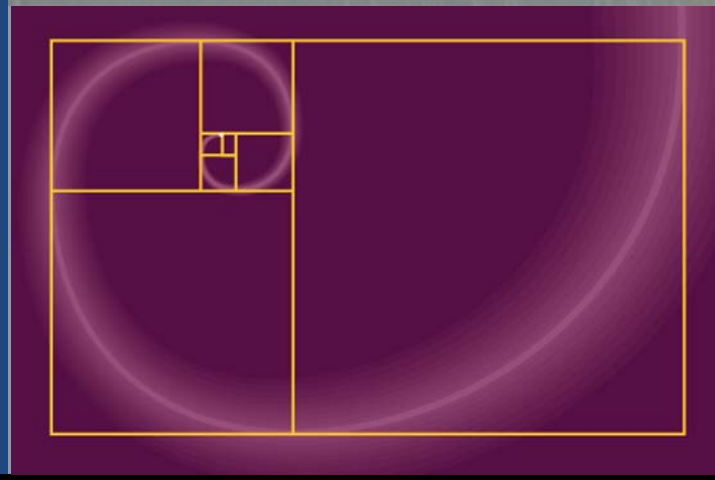
Agile in Government: Practical Considerations

CTSD Agile Adoption in Regulated Settings Team
Software Engineering Institute
July 2015

AIG Module 1 (Introduction)



© 2015 Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

This material has been approved for public release and unlimited distribution except as restricted below.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study. Except for the U.S. government purposes described below, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose this material are restricted by the Rights in Technical Data-Noncommercial Items clauses (DFAR 252-227.7013 and DFAR 252-227.7013 Alternate I) contained in the above identified contract. Any reproduction of this material or portions thereof marked with this legend must also reproduce the disclaimers contained on this slide.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

This material was prepared for the exclusive use of NRO Agile in Government Workshop and may not be used for any other purpose without the written consent of permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0002617

Overall Messages

- Agile is not a silver bullet for software acquisition, <u>but</u> it can be used in a government setting when appropriate
- "Agile" isn't one approach or method – it's an umbrella term and is best thought of as reflecting the Agile tenets and principles of the Agile Manifesto
- <u>How</u> the program office tasks for oversight are performed in a program using Agile is different from a traditional acquisition, but the responsibility of the PMO remains
- There are multiple aspects of a software acquisition that are affected when Agile methods are in use
- As with many adoptions of new practice, "the soft stuff is the hard stuff" – cultural aspects are usually more challenging than technical or management

© 2015 Carnegie Mellon University

Contents

Module 1: Introduction

Module 2: Basic Agile Concepts

- Agile Introduction
- Agile Tenets and Principles
- Differences Between Traditional and Agile
- Common Agile Concepts and Practices
- Agile Mythbusting

Module 3: New Role for PMO

- Why Agile
- The Product Owner Role

© 2015 Carnegie Mellon University

Contents (continued)

Module 4: Agile Program Oversight

- System Engineering
- Integration and Test
- Technical Reviews
- Measurement

Module 5: Agile in the Larger Eco-System

- Documentation
- Contracting
- Regulatory Oversight

Module 6: Enabling an Agile Acquisition Culture

- Enabling and Transitioning to an Agile Culture
- Suggestions for Implementing Agile in Government

Module 7: Summary

© 2015 Carnegie Mellon University

Philosophy for Course

"You are here to make a difference"

- Don't just sit at your desk checking off stuff on a list -- go out, engage in the Agile process productively, and make a difference

Most programs today are software-reliant, if not software-centric!

It isn't unusual for embedded systems to be managed primarily as hardware systems from an acquisition viewpoint.

© 2015 Carnegie Mellon University

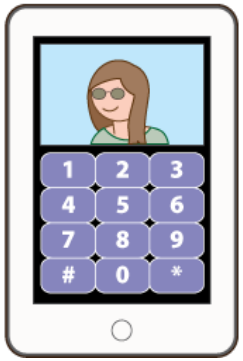
Audience—Who are You?

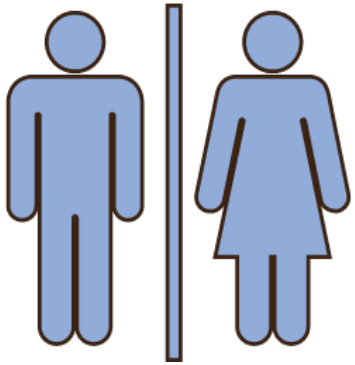
What role do you have in software acquisition?

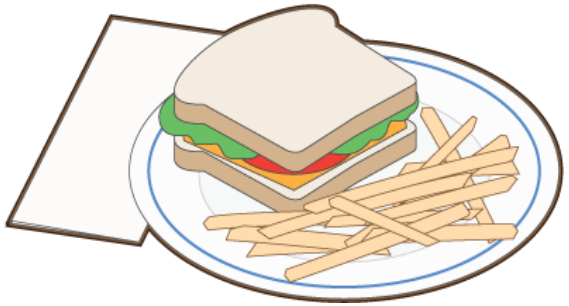
- ***Engineer***
- ***PM***
- ***Test***
- ***Finance Staff***
- ***Contracting Staff***
- ***[your role here]…***

© 2015 Carnegie Mellon University

Course Logistics







© 2015 Carnegie Mellon University

Notional Schedule

Morning:

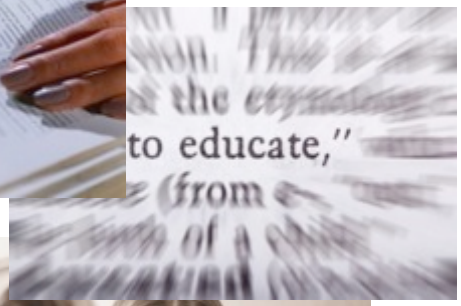
- Introduction
- Agile Basics

Afternoon:

- The Role of Government in Agile Settings
- Agile Oversight
- Agile in the Larger Eco-System
- Agile Culture/Enablement
- Summary

© 2015 Carnegie Mellon University

What Else do You Need to Learn?



Break into pairs and discuss what one thing **BEYOND** what we've discussed will be included that you want to get out of the course (3 min)

Each pair provides two sticky notes, each with one idea, to instructor's flip chart

Instructor will quickly group them and tell group which ones are in and which are out of scope for the course.

© 2015 Carnegie Mellon University

Summary

This course is a first step in building your skills and knowledge of software acquisition issues and solutions in a government program using Agile methods

- Much of the material is generic; we will incorporate program specific information where we can via the soundtrack and supplemental materials

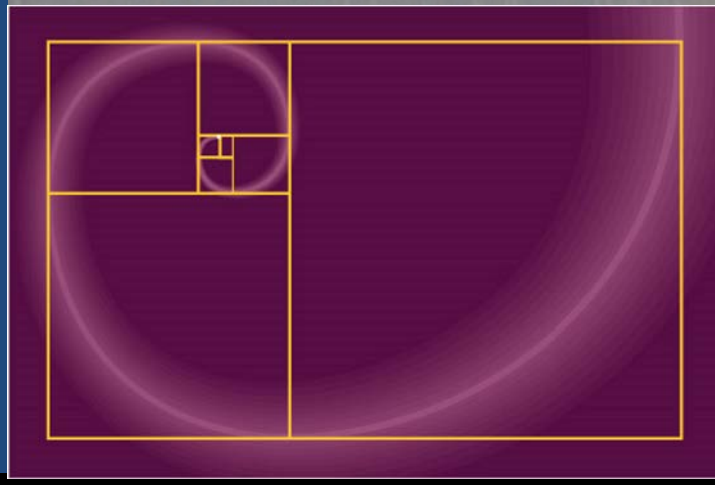
We will accommodate your knowledge needs throughout the course to the best of our ability and will point to resources where time does not permit us to address your needs in the context of this training

© 2015 Carnegie Mellon University

Agile in Government: Practical Considerations

CTSD Agile Adoption in Regulated Settings Team
Software Engineering Institute
July 2015

Module 2 (Basic Agile Concepts)



© 2015 Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

This material has been approved for public release and unlimited distribution except as restricted below.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study. Except for the U.S. government purposes described below, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose this material are restricted by the Rights in Technical Data-Noncommercial Items clauses (DFAR 252-227.7013 and DFAR 252-227.7013 Alternate I) contained in the above identified contract. Any reproduction of this material or portions thereof marked with this legend must also reproduce the disclaimers contained on this slide.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

This material was prepared for the exclusive use of NRO Agile in Government Workshop and may not be used for any other purpose without the written consent of permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0002617

Contents

Module 1: Introduction

Module 2: Basic Agile Concepts

- ***Agile Introduction***
- ***Agile Tenets and Principles***
- ***Differences Between Traditional and Agile***
- ***Common Agile Concepts and Practices***
- ***Agile Mythbusting***

Module 3: New Role for PMO

- Why Agile
- The Product Owner Role

© 2015 Carnegie Mellon University

Contents (continued)

Module 4: Agile Program Oversight

- System Engineering
- Integration and Test
- Technical Reviews
- Measurement

Module 5: Agile in the Larger Eco-System

- Documentation
- Contracting
- Regulatory Oversight

Module 6: Enabling an Agile Acquisition Culture

- Enabling and Transitioning to an Agile Culture
- Suggestions for Implementing Agile in Government

Module 7: Summary

© 2015 Carnegie Mellon University

Learning Objectives:
Module 2: Basic Agile Concepts

After Module 2, participants will be able to:

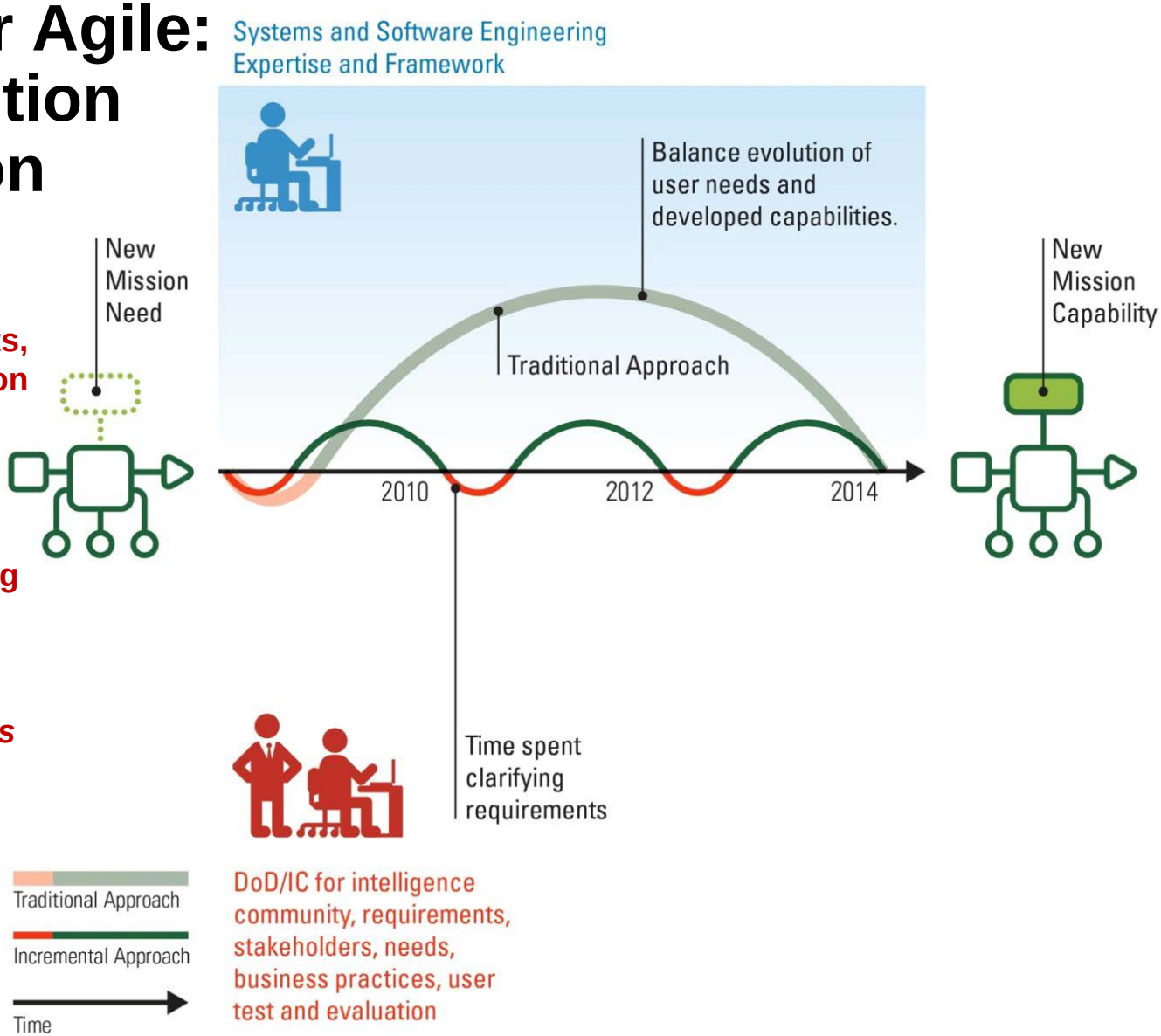
1. Describe the purpose of the Agile tenets and principles
2. Explain at least one reason that Agile could be useful in today's government acquisition environment
3. Recognize common Agile terms and concepts
4. list the main elements of Scrum
5. Describe at least two things that are different between Agile methods and traditional acquisition methods

© 2015 Carnegie Mellon University

Motivation for Agile: Gov't Acquisition and Innovation

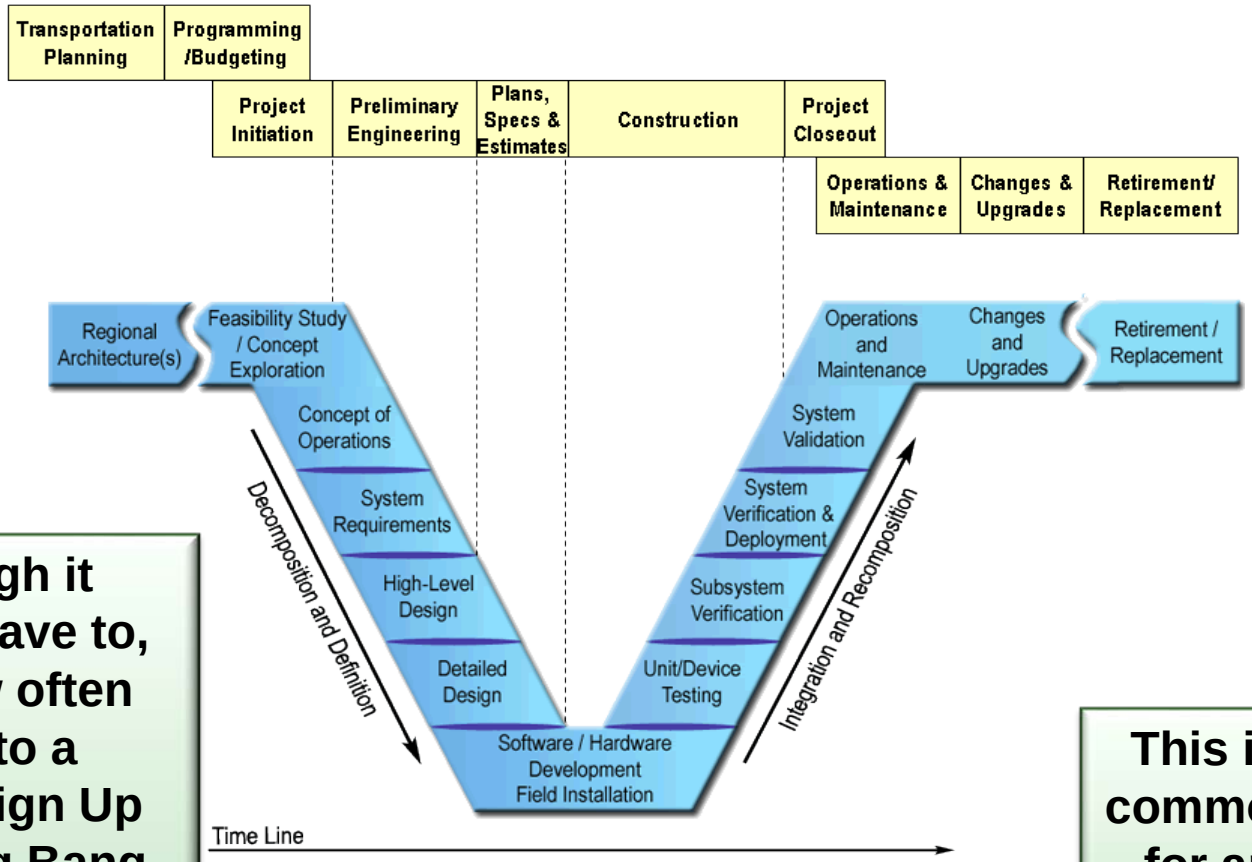
Many regulated environments, like the DoD, NEED innovation and NEED incremental improvements to their systems.

Many of them are now willing to consider changing their approach *if they can do it without getting in trouble with their governing statutes and regulations.*



© 2015 Carnegie Mellon University

The Classic Engineering "V Model"



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

Although it doesn't have to, this view often leads to a "Big Design Up Front, Big Bang Delivery" Approach

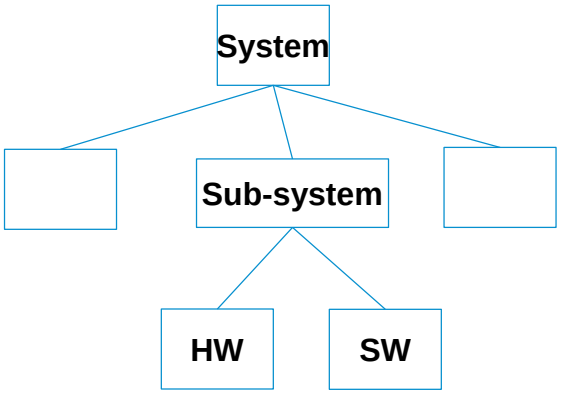
This is the most common life cycle for any system that includes hardware

© 2015 Carnegie Mellon University

V Model was Built Assuming a Hardware-Centric System

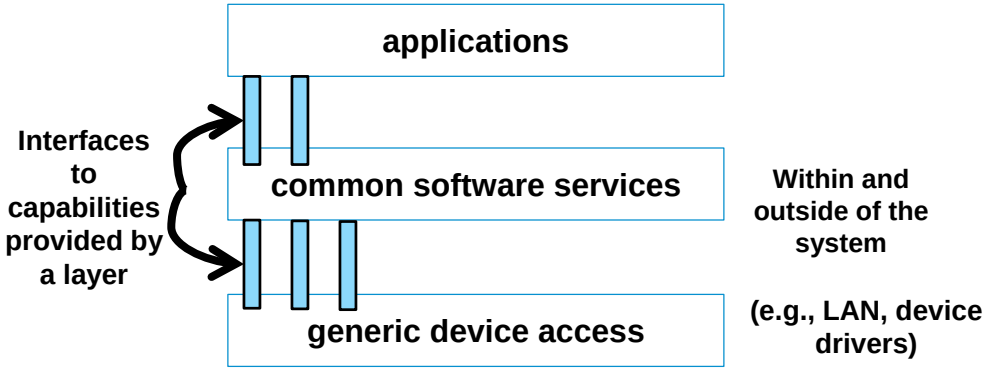
Hardware Sys Development Assumptions

- Systems can be decomposed into discrete, independent, and hierarchically related components (or subsystems).
- **Is part of:** Components can be constructed and integrated with minimal effort based on the original decomposition.
- Quality attributes can be allocated to specific components.



Software Realities

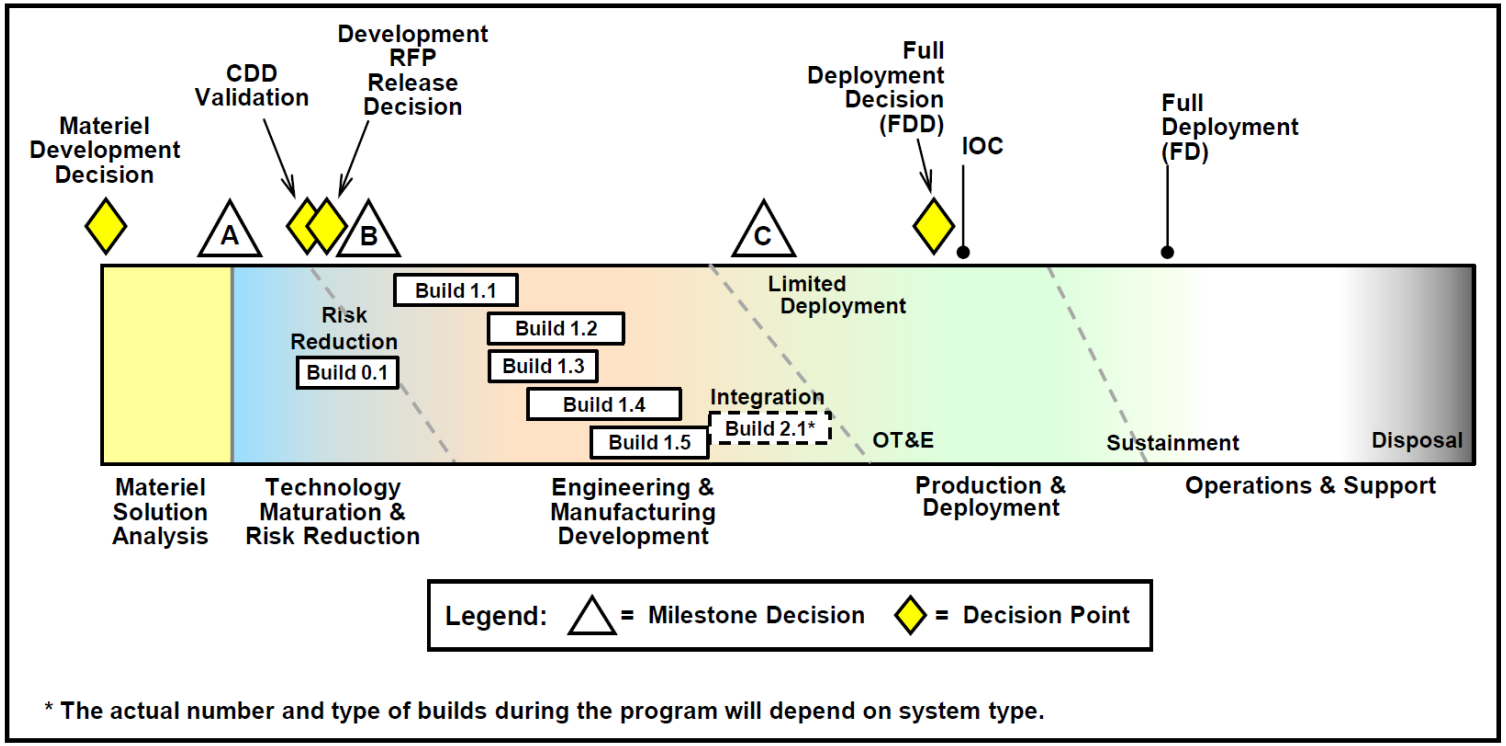
- Software components are often related sets of layered functionality (one layer is *not* contained inside another layer).
- **Is used by:** Interactions of the components (*not* the decomposition) must be managed.
- Quality attributes relate to composite interactions (*not* to individual components).



© 2015 Carnegie Mellon University

Recent Changes in DoD Acquisition Guidance Explicitly Address IT Systems-1

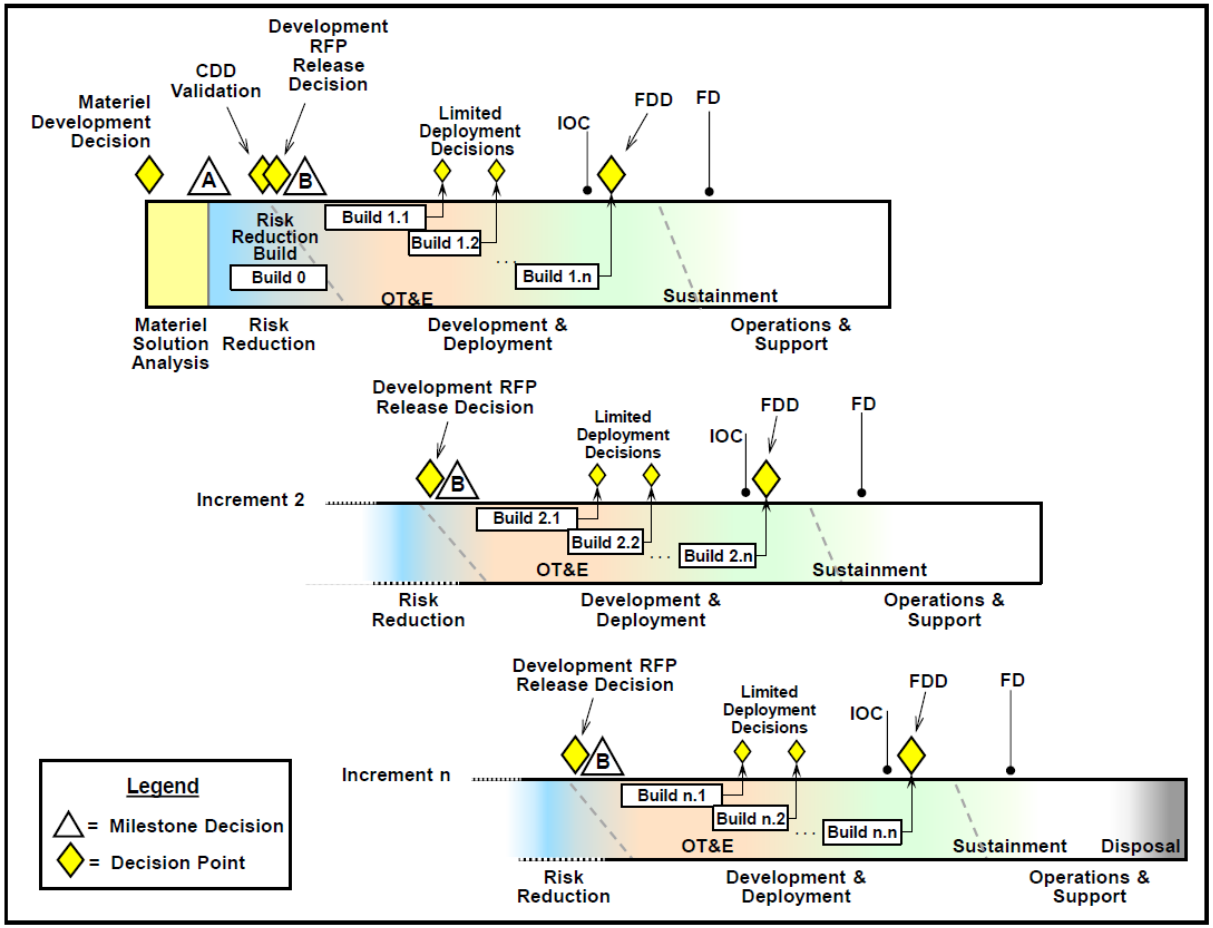
Figure 4. Model 2: Defense Unique Software Intensive Program



© 2015 Carnegie Mellon University

Recent Changes in DoD Acquisition Guidance Explicitly Address IT Systems-2

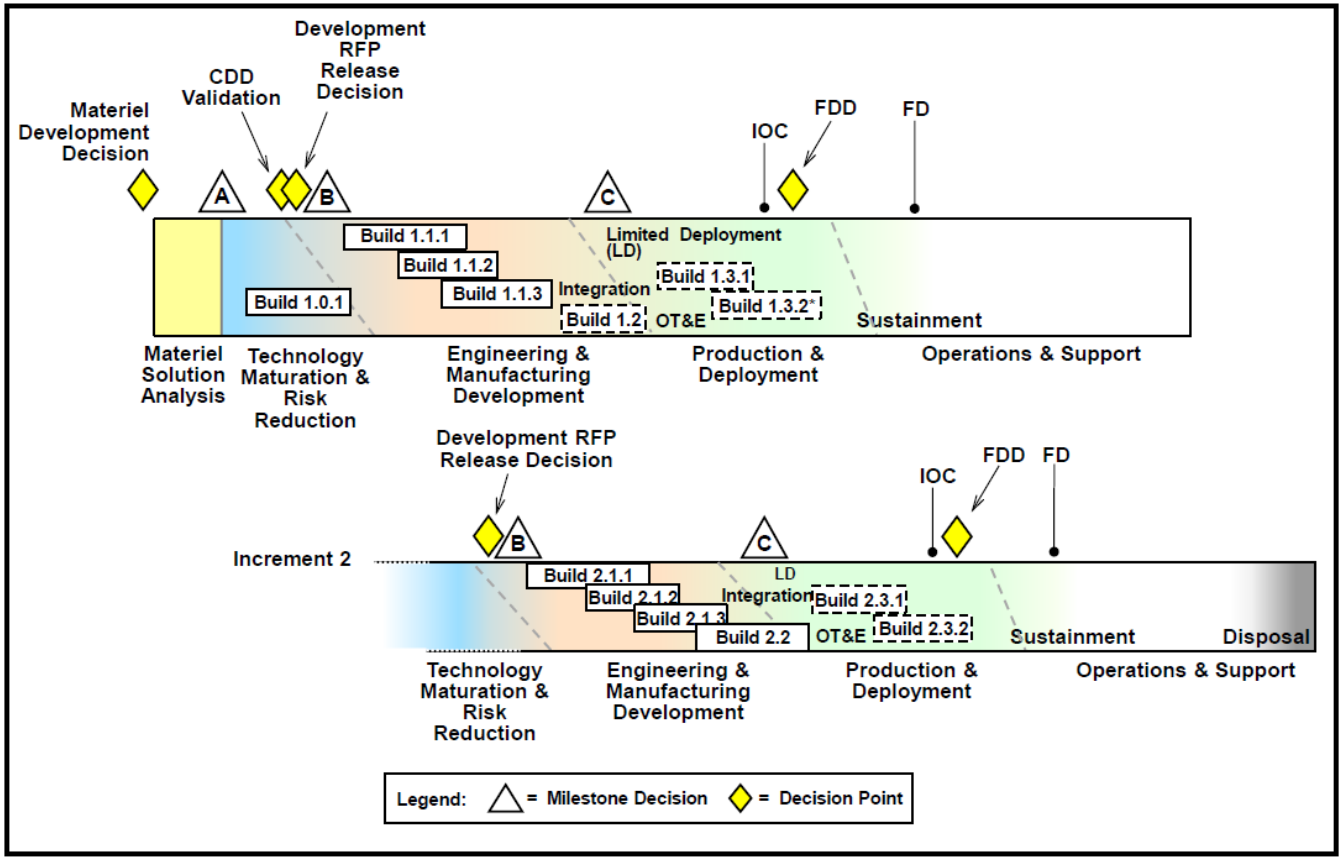
Figure 5. Model 3: Incrementally Deployed Software Intensive Program



© 2015 Carnegie Mellon University

Recent Changes in DoD Acquisition Guidance Also Address SW-Dominant HW/SW Systems

Figure 8. Model 6: Hybrid Program B (Software Dominant)



© 2015 Carnegie Mellon University

Discussion: Which of These Best Reflects How You Think Your Program Would Best Be Managed from an Acquisition Viewpoint?

Figure 4. Model 2: Defense Unique Software Intensive Program

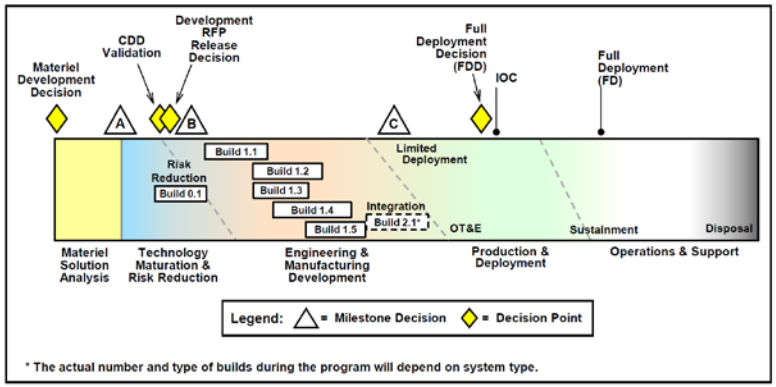


Figure 5. Model 3: Incrementally Deployed Software Intensive Program

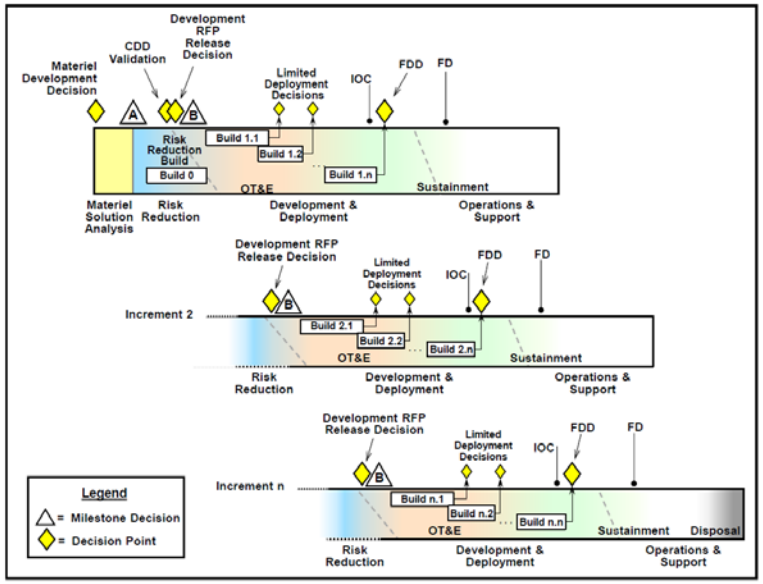
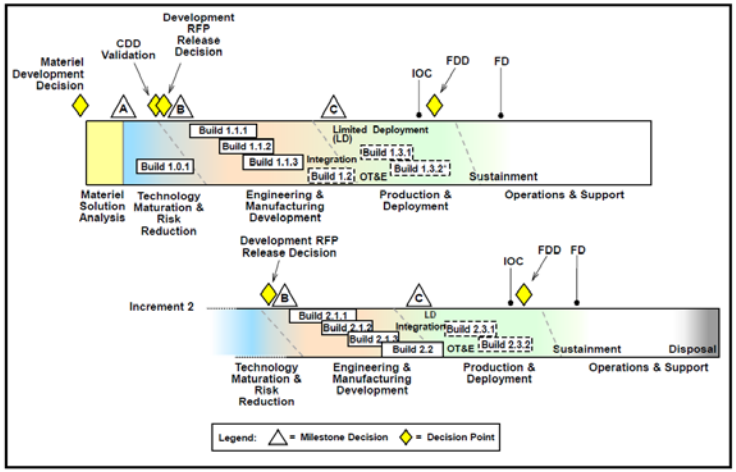


Figure 8. Model 6: Hybrid Program B (Software Dominant)



- Note that any of these could accommodate agile OR waterfall approaches

© 2015 Carnegie Mellon University

Agile Tenets, Principles, and Concepts

© 2015 Carnegie Mellon University

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**.

http://www.agilemanifesto.org/

Common myth:

The manifesto is often misinterpreted to mean:

no documentation, no process, and no plan!

© 2015 Carnegie Mellon University

Agile Principles-1

All are important aspects of building an Agile culture

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.

© 2015 Carnegie Mellon University

Agile Principles – 2

All are important aspects of building an Agile culture

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

© 2015 Carnegie Mellon University

© 2015 Carnegie Mellon University

Discussion

Which of the above principles and tenets

- Are supported by/compatible with current government acquisition practice?
- Are **un**supported by or **in**compatible with current government acquisition practice?

Which of the principles/tenets do you think are most difficult to express in the your program environment?

© 2015 Carnegie Mellon University

BUT, These were Designed & Focused on Small Teams

Lots of things beyond supporting a small team that you have to focus on when scaling above a few small teams:

- Managing the interfaces among the many products/system components that multiple teams are working on…
- Figuring out how to synchronize releases and events across multiple teams…
- Figuring out how to get the inventory (backlog) of requirements organized 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….

© 2015 Carnegie Mellon University

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

© 2015 Carnegie Mellon University

Lean Thinking and Engineering Principles Can be Applied to Help Scaling Issues (and apply at team level, as well)

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 Work in Process (WIP), reduce batch sizes, and manage queue lengths (apply concepts of product development flow)
7. Apply cadence; synchronize with cross-domain planning
8. Unlock the intrinsic motivation of knowledge workers
9. Decentralize decision-making

Adapted from Leading Safe, Scaled Agile Academy, 2015

© 2015 Carnegie Mellon University

	We can't do this here	We might be able to do this		We do this
Take an economic view				
Apply systems thinking				
Assume variability; preserve options				
Build incrementally with fast, integrated learning cycles				
Base milestones on objective evaluation of working systems				
Visualize and limit Work in Process (WIP), reduce batch sizes, and manage queue lengths				
Apply cadence; synchronize with cross-domain planning				
Unlock the intrinsic motivation of knowledge workers				
Decentralize decision-making				

© 2015 Carnegie Mellon University

Agile is a Team Approach

Agile can't succeed in a vacuum. Different roles to play by:

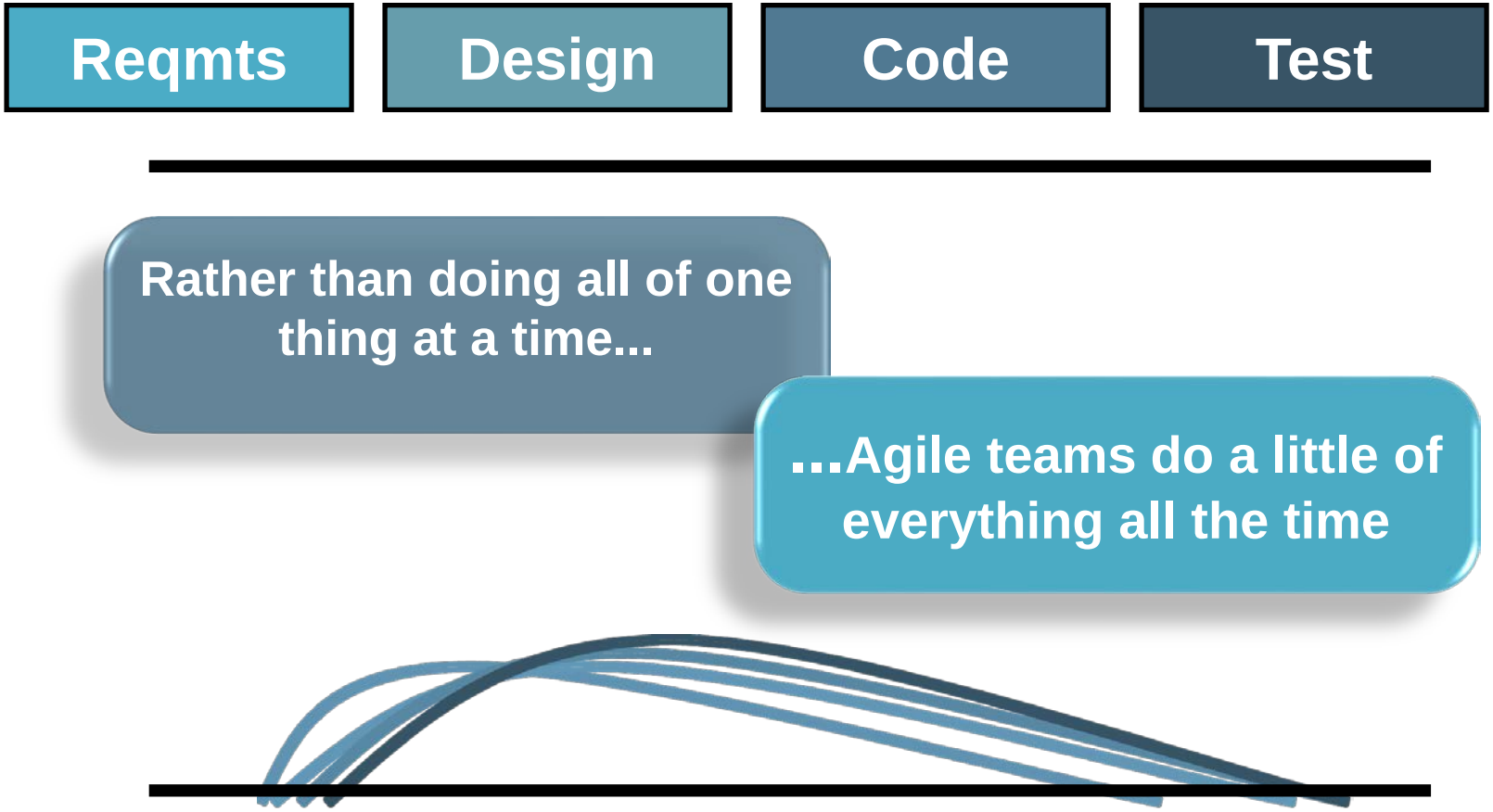
- Developers
- Testers
- End Users
- Customer Representative
- Subject Matter Experts
- Program Office
- Contracts
- Finance
- Certifiers
- ….

© 2015 Carnegie Mellon University

Common Agile Concepts and Practices

© 2015 Carnegie Mellon University

Agile is Incremental AND Iterative



Source: "The New New Product Development Game" by Takeuchi and Nonaka. *Harvard Business Review,* January 1986.

© 2015 Carnegie Mellon University

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—relatively short-duration development cycles that permit changes in scope rather than changes in delivery time frame

Adapted from Nidiffer, Miller, & Carney. Potential Use of Agile Methods in Selected DoD Acquisitions: Requirements Development & Management, SEI-2013-TN-006,

© 2015 Carnegie Mellon University

Methods Generally Termed “Agile”

Scrum

- focused on team management practices

XP (Extreme Programming)

- focused on team technical practices

Crystal

- Encourages risk-based selection of practices; different patterns for different contexts

Test Driven Development (TDD)

- Technical and management practices focused on writing the test that proves acceptance, then coding to that

Dynamic Systems Development Method (DSDM)

- Popular in UK, based on RAD (Rapid Application Development)
- One of the oldest Agile methods, designed for large scale

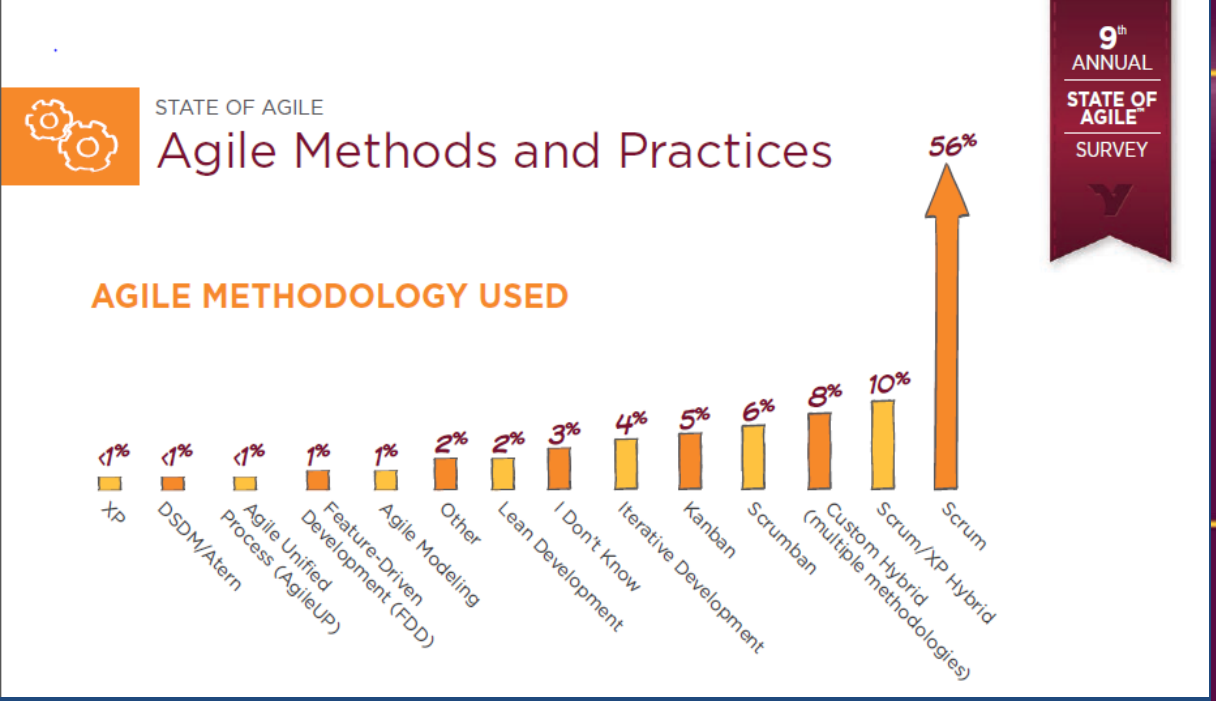
Disciplined Agile Delivery

- Derived from Rational Unified Process, designed to scale

Scaled Agile Framework

- Merger of lean, Kanban, and other Agile methods to support large scale projects

© 2015 Carnegie Mellon University



Source: 9th Annual Survey on State of Agile, Version One

Scrum—The Team Level Practices of Most Agile Scaling Frameworks

© 2015 Carnegie Mellon University

Key Elements of Scrum

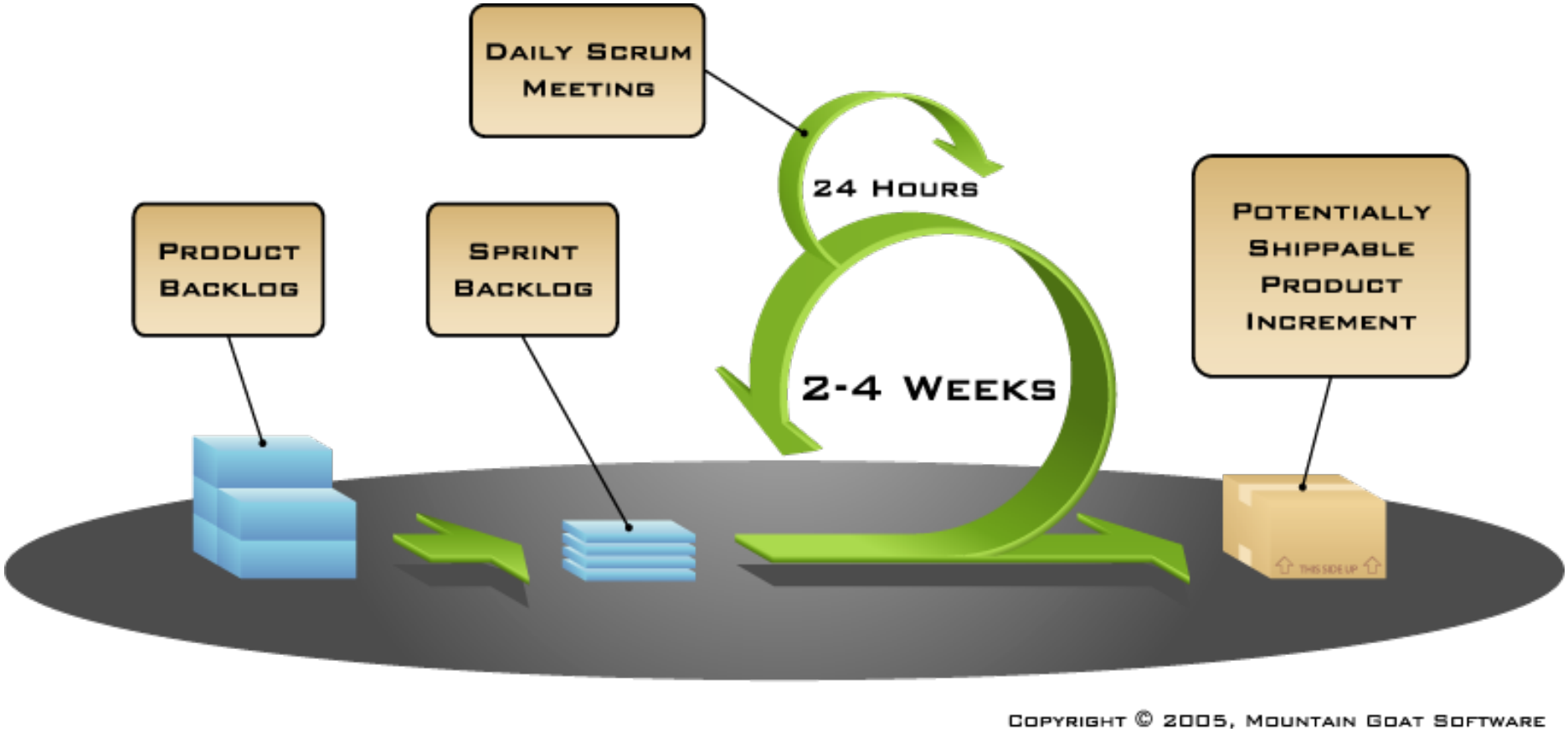


Image available at www.mountaingoatsoftware.com/scrum

© 2015 Carnegie Mellon University

Scrum Framework

Roles

- Product owner
- Scrum Master
- Team

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts
- Potentially Shippable Product Increment

Ceremonies

- Release Planning
- Sprint planning
- Daily scrum meeting
- Sprint review
- Sprint retrospective

Other Concepts Used with Scrum

- Epics
- User Stories
- Sprint

© 2015 Carnegie Mellon University

New Terms - Roles

Product Owner:

- the “voice of the customer,” accountable for ensuring business value is delivered by creating customer-centric items (typically stories (or user stories), prioritizing them, and maintaining them in the product backlog

Scrum Master:

- the process facilitator for a development team who works with the team to identify and find solutions to impediments to progress, as well as maintaining the information radiators that show progress to stakeholders and representing the team in management activities related to the project

Team:

- Cross-functional self-organizing team of 5-10 members including Agile Team Lead, Product Owner or Capability Owner, and developers, testers and SMEs

© 2015 Carnegie Mellon University

New Terms - Artifacts

Product Backlog:

- a prioritized list of (user) stories and defects ordered from the highest priority to the lowest

Sprint Backlog:

- a list of tasks that the team believes need to be completed to satisfy the user stories for that sprint and meet the sprint goal

Burndown Charts:

- a visual tool displaying progress via a simple line chart representing remaining work (vertical axis) over time (horizontal axis)

Potentially Shippable Product Increment:

- the result of a sprint—even though it is unlikely that the project would be cancelled after a particular sprint, the idea is that the product is implemented in such a way that some value could be derived no matter when the project stops

© 2015 Carnegie Mellon University

New Terms – Ceremonies$_1$

Release Planning: planning activities across a defined number of sprints that implement a desired set of features to the point of delivery to the next customer (could be external, often is internal, e.g. system test)

Sprint Planning: planning activities that occur at the beginning of a sprint, including determining the capacity for the sprint, establishing a sprint goal, performing relative estimation on the candidate stories for the sprint, and creating the task list (sprint backlog) the team will use to self-manage the work of the sprint

Daily Scrum Meeting: daily standups that are used to communicate what was accomplished yesterday, what will be accomplished today and identify any impediments to the Scrum Master in order for action to be taken to eliminate the issue.

© 2015 Carnegie Mellon University

New Terms – Ceremonies$_2$

Sprint Review:

- the activity at the end of a sprint that demonstrates the code that has been completed and verifies, with the product owner, that the sprint goal has been met

Sprint Retrospective:

- an activity at the end of the sprint review/demo where the team and the Scrum Master review the processes and practices used in the sprint to identify improvements to be tried in the next or future sprints—Agile's way of "inspect and adapt" for processes

© 2015 Carnegie Mellon University

New Terms - Other

Epics:

- user stories that are too large to directly implement. There is no official threshold to differentiate between an epic and a user story

User stories:

- used in several Agile methods, derive from Extreme Programming; used as the basis for defining the functions a system must provide, and include a written sentence or two and a series of conversations about the desired functionality, to shift the focus from writing about requirements to talking about them

Sprint:

- an iteration of a defined, consistent time span (2-4 weeks is typical) during which the backlog items selected for the iteration are planned, designed, implemented, tested, and demonstrated to the product owner/customer

Information Radiator

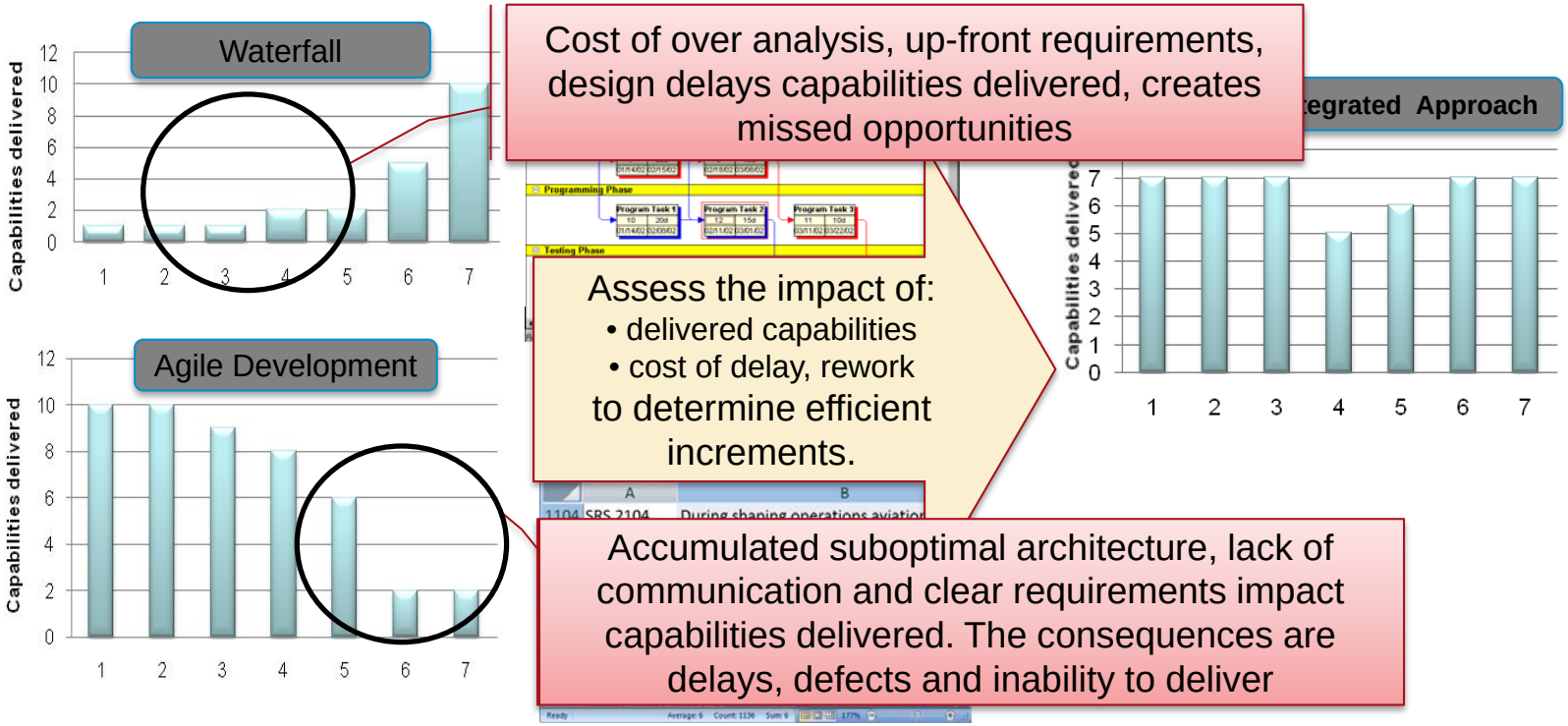
- a physical or virtual display of tasks and progress that is accessible to all stakeholders throughout the project

© 2015 Carnegie Mellon University

Differences between Traditional & Agile Approaches in DoD

© 2015 Carnegie Mellon University

Both Waterfall (HW-centric) and Agile Development (SW-centric) Have Risks



Ozkaya, Ipek. *Internal SEI customer presentation, 2012.*

© 2015 Carnegie Mellon University

Traditional Approach

Strengths of the traditional approach include:

- enables the comparability and repeatability that standardization provides
- enables a contractually verifiable definition of completed intermediate work products
- reduces risks by means of contractually assured baselines

Weaknesses of the traditional approach include:

- the process drives measurement of compliance with itself as a primary measure of success (i.e., rather than measuring success as deploying a workable solution)
- it depends on documents as the basis to verify and validate the requirements, the architecture, and the detailed design
- most of the requirements are completed before any code is written, thus extending development timelines

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.

© 2015 Carnegie Mellon University

Agile Approach

Strengths of this approach include

- early insight by the users into the shape of the solution
- early course correction
- “fail fast” (If the early solution ideas turn out to be flawed, little time or money is spent before that learning occurs.)
- explicit understanding that the requirements are expected to evolve

Weaknesses of this approach (particularly in large acquisition settings) include

- more dependence on tacit knowledge (e.g., lack of explicit documentation) as the basis for decision-making than is comfortable for most acquisition organizations
- dependence on availability of actively engaged user/customers
- difficulty in aligning implementation-driven artifacts and measures with those of the larger traditional acquisition setting.

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.

© 2015 Carnegie Mellon University

Applying Traditional vs. Agile Approaches

Traditional approach

- Is consistent with the acquisition lifecycle guidance provided in the DoD Acquisition Deskbook and its supporting documents.
- 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 can work 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.

© 2015 Carnegie Mellon University

Top Level View of Difference in Traditional/ Agile Programs

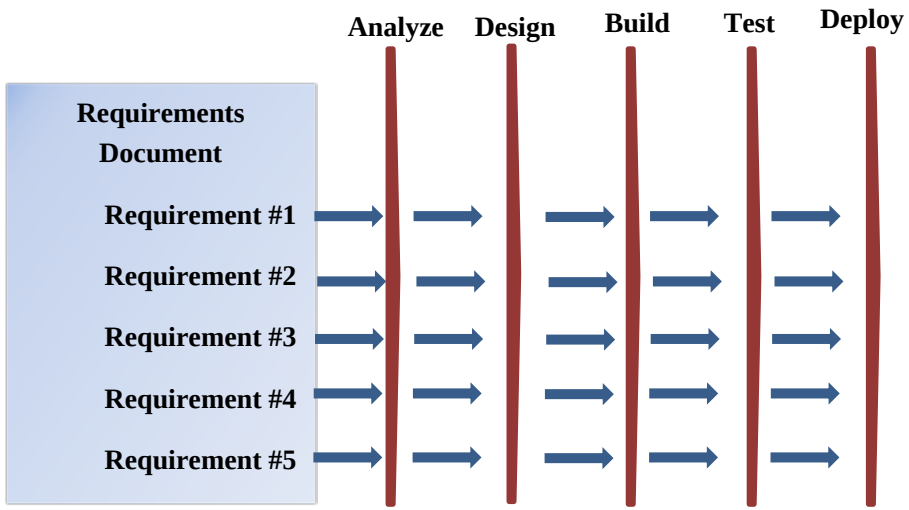
The Worst of Traditional	The Best of Agile
Plan the work—especially the budget, schedule, and deliverables—to the maximum extent possible before beginning any design or code.	• Near-term plans contain more detail, while plans further out on the time horizon contain fewer details. • The overall vision is broken down into a roadmap, which is further broken down into release plans, which are further broken down into sprint or iteration plans, which are further broken down into daily plans. • Requirements are rank ordered, not just prioritized as Hi/Med/Lo. • Cost and schedule estimates are prepared for each capability at a high level. Relative estimation versus absolute estimation is employed. • Frequent planning sessions (at the beginning of each iteration) result in detailed, high-fidelity plans. • Risks are assessed and risk mitigation influences planning.
Lock down requirements to prevent gold-plating and scope creep.	• No requirements can be added to an iteration once it has started. • New requirements are evaluated by the stakeholders and prioritized, thus preventing gold-plating and scope creep.
Institute multiple reviews to provide senior leadership oversight as well as to serve as gates for continued work.	• The customer is involved in all aspects of planning and testing. Customer (in the form of the product owner) is involved daily. • There are reviews at the end of each iteration that serve as gates to further work.
Move forward in a step-by-step, sequential manner and only when all parts of the previous steps were complete.	• The code base is integrated and tested daily. • The code base must pass all tests before and after integration. Regression testing is typically done each night.
Capture all details with extensive documentation. *Source: Palmquist, Steve, et al. Parallel Worlds:*	• There is an overall plan. • There are requirements descriptions. • There are cost and schedule estimates. • There are risk assessments. • There is training material (as appropriate). • There is documentation (as appropriate). • There are lessons learned (based on retrospectives).

© 2015 Carnegie Mellon University

Building Blocks of Development-1

Waterfall and Agile use the same development basic building blocks—analyze, design, build, test, and deploy

In waterfall, <u>all</u> requirements move through these stages *en masse* with heavy documentation and formal approval at each stage:



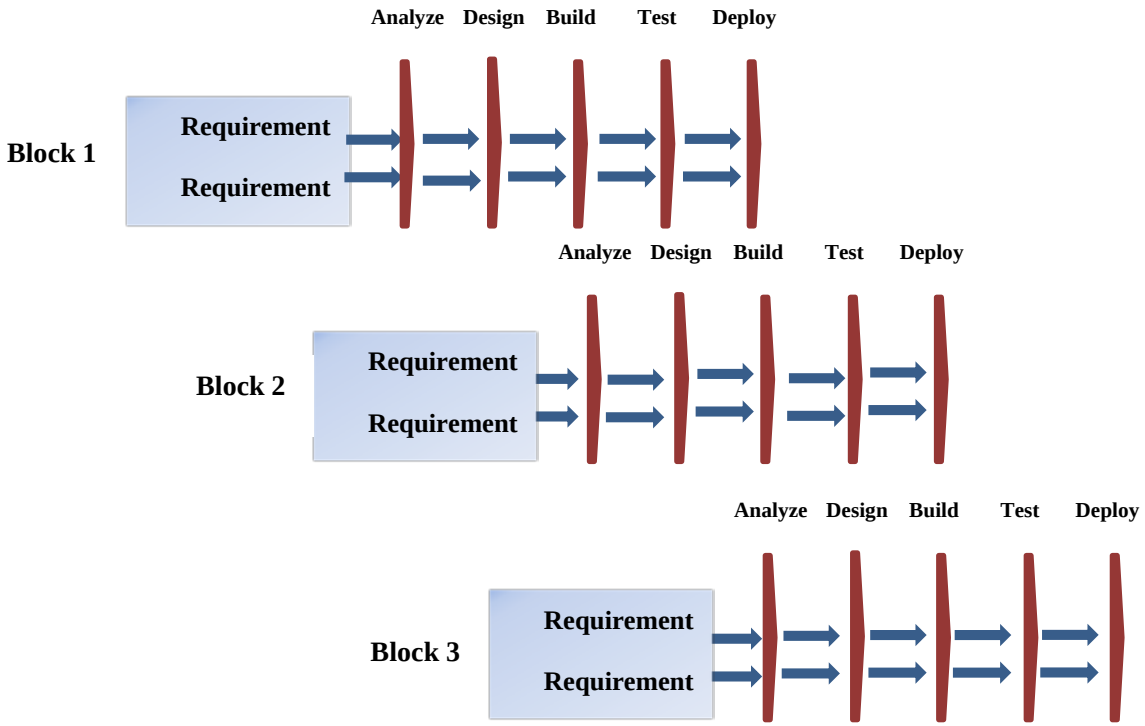
Graphic adapted from http://www.agile-process.org/process.html

© 2015 Carnegie Mellon University

Building Blocks of Development-2

At times the requirements are “blocked out” or delivered in increments, but the requirements for all of the blocks are still determined up front

- So while blocking and increments are techniques to manage schedule and resources, the sequential, one-pass paradigm remains



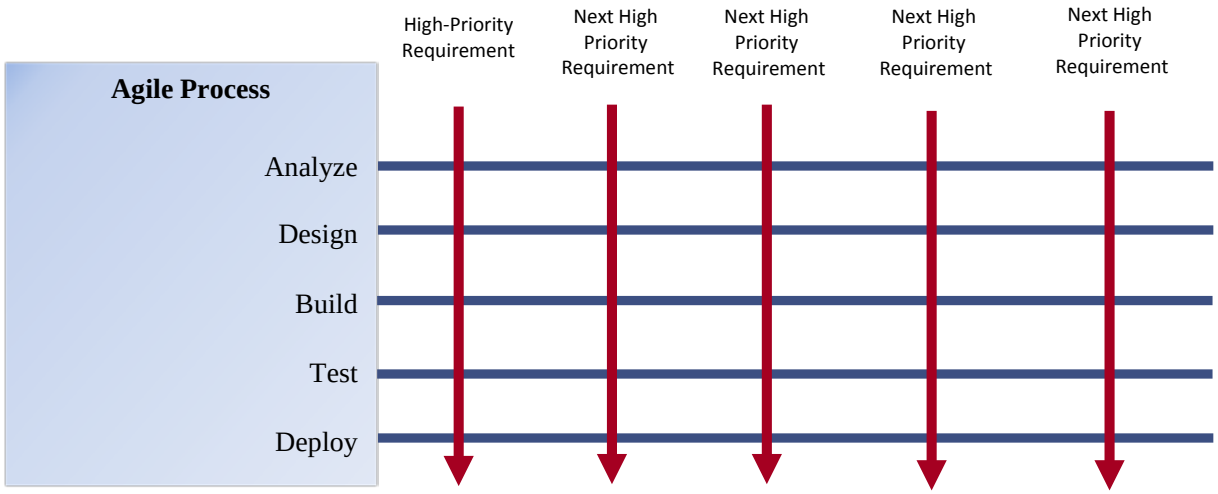
Graphic adapted from http://www.agile-process.org/process.html

© 2015 Carnegie Mellon University

Building Blocks of Development-3

Agile uses the same building blocks—analyze, design, build, test, and deploy—but it looks at them differently

- In Sprint 1, Agile does all of these for the highest priority requirements
- In Sprint 2, Agile does all of these for the next highest priority requirements
- In Sprint 3, Agile does all of these for the next highest priority requirements
- Etc.



Graphic adapted from http://www.agile-process.org/process.html

© 2015 Carnegie Mellon University

Agile Myths, Fables, and Monsters

© 2015 Carnegie Mellon University

Which myths do you hear the most?

	Your Vote
1. Agile is a fad—if I wait long enough, it will go away	
2. Agile teams don't document anything	
3. Agile is "cowboy" programming	
4. Agile only works in co-located environments	
5. Agile is just spiral renamed (or incremental or iterative)	
6. Agile won't work in DoD or Government environments	
7. Agile only works with small projects	
8. You can't use Earned Value Management on Agile Software Developments	

© 2015 Carnegie Mellon University

Myth: Agile is a Fad…if I Wait Long Enough, It Will Go Away…

© 2015 Carnegie Mellon University

For something that's a fad, there sure is a lot of activity related to Agile guidance

DoD and NDAA documents tend to suggest that DoD IT projects follow Agile-like processes and lifecycles

Federal working groups/task forces in place to support these directives (e.g. Section 804 Task Force) [AFE2012]

Government is looking at alternative development processes to enable earlier delivery of capability to users.

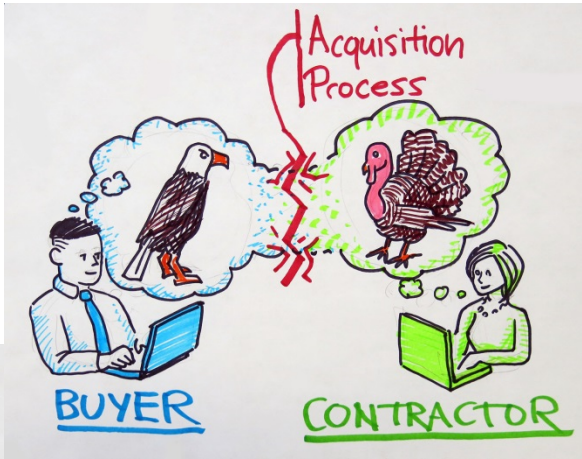
DoD 5000.02 guidance include hybrid life cycle examples that more easily accommodate Agile methods implementation.

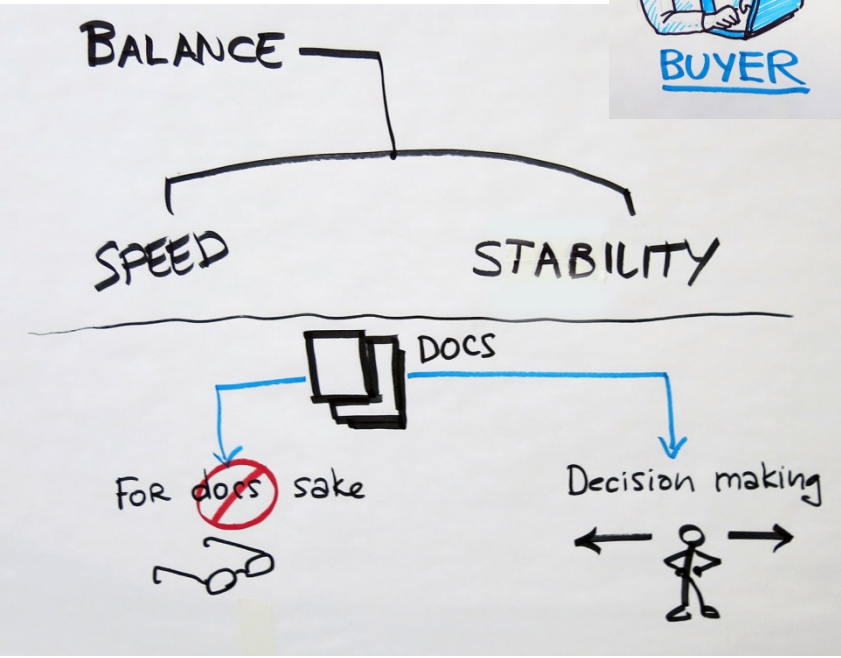
© 2015 Carnegie Mellon University

Myth: Agile Teams Don’t Document Anything

© 2015 Carnegie Mellon University

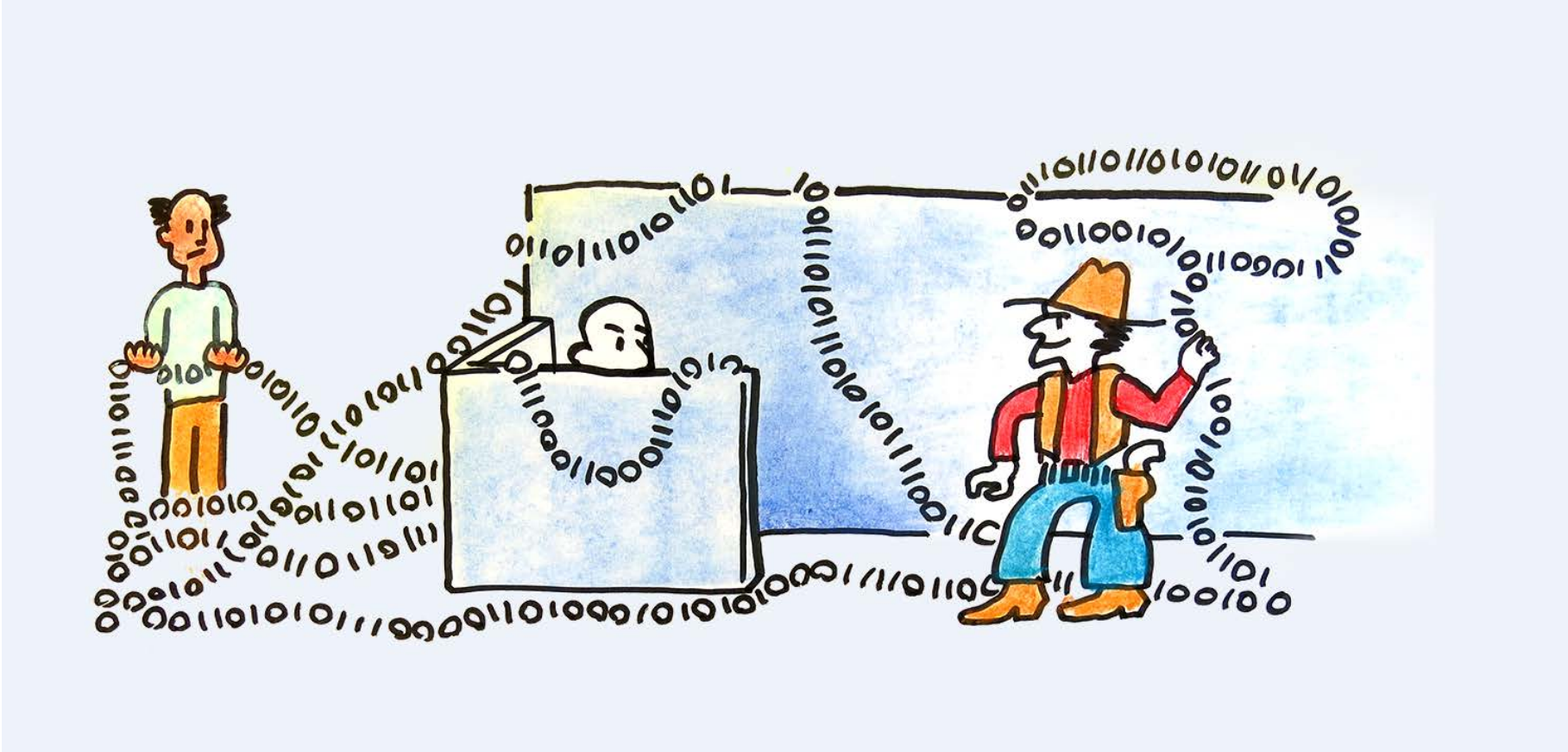
Balance between Speed and Stability is a Hallmark of Agile Documentation





© 2015 Carnegie Mellon University

Myth: Agile is “cowboy programming”



© 2015 Carnegie Mellon University

The Code Cowboy – definitely not an Agile team player!

The Code Cowboy is a force of nature that cannot be stopped. He or she is almost always a great programmer and can do work two or three times faster than anyone else. The problem is, at least half of that speed comes by cutting corners. The *Code Cowboy feels that checking code into source control takes too long, storing configuration data outside of the code itself takes too long, communicating with anyone else takes too long...* you get the idea.

The *Code Cowboy's code is a spaghetti code mess*, because he or she was working so quickly that the needed refactoring never happened. Chances are, seven pages' worth of core functionality looks like the "don't do this" example of a programming textbook, but it magically works. The *Code Cowboy definitely does not play well with others.* And if you put two Code Cowboys on the same project, it is guaranteed to fail, as they trample on each other's changes and shoot each other in the foot.

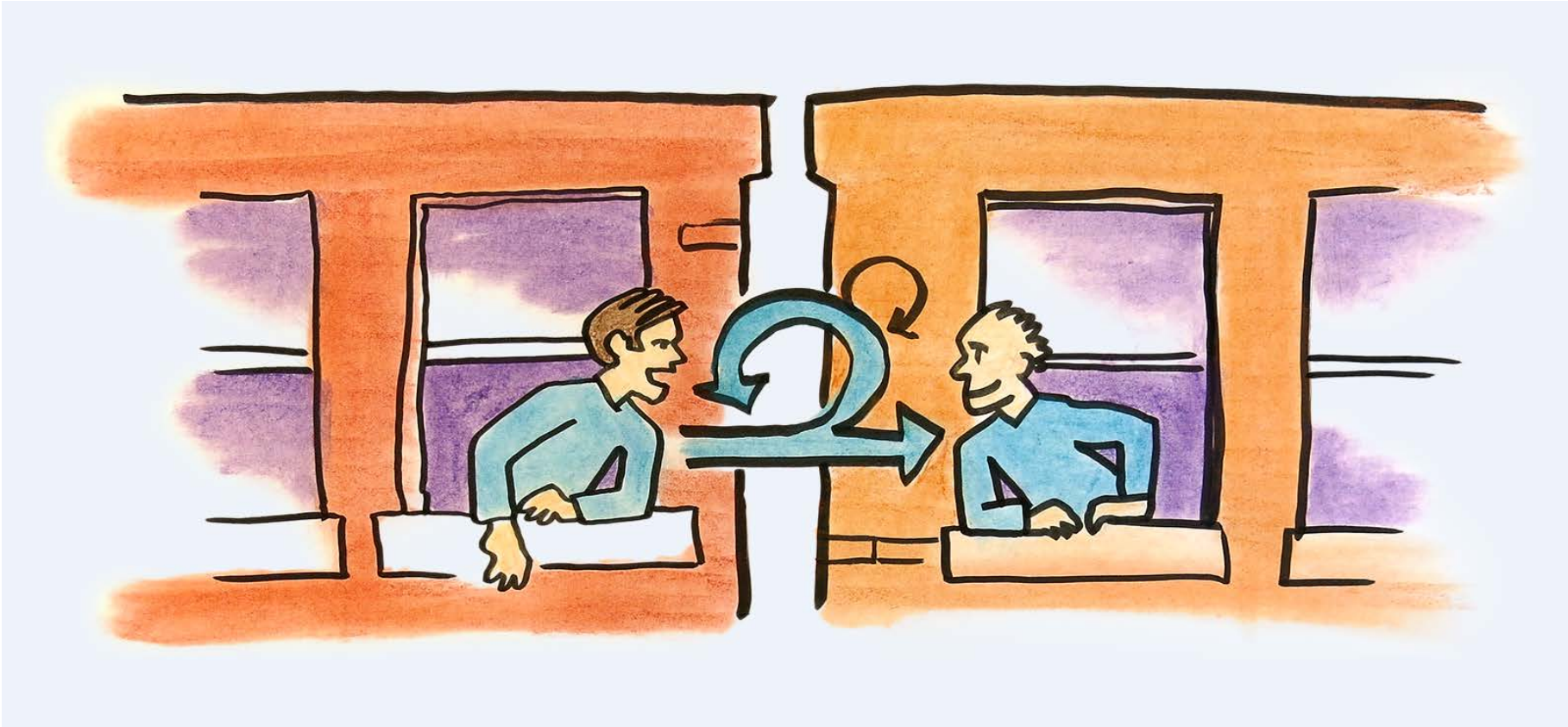
Put a Code Cowboy on a *project where hitting the deadline is more important than doing it right,* and the *code will be done just before deadline every time*. The Code Cowboy is really just a loud, boisterous version of The Ninja. While The Ninja executes with surgical precision, The *Code Cowboy is a raging bull and will gore anything that gets in the way.*

[Above description taken from “10 types of programmers you'll encounter in the field” by Justin James]

Source: http://www.techrepublic.com/blog/10-things/10-types-of-programmers-youll-encounter-in-the-field/262/

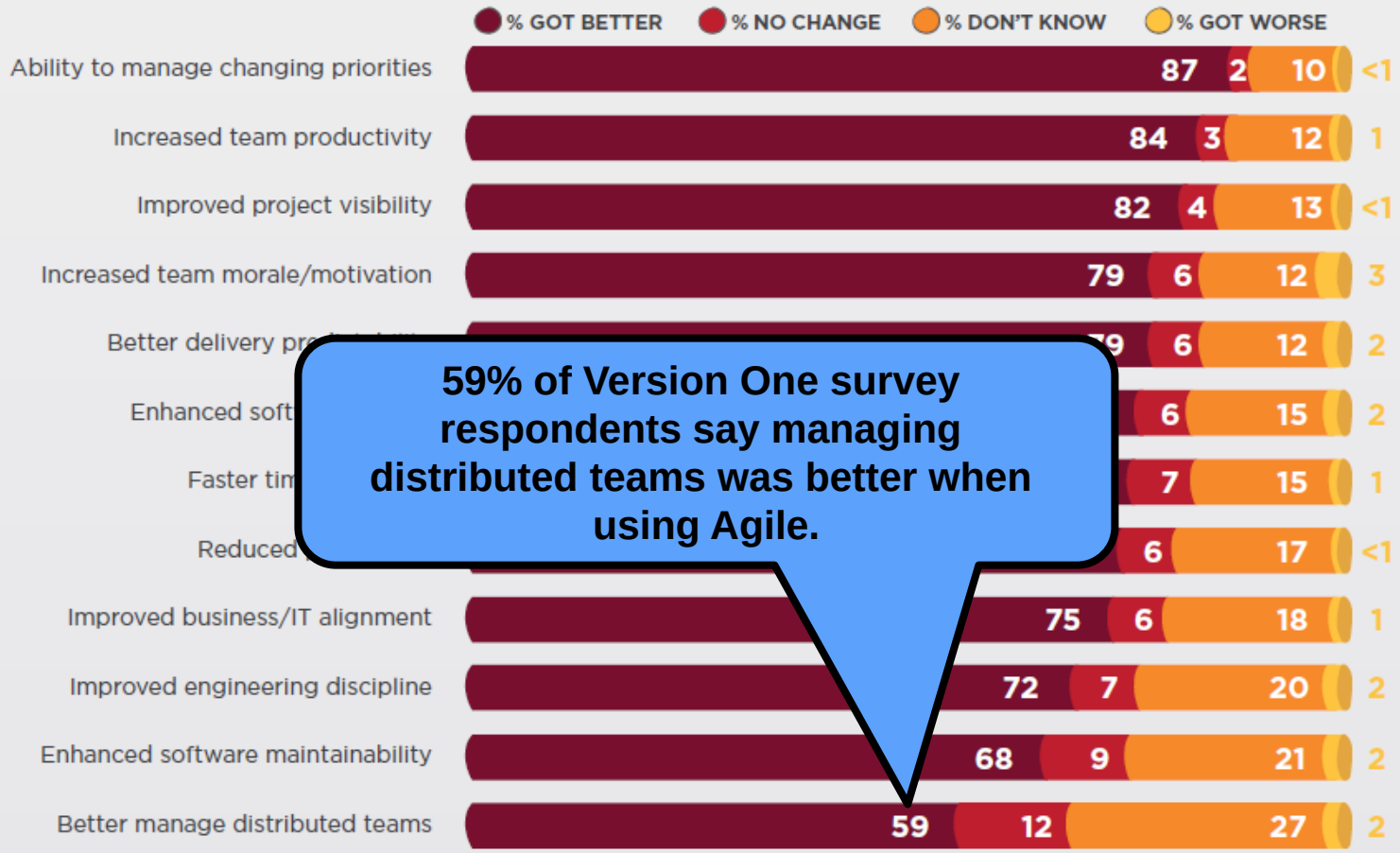
© 2015 Carnegie Mellon University

Myth: Agile Only Works in Co-Located Environments



© 2015 Carnegie Mellon University

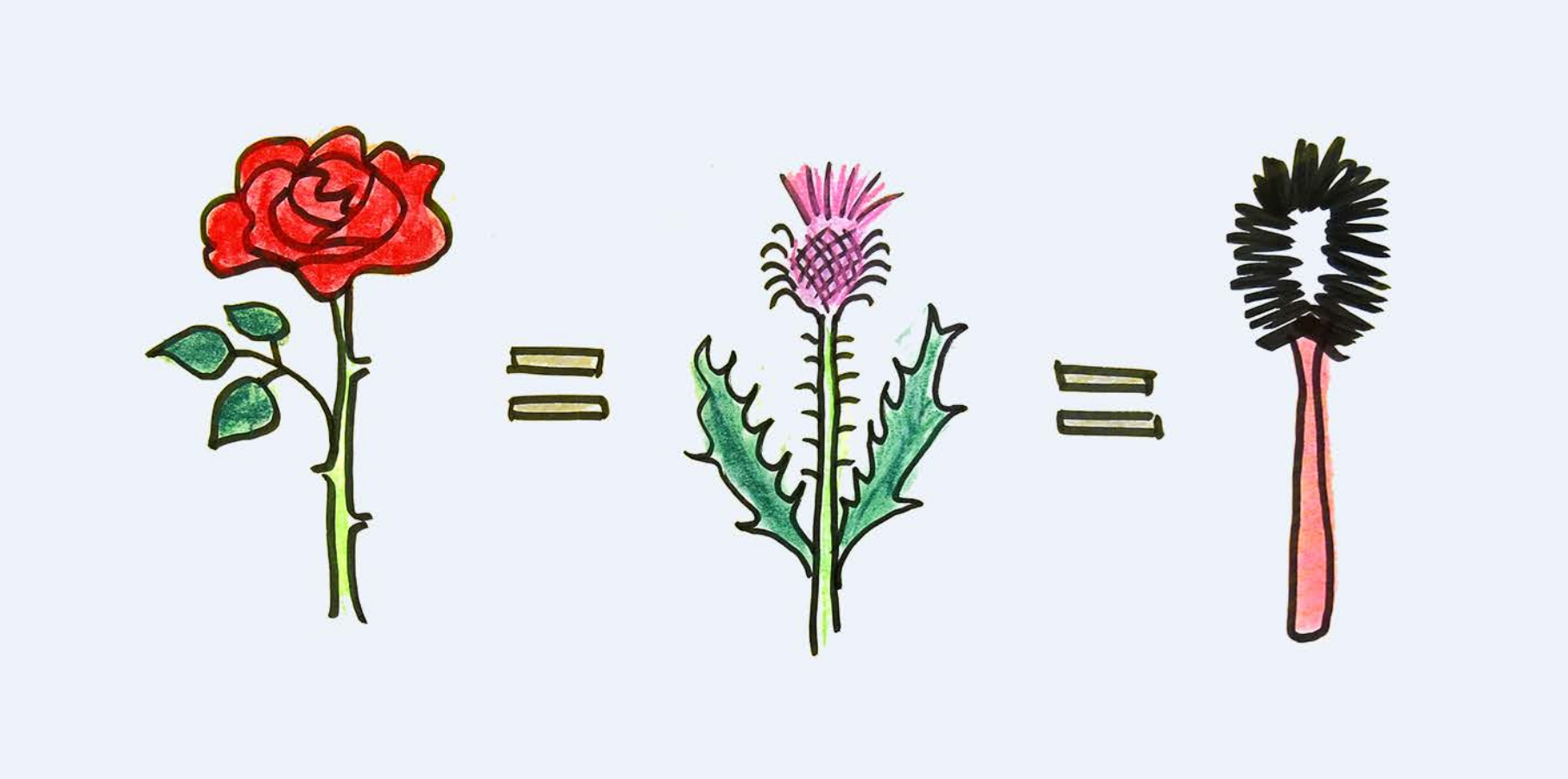
Co-Location is a Reality of Today's Development Environments



Source: 9th Annual Survey on State of Agile, Version One

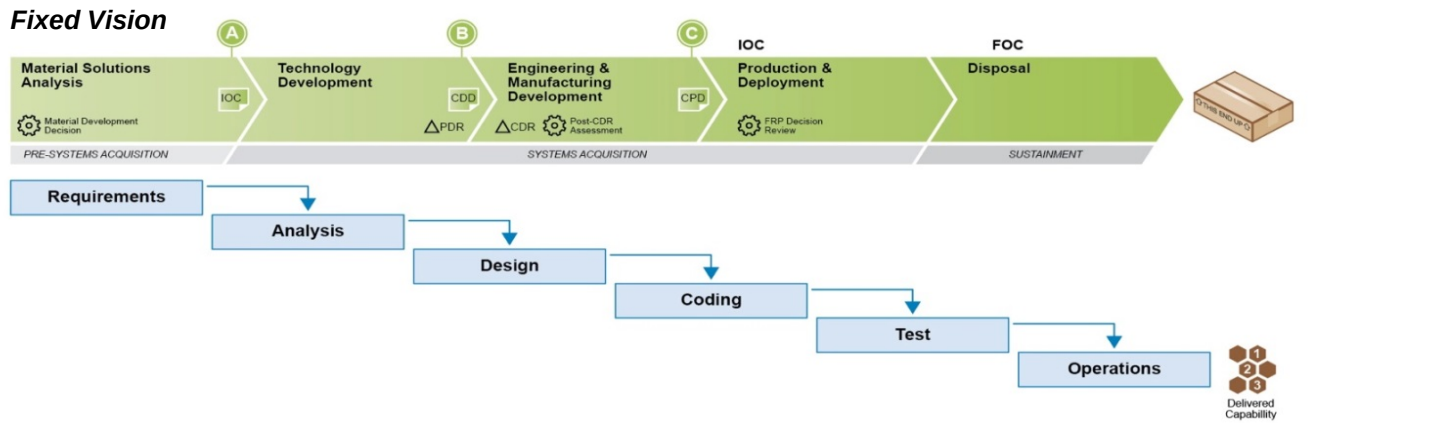
© 2015 Carnegie Mellon University

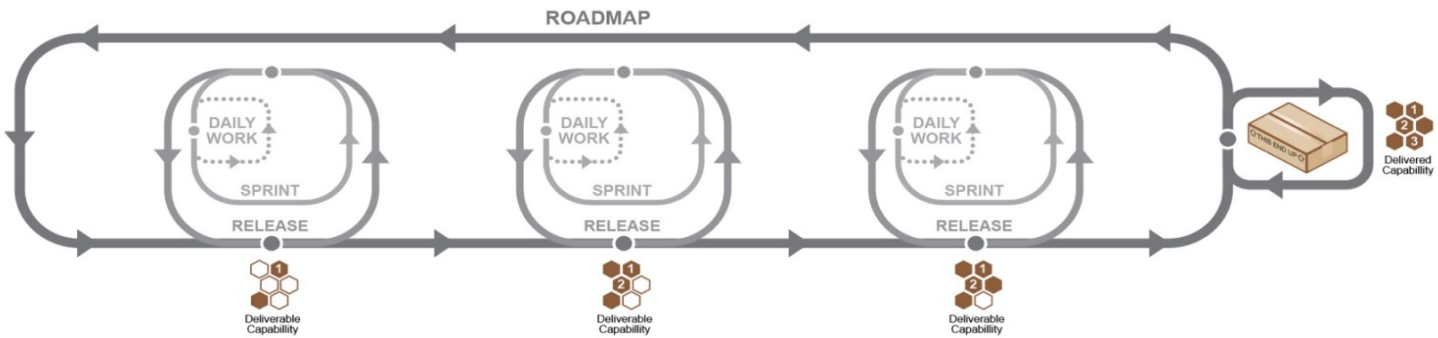
Myth: Agile is Just Spiral Renamed (or Incremental or Iterative)



© 2015 Carnegie Mellon University

Agile is Both Incremental AND Iterative

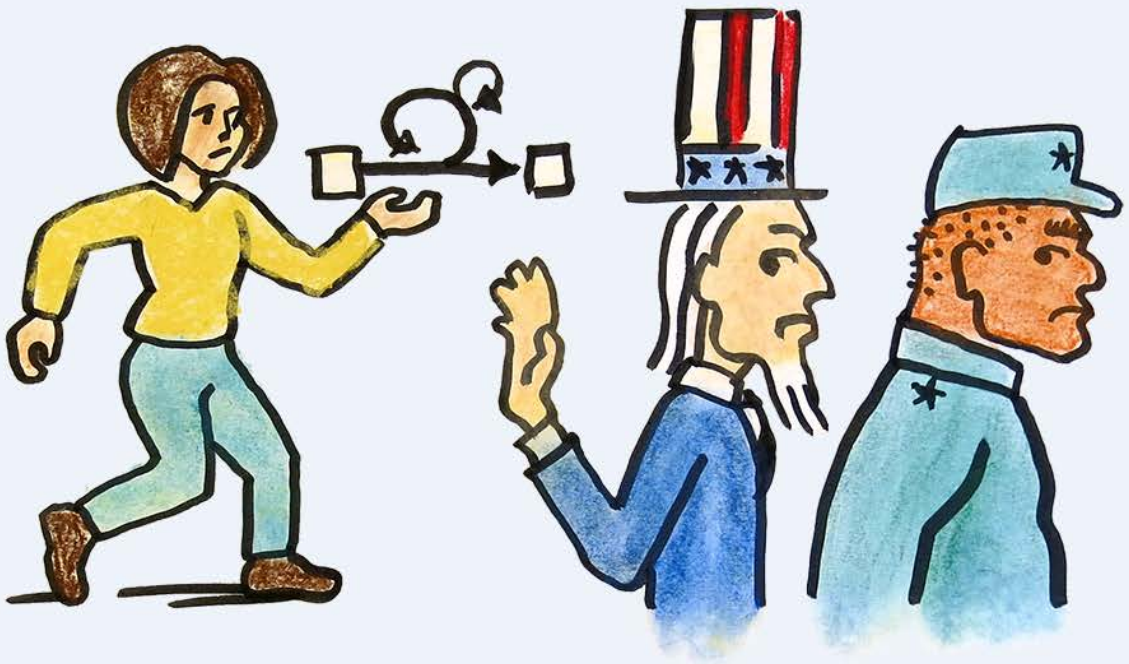




Source: Palmquist, Steven; Lapham, Mary Ann; Garcia-Miller, Suzanne; Chick, Timothy; & Ozkaya, Ipek. *Parallel Worlds: Agile and Waterfall Differences and Similarities* (CMU/SEI-2013-TN-021). Software Engineering Institute, Carnegie Mellon University, 2013.

© 2015 Carnegie Mellon University

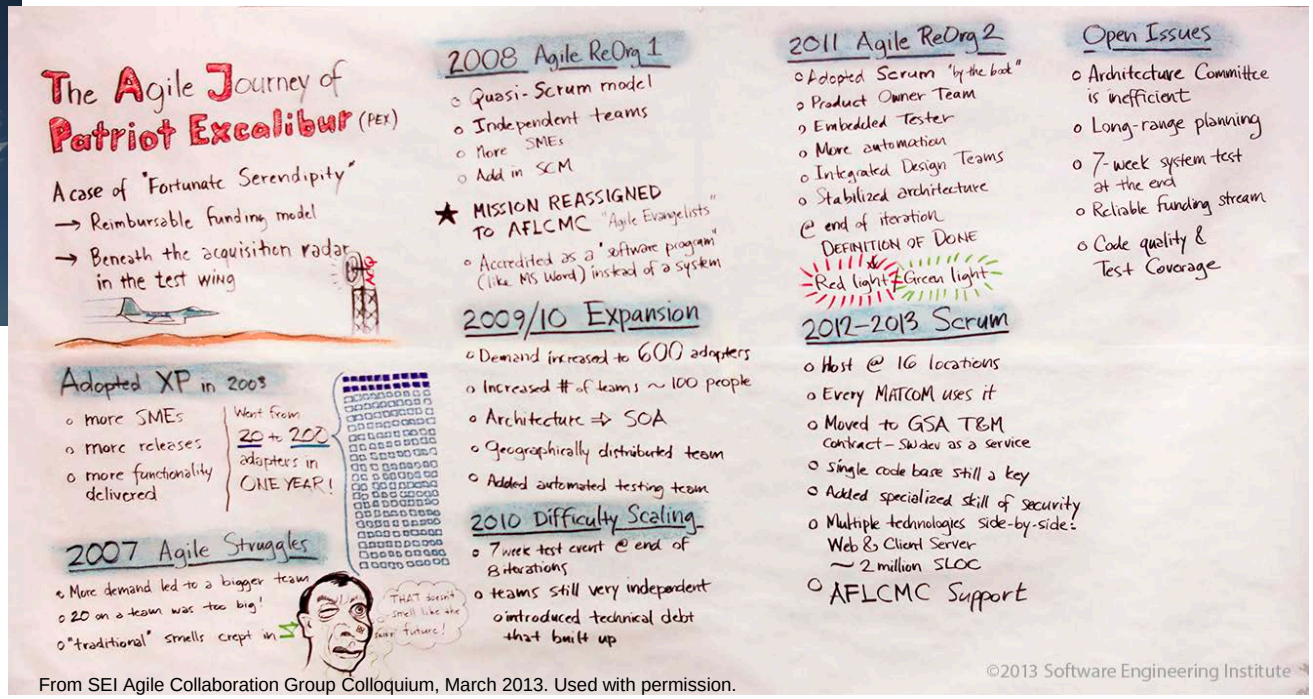
Myth: Agile Won't Work in DoD or Government Environments



© 2015 Carnegie Mellon University

Patriot Excalibur was One of the First, Going Strong since 2003!

It's a Journey…Patriot Excalibur switched to Agile methods in 2003 and successfully continues today



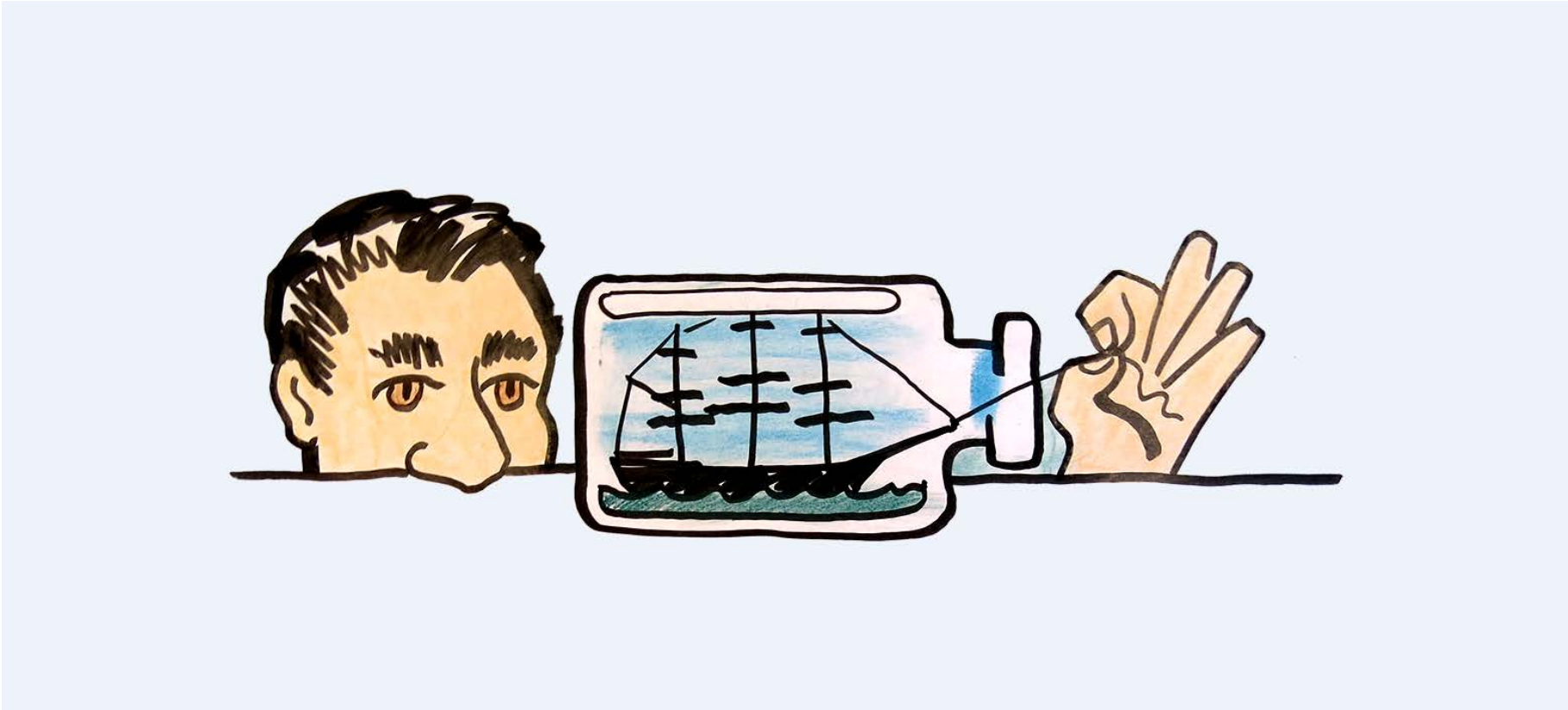
From SEI Agile Collaboration Group Colloquium, March 2013. Used with permission.

http://www.crosstalkonline.org/storage/issue-archives/2004/200407/200407-Fortier-Lozancich.pdf,

http://www.dtic.mil/ndia/2011agile/NDIAAgileProcessinDoD.pdf

© 2015 Carnegie Mellon University

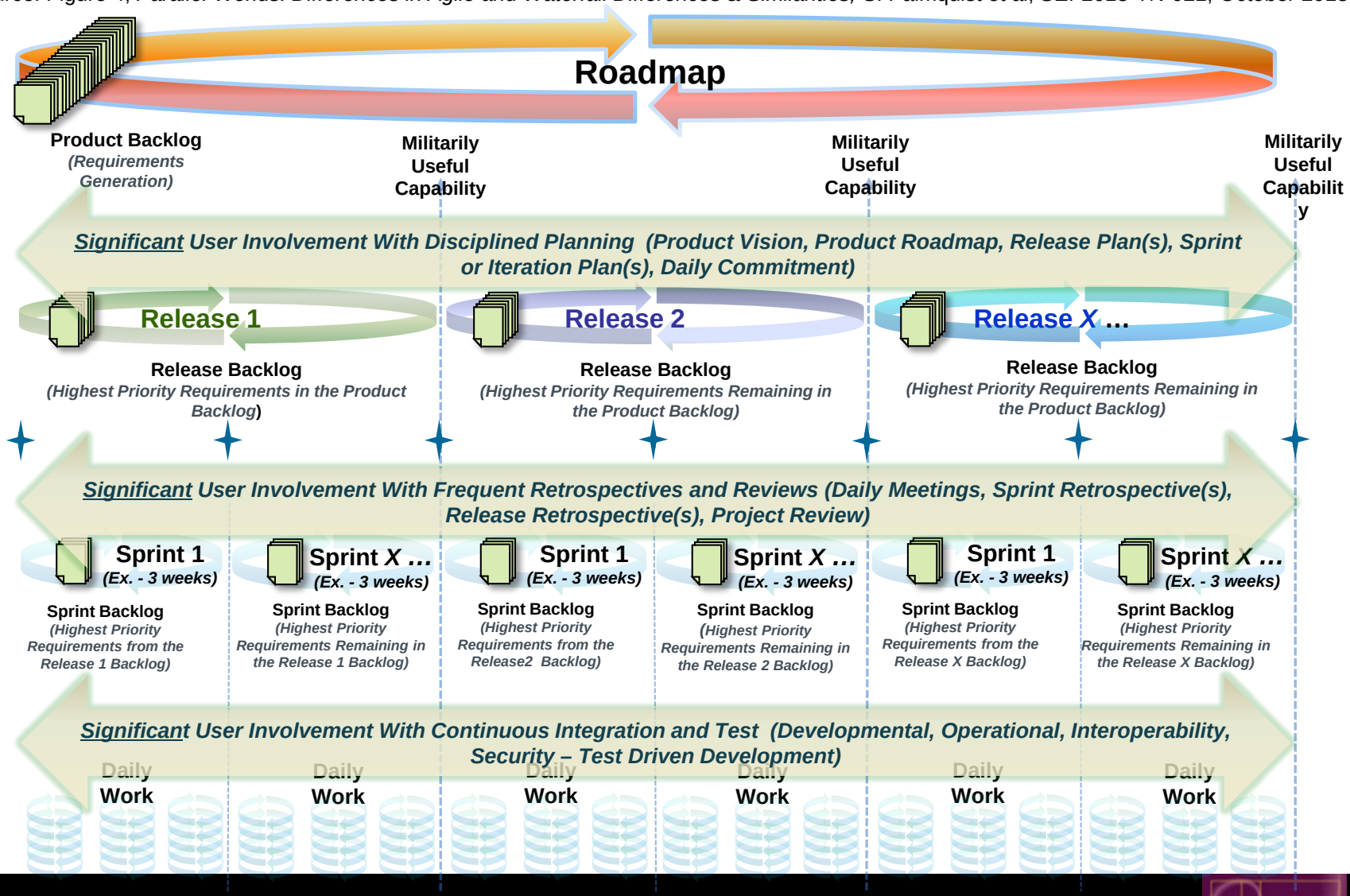
Myth: Agile Only Works for Small Projects….



© 2015 Carnegie Mellon University

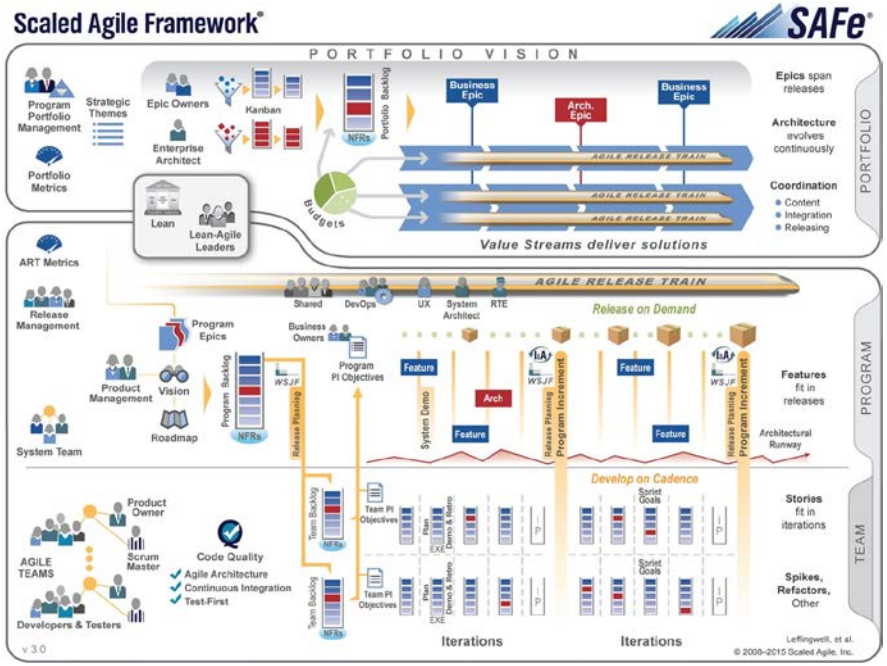
One View of a Large Scale Agile Development Model

Source: Figure 4, *Parallel Worlds: Differences in Agile and Waterfall Differences & Similarities,* S. Palmquist et al, SEI-2013-TN-021, October 2013.



© 2015 Carnegie Mellon University

Multiple Commercial Scaling Frameworks to Choose From

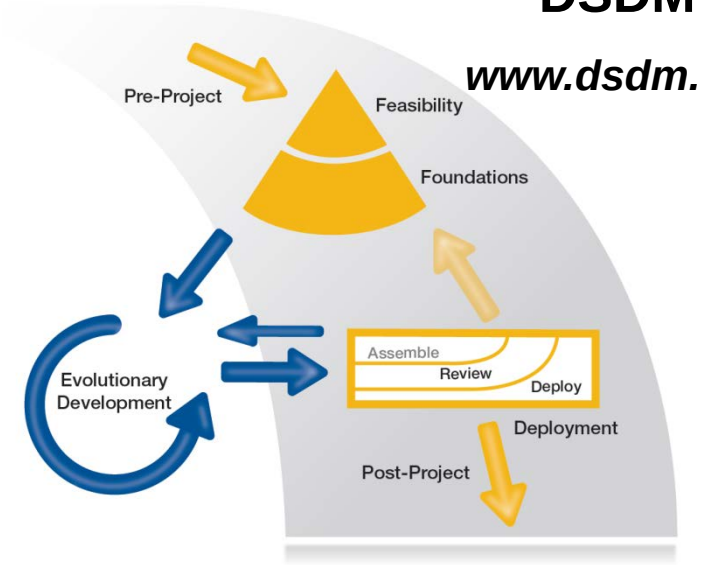


SAFe

www.scaledagileframework.com

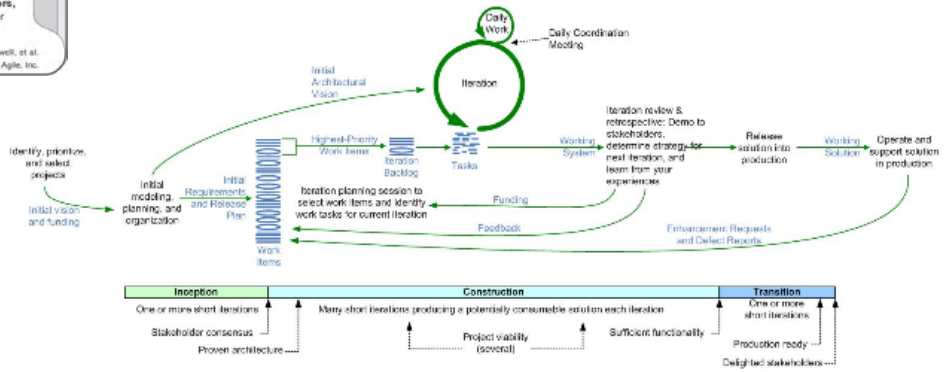
DSDM

www.dsdm.org



DAD

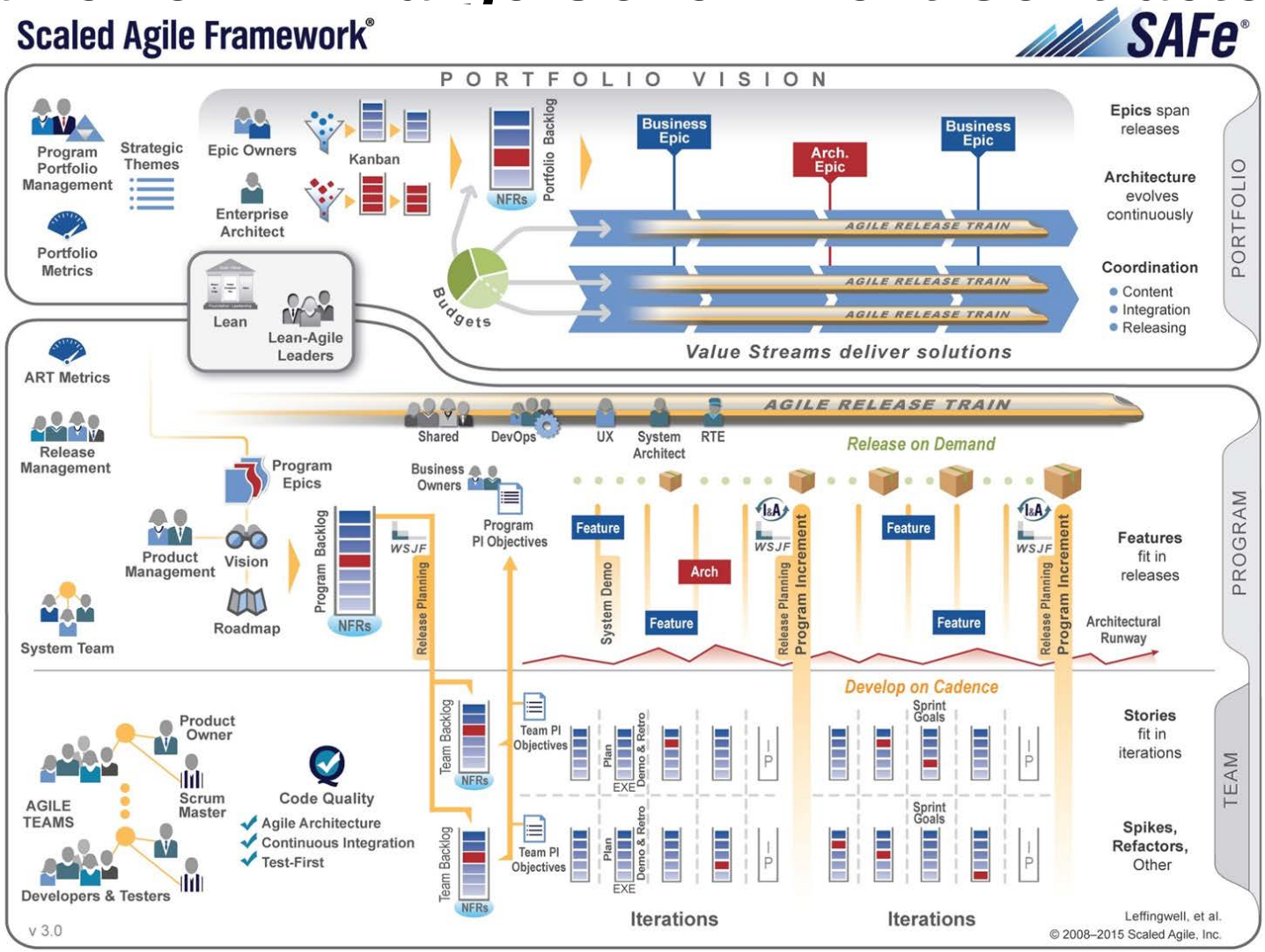
https://www.ibm.com/developerworks/community/blogs/ambler/entry/disciplined_agile_delivery_dad_lifecycle14?lang=en



And a couple of others

© 2015 Carnegie Mellon University

To Date, SAFe is the Most Frequently Seen Framework in Large Government Contractors



www.scaledagileframework.com

© 2015 Carnegie Mellon University

Myth: You Can't Use Earned Value Management with Agile Software Development

- **"Agile EVM" is successfully used in multiple environments, including DoD programs.[1]**
- **Lots of legacy data using non-Agile approaches makes it challenging to apply traditional measures, including EVM, in programs transitioning to Agile methods**

[1]Start with "Agile EVM in Scrum Projects" from AGILE 2006 to get started learning about Agile & EVM.
http://www.computer.org/csdl/proceedings/agile/2006/2562/00/25620007-abs.html

© 2015 Carnegie Mellon University

Summary-1

The key to successful Agile implementation is understanding how you will instantiate the Agile manifesto and principles.

The Agile principles have implications for the characteristics of the lifecycle that can be used

- But there's still more than one valid way of implementing the principles

© 2015 Carnegie Mellon University

Summary-2

The family of Agile methods has grown since 2000 to incorporate methods that address team, project, and enterprise levels of scaling

- It is likely there will never be a “single” Agile method
- SAFe-Scaled Agile Framework, is the most frequently adopted scaling framework in the DoD so far

Hybrids of multiple methods are common practice in both industry and government

- One reason it’s so difficult to say a program is “Agile” or not

© 2015 Carnegie Mellon University

Summary-3

Scrum focuses on the team management aspects of Agile software development.

As the most commonly practiced Agile methodology, it is the one that most practitioners are familiar with.

Many of the scaling approaches, including SAFe, leverage Scrum practices as the team component of their methodology.

© 2015 Carnegie Mellon University

Backup Slides

© 2015 Carnegie Mellon University

Is Incremental Development Another Term for Agile? (maybe, maybe not….)

Traditional Incremental Delivery	Agile Methods
Developer-Acquirer relationship at arm's length	Develop-Acquirer-End User collaboration
Hierarchical, command-and-control based teams	Collocated teams or strong communication mechanisms when teams are distributed
Leader as keeper of the vision and primary source of authority to act	Facilitative leadership and leader as champion and team advocate
Traditional, representational documents used by PMO to oversee the progress of the developer	"Just enough" documentation, highly dependent on product context
Lifecycle model with separate teams, particularly for development and test; some IPTs to involve multiple functions	Cross-functional teams including all roles across the lifecycle throughout the lifespan of the project

http://www.sei.cmu.edu/library/abstracts/reports/11tn002.cfm?DCSext.abstractsource=SearchResults

© 2015 Carnegie Mellon University

Extreme Programming (XP) – Commonly Used Technical Practices

User stories, used in several Agile methods, derive from Extreme Programming

Technical practices from XP commonly incorporated into other Agile methods:

- Continuous integration
- Daily build/automated regression test

Pair programming, another XP technical practice, is not used as often as the above outside of XP environments

© 2015 Carnegie Mellon University

Crystal

From A. Cockburn's description of Crystal:

"Crystal is a family of human-powered, adaptive, ultralight, "stretch-to-fit" software development methodologies.

- "Human-powered" means that the focus is on achieving project success through enhancing the work of the people involved (other methodologies might be process-centric, or architecture-centric, or tool-centric, but Crystal is people-centric).
- "Ultralight" means that for whatever the project size and priorities, a Crystal-family methodology for the project will work to reduce the paperwork, overhead and bureaucracy to the least that is practical for the parameters of that project.
- "Stretch-to-fit" means that you start with something just smaller than you think you need, and grow it just enough to get it the right size for you (stretching is easier, safer and more efficient than cutting away).

Crystal is non-jealous, meaning that a Crystal methodology permits substitution of similar elements from other methodologies."

Source: http://alistair.cockburn.us/Crystal+methodologies

© 2015 Carnegie Mellon University

Test Driven Development (TDD)

Two common flavors of TDD:

- Within an iteration, at unit/component level
- Acceptance TDD—across the lifecycle

Unit-level TDD

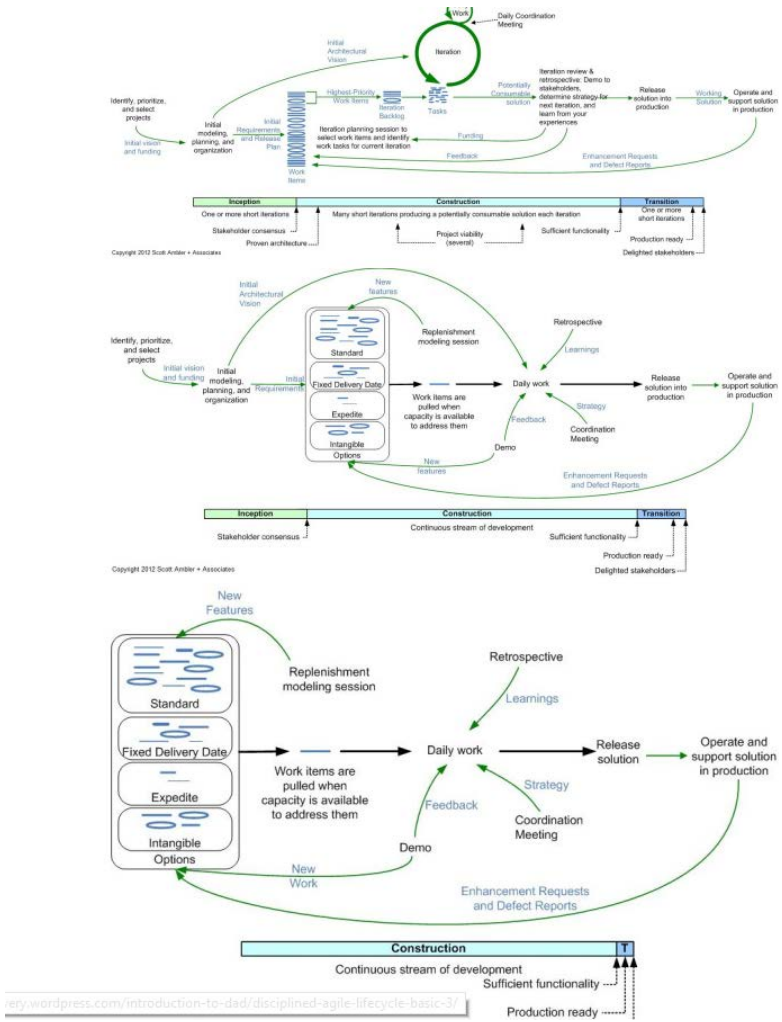
- For each requirement/user story, a test is written BEFORE coding starts on that element
- The minimum amount of code needed to pass the test is written and integrated into the code base

Acceptance TDD

- Expands the role of the product backlog to include the acceptance tests that will demonstrate satisfaction of the requirements
- Usually user story-based
- Cross-functional teams collaborate to build the acceptance criteria for the stories/requirements

© 2015 Carnegie Mellon University

Agile at Scale—Disciplined Agile Delivery



Leverages concepts from Rational Unified Process and designed to easily align with projects using Rational Unified Process

Risk+value-driven lifecycle

Enterprise Aware

Focused at project level

© 2015 Carnegie Mellon University

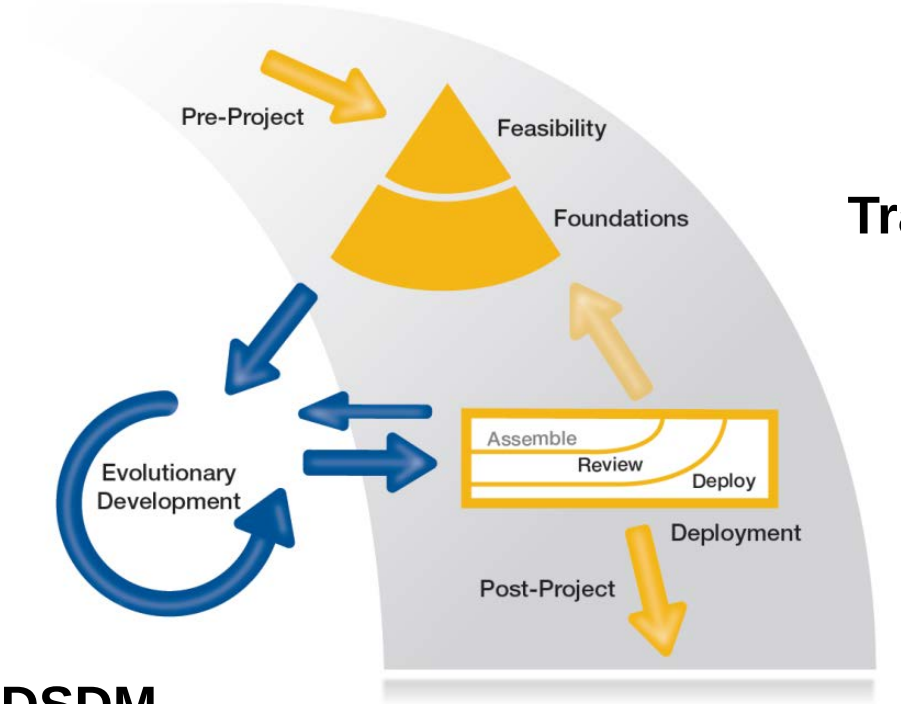
Agile at Scale-DSDM

Eight Principles for DSDM

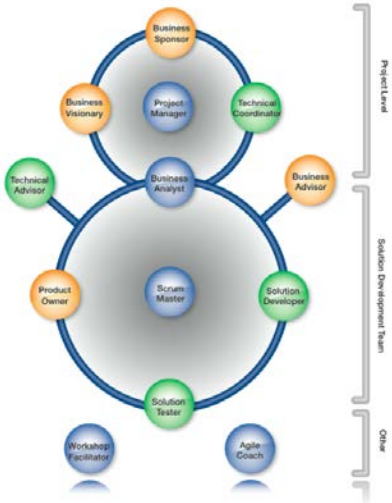
The eight Principles are:

1. Focus on the business need
2. Deliver on time
3. Collaborate
4. Never compromise quality
5. Build incrementally from firm foundations
6. Develop iteratively
7. Communicate continuously and clearly
8. Demonstrate control

DSDM System Lifecycle



DSDM Roles Map Well to Traditional Project Management



DSDM

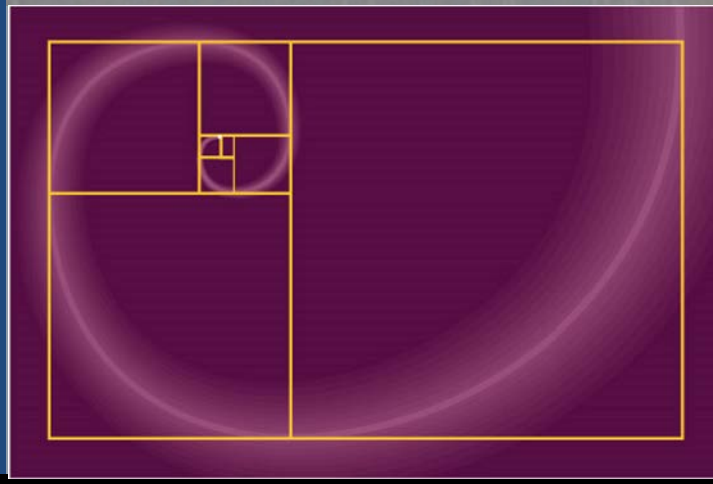
www.dsdm.org

© 2015 Carnegie Mellon University

Agile in Government: Practical Considerations

CTSD Agile Adoption in Regulated Settings Team
Software Engineering Institute
July 2015

Module 3— Government Agile Interactions: Role in Agile Programs; Dealing with Requirements



© 2015 Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

This material has been approved for public release and unlimited distribution except as restricted below.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study. Except for the U.S. government purposes described below, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose this material are restricted by the Rights in Technical Data-Noncommercial Items clauses (DFAR 252-227.7013 and DFAR 252-227.7013 Alternate I) contained in the above identified contract. Any reproduction of this material or portions thereof marked with this legend must also reproduce the disclaimers contained on this slide.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

This material was prepared for the exclusive use of NRO Agile in Government Workshop and may not be used for any other purpose without the written consent of permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0002617

Contents

Module 1: Introduction

Module 2: Basic Agile Concepts

- Agile Introduction
- Agile Tenets and Principles
- Differences Between Traditional and Agile
- Common Agile Concepts and Practices
- Agile Mythbusting

Module 3: New Role for PMO

- ***Why Agile***
- ***The Product Owner Role***

© 2015 Carnegie Mellon University

Contents (continued)

Module 4: Agile Program Oversight

- System Engineering
- Integration and Test
- Technical Reviews
- Measurement

Module 5: Agile in the Larger Eco-System

- Documentation
- Contracting
- Regulatory Oversight

Module 6: Enabling an Agile Acquisition Culture

- Enabling and Transitioning to an Agile Culture
- Suggestions for Implementing Agile in Government

Module 7: Summary

© 2015 Carnegie Mellon University

Learning Objectives:
Module 3: Government Agile Interactions 1

After completing this module, participants will be able to:

- Explain the primary responsibilities of the Product Owner role in Scrum
- Explain why someone in the government program office might be an appropriate Product Owner in an Agile program
- Describe how interacting with a developer contractor should be different in an Agile program
- Describe the main challenge in dealing with requirements in an Agile government setting with traditional expectations

© 2015 Carnegie Mellon University

Historical Reasons SW Acquisitions Fail

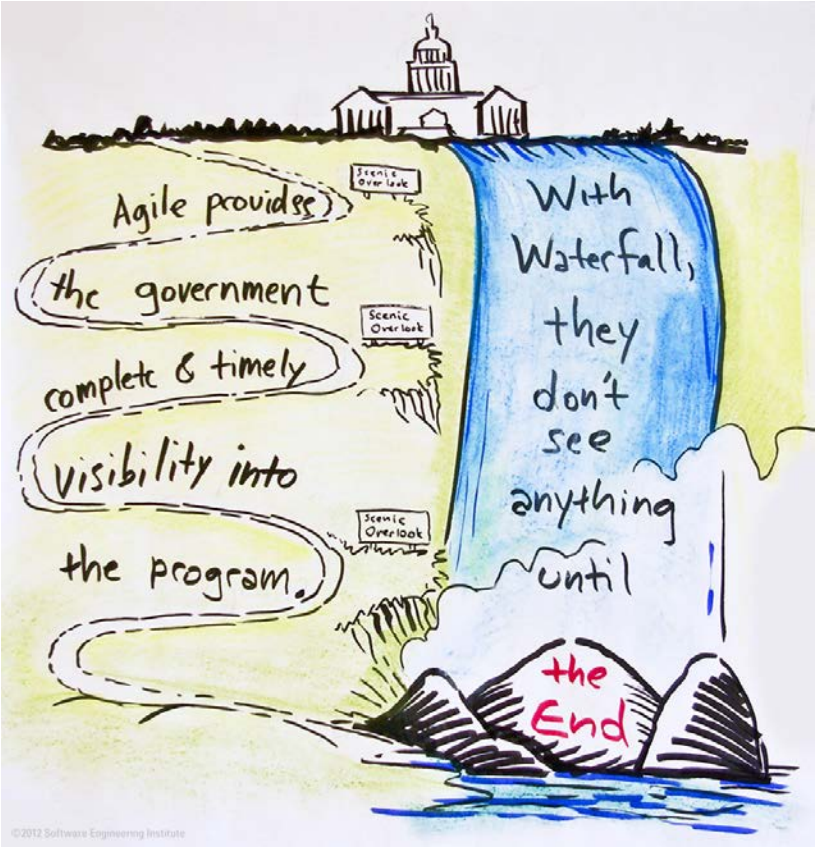
Top 10 Reasons	Your Perspective
10. Technology used is new to the organization	
9. Software issues are considered too late in the system-development process	
8. Inadequate planning and estimating; long duration programs	
7. Size matters—large projects get into trouble more frequently than smaller ones	
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	

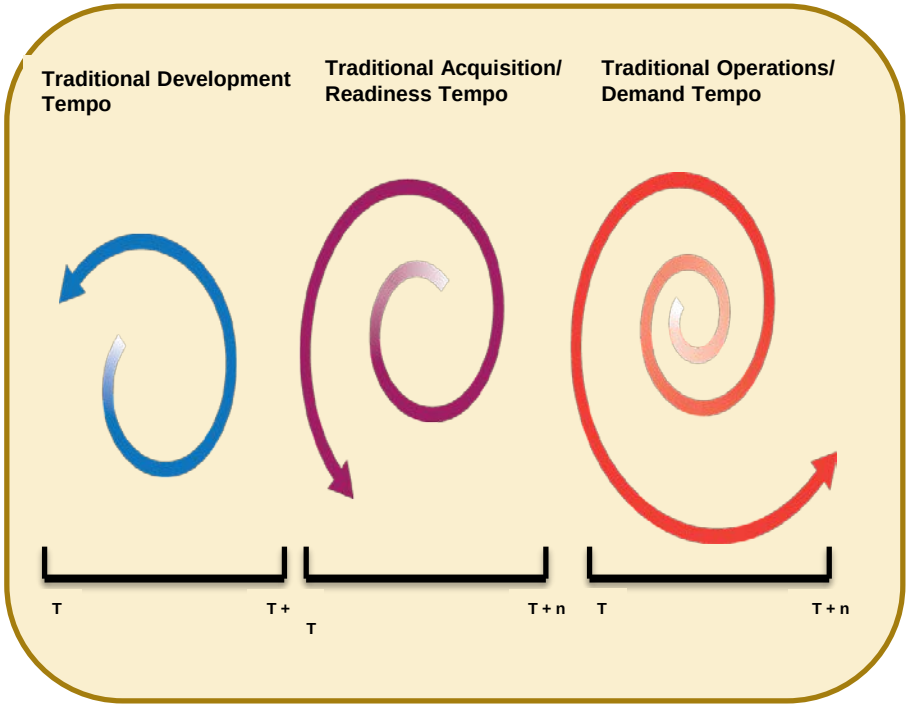
Source: Nielsen, P. Congressional Testimony July 9, 2009.

Provide *your* rank order of these failure modes. Which of these might be reduced or eliminated by Agile approaches?

© 2015 Carnegie Mellon University

Increasing Delivery Tempo to the Field



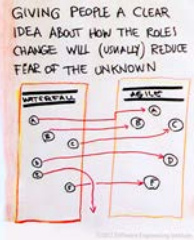


© 2015 Carnegie Mellon University

The Product Owner Role

© 2015 Carnegie Mellon University

What does a Product Owner *Do*?



Write epics, user, and technical stories (requirements) that start the conversation with developers

- Work with end users, testers, and other stakeholders to establish Acceptance Criteria for the stories and epics

Prioritize and rank order epics and stories

Participate in release planning events that map the stories into implementation packages

Establish Release Goals for each release that are objective and measurable

Establish Iteration (or Sprint) goals that are objective and measurable

Proactively monitor developer progress via physical or electronic Information Radiators

Evaluate the satisfaction of Sprint and Release goals through attendance at Sprint and Release Reviews

“Accepts” the Release for further integration and system testing or delivery into a sandbox

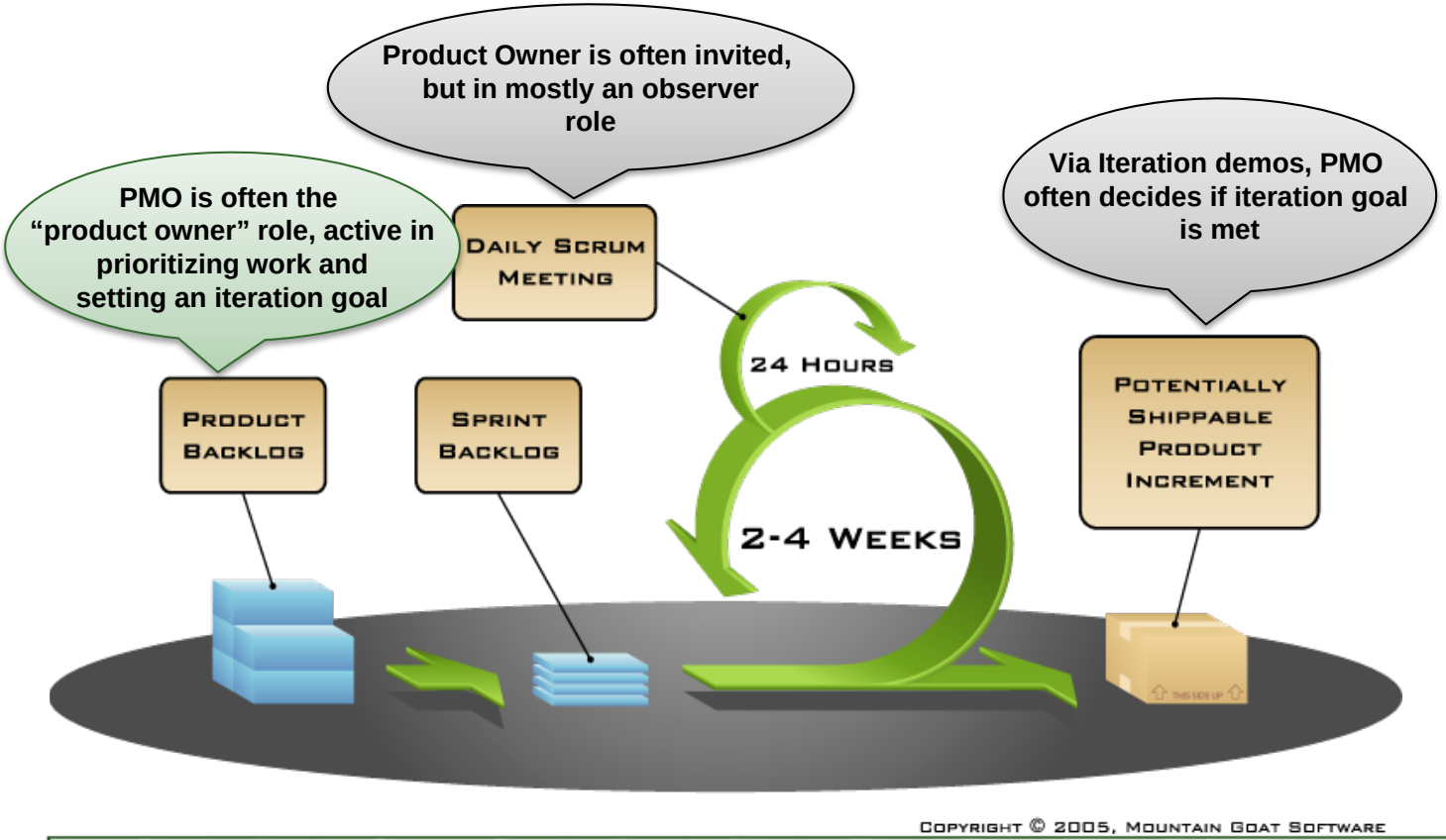
Answers (as often as daily) questions from developers to clarify their understanding of a story or its acceptance criteria

Advocates for the product to business or operational management who are investing in the product

Oh yeah, don’t forget to BREATHE!!!!

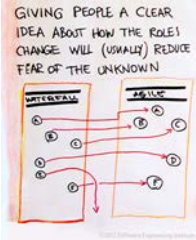
© 2015 Carnegie Mellon University

Potential Product Owner interactions in Scrum



Battle rhythm philosophy for Agile is: more frequent interaction, but on smaller batches of content

© 2015 Carnegie Mellon University



The Big Challenges for Government Program Office as Product Owner

TIME!

- Moving from less frequent, higher content interactions to more frequent, lower content level interactions

Many program office staff support more than one program

- Engaging in an Agile program can upset the rhythm of interactions across others

Navigating the “constructive change” landscape when interactions are frequent can be challenging

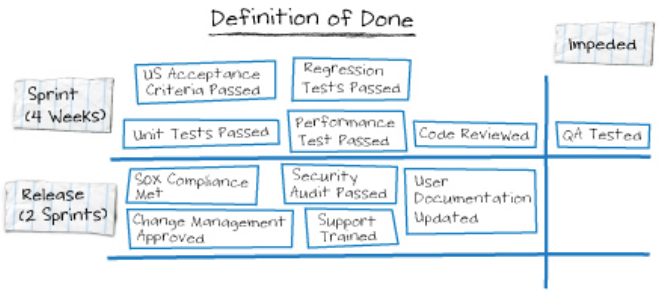
Representing all stakeholders as a single voice at all times (in real time)

On larger programs, having enough staff to interact with all the teams in play is challenging

The payoff: earlier course corrections lead to products with less rework

© 2015 Carnegie Mellon University

Definition of Done (DoD) is a Key Concept for a Product Owner



What is the Definition of Done (DoD)?

Definition of Done is an explicit declaration of the completion criteria for some aspect of an Agile lifecycle. DoD can be applied to an individual artifact (e.g., a user story), a sprint (as a companion to the Sprint Goal), or a release.

Why do we need a specific Definition of Done?

One of the ways that Agile methods achieve the speed they are known for is that developers have confidence that when they are "done" with some task or artifact, it's safe to move on to the next one. The explicit Definition of Done is a key contributor to enabling this confidence.

When is the Definition of Done established?

The DoD is established prior to the work being done (for a User Story, DoD is established before the Product Backlog items are estimated; for a Release DoD is established prior to the completion of Release Planning, etc.)

When do you determine that the Definition of Done has been met?

Verifying that the DoD has been met depends on what DoD is being applied to. If DoD is being applied to an artifact, it is verified before the item is marked as "Done" in whatever Information Radiator the team is using to communicate status.

If DoD is being applied to an Agile lifecycle phase, like the Sprint Goal, it would be determined during the review meeting that occurs at the end of the sprint or release.

Other Notes on Definition of Done:

There is no "universal" Definition of Done. However, a good definition of done should follow SMART rules: Specific, Measurable, Actionable, Relevant, and Timely. Usually the time is predetermined (e.g. your sprint timebox)

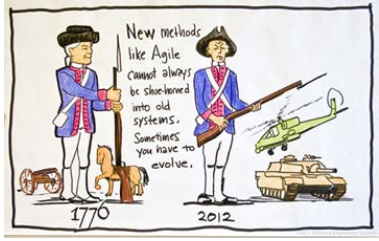
Make the definition public and review it in between sprints, releases, etc. —reviewing the DoD as part of the Retrospective is a frequent approach, or including a review during the Sprint Planning Meeting.

© 2015 Carnegie Mellon University

Requirements Challenges in Agile Settings

© 2015 Carnegie Mellon University

Requirements Challenges: Agile in DoD Settings



DoD Guidance

- DoD 5000.02 helps, but stops short of encouraging Agile

Translation of Requirements Progress Measures

- Progress=document completion is problematic in Agile settings

Risk Averse Culture

- “build to the requirements”-safe, but doesn’t account for inevitable learning

Work Breakdown Structure

- Especially HW-centric WBS can result in software requirements at too low a level of detail

Effect of Requirements Changes on Contracts

- Assumption that change is an exception vs. change is expected and planned for

Perception that Reduced Documentation is Cause for Concern

- Letting go of “documents=progress” is difficult

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.

© 2015 Carnegie Mellon University

Product Backlog: Scrum's Way of Organizing and Prioritizing Requirements

This is the product backlog

- The requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the Product Owner
- Reprioritized at the start of each release and each sprint

© 2015 Carnegie Mellon University

Stories—an Agile Way of Including the "Why" of a Requirement

USER Stories express things that an operational user would find valuable

- Example User Story template: "*As a "role," I want to "function" so I can "operational goal"*

TECHNICAL Stories express quality attributes of a system, subsystem or component that may not be directly seen by the user but are essential to meeting mission goals

- Example Technical Story template: To meet "quality attribute," system/subsystem/component must "do function."

© 2015 Carnegie Mellon University

Multiple Levels of Abstraction for Stories Accommodated in Most Agile Scaling Frameworks

Typical hierarchy (from SAFe, in this case):

- Strategic theme – could be analog to ORD requirements
- Epic – could be analog to System Level 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

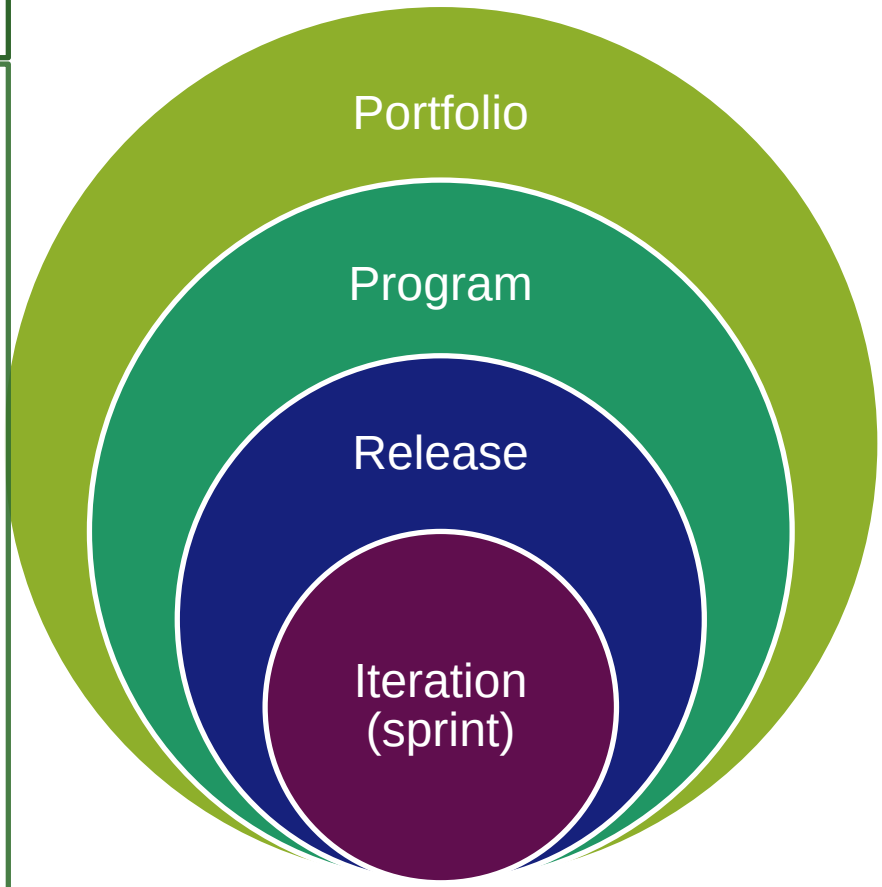
- One dependency is how the software part of the program interacts with systems engineering/other stakeholders
- Another criteria is how requirements change will be accommodated
  - Level at which technical baseline is established is crucial to having flexibility in requirements change

© 2015 Carnegie Mellon University

Addressing Requirements at Multiple Levels in Agile Settings

Issues in Expressing Requirements

- **Portfolio**: Conops level, trying to establish value stream
- **Program**: moving from "shall" statements to Business & Architecture Epics
- **Release**: Bundling Epics into meaningful Features that are executable in a few iterations; translating Epics into User & Technical 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

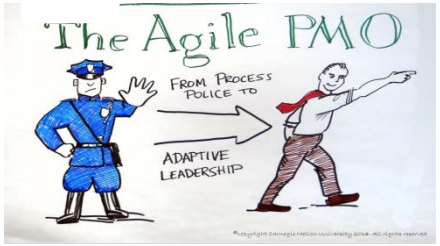


Issues in Governance Requirements

- **Portfolio**: Assuring that the value stream is representative of operations
- **Program**: assuring that user representatives are engaged and relevant
- **Release**: Assuring that Product Owners are actively engaged in refining and prioritizing stories and epics ahead of the development teams
- **Iteration**: Assuring that Product Owners appropriately represent user needs and management goals when interacting with development teams

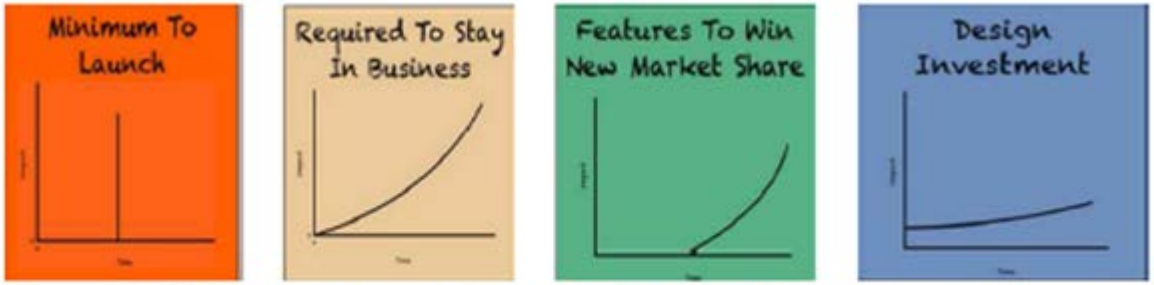
© 2015 Carnegie Mellon University

Product Backlog Feeds Releases



List of high-level requirements

- Prioritized by Product Owner
  - "Value points" are not story points, but are something the product owner can do to help developers understand operational priorities
  - Instead of relative estimation of complexity, product owners estimate relative **value** of the backlog items



http://agileconsulting.blogspot.com/2011/03/using-cost-of-delay-functions-to.html

© 2015 Carnegie Mellon University

Frequent Question About Requirements and Sprints

Probably the most frequently asked question about overseeing Agile that we're asked is some variation of:

- ***How do we know if deferral of requirements from one iteration/sprint to another is "OK" vs. a sign of a problem?***

Some of the things to look for to answer that question:

- How early is it in the development?
  - Most teams take at least three, as many as six, iterations to get their estimation heuristics consistent enough to achieve their estimates
- Especially early, is there an identifiable "new" dependency that has been discovered that makes deferral of stories appropriate?
- Does the developer recognize they are incurring "technical debt" by deferring stories, and have a strategy for addressing?
- Are the deferrals a result of a larger amount of rework due to defects in previously delivered code?
  - Often occurs when not enough automated testing is used for build integration

© 2015 Carnegie Mellon University

User Stories and Technical Stories - 1

User stories represent the functional requirements from the perspective of an actor

- Actors can be human (e.g., User)
- A system can also be an actor

As an [actor], I can [function] so that [reason]
Acceptance criteria:
1)
2)
⋮
n)

© 2015 Carnegie Mellon University

User Stories and Technical Stories-2

There is a class of user stories—*technical user stories*—that represent non-functional support of the system (e.g., security, database schema, networking protocols), or towards resolving technical debt and refactoring.

Five broad categories[1]:

- Product infrastructure
- Team infrastructure
- Refactoring
- Bug fixing
- Spikes

1. This discussion about technical stories is taken from Robert Galen, “Technical User Stories – What, When, and How?, 16 November 2013

© 2015 Carnegie Mellon University

Recognizing "Good" User Stories

How to know Good User Stories?

- Story Card Format
  - As a _____,
  - I want to be able to _____
  - So that _____ .
- Acceptance Criteria
  - I will know this is done when _____ .
- Key Points to Writing Stories
  - Keep stories short & business language focused
  - Seek a level of granularity that can be completed in a few days
  - Keep stories mutually independent
  - Do not include implementation details
  - Do not stop talking –

Independent

Negotiable

Valuable

Estimable

Small

Testable

© 2015 Carnegie Mellon University

User Stories and Technical Stories - Example

Product infrastructure stories support functional stories. This could include new/modified infrastructure, or functionally-driven refactoring opportunities

Example:

The code base needs to operate on an x86 blade infrastructure so we can reduce maintenance cost and improve our ability to satisfy new functional and performance requirements.

Acceptance criteria:

- *Sourced from commodity hardware*
- *Life cycle maintenance cost is lower than current system on an annual basis*

© 2015 Carnegie Mellon University

What about Epics?

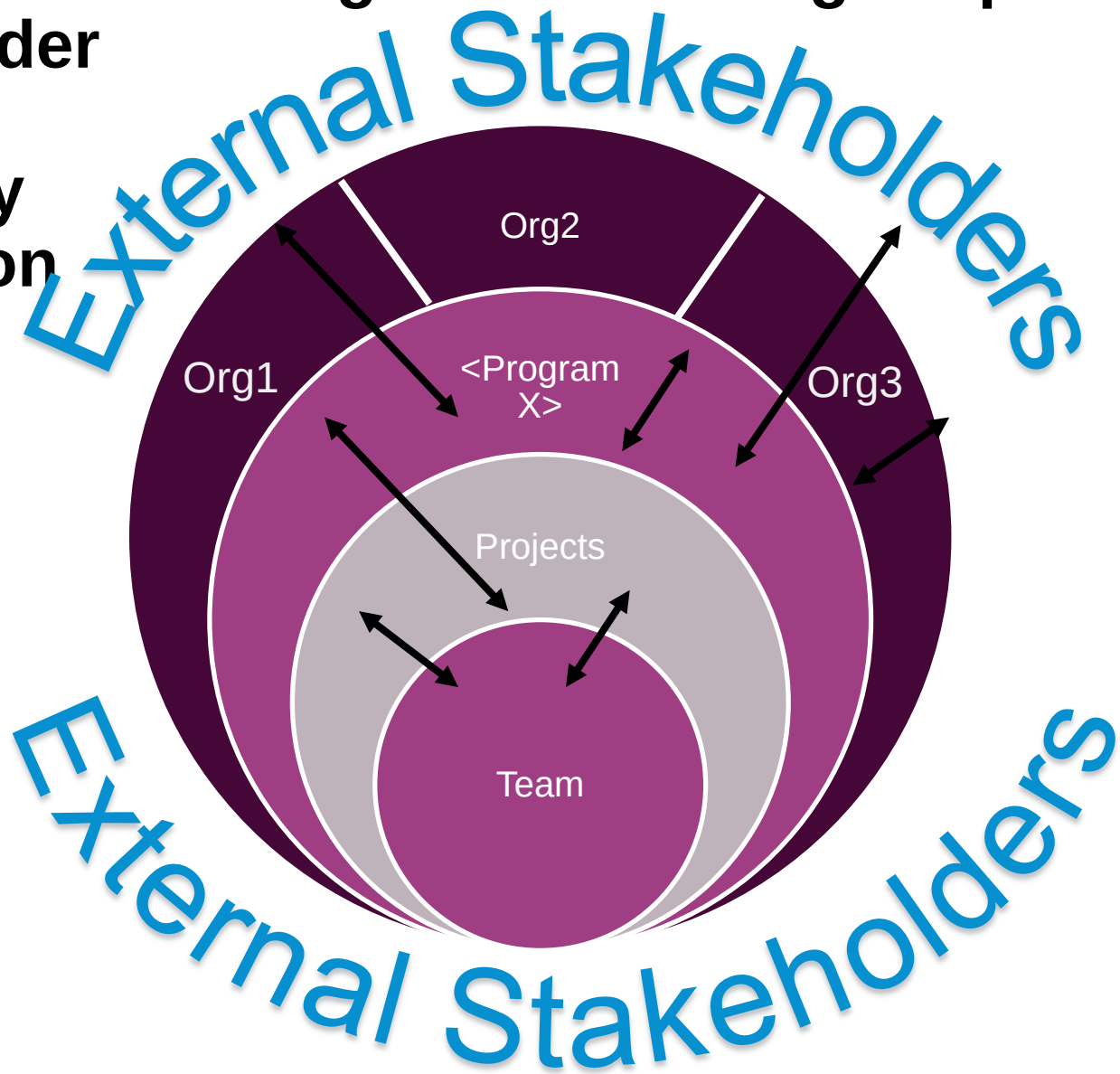
Themes that break down into product features and Stories (User, Technical)

Template from SAFe:

Forward-Looking Position Statement	
For	<customers>
who	<do something>
the	<solution>
is a	<something – the "how">
that	<provides this value>
Unlike	<competitor, current solution, or non-existing solution>
our solution	<does something better – the "why">
Scope	
Success Criteria:	▸ ▸
In Scope:	▸ ▸
Out of Scope:	▸ ▸
NFRs:	▸ ▸

© 2015 Carnegie Mellon University

A Common Challenge in Governing Requirements: Stakeholder Diversity and Many Interaction Modes



© 2015 Carnegie Mellon University

Summary

Agile methods embody a change in philosophy of interaction

- Program Office must plan for that difference to be successful

DoD 5000.02 is more supportive of Agile approaches

- But still contains constructs (like Technical Reviews) that are challenging if not planned for

The Product Owner role provides an opportunity for collaborative interaction with developers that can lead to better operational outcomes

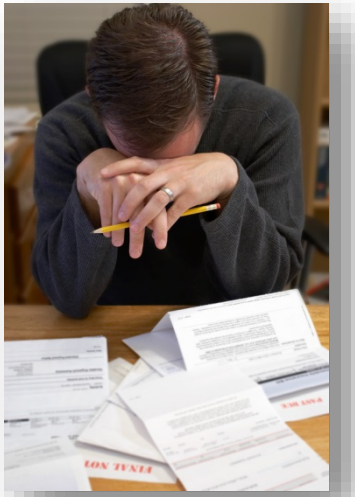
- But it's a different cadence of interaction than is typical in most traditional programs
- Constructive change parameters must be understood and respected

© 2015 Carnegie Mellon University

Discussion

What is your reaction to the possibility of taking on the Product Owner role?

- What excites you?
- What puts you off?
- What terrifies you?

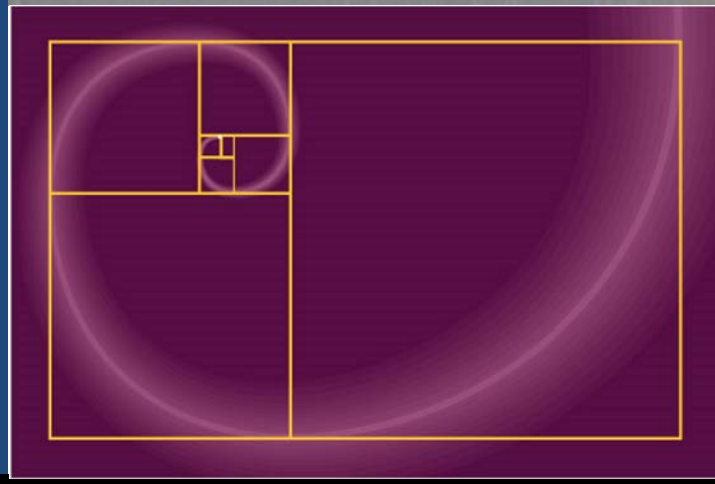


© 2015 Carnegie Mellon University

Agile in Government: Practical Considerations

CTSD Agile Adoption in Regulated Settings Team
Software Engineering Institute
July 2015

Module 4 (Systems Engineering, Technical Reviews, Measures)



© 2015 Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

This material has been approved for public release and unlimited distribution except as restricted below.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study. Except for the U.S. government purposes described below, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose this material are restricted by the Rights in Technical Data-Noncommercial Items clauses (DFAR 252-227.7013 and DFAR 252-227.7013 Alternate I) contained in the above identified contract. Any reproduction of this material or portions thereof marked with this legend must also reproduce the disclaimers contained on this slide.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

This material was prepared for the exclusive use of NRO Agile in Government Workshop and may not be used for any other purpose without the written consent of permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0002617

Contents

Module 1: Introduction

Module 2: Basic Agile Concepts

- Agile Introduction
- Agile Tenets and Principles
- Differences Between Traditional and Agile
- Common Agile Concepts and Practices
- Agile Mythbusting

Module 3: New Role for PMO

- Why Agile
- The Product Owner Role

© 2015 Carnegie Mellon University

Contents (continued)

Module 4: Agile Program Oversight

- ***System Engineering***
- ***Integration and Test***
- ***Technical Reviews***
- ***Measurement***

Module 5: Agile in the Larger Eco-System

- Documentation
- Contracting
- Regulatory Oversight

Module 6: Enabling an Agile Acquisition Culture

- Enabling and Transitioning to an Agile Culture
- Suggestions for Implementing Agile in Government

Module 7: Summary

© 2015 Carnegie Mellon University

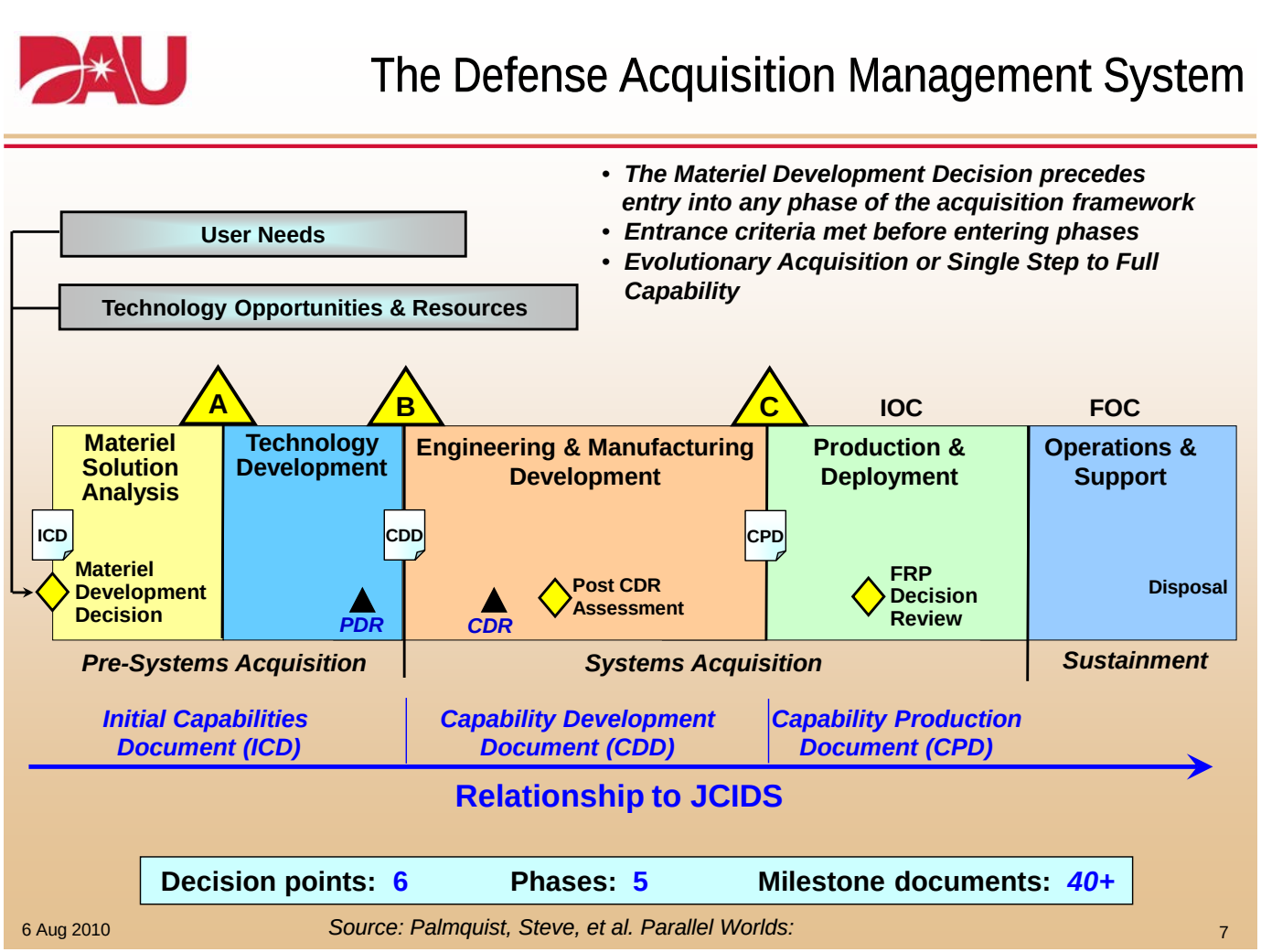
Learning Objectives:
Module 4: Agile Program Oversight

After this module, participants will be able to:

1. List three common modes of interaction between an Agile software team and a program's systems engineering function
2. List at least two challenges related to bringing test activities into an Agile development
3. List 3 options for Technical Reviews in a program using Agile methods
4. Discuss how Agile methods attempt to achieve INSIGHT into progress, not just OVERSIGHT
5. Recognize velocity, burn up/burn down, and cumulative flow diagram visualizations for measuring Agile program progress

© 2015 Carnegie Mellon University

Remember-Agile Methods are Used in Context of a Larger Program (More Often Than Not)



© 2015 Carnegie Mellon University

Agile and Systems Engineering

© 2015 Carnegie Mellon University

Systems Engineering is Important Player in Programs Adopting Agile -1

Key facets of systems engineering

- ***Product side***—transforming artifacts that communicate the intent of the system as understanding of the system evolves
  - Leverage incremental, iterative approach with heavy user involvement to increase speed of development of key requirement and design artifacts needed to implement different mission/system threads.
  - Incorporate acceptance-test-driven development into activities of systems engineering to increase connection between two sides of systems engineering "V"

© 2015 Carnegie Mellon University

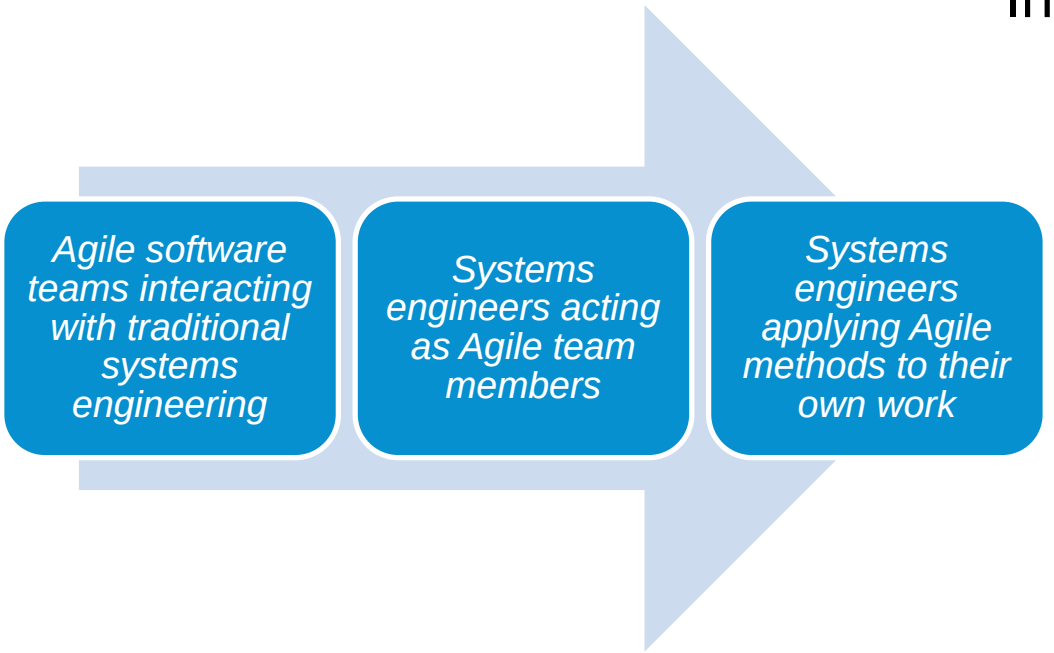
Systems Engineering is Important Player in Programs Adopting Agile -2

Key facets of systems engineering

- ***Service side***—communicating and coordinating important information about evolving knowledge of system among many stakeholders.
- Additionally, systems engineers have a strong conflict resolution role when technical and programmatic conflicts arise among stakeholders
  - When program scale requires a separate systems engineering function, their coordination, communication and conflict resolutions services could translate into product owner surrogate role, Scrum of Scrums facilitator role or other specialty roles that show up in Agile scaling approaches.

© 2015 Carnegie Mellon University

Three Approaches We Observed of Systems Engineering Interacting with or Being Part of Agile Teams



In reality it is a continuum:

- As software teams demonstrated/continued to demonstrate success, systems engineering teams and leaders got engaged with software processes
- Successful activities led, in at least one case, to application of Agile methods to part of systems engineering process

© 2015 Carnegie Mellon University

Suggestions for Program Office Interacting with Systems Engineering

Ascertain what mode of interaction systems engineering is planning in relation to Agile software development activities

- Plan software activities to account for the selected mode

In CDRL, include concept of incremental delivery of content

In scheduling, be aware of places where hardware emulators/simulators are needed by software development team to appropriately progress the software iteratively

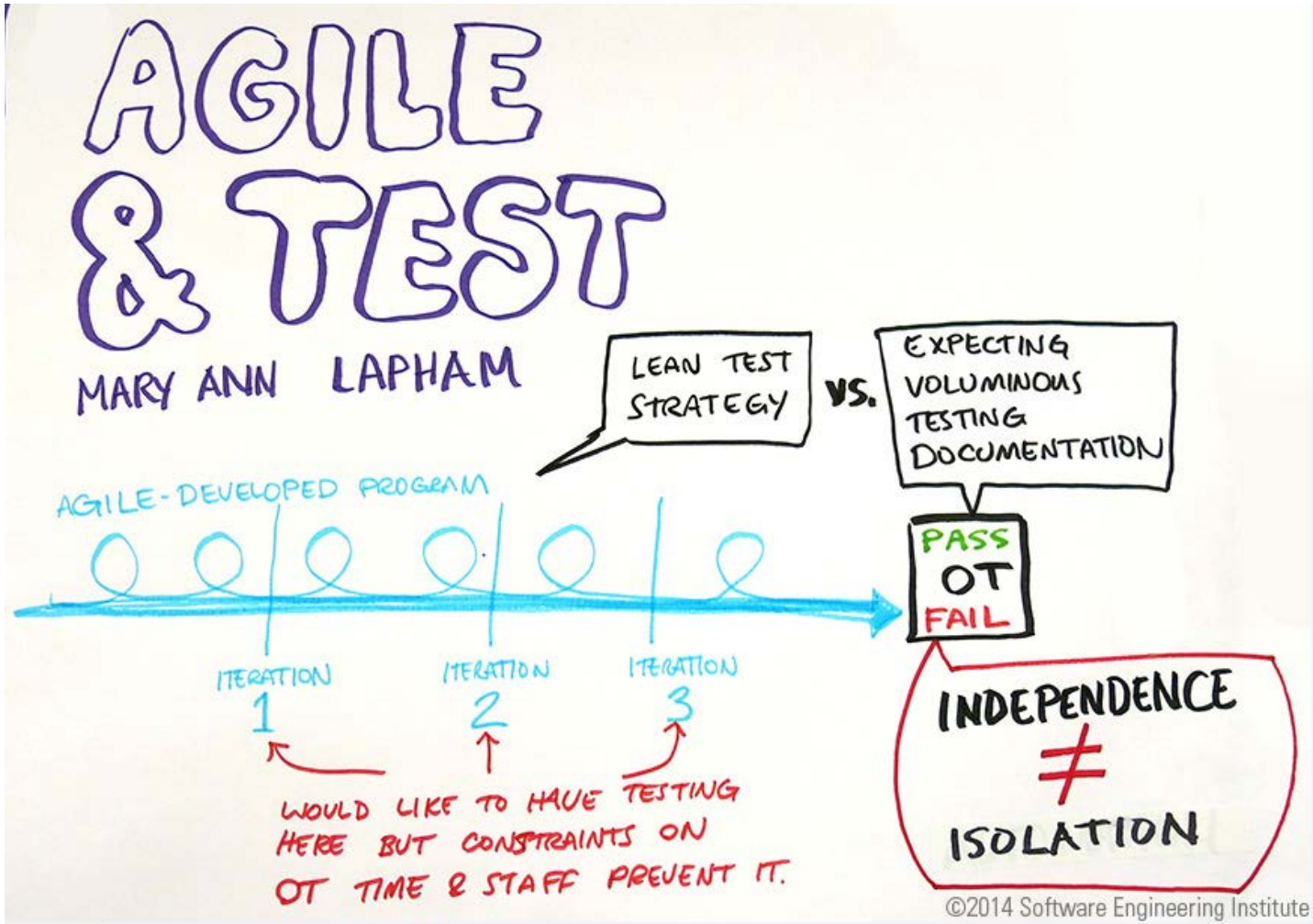
DISCUSSION: what other steps should the program office take to ensure productive interaction with systems engineering in an Agile setting?

© 2015 Carnegie Mellon University

Agile and Integration & Testing

© 2015 Carnegie Mellon University

SEI Research In This Area is Still In Process



© 2015 Carnegie Mellon University

Role of Software Testing in SW-Reliant Systems

Testing plays many roles throughout the development of software:

- Providing input to requirements definition and design efforts
- Exercising or simulating a system or program operation
- Establishing a level of confidence that software does what it is supposed to, and doesn't do what it isn't supposed to
- Analyzing a program with the intent of finding problems
- Measuring system functionality and quality
- Evaluating the capabilities of programs and work products, and assessing whether they achieve acceptable results

T&E (Test & Evaluation) includes inspections and structured peer reviews of requirements and design, as well as execution test of code

- Good practice references include peer reviews as a recommended practice

Adapted from *SW Program Manager's Network Guide*

© 2015 Carnegie Mellon University

Agile Provides a Unique Opportunity for Early *Validation* of User Requirements

Agile principles and practices focus on validation issues:

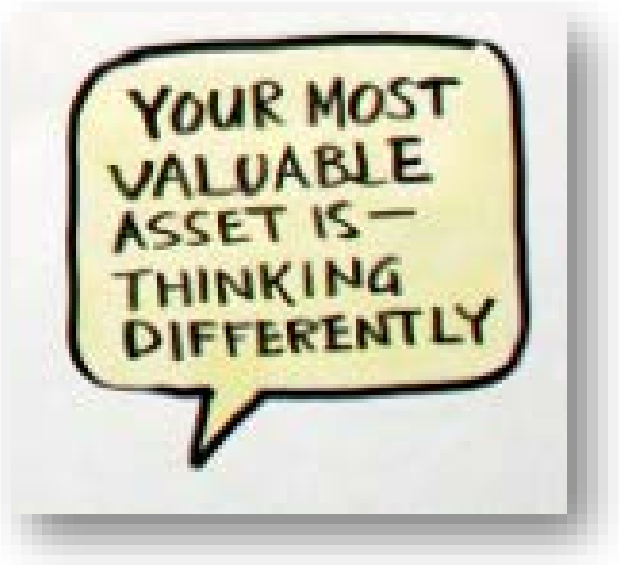
- Value of stories to the end user
- Conversations with end users during development to validate understanding of requirements
- End-Iteration and End-Release demonstrations of working software

© 2015 Carnegie Mellon University

This Area is Still a Struggle in Many Settings

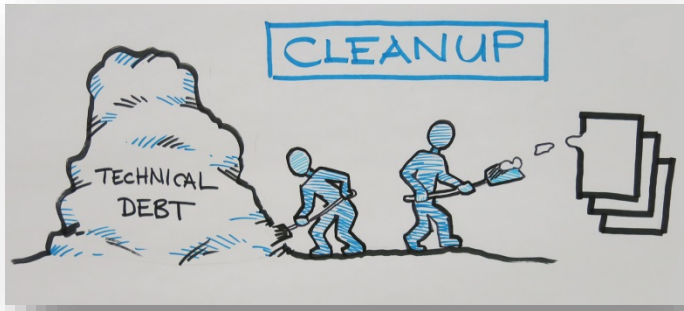
Where test communities are willing to work incrementally, results are mixed

- Sometimes more effort earlier in the process which isn't budgeted for
- BUT, problems found earlier are usually easier to fix
- AND, early insight into the shape of the product sparks productive dialogue



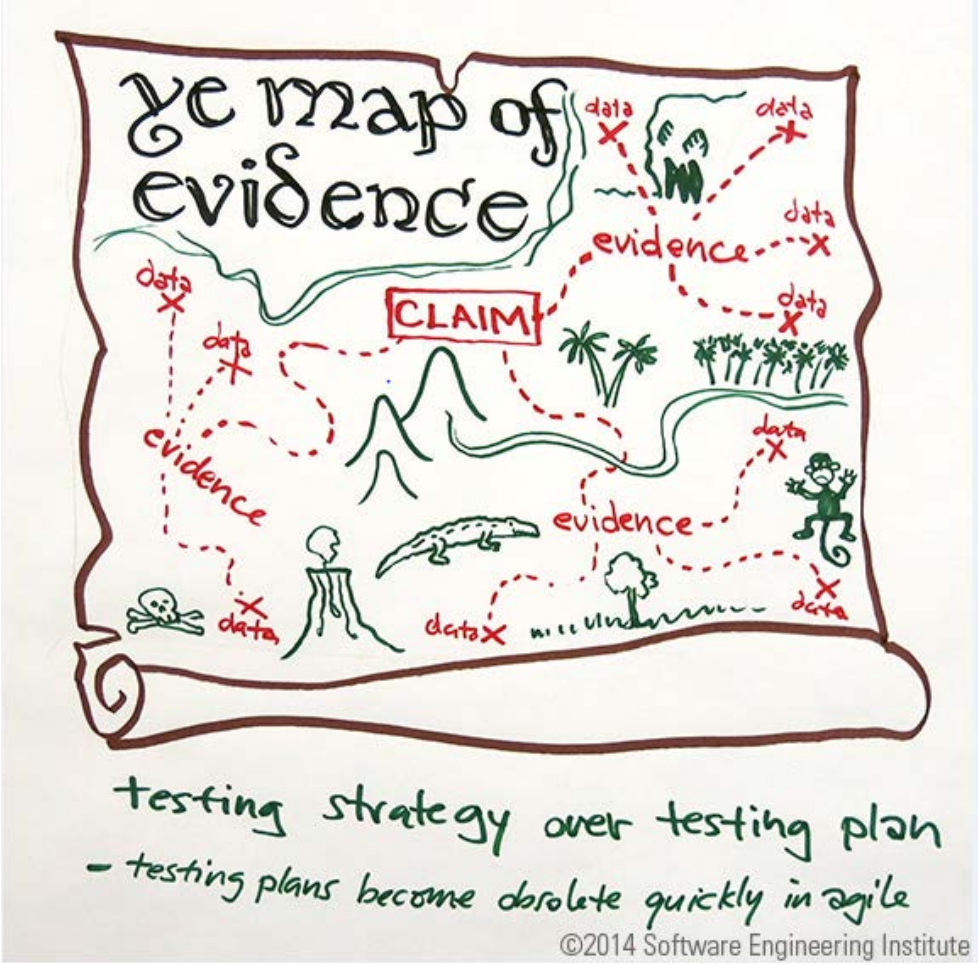
Where test communities demand total independence and do *not* participate in early demos and reviews

- Some of the benefits of early validation are lost
- Rework can be similar to traditional systems



© 2015 Carnegie Mellon University

The Crux of the Matter: How You "Get Evidence" in Agile is Seen Differently than Traditional Test Approaches



© 2015 Carnegie Mellon University

Agile and Technical Reviews

© 2015 Carnegie Mellon University

Insight and Oversight…

There is a great deal of OVERSIGHT activity that is required by the acquisition system to progress a program, software or otherwise

- Many of the mechanisms used for acquisition oversight could be seen as substitutes for the communication that naturally occurs in a trust-based relationship
- Regardless of the informal communication on the program, required oversight has to be accomplished

The other goal for contract monitoring is to achieve INSIGHT into the program

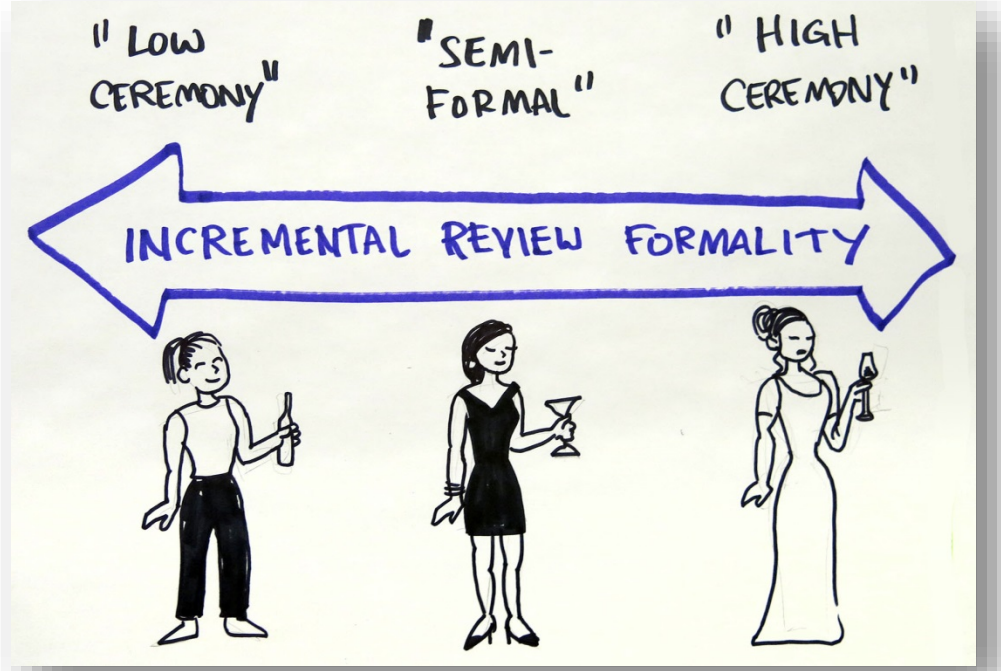
- Acquisition CDRLS and required events are not always the best way to achieve insight
- Agile development settings, in particular, promote transparency and have built in mechanisms for achieving ongoing insight
  - These mechanisms, however, require proactive participation from the acquirer to be effective

© 2015 Carnegie Mellon University

One of Top Three Questions SEI Hears about Agile

How do I accommodate Technical Reviews like PDR, CDR, etc.?

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



© 2015 Carnegie Mellon University

SEI Found 3 Patterns in Agile Settings for PDR, CDR Design/Execution

Pattern A

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

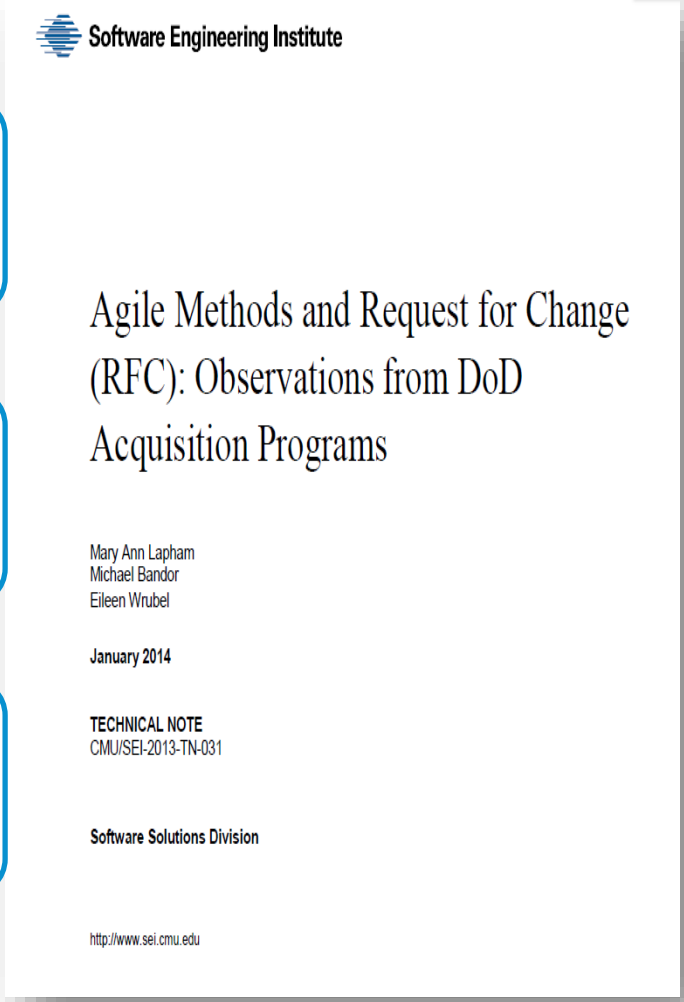
Pattern B

- 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


Software Engineering Institute

Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs

Mary Ann Lapham
Michael Bandor
Eileen Wrubel

January 2014

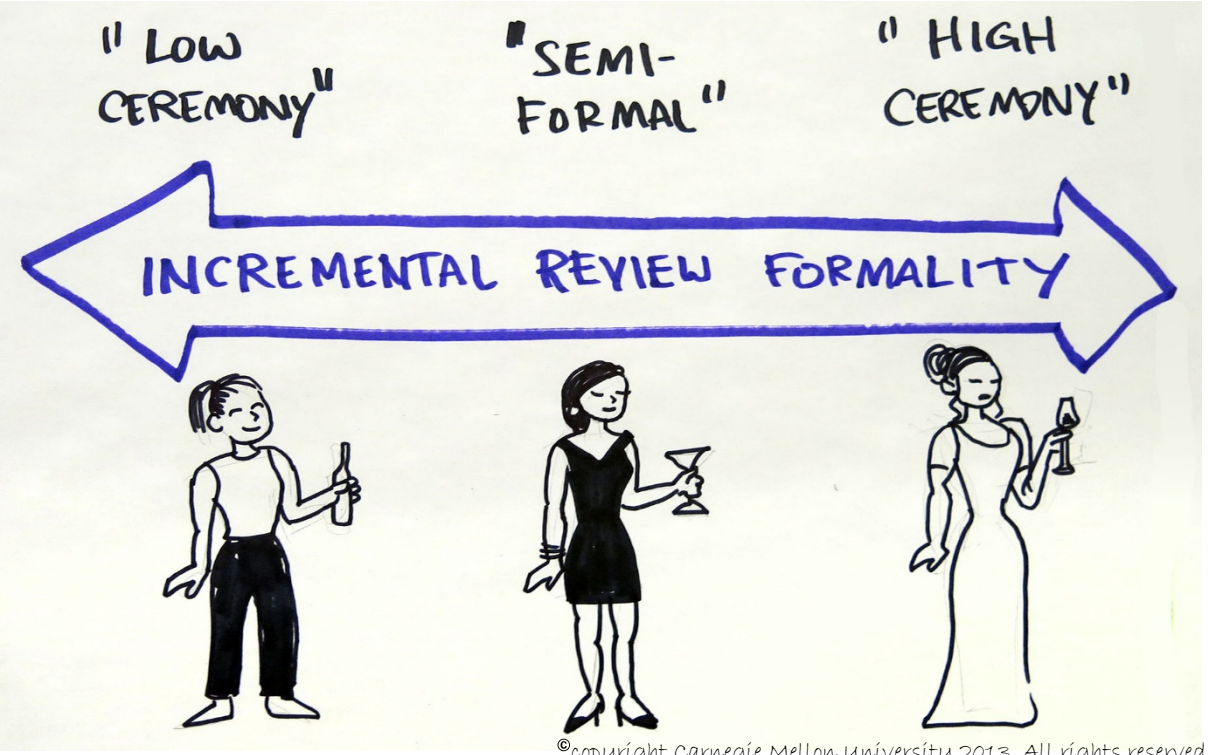
TECHNICAL NOTE
CMU/SEI-2013-TN-031

Software Solutions Division

http://www.sei.cmu.edu

© 2015 Carnegie Mellon University

Agile Acquisition Challenges: Technical Review Strategies



©copyright Carnegie Mellon University 2013. All rights reserved.

Which of the approaches discussed in the previous slide is your program using? What is the rationale for your choice?

© 2015 Carnegie Mellon University

Review not Mentioned in Most Acquisition Guidance—End of Sprint/Iteration Demonstration

Practitioners of Agile/iterative methods typically conduct an iteration review at the end of each iteration to which product owners and other stakeholders are invited—SAFe incorporates this demonstration, as well as an end of Release demonstration

- Working software is demonstrated
- Other required documentation may be reviewed
- The success of the iteration is determined
- Inputs for backlog changes for next iteration may be solicited

Key event in iterative development for keeping up with progress and influencing the shape of the development

- Take advantage of participating whenever you can

© 2015 Carnegie Mellon University

Releases Prior to Engineering Release—Agile Considerations

When a contractor is using Agile methods, one of their risk mitigation approaches is to provide multiple internal releases of working software to program stakeholders

- So that stakeholder feedback can provide course corrections as the implementation progresses
- To incrementally provide value

Agile releases are typically made up of 3-6 short iterations (2-4 weeks)

- In non-agile formats, these types of reviews should occur informally (TIMs and other types of notifications)

Release Reviews, in addition to iteration/sprint reviews, are opportunities for the Program Office to interact directly with the software as it progresses

- Make them more useful by building relationships with IA, OT&E, and other certification authorities that encourage them to work early with internal releases in building their own activities and providing early feedback and course corrections to the acquisition team

© 2015 Carnegie Mellon University

Agile & Progress Measurement

© 2015 Carnegie Mellon University

Context-Everyone Wants Some Kind of Progress Measure

© 2015 Carnegie Mellon University

Sample Regulatory References

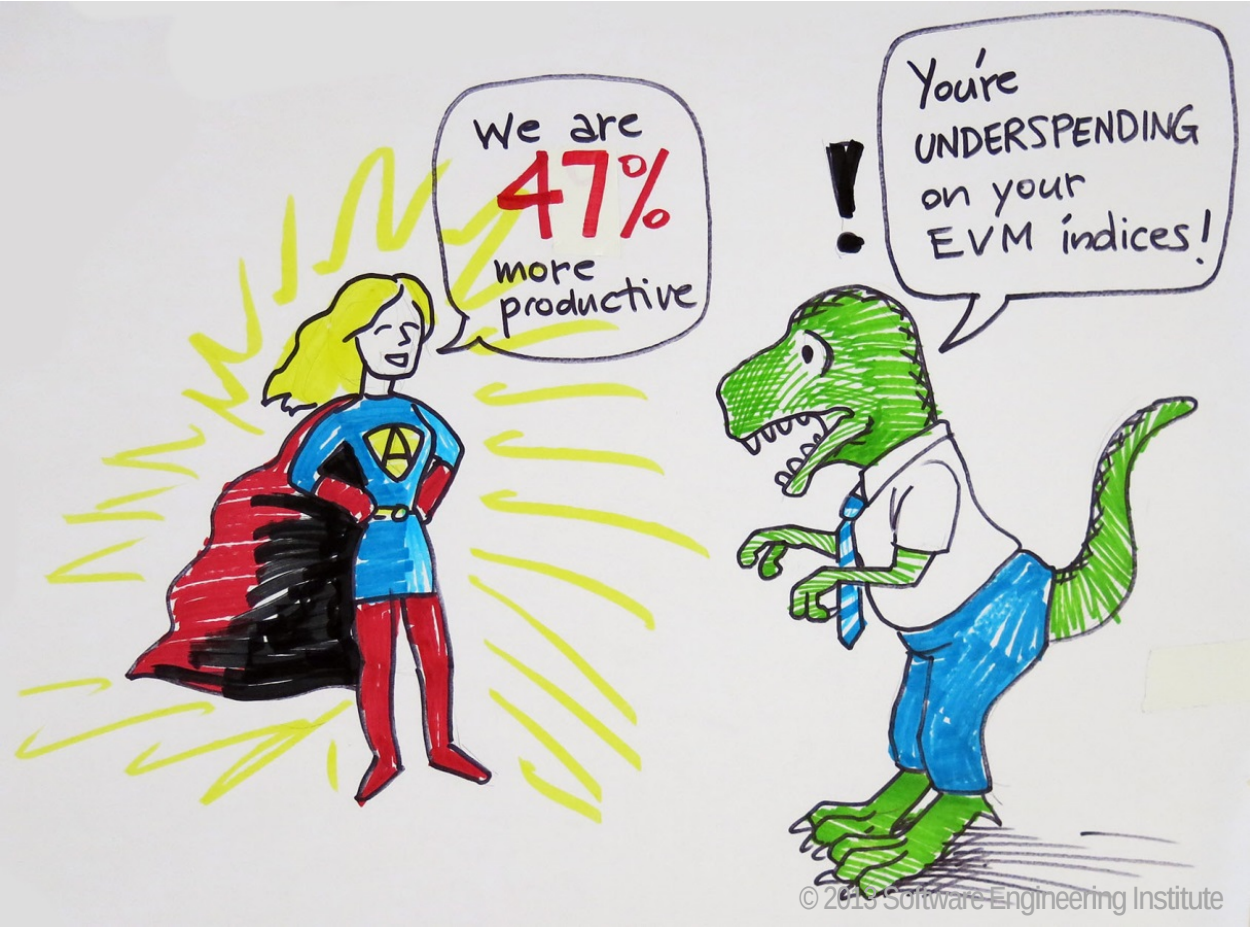
USAF Software Core Metrics	Army Regulation (AR) 70-1 Army Acquisition Policy
Software size Software development effort Software development schedule Software defects Software requirements definition and stability Software development staffing Software progress (design, code and testing) Computer resource utilization	Section 7-13 Software Metrics: PMs will negotiate a set of software metrics with the software developer to affect the necessary discipline in the software development process and to assess the maturity of the software product. At a minimum, the metrics should address— • Schedule and progress regarding work completion. • Growth and stability regarding delivery of the required capability. • Funding and personnel resources regarding the work to be performed. • Product quality regarding delivered products to meet the user's need without failure, as reflected in associated requirements documents. • Software development performance regarding the capabilities to meet documented program requirements. • Technical adequacy regarding software reuse, programming languages, and use of standard data elements.

Reference: United States Air Force. *United States Air Force Weapon Systems Software Management Guidebook, Version 1 (Abridged)*. 2008

Reference: United States Army. *Army Regulation (AR) 70-1 Army Acquisition Policy*, Sections 7-12 and 7-13. 2011.

© 2015 Carnegie Mellon University

Agile EVM Does Exist, But is Not Prevalent



Google search on "Agile EVM" will provide latest thinking and tool support—an evolving research area

© 2015 Carnegie Mellon University

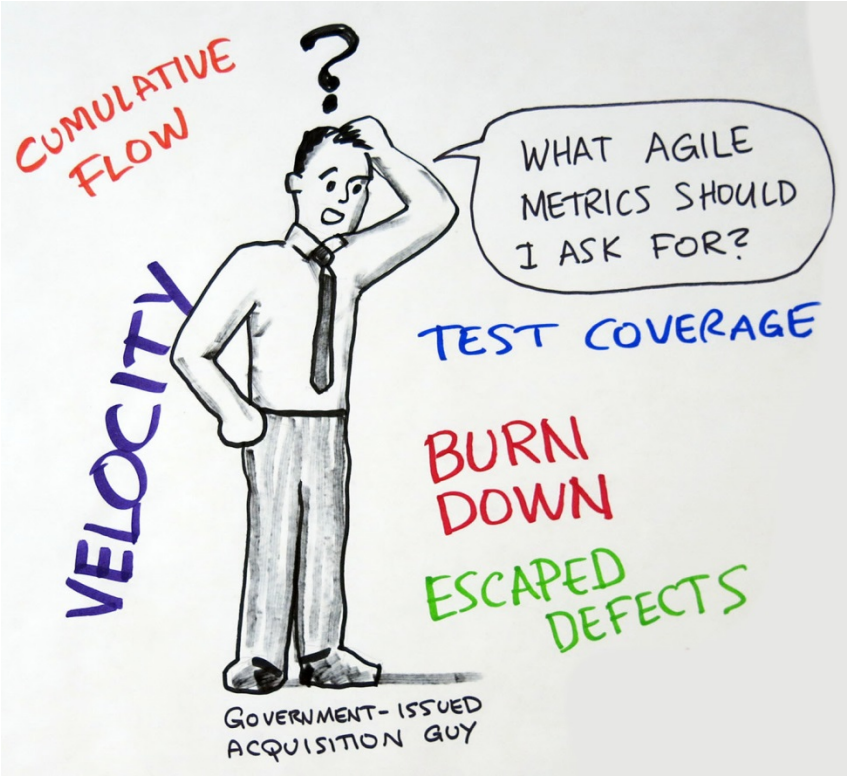
Integrating Agile Measurement into Your Overall Management Metrics

Agile/Iterative measures for teams leverage the approach of time-boxing (fixing the schedule for the iteration, and varying the target requirements to be fulfilled based on the team's capacity and capability)

- Opposite of traditional acquisition where schedule is based on estimates of what it takes to implement a fixed set of requirements

Agile/iterative iteration measures are meant primarily for the team's use

- External use of *iteration* metrics is generally discouraged
- *Release* metrics are generally used for monitoring/management purposes



© 2015 Carnegie Mellon University

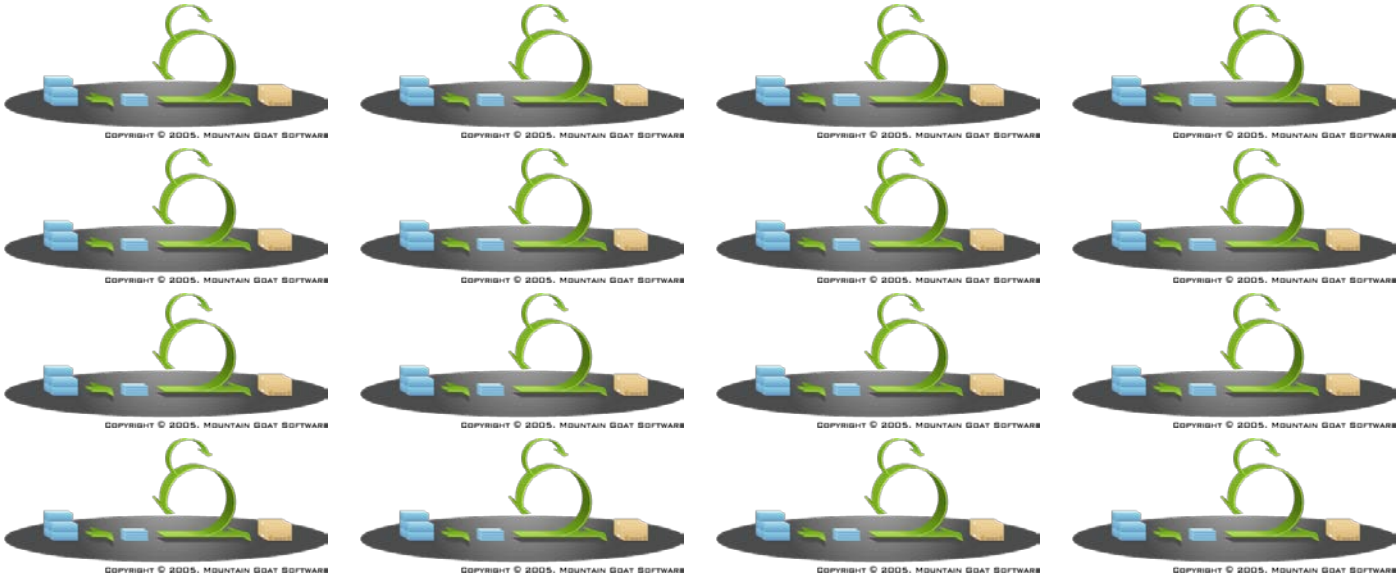
Program Level vs. Team Level Measures

Geared to External Stakeholders

Release 1 → Release 2 → Release 3 → Release 4

Intended to Serve Needs of the Team

Typically Not Shared Out-side the Team

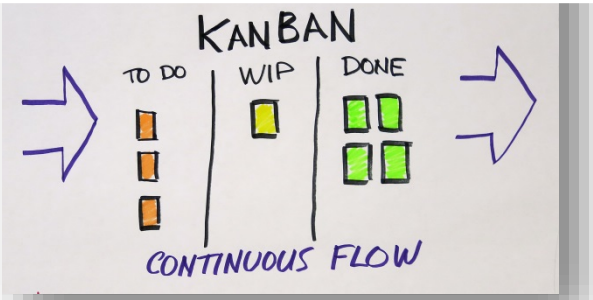


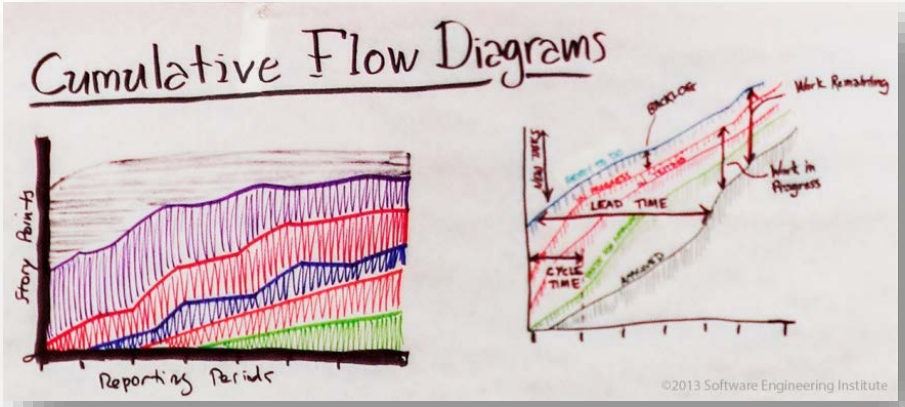
© 2015 Carnegie Mellon University

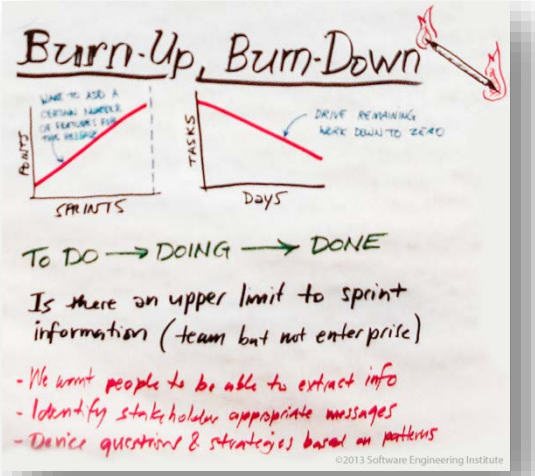
Typical Team Measures for Agile Development

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







© 2015 Carnegie Mellon University

Program Level Measures

Because teams focus on delivering working code:

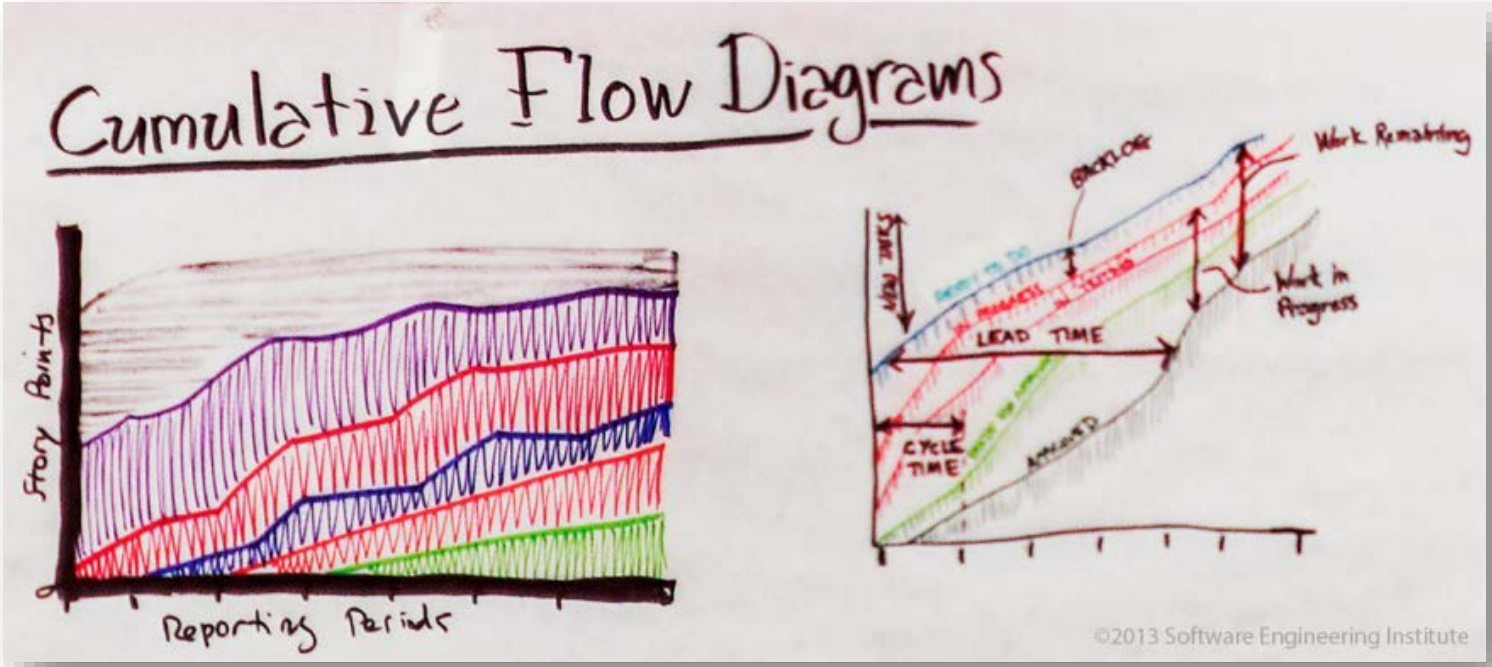
- The program can measure finished product (size, complexity, quality…)
  - Rather than estimates of the finished product being carried (and revised) across the program timeline, we can know *actual* values for incrementally completed work
- The program can focus on 'concept-to-capability' cycle
  -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

© 2015 Carnegie Mellon University

Typical Program Measures for Agile Development

Team data for release level insight

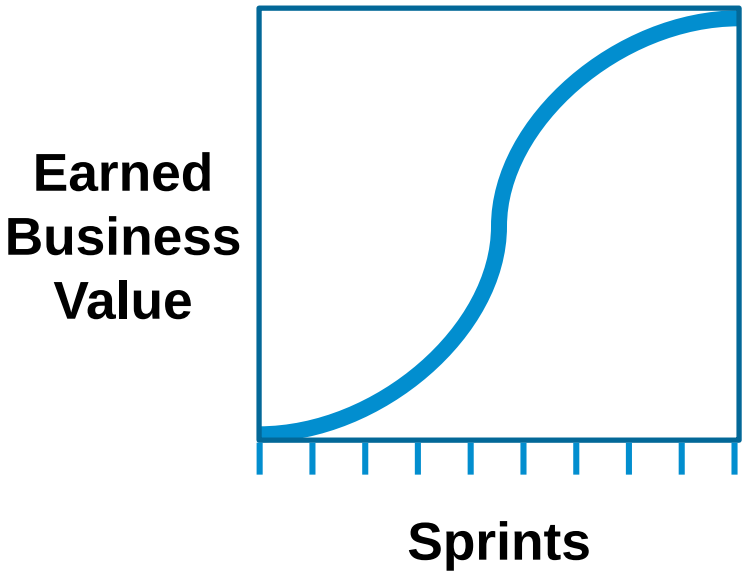
- Burn Up Charts
- Story Aging Histograms
- Cumulative Flow Diagrams



© 2015 Carnegie Mellon University

The (Earned) Value Proposition

Contract progress payments are based on "earned value" for the accounting period and therefore are considered our "rice bowl," something you simply do not mess with



From: Alleman, G. B., Henderson, M., & Seggelke, R. "Making Agile Development Work in a Government Contracting Environment: Measuring Velocity with Earned Value," in *Agile Development Conference*, June 25-28, 2003.

http://www.informatik.uni-trier.de/~ley/db/conf/agiledc/agiledc2003.html

EVM and Agile is an area of intense scrutiny and activity at this time—IDA study may bring some clarity, but no new guidance yet

© 2015 Carnegie Mellon University

Summary-1

In government settings, Agile methods are performed in a larger context with stakeholders who may or may not be bought in to the Agile approach

When systems engineering is actively engaged in an Agile mindset, shared understanding of benefits and problems can make the journey smoother

- BUT, if not integrated with contracting and budgeting approach, systems engineering may not have the resources at the right time to effectively engage in Agile practices

Integration and test continues to be a challenge area

- Independence of various test functions vs collaboration
- Integration into the larger system

© 2015 Carnegie Mellon University

Summary-2

Technical reviews and *what is reviewed when* is one of the most challenging aspects of harmonizing Agile methods and the DoD Acquisition Life Cycle

- All three approaches suggested offer advantages and disadvantages

"Go in with your eyes open!"

- Once you've selected an approach, do what you can to mitigate its risks and get full benefit from its advantages

© 2015 Carnegie Mellon University

Summary-3

Measuring progress for Agile can occur at multiple levels

- Measurement of individual team progress is best left inside the team
- Visualizations like Cumulative Flow Diagrams provide multiple insights for release-level progress

Focus of measurement is not “are we on time?” but “are we producing sufficient value fast enough?”

- Not the typical way programs are measured
- “Value point delivery” may be a way to address this

Earned value management has been adapted for use in Agile projects

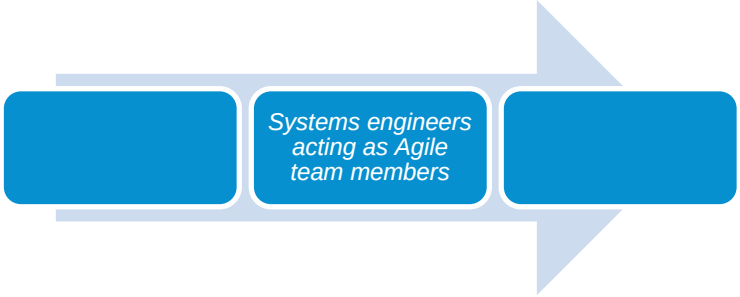
- Not well adopted yet in government settings

© 2015 Carnegie Mellon University

Backup Slides

© 2015 Carnegie Mellon University

Pattern 2-1 Systems Engineers Acting as Agile Product Owner



Surrogate for operational user (unavailable or large, diverse population of operational users)

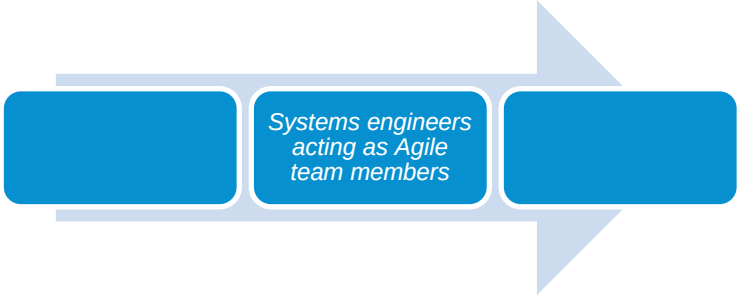
More of a product management team role

- Convened team of end users or their representatives
- Synthesized their prioritizations for requirements backlog
- Ensured questions asked by software team were answered

Challenge: availability, as systems engineers have their own set of artifacts and communications to do; rarely could they be dedicated full time as a product owner

© 2015 Carnegie Mellon University

Pattern 2-2 Systems Engineers Acting as Agile Systems Architect



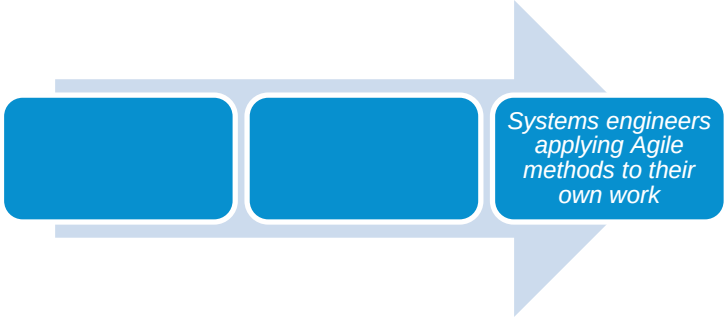
SAFe, among other scaling approaches, acknowledges a special role for the systems architect

Challenge: availability

- One solution: define kinds of questions the architect needed to answer and agree on turnaround time to make part-time architect more productive on Agile team

© 2015 Carnegie Mellon University

Pattern 3: Systems Engineers Applying Agile Methods to Their Own Work



Most often seen in ***IT settings*** (no significant hardware development)

- At least one case of Agile systems engineering methods applied across system, hardware, and software tasks

Used ***Scrum*** method when applying Agile principles

- Adapted to the **artifact transformation** aspect of systems engineering

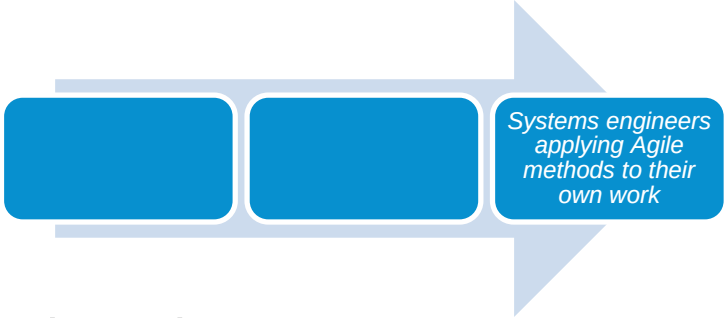
Used ***Kanban*** (Lean) method for ***communication and similar functions***

Translation of "***working software***" to "***working product***" was biggest issue

- Early SE phases focus strongly on producing ***documents*** that guide implementation
  - If no regulatory relief, then documents become the working product
- Breadboard/brassboard hardware ***prototypes***, and ***simulators and emulators*** supporting early software development are more natural fit

© 2015 Carnegie Mellon University

Systems Engineers Applying Agile Methods to Their Own Work -2



Enablers for development of working prototypes include

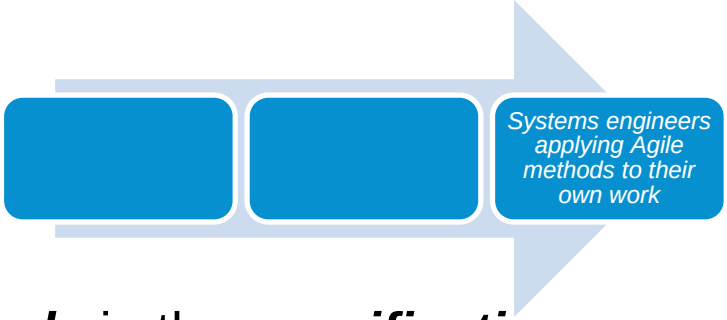
- Product backlog that includes items systems engineering can produce
- Technical or architecture stories for their content
- Explicit iteration/sprint planning
- Short iterations (< 1 month)
- Iteration demonstrations and reviews

Most important—mindful application of ***Agile principles*** focusing on

- User/engineer collaboration
- Learning versus “shutting down” requirements or design constraints too soon
- Empowering a cross-functional team to self-organize

© 2015 Carnegie Mellon University

Systems Engineers Applying Agile Methods to Their Own Work -3



[+] SE ***interacting closely with end users early*** in the ***specification*** process ***produced a level of trust*** that provided ***benefit throughout*** the entire development cycle.

[-] ***change in program manager*** who ***did not understand or support Agile*** methods resulted in

- ***Substitution of surrogates*** by the contractor as end users were no longer authorized to provide the ongoing support for the document development processes
  - ***Visible erosion of the trust*** between the development team and stakeholders
    - Resulted in ***documentation*** products that ***required*** later ***rework***
      - ***Due to insufficient understanding*** of the end user's perspective
    - Resulted in ***return to a lower level of detail for change authorization***

© 2015 Carnegie Mellon University

Definitions

Software Verification (typically DT):

The process of providing objective evidence that the software and its associated products

- *conform to requirements* (e.g., for correctness, completeness, consistency, accuracy) for all life cycle activities during each life cycle process (acquisition, supply, development, operation, and maintenance);
- *satisfy standards, practices, and conventions during life cycle processes*; and
- *successfully complete each life cycle activity and satisfy all the criteria for initiating succeeding life cycle activities* (e.g., building the software correctly). [IEEE 2004 (1012)]

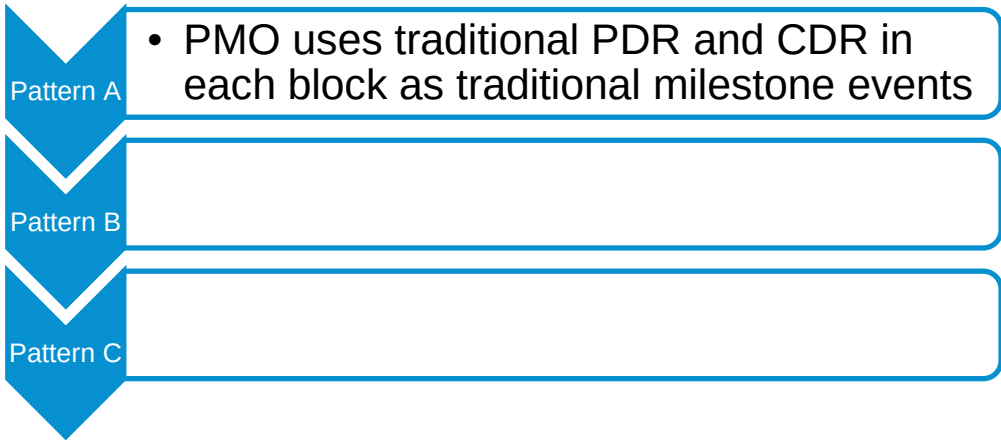
Software Validation (typically OT):

The process of providing evidence that the software and its associated products

- *satisfy system requirements allocated to software at the end of each life cycle activity,*
- *solve the right problem (e.g., correctly model physical laws, implement business rules, use the proper system assumptions), and*
- *satisfy intended use and user needs*. [IEEE 2004 (1012)]

© 2015 Carnegie Mellon University

PDR, CDR Pattern A



Advantages:

- Fits the more traditional acquisition life cycle
- Minimizes personnel travel costs

Disadvantages:

- Less synchronicity between development life cycle and review life cycle. CDRs and PDRs (per block) may be accomplished well into the iteration cycle raising the distinct possibility of rework in the event there is a direction change (e.g., requirements, etc.)
- PDR and CDR events end up being very long (3-5 days possibly) as information on each iteration will most likely be presented
- Decreases the in-process communication between the contractors and PMO regarding development efforts

© 2015 Carnegie Mellon University

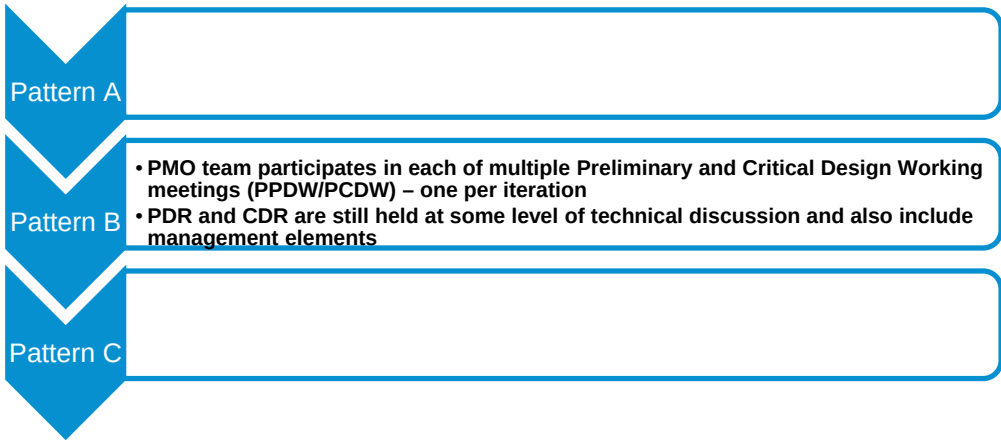
PDR, CDR Pattern A-Impact

Pattern A	• PMO uses traditional PDR and CDR in each block as traditional milestone events
Pattern B	
Pattern C	

- Artifacts under review different maturity levels
  - some complete
  - some under construction
  - some not yet started
- Result - confusion and unnecessary comment adjudication
- Fix - Insist on proper training and coordination so not to defeat purpose of PDR/CDR

© 2015 Carnegie Mellon University

PDR, CDR Pattern B



Advantages:

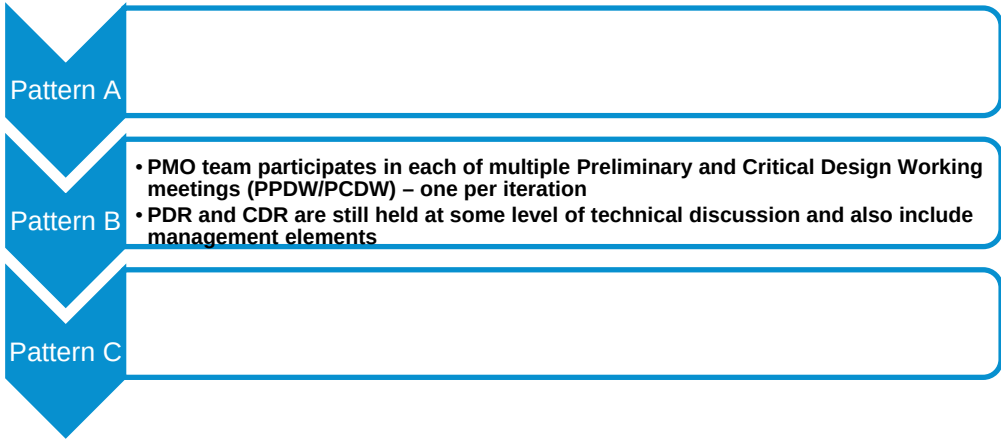
- Allows for earlier looks at the evolving products by program office staff and end users or their surrogates
- Allows for direction change if needed much earlier in the delivery cycle of the increment
- Potentially shortens the PDR and CDR to a high-level review (summarization of development efforts and outstanding action items)
- Allows for better communication between contractors and PMO regarding development efforts
- Aligned well with the DAG guidance on incremental development
- Potentially increased synchronicity between development life cycle and review life cycle.

Disadvantages:

- Could require more travel (if not done remotely) and resource allocation to review activities by PMO personnel; however, costs related to these activities are highly likely to result in lower overall program cost and risk
- Potential for loss of big picture view due to dependencies across iterations. The work required to maintain the big picture could be greater than for traditional approaches.
- Risk that required replanning may not receive appropriate emphasis as contents and requirements move from iteration to iteration.

© 2015 Carnegie Mellon University

PDR, CDR Pattern B-Impact



- Only review artifacts for the iteration being reviewed.
- Artifacts should be relatively same maturity
  - set by appropriate entry / exit criteria
  - if different maturity, watch for confusion
    - different maturity may lead to unnecessary comment adjudication

© 2015 Carnegie Mellon University

PDR, CDR Pattern C

Pattern A

Pattern B

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

Advantages:

- Potentially shortens the CDR to a high-level review (summarization of development efforts and outstanding action items)
- Allows for earlier looks at the evolving products by program office staff and end users and their surrogates
- Allows for direction change if needed much earlier in the delivery cycle of the increment
- Allows for better communication between contractors and PMO regarding development efforts
- Aligned well with the DAG incremental guidance
- Potentially increased synchronicity between development life cycle and review life cycle.

Disadvantages:

- Requires more travel and resource allocation for review activities by PMO personnel
- Requires increased communications by technical staff due to conveying interim review results to PMO management personnel.

© 2015 Carnegie Mellon University

PDR, CDR Pattern C-Impact

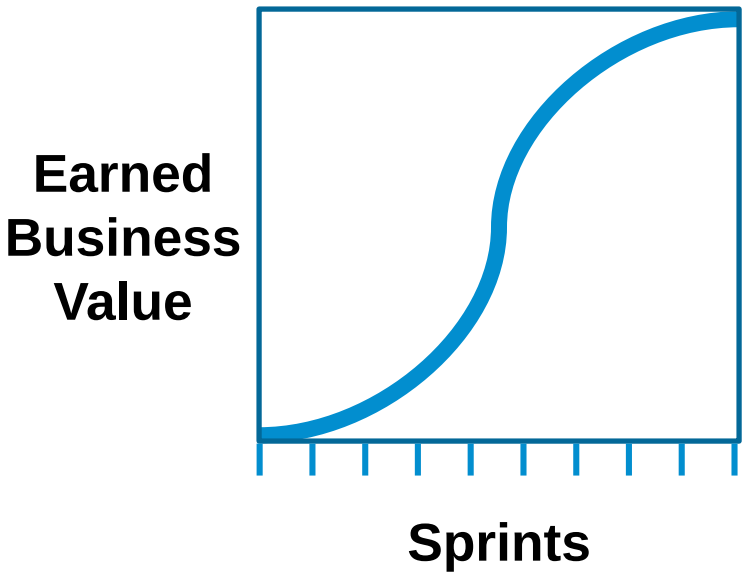
Pattern A	
Pattern B	
Pattern C	• PMO technical staff (engineers) participate in each PPDW/PCDW (per iteration) • PDR and CDR become management level reviews • No technical detail is discussed in PDR and CDR other than a summary for management

- Additional communications required to ensure effective information flow from technical staff to PMO personnel.
- Communication must accurately present results and corresponding context for each iterative review.
  - Further, non-verbal aspects of the iterative review as well as management level nuances can be difficult to capture in prose.

© 2015 Carnegie Mellon University

The Value Proposition

Contract progress payments are based on "earned value" for the accounting period and therefore are considered our "rice bowl," something you simply do not mess with



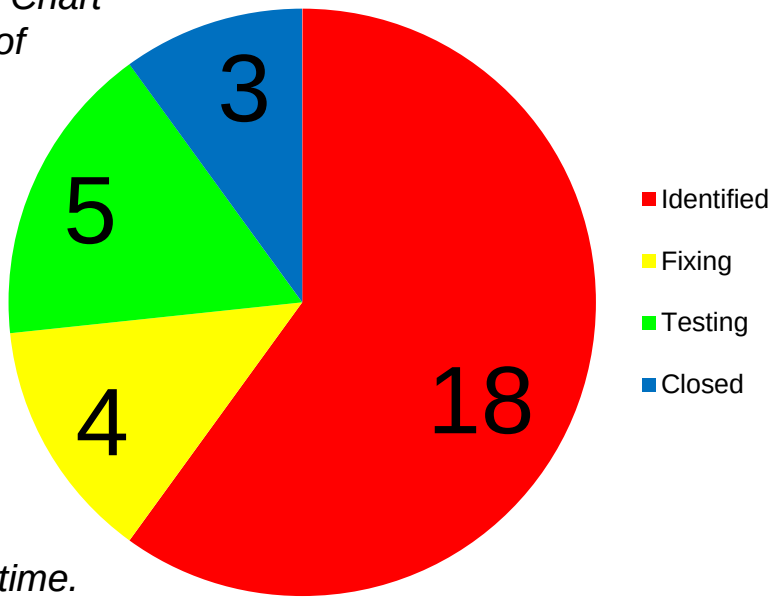
From: Alleman, G. B., Henderson, M., & Seggelke, R. "Making Agile Development Work in a Government Contracting Environment: Measuring Velocity with Earned Value," in *Agile Development Conference*, June 25-28, 2003.

http://www.informatik.uni-trier.de/~ley/db/conf/agiledc/agiledc2003.html

© 2015 Carnegie Mellon University

Constructing a Cumulative Flow Diagram-1

Here we have a Pie Chart showing the status of 30 defects across the four stages of the defect handling life-cycle.



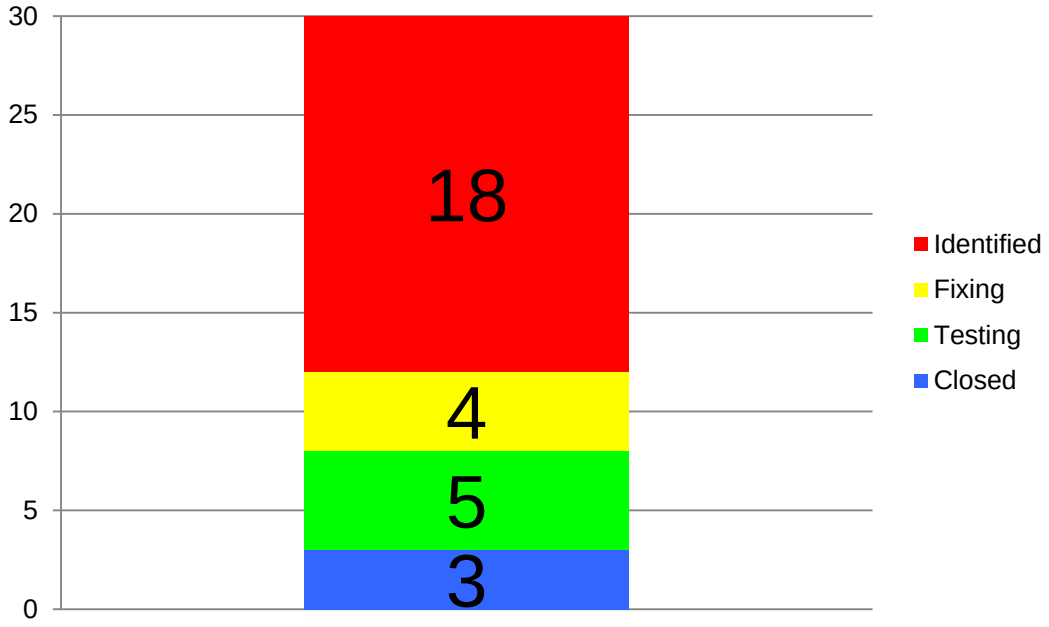
This is a snapshot for a single point in time.

© 2015 Carnegie Mellon University

Constructing a Cumulative Flow Diagram-2

Same data, but presented in a stacked column chart

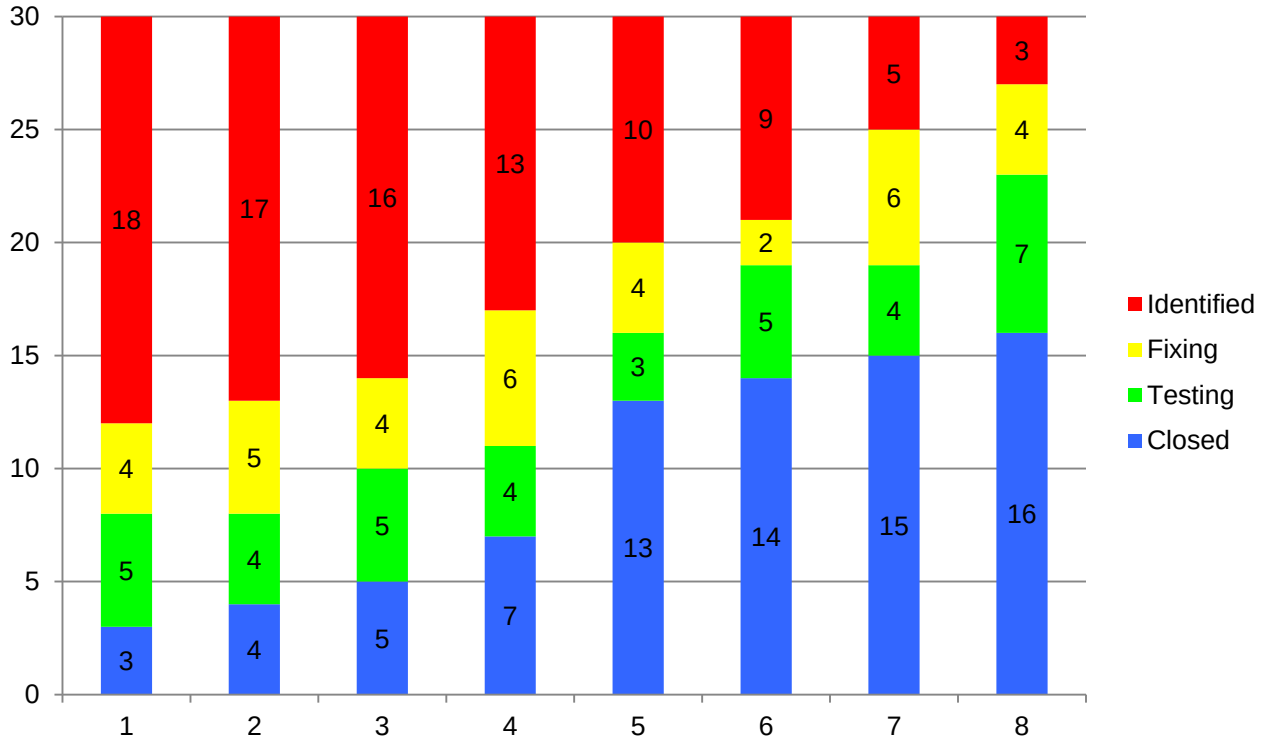
For a single point in time.



© 2015 Carnegie Mellon University

Constructing a Cumulative Flow Diagram$_3$

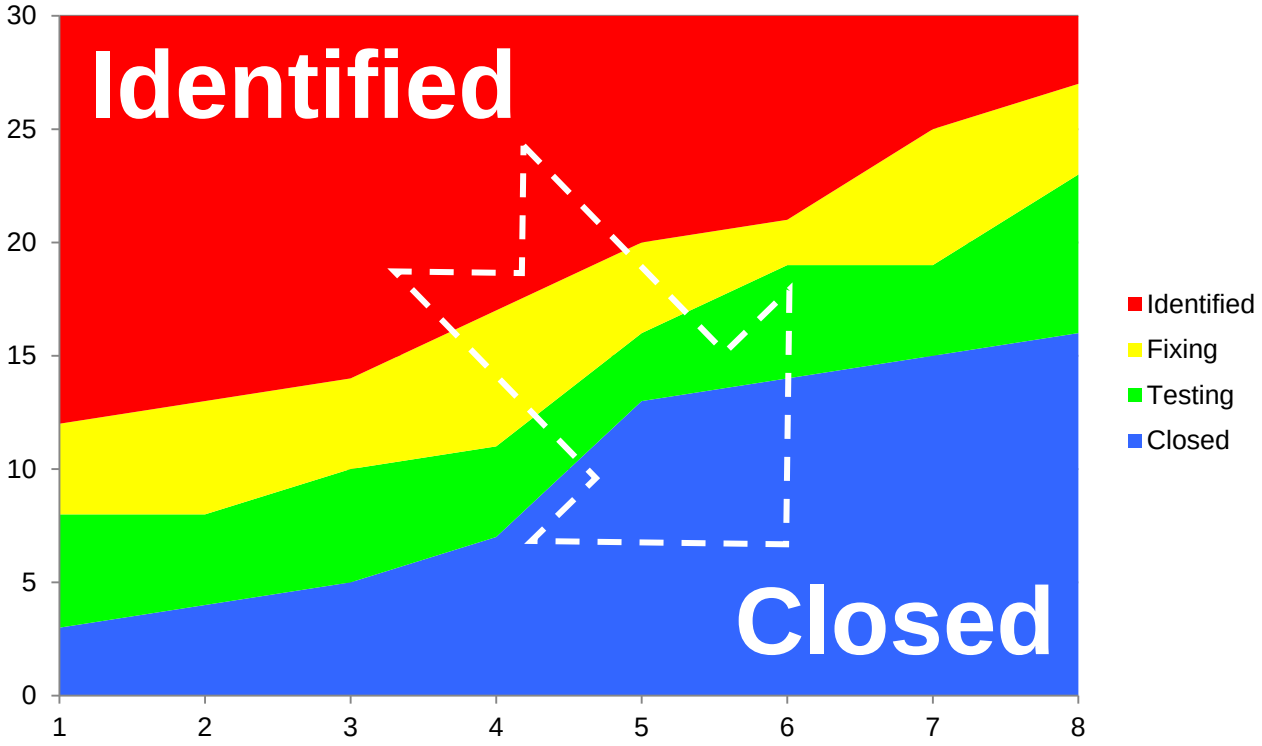
… adding the next 7 times



© 2015 Carnegie Mellon University

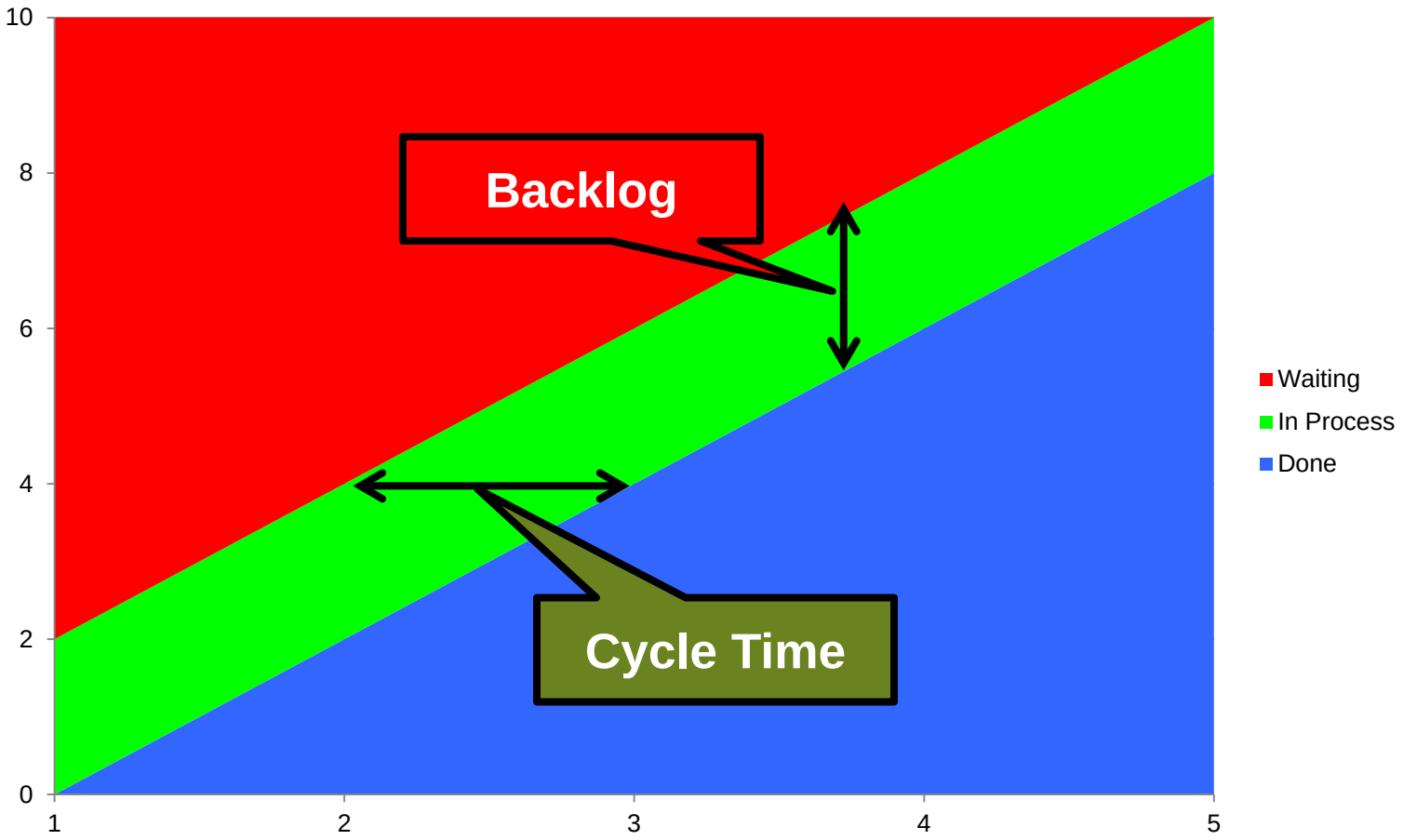
Constructing a Cumulative Flow Diagram$_4$

… now we are looking at the flow from "identified"… to "Closed"…
This view starts to show patterns a little easier…



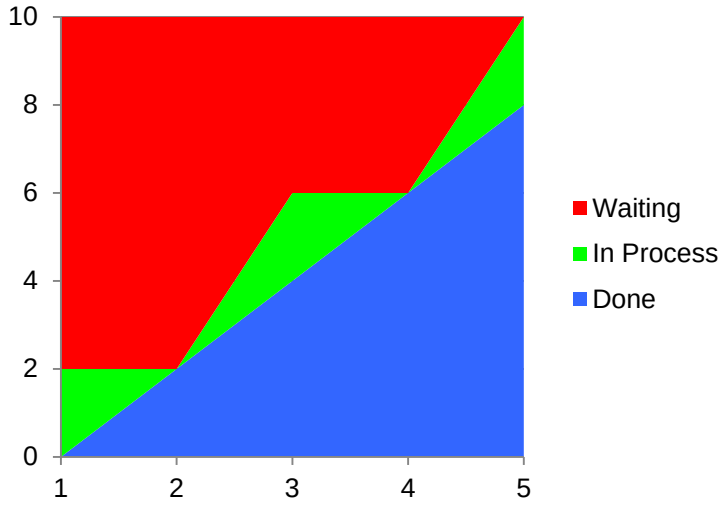
© 2015 Carnegie Mellon University

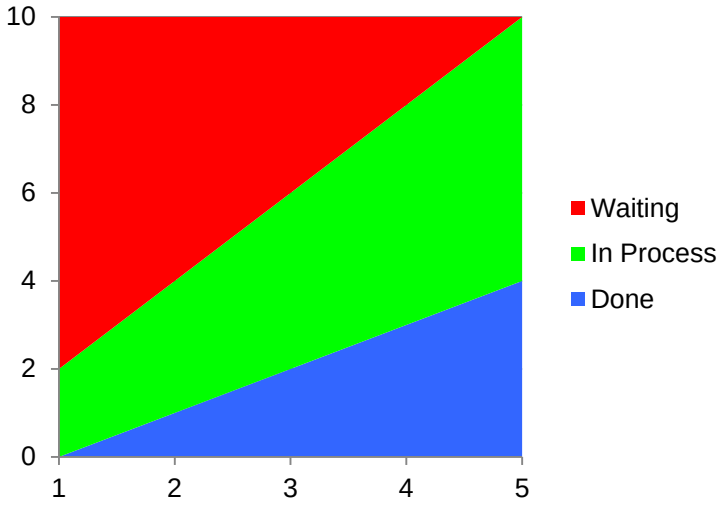
Tell-Tale Signals



© 2015 Carnegie Mellon University

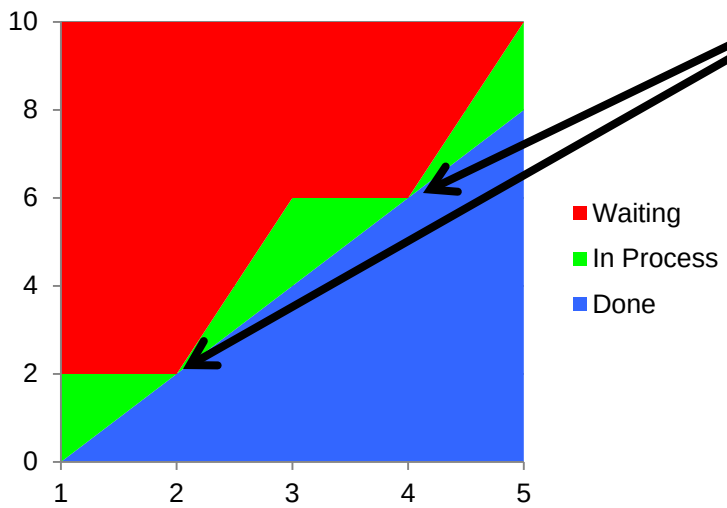
Exercise: What is Going on Here?





© 2015 Carnegie Mellon University

Exercise: What *MIGHT BE* Happening$_1$



At time 2, and then again at time 4, the number of items "In Process" goes to zero.

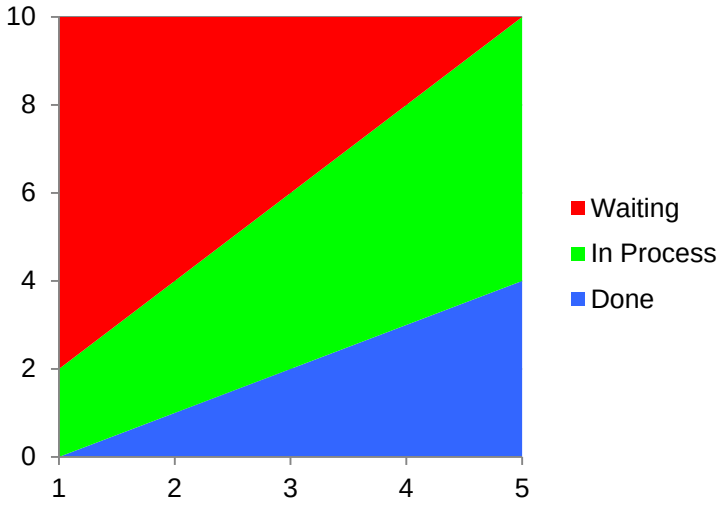
- **Have we lost the resource(s) that were preparing the items in the "Waiting" state?**
- **Is this intentional, due to limited resource(s) who can work on items in the "In Process" state?**

© 2015 Carnegie Mellon University

Exercise: What *MIGHT BE* Happening$_2$

The number of items that are "In Process" is growing over time.

- **The rate at which things enter "In Process" is greater than the rate at which things leave "In Process."**
- **Are people moving onto new items without completing their work?**
- **Are new resources being added, who start new work at each time period?**
- **Are things moving into the "Done" state quickly enough?**

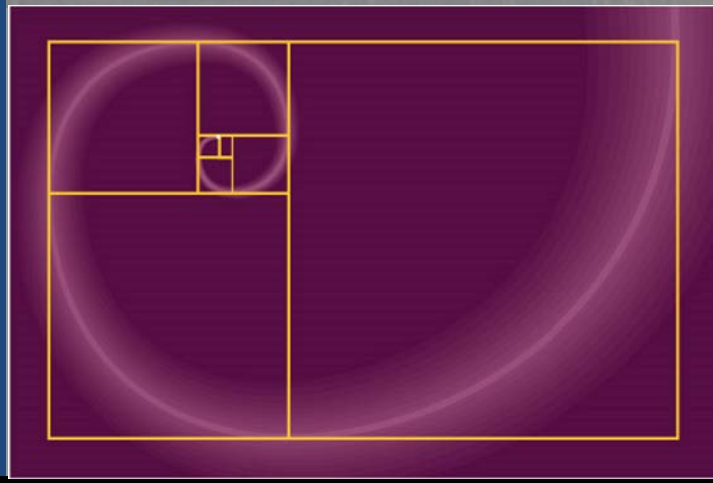


© 2015 Carnegie Mellon University

Agile in Government: Practical Considerations

CTSD Agile Adoption in Regulated Settings Team
Software Engineering Institute
July 2015

Module 5 (Documentation, Contracts, Regulatory Oversight)



© 2015 Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

This material has been approved for public release and unlimited distribution except as restricted below.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study. Except for the U.S. government purposes described below, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose this material are restricted by the Rights in Technical Data-Noncommercial Items clauses (DFAR 252-227.7013 and DFAR 252-227.7013 Alternate I) contained in the above identified contract. Any reproduction of this material or portions thereof marked with this legend must also reproduce the disclaimers contained on this slide.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

This material was prepared for the exclusive use of NRO Agile in Government Workshop and may not be used for any other purpose without the written consent of permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0002617

Contents

Module 1: Introduction

Module 2: Basic Agile Concepts

- Agile Introduction
- Agile Tenets and Principles
- Differences Between Traditional and Agile
- Common Agile Concepts and Practices
- Agile Mythbusting

Module 3: New Role for PMO

- Why Agile
- The Product Owner Role

© 2015 Carnegie Mellon University

Contents (continued)

Module 4: Agile Program Oversight

- System Engineering
- Integration and Test
- Technical Reviews
- Measurement

Module 5: Agile in the Larger Eco-System

- ***Documentation***
- ***Contracting***
- ***Regulatory Oversight***

Module 6: Enabling an Agile Acquisition Culture

- Enabling and Transitioning to an Agile Culture
- Suggestions for Implementing Agile in Government

Module 7: Summary

© 2015 Carnegie Mellon University

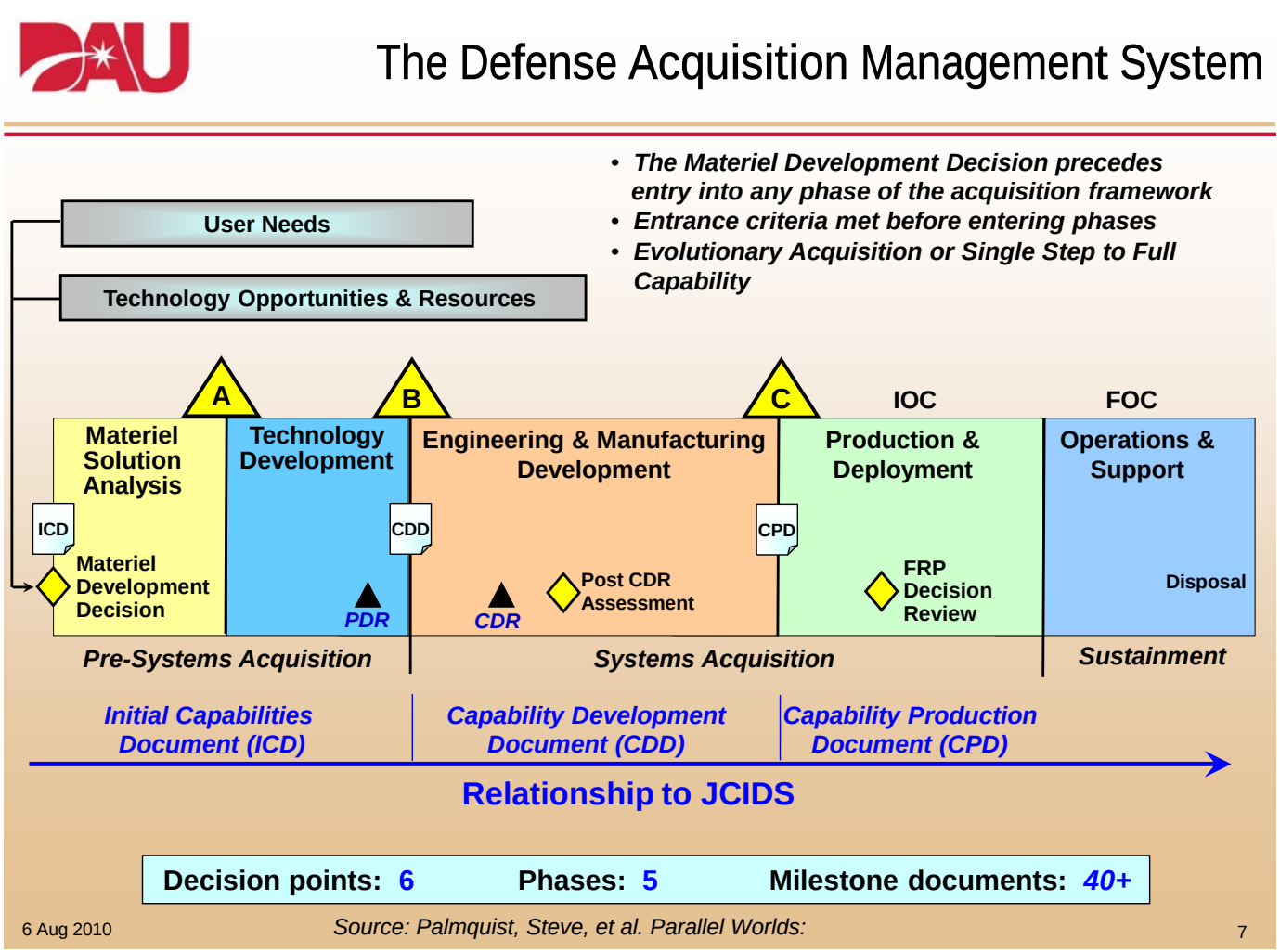
Learning Objectives
Module 5: Government Agile Interactions 3

After this module, participants will be able to:

1. Discuss why contract type is not the primary consideration when trying to contract effectively for Agile contractors
2. List recent changes to DoD 5000.02 that may make use of Agile in DoD settings easier than the 2008 version
3. List 2 different ways to address documentation requirements in Agile settings

© 2015 Carnegie Mellon University

Remember-Agile Methods are Used in Context of a Larger Program (More Often Than Not)



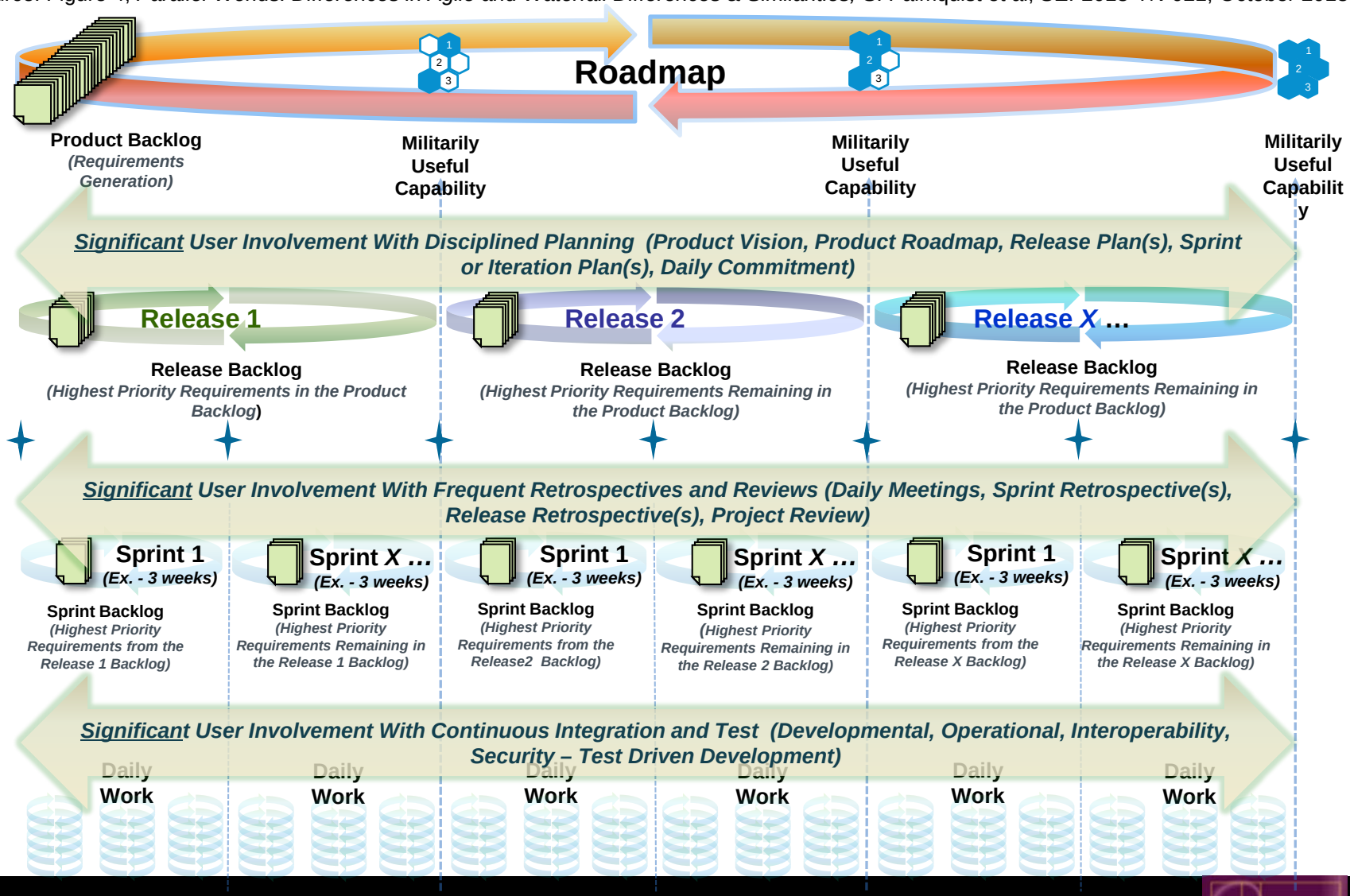
© 2015 Carnegie Mellon University

Agile & Regulatory Oversight

© 2015 Carnegie Mellon University

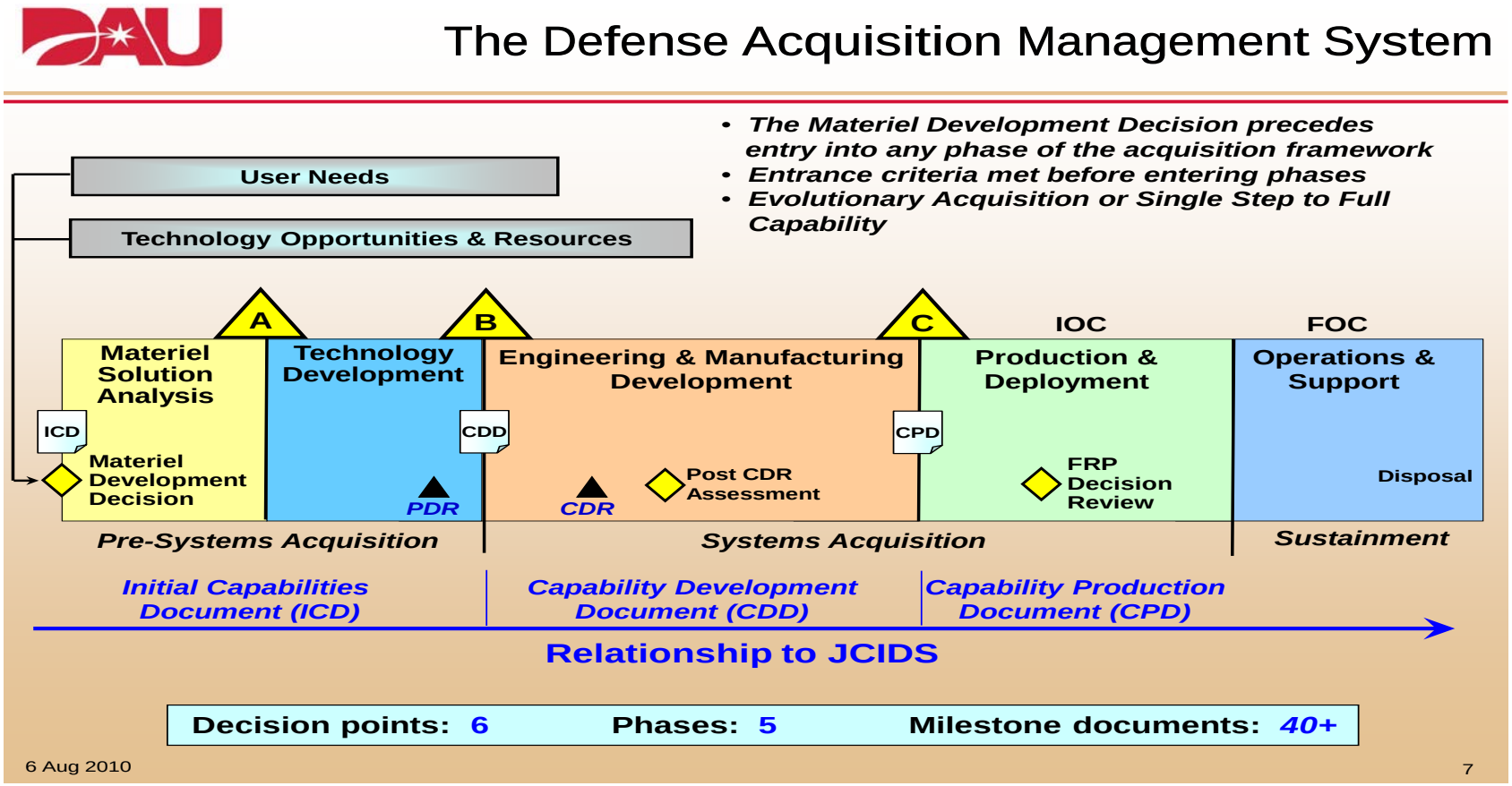
This is What an Agile Program "Could" Look Like

Source: Figure 4, *Parallel Worlds: Differences in Agile and Waterfall Differences & Similarities,* S. Palmquist et al, SEI-2013-TN-021, October 2013.



© 2015 Carnegie Mellon University

This is the Acquisition Cycle That is Still (Generally) Expected

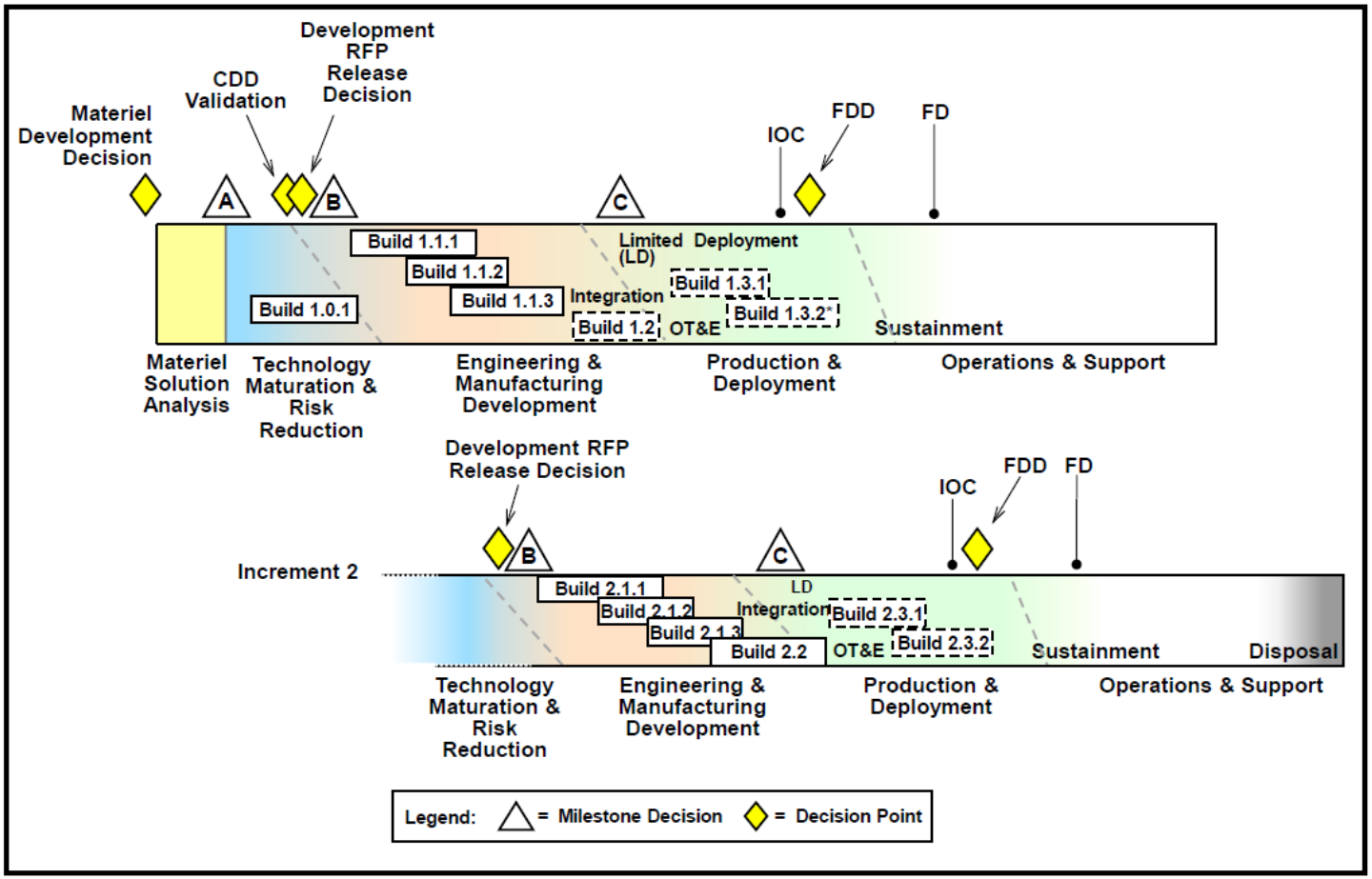


Source: Palmquist, Steve, et al. Parallel Worlds:

© 2015 Carnegie Mellon University

Recent Changes in DoD Acquisition Guidance Address SW-Dominant HW/SW Systems

Figure 8. Model 6: Hybrid Program B (Software Dominant)



© 2015 Carnegie Mellon University

Points Where Agile Makes a Difference

Acquisition Strategy

- If Agile will be allowed/enabled, appropriate life cycle needs to be selected

RFP Creation

- How will cost estimates be evaluated if no prior Agile projects in the relative independent costing databases (on gov't side)?
- Adjusting review expectations, CDRL contents and cadence, technical baseline level of abstraction, incentive fee plans within acceptable parameters from oversight agencies

Proposal Review:

- Understanding if/how offerors understand "Agile"
- Evaluating proposals that have a different cadence than traditional oversight expects

Contract Execution:

- Reporting cost, schedule, technical parameters back to oversight agencies in the ways they want it vs the way the Agile project is more productively managed
- Navigating acquisition milestone requirements

© 2015 Carnegie Mellon University

Agile & Documentation

© 2015 Carnegie Mellon University

© 2015 Carnegie Mellon University

Alternatives for Producing Documentation

Tailored documentation requirements; what is provided is produced incrementally with the code

SETA contractor hired by the program office to review the repository of development information (embedded in a tool that supported Agile methods) and produce required documentation from it.

Technical writers embedded with the Agile team produced documentation in parallel with the development activities.

Contractor personnel doing program controls activities produced required documentation toward the end of each release. The contractor program control personnel took the outputs from the Agile process and formatted them to meet the 5000 required documents.

If particular documentation produces no value for particular program – then seek waivers

© 2015 Carnegie Mellon University

Two Major Types of Documentation

PROJECTIVE documentation (documentation that projects how the system will behave or what it will do) is generally seen as less valuable in Agile settings

- Build-to detailed design specification
- Detailed software requirement specifications
- …(the word specification is a clue that the document is projecting what is expected, not documenting what exists)

AS-IS documentation (documentation that describes the completed system or functions for a particular stakeholder group) is usually still needed

- Maintenance manual for sustainment programmers
- Database schema for DBAs
- User manual for end users
- Install manual for deployers
- Configuration listing for IT operations staff

© 2015 Carnegie Mellon University

Weight or Wait?

Agile seeks to overcome weighty processes of the past — However, it's not the weight of documentation, but the wait for feedback we are trying to reduce.

©2014 Software Engineering Institute

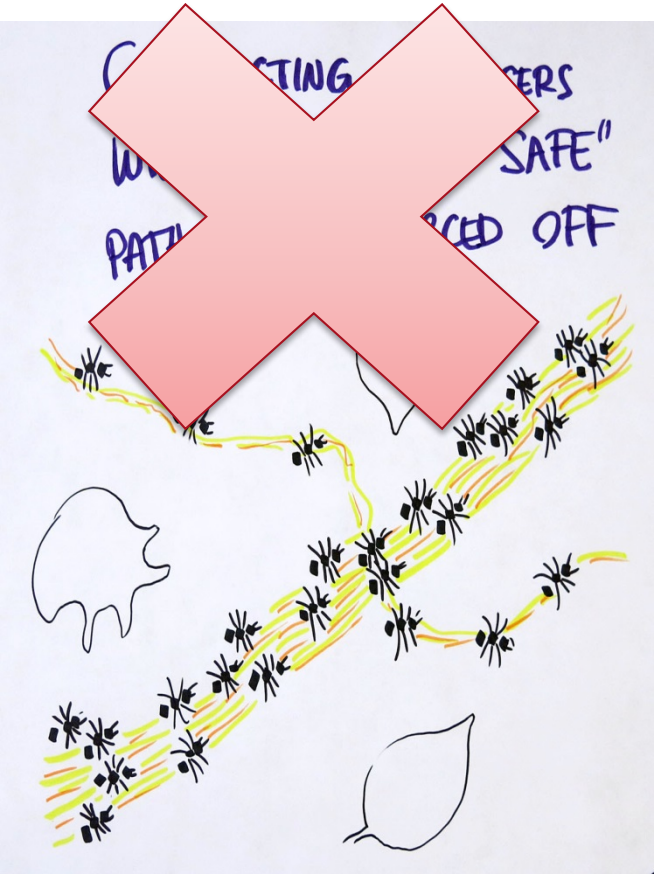
© 2015 Carnegie Mellon University

Agile & Contracting Issues

© 2015 Carnegie Mellon University

Innovative Contracting Case Studies - OSTP

"the Office of Science Technology Policy (OSTP) and the Office of Management and Budget are pleased to release the first version of Innovative Contracting Case Studies, an iterative, evolving document that describes a number of ways Federal agencies are getting more innovation per taxpayer dollar – all under existing laws and regulations. "



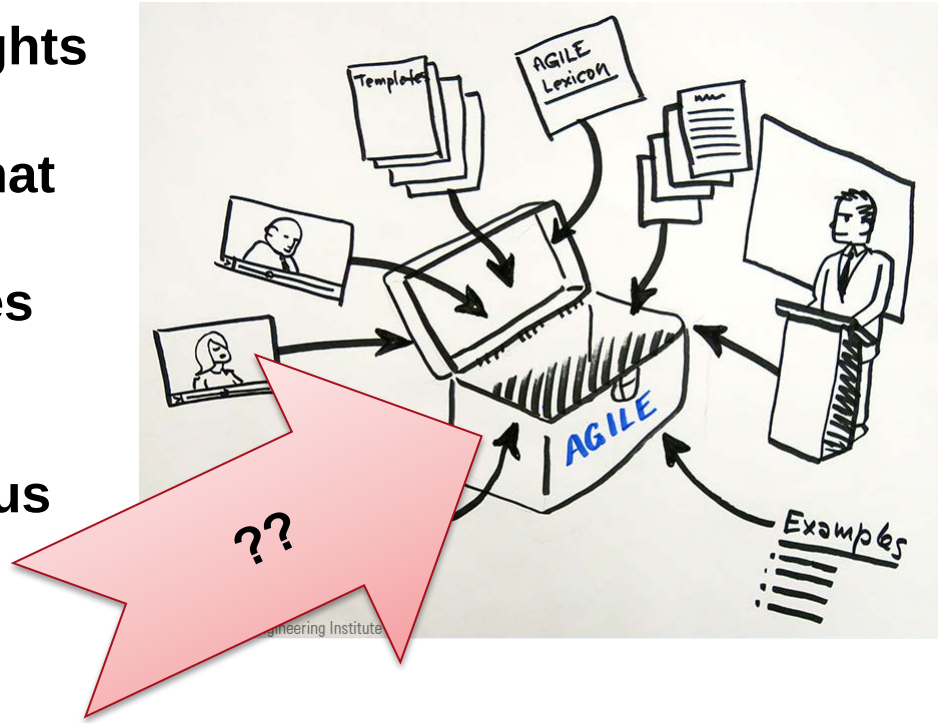
©copyright Carnegie Mellon University 2013. All rights reserved.

http://www.whitehouse.gov/sites/default/files/microsites/ostp/innovative_contracting_case_studies_2014_-_august.pdf

© 2015 Carnegie Mellon University

Federal Government Initiatives Are Starting to Deal Directly with Contracting/Reporting Issues

The TechFAR Handbook highlights the flexibilities in the Federal Acquisition Regulation (FAR) that can help agencies implement "plays" from the Digital Services Playbook that would be accomplished with acquisition support — with a particular focus on how to use contractors to support an *iterative, customer-driven software development process,* as is routinely done in the private sector.



https://playbook.cio.gov/techfar/

© 2015 Carnegie Mellon University

The Purpose of Government Contracts and Contract Oversight

In theory, contracts reduce risk and protect both parties from mismatched expectations by:

- Clarifying roles and role boundaries
- Establishing objective criteria for success
- Providing redress approaches for perceived incorrect actions or products that don't meet expectations
- Establishing the boundaries of the scope of the work
- Establishing expectations of interaction among the parties

Over time, government regulations related to contracts have been added to the Federal Acquisition Regulations (FARs) and to policy documents like DoD 5000.02

- Attempting to minimize risk to government and taxpayers for ill-conceived or poorly designed or executed products
- Attempting to prevent a negative outcome that DID occur from re-occurring, mostly by adding oversight points and constraints

© 2015 Carnegie Mellon University

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

© 2015 Carnegie Mellon University

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 is trying to build sample fragments – stay tuned!

© 2015 Carnegie Mellon University

More on RFPs….

Things we saw in our contracting research regarding RFPs:

- “Iterative development with frequent demonstrations” was seen multiple times in contract language
- Language that supports Agile principles and process flows, but not using “do Agile” per se
- Including Agile from beginning resulted in projects having improved agility
- Successful projects also had more involved Agile-educated stakeholders who provided “top cover;” key factor in success of contract, including trust

© 2015 Carnegie Mellon University

More Observations from research on Contract Type

Key factor ***NOT*** specific contract type, but contract deliverable & reporting structure

- Specific language, artifacts and oversight required within contract
- Milestones and CDRLs: many required to adhere to ***traditional review cycle***
- Many had ***concurrence to tailor*** SDR/PDR/CDR reviews with Agile components ***providing visibility*** into project ***via product demos***

IDIQ popular contract type to get capability on contract and provide visibility

- With CPAF, CPIF or T&M-based task orders to streamline time, effort and cost in contracting effort
- Having a pool of pre-qualified Agile-capable contractors enabled at least one site to do more performance-based contracting

Remember not all Government contracts are "Contracts" per se!

- ***Organic*** software development, especially in sustainment centers use MOAs/MOUs between acquirer and developer, with negotiated SOWs or SOOs
- Same challenges with respect to deliverables and milestones; no apparent success differences
- Determine if organic organization is *in GENERAL* compatible with contracting organization

© 2015 Carnegie Mellon University

Summary

"How do I appropriately contract for Agile?" is still one of the top questions we get about Agile in government settings

- Answers are highly dependent on
  - Program product and project context,
  - Attitude of senior government management in the chain towards/against Agile,
  - Local Program Office culture
  - Prior knowledge/experience of Program Office and Contracting staff with Agile
  - Experience of bidder pool with Agile in the product context being solicited

Contract type doesn't seem to be as much a predictor of ease of Agile implementation as

- Documentation requirements
- Technical review norms
- Enabling incremental integration and delivery

© 2015 Carnegie Mellon University

Summary-2

Documentation choices for the program need to account for how stakeholders are involved

- Projective vs As-Is documentation
- Incremental vs Big-Bang Delivery
- Documentation “as you go” vs at specific release points

Regulatory oversight is starting to change

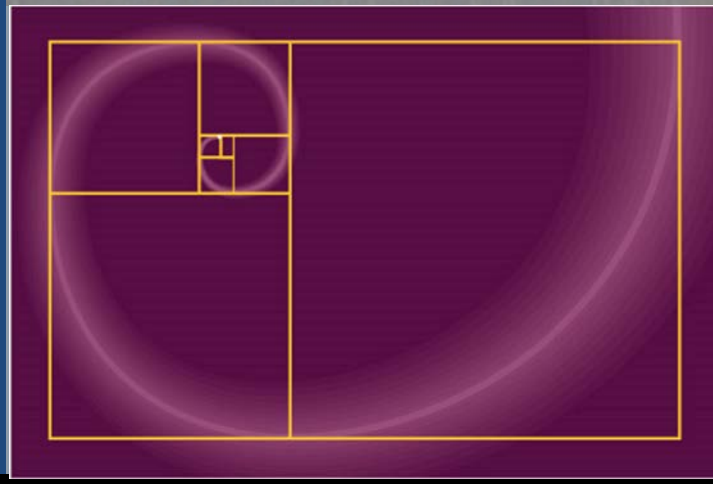
- Several DoD-related working groups in industry associations have initiatives related to contracting and reporting
- Some government working groups are dealing with issues of contracting and reporting
- Involving oversight stakeholders early can help manage expectations

© 2015 Carnegie Mellon University

Agile in Government: Practical Considerations

CTSD Agile Adoption in Regulated Settings Team
Software Engineering Institute
July 2015

Module 6 (Enabling an Agile Acquisition Culture)



© 2015 Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

This material has been approved for public release and unlimited distribution except as restricted below.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study. Except for the U.S. government purposes described below, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose this material are restricted by the Rights in Technical Data-Noncommercial Items clauses (DFAR 252-227.7013 and DFAR 252-227.7013 Alternate I) contained in the above identified contract. Any reproduction of this material or portions thereof marked with this legend must also reproduce the disclaimers contained on this slide.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

This material was prepared for the exclusive use of NRO Agile in Government Workshop and may not be used for any other purpose without the written consent of permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0002617

Contents

Module 1: Introduction

Module 2: Basic Agile Concepts

- Agile Introduction
- Agile Tenets and Principles
- Differences Between Traditional and Agile
- Common Agile Concepts and Practices
- Agile Mythbusting

Module 3: New Role for PMO

- Why Agile
- The Product Owner Role

© 2015 Carnegie Mellon University

Contents (continued)

Module 4: Agile Program Oversight

- System Engineering
- Integration and Test
- Technical Reviews
- Measurement

Module 5: Agile in the Larger Eco-System

- Documentation
- Contracting
- Regulatory Oversight

Module 6: Enabling an Agile Acquisition Culture

- ***Enabling and Transitioning to an Agile Culture***
- ***Suggestions for Implementing Agile in Government***

Module 7: Summary

© 2015 Carnegie Mellon University

Learning Objectives
Module 6: Enabling an Agile Acquisition Culture

After Module 6, participants will be able to:

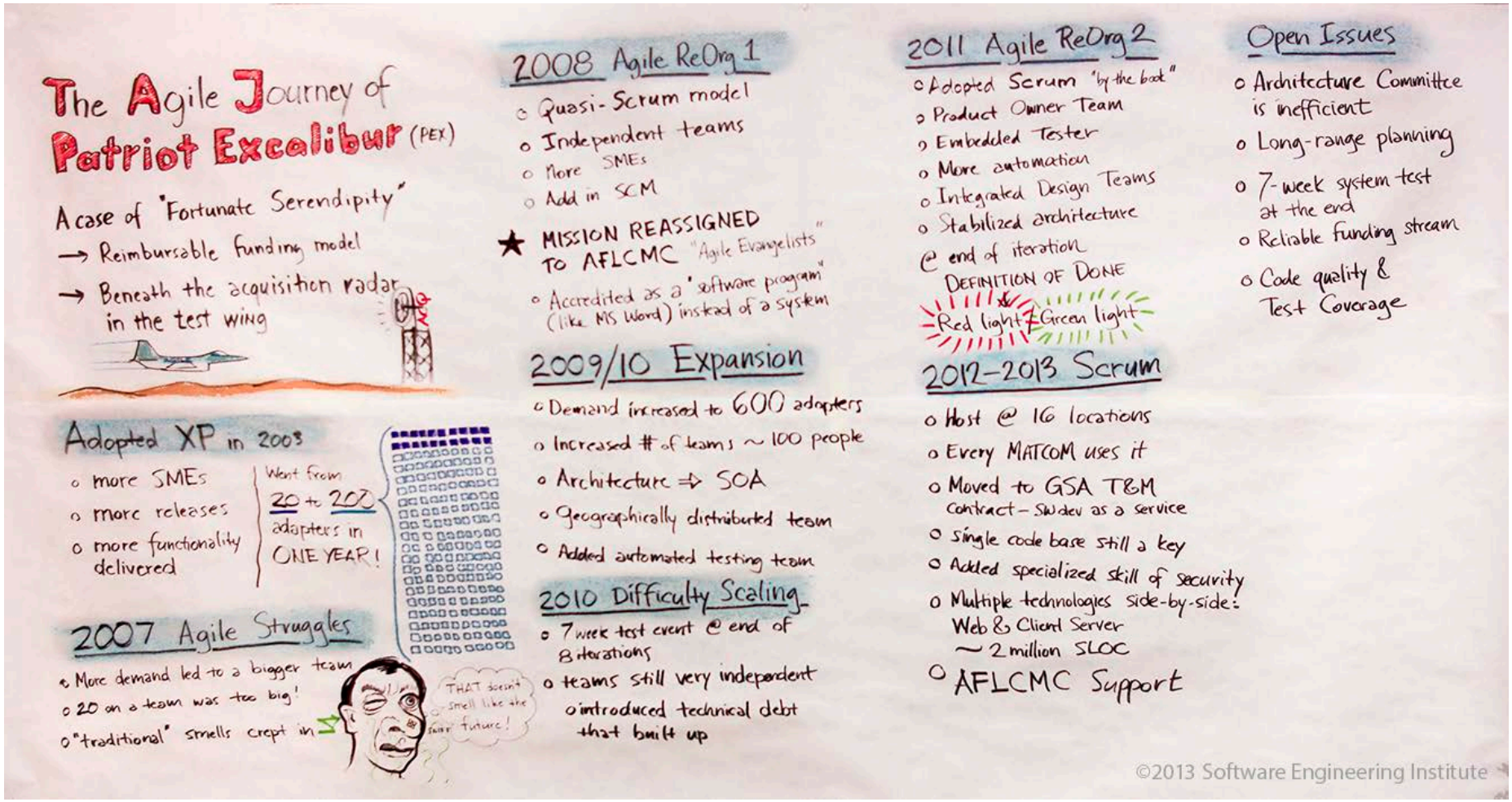
1. Describe two or more differences in the culture in a well-functioning Agile program and a traditionally managed program
2. Recognize training and other transition mechanisms that could be useful in implementing Agile methods in their program
3. Explain to a senior manager why they are considering using Agile on a program

© 2015 Carnegie Mellon University

Enabling and Transitioning to an Agile Culture

© 2015 Carnegie Mellon University

Patriot Excalibur: An Agile Success Story in DoD



© 2015 Carnegie Mellon University

Comparison of Agile and Traditional DoD Cultural Elements[1]

Knowledge Piece	Method	
Organizational Structure	**Agile DoD**	**Traditional DoD**
	• Flexible and adaptive structures • Self-organizing teams • Collocated teams or strong communication mechanisms when teams are distributed	• Formal structures that are difficult to change • Hierarchical, command-and-control-based teams • Integrated product teams that have formal responsibilities
Leadership Style	**Agile DoD**	**Traditional DoD**
	• Facilitative leadership • Leader as champion and team advocate	• Leader as keeper of vision • Leader as primary source of authority to act

http://www.sei.cmu.edu/library/abstracts/reports/11tn002.cfm?DCSext.abstractsource=SearchResults

© 2015 Carnegie Mellon University

Comparison of Agile and Traditional DoD Cultural Elements$_2$

Knowledge Piece	Method	
Rewards System	**Agile DoD**	**Traditional DoD**
	• Team is focus of reward systems • Sometimes team itself recognizes individuals	• Individual is focus of the reward system
Staffing Model	**Agile DoD**	**Traditional DoD**
	• Cross-functional teams including all roles across the life cycle throughout the lifespan of the project • Includes an Agile advocate or coach who explicitly attends to the team's process	• Uses traditional life-cycle model with separate teams, particularly for development and testing • Different roles are active at different defined points in the life cycle and are not substantively involved except at those times

http://www.sei.cmu.edu/library/abstracts/reports/11tn002.cfm?DCSext.abstractsource=SearchResults

© 2015 Carnegie Mellon University

Comparison of Agile and Traditional DoD Cultural Elements[3]

Knowledge Piece	Method	
	Agile DoD	**Traditional DoD**
Communications & Decision Making	• Daily stand-up meetings • Frequent retrospectives to improve practices • Information radiators to communicate critical project information • Evocative documents to feed conversation • "Just enough" documentation, highly dependent on product context	• Top-down communication structures dominate • External regulations, policies and procedures drive the focus of work • Indirect communications, like documented activities and processes, dominate over face-to-face dialogue • Traditional, representational documents used by the PMO throughout the development life cycle to oversee the progress of the developer • PMO oversight tools focused on demonstrating compliance vs. achieving insight into progress

http://www.sei.cmu.edu/library/abstracts/reports/11tn002.cfm?DCSext.abstractsource=SearchResults

© 2015 Carnegie Mellon University

Attributes of Agile Success in DoD Organizations

Top cover

Permission to “fail fast”

Dedicated staff

Willing and open to adopt new modes of operation

Training in Agile

Use of agile coach

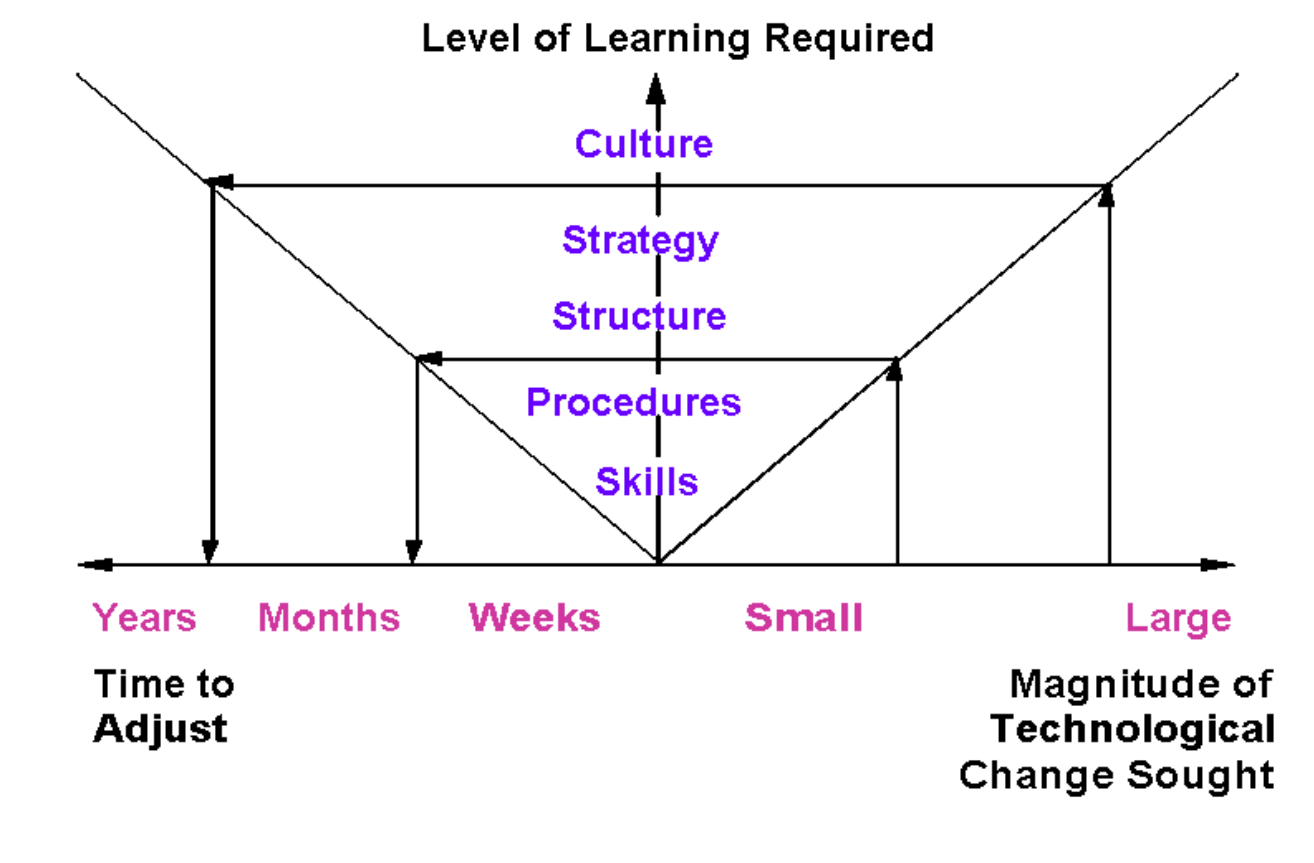
Willing to work collaboratively across government/contractor boundary

Enough up-front system and software architecture work to create sufficiently stable environments for Agile implementation teams

© 2015 Carnegie Mellon University

Polling Question

How Big a Challenge is Your Adoption of Agile Practices?



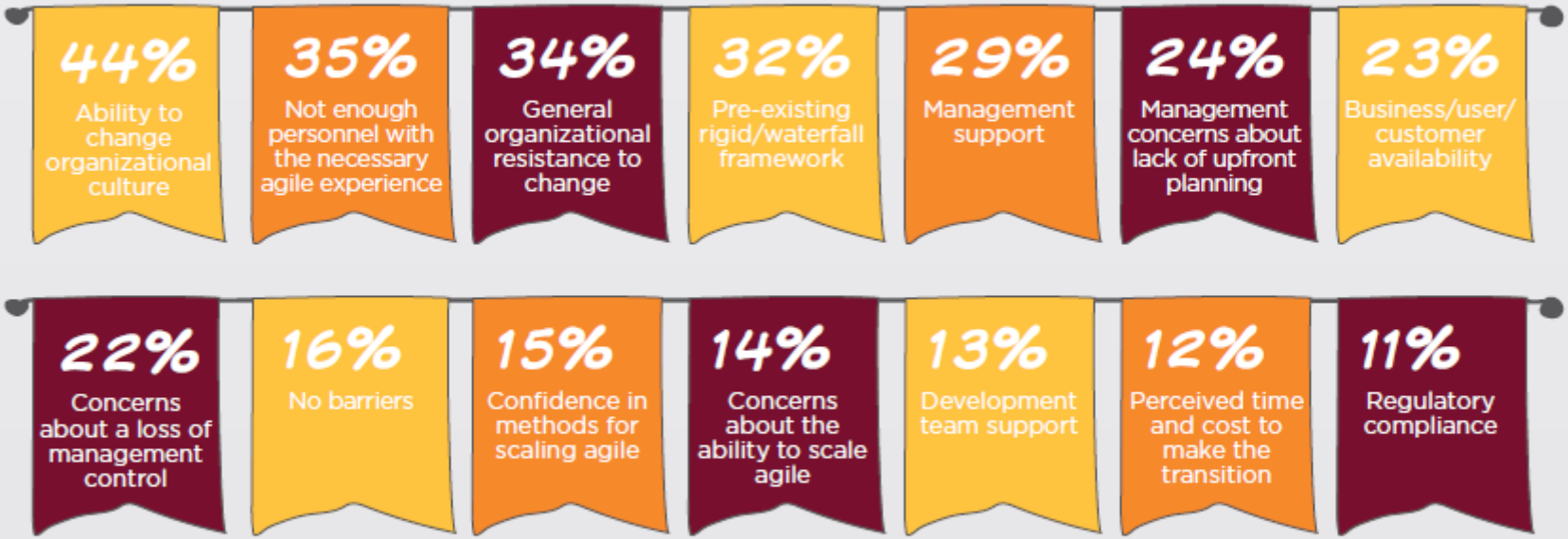
© 2015 Carnegie Mellon University

Culture Change is an Issue Throughout Industry

BARRIERS TO FURTHER AGILE ADOPTION

At the agile initiative level, respondents cited organizational culture or a general resistance to change as their biggest barriers to further agile adoption, followed by not having the right skill set.

*Respondents were able to make multiple selections.



VERSIONONE.COM
©2015 VersionOne, Inc. All rights reserved.
State of Agile is a trademark of VersionOne, Inc. and VersionOne is a registered trademark of VersionOne, Inc.

VERSIONONE Agile Made Easier

Source: 9th Annual Survey on State of Agile, Version One

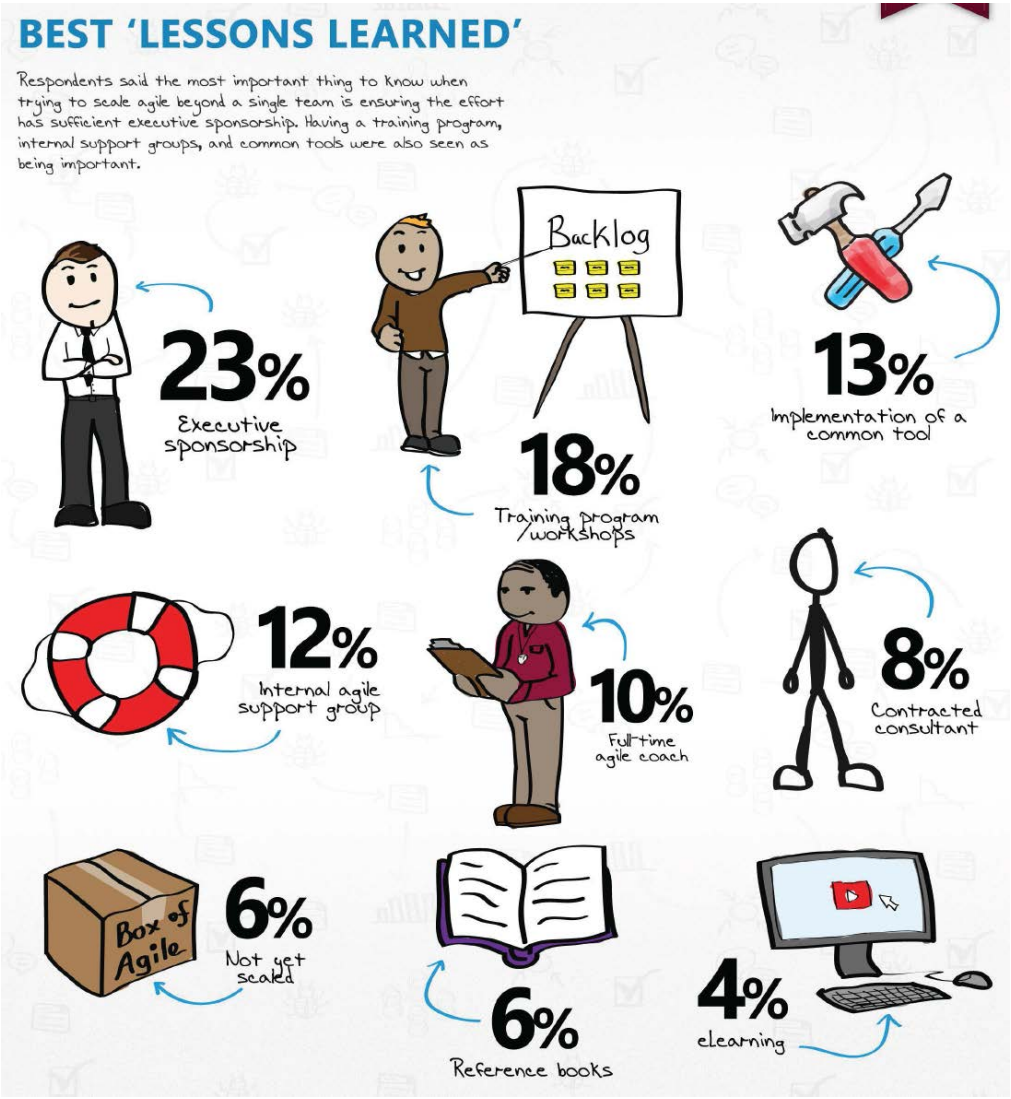
© 2015 Carnegie Mellon University

Suggestions for Successful Use of Agile Methods in Government Acquisition

© 2015 Carnegie Mellon University

Industry Lessons Learned

Many of the lessons learned in commercial industry for Agile adoption can apply in government settings



Source: 8th Annual Survey on State of Agile, Version One

© 2015 Carnegie Mellon University

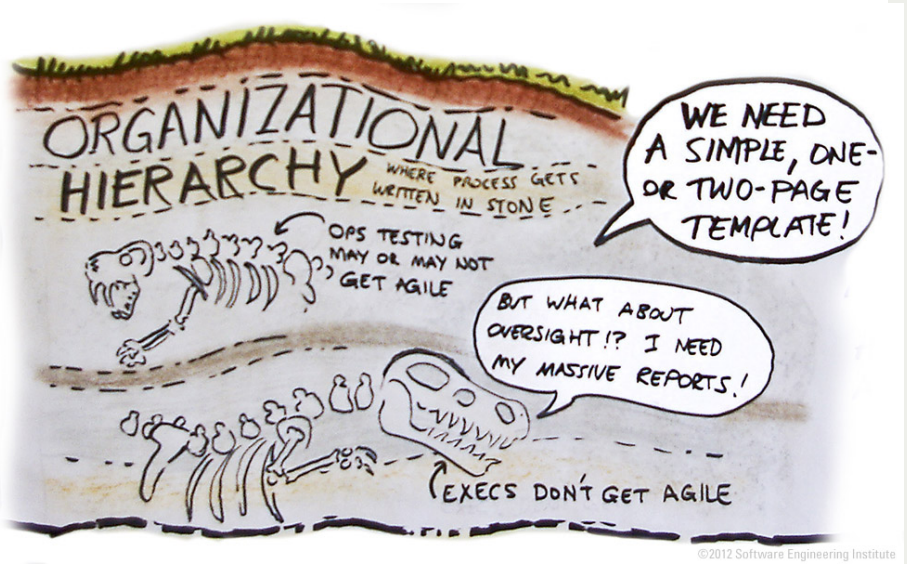
Suggesting Successful Approaches

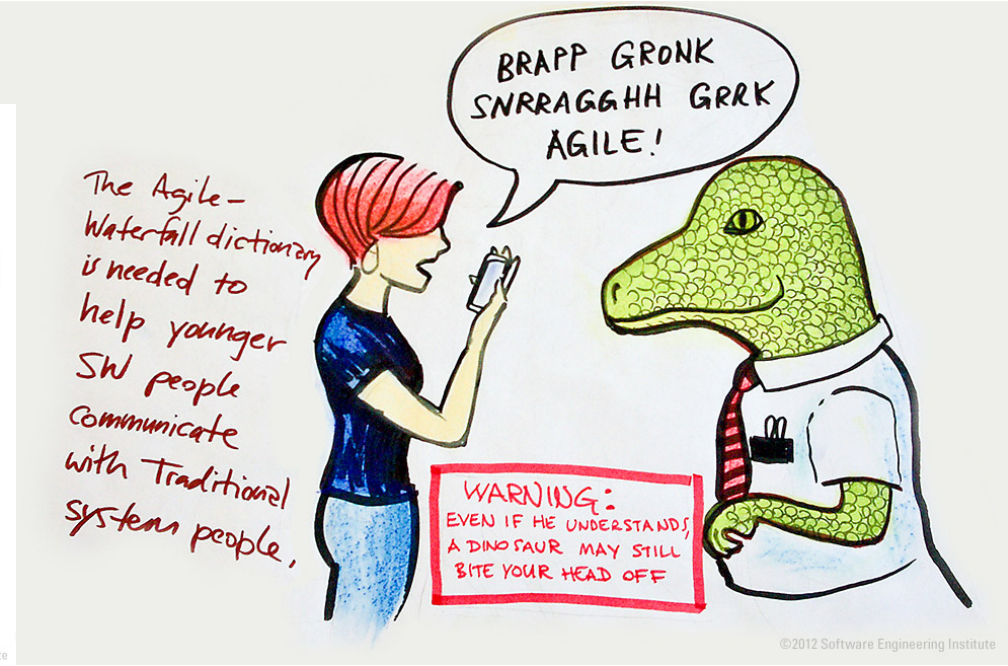
Educating leadership and staff on differences they will see

Reminding organizations of the typical challenges they face for a big change

Disseminating successful approaches when we find them

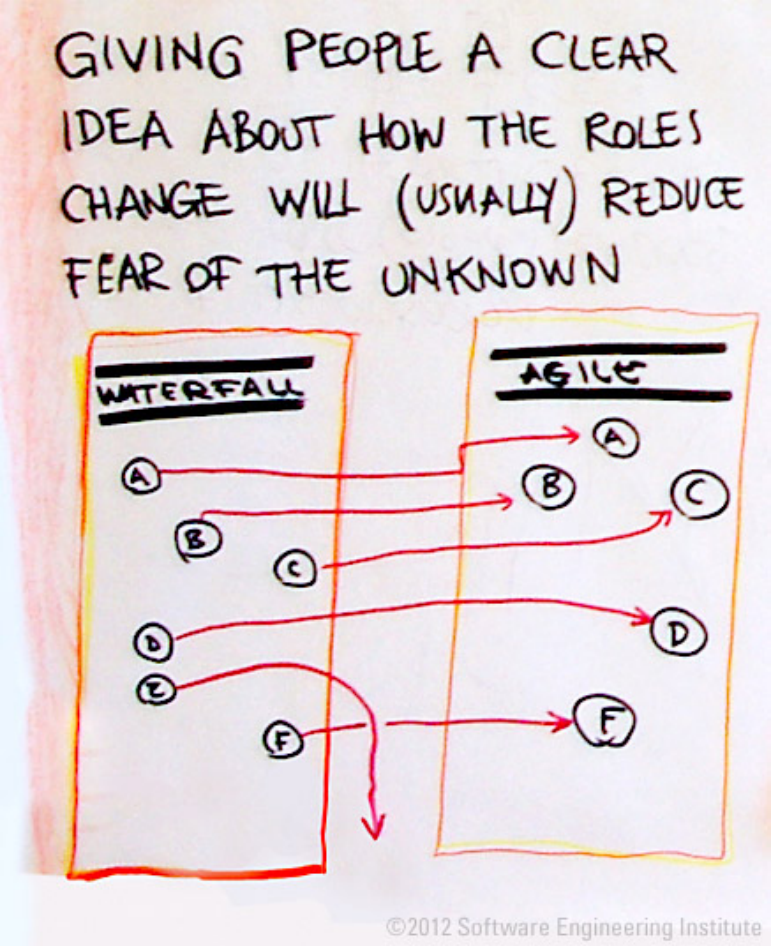
Adding in a little humor along the way…





© 2015 Carnegie Mellon University

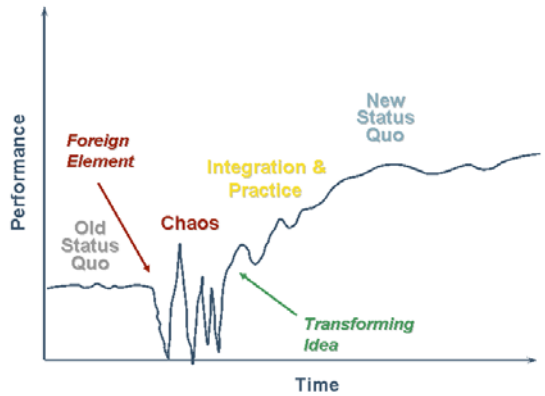
Help People Understand What's Coming

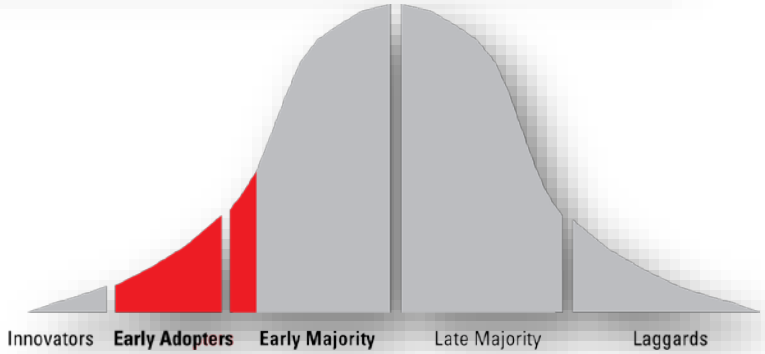


© 2015 Carnegie Mellon University

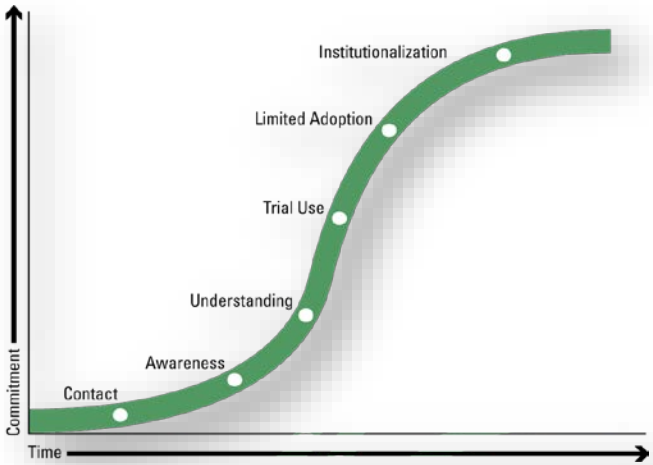
“Traditional” Adoption Tools and Methods Work Well with Agile Adoption

Understand the Change Cycle and Your Adoption Population





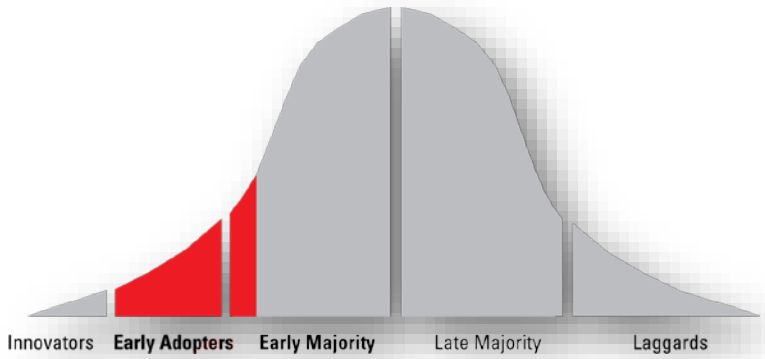
Prepare for Both Communication and Implementation Support Mechanisms that are Needed



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

© 2015 Carnegie Mellon University

Understand Your Adoption Population

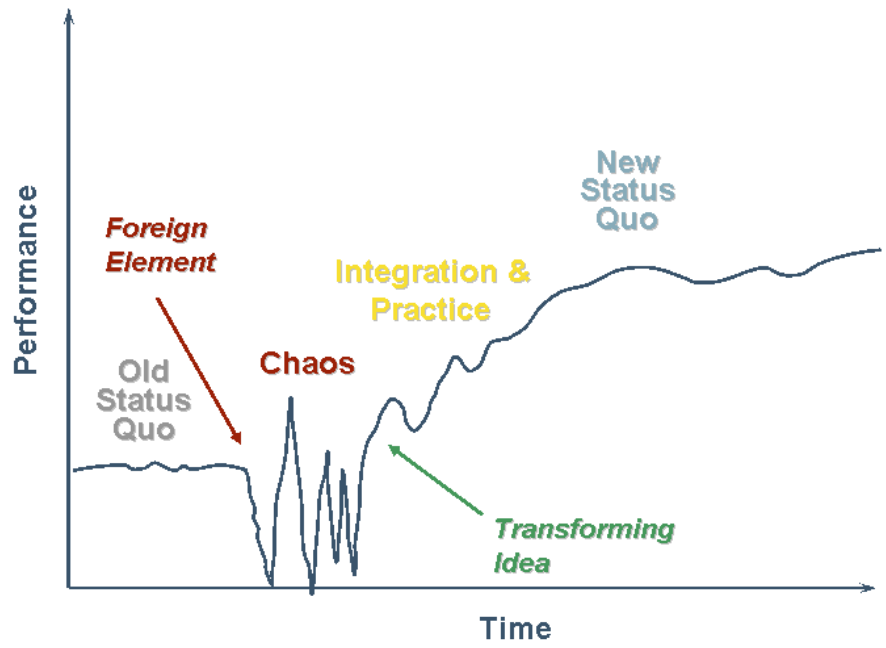


Not all changes affect an individual the same way

- Some people who are early adopters for one type of technology are laggards for another
- Innovators and Early Adopters are good candidate for “technical feasibility” pilots – answering “Will the technology work at all?”
- Early Majority and Late Majority are better for “adoption feasibility pilots”– answering “What does it take to make the technology work HERE?”

© 2015 Carnegie Mellon University

Understand the Cycle of Change for Individuals and Groups

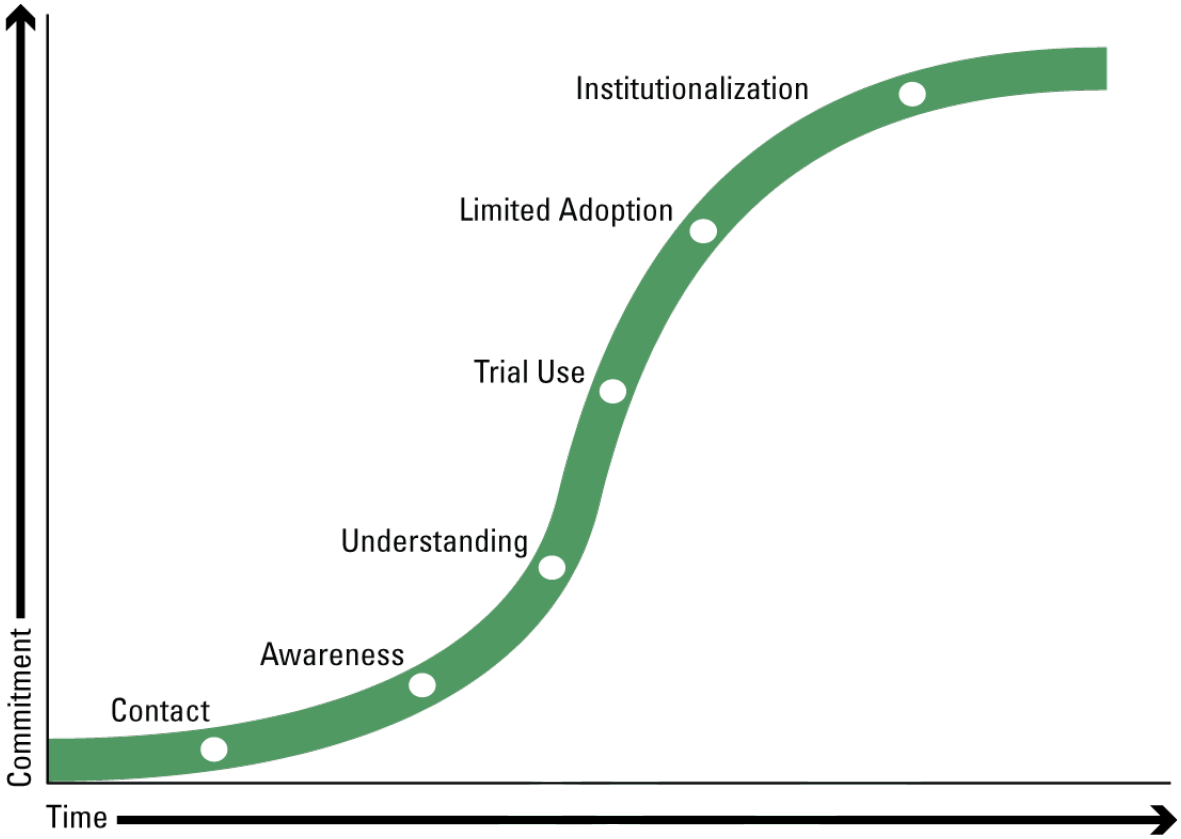


New Agile practices are your “Foreign Element”

Even after you have figured out your Transforming Idea, you still need to leave room for Integration & Practice before the desired new performance level is achieved

© 2015 Carnegie Mellon University

Prepare for Both Communication & Implementation Support Transition Mechanisms



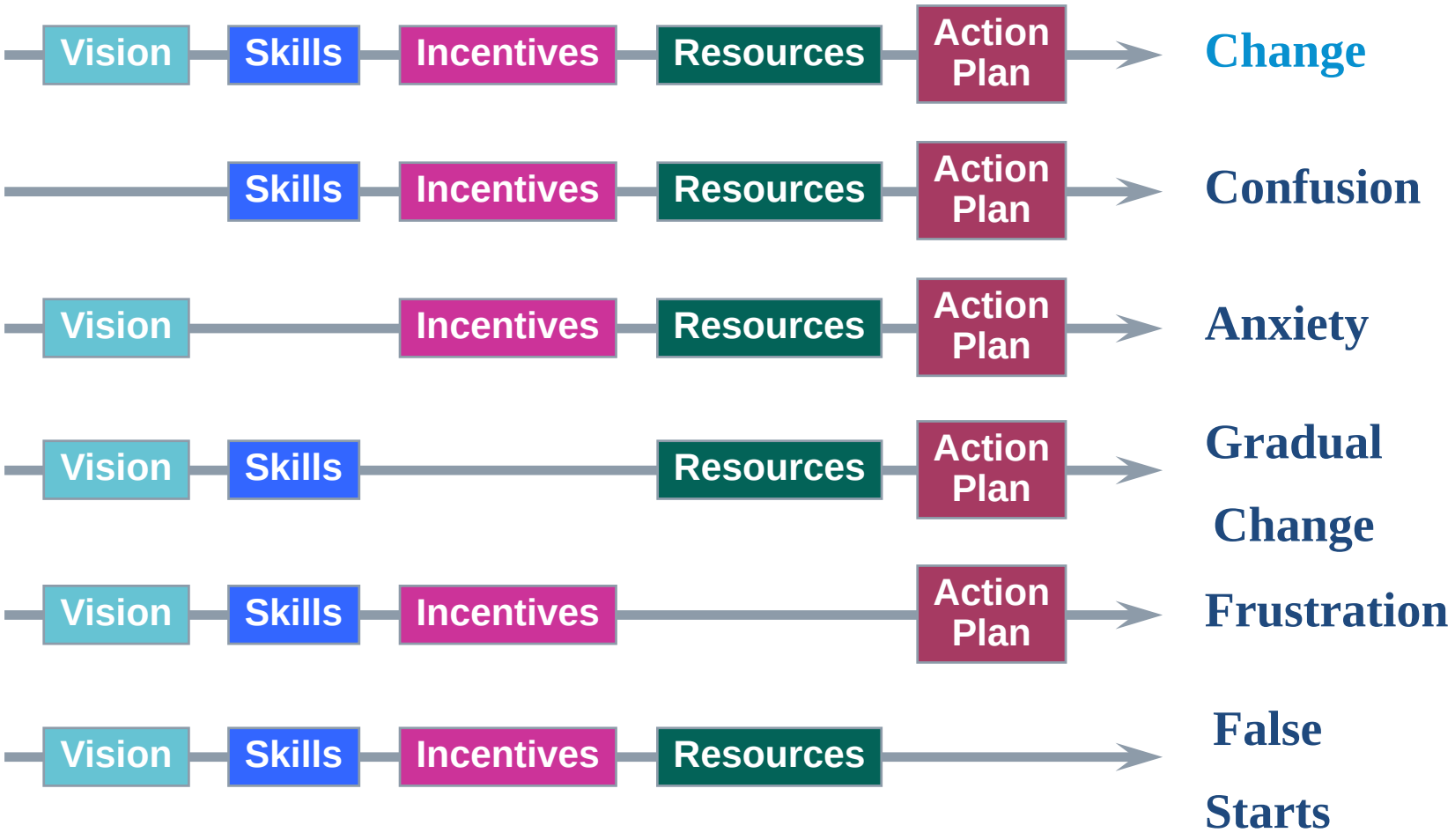
Communication mechanisms support Contact, Awareness, & Understanding

Implementation mechanisms support Trial Use, Adoption, and eventually, Institutionalization

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

© 2015 Carnegie Mellon University

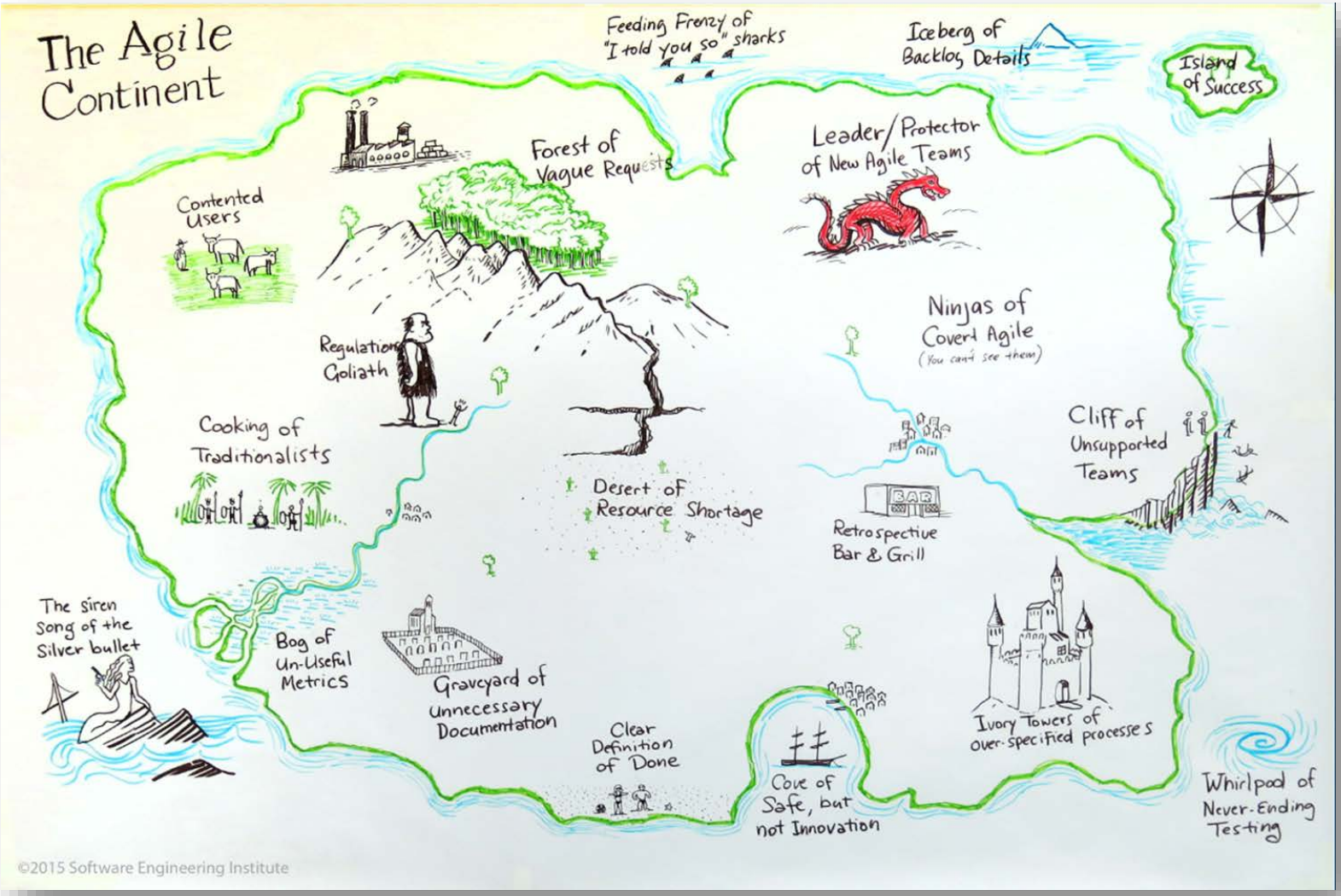
Effects of Missing Elements



Original source unknown.

© 2015 Carnegie Mellon University

Agile Adoption Landscape is Unique to Each Organizational Setting



© 2015 Carnegie Mellon University

Summary-1

Several DoD and government civil programs have successfully navigated their larger eco-systems—more are getting engaged

- Air Force Patriot Excalibur program
- Army battle management systems
- DHS Customs and Immigration
- Federal Housing Finance Administration
- Navy programs supported by SPAWAR

© 2015 Carnegie Mellon University

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

© 2015 Carnegie Mellon University

BACKUPS

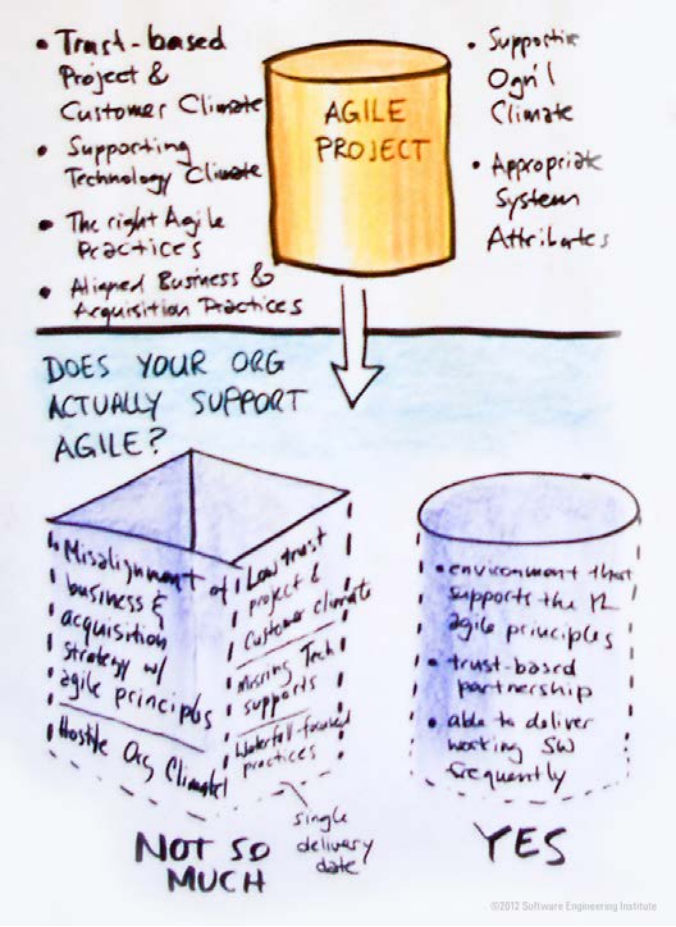
© 2015 Carnegie Mellon University

Understanding Your Readiness for Agile Adoption

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



© 2015 Carnegie Mellon University

Agile RFA Categories

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

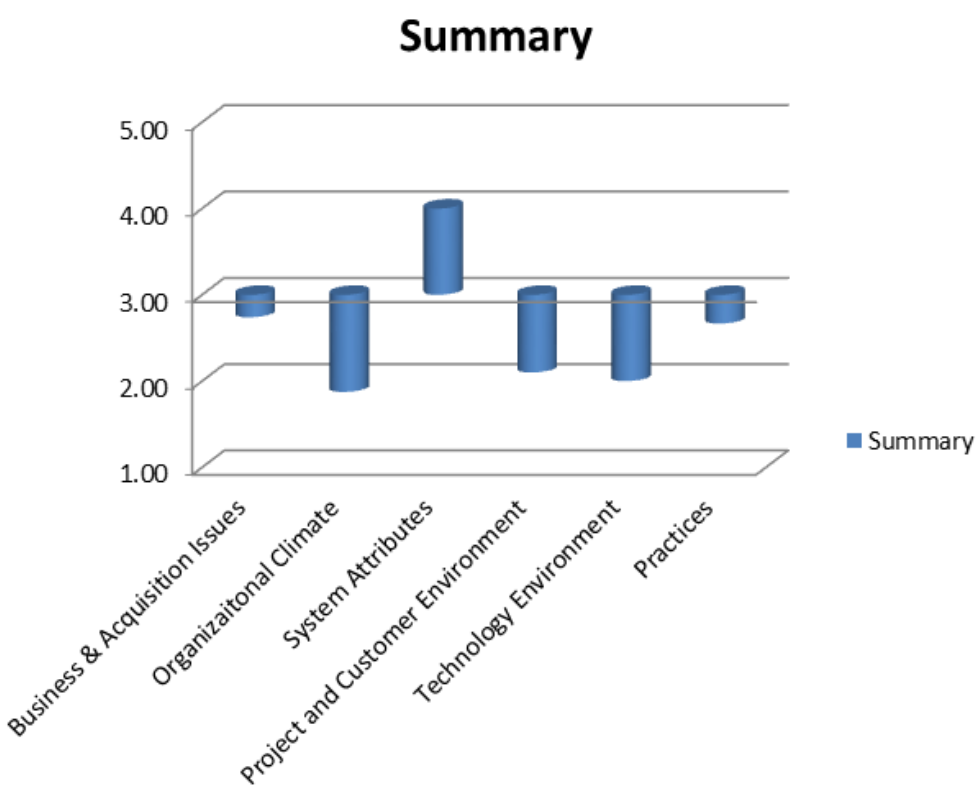
Organizational climate – adoption factors related to sponsorship, leadership, reward systems, values, and similar “soft” issues

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.

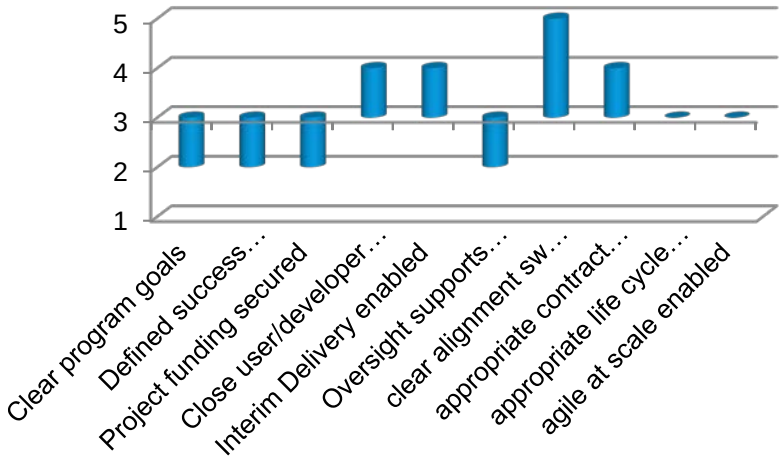
More important than the rating is the specific risks that RFA participants identify.

© 2015 Carnegie Mellon University

Business and Acquisition Category Particularly Affects DoD Agile Adoption

- 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

Business/Acquisition Factors



- 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

© 2015 Carnegie Mellon University

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:

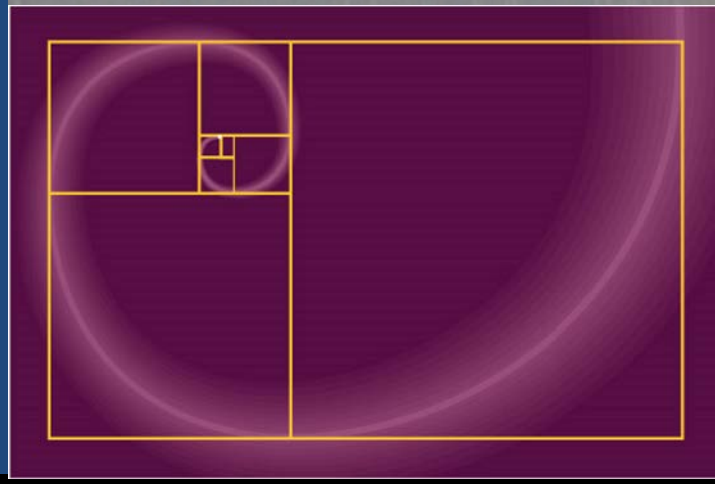
- Was able to identify the root cause of some of their adoption issues
- Changed how they communicated some of their progress to management
  - Improved management’s ability to understand both the opportunities and risks that Agile presented to the program

© 2015 Carnegie Mellon University

Agile in Government: Practical Considerations

CTSD Agile Adoption in Regulated Settings Team
Software Engineering Institute
July 2015

Module 7 (Summary)



© 2015 Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

This material has been approved for public release and unlimited distribution except as restricted below.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study. Except for the U.S. government purposes described below, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose this material are restricted by the Rights in Technical Data-Noncommercial Items clauses (DFAR 252-227.7013 and DFAR 252-227.7013 Alternate I) contained in the above identified contract. Any reproduction of this material or portions thereof marked with this legend must also reproduce the disclaimers contained on this slide.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

This material was prepared for the exclusive use of NRO Agile in Government Workshop and may not be used for any other purpose without the written consent of permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0002617

Contents

Module 1: Introduction

Module 2: Basic Agile Concepts

- Agile Introduction
- Agile Tenets and Principles
- Differences Between Traditional and Agile
- Common Agile Concepts and Practices
- Agile Mythbusting

Module 3: New Role for PMO

- Why Agile
- The Product Owner Role

© 2015 Carnegie Mellon University

Contents (continued)

Module 4: Agile Program Oversight

- System Engineering
- Integration and Test
- Technical Reviews
- Measurement

Module 5: Agile in the Larger Eco-System

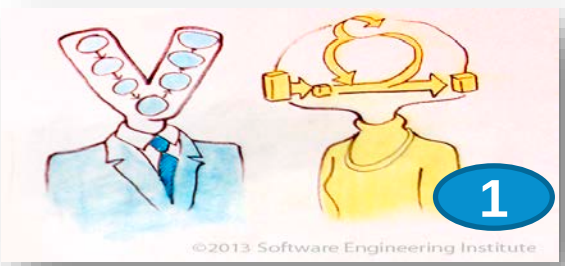
- Documentation
- Contracting
- Regulatory Oversight

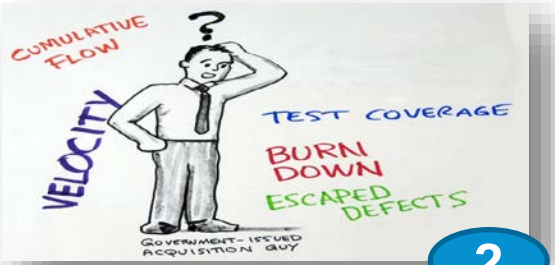
Module 6: Enabling an Agile Acquisition Culture

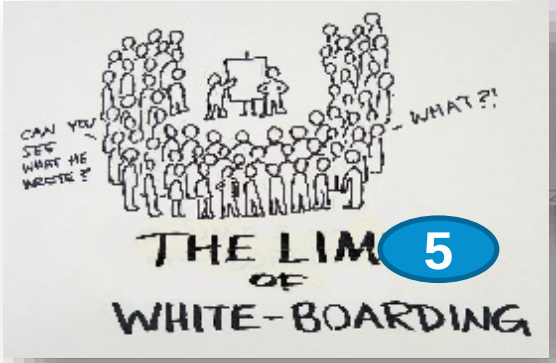
- Enabling and Transitioning to an Agile Culture
- Suggestions for Implementing Agile in Government

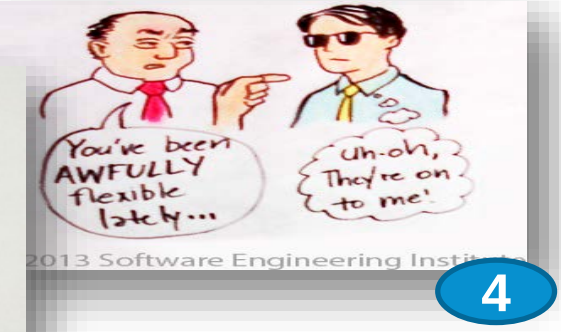
Module 7: Summary

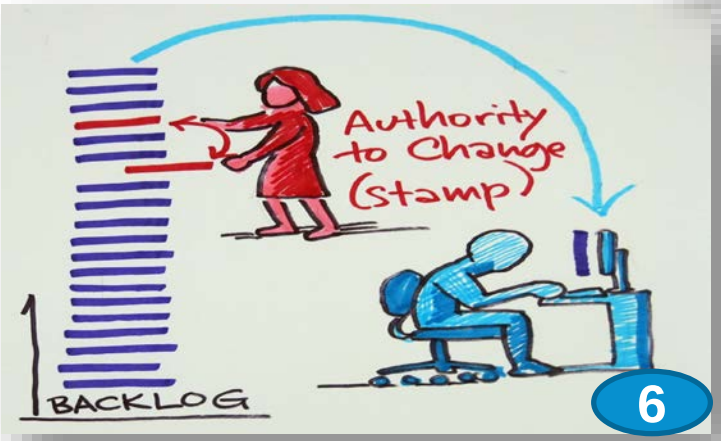
© 2015 Carnegie Mellon University











© 2015 Carnegie Mellon University

Image Quiz

At your tables, for each image:

- List the challenge that the picture represents
- List one or two strategies for addressing that challenge

Be prepared to share your insights with the class.

© 2015 Carnegie Mellon University

Review of YOUR Learning Objectives

Discussion:

- Look at the flip charts of your desired learnings from the beginning of the course
- Instructor will review them with class as a whole
  - For learning objectives we couldn't address in this course, instructor will try to provide references to other resources to fill that knowledge gap

© 2015 Carnegie Mellon University

Contact Information

Mary Ann Lapham
Principal Engineer, CTSD
Telephone: +1 412-268-9143
Email: mlapham@sei.cmu.edu

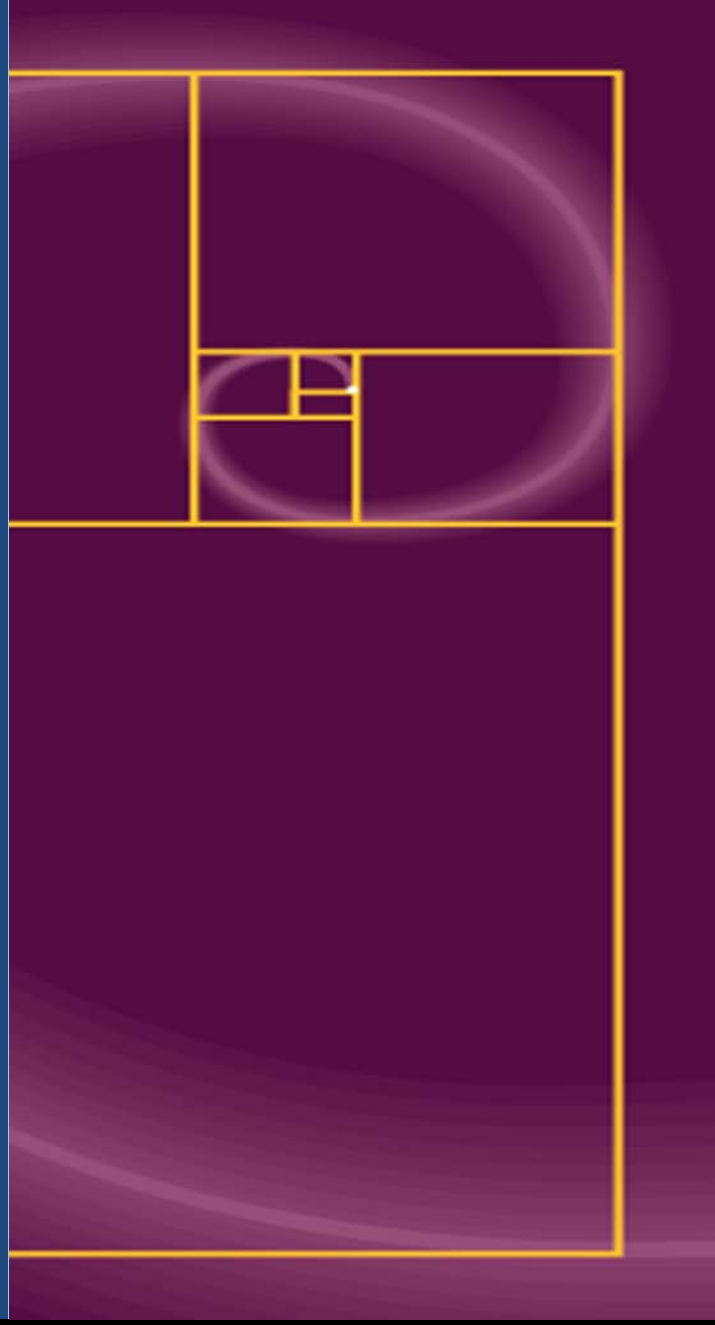
U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

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

© 2015 Carnegie Mellon University

THANK YOU!!!



© 2014 Carnegie Mellon University