Software Defines Tactics: Fielding Transformational Capability without Transforming Acquisition

TRANSCRIPT

Discussion

- **Jason Weiss**, Chief Software Officer, Conquest Cyber
- **James Geurts**, Executive Vice Chairman, Sarcos Technology and Robotics and former Assistant Secretary of the Navy for Research, Development, and Acquisition
- **Dan Patt**, Senior Fellow, Center for Defense Concepts and Technology
- **Bryan Clark**, Senior Fellow and Director, Center for Defense Concepts and Technology

Disclaimer: This transcript is based off of a recorded video conference and breaks in the stream may have resulted in mistranscriptions in the text.

A video of the event is available: [https://youtu.be/YEENoM2mzrM](https://youtu.be/YEENoM2mzrM)

About Hudson Institute: Founded in 1961 by strategist Herman Kahn, Hudson Institute challenges conventional thinking and helps manage strategic transitions to the future through interdisciplinary studies in defense, international relations, economics, health care, technology, culture, and law.

Hudson seeks to guide public policy makers and global leaders in government and business through a vigorous program of publications, conferences, policy briefings, and recommendations.
Bryan Clark:

Welcome to the Hudson Institute. I'm Bryan Clark, a senior fellow at the institute and Director of the Hudson Center for Defense Concepts and Technology. Today we've got an outstanding panel to have a discussion about Hudson's latest report, “Software Defines Tactics,” which talks about how we should be structuring military software acquisitions for adaptability and advantage in a competitive era. The authors of that report are here with us today. Jason Weiss was the Department of Defense's Chief Software Officer. He is now at Conquest Cyber, where he is also the Chief Software Officer. While at DoD, Jason was pivotal in implementing the DoD's new software modernization strategy and many of the implementation thereof.

Also, Dan Patt is one of the authors of that report as well. He is a senior fellow at Hudson, where he works in the Defense Concepts and Technology Center along with me. Right now, he supports strategy at the artificial intelligence company STR, as well as supporting Thomas H. Lee Partners' automation and technology investment practice. Also with us for this discussion is Jamie "Hondo" Geurts, who most recently served doing the duties of the undersecretary of the Navy, but also served as the assistant secretary of the Navy for research development and acquisition, which capped a 34-year career in government, where he served in the Air Force, as well as the SOCOM, or special operations command chief acquisition officer, and a number of other positions. Gentlemen, thank you very much for being here with us today to talk about the report Software Defines Tactics.

Jason Weiss:

Yeah. Excited to have this discussion, Bryan. Thank you.

Bryan Clark:

We are as well. Let me just start off here with an opening question that can allow you all to talk a little bit about what the challenges are that the DoD is facing, and some of the opportunities that are highlighted by this report. Marc Andreessen famously wrote more than a decade ago, "Software was eating the world." Now, in the commercial sphere, that's clearly been the case. We've seen the impact of software on commercial and consumer electronics. Every product we buy has an enormous software component that is often managed and updated from outside of our devices. DoD maybe has been slow to come to this realization and embrace the change that software is bringing to military capabilities. Military systems clearly have a major software component, but hardware, in a lot of ways, still defines military capability and what drives the acquisition process.

I'm going to ask each of you, why has it taken so long for DoD to really embrace the idea that software is a defining characteristic military capability and start to change the way that it both buys and develops military capabilities in response to that? I'll start with you, Jason, since you're the lead author on the report. Why do you think this has taken so long?

Jason Weiss:

Well, I think part of it, Bryan, is what I would term bureaucracy blindness. Obviously, the large workforce across the DoD, our servicemen and women, they have mobile phones. They use these services that the commercial industry are putting out. It isn't like there's this era of ignorance where we don't know about these capabilities, that we don't know they exist. We
clearly do know they exist. But I think when you start to look at the bureaucracy of the DoD and the way that we think about our budget cycles, the way that we look at the FYDP, the way that we talk about multi-year execution plans, even the way we approach research and engineering, is a different construct than what we see taking place over in the commercial sector. So the DoD does, I think, have a recognition that software started to eat the world, but the ability to pivot in turn and begin to embrace the impacts, the positive benefits that start to come from these cascading capabilities, was a bit slow.

Bryan Clark:

Dan is the other co-author on this paper. You obviously deal with this challenge, and dealt with it in your time at DARPA as the Deputy Director of the Strategic Technologies Office. Why do you think it's taken so long for DoD to really wrap its head around how to deal with software as an independent capability source from hardware? Why are we not taking advantage of the opportunity that software might present?

Dan Patt:

Yeah. Yeah. I'll try to sharpen what I think some of what's new here, really, is internet scale complexity. It's not just that software is everywhere. It's that these systems are interconnected, and then there's all these cascading impacts. Just to be clear, it's not like all of the commercial world does amazing things. This has been a struggle in the commercial space, too. So often, when we speak in glowing terms about where the commercial world does this well, we point at things like the internet giants who've grown up trying to wrangle this complexity and actually figuring out ways of how to iterate between development and operations. They figured out methods to tackle this complexity.

DoD has been doing software a long time. You go back and look at '60s acquisition programs like the F-4 Phantom. There was a little bit of software in there, but it was constrained. It was in a few places. What's new is the fact that everything is coupled together, and I can't put everything into a requirement. I now have to couple these. I have to iterate between development and operations. That's really where we're struggling. It wasn't that long ago that General Electric talked about recasting itself as a software company. Now it's decided that it probably can't do that. DoD's in a similar place as the old industrial giants in industry. It's right in that place, except DoD has no choice but to figure out how to adapt and do this. Hopefully, that's what all of us on this call are trying to do, bring perspectives to bear about how DoD can be successful in doing that transformation.

Bryan Clark:

Yeah. Hondo, that brings us to your long experience in the acquisition world and the operational world, in both the Air Force and the Navy. You've grown up watching this evolution happen from a very hardware-centric force, to a more software-centric force, especially at SOCOM where capability development was happening much more rapidly. Why is it that maybe SOCOM is so far ahead of the rest of DoD at maybe embracing the idea that you can get into a development operations cycle and that you can take advantage of software to gain capability more quickly than you could with hardware alone?

James Geurts:
Yeah. Sure. I mean, it's a great question. I'm laughing because as a second lieutenant, the first training class I went to was a software acquisition class in the late '80s, because we back then knew we had a problem. In fact, I got recruited to come in the Air Force with a double E degree because I think they were hoping to bring some talent in. Listen, there's, I think, a couple different things that make it tough. Dan and Jason have alluded to it. One of them is just scale. The DoD and, quite frankly, the industrial base of the country largely is a World War II plus four percent industrial base. It's built on industrial practices. You design something. Then you hand it to blueprint people. Then they make blueprints. Then you produce it. Your goal was to produce as fast and cheap as you could.

When you're in industrial age thinking, it's about a serial process. Software network age thinking is much more in parallel, much more network, much more fluid. All of our processes, the way we think about planning, the way we budget programs, the way we artificially create design reviews, are all built on that industrial age thought, and we're pretty darn good at it. But that doesn't mean we're going to be relevant for the next 50 years using that same process.

The second piece, I think it's a scaling issue. The bureaucracies tend to want to do everything one way or another, and tend not to embrace being multivariate and so all of us multidextrous. We try and make software look like hardware. Then we flip the other way. We try and make hardware look like software. What I really like about what this report does is actually take that on. It's like logistics, which I really liked in the report. You can fly it. You can put it on a boat. You can put it on a train. Each has goods. Each has bads. But it's all about the outcome you're trying to achieve. I think if we focus more on the outcomes, which I think the heritage of special ops is all about an outcome, and not the activity or defining one way to do everything, we can get to the point where we're still going to have to be good at producing stuff, hardware stuff, really hard, hypersonic stuff. But we also got to be good at producing software. By the way, most of our industrial machines that produce hardware run on software. So it's everywhere.

Bryan Clark:

That's a great point, talking about outcomes as a way to measure progress as opposed to performance inputs, which is traditionally how the industrial model acquisition management process works, which brings up a point about how the commercial space has really maybe gone there first. Dan, to your point, commercial companies aren't really necessarily all good at this, but some are good at this. Jason, commercial companies have worked through the challenge of figuring out how to acquire software, how to pursue a process that's much more iterative and multivariate to talk about what Jamie's mentioning. How have they worked through that, and what could DoD learn from that commercial model of acquisition?

Jason Weiss:

Bryan, I think part of it stems from a lot of these companies that we point to as Northern Lights or lighthouses. They were born digital. They came up in an era where software is a service, where the idea of the Gantt chart, the Gantt chart was great for looking at tasks and coordinating how to move physical goods onto a floor at a particular time. But they recognized that in a software world, those tools from the 20th century aren't as valuable to a software ecosystem. I think they have an inherent advantage in that they were born digital. They think digital. We try to capture that in a number of places in the report. For example, they think about software first in terms of the interfaces that they need. We highlighted in the report as ACTs
Three, where you need to own your interface and your APIs more than the actual implementation of the code.

I think when we look at these tech giants today, they've recognized that owning that capability and thinking in service levels, being able to build upon and build upon and cascade and get bigger and bigger, is a huge competitive advantage that they have in what historically has been considered more of a physical market. If you look at Uber, they have more cars available to get people from point A to point B than any taxi system in the world. The same thing can be said for Airbnb. They have more rooms available per night that somebody can stay at than any hotel chain that's out there. Yet Uber doesn't own taxis, and Airbnb doesn't own hotels. They really have managed to think in service levels and interfaces and use that to construct new business models. It in terms of the DoD, I think that pivot, thinking about things in service level agreements, is a journey that we're currently undergoing.

**Bryan Clark:**

Dan, I'm to jump to you real quick. This raises something that you've brought up in previous reports that we've done, this idea that scale is something that DoD has alternately employed and not employed. MAST has been really important in some eras of military operations, less so maybe today in the current era of precision strike warfare. Do we need to be able to take advantage of software and what it can do? Do we need to re-embrace MAST and scale to be able to make the most of it?

**Dan Patt:**

Yeah. Look, I think that one of the really exciting things that's that we see emerging is the ability to innovate in, for lack of a better term, kill chains, how we conduct our missions and how we use technology together with that. I mean, this is something that we see the Ukrainians do, improvising these battle networks with Starlink and their home-developed applications and allied support and open source intelligence. There's not a program office supporting them. There are people who act as acquisition agents, but people are aligned around outcomes. The same that Hondo talked about. They're aligned around mission. What's new now is you don't necessarily always need highly specialized skills to bring this together. There's a role for that. I mean, there is a role for highly specialized, industrial-based talent on these really high-end mission systems that nobody else can do. There's a role. But that's not the whole game.

There's this whole spectrum of what you can do. Part of what's new is taking advantage of all of this customization that you can do. Let's get both scale, let's get exquisite systems, and let's push out the ability to innovate in how it comes together, and how you come up with those tactics, how you come up with those, let's say, users innovate on how they get their mission done. That's what's exciting. That's what's new. There's a real opportunity for US advantage there that plays to our institutional and cultural strengths.

**Bryan Clark:**

The scale comes into play for the variety of combinations and force compositions that the Ukrainians, for example, could put together.

**Dan Patt:**
Absolutely.

**Bryan Clark:**

It seems like, also, they take advantage of commercial capabilities, which, again, gives you a lot of scale that you wouldn't have if you were just using exquisite military systems.

**Dan Patt:**

That's right. They can inherit the scale of somebody else's communication system and somebody else's sensor network. Then they can harness that to develop more offensive options.

**Bryan Clark:**

Hondo, when you look at the model or the example of Ukraine and how they're innovating their battle networks on the fly, and how they're essentially using software to define the services that are going to be provided to integrate forces in the field, can DoD do this within its existing acquisition authorities? Can DoD adopt these kinds of models?

**James Geurts:**

Yep.

**Bryan Clark:**

If so, what are the challenges going to be in order to make this happen? I'm sure there's a lot of impediments that are not just legal and regulatory.

**James Geurts:**

Yeah, no. I think there are. I mean, a couple things. I worry a little bit if we overplay the Ukrainian ... They're very creative, very innovative. It heartens my SOCOM soul to see what they're doing. But the challenge, largely in modern trench warfare, with very isolated lines, is not the challenge we're going to be facing going forward. I think we got to be a little cautious that we think, just because we can use a commercial SATCOM and codge together a couple apps, that's going to help us in a Chinese thing. I do think there is a scaling issue we have to confront with China. It's how do we take that ingenuity and apply it at scale? That's what we're really talking about here.

I think, largely, we always want to go find the perfect process or the perfect authority. Quite frankly, we have the authorities we need. We know how to do this if we would just ... I would say almost unleash the hounds. A lot of what I liked about Jason and Dan's view of this was looking at the Program Executive Officer as really the battle space owner. Instead of just asking the PEO to respond to a piece of paper a requirements officer wrote five years ago, by the time he made it through the system, really focus on outcomes. I've seen if you empower teams, you tell them the outcomes you want, and we get much more fluid, you can have great things happen.

Now, again, we got to remember to differentiate the work. I don't want to be changing a nuclear reactor 24 hours at a time. There's some things where we do want very thoughtful, long
development programs because we're inventing new things, or the risk to failure is so high. But that's a part of the work, not all of the work. I think if you start empowering teams to differentiate the work and then pick the right work or the right tool for the work you're trying to get done, then that's how we get scale, because now, rather than over bureaucratically attack it, we empower and hold folks accountable for outcomes.

I think there's a lot more available to us than we sometimes give. Now we have a culture, we have a practice, and we've got to get over those cultures and tribal practices. I think some of the very simple North Stars that are laid out in this report would help me back if I was a PEO and wanted to tackle this. "Okay. These are five or six things I need to know how to do. Here's some ways to go at it. Now let me go take on the problem with that new knowledge and idea set."

Bryan Clark:

Does that mean that we need to rethink how we define your requirements? Dan and I talk to PEOs pretty often. Their response is, "Well, I'm pretty hemmed in because my requirements are actually very strictly defined. I don't have a lot of flexibility." Is that part of what has to happen here?

James Geurts:

Yeah. Absolutely. I mean, software development is production. You don't have this create a blueprint, and then send it down to the shop floor and have them create 10,000 of them, which is an industrial age thinking. It's very much more network thinking. What's the product I need to get out in the field? What does it have to do? Link the right architecture so I can add features as I go. A much more fluid process. Now, of course, that goes against our practice of defining to the Nth degree the program and defining milestones. We have a very industrial age architecture that we're trying to bring this into. If we can differentiate the work and find the right ways to oversee that work, as opposed to industrial age work, we've got to be able to do both. I think if we can do that, then we can unleash all the creativity we have in the force.

Bryan Clark:

Right. I guess that's creativity, too, in terms of the acquisition structure. A PEO might need the flexibility to change what the PMs do. We have a PM for this box or that box. Maybe there's a PM that's just mission systems. Yeah.

James Geurts:

Yeah. Where we saw success in the past is when you can close down that distance between whoever's on the government in charge of the PM, whoever's providing it, whether that's a government or a bunch of entities, and then the end user. Again, we make it sound like this is an easy thing. The automotive industry struggled with this. Still struggles with it over decades and decades. But as we bring up our digital competency as a force, we've got to then use that competency to create new ways of doing things. I'm optimistic we can get there. We just have to get to stepping.

Bryan Clark:
Yeah, yeah. Makes sense. Dan, to build on what Hondo was just saying, do we need to think about having different structures or different program managers that are dealing with hardware versus software? Today, I think, in large part, the program manager handles both. You got the same person in office that’s supposed to be managing the software associated with the mission system and the software that might knit it together with other mission systems as the actual hardware at the mission system itself. It seems like that model wouldn't work, given the challenges that you guys are laying out, or the opportunities, also.

Dan Patt:

Look, I mean, think DoD covers a lot of different applications. By our accounting, actually, every PEO in the US government has some software under their purview. So I think the answer is probably some version of it depends. But I think in places where your ability to update that software in the future matters, which is going to be, for example, any mission system that is deployed, many business systems that the DoD has, and certainly places where I'm going to need combinations of systems working together, manned-unmanned teaming, C2 systems, et cetera, where I know that it's a foundational need to be able to change and update that software in the future, one of the things that a PEO needs to do, in structuring what their PMs and PM responsibilities are, is to think about their mechanisms for delivery.

How am I going to get the software out there? How can I, for example, establish a software factory? How can there be an ecosystem of providers that feed into that? That is a foundational need. I'm not sure there's always just one recipe for doing that, but the idea that this is a foundational concern, I think, becomes clear.

Bryan Clark:

Yeah. That's a great segue, Dan, because I think the last software class I took was maybe Fortran, so I'm a little bit out of date on this, maybe. Jason, can you explain what a software factory is? How does DoD or how should DoD be using those as part of its software delivery infrastructure?

Jason Weiss:

Yeah. Great question. I think, first and foremost, it's important for folks to appreciate that software factory is really just a moniker. At the end of the day, when we talk about building, managing the delivery, the runtime environment of any software application, whether it's something running on a mobile device, something running at the tactical edge, or something running on an aircraft, there's a lot of complexity that's involved there. So part of the challenge early on is helping the industrial base get their head around some of the concepts of software development.

I think, from my perception, when the word software factory started to be used more prevalent across the industrial base, it was a way to help our large primes, our defense contracting base, really start to think about things like, "Oh, hey, there's QA in a physical world where you've got bent metal. There's QA in a software world that we need to think about." You need to have the ability to have space if you're going to be expanding the compute platform in there. You need to have room within your software architecture if you're going to be dropping in an AI or ML model. There is a lot of nuances there that, by adopting that concept of a software factory, I think a lot
of people were able to have more of an aha moment, that there's more parallels there that they can pull out from their past and really act upon.

In terms of what actually is a software factory, it's really the foundation and the mechanism for doing your development and what we call continuous integration and continuous deployment, or continuous delivery into a production environment. I think that touches on a couple of points that my colleagues here on the call have pointed out. Hondo mentioned that software is production. The software development that you’re working on is production. I think one of the things that I heard often in the halls of the Pentagon was people would go, "Oh, well we're just a development environment. We're not production."

That's really a dangerous mentality, in my opinion, because when you start looking at something called drift, where you have a configuration over here in a development environment, and all your tests work flawlessly, and you think the software is perfect, then you go and you push it over into a production environment, where it's running in a completely different configuration, and nothing works. That's where you get the memes and the jokes about, "Well, it worked fine in dev," or "It worked fine on my laptop." The software factory is really a way for the Department of Defense to codify that development is production. We have to manage the drift between those environments, to Hondo's points, very meticulously.

We also need to think about the way we integrate other parts of the DoD process. For example, operational test and evaluation, OT&E applications. When we talk about our commercial lighthouses and what they've been able to do, they break down value delivery into very small pieces. When you hear that Amazon is pushing hundreds, if not thousands, of updates into production each week, they're real small changes. The ability to test those and even react if one of those changes didn't work quite as expected is simple. It's easy to roll that back. It's easy to fall forward, whatever approach you want to call it. When you start aggregating months and months of value creation, months and months of features, and then do one large push, well, that creates a lot of complexity. Which one of those software commits created the problem in production? At what point in time did production fall out of sync with our development environment? The longer you wait, the more you aggregate into that batch, the more risk you actually adopt.

The software factory is really a way to break these concepts down and say, "Look, this thing is a production line, just like there's an F-35 production line. When we hit the end and we roll off that jet, it should be good to go. The same is for software, except the scale is different. Instead of waiting weeks or months, or whatever the timeframe is to create a brand new F-35 from scratch, in the software world, we are talking about, in more mature software organizations, that being measured in days or weeks before that code is actually out there in a production ecosystem.

The more we help folks understand what the software nomenclature value is to the DoD, and we help them think about, ultimately, this is about value creation and getting small batches into the hands of the war fighter faster, I think the more they can resonate and appreciate the nuance of software development.

**Bryan Clark:**

Jason, that that's a great way of explaining it. It brings up a couple of follow-on questions. I think there seems like there's one, the environments are going to be very different when you're building a new system like the software that's associated with a B-21 Raider or an F-35 versus the software that's being pushed to allow forces to integrate in the field. You're making software
that allows a unit or a set of units to be able to communicate with each other, build tactics, develop and evaluate a potential course of action. Changes to that software environment are very different than what might be happening in an acquisition program. One, is that true?

Then, two, when we are out in the field and you’re pushing new software, how do you manage this challenge of needing to push out frequent updates? Do you need constant connectivity to do that? Do you need to put these software engineers out in the field co-located with the troops? Who does that? Is that a contractor function or a civilian function? How do we manage that process when you’re out in the field pushing new software to enable new force compositions?

Jason Weiss:

Great follow-on, Bryan. In the first part, when we think about the totality of DoD software, I often like to remind people that there’s a dramatic difference between the enterprise software that we have that runs in the cloud, does things like travel voucher systems. That’s never going to be forward deployed to the tactical edge running over a SATCOM link. Then there’s the tactical systems that are size, weight, and power constrained. Those are completely different beasts. What one model works well for A may not work well for B. I think that’s really the adaptability and the idea that we have to embrace this heterogeneous environment that is software, that we try to drive from our report. The notion of one size fits all for the DoD is a fallacy because the needs and the requirements and the operational ecosystems are fundamentally different between these.

As far as the second question, that tactical edge is very interesting. I remember one of my earlier assignments I was working on for SOCOM. We were testing it all out at the range. One of the first takeaways we had when we came back to the lab was written on the board in big letters. That is, "Build lighter stuff," because carrying it across the sand at the beach was really hard. But until you actually go and experience the tactical edge, until you actually experience austere environments, it’s substantially difficult for engineers to understand how to pivot the way they think about a solution set to a problem.

One of the things that I worked on very early in my career was updating software in a metropolitan bus system. You think about all the buses driving around, moving people from point A to point B. We had the opportunity, whenever the bus would come back into the depot, it could get gas, change its tires, whatever it needed to do. We had wifi access points throughout. We would actually detect when that bus would come back. We’d begin a communications chain with the bus to figure out if we needed to update that software. The drivers and the actual mechanics of those buses had no clue what we were doing with the software components as it related to the pavement systems that the patrons were using.

We’ve already seen the DoD demonstrate proof of concepts and capabilities here. We’ve put Kubernetes on aircraft. We’ve demonstrated we can deploy things out well in route, in flight. So part of it is breaking down what you need to update, versus what can you wait for when the thing’s actually on the ground during a lull? Does this need to be updated at the speed of relevance, like replacing a munition on a wing tip? Or does this need to be updated at the end of the month when I’m doing some sort of quarterly preventative maintenance? Until we start thinking about software as this is the really move fast gear, and this is I can do this at a slower pace gear, we’re going to be hamstrung because we can’t think about all of it being upgraded all of the time. I’m going to go out on a limb and say that there’s probably very few pilots who want us to upgrade an operating system while they’re mid-flight and being shot at.
Bryan Clark:

Yeah. Absolutely. Which, heterogeneity, I think that's an important thing to embrace. We're used to that on the hardware side. I mean, operators don't even think twice about the fact that certain things will get changed out in the field, and certain things will only get changed out during an overhaul. Hondo, going back now to the role of software during the development of a new system, or the production of a new set of hardware, do we need to start thinking about the software being built for a new platform or a new mission system in parallel with that system, or even before the system's hardware is even built? I know, as a submariner, we would often get the software at the end of the whole acquisition process. Sometimes it would get changed out after we built the submarine. Our combat system software would change out a couple of years after the submarine's supposed to be delivered. Do we need to start thinking about those processes being much more in parallel now when we're building new platforms?

James Geurts:

Yeah. Absolutely. I mean, we have to remember, we've done this before. We used to, back in the day ... I mean, I'm sounding like that old has-been again. We used to change out the EW tapes on fighter jets every night. To Jason's point, we didn't change out the operational flight program. We architected it so the things that we needed to change fast, we could, and the things we didn't need to change fast, we didn't. In the submarine example, the Navy, they've started to differentiate the work. We change the outside of the submarine about every 10 years. I often say, "I think about what's wet different than what I think about what's dry," because we want to be very deliberate in that. When you're spending three or four billion and putting a bunch of Americans underwater, you don't want to take risk in whole design, 24-hour changes in that.

We got to the point where we were changing in the computer about every three years, and the software about every 18 months. This is where the operational for it. We actually were changing it faster, and we ran into a problem because if you get too quick, then you can't train. You can't get mission coherence and all that. I think, to both Bryan and Dan's point, you've really got to differentiate the work. You've got to be able to do things that you need to do fast, fast, and you've got to do the things you need to do with high precisions of confidence and low tolerance of failure, maybe slower. The art is figuring all that out. The challenge is, bureaucracies like to pick one thing to do everything, and you tend to get a sawzall that can perform every woodworking function poorly. That's what I think we want to get away from.

I think technology's also letting us get there faster. It's allowing us to change things without having to have the precision coders. You can have coders create things that you can customize operationally much easier. It's this mindset issue, I think, over everything. We've got to get out of this mode that we can't do it, and focus on we need to do it now. How's the best way to do it? I think the report that was put together here, again, I'm a big guy with a big neck. I'm a practitioner. I'm not the theoretical guy. When I read through that, there's some very practical things that I think the acquisition team and the operational requirements team can be thinking about, and still stick with all of our principles of being safe and having the right level of risk, but not treating everything the same, or we're not going to get the results we need to be competitive as a nation.

Jason Weiss:
Right. If I could just follow up-

Bryan Clark:

Yeah. Please, Jason.

Jason Weiss:

... on that real quick, one of the things that I really want to emphasize is we have done this before. Just like I had talked about the software factories and helping connect metaphors and mindsets, this is really the same topic here. Even when I served during the First Gulf War, we pulled out a tape, we put another tape in, and we were back at it. We were off. There was very little latency there. The more we can help those individuals connect the dots and go, "Oh, this is just like this," I think we can help them move faster and accelerate. The second part of this really comes down to that bit about talent. If we've designed this concept before, but it was with tape instead of digital, for example, we really begin to think about the fact, and this is something we called out in the report, that you need some people all of the time, but you don't necessarily need all of the people all of the time. Software development is no different.

When you come in and you have to do the hard architecture work, the interface designs with the APIs, you need that extra level of competence. You need that expertise of the industrial base. But once you pivot and you're actually in that 18-month deployment of that tech on a platform somewhere, those architects and those API designers, you don't necessarily need them at that point. You need them for the next iteration of evolution of that software system and those capabilities. But the more we can help PEOs and program managers in particular recognize how we need to think about these problems and help them tie this back to experiential learning that they've already had in their past, the more we're going to be able to accelerate.

Dan Patt:

Yeah. Let me also just-

Bryan Clark:

Great point. Yeah, Dan.

Dan Patt:

... chime in on that if I could. I mean, it's just overwhelming. I mean, if you think about what the modern PM faces, that sheer noise about how do you manage intellectual property, what's the right contracting model? Should I do software as a service and work with these new companies which want to deliver the whole capability as a service, or should I try to own all of the source code? How do I think about APIs and manage the APIs? Which ones are important? How do I think about cloud? Do I need my own cloud? How do I think about security, and how do I think about managing? I mean, there's just so many variables. Jason actually takes an estimate about how many choices they make. It is an astonishingly large number.

Part of what we want to do, and I think what report does, it just comes up with six principles, six ideas. We call them ACTs, Acquisition Competency Targets, that help you just work through those. Here's six ideas that, for whatever the problem you have, helps you work through. Here's
the set of logic to work that out. What makes sense? Then there's a whole toolbox of things available. Sometimes it's going to be new industrial-based members which have new business models, and there's places where that makes a lot of sense. Sometimes it's going to be old, comfortable practices. Sometimes that makes sense. There's just a set of ideas to help work through the dizzying complexity and all of the jargon and new technologies and new ideas and old principles and standards.

Bryan Clark:

Yeah. Dan, as Jason pointed out, a lot of this is not new if you just think of the analogy to hardware. Because you've got certain pieces of hardware, you're going to have to go back to the OEM to say, "Well, I need an expert to build this from scratch." Then there's certain mission systems where the company is really just a small business that assembles parts from other people and then turns that into something you can use. But it seems like the one thing that's different or new here is this idea of the service model. I don't know if we've got a service model for buying a radar or a valve that goes into a steam system. The hardware doesn't seem to come with a service model. Is that the big difference here, that you've got ways of delivering product to the DoD on a continuous basis that's different than how we did this with hardware?

Dan Patt:

Yeah. I mean, that is a new tool here. There's probably some argument you can make that the DoD's long also had service contracts in other areas. I think the essence there is recognizing service contracts, like a service-level agreement, make a lot of sense when you can cleanly define it. If you can really cleanly think about the metrics that make sense, like, "This is the uptime or reliability around this particular capability that I need, and I know how to define that," it makes a lot of sense to try to contract that out. You move out all of this management headache outside. But sometimes there are problems where it's just not well-defined. There's a lot of scope, and you haven't actually figured ... You have to work awhile to figure out what the requirement is. Or maybe there's an inherently governmental or even operational risk associated with it, and it's difficult. You can't just move that whole thing out under an SLA.

Bryan Clark:

Right. Going back to talent management, I think, Jason, you brought up an excellent point. With the heterogeneity software and the amount of choices available to acquisition professionals, you really need people that are both proficient in different ways of writing software, or different types of software, or at least capable of doing different types of software, as well as managing the process of buying software. There's a bunch of new talents that need to be brought into DoD that maybe have not been there quite as much resident in the past. Jamie, in the Navy, I know that you had been working a lot to try to improve the acquisition, professionals' expertise with regard to new technologies and software. How do we go about trying to bring these skills into the department? To Dan's point, do we need to start thinking about some of this has to just be acquired via a service model, or do we need to think about hiring new uniform and civilian acquisition people?

James Geurts:

Yeah. I mean, I think it's a little of all of the above. Again, going back to that PEO or PM, what do you need? What outcome are you trying to achieve? I do see, over the last probably three,
four years, I think there was a notion for a while, particularly from these commercial-based software, VC-backed software, they could ignore national security. They're just going to go do their thing, and we could separate national security from national prosperity. I have seen in the last couple years this willingness to serve, thirst to serve. You're starting to see the services get more and more creative of using Reserve forces. You take advantage of the talent we have in the Reserve and the Guard in new ways. Come up with talent-sharing models where folks can flow in industry and then out of government and all that.

We're certainly not where we need to be. If you're a GS-13 and you go to Google for four years and you come back to the government, they're going to say, "Great. Now you're ready for GS-14." They don't know how to yet equate that time equivalency. But I am seeing a lot more thirst to serve from folks outside national security, whether that's in a company or part of the force, and a lot of new entrants into the marketplace, which I think are good. But I think we're about 20% there yet of really getting the fluidity that you see in somewhere like in Israel or somewhere else, where you've got this very fluid ecosystem. The benefit of that, back to one of Jason's points, is that you can then have a much better understanding of the operating environment. You don't have to be on the front lines, but you have an appreciation of the operating environment in different ways. You can bring that creativity in a way that's relevant.

The other advantage that I always have in the back of my mind is we talk a lot about mobilizing industrial based, but we still think about building a victory ship every other day. We've got to think about how are we going to mobilize our software capabilities, should we have to scale up? That's an area that I'm having a lot of dialogue with my government teammates, on really thinking about this idea of how do we mobilize software capability? I think that is a change. That's not just calling it something else. That's going to be something different that we're going to have to get our arms around if we're going to be a prepared and ready force.

Bryan Clark:

It's interesting you bring that up because this is something Dan and I have been talking about. It seems like there's opportunities with software to where we can mobilize in a way that we couldn't. I mean, if we had the—

James Geurts:

Yeah. You don't have to create a factory and create a greenfield factory, but it's going to be different. You can't just think about it when you're into crisis. I think that's an area for the department as we think about our future competitiveness to keep in mind.

Bryan Clark:

Do you think that, with this more fluid model, we'd be looking at a lot more civilian software acquisition and software development personnel inside the department, and they would rotate in and out, which is not too dissimilar to the model we have? Professors at military war colleges, for example, have that model where they jump in and out to be able to maintain their currency outside in their field.

James Geurts:
Yeah. I think it's going to be a lot of things. There's certainly a core training element, understanding what's our digital competency of the force. I think we need to raise the digital competency of the force, whether that's here on the frontline as an operator or whether you're in a combat support role. Then, as we raise that competency, really looking at our talent. Do we have our talent allocated and available at the right ratios in there? Obviously, I think that's going to have to change some. We've talked a lot about it in AI and ML as the buzzword, but I think there's a core element that has to occur before we get to the AI and ML artificial intelligence and machine learning element.

**Dan Patt:**

Hondo, you talked earlier about this triangle between who's managing an effort, who's performing on the effort, and who's the user of the effort, and bringing them together. As you look at successful acquisition efforts, I mean, you talked about drawing from Reservists' pool and the talent that's out in Silicon Valley that's motivated by national security. I mean, are there some examples you can give of creative recruiting, or things where you've seen program managers find a way to bring diversity and other perspectives onto their team to get better outcomes?

**James Geurts:**

Yeah. That's great. We patterned that when I was at Special Operations Command. We called them Joint Acquisition Task Force. It was akin to a JTF, a joint task force, small command and control element. You're bringing the expertise from the units you need to get the outcome. We did that when we were building the next gunship, and we brought in talent from Reservists. We brought all the different services, and we did those gunships in a 10th of the cost and a 10th of the time. You're seeing in the Navy adopting Taskforce 59 on the unmanned space. I mean, you need all the acquisition tools, but you've got to have that operational outcome in there. I think those are great models to bring even within a PEO, a diverse set, a JTF-like mindset. You have a mission. Bring the right folks together, a small community control element on the top. Achieve your outcome, and then disband. Disband is a good underliner in there that I think we could adopt. You're starting to see some of the services do that.

**Bryan Clark:**

Yeah. Hondo and Jason both, a couple of examples come to mind. Dan and I have been out talking to the folks in the Navy doing Project Overmatch, where Navy Information Warfare Command is out there leading an effort to integrate new software onto battle groups or strike groups as they prepare to deploy. They go out with this new capability for force integration and communications. Also, you've got the Air Force 350th Spectrum Warfare Wing, which is doing a lot of this real-time, or as close to real time as they can get, software updates for aircraft as they're deploying, or even when they're out in the field. Are these the kinds of models you're talking about, where the DoD is able to start thinking about fielding changes to software in more of a relevant timescale?

**James Geurts:**

Yeah. I'll go quick, and I'll turn it over Jason. Again, I think the formation depends on the problem. If the EW Squadron is about, "Hey, I got to get whatever the relevant signal is into my sensors overnight," that's one construct. Overmatch or convergence or whatever those are,
those are big functions. I think you need both of those. But I think within the acquisition community, they also have to be a little bit more dynamic than I have a PEO that's been around forever, and I have a PM that's been there with a strict mission, and organized for outcome, not activity. Jason, I don't know what your thoughts are, but I'd love to hear your perspective.

**Jason Weiss:**

Completely agree. It really is about the outcome. Having a little bit of a background in electronic warfare, concepts like Kubernetes, they're not ready to manage down at the microsecond level, where if you don't catch that signal, and it may only be a few microseconds at best, it's gone. You're not going to get it again. So the role of custom chips and ASICs and being able to operate at that scale is not necessarily the same scale as some of the other systems on that aircraft or on that ship or whatever the service might be looking at. I think, really, again, to me, it goes back to understanding the mission outcome that you seek and finding the right tools and the right capabilities. That's the heterogeneity that we're talking about. One size fits all is not going to work well if you're talking about a team building, a travel voucher system and a team trying to create the next-gen electronic warfare pod.

**Bryan Clark:**

Right. But they work together, right, Jason? Because you've got to build your systems in a way that enables their future adaptability. You've got to build them in a way that they can be mixed and matched in different compositions downrange. You also have to have the process downrange to write software that allows them to be recomposed in those ways.

**Jason Weiss:**

Absolutely. I always have viewed it from my architectural experience as a slider on a continuum. There's all these trade-offs that come from the illies, where, as an architect, you often say you pick the least worst architecture possible. They all have trade-offs. They all have things that you're sacrificing if you go for this approach versus this approach. You just want to find the one that is going to get you the closest there in the timeframe that you need to deliver.

**Bryan Clark:**

Yeah. Dan, to wrap up the discussion, it seems like a lot of what we're talking about here is both dealing with the challenges associated with heterogeneity and software, but also the opportunities. We talk about outcomes. It seems like, in general, a lot of the outcomes we're talking about is this ability for the force to be used in ways that are matched, to Hondo's point, to the outcomes you're trying to achieve at that time, at that place, on the battlefield. Our current force is very much a one-size-fits-all military, where we build it to do a certain set of things, and then we just expect people downrange to make it happen, figure out how to adapt. But it seems like there's ways with software so we could enable that adaptation to happen with much greater effectiveness, much greater efficiency. Is that what we're talking about here, leveraging heterogeneity and software to give us that ability to define more customized output outcomes in the field?

**Dan Patt:**
That's right. Look, I think there's a strong analogy to mission command, which American military people are certainly familiar with. It's their ability to adapt in the field to circumstances at hand. We believe that gives superior outcomes. I just want to point out that's something we choose to train and develop. But all humans are capable of change. Sometimes we get hidebound. It depends on which piece you want. I think it's more or less the same thing. It's about enabling the right flexibility down the road. Sometimes that means just realizing, "Hey, I need a software system to help me with something." It's no longer just the F-35 that delivers the capability. Logistics systems matter as well. The airplane alone no longer solves that. I've now inherited this other system. It's not just having a joint force. I need commander control tools to help me with that. It's not just the EW hardware. I need a mission data file. Sometimes it's about focusing there.

Sometimes it's about, "Hey, I have a kill chain that has multiple systems, and someday there's going to be a commercial satellite that I need its data, but I don't know what that is yet. I need to be able to adapt down the road." Sometimes it's new classes of software. Sometimes it's being able to modify things later. Certainly, when we talk about data and AI, really, that's built on foundational software infrastructure. Those are the things that are going to help us adapt. Ultimately, the strategic imperative is the same as it is in mission command. It's about enabling. It's about giving the US a set of strategic advantages to be able to make better decisions at better scale and faster than opponents to be able to achieve better outcomes. We can do that.

James Geurts:

Yeah. Bryan, if I can, I mean-

Bryan Clark:

Yeah.

James Geurts:

... I think our core competitiveness rests on two things: our ability to leverage our learning velocity and our ability to attract an integrated joint and international force. If our ability to learn fast, adapt, and overcome is a core trait, and our ability to attract and integrate joint and coalition and allied partners, then software is one of those key enablers that will allow us to overcome the efficiency of a centrally planned approach. So I think it's at a fundamental element for us to continue to be successful as a fighting force.

Bryan Clark:

Yeah. Well said, Hondo. Thank you very much. I'd like to thank all of our panelists and our co-authors here, Hondo Geurts, Executive Vice Chairman for Sarcos Technology and Robotics Corporation, and former Navy SOCOM and Air Force Official, and our co-authors of the paper "Software Defines Tactics," Jason Weiss, former Chief Software Officer for DoD, and now Chief Software Officer for Conquest Cyber, and our own Dan Patt, Senior Fellow at the Hudson Institute. Thank you, gentlemen. We really appreciate it. Also, thank you to Morgan DeWitt for producing this session. From the Hudson Institute, thank you for being with us. We will see you next time.