# Agile in Government Introduction

For VM-Fusion

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

# Document Markings



This material is based upon work funded and supported by the Department of Homeland Security 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 sponsored by the United States Department of Defense.



DM22-0423

# Meeting Conventions for Today

Please stay on mute for the lecture portion of the course module.

If you are “in” the Teams meeting via web or app, please ask questions via the Chat window.

- A facilitator will collect the questions and either pass them to the facilitator if something immediate, or organize them for the Q&A portion of the course module

Instructor will call for participation and discussion at break points. Please remember to come off mute before talking.

When you are done talking, before going back on mute, please say “Over” so others know you are finished.

SEI PowerPoint Template

# Practical Considerations

# Learning Objectives

Describe agile and how to confirm that VM-Fusion is adopting this mindset

Describe agile values & principles and identify ways that these are relevant to VM-Fusion

Recognize Scrum and Kanban practices including how they apply to VM-Fusion activities

**Practical Considerations**

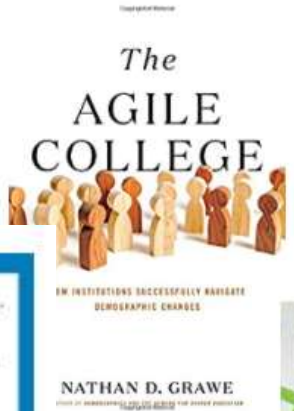
# Agile Principles

# Working Definition of Agile

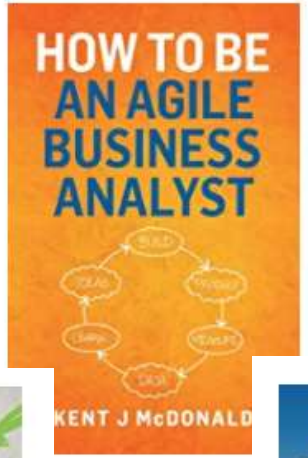

Agile An *iterative* and *incremental* (evolutionary) approach to software development which is performed in a *highly collaborative manner* by *self-organizing teams* within an *effective governance framework* with *"just enough" ceremony* that produces *high quality software* in a *cost effective and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

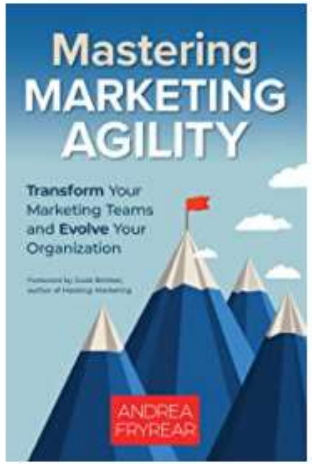
[Ambler 2013] Ambler, Scott. *Disciplined Agile Software Development: Definition*.
http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

# Agile is Everywhere

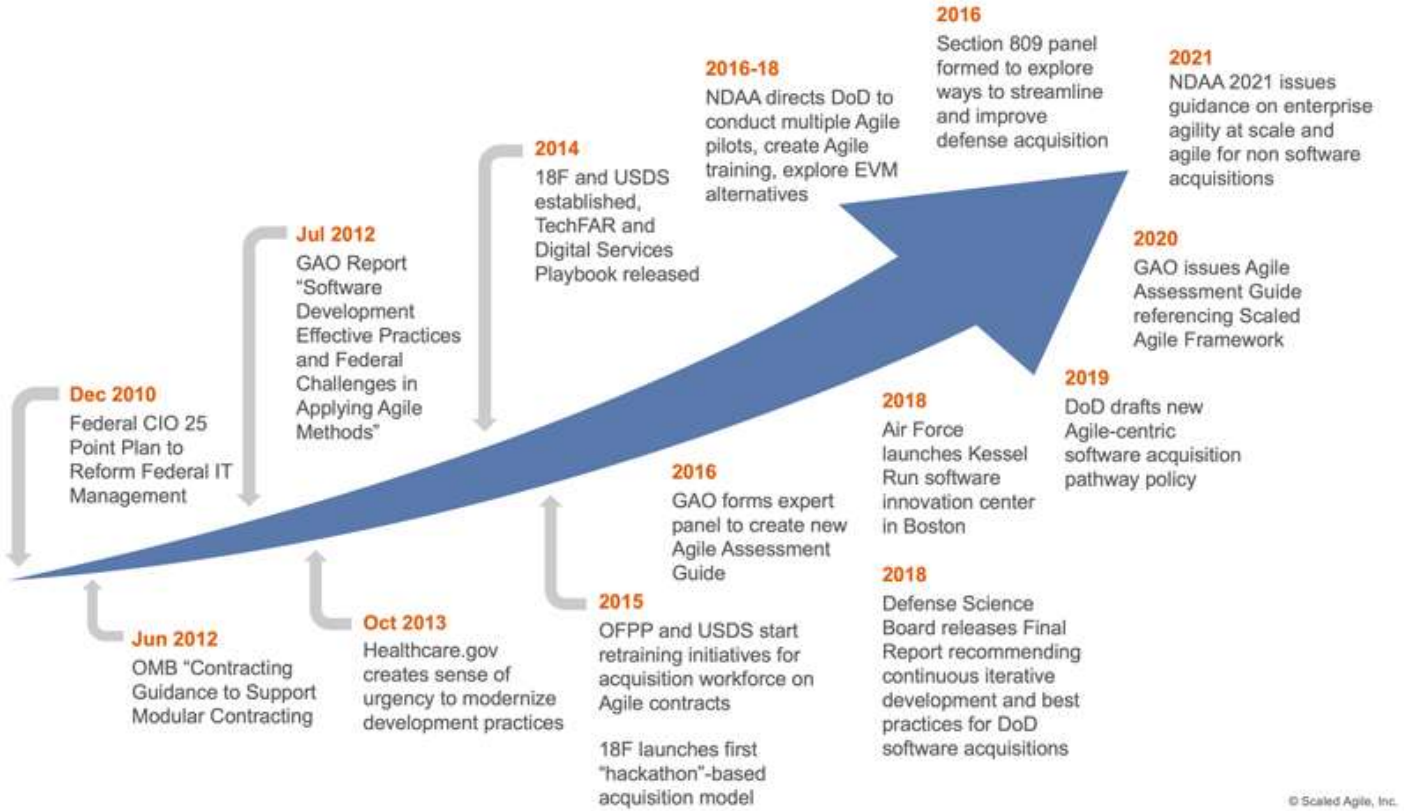







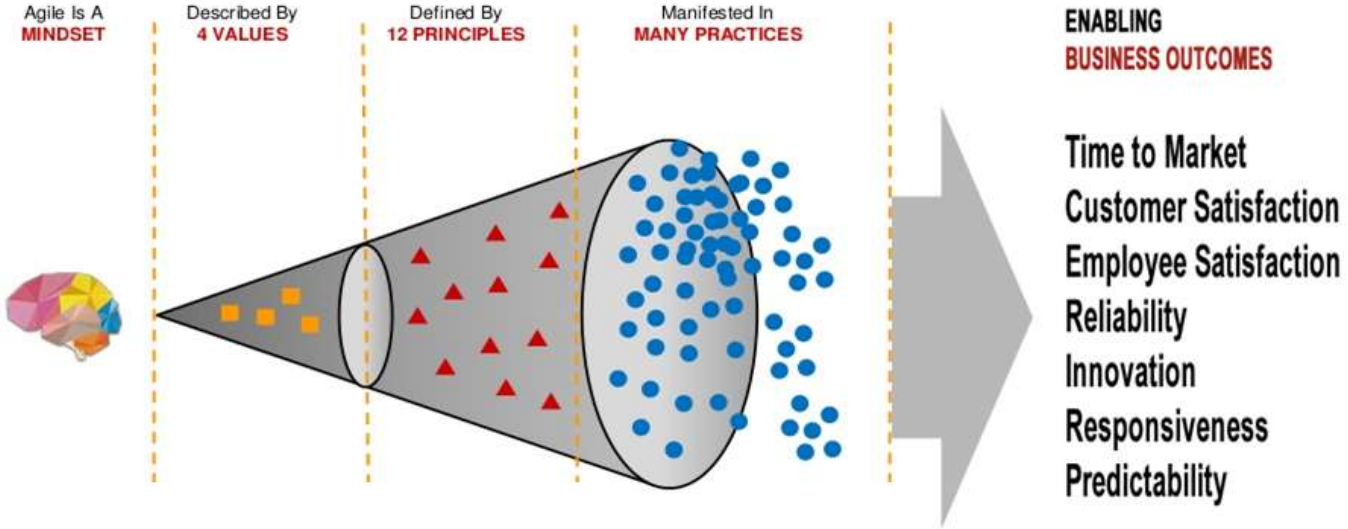




# Events Driving Lean-Agile Adoption in US Government




Source: https://www.scaledagileframework.com/government-article/, as of Apr 13, 2021

# What is Agile?




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

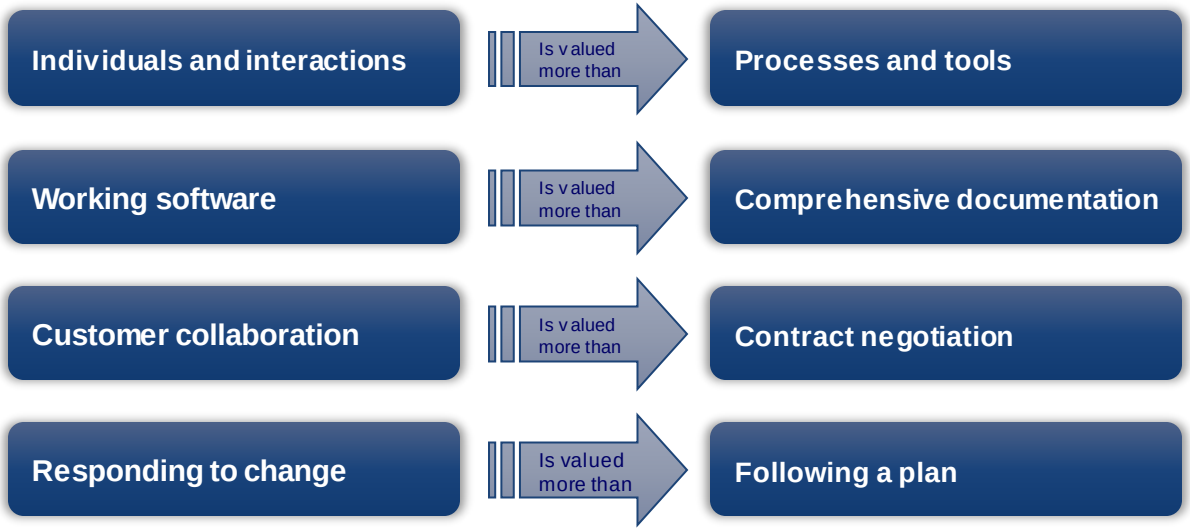
Implementing the practices, tools and processes **without** the Agile mindset, values, and principles of the Agile Manifesto

***Is NOT Agile!***


**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.**

# Agile Values

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




➢ **Activity: Which side do you think will benefit your customer more?**

http://agilemanifesto.org

# Agile Principles

For most of these, substituting "working product" allows translation into non-software environments.

1. Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable **software.**
2. Welcome **changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. **Business people and developers must work together** daily throughout the project.
5. **Build projects around motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation.**
7. **Working software is the primary measure of progress.**
8. Agile processes **promote sustainable development.** The sponsors, developers, and users should be able to maintain 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.

Agile Principles: http://agilemanifesto.org

➢ **Activity: Which principle(s) do you think are most applicable to the work you do?**

# Please Open This Web Site for the Next Portion of the Training

https://pollev.com/mainsummit799

# Don’t Forget Lean Principles


SEVEN WASTES

#1 Take an economic view

#2 Apply systems thinking

#3 Assume variability; preserve options

#4 Build incrementally with fast, integrated learning cycles

#5 Base milestones on objective evaluation of working systems

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

#7 Apply cadence, synchronize with cross-domain planning

#8 Unlock the intrinsic motivation of knowledge workers

#9 Decentralize decision-making

#10 Organize around value

**SAFe Lean-Agile Principles**

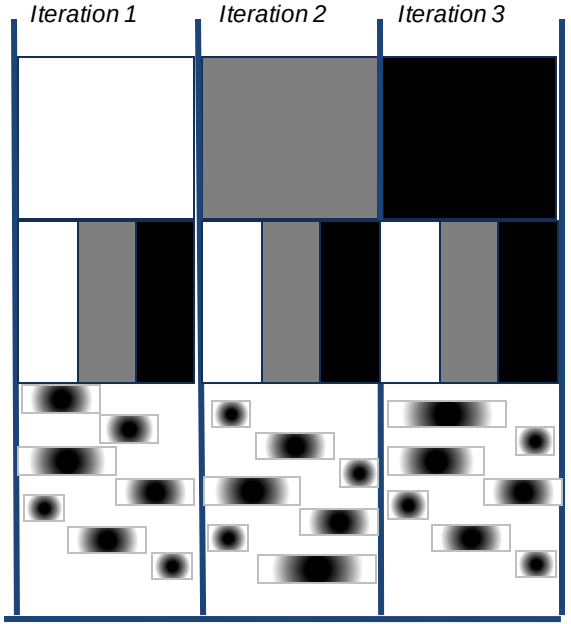
**Lean principles focus on more of an enterprise level than Agile’s**
**Lean principles are compatible with Agile and DevSecOps**

# 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



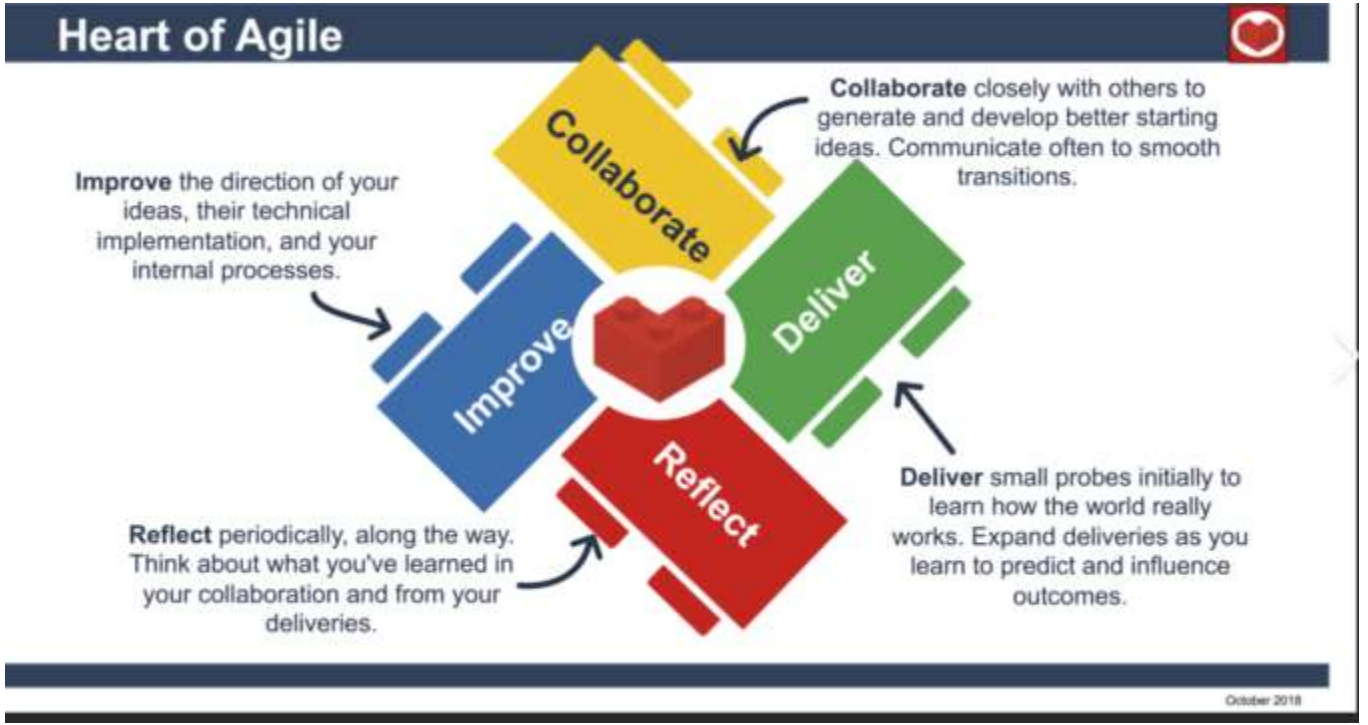

Distillation of a number of lean principles

# Start with Teams Using Common Concepts But Allow Methods That Suit Their Context

All team-focused Agile approaches have commonalities and differences

Focus on the commonalities whenever possible and choose the best way forward for your team that suits your context

Cockburn's "Heart of Agile" provides a way to look at commonalities across team methods

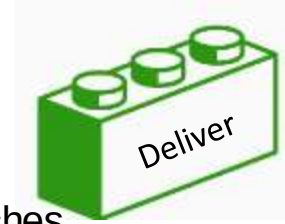



*Source: heartofagile.com*

# Agile Team Method Commonalities: Scrum, XP, Kanban Are The Most Used


- Collaborate across functions and stakeholders
- Communicate often to smooth handoffs
- Build trust-based relationships
- Use common tools, where feasible
- Establish & evolve prioritized backlogs of work items


- Learn from each iteration's products
- Enable fast feedback




- Work in small batches
- Build quality in
- Design modular work items
- Divide work into short iterations
- Deliver incrementally


- Act on learning, don't just capture it
- Improve direction of ideas, technical implementation, and internal processes

# All Common Agile Team Methods are Based on PDCA Cycle

Plan - a task, change or test, aimed at improvement.

- Analyze what you intend to improve - choose areas with highest rate of return

Do - Carry out the change or test (preferably on a small scale).

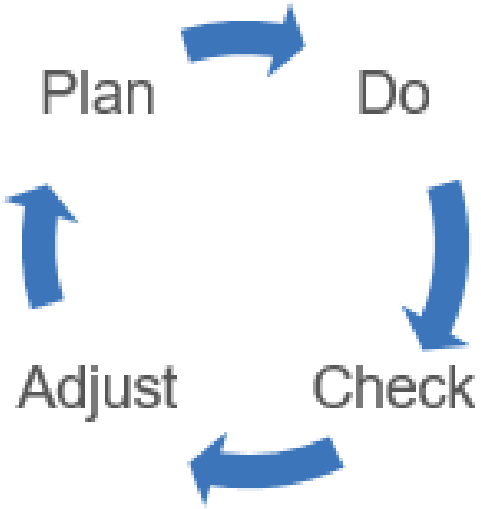
- Implement the change you decided on in the plan phase.

Check - the results. What was learned? What went wrong? (We have received empirical data! We are NOT guessing!)

- Measure / monitor the level of improvement.

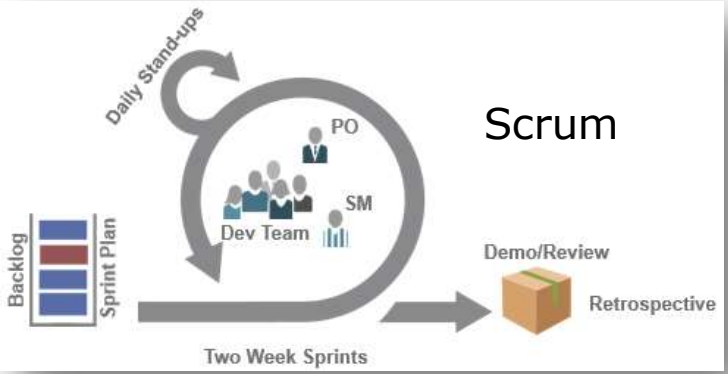
Adjust – Make the change based on the empirical data.

- Adopt the change, abandon it, or run through the cycle again.




*Plan – Do – Check – Adjust (PDCA) cycle is inherent in all of our work.*

# Choose One or More Methods to Fit Your Team Context

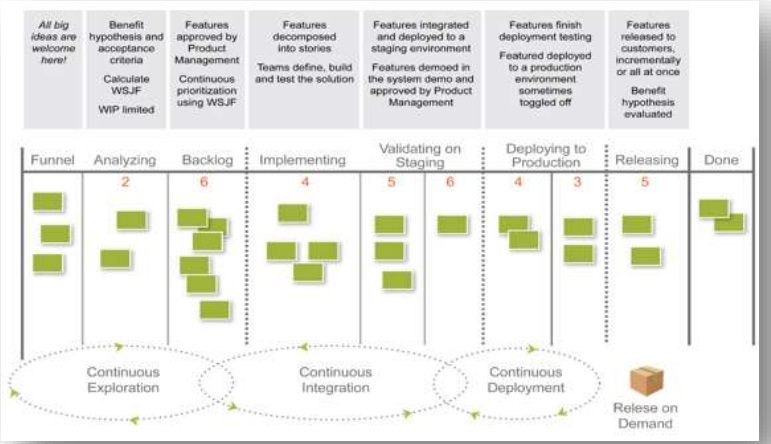



## Scrum

- Most common product dev approach
  Not just SW dev
- *Use when you have a dedicated team working on a product and timebox is an advantage*

- Applies to almost any type of work
- Commonly used for teams not focused on products per se (e.g., teams who do lots of review/ coordinate tasks)
- *Use when services are more in play than product delivery, and when team is not dedicated to single product*

## Kanban



# Common Tools Will be Used Across Different Team Types

Tool focused on workflow management: JIRA

Tool focused on artifact (document) evolution and sharing: CONFLUENCE

Tool focused on inter/intra-team collaboration: MATTERMOST and MS Teams

Development, integration, and test teams of evolving software and hardware products will have lots of other tools specific to their context



Agile in Government

**Practical Considerations**

# Scrum

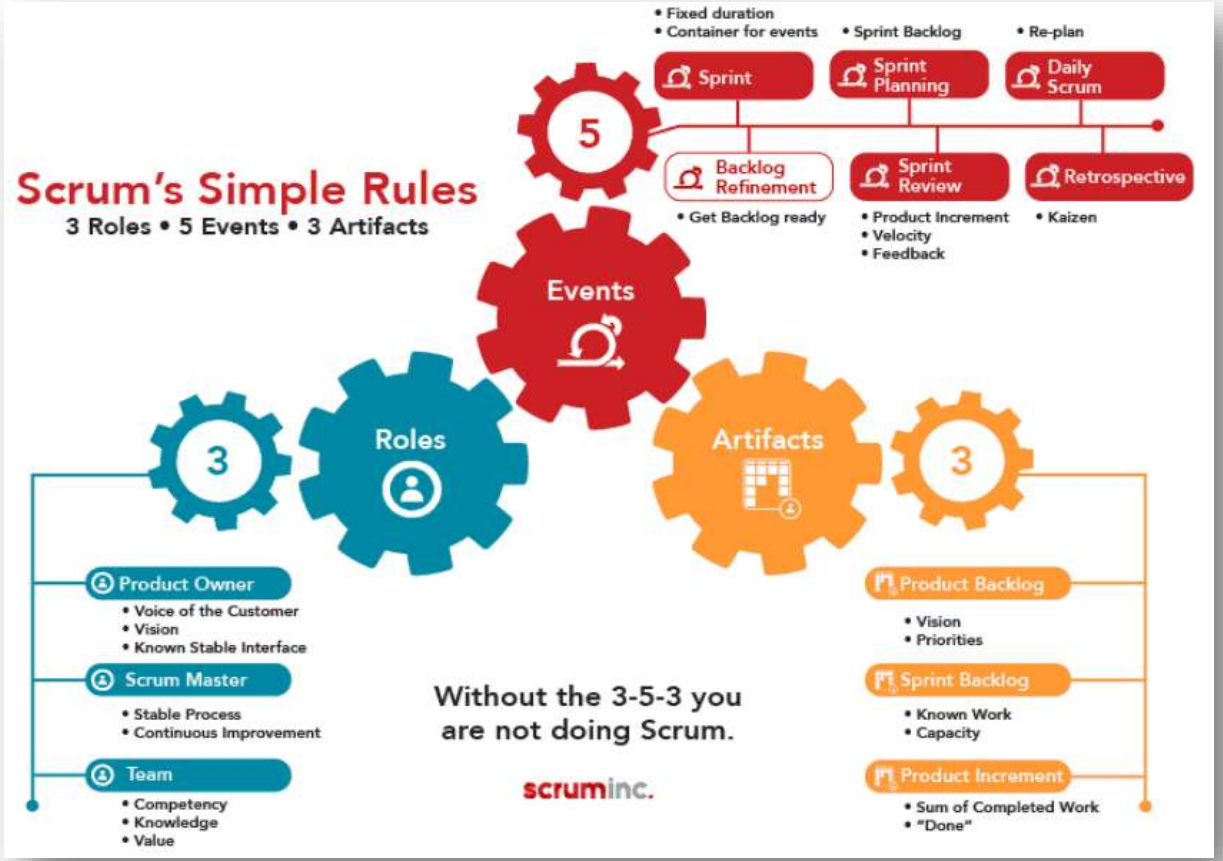
# Scrum

The most popular approach to agile implementation

Really about team management

Almost invariably mixed with development practices from eXtreme Programming (XP)

# Scrum's Simple Rules




Scrum is founded on empirical process control theory, or **empiricism**

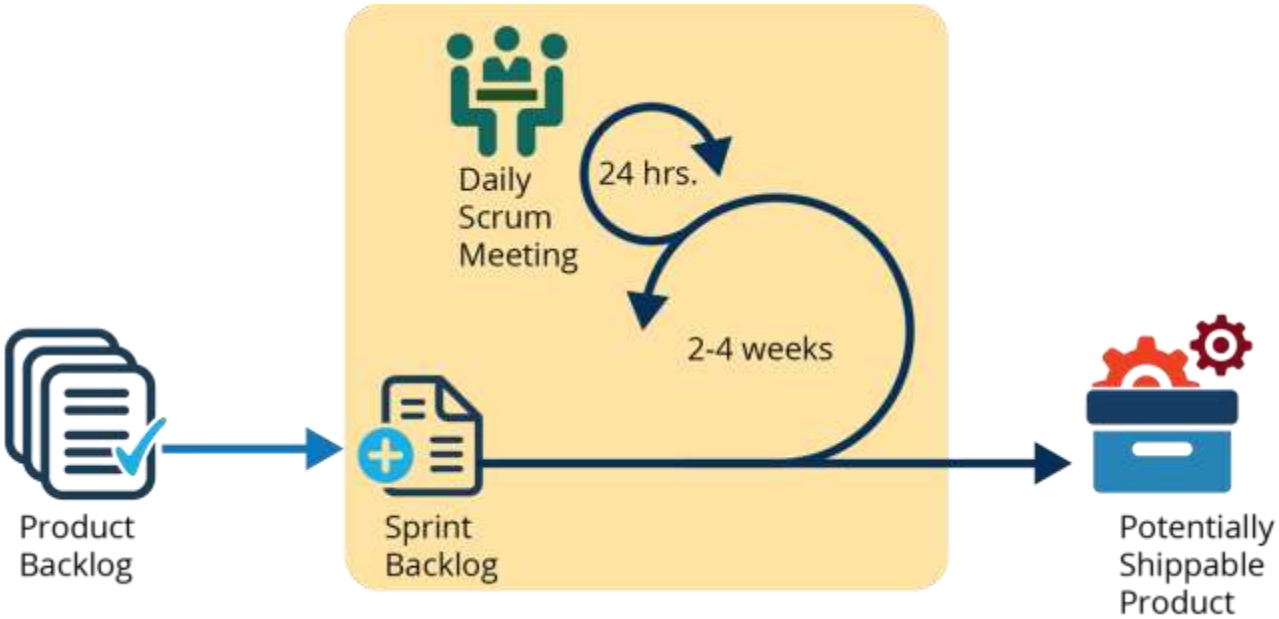
**Empiricism** asserts that knowledge comes from real data and experience; and that making decisions is based on what is known.

Scrum employs an iterative, incremental approach **to optimize predictability and control risk**

**Optimal Team size is 3-9 members, ("2 pizza rule" (4.6 optimum)) not counting SM and PO**

**Scrum Teams are cross-functional, self-organizing and self-managing**

# Key Elements of Scrum



# 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 3-9 members (usually developers and testers) plus the Scrum Master (sometimes called Agile Team Lead) and Product Owner

# New Terms – Events$_1$

***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

***(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:***

- 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.

# New Terms – Events$_2$

***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

***Backlog Refinement:***

- An ongoing activity that looks at the uncommitted backlog, reprioritizes it as needed and ensures that the top items are ready for the next sprint

# 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

***Product Increment (potentially shippable):***

- 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

***(Visualizations of Progress:)***

- E.g., Burndown Charts: a visual tool displaying progress via a simple line chart representing remaining work (vertical axis) over time (horizontal axis)

# 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

***Definition of Done[1]:***

- A checklist of value-added activities that must be performed before a story can be declared to be Done. The checklist is determined by the Agile Team. ***Information Radiator***: a physical or virtual display of tasks and progress that is accessible to all stakeholders throughout the project

1 https://www.scrumalliance.org/community/articles/2008/september/what-is-definition-of-done-(dod)

Agile in Government

**Practical Considerations**

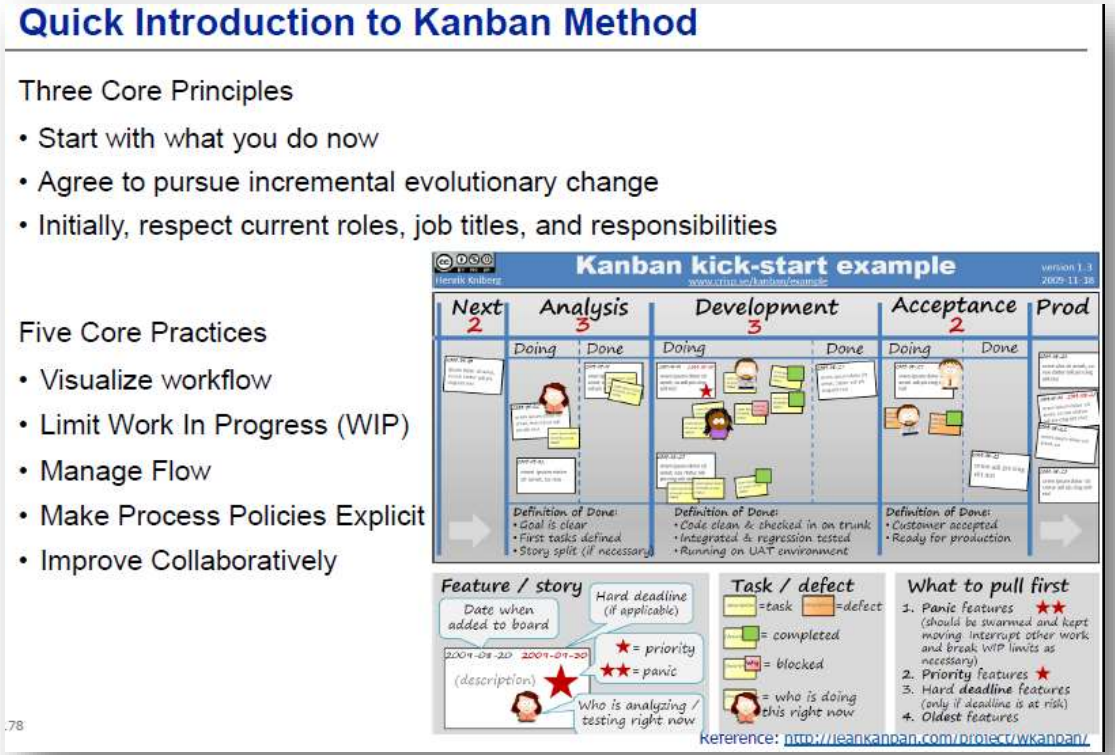
# Kanban

# Kanban

Although Kanban can be used on its own, it is frequently used within the context of Scrum & XP when the team is a product development team

- In that context, Kanban is frequently used in Sprint execution to visualize the work the team is currently committed to

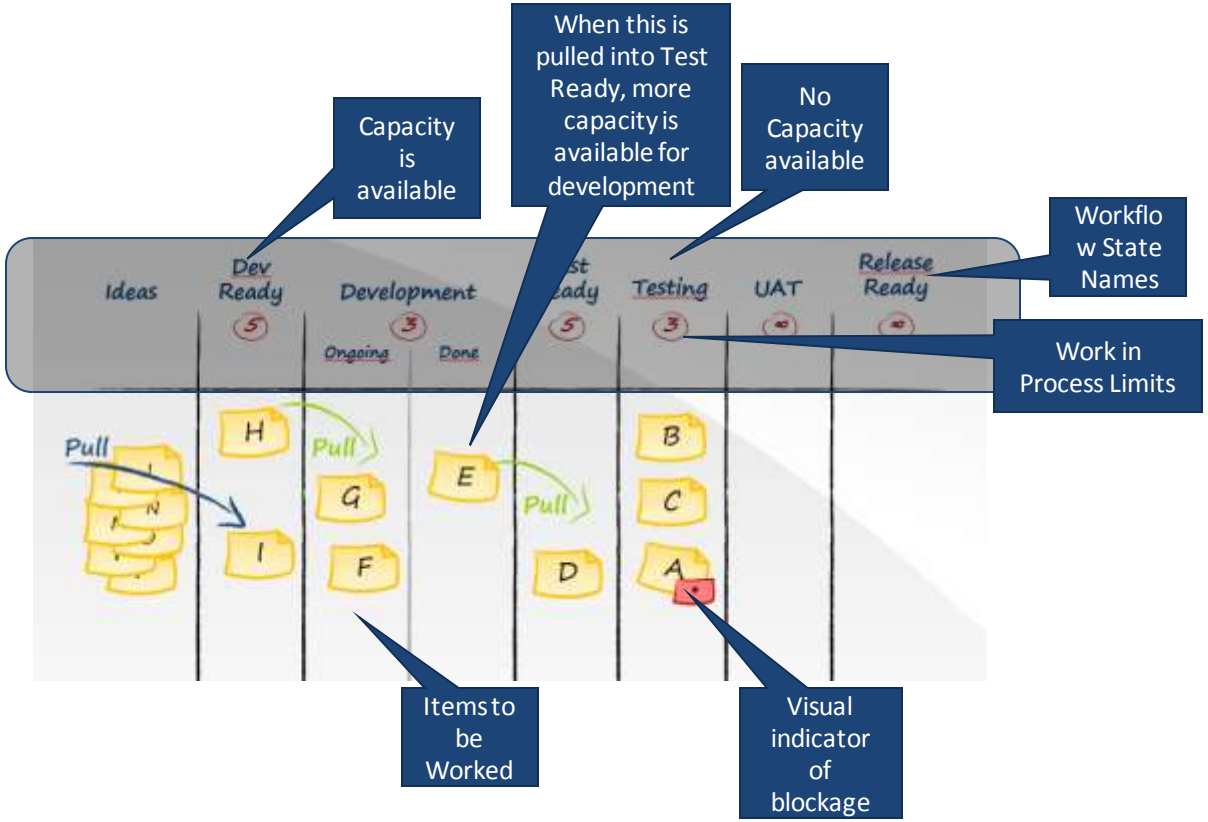
Kanban is the most frequently used team method for non-SW development teams (e.g., systems engineering, certification, contracting)



# Anatomy of a Development-Focused Kanban Board

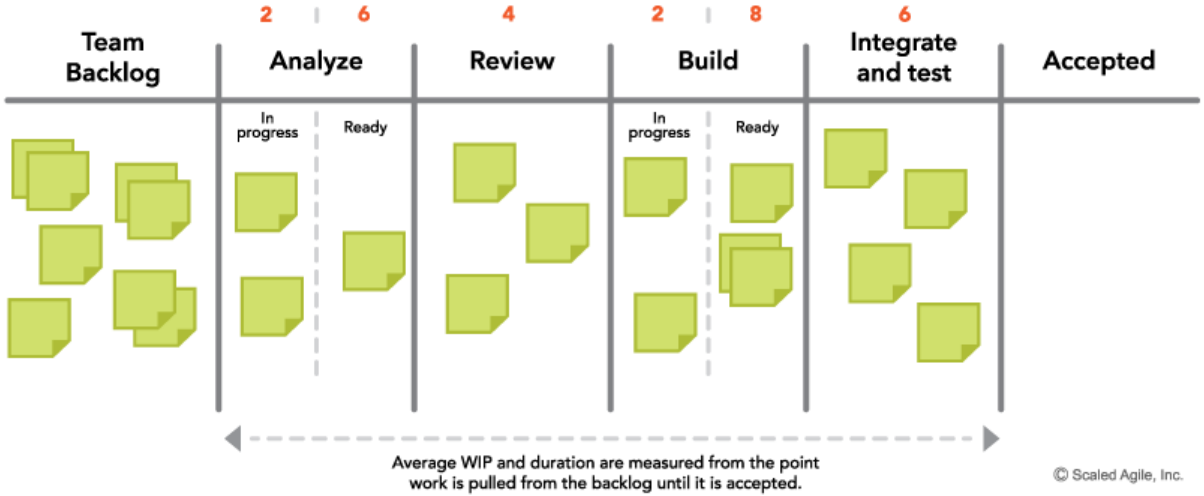
***Many Kanban examples focus on development. How would the workflow states differ for systems engineering, or contracting services?***

***Add different workflow categories in the Mural where indicated.***

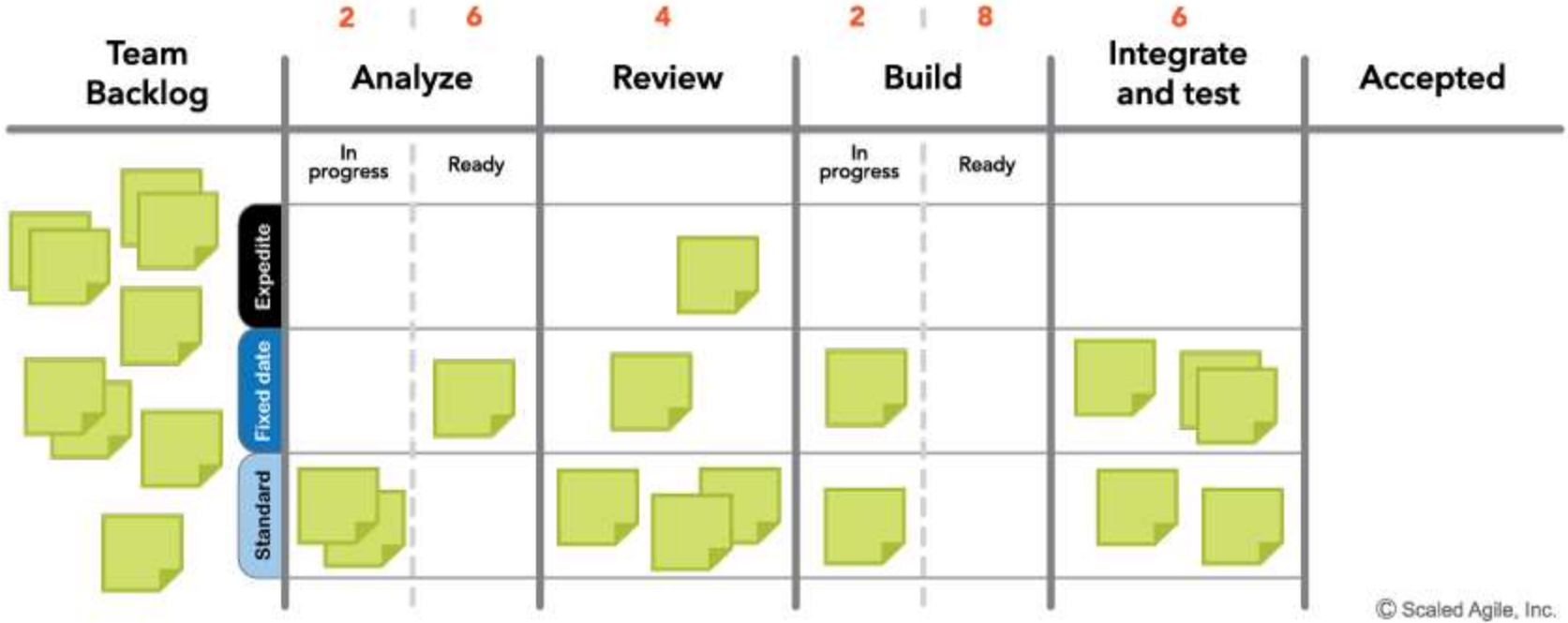


# Kanban Example

To get started teams typically build an approximation of their current process flow and define some initial WIP limits. Here is an example of a team's initial Kanban board, which captures their current workflow states: Analyze, Review, Build, and Integrate and test.



# Classes of Service on the Kanban Board



# Improving Flow with Classes of Service

Teams need to be able to manage dependencies as well as ensure alignment with Milestones. Kanban uses the concept of classes of service to help teams optimize the execution of their backlog items.

Classes of service help differentiate backlog items based on their Cost of Delay (CoD).

Each class of service has a specific execution policy that the team agrees to follow.

Classes of service are typically visualized as ‘swim lanes.

# Class of Service Example

Standard – Baseline class of service, work items that are neither expedited nor fixed date.

- Most backlog items should fall into this category.
- The CoD is linear for standard items, meaning that value cannot be achieved until delivery occurs, but there's no specific date by which the work must be completed.

# Class of Service Example

Fixed date – Work items that must be delivered on or before a specific date.

- CoD of these items is nonlinear and is highly sensitive to small changes in the delivery date.
- These must be actively managed to mitigate the schedule risk.
- These items are pulled into development when necessary, to be finished on time.
- Some items may require additional analysis to refine the expected lead time.
- Some may need to be reclassified as expedite if the team falls behind.

# Class of Service Example

Expedite – Work item has an unacceptable CoD and therefore requires immediate attention.

- Can be pulled into development, even if it is in violation of current WIP limits.
- Typically, there can be only **one** expedite item in the system at a time.
- Teams may set a policy to swarm on that item to make sure it moves through the system rapidly.
- If teams find that many items require expediting, then the system may be overloaded. Either demand exceeds capacity, or the input process may need more discipline. Whatever the case, the process needs to be adjusted.

SEI PowerPoint Template

# Requirements Prioritization

# Learning Objectives

Select and apply prioritization techniques to VM-Fusion work

Recognize the value of periodic re-prioritization

Recognize roles of VM-Fusion PMs and business stakeholders in task sizing

Generate an initial prioritization of VM-Fusion projects

**Requirements Prioritization**

# Principles

# Prioritization Principles

Prioritization is essential for any agile project

- Can't work on everything at the same time
- Arrange work in order of importance
- Also allows the team to consider the minimum necessary to create value

Goal is to produce a ranked order of work items

Must be performed periodically

- New work items arrive and must be inserted into the backlog periodically
- Priorities change as we learn

Don't be stupid!

- Don't ignore dependencies, regardless of prioritization, if A depends on B, do B first
- Generated priorities can be adjusted

# Prioritization Techniques

There are many different prioritization techniques – so choose what works for you

List of some prioritization techniques:

- MoSCoW
- Kano model
- Relative Weighting Method
- Opportunity Scoring
- Stack Ranking
- Priority Poker
- Cost of Delay & WSJF
- 100 Dollar Test
- Mission Based Prioritization

# MoSCoW

All work items are place into one of the following categories:

**Must** – The must requirements are given the topmost priority

**Should** – Next priority is given to the requirements that are highly desirable, though not mandatory

**Could** – The next priority is given to the requirement that is nice to have

**Won't** – And the final consideration is given to the requirements which will not work in the process at that point of time

Can use time as a forcing function by asking "How do we place the work items if the work has to be done within the next 3 months?"

# Estimation Mechanisms

We can't be precise about the various factors that play into the different prioritization techniques so must come to consensus on estimates for the factors

- Use relative and not absolute estimation
- Typically use an arbitrary scale (Fibonacci sequence/T-shirt sizing)
- Don't have to be precise – the goal is speed not precision

Popular estimation techniques

- Planning Poker
- Bucket System
- Affinity Estimation
- Dot Voting
- White Elephant Estimation

# White Elephant Estimation

Team stand-up in a semi circle facing their sizing or white board

Have deck of story cards, in arbitrary order, in front of the board

Signal for the next member to perform the following steps:

- *Either take a card from the top of the deck read it out and place in one of the columns (a.k.a **propose the estimation** for that item)*
- ***OR** take one of the cards already placed on the board and move to another column (a.k.a **change the estimate**). If someone is moving card, he/she need to provide some reasons for doing it*
- ***OR** pass, if all the stories are placed and they are satisfied with the story placement*

Once all the stories are placed on the board, the team inspects the board and each member can propose to move one of the stories' place. They can, later on, discuss it with the product owner and ask questions that will help them estimate those stories together.

# Exercise

We have the list of tickets in the Fusion Center Ticket Inventory

- The keys have been placed on virtual stickies on a virtual whiteboard
- Need to prioritize them via MoSCoW

https://app.mural.co/invitation/mural/seissd7195/1651239756139?sender=ccampus9600&key=e7769018-3140-4d61-95e5-e418c400b67e

**Requirements Prioritization**

# Weighted Shortest Job First (WSJF)

# Weighted Shortest Job First (WSJF)

WSJF extends the notion of Cost of Delay by factoring in job duration

Produces maximum economic benefit by focusing on sequence of jobs rather than individual job ROI

Very simple: Weight = Cost of Delay / Job size

Cost of Delay can be measured in terms of:

- User business/mission value – relative value to the customer/organization
- Time criticality – the urgency of the job (e.g., is there a date after which it has no value)
- Risk reduction/opportunity enablement (RR/OE) – what else does this do for the organization? Does it reduce risk? Can it enable other jobs to proceed more rapidly?

Cost of Delay = User value + Time criticality + RR/OE

# Applying WSJF

For the selected jobs:

1) Estimate user value for all jobs
2) Estimate time criticality for all jobs
3) Estimate RR/OE for all jobs
4) Estimate job size for all jobs
5) Perform the calculations and prioritize accordingly

https://app.mural.co/invitation/mural/seissd7195/1651239756139?sender=ccampus9600&key=e7769018-3140-4d61-95e5-e418c400b67e

**Requirements Prioritization**

# Mission Based Prioritization

# Prioritization

# Backlogs in Real Life

# A Solution

- Provides strategic or tactical advantage
- Foundational (other features depend on this feature)
- Replaces deficient functionality
- Supplies missing functionality in existing system
- High sponsorship / visibility

- Fulfills direct user request
- Required to field
- Hardens security
- Time critical / urgent
- Previously de-prioritized

- Difficulty-to-implement
- Uncertainty

# Mission Based Prioritization Demo

SEI PowerPoint Template

# Requirements Elicitation and Splitting

# Learning Objectives

Know when requirements are developed to sufficient detail

Practice approaches to requirements decomposition

Agile in Government

**Requirements Elicitation and Splitting**

# Providing Detail for Requirements

# 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

# 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?*

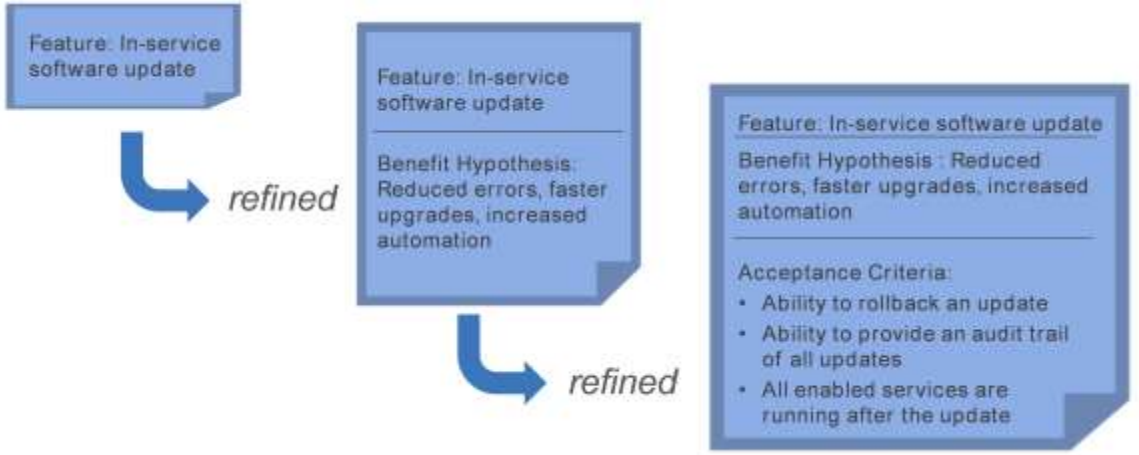
*Product Owners and Managers!*

# Analysis and refinement ensure Features and Stories are ready for implementation

***Epics/Features*** may start as a one-sentence overview with more details added in backlog refinement meetings.

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

This refinement is the "bread and butter" of Product Managers prior to planning meetings



# Making It Real

Need to know more than just functionality

- For stories we have who wants the functionality and why they want it
- Need to have some understanding of user value, urgency, RR/OE, and size
- Need to have understanding of what's most important to be delivered early and what can be delayed

The only way to do this is by:

- Providing templates that guide the customers into providing the right information
  - Epic Hypothesis (describes the work)
  - Lean business case (to assist in making the go/no go situation)
- Asking clarifying questions of the customers (and educating them in the process)

# Epic Hypothesis Statement

| Epic Hypothesis Statement | |
|---|---|
| **Funnel Entry Date:** | <The date that the epic entered the funnel.> |
| **Epic Name:** | <A short name for the epic.> |
| **Epic Owner:** | <The name of the epic owner.> |
| **Epic Description:** | <An elevator pitch (value statement) that describes the epic in a clear and concise way.><br><br>**For** <customers><br>**who** <do something><br>**the** <solution><br>**is a** <something – the 'how'><br>**that** <provides this value><br>**unlike** <competitor, current solution or non-existing solution><br>**our solution** <does something better – the 'why'> |
| | |
| **Business Outcomes:** | <The measurable benefits that the business can anticipate if the epic hypothesis is proven to be correct.> |
| **Leading Indicators:** | <The early measures that will help predict the business outcome hypothesis. For more on this topic, see the Innovation Accounting advanced topic article.> |
| **Nonfunctional Requirements (NFRs):** | <Nonfunctional requirements (NFRs) associated with the epic.> |



https://www.scaledagileframework.com/epic/

Very generic statement that needs to be tailored for VM-Fusion

- Still has the “who” and “why” seen in stories
- Recast in terms of mission
- Other changes as needed

Provides a focus for analysis where the Minimum Viable Product is defined

Is the basis of analysis

# Lean Business Case


https://www.scaledagileframework.com/epic/

Again, needs tailoring for VM-Fusion

Defines the MVP in detail

- The necessary stories that, when complete, can confirm/refute the epic hypothesis

Is the basis for a go/no-go decision

The Lean Business Case provides enough information to be able to prioritize

# 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 specification expression of a team-level derived requirement:

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

*What is the benefit of adding "who" and "why" to low-level backlog items?*

# More on Stories

Stories should satisfy the 3 Cs

- **Card** – The story must be concise e.g., can be written on an index card or a sticky note
- **Conversation** – A promise that, when the story is scheduled to be worked that there will be a conversation between the team and the product owner to fully understand the story
- **Confirmation** – The collection of acceptance criteria that, when satisfied, allow the team to know that the story is functionally correct

Stories should also satisfy then INVEST attributes

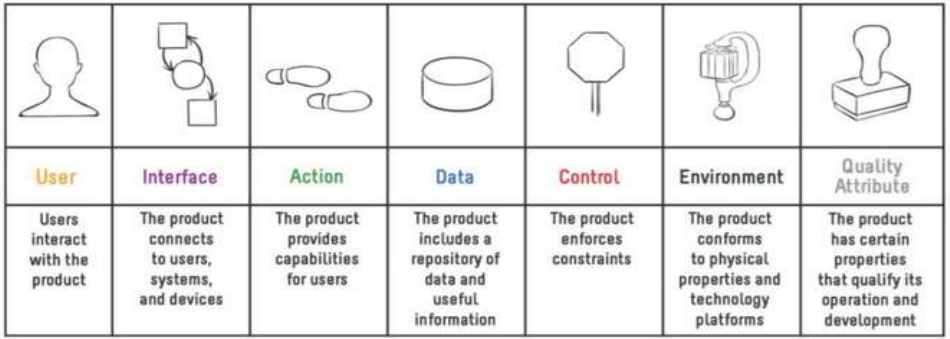
- I – independent of each other
- N – negotiable as they express intent, not a contract
- V – valuable as they represent a piece of visible functionality
- E – estimable as they are small and negotiable
- S – small enough to fit within an iteration
- T – testable as they are understood well enough to know how to test

# Richer Approach

Guide the conversations with the customer by focusing on the 7 Product Dimensions of agile products

- Framework for factors to consider when developing a product

## 7 Product Dimensions



| User | Interface | Action | Data | Control | Environment | Quality Attribute |
|---|---|---|---|---|---|---|
| Users interact with the product | The product connects to users, systems, and devices | The product provides capabilities for users | The product includes a repository of data and useful information | The product enforces constraints | The product conforms to physical properties and technology platforms | The product has certain properties that qualify its operation and development |



See also: https://www.ebgconsulting.com/blog/the-7-product-dimensions-a-guide-to-asking-the-right-questions/

# Dimensions

Users – the person or thing that will interact with the product (e.g., admin, regular user, another system)

Interface – how that interaction will occur (e.g., web page or API)

Action – what the product will do for the users, needs a verb phrase

Data – the input data users will provide and the output data they will get back

Control – rules that govern the data or product (e.g., each user can have only one SSN)

Environment – the technology and infrastructure needed for the product to work correctly

Quality Attribute(s) – often a reflection of the controls (e.g., software cannot permit the user to enter more than one SSN), but should also focus on the “ilities” such as Availability, Reliability, and Maintainability

# Sample User Story

As the Account Manager (**User**), I need to add a dependent (**Action**) to my account management page (**Interface**) to provide access to this dependent (another **Action**).

**(Data)**

- Input Data: Name, ID Number, Relationship status (Husband, Son, Parent, etc.)
- Output Data: Non-Applicable in this case.

**Control (Business rules and/or restrictions)**

- Only a valid ID number can be accepted by system. When it is invalid, block the operation and show error message.
- If the informed ID number is already associated to the Account, block the operation, and show message.

**(Environment)**

- Implement in both web and mobile versions of online banking.

**(Quality Attributes)**

- Not allowed to use an invalid ID Number for the dependent.
- Not allowed to use duplicate ID Numbers for an account.
- Only after being added can the dependent person access the balance for the linked account.

**Requirements Elicitation and Splitting**

# Requirements Splitting

# Splitting Epics and Stories

Split by:

- Workflow steps
- Business rule variations
- Major effort
- Simple/complex
- Variations in data
- Data entry methods
- Deferred system qualities
- Operations (ex., Create, Read, Update, Delete [CRUD])
- Use-case scenarios
- Break-out spike

Summarized from: *Agile Software Requirements*. By Dean Leffingwell

Functional/project/geographic/mission-focused groups

Actors: Analyst Type A, Analyst Type B, Customer

NFRs: ability to share/collaborate with working files

Interfaces: database of user profiles (need to identify user profile)

Acceptance Criteria: CRUD

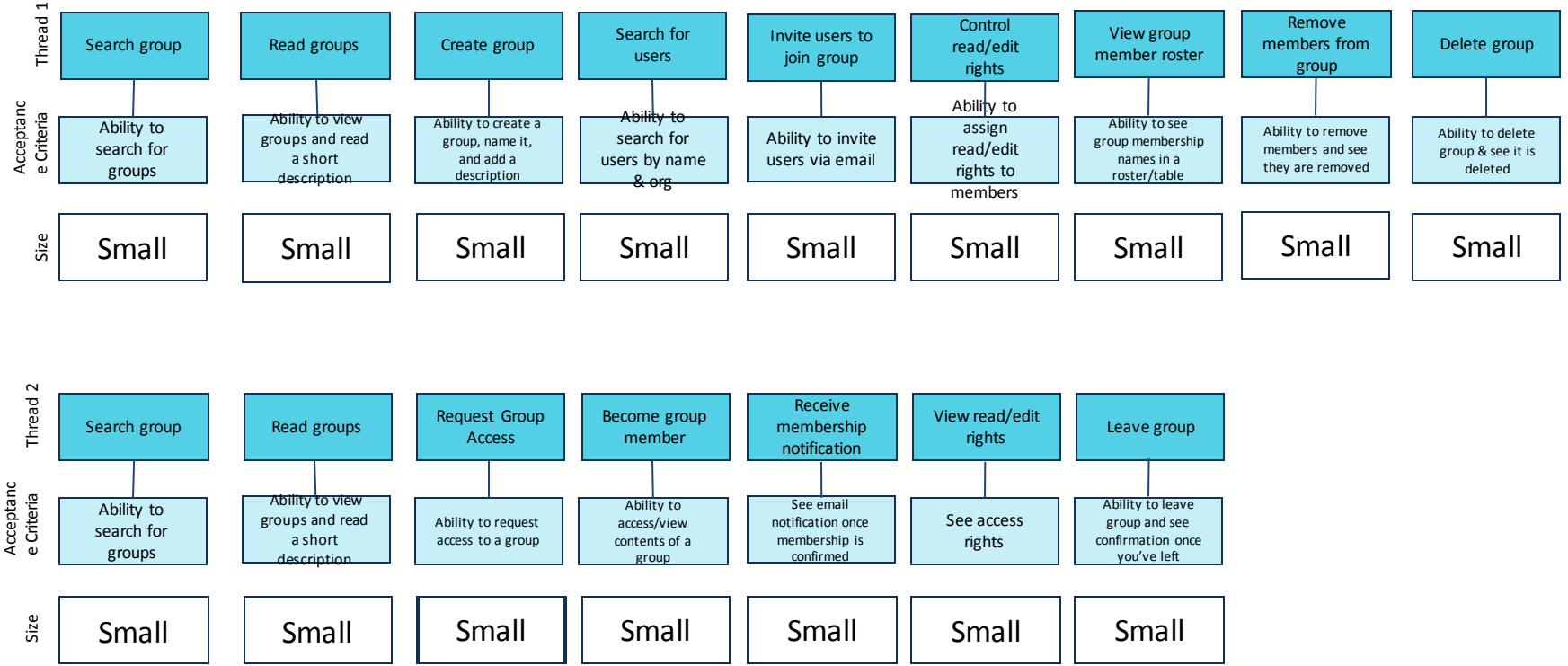
1. Identify Actors
2. Identify Interfaces
3. Identify NFRs
4. Build the Mission Thread
5. Define Acceptance Criteria for Epic / Enabler
6. Define Acceptance Criteria for each Box
7. Define Size for each Box

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Thread 1 | Search group | Read groups | Create group | Search for users | Invite users to join group | Control read/edit rights | View group member roster | Remove members from group | Delete group |
| Thread 2 | Search group | Read groups | Request Group Access | Become group member | Receive membership notification | View read/edit rights | Leave group | | |
| Acceptance Criteria | | | | | | | | | |
| Size | | | | | | | | | |

- Origination: Users want the ability to create groups/teams to work on projects or RFIs and be able to collaborate amongst themselves, but also be able to share the work with other sites and allow them access to the work, and govern access with the ability to assign "read only" and "edit" rights to other users.
- Step 1, the analyst gets an RFI and creates a group to answer the RFI via , chat, or email and include all participants to work and share data within the sites. Step 2, share the work/data/notes with other users outside of the team, which could be other sites or customers, so that they could share information and collaborate.

Functional/project/ geographic/mission -focused groups

# Program Epic - Data sharing within



Pgm Epic

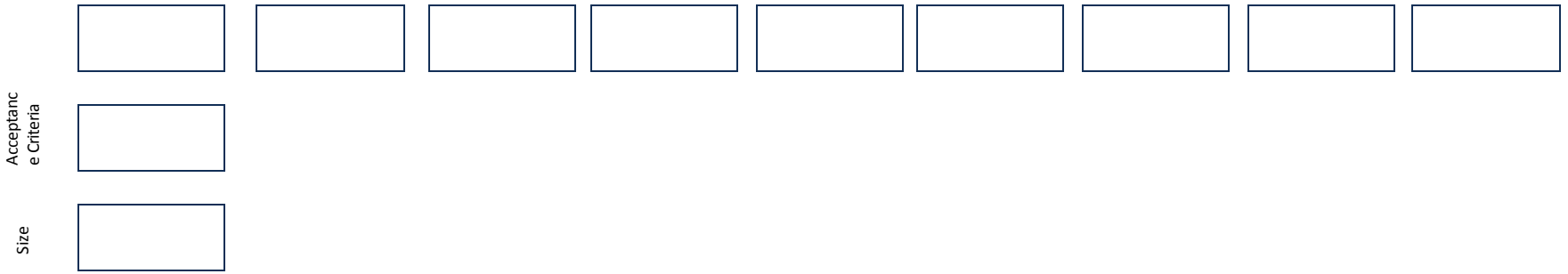
Actors:

NFRs:

Interfaces:

Acceptance Criteria:

1. Identify Actors
2. Identify Interfaces
3. Identify NFRs
4. Build the Mission Thread
5. Define Acceptance Criteria for Epic / Enabler
6. Define Acceptance Criteria for each Box
7. Define Size for each Box



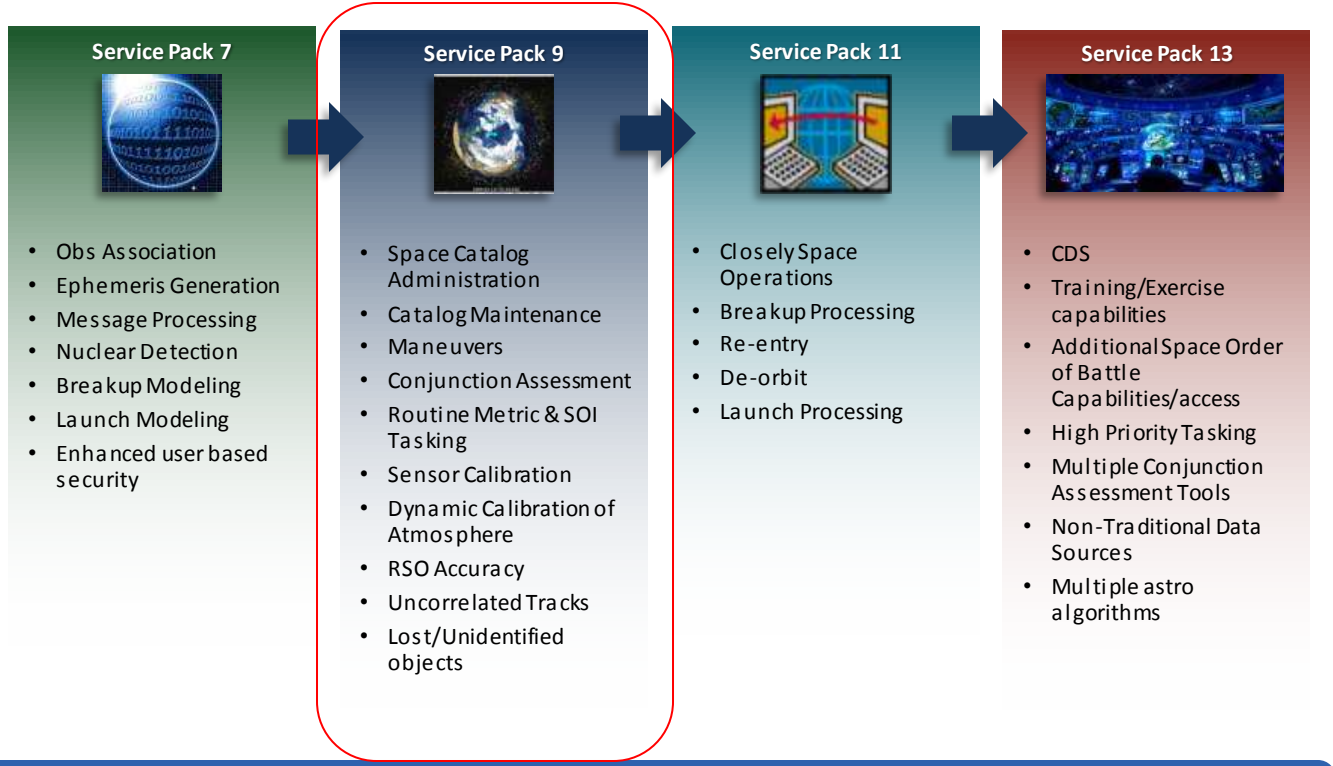
Agile in Government

**Requirements Elicitation and Splitting**

# Requirements – Communicating the Plan

# Letting Leadership Know What to Expect – And When

# XXX Increment 2
# Service Packs-Capability Delivery




Impact for <using organization>: Increased functionality starting as early as 2014, migrate off legacy catalog, and end reliance on XXX by 2015.

# XXX Increment 2
# Service Pack 9: Complete Space Catalog and SAP

**Service Pack 9**

- Catalog Administration
- Catalog Maintenance
- Maneuver
- Conjunction Assessment
- Routine Metric & SOI Tasking
- Sensor Calibration
- RSO Accuracy
- Dynamic Calibration of Atmosphere
- Uncorrelated Tracks
- Lost/Unidentified objects
- SAP (MIT/LL)

## IMPACT TO JSpOC C2 & SSA

- **Catalog Administration** will be coordinated, exhaustive and CM controlled, reducing time required to perform actions, potential data spills, and repeatability. Posture for SISP replacement in future (post Inc 2)
- **Maneuver capability** will report processed maneuvers 1 order of magnitude smaller than XXX, and provide a federated service capable of publishing processed, predicted and detected maneuvers
- **Catalog Maintenance** will be SP-based using more accurate observation association techniques and streaming in nature vice batch catalog updates
- **Conjunction Assessment** will be event-based vice detection-based, streaming in nature resulting in more objects compared more frequently, automated and largely a background process eliminating tedious and needless operator steps
- **RSO Accuracy** will be automated, dynamic and configurable via a dedicated UDOP report generation screen
- **Routine Metric &SOI Tasking** will be automated on a single system eliminating data and tape transfers, modular to support future improvements
- **Lost/Unidentified objects** will leverage Nyx-based capability for more effective UCT and Lost-list processing

Impact for <user organization>: Complete establishment of …