Agile in Government: Executive Overview

May 2018

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.

DM18-0857

Today's agenda

Today's landscape
Agile basics: let's get a consistent vocabulary

Beyond the small team: Agile in the larger ecosystem

Special issues in the government & DoD:
- **Oversight/measurement**
- **Technical reviews**
- **Contracting Questions**

How do we get there: enabling Agile culture

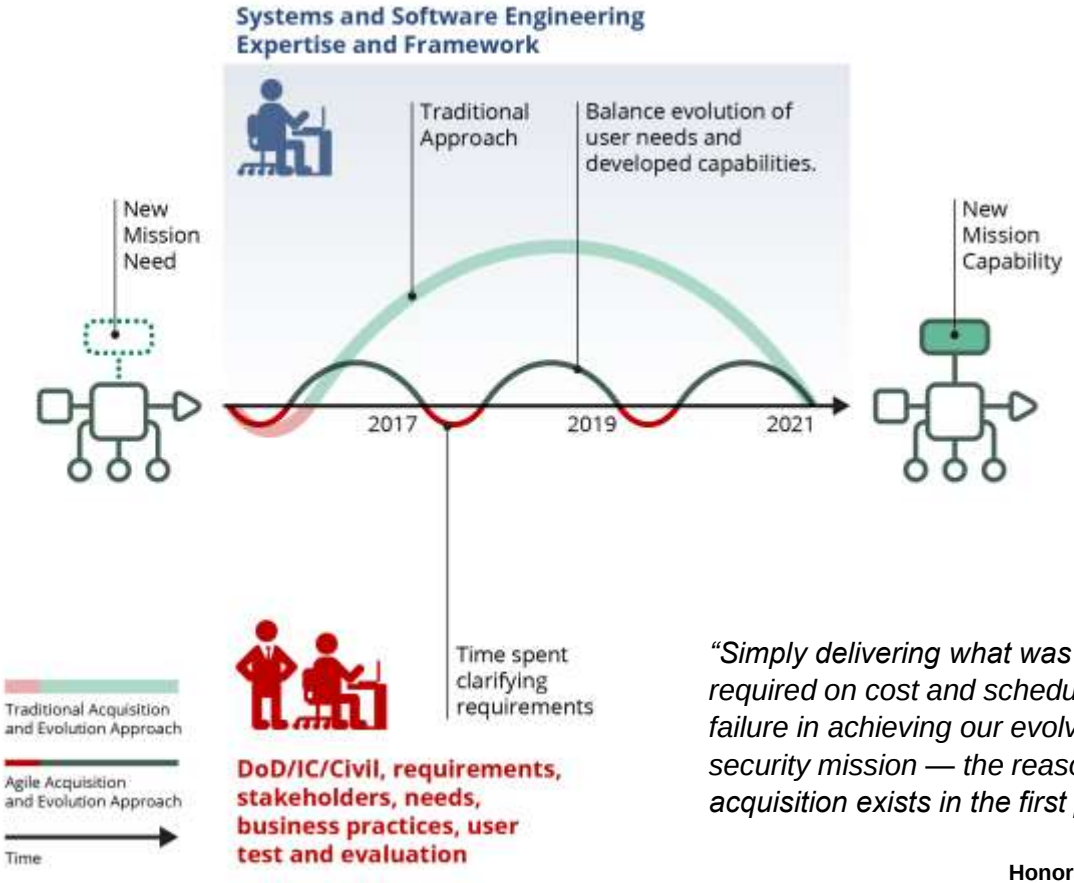
Is your program ready: An Agile Readiness & Fit Exercise

Why does the DoD/Govt care?

Deliver performance at the speed of relevance

Streamline rapid, iterative approaches from development to fielding

National Defense Strategy Summary
Jan 2018



"Simply delivering what was initially required on cost and schedule can lead to failure in achieving our evolving national security mission — the reason defense acquisition exists in the first place."

Honorable Frank Kendall
Under Secretary of Defense (AT&L)
2015 Performance of The Defense Acquisition System

Sample of Reported Results on DoD/Federal Programs

- Quantifiable cost savings and 6-month early delivery
- Significant cost avoidance
- Reduced rework & unplanned releases
- Dramatically increased productivity/capacity, with reduced cost of delivery
- Improved insight into contractor performance and progress
- Early discovery & resolution of Cat 1 defects (one year prior to integration test event)
- Early discovery & resolution of interface issues
- Improved flight test efficiency
- Early insight for end users into functionality of delivered system
- Better responsiveness to users with rapidly fluctuating requirements
- Heightened awareness & collaboration, improved realization of tradeoffs
- Improved workflow management

Large Software Projects Rarely Succeed

Project Size	Successful*
Grand	6%
Large	11%
Medium	12%
Moderate	24%
Small	61%

Source: Standish Group 2015 CHAOS Report

Advantages of small, incremental deliveries

- Fast feedback from stakeholders
- Less investment to move project goals forward
- Less time spent refining low priority items

* **Success:** On Time, On Budget, Satisfactory Result

MITRE

Software Engineering Institute

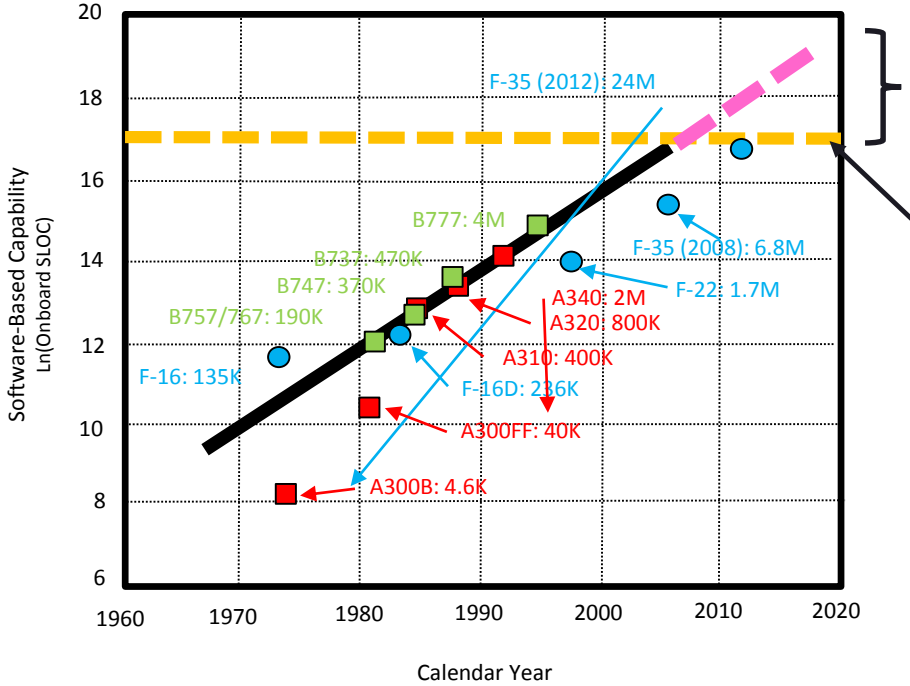
Historical Reasons SW Acquisitions Fail

Top 10 Reasons	Your Perspective
10. Technology used is new to the organization	
9. Software issues are considered too late in the system-development process	
8. Inadequate planning and estimating; long duration programs	
7. Size matters—large projects get into trouble more frequently than smaller ones	
6. Software objectives/requirements are not fully understood or specified; they change frequently (and grow) during the project; growth often uncontrolled/mismanaged	
5. Inadequate project management methodology	
4. Inadequate process emphasis	
3. Inadequate contract incentives to encourage use of modern software engineering practices	
2. Acquirers and developers lack experience working as a team	
1. Insufficient senior staff and/or inexperienced software engineering cadre	

Source: Nielsen, P. Congressional Testimony July 9, 2009.

Provide *your* rank order of these failure modes.

Complex software costs pose a military threat (e.g., in Aviation Software)



We are now in an era where software costs limit military capability

SAVI projects a limit of affordability at 27.5MSLOC or $10B in software costs

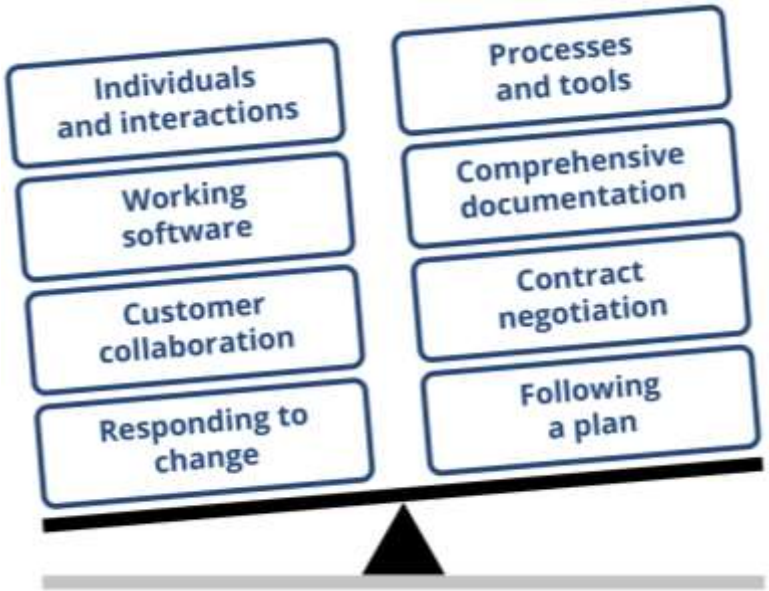
Augustine's Law #16
"In the year 2054, the entire defense budget will purchase just one tactical aircraft. This aircraft will have to be shared by the Air Force and Navy 3½ days each per week except for leap year, when it will be made available to the Marines for the extra day."
Norman Ralph Augustine

Software as percentage of total system cost : 1997: 45% 2010: 70% 2020: 80+%

SLOC: Source Lines of Code (a proxy measure of software complexity/functionality)
SAVI: System Architecture Virtual Integration (incl. members Airbus, Boeing, Embraer, US FAA/NASA, Honeywell, Rockwell Collins, CMU and UTC)

Agile Manifesto

Through this work we have come to value:



That is, while there is value in the items on the right, we value the items on the left more.

Common myth:

The manifesto is often misinterpreted to mean:

no documentation, no process, and no plan!

http://www.agilemanifesto.org/

Agile Principles-1

1. Highest priority is satisfy the customer through early and continuous delivery of software.
2. Welcome changing requirements, even late in development…
3. Deliver working software frequently, from a couple of weeks to a couple of months...
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Provide environment and support they need…
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Agile Principles – 2

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development…a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Adapted from http://agilemanifesto.org/principles.html

Working Definition of Agile

Agile (*adj.*): 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

Air Force Early Agile Adopters

ISPAN		GCSS-AF
AF Weather		C-17 Sustainment
F-22		F-15 Modernization
DCGS-AF		B-2
AFRL-MS177		Blue Devil
PEX		OCX
JMS		UL-C2
MPS		AOC-WS*

Growing adoption of Agile across large and small programs

* Contemplating Agile Adoption

MITRE

Software Engineering Institute

Some Observable Characteristics of Agile Implementations

Iterative—elements are expected to move from skeletal to completely fleshed out over time, not all in one step

Incremental—delivery doesn't occur all at once

Collaborative—progress is expected to be made by stakeholders and the development team working collaboratively throughout the development timeframe

Loosely-coupled Architecture—multiple self-organizing, cross-functional teams work concurrently on multiple product elements (e.g., requirements, architecture, design, and the like) for multiple loosely coupled product components

Dedicated—team members are allowed to focus on the tasks within an iteration/release as opposed to multi-tasking across multiple projects

Time-boxed or Flow-based—relatively short-duration development cycles that permit changes in scope rather than changes in delivery time frame

Traditional vs. Agile Approaches

Traditional approach

- Is consistent with the acquisition lifecycle provided in typical acquisition guidance
- Works well for
  - programs with stable requirements and environment, with known solutions to the requirements
  - programs with a homogeneous set of stakeholders who communicate well via documents
  - programs for which the technology base is evolving slowly (technology is not expected to be refreshed/replaced within the timeframe of the initial development)

Agile approach works well for

- programs with volatile requirements and environment
- programs where solutions are sufficiently unknown that significant experimentation is likely to be needed
- programs for which the technology base is evolving rapidly
- *programs with stakeholders who can engage with developers in ongoing, close collaboration*

Nidiffer, K. Miller, S. & Carney, D. *Potential Use of Agile Methods in Selected DoD Acquisitions: Requirements Development and Management* (CMU/SEI-2013-TN-0006), September 2013.

Important Points to Remember: Agile Basics

Agile is an iterative, incremental, highly collaborative approach that prioritizes responsible responsiveness to changing conditions and as-built product over projections

- There are many valid ways to implement the principles
- A wide variety of popular engineering methodologies fall under the umbrella of “Agile”

Agile approaches require collaboration across the enterprise to be successful

- Contracts, finance, test, end users…

Agile approaches support fast learning cycles and adaptation to changing conditions/volatility

- Changes in technology, threats, priorities and diverse stakeholders, unknown solutions/experimentation
- Traditional highly sequential (“waterfall”) approaches are well-suited to homogeneous, stable environments with slowly changing requirements

Agile Principles were Designed & Focused on Small Teams

We operate on a massive scale – ***how does Agile work "in the large"?***

Some considerations when scaling above a few small teams:

- Managing interfaces among the many products/system components that multiple teams are working on…
- Synchronizing releases and events across multiple teams…
- Organizing inventory (backlog) of requirements productively to support the development pace of multiple small teams….
- Dealing with specialty disciplines (UX, security, etc.) that have significant inputs to the evolving product, but aren't needed as full time team members….
- Mindfully specifying architecture ("just enough") and other far-reaching concerns…
- Incorporating high assurance requirements (safety of flight, IA, nuclear surety…)

Foundations of the Scaled Agile Framework® (SAFe®) 4.5

V4.5.0

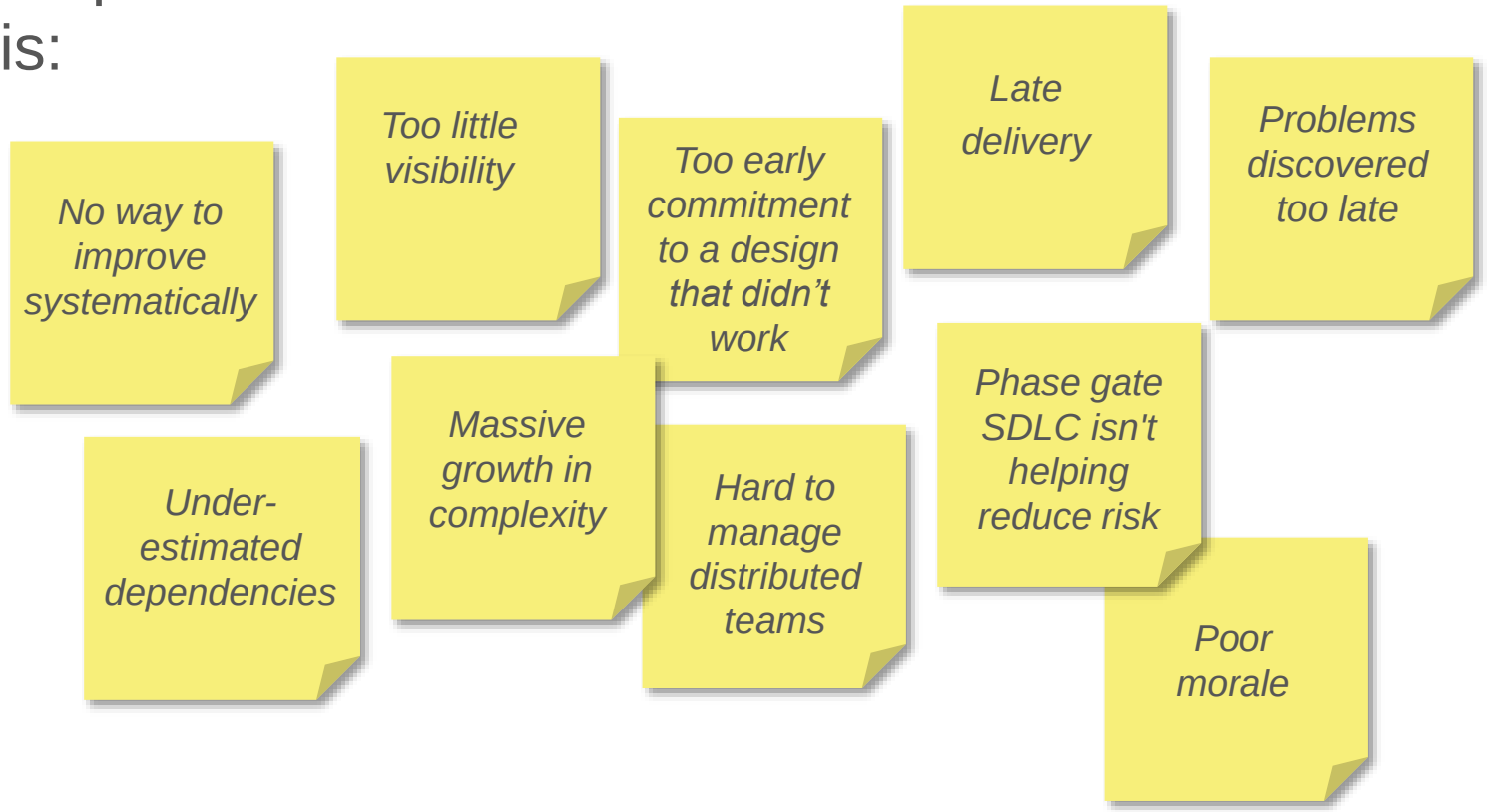
SCALED AGILE®

We thought we'd be developing like this.

But sometimes it feels like this.

Library of Congress

And our retrospectives read like this:



Management's challenge

It is not enough that management commit themselves to quality and productivity. … They must know what it is they must do.

Such a responsibility cannot be delegated.

—W. Edwards Deming

"… and if you can't come, send no one."
—Vignette from *Out of the Crisis*, Deming,1986

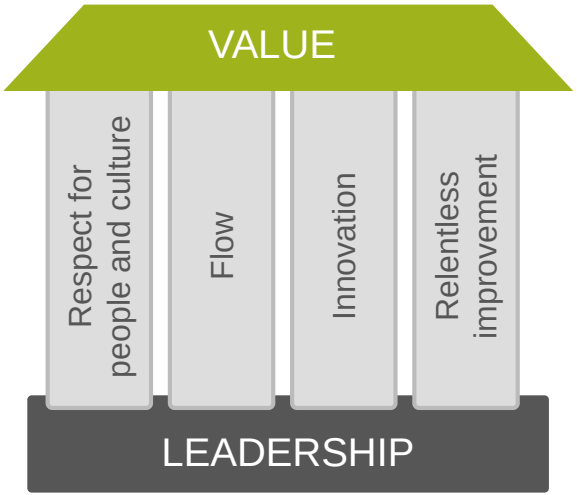
What it is they must do

- Embrace a Lean-Agile mindset
- Implement Lean-Agile practices
- Lead the implementation
- Get results

Embrace a Lean-Agile mindset

Embrace Lean-Agile values

House of Lean



Value in the shortest sustainable lead time

Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

SAFe Lean-Agile principles

#1 - Take an economic view

#2 - Apply systems thinking

#3 - Assume variability; preserve options

#4 - Build incrementally with fast, integrated learning cycles

#5 - Base milestones on objective evaluation of working systems

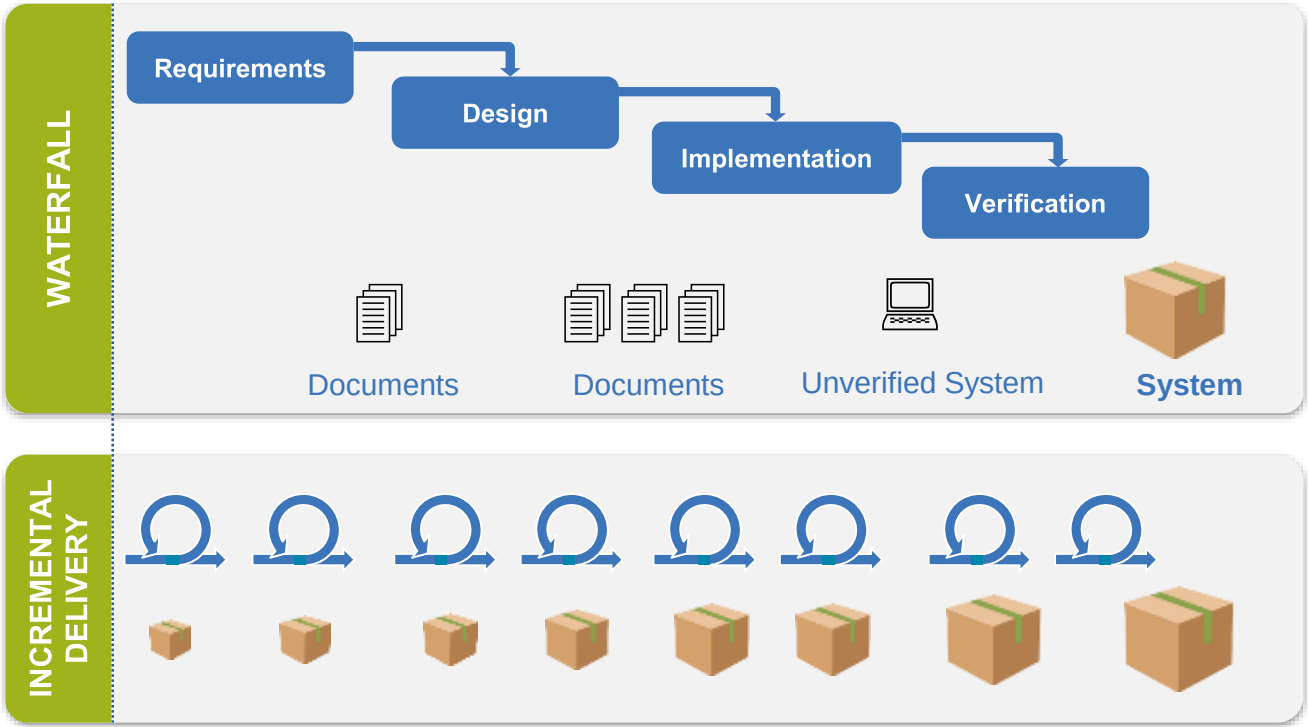
#6 - Visualize and limit WIP, reduce batch sizes, and manage queue lengths

#7 - Apply cadence, synchronize with cross-domain planning

#8 - Unlock the intrinsic motivation of knowledge workers

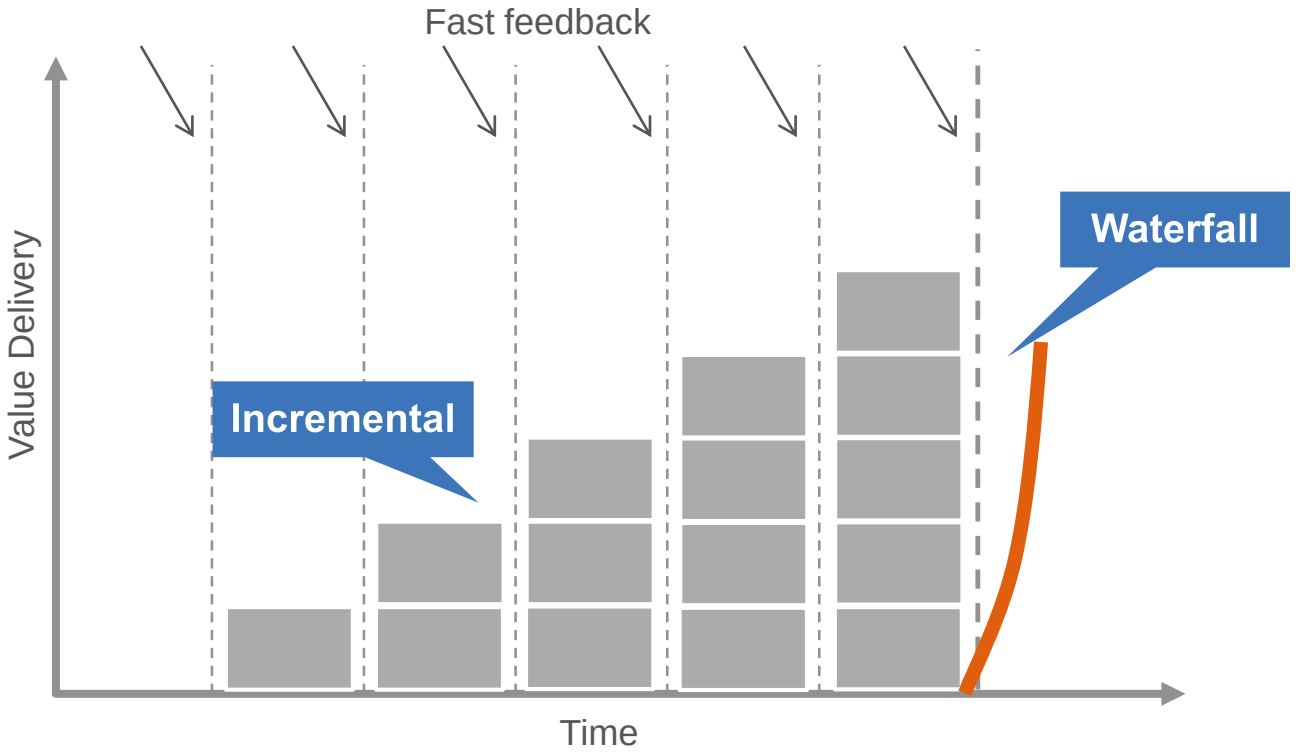
#9 - Decentralize decision-making

Building incrementally accelerates value delivery



And delivers better economics

Early delivery provides fast value with fast feedback



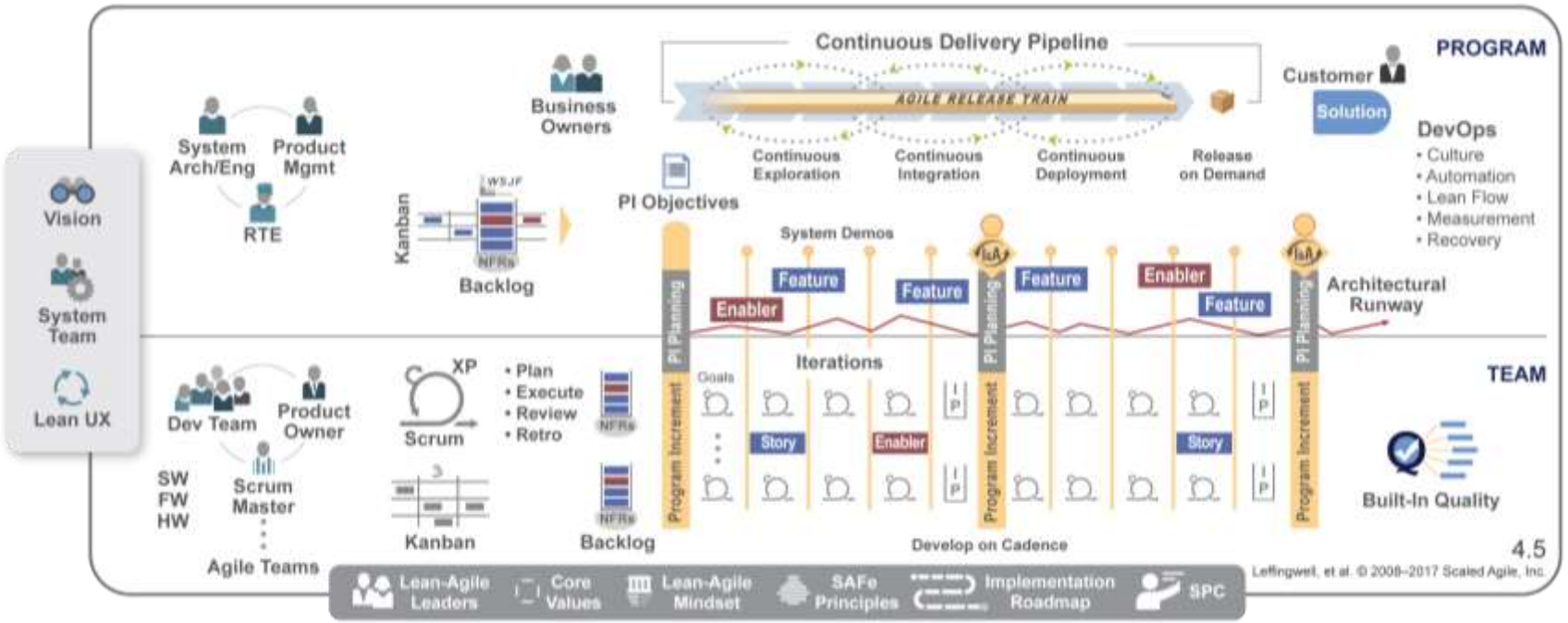
Implement Lean-Agile practices

Knowledge for people building the world's most important systems

SAFe® is a freely revealed knowledge base of integrated, proven patterns for enterprise Lean-Agile development.

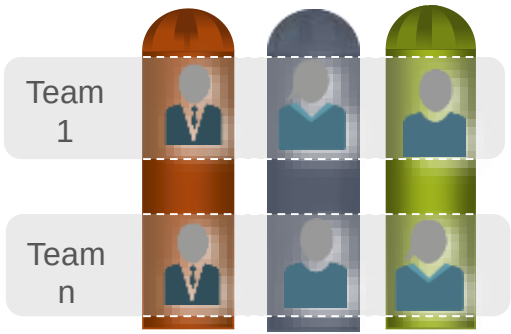
scaledagileframework.com

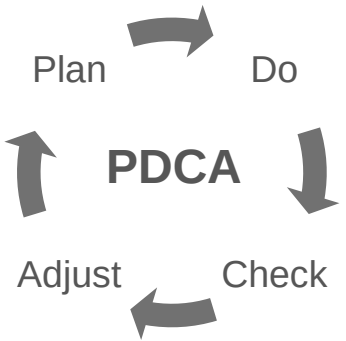
Essential SAFe provides the basis for success



Nothing beats an Agile Team

- Cross-functional, self-organizing entities that can define, build and test a thing of value
- Applies basic scientific practice: Plan—Do—Check—Adjust
- Delivers value every two weeks



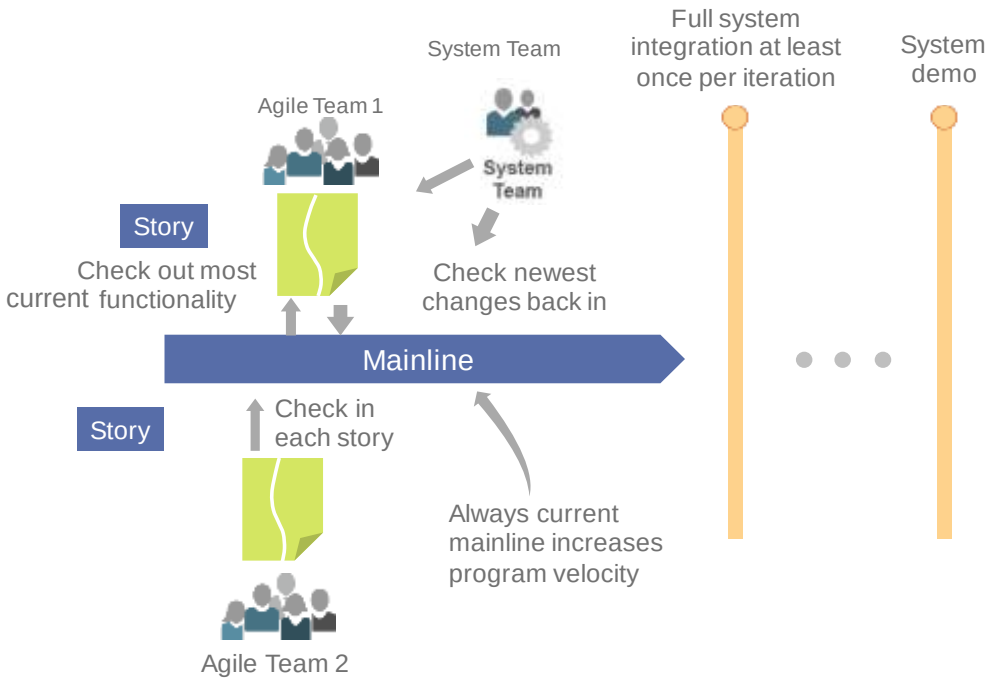


That integrates frequently

Integration points control product development.

— Dantar Oosterwal, The Lean Machine

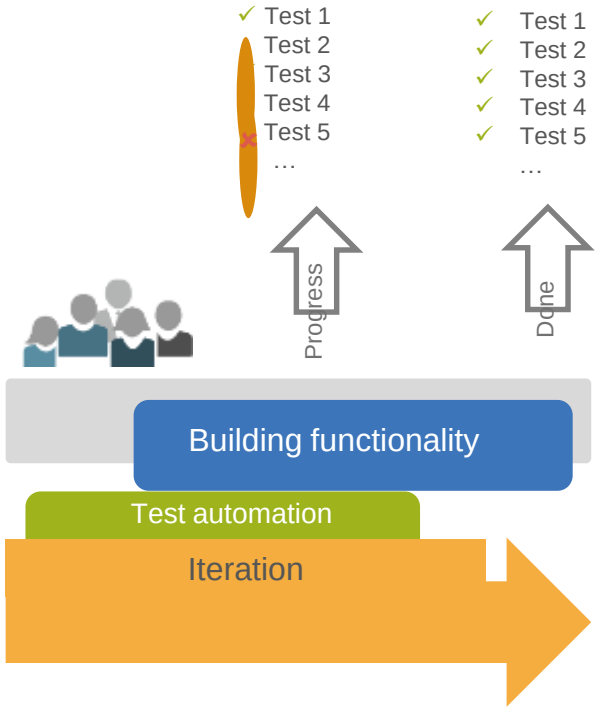
- Avoid physical branching for software
- Frequently integrate hardware branches
- Use development by intention in for inter-team dependencies



Applies test automation

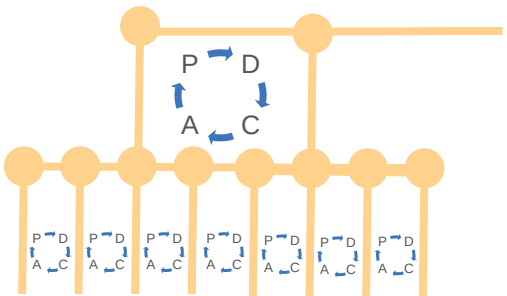
Test automation supports rapid regression testing

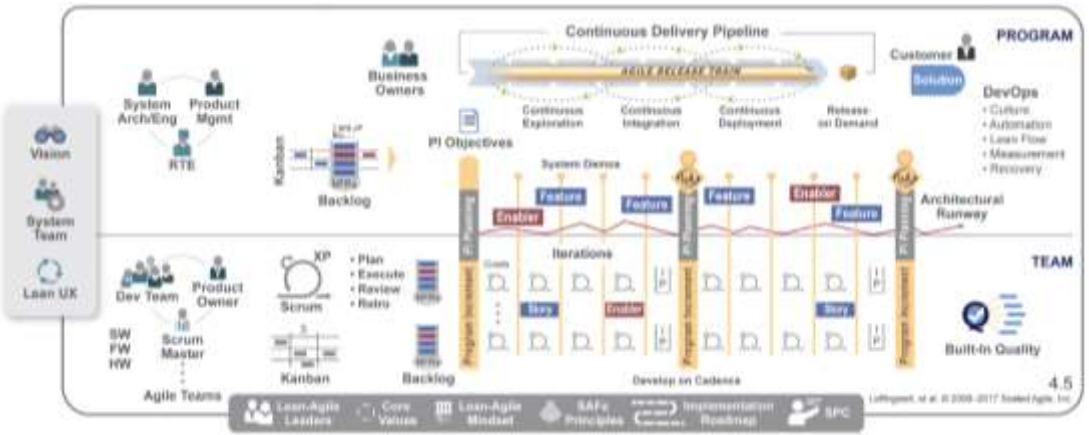
- Implemented in the same iteration
- Maintained under version control
- Passing vs. not-yet-passing and broken automated tests are the *real* iteration progress indicator



Except a team of Agile Teams

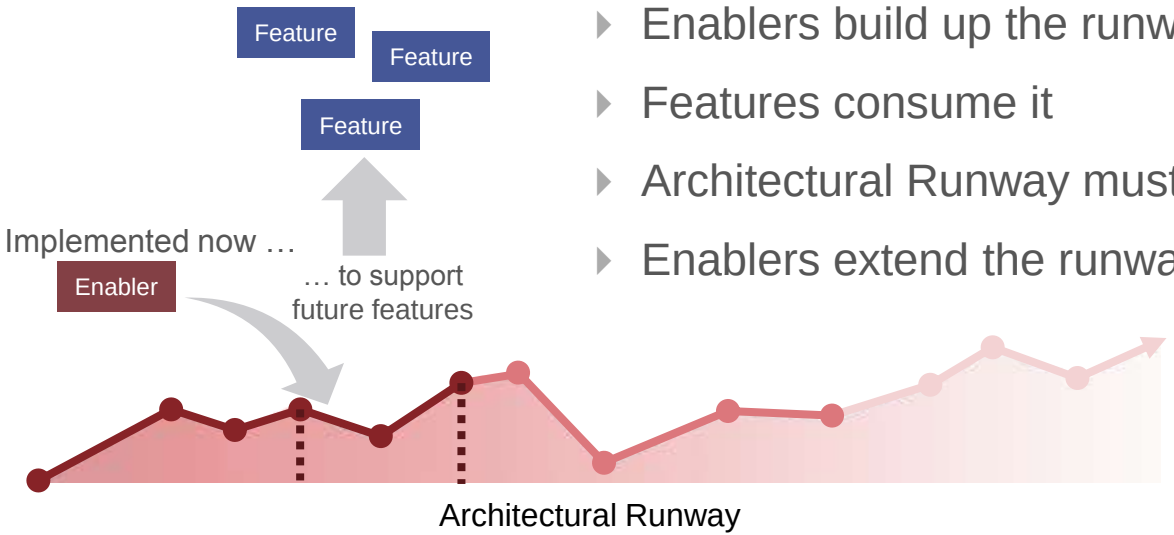
- Align 50-125 practitioners to a common mission
- Apply cadence and synchronization, Program Increments every 6-12 weeks
- Provide Vision, Roadmap, architectural guidance





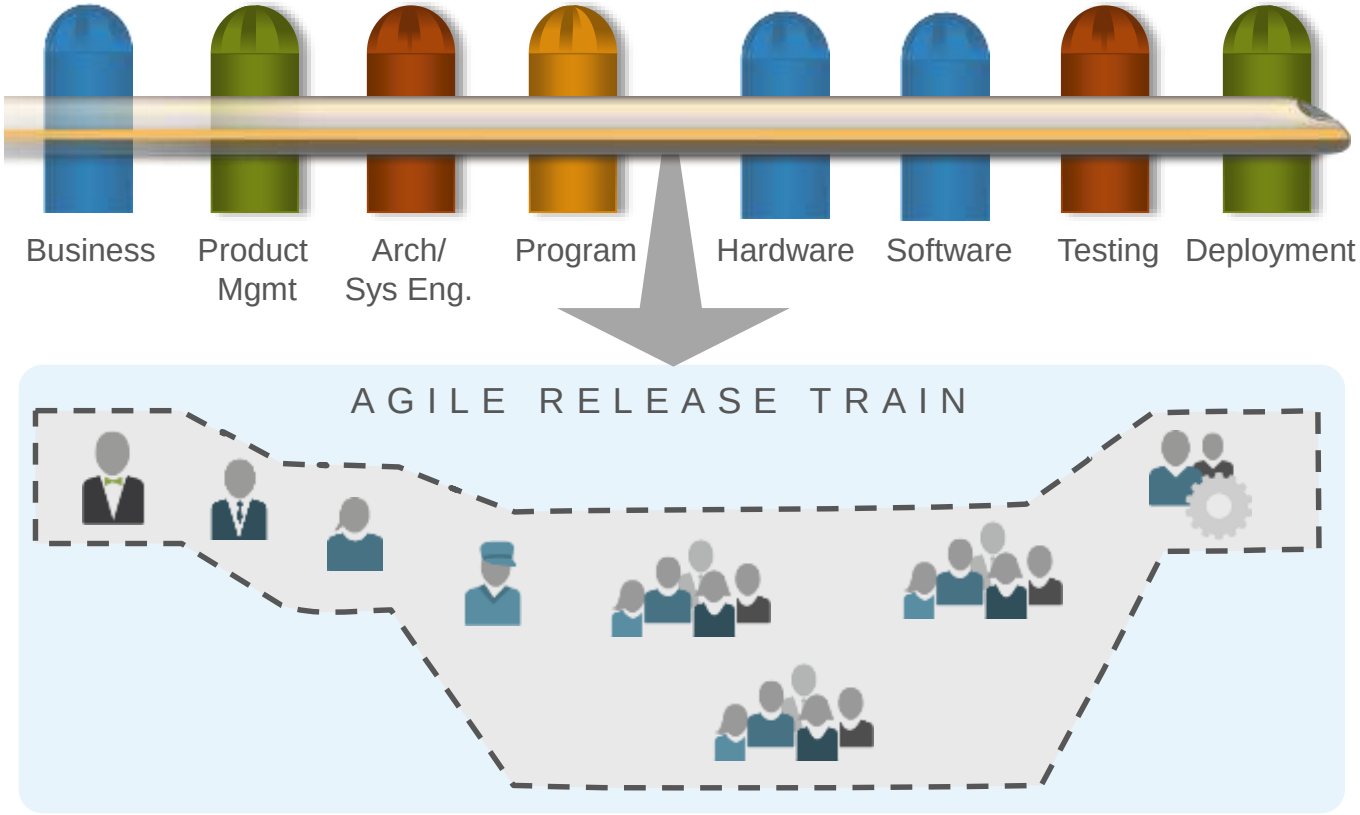
With some Architectural Runway

Architectural Runway—existing code, hardware components, etc. that technically enable near-term business features



- Enablers build up the runway
- Features consume it
- Architectural Runway must be continuously maintained
- Enablers extend the runway

Bringing together the necessary people

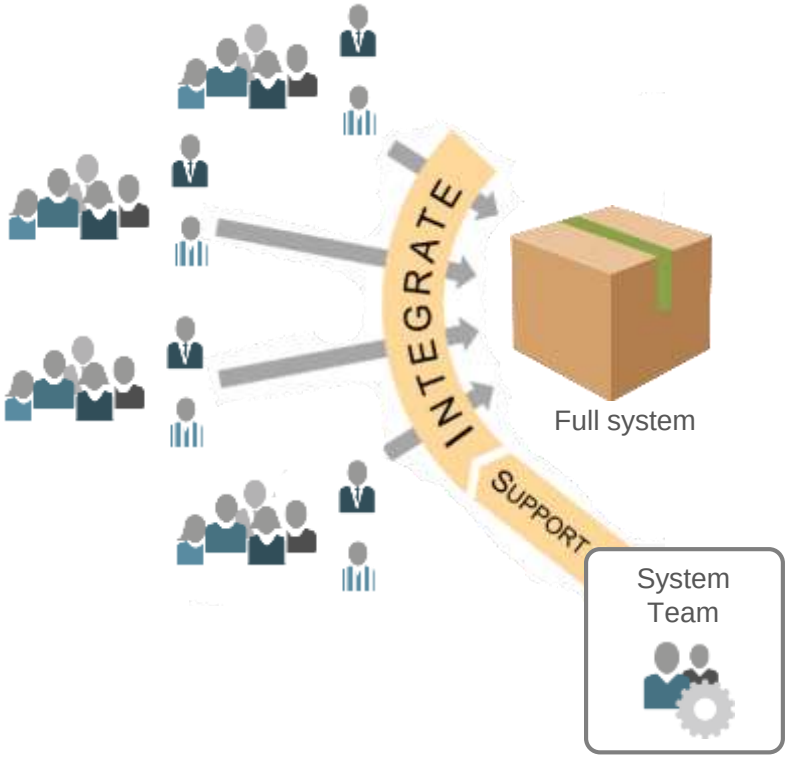


Synchronizes with PI Planning

Future product development tasks can't be pre-determined. Distribute planning and control to those who can understand and react to the end results. — Michael Kennedy, Product Development for the Lean Enterprise

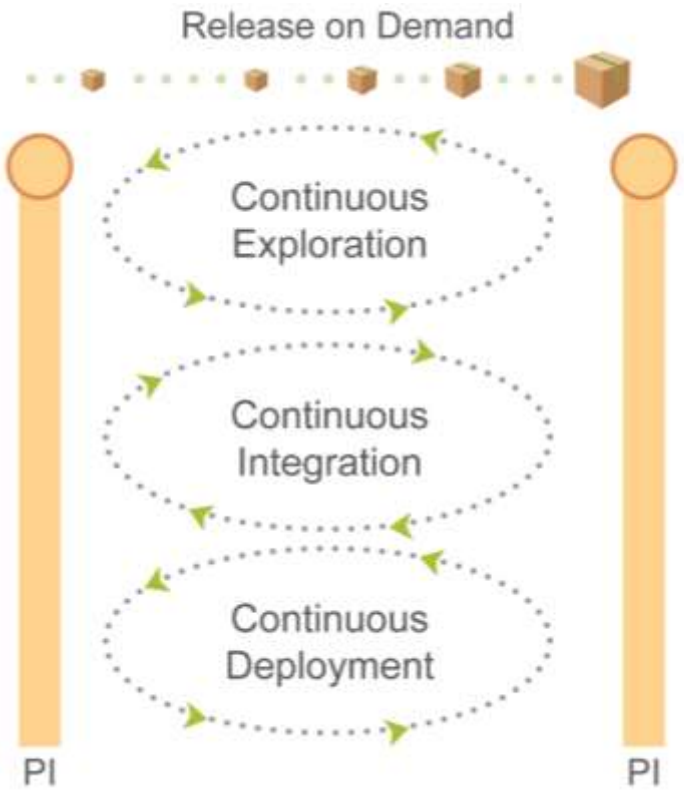
- All stakeholders face-to-face (but typically multiple locations)
- Management sets the mission, with minimum possible constraints
- Requirements and design emerge
- Important stakeholder decisions are accelerated
- Teams create—and take responsibility for—plans

Demonstrates the full system every two weeks



- An integrated solution demo
- Objective milestone
- Demo from the staging environment, or the nearest proxy

Continuously delivers value to customers with DevOps



Inspects and Adapts every PI

Every PI, teams systematically address the larger impediments that are limiting velocity.

Agree on the problem to solve

Insufficiently reliable release commitments?

Apply root cause analysis (+ five whys)

Identify the biggest root cause using Pareto Analysis

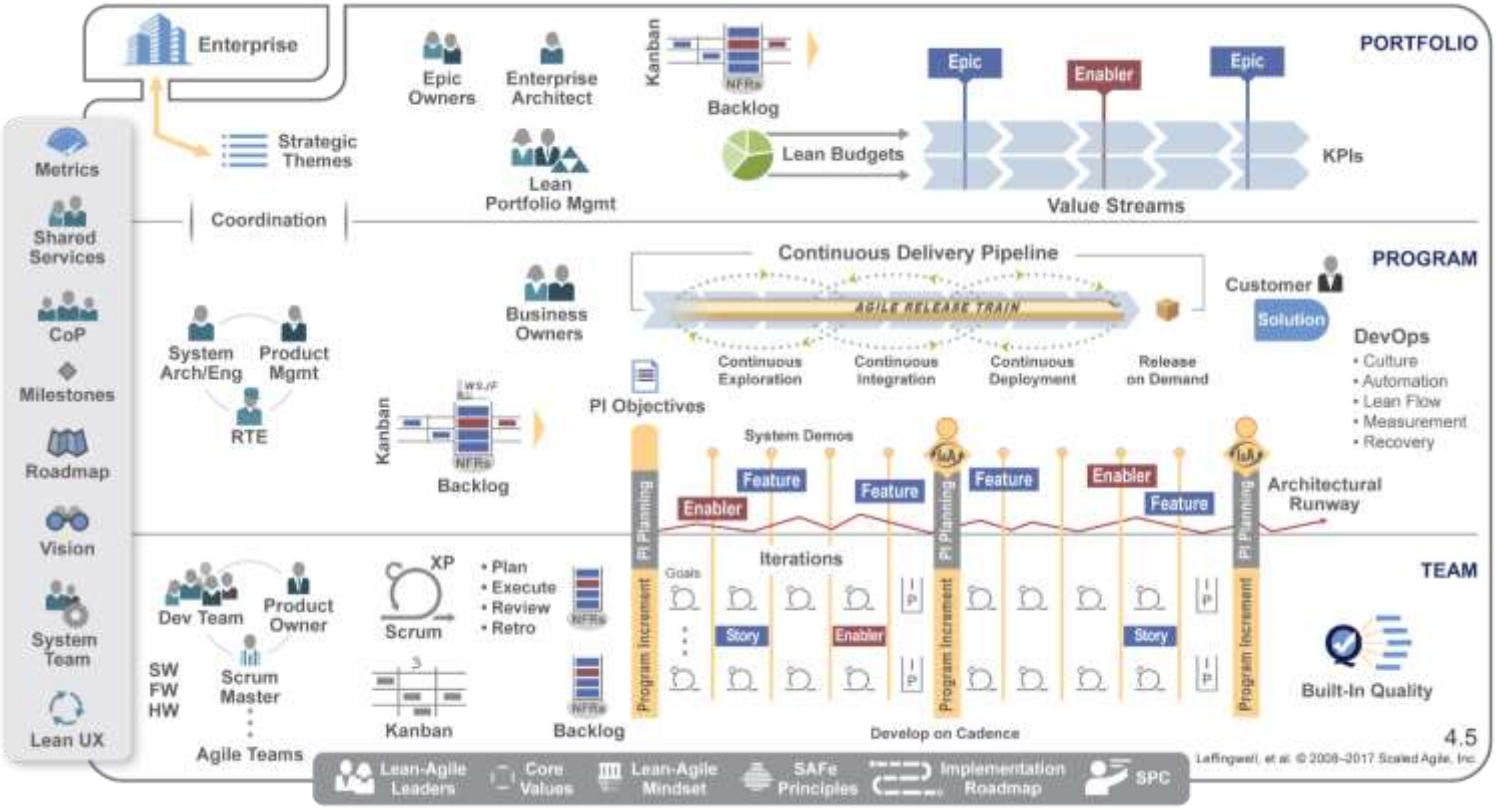
Restate the new problem for the biggest root cause

Insufficient architectural runway

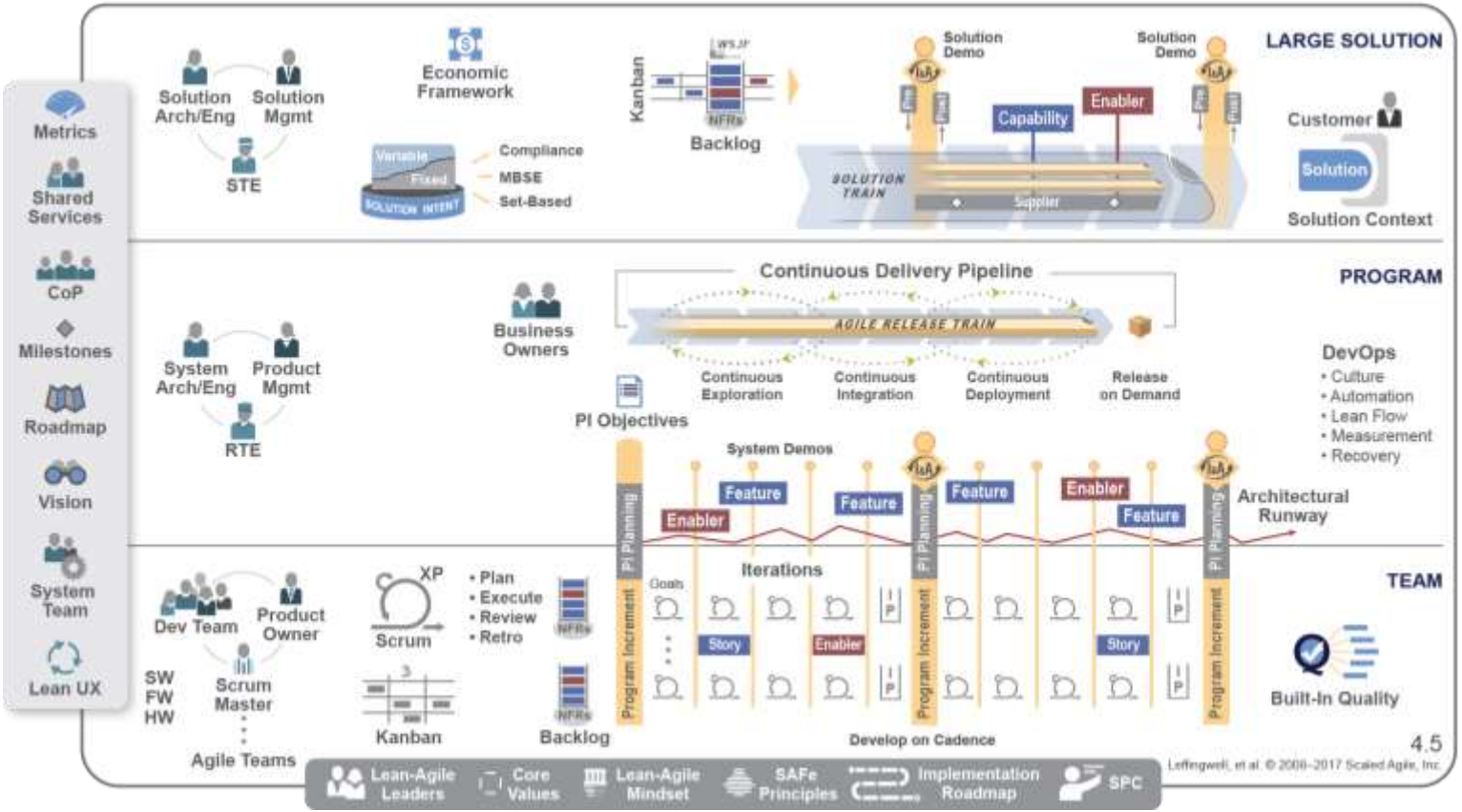
Brainstorm solutions

Identify improvement Backlog items

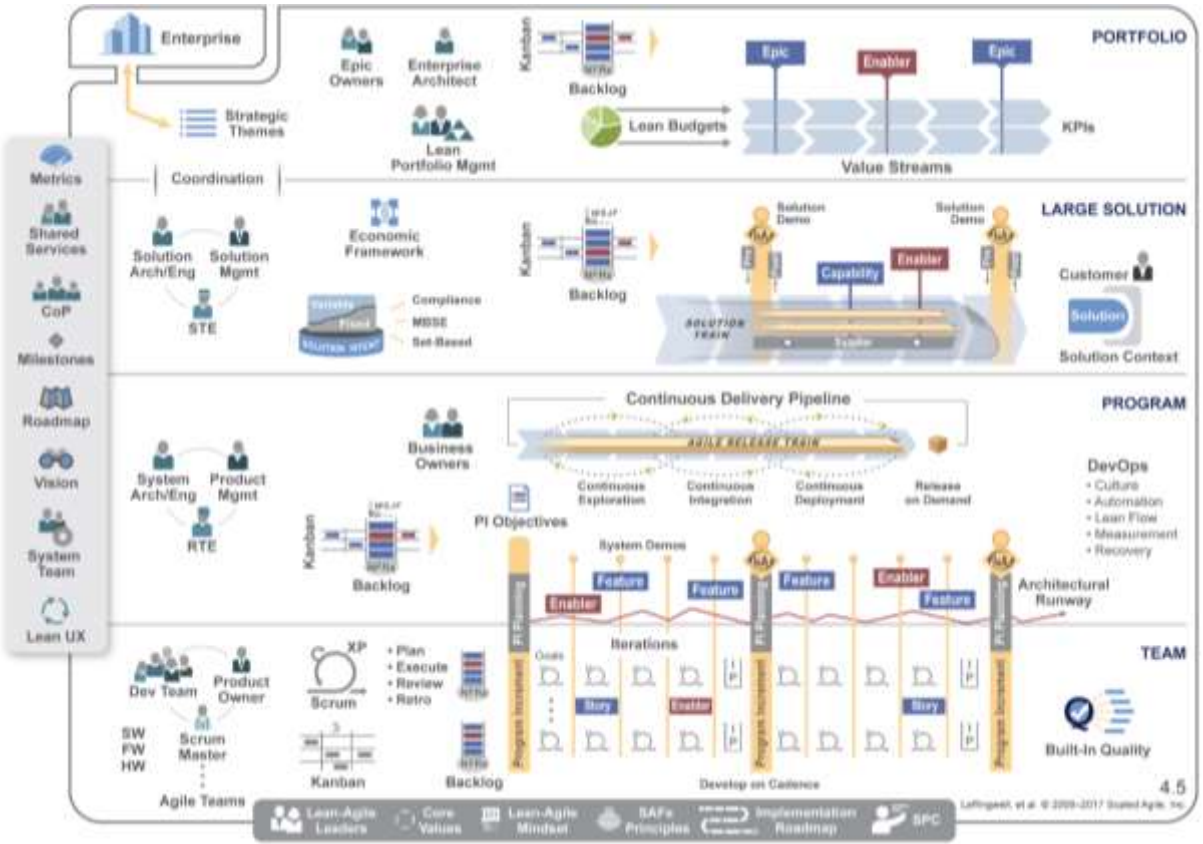
Portfolio SAFe aligns strategy and execution



Large Solution SAFe coordinates ARTs with a Solution Train

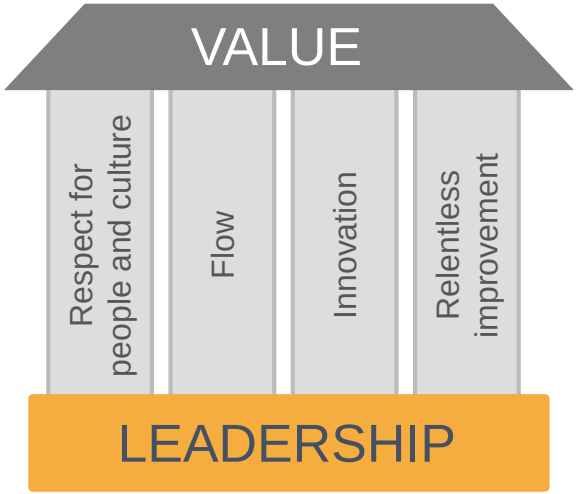


Full SAFe for large enterprises



Lead the implementation

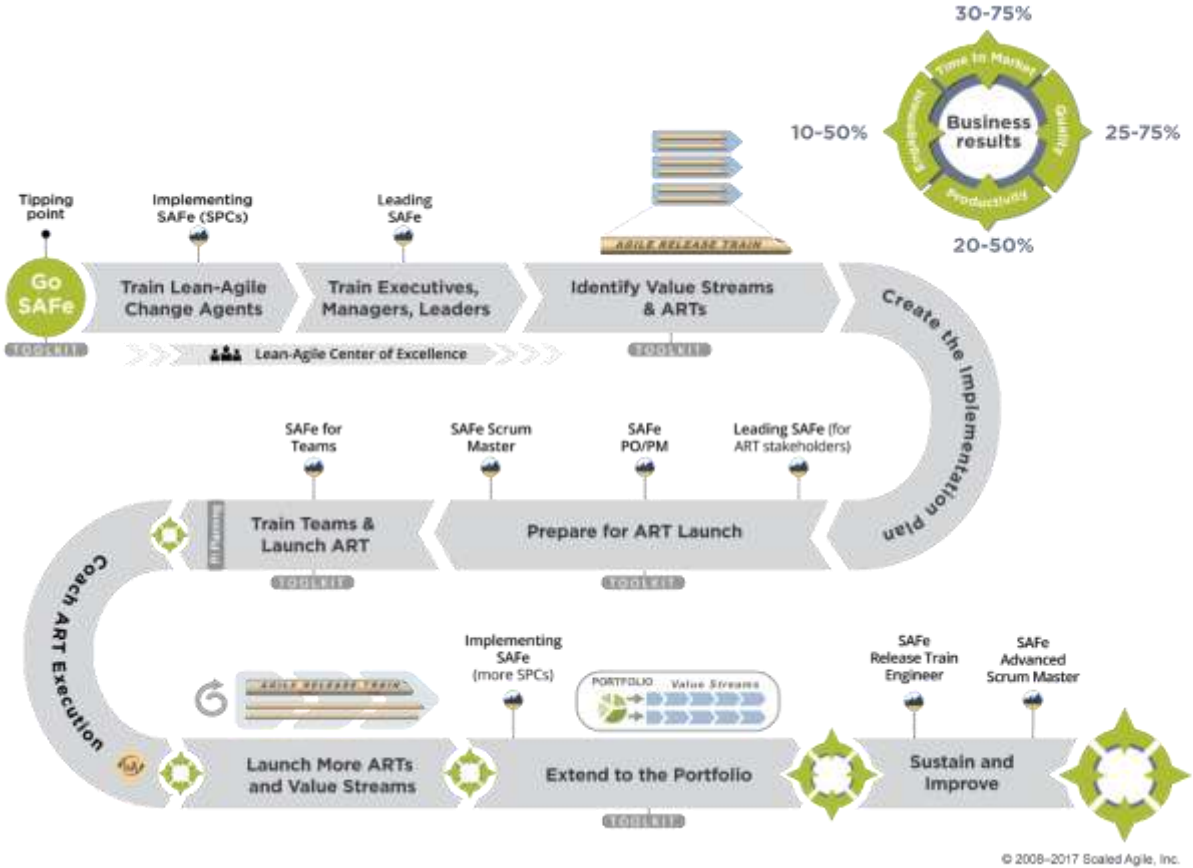
Leadership foundation



People are already doing their best; the problems are with the system. Only management can change the system.

—W. Edwards Deming

Implementation Roadmap



Get results

Business results

10 – 50% happier, more motivated employees

30 – 75% faster time-to-market

20 – 50% increase in productivity

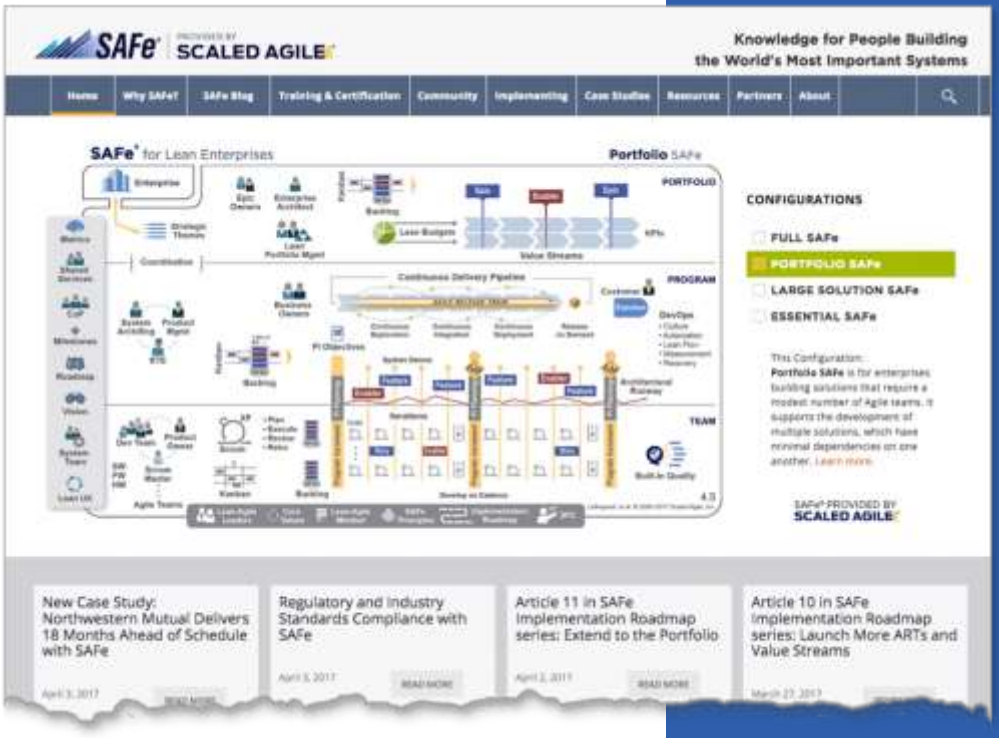
25 – 75% defect reduction

See ScaledAgileFramework.com/**case-studies**

Financial Services / Electronics / Software / Telecom / Retail & Distribution / Government / Healthcare / Insurance / Medical Technology / Pharmaceutical / Media / Manufacturing / COTS Software / Customer Care & Billing / Outsourcing

See ScaledAgileFramework.com/**case-studies**

Gain the Knowledge



ScaledAgileFramework.com

Explore the SAFe knowledge base and find free resources:

- Articles
- Guidance
- Presentations
- White papers
- Videos
- Case studies

ScaledAgile.com

Find SAFe
training worldwide

CORE

- Leading SAFe
- SAFe for Teams
- SAFe Scrum Master
- SAFe PO/PM

ADVANCED

- SAFe Advanced Scrum Master
- Implementing SAFe
- SAFe Release Train Engineer

SAFe® for Lean Enterprises

180,000
SAFe certified professionals in 100+ countries

150
Scaled Agile Partners in 50 countries

70% US *Fortune* 100 enterprises have SAFe certified professionals

2 million
Annual visitors to SAFe and Scaled Agile websites

Pledged 1%
Scaled Agile stock equity & employee time to Pledge 1% campaign

SAFe:
Freely available knowledge base, downloads, and resources for people building the world's most important software and systems

SAFe SUMMIT

Freely Available
SAFe's knowledge base is freely available at **scaledagileframework.com**

Configurable
SAFe is able to accommodate enterprises of all sizes and industries

Fastest Growing Method
- **11th Annual State of Agile Report by VersionOne**
- **2017 Scaling Agile Report by cPrime**

SAFe cited as preferred solution for scaling Agile, making SAFe the most popular scaling method above Scrum, Scrum of Scrums, and all other frameworks

SAFe® | PROVIDED BY SCALED AGILE

SAFe Lean-Agile principles

#1 - Take an economic view

#2 - Apply systems thinking

#3 - Assume variability; preserve options

#4 - Build incrementally with fast, integrated learning cycles

#5 - Base milestones on objective evaluation of working systems

#6 - Visualize and limit WIP, reduce batch sizes, and manage queue lengths

#7 - Apply cadence, synchronize with cross-domain planning

#8 - Unlock the intrinsic motivation of knowledge workers

#9 - Decentralize decision-making

Why “Visualize and Limit WIP, Reduce Batch Sizes, and Manage Queue Lengths”?

To Optimize Flow and Reduce Time to Value

Little's Law – Average Longer Queues Lead to Average Longer Wait Time

Average wait time = average queue length / average processing rate

WQ = LQ / (#of items / time period) OR

WQ = LQ * (time period / # of items)

If the average time to create a latte is 2 minutes and the average queue of people waiting for a lattes is 8 deep, the average wait time for a latte is 16 minutes

Little's Law and Batch Size

Little's Law argues for reduced batch size as a way to speed up value delivery

BUT ..

Holding cost and transaction cost strongly influence the economies of batch size

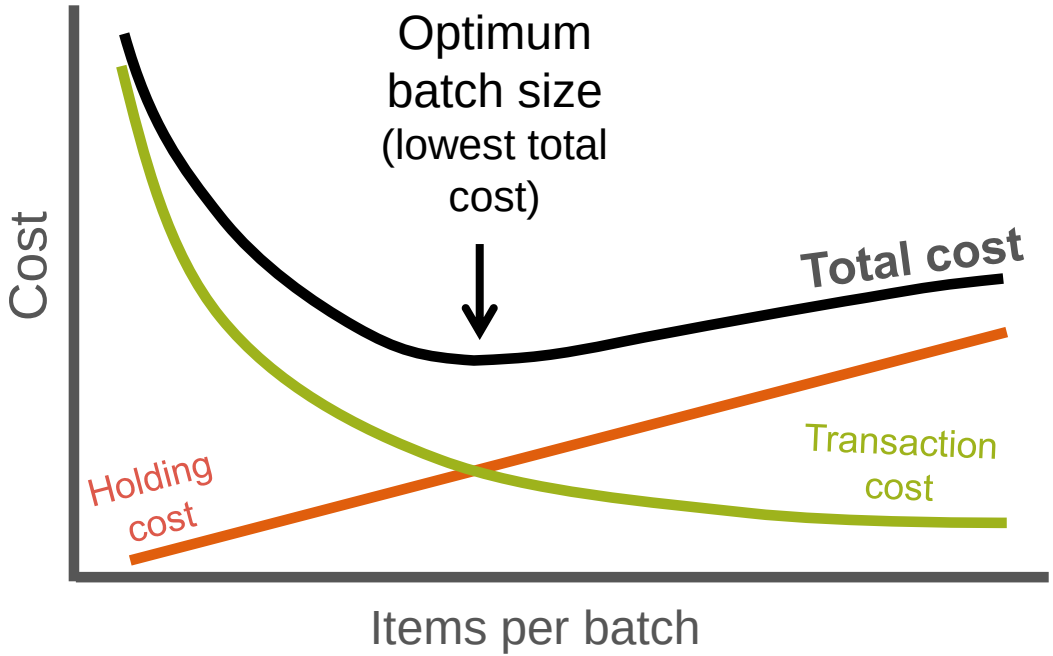
Transaction cost – The cost of sending a batch to the next process

Holding cost – The cost of not sending it

Principles of Product Development Flow, Don Reinertsen

Finding optimum batch size

Optimum batch size is an example of a U-curve optimization.

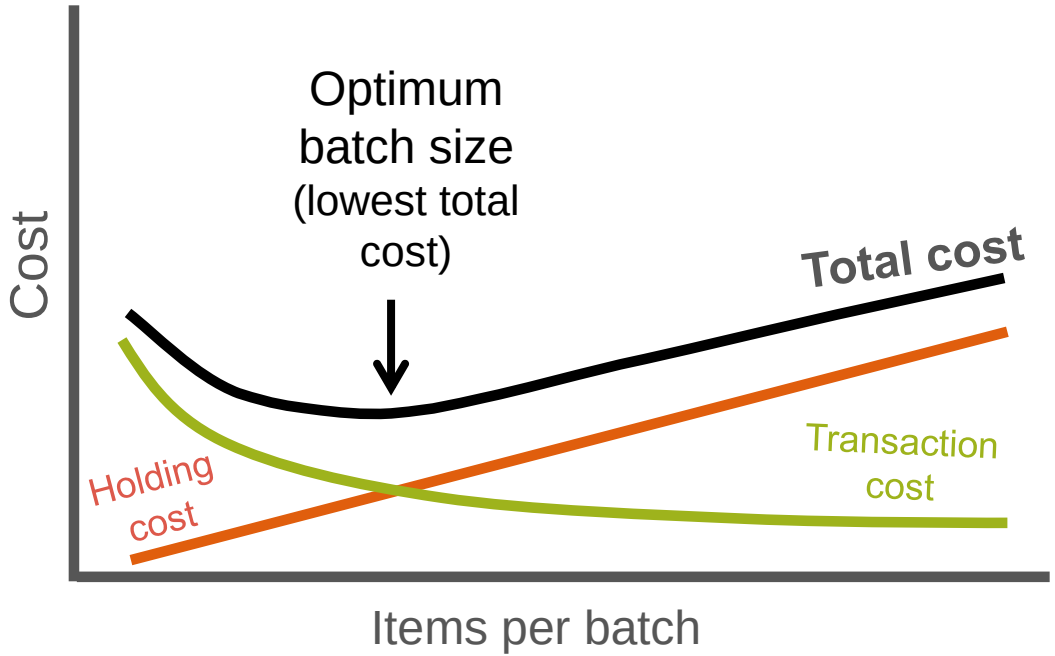


Principles of Product Development Flow, Don Reinertsen

- *Total* costs are the sum of holding costs and transaction costs
- Higher transaction costs shift optimum batch size higher
- Higher holding costs shift batch size lower

Reducing optimum batch size

Reducing transaction costs reduces total costs, and shifts optimum batch size lower.



Principles of Product Development Flow, Don Reinertsen

- Reducing batch size:
  - Increases predictability
  - Accelerates feedback
  - Reduces rework
  - Lowers cost
- Batch size reduction probably saves twice what you think

Reducing transaction costs example
https://youtu.be/RRy_73ivcms
2:09

An Example Would Come in Handy Right About Now

The Realm of the Personal

- Laundry
- Amazon

The Realm of Development

- Testing
- Deployment

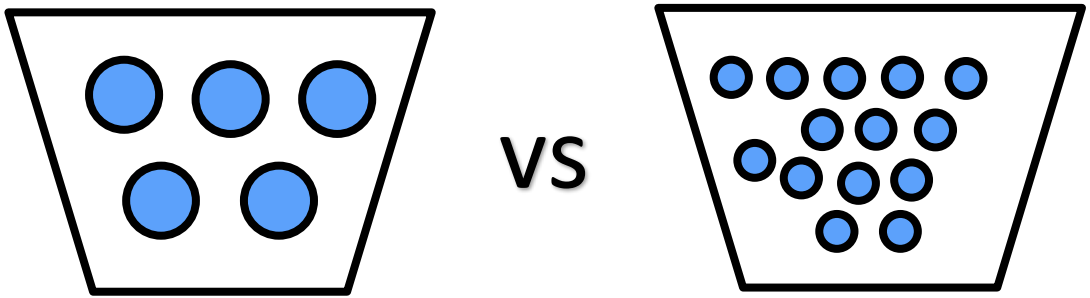
The Realm of Manufacturing

- Toyota and Single Minute Exchange of Die (SMED)

Key Take-Away

Reduced transaction costs are a major enabler of smaller batch size and can dramatically alter the economies of how we work

Story Splitting is an Enabler of Smaller Batch Size Too



Splitting stories requires engineering judgment

Besides Longer Cycle Time, Queues are Just Generally Bad

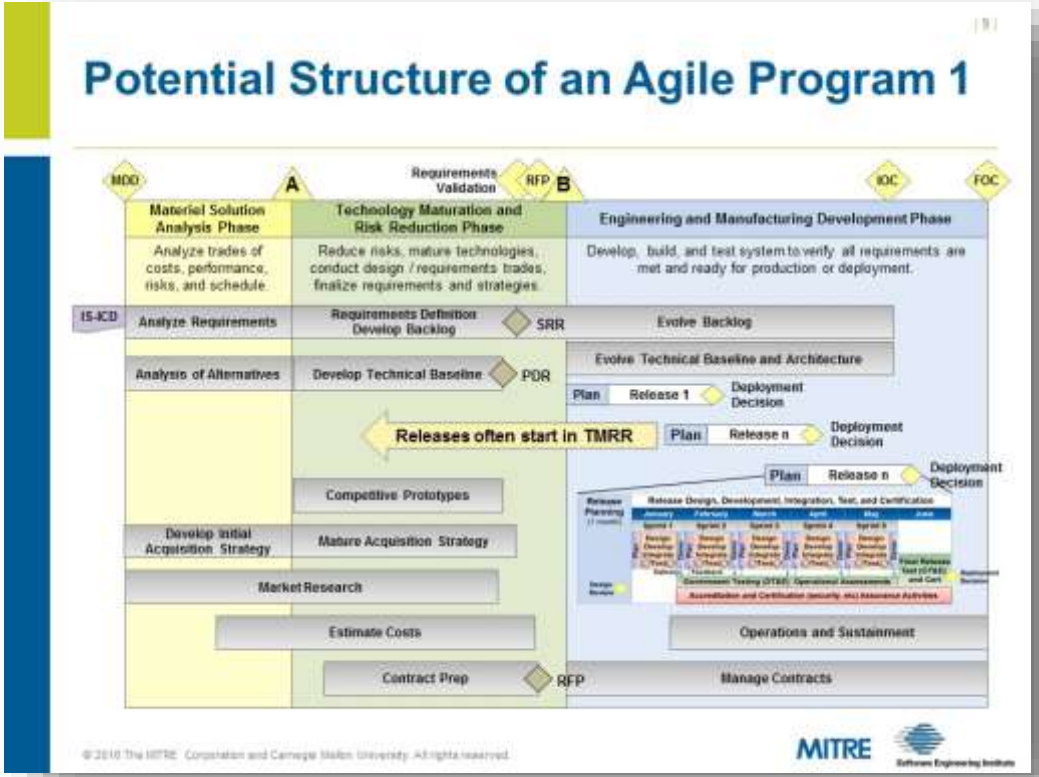
The Principle of Queueing Waste: Queues are the root cause of the majority of economic waste in product development.

Queues create:

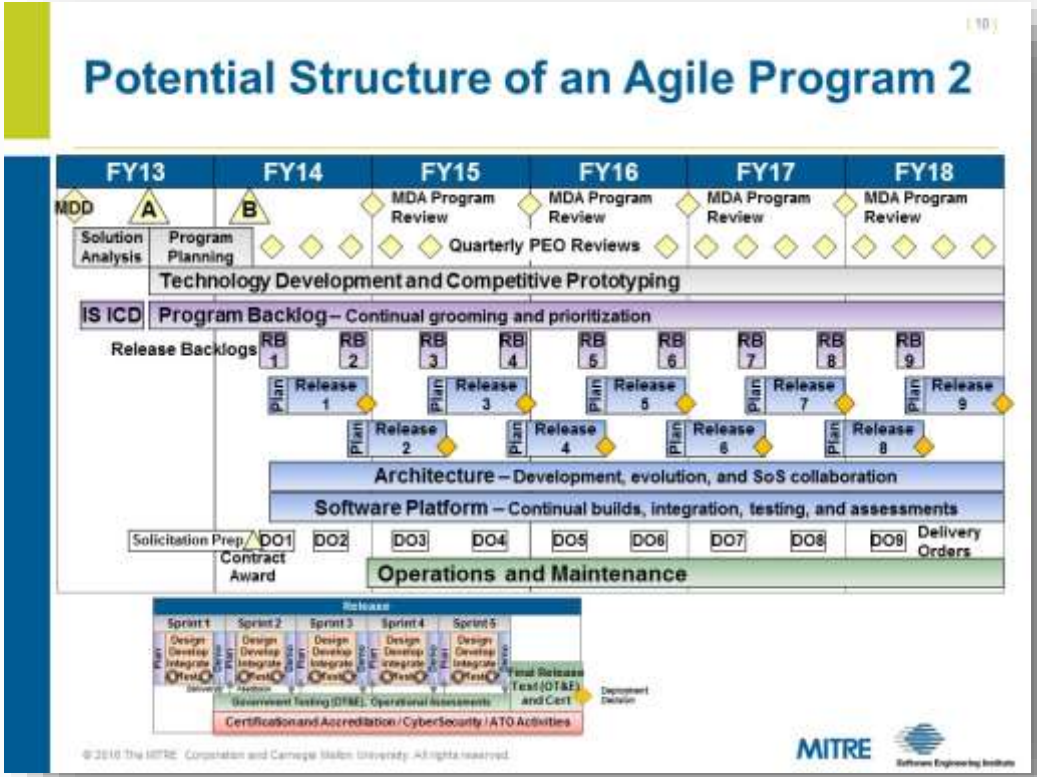
- Longer Cycle Time
- Increased Risk
- More Variability
- More Overhead
- Lower Quality
- Less Motivation

Principles of Product Development Flow,
Don Reinertsen

How SAFe Might Translate into a DoD Acquisition Environment



A More Detailed Look at a Possible Agile Implementation in DoD



How do we think & talk about requirements?

SAFe Requirements Hierarchy

Typical hierarchy (from SAFe, in this case):

- Epic – could be analog to contract-level requirements
- Capability – could be analog to System Level requirements
- Feature– could be analog to software capability requirements
- Story – could be analog to software component level requirements or below

One of the decisions to make is how different levels of requirements will be treated

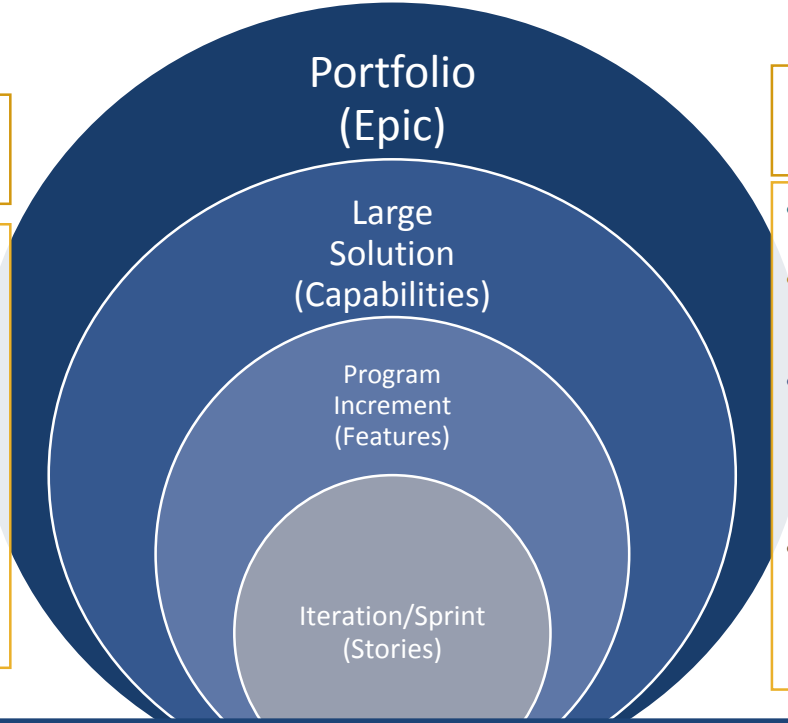
- One dependency is how the software part of the program interacts with systems engineering/other stakeholders

- Another criterion is how requirements change will be accommodated
  - Level at which allocated baseline is established is crucial to having appropriate flexibility for requirements evolution

Addressing Requirements at Multiple Levels (SAFe Terminology)

Issues in Expressing Requirements

- **Portfolio**: Conops level, trying to establish Business/Enabling **Epics**
- **Program/Large Solution**: moving from "shall" statements to **Capabilities**
- **Release**: Decomposing Capabilities into meaningful **Features** that are executable in a few iterations; translating Features into User & Enabling **Stories** that can be allocated to iterations (sprints)
- **Iteration**: "slicing" **Stories** in such a way that meaningful working software can be produced in short (2-3 week) iterations



Issues in Governing Requirements

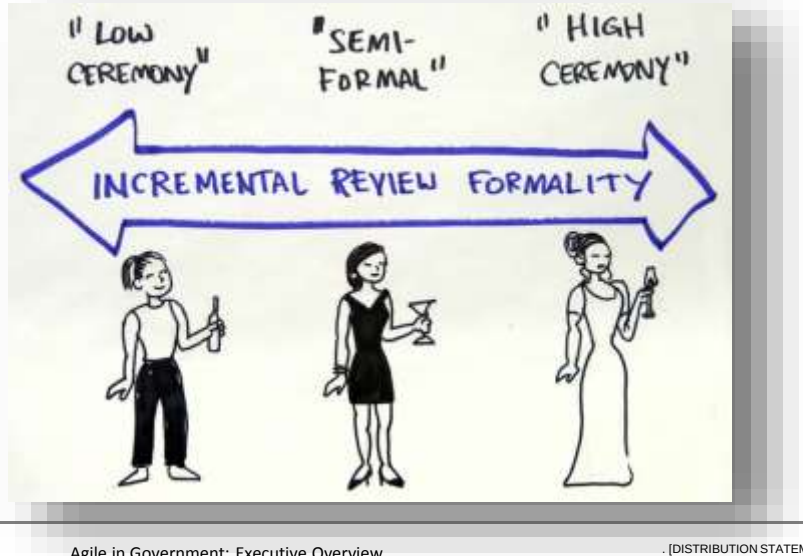
- **Portfolio**: Assuring that the value stream is representative of operations
- **Large Solution**: assuring that acquisition and users or their representatives are engaged and relevant
- **Release**: Assuring that Product Managers (or Chief Product Owners) are actively engaged in refining and prioritizing stories and features ahead of the development teams
- **Iteration**: Assuring that Product Owners appropriately represent user needs and management goals when interacting with development teams

Where should acquisition program offices be *controlling* and/or *participating?*

One of Top Questions SEI Hears about Agile

How do I accommodate Technical Reviews like PDR (preliminary design review), CDR (critical design review), etc.?

- Especially if contract was formulated as traditional and program office or developer wants to use Agile after the fact



S3 Patterns in Agile Settings for PDR, CDR Design/Execution

Pattern A

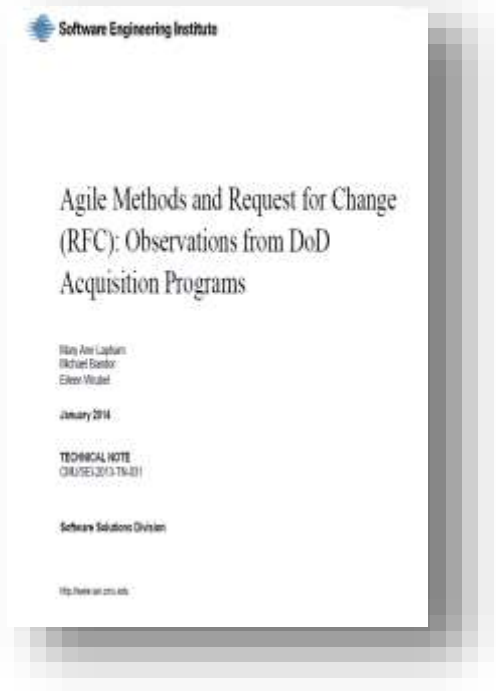
- PMO uses traditional PDR and CDR in each block as traditional milestone events

Pattern B

- PMO team participates in each of multiple Preliminary and Critical Design Working meetings (PPDW/PCDW)* – one per iteration
- PDR and CDR are still held at some level of technical discussion and also include management elements

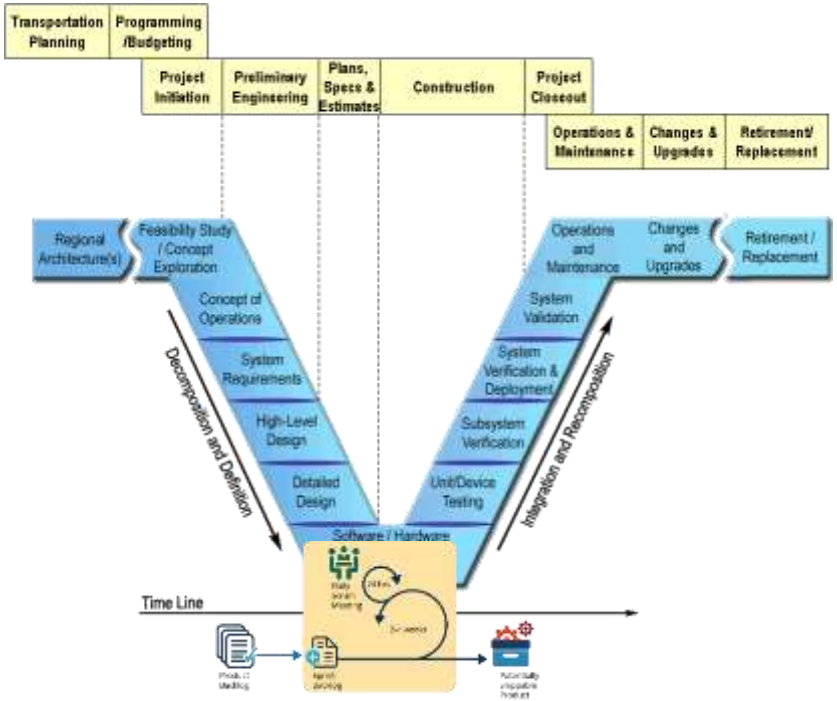
Pattern C

- PMO technical staff (engineers) participate in each PPDW/PCDW (per iteration)
- PDR and CDR become management level reviews
- No technical detail is discussed in PDR and CDR other than a summary for management



*PPDW=Partial Preliminary Design Walkthrough;
PCDW=Partial Critical Design Walkthrough

The Classic Engineering “V Model”



Source: Palmquist, Steve, et al. Parallel Worlds:

This isn’t enough

Optimizing one part of the process:

- Doesn’t optimize the whole process
- Simply exposes roadblocks by other parts of the process

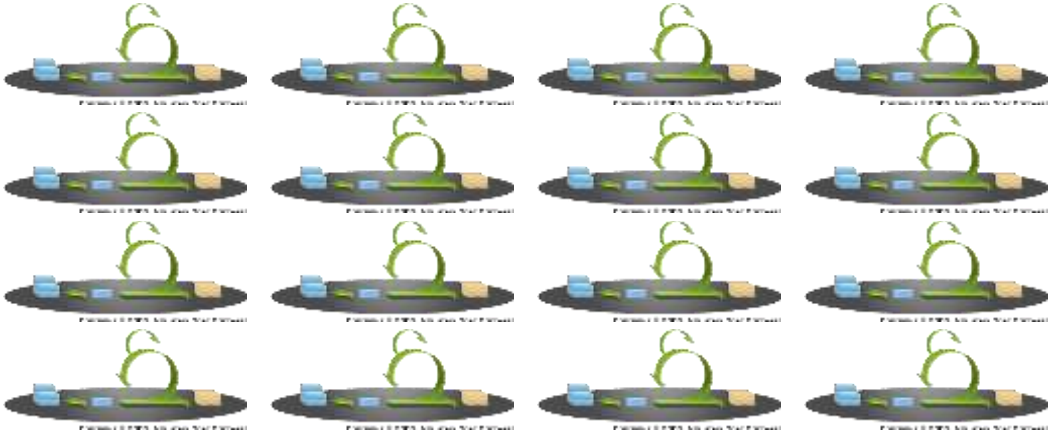
“Agile at the bottom of the V”
loses benefits of agility:

- Too many decisions are made too early
- No learning opportunities

Program Level vs. Team Level Measures

Geared to External Stakeholders

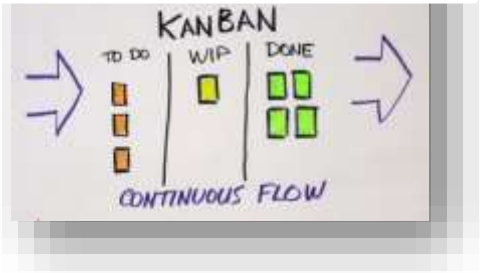
Intended to Serve Needs of the Team Typically Not Shared Out-side the Team

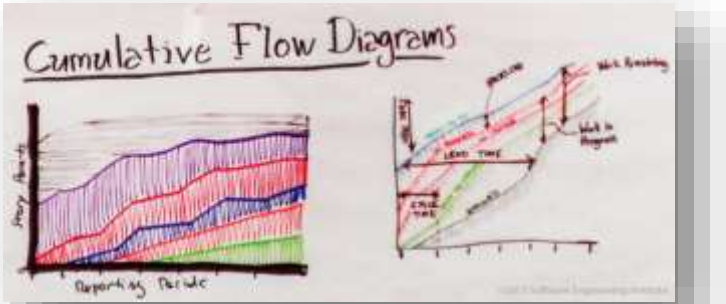


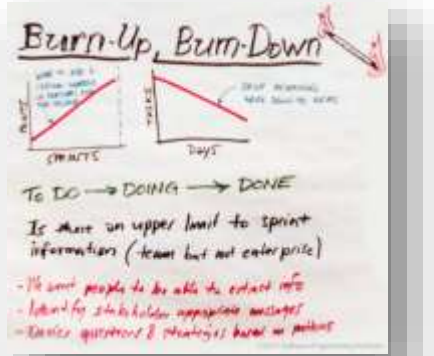
Typical Team Measures for Agile Development

Metrics used by and for the development team

- Kanban Board for Task Tracking
- Sprint Burn-Down Charts
- Release Burn-Up Charts
- Velocity Tracking
- Cumulative Flow Diagrams







Program Level Measures

Because teams focus on delivering working code:

- The program can measure finished product (size, complexity, quality…)
  - Rather than estimates of the finished product being carried (and revised) across the program timeline, we can know *actual* values for incrementally completed work

- The program can focus on 'concept-to-capability' cycle
  - Hidden tradeoffs can compromise design time, or squeeze testing schedules in a waterfall lifecycle – because they are not necessarily visible until later.
  - Cycle time measures in agile lifecycles can show the entire value stream within each incremental delivery.

- Overall capacity can be understood earlier
  - Rather than measuring the productivity of individual disciplines, overall program capacity to achieve the desired schedule can be estimated

Categories of oversight metrics: Ask new questions

Category	Description
Flow	Flow measures come out of the lean engineering and management environment. They focus on understanding the "idea to realization" cycle time. Flow measures for senior oversight focus on the development organization's ability to consistently meet timelines for deployment of IT functions according to a roadmap. These are cycles measured in weeks and months, rather than quarterly or annual cycles seen traditionally.
Engagement	Engagement measures help oversight organizations understand the level of collaboration that has been achieved. Timely involvement of stakeholders from the workflow supported by the IT system results in a deeper understanding of intended usage. Evolution of the workflow to better utilize technology results from engagement with the correct decision makers.
Quality	Quality measures at senior oversight levels have less to do with software defect rates than they do with the quality of the services supported by the IT systems. For example, improvements in wait times for key services, or percentage of "made it through in one pass" attempts to use a service are potential quality measures. These measures, in turn, drive the priorities for quality measures among software teams.
Risk	Risk measures for senior oversight can focus on the development organizations' performance in managing threats to their success, more than those threats themselves. When using Agile methods, confidently asserting the expected success of a program is no longer based on the comprehensive- ness of up-front specification documents. Therefore, an oversight approach for Agile cannot rely on review and approval of such projective documents as the primary mode of risk identification. The short and steady cadence of Agile promotes rapid learning.

Source: SEI Congressional testimony July 14, 2016 to House Ways and Means Committee.

Writing an “Agile RFP”…

What are the pros/cons of using a statement like “The contractor shall use Agile methods in executing the software development” in an RFP?

Not as easy as it seems

- Providing an SOO or SOW that doesn’t say “how” contractors should work while encouraging the mindset and behaviors based on Agile principles is harder than it looks!
- This is where process tailoring on the government side can occur if warranted
- This is where CDRL requirements should be modified to accommodate the incremental nature of document delivery common in Agile settings
- This is where the nature of Technical Reviews should be agreed upon (could be one of the tailoring areas)
- Establishing reward/incentive criteria supporting Agile principles can be tough

No “iconic” RFP language for encouraging Agile development practices exists

Agile/SAFe and Contracting

Moving from a WBS based on *components* to *capabilities*:

- Contractors generally have databases of historical performance data based on a component-based WBS and waterfall-based life cycle
  - Estimating and negotiating contract costs based on capabilities and iterative development is different
- Master contracts with smaller IDIQ or time/materials task orders are more in tune with the Product Increment approach in SAFe

DoD and Federal programs report success using supply, services, and even commercial item contracts

Cost-based or FFP contracts all work

Finding ways to contract for *development capacity*

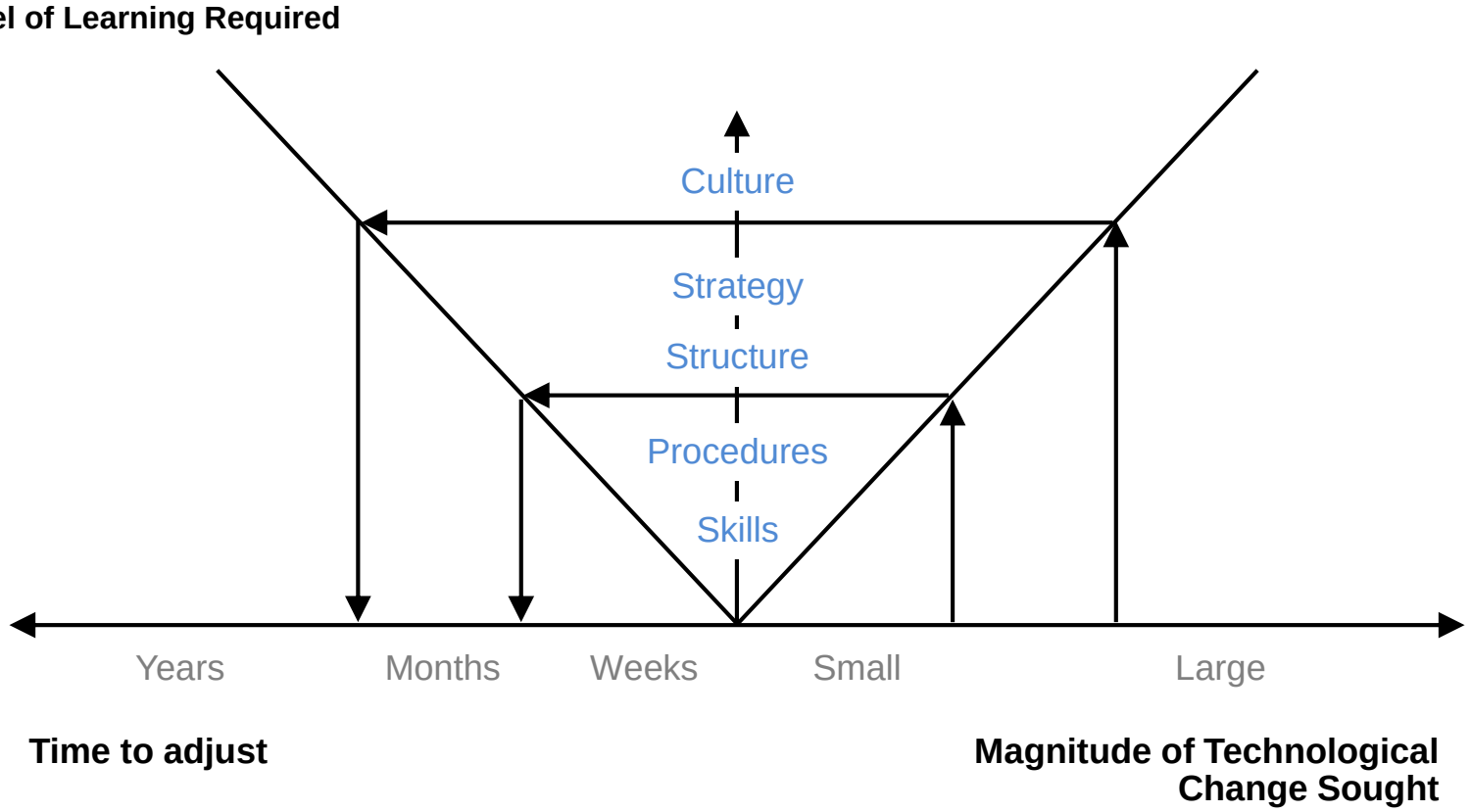
SEI & NDIA System Engineering Agile working group developed a Special Report on this topic:

RFP Patterns and Techniques for Successful Agile Contracting

www.sei.cmu.edu > Digital Library > search on "RFP Patterns"

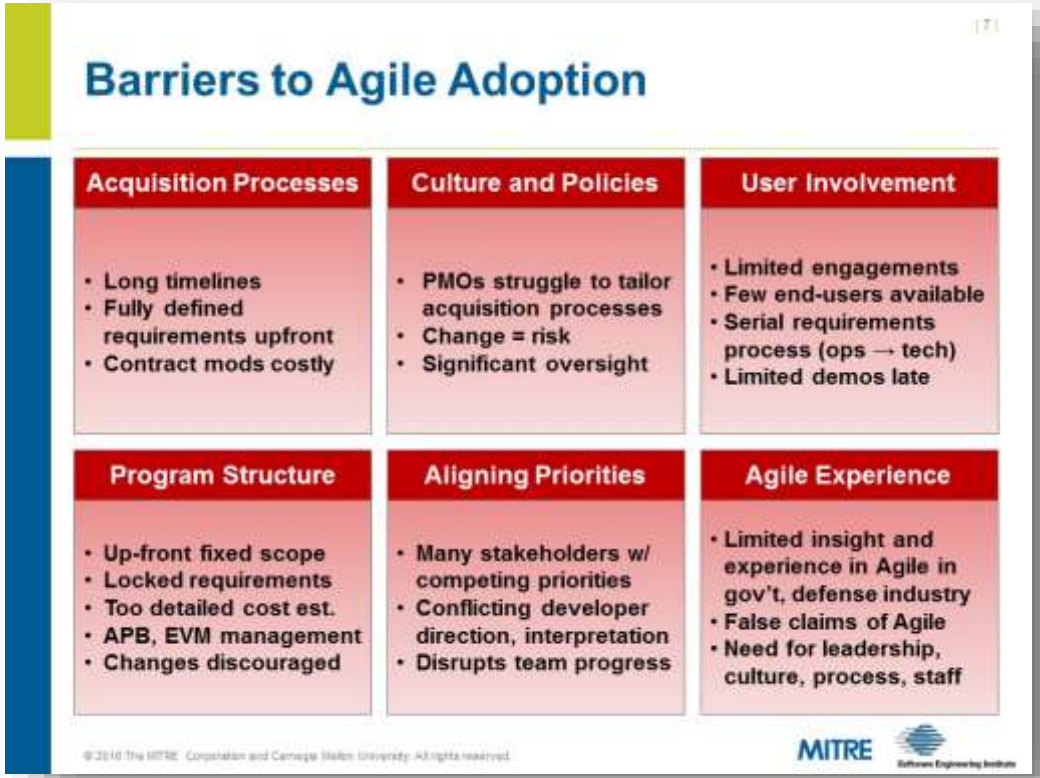
http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=484056

If this is so great, why isn't everyone already doing it?



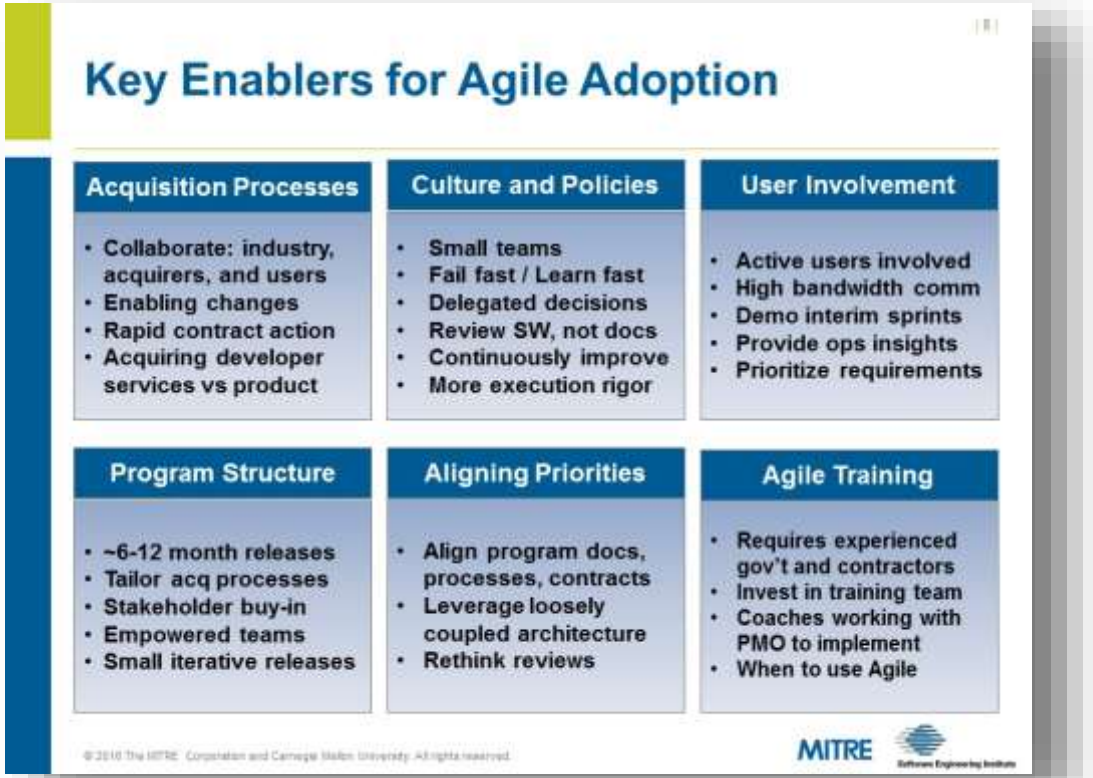
SEI Observations on Agile Adoption Barriers

Which of these do your programs face?



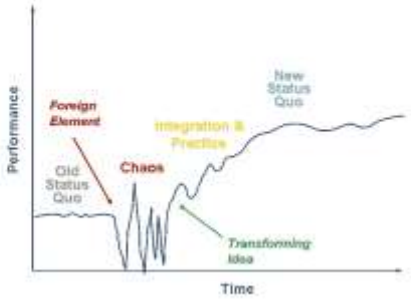
SEI Observations on Key Enablers to Agile Adoption

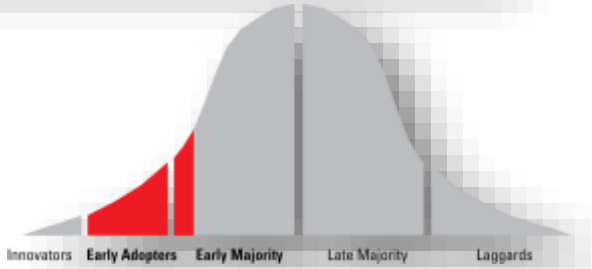
Which of these do your programs exhibit?



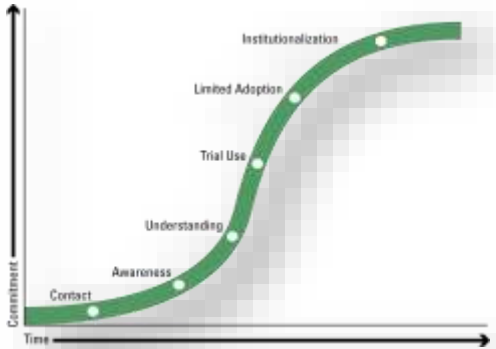
“Traditional” Adoption Tools and Methods Work Well with Agile Adoption

Understand the Change Cycle and Your Adoption Population



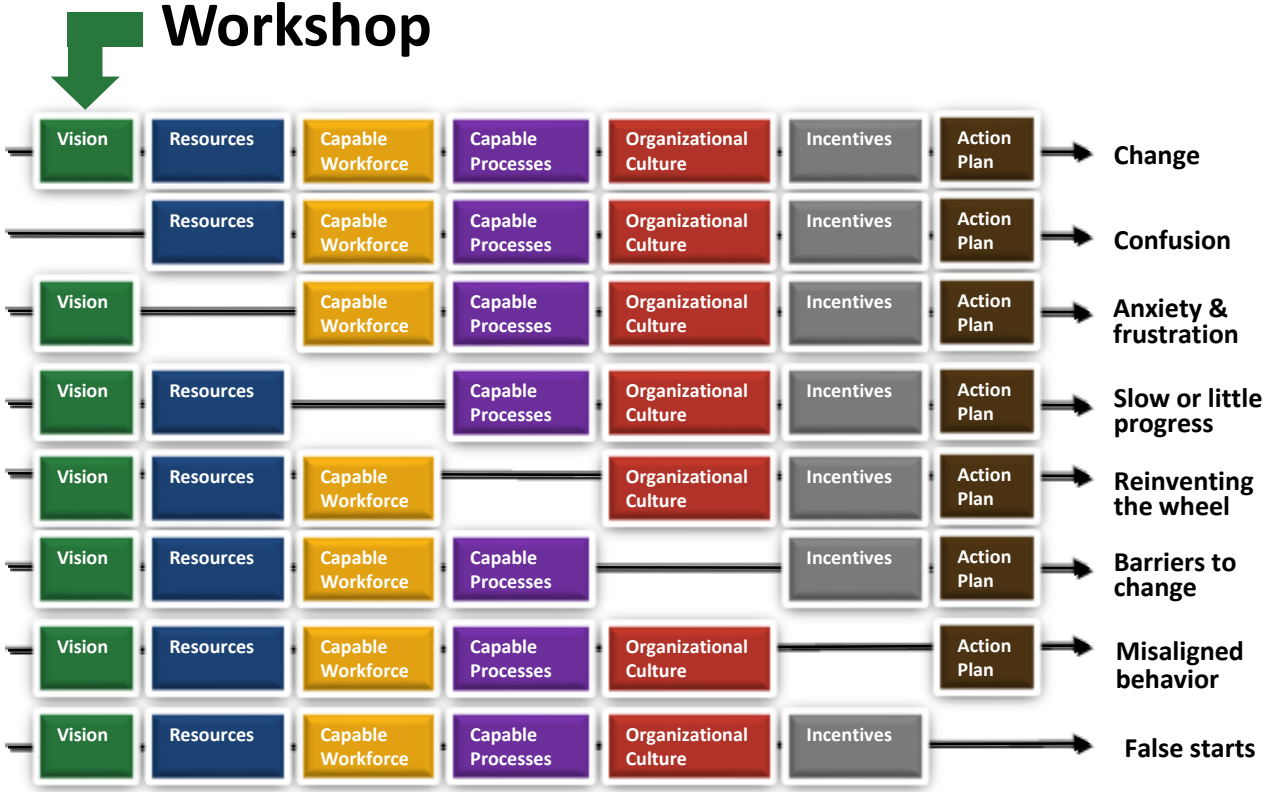


Prepare for Both Communication and Implementation Support Mechanisms that are Needed



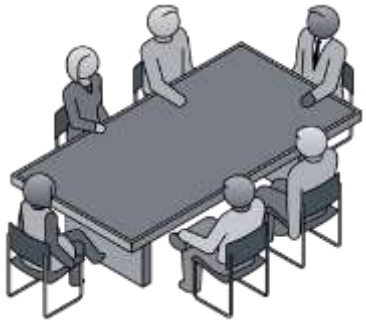
*Adapted from Daryl R. Conner and Robert W. Patterson, “Building Commitment to Organizational Change,” *Training and Development Journal* (April 1983): 18-30.

Where Leadership, Vision, and Goals Fit into Organizational Improvement



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

Attributes of Agile Success in Government Organizations



Top Cover

Permission to "fail fast" (learn fast)

Training in Agile

Use of Agile coach

Enough up-front system and software architecture

Willing to work collaboratively across government/ contractor boundary

Willing and open to adopt new modes of operation

Dedicated staff

What do leaders have to do to change the environment?

About Agile: Summary

Agile is an iterative approach to software delivery that builds and delivers software incrementally from the start of the project, instead trying to deliver it all at once near the end.

- Early opportunity for course correction, especially when the environment changes after a program has begun
- Early risk reduction, especially in user-facing areas of the system
- Shorter “idea to realization” cycle resulting in fast user feedback for future increments of functionality

But it’s about more than software engineering to do it right: Needs business/acquisition process support

Oversight: Responsibility for oversight and due diligence doesn’t change; approach to oversight in an Agile setting does. Some examples:

Flow: Predictable delivery volume, deployment speed	**Engagement:** stakeholder involvement
Quality: Defect backlog	**Risk:** Deferred complexity

Contracting: Benefits can’t be realized without contracting approaches that allow for fast learning & pivoting. Some examples:

Supply contracts	Blanket contracts w/pre-qualified contractors/IDIQ pools
Service contracts	Commercial item contracts for development services (FAR 13.5)

The FAR/DFARS encourage bold innovation – the culture has a long way to go

About Agile: Summary (contd.)

(adapted from SEI Testimony to House Ways and Means Social Security Subcommittee)

Agile will not solve all the complex problems associated with software-dominant systems acquisition and sustainment efforts

- But it has contributed significantly to successful efforts (both in IT and weapons systems)

Benefits from using Agile methods only manifest when the developer and acquisition efforts are aligned

Government obligations in oversight must change when Agile is the focus of development

- SEI has observed negative consequences in organizations that do not address these changes.

Changing the oversight approach in Agile settings means asking different questions on a new cadence

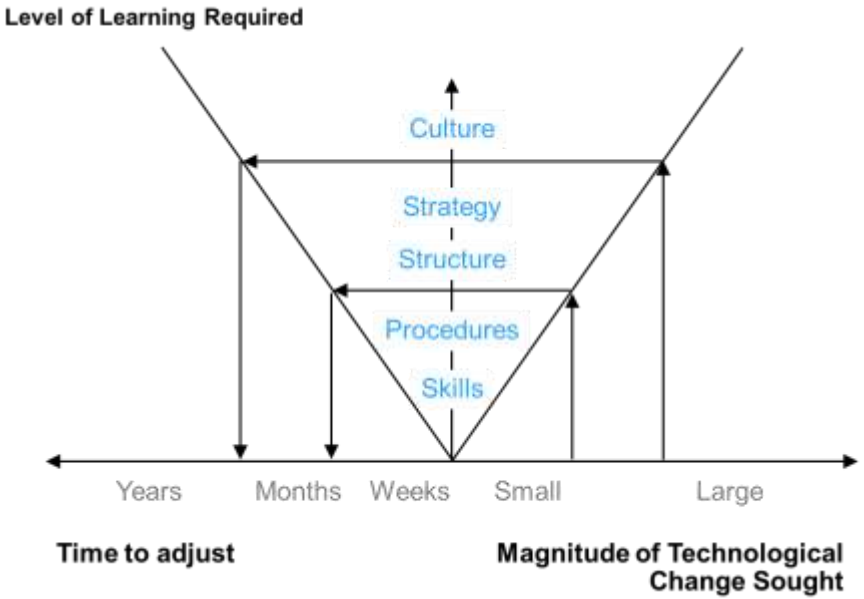
- Leads to different measurement and reporting approaches as well.

A focused government workforce development effort is required to enable the knowledge, skills, and abilities needed for effective oversight and interaction in Agile settings.

[1]July 14, 2016, Link: http://waysandmeans.house.gov/event/hearing-modernizing-social-securitys-information-technology-infrastructure/

Moving Forward: Are You Ready?

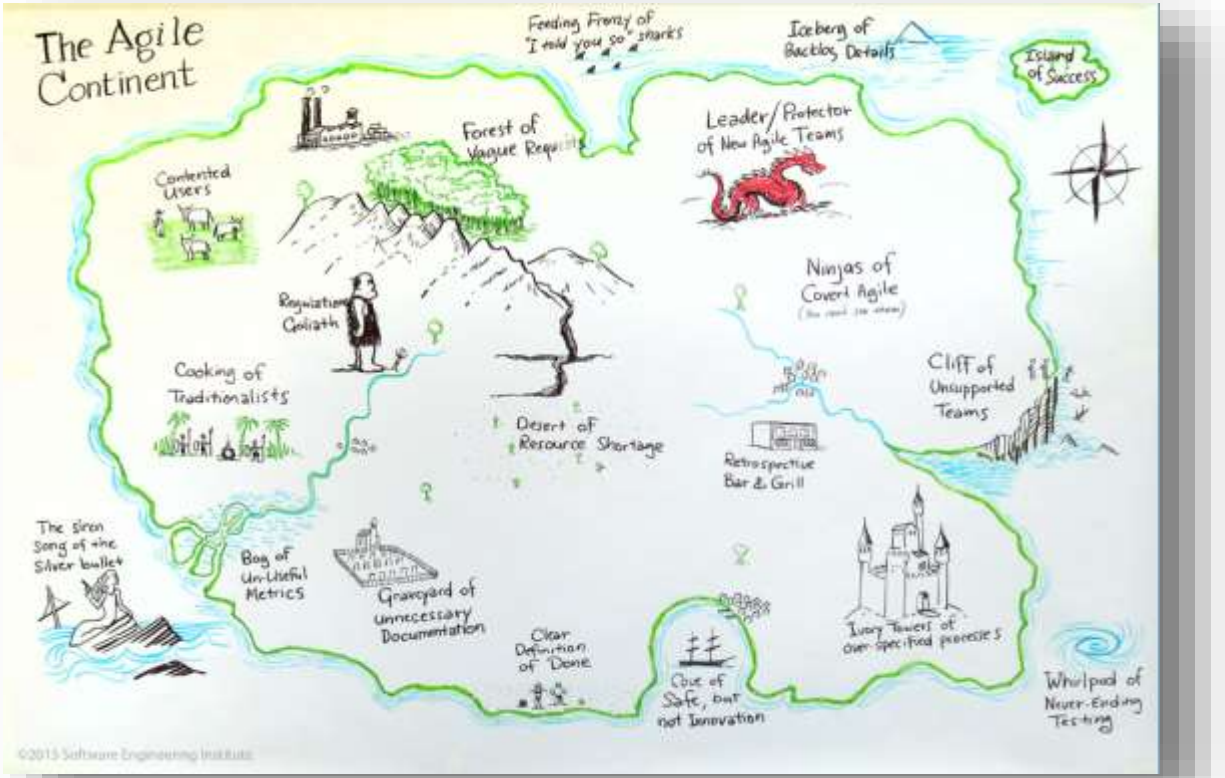
Simplistic View of Way Forward



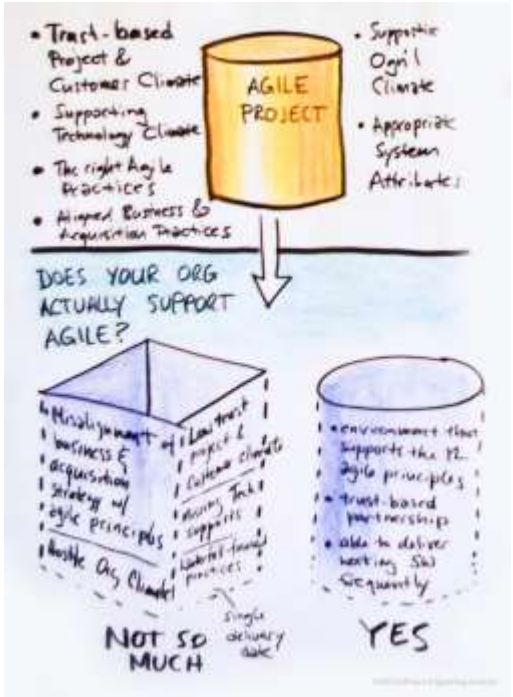
Work your way up this scale:

- Start skill building early – it will be a long term journey
- Determine what procedural/life cycle changes need to be made
- Re-align structures around the different processes/procedures needed to satisfy execution needs
- Determine if/how acquisition and other program strategies change over the long term
- Analyze cultural fit early, and keep working incrementally to desired cultural state while working issues from the other categories

Agile Adoption Landscape is Unique to Each Organizational Setting



Is your program ready for Agile Adoption?



General cultural analyses for adoption of Agile methods don't tend to pick up some of the acquisition issues inherent in regulated environments.

SEI Readiness and Fit Analysis (RFA) and its underlying model explicitly include risk areas known to impede Agile adoption in regulated environments

- More emphasis on business models, goal alignment, and acquisition strategy
- More focus on alignment issues—especially related to staff turnover
- Some particular issues around interfacing with systems engineering in large systems developments

Overview of SEI Readiness & Fit Analysis Process

Agile RFA Categories

Business and acquisition —adoption factors related to business strategy, acquisition strategy, and contracting mechanisms

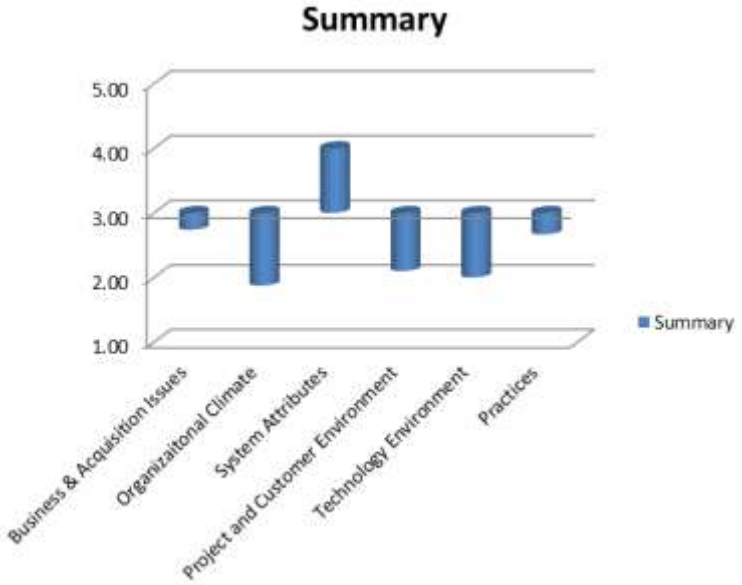
Organizational climate – adoption factors related to sponsorship, leadership, reward systems, values, and similar “soft” issues

System attributes – adoption factors related to the actual characteristics of the system(s) being developed

Project and customer environment – adoption factors related to project management norms, team dynamics and support structures, and customer relationships and expectations

Technology environment – adoption factors related to the technologies that are in place or planned to support the selected Agile methods

Practices – a taxonomy of Agile practices commonly adopted within DoD that is used to understand which practices an organization plans to adopt so that other factors can be calibrated around those expectations



Rating of 3 indicates some issues likely, but nothing unusual for an Agile adoption. Below 3 indicates issues that will negatively impact adoption success. Above 3 indicates issues that could enable adoption success. More important than the rating is the specific risks that RFA participants identify.

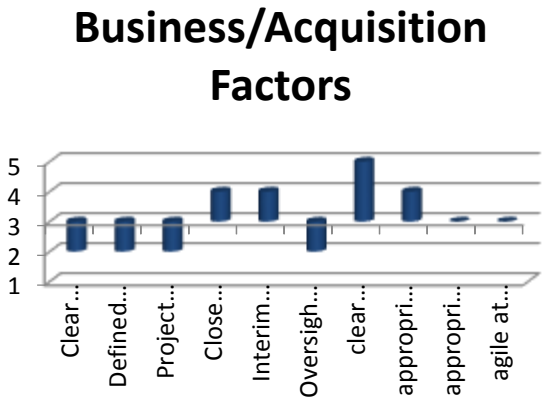
Agile-Specific Factors for Acquisition Contexts

Source: Miller, S. "Is Your Organization Ready for Agile? Part 1", SEI blog, https://insights.sei.cmu.edu/sei_blog/2012/10/readiness-fit-analysis.html

There are some specific acquisition factors for Agile (e.g. enabling interim, incremental delivery) that go beyond typical technology adoption factors

Business and Acquisition Category Particularly Affects Agile Adoption in Regulated Settings

- Business or program goals are clear and reflect stakeholders concerns
- Success strategies (e.g. roadmaps, product portfolios) are defined and clearly communicated
- Funding for the project has been secured
- Mechanisms are in place in the contract and acquisition strategy to allow close collaboration between developers and end users
- Mechanisms are in place in the contract and acquisition strategy that allow for interim demonstration and delivery between official releases



- Contract oversight mechanisms are aligned with Agile principles
- The alignment of software-related goals with program-level goals is clear
- Contract type accounts for use of Agile/Lean methods in the program
- Lifecycle activities are planned in the acquisition strategy that are compatible with Agile/Lean methods
- Acquisition strategy takes into account the use of Agile methods at the scale needed for the program

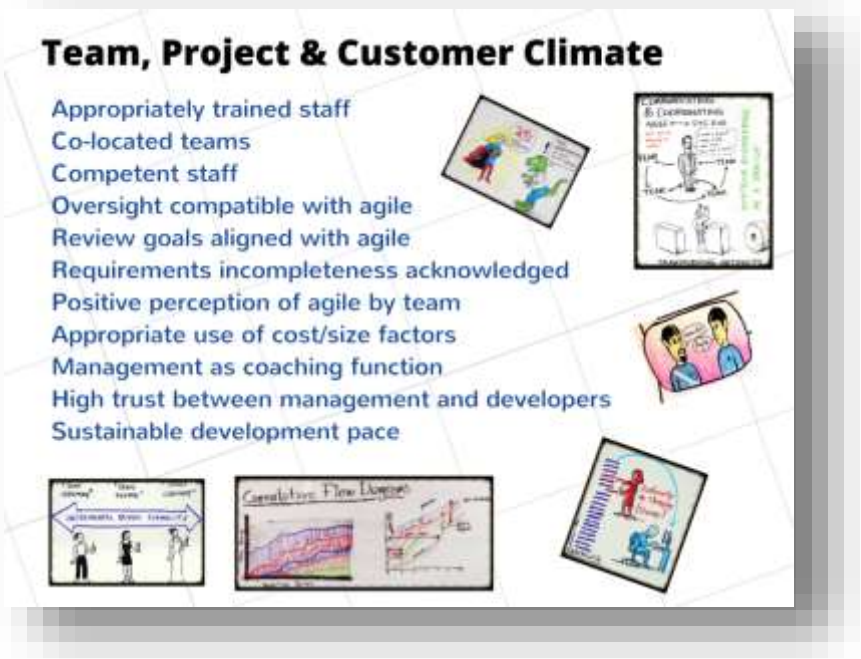
Organizational Climate Enablers for Agile Adoption

Source: Miller, S. "Is Your Organization Ready for Agile? Parts 2 and 3 ",
SEI blog, https://insights.sei.cmu.edu/sei_blog/2012/10/readiness-fit-analysis.html

"Alignment" is a key term for Organizational Climate and other categories of adoption enablers –

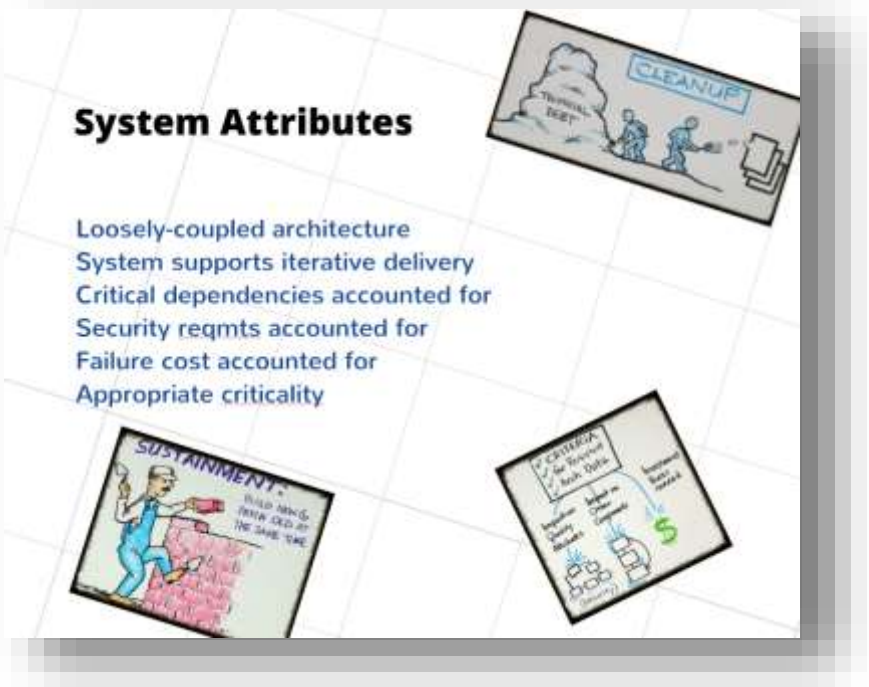
- alignment between the implementation and oversight,
- alignment between the Agile tenets and principles and those of the organization
- Alignment between the leadership and the managers and practitioners

Project, Customer & Team Environment Enablers for Agile Adoption



Although much Agile adoption focuses at the team level, the project and customer climate have significant effect on the program's ability to execute an Agile development successfully, although some of the factors (e.g. appropriately trained staff) are factors in ANY successful program

System Environment Enablers for Agile Adoption



Although most of the Agile adoption enablers are focused on people issues, there are some aspects of the system being developed or evolved that have an effect as well

- Loosely-coupled architecture is a particularly powerful enabler, if present

Technology and Practice Enablers to Agile Adoption

Technology Environment

Technology and tools support agile
Project tools support agile
Project technologies accounted for

Agile Practices

Test-driven development
Short (<6 wk) iterations
User stories
Continuous integration
Prioritized Product Backlog
Product Owner
Relative estimation
Management as Barrier Remover
Self-organizing team
Self-managing team
Pair programming
Daily stand up meetings
End-Iteration Demos
End-Iteration Retrospectives

Early Application of Agile RFA

Program that applied RFA prior to implementation:

- Was able to continue in a very volatile management environment (3 directors in 4 months—no connection to Agile implementation!)
- Agile RFA was a primary communication mechanism to new leadership about risks and opportunities with their culture
- One director “didn’t like the answers” (as to their culture fit at the time) but appreciated “going in with eyes open”

Program that applied RFA during implementation:

- Was able to identify the root cause of some of their adoption issues
- Changed how they communicated some of their progress to management
  - Improved management’s ability to understand both the opportunities and risks that Agile presented to the program

Summary-1

Agile is not a “silver bullet”

Several DoD programs and Federal programs of varying size and complexity have experienced benefits from working productively with contractors using Agile/Lean methods

SAFe is the most frequently adopted scaling framework among DoD contractors

- Lean Engineering on top of Agile
- Flexible but well-described framework
- Not inherently aware of DoD acquisition life cycle and its constraints; requires adaptation

Summary-2

Adopting Agile methods involves (sometimes significant) cultural shifts as well as practice changes

- Transition and adoption approaches for other major organizational changes will work for Agile adoption as well
- Many adoption support mechanisms exist out in the commercial world that can be adapted to regulated settings
  - The SEI technical notes and other resources (blogs, podcasts, etc.) on Agile adoption are meant to support acquisition practitioners in becoming knowledgeable about different issues they may encounter when adopting Agile or Lean methods

BACKUP MATERIALS

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

A Word about Sample RFP Language

No “iconic” RFP language for encouraging Agile development practices exists

- Lots of factors go into what language would be appropriate
- DCMA is considering changes to their policies related to audit points, etc, which could point to some new language—not expected for another year
- NDIA System Engineering Agile working group developed a Special Report on this topic:

RFP Patterns and Techniques for Successful Agile Contracting

www.sei.cmu.edu > Digital Library > search on “RFP Patterns”

http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=484056

Useful Interpretation of Agile Principles for Government Settings (1/3)

Agile Principle	Useful Interpretations in Government Settings
The highest priority is to satisfy the customer through early and continuous delivery of valuable software.	In government, the "customer" is not always the end user. The customer includes people who pay for; people who use; people who maintain; as well as others. These stakeholders often have conflicting needs that must be reconciled
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Rather than saying "competitive" advantage, we usually say "operational" advantage. This principle causes culture clash with the "all requirements up front" perspective of many large, traditional approaches.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.	What it means to "deliver" an increment of software may well depend on context. With large embedded systems, we are sometimes looking at a release into a testing lab. Also, for some systems, the operational users are not able to accept all: "deliveries" on the development cadence – because there are accompanying changes in the workflow supported by the software that require updates.
Business people and developers must work together daily throughout the project.	In government settings, we interpret "business" people to be end users and operators, as well as the other types of stakeholders mentioned in Principle 1, since in many government settings, the business people are interpreted as the contracts and finance group.

Source: SEI Congressional testimony July 14, 2016 to House Ways and Means Committee.

Useful Interpretation of Agile Principles for Government Settings (2/3)

Agile Principle	Useful Interpretations in Government Settings
Build projects around motivated individuals. Give them environment and support they need, and trust them to get the job done.	A frequent challenge in government is to provide a suitable technical and management environment to foster the trust that is inherent in Agile settings. Allowing teams to stay intact and focused on a single work stream is another challenge.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	In today's world, even in commercial settings, this is often interpreted as "high bandwidth" rather than only face-to-face. Telepresence via video or screen-sharing allows more distributed work groups than in the past.
Working software is the primary measure of progress.	Our typical government system development approaches use *surrogates* for software – documents that project the needed requirements and design – *rather than the software itself*, as measures of progress. Going to small batches in short increments allows this principle to be enacted, even in government setting, although delivery may well to be a test environment or some internal group other than users themselves.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	This principle is a caution against seeing agility just as "do it faster." Note that this principle includes stakeholders outside of the development team as part of the pacing.

Source: SEI Congressional testimony July 14, 2016 to House Ways and Means Committee.

Useful Interpretation of Agile Principles for Government Settings (3/3)

Agile Principle	Useful Interpretations in Government Settings
Continuous attention to technical excellence and good design enhances agility	This is a principle that often is cited as already being compatible with traditional government development.
Simplicity– the art of maximizing the amount of work not done– is essential.	One issue with this principle in government setting is that our contracts are often written to penalize the development organization if they don't produce a product that reflects 100% of the requirements. This principle recognizes that not all requirements we think are needed at the onset of a project will necessarily turn out to be things that should be included in the product.
The best architectures, requirements, and designs emerge from self-organizing teams.	Note that the principle does not suggest that the development team is necessarily the correct team for requirements and architecture. It is however, encouraging teams focused in these areas to be allows some autonomy to organize their work. Another complication in many government settings is that we are often re-architecting and re-designing existing systems.
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	This principle is an attempt to ensure that “lessons learned” are actually learned and applied rather than just being “lessons written”

Source: SEI Congressional testimony July 14, 2016 to House Ways and Means Committee.

Contact Information

Eileen Wrubel
Initiative Lead, Agile in Government
Continuous Lifecycle Solutions Initiative
Software Engineering Institute
Email: eow@sei.cmu.edu
Phone: +1 412 268-9976

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Suzanne Miller
Principal Researcher, Agile in Government
Continuous Lifecycle Solutions Initiative
Software Engineering Institute
Email: smg@sei.cmu.edu
Phone: +1 412 268-9143

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257

Web www.sei.cmu.edu> Research and Capabilities> All Work > Filter by “Agile”