One Hour Tip of the Iceberg Overview: Requirements in an Agile Government Setting

For F-35 Training Systems PMO

Suzanne Miller
Crisanne Nolan

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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

DM21-0565

Assumptions

You have attended an Agile overview of some sort

You want to know how working with requirements will be different in an Agile government setting

You are working with contractors who design and implement based on the requirements you provide to them

Target Audience

- **Training Systems PMO Leadership who will be asked to change the way they write, manage, and accept the implementation of requirements to support iterative, incremental delivery using Agile and Lean methods**

Expectations/Goals for this Overview

You will Learn:

- Concepts of Agile Requirements Expression and Management in an Agile Government setting

You will NOT Learn:

- Detailed procedures and techniques for writing and managing requirements in an Agile Government setting
- Day to Day processes for managing a contractor deriving from and implementing requirements in an Agile contracted setting

Agenda

- Reminders: General Agile Approach Considerations
- How Dealing with Requirements Changes When Moving from Big Batch to Iterative, Incremental Approaches
- Governance and Role Considerations

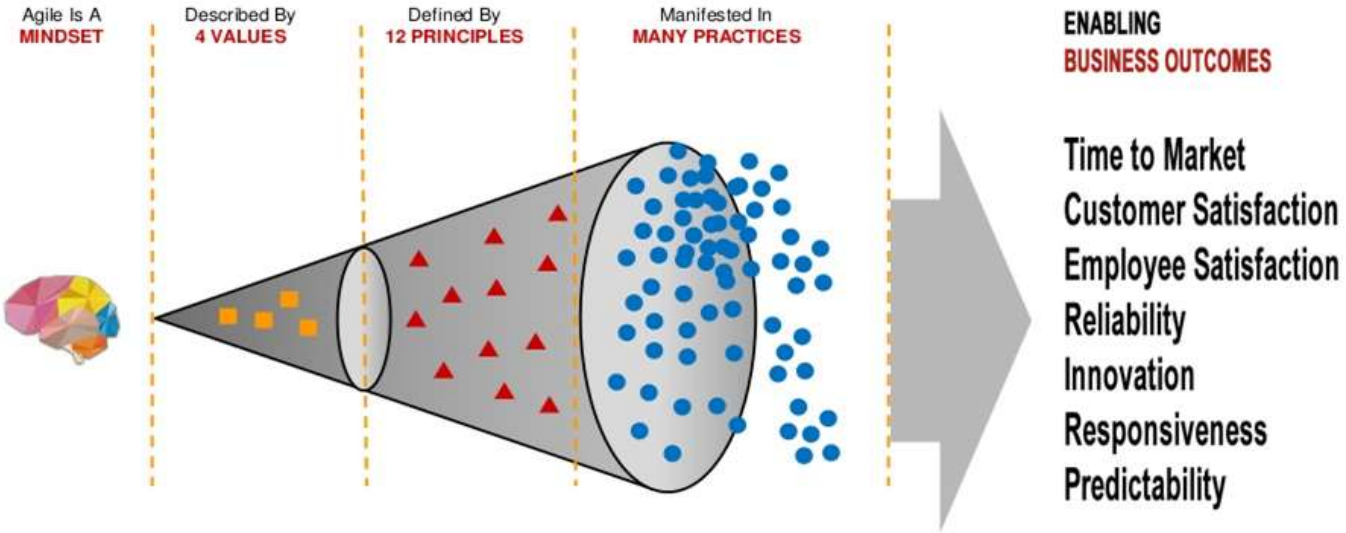
BLUF

Moving from big batch requirements development and management to iterative, incremental (Agile) requirements and development:

- Requires shifts in how we think about requirements
  - FROM “must be perfect” TO “must be perfect enough to produce the next, most valuable element”
- Requires shifts in how we manage requirements
  - FROM reviewing a large requirements document once and then preventing change to the extent possible, TO reviewing small batches of requirements throughout the development, allowing change based on learning as a natural part of the process

Reminders: Agile Approach Considerations

What is Agile?



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

It isn't enough to adopt the practices of a successful team. You must adopt attitudes and a mindset for making decisions to adopt practices that will lead to your success.

Implementing the practices, tools and processes **without** the Agile mindset, values, and principles of the Agile Manifesto

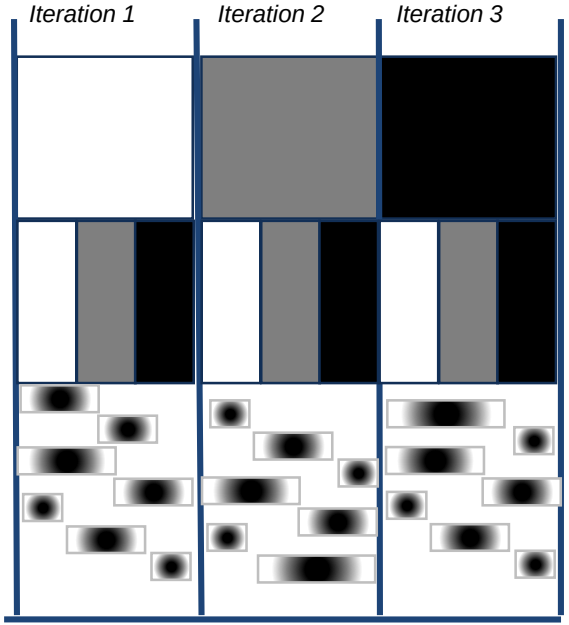
Is NOT Agile!

Taking an Iterative Approach

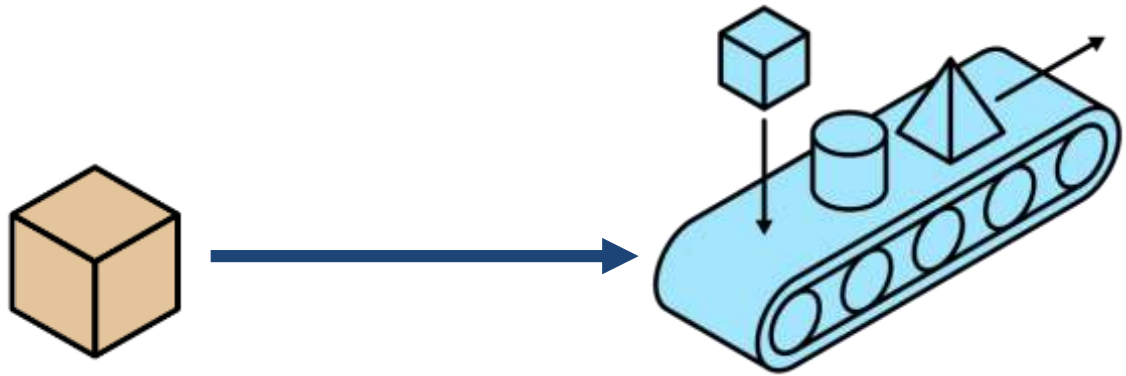
Single batch – one process step per iteration

Multiple batches - complete each batch at the end of an iteration; siloed process steps within each iteration

Multiple really small batches - decompose each batch into small packages, with multiple start-to-finish cycles in each iteration



Move from Buying a Box to Buying an Ongoing Delivery Stream

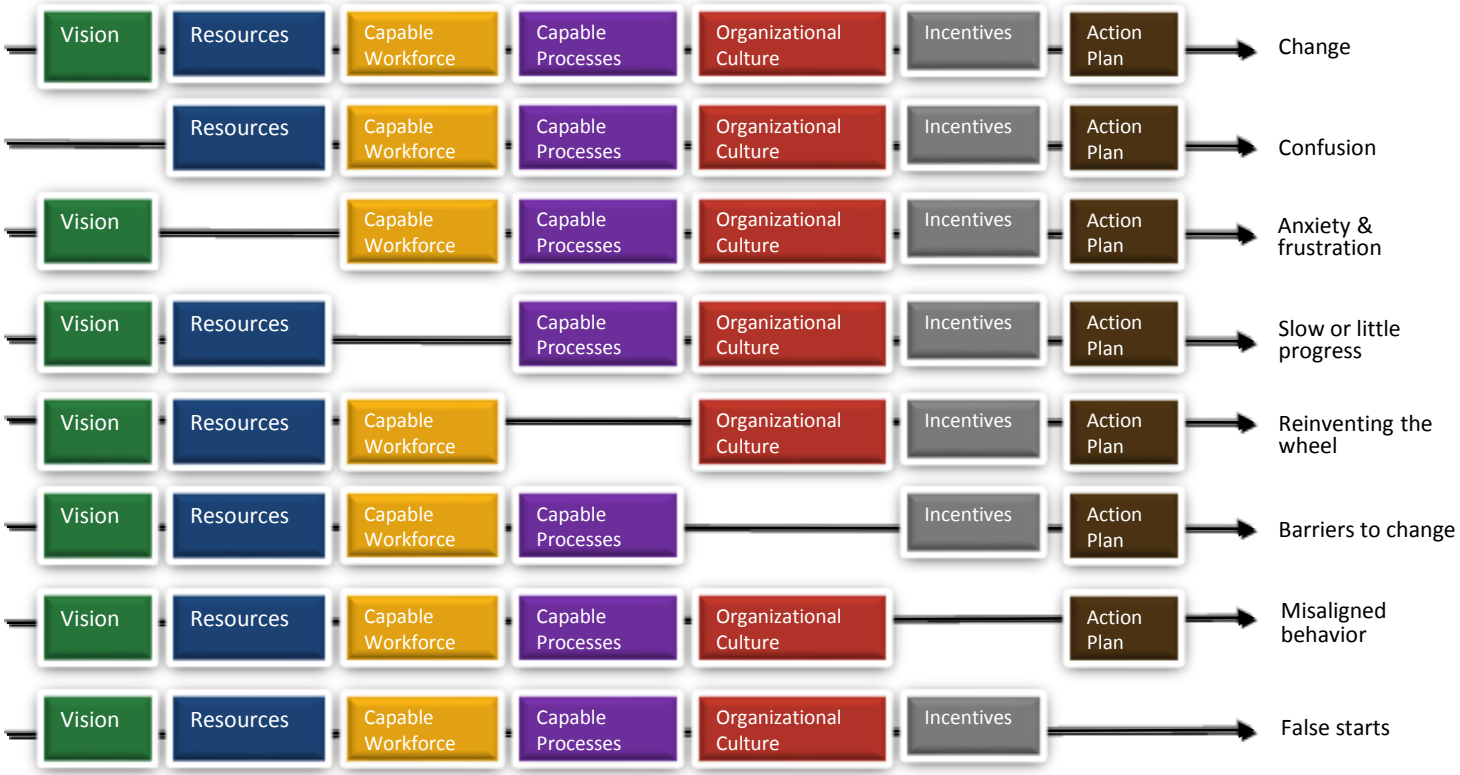


Buying a Box

Buying an ongoing delivery stream

Because delivery is ongoing, someone from the PMO needs to continually be involved in framing the deliveries needed and prioritizing development items from the customer/user perspective.

Why is Agile Adoption Challenging?



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

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

What Changes to be Successful with Agile Requirements?

Different Roles, Focus, Terminology….1

Key Terminology:

- **Product Backlog Item**: the umbrella term for the various types of requirements at different levels of abstraction
  - Epics: requirements that will take more than 3 months to complete and likely to take multiple teams
  - Features: requirements that will take one team 3 months or less to complete (assuming 3 month Increments)
  - Stories: requirements that will take one team between 4 hours and 40 hours to complete (assuming 2 week iterations)
- **Backlog:** container for the Product Backlog Items; multiple backlogs, each at a different level of abstraction
  - **Epic/Capability backlog**: the likely CDD or SOW top level requirements; possible allocated baseline
  - **Feature or Program Increments backlog:** the likely HW/SW Reqmts specification-level requirements; possible allocated baseline
  - **Team or Iteration Backlog: lowest level product backlog**; usually derived requirements below the allocated baseline

Different Roles, Focus, Terminology….2

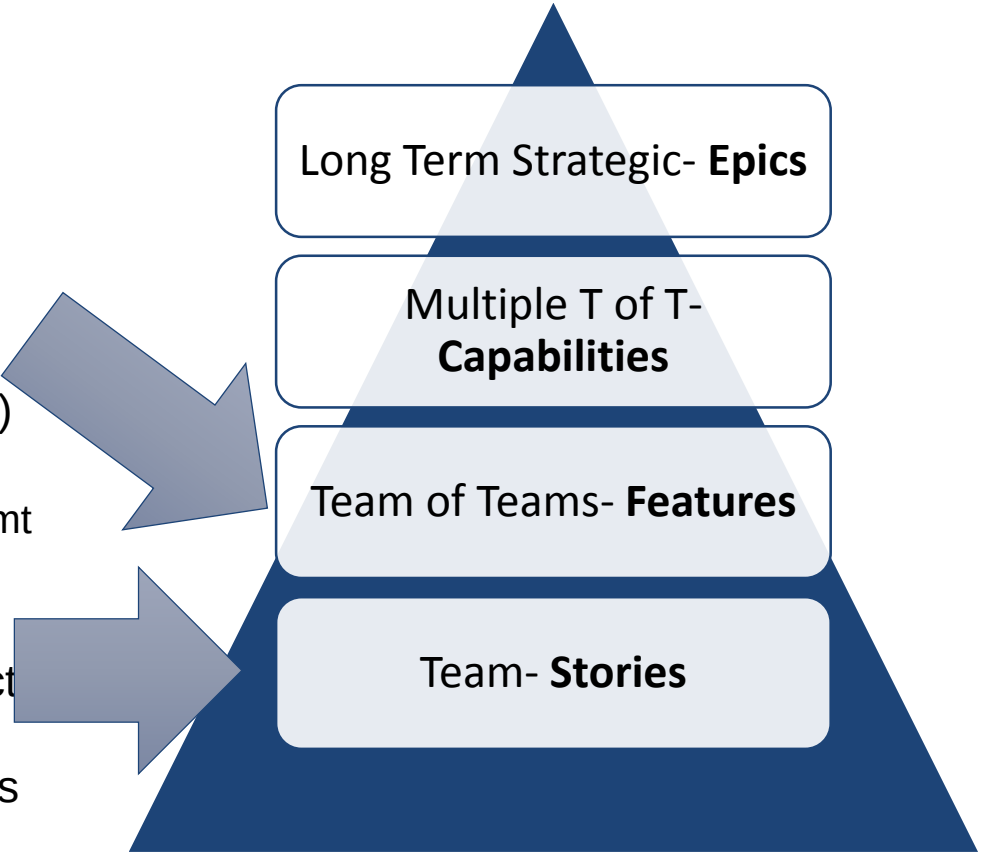
Key Terminology:

- **Product Manager:** The role responsible for establishing and accepting the 3-month level of product backlog content and the roadmaps that lay out the longer term goals; often a government/prime contractor joint role
- **Product Owner:** The role responsible for establishing and accepting the iteration (typically 2 weeks long) level of product backlog content and ensuring integration of that content with the Feature level implementations; could be either government or contractor role, depending on product context
- **Acceptance Criteria:** The criteria used at iteration and program increment level to ensure that the implementation meets the need stated in the backlog item. Specific to a particular backlog item
- **Definition of Done:** The criteria used at each level of the backlog to ensure that the quality attributes (e.g. security, safety, usability) and programmatic attributes (e.g., documentation, model artifacts) are present when implementations are delivered

Focus for this Overview

We are focusing on two levels of activity related to requirements:

- Team of teams level – the requirements that 50-125 people broken up into multiple teams of 6-10 work on over the course of a 3 month planning/execution cycle (often called a Program Increment)
  - One of the likely levels of SOW requirements, and likely level of SW Reqmt Specification artifacts
- Team level—the (typically) derived requirements that a team of 6-10 product developers and testers produce on a short iterative cadence (typically 2 weeks or less)



The Backlog is the “Container” for Requirements in Lean/Agile Settings

Backlog Attributes:

- **List format typically organized by rank-order (not priority category!)**
  - **There cannot be two “Number One” priorities**
- **Expected only to be complete enough for people close to the work to plan and specify implementation**
- **Abstraction level at which backlog items are specified is tied to which layer of the team structure they relate to**
  - **“feature” level items are often allocated baseline types of items**
  - **“story” level items are usually derived requirements level and wouldn’t require ECPs for change**
- **Story level items often include both “who” and “why” in the description of the item**

Requirements Specification Attributes:

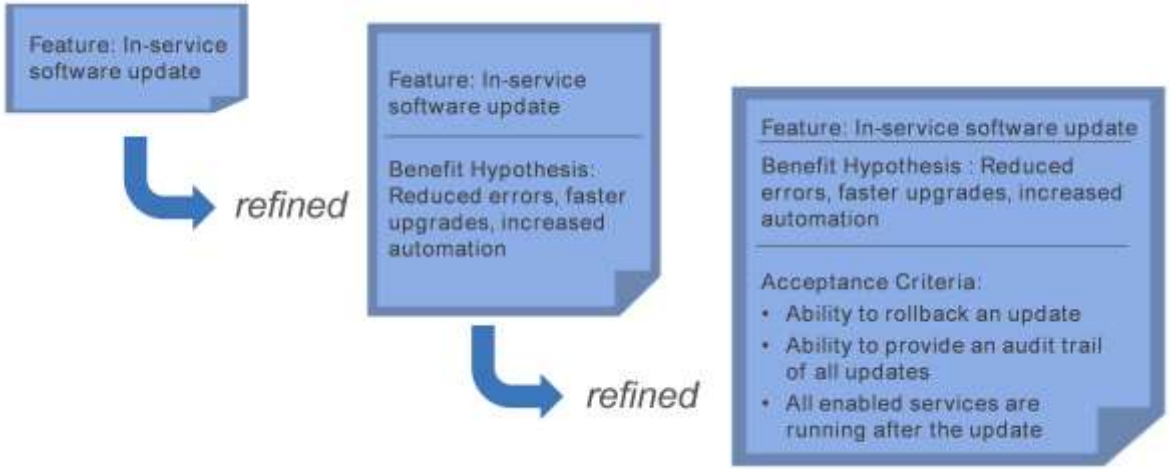
- Typically organized by topic or by Product-based Work Breakdown Structure
- Expected to be “complete” before implementation begins
- Level at which requirements are specified is usually a baseline that requires configuration control permission to change (SRS or HRS typical)
- “Who” needs the “what” of the requirement and “why” they need it not typically included at any level

Analysis and refinement ensure Features and Stories are ready for implementation

This refinement is the "bread and butter" of Prod Mgr activity in preparation for Quarterly PI Planning

Features may start as a one-sentence overview with more details added in PI Planning and backlog refinement meetings.

Stories are more detailed, and add "who" and "why" to typical "what" requirements (see section on Stories)



Example in difference of expression of Stories vs Specifications

Typical backlog expression of a team-level story:

- As an operator of the xxx component, I need to control access to the yyy interface to ensure that all security parameters have been satisfied prior to transmitting data across the interface.
- Who=operator of xxx
- Why=ensuring security parameters are satisfied

Typical requirements expression of a team-level derived requirement (note—this is typically below the level of the allocated baseline):

- The xxx component shall control access to the yyy interface
- Who needs this? Not specified
- Why do they need it? Not specified

Governance and Role Considerations

Agile Motto for the Backlog Process

Who makes the backlog? And who ensures that what is in the backlog is prioritized? And who ensures that the committed backlog items meet expectations of the customer?

Product Owners and Managers!

Differences Between ProDUCT Mgr & ProGRAM Mgr

Both roles are present in most Lean/Agile government programs

Some individuals have successfully combined both roles

- Usually in smaller settings where the demands of the content (product ownership) are not that large or challenging

Once through a transition from large batch to small batch, staffing may not be that different a level

- Often see a need for more staffing during transition while there are both large and small batch tasks to accomplish

What Does This Mean for Writing Requirements for a Contract?

START

- Preparing staff to work in small batches of requirements on a continuing basis vs large batches of requirements on an infrequent basis
- Establishing governance that expects relevant change, and supports ongoing learning through implementation vs specification perfection
- Establishing contract mechanisms that are shorter delivery cycles for ongoing capability delivery and allow for pivoting as needed as learning occurs
- Framing requirements by adding in “who needs it” and “why do they need this”

STOP

- Considering the requirements to be “perfect” when they are send out in a SOW
- Governing in a way that expects “nothing is done until everything is done”
  - Incremental delivery is the best way to learn what needs to be delivered next
  - “ongoing capability delivery” is a tenet of DoD 5000.87
- Establishing long “buy a box” types of contracts
- Framing requirements just as a “what” without context

Other Governance Considerations

Governance in relation to Contractor

Key to establishing requirements governance with contractors in an Agile setting is getting the Allocated Baseline at the right level of abstraction:

- Requirements at too low a level means that there will be LOTS of ECPs and the overhead that goes along with that
- Requirements at too high a level means that there may too little understanding of what the problem to be solved really is

Leveraging contractor activities like Increment Planning and System Demos provides the insights government stakeholders need to guide the requirements evolution in the right direction

Governance in relation to PMO, Users, and Stakeholders

Keeping the “longer term” requirements at a higher level means that the lower level conversations need to happen with relevant stakeholders in a timely fashion

More engagement with users and stakeholders like security and safety means their time has to be factored in to events

- Things like quarterly Stakeholder Workshops and Office Hours mechanisms have been used successfully elsewhere in the program

Summary

You are Moving Forward on Your Lean/Agile Journey!

Incremental, iterative ways of working are here to stay in DoD and in industry

Small batch work means consistently paying attention to the product content

Contracts shift from “buying a box” to “ongoing capability delivery”

- Adjust our thinking to optimize for that situation

A Few Relevant Resources

SMC Agile/Lean Self-Service Learning Paths Catalog

- Some of the materials in the learning package came from this site. If you have a CAC, you can access a large set of curated materials on your own

https://www.milsuite.mil/wiki/Portal:AtlasX_Agile_Self-serve_Learning_Paths

Software Engineering Institute Agile Resources: podcasts, blog posts, Technical Notes on many Agile in Government topics (including Mission-based Prioritization!):

https://www.sei.cmu.edu/research-capabilities/all-work/display.cfm?customel_datapageid_4050=21345

Contact Information

SuZ Miller, Principal Researcher
Software Engineering Institute

smg@sei.cmu.edu

Crisanne Nolan
Software Engineering Institute

ccampus@sei.cmu.edu

Backup Slides

Multiple Ways to Engage in Product Ownership in Government Programs

Team Level Engagement—The Product OWNER

Work with teams of 6-10 developers/testers; teams could be using any or combinations of these in most software-intensive Programs

- Scrum
- XP
- Kanban

All of these methods have the following in common:

- A prioritized backlog of items that can be built within a defined timebox
- A role that is focused on
  - Framing the requirements in a way that developers can effectively estimate and execute the work
  - Prioritizing the work of the team to optimize results for the end user/customer
  - Answering developer questions or finding answers for developer questions that relate to the end user intent
  - Evaluating the work of the team for its acceptance to the end user/customer
  - Coordinating their team's technical work with other related team

Roadmaps Become a Focus: Link Strategy to Tactics

Some kind of contractual specification document often provides the Roadmap focus for Product Managers

…according to SAFe

Attributes of Good Government Product OWNERS (Team Level)

Understand the vision and roadmap for their portion of the system—often just software or hardware

Collaborative mindset working with program office, contractor and suppliers—not a blame game when (not if!) something unexpected happens

Capable of communicating effectively with engineering and management and operational staff, especially individual developers or operators

Analytic mindset capable of rank-order prioritization and rationale for their decisions

Effective in suggesting feasible risk mitigations as risks emerge

Empowered to make decisions for evolution of the product in the **2 week to 3 month** month time horizon

Capable of evaluating if Acceptance Criteria for their Stories have been met

What Traditional Staff Roles Are Candidates for Product OWNER Role?

Engineers or Program Management staff who are experienced

- Writing and analyzing requirements
- Engendering trust among developers and users
- Prioritizing work in the 2 week to 3 month time horizon
- Analyzing, communicating, and mitigating technical risks
- Communicating with end user and other important stakeholder communities about their requirements/desirements
- Evaluating and verifying a team’s work

Attributes of Good Government Product MANAGERS (Quarterly PI Planning Level/Feature Level)

Understand the vision for their portion of the system—typically more than software

Collaborative mindset working with program office, contractor and suppliers—not a blame game when (not if!) something unexpected happens

Capable of communicating effectively with engineering and management and operational staff from Chief Engineer to individual developers or operators

Analytic mindset capable of rank-order prioritization and rationale for their decisions

Effective in suggesting feasible risk mitigations as risks emerge

Sufficient authority/rank to be invited to meetings where roadmaps, architectures, and programmatic decisions are under discussion

Empowered to make decisions for evolution of the product in the **3-6 month** time horizon

Capable of evaluating if Definition of Done for their product elements have been met

What Traditional Staff Roles Are Candidates for Product Manager Role?

Systems engineers or program management staff who are experienced

- Writing and analyzing requirements
- Creating product roadmaps
- Prioritizing work in the 3-6 month horizon
- Analyzing, communicating, and mitigating technical and programmatic risks
- Communicating with end user and other important stakeholder communities about their requirements/desirements

Program Managers or Deputy Program Managers who understand that the ProDUCT Manager role is about managing product content, not managing the people doing the work

.

User and Enabler Stories as Backlog Items

Building a Team-Level Initial Backlog

Take a set of objectives

Bin them in “will take more than 3 months”/”will take less than 3 months”—those are Features

For the “less than 3 months” Features

- Break down into items that take between 4 hours and 20 hours to accomplish by one person
- Those are your starter “stories” or iteration product backlog items (PBIs)

For the “more than 3 months” items, those are Epics

- Break those down into “Features” that would demonstrate something meaningful but will take 3 months or less by a team of 10 or fewer people

Stories - Including the “Who” and “Why” of a Requirement

User Stories

Expresses concepts in a way operational user would find useful

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

Template: *Typically free-form; should incorporate the “why”*

Key Points in Writing Stories

- Keep stories short and expressed in business/operator language
- 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

Writing/Using Acceptance Criteria

Acceptance Criteria

Difficult to write good acceptance criteria without getting into designing solutions.

- Requires practice
- Take perspective of the system user
- Let the developer do their best work

Bad Acceptance Criteria

Constrain design unnecessarily

(e.g., Design is stated in acceptance criteria)

May compromise architecture

(e.g., the criteria calls for use of .NET when the system architecture is J2EE-based)

Make maintenance more difficult

(e.g., References another document)

Limit searching for good solutions

(e.g., Narrows down to one design solution)

Contributing to Definition of Done

#6: Definition of Done (DOD)

What is the Definition of Done (DOD)?

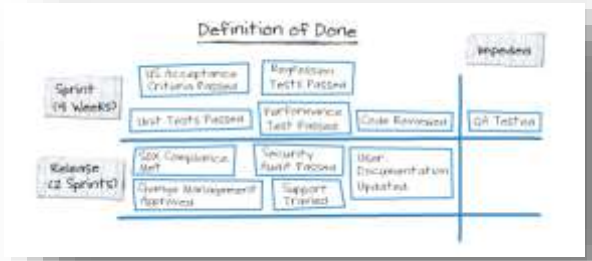
Definition of Done is an explicit declaration of the completion criteria for some aspect of an agile life cycle. DOD can be applied to an individual artifact (e.g., a user story), a sprint (as a companion to the **Sprint Goal**), or a release.

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 life cycle 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**.