# Agile in Government: Requirements Strategies

April 2017
AFEI Agile in Government Summit

Suzanne Miller
Eileen Wrubel
Will Hayes

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA  15213

Software Engineering Institute | Carnegie Mellon University

DM17-0128

# Topics

***Expressing and Governing Requirements in Agile Government Settings***

***Strategies to Think About***

***Summary***

# What Questions Do You Have About Requirements in an Agile Setting?

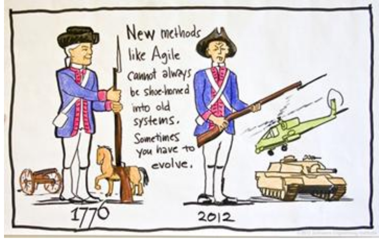
Create a sticky note for each question and post to specified flipchart

We won’t have time to address in 30 minutes, but your questions will help to inform the evolution of our Agile in Government training products related to requirements definition and management.

THANK YOU!

# Expressing and Governing Requirements in Agile Settings

# Requirements Challenges: Agile in Regulated  Settings


New methods like Agile cannot always be shoe-horned into old systems. Sometimes you have to evolve.
1776
2012

Governance Guidance from Policy Makers

- Often hints at, but stops short of encouraging Agile (e.g, DoD 5000.02)

Translation of Requirements Progress Measures

- Traditional view that 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

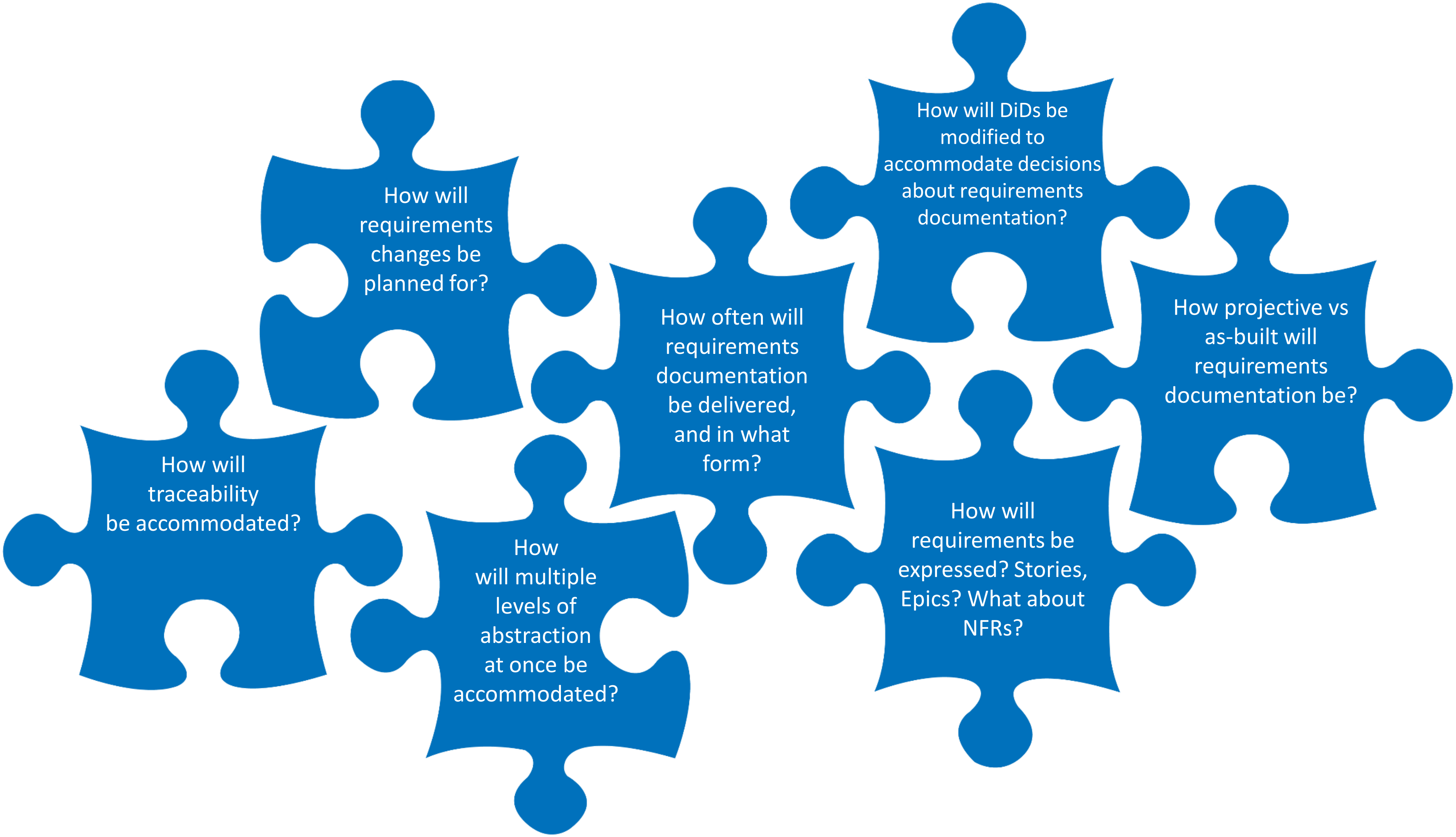
- Especially HW-centric WBS can result in software requirements at too low a level of detail

Effect of Requirements Changes on Contracts

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

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.

# Multiple Related Requirements Considerations for Acquisitions Using Agile Contractors



How will traceability be accommodated?

How will requirements changes be planned for?

How will multiple levels of abstraction at once be accommodated?

How often will requirements documentation be delivered, and in what form?

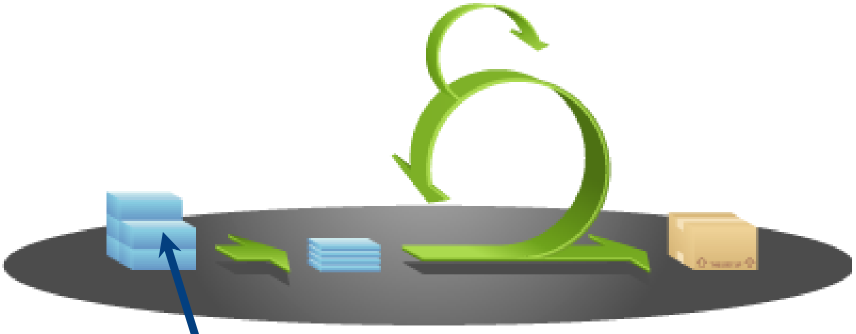
How will DiDs be modified to accommodate decisions about requirements documentation?

How will requirements be expressed? Stories, Epics? What about NFRs?

How projective vs as-built will requirements documentation be?

# Product Backlog: Scrum's Way of Organizing and Prioritizing Requirements at Team Level

*Many other Agile/Lean methods adopt the backlog idea in some way*



Copyright © 2005, Mountain Goat Software

This is the product backlog

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* (represents the business/acquirer)

Reprioritized at the start of each release and each sprint

# Stories—an Agile Way of Including the "Why" of a Low-Level 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

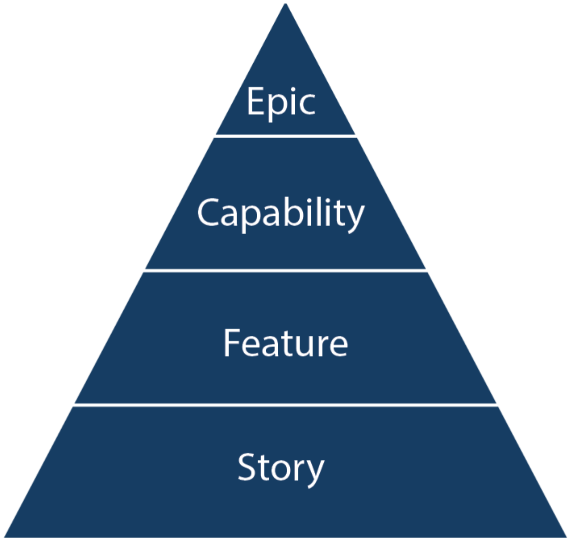
Template: To meet "quality attribute," system/subsystem/component must "do function."

## Use Cases

Not seen as often, but are effectively used in some Agile environments

# Multiple Levels of Abstraction for Requirements Constructs Accommodated in Most Agile Scaling Frameworks

SAFe Requirements Hierarchy



Typical hierarchy (from SAFe, in this case):

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

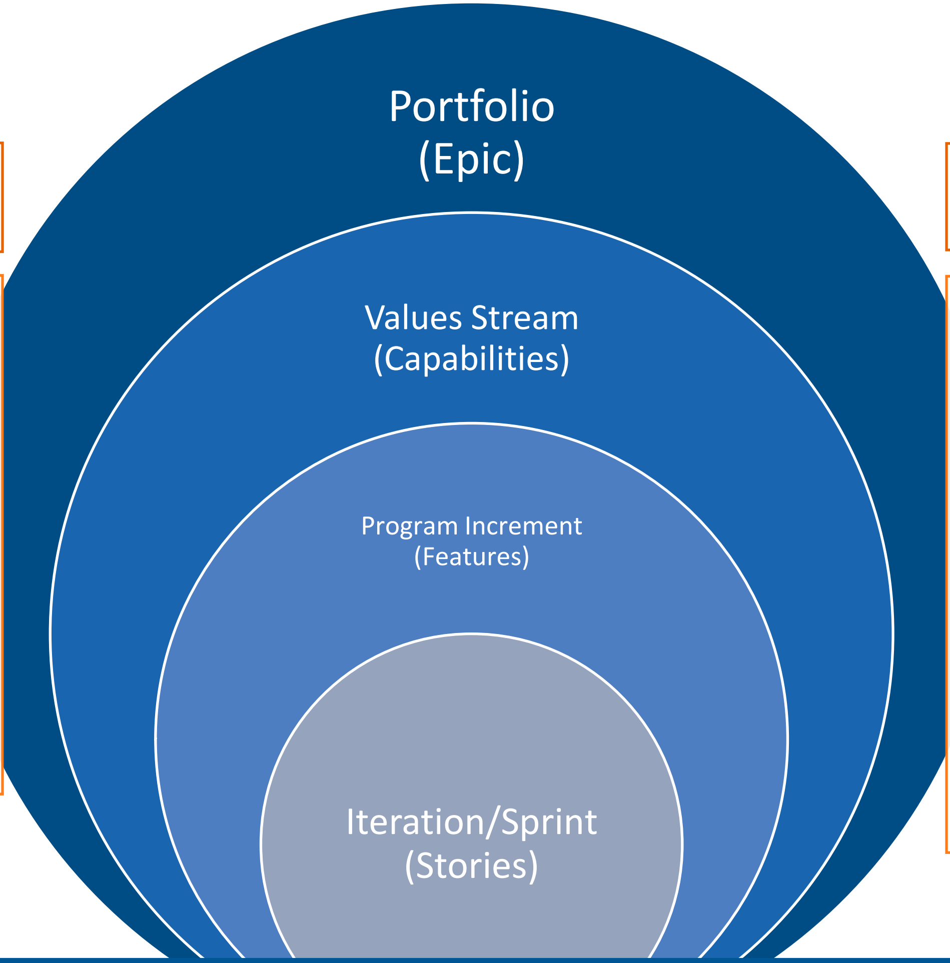
One of the decisions to make is how different levels of requirements will be treated

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

# Addressing Requirements at Multiple Levels in Agile Settings (SAFe Terminology)

## Issues in Expressing Requirements

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



## Issues in Governing Requirements

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

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

# Product Backlog Feeds Releases

Multiple Layers of Backlog

Kanban is Common Method for Prioritizing at Each Level

List of high-level “requirements”

- Prioritized by Product Owner/Manager
  - “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/managers estimate relative **value** of the backlog items

# Agile Motto for the Backlog Process

If it's IN the backlog, it MIGHT get done…

If it ISN'T in the backlog, it WON'T get done!

# Frequent Question About Requirements and Sprints

Probably the most frequently asked question about overseeing Agile teams 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 Possible Answers to Deferring Requirements

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

# Strategies to Think About

# Example Strategy for Requirements Management in Agile Setting: JMPS AMPD

JMPS AMPD is a USAF ground system that interacts with multiple air platforms to assist in Mission Planning

They have a master contract and 5 contractors who have qualified to bid on Delivery Orders over a 5 year period.

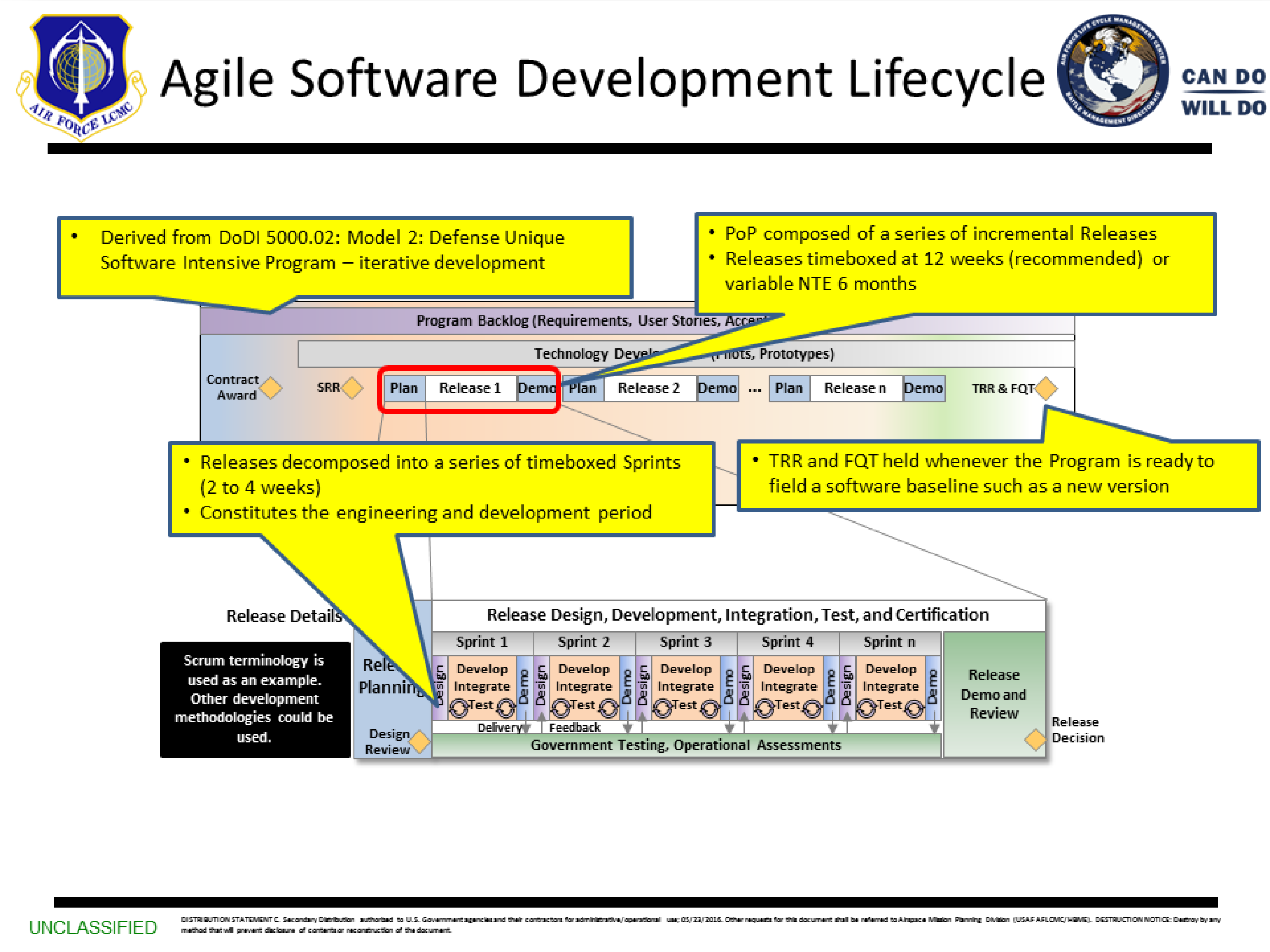
Some contractors, for some RFPs, want to use Agile approaches to accomplish the work; for other jobs, they might bid a waterfall approach

AMPD, Mitre, and SEI worked together to produce a "Mission Planning Agile Acquisition Model" as a framework to guide production of more detailed process assets to support both Agile and waterfall deliveries

- Agile approach is SAFe-based, with some tailoring
- SEI is currently working with AMPD to build work instructions and other process assets to support use of MPAAM

Slides below are from a May 23, 2016 presentation by Mitre about various aspects of MPAAM.

# USAF Program Example of Designing Tailoring for Acquisition Life Cycle to Accommodate Agile Contractors



## Agile Software Development Lifecycle

CAN DO
WILL DO

- Derived from DoDI 5000.02: Model 2: Defense Unique Software Intensive Program – iterative development

- PoP composed of a series of incremental Releases
- Releases timeboxed at 12 weeks (recommended) or variable NTE 6 months

- Releases decomposed into a series of timeboxed Sprints (2 to 4 weeks)
- Constitutes the engineering and development period

- TRR and FQT held whenever the Program is ready to field a software baseline such as a new version

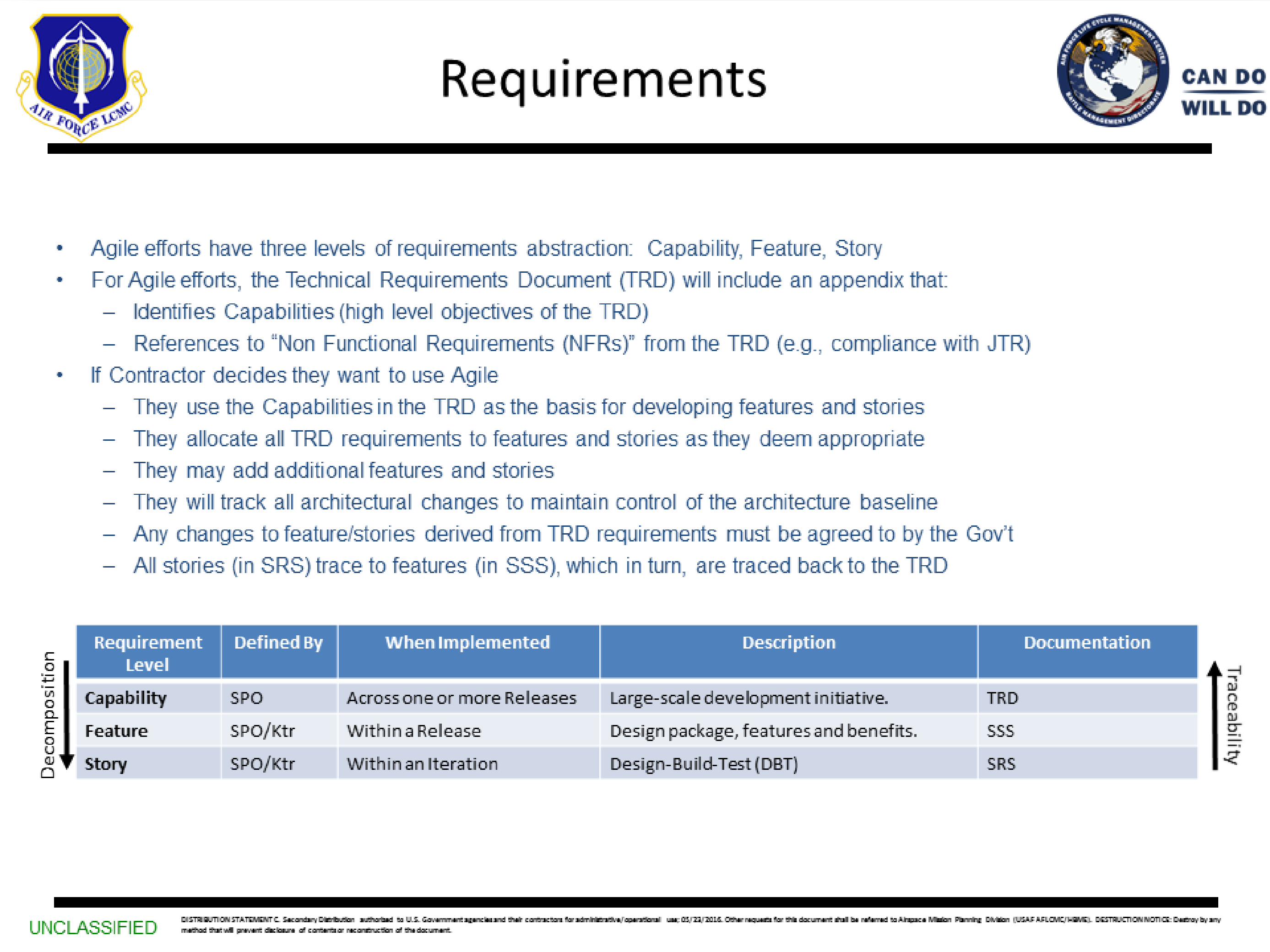
Scrum terminology is used as an example. Other development methodologies could be used.

UNCLASSIFIED

DISTRIBUTION STATEMENT C. Secondary Distribution authorized to U.S. Government agencies and their contractors for administrative/operational use; 05/23/2016. Other requests for this document shall be referred to Airspace Mission Planning Division (USAF AFLCMC/HBME). DESTRUCTION NOTICE: Destroy by any method that will prevent disclosure of contents or reconstruction of the document.

*Source: Pattee, J. et al. "Mission Planning Agile Acquisition Model (MPAAM)", Interagency SDLC Seminar, May 23, 2016* Reproduced with permission

# How Will You Organize Requirements Documentation? Example….



CAN DO
WILL DO

## Requirements

- Agile efforts have three levels of requirements abstraction: Capability, Feature, Story
- For Agile efforts, the Technical Requirements Document (TRD) will include an appendix that:
  - Identifies Capabilities (high level objectives of the TRD)
  - References to "Non Functional Requirements (NFRs)" from the TRD (e.g., compliance with JTR)
- If Contractor decides they want to use Agile
  - They use the Capabilities in the TRD as the basis for developing features and stories
  - They allocate all TRD requirements to features and stories as they deem appropriate
  - They may add additional features and stories
  - They will track all architectural changes to maintain control of the architecture baseline
  - Any changes to feature/stories derived from TRD requirements must be agreed to by the Gov't
  - All stories (in SRS) trace to features (in SSS), which in turn, are traced back to the TRD

Decomposition

| Requirement Level | Defined By | When Implemented | Description | Documentation |
|---|---|---|---|---|
| Capability | SPO | Across one or more Releases | Large-scale development initiative. | TRD |
| Feature | SPO/Ktr | Within a Release | Design package, features and benefits. | SSS |
| Story | SPO/Ktr | Within an Iteration | Design-Build-Test (DBT) | SRS |

Traceability

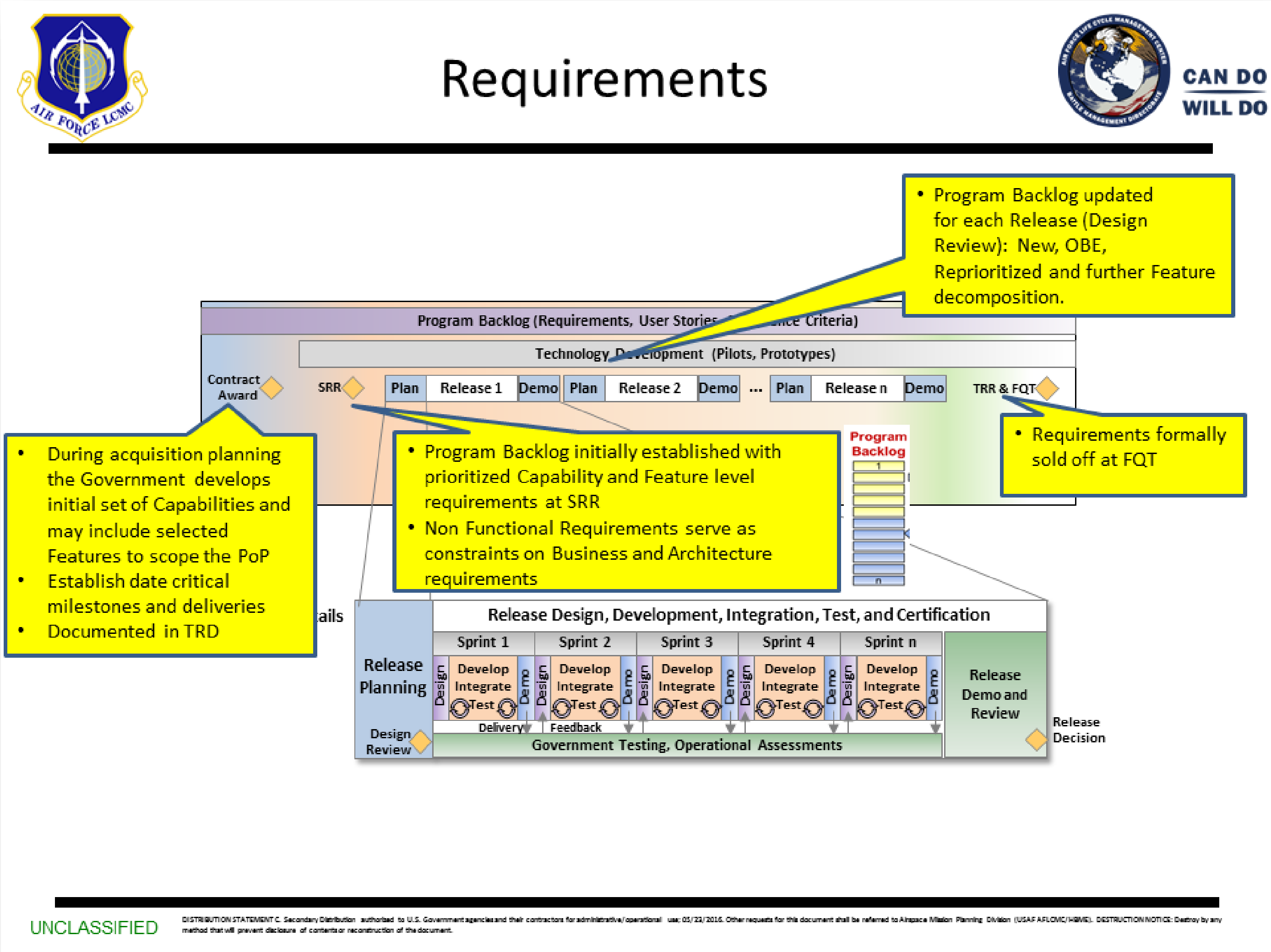
UNCLASSIFIED

DISTRIBUTION STATEMENT C. Secondary Distribution authorized to U.S. Government agencies and their contractors for administrative/operational use; 05/23/2016. Other requests for this document shall be referred to Airspace Mission Planning Division (USAF AFLCMC/HBMS). DESTRUCTION NOTICE: Destroy by any method that will prevent disclosure of contents or reconstruction of the document.

*Source: Pattee, J. et al. "Mission Planning Agile Acquisition Model (MPAAM)", Interagency SDLC Seminar, May 23, 2016*

Reproduced with permission

# Be Explicit About Requirements Signoff: Example



## Requirements

- Program Backlog updated for each Release (Design Review): New, OBE, Reprioritized and further Feature decomposition.

- During acquisition planning the Government develops initial set of Capabilities and may include selected Features to scope the PoP
- Establish date critical milestones and deliveries
- Documented in TRD

- Program Backlog initially established with prioritized Capability and Feature level requirements at SRR
- Non Functional Requirements serve as constraints on Business and Architecture requirements

- Requirements formally sold off at FQT

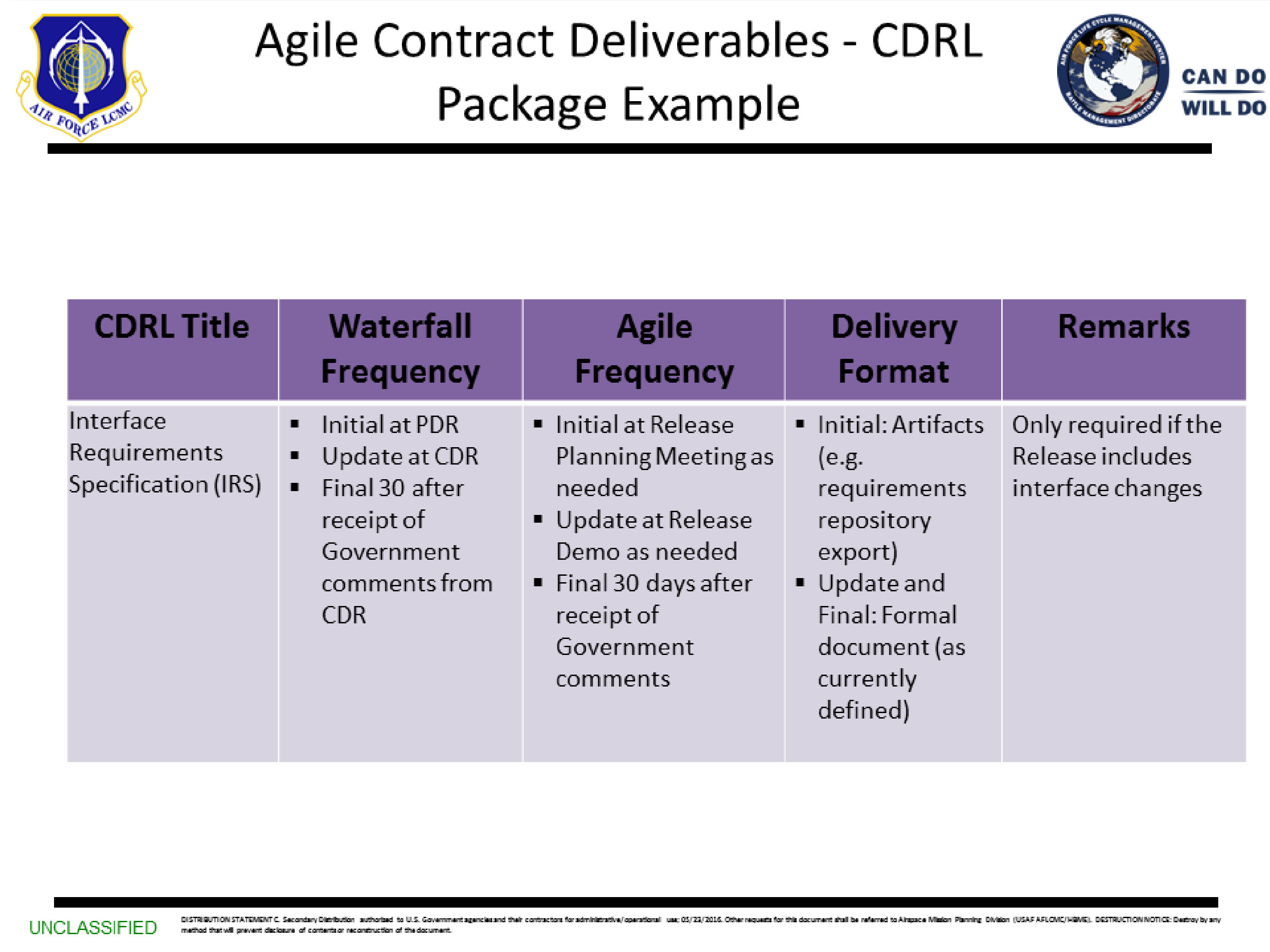
UNCLASSIFIED

DISTRIBUTION STATEMENT C. Secondary Distribution authorized to U.S. Government agencies and their contractors for administrative/operational use; 05/23/2016. Other requests for this document shall be referred to Airspace Mission Planning Division (USAF AFLCMC/HBME). DESTRUCTION NOTICE: Destroy by any method that will prevent disclosure of contents or reconstruction of the document.

*Source: Pattee, J. et al. "Mission Planning Agile Acquisition Model (MPAAM)", Interagency SDLC Seminar, May 23, 2016*

Reproduced with permission

# This Means Reviewing DIDs for Needed Changes: Example



## Agile Contract Deliverables - CDRL Package Example

AIR FORCE LCMC
CAN DO
WILL DO

| CDRL Title | Waterfall Frequency | Agile Frequency | Delivery Format | Remarks |
|---|---|---|---|---|
| Interface Requirements Specification (IRS) | ▪ Initial at PDR<br>▪ Update at CDR<br>▪ Final 30 after receipt of Government comments from CDR | ▪ Initial at Release Planning Meeting as needed<br>▪ Update at Release Demo as needed<br>▪ Final 30 days after receipt of Government comments | ▪ Initial: Artifacts (e.g. requirements repository export)<br>▪ Update and Final: Formal document (as currently defined) | Only required if the Release includes interface changes |

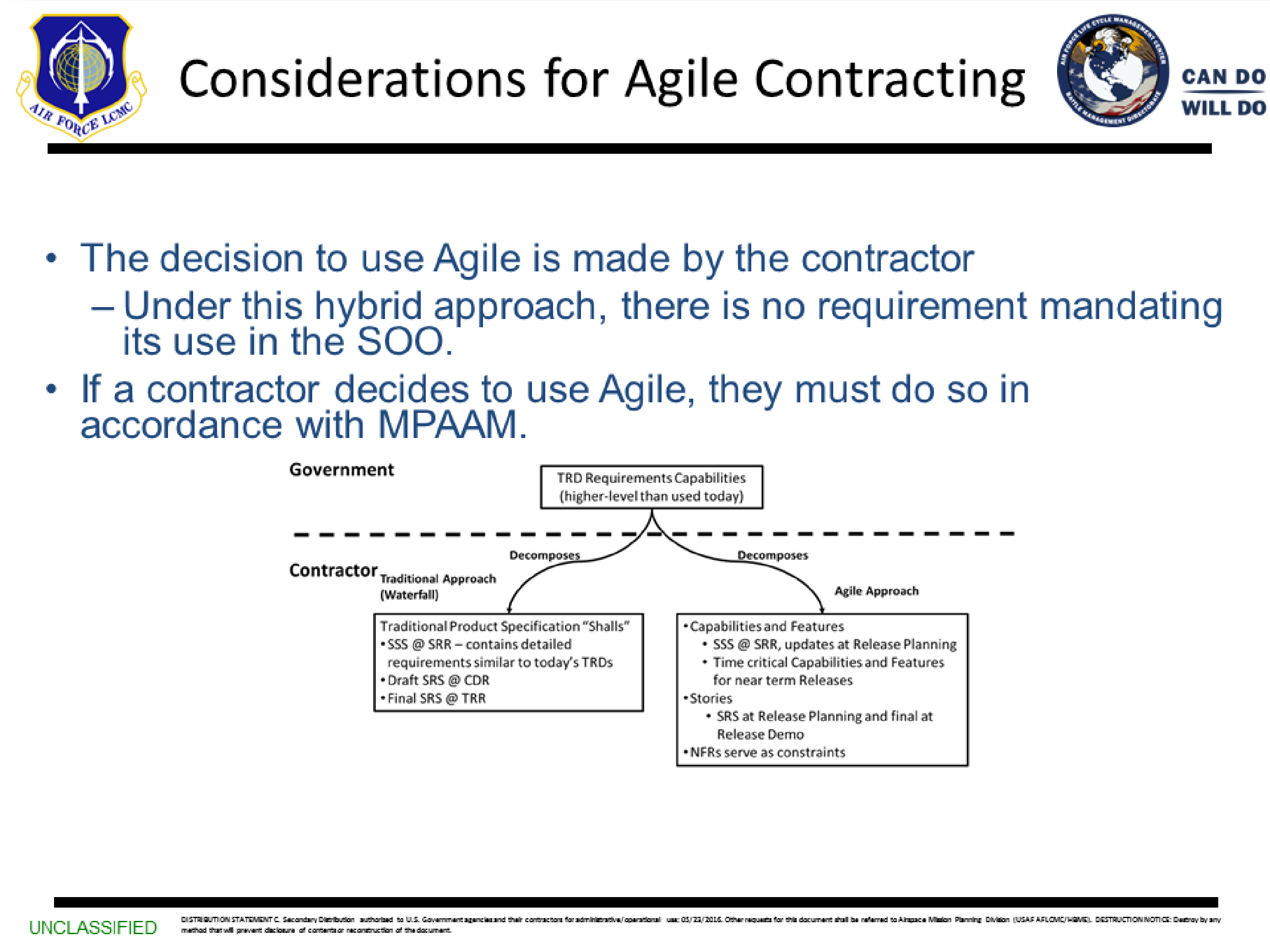
UNCLASSIFIED

DISTRIBUTION STATEMENT C. Secondary Distribution authorized to U.S. Government agencies and their contractors for administrative/operational use; 05/23/2016. Other requests for this document shall be referred to Airspace Mission Planning Division (USAF AFLCMC/HBME). DESTRUCTION NOTICE: Destroy by any method that will prevent disclosure of contents or reconstruction of the document.

*Source: Pattee, J. et al. "Mission Planning Agile Acquisition Model (MPAAM)", Interagency SDLC Seminar, May 23, 2016*

Reproduced with permission

# (How) Will You Accommodate both Agile and Waterfall? Example



## Considerations for Agile Contracting

AIR FORCE LCMC
CAN DO
WILL DO

- The decision to use Agile is made by the contractor
  - Under this hybrid approach, there is no requirement mandating its use in the SOO.
- If a contractor decides to use Agile, they must do so in accordance with MPAAM.

Government

TRD Requirements Capabilities (higher-level than used today)

Contractor

Decomposes — Traditional Approach (Waterfall)

Traditional Product Specification "Shalls"
- SSS @ SRR – contains detailed requirements similar to today's TRDs
- Draft SRS @ CDR
- Final SRS @ TRR

Decomposes — Agile Approach

- Capabilities and Features
  - SSS @ SRR, updates at Release Planning
  - Time critical Capabilities and Features for near term Releases
- Stories
  - SRS at Release Planning and final at Release Demo
- NFRs serve as constraints

UNCLASSIFIED

DISTRIBUTION STATEMENT C. Secondary Distribution authorized to U.S. Government agencies and their contractors for administrative/operational use; 05/23/2016. Other requests for this document shall be referred to Airspace Mission Planning Division (USAF AFLCMC/HBME). DESTRUCTION NOTICE: Destroy by any method that will prevent disclosure of contents or reconstruction of the document.

*Source: Pattee, J. et al. "Mission Planning Agile Acquisition Model (MPAAM)", Interagency SDLC Seminar, May 23, 2016*

Reproduced with permission

# Summary

# Summary

**Both Expressing and Governing Requirements Need Attention in Agile Contracting Settings**

- Expressing—what forms, levels of abstraction, etc
- Governing – when, how change will be accommodated; what level of detail is Allocated Baseline

**Stakeholder Management is Key in Managing Requirements in Agile Settings**

- Those outside the software/Agile part of the development expect what they've always received
- Being explicit and communicating often about what is same/different than "usual" is important

**Requirements in Agile Add 2 Key Elements to Typical Requirements Expression**

- Who needs it? ( via "As a <role>")
- Why is it needed? (via "so that I can <goal>")
- These elements provide needed foundation for ongoing discussions for refinement