Agile Overview

Will Hayes, Keith Korzec
17 May 2022

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM22-0431

Agenda

Defining Agile

Agile in Government

What's New? What's Different?

// Break

Difficult Learning

Discussion

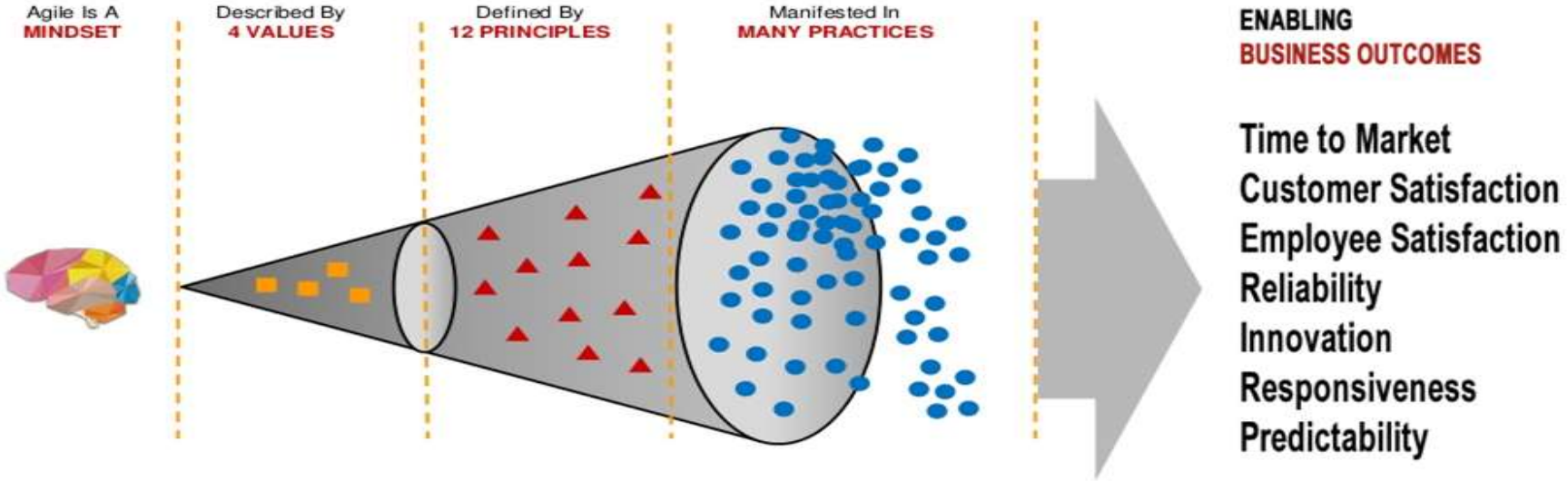
Under 40 Slides for a 4-Hour Session = Please Speak Up

Working Definition of Agile

Agile An *iterative* and *incremental* (evolutionary) approach to software development which is performed in a *highly collaborative manner* by *self-organizing teams* within an *effective governance framework* with *"just enough" ceremony* that produces *high quality software* in a *cost effective and timely* manner which *meets the changing needs of its stakeholders*. [Ambler 2013]

[Ambler 2013] Ambler, Scott. *Disciplined Agile Software Development: Definition*.
http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

What is Agile?

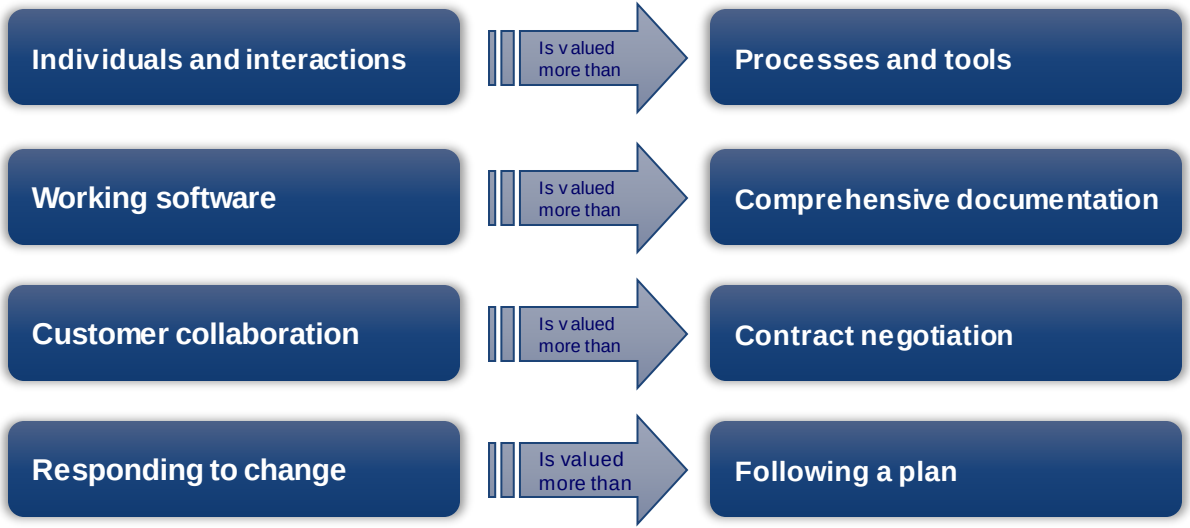


Source: https://www.slideshare.net/MichaelTarnowski/agile-mindset-for-executives

Implementing practices meant for a given environment, outside of that environment, could make things worse – 'installing Agile' doesn't fix anything

Agile Values from Manifesto for Agile Software Development

While there is value in the items on the right, **we value the items on the left more.**



Which side do you think will benefit your users more?

http://agilemanifesto.org

Major Program Challenge to Overcome

The Perfect Plan is Unattainable

Agile In Government

Agile in the Larger DoD Context

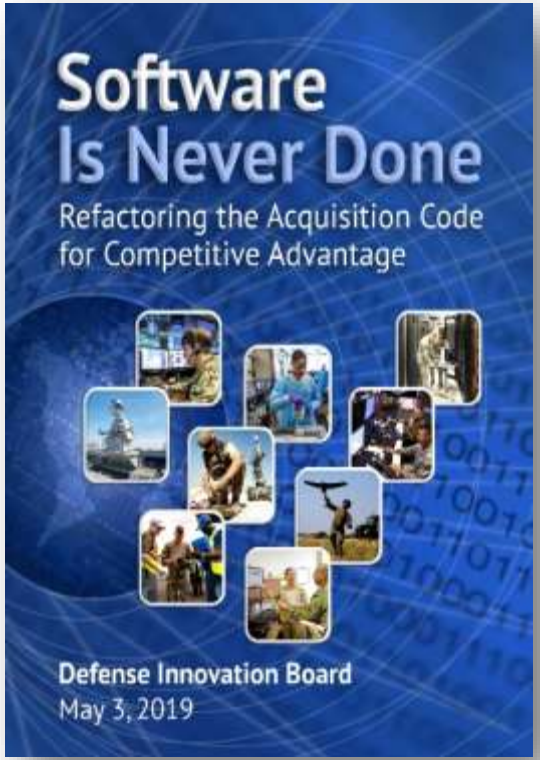
DoD continues to track anecdotal success and failures of programs using Agile

2019 Defense Innovation Board Software Acquisition Practices Study (referred to as DIB SWAP Study) frames challenges in the eco-system that slow down Agile/Lean adoption (blockers)

Agile/Lean Successes:

- 2017: JIDO (Joint Improvised-Threat Defense Orgn) reduced time for critical software delivery capability from 6 months to 12 days with a 9X increase in deployment frequency and a 90% reduction in operating costs
- 2018-2019: 4 of 7 programs piloting Agile approaches delivered working capabilities to USERS within three months
  - The other 3 of 7 delivered working capabilities to INTEGRATION TESTING for larger capabilities within three months
- 2019: Space Force Space C2 program developed and fielded two applications within six months of initiation; one of these documented 4 hours per shift time savings for over 1000 users

2019 DIB SWAP Study



Strong influence on the 2019 and 2020 National Defense Authorization Acts

Also home of the “Agile BS” Appendix

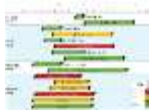
Main lines of effort:

- **Congress and OSD: Refactor statutes, regulations, and processes for software**, providing increased insight to reduce the risk of slow, costly, and overgrown programs, and enabling rapid deployment and continuous improvement of software to the field.
- **OSD and the Services: Create and maintain cross-program/cross-Service digital infrastructure** that enables rapid deployment, scaling, testing, and optimization of software as an enduring capability; manage them using modern development methods; and eliminate the existing hardware-centric regulations and other barriers.
- **Services and OSD: Create new paths for digital talent (especially *internal* talent)** by establishing software development as a high-visibility, high-priority career track and increasing the level of understanding of modern software within the acquisition workforce.
- **DoD and industry: Change the practice of how software is procured and developed** by adopting modern software development approaches.

Source: https://innovation.defense.gov/software/

Why Agile (and Lean and DevSecOps)?

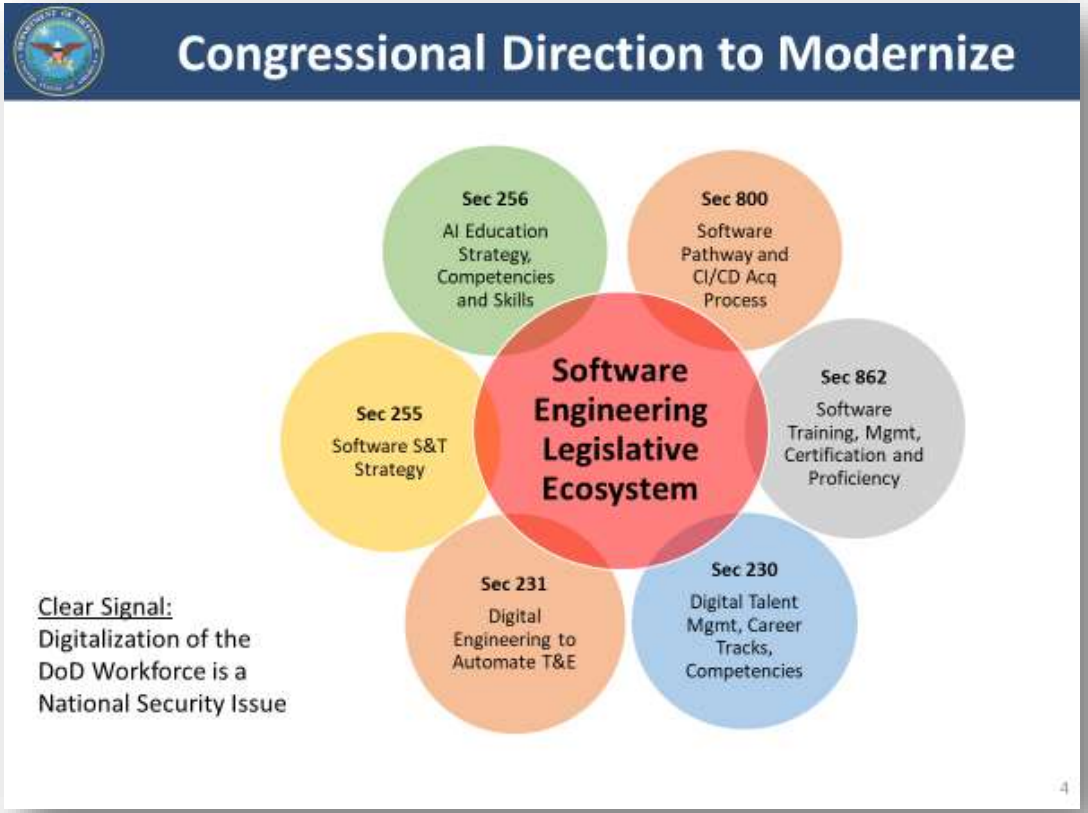
Direction from OSD across multiple NDAA's reinforces need for fast feedback all along the development path (Lean/Agile)



To assure time-certain delivery, parallelization of certification and development activities is needed (DevSecOps or Dev*Ops)

Longevity of system in a rapidly evolving threat space (Agile, Lean, DevSecOps)

Multiple Sources of SW Engineering Legislative Direction

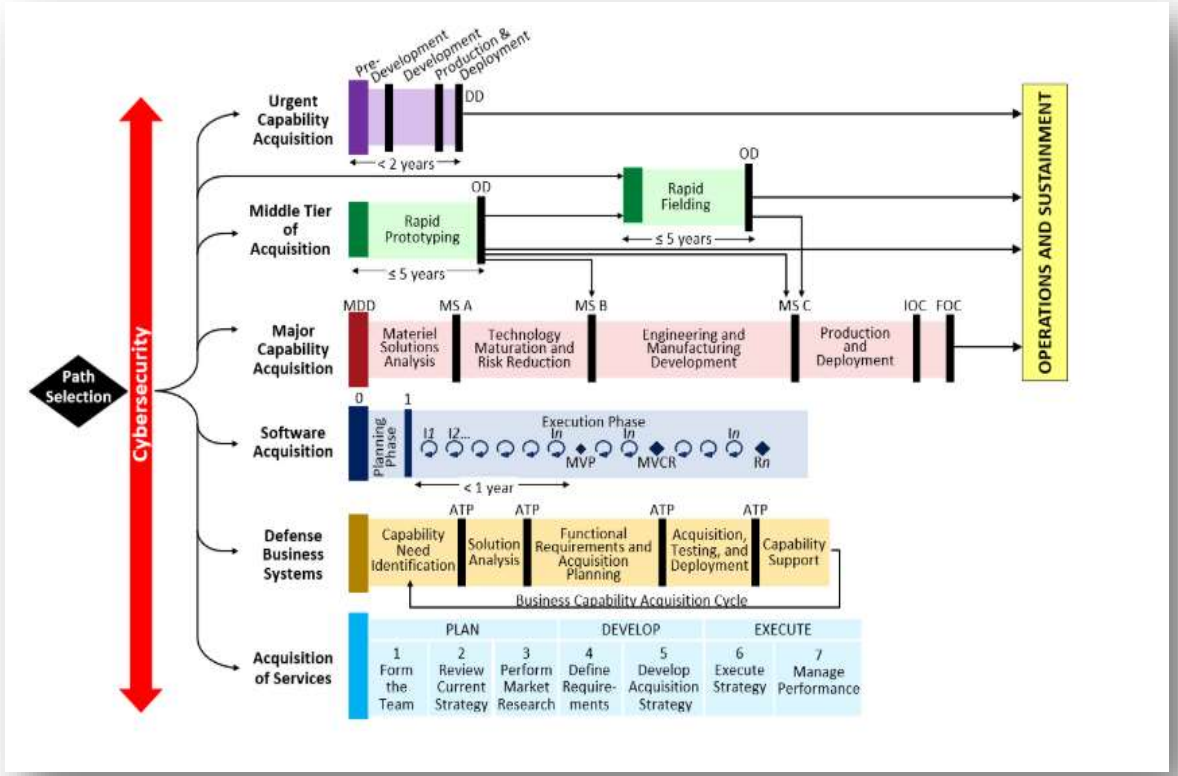


Source: Brady, Sean. *How Software Acquisition & DevSecOps Increase the Lethality of the DoD*, DSO Days, Oct 2020.

Adaptive Acquisition Framework

New ways of acquiring systems. Note the Software Acquisition Pathway (SWP) for software-dominant products.

https://aaf.dau.edu/



Key Elements of SW Acquisition Pathway

- Modern software development practices (Agile, DevSecOps, Lean)
- Capitalizing on active user engagement and enterprise services
- Software is rapidly and iteratively delivered to the operational environment to meet the highest priority user needs
- Tightly coupled mission-focused government-industry software teams
- Automated tools for development, integration, testing, certification

Source: DODI 5000.02 Section 4.2

10

If these resonate and more information is desired, we can follow-up with a SWP 101 briefing

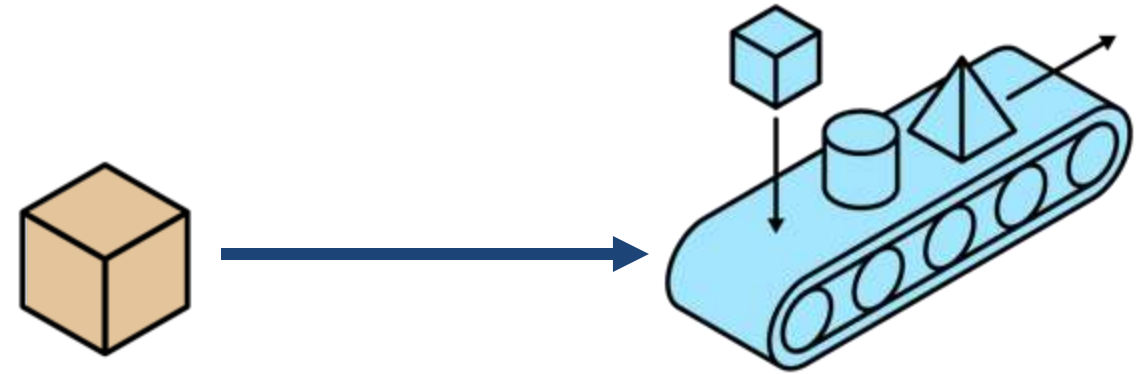
Source: Brady, Sean. *How Software Acquisition & DevSecOps Increase the Lethality of the DoD*, DSO Days, Oct 2020.

Major Program Challenge to Overcome

Unrelenting Needs, Limited Budget

What's New? What's Different? Let's Talk Requirements

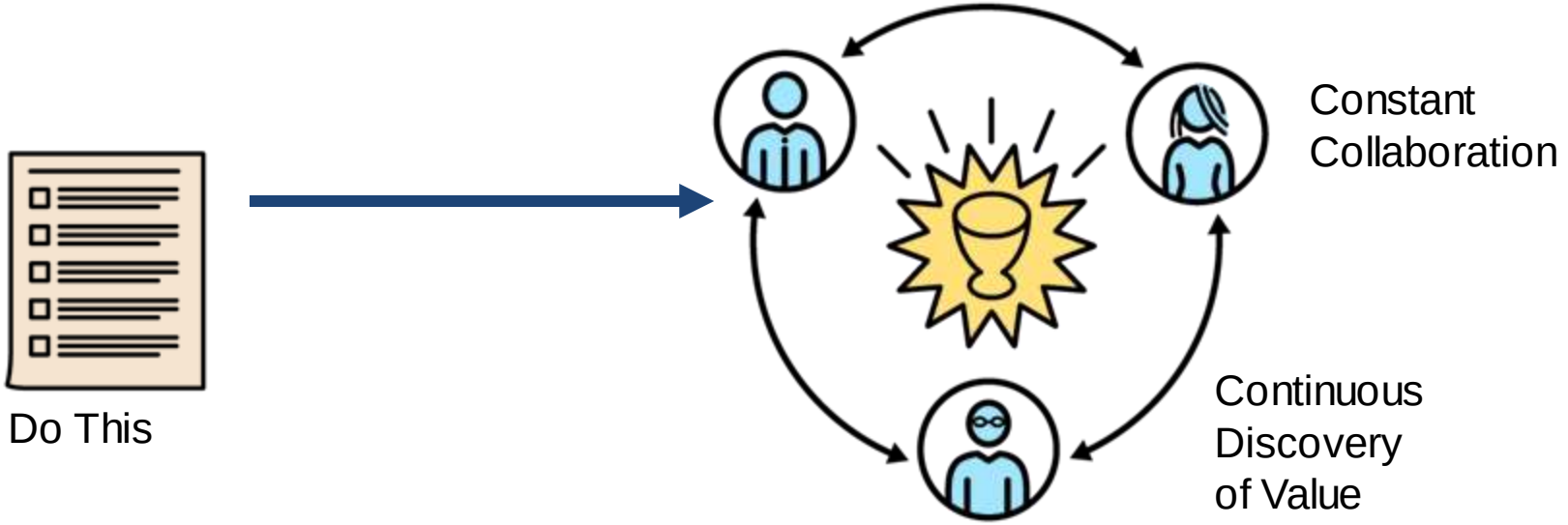
Agile Represents a Major Requirements Transition – Different Acquisition Objectives



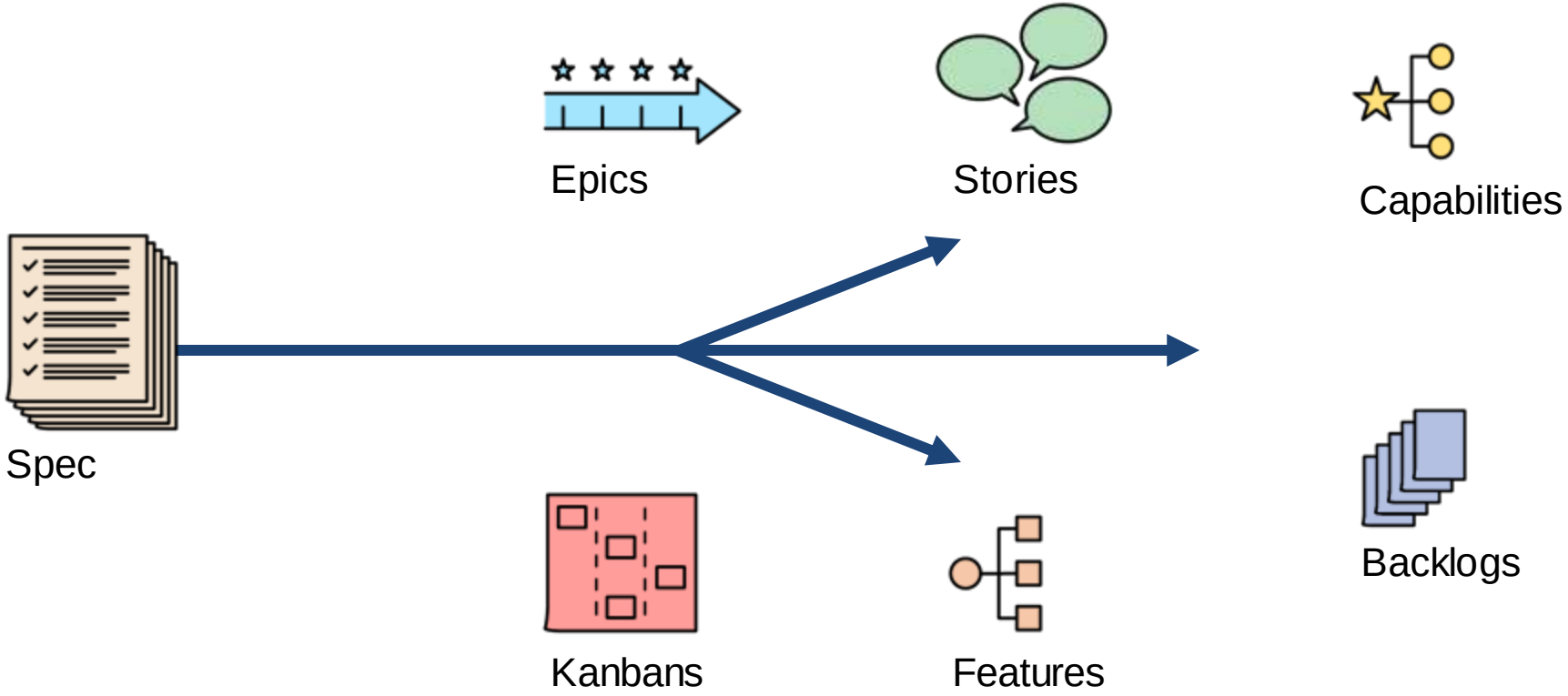
Buying a Box

Buying an ongoing delivery stream

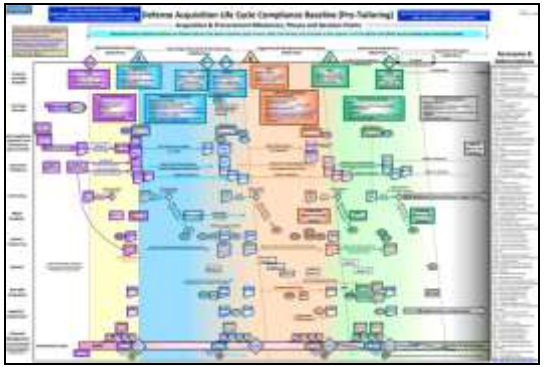
Agile Represents a Major Requirements Transition – Different Interactions



Agile Represents a Major Requirements Transition – Different Artifacts



Agile Represents a Major Requirements Transition – Different Practices for Requirements Management



Date-based Milestones

Increment-Based Demos

Definition of Done

Team Increment

System Increment

Solution Increment

Release

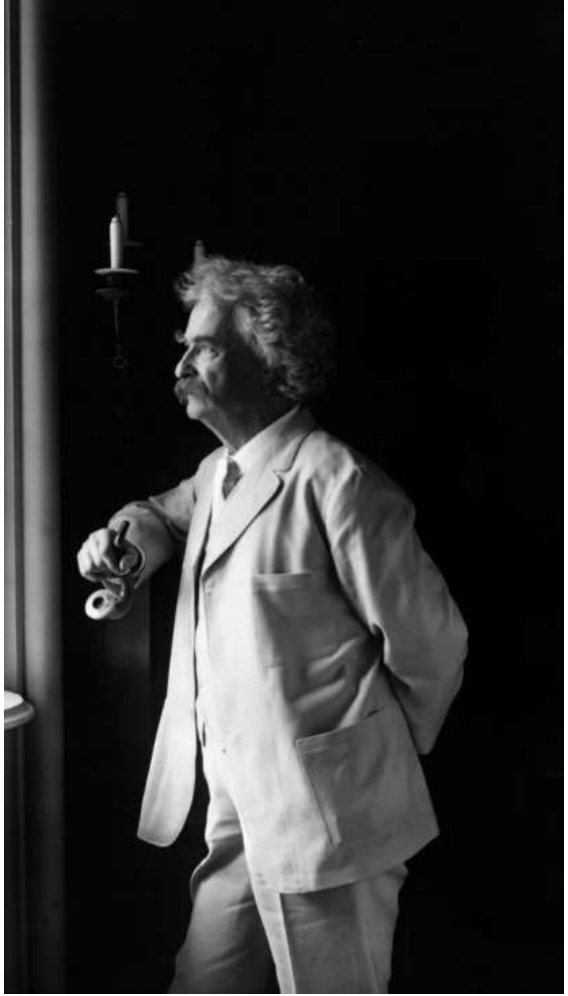
Are We On The Right Path?

Logical Breakpoint

Major Program Challenge to Overcome

It ain’t what you don’t know that gets you in trouble. It’s what you know for sure that just ain’t so.

Samuel Langhorne Clemens

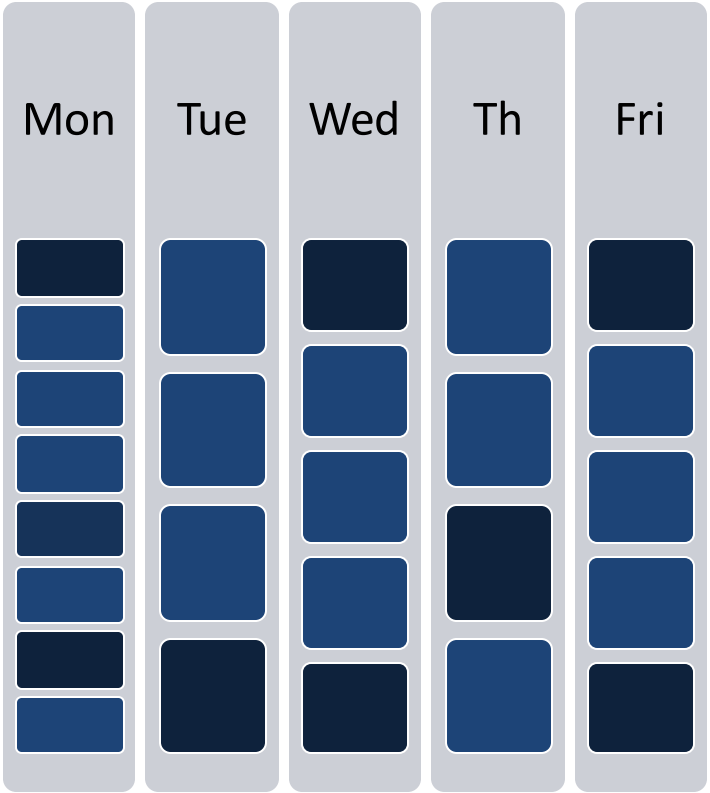


Major Program Challenges to Overcome

New Lessons Being Learned

Flow

Value Flow: Utilization is the Wrong Goal



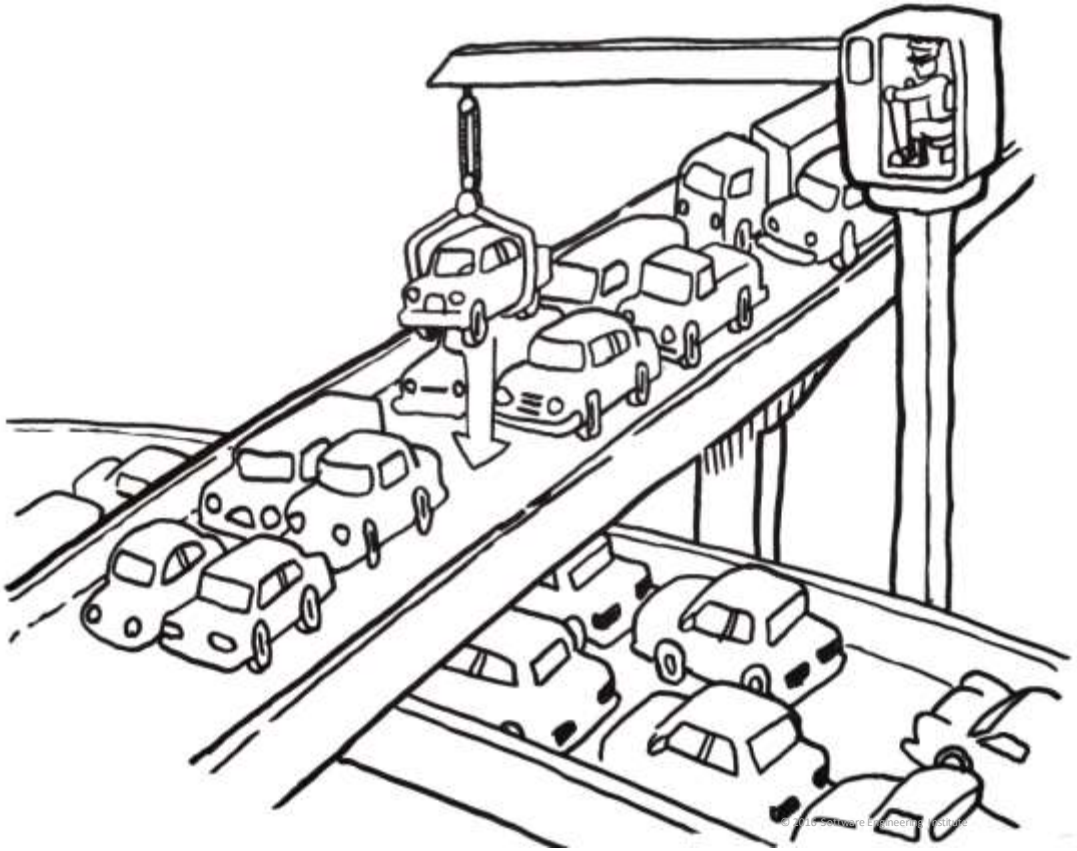
100% Utilization:

- Magnifies the impact of variation
- Maximizes task-switching overhead
- Assures slower overall progress

Change is inevitable, plan to learn

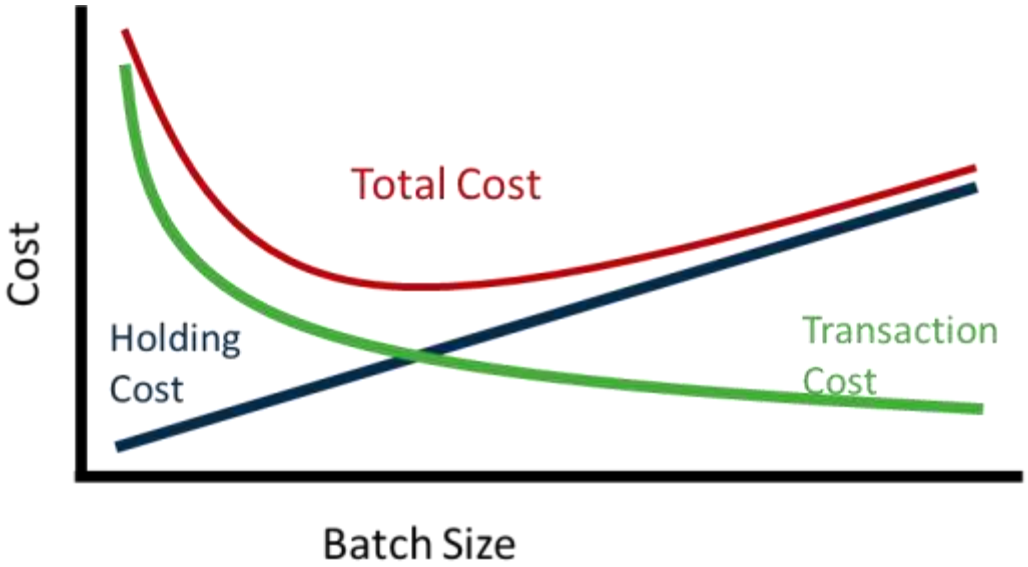
Multi-tasking is a myth we don't accurately comprehend

Maximum Utilization is Counterproductive

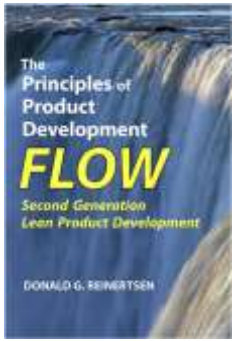


Batch Size

Economies of Batch Size



U-Curve optimization problem as described in *Principles of Product Development Flow*, by Don Reinertsen



Iterative & Incremental

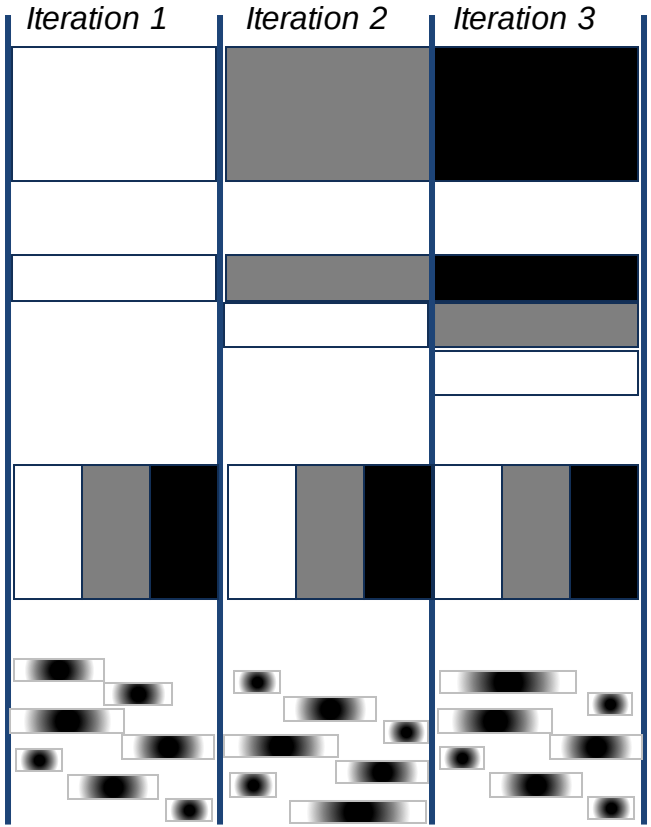
Iterative Development

A stronger break between major process steps used to define iterations

Divide into multiple batches, performing a single lifecycle step per iteration on each

Same division of work, completing all work on each batch during each iteration

Decomposing work into smaller packages, leading to multiple start-to-finish cycles in each iteration. This is the intent of Agile.



Incremental Development/Delivery

Not Incremental

- Single increment of work, delivered once in a single package

Incremental Development, Single Delivery

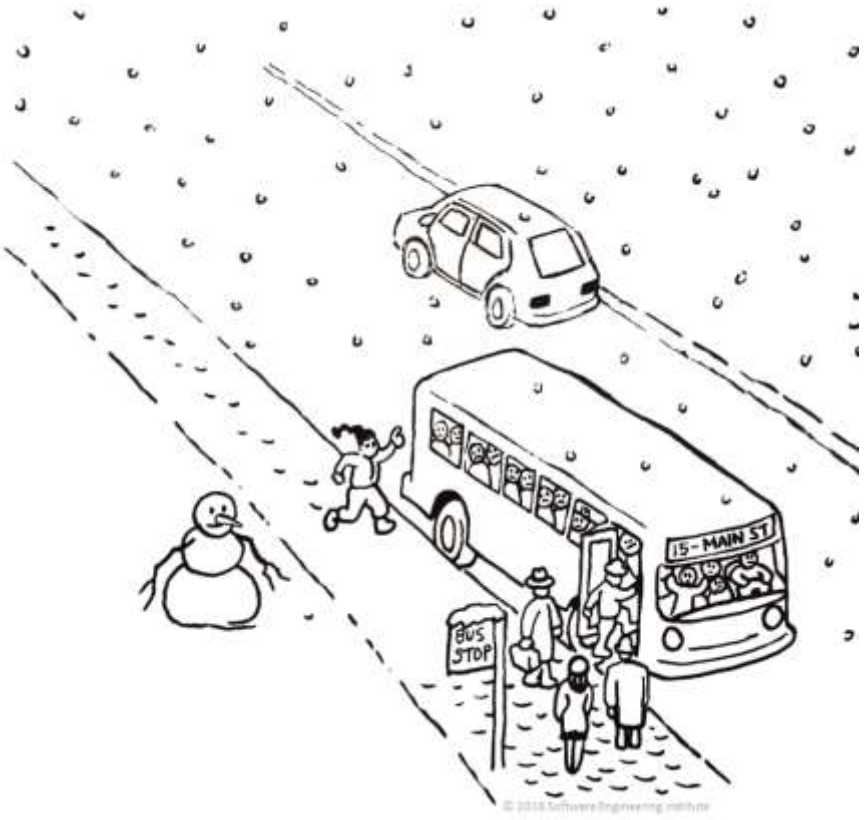
- Work divided into logical subsets for development in pieces, delivered once in a single package

Incremental Development & Delivery

- Work divided into meaningful slices of the total end result, delivered in gradually more complete versions
- Alternatively, delivering new pieces rather than total new versions

Cadence

Cadence and Predictability

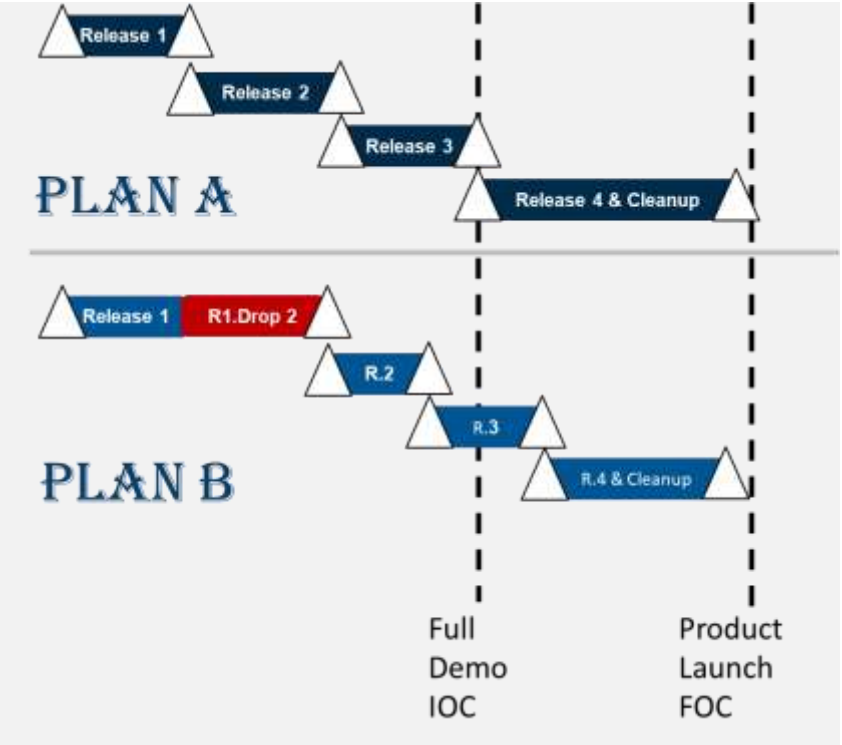


A Late Bus:

- Makes people scramble to get aboard
- They don't know when the next one will get here

Then the next bus comes along empty

Late Releases Become “Feature Magnets”



As things start to slip

- Influential people get ‘their priorities’ moved up, rather than deferred
- Pressure increases on early releases
- Functions slated for final release can’t be guaranteed…

Understanding Progress

Large Program Algebra

Please Share Your Thoughts

Potential Walk-On Topics

Backup Material

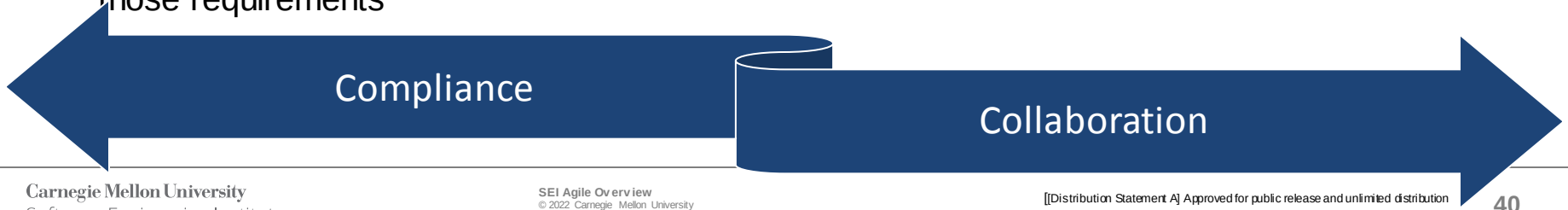
Bottom Line for Programmatics of Complex Systems

OVERSIGHT is an element of what makes the DoD Acquisition Ecosystem Work

- Oversight mechanisms established by program management determine:
  - Nature of information made available
  - Frequency of communication
  - Urgency/importance
- Well-established procedures & templates convey oversight requirements
  - Recent developments like Adaptive Acquisition Framework change some of those requirements

INSIGHT is a necessary ***enabler*** to effective oversight

- Well-established CDRLs and DIDs may not always the best source of insight
  - Aversion to all off-nominal conditions
  - Conformance to plan becomes the goal
- Agile development settings promote transparency and ongoing insight
  - ***Available mechanisms, however, require proactive participation from the acquirer to be effective***



Batch Size

Typical Large Batch Realities:

- "Nothing is done until everything is done"
- More Work in Progress is good
- 100% utilization of resources is a goal
- Optimistic reporting of progress in order to "keep the program sold"
- Large scope integration events identify defect levels that strain resources
  - Increases number of potential defects that affect multiple areas of the system
  - Reduces confidence in system robustness
  - Harder for engineers to find sources of defects
- Tendency toward "test quality in"

Aspirations for Small Batch:

- We can learn from even small pieces being implemented/done
- "Stop starting, start finishing"
- Work in Progress is limited to enhance flow through the system
- 100% utilization of resources is recognized as limiting flow, flexibility, and work accomplishment
- Short time between when a defect is found and when it was created
  - Root cause analysis easier with current work, rather than work done in the past
- LOTS of integration happening across entire system, building confidence
- Tendency to "build quality in"

Feedback

Typical of "Primarily Document" Focus:

- Prefer larger, less frequent demos
- Requirements documents seen as "ground truth" for user needs, even when known to be superseded
- Constraints on opportunities for feedback
- Rushed feedback on documents
- More investment in documenting "to be" state than in documenting "as built"
  - Using documents to "lock down" design,
  - Then struggling to keep them current?

Aspirations for a Broader Aperture:

- Recognition that demo doesn't EQUAL test, but INFORMS it
- Stakeholder participation in demos of small pieces of functionality
- Open, continuous feedback about both the fact of and the meaning of progress or lack thereof
- Info from demos is fed forward to testing and certification staff to ensure alignment
- Definition of Done that includes certification needs (cyber, DT/OT, ATC, ATO, etc.)
- Participation on continuous integration team by govt staff seen as a high priority

Requirements

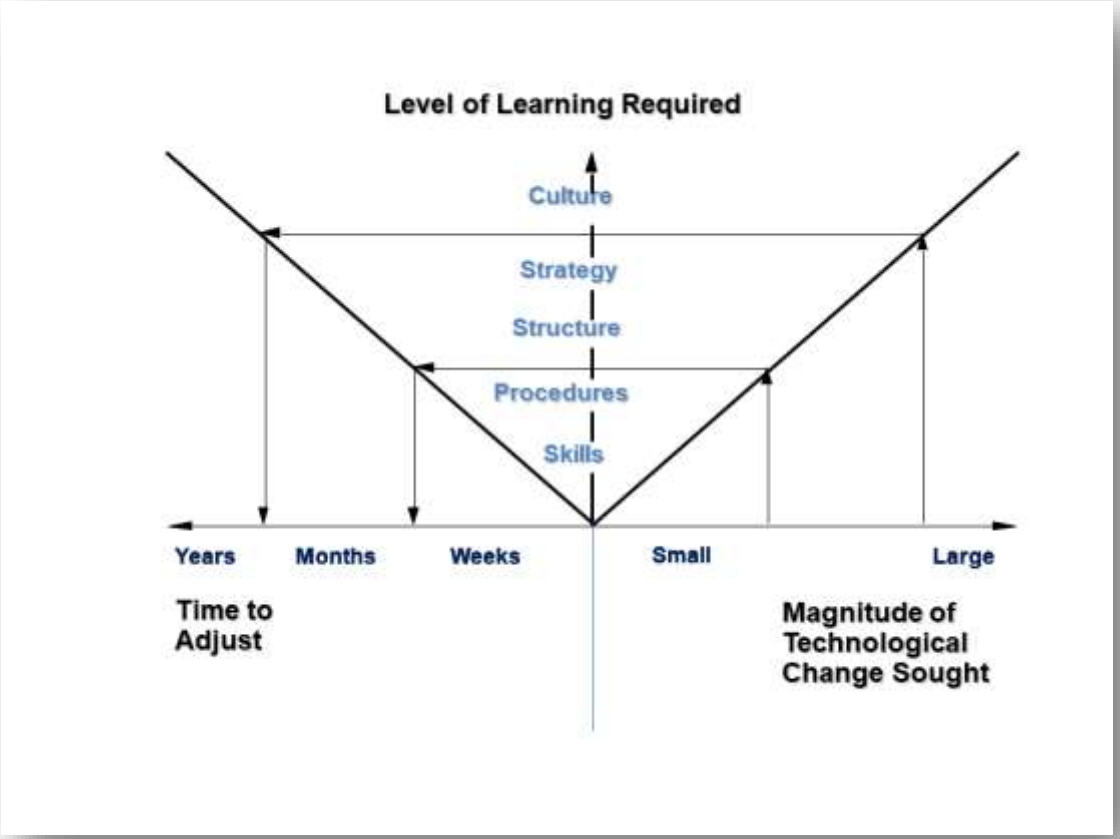
Typical of "Single Delivery" Focus:

- Work must commence early to limit risk
- Narrow time window to set the baseline
- Increasing resistance to requirements change over time, though knowledge of real user need continues to evolve
- Favoring breadth over depth in reviews
  - Hard to take in the large requirements set
  - Time for "digging in" on critical issues is rarely available during the review
- Sometimes we get as far as we can, declare success, and track action items until the next event

Aspirations for Iterative Approach:

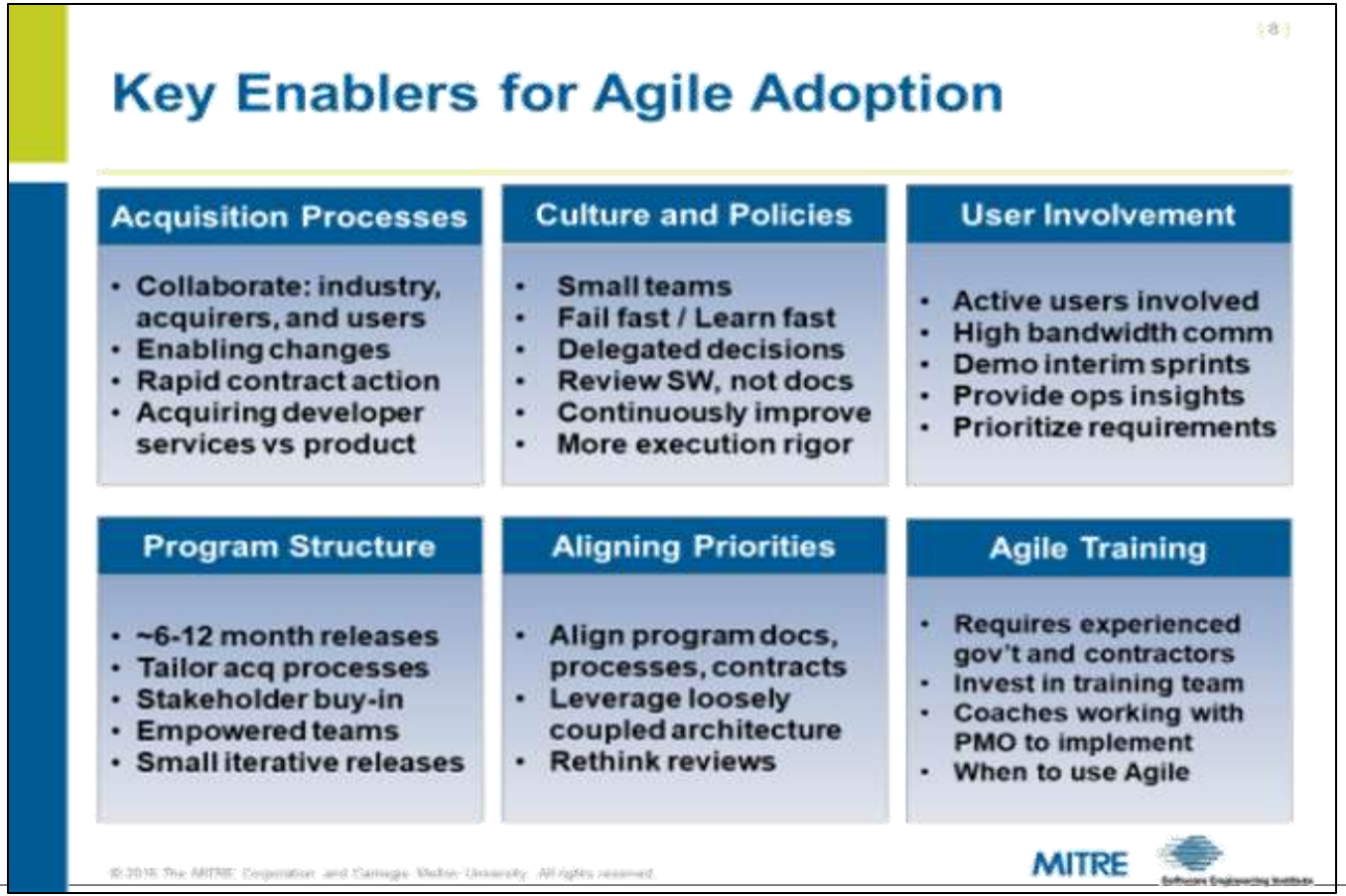
- Mix of "push" and "pull" communication across govt/contractor interface as requirements are elaborated/refined
  - Facilitated by workflow and collaboration tools
- Frequent high bandwidth meetings keep the relationship going, not just technical work
- Transparency among stakeholders is an essential ingredient to build trust
- Frequent small batch prioritizations build a solid base of understanding of current state and progress

Culture Change will take Time and Effort!



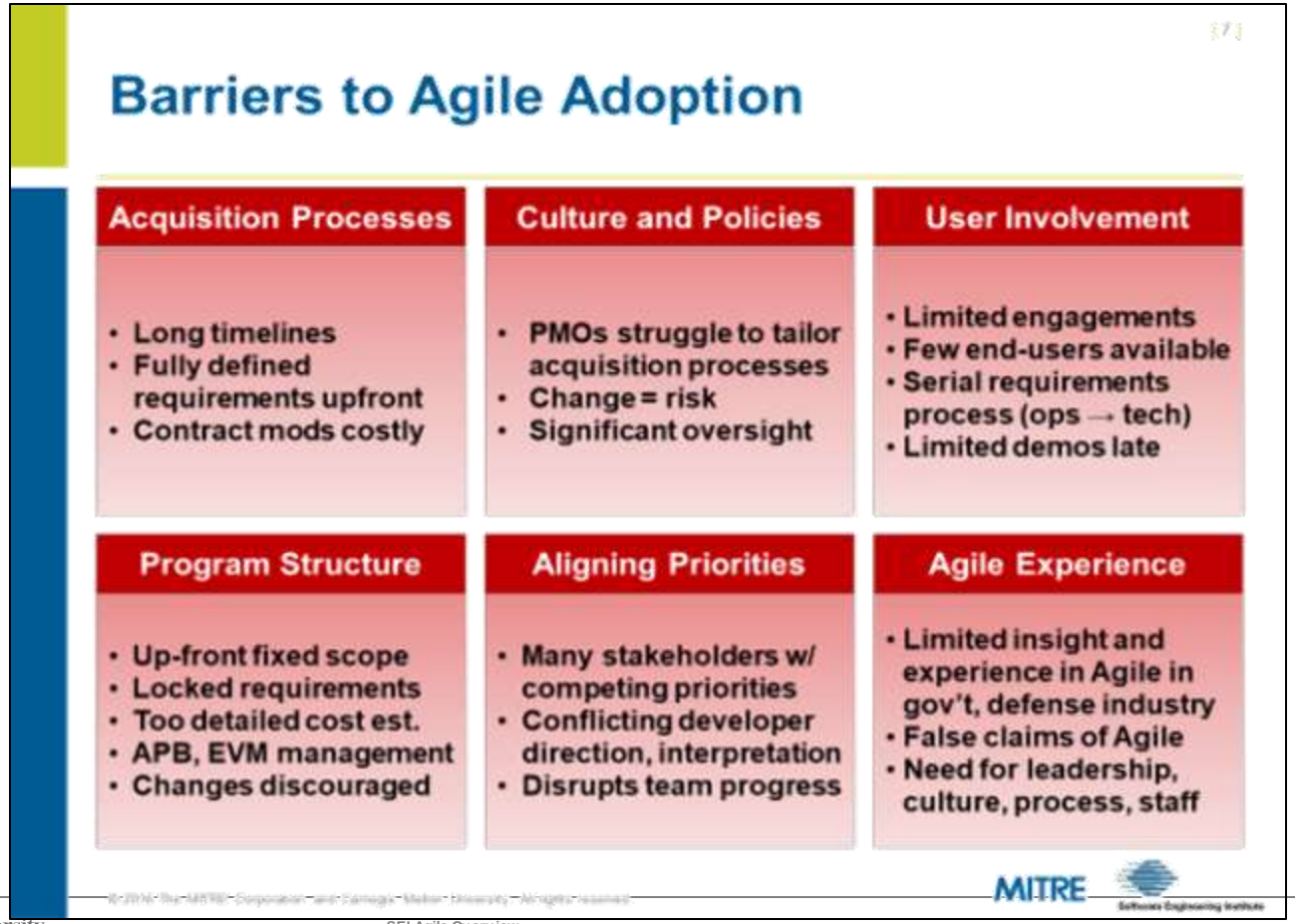
Source: Adler, Paul. "Adapting Your Technological Base: The Organizational Challenge", Sloan Mgmt Review, 1990.

Some Advantages We Want to Leverage



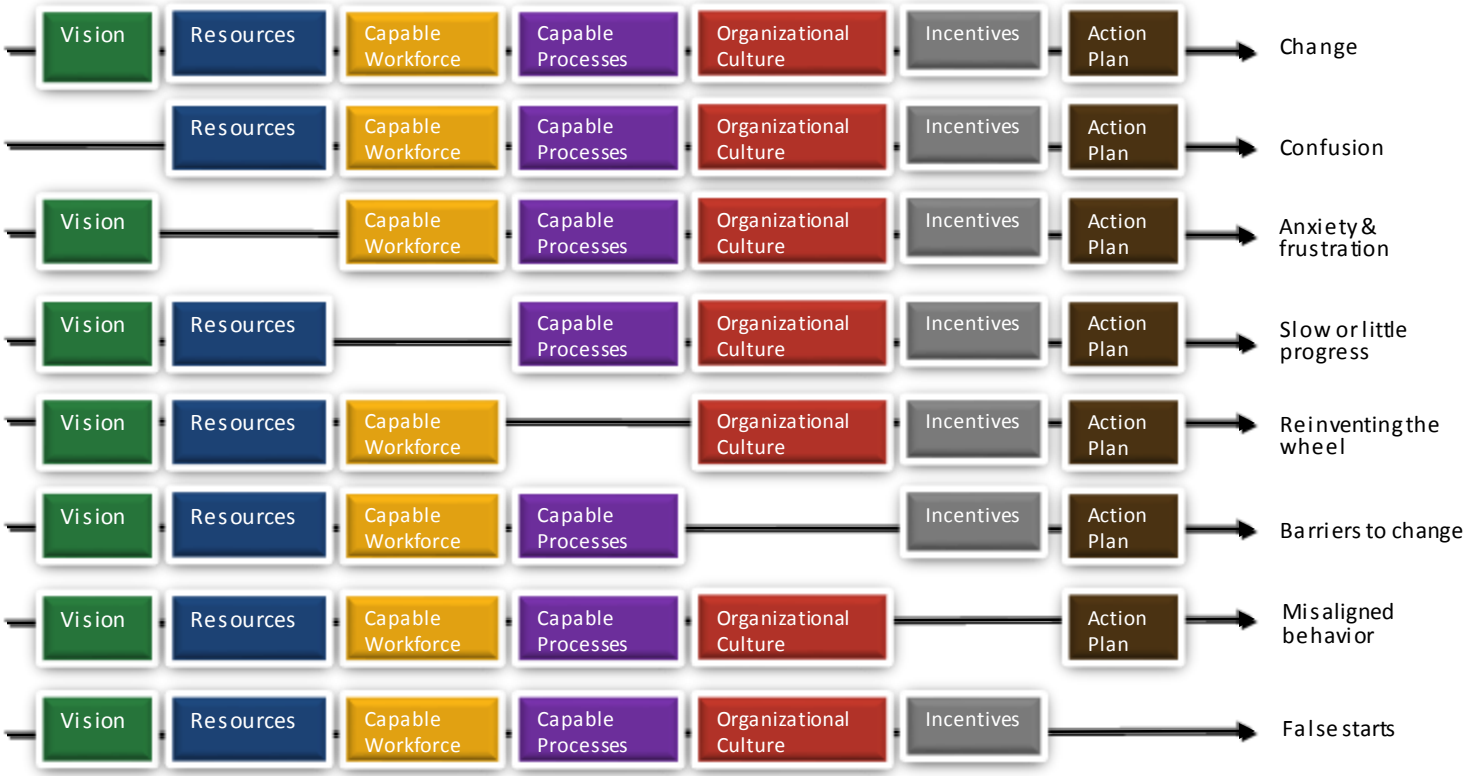
Source: 2016 briefing to General E. Pawlikowski on USAF Agile Adoption, SEI & Mitre

Some Barriers We May Run Into



Source: 2016 briefing to General E. Pawlikowski on USAF Agile Adoption, SEI & Mitre

Success Factors in Adopting New Practices: What Can We Learn?



The symptoms on the right can be used to diagnose what might be missing in our adoption support approach

Adapted by Buttles (2010) from:
Delorise Ambrose, 1987

Contact Information

Will Hayes

Principal Engineer
Agile Transformation Team Lead
Software Engineering Institute

wh@sei.cmu.edu

Keith Korzec

Senior Engineer
Agile Transformation Team
Software Engineering Institute

kkorzec@sei.cmu.edu