and defense contractors will need to build on a common set of tools (instead of inventing their own) *without* just requiring that everyone use one DoD-wide (or even service-wide) platform.

Digital twin. A digital twin is a digital synthetic representation of a system or capability. Digital twins are useful in concept development, designing, developing, testing, and validation of software. The use of high fidelity simulations and digital models enables software developers to develop and validate software more quickly with greater reliability. In the future, as we leverage the use of Machine Learning (ML) and Artificial Intelligence (AI) in software design, development, and test, the ability to leverage simulation and modeling will be critical. For example, today, in the commercial world where self-driving cars are being pioneered, sensors are used to collect data on millions of miles of roads. Before software updates are pushed to the autonomous driving cars and before the first mile is every driven on real roads, the software "drives" those millions of miles through simulation. It is important that the Department and our defense industrial base develop and support similar capabilities to that of commercial industry.

Embedded computing [Sec 1.2]**.** Computing that is tied to a physical, often-customized hardware platform and that has special features that requires careful integration between software and hardware.

Enduring capability. Refers to a class of mission software needs that will persist for the foreseeable future and should be budgeted and managed as an ongoing level of effort with a portfolio management approach to balance—in real time—maintenance, upgrades and major new functionality. An example is the acquisition, processing and distribution of data and information from overhead assets which, when separated from the sensor and satellite programs to which each iteration is traditionally attached, is an area of investment we will always be making.

Throughout this report we make reference to the modern view of software as a continuously, incrementally delivered capability and we use that definition to drive many of the recommendations we propose, especially around the use of DevSecOps. This view is characterized by rapid user feedback loops and continuous deployment to deal with that feedback and with such "maintenance" functions as cyber protection, operating system upgrades, etc. This is the overall vision we espouse for the acquisition and delivery of most types of software—think about the software to deliver spare parts management for a fighter fleet, the software to manage the movement of service personnel and their families, or the software to provide tanker scheduling for a combat air fleet in an AOR.

We believe it is also important to look at certain kinds of software that will need to be delivered against a *mission* need that will persist for long enough into the future that we should think about it as an *enduring capability* need. A good example of an enduring capability is the processing, exploitation, and distribution (PED) software that ingests data from multi-domain overhead assess, processes that data into a series of information products and makes those products available to a wide array of global users. Satellites will change, sensors will change, and the kinds of analyses will change, but the underlying software to process this chain will endure.

Historically PED has been mapped to new or upgraded satellite launches—new satellite, new ground station—and as such are mapped to long cycle times, large, non-incremental programs and oversized budgets broken into the traditional buckets of R&E, acquisition and maintenance. A different model would be to recognize the enduring need for PED capability, fund as a stable ongoing effort, manage the capability through an integrated program team/PEO responsible in real time for the portfolio trades between fixes, upgrades and new capabilities. The core is to separate software from the hardware platforms that provide it data and from the downstream systems that consume the output of the software, recognize that this software need will persist for the foreseeable future, and fund and manage the program in this fashion.

Enterprise systems [Sec 1.2]. Very large-scale software systems intended to manage a large collection of users, interface with many other systems, and generally be used at the DoD level or equivalent. These systems should always run in the cloud and should use architectures that allow interoperability, expandability, and reliability. In most cases the software should be commercial software purchased (or licensed) without modification to the underlying code, but with DoD-specific configuration. Examples include email systems, accounting systems, travel systems, and HR databases.

Logistics systems [Sec 1.2]. Any system that is used to keep track of materials, supplies, and transport as part of operational use (versus Service-scale logistics systems, with which they should interoperate). While used actively during operations, logistics systems are likely to run on commercial hardware and operating systems, allowing them to build on commercial off-the-shelf (COTS) technologies. Platform-based architectures enable integration of new capabilities and functions over time (probably on a months-long or annual time scale). Operation in the cloud or based on servers is likely.

Mission systems [Sec 1.2]. Any system used to plan and monitor ongoing operations. Similar to logistics systems, this software will typically use commercial hardware and operating systems and may be run in the cloud, on local services, or via a combination of the two (including fallback modes). Even if run locally (such as in an air operations center), they will heavily leverage cloud technologies, at least in terms of critical functions. These systems should be able to incorporate new functionality at a rate that is set by the speed at which the operational environment changes (days to months).

Mobile computing [Sec 1.2]. Computing that is carried out on a mobile device, usually connected to the network via wireless communications. These systems will usually be running on commercial operating systems using commodity chipsets.

MVP (minimum viable product). A minimum viable product is a first iteration of a software project that has just enough features to meet basic minimum functionality. It provides the foundational capabilities upon which improvements can be made. The goal of an MVP is to quickly get basic capabilities into users hands for evaluation and feedback.

Security-at-the-perimeter. An approach to security that relies on perimeter access control as the primary mechanism for protecting against intrusion.

Software-defined systems. Software-defined systems make use of the increased capability of digital computing to carry out functions that are traditionally associated with hardware. Examples include software-defined radios and software-defined networking.

Software factory [DSB18]. **A** set of software tools that programmers use to write their code, confirm it meets style and other requirements, collaborate with other members of the programming team, and automatically build, test, and document their progress. This allows teams of programmers to do iterative development with frequent feedback from users.

Technical debt. The cost that is incurred by implementing a software solution that is expedient rather than choosing a better approach that would take longer. Technical debt often accrues over the life of a program as code is expanded and patched. Technical debt can often be "paid down" by investing in refactoring or re-architecting the code.

Unit testing. A software testing method in which software programs, modules, or functions tested to determine whether they satisfy a desired set of specifications, typically by testing a large number of individual tests cases (unit test). Unit testing provides a means of detecting when errors have been inadvertently introduced into a code base.

Weapons system [Sec 1.2]. Any system that is capable of the delivery of lethal force, as well as any direct support systems used as part of the operation of the weapon. Note that our definition differs from the standard DoD definition of a weapons system, which also includes any related equipment, materials, services, personnel, and means of delivery and deployment (if applicable) required for self-sufficiency. The DoD definition would most likely include the mission and logistics functions, which we find useful to break out separately. Software on weapons systems is traditionally closely tied to hardware, but as we move to greater reliability of software-defined systems and distributed intelligence, weapons systems software is becoming increasingly hardware independent (similar to operating systems for mobile devices, which run across many different hardware platforms).

Catch Phrases

Self denial of service attack. Not letting your organization make use of tools or processes that are available to others.

Staple test. Any report that is going to be read should be thin enough to be stapled with a regular office stapler. A standard office stapler is able to staple 25 sheets of paper together => staple test limit is ~50 pages (but you can get a bit more if you bend over the staples manually).

Takeoff test. Reports should be short enough to read during takeoff, before the movies start and drinks are served (assuming you got upgraded). The average time from closing the door to hitting

10,000 ft (wifi on) at IAD is 25 minutes (15 taxi + 10 cruise). Average reading time for a page is 2 minutes => takeoff test limit is ~12 pages.

Waterfall with sprints. A too common approach to implementing agile development principles in a DoD environment. Development teams work on a rapid sprint cycle and deliver code into a test environment that takes months to complete (versus actual agile, where code would be released to users at the end of the spring).

Appendix J: Study Information

Study Membership

CHAIRS

Richard M. Murray
California Institute of Technology

J. Michael McQuade
Carnegie Mellon University

MEMBERS

Milo Medin
Google

Jennifer Pahlka
Code for America

Trae' Stephens
Founders Fund

Gilman Louie
Alsop Louie Partners

GOVERNMENT ADVISORS

Jeff Boleng
Special Assistant for Software Acquisition

Bess Dopkeen (Jan '18-Jan '19)
Cost Assessment/Program Eval

FEDERALLY FUNDED RESEARCH AND DEVELOPMENT CENTERS SUPPORT

Forrest Shull
Software Engineering Institute

Kevin Garrison
Institute for Defense Analyses

Craig Ulsh
MITRE

Nicolas Henri-Guertin
Software Engineering Institute

DATA ANALYSES AND MACHINE LEARNING SUPPORT

John Piorkoski
Johns Hopkins University Applied Physics Lab

Steven Lee
Rotunda Solutions

Linda Harrell
Johns Hopkins University Applied Physics Lab

Vladimir Filkov
University of California Davis

Yun Kim
Tecolote

Tom Schaefer
Tecolote

Kevin Jackameit
Tecolote

Dave Zubrow
Software Engineering Institute

STUDY SUPPORT

Courtney Barno
Artlin Consulting

Devon Hardy
Artlin Consulting

Sandra O'Dea
E-3 Federal Solutions

Software Acquisition and Practices (SWAP) Working Group
Subgroup Participation

<u>Acquisition Strategy</u>
Melissa Naroski Merker (lead)
Larry Asch
Jeff Boleng
Nicolas Chaillan
COL Harry Culclasure
Ben FitzGerald
Nick Kosmidis
Jonathan Mostowski
Nick Tsiopanas

<u>Appropriations</u>
Jane Rathbun (lead)
Maj Gen Patrick Higby
Paul Hullinger
Melissa Naroski Merker
Shannon McKitrick

<u>Contracts</u>
Jonathan Mostowski (lead)

<u>Data & Metrics</u>
Ben FitzGerald (lead)
Jeff Boleng
Victoria Cuff
Bess Dopkeen
Jon Engelbrektson
Mark Krzysko
Melissa Naroski Merker
John Seel
David Zubrow

<u>Infrastructure</u>
Jeff Boleng (co-lead)
John Bergin (co-lead)
Nicolas Chaillan
Victoria Cuff
Robert Gold
Amy Henninger
Richard Kutter

<u>Infrastructure (Cont.)</u>
Jane Rathbun
John Rusnak
Zack Schiller
Philomena Zimmerman

<u>Requirements</u>
Fred Gregory (lead)
Jeff Boleng
Victoria Cuff
Jennifer Edgin
Donald Johnson
Margaret Palmieri
Owen Seely
Philomena Zimmerman

<u>Security & Accreditation</u>
Leo Garciga (lead)
Jeff Boleng
Nicolas Chaillan
Amy Henninger
Maj Gen Patrick Higby
Ana Kreiensieck
Nicolas Lanham
Tom Morton

<u>Sustainment & Modernization</u>
Kenneth Watson (lead)
Stephen Michaluk
Bernard Reger

<u>Testing & Evaluation</u>
Greg Zacharias (lead)
Chad Bieber
Chris DeLuca
Amy Henninger
Lt Col Mark Massaro
Ryan Norman
Tom Simms
Heather Wojton
Philomena Zimmerman

Defense Innovation Board (DIB), Software Acquisition and Practices (SWAP)
Study Team and Support Team Site Visits

As part of its data gathering activities, the SWAP study team visited a cross-section of ongoing software programs (both business and weapon systems) across DoD and the Services. Despite their demanding schedules, program managers and their teams (civilian and contractor) welcomed members of the study team and shared their valuable experiences in software acquisition and development, testing, and security. The knowledge gained from these collaborative sessions provided tremendous input into the study and the development of the final recommendations. The SWAP study team would like to thank all those individuals who participated in these site visits for their invaluable contribution to this study.

Date	Companies/Organizations	Locations
Mar 2018	Lockheed Martin	Fort Worth, TX
Apr 2018	Pivotal, Raytheon	Boston, MA
Aug 2018	Raytheon	Los Angeles; Aurora, CO
Aug 2018	SPAWAR, ARL	Colorado Springs, CO
Sep 2018	Lockheed Martin	Moorestown, NJ
Oct 2018	Leidos, Cerner	Rosslyn, VA
Nov 2018	Raytheon	Tucson, AZ

Special thanks to: Samantha Betting, Richard Calabrese, Tory Cuff, RDML Tom Druggan, Lt Col Thomas Gabriele, Leo Garciga, Jack Gellen, Arturo Gonzalez, Jill Hardash, Brian Henson, Cori Hughes, Lisa Jollay, CAPT Bryan Kroger, Col Jennifer Krolikowski, Lt Col Jason Lee, Myron Liszniansky, Maj Zachary McCarty, Lt Col Steve Medeiros, Kenneth Merchant, Anna Nelson, David Norley, Scott Paulsen, Kelci Pozzi, Sandy Scharn-Stevens, Terry Schooley, Thomas Scruggs, Lt Col Kenneth Thill, and Eric Todd

Government and Supplemental Program Meetings

In addition to conducting site visits, the SWAP study team engaged with a broad spectrum of offices within DoD and the Services that possess ownership of the regulations and policies that relate to the software acquisition and/or development life cycle and their associated challenges. In the spirit of practicing an agile methodology, these regular collaborative sessions resulted in cyclic user feedback. The meetings listed below are not exhaustive, but we aimed to capture the wide array of offices that provided feedback to the SWAP study team, highlighting the myriad and assorted offices within DoD that are intertwined with software.

3 – 4 APRIL 2019 MEETINGS

Office of the Under Secretary of Defense (Acquisition & Sustainment)

Office of the Under Secretary of Defense (Comptroller) & Chief Financial Officer

Department of Defense Chief Information Officer

Office of the Director, Operational Test and Evaluation

Office of the Secretary of the Air Force/AQR - Science, Technology, and Engineering

Assistant Secretary of the Navy for Research, Development and Acquisition
Office of the Chief of Naval Operations

Office of the Chief Information Officer (CIO)/G6, Department of the Army

20 – 22 MARCH 2019 MEETINGS

Office of the Secretary of Defense

Office of the Under Secretary of Defense (Acquisition & Sustainment)

Office of the Director, Operational Test and Evaluation

Cost Assessment and Program Evaluation

Office of Personnel & Readiness

Office of the Under Secretary of Defense (Research & Engineering)

Department of Defense Chief Information Officer

4 – 6 DECEMBER 2018 MEETINGS

Office of the Under Secretary of Defense (Comptroller) & Chief Financial Officer

Office of the Director, Operational Test and Evaluation

19 NOVEMBER 2018 MEETING

Supplemental Program Session

Lockheed Martin

Air Force Life Cycle Management Center
Office of the Secretary of the Air Force/Acquisition

24 SEPTEMBER 2018 MEETING

Office of the Under Secretary of Defense (Acquisition & Sustainment)

Office of the Under Secretary of Defense (Research & Engineering)

2 OCTOBER 2018 MEETING

Supplemental Program Session

Lockheed Martin

Naval Sea Systems Command

17 AUGUST 2018 MEETING

Air Force Materiel Command/Air Force Life Cycle Management Center

27 – 28 FEBRUARY 2019 MEETINGS

Joint Rapid Acquisition Center

Office of Personnel & Readiness

Defense Digital Services

Defense Security Cooperation Agency

17 – 18 JANUARY 2019 MEETINGS

Office of the Under Secretary of Defense
(Acquisition & Sustainment)

Office of the Under Secretary of Defense
(Research & Engineering)

Department of Defense Chief Information Officer

Office of the Under Secretary of Defense
(Comptroller) & Chief Financial Officer

Cost Assessment and Program Evaluation

Office of the Chief Management Officer

Office of the Secretary of the Air Force for
Acquisition, Technology, & Logistics

Office of the Secretary of the Navy
Office of the Assistant Secretary of the Navy for
Research, Development, & Acquisition - Command,
Control, Communications, Computers, Intelligence

Representatives from Industry

23 JULY 2018 MEETING

U.S. Navy

Naval Air Warfare Center Weapons Division

3 JULY 2018 MEETINGS

Congressional Research Service

U.S. Army Contracting Command

Charge from Congress

2018 NATIONAL DEFENSE AUTHORIZATION ACT

SEC. 872. DEFENSE INNOVATION BOARD ANALYSIS OF SOFTWARE ACQUISITION REGULATIONS.

(a) STUDY.—

(1) IN GENERAL.—Not later than 30 days after the date of the enactment of this Act, the Secretary of Defense shall direct the Defense Innovation Board to undertake a study on streamlining software development and acquisition regulations.

(2) MEMBER PARTICIPATION.—The Chairman of the Defense Innovation Board shall select appropriate members from the membership of the Board to participate in the study, and may recommend additional temporary members or contracted support personnel to the Secretary of Defense for the purposes of the study. In considering additional appointments to the study, the Secretary of Defense shall ensure that members have significant technical, legislative, or regulatory expertise and reflect diverse experiences in the public and private sector.

(3) SCOPE.—The study conducted pursuant to paragraph (1) shall—

(A) review the acquisition regulations applicable to, and organizational structures within, the Department of Defense with a view toward streamlining and improving the efficiency and effectiveness of software acquisition in order to maintain defense technology advantage;

(B) review ongoing software development and acquisition programs, including a cross section of programs that offer a variety of application types, functional communities, and scale, in order to identify case studies of best and worst practices currently in use within the Department of Defense;

(C) produce specific and detailed recommendations for any legislation, including the amendment or repeal of regulations, as well as non-legislative approaches, that the members of the Board conducting the study determine necessary to—

(i) streamline development and procurement of software;

(ii) adopt or adapt best practices from the private sector applicable to Government use;

(iii) promote rapid adoption of new technology;

(iv) improve the talent management of the software acquisition workforce, including by providing incentives for the recruitment and retention of such workforce within the Department of Defense;

(v) ensure continuing financial and ethical integrity in procurement; and

(vi) protect the best interests of the Department of Defense;
and

(D) produce such additional recommendations for legislation as such members consider appropriate.

(4) ACCESS TO INFORMATION.—The Secretary of Defense shall provide the Defense Innovation Board with timely access to appropriate information, data, resources, and analysis so that the Board may conduct a thorough and independent analysis as required under this subsection.

(b) REPORTS.—

(1) INTERIM REPORTS.—Not later than 150 days after the date of the enactment of this Act, the Secretary of Defense shall submit a report to or brief the congressional defense committees on the interim findings of the study conducted pursuant to subsection (a). The Defense Innovation Board shall provide regular updates to the Secretary of Defense and the congressional defense committees for purposes of providing the interim report.

(2) FINAL REPORT.—Not later than one year after the Secretary of Defense directs the Defense Advisory Board to conduct the study, the Board shall transmit a final report of the study to the Secretary. Not later than 30 days after receiving the final report, the Secretary of Defense shall transmit the final report, together with such comments as the Secretary determines appropriate, to the congressional defense committees.

[redacted]

THE UNDER SECRETARY OF DEFENSE
3010 DEFENSE PENTAGON
WASHINGTON, DC 20301-3010

APR 05 2018

ACQUISITION
AND SUSTAINMENT

MEMORANDUM FOR CHAIRMAN, DEFENSE INNOVATION BOARD

SUBJECT: Terms of Reference - Establishment of the Software Acquisition and Practices Subcommittee of the Defense Innovation Board

Today's advances in software are pushing new frontiers in lethality, speed, precision, accuracy, and efficiency. The Department of Defense's (DoD) ability to field and sustain weapon systems will increasingly depend on its ability to upgrade, develop, and deploy software or acquire commercial software. The technology and business of software development has undergone a radical transformation over the last decade, yet DoD's approach to assess and acquire commercial off-the-shelf software (COTS) products, use and improve existing Government off-the-shelf (GOTS) software, further develop commercial software products to meet unique government needs, or independently develop software products has changed little. This stymies progress and represents significant risk. Software is increasingly the decisive factor in determining the capabilities of modern weapon systems and is often the limiting factor for integrating sensors, platforms, and weapons. For these reasons, an analysis of the DoD's software development and acquisition practices across the range of business and weapon systems is urgently needed as part of the DoD's broader efforts at modernization and reform.

Modernizing the DoD's approach to software development and acquisition has the potential to accelerate fielding of new capabilities, reduce cost, and increase the lethality of our forces. Failure to modernize also carries costs, perpetuating the often slow, unwieldly, requirements-driven approach to software that no longer serves the warfighter or taxpayer well. Moreover, as the field of artificial intelligence progresses, employing rapid, iterative software development, as well as leveraging COTS and GOTS alternatives, will provide critical warfighting capabilities and competitive advantages.

Section 872 of the National Defense Authorization Act (NDAA) for Fiscal Year (FY) 2018 (Public Law 115-91), requires the Secretary of Defense to direct the Defense Innovation Board (DIB) to undertake a study on streamlining software development and acquisition regulations. The Secretary of Defense delegated this authority to the undersigned on 2 January 2018. As such, I am establishing the Software Acquisition and Practices (SWAP) Subcommittee of the DIB to undertake a data-driven analysis of how DoD develops, acquires, and employs software technologies and capabilities.

The NDAA for FY 2018 stipulates that the study must:
(1) Review the acquisition regulations applicable to, and organizational structures within, DoD with a view toward streamlining and improving the efficiency and effectiveness of software acquisition in order to maintain defense technology advantage;
(2) Review ongoing software development and acquisition programs, including a cross section of programs that offer a variety of application types, functional communities, and ***scale, in*** order to identify case studies of best and worst practices currently in use within DoD;

[redacted]

(3) Produce specific and detailed recommendations for any legislation, including the amendment or repeal of regulations, as well as non-legislative approaches, that the members of the Board conducting the study determine necessary to–

(a) Streamline development and procurement of software;

(b) Adopt or adapt best practices from the private sector applicable to Government use;

(c) Promote rapid adoption of new technology;

(d) Improve the talent management of the software acquisition workforce, including by providing incentives for the recruitment and retention of such workforce within DoD;

(e) Ensure continuing financial and ethical integrity in procurement; and

(f) Protect the best interests of DoD; and

(4) Produce such additional recommendations for legislation as such members consider appropriate.

The SWAP Subcommittee will provide its recommendations for any legislation, including the amendment or repeal of regulations, and actions to be considered by DoD to the DIB for full and thorough public deliberation and approval. The DIB will submit an interim report to my office not later than May 11, 2018, and a final report not later than April 5, 2019, reporting directly back to me on the study's progress as appropriate.

In conducting its work, the DIB and its subcommittees have my full support in all requests for information, data, resources, and analysis that may be relevant to its research and fact-finding under this Terms of Reference so that the DIB may conduct a thorough and independent analysis as required by section 872 of the NDAA for FY 2018. As such, the Office of the Secretary of Defense, Component Heads, and the Military Departments are directed to promptly facilitate the work of the DIB and the SWAP Subcommittee by ensuring that the DIB staff and members have timely access to any relevant personnel and information necessary to perform their duties consistent with the requirements and limitations of existing law that may be applicable.

As a subcommittee of the DIB, the SWAP Subcommittee shall not work independently of the DIB's charter and shall report its recommendations to the full DIB for public deliberation and approval, pursuant to the Federal Advisory Committee Act of 1972, as amended, the Government in the Sunshine Act of 1976, as amended, and other applicable Federal statutes and regulations. The SWAP Subcommittee does not have the authority to make decisions on behalf of the DIB nor can it report directly to any Federal representative. The members of the SWAP Subcommittee and the DIB are subject to title 18, United States Code, section 208, which governs conflicts of interest.

Ellen M. Lord

CONTACT

Jeff Boleng | SWAP Study Program Manager

jeffrey.l.boleng.civ@mail.mil | Office (703) 571-9029

SWAP Study Staff

Courtney Barno | courtney.e.barno.ctr@mail.mil | (703) 614-5147

Devon Hardy | devon.k.hardy2.ctr@mail.mil | (703) 614-5148

Sandy O'Dea | Sandra.l.odea.ctr@mail.mil | (703) 614-5139