Tom Longstaff: Hi everyone. Welcome to today's installment of the SEI webcast, a discussion on DoD software advances and what's next for the SEI. So sorry for the delay for a couple of minutes; we had a few technical difficulties, looked like focused around improvements in security for YouTube. So with any luck, we are rolling now, and I'm seeing your comments on the top chat, so this is great. Again, we want this session to be interactive, so please, at any time, we'll be watching the chat and we'll try to get to your questions as soon as we see them.

I have with me here today Jeff Boleng. Jeff, if you could please introduce yourself and tell me a little bit about your background.

Jeff Boleng: Sure. Yeah, I'm a retired Air Force officer, and I started at Carnegie Mellon at the Software Engineering Institute in about 2012, and in 2018 had the privilege of being recruited to go to work in Acquisition and Sustainment at OSD to work on software modernization efforts.

Tom Longstaff: Okay, great. What'd you do before that? Where have you been around? How did you get to where you are now?

Jeff Boleng: Well, most of the time, post-2012, working at the Software Engineering Institute. Was principal researcher there, worked on a lot of embedded software support for SOCOM and some other projects. For the last couple of years there I was the acting CTO, the job that you've got now, Tom, and then for the last two years been working at OSD. It started off-- a lot of the momentum started off really around the software modernization effort. In early 2018-- actually, probably 2017, when the Defense Science Board did their study on software practices in the DoD and the Defense industrial base, and they released their final report at the first of February on 2018, and I joined A&S, Acquisition and Sustainment, in April of 2018.

Tom Longstaff: That sounds great. By the way, thank you for warming up the seat for me in the CTO chair. It was really great. I've been here for about a year and a half now, coming into sort of do it, and you sort of left us in good hands before going off to an IPA.

So for those of you don't realize it, IPA is an inter-personnel agreement. It's an arrangement where the government can actually hire somebody in from like an FFRDC or a contractor and bring them in and have them operate with all the rights and privileges as a full government employee, in this case a full government senior. So Jeff, can you tell us a little bit about what it was like to be an IPA down at the Pentagon and sort of working day to day in that situation?

Jeff Boleng: Yeah, absolutely. Interestingly, 21-plus years in the Air Force and I never had a Pentagon assignment, so this was my first real time to join and work day to day at the Pentagon. I learned an awful lot. It is the bureaucracy that we all feared and loved. There's pros and cons with the bureaucracy at the Pentagon. It was really a privilege to work for Ms. Lord, for Ellen Lord, as the undersecretary for Acquisition and Sustainment. She gave me a pretty clear

mandate to help modernize not just defense programs' software approaches, but also help the Defense industrial base as much as we could, and that was what we embarked on over a couple-year journey.

Tom Longstaff: Sounds really fun. I myself served as an IPA for a few years at NSA, so this was one of the more exciting times in my life in terms of trying to get some of this work done. I assume for you it's been that kind of a really interesting rollercoaster ride. So while you've been there, while you've sort of been working that area, what kinds of policy initiatives, what kinds of things were you able to work on?

Jeff Boleng: So I was lucky that when I got there, not just the Defense Science Board report had been released just a couple of months before I got there, and it had a number of recommendations, seven recommendations, on ways to improve software practices of DoD and the industrial base, but then also right about the same time, in I think the 2018 National Defense Authorization Act, Congress also told the Defense Innovation Board to do a study on software acquisition and practices. That was executed through a lot of 2018, and I actually really got to piggyback and travel with them to a lot of programs and interact with them a lot on a lot of their findings; and then in January of 2019 the lead for that study moved to the HASC-- Bess Dopkeen-- she moved over to become a HASC staffer, and I took over as the government rep for that study, which we delivered in April of 2019, and then from that point on, Ms. Lord asked me to stay on and implement as many of the recommendations as I could-- or not just me, obviously-- but help OSD and the services implement as many of the recommendations from both the Defense Innovation Board study and also the Defense Science Board study, and that's led to a whole number of things that we'll talk more about, I think.

Tom Longstaff: That sounds really excellent. Just as a side note, during this time period, Lee online has asked: Did you ever work with Kessel Run? So what was your relationship with the things that are going on within the software factories that are happening in Air Force and more broadly in DoD?

Jeff Boleng: Yeah, in fact I have worked very closely with Kessel Run. I was on a call with some of the Kessel Run leadership right before this, actually, and the F-35. So one of the things, I'm really a big proponent of learning by doing, and I think that policy, especially at the OSD services level, at the highest levels, really needs to be informed by practical experience and by doing, and one of the things I've been heavily involved with is the F-35 program, the joint strike fighter. In that program, we're working with Kessel Run as one of the primary application developers to replace the Autonomic Logistics Information System, which is the maintenance and supply system for the F-35. So working closely with them. They've been developing-- they've got at least 10 or 12 application development teams that have been working on that software with us; and then more broadly, getting to know many of the other software development activities-- "software factory"-- that's the term that was coined in the Defense

Science Board report. I've got mixed feelings about the term. It conjures some really good things but also maybe some things that maybe trivialize the difficulty of software production sometimes.

But organizations like NIWC, was SPAWAR, I think, they have an initiative called CANES and C2C24, which is Code to Combat in 24 hours. The Army, especially Army Intel, has been doing a lot of software modernization with their work. JIDO, the Joint Improvised Defeat Organization, they had one of the very early, really agile responsive software development activities. So there's been a whole group of programs that have been transitioning and doing this. Interesting news reports came out this morning about the F-16 has really pivoted with their flight software in the jet to deliver much more quickly and even to the point where they're talking about online updates, probably not in flight, but definitely being able to update very quickly the software in the cockpit of the F-16. So yeah, there's a lot of momentum now. We have a long way to go, but there's a lot of momentum.

One of the findings and recommendations from both the DSB, the Defense Science Board, and the Defense Innovation Board, was that DoD and our industrial base, they actually really did say about two decades behind in our software practices. I find that to be true. We've come a long way; we've still got a long ways to go. The effort really has been on bringing our practices in the industrial base and in software programs in the DoD to sort of commercial best practice state of the practice. There's going to be more effort-- which I think, Tom, you and I will talk about later-- about what's next, what do we do next.

Tom Longstaff: Yeah, it's interesting, because one of the major initiatives at the SEI, one of the things that we're trying to do next, is to build on all those sort of practices from industry that have been sort of out there, tailor them such that they deal with sort of the unique nature of some of the DoD missions, and then try to get them actually embedded in some of these larger programs-- very, very tricky in some of these cases to take what is a long-term culture and then sort of evolve it into what we know is going to be a lot more effective in terms of software creation, delivery, feedback, using what we know about DevOps, what we know about Agile, what we know about resilience and security tools, and what we're learning about AI engineering. So when you look to all of these areas and the places within DoD that are really leaning forward, like F-35, do you see them as very similar to what you know about in industry? Do you see much difference between what they're doing and what you've seen in the industry side? How big are the differences between DoD and industry?

Jeff Boleng: Wow. I do think-- we try hard-- DoD has a history of considering-- government in general has a history of considering that it's-- the phrase we use is a "special snowflake"-- and we've been trying hard to dispel that notion, that we're not all that different, that the software challenges we have and the software programs that we have can use, by and large, a lot of commercial best practices. I think some of the differences though are a lot of our programs are

embedded program. So some of the continuous integration, continuous delivery and agile approaches-- I'm a proponent that they can and do work well on embedded programs, on safety-critical and embedded programs. That's something the industry is still working to discover as well, and there's open debate there.

And then a lot of our programs have humungous legacy code bases. Programs like Aegis is one of the ones that I visited early on and have done some interaction with. They've got a common source repository that is upwards of 20-ish million lines of code, and the Aegis weapons system writ large is a massive amount of complexity and subsystems, and I think they have well over a thousand developers that contribute to that code base from a whole variety of companies, not just from the prime contractor but a variety of subs, and also there's actually good government partnering and government interaction with that code base, which I think is one of the keys to success, of good programs, is government takes ownership and is involved day to day on the agile teams with the contractors. I think that kind of partnering is essential. That's where we see a lot of success.

Another big difference I think is the level of required rigor for developmental test, operational test, which developmental test is really sort of verification, "Did I build what I was supposed to build?" Operational test really is validation, is, "Is it fit for purpose? Does the thing that we built meet the mission requirements that we need?" And then also the safety certification and flight testing and all of that other stuff. Anytime you build a system, especially a complex weapons system that yields deadly force, there's a huge responsibility there to make sure that it's safe and secure and high quality. So what have been referred to as "tailgate processes"-- that I think diminishes the importance of them a little bit-- but those end processes are necessary, and one of the big challenges we're having and seeing is how do we integrate those. Another term that's used often is "shift-left", "shift-left testing". How do we integrate those activities more into the DevOps cycle, the DevSecOps cycle, and create confidence as early as we can during development so that those other processes at the end can be shortened because we've got things like automated testing that-- we're never going to get away from some level of developmental test, operational test and safety certification flight test-- we're never going to get away from that entirely, but I do think that we can develop ways, or build confidence into the development tool chain so that those processes are shortened and we can move on more rapid iterations.

And then also the big principle of DevOps-- another big principle of Agile and DevOps is batch size, decreasing batch size. If we introduce smaller changes more often, those things are also easier to certify. But then another challenge with that one is the legacy code bases we have in some cases-- in many cases, actually-- don't lend themselves to isolation as well. So a lot of the testing community will always say, "If you touch one line of code, I got to retest the whole thing." In a good architecture and a good design, we ought to be able to implement those changes and localize their impact so that we maybe don't have to retest the whole thing or recertify the whole thing, and actually that's the case in some other more modern programs that

we've been working on. There's been very good architectural and design approaches that have helped localize. Like the mission systems in the cockpit are completely segregated from the flight control systems. So if we make a change to the mission systems to be more responsive to an adversary threat, we don't necessarily have to recertify the flight control systems. So in some of our newer programs, that design architecture approach has been showing fruit, so that we can move some of the certification and testing earlier into the lifecycle.

Tom Longstaff: Actually, Jeff, that is a wonderful segue to a question we got from David in the chat. By the way, hi David. It's been a long time. One of the things that we're looking at in terms of helping to solve that problem, how do you get automation and isolation of the pieces so you don't have to retest the whole thing, is in model-based software engineering, in creating formalized models that allow us to do guaranteed separation, and some of the automation that would test against these kind of requirements. Do you see that as something that could really be effective at helping us get to some of the automation and some of the real-time operational T&E that needs to happen in order to make this cycle time short?

Jeff Boleng: I do. I'm a huge proponent, obviously-- people that have heard me talk before or met me-- huge proponent of automation and tools. Tools do not trump competence-- I have to say that up front. At the end of the day, the single most important thing about software development is the competence of the developers and the engineers involved. But tooling can go a long way to assist, and especially automation. There's a lot of promise in model-based system engineering, especially when you couple model-based system engineering-- the question I think David was really asking is: Do the tools need more help in model-based system engineering? And I think they absolutely do. We have some tooling around it. I think a recent program-- actually maybe not as recent nowadays-- but there was a DARPA program called HACMS, the High-Assurance Cyber Military Systems program, that was about a four- or five-year program that showed incredible amounts of power from model-based system engineering, from formal verification, and from automating the tool chain.

One of the things that I see though is we've got these model-based system engineering tools and we create digital twins and we create a simulation environment so that we can work analysis of alternatives and tradeoffs and design alternatives and even test different code changes, but the digital thread from actual engineering and creation to the end, where we actually deploy to operations with those code, that digital thread is not always preserved, and one of the things that I have said for a long, long time now, since I was a wee developer years ago, was the only thing that really gets maintained in our systems and sustained is the actual source code, and until we have a way to kind of round-trip from our tool chains, if I make a design change and that thing generates source and I need to fix something, I need fix a complex race condition or something like that, if I can fix it in the tool or even fix it in the code and round-trip it back to the model-based engineering tool, there's got to be parity there between all of the digital artifacts in the tool

chain, all the way from conception and engineering all the way to operations. If we can't fully close that loop I think the tools are going to continue to struggle.

Tom Longstaff: Yeah, I completely get you, and of course the other thing that really resonated with me a lot was your statement that tools are really no substitute for expertise. They can help that human-machine teaming, where we can get together and really take the most advantage of the human designer, the human coder, the people who are actually giving the feedback from the operational perspective and sort of coming back, so that the tools are even more powerful in the hands of expertise and things that are happening there. Completely agree though-- the tooling that we have right now, probably insufficient to link the feedback and design requirements all the way back to a modeling change that would then lead to potentially automated code construction, that would go down that way. Do you think that-- these were solutions or items that are really called out within the DIB study and the DSB recommendations. Is this in line with where those studies came down in terms of the real problem with software never being finished and things that are kind of out there? Do you see that as all lined up in the same direction?

Jeff Boleng: Let me make sure I understood the question properly. So certainly I think the DIB, the Defense Innovation Board report, touched more on model-based system engineering, and we had the participation of the folks in OSD from research and engineering that lead the digital engineering strategy, Philomena Zimmerman and some of the other folks in R&E-- so shout-out to them; they were very helpful. I don't think the DSB study did it as much. I actually really have to say a lot of the recommendations are more fundamental than advanced tools and advanced approaches around model-based system engineering. I think our biggest struggle right now is talent, digital talent, attracting and retaining digital talent. The DoD has a personnel system that's been around for a long time and it's not really responsive to talent.

The DDS, the Defense Digital Service, has done a lot of work, and continue to do work, and there's some new language in the most recent 2020 National Defense Authorization Act. They've done a lot of work on helping attract and maintain talent, and one of the vignettes and the stories they tell is if you go out on USAJobs, which is the place where we try to attract our talent, and you do a search for modern titles for software developers, like User Experience Designer, or a product manager or a product owner, you won't find those in any of those job announcements. So we really-- like I said, I think really the reports were less about high-tech tools and processes and more about really simplicity and adopting commercial best practices. The commercial world-- I mean, the Facebooks, Netflix, Google, Microsoft, all of the big software companies, they are able to iterate on a capability and deliver new features in hours and minutes and dozens of times or hundreds of commits per day, or thousands of commits per day, that actually go through the tool chain with automated test and security scanning to deployment, sometimes to sort of red/black AB deployments, and then 1 percent, 5 percent, 10 percent rollout, but they're able to actually release capability at incredibly rapid paces in order to continue to innovate, and that's as much of a culture and people thing as it is anything to do with tooling and processes.

Tom Longstaff: Yeah, I can absolutely see that as true. Actually, this kind of gets into some of the things that Edward is saying in the chat, where if you're going to use things like auto-coders, they have to generate actual working source. I think that's also sort of in line with this, and making sure the tools are working in this way. Robin has sort of an interesting thing: Do you think the functional-based organization is impacting success for DevSecOps? So in your travels around DoD and your visits to commercial areas, have you seen a lot of success within functional-based organizations and that entire movement toward functional languages and representations and models, as opposed to some of the more traditional procedural ways that we've constructed these codes?

Jeff Boleng: There's a lot there. I want to comment on the auto-coding thing really quickly and then I'll get to that part-- is I completely agree that auto-coding, if we use it and we do it, it's got to create executable code, workable code. I think it also has to create human-intelligible code. A lot of times-- especially in the era where we can't round-trip things. So human-intelligible code is really important from automatic code generation, and I'm glad to see Robin's participating. Thanks, Robin, for the question. You made the leap to sort of functional languages. I'm also a big proponent of modern languages and using modern languages to help provide guardrails and safety checks for programmers and developers. I actually haven't seen-- or this could be my own-- I just didn't see it and they could exist, but I haven't seen a real uptick of really modern language and even functional languages in any really of the programs that I reviewed. Languages, a teeny bit of use of things like Go, not really a lot of use of things like Scala or Rust, which provide a whole bunch of benefits.

There's an interesting story about the Firefox team at Mozilla. They rewrote their code rendering engine in Rust and it was not only far more defect-free because of the inherent strong typing of the language, it's also faster-- it was like 30 percent faster-- and they rewrote the entire rendering engine in a matter of less than a year with a fairly small development team. Some of those things-- the expressiveness and productivity of some of the new languages, while they're a lot more complex to learn and master, really do bring a lot of benefits, and I haven't seen a real uptick in those.

Interesting, I got to go to Oklahoma City and talk to some of the B2 team there, and they have in their-- this goes back to our legacy code base issue-- they've got massive piles of JOVIAL code that they have to maintain, and I actually said, "Wow, how many graduates do you have that are fluent in JOVIAL that you can hire?", and the answer of course is zero. So that's a challenge.

One of the things-- I don't know, I might have seen it in the chat window over on my left-- I'm a fan-- the legacy code bases we deal with, I'm actually a fan of, in some cases-- in fact, maybe in many cases-- we need to do clean sheet. We should not be afraid to abandon some of our legacy code bases because they're so hard to maintain and use so much old technology that we need to

really start strangling the leviathan, as I think it was-- I'm trying to remember who coined that term. But we really need to start replacing a lot of these legacy code bases with modern languages, modern tools, and modern design approaches, because we can't forklift the old ones. I just don't think it's feasible to do effectively.

Tom Longstaff: Yeah, I can absolutely see and completely agree, and this sort of leads directly into the next comment from Young Thang, where they basically were saying: Are you considering refactoring, or tool-aided refactoring? We at the SEI absolutely-- we're doing a significant amount of investment in refactoring, Jeff, for exactly the reason you're talking about, for trying to understand how are we finally going to get away from some of this legacy code, and one of most important things in refactoring that we're trying to do is isolating various portions that in a very large code base we can refactor out, bring into a modern language, bring into sort of a modern set of tools, and then essentially reimplement, using sort of much better and much more solid and much more resilient techniques that might kind of come into that way. So I think that's a great direction.

I'm encouraged to see that we have so much of the language work and some of the things happening in industry and sort of moving that way, because that is probably the way that we're going to pick up what the real benefits are and then move them sort of into DoD. We used to have DoD spin-out, and we talk a lot about spin-in now, and spinning these technologies back into DoD and sort of making that work. Hasan, of course, talks about: What do you think about the requirements and use cases for functional programming? Can it be applied to many DoD systems? So I think you started to answer that. Do you want to go down a little bit further down that to talk a little bit about where you see the primary areas? One of the things that, for example, people talk a lot about the use of functional languages within embedded systems and various kinds of real-time systems. So what do you see there?

Jeff Boleng: Interesting. Yeah, I want to keep the discussion a bit more broad and not maybe do a deep-dive on just functional languages. There are different programming paradigms that are very appropriate for different problem sets, and I think we haven't always done a good job of applying the right language that supports the right programming paradigm to the right piece of the design, and we also sometimes have-- I think we have a resistance to use multiple paradigms and multiple languages for a single system and we shouldn't be afraid of that. Modern design techniques that create a lot of loose coupling allow a lot of the components to be implemented separately and in the most effective technology that is available, and I think those are disciplines that we need to push on. The auto-refactoring-- I know there's a lot of research at SEI in Ipek's group that-- they've got a research project called Untangling the Knot, which is understanding legacy code bases. I think it's important to be able to use some level of automated tools to refactor legacy code bases, but I'm actually maybe a bigger proponent of constantly refactoring your existing code base and reimplementing.

If you look at some of the articles that are out there about some of the really big commercial code bases, some of the big commercial code bases, there's not code in them that's more than five or ten years old, because the teams are constantly refactoring, and this is a problem-- the subheading of the Defense Innovation Board's report was "Software's Never Done", and in the government and in DoD, we haven't yet embraced the fact that software's never done, that we need to constantly refactor our code bases and ferret out the legacy pieces and reimplement the legacy pieces. Part of it is we never actually-- from a programmatic standpoint, from an acquisition standpoint, we never actually budget for that. We think that we can have a long list of requirements that once we satisfy them all with some level of software, we're done, and then we turn it over to maintenance, sort of like you would a hardware, like a tank or a jeep. But software is just not like that, and that's one of the big things that-- I'm a bit of a crusade to kill the term "software maintenance". "Software sustainment" is a little better, but even that I'm not a big fan of. We need to continuously refactor our code bases and continue to modernize them. I actually have been an advocate with program managers I've talked to of budgeting between 20 and 30 percent of your ongoing software budget just to refactoring and improvement of the code base, just to caring and feeding for the code base.

Tom Longstaff: Yeah, that would be a real game-changing, if we actually had that level of budget in refactoring and keeping code from becoming too legacy. Young Thang's point on the chat was: Do you have difficulty understanding legacy code? Yeah, the older it gets, the harder it is to understand it, and the fewer number of software professionals that remember what the code was really like. This is, I think, one of the reasons why this sort of constant refactoring and constant movement and budgeting for it, as you say, is sort of the way around that. Maybe the other area that I find there that's required if we're going to do that level of refactoring and pushing out updates on a regular basis is we're going to have to break down some of the barriers between acquisition and sustainment. We're going to have to create common infrastructure that actually sort of works between deployed systems and the acquisition systems and the models that generate the acquisition systems. These all come together for really an entire new environment or new approach to the way software happens within DoD. I think that's an incredible direction that you're sort of pointing out here.

Jose actually points to a huge elephant in the room, so I'm going to go ahead and go there: Classified systems and high-security systems and security requirements on development systems and deployed systems. Is this a barrier? Is this something that helps us? What is the interaction between all that we're talking about in software and the security requirements, some of which have been handed down for an awfully long time, that are within our systems?

Jeff Boleng: Yeah, I'm going to comment on that and take it a bit more broadly also, as most of our systems, most of our weapons systems at some level, some of the software capabilities that they deliver are classified. Especially offensive capabilities become special access programs. We've seen a lot of-- we've seen the ability to do an awful lot of source code development in an

unclassified space. It's probably still going to be a highly controlled place, like for official use only, or in a GovCloud instance that's still highly controlled, but we can do a lot of development in an unclassified sense. There was one of the programs that I was associated with that upwards of 80 to 90 percent of their software development is done unclassified, and if you follow the sort of cloud native development and 12-Factor App, you want to localize a lot of your specific changes to configuration files and not actually embed those kinds of dependencies in your source code.

So a big challenge that we have is we have these enclaves, these classified network enclaves, and they're still to this day-- this has been-- I've witnessed this for three decades. We've always talked about interconnecting our classified enclaves with each other so that we can share data in a secure, appropriate way from low to high or whatever. We still struggle with that and are challenged with that, and where this becomes a real issue is if we have to do something on a classified enclave and new development, the tool chains and the features lag sometimes significantly behind what's available in commercial cloud or commercial tools. An example of this is there's a thing call C2S, which is the classified cloud computing enclave that has the TS-SCI enclave that the intelligence computer instituted with AWS and now they've broadened it to Azure. The feature sets available in those APIs lag from 12 months to two years, for example. So if you develop most of your software in an unclassified enclave, even though it's a secure enclave-- I don't want anybody to misunderstand that-- and you depend upon a certain feature set, like a certain data API or a certain backing store or something like that, and there's not feature parity when you move that up to your production environment, which oftentimes the production environments are definitely classified even if most of the development is not, and there's not feature parity on the infrastructure, we run into a lot of problems, and that's a real technical challenge that we have to address, we've got to get better at.

Tom Longstaff: Yeah, I agree. This is, I think, leading to Martin's comment in the chat of: It's close to impossible to develop on a separate system. I wouldn't go so far as to say close to impossible, but clearly there's a lot of the infrastructure and modern tools that we expect to have on these systems, but my perception is we have two major issues. In the first place, some of these modern software tools are difficult to run in isolation. They count on having the entire backend of sort of the internet in the place; and the second thing is the whole-- how do you do C&A? How do you do certification and accreditation fast enough to keep the development system on these secure enclaves that are secure for a reason? How do we really keep those up to date? I think that gets to Paul's question: Do we have modern software tools on classified clouds? So, what do you think, Jeff?

Jeff Boleng: I think we have-- it depends on how you define "modern". I think our classified clouds are definitely in much better-- much more modern than they ever have been, but like I said, there's still some feature lag of, I don't know, twelve months to two years in some cases. That's getting better but it's still a really manual process to keep those featured updated when a

new software release comes out for a library, like identity management library, or for Kubernetes or something. When a new release comes out, even as hard as we try to move those releases into our classified enclaves, in our classified development enclaves,. It's still a largely manual process. So while that's getting better, we're still not there.

I would say the SAP community has done a really good job with a thing called FENCES, which is a capability that they've built out to allow the kind of work that's actually helping some of our programs that are really getting started with new software.

I wanted to comment and actually make a little point that it's my belief that a lot of software today, if not a predominant amount of software today, is really software by construction. There are so many good open-source libraries and even commercial libraries that are available that you can use that really-- I take a phrase from Leo Lessig from the Free Software Foundation-- or not Free Software Foundation, sorry-- I think from his book called "Rip, Mix, Burn", and I think a lot of our software now is we rip capabilities from a lot of places, we mix them together and we burn them, and we've got our own new capability that we provided some level of code but it's a lot less than what we have been doing in the past, and in that kind of an era, having access to those widely available libraries that are both commercial and open source and having feature parity is just essential. So like I said, it's another challenge that is recognized and is being worked on, but I think we're maybe not making the progress that we need to make on that one.

Tom Longstaff: Yeah, that makes absolute sense. I think looking to David's comment online, the Federated Kubernetes cluster, yeah, I think that's directly in line with the kind of stuff that you're talking about. Completely agree that greenfield programming almost doesn't exist anywhere. For a long time, earlier in my career, we talked a lot about clean room software engineering and reinventing from scratch and redoing that thing. But going back to an earlier discussion, when we were talking about refactoring and doing sort of constant refactoring and keeping that up to date and keeping code fresh, again, it really isn't about always recoding everything; it's always about adapting sort of latest things that other people have coded, mixing them up, gluing them in, twisting them until they fit just right, and then sort of having the new capabilities sort of in there.

That doesn't seem to be the way acquisition works. That doesn't seem to be the way that within DoD the regs and the FAR and everything else-- it just doesn't seem to fit with that. So where are our disconnects in DoD acquisition versus sort of code by construction?

Jeff Boleng: This is a much bigger challenge that goes more broadly, is of course we all want to be good stewards of our taxpayer dollars. A lot of the way that that gets interpreted is we never want to have to pay for something more than once. But that ignores the fact that budgeting to actually increase or manage a code base and increase our capability because we're refactoring-- that's part of the challenge. I mean, it goes all the way to Congress, where-- I mean, Congress, of

course, both authorizes and appropriates money for the DoD. It's hard for them-- and I'll use a concrete example, the F-35 maintenance and sustainment system, ALIS. We already paid for that. We paid to have a maintenance and sustainment system implemented so that we could properly maintain the joint strike fighter. We're currently paying to reimplement it, because the current system had design issues and couldn't scale as the fleet grew. That's a hard pill to swallow for the appropriators and the authorizers in Congress. It's a hard pill to swallow for program managers and senior leaders in DoD. But it's something that I think we're just going to have to accept, that we need to keep pace with commercial tech and modernize at the same speed as commercial.

One of the actual things that we talk about with some of the policy changes that I've always made emphasis of, we have this-- we come and go-- DoD comes and goes about build versus buy, or buy versus build. Sometimes we think we want to build everything from scratch. Other times we really look and we want to adopt commercial products and as much commercial things as we can. I'm a fan of buy before build. If the commercial industry offers a software capability, we should adopt that software capability. There's a couple caveats there though. One of the caveats is we should never modify commercial software because our workflows and processes are different than commercial workflows and processes; we should actually modify our workflows and processes to fit the commercial software because it's widely used by a whole bunch of companies, and so we should modify our processes. The other thing is when and if we adopt commercial software, we need to commit to staying on the upgrade pace of the commercial software. I mean, there's large chunks of commercial software that get used in the DoD, but we tend to be so risk-averse and not implement the changes and the fixes as they come in that we fall further and further behind. I mean, there's a bunch of Windows XP floating around in Navy ships that we're going to extraordinary lengths to secure now because we didn't continually modernize, because we committed to a commercial operating system but we didn't continually modernize.

One of the Defense Innovation Board members, Milo Medin, actually used to say that DoD, their risk aversion is what leads them to not upgrade and patch at the same pace that the commercial world does, and he actually made the point that that doesn't help our risk profile. That actually-- we swallow massive amounts of risk by doing that, by not keeping pace with updates, and we think we're being risk-averse by, "If it's not broke, don't fix it," but really we're actually incurring way more risk because we're falling further and further behind, and our tech and our ability to respond to things like cyber vulnerabilities just gets further and further behind, and then we get to a point where we're almost forced into greenfield again. We have to reimplement. If we had kept up, we maybe wouldn't have the same pressure to reimplement.

Tom Longstaff: Yeah. By the way, Fitch, yes, this is live. I'm seeing the comments as they're coming in, so absolutely is live. You actually referenced Ada, and that's a real blast from the past for me in terms of how Ada is actually being used and sort of where it's there. So Jeff, I'll

ask you to sort of respond to the Ada comment, but what I really want to do is I want to follow up on some of your comments around the congressional allocation. I specifically want to talk about NDAA and the different clauses in NDAA that are beginning to help us get to where we need to go. So first, talk to me a little bit about where you think Ada sits in the ecosystem of where we are, and then, if you could, let's talk about the recent sections in 2019, 2020 NDAA that are moving us in the right direction.

Jeff Boleng: All right, I'll address the Ada one first and then I'll probably pass it back to you to hone in on some of the NDAA sections.

I don't make it a secret that I am still a supporter of Ada. In fact, it's not a secret I was the SIGAda chair at one of the SIGAda conferences a few years back. I think the U.S. government and DoD have largely abandoned the use of Ada. It has shown good traction in embedded communities in Europe. I think the Ada ecosystem in Europe is still pretty healthy, in the areas that it's good. It's still a fundamentally good language. It's continually being modernized, actually. There was a 2012 update to Ada and I think there maybe was a 2017 update to the Ada standard that's included more things like syntactic (inaudible) and some other things. So I would say it's definitely not a dead language but it really has been adopted in very specific instances where safety criticality is important.

Now, there still-- our large legacy code bases, there's still a fair bit of Ada in the DoD programs and systems, I think particularly in the space community. Some of our satellites and our spacecraft have got a fair bit of Ada in them, and that's sort of like-- it's not quite as old as JOVIAL, but it's, again, a hard place to get people that have that level of expertise to use the language features. Ada is an actual fairly complex language, and I think that part of the reason it struggled with adoption is the design of the language was ahead of compiler technology, so it was hard to actually implement compilers that could enforce all of the typing rules and all of the constraints in the language. But anyway, still a proponent. But I'm going to pass it back, Tom, to you. Which ones of the NDAA provisions do you want to talk about first?

Tom Longstaff: To start with, why don't we jump back to 2019, with 860-- I'm trying to remember exactly what the number is.

Jeff Boleng: Eight sixty-eight?

Tom Longstaff: Yes, 868. That's exactly right. Let's start there.

Jeff Boleng: Okay. I apologize, I shouldn't have done that. I'm going to go back a little further because that's what 868 references, but first I want to share a little anecdote that the Defense Innovation Board-- an observation they made which I thought was funny. There's been an awful lot of legislation coming from Congress from the HASC and the SASC, the two Armed Services

Committees, directing DoD on how to do software better. So there's been a whole bunch of provisions, and it's funny that-- the Defense Innovation Board jokes that Congress is inside DoD's OODA loop when it comes to software modernization, that being the decision loop-- the observe, orient, decide, act loop-- that Congress is legislating faster than DoD can actually take action to modernize. So the folks that I know, the staffers in the HASC and the SASC-- I've asked them several times, "Give us a little bit of a break. Can you slow down and let us actually implement some of the things you're telling us to implement so that we can get some breathing room?"

But yeah, specifically the 2019 868 provision, it actually required a report-- it basically was a provision in 2019 that basically said, "Hey, that 2018 Defense Science Board set of recommendations, the seven recommendations in the Defense Science Board report from 2018, go implement those. Go do those. Send us your report telling us how you're doing, and if you're not going to do some of those recommendations, you need to tell us why you're not going to do them."

So what we did-- and it actually goes back and references some stuff from 2018, I think, also. In the 2018 NDAA there was a provision-- there were several provisions-- 872, 873 and 874-- that also talked about software modernization. The 872 provision is the one that actually tasked the Defense Innovation Board also to do a software study, and 873 and 874 basically mandated a number of agile pilot programs that have been in execution and being reported on. So the 868 report from 2019, interestingly, I sort of volunteered to author that report as a result of all the effort we put behind both the Defense Science Board study and the Defense Innovation Board study, and I actually really want to point out that the implementation of these recommendations, it was really-- my predecessor on the SWAP study, Bess Dopkeen, set up a series of working groups from across the agencies and the services in DoD that was really the key to the successes we've had, and so it's really not a single individual. I don't want anybody to mistake that I'm taking credit for any of this. I was involved, I think I contributed, but really is was a very big effort by a whole bunch people at OSD and the services to actually make the program we've made.

And so the 868 report, which is-- it's in final coordination right now-- is really a report on what we've done over the last two years for software modernization specifically to address the seven recommendations from the Defense Science Board report and the ten major recommendations from the Defense Innovation Board report. So I think we're going to be able to deliver that to Congress by the end of May, at which point it'll be a public report and it'll be available for everybody, but I'm actually looking at a matrix in front of me of the 16 or so major efforts that we did in order to answer that requirement from Congress to report on what we've been doing for software modernization, and we can get into-- I'll try to summarize them.

Most of them are around really four different lines of effort. Policy-- there's a new software acquisition policy that was published in January of last year, actually, I think, or this year-- anyway, new software acquisition progress is part of the Adaptive Acquisition Framework. There's multiple acquisition pathways now. We're working on a pilot program with the appropriators to actually create a single kind of budget authority for software and digital technology, what some people call Colors of Money. We're prototyping a single budget type for software and digital technology. Done a lot of work with enterprise services with both the partnership with CIO, the DoD CIO, and DISA to provide enterprise services. The Air Force has really been helping us with this as well, the Air Force chief software office-- creating enterprise tooling, cloud-based development environments, comprehensive DevSecOps reference implementations.

The big, tough one we've been working on is workforce improvements on how we, like I said, attract, retain, hire, develop-- right? We want to develop internal talent and keep it modern as well. And then the last one is really culture and things like security accreditation, working towards things like continuous ATOs, Authority to Operates, the rapid ATOs, those kinds of things. Those are the four sort of broad areas that we've done work, and there's been a lot of specific things that we've done in each of those areas to modernize our ability to do software, both internal to DoD and with our industrial base, essentially providing a framework or capabilities or authorities for the industrial base to be more innovative as well.

Tom Longstaff: That's fantastic. So we're getting closer to the end of the time, and where I really want to go with the NDAA clause closer to my heart, 255. So let's talk a little bit about what the feature of software looks like and how NDAA is really leading us toward a future that might pick up on a lot of the things that we've talked about over the last 50 minutes.

Jeff Boleng: Sure. Yeah, so in 2020-- I think it was the SASC, actually, put this language in there. Section 255-- I refer to it as 0xff, which is a little bit of a nerd joke since that's hex for 255. I don't know if they did that on purpose for not. The 255-- basically we've spent a couple of years-- we're not done, and I'm not at all claiming we're done-- really working on modernizing policy, tooling, enterprise services, personnel, everything that I just discussed. But really what that's doing is getting us closer to-- or the goal is to get us to commercial best practice, commercial stated of the practice.

We haven't, as an organization, done as much work on sort of what's next. So what 255 does is it calls for a science and technology strategic roadmap for software research and development, and it's something that the SEI is working on. We're working closely with Research and Engineering-- both Acquisition and Sustainment, Research and Engineering, and the CIO are sort of partnered on this, working together, to say, "What does that roadmap look like?"

Another sort of claim that I've made or opinion that I have is that I think the predominance of software engineering innovation is really done in the commercial sector right now, not just in commercial companies on the product side but also in the research pieces of commercial companies. Places like Microsoft Research and a lot of the big companies have their own R&D shops. And-- excuse me-- and so an awful lot-- see, it's live, so everybody probably just saw that. An awful lot of innovation is done on the commercial sector, and so working to harness that, both commercial research, academic research, and the government labs, working to harness those and say, "Where should DoD invest to bring about the next big advancement in the ability to produce good software?" So some of that is model-based system engineering; some of it is virtual environments. I think there's a whole lot of ways to go, which I'm going to flip the script a little and ask Tom: Where does SEI see this?

Tom Longstaff: So I'm actually very optimistic, and the way I see this is the SEI has had a long history of working directly with the private sector, going back into our product line work and architecture work and various places where we've partnered closely with the private sector, and then brought that technology and sort of the new techniques into DoD and applied them directly. I see especially the study and the recommendations that are going to come out of the NDAA 255 report as taking us in that direction, as really helping us to say: Let's take the best of what academia, the private sector, and a few places in the government are really offering and create an R&D roadmap that will fill in the gaps, that will fill in those specific areas that the private sector just isn't really investing in very deeply, and some of the areas that we need some additional investment, also maybe even to help the private sector succeed more greatly. So I see this as not an SEI-only effort. I see this as a huge collaborative effort across the world, where the SEI, much like we've done in reports in the past, like the Ultra-Large-Scale Systems study and some other things that we've done in the past, can really begin to get an R&D roadmap that looks not to just fix kind of the tactical problems of today, but really understanding what the future of merging everything we're learning in AI engineering, all the stuff that we're learning within the Agile DevOps community and that world, everything that we're learning in resilience, formal methods, formal modeling, and really understand what research will round out our ability to apply this into the feature, and really sort of make it really effective at getting to the implementation of the future construction of software, to sort of use the term that we've come up this way.

So that's what I see. It's where-- I feel like we at the SEI are all in in bringing all of the different best elements of the world together to help us understand this, to really get the very best advice we can get from everywhere to create that R&D agenda, and then get buy-in throughout the federal government and throughout other areas in the private sector to begin to fill in some of these areas that have just been so difficult and so thorny all the way through the end.

So anyway, that's my feeling. That's where I come off very optimistic. It's why I feel good about where things are actually going over the next few years, and it keeps me excited to sort of

being here. So Jeff, any sort of final thoughts or things that you want to share before we close out?

Jeff Boleng: I just want to throw out one more thing, is I really want to encourage all of us to collaborate closely with commercial industry, and one of the things I want to-- another, I guess, elephant in the room that I wanted to close on is the government and DoD needs our commercial industry-- I mean, the innovation in our commercial industry, especially our tech sector and our software sector-- I think it could be an element of our national safety, international power. So I wanted to call to mind-- there was an interesting sort of op-ed or editorial that one of the Defense Innovation Board members wrote. Jen Pahlka wrote something called "Sharp Knives", which is there's been some resistance in some parts of our tech sector to not want to collaborate with the Department of Defense. But I would refer people to that article, and it makes a really good case for why we all should contribute and we're all in this together, and having a strong national security is important I think for everybody in America.

Tom Longstaff: Thank you, Jeff. Thank you so much. This has been a really wonderful session. Thanks all of you for participating in the chat and to watching and spending an hour with us today. Again, I apologize for the late start, but I hope that you were able to take something away from this discussion. Please join us for the next webcast June 2, which will be hosted by Linda Northrup and Philippe Kruchten, and you'll get the registration link, especially if you have subscribed, so please do, and hopefully we will see you all again next time. Thanks for being here.