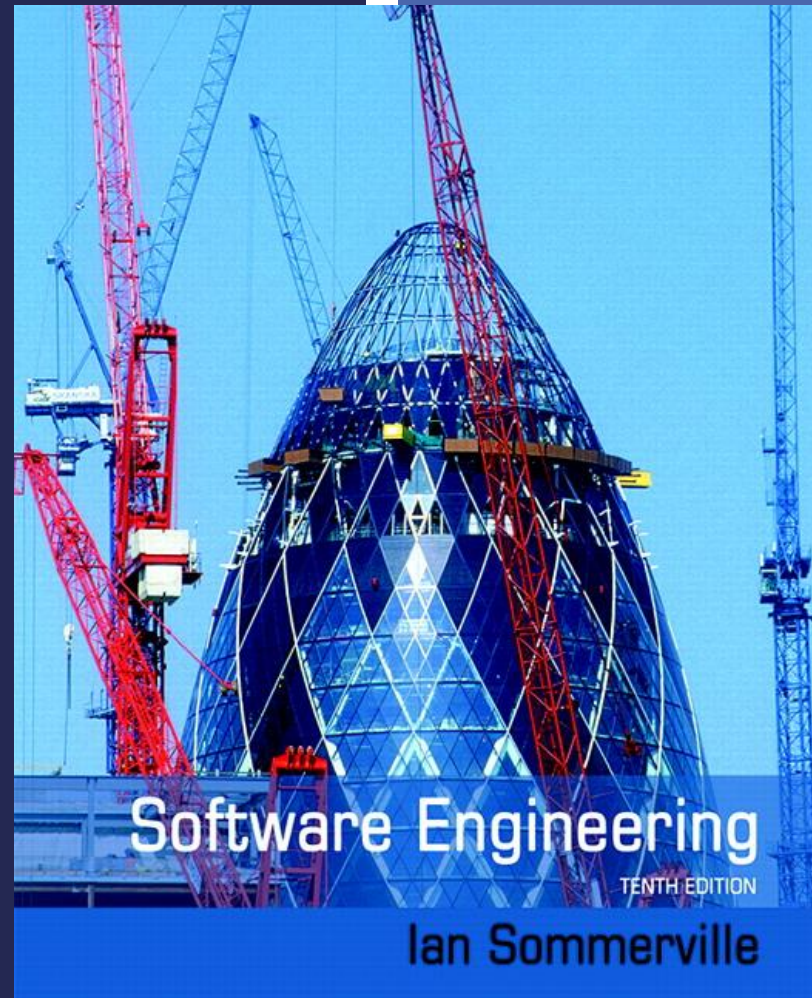


Software Engineering I

Chapter 3

Agile SW Development

plus Scrum in-depth



Plan-driven and agile development

■ Plan-driven development

- A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
- Not necessarily waterfall model – plan-driven, incremental development is possible
- Iteration occurs within activities.

■ Agile development

- Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.
- A cadence of multiple deliverables is better able to help a project keep progress and provide better estimates rather than having everything be "80% done"

Why the break from Waterfall?

On February 2001, 17 software developers met at the Snowbird resort in Utah to discuss lightweight development methods.

They published the *Manifesto for Agile Software Development*

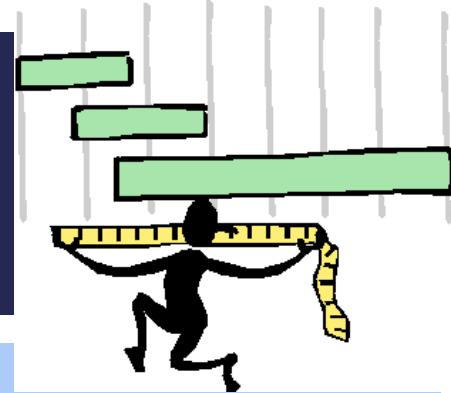
Manifesto for Agile Software Development

- 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.

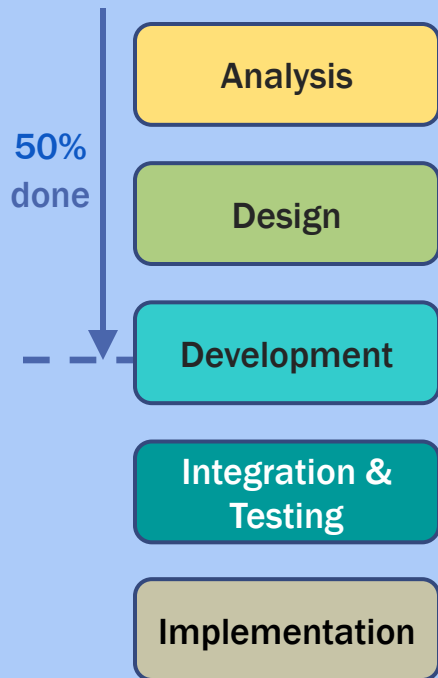
Twelve Principles of the Agile Manifesto

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

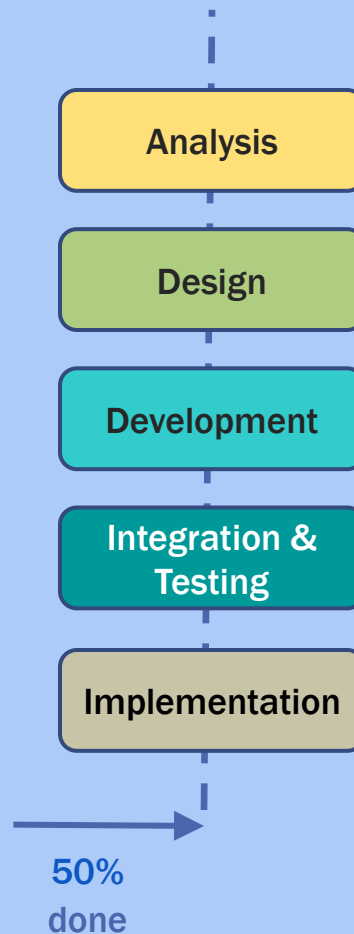
The primary measure of progress



TRADITIONAL WATERFALL



AGILE



Agile vs Scrum

"The Agile movement is not anti-methodology, in fact many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment."

Jim Highsmith, 1 of the 17 original signatories of the Agile Manifesto

"Scrum is an Agile framework for completing complex projects. Scrum originally was formalized for software development projects, but it works well for any complex, innovative scope of work"

ScrumAlliance, founded by Ken Schwaber, another original signatory of the Agile Manifesto

"Scrum is a management and control process that cuts through complexity to focus on building software that meets business needs... Scrum itself is a simple framework for effective team collaboration on complex software projects."

scrum.org, founded by Ken Schwaber,

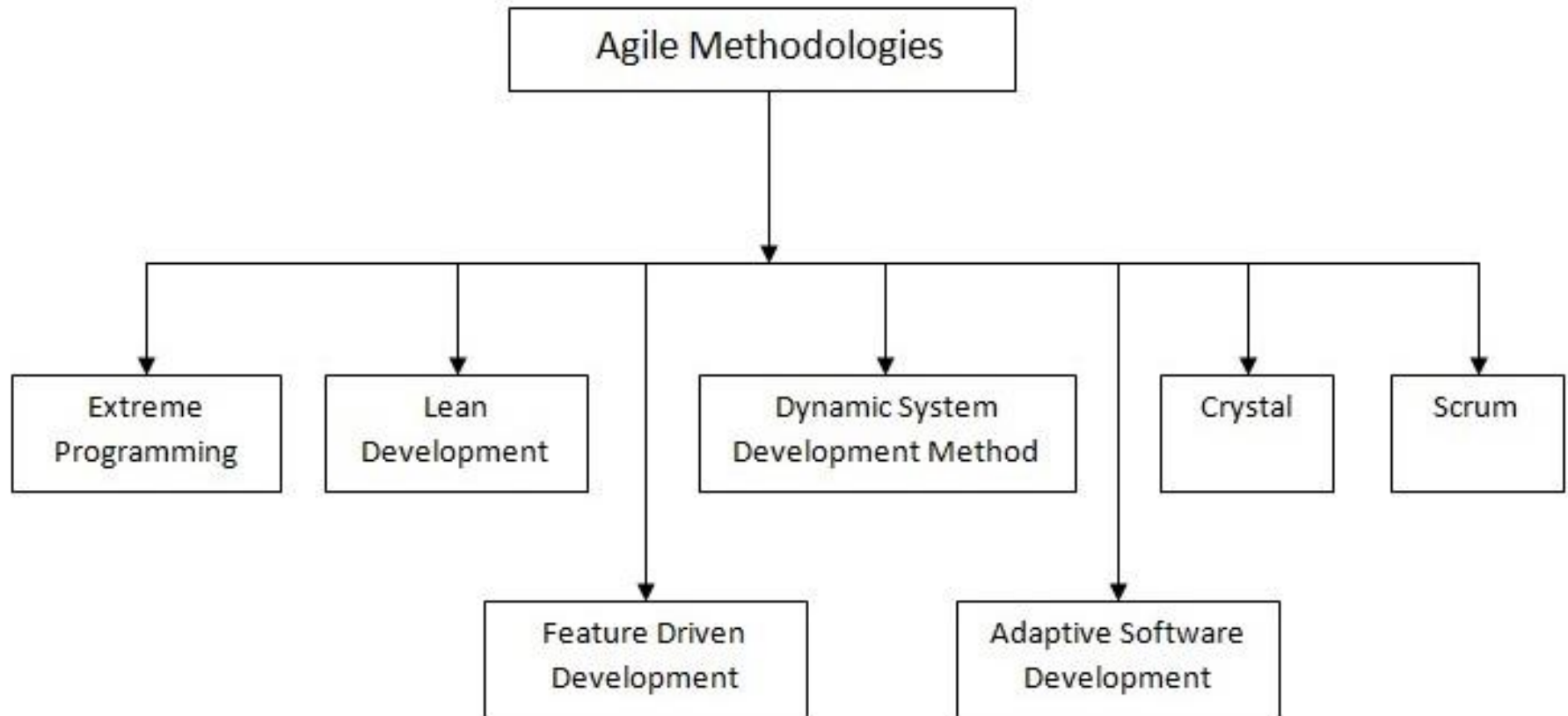
Agile Programming

The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.



Scrum

The most prevalent of the Agile Methodologies



Definition of scrum

- Scrum (n): A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.
- Scrum is:
 - Lightweight
 - Simple to understand
 - Difficult to master
- Scrum is a process framework that has been used to manage complex product development since the early 1990s.
- Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques.
- The Scrum framework consists of Scrum Teams and their associated **roles, events, artifacts, and rules**. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.
- The rules of Scrum bind together the events, roles, and artifacts, governing the relationships and interaction between them. The rules of Scrum are described throughout this document.
- Specific tactics for using the Scrum framework vary and are described elsewhere.

Scrum Theory: Process Control Models

PREDICTIVE

- Work and outcomes are understood before execution.
- Given a well-defined set of inputs, the same outputs are generated every time.
- Follow the pre-determined steps to get known results.

Customer knows what he wants.
Engineers know how to build it.
Nothing changes along the way.

EMPIRICAL

- Frequent inspection and adaptation occurs as work proceeds.
- Processes are accepted as imperfectly designed.
- Outputs are often unpredictable and unrepeatable.

Customer discovers what he wants.
Engineers discover how to build it.
Things change along the way.

The Three Legs of Empiricism

Empiricism asserts that knowledge comes from experience and making decisions based on what is known

■ TRANSPARENCY

- Significant aspects of the process must be visible to those responsible for the outcome.
- Transparency requires those aspects be defined by a common standard so observers share a common understanding of what is being seen.
- For example:
 - A common language referring to the process must be shared by all participants; and,
 - Those performing the work and those accepting the work product must share a common definition of “Done”.
- Transparency allows all facets of any Scrum process to be observed by anyone. This promotes an easy and transparent flow of information throughout the organization and creates an open work culture. In Scrum, transparency is depicted through:
 - Artifacts
 - Project Vision Statement
 - Prioritized Product Backlog
 - Meetings
 - Sprint Review Meetings
 - Release Planning Schedule
 - Burndown Chart
 - Scrumboard
 - Inspection

The Three Legs of Empiricism

Empiricism asserts that knowledge comes from experience and making decisions based on what is known

■ INSPECTION

- Scrum users must frequently inspect Scrum artifacts and progress toward a Sprint Goal to detect undesirable variances.
- Their inspection should not be so frequent that inspection gets in the way of the work. Inspections are most beneficial when diligently performed by skilled inspectors at the point of work.
- Inspection can be accomplished through:
 - Use of a common Scrumboard and other information radiators
 - Collection of feedback from the customer and other stakeholders during Sprint reviews, when creating a Prioritized Product Backlog, and during Release Planning processes
 - Inspection and approval of the Deliverables by the Product Owner and the customer in the Demonstrate and Validate Sprint process.

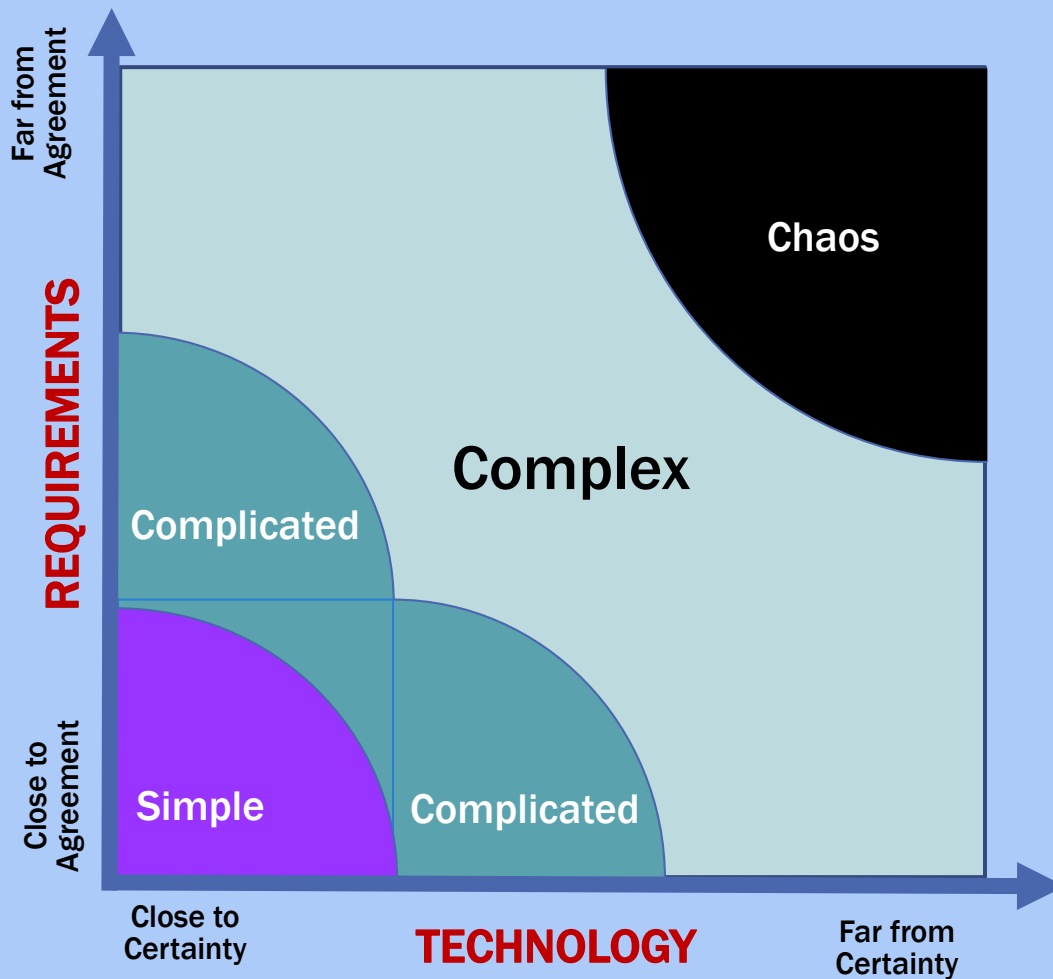
The Three Legs of Empiricism

Empiricism asserts that knowledge comes from experience and making decisions based on what is known

■ ADAPTATION

- Adaptation happens as the Scrum Team and Stakeholders learn through transparency and inspection and then adapt by making improvements in the work they are doing., especially through Sprint Retrospectives and constant risk identification.
- If an inspector determines that one or more aspects of a process deviate outside acceptable limits, and that the resulting product will be unacceptable, the process or the material being processed must be adjusted. An adjustment must be made as soon as possible to minimize further deviation.
- Scrum prescribes four formal events for inspection and adaptation, as described in the Scrum Events section of this document:
 - Sprint Planning
 - Daily Scrum
 - Sprint Review
 - Sprint Retrospective

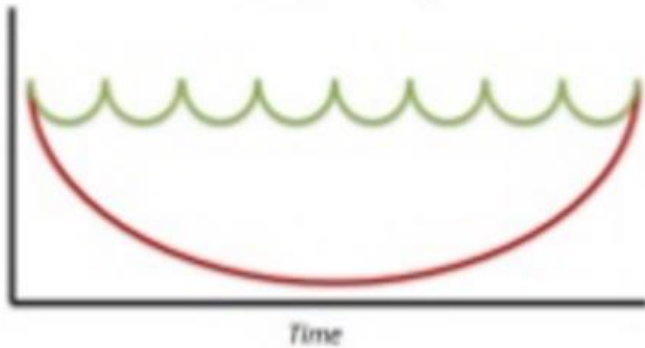
The Complexity of Software Development



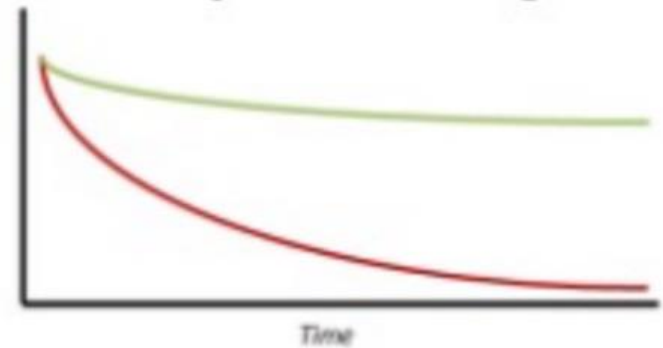
- **Simple**
 - Everything is known
- **Complicated**
 - More is known than unknown
- **Complex**
 - More is unknown than known
 - Scrum thrives here
- **Chaotic**
 - Very little is known

Characteristics of Waterfall vs Scrum

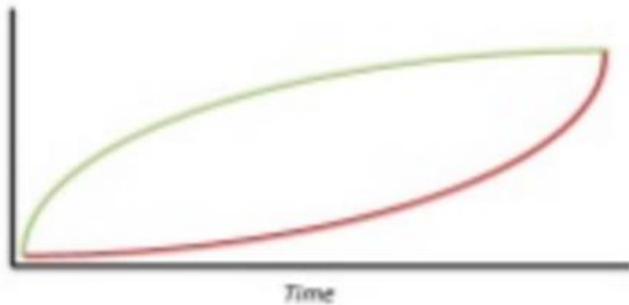
Visibility



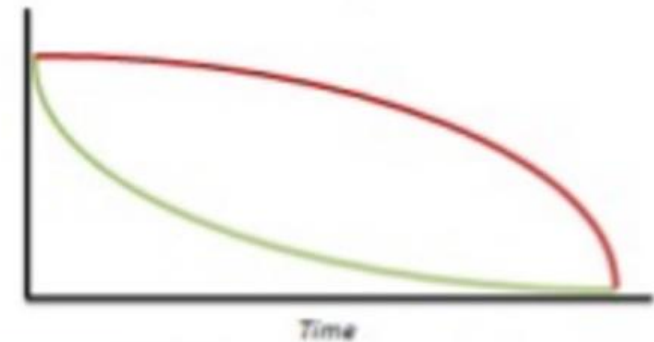
Ability to Change



Business Value



Risk



Waterfall

Scrum

Scrum in a nutshell

- A Team commits to delivering working software in 30 days or less.
- A time is scheduled to show that software
- The Team creates the software
- The Team offers their work for inspection and adapts the plan for the next cycle

Scrum Roles, Artifacts and Events

Prescribed by Scrum

ROLES

- **PRODUCT OWNER**
 - **DEVELOPMENT TEAM**
 - **SCRUM MASTER**
- 
- ```
graph BT; DM[DEVELOPMENT TEAM] -- serves --> PO[PRODUCT OWNER]; SM[SCRUM MASTER] -- serves --> DM;
```

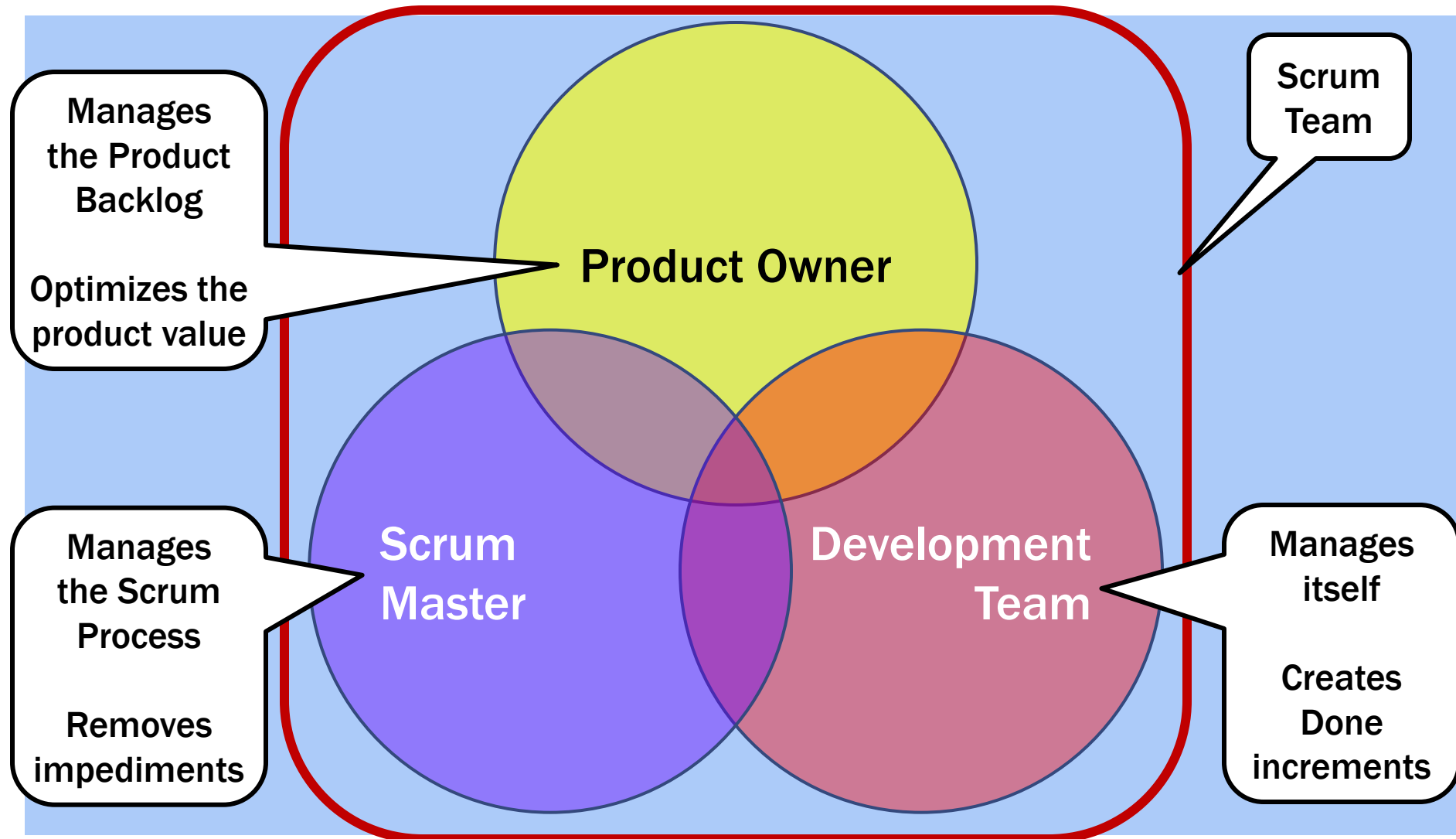
## ARTIFACTS

- **PRODUCT BACKLOG**
- **SPRINT BACKLOG**
- **INCREMENT**

## TIME BOXED EVENTS

- **SPRINT**
- **SPRINT PLANNING MEETING**
- **DAILY SCRUM**
- **SPRINT REVIEW**
- **SPRINT RETROSPECTIVE**

# Roles (high level)



# Roles

## ■ PRODUCT OWNER

- Optimizes the value of the Product
- Manages the Product Backlog
- Chooses what and when to release
- Represents stakeholders and customers to the Development Team
- Ideally from the business with profit and loss accountability for Product

*Does not do all this in a vacuum,  
works with full Scrum Team*

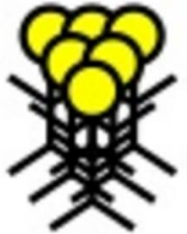


Product  
Owner

## ■ DEVELOPMENT TEAM

- Creates the product Increment
- Operates in a series of Sprints
- Organizes itself and its work
- Collaborates with Product Owner to optimize value
- Must have all the skills it needs to deliver a done Increment – breadth and depth; ideally more than one team member has the competency

*Scrum does not name sub-roles, e.g.  
Testers, Coders, Business Analysts...*



Dev. Team

## ■ SCRUM MASTER

- Enacts Scrum values, practices, and rules throughout the organization
- Ensures the Scrum Team is functional and productive – "healthy"
- Provides guidance and support for the Scrum Team – has a helpful personality



Scrum  
Master

# For our section



Product  
Owner

- The Product Owner will be the direct interface to MSE.



Dev. Team

- One member of the development team will be in charge of the GitHub repository



Scrum  
Master

- The Scrum Master will be the direct interface to me.

*Plus every team member will run a Sprint  
and produce a formal deliverable.*

# Artifacts

## ■ PRODUCT BACKLOG

- Holds the requirements for the Product
  - A minimal but sufficient inventory of work
  - An ordered list of desirements
  - Potential features of the product
  - The single source of truth for what is planned in the product
- Managed by the Product Owner
- Consists of Product Backlog Items (PBIs)
- Public and available

## ■ INCREMENT

- Working addition to the product
- Should be releasable
- The sum of all the PBIs completed during the Sprint
- Is usable and it works
- Must be DONE per Scrum Team standards with no work remaining

## ■ SPRINT BACKLOG

- Holds all work for the Sprint Goal
- The selected Product Backlog Items ("forecast" for the Sprint)
- A plan, often a list of tasks, to deliver the Product Backlog Items against the Sprint Goal
- Created by the Development Team during Sprint Planning
- Often derived by examining and decomposing PBIs
- Might be a simple To Do list
- Work for the Sprint emerges during the Sprint
- Development Team members sign up for work, they aren't assigned
- Development Team members may modify the Sprint Backlog anytime as they see fit
- Progress within the Sprint must be visualized
- Owned and managed by the Development Team

# Events (1)

## ■ SPRINT PLANNING

*from Product Backlog to Forecast, Sprint Goal and Sprint Backlog*

- Product Backlog is inspected
- A Sprint Goal is created
- Sprint Backlog is created
  - The Development Team selects PBIs for the Sprint in collaboration with the Product Owner
  - The selection of Product Backlog items into the Sprint Backlog represents the Development Team's forecast for what can be completed in this Sprint
- The entire Scrum Team attends

## ■ DAILY SCRUM

*from Daily Progress to Updated daily plan*

- 15 minute daily meeting
- Same time and place  
(this might not be possible in a University setting)
- An opportunity for the Development Team to
  - Assess progress toward the Sprint Goal
  - Synchronize activities
  - Create a plan for the next 24 hours



# Events (2)

## ■ SPRINT REVIEW

*from Sprint, Increment  
to Updated Product Backlog*

- The product Increment is inspected with the stakeholders
- Stakeholders are encouraged to provide feedback and to collaborate
- The Product Backlog is updated upon the new insights

## ■ SPRINT RETROSPECTIVE

*from Past Spring  
to Improvements for next Sprint*

- The Scrum Team discusses
  - What went well in the Sprint
  - What could be improved
  - What we will commit to improve in the next Sprint

## ■ SPRINT

*container event: 30 days or less in duration*

- A container for all activities and the other Scrum events
- Focus is on development activities
- Starts with Sprint Planning
- Ends with Sprint Retrospective
- 30 days or less to enable regular feedback

# Roles, Artifacts and Events in Action

have clear  
accountability

provide  
transparent  
information

serve  
inspection,  
adaptation and  
transparency

## Roles

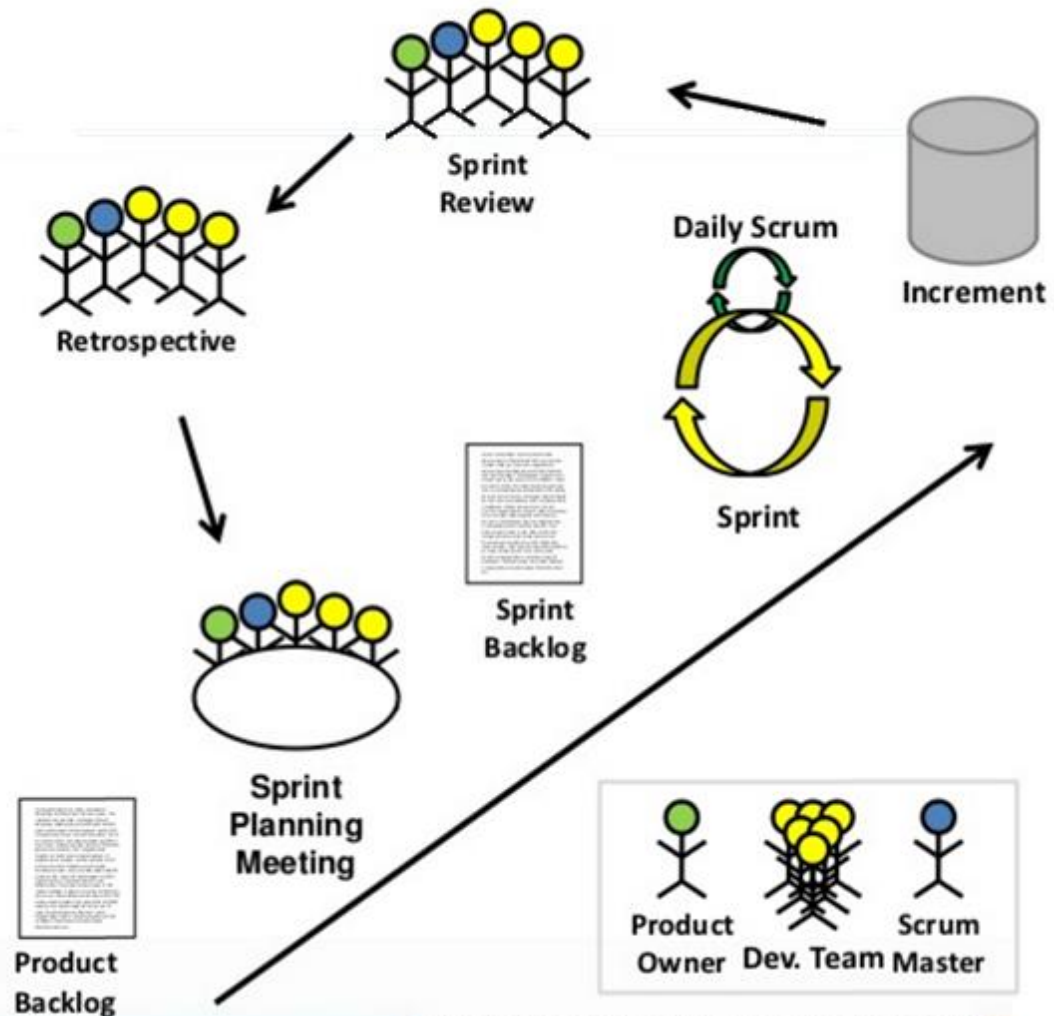
- Product Owner
- Development Team
- Scrum Master

## Artifacts

- Product Backlog
- Sprint Backlog
- Increment

## Events

- Sprint
- Sprint Planning
- Daily Scrum
- Sprint Review
- Retrospective



# Roles Revisited

# Who the Product Owner Is

- **Defines features and functionality**
  - The level of detail provided will vary
  - Some Product Owners will work closer to implementation details than others
- **Has the final word on the content and the ordering of the Product Backlog**
- **Not the Development Team's assistant**
  - If a member of the business, it is preferred that the Product Owner not be tasked with writing a User Story
  - Needs to spend as much time with the Development Team as needed

# Who the Scrum Master Is

- A servant-leader, management position
  - Managing the use and adoption of Scrum by the team and within the organization
  - Serving and coaching the Scrum Team
  - Embodying agility for all to see
- Not the Development Team's assistant
  - Optional to the Daily Scrum
  - Is not there to "drive the team" or "drive results"
  - Is not the agile variant of a Project Manager
- Guides and cares for the Team
  - Adheres to Scrum values, practices and rules
  - Understands and uses self-organization
  - Is productive
  - Increases the quality of Increments
- Removes impediments to the Development Team's success **that they are unable to remove themselves.**

# More on the Development Team

- The team composition is constant throughout a Sprint
- Typically has 3-9 members, with the sweet spot around 6
- May have partially allocated members
  - Although this is often considered an impediment as they have other priorities
  - For instance, Database Administrators, User Interface Design experts, Technical Writers
- Non-sequential execution is key
- Everyone pitches in regardless of individual skill specialty
- The Development Team is held to account **as a unit.**

# Techniques Complementary Practices

# PBIs

- **Transparent unit of deliverable work**
- **Contains clear acceptance criteria**
  - Criteria for successful completion
  - Answering "what will be true when this works"
- **May reference other artifacts like**
  - Specifications
  - Mockups
  - Architecture Models
- **Must be sized appropriately**
  - Must be completable (able to be "DONE") within a single Sprint
  - Best to have several PBIs in a Sprint
- **Valid Product Backlog Items**
  - Feature definitions
  - User actions or stories
  - Constraints
    - e.g., The system responds to all search requests within 10 seconds of receiving the request
  - Bugs / Defects
  - Behaviors (execution of tasks that can be verified with observation)
  - Use Cases
  - Desirements (aka functional require(?)ments)
  - Non-functional requirements
    - e.g., all database records must be replicated in a remote database environment





## Set up continuous integration system

in list [Sprint 1 Done](#)

Members

Labels

JM



Priority Medium

Difficulty 8



Description [Edit](#)

As a developer, I want to have a code repository that will enable integrating new or changed code from the development team and ensure that no errors can arise without developers noticing them and correcting them immediately.

Acceptance Criteria

- I can check code in and out of the repository
- I can inspect the version history of any code units including branches
- I can visualize code branches using a graphical representation of versions
- I can automate the build of a complete product or increment
- The system notifies me when code I have authored is changed
- The system provides the capability to integrate bugs and issues with code



### Checklist

[Hide completed items](#) [Delete...](#)

100%



~~Select the version control system~~



~~Determine strategies for development, delivery and application versioning~~



~~Test the build automation tool~~



~~Create and implement a sample unit test~~



~~Build a test environment~~



~~Build an acceptance environment~~

Add an item...

# Example PBI





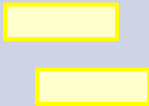


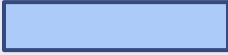

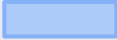
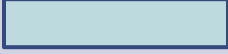

■ User story

■ Acceptance criteria

■ PBI decomposed into tasks

# A Scrum Board

- Scrum says to visualize progress on the Sprint Backlog. A Scrum board is an often used means to do so
- Note PBIs are decomposed to Tasks by the time they are represented in the To Do column

| Sprint #1                                                                           |                                                                                     |                                                                                      |                                                                                      |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| PBI                                                                                 | To Do                                                                               | In Progress                                                                          | Done                                                                                 |
|    |                                                                                     |   |   |
|   |   |  |  |
|  |  |  |                                                                                      |
|  |  |                                                                                      |                                                                                      |

*Each PBI is ideally discrete without dependencies*

# The Definition of DONE (DoD)

- An increment is deemed complete after all PBIs for the corresponding Sprint meet their done criteria and their acceptance criteria.
- In pure Scrum, **the team** defines the definition of done. However, those performing the work and those accepting the work product must share a common definition of “Done”.

## The DoD is NOT the same as Acceptance Criteria

- The Definition of Done is a clear and concise list of requirements that the user story must satisfy for the team to call it complete.
- The DoD is a list of technical and non-technical requirements for each user story to meet for the project. It must apply to all items in the backlog.
- DoD is a contract between the Scrum Team & the Product Owner.
- User story Acceptance Criteria can be modified over the course of the Sprint as the user story is further refined.
- The DoD is set during the Release Planning or Sprint 0 – not generally modified once active development print cycles are underway.
- The DoD is often confused with acceptance criteria because, like acceptance criteria, it is used to prove that a story is complete.
- The difference with the DoD is that it is a list of requisites that determine that a story meets company or project requirements, instead of only technical requirements.

# Sample Definition of Done (excerpted)

## ■ Preliminary Design.

- is done by showing artifacts that focus on identifying actors, black box software components and their inter-relationships. The architecture is represented by the following artifacts:
  - Subsystem context diagrams
  - Leaf subsystem level collaborations
  - Subsystem collaboration requirements trace diagram
  - Software Model Questions are all addressed.
  - Level 2 test plan is developed or updated
  - ...

## ■ Detailed Design

- is done by showing artifacts that focus on identifying relationships between a software component's architecturally significant classes. The Detailed Design is represented by the following artifacts:
  - Component level collaboration sequence diagrams
  - Component level collaboration class diagrams
  - ...

## ■ Code & Unit Test (CUT)

- Level 1 tests are done using CppUnitLite for C++ and JUnit for Java. Model updated to reflect design impact of code changes. All implementation artifacts from the story are ready for formal review.
  - All code written will include Code Headers & Commenting standards
  - Each new or modified level 1 test is named to clearly indicate its purpose
  - Code and level 1 test drivers are in the code repository on the appropriate branch
  - A subject matter expert has peer reviewed level 1 test procedures to verify sufficiency of preconditions tested and expected results verifications
  - Level 2 test procedures have been created

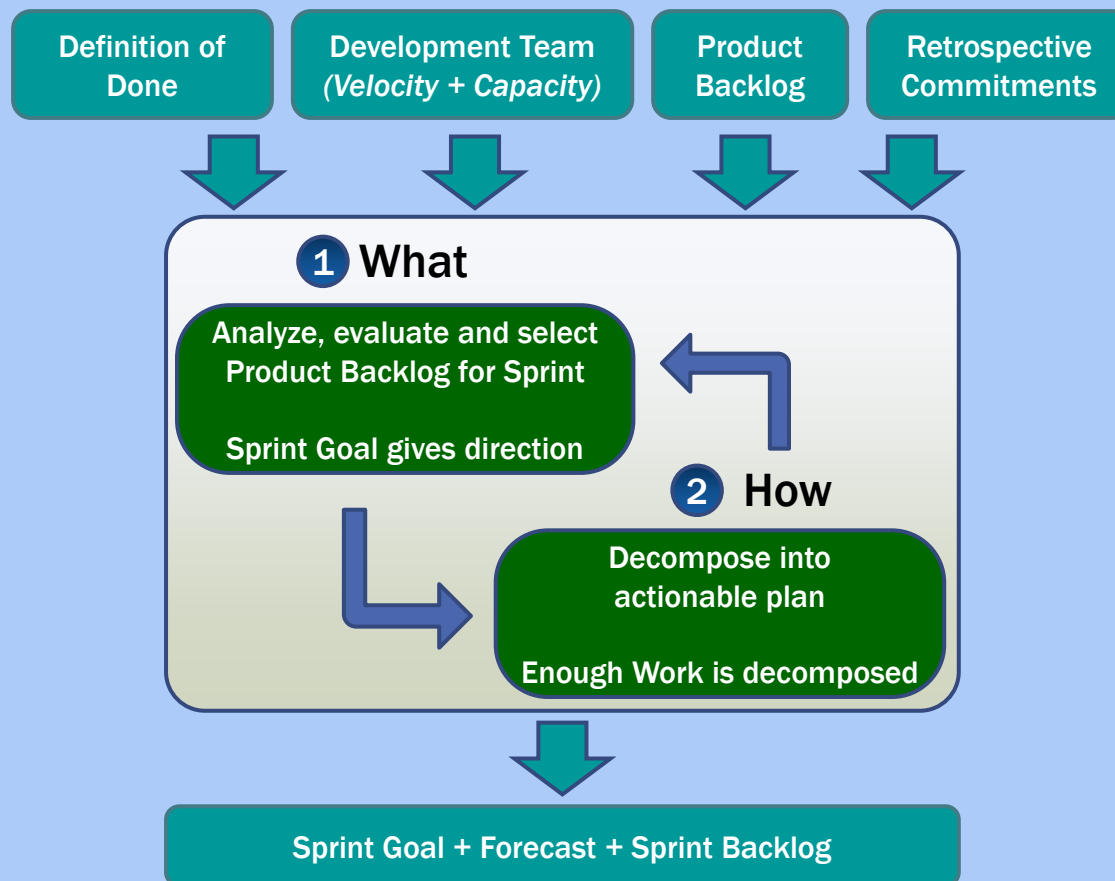
## ■ Integration

- is done when:
  - Verification and regression tests are complete
  - All open Level 2 open items have been documented in the test record and in additional software code records and stories.
  - Updated Level 2 test plans and procedures
  - ...

# Planning for the Increment and the Product

- The **product** is the sum of all increments
- **Release Planning** is used to plan out all sprints needed to produce the product.
  - Can be done at the PBI level.
  - This way the team can know the approximate size of the effort and can make sure they are marching toward completing that amount of work based in their **velocity** and total work plan.
  - **Burn up charts** and **feature burn down charts** can be used for this.
- **Sprint Planning** is used to plan the current Sprint, i.e., the current Increment
  - Has two main topics: What will we do? How will we do it?
  - 8 hours for a 4 week Sprint
    - Usually less for shorter Sprints
  - Often the day after the Sprint Retrospective
  - The full Scrum Team participates
  - What do you think the activities and responsibilities are during Sprint Planning are for:
    - Product Owner
    - Development Team
    - Scrum Master

# Sprint Planning Meeting Flow



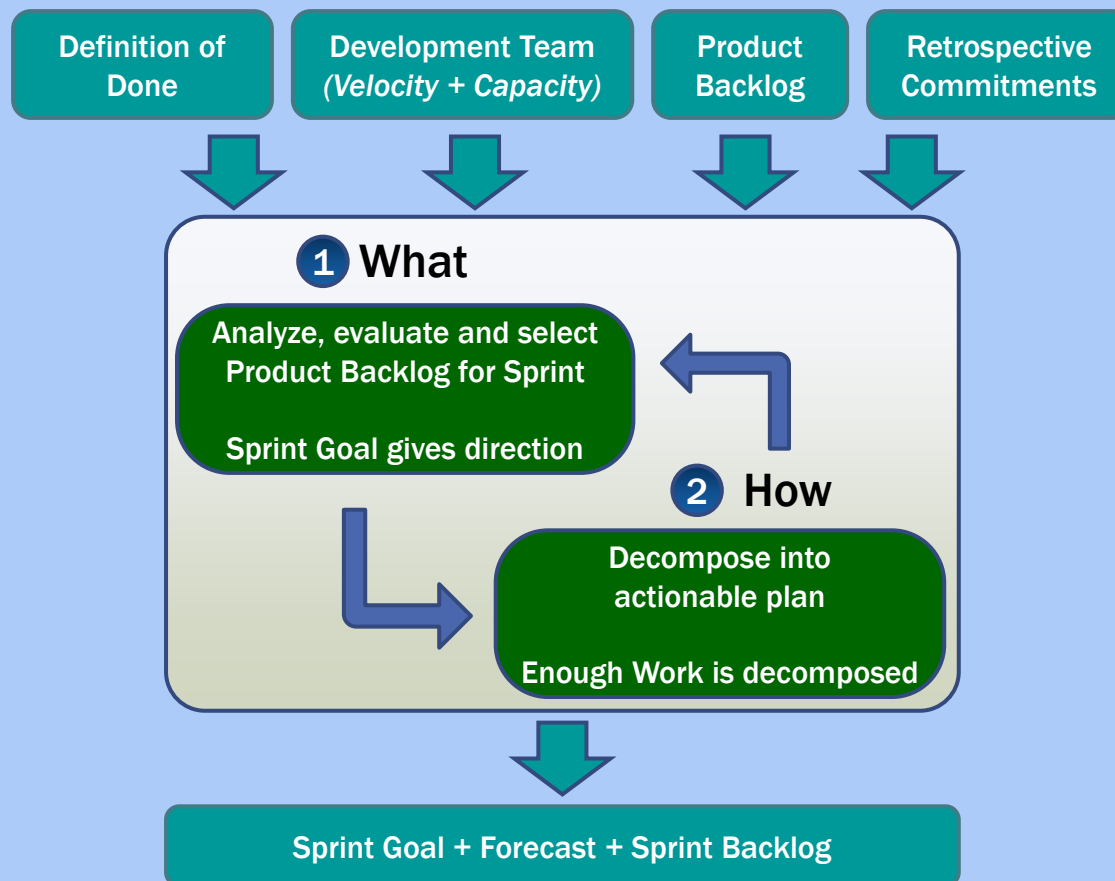
## Inputs

- Definition of Done
- Development Team Capacity for next Sprint
- Past performance of the Development Team
- The Ready Product Backlog
- The latest Increment
- Retrospective commitments

## Outcome

- The Sprint Goal
- Forecast
- Sprint Backlog
  - 60-70% known and accurate
  - Some work is stubbed out for discovery later
  - 100% time commitment is going to fail

# Sprint Planning Roles



## ■ Product Owner

- Brings in new PBIs, prioritized
- Clarifies information

## ■ Development Team

- Create plan
- Ask questions

## ■ Scrum Master

- Identifies impediments
- Conducts risk management

# Monitoring Sprint Progress

- Measurement is for the Development Team and no one else
- This is part of self-managing the Sprint's work
- Measurements are indications of
  - Progress in the Sprint
  - If scope should be re-negotiated with the Product Owner
- Sprint progress monitoring can be easily misused
  - To micromanage the Development Team
  - To demonstrate false progress
- Progress may change abruptly when
  - New work is added or removed during the Sprint
  - Scope is renegotiated with the Product Owner
  - New things are learned about the work of the Sprint



# Sprint Goals

**An objective to be met in the Sprint**

- Through the implementation of the PBIs selected in Sprint Planning
- Providing guidance to the Development Team

**Allows flexibility in delivering the increment**

- Allows wiggle room for exact implementation of PBIs
- Although the Sprint Goal is fixed

**Is sacrosanct throughout the Sprint**

- As the development team works, it keeps this goal in mind
- Each Daily Scrum assesses the Team's progress toward meeting the Sprint

Make the app run on SQL Server in addition to Oracle

Increase find accuracy of search by 20%

Deliver a minimal set of admin features

Decrease upstream payloads by 30%

# The Daily Scrum



## ■ DAILY SCRUM

*from Daily Progress to Updated daily plan*

- 15 minute daily meeting
- Same time and place  
(this might not be possible in a University setting)
- An opportunity for the Development Team to
  - Assess progress toward the Sprint Goal
  - Synchronize activities
  - Create a plan for the next 24 hours

## ■ Why a Daily Scrum?

- Share commitments
  - Identify impediments
  - Create focus
  - Increase and maintain situational awareness
- ## ■ Good idea to update the Sprint Backlog and Burndown charts at the Daily Scrum

What did I do yesterday  
that helped the  
Development Team meet  
the Sprint Goal?

What will I do  
today to help the  
Development Team  
meet the Sprint  
Goal?

Do I see any impediment  
that prevents me or the  
Development Team from  
meeting the Sprint Goal?

# The Sprint

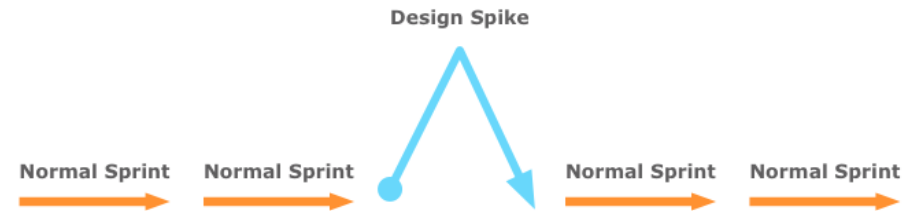


## ■ SPRINT

*container event: 30 days or less in duration*

- A container for all activities and the other Scrum events
  - Focus is on development activities
  - Starts with Sprint Planning
  - Ends with Sprint Retrospective
  - 30 days or less to enable regular feedback
- All Sprints have consistent duration
  - Sprints begin as soon as the previous Sprint ends
  - Scope is negotiated constantly throughout
    - Between Development Team and Product Owner
    - This recognizes uncertainty even within the Sprint

# Hitting a Spike



- Sometimes the Development Team has difficulty estimating the unknown
  - "We don't know enough to size this PBI"
  - "The technology is foreign to us"
  - "We don't know enough about the security capabilities of this software"
- The team may have hit a spike

*A user story or task aimed at answering a question or gathering information, rather than at producing shippable product.*

- The spikes teams take on are often proof-of-concept types of activities.
- Work is not focused on the finished product. It may even be designed to be thrown away at the end.
- Product Owner given the proper expectation that spike solution will most likely not be implemented.

## DEALING WITH SPIKES

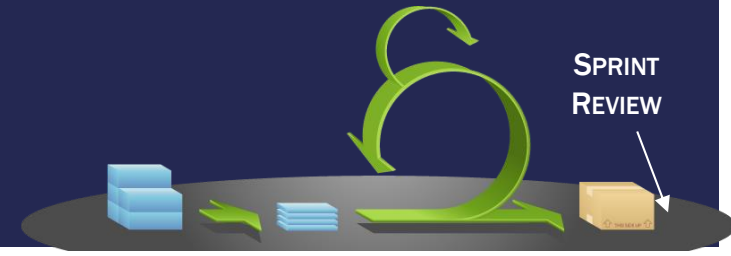
- **Have clear objectives and outcomes for the spike.**

Be clear on the knowledge you are trying to gain and the problem(s) you are trying to address. It's easy for a team to stray off into something interesting and related, but not relevant.

- **Be timeboxed.**

Spikes should be timeboxed so you do just enough work that's just good enough to get the value required.

# Sprint Review Meeting



## ■ SPRINT REVIEW

*from Sprint,  
Increment  
to Updated Product  
Backlog*

- The product Increment is inspected with the stakeholders
- Stakeholders are encouraged to provide feedback and to collaborate
- The Product Backlog is updated upon the new insights

- A collaborative working session, not a demo
- The Scrum Team shows the increment
- Feedback is heard from all present
  - Feedback used to guide the next increment
  - No slides! Just working software
  - 4 hours for a 4 week Spring
    - Usually less for shorter Sprints
  - The full Scrum Team participates
    - Complemented by the stakeholders

### Product Owner Shares

- What was done
- What wasn't
- State of the Product Backlog
- Likely Release projections

### Development Team Shares

- The actual Increment of software
- What happened in the Sprint
- How many problems were addressed and the effect on the Increment

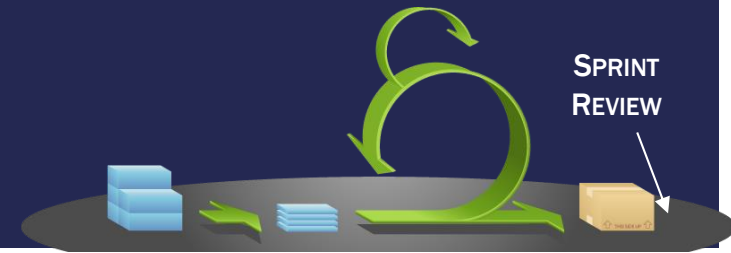
### Everyone

- Provides and hears feedback

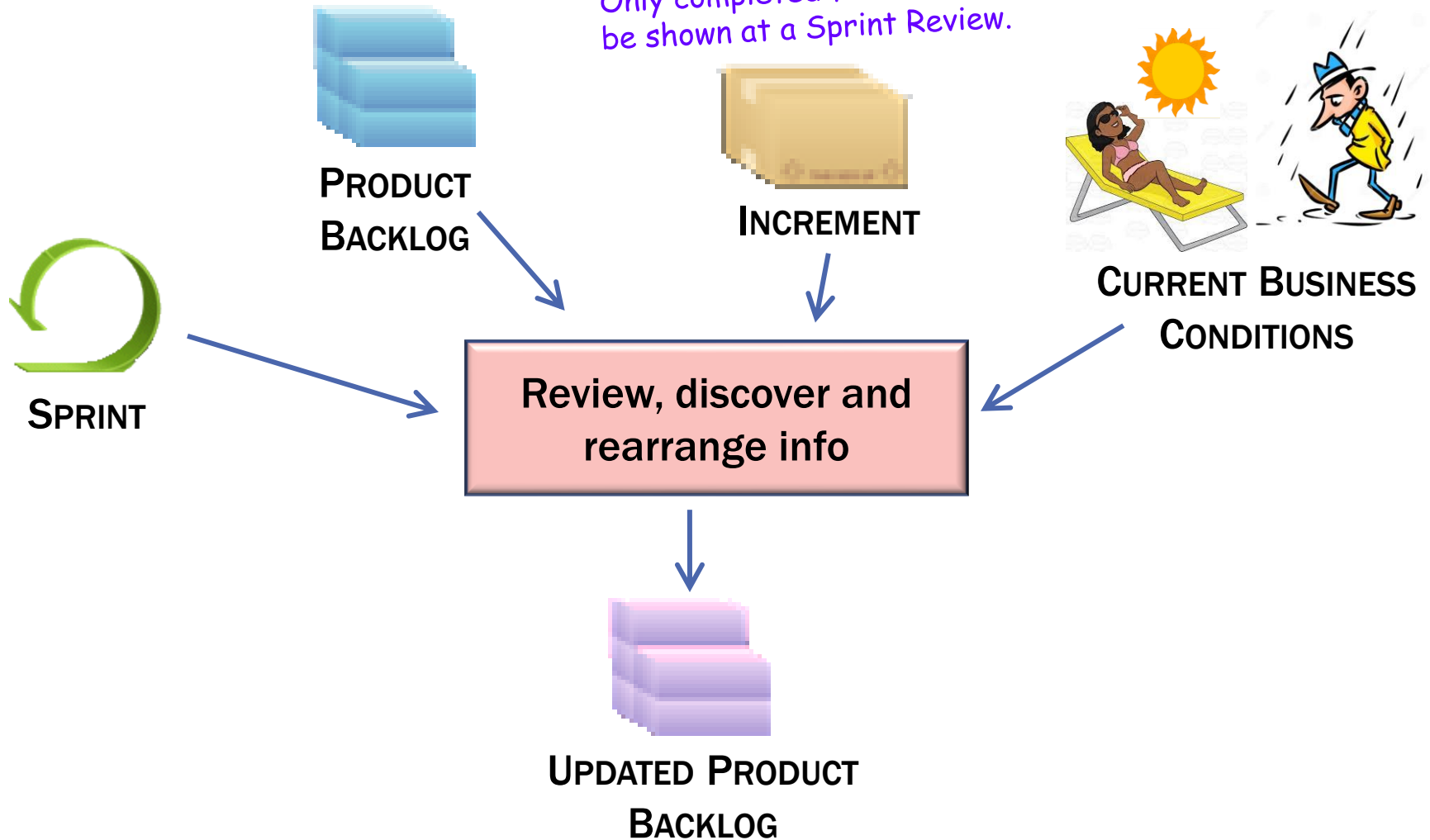
This is not an excuse for self-congratulations!

This is an inspect and adapt opportunity!

# Sprint Review Meeting

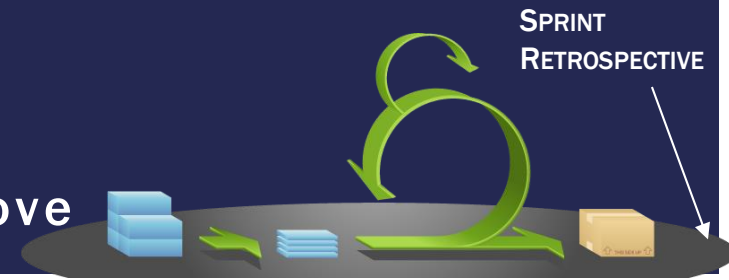


Only completed features may be shown at a Sprint Review.



# Sprint Retrospective:

## The Scrum Team's Opportunity to Improve



### ■ SPRINT RETROSPECTIVE

*from Past Sprint  
to Improvements for  
next Sprint*

- The Scrum Team discusses
  - What went well in the Sprint
  - What could be improved
  - What we will commit to improve in the next Sprint

- A discussion of
  - The Scrum process
  - Scrum Team members behaviors
  - Tools used and needed
  - Expanding the Definition of Done (based on Sprint Review)
- Goal: to find actionable improvements
  - The Scrum Team can enact on the next Sprint
  - To adapt common practices and techniques
  - To increase the Definition of Done
- Typically 3 hours for a 4-week Sprint
  - Usually less for shorter Sprints
- Occurs after every Sprint Review
- Full Scrum Team participation
  - Product Owner
  - Scrum Master
  - Development Team

Scrum Team members make  
actionable commitments

What worked well?

What will we commit to doing in the next Sprint?

What could be improved?

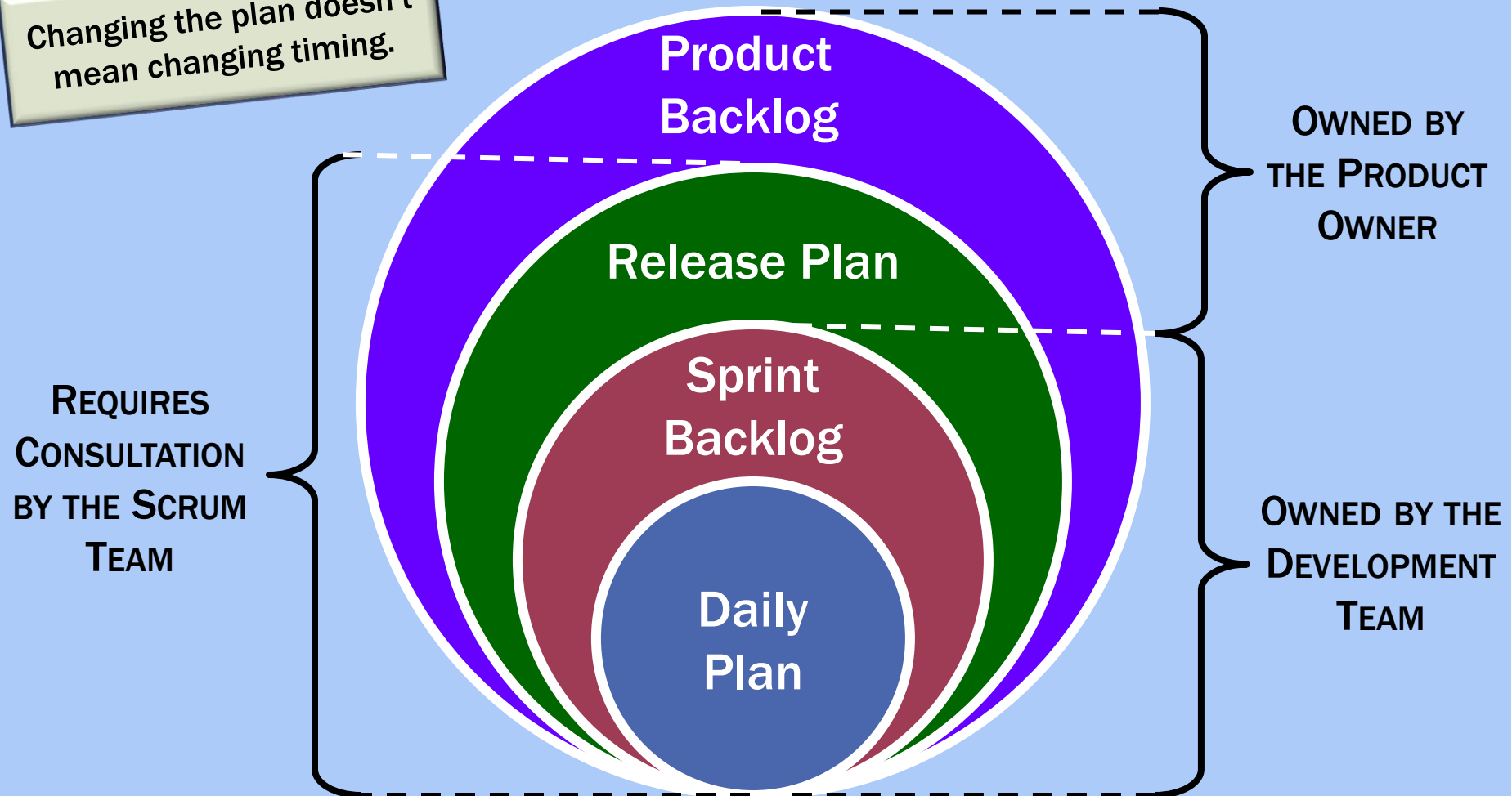
# Event lengths and Sprint length

| Event                | 20 days    | 3 weeks                         | 2 weeks                        | 1 week                          |
|----------------------|------------|---------------------------------|--------------------------------|---------------------------------|
| Sprint Planning      | 8 hours    | Less than 8 hours (~6 hours)    | Less than 8 hours (~4 hours)   | Less than 8 hours (~2 hours)    |
| Daily Scrum          | 15 minutes |                                 |                                |                                 |
| Sprint Review        | 4 hours    | Less than 4 hours (~3 hours)    | Less than 4 hours (~2 hours)   | Less than 4 hours (~1 hour)     |
| Sprint Retrospective | 3 hours    | Less than 3 hours (~2.25 hours) | Less than 3 hours (~1.5 hours) | Less than 3 hours (~45 minutes) |

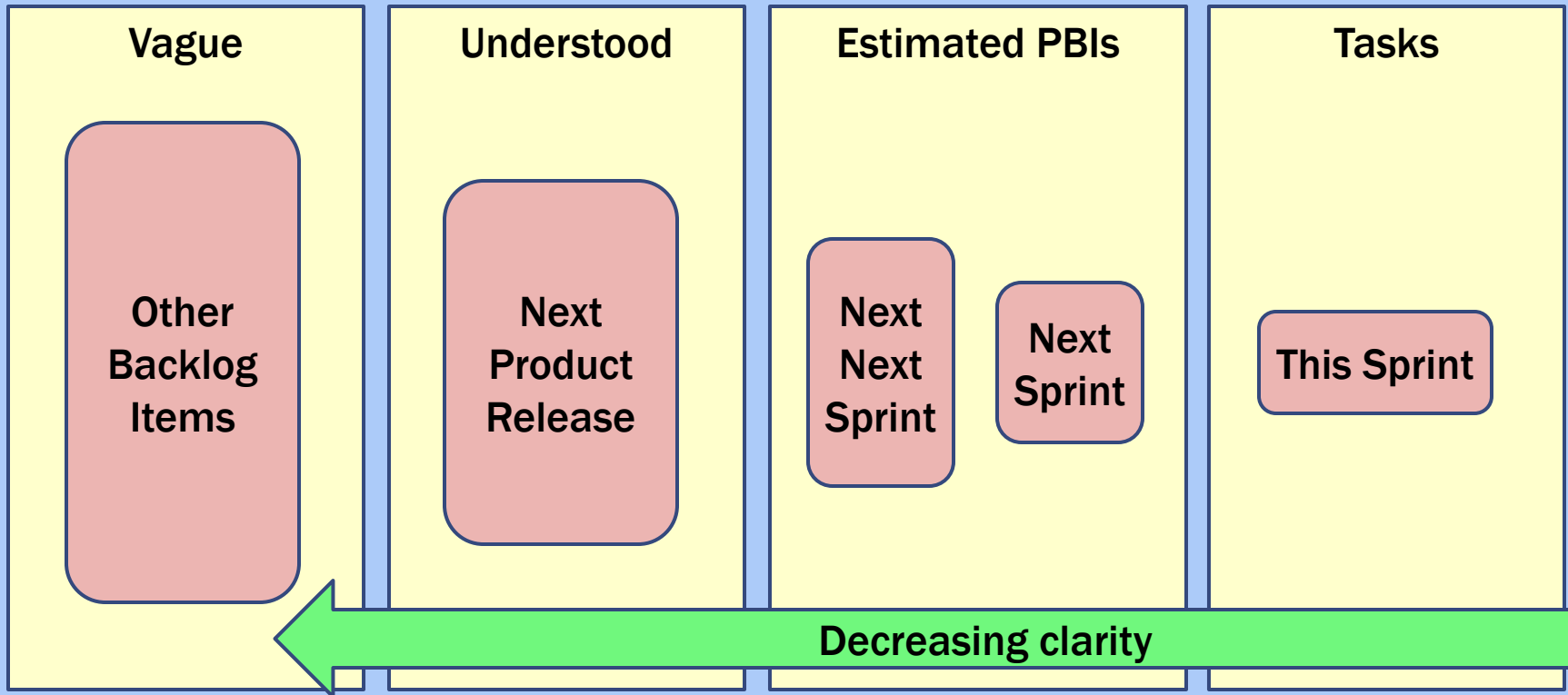


# Planning Horizons

Changing the plan doesn't mean changing timing.



# Backlog Accuracy and Item Detail



- PBIs are estimated and ordered for approximately the next 3 Sprints
- The current Sprint is detailed
  - Broken into Sprint Backlog Tasks
  - Very granular detail
- The next 2 Sprints are understood by the entire Scrum Team. They are estimated, valued, ordered, loosely planned.

# A 'prescribing medication' story

## **Prescribing medication**

The record of the patient must be open for input. Click on the medication field and select either 'current medication', 'new medication' or 'formulary'.

If you select 'current medication', you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, 'new medication', the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose 'formulary', you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

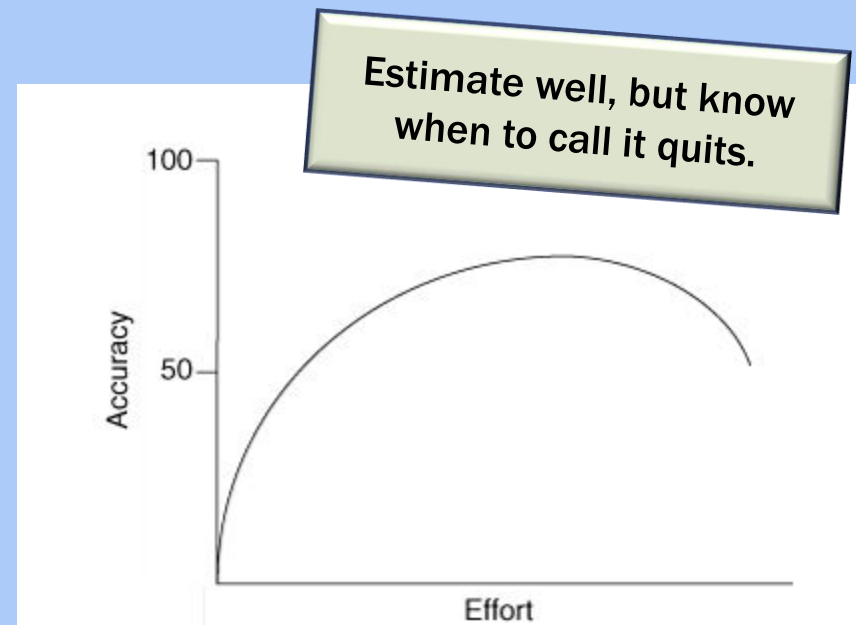
In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click 'OK' or 'Change'. If you click 'OK', your prescription will be recorded on the audit database. If you click 'Change', you reenter the 'Prescribing medication' process.

# PBI Refinement:

## Progression from vague idea to specific tasks

- Refining means
  - Planning the PBI to an actionable level of detail
  - Maintaining a Rolling Backlog Projection
- Plan 10% of the Sprint capacity of the Development Team to be spent on refining the Product Backlog
- Top ordered Product Backlog items are well understood and easily selected in Sprint Planning. These are **READY**.
- Estimate in groups. Group derived estimates are demonstrably more accurate than estimates by individuals.



Source: Mike Cohn ("Agile Estimating and Planning")

# Gherkin Syntax

- Gherkin is a line-oriented language that uses indentation to define structure.
- Line endings terminate statements (called steps) and either spaces or tabs may be used for indentation.
- Every scenario consists of a list of steps, which must start with one of the keywords Given, When, Then, But or And
- Serves two purposes
  - Acts as project documentation
  - Useful in automating feature testing



**Feature:** Some terse yet descriptive text of what is desired  
In order to realize a named business value  
As an explicit system actor  
I want to gain some beneficial outcome which furthers the goal

**Scenario:** Some determinable business situation

**Given** some precondition  
**And** some other precondition  
**When** some action by the actor  
**And** some other action  
**And** yet another action  
**Then** some testable outcome is achieved  
**And** something else we can check happens too

**Scenario:** A different situation

...

*Feature: LogIn Action Test*

*Description: This feature will test a LogIn and LogOut functionality*

*Scenario: Unsuccessful Login with Invalid Credentials*

*Given* user is on Home Page  
*When* user navigates to the LogIn Page  
*And* user enters UserName and Password  
*But* the user credentials are wrong  
*Then* wrong UserName & Password message displayed

# Early PBI



## Set up continuous integration system

Members

Labels

Priority or  
Business Value

Priority Medium

Description [Edit](#)

User Story

As a developer, I want to have a code repository that will enable integrating new or changed code from the development team and ensure that no errors can arise without developers noticing them and correcting them immediately.

- This PBI is only vaguely understood
- A user story is developed and the PBI may or may not be prioritized.
- A full refined PBI

### A Refined PBI

**Name:** *clear, short, atomic (single entity)*

**Value:** *business value, acts as priority*

**Description:** *the user story*  
*"As a \_\_\_, I want a \_\_\_ that \_\_\_"*

**Acceptance**

**Criteria:** *"Given \_\_\_ when \_\_\_ then \_\_\_"*

**Sizing:** *how difficult is the development*

**READY**

# Refined PBI

The screenshot shows a Jira Product Backlog item. The title is "Set up continuous integration system". Below the title, there are tabs for "Members" and "Labels". The "Labels" tab is active, showing two labels: "Priority Medium" (yellow) and "Difficulty 8" (green). To the right of the labels is a "Sizing" button with a plus sign. Below the labels, there is a "Description" field with an "Edit" link. The description text is: "As a developer, I want to have a code repository that will enable integrating new or changed code from the development team and ensure that no errors can arise without developers noticing them and correcting them immediately." Below the description, there is an "Acceptance Criteria" section with a list of five criteria. Annotations with purple arrows point to various parts of the interface: "Name" points to the title, "Priority or Business Value" points to the "Priority Medium" label, "Sizing" points to the "Difficulty 8" label, "User Story" points to the description text, and "Acceptance Criteria" points to the list of criteria.

**Set up continuous integration system** ← Name

Members Labels ← Priority or Business Value

Priority Medium Difficulty 8 ← Sizing

Description Edit ← User Story

As a developer, I want to have a code repository that will enable integrating new or changed code from the development team and ensure that no errors can arise without developers noticing them and correcting them immediately.

Acceptance Criteria ← Acceptance Criteria

- I can check code in and out of the repository
- I can inspect the version history of any code units including branches
- I can visualize code branches using a graphical representation of versions
- I can automate the build of a complete product or increment
- The system notifies me when code I have authored is changed
- The system provides the capability to integrate bugs and issues with code

- Because requirements in Scrum are only loosely defined, they need to be revisited and clearly defined before they come into the Sprint.

- This is done during the current Sprint in a ceremony called Product Backlog Refinement.

- Acceptance Criteria that follows Gherkin syntax are often more specific and clear because they set up pre-conditions and post-conditions.
- This PBI is **Refined** and **Ready** to be moved from the Product Backlog to the Sprint Backlog

# Decomposed PBI

- A Refined and Ready PBI may be pulled into the Sprint backlog.
- However, the PBI must be **Decomposed** into Tasks before it can be worked on.
- Caution: Don't over focus on defining tasks

## Set up continuous integration system

in list Sprint 1

Members

JM

- A Refined and Ready PBI may be:
  - placed into the Sprint Backlog
  - chosen by a member of the Development Team

Description Edit

As a developer, I want to have a code repository that will enable integrating new or changed code from the development team and ensure that no errors can arise without developers noticing them and correcting them immediately.

Acceptance Criteria

- I can check code in and out of the repository
- I can inspect the version history of any code units including branches
- I can visualize code branches using a graphical representation of versions
- I can automate the build of a complete product or increment
- The system notifies me when code I have authored is changed
- The system provides the capability to integrate bugs and issues with code



## Checklist

Hide completed items Delete...

100%



~~Select the version control system~~



~~Determine strategies for development, delivery and application versioning~~



~~Test the build automation tool~~



~~Create and implement a sample unit test~~



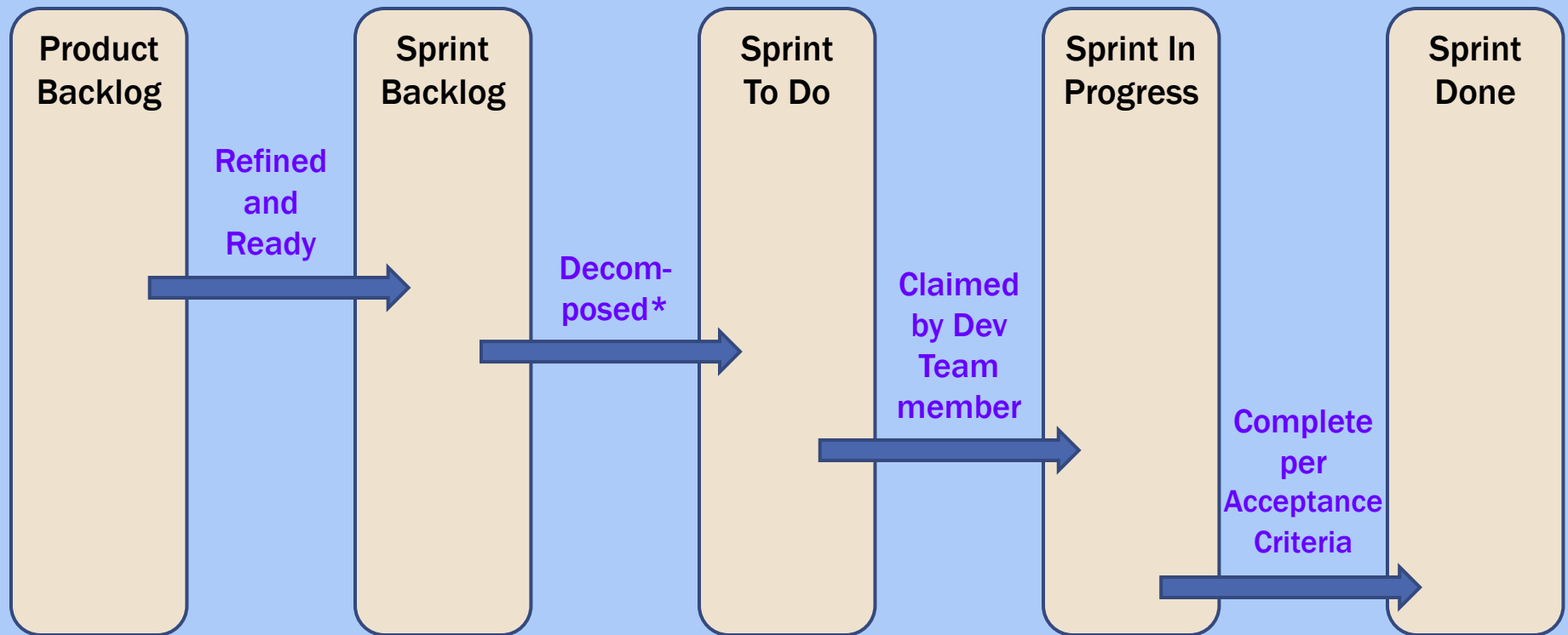
~~Build a test environment~~

PBI is  
decomposed  
into Tasks



# Proper Use of Scrum Boards (and Trello)

## ■ When to move a PBI in a Scrum Board



\* In fully functional scrum boards, the decomposed tasks move to "To Do" but the tasks retain their association with the PBI. In Trello, PBIs move as a whole.

# Assigning Points for PBI Sizing

## ■ Story points


- A very common way to estimate work based on size, effort and complexity, not duration
- Unit less and numerically relative
- Points are additive, based on historical reality and easy to use and understand
- Common practices: Fibonacci sequence [1, 2, 3, 5, 8, 13, 21, 34 ]
- Sometimes other cards are added: ? (I stil have questions) and  $\infty$  (too big to estimate)

## ■ Planning Poker

- Each PBI has a size.
- Each developer has a deck of estimation cards.
- Product Owner reads a user story and it's discussed briefly
- Each developer selects a card that's his or her estimate
- Cards are turned over at the same time so all can see them
- Differences are discussed (especially outliers)
- Re-estimate until estimates converge
- Eventually, as the team gets better at estimating, use older PBIs as a relative measure of new work. "Remember that previous bit of work? This one is like it. And the previous one was a 13."

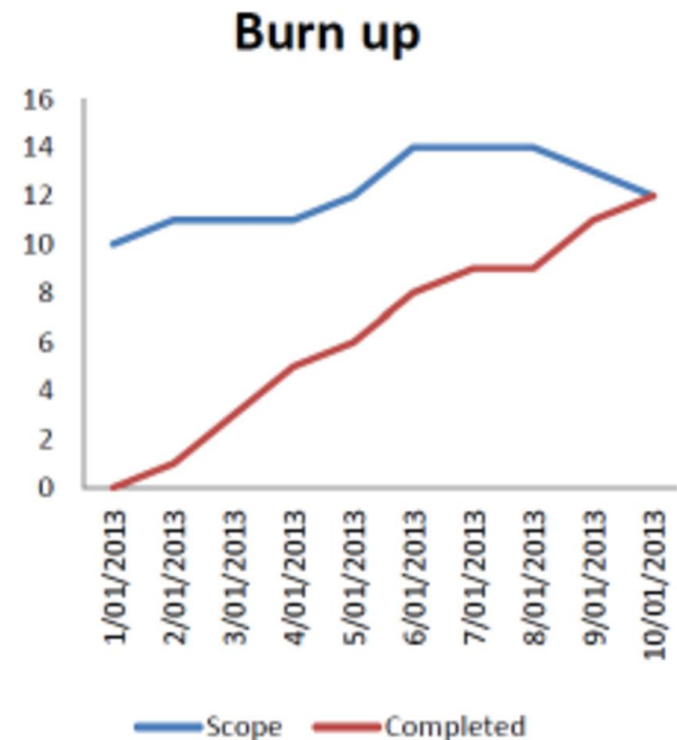
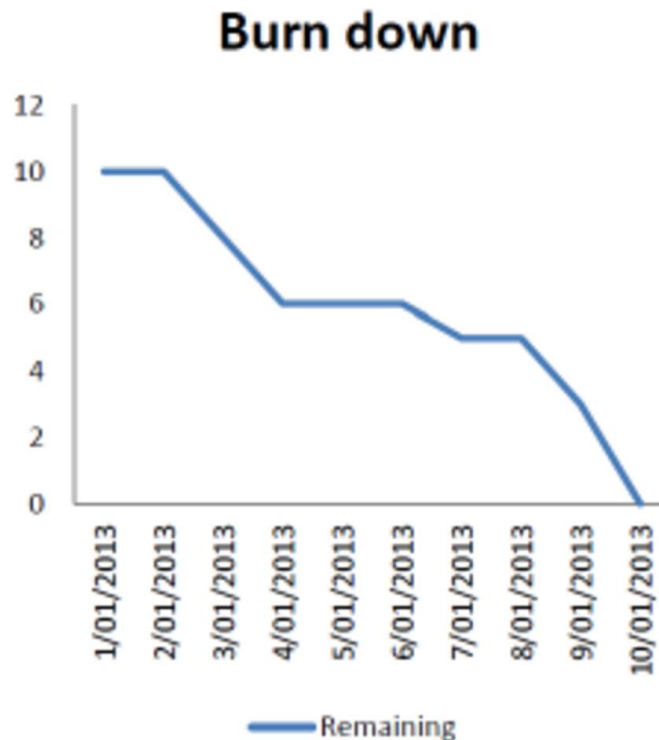


# Business Value

- The Product Owner is responsible for this
- It isn't always just revenue
- It can be estimated or calculated
  - Can be High / Medium / Low
  - Can use MoSCoW 
  - Can be numeric value
- **M**ust have
  - Critical to the current delivery timebox in order for it to be a success. If even one MUST requirement is not included, the project delivery should be considered a failure
  - MUST can be considered an acronym for the Minimum Usable SubseT.
- **S**hould have
  - Important but not necessary for delivery in the current delivery timebox.
  - While SHOULD requirements can be as important as MUST, they are often not as time-critical or there may be another way to satisfy the requirement, so that it can be held back until a future delivery timebox.
- **C**ould have
  - Desirable but not necessary, and could improve user experience or customer satisfaction for little development cost. These will typically be included if time and resources permit.
- **W**on't have
  - the least-critical, lowest-payback items, or not appropriate at that time. As a result, WON'T requirements are not planned into the schedule for the delivery timebox. WON'T requirements are either dropped or reconsidered for inclusion in later timeboxes.

# Assigning Points for PBI Value

- Business Value can also be assigned points as well
- Y axis of burn down and burn up charts can be Sizing or Business Value
- Burn up charts may be more accurate – they show scope change



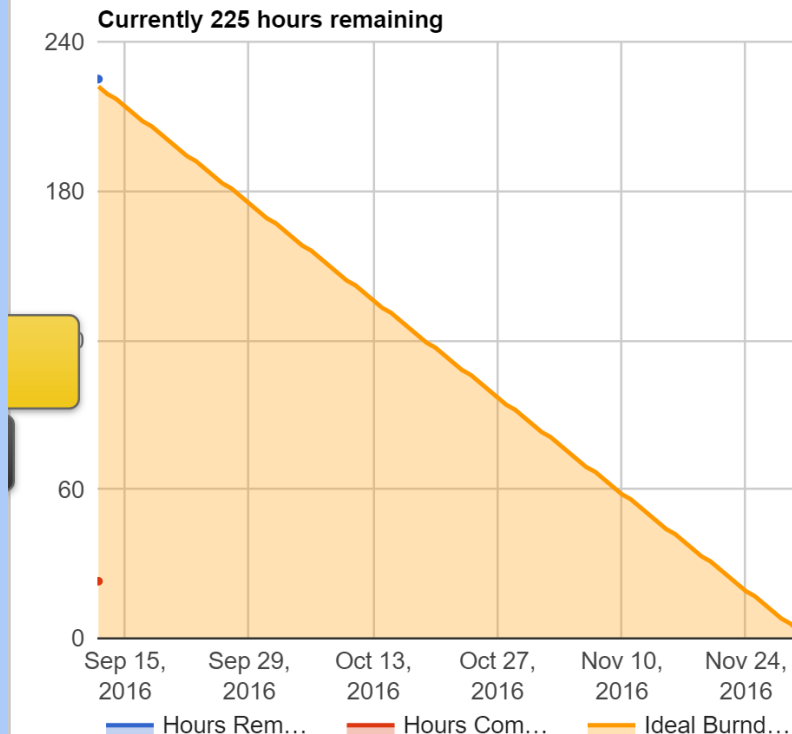
A burn down and burn up chart of the same project. In the burn down chart it appears that the team did not accomplish much in the middle of the project but heroically finished everything at the end. The burn up chart shows the complete picture - that the scope increased at the beginning of the project, and some scope was removed to finish the project by the deadline, whilst the team made steady progress through the entire duration of the project.

# Burndown for Trello

Loaded directly from the Trello API

Reload from Trello

## Burndown Chart



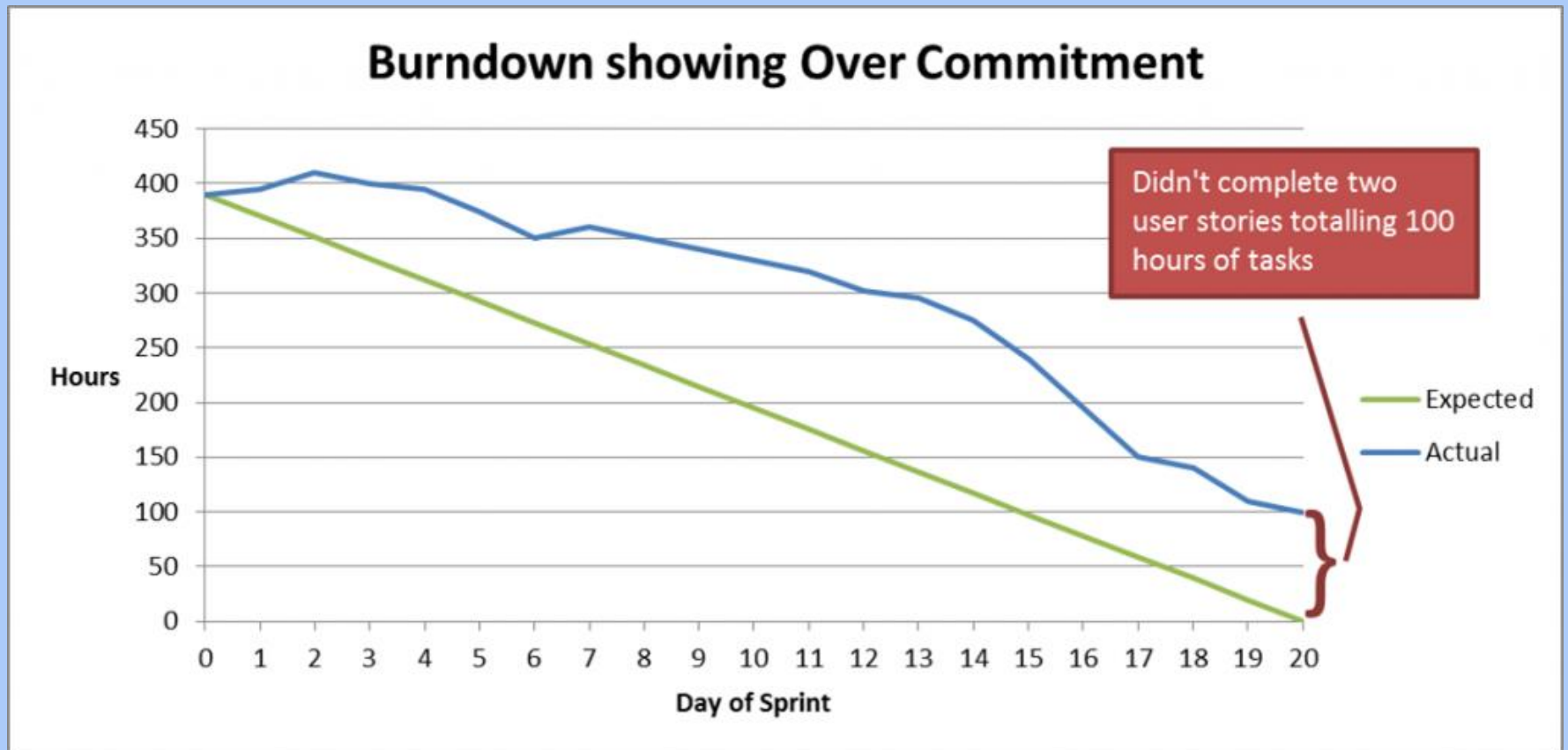
(+)

## Summary stats

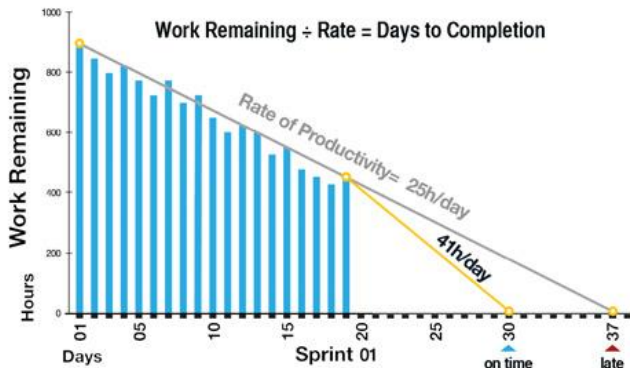
|                        |                            |
|------------------------|----------------------------|
| Total Cards:           | 18                         |
| Remaining Cards:       | 16                         |
| Done Cards:            | 2                          |
| Percent of cards done: | <div><div></div></div> 11% |
| Hours at start:        | 225 (edit)                 |
| Hours est total:       | 248                        |
| Hours remaining:       | 225                        |
| Hours done:            | 23                         |
| Percent of hours done: | <div><div></div></div> 9%  |
| ----                   | ----                       |
| Days Elapsed           | 0                          |
| Daily Burndown         | 23                         |
| Est. Days Left         | 10                         |
| Est. Completion Date   | 09/14/16                   |

Burndown for Trello bases burndown on number of hours (Y axis)

# Burndown can show over-commitment problems



# Visualization Suggestions for Project

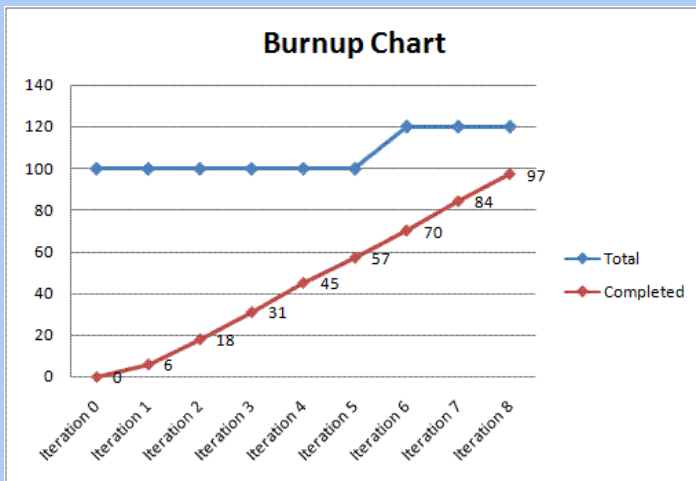


## ■ Sprint Visualization

- Use Trello's Burn Down capability (Or Excel with difficulty on Y axis)
- Accept hours as unit for Y axis
- Update at Daily Scrum

## ■ Release Visualization (Sprints 1 to 5)

- Use Burn Up charts
- Create two charts
  - 1 with Business Value on the Y axis
  - 1 with Effort (PBI Sizing) on the Y axis
- Update prior to each Spring Review

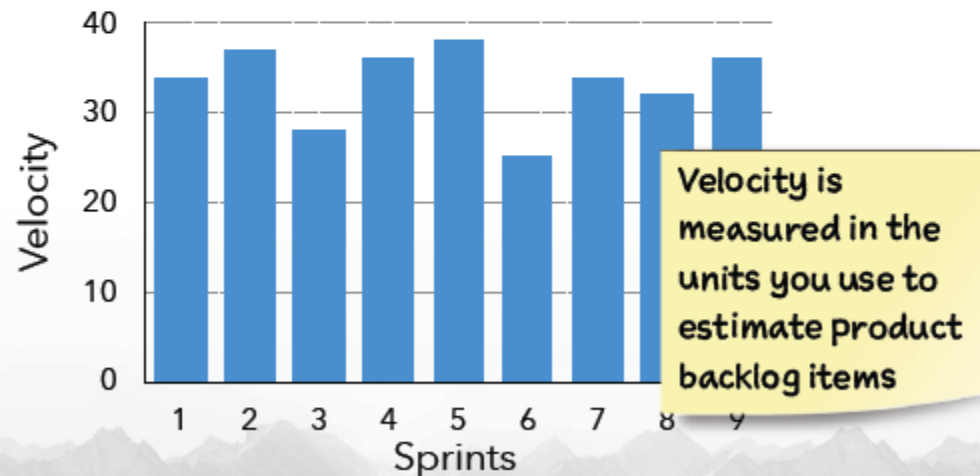


# Velocity

- A measure of Product Backlog Items delivered per Sprint
- Used by the Development Team to gauge how much work to pull in a Sprint Planning meeting
- Used by Product Owner to provide forecasts on the Product Backlog completion

## Velocity

- ❖ The amount of work completed in a sprint
- ❖ Most useful over at least a handful of sprints



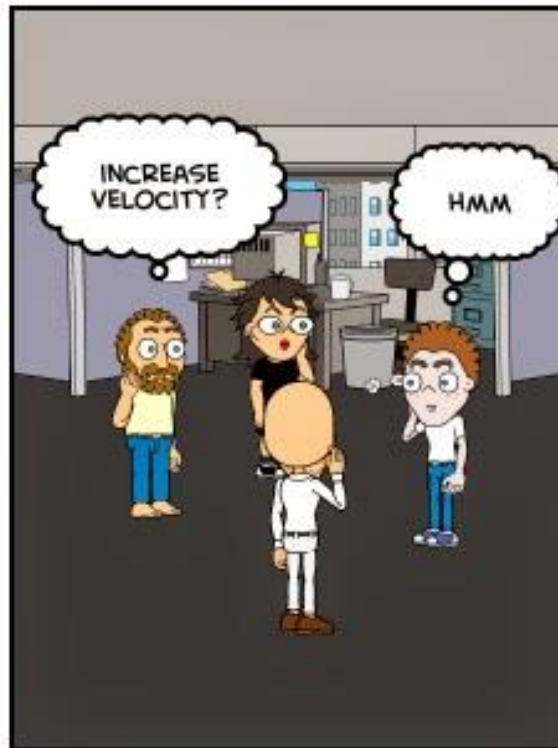


# Velocity

## Don't Do This

- Velocity is supposed to help Development Teams plan better Sprints
- It is NOT supposed to be a tool for managers to assess teams!

OBSERVER EFFECT - AGILE



BY ONTIMENOW.COM



# Ordering the Product Backlog

- Risk
  - Identify risk for items in the Backlog
  - Do the highest risk items first
- Return on Investment (ROI)
  - Simple business ranking system
  - Gives a single number by which to rank work
  - You can see value at a glance and this is less subjective
- Because the Product Owner says so

## RULE #1

An accurate release plan requires an ordered and estimated backlog

## RULE #2

An accurate release plan requires known Velocity

# Two Basic Types of Release Planning

## DATE TARGET PLANNING

- The product will release on a specific date

### We Must Answer

- How much of the backlog will be complete by a given date?

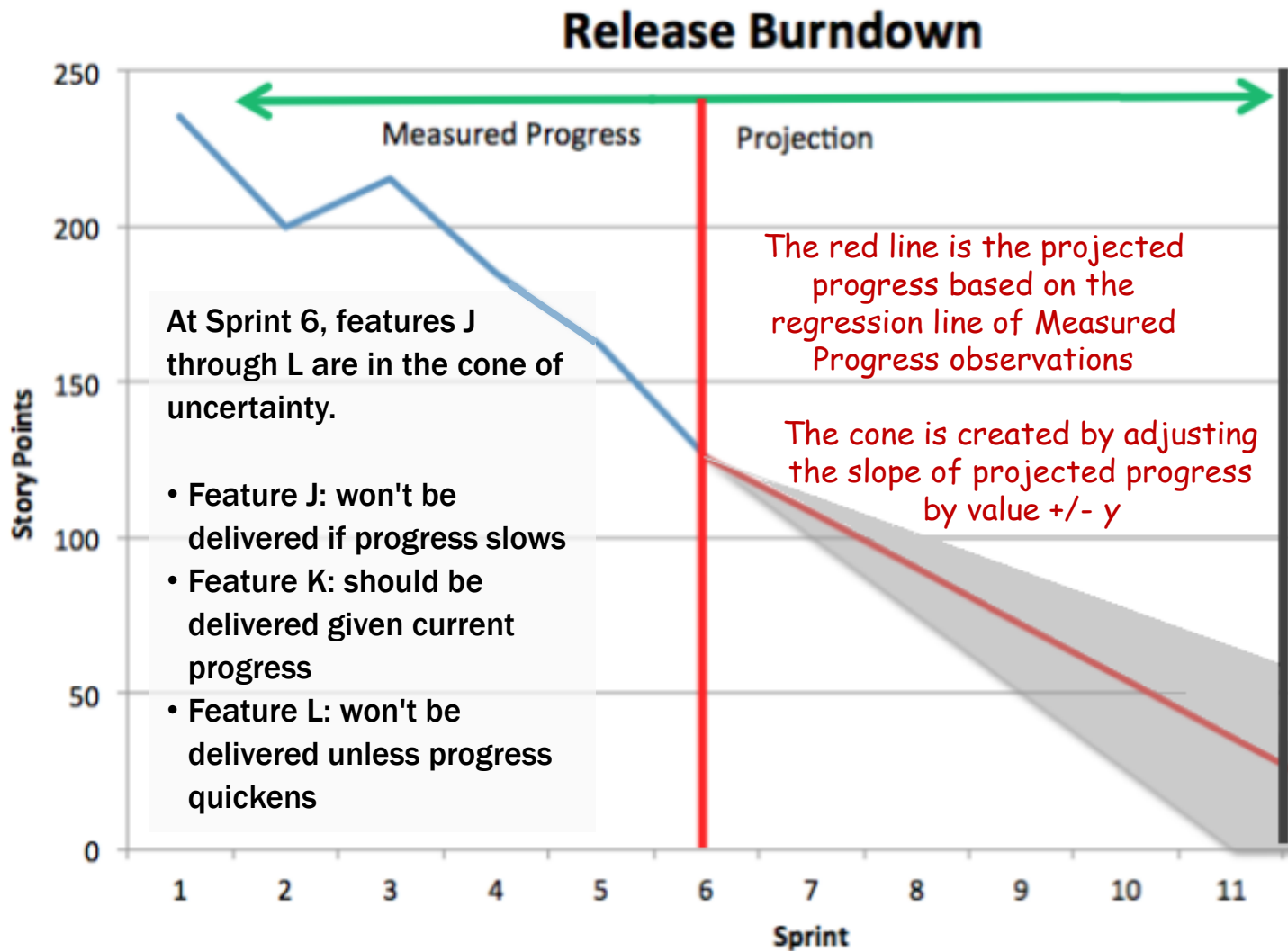
## FEATURE TARGET PLANNING

- The product will release when specific features are ready

### We Must Answer

- When will features A, B, and C be ready?

# Cone of Uncertainty



## Prioritized Product Backlog

Feature A

Feature B

Feature C

Feature D

Feature E

Feature F

Feature G

Feature H

Feature I

Feature J

Feature K

Feature L

# Sprint 0

- Per Scrum, there is no Sprint 0
- However, many organizations create a Sprint 0, for example:
  - The planning team is responsible for producing 3 deliverables by the end of the planning iteration:
    - A list of all prioritized features/stories with estimates
    - A release plan that assigns each feature/story to an iteration/sprint
    - A high-level application architecture, i.e. how the features will likely be implemented
- Ken Schawber, co-creator of Scrum agrees: "Sprint 0 has become a phrase misused to describe the planning that occurs prior to the first sprint".

# Hybrid models

# Agile and plan-driven methods

- Most projects include elements of plan-driven and agile processes. Deciding on the balance depends on:
  - Is it important to have a very detailed specification and design before moving to implementation? If so, you probably need to use a plan-driven approach.
  - Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic? If so, consider using agile methods.
  - How large is the system that is being developed? Agile methods are most effective when the system can be developed with a small co-located team who can communicate informally. This may not be possible for large systems that require larger development teams so a plan-driven approach may have to be used.

# Collaborative Model

## ■ Collaborative Model

### ■ See

<http://jackmyers.info/sweng/hybridCollaborative.html>

- "The Collaborative Model aims to combine both the models – Waterfall and Agile. Leveraging both the Waterfall and Agile approach ensures the success of the project. It removes the disadvantages of both the models; while bringing together the advantages of both."

Requirements Analysis and Design



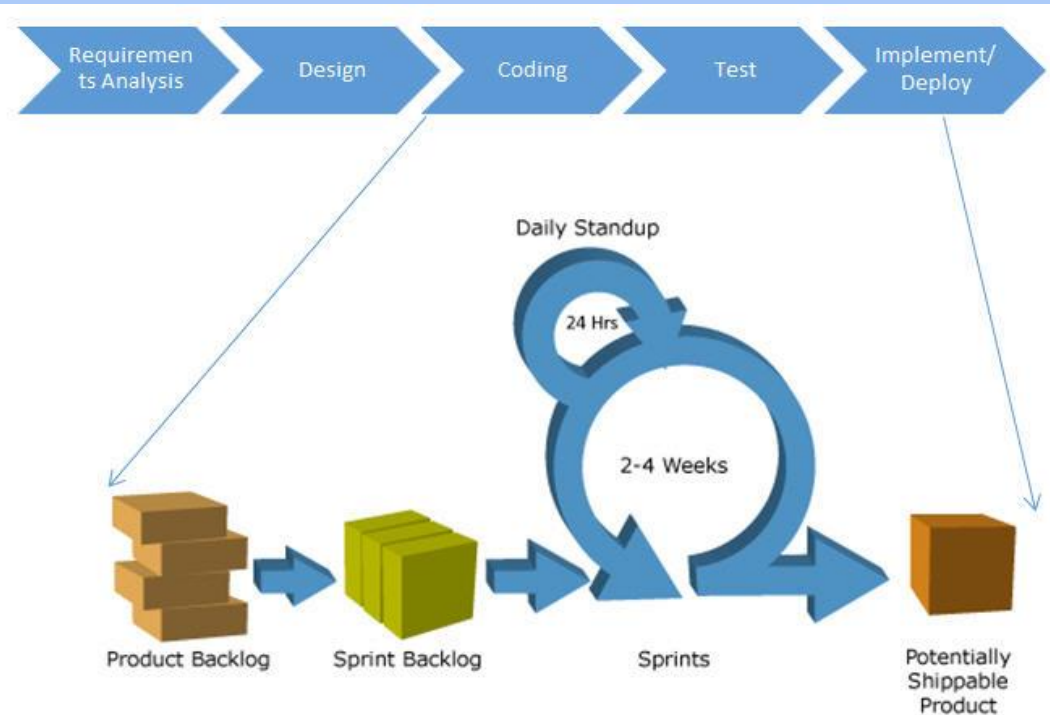
as per Waterfall Model

Coding Test and Deploy/Implement



as per Agile Model.

Collaborative (Hybrid) Model



Collaborative (Hybrid) Model



# Variations

- The collaborative model divided the phases of the SDLC into
  - Waterfall phases
  - Agile phases
- Other hybrid models blend the approaches
  - See <http://jackmyers.info/sweng/Agile-Waterfall-Hybrid-Template.pdf>
- Neither scrun.or nor ScrumAlliance believe in or support hybrid methodologies.
- For an interesting take on hybridization, see <https://www.scrumalliance.org/community/articles/2015/may/hybrid-agile-versus-agile>

# Larger Project can use Multi-team Scrum

## ■ *Role replication*

- Each team has a Product Owner for their work component and ScrumMaster.

## ■ *Product architects*

- Each team chooses a product architect and these architects collaborate to design and evolve the overall system architecture.

## ■ *Release alignment*

- The dates of product releases from each team are aligned so that a demonstrable and complete system is produced.

## ■ *Scrum of Scrums*

- There is a daily Scrum of Scrums where representatives from each team meet to discuss progress and plan work to be done.

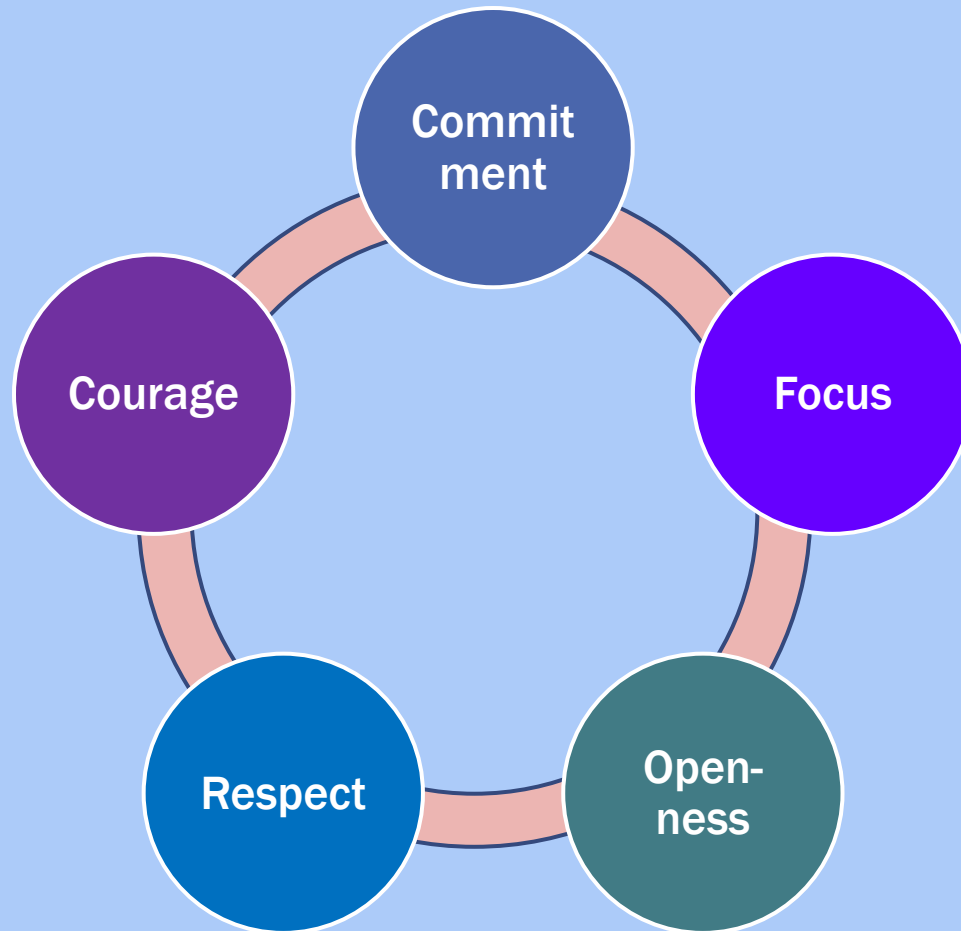
# It's all about the team



# Teamwork in Scrum

- The ‘Scrum master’ is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with customers and management outside of the team.
- The whole team attends short daily meetings (Scrums) where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day.
  - This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them.

# Scrum Values



# Scrum Values

| Value             | Misunderstanding                                                                        | Getting the value right                                                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Commitment</b> | Committing to something that you don't understand because you are told to by your boss. | Committing yourself to the team and Sprint Goal.                                                                                                            |
| <b>Focus</b>      | Focusing on keeping the customer happy                                                  | Being focused on the sprint and its goal.                                                                                                                   |
| <b>Openness</b>   | Telling everyone everything about all your work                                         | Highlighting when you have challenges and problems that are stopping you from success                                                                       |
| <b>Respect</b>    | Thinking you are helping the team by being a hero                                       | Helping people to learn the things that you are good at and not judging the things that others aren't good at.                                              |
| <b>Courage</b>    | Even after the decision has been made continuing to push back                           | Being transparent, but willing to change even if that means accepting that you are wrong, or that your opinion is not the direction that the team is going. |