

Fundamentals of Agile




WHAT IS AGILE?

- A value-based, iterative approach under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams
- Advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change

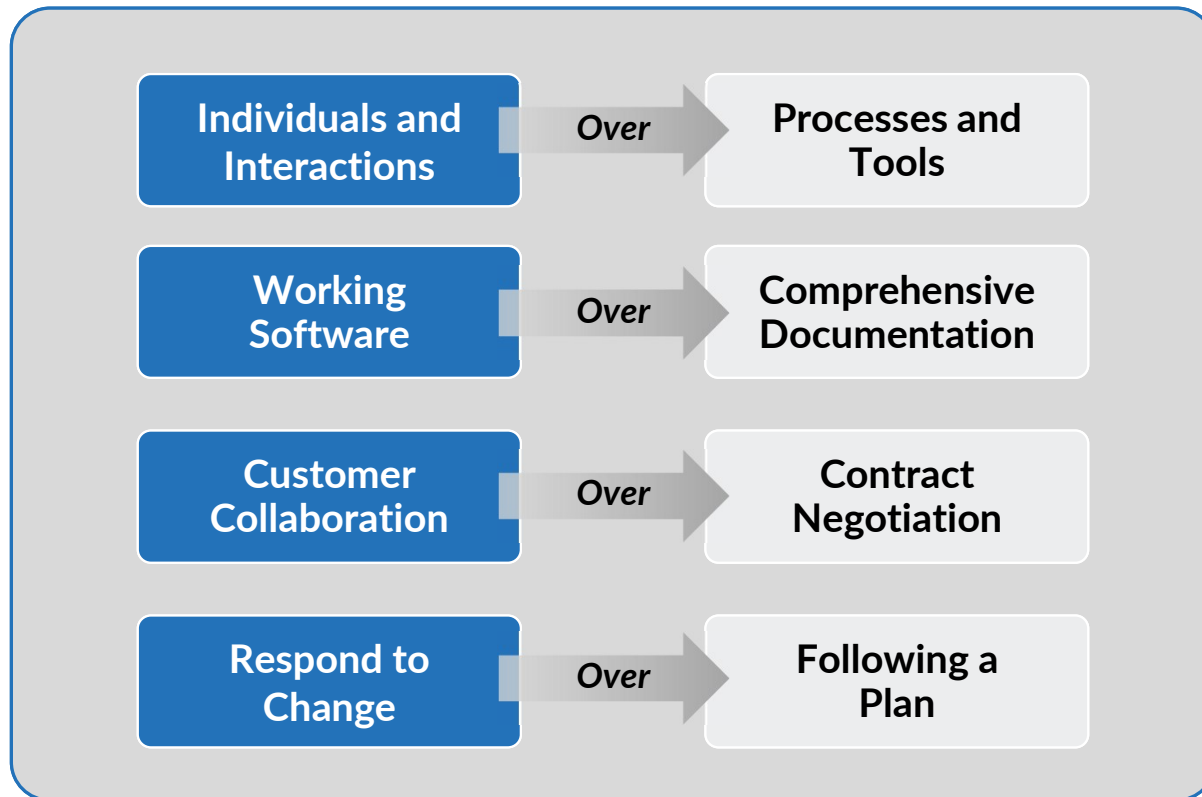




AGILE MANIFESTO

- Seventeen representatives from a variety of software development practices gathered at the Lodge at Snowbird, Utah in 2001 to discuss better ways to develop software
 - The Manifesto for Agile Software Development (Agile Manifesto) was born
 - The 12 principles behind the Agile Manifesto are a set of guiding concepts that support the implementation of Agile projects
 - They are value statements, not a concrete plan or process
 - Applicable to all processes, practices, and techniques
 - The emphasis is on people, collaboration, learning, and value
- 

FOUR VALUE STATEMENTS OF AGILE



SUMMARY OF THE 12 AGILE PRINCIPLES


1 - Satisfy the customer	7 - Working software is the primary measure of progress
2 - Welcome changing requirements	8 - Promote sustainable development
3 - Deliver working software frequently	9 - Continuous attention to technical excellence
4 - Business people and developers must work together daily	10 - Maximize amount of work not done
5 - Build projects around motivated individuals	11 - Self-organizing teams
6 - Face-to-face conversation	12 - Reflect...and tune

DEVELOPMENT METHODOLOGIES

Methodology	Key Concepts
Scrum	Highly adaptable and applied to non-software development projects
Extreme Programming (XP)	Coding, testing, listening, designing, paired programming
Dynamic System Development Method (DSDM)	Prioritizes scope into must, should, could, and won't have versus backlogs
Feature Driven Development (FDD)	Build feature list, plan by feature, design by feature, build by feature, inspection
Adaptive Software Development (ASD)	RAD, Adaptive
Lean Software Development (LSD)	Eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible
Behavior Driven Development (BDD)	User stories are converted into sets of tests or behaviors, XP/Scrum



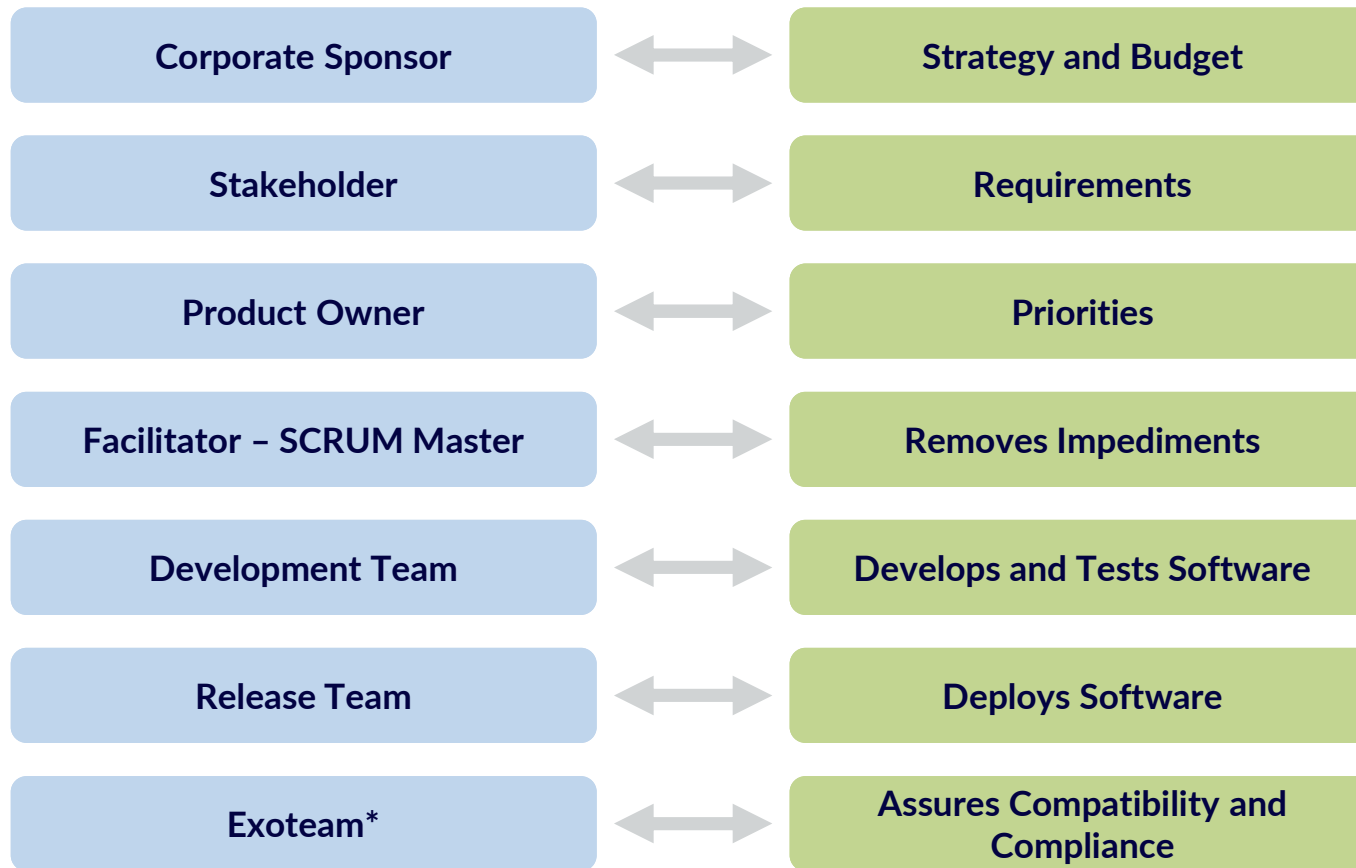
DEFINITION OF DONE

- The criteria for accepting work as completed
 - Although the exact criteria for what constitutes “Done” varies to meet the specific needs of different organizations and initiatives, there are generally three levels of “Done” (also known as Done-Done-Done)
 - Done: Developed, runs on developer’s box
 - Done: Verified by running unit tests, code review, etc. (Agile Team)
 - Done: Validated as being of deliverable quality with functional tests, reviews, etc. (Client/Customer Acceptance)
- 

A photograph of a diverse group of business professionals in a meeting. An older man in a suit is looking at a laptop screen, while a woman with glasses and a man in a blue shirt look on attentively. The scene is set in a modern office environment with a wooden table, water glasses, and a coffee cup.

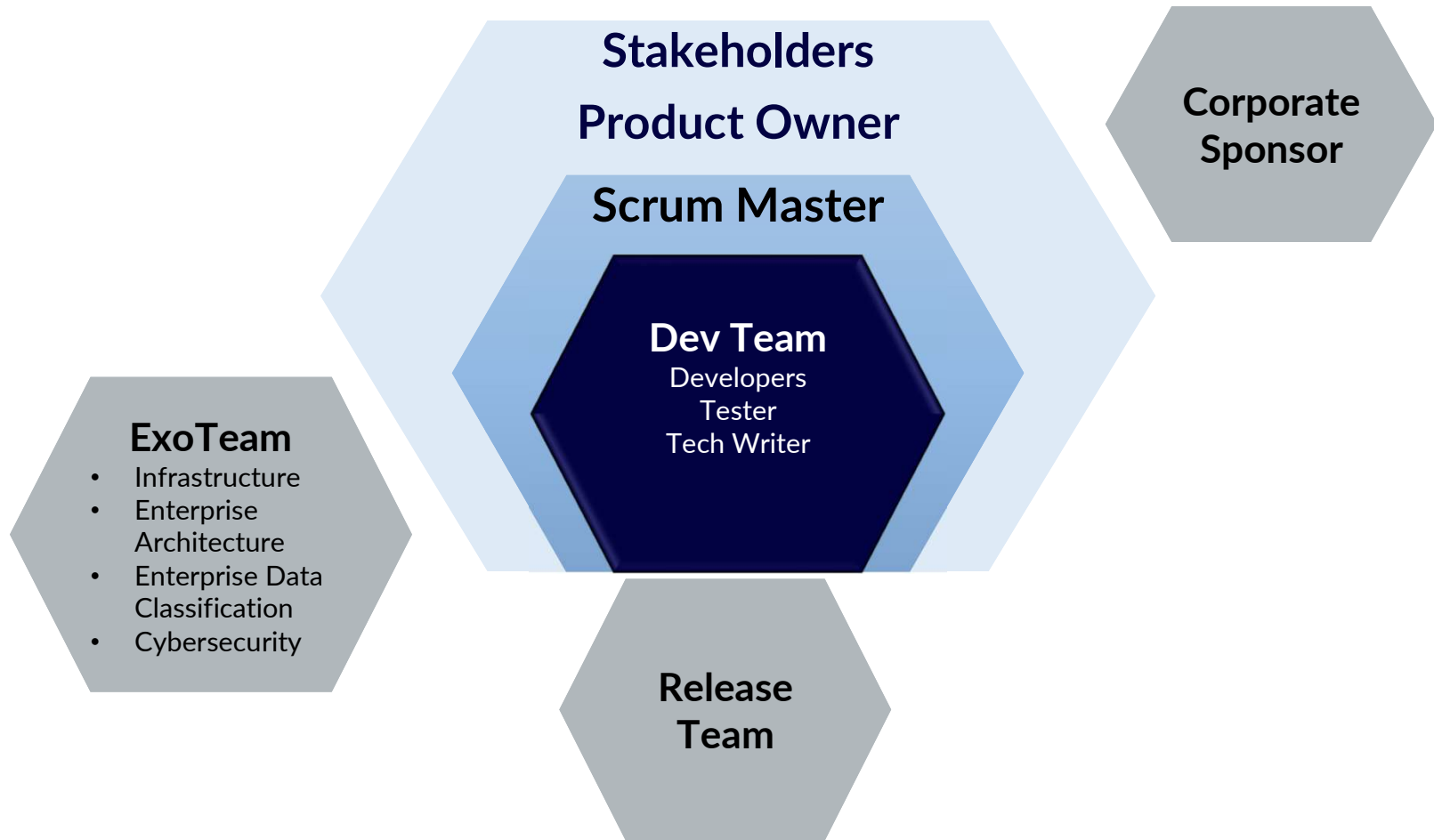
Agile Teams and Roles

TEAM ROLES




*Staff outside of the core development team - infrastructure engineers, security team, etc.

AGILE PARTICIPANT RELATIONSHIPS





TEAM CHARACTERISTICS

- Self-organizing
 - Cross-functional, with multidisciplinary members with numerous skillsets from across the organization
 - Structured to allow for its own autonomy so that it need not rely on outside teams to complete its work
 - Empowered to collectively own the whole product and decide how work will be accomplished
 - Collaborate together to deliver a given product
 - Decide how to communicate
 - Decide team roles
- 

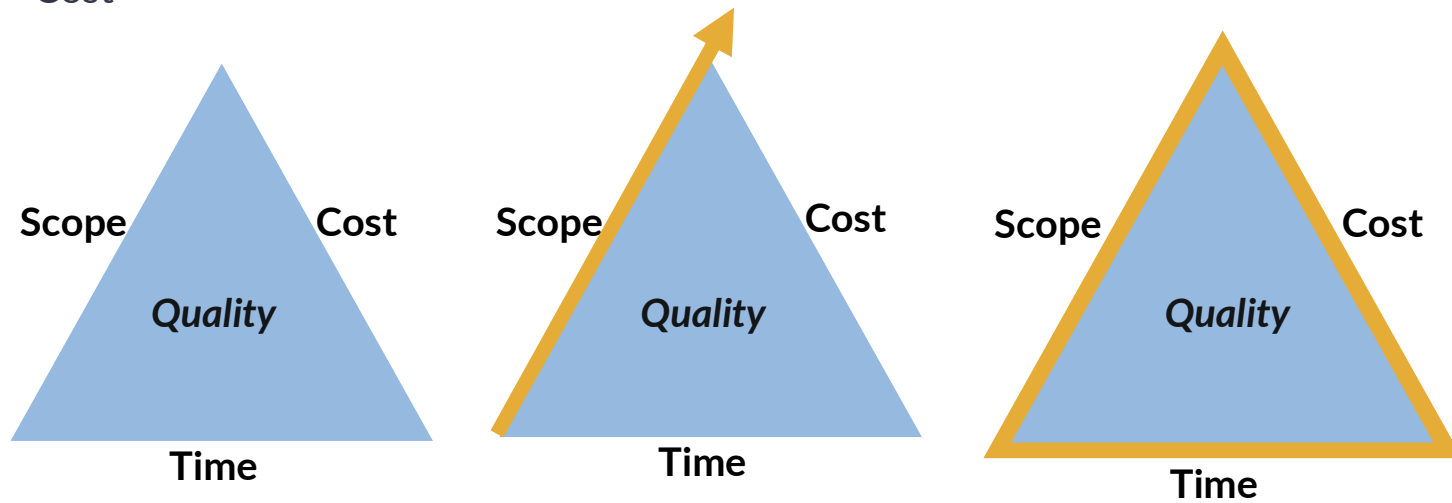


Lesson 3

Agile Scope

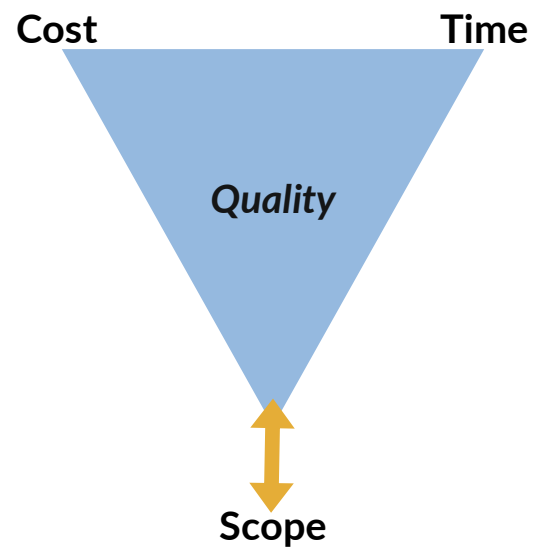
IRON TRIANGLE

- What “Binds” a traditional project?
 - Quality
 - The Triple Constraint
 - ✓ Scope
 - ✓ Time
 - ✓ Cost




RUBBER TRIANGLE

- What “Binds” an Agile project?
 - Quality
 - The Triple Constraint
 - ✓ Scope
 - ✓ Time
 - ✓ Cost






USER STORIES

- Short, simple descriptions of a feature
 - Told from the perspective of a user or customer of the system
 - Follow a template
 - As a < type of user >, I want < some goal > so < some reason >
 - Written on index cards or sticky notes
 - Arranged on walls or tables to facilitate planning and discussion
 - Shift the focus from writing about features to discussing them
- 

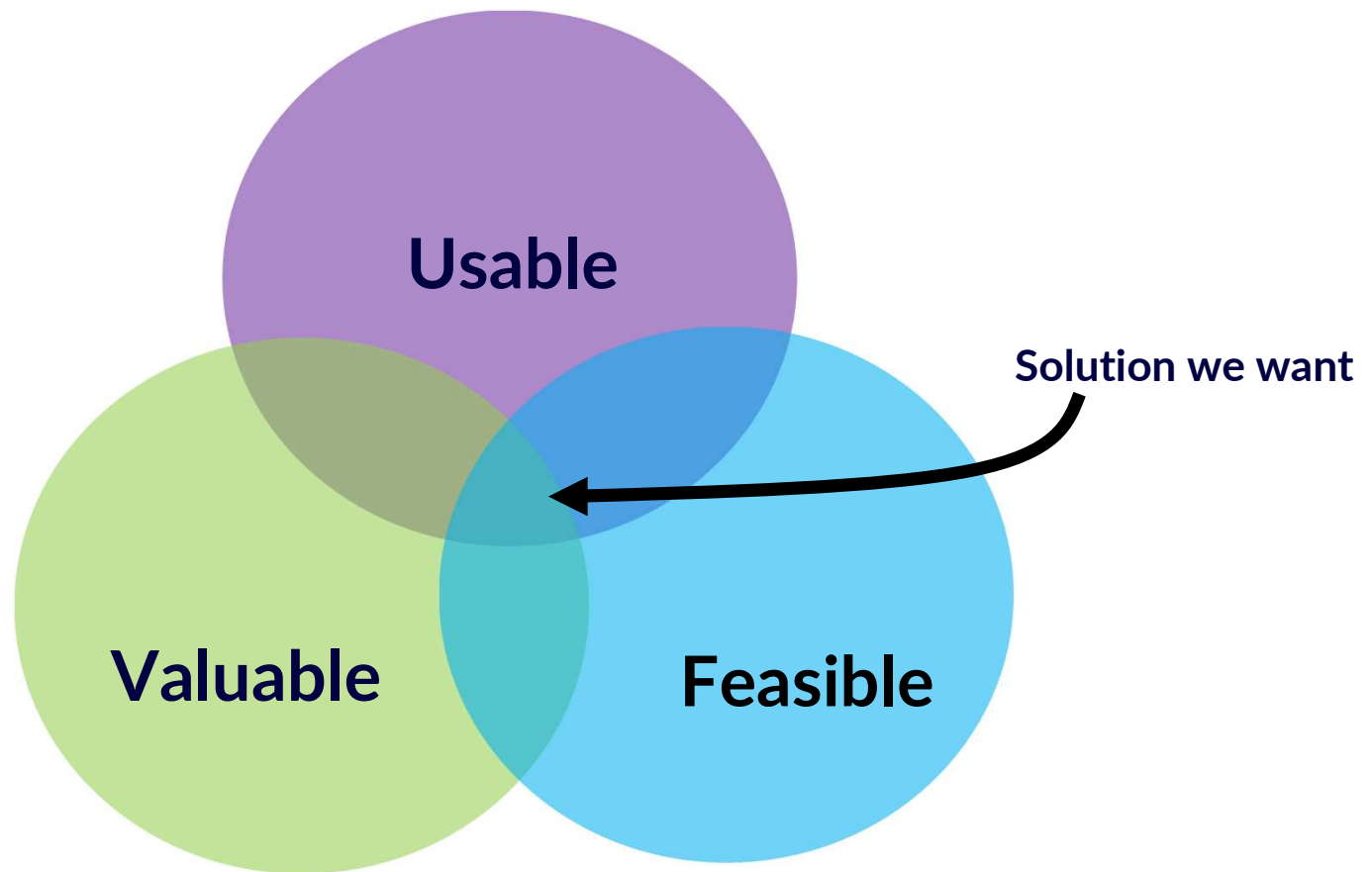


STORY TYPES

- Feature (User Stories)
 - Functional capability (accomplishes user needs)
 - Chores
 - Non-functional work (documentation, meetings, reports)
 - Refactor
 - Technical clean-up (code optimization, restructuring, integration)
 - Defects (Bugs, Undocumented Features)
 - Something found after acceptance (passed the Done, Done, Done)
- 


USER STORIES DEFINE THE SOLUTION

- The product owner identifies what is valuable, usable, and feasible as the solution we want





BACKLOG

- A list of features, user stories, and tasks to be addressed by the team, program or portfolio and is ordered from the highest priority to the lowest priority
 - Includes both functional and non-functional work, including technical team-generated user stories, features, or epics
- 

AGILE ESTIMATION

- What is an estimate?
 - A rough calculation of the value, number, quantity, or extent of something
 - A point-based, valuation system for estimating the level of effort it takes to deliver a requirement, or user story
 - The valuation scale for estimates is usually reflected in terms of size, but most often as story points

Point Based System



STORY POINT CONCEPT

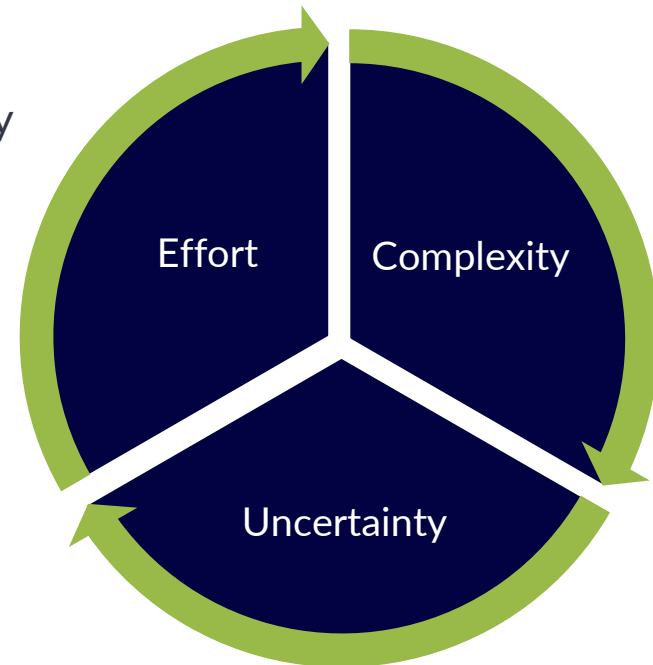
- Story Points – subjective estimate of effort to get a story done
 - Done = developed, tested, and demonstrated
 - In simple terms, it's a number telling how difficult the story is to complete
- Story point estimation is done using relative sizing by comparing one story with a sample set of previously sized stories
- Relative sizing across stories tends to be much more accurate over a larger sample in lieu of trying to estimate each individual story

Example:

It is easier to understand Cleveland to Chicago is three times the distance of Philadelphia to New York City

STORY POINT CHARACTERISTICS

- Story points
 - Rate the relative effort of work, complexity, and uncertainty in a Fibonacci Sequence
format: 0,1, 2, 3, 5, 8, 13, 21, 34, ∞
 - Should be broken into smaller stories when there are large stories
 - Are based upon effort, complexity, and uncertainty



PRIORITIZE STORIES - BACKLOG

Biggest bang for buck

Most important



Least important

Master Story List*

1 pt	Add user
2 pts	Print itinerary
13 pts	Cancel Trip
3 pts	Book permit
1 pt	Update permit
.....
5pts	Create device
13pts	Add swap trade
1pt	Add option

Most complex story

Nice to haves

Ones we may never get to

MVP – Minimum Viable Product

- Stories that must be in 1st release



Scrum Methodology

SCRUM METHODOLOGY

- Lightweight Agile project management framework with broad applicability for managing iterative projects of all types
 - Most popular approach in the U.S. – 80+%
 - Teams work tightly together to succeed and produce a working product
 - Scrum is popular because of its
 - ✓ Simplicity
 - ✓ Proven productivity
 - ✓ Can act as a wrapper other engineering practices promoted by other Agile methodologies
 - ✓ Proven scalability to multiple teams across very large organizations


SCRUM TEAM

- Consists of one Scrum Master, one Product Owner, and Developers
 - Within a Scrum Team, there are no sub-teams or hierarchies
 - It is a cohesive unit of professionals focused on one objective at a time, the Product Goal






SCRUM MASTER

- Accountable for establishing Scrum as defined in the Scrum Guide
 - Accountable for the Scrum Team's effectiveness
 - Facilitates meetings, removes impediments, and maintains the team's focus
 - Ensures that all Scrum events take place and are positive, productive, and kept within the timebox
- 



PRODUCT OWNER

- Voice of the Stakeholders
 - Maximizes the value of the product resulting from the work of the Scrum Team
 - Accountable for effective Product Backlog management
 - Developing and explicitly communicating the Product Goal
 - Creating and clearly communicating Product Backlog items
 - Ordering Product Backlog items
 - Ensuring that the Product Backlog is transparent, visible and understood
- 

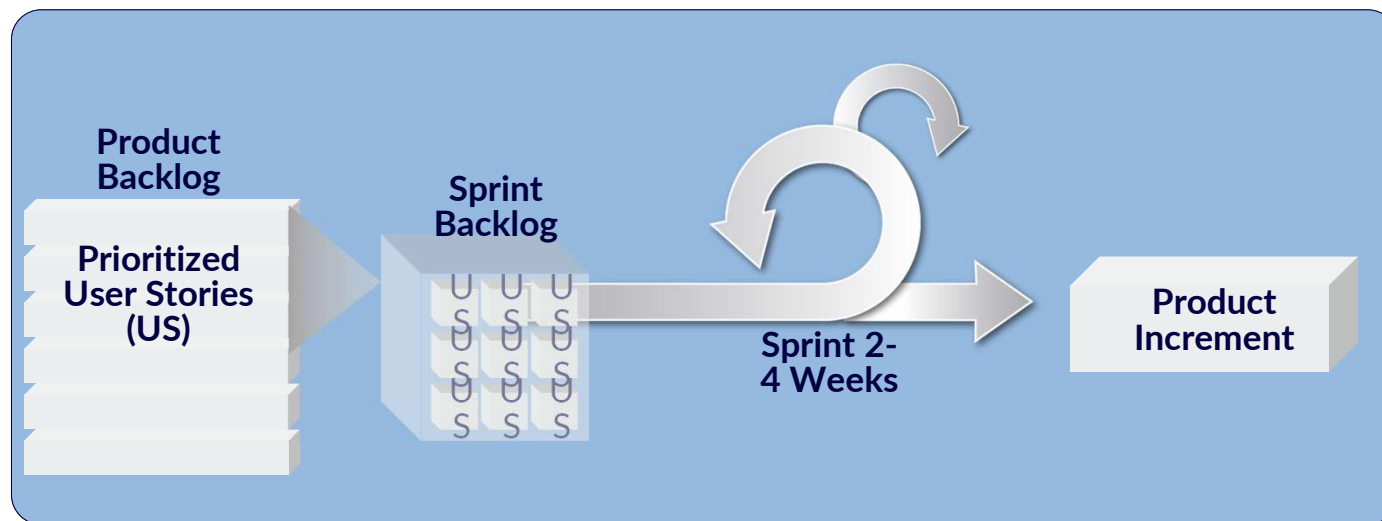
DEVELOPMENT TEAM

- Committed to creating any aspect of a usable Increment each Sprint
- Accountable for
 - Creating a plan for the Sprint, the Sprint Backlog
 - Instilling quality by adhering to a Definition of Done
 - Adapting their plan each day toward the Sprint Goal
 - Holding each other accountable as professionals



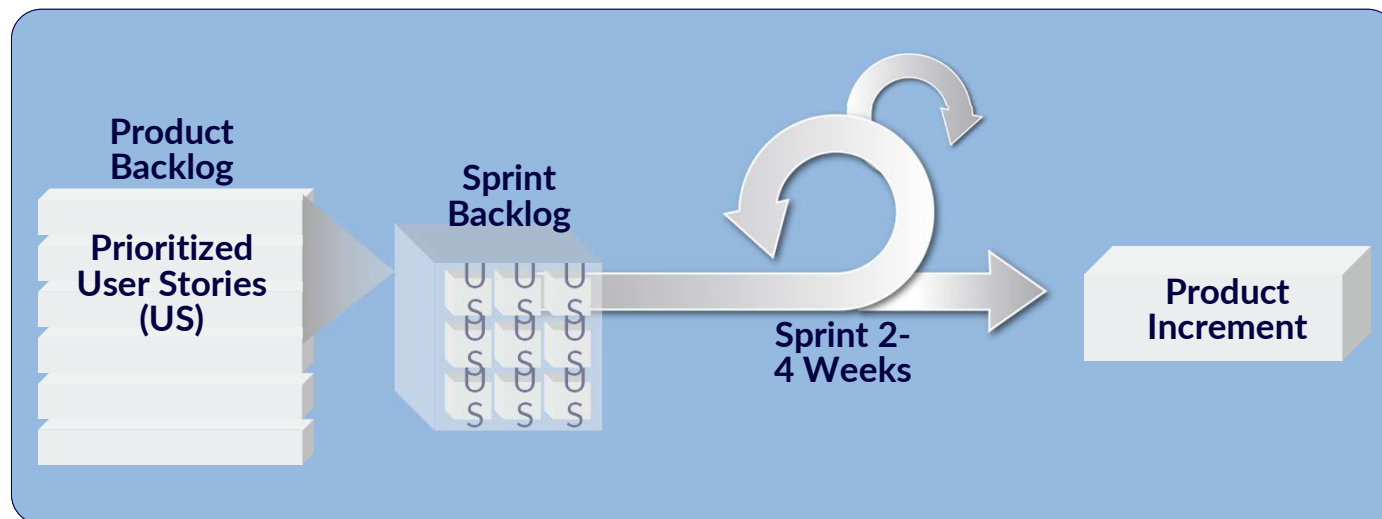
PRODUCT BACKLOG

- The Product Backlog is an emergent, ordered list of what is needed to improve the product
- Defines “what” will fulfill the Product Goal
- Prioritized



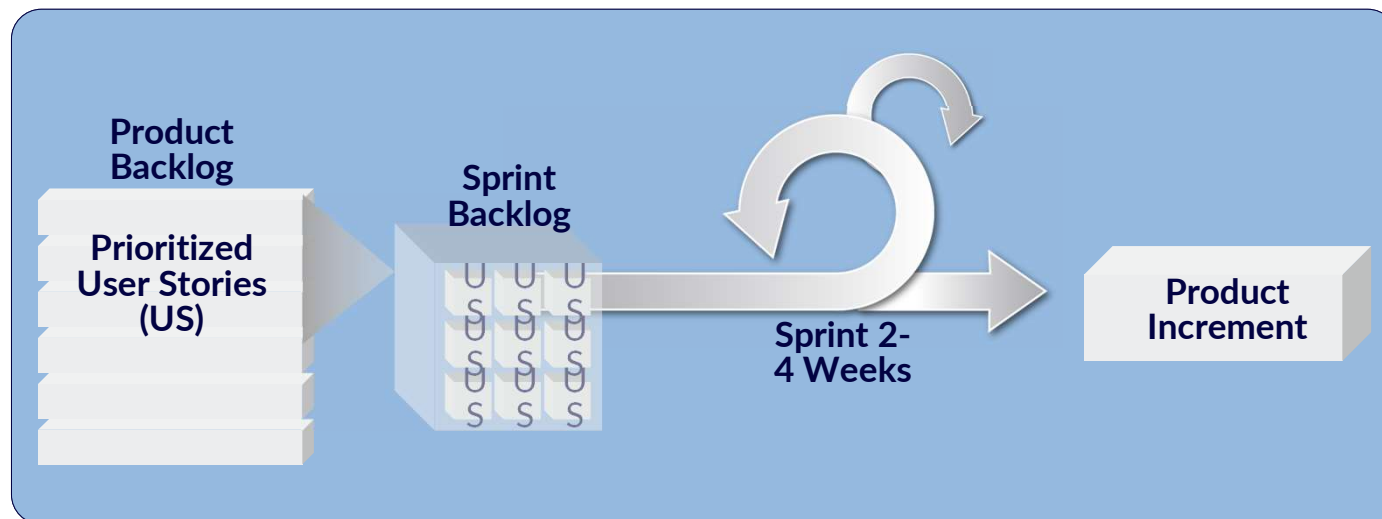
SPRINT BACKLOG

- Sprint Goal (why)
- The set of Product Backlog items selected for the Sprint (what)
- An actionable plan for delivering the Increment (how)




PRODUCT INCREMENT (PI)

- Concrete stepping stone toward the Product Goal
- Each PI is additive to all prior PIs and thoroughly verified
- Fully tested working software, “potentially shippable”






SCRUM EVENTS

- The Sprint is a container for all other events
 - Each event in Scrum is a formal opportunity to inspect and adapt Scrum artifacts
 - These events are specifically designed to enable the transparency required
 - Optimally, all events are held at the same time and place to reduce complexity
- 



THE SPRINT

- Fixed length events of one month or less to create consistency
 - Collection of stories worked over the sprint time period
 - During the Sprint
 - Scope may be clarified and renegotiated with the Product Owner as more is learned
 - Quality does not decrease
 - No changes are made to agreed upon Stories
 - The Product Backlog is refined as needed
- 

SAMPLE SCRUM ACTIVITIES

Stakeholder, end-users, and Scrum team provide input to product owner

Scrum team selects user stories and commits to delivery by sprint's end

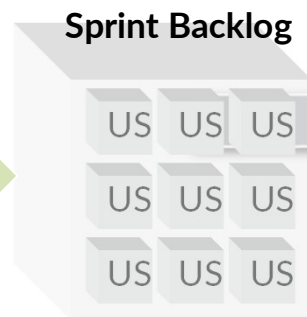
Sprint Planning

Team completes user stories and holds daily Scrum to re-plan remaining work

Team demonstrates the product increment and reflects on processes

Sprint Review


Sprint Retrospective



Scrum Master documents unfinished user stories and updates product backlog



SPRINT PLANNING

- Initiates the Sprint by laying out the work to be performed for the Sprint
 - The Product Owner ensures that attendees are prepared to discuss the most important Product Backlog items and how they map to the Product Goal
 - Create the Sprint Backlog
 - The resulting plan is created by the collaborative work of the entire Scrum Team
 - The Scrum Team may also invite other people to attend Sprint Planning to provide advice
 - Timeboxed to a maximum of two hours for each week of Sprint
- 


DAILY SCRUM

- Allows the Scrum Team to highlight what they are working on and any impediments to progress
- Provides for rapid communication and decision-making
- Timeboxed to no more than 15 minutes

What did you do yesterday?
What are you doing today?
Do you have any blockers / impediments?



SPRINT REVIEW

- The Scrum Team, Product Owner, and stakeholders review what was accomplished in the Sprint and what has changed in their environment
 - Completed work is either Accepted or Rejected
 - Based on this information, attendees collaborate on what to do next.
 - The Product Backlog may also be adjusted to meet new opportunities
 - Timeboxed to a maximum of one hour for each week of Sprint
- 


SPRINT RETROSPECTIVE

- Scrum Team reviews their previous sprint's activities and processes in an effort to continuously improve
- Timeboxed to a maximum of 45 min. for each week of Sprint
- The retrospective concludes the Sprint

What went well during the sprint cycle?
What went wrong during the sprint cycle?
What could we do differently to improve?




BACKLOG GROOMING

- A Product owner and Agile team periodically review the backlog to ensure appropriate stories are at the top of the backlog and ready for delivery
 - Grooming activities
 - Remove user stories that no longer appear relevant
 - Create new user stories in response to newly discovered needs
 - Reassess the relative priority of stories
 - Assign estimates to stories and modify them when necessary
 - Split user stories that are high priority, but too coarse to fit into an upcoming iteration
- 

Agile Metrics




AGILE METRICS

- Favors empirical and value-based measurement instead of predictive measurements
 - Measures what the team delivers, not what the team predicts it will deliver
 - Based on working products of demonstrable value to customers
- 

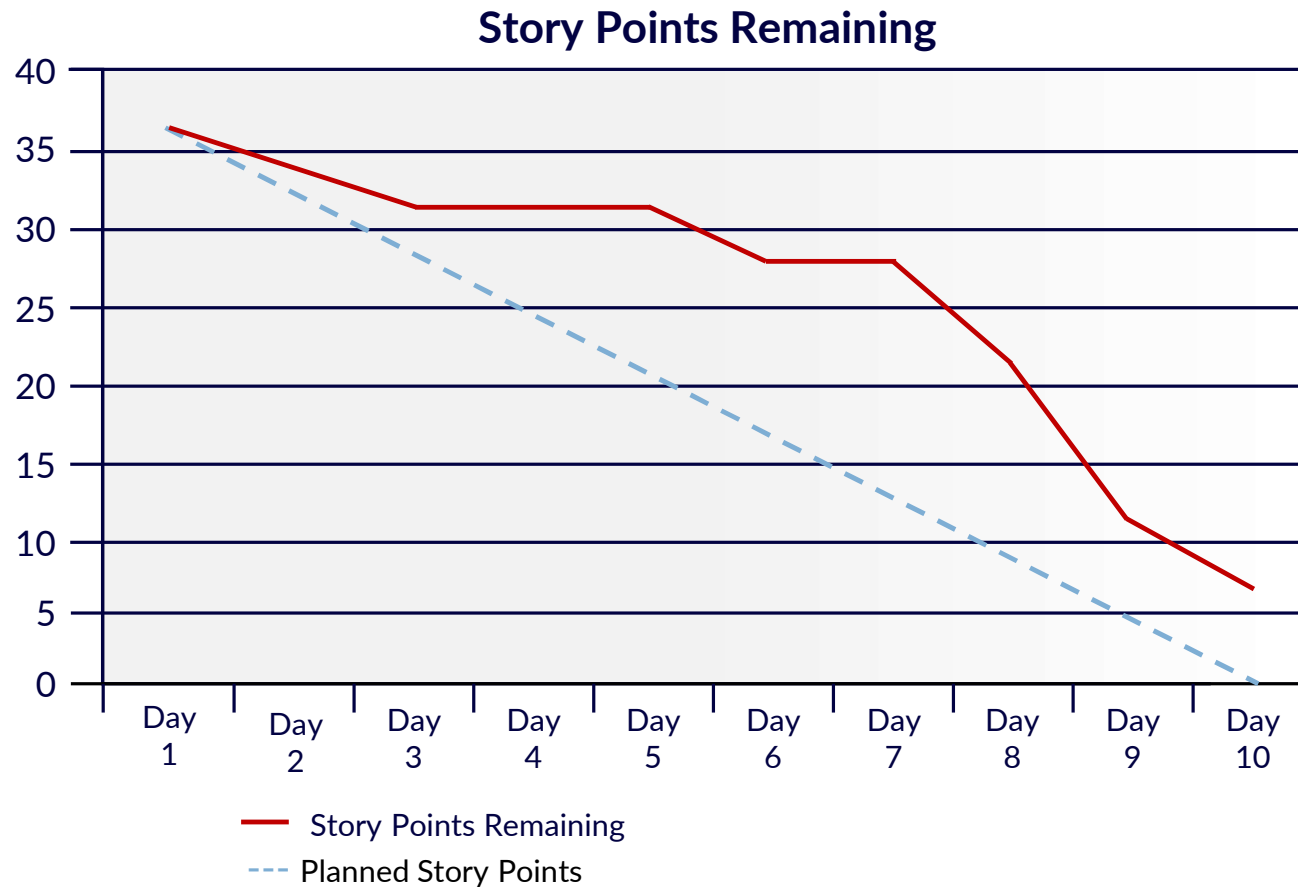


VELOCITY

- Measures how much work a team can complete in an iteration
 - Measured in stories or story points
 - Should only be computed after at least 3 Sprints
 - Used to measure how long it will take a particular team to deliver future outcomes by extrapolating on the basis of its prior performance
- 

BURNDOWN CHART

- A graphical representation of the work remaining versus the time left in a timebox



EFFICIENCY

- A little secret...
 - When addressing a prioritized list of stories, not all stories may be accomplished
 - Since these stories were relatively less important, they have little impact on the functionality
 - Users may receive 80% or 90% of what they need within schedule and budget, resulting in huge efficiency

What is better?

- 100% of requirements that work 80% of the time
- 80% of requirements that work 100% of the time*

* - and they are the most important requirements!

Questions?

J. FLETCHER HEARNS

FHearns@edwps.com

443.561.1340



LET'S TALK

To learn about how we can help your organization, please contact us.



10980 Grantchester Way, Suite 300
Columbia, MD 21044

800.556.2506 | www.EdwPS.com

This presentation is proprietary to Edwards Performance Solutions