Agile Systems Development

In a Medical Environment

Meet Jama

Cary Bryczek

Jama Software

Requirements & Test Management

“Simplify Complex Product Development”

https://www.jamasoftware.com/

Meet Kelly

Agile Quality Systems

Kelly Weyrauch

Agile Quality Systems

Agenda (2hrs)

Organizing your Agile Data (1.5 hour)

- Intro
- Mechanisms, Roles, Cadence
- Your System

Agile Reqs (1 hour)

- Taxonomy, Writing Tips, Examples
- Group Exercise – Stories
- Team Review

Transitioning to Agile (1 hour)

- Group Exercise: Transform real examples to Agile
- Team Discussion: Challenges and Barriers

Close and Retrospective (½ hour)

What is Agile?

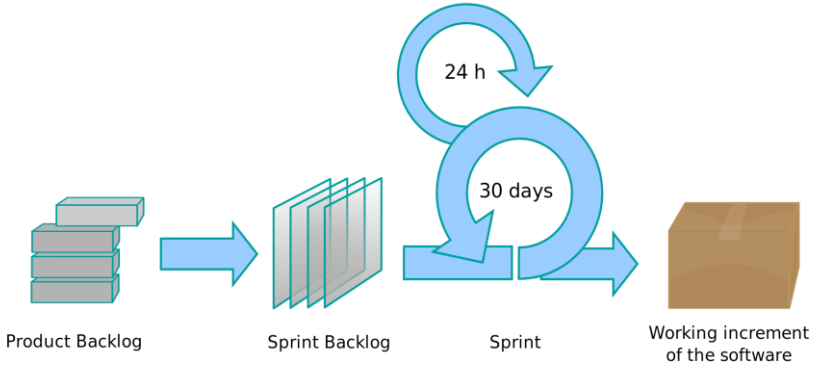
We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

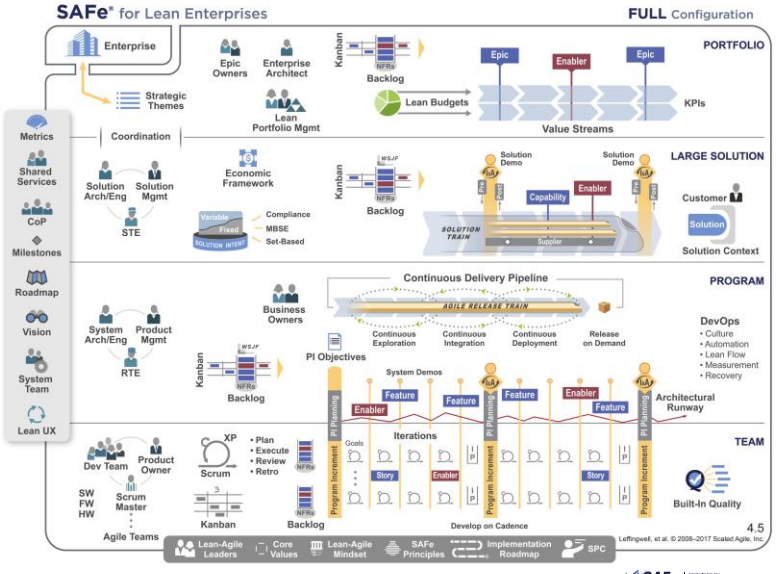
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

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

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.

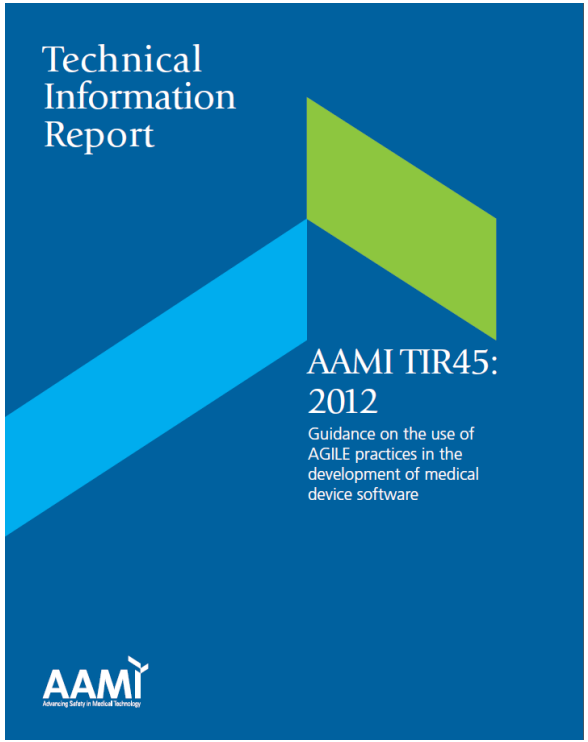
Manifesto for Agile Software Development, www.agilemanifesto.org





AAMI TIR45:2012 Guidance on the Use of AGILE Practices in the Development of Medical Device Software

- Industry and FDA participation
- FDA Recognized Consensus Standard



The Agile Manifesto

The Agile Manifesto

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

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

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

Manifesto for Agile Software Development, www.agilemanifesto.org

Principles behind the Agile Manifesto

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.

Manifesto for Agile Software Development, www.agilemanifesto.org

Agile Mechanisms

Layers / Kinds of Backlog Items

(as defined by the Scaled Agile Framework, SAFe®)

- Four Levels
  - Portfolio-Level Epics
  - Large-Solution-Level Capabilities
  - Program-Level Features
  - Team-Level Stories

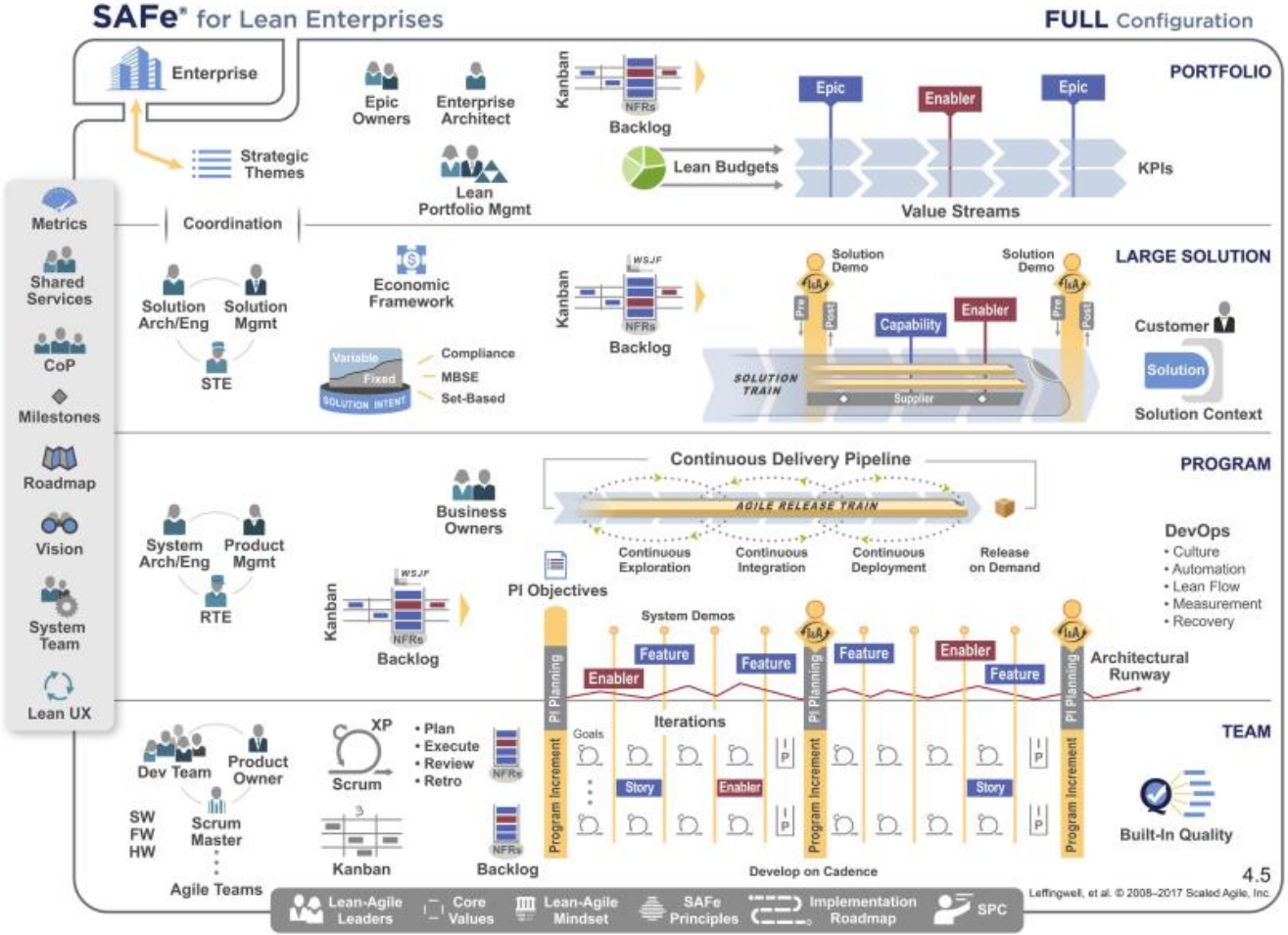
- Two Perspectives
  - Customer-Facing, Business Value
  - Solution-Facing, Enablers

	Epics (Portfolio)	Capabilities (Large Solution)	Features (Program)	Stories (Team)
Responsible	Portfolio Management Enterprise Architect (+PM, +SE)	Solution Management, Solution Architect (+PM, +SE)	Product Management (+PO, +SE)	Product Owner (+Team)
Provides Value to	Customers, Business	“Who” of the Capability, Customers, Enterprise Architect, The Epic	“Who” of the Feature, Business Owners, Solution Architect, The Capability/Epic	“Who” of the Story, System Architect, The Feature
Delivered by	N/A, delivered through Capabilities & Features	Agile Teams, System Team, Specialty Team?	Agile Teams, System Team	Agile Teams
Delivered when	Depends - When all Capabilities/Features complete? Only some?	When all Features complete (1 or more Increments)	Each Program Increment	Each Sprint
Demoed	N/A, Demo done with lower layers	System Demo, Validation Studies	System Demo	Team’s Sprint Demo
Content	Lightweight Business Case	Description & Benefit, Acceptance Criteria, Definition Of Done	Description & Benefit, Acceptance Criteria, Definition Of Done	Story Pattern, Acceptance Criteria, Definition Of Done

Solution-Facing Backlog Items

- SAFe® term: “Enablers”
  - Enabling the “Architectural Runway” upon which customer-facing value can be delivered.

- Infrastructure (Development, Product)
- Debt
- Spikes (Definition, Technical, Decision)
- System Integration
- Quality System Satisfaction
- …



SAFe® PROVIDED BY SCALED AGILE

Sprint / Increment

Definition: a fixed time box (e.g. 2 weeks) where team build an incremental business or product functionality.

Purpose:

- Delivers value to customers more quickly
- Allows faster feedback from customers
- Allows teams to incorporate feedback and adjust priorities and improve process

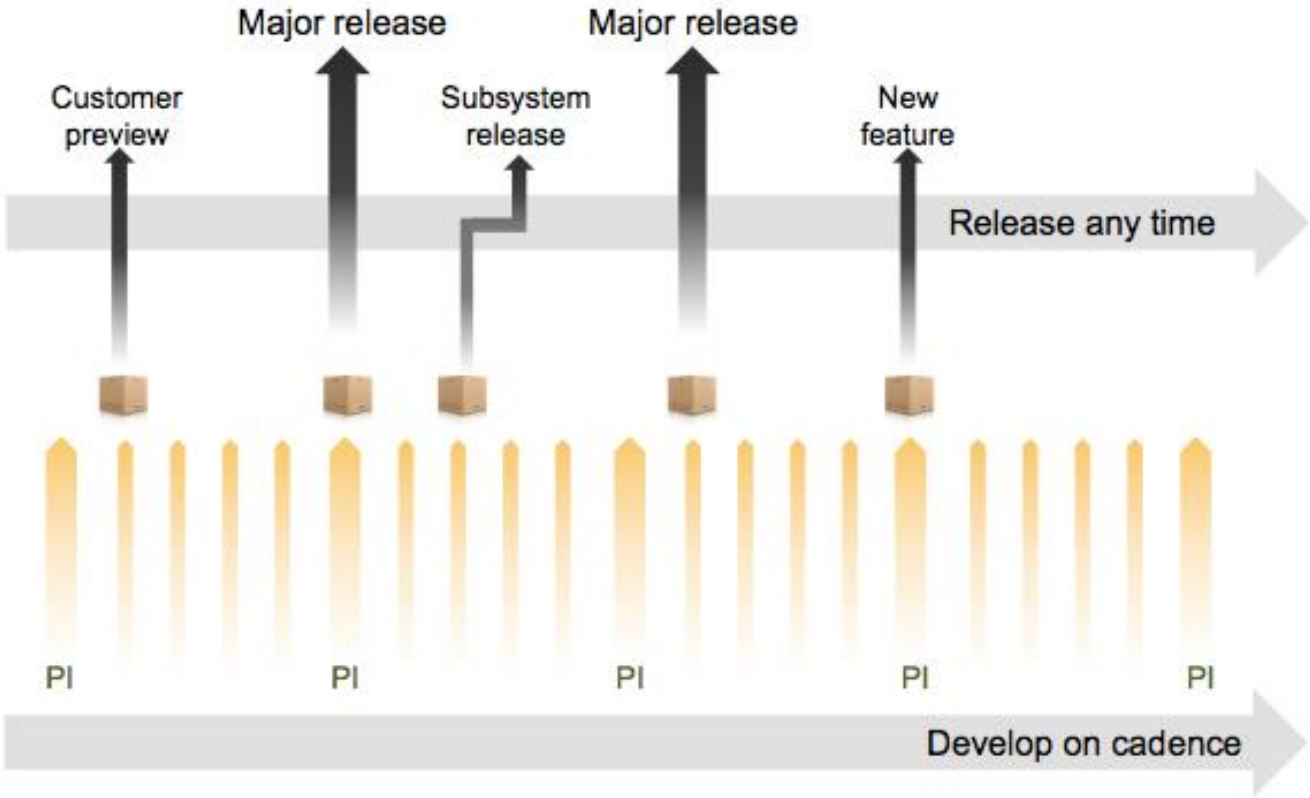
Release

Definition: Traditionally, when value is delivered to a customer. In Agile, **a release could be anytime.**

Examples

- After several sprints and demos to customers, functionality is packaged and delivered to production
- After each sprint, developed value is released to customers
- For mature, continuous integration organizations, release every time code is checked in!

Scaled Agile Framework® (SAFe®)



Source: http://www.scaledagileframework.com/develop-on-cadence-release-any-time/

Your System?

ISO 15288 System Life Cycle Processes

Technical processes

- Business or mission analysis process
- Stakeholder needs & requirements definition process
- System requirements definition process
- Architecture definition process
- Design definition process
- System analysis process
- Implementation process
- Integration process
- Verification process
- Transition process
- Validation process
- Operation process
- Maintenance process
- Disposal process

Technical management processes

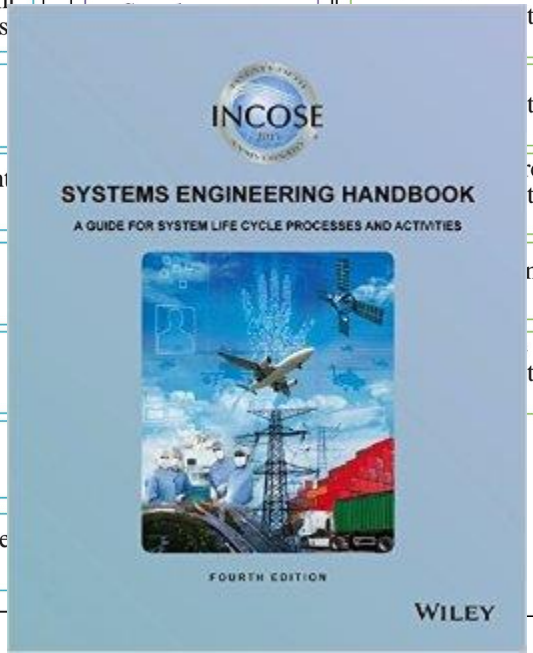
- Project planning process
- Project assessment and control proces
- Decision management process
- Risk management process
- Configuration management process
- Information management process
- Measurement process
- Quality assurance process

Agreement processes

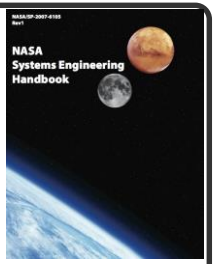
- Acquisition process

Organizational project-enabling processes

- Life cycle model management process
- Infrastructure



The Systems Engineering Engine



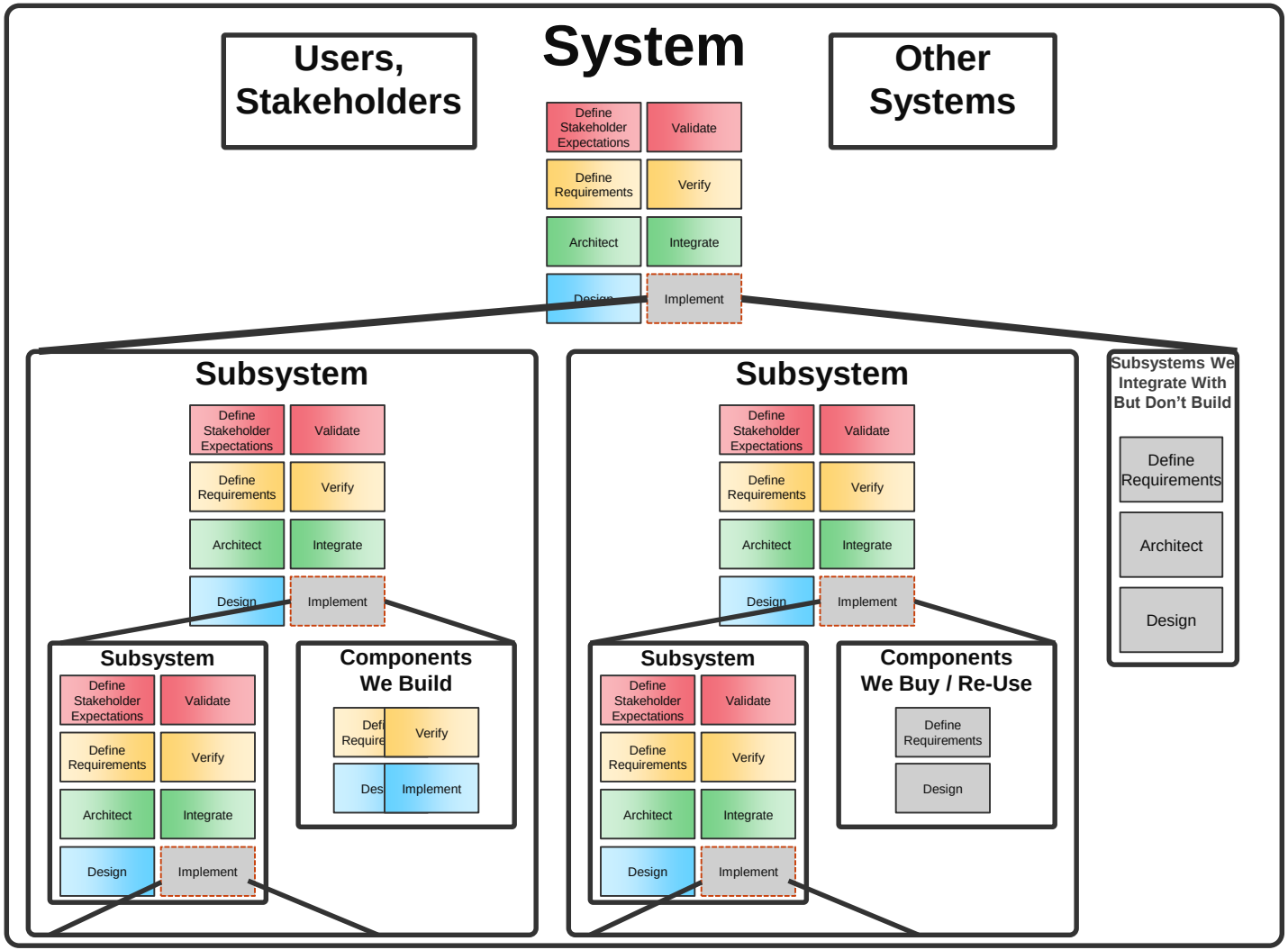
System Development Processes

System Design	Product Realization
Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Integrate
Design	Implement

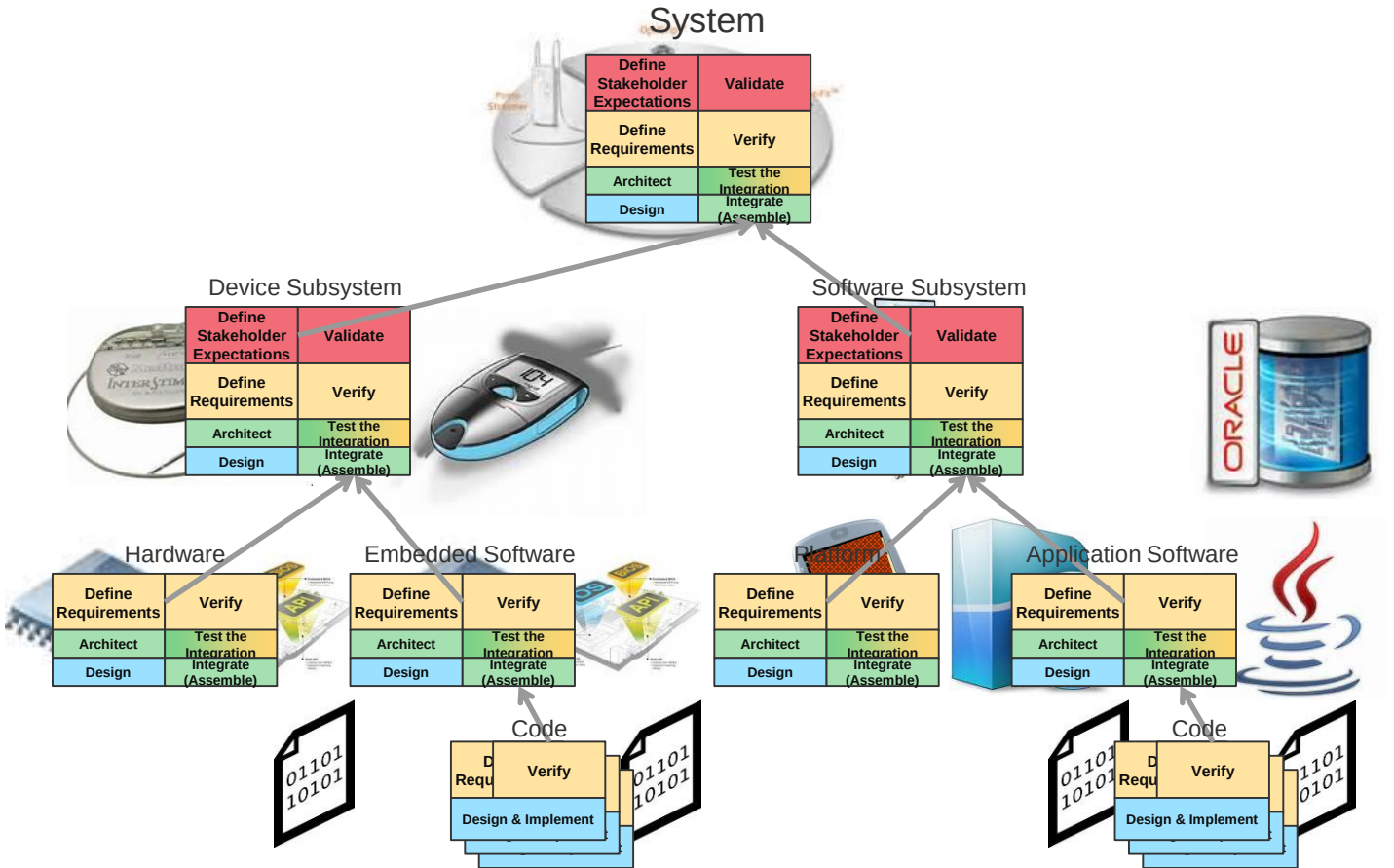
Technical Management Processes

- Technical Planning
- Requirements Management
- Interface Management
- Risk Management
- Configuration Management
- Data Management

Adapted from: NASA Systems Engineering Handbook, NASA/SP-2007-6105


Users, Stakeholders
System
Other Systems
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Subsystem
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Subsystem
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Components We Build
Verify
Implement
Subsystem
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Components We Buy / Re-Use
Define Requirements
Design
Subsystems We Integrate With But Don't Build
Define Requirements
Architect
Design

What is the System?
And What are the Activities & Deliverables?



Exercise in Jama

1. Draw a block diagram of your System.
2. Identify levels of requirements
3. Identify levels and kinds of testing

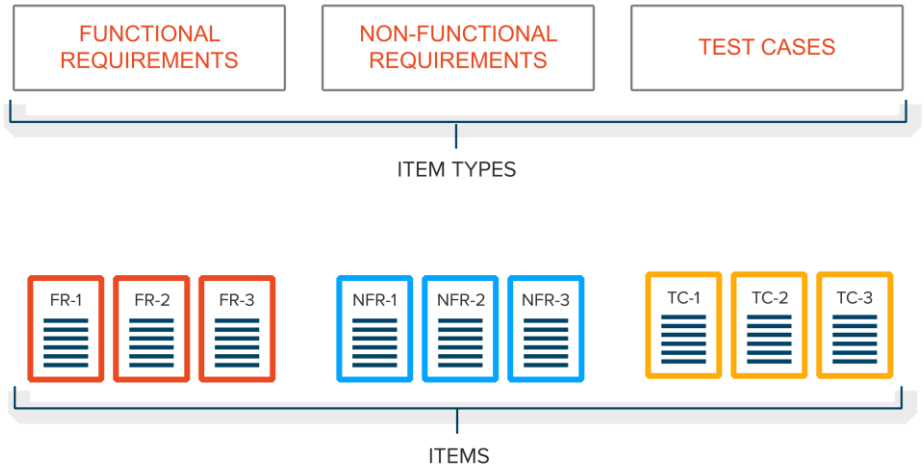
Organizing Information in Jama

Item Type:

- How the types of information that exist in a project are defined and categorized in Jama (functional requirements, user stories, test cases, etc.)
- Created and/or customized by an Organization Admin

Item:

- A single unit of defined and organized information a single functional requirement, non-functional requirement, or test case)

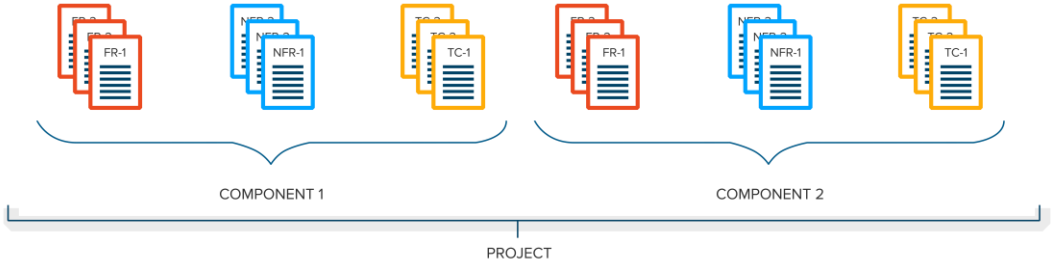


Set:

- A structural container of like items
- May contain folders, Text Items

Component:

- A structural container used to organize a project
- Each Component may contain multiple Sets of items



Project: The highest level of organization. A project, product, application, etc.

May contain: Components, Sets & Text Items

Component: Logical subset within a project

May contain: Components, Sets & Text Items

Set: A grouping of like items

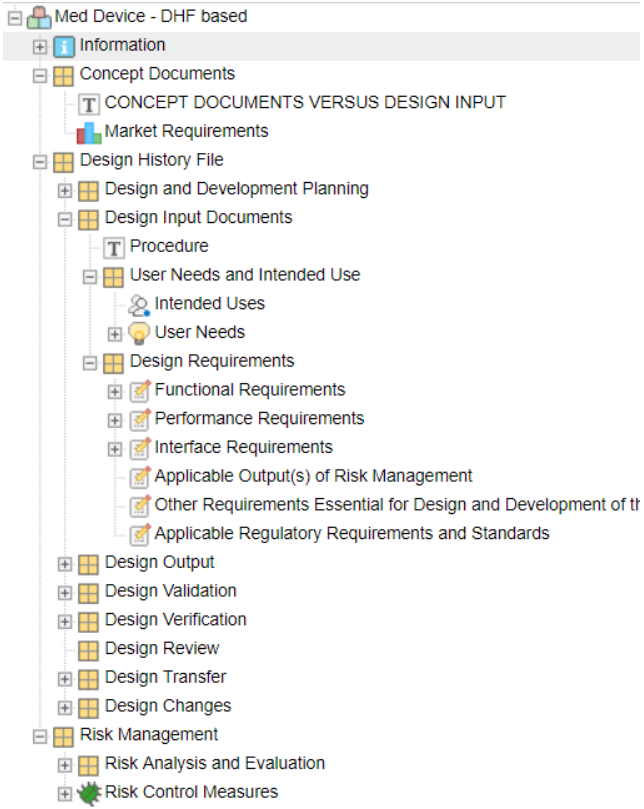
May contain: Folders, Text Items & Items

Folder: Logical subset within a set

May contain: Folders, Text Items & Items

Item: An individual artifact with fields

Text Item: Provides context



Project: The highest level of organization. A project, product, application, etc.

May contain: Components, Sets & Text Items

Component: Logical subset within a project

May contain: Components, Sets & Text Items

Set: A grouping of like items

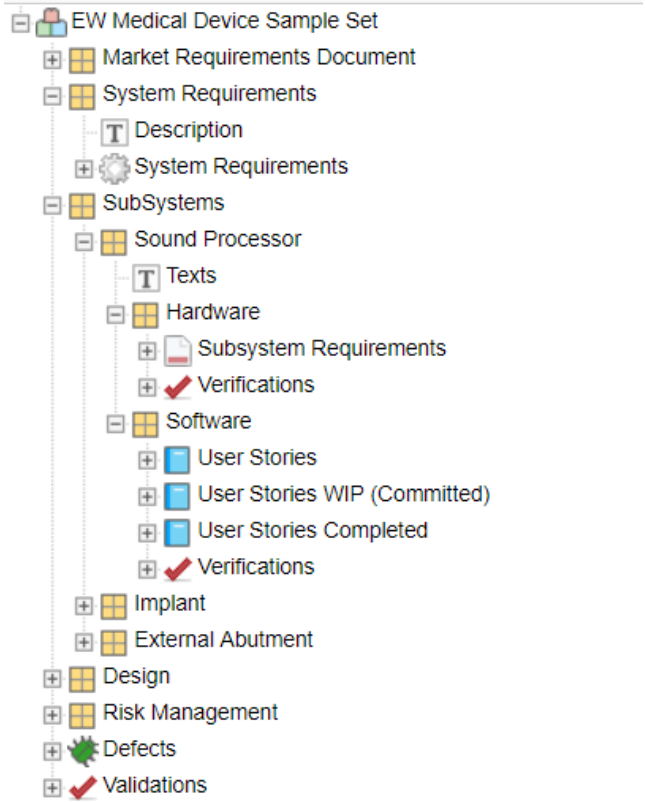
May contain: Folders, Text Items & Items

Folder: Logical subset within a set

May contain: Folders, Text Items & Items

Item: An individual artifact with fields

Text Item: Provides context



Project: The highest level of organization. A project, product, application, etc.
May contain: Component

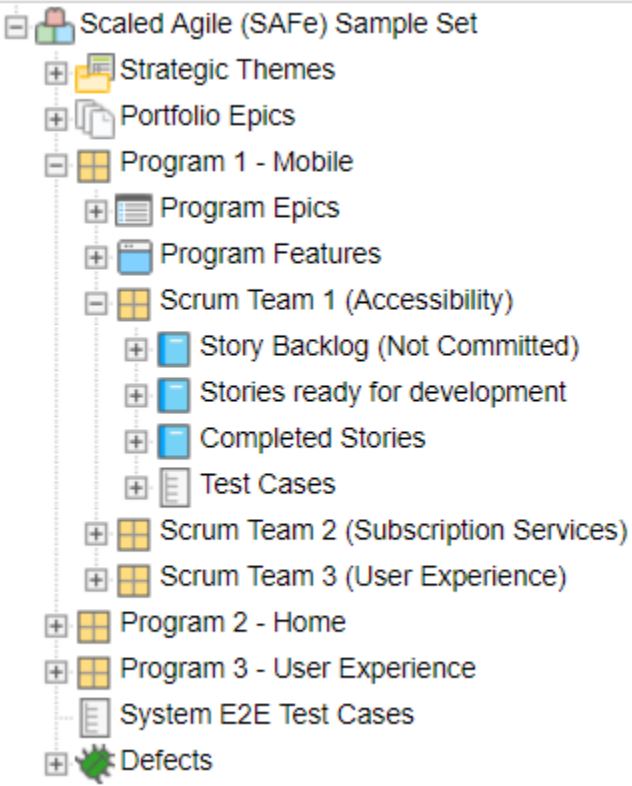
Component: Logical
May contain: Component

Set: A grouping of lik
May contain: Folders, Tex

Folder: Logical subse
May contain: Folders, Tex

Item: An individual a

Text Item: Provides context



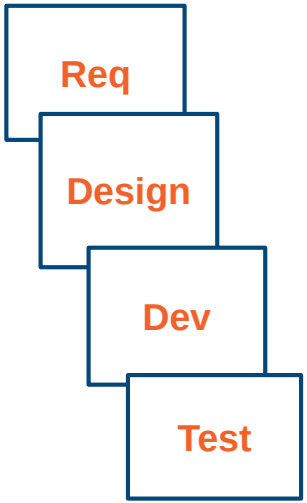
Cadence of Writing Requirements?

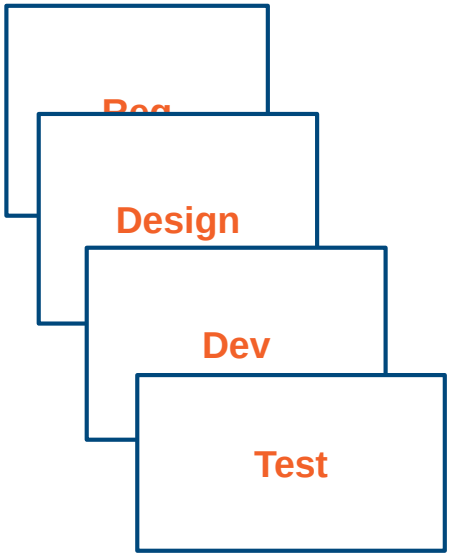
Large System Reqs Doc

Multiple Sub-System Spec

Long Dev Lifecyle

Long V&V Cycle





Comprehensive documentation (if needed) grows over time and complete after several iterations

Methods of Eliciting Requirements

Customer Feedback

- Qualitative: regularly scheduled customer interviews and feedback sessions
- Quantitative: “In Product” feedback and NPS surveys

Innovation

- Market Research
- Customer Obsession

Tips

Think in terms of the “problem” rather than the “solution”

Include team members in customer interviews

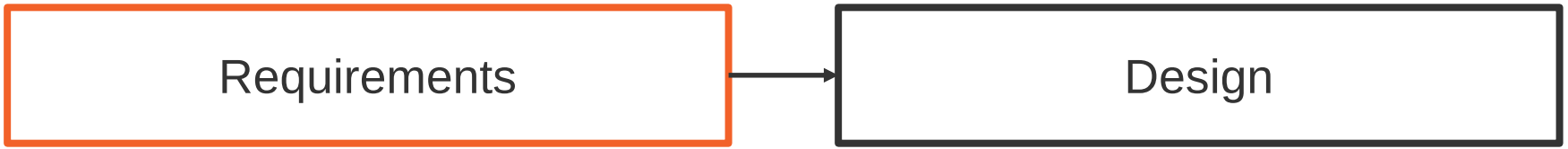
Anti-Patterns to watch for

- The entire project is spec'd out in great detail before engineering work begins
- Thorough review and iron-clad sign-off from all teams before work even starts
- Designers and developers don't know when requirements have changed
- The product owner writes requirements without the participation of the team

Atlassian: https://www.atlassian.com/agile/requirements

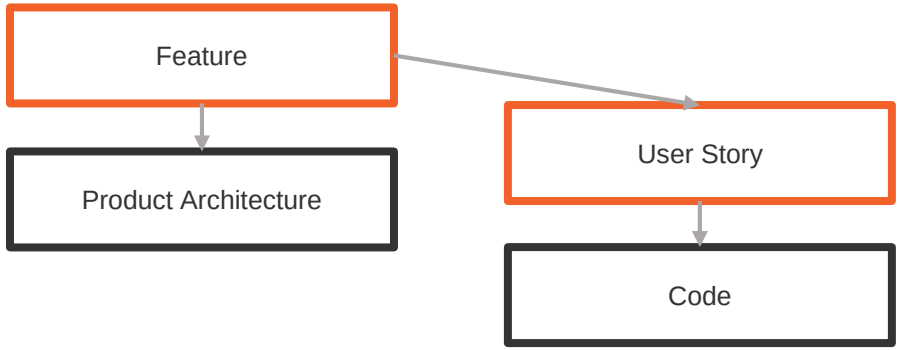
So when you say “requirements”…

Requirements vs Design



Describes the **NEED** for something

A **RESPONSE** to the requirements that describes something that meets the requirements

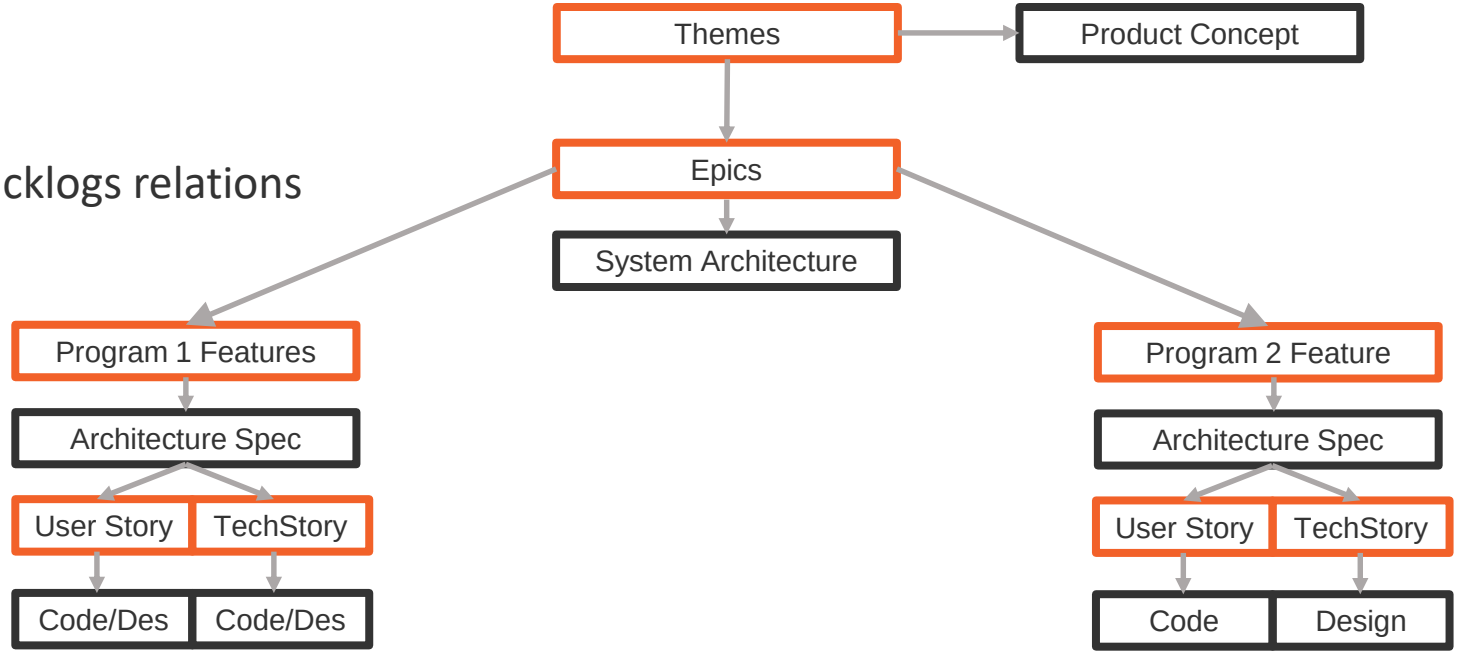


Kelly's layer

Sys
Sub
Comp

Establish a clear hierarchy.

Backlogs relations



Building Something Complex?

Common Agile Requirements Taxonomy

Theme

Definition

- High-Level Goal
- Strategic Objective
- Innovation or Differentiator
- May span multiple releases and teams

Examples:

- Lower distribution center costs
- Implement new billing system
- Deliver new upgrades on a more frequent basis

Related to:

- Epics

Epic

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:	• •

SAFe template for an Epic

Feature

Definition

- Short description of a system feature and benefit
- Easy for business stakeholders to understand
- Higher-level than a User Story – it may span multiple user roles/user stories
- Used in some methodologies to bridge Epics to Stories

Example:

- In service software update
  - Benefit: Reduces software downtime
  - Acceptance Criteria: 1) Choice of Automated or Manual 2) Rollback option after update 3) support configuration through GUI

Related to:

- Higher-Level Epics
- Lower-Level Stories

User Story

Definition

- Derived from Epics and Features
- Fits within 1 sprint
- As a <role> I can <activity> so that <business value>
- Role can be human or system

Example:

- As an administrator I can configure automated or manual software updates so that I can deliver new features to my users
  - Acceptance Criteria: 1) Pick-List option of Automated or Manual 2) If Automated, choice of day/time to pull updates

Related to:

- Higher-Level Features/Epics

Sample Agile Models in Jama - Replace with Customer trace model?

Tips for Writing “Good” Agile Requirements

Requirement Rules?

Examples:

1. An Agile requirement may not contain the word “and.” An “and” indicates the presence of two requirements, which must be separated.

2. Requirements must be written in the form of a scrum user story

3. A requirement may not contain more than 22 words.

Strict rules are not realistic. The “golden rule” of requirements is:

Clear and effective communication among your stakeholders.

Stories to Choose

Is backlog planning?
Or is backlog the artifacts/req themselves

The output of Stories are requirements (support baselines and version management) “snapshot that represents the state of the final product”

Authoring Requirements ACC

Recommendation – Use a Template

User type:
As a [user class or actor name]...

Result type:
... I need to [do something]...

Object:
... [to something].

Qualifier:
…so that I can do [response time goal or quality objective]

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:	• •

SAFe template for an Epic

Recommendation – Use Active Voice

Passive: "As a user, I need to change the state of a requirement, so that **it** is logg**ed** in the event history."

Whenever possible, recast such requirements in the much clearer active voice

Active: "As a user, I need to change the state of a requirement, so that I can see the new state and the time of the state change in the event history log."

Recommendation – Be Positive!

Negative

- Feature: The migration tool will not migrate users with more than three accounts
- As a Project Admin, I should not have ability to change the web user accounts.

Positive

- Feature: The migration tool will migrate only users with one or two accounts
- As a System Administrator, I need to change web user accounts so that I can change user's desired email address and display name

Recommendation – Avoid “ly” words

Adverbs provide ambiguity:

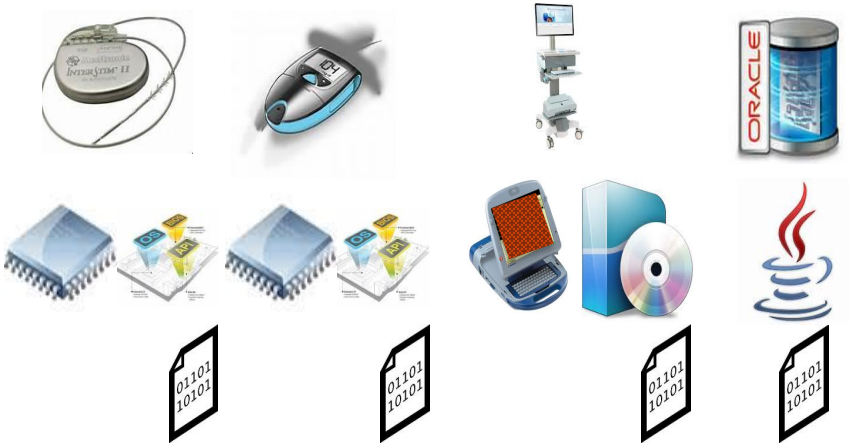
- …so that I can provide a reasonab**ly** predictable end-user experience.
- …so that I can offer significant**ly** better download times.
- …so that I can optimize upload and download to perform quick**ly**.

It’s hard to test ”quickly” or “reasonably.”

When possible - include a qualifying objective (acceptance criteria) that is measurable and testable.

Exercise in Jama

1. Write some User Stories in Jama and use different templates



Refining the Requirements

Recommendation - Review and Discuss

The point is a shared understanding of the need.

Taking time up-front to review requirements:

- Gives you feedback and makes you a better auth
- Increases shared understanding amongst team
- Helps define acceptance criteria and ensure testability
- Reduces surprises and missed requirements

Methods: Face-to-face conversations, Jama comments, Jama reviews (collection of requirements).

Recommendation – Build Detail Iteratively

We need details but it can be "negotiated" throughout 3 major phases:

- Initial draft
- Backlog Grooming & Iteration Planning (more detailed)
- Test Development (e.g. for Stories, all acceptance criteria defined)

Too Detailed	Just Right
As a team member, I can click a red button to expand the table to include detail, which lists all the tasks, with rank, name, estimate, owner, status so that I understand development progress	As a team member, I can view the iteration's stories and their status with main fields so I understand development progress *<acceptance criteria of specific fields defined later>*

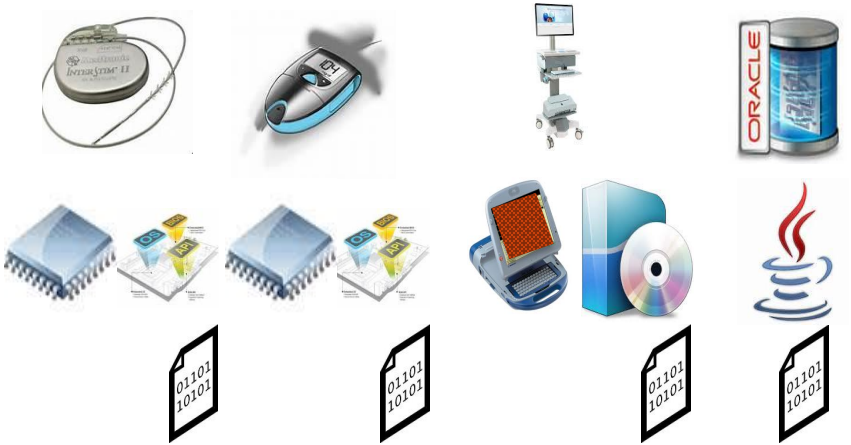
Anti-Patterns to watch for

- The entire project is spec'd out in great detail before engineering work begins
- Thorough review and iron-clad sign-off from all teams before work even starts
- Designers and developers don't know when requirements have changed
- The product owner writes requirements without the participation of the team

Atlassian: https://www.atlassian.com/agile/requirements

Exercise in Jama

1. Create some comments on items
2. Create a Review



Summary: Agile Requirements

- Any hierarchy that describes the “**need**” - not the ”how”
- Gathered, prioritized, improved on a regular cadence
- May be owned by a specific role (e.g. Product Owner) but is a team effort to author and refine
- Constructive feedback, co-authoring with colleagues, and conversations can help anyone become a better reqs writer.

BREAK: 10 min

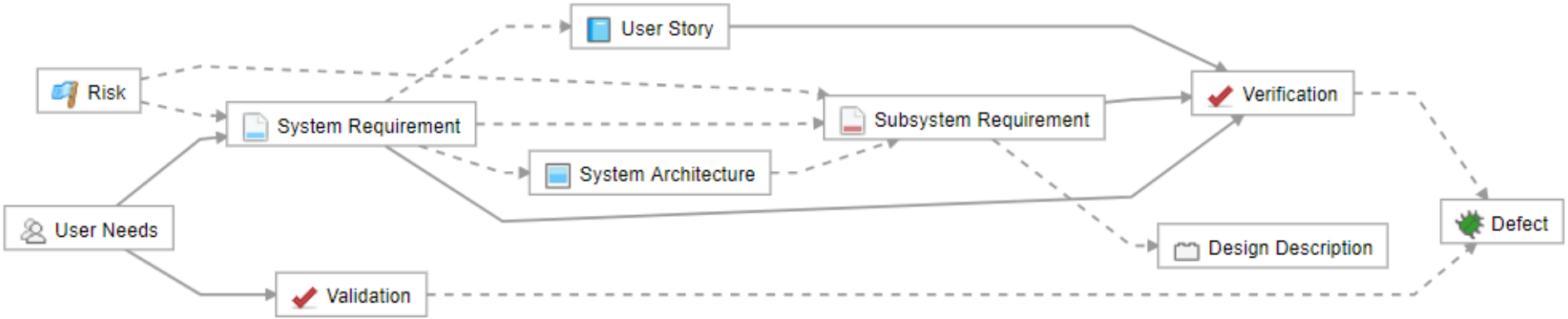
Traceability and V&V

“Traceability” Sounds waterfall but when done right, enables agility and fast response to change

Solution Traceability

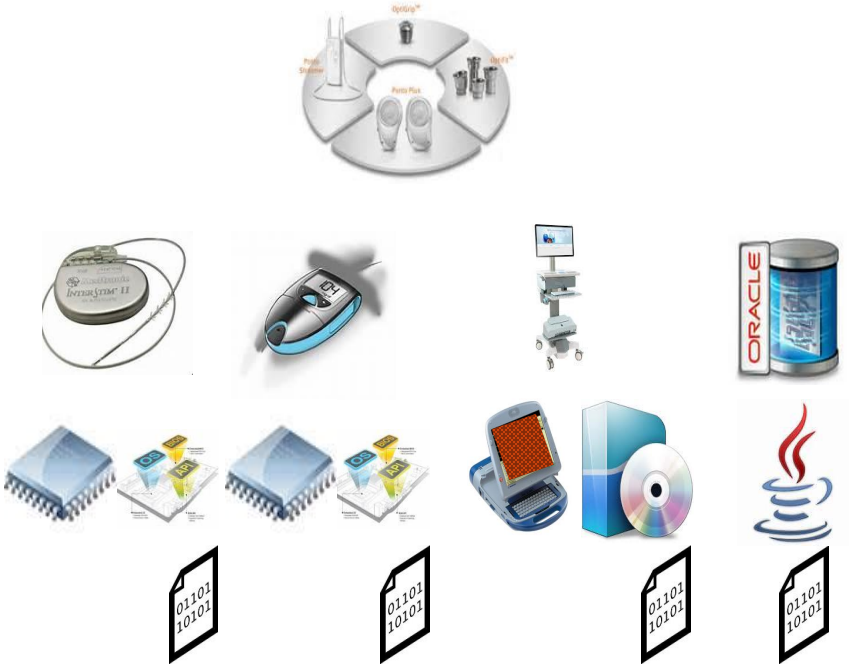
Large and Complex Systems often need to track much more than User Stories

Solution Traceability



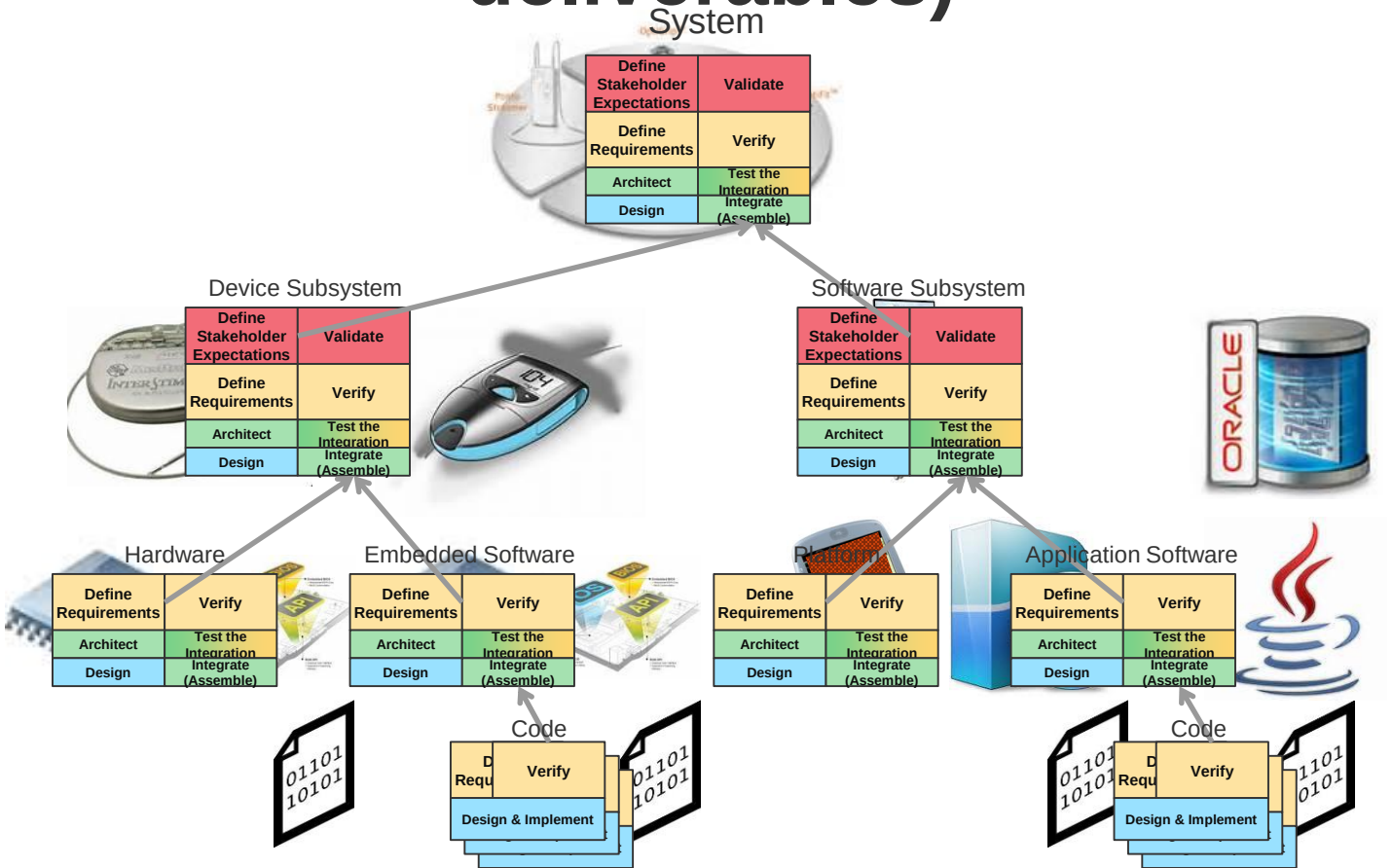
Exercise in Jama

1. Establish Requirements Traces
2. Establish Testing Traces
3. Identify gaps in Traces

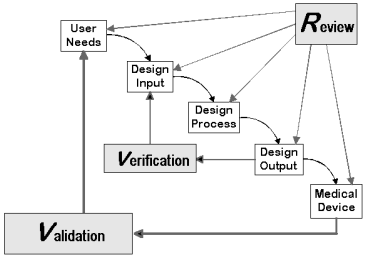


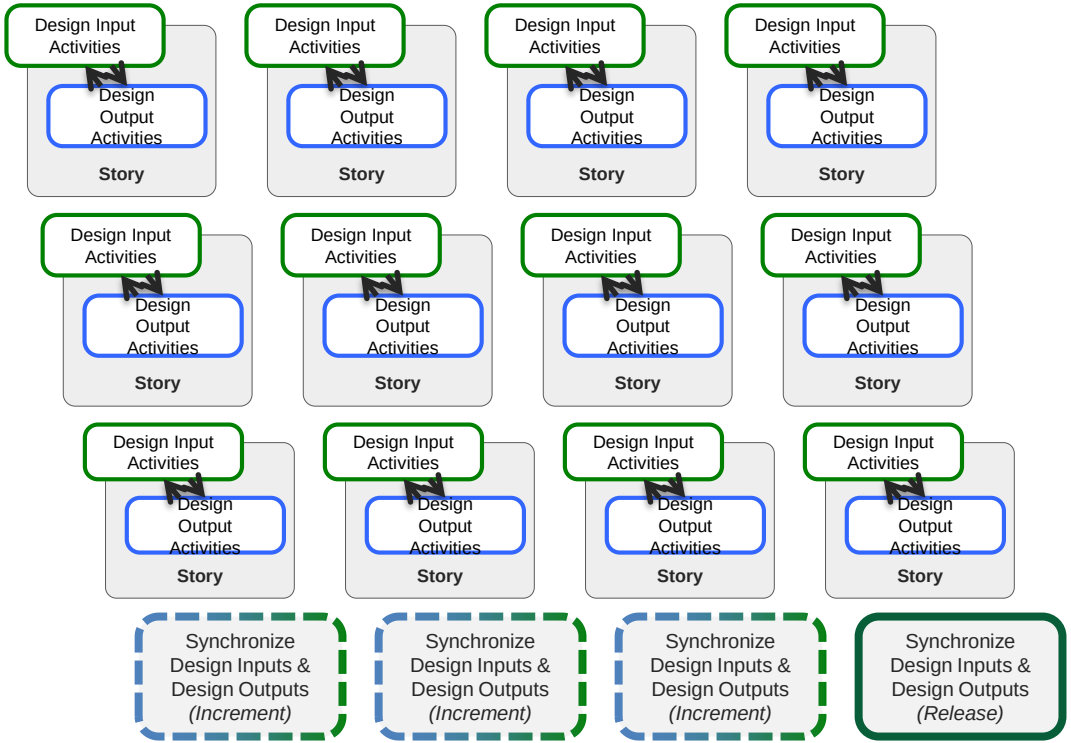
System Activities and Deliverables

System Activities (and deliverables)

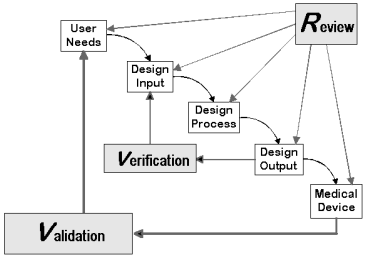


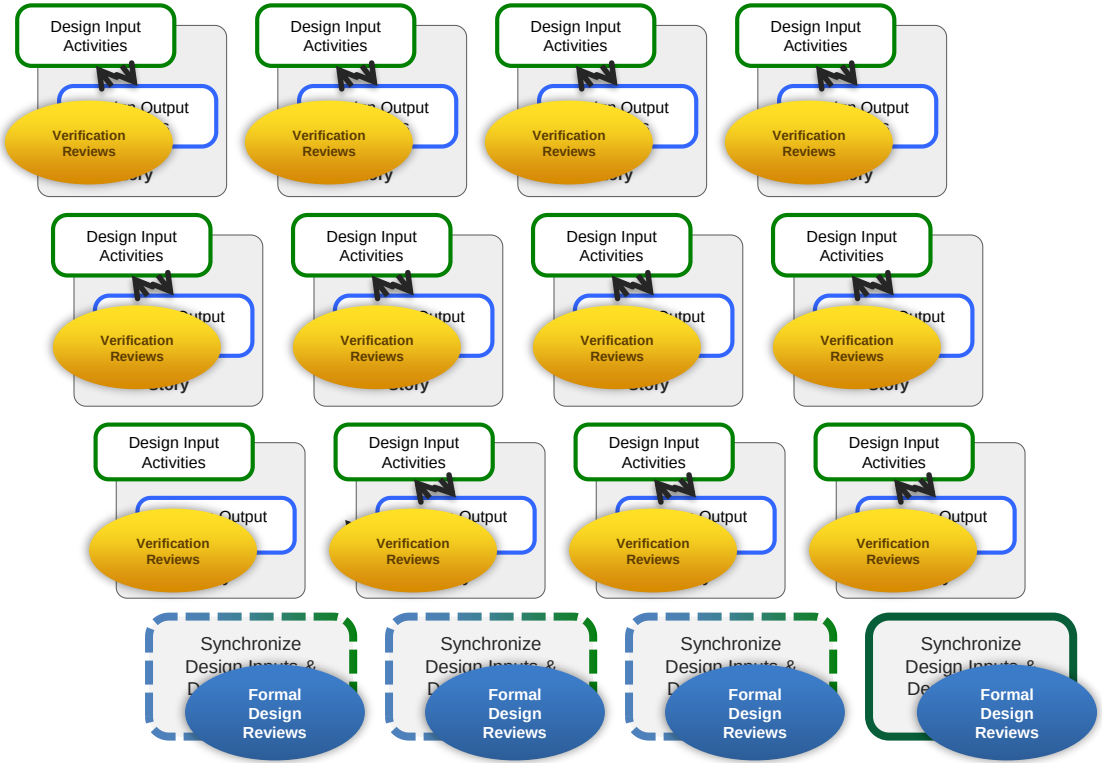
Synchronization Activities from TIR45:2012 Design Inputs & Design Outputs



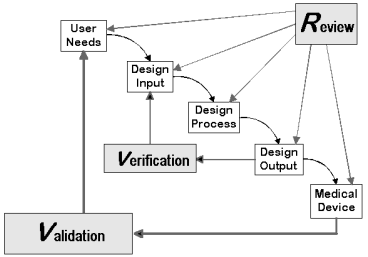


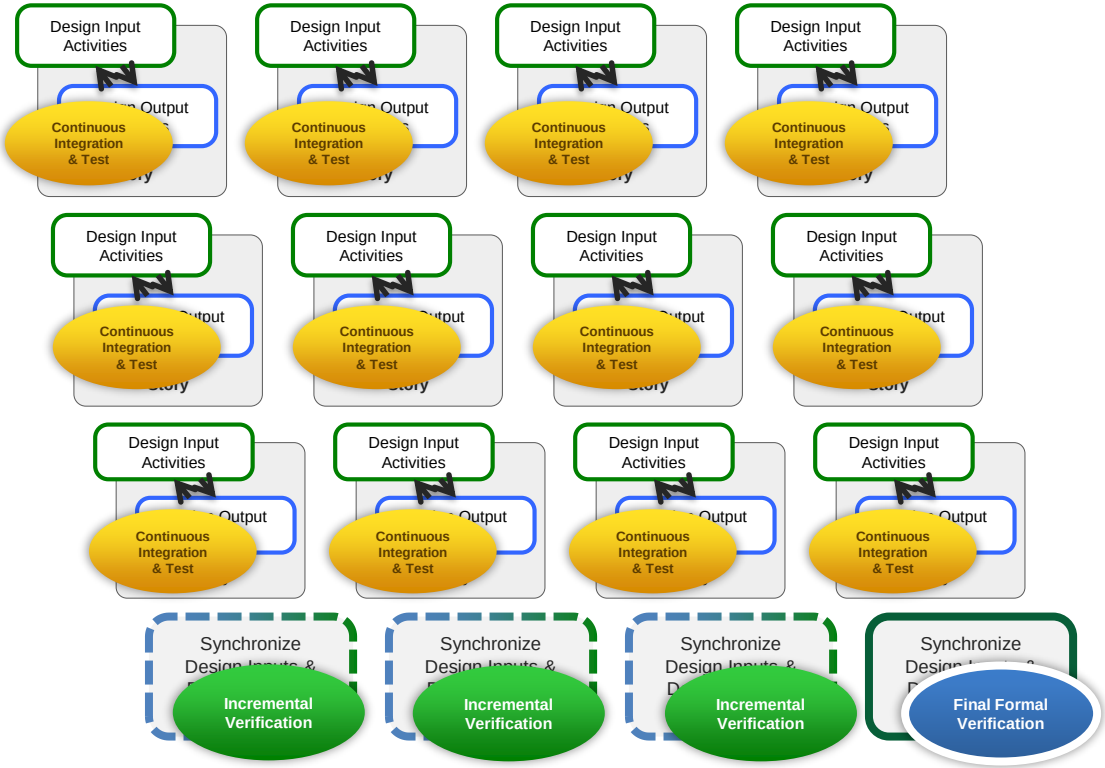
Synchronization Activities from TIR45:2012
Verification Reviews & Design Reviews



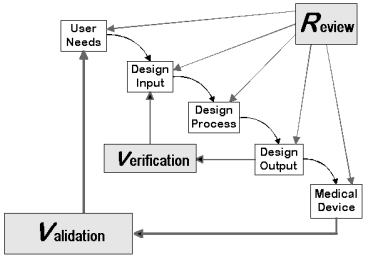


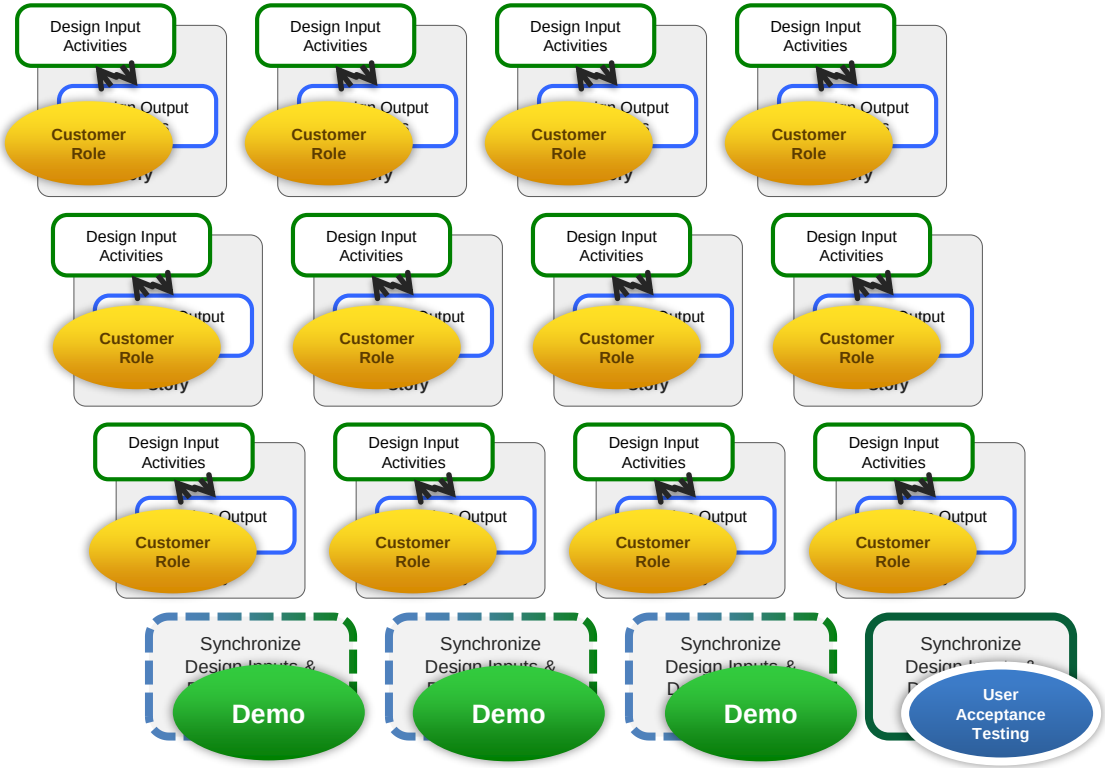
Synchronization Activities from TIR45:2012 Verification Testing





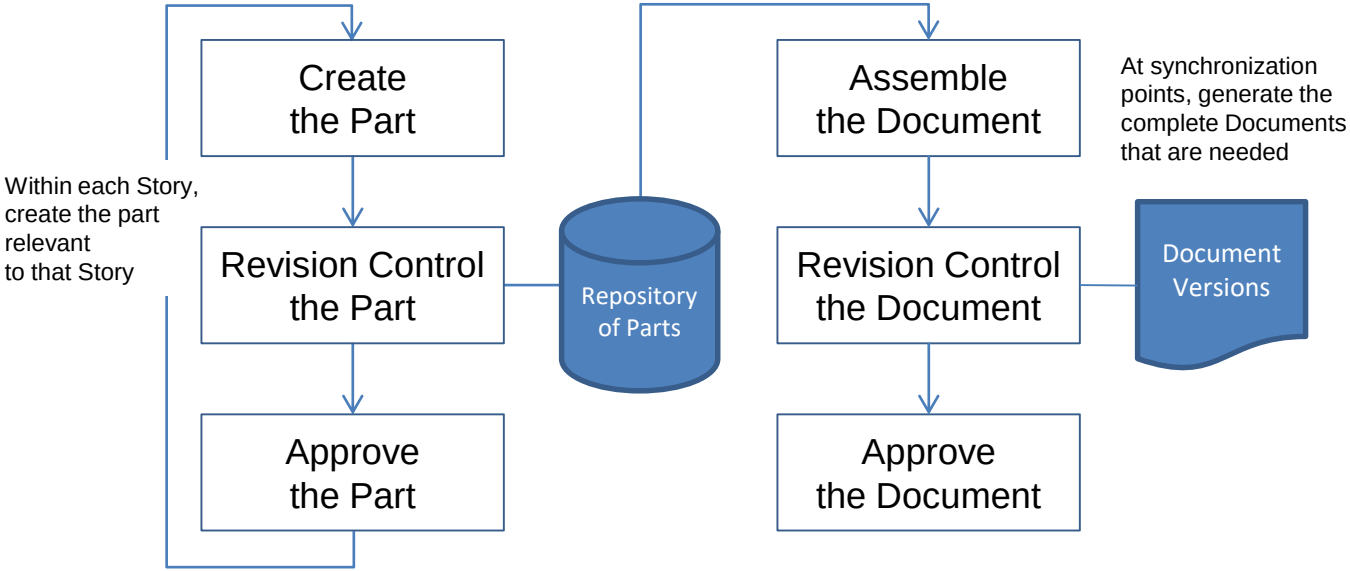
Synchronization Activities from TIR45:2012 Design Validation

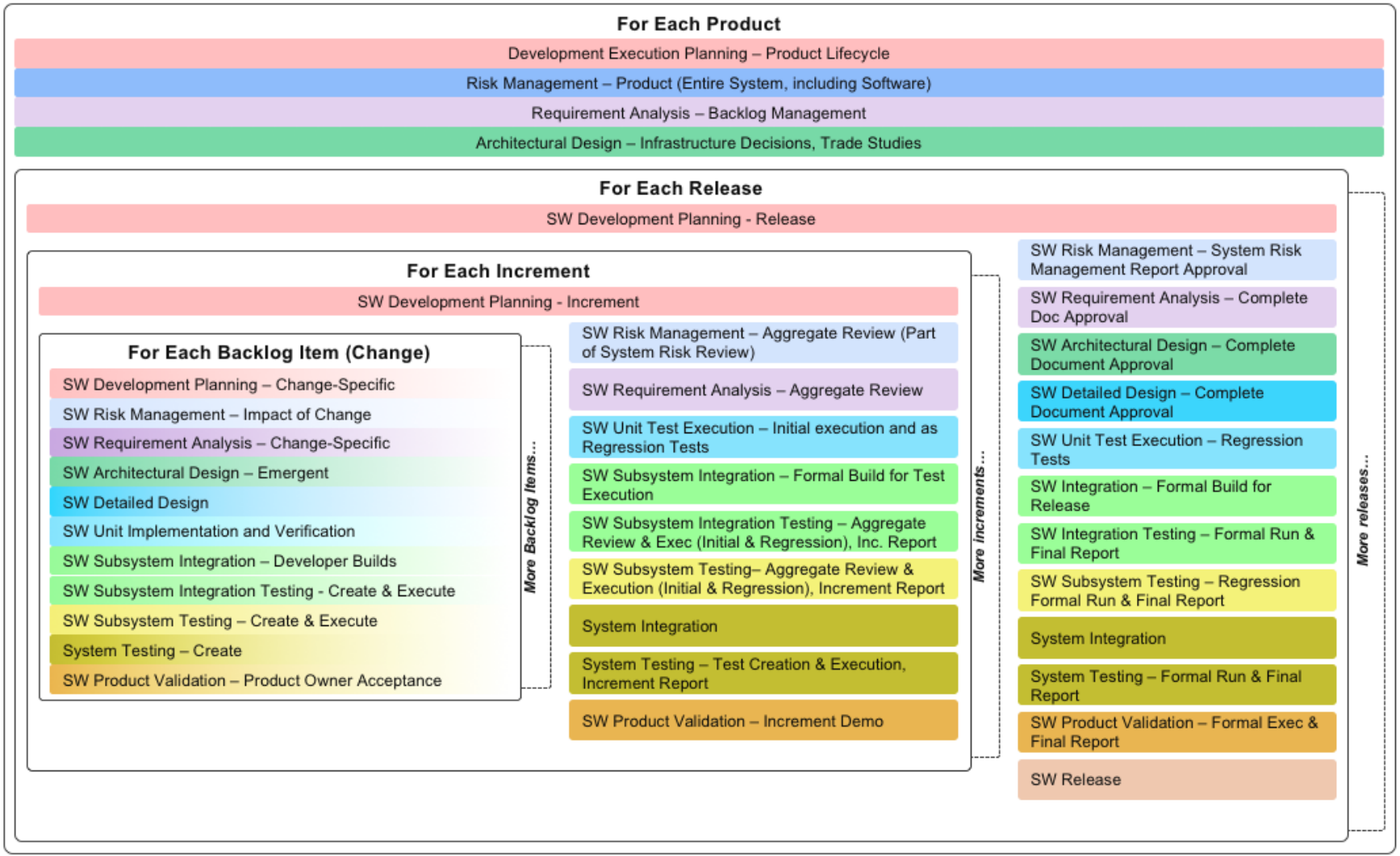




Documentation in the Agile Model

- TIR45:2012 describes how documentation is produced in an Agile model, using a “Sum Of The Parts” concept




For Each Product
Development Execution Planning – Product Lifecycle
Risk Management – Product (Entire System, including Software)
Requirement Analysis – Backlog Management
Architectural Design – Infrastructure Decisions, Trade Studies
For Each Release
SW Development Planning - Release
For Each Increment
SW Development Planning - Increment
For Each Backlog Item (Change)
SW Development Planning – Change-Specific
SW Risk Management – Impact of Change
SW Requirement Analysis – Change-Specific
SW Architectural Design – Emergent
SW Detailed Design
SW Unit Implementation and Verification
SW Subsystem Integration – Developer Builds
SW Subsystem Integration Testing - Create & Execute
SW Subsystem Testing – Create & Execute
System Testing – Create
SW Product Validation – Product Owner Acceptance
More Backlog Items…
SW Risk Management – Aggregate Review (Part of System Risk Review)
SW Requirement Analysis – Aggregate Review
SW Unit Test Execution – Initial execution and as Regression Tests
SW Subsystem Integration – Formal Build for Test Execution
SW Subsystem Integration Testing – Aggregate Review & Exec (Initial & Regression), Inc. Report
SW Subsystem Testing– Aggregate Review & Execution (Initial & Regression), Increment Report
System Integration
System Testing – Test Creation & Execution, Increment Report
SW Product Validation – Increment Demo
More Increments…
SW Risk Management – System Risk Management Report Approval
SW Requirement Analysis – Complete Doc Approval
SW Architectural Design – Complete Document Approval
SW Detailed Design – Complete Document Approval
SW Unit Test Execution – Regression Tests
SW Integration – Formal Build for Release
SW Integration Testing – Formal Run & Final Report
SW Subsystem Testing – Regression Formal Run & Final Report
System Integration
System Testing – Formal Run & Final Report
SW Product Validation – Formal Exec & Final Report
SW Release
More releases…

Recap

Discussion

What are the challenges or barriers to using Agile Requirements?

What challenges to you see in satisfying regulatory expectations?

What changes would be necessary to overcome the challenges?

Contact Information

- Cary Bryczek
- cbryczek@jamasoftware.com
- 202-236-2227

- www.jamasoftware.com

- Connect with me on LinkedIn

- Kelly Weyrauch
- Kelly@AgileQualitySystems.com
- 763-688-0980

- www.AgileQualitySystems.com

- Connect with me on LinkedIn

Appendix (if needed)

3 Challenges at Scale

- Documentation – for some, documentation will still exist. How do we make it “just enough?”
- Prioritization & Alignment – what should we work on next, how do we align multiple teams?
- Traceability – being agile in a more complex/regulated environment still requires complex traceability

Documentation:

more important at scale, how to balance “just enough?”

Documentation

Large systems may still require documentation (e.g. traceability matrices, documented specifications, regulatory compliance).

Traditionally, documentation done “up-front” before beginning design & development.

Lean and Agile principles recommend keeping design options open – and finalizing documentation at the end.

Economic Prioritization

Business Value	Job Size
High (5)	High (1)
Med (3)	Med (3)
Low (1)	Low (5)

Light-weight, relative ranking that considers both business value and LOE

Feature / Req 1

Biz Value: High (5)
LOE: Med (3)
Priority Score 15

Feature / Req 2

Biz Value: High (5)
LOE: Low (5)
Priority Score 25

Weighted Shortest Job First (WSJF)

The jobs (Features, Epics, Reqs) get weighted with the cost of delay so that the most costly jobs get done sooner.

$$\text{WSJF} = \frac{\text{User|Business Value} + \text{Time Criticality} + \text{RR|OE Value}}{\text{Job Size}}$$

Reinertsen, Donald (2008). Principles of Product Development Flow: Second Generation Lean Product Development.

Economic Prioritization:

At Scale, consider economics when prioritizing.
Understand the cost of delay.