Why doesn't anyone program anymore?

Tom Longstaff
Chief Technology Officer

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Who am I?

I was a programmer! Math -> Physics -> Computer Science -> Software Engineering -> Acquisition -> Systems Engineering -> Software intensive systems and human impact

Self identify as a Computer Scientist, specializing in network cybersecurity and incident response

PhD in automated programming

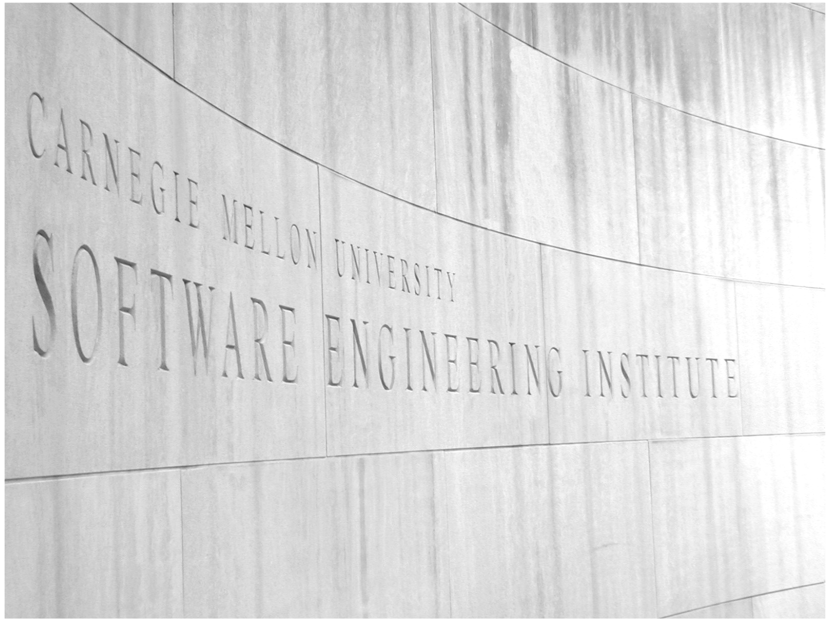
Well-versed in a dozen or so languages, focused on OO and functional programing

Conversant with a dozen or so more

Worked on many software projects using a variety of standards for software process, culminating in DevSecOps

No one wants me to write code anymore ☺

Where am I?



CTO at the Software Engineering Institute

A DoD sponsored R&D center at Carnegie Mellon University (CMU)

Charged to improve the state of the art and practice of software engineering and cybersecurity

Added AI Engineering in 2018

Capable of conducting both fundamental research and classified work

I'm here because of Watts and Rich Pethia, its all about the data.

What do I think about programming?

Observations

- Real SW engineering projects never start from scratch, never just write code, and rarely focus on unintended consequences of their system when it is reused.
- Best practice treats CODE as a first-class object, but SW architecture is really the first-class object.
- System quality attributes hold the key to success.

Suggestions

- Modify assumptions to progress in SW process.
- Use quality attributes of the desired system and the systems to be repurposed as the primary artifacts.
- Focus on code as supporting quality attributes and emergent behavior of the combined system.

F-35 SLOC as composed

7% Assembly

5% Ada83

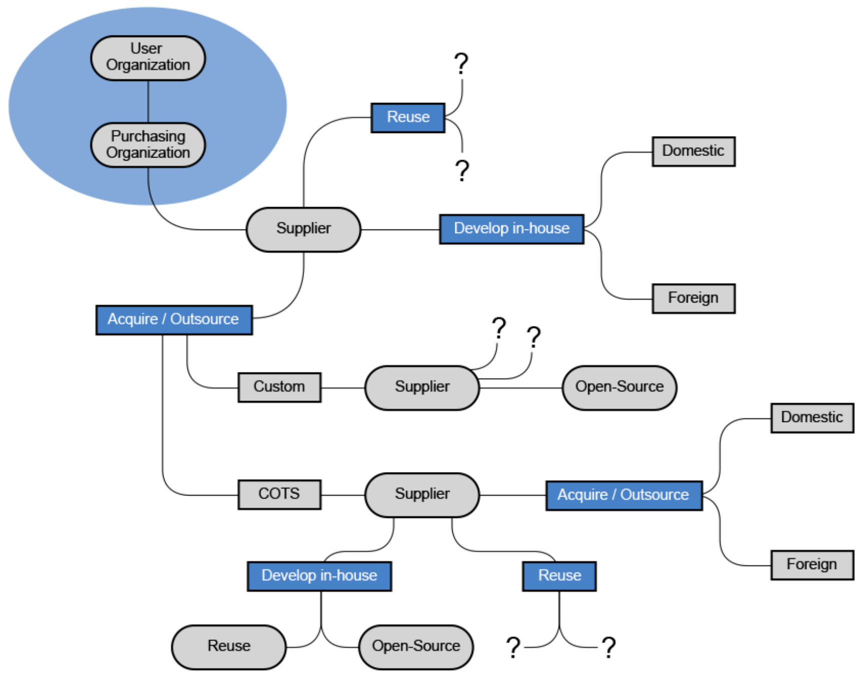
35% C++

53% C

Leverages COTS components

About 5% of the millions of on-board software lines of code come from the F-22, written in an Ada language first deployed nearly 40 years ago.

Software Supply Chain



Reuse of systems forms a complex software supply chain.

Systems are informally defined; quality attributes are rarely described at all.

Mix of public and private/proprietary repositories leads to licensing nightmares and emergent behavior.

Lots of functionality is "not used" but included in the final system.

Complexity leads to potential vulnerabilities; instability; brittle systems; and difficult, costly maintenance.

Case study: SolarWinds Cyber-Attack

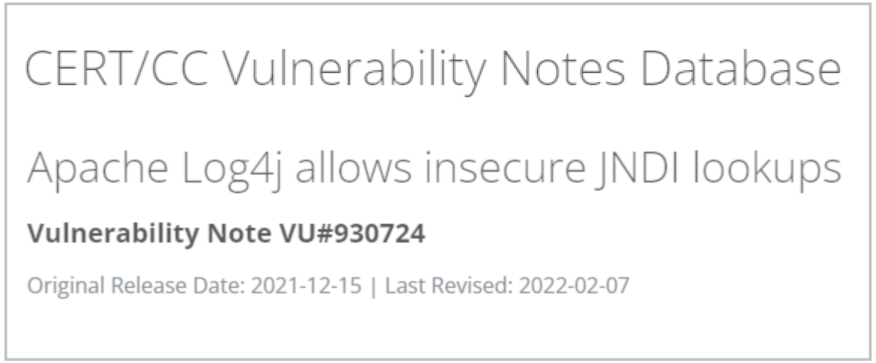
Intrusion that leveraged a commercial software application made by SolarWinds

Initiated by advanced persistent threat (APT) actors infiltrating SolarWinds supply chain by inserting a back door

Customers downloaded the Trojan Horse installation packages from SolarWinds, giving attackers access the systems running the SolarWinds product(s).

Result: emergent vulnerable infrastructure across many companies and sectors

Case study: Apache Log4j zero-day exploit



CERT/CC Vulnerability Notes Database

Apache Log4j allows insecure JNDI lookups

Vulnerability Note VU#930724

Original Release Date: 2021-12-15 | Last Revised: 2022-02-07

SEI CERT Coordination Center (CERT/CC) Vulnerability Note on Apache Log4j. For all CERT/CC vulnerability notes, visit https://kb.cert.org/vuls/bypublished/desc/

Highlights software supply chain security

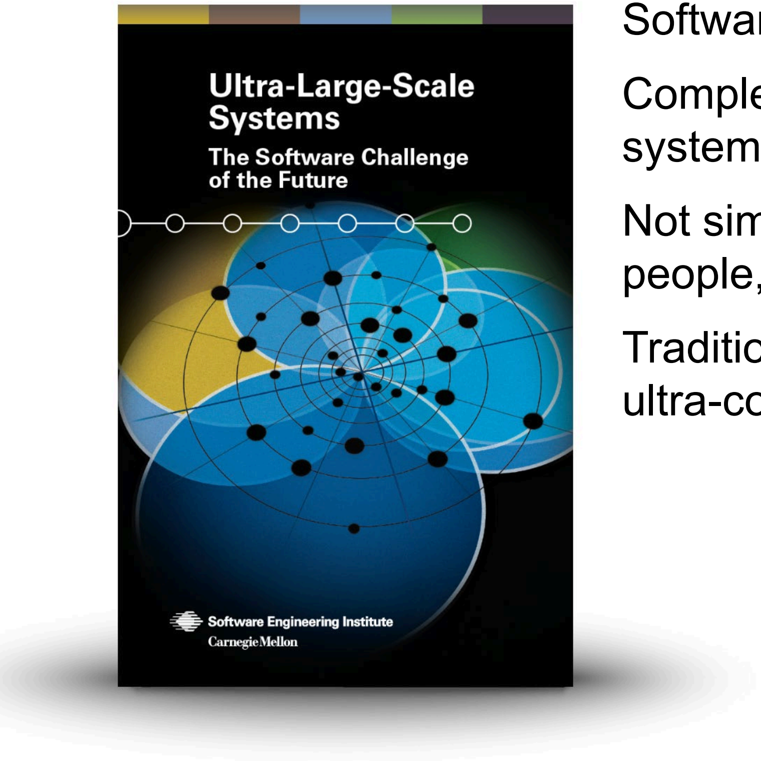
Java-based logging utility

Commonly used by apps and services across the internet

Flaw allows hackers to run code on vulnerable machines or hack into any application directly using the Log4j framework

Not the first or last time this was discovered after wide-scale distribution in an ultra-large-scale system

Ultra-large-scale systems


Ultra-Large-Scale Systems
The Software Challenge of the Future
Software Engineering Institute
Carnegie Mellon

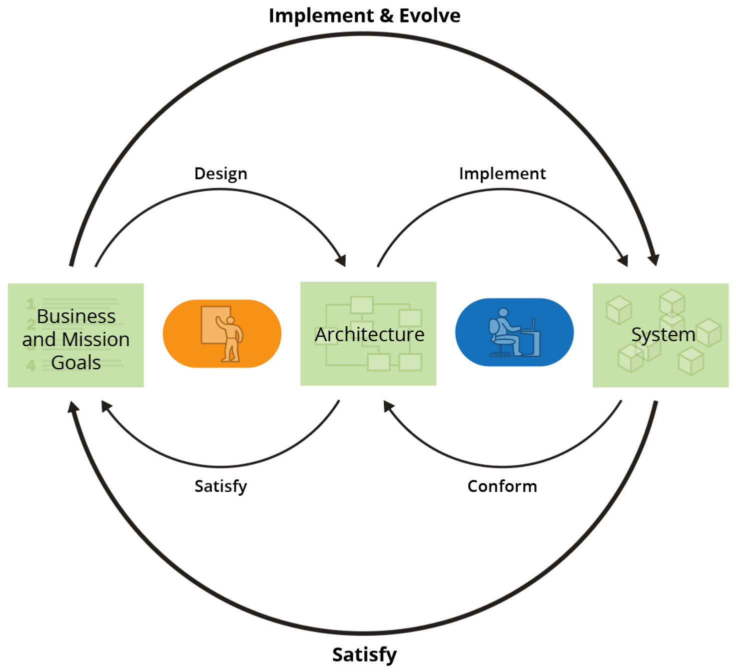
Software-intensive

Complex system-of-systems where components of the system are always changing

Not simply bigger: Interdependent webs of systems, people, policies, economics, cultures

Traditional, centralized engineering no longer adequate for ultra-complex systems

Quality (aka non-functional) attributes



Describe visible properties of a software system and the expectations for that system’s operation

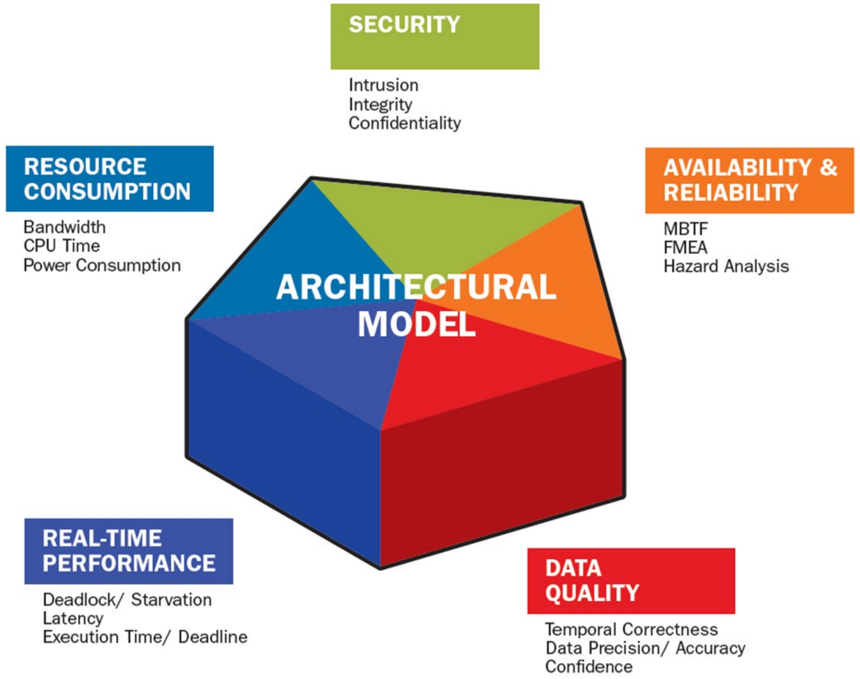
Define how well a system should perform some action

Influence decisions about software architecture: Designers need to analyze trade-offs between attributes to satisfy user requirements

Often called "ilities"— such as availability, reliability, maintainability, deployability

Some qualities may arise from emergent behavior of a complex system and be difficult to characterize

Quality attribute as first-class object

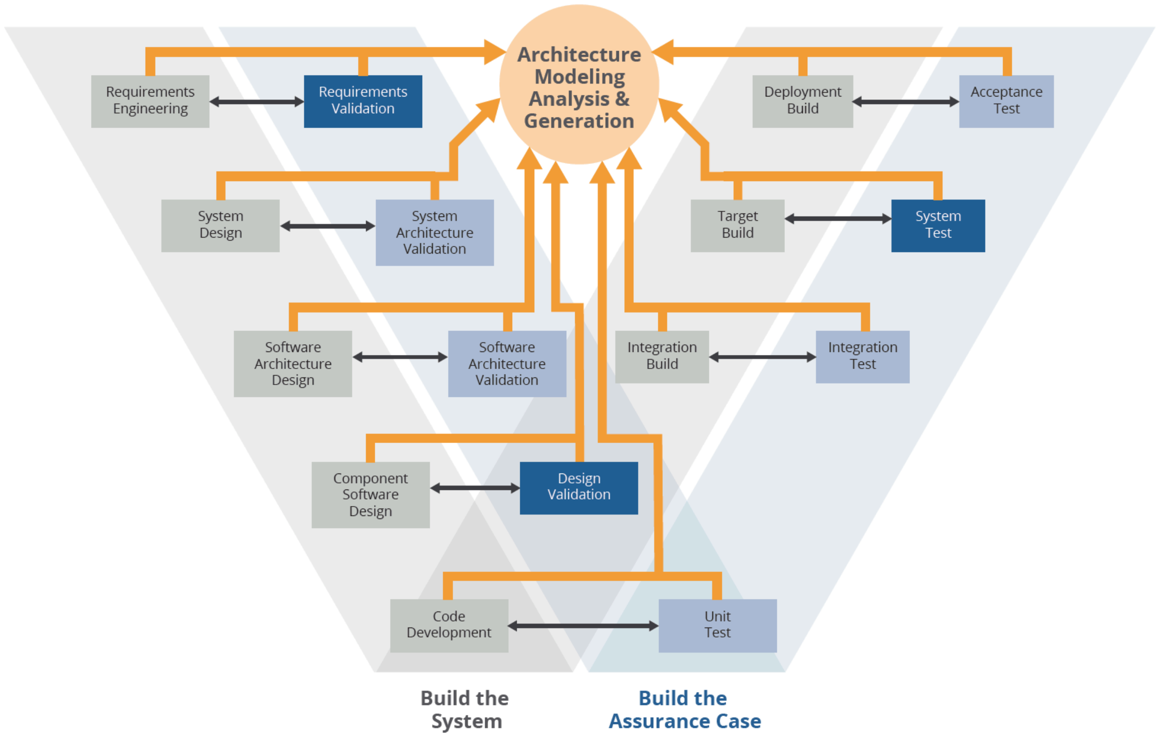


In programming, first-class object is a function or variable that operates as other entities in a language.

An architectural model provides a framework to reason about software quality attributes.

Reasoning About Software Quality Attributes (cmu.edu)

SEI integration of SWE, AI, and Cyber



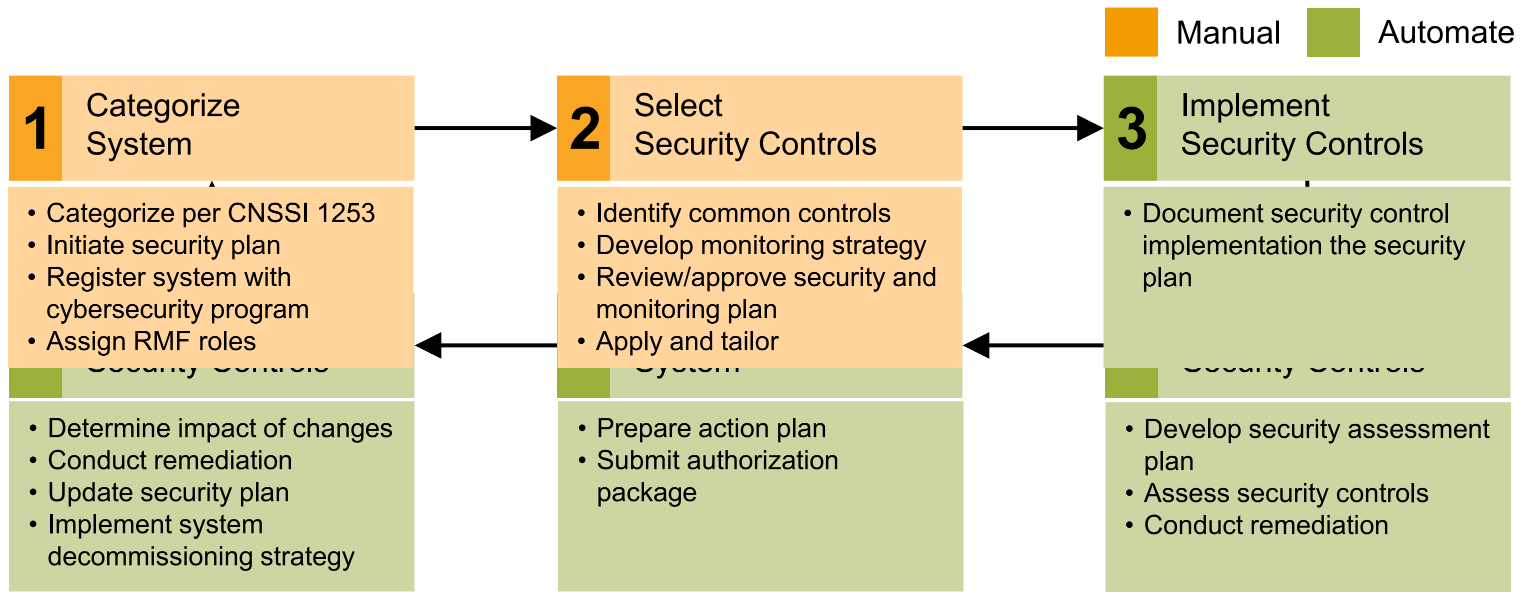
Answer to software complexity

Model-based system engineering using architecture modeling of software and hardware

AI for system analytics, evolution, security enforcement, affordability

Supports continuous integration and delivery processes

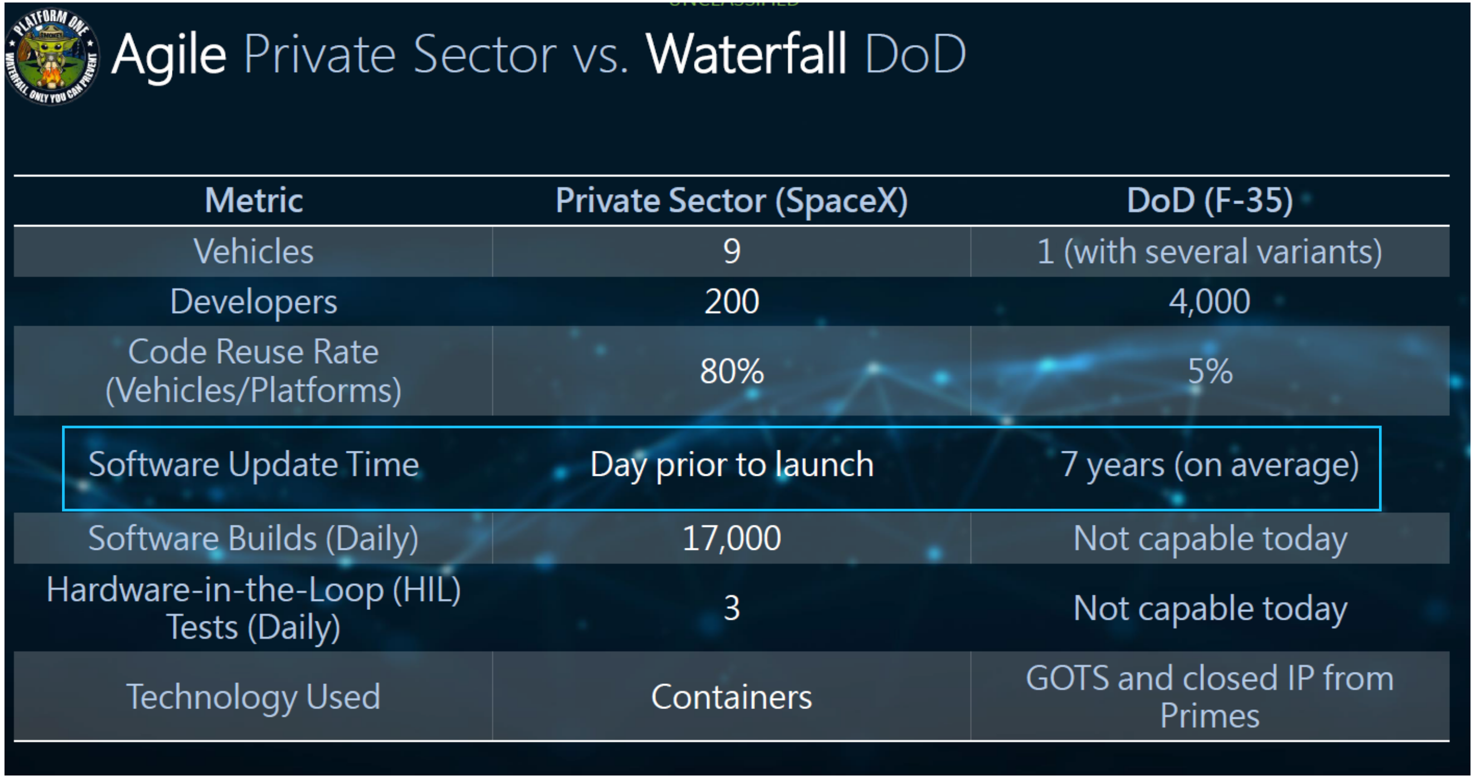
Assurance arguments, T&E, and Continuous ATO



🟧 Manual
🟩 Automate
1 Categorize System
• Categorize per CNSSI 1253
• Initiate security plan
• Register system with cybersecurity program
• Assign RMF roles
2 Select Security Controls
• Identify common controls
• Develop monitoring strategy
• Review/approve security and monitoring plan
• Apply and tailor
3 Implement Security Controls
• Document security control implementation the security plan
• Determine impact of changes
• Conduct remediation
• Update security plan
• Implement system decommissioning strategy
• Prepare action plan
• Submit authorization package
• Develop security assessment plan
• Assess security controls
• Conduct remediation

- ATO: Authority to operate—approval for information systems
- Federal agencies expend considerable resources seeking ATO approval for information systems.
- Updates require reapproval and delay deployment.
- DevSecOps pipeline can automate processes for continuous ATO.

From The Role of DevSecOps in Continuous Authority to Operate*, an SEI Blog posted on October 4, 2021.*

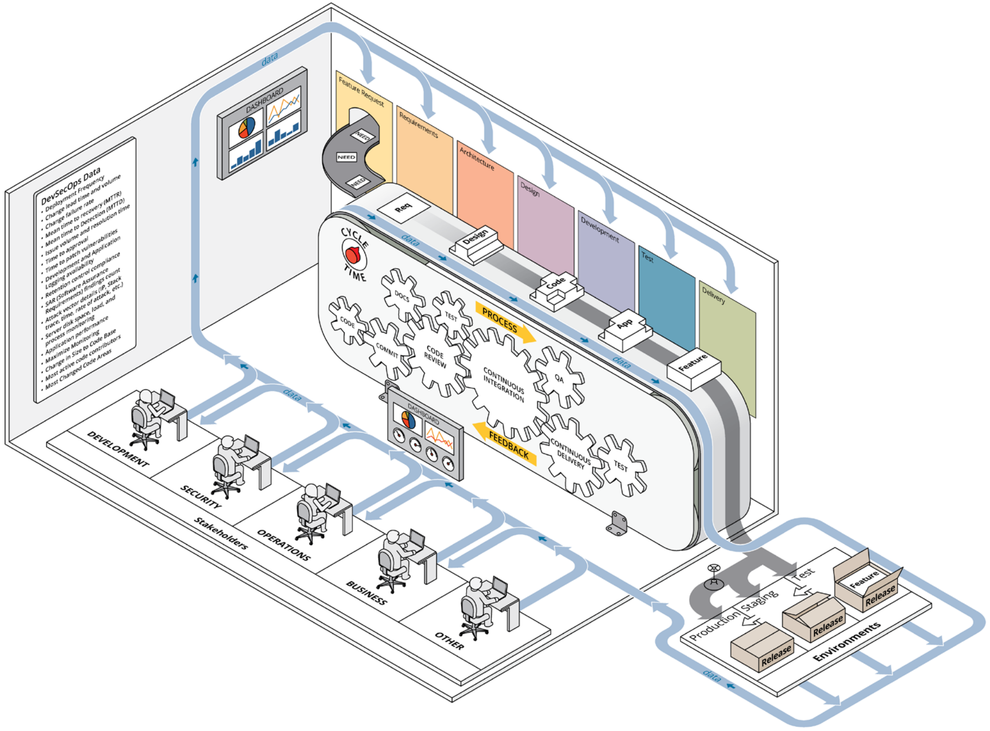


Agile Private Sector vs. Waterfall DoD

Metric	Private Sector (SpaceX)	DoD (F-35)
Vehicles	9	1 (with several variants)
Developers	200	4,000
Code Reuse Rate (Vehicles/Platforms)	80%	5%
Software Update Time	Day prior to launch	7 years (on average)
Software Builds (Daily)	17,000	Not capable today
Hardware-in-the-Loop (HIL) Tests (Daily)	3	Not capable today
Technology Used	Containers	GOTS and closed IP from Primes

*U.S Air Force DoD-wide DevSecOps Managed Services Platform **One** merges top talent from across the U.S. Air Force using various factories (Kessel Run, Kobayashi Maru, SpaceCAMP, and Unified Platform). It is an official DoD DevSecOps Enterprise Services team for the DoD.*

Looking ahead



Model-based engineering (MBE) linked with DevSecOps practices

Shift security left in SDLC to make continuous security (toward zero trust) a priority

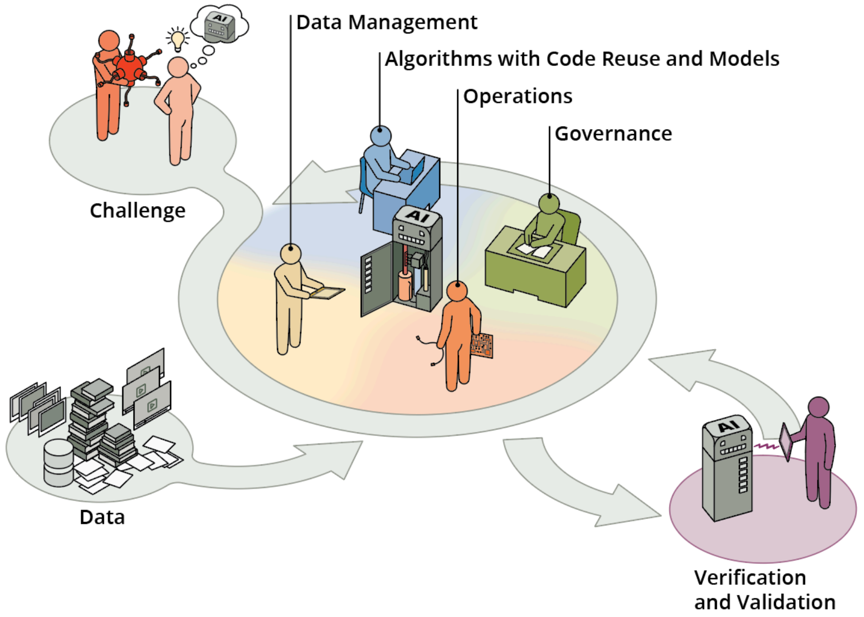
Incorporate operational feedback into MBE and DevSecOps

MBE and secure code in a single pipeline

Reusing models rather than code to start with

Updating code becomes a customization activity

Incorporation of AI


Data Management
Algorithms with Code Reuse and Models
Operations
Governance
Challenge
AI
Data
Verification and Validation

Automation and autonomy in key cyber tradecraft areas

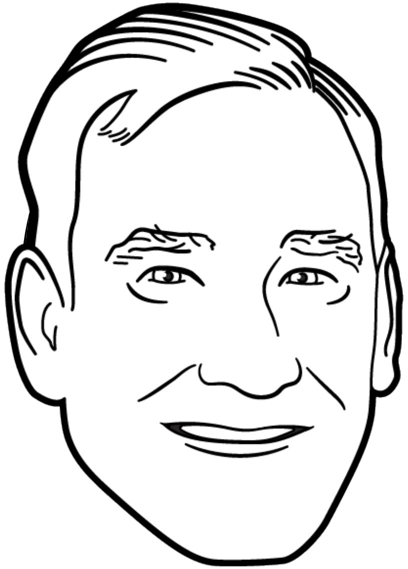
Trustworthiness

Risk bounds for AI/ML

New architecture AI solutions across software lifecycle

An AI Engineering discipline

AI components that are engineered to the size and complexity of future system needs

CDE