Objectives

- Provide an understanding of what Agile is
- Understand the Agile Values & Principles and how they equate to a ‘Value Centre’
- Provide an introduction to the Scrum Framework
- Introduction to the Roles in Scrum
- Discuss Stories / Epics
- Introduction to the Meetings / Flow of Scrum
- Work together as a Team to create and document:
  - Definition of Ready
  - Definition of Done

Timings

- Start at 9.00 am each day
- Finish at 5pm
- Lunch 12pm – 1pm
- Breaks will be taken approximately every 90 minutes.

- Participants should be in the room on time and return from breaks accordingly.
- Cell phones should be switched off or kept in silent mode.
- Attention is required –
- All questions are good
- Participation is expected from participants…

scrum

Topics

- Principles
- Aspects
- Roles
- Terminology
- Sprint
- Stories and Epics
- Definitions
- Story Points and Velocity
- Backlogs
- The Board
- Sprint Flow
- Summary

- Test Driven Development
- Dev. Opps

What is Scrum

Many people think Scrum and Agile are the same thing; they are not.

Agile can be described as a Value Center – embracing the many frameworks (XP, Kanban etc.) that can be used to implement Agile ways of thinking - Scrum is just of them…

"Agile methods are a family of development processes, not a single approach to software development" – Wikipedia

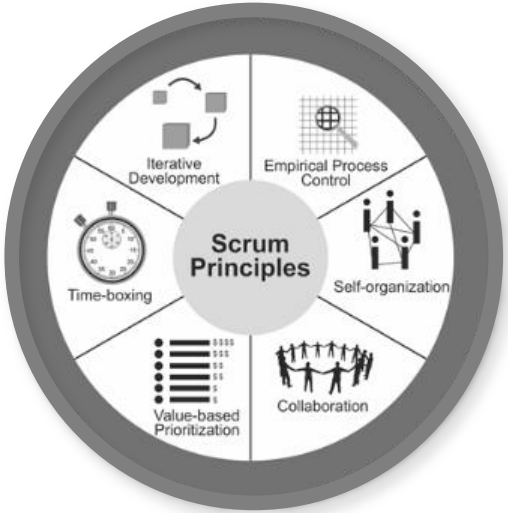
**Scrum is the most popular Framework– 72% of companies working with Agile, use Scrum.

principles

PRINCIPLES

Scrum is based on **6** core **principles**:

- **Iterative Development**
- **Empirical Process Control**
- **Self-organization**
- **Collaboration**
- **Value Based Prioritization**
- **Time-boxing**



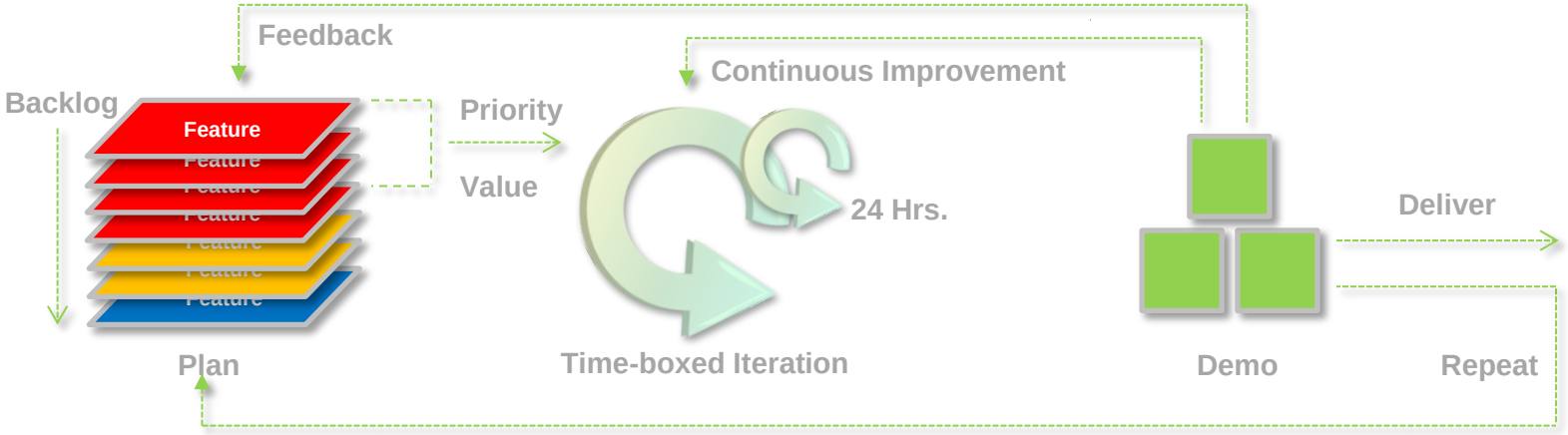
***NOTE**: Although many people think Scrum and Agile are the same thing; they are not. Agile can be described as a Value Center – embracing the many frameworks (XP, Kanban etc.) that can be used to implement Agile - Scrum is just one of them…*

****Scrum** is the most popular **Framework**– 72% of companies working with Agile, use Scrum.
** **Agile** is a **Value Center** and not a framework in it's self…

PRINCIPLES - Iterative Development

Agile methodologies are driven by the goal of delivering maximum **business value** in the least **time**. This is achieved by **iteratively** delivering high-priority / high value customer driven requirements quickly and often.

1. **Requirements** or **'Features'** are collated as '**User Stories'** and prioritized (top to bottom) in a '**Product Backlog**' The Backlog is refined throughout the project and is updated as requirements or priorities change.
2. **Priority** (highest **Business Value**) Stories are refined, planned and selected to be worked on. Priority is determined by the Customer and changes can be requested at any point, prioritized and queued to be worked on.
3. **Iterations** (or *Sprints)* are worked on based on the priorities set.
4. **Customers** are invited to view working software at the end of each Iteration in order to understand progress and to provide **Feedback**.
5. **Continuous Improvement** is achieved through inspecting and adapting processes at the end of each iteration, and the process **repeated**.

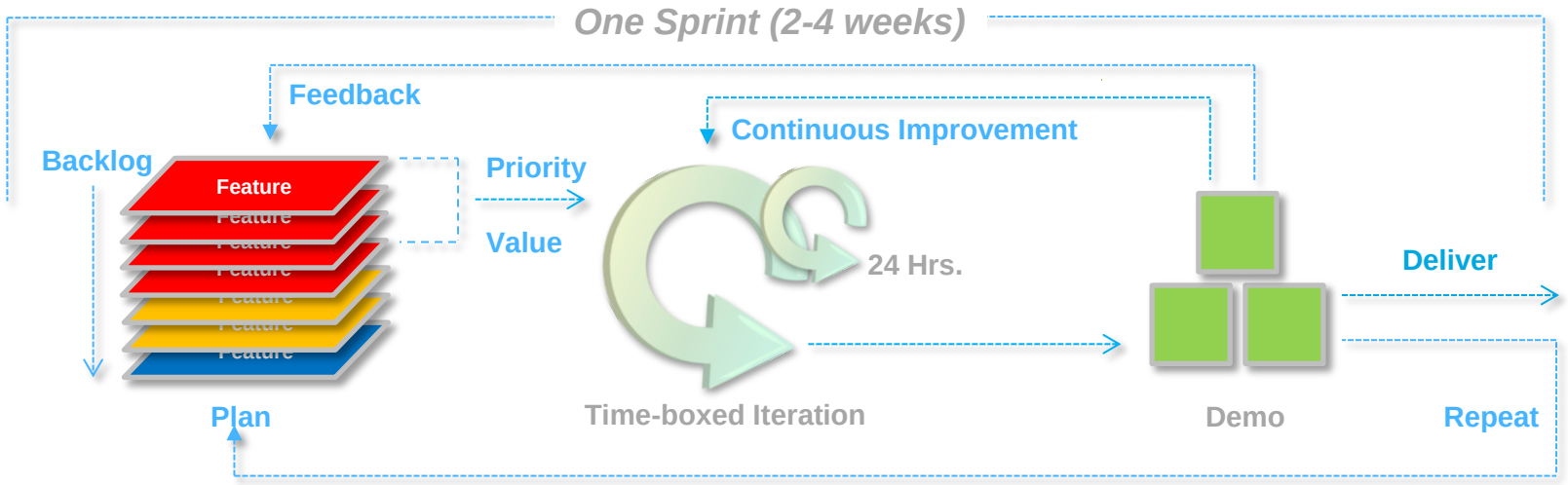


PRINCIPLES - Iterative Development

Maximum **business value** in the least **time**.

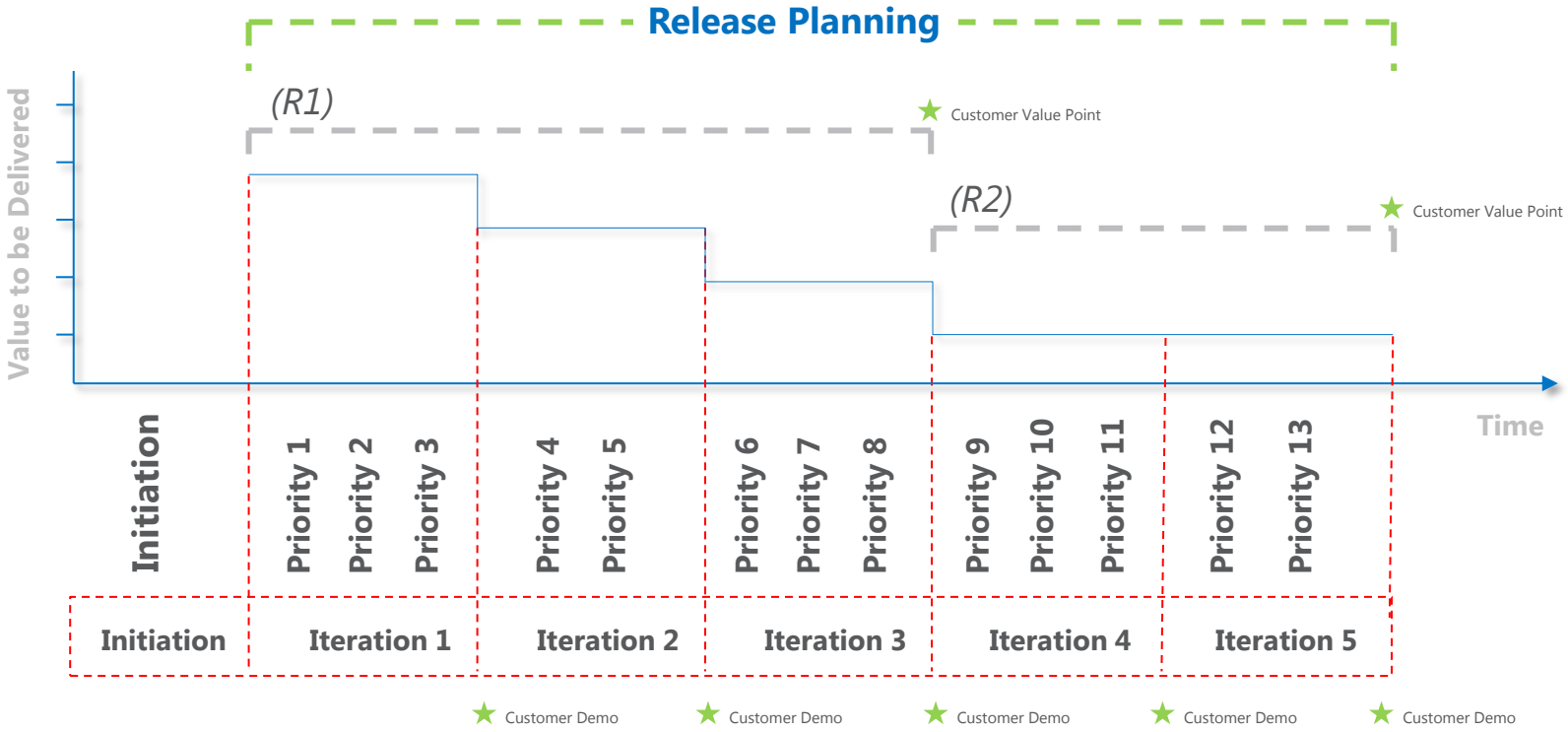
Some of the benefits of Iterative Development are:

- **'Fail Fast'** – continual review allows for reassessment of work without wasting large amounts of time
- **Changes** to requirements are better accommodated – they are requested throughout development and prioritized in the backlog
- **Information** is updated throughout the development and therefore remains current
- **Incremental** delivery, quicker releases of value to customers – most valuable features first. Not all at once.
- **Feedback** is continually gathered allowing process improvements to be recognised and implemented.



Agile – Iterative Development / Value Delivery

Agile methodologies are driven by the goal of delivering maximum **business value** in the least **time**. This is achieved by **iteratively** delivering **high-priority** / high value customer driven requirements as often as possible.



PRINCIPLES - Empirical Process Control - *decisions are made based on observation and experimentation…*

Transparency

Allows the Scrum process to be viewed openly by anyone:

Artifacts
- Project Vision Statement / Sprint Goal
- Prioritized Product Backlog
- Release Planning Schedule

Meetings
- Sprint Review Meetings
- Daily Standup Meetings

Visuals Information Indicators
- Burndown Chart
- Scrumboard

Inspection

Transparency allows for thoughtful inspection of processes and practices

- Use of a common Scrum-board and visual information indicators
- Collection of feedback from the customer and other stakeholders
- Review of deliverables by the Product Owner and the customer at Demo's

Adaptation

Through transparency and inspection the team can then adapt by making improvements in the work they are doing.

- Constant Risk Identification / User Stories / Sprints
- Daily Standup Meetings
- Retrospectives

PRINCIPLES - Self Organization

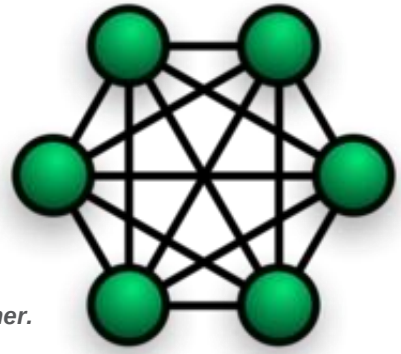
This principle reflects the management philosophy whereby:

Operational decisions are, as much as possible, made by those who have the most detailed knowledge of the consequences and practicalities associated with those decisions – **THE TEAM**

Some benefits of Self-organization are:

- **Involvement** and shared ownership
- **Responsibility** and accountability
- **Motivation**, which leads to an enhanced performance level of the team
- **Innovative** and creative environment conducive to growth

"Self Organization is only effective through strong teamwork. The team is accountable and has a responsibility to play an active role in ensuring the output and quality of work of the team."



NOTE: What to work on and the priority of that work is managed by the Product Owner with direct input from the customer.

PRINCIPLES - Collaboration

Through the team working together and interfacing with the stakeholders to create and validate the deliverables of the project; expectations are better managed, developed and delivered.

The core pillars of collaboration are:

Awareness — Individual team members working together need to be aware of each other's work.

Articulation — Collaboration involves dividing work into units, distributing the units among team members, and then after the work is done, re-integrating to the desired product.

Appropriation — Taking ideas and practices from different areas of the team to improve delivery of the common goal.

Some of the benefits of Collaboration are:

- Minimization of change requests – the customer is involved all along
- Risk Mitigation
- Increase in efficiency
- Continuous improvement
- Consistency of expectation and product delivery

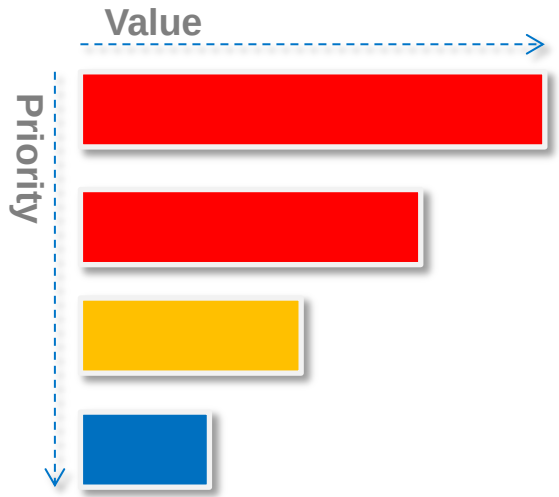
PRINCIPLES - Value Based Prioritisation

Prioritizing can be defined as determining the order and separating what must be **done now**, from what can be **done later** – prioritizing the backlog is an on-going exercise .

This core principle drives the entire Scrum framework - ***… satisfy the requirements of the customer with the objective of delivering the maximum business value in the least amount of time.***

Prioritization should be based on the following 3 factors:

1. **Value** to the customer
   - Must have
   - Should have
   - Could have
2. **Risk** and **Opportunity** Cost
3. **Dependencies** / Constraints



aspects

ASPECTS

In Scrum there are **5 key Aspects** which are maintained throughout the methodology:

1. **Organisation:** At the heart of Scrum is the organization. There are 3 key roles that each Scrum project must have fulfilled:

    - **Product Owner** – Business value, requirements, the voice of the customer, prioritization
    - **Scrum Team** – Converting requirements to tasks to value – implementing through Sprints
    - **Scrum Master** – The 'conductor' through iterations / Sprints, protecting and providing for the team

2. **Business Justification** – As with any process – The business must conduct relevant analysis for the need for a project output. Scrum focuses on ***Value-driven Delivery*** – delivering incremental value to the customer, often. Continual integration ensures the business need is continually assessed and met though Scrum's inherent ability to adapt.

3. **Quality:** Is defined as the ability of the completed product or deliverables to meet the **Acceptance Criteria** and achieve the business value expected by the customer. This is achieved through short sprints and the ability to quickly recognize issues and adapt - continually improve.

4. **Change:** Every project has Change. Scrum is geared to accept change and adapt through short sprints and **incremental delivery** – never getting too far down a path to be able to back out if needed.

5. **Risk:** As with any project and methodology, risk must be proactively assessed and mitigated by a robust risk management process encompassing **all Sprints**, throughout the entire project life-cycle – each Story is assessed for risk.

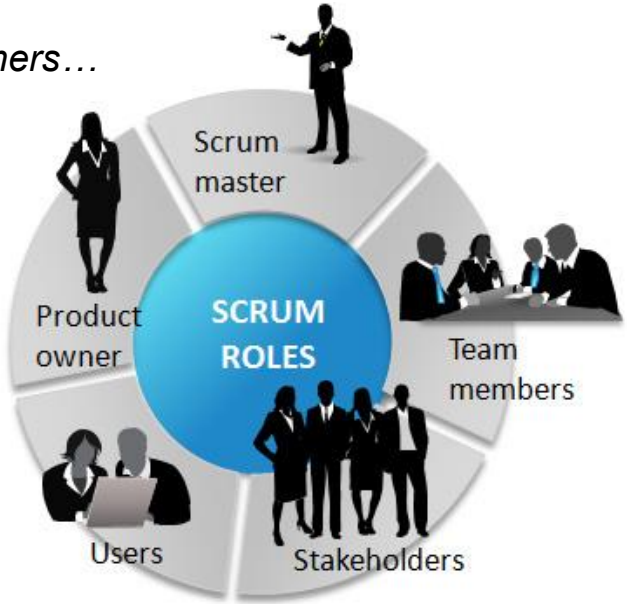
roles

ROLES

Three essential (mandated) roles in a Scrum Team:

A scrum team has a slightly different composition than a traditional waterfall project, with three mandated roles as well as the User (customer)

1. **Product Owner** - *The voice of the customer*
2. **Scrum Team** – *Cross-functional - Developers, Testers, Designers…*
3. **Scrum Master** – *Servant-Leader to the team*
4. **Users** – *End Users of the System*
5. **Stakeholders** – *Interested parties*

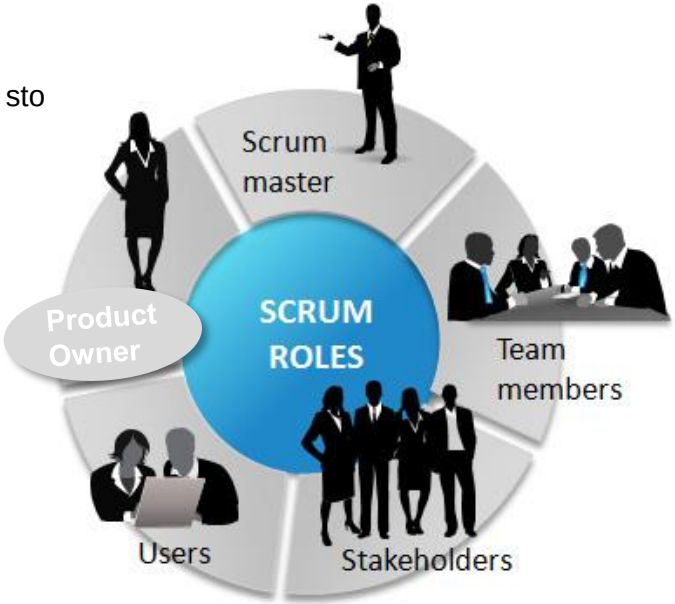


ROLES

The Product Owner

The responsibilities of the Product Owner are key to the success of any Agile team. The right Product Owner can make the difference for the entire project / product.

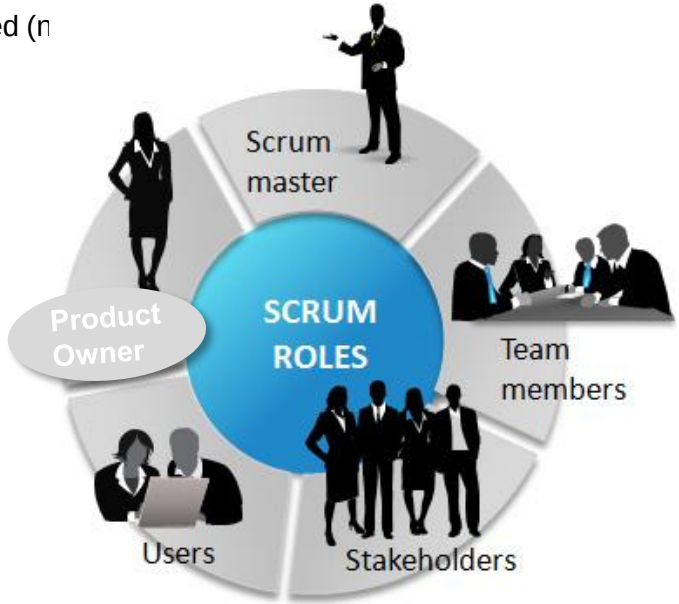
- Creates and maintains the Product Backlog.
- Prioritizes and sequences the Backlog according to business value.
- Works with the customer and the team to elaborate Epics and Features into user sto granular enough to be achieved in a single sprint.
- Conveys the Vision and Goals at the beginning of every Release and Sprint.
- Represents the customer, interfaces and engages the customer with the team.
- Participates in the daily Stand-ups, Sprint Planning Meetings and Sprint Reviews and Retrospectives if required.



ROLES

The Product Owner cont…

- Inspects the product progress through PO Verification of each story and has complete authority to accept or reject work done.
- Can change the course of the project at the end of every Sprint (… or during Sprints in very rare circumstances).
- Communicates status externally to Customers and Stakeholders.
- Terminates a Sprint if it is determined that a drastic change in direction is required (n



Scrum Events

1. Backlog Refinement– Leads
2. Sprint Planning 1 & 2 - Attends
3. Daily Stand-ups – Attends (where possible)
4. Sprint Review/Demo - Leads
5. Retrospectives – Attends where required

ROLES

The Scrum Master

The Scrum Master is responsible for ensuring Scrum is understood and enacted. Scrum Masters do this by ensuring that the Scrum Team adheres to Scrum theory, practices, and rules.

- Champions Velocity
- Shields the team from interruptions during the sprint
- Helps the team maintain their Burn-down chart/s
- Setting up retrospectives, sprint reviews or sprint planning sessions
- Encouraging collaboration between the Scrum team and Product Owner

ROLES

The Scrum Master cont.

The Scrum Master is responsible for ensuring Scrum is understood and enacted. Scrum Masters do this by ensuring that the Scrum Team adheres to Scrum theory, practices, and rules.

- Removing obstacles that affect the team
- Protect the Sprint Backlog – defend scope creep
- Facilitating (not participating in) the daily standup

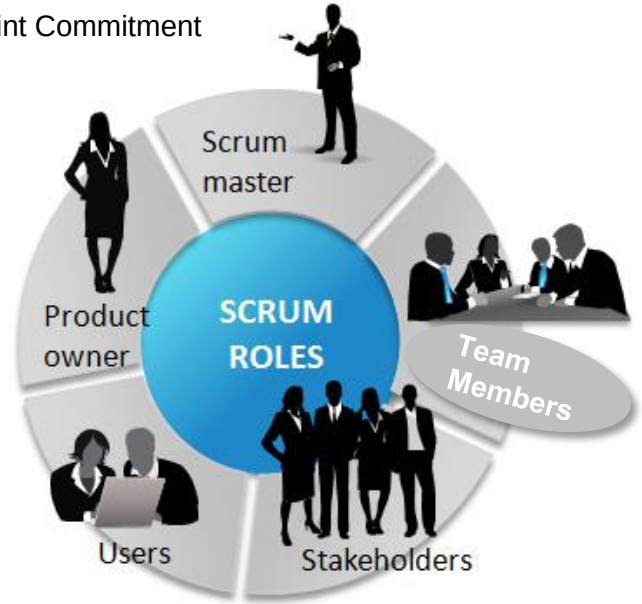
Scrum Events:

1. Backlog Refinement - Attends
2. Sprint Planning 1 & 2 - Leads
3. Daily Standup - Leads
4. Sprint Review/Product Demo - Attends
5. Sprint Retrospective - Leads

ROLES

The Scrum Team

- A cross-functional team including: Developers/Tester/s/DBA/Designers etc. required to deliver Product increments
- Typically 5 to 9 in size,
- Collaborates with the PO to ensure Stories are developed (‘Ready’) and have clear Acceptance Criteria
- Drives the plan for each sprint by managing the amount of work using Velocity / Sprint Commitment
- Communicate frequently and openly.
- Ideally the team remains persistent and is co-located to facilitate daily interaction, where not possible, adequate communication mechanisms need to be established
- Self Organising to deliver product implements



Scrum Events

1. Backlog Refinement– Attends
2. Sprint Planning 1 & 2 - Attends
3. Daily Stand-ups – Attends
4. Sprint Review/Demo - Presents
5. Retrospectives – Attends

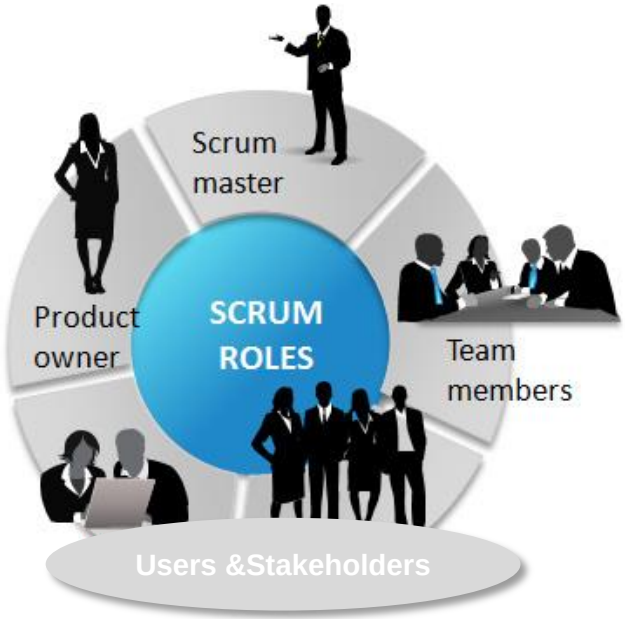
ROLES

Users:

- The Users (customers) are generally the group or individual who request the features or system
- They are crucial to the ongoing development and acceptance of the development
- Work with the PO to request, write and build user stories
- Work with the PO to set Backlog Priority
- Participate in the Sprint Demo to review completed work

Stakeholders

- Anyone with a direct or indirect interest in the product or the work being done



Scrum Events:

1. Backlog Refinement – As Invited
2. Sprint Planning 1 & 2 – No Attendance
3. Daily Standup - Leads - As Invited
4. Sprint Review/Product Demo - Attends
5. Sprint Retrospective - No Attendance

terminology

TERMINOLOGY

Sprint - this is a time-boxed iteration usually 2 – 4 weeks in duration during which the Scrum Team works on and creates the Sprint deliverables. The Team and the Scrum Master self-organize around meeting the committed stories for the Sprint

User Stories - are the heart of Scrum and are what is delivered in Sprints. These are self-contained requirements / work that have been collaboratively agreed on by stakeholders, the Product Owner and the Team.

Epics - are made up of multiple stories and can often initially appear like LARGE user stories.

Product Backlog - is the *prioritized* list of all features and requirements for the product / project expressed in User Stories and Epics. The Product Owner manages the backlog.

The Sprint Backlog - is the list of Stories to be worked on in the Sprint. Stories included in a compiled Sprint Backlog are complete and Ready to be worked on by the team. This prioritized by the Product Owner and is managed by the Scrum Master and the Team

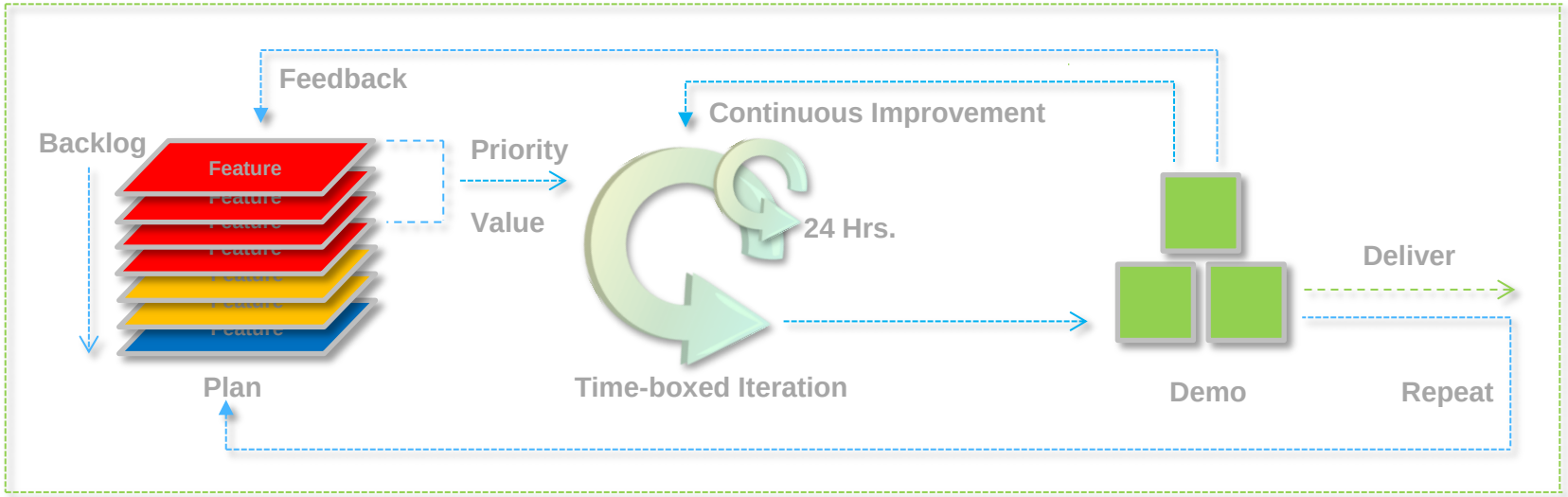
Story Points – An arbitrary measure used by the team to measure the complexity of a story and the effort required to implement. The team determines the Story Point allocation through the Backlog Refinement and Sprint Planning meetings.

Velocity – Is defined by the average number of Story Points delivered over the past 3 Sprints. This number reflects the amount of work the team can mange in an Sprint. Sprint commitment should be based on this number.


sprint

SPRINT

The Sprint is the heart of Scrum, it contains all the elements and ceremonies of the Scrum process.



1. Backlog Refinement
2. Sprint Planning
3. Daily Stand Up
4. Demo
5. Retrospective

SPRINT

- **Delivers on a Sprint Goal**.

    - A *potentially* shippable increment of software and/or
    - a useable iteration of a hardware product and/or
    - a ‘Design Spike’ that facilitates future delivery that directly produces customer value.

- **Gives the Team regular, high quality feedback on delivered value**. The Team can then inspect and adapt both their process and their product based on real and actionable customer input.

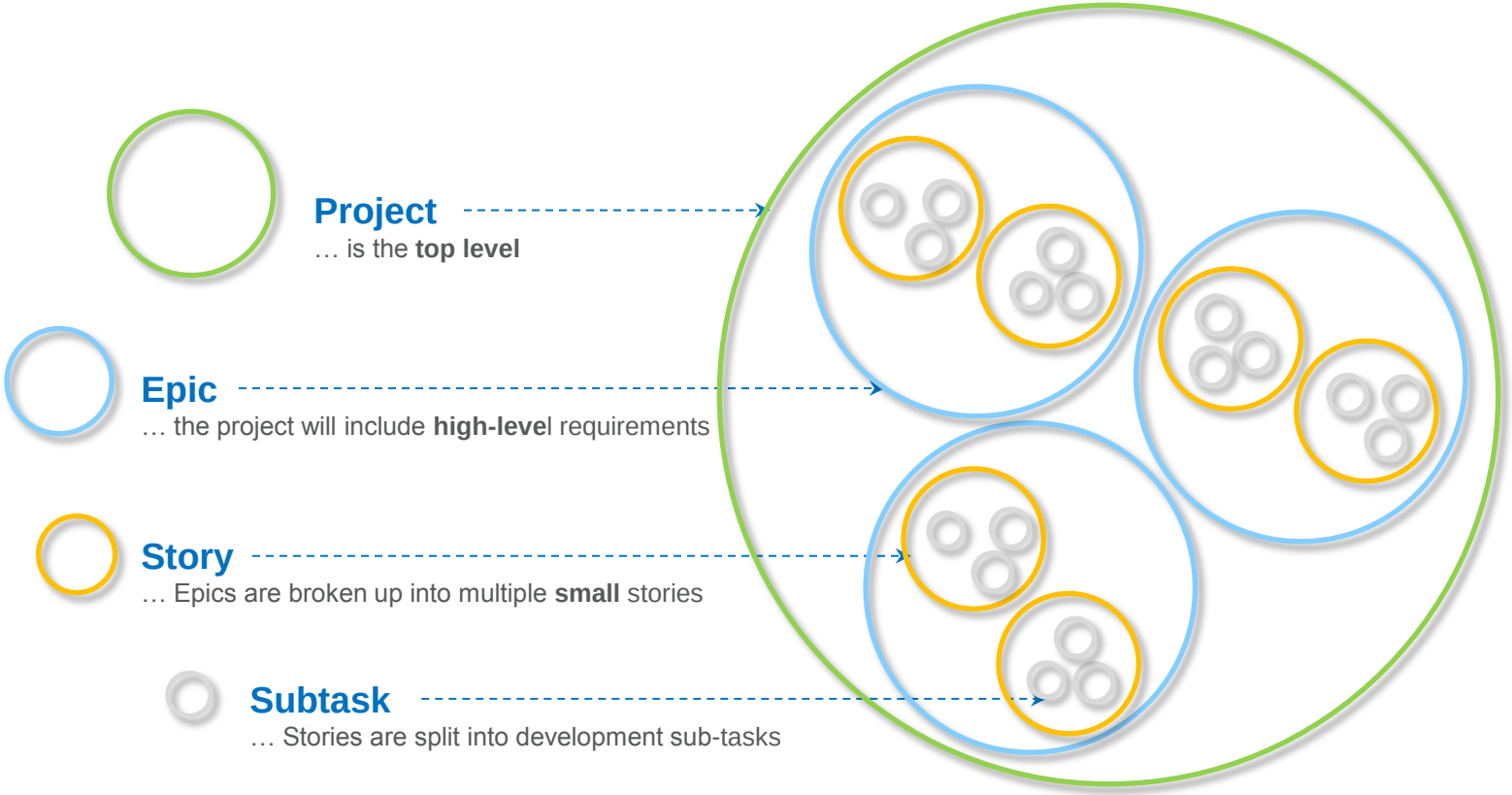
- **Measures Team output** over a consistent and recurring period of time which facilitates planning. This is called the Team's ’**velocity’**.

Key Governance

- Repeatable, ‘time boxed’ (typically 2-4 weeks)
- Once the Sprint Starts, the scope should not change – the Scrum Master protects the Sprint
- Product Owner is available to answer any questions that come up, or find the answers
- Only the Product Owner can terminate the Sprint (very rare)
- Sprint commitment is a *Forecast*… not a *Guarantee*

stories & epics

STORY HIERARCHTY



Project
… is the **top level**

Epic
… the project will include **high-level** requirements

Story
… Epics are broken up into multiple **small** stories

Subtask
… Stories are split into development sub-tasks

USER STORIES

1. *A short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.*

2. *Through conversation and collaboration between the Customer, Product Owner and the Team, Stories are embellished with just enough detail to facilitate being developed and tested.*

3. *Prior to be included in a Sprint, Stories can change as more information is understood through the conversations noted above – incomplete stories do not get included in a Sprint – only Stories that are 'Ready' are included.*

4. *Small enough to be able to be completed within a Sprint.*

USER STORIES

Anatomy of a Story

All stories should contain the following…

Summary: A short, few word overview, like a ***title***. This helps when Stakeholders are viewing the Backlog, they can quickly see what is being worked on at a high level.

Description: Going into more detail, using the following structure:

As a ***'user'*** I want ***'do something'*** so that I can ***'achieve something'***.

1. **User** – a specific user of the product – **who** we are building for – i.e. ***Customer***
2. **Do Something**– the feature user wants – the **intention** of the delivery – i.e. ***login***
3. **Achieve Something** – why the feature is important – **business value**– i.e. ***So I can access site content***

Note: *Use this structure when it is helpful, different scenarios may require a different structure – experiment and find what is best for the team at different times – i.e. a Bug story may be best described slightly differently.*

Acceptance Criteria: *"the conditions that a software product must satisfy to be accepted by a user, customer, or in the case of system level functionality, the consuming system."*

Adding Acceptance Criteria completes the story and helps the Development team plan what they are going to do to deliver the customers expectations for the story. It also makes it **testable** and ensures it can be finished and considered '**Done**'

***NOTE:** Acceptance Criteria should be written **before** starting work on the story – the helps to ensure the criteria is based on providing the customer what they require and expect rather than being a retrospective list of what has been developed.*

USER STORIES

A ‘good’ User Story

Will typically include the following:

- A description and summary,
- Defines the acceptance criteria,
- Able to be delivered within a single sprint,
- All (probable) dependencies identified,
- Required performance criteria / Non Functional Requirements,
- Has been ‘*estimated*’ by the team
- Meets the agreed ‘*Definition of Ready*’

USER STORIES

I N V E S T

To ensure the quality of Stories in the Backlog, the above acronym (*INVEST*) is a good way of reviewing and confirming they are '*READY*:

- **I**ndependent - Of all others, not tightly coupled to another story – if so, can the stories be combined?
- **N**egotiable - Not a specific 'contract' for features – the story may be rewritten once discussed with the team
- **V**aluable – The story must bring specific project value to the stakeholder
- **E**stimable – The story must be estimable to a fairly good approximation so it can be planned, there must therefore be enough information contained to facilitate this.
- **S**mall - A good rule of thumb is that Stories do not take more than 50% of an iteration – anything larger should look to be grouped as an Epic and disaggregated accordingly
- **T**estable – If a story cannot be tested it will never get to 'DONE'. Ensure there is enough information in the story to be able to write and run a successful test. As with Estimation above; without ample information, the story cannot be planned and completed.

WRITING STORIES / DISAGGREGATION

Break it down

1. An **Epic** is a ‘headline’ or ‘holder’ - a **large story** which is broken down to smaller, more manageable stories to be developed through Sprints.

‘Members can book a flight and pay for it’

2. The next step is to **disaggregate** into more granular Story’s that make up the component parts of the desired Feature / Epic…

a) As a customer I want to **login** to the site so I can access the booking screen

b) As a customer I want to **select a date**, or date range so I can see relevant flights

c) As a customer I want to be able to **specify journey** parameters so I can select One Way or Return

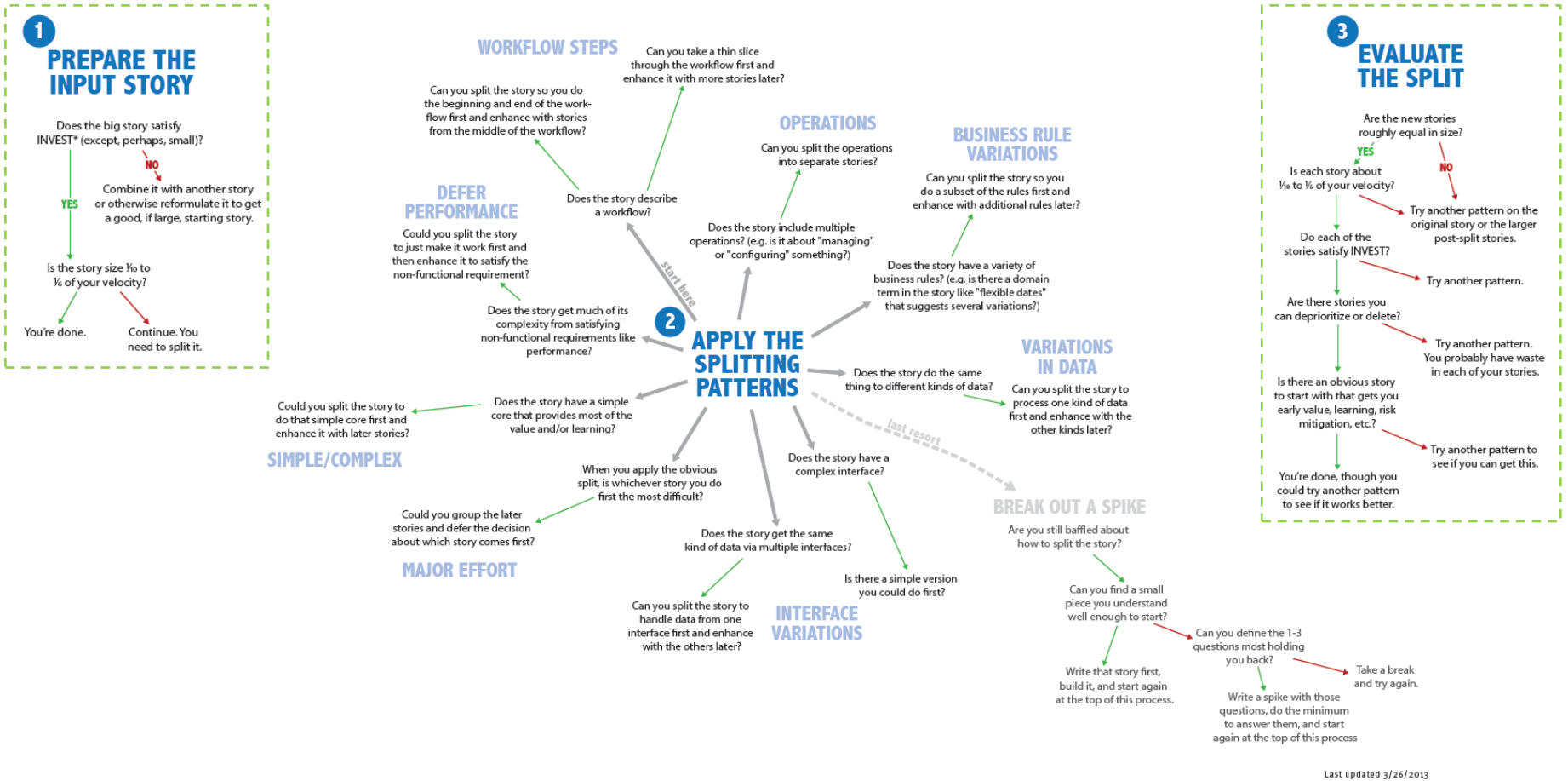
d) As a customer I want to be able to **select the class** of travel so I only see relevant results after searching

e) As a customer I want to be presented with a **list of possible flights** so I can select my preferred booking

f) As a customer I want to use my credit card so I can pay for my ticket

NOTE: *The process of breaking down Epics can often unearth new Epics i.e. establishing the payment gateway etc. for ‘story f’ is large enough to be considered an Epic and broken down into further stories as per the above process.*

WRITING STORIES / DISAGGREGATION



Last updated 3/26/2013

WRITING STORIES / DISAGGREGATION

3. Acceptance Criteria

- ***Given*** *some precondition* ***When*** *I do some action* ***Then*** *I expect some result*

a) As a customer I want to login to the site so I can access the booking screen

- Able to enter valid Username and Password
- Upper, Lowercase, Numbers and Special Characters OK
- Minimum 6 characters – show error message if fewer
- Able to Click 'Enter' button
  - Credentials OK – go to booking screen
  - Credentials FAILED – present Failed Message
  - After 3 wrong attempts - present locked out message
- Validation response less than 1 second.
- *Etc. as needed…*

WRITING STORIES / DISAGGREGATION

4. Once the Stories are created, the next step is for the Developers to **break down** the Stories into the **Sub-tasks** (sT) that will deliver the story through a Sprint…

a) As a customer I want to login to the site so I can access the booking screen

sT Enter Username and Password
Create input field for Username and Password to be authenticated through Active Directory

sT Click OK / Authenticate / Enter to Flight-Search page
- Create OK button, when clicked submit credentials to Active Directory
- Receive message form Active Directory authenticating credentials
  - Positive Response – Open page

sT Negative response pop-up –
Receive negative response from Active Directory,
Generate error pop-up screen, inform user of the error. "The credentials supplied…."
Provide a link to start again, or to '***reset the password…'***

NOTE: *Breaking down the stories to sub-tasks can bring to light tasks that in themselves may be stories i.e. 'resetting the password' would be significant enough of a flow to create a new story or even Epic and repeat the above disaggregation process…*

USER STORIES

Collaborate & Elaborate

The **Product Owner** will discuss and refine all stories with the **Customer / SME and Team** to get them '*Ready*' for inclusion in a Sprint. The Product Owner manages the backlog and stories - the Customer and the Team help to elaborate where required.

The Discussion and Collaboration are integral to writing good, usable stories.

Product Owner* + *Customer* + *Team* = **High Quality Stories.

definitions

DEFINITIONS

The Definition of *READY*

Ensures the team can work productively, effectively and be able produce the desired features of a Story.

Each prospective Story should be assessed against the following *(below is only an example, the Definition of Ready should be tailored by the team and agreed):*

1. The **Business value** is clearly articulated in the story.
2. Details are sufficiently **understood by the developer / team** so it can make an informed decision as to whether it can complete the story.
3. All **Dependencies** are identified and no external dependencies would block the story from being completed
4. The story is estimated and small enough to comfortably be **completed in one sprint**.
5. The **Acceptance Criteria** are clear and testable.
6. NFR (Non Functional Requirements), if any, are defined and testable.
7. Scrum team understands how to **demonstrate the story** to the customer at the sprint review
8. …

Ensuring the Definition of ***Ready*** is met, prior to accepting the Story into a Sprint, facilitates delivering the desired outcome at the end of the Sprint – meeting the 'Sprint Commitment'.

***NOTE**: Different kinds of stories may require different definitions, i.e. a Design Spike or Bug story.*

The Definition of Ready will be agreed with the team

DEFINITIONS

The Definition of *DONE*

*Ensures that DONE means DONE. For a story to be considered **complete**, basic criteria or guidelines are set and each story is reviewed against the criteria.*

1. Development Complete
2. All Acceptance Criteria satisfied
3. All test Data available / Dev test Complete
4. No (known) bugs
5. Unit Tests written / Run
6. Successfully QA tested
7. Code peer reviewed
8. PO Verified
9. Relevant documentation updated
10. ….

**NOTE:* *Different types of stories will require different definitions i.e. a Design Spike or Technical story will include different items.*

Above is an example, The Definition of Done will be agreed with the team

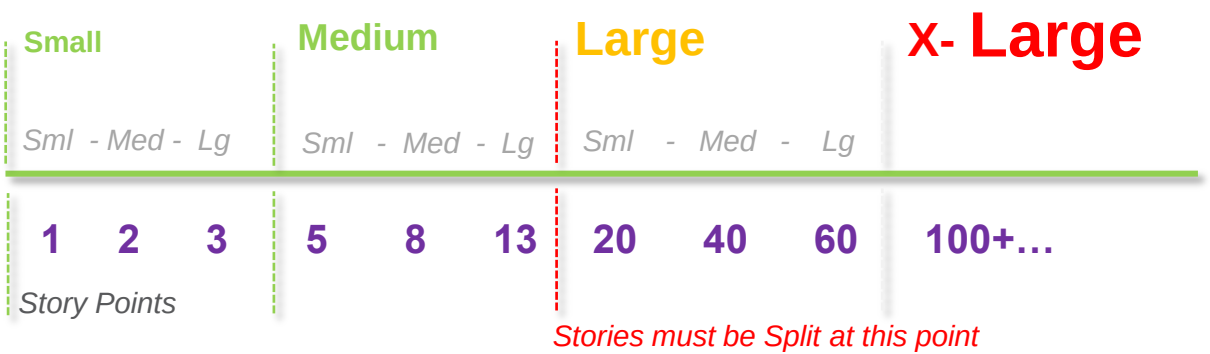
story points & velocity

STORY POINTS

How ‘complex’ is a Story…

A story can be thought of in terms of complexity to implement (i.e. Small, Med, Large etc.)

This is done to assist in prioritization and to help determine how many Stories can be worked on in a Sprint.



Based on the Fibonacci sequence of numbers, a number value is appointed to each ‘Size’. This is not a measure of time, it must be a measure of complexity.

STORY POINT ESTIMATION

Planning Poker

A method of estimation.

A User Story is presented to the team:

'As a customer I want to login to the site so I can book a ticket'

Each team member then turns over a card (at the same time), showing the Story Points they feel the story warrants in terms of complexity to deliver.

Team Member A Team Member B Team Member C

STORY POINT ESTIMATION

Planning Poker

A method of estimation.

A User Story is presented to the team:

'As a customer I want to login to the site so I can book a ticket'

Each team member then turns over a card (at the same time), showing the Story Points they feel the story warrants in terms of complexity to deliver.

Team Member A

Team Member C

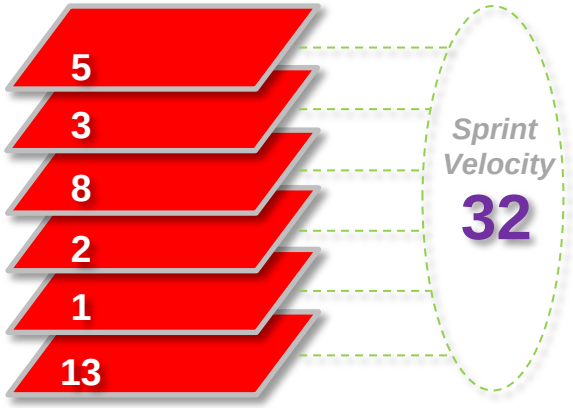
Facilitated by the Scrum Master - the highest and lowest numbers are discussed. This ensures understanding of the estimate, making sure nothing is overlooked from each team members perspective i.e. one estimate may take into account a better way to implement, or another may not have realized some complexity.

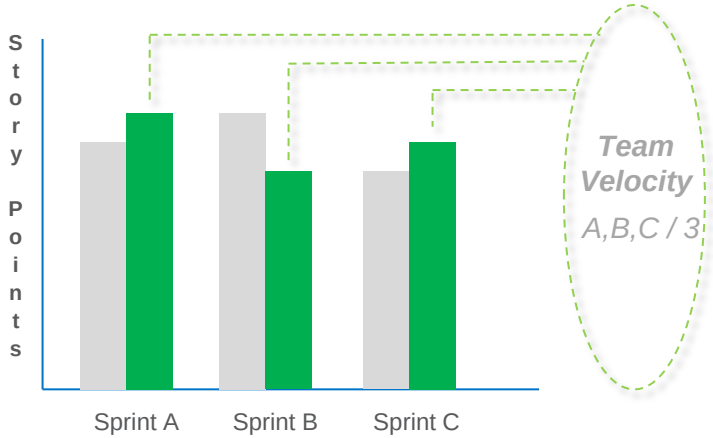
Estimation is carried out initially at a high-level, and then repeated once the story is ''Ready' and to be added to a Sprint.

VELOCITY

… how many Stories can be completed in a Sprint.

- During Backlog Refinement and Sprint Planning, Stories are estimated using Story Points
- The combined number of points to be added to a Sprint is called **Sprint Velocity**.
- The average number of Story Points *delivered* over the last 3 completed Sprints, represents the **Team Velocity**.
- When planning Sprints, it is the **Team Velocity that** sets the boundaries for **Sprint Capacity Planning** – the guideline for how many Stories are included in a Sprint.
- The Team Velocity assists in **Release Planning** to give a high-level estimate on how many Sprints a Release may take
- Metrics are also captured around Planned vs Actual (PVA) Story Points / Velocity.





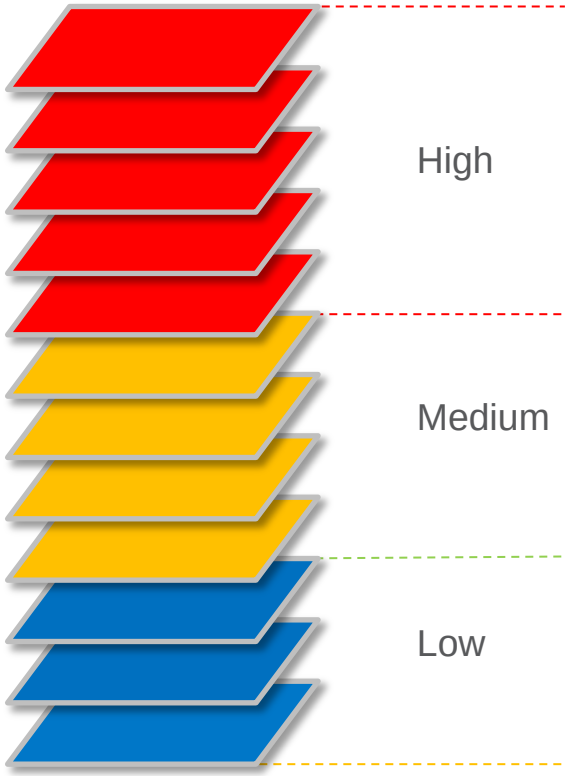
backlogs

BACKLOGS

The Product Backlog –

The single source of all features and requirements to be developed for the product or project expressed as prioritized User Stories and Epics.

- Stories are prioritized - the top (high priority) stories have more information and are generally more complete and ready to be included in a Sprint.
- Time is invested in Stories proportionate to priority – i.e. more at the top, less at the bottom.
- New requirements, captured as the project progresses are added and prioritised as needed.
- If it is not in the Product Backlog – it does not exist and wont get developed



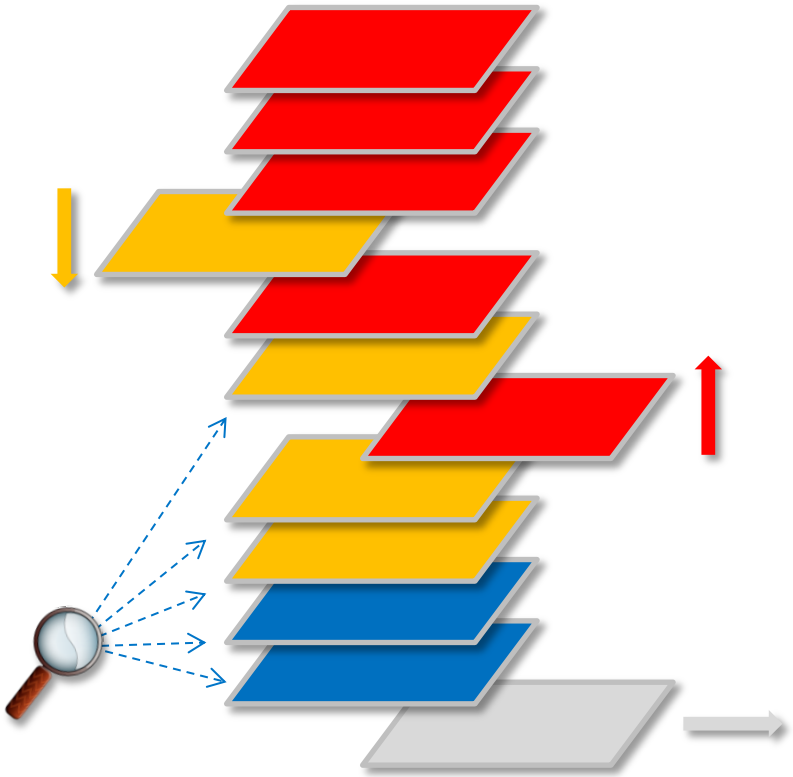
Responsibility – Product Owner
Assists – Customer, Team

BACKLOGS

Backlog Management.

The backlog is continually reviewed and managed by the Product Owner, focusing on the following key areas of Backlog Management:

- **Refine –** Stories are refined and clarifications are continuously updated in the stories . It is considered good practice to have 2 Sprints worth of stories refined and '*Ready*' in the Backlog
- **Reprioritise** - As Customer needs change, story priorities also change in the backlog.
- **Remove –** Any stories that are no longer needed, or have become obsolete as requirements change, are removed to keep the backlog 'Lean'.



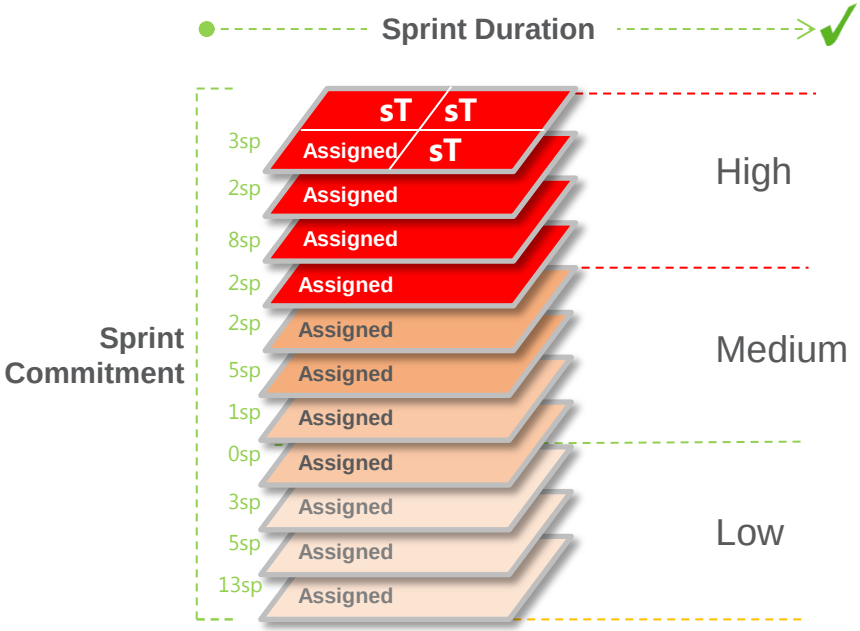
Responsibility – Product Owner
Assists – Customer, Team

BACKLOGS

Sprint Backlog

The list of features selected by the team to be worked on and delivered in an iteration (Sprint) – the Sprint Commitment

- Ordered in priority – top to bottom
- Each story has been estimated and the work planned
- Each ticket should show who is assigned to the task one the work is underway
- Forms the focus for the team during daily Stand-Up meetings.
- Each ticket will meet the Definition of Done within the Sprint

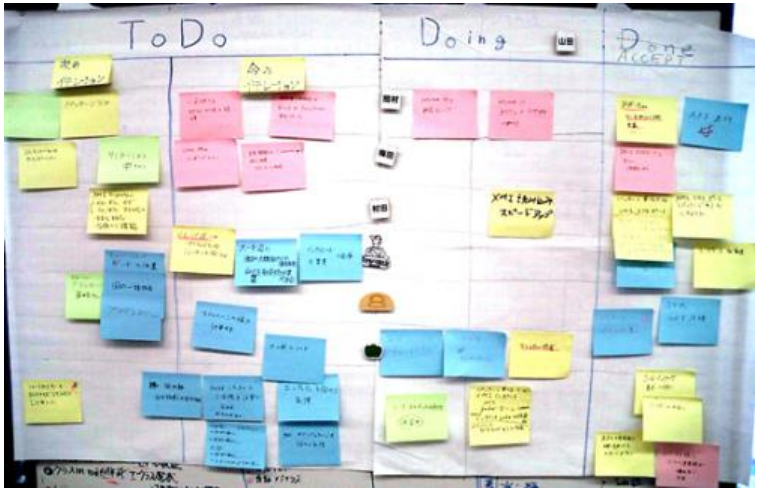


Responsibility – Team & Scrum Master
Assists – Product Owner

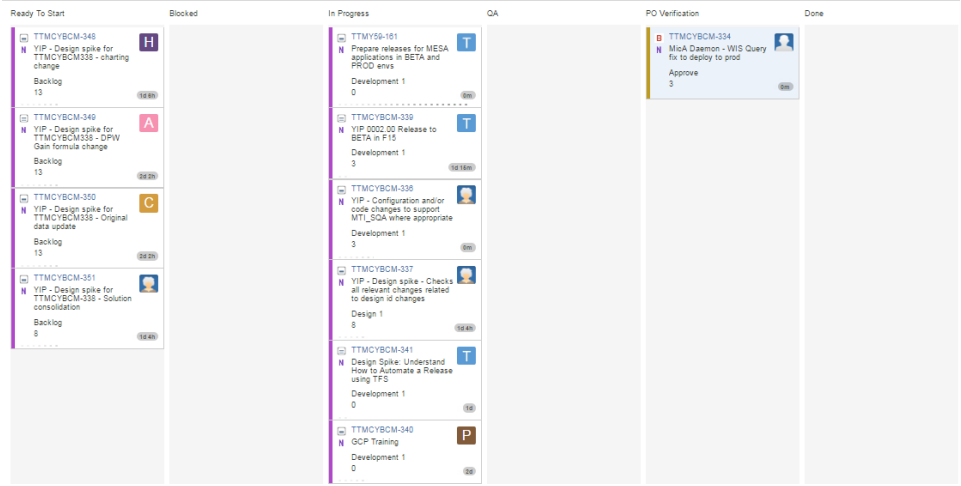
the board

THE BOARD

The Board can be either a Physical Board using Post-it notes to represent the tickets / Stories.



…Or a Virtual Board, using an online tool such as JIRA (which is what Micron uses)

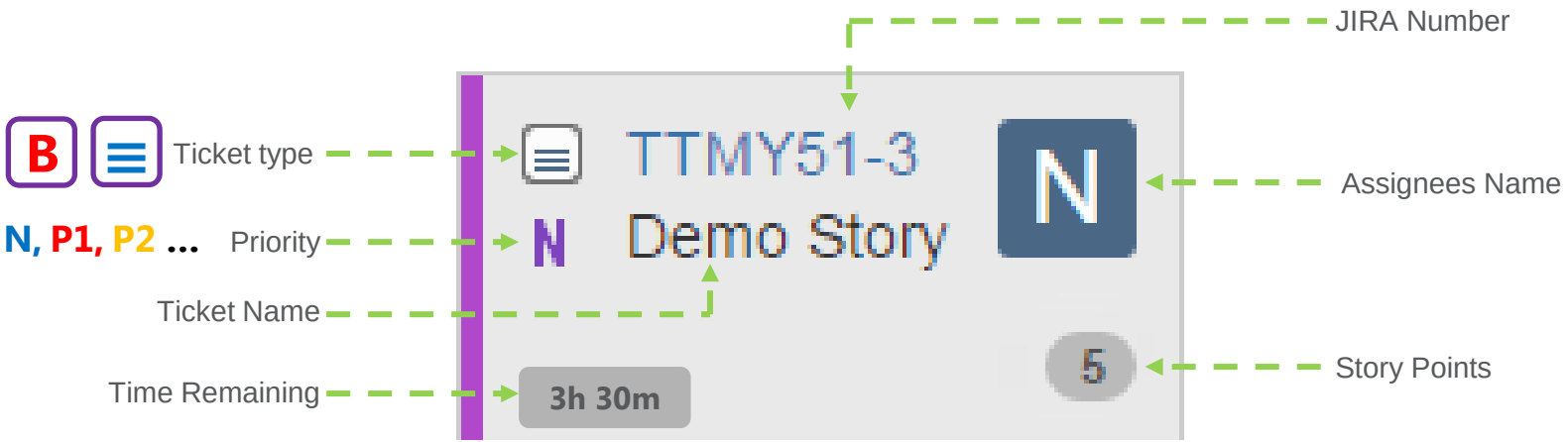


The Board is a key part of Scrum …or any Agile methodology.

- Each **'Ticket**' moves through the different '**statuses**' from Ready to Done during the Sprint
- This is the 'window' to **ALL** the work being undertaken in a Sprint, providing **transparency** to how the Sprint is progressing
- Provides the **focus point** for the team in the daily stand-up meetings (Scrum)
- Highlights when **impediments** affecting a story may prevent it from being completed (allowing for timely corrective action to be taken)

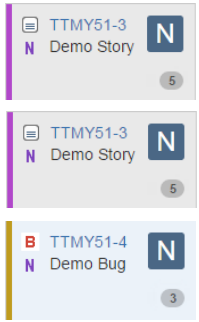
THE BOARD

‘Anatomy of JIRA Tickets’ - Tickets represent ‘tasks’ that are worked on during a Sprint



THE BOARD

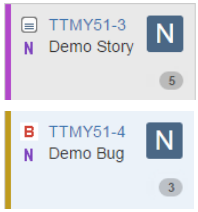
To Do



This is the committed & prioritised Sprint Backlog. All stories here should have met the **Definition of Ready**.

Stories remain here until the team member is ready to start the work.

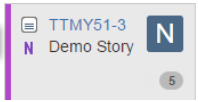
In Progress



Stories being worked on by a team member, currently under way and have no issues blocking completion of the story to pass to **QA**.

Should stories fail **QA** or **PO Verification**, they will be moved back **to In Progress** for resolution of the issue reported by the original assignee.

Blocked / Pending

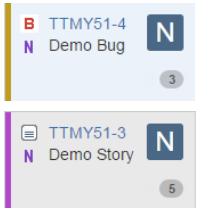


Any work that has been moved to **In Progress, QA or PO Verification** but has subsequently run into difficulties and is now **Blocked** from being able to progress through its current state (column) and moved to the next relevant state.

This is an '*exceptional*' state and only used to highlight issues for resolution.

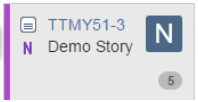
Once unblocked, the task is moved back to the relevant state to progress through the Workflow.

QA



Stories that have finished development and unit testing and are ready for **QA** and verification.

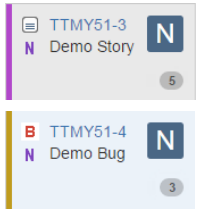
PO Verification



Stories that are complete – developed and passed **QA** – are reviewed by the PO .

The PO reviews the story against the **Definition of Done** and either moves the story to **DONE**, or seeks further clarification from the team.

DONE

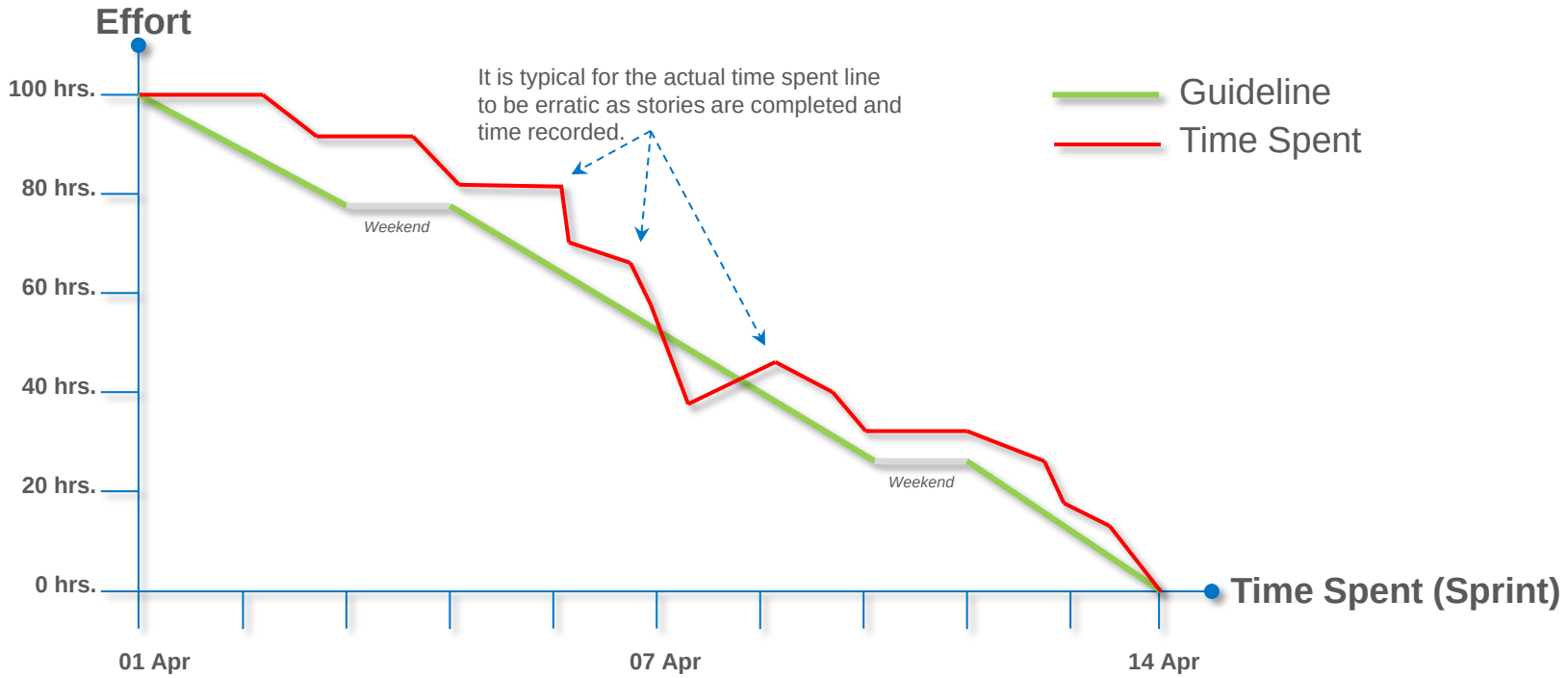


Stories that have completed all previous states and meet the **Definition of Done** and are ready to be showcased in the **Demo**.

Sprint Duration

SPRINT BURN DOWN CHART

The Burn Down chart shows the progress of the Sprint in terms of effort committed and time spent. This graph is used as an estimation for guidance purpose.



meetings & sprint flow

SPRINT FLOW

At the heart of **Scrum** are **5** main **Meetings** (often referred to as Ceremonies):

1. Backlog Refinement (1 / 2 meetings per Sprint)
2. Sprint Planning
    a) Part 1
    b) Part 2
3. Daily Stand Up (Scrum)
4. Sprint Demo
5. Sprint Retrospective

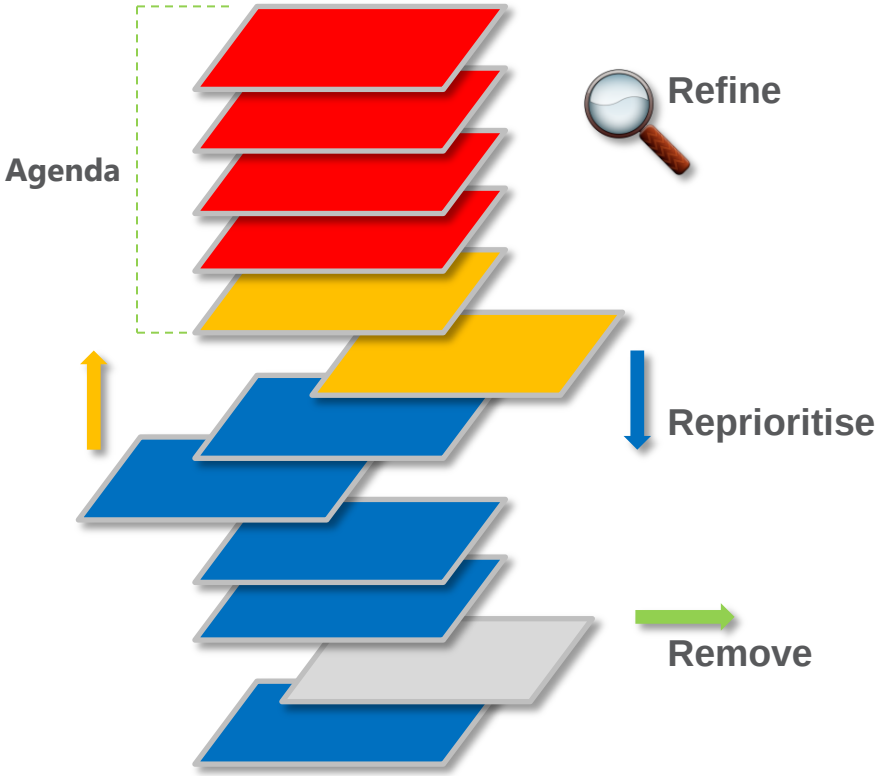
NOTE: *Technically speaking, Scrum only prescribes 4 ceremonies (2, 3, 4 & 5 above). However, it is now generally accepted that there is a need for a 5th meeting – Backlog Refinement.*

SPRINT FLOW

The Sprint ‘flow’ is the repeatable process and meetings that facilitate having a well managed Product & Sprint Backlog which ensures productive Sprints leading to rapid delivery of value to customers.

1. **Backlog Refinement** – The practice of ensuring the backlog is prepared and ‘Ready’ for inclusion in a Sprint

SPRINT FLOW – BACKLOG REFINEMENT



The backlog is continually updated and refined (‘groomed’) – always from the top down.

- The Product Owner prepares for the meeting by ensuring the Product Backlog is effectively prioritised
- The highest priority Stories form the agenda for the time-boxed meeting - Stories are then presented to the team for review
- The team asks questions regarding details to facilitate development
- The Product Owner collates questions and finds answers and updates the Stories after the meeting

It is good practice to have (roughly) 2 Sprints worth of Stories ready – This also ensures ‘ready’ stories are in the backlog should the opportunity arise to pull a story into the active sprint.

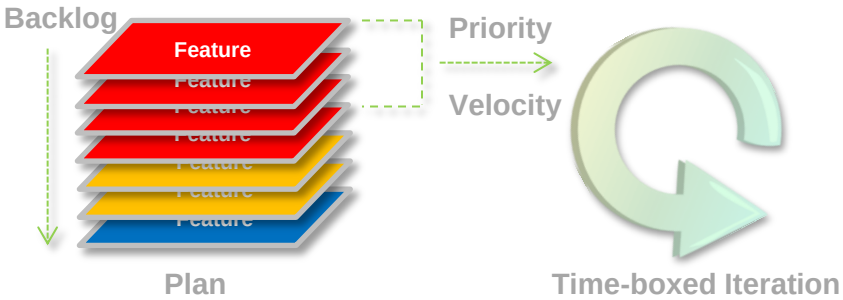
Participants

Product Owner – *Leads*
Scrum Master – *Participates*
Scrum Team - *Participates*

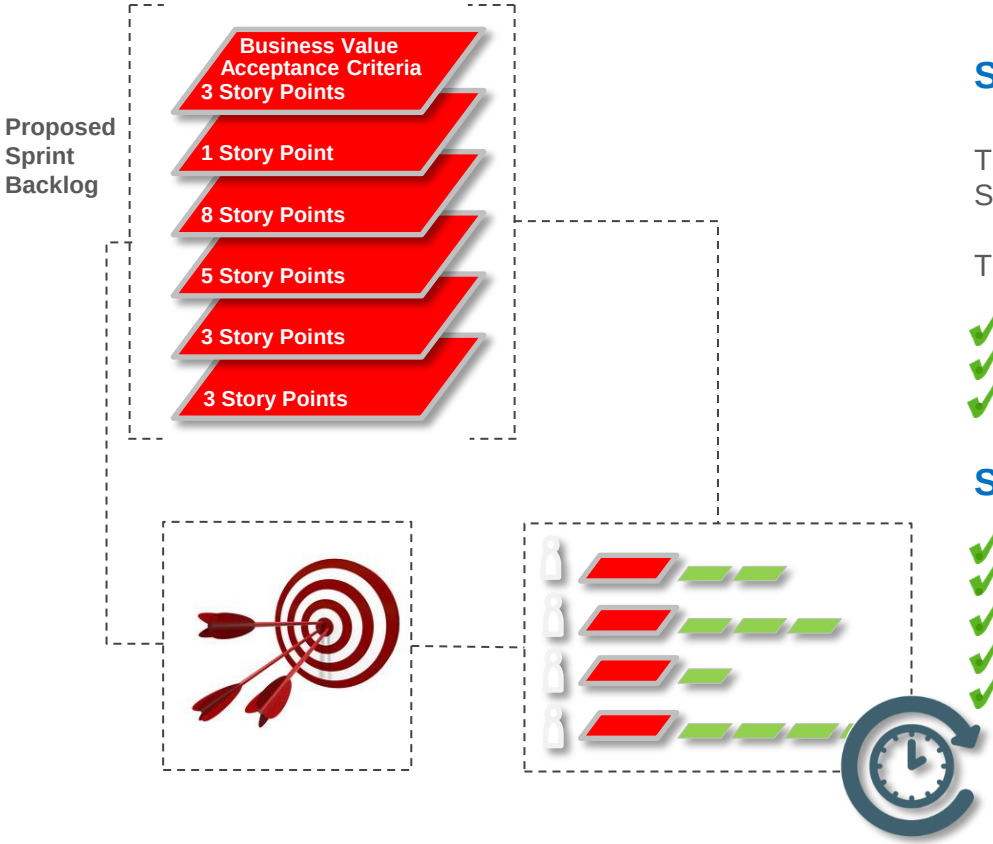
SPRINT FLOW

The Sprint ‘flow’ is the repeatable process and meetings that facilitate having a well managed Product & Sprint Backlog which ensures productive Sprints leading to rapid delivery of value to customers.

1. **Backlog Refinement** – The practice of ensuring the backlog is prepared and ‘Ready’ for inclusion in a Sprint
2. **Sprint Planning** – What will be developed, tested and demonstrated at the end of the Sprint.



SPRINT FLOW – SPRINT PLANNING



Sprint planning (pt.1)

This meeting determines which Stories will be brought into the next Sprint.

The ‘proposed’ sprint is presented by the PO for the team to review:

- ✓ Are the stories ‘Ready’
- ✓ Is the initial Story Point estimation still correct / check Velocity
- ✓ The PO proposes the Sprint Goal

Sprint planning (pt.2)

- ✓ Allocation of tasks to team members.
- ✓ Breakdown stories to small manageable sub-tasks
- ✓ Task Estimation – in time.
- ✓ Sprint Review / Objective Definition
- ✓ Revise Sprint Goal / Agree Sprint Commitment

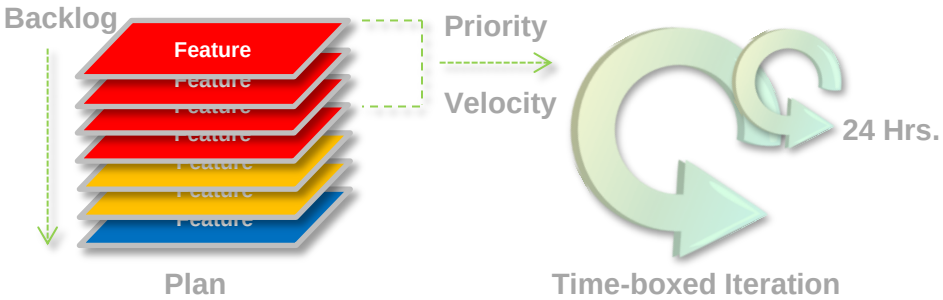
Participants

Product Owner – *Participates*
Scrum Master – *Leads*
Scrum Team - *Participates*

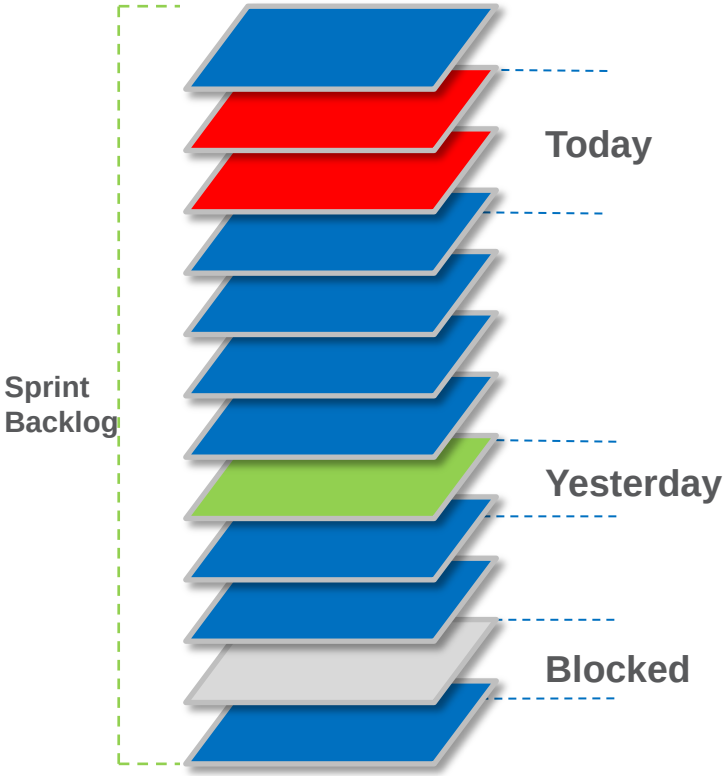
SPRINT FLOW

The Sprint ‘flow’ is the repeatable process and meetings that facilitate having a well managed Product & Sprint Backlog which ensures productive Sprints leading to rapid delivery of value to customers.

1. **Backlog Refinement** – The practice of ensuring the backlog is prepared and ‘Ready’ for inclusion in a Sprint
2. **Sprint Planning** – What will be developed, tested and demonstrated at the end of the Sprint.
3. **The Sprint / Daily Standup** – Once the Sprint gets underway, the team meet daily for 15 minutes to discuss progress



SPRINT FLOW – DAILY SCRUM



Daily stand-up (…or Scrum)

A **time-boxed (15 minutes)** daily checkpoint meeting: is all going as planned, or does the team need to adjust?

1. What did I work on yesterday?
2. What will I work on today
3. Do I have any blockers / impediments

***** *This is not a 'Status' meeting - reports on the technical intricacies of what team members are doing – it is to understand high-level progress and to identify blockers and when corrective action is required.*

Where further technical discussion is required, these are held 'post-scrum' – this helps keep the meeting focused and within the time-box.

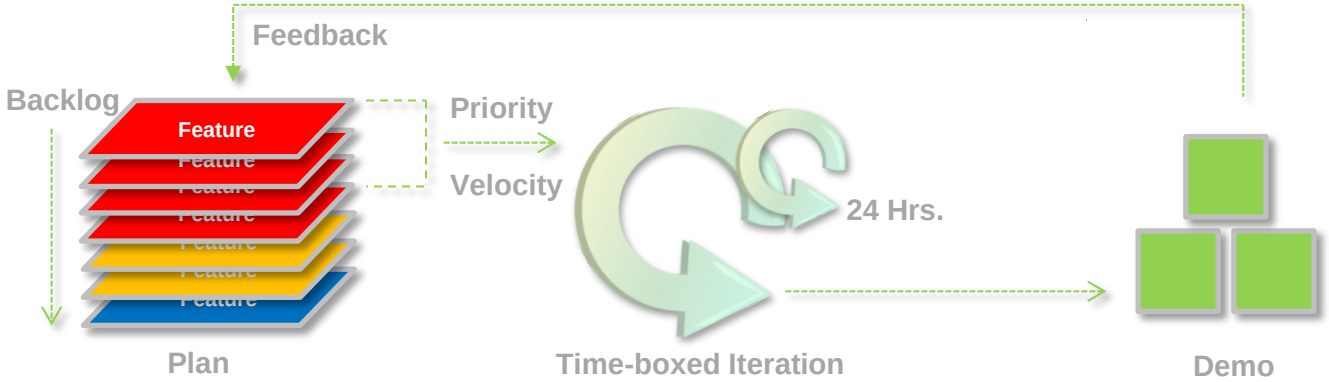
Participants:

Prodcut Owner – Participates
Scrum Master – Leads
Scrum Team - Participates

SPRINT FLOW

The Sprint ‘flow’ is the repeatable process and meetings that facilitate having a well managed Product & Sprint Backlog which ensures productive Sprints leading to rapid delivery of value to customers.

1. **Backlog Refinement** – The practice of ensuring the backlog is prepared and ‘Ready’ for inclusion in a Sprint
2. **Sprint Planning** – What will be developed, tested and demonstrated at the end of the Sprint.
3. **The Sprint / Daily Standup** – Once the Sprint gets underway, the team meet daily for 15 minutes to discuss progress
4. **Demo** - Customers are invited to view working software at the end of each Iteration in order to view progress and to provide **Feedback**.



SPRINT FLOW – DEMO

Sprint demo

The Scrum Team organizes and demonstrates the Story's from the Sprint that have met the Definition of Done to the Product Owner and relevant Customers / stakeholders.

- Customers can either **accept** the Stories as complete (meeting expectations) – or take the opportunity to **request enhancements**
- Enhancements are considered **New Stories**.
- The **PO** works with the **customer** to **create** the new stories
- The new Stories are then **prioritised** in the **Backlog** for future Sprints

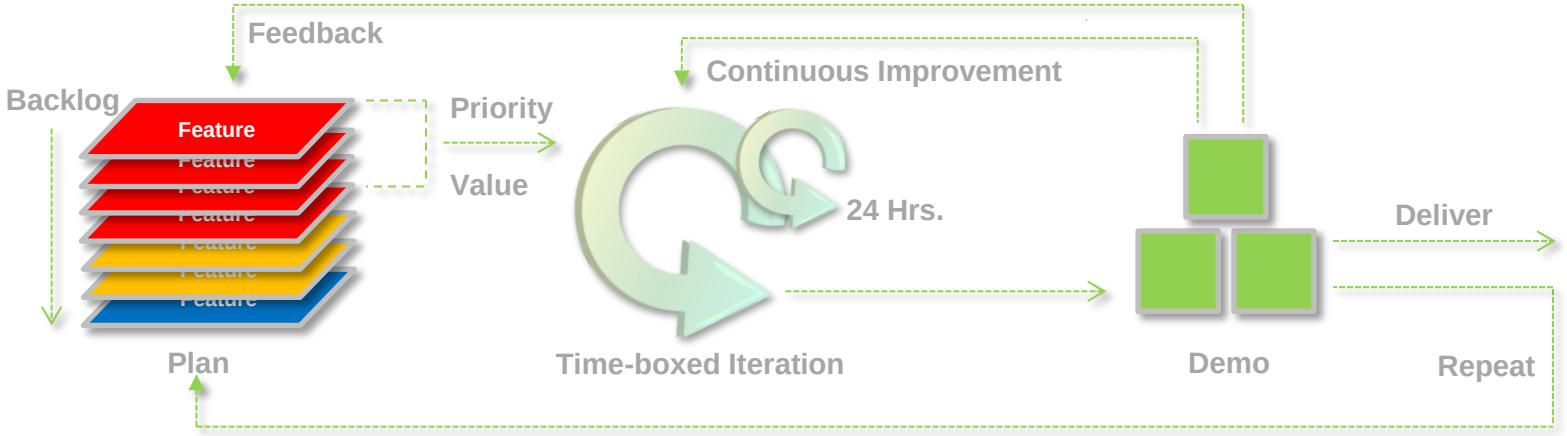
Participants

Product Owner – Leads
Scrum Master – Participates
Scrum Team – Participates
Customer / User - Participates

SPRINT FLOW

The Sprint ‘flow’ is the repeatable process and meetings that facilitate having a well managed Product & Sprint Backlog which ensures productive Sprints leading to rapid delivery of value to customers.

1. **Backlog Refinement** – The practice of ensuring the backlog is prepared and ‘Ready’ for inclusion in a Sprint
2. **Sprint Planning** – What will be developed, tested and demonstrated at the end of the Sprint.
3. **The Sprint / Daily Standup** – Once the Sprint gets underway, the team meet daily for 15 minutes to discuss progress
4. **Demo** - Customers are invited to view working software at the end of each Iteration in order to view progress and to provide **Feedback**.
5. **Retrospective** – After the Demo, the team meets to discuss possible areas for improvement in the Scrum process based on the last Sprint.



SPRINT FLOW – RETROSPECTIVE

Retrospective

Is held on the last day of the Sprint (usually after the Demo).

.

- 1 hour time-boxed Team only meeting reflecting on its own process, the last Sprint

- Teams focus on:
  1. What to Continue (what went well)
  2. What to Start (what can we change / introduce)
  3. What to Stop (what is not working)

- The team inspects their behavior in order to take action and adapt for future Sprints

This continual introspection is crucial to the ongoing adaptability and fine tuning of the Scrum process.

Participants

Product Owner – Participates when required
Scrum Master – Leads
Scrum Team - Participates

SPRINT FLOW – RETROSPECTIVE

Retrospective cont.

The Retrospective **is**:

- A point to reflect on the working processes of the Sprint
- An opportunity to provide open and candid feedback
- Document issues
- Collate actions to be implemented to improve the processes

The Retrospective is **not**:

An opportunity to blame people

.

Participants

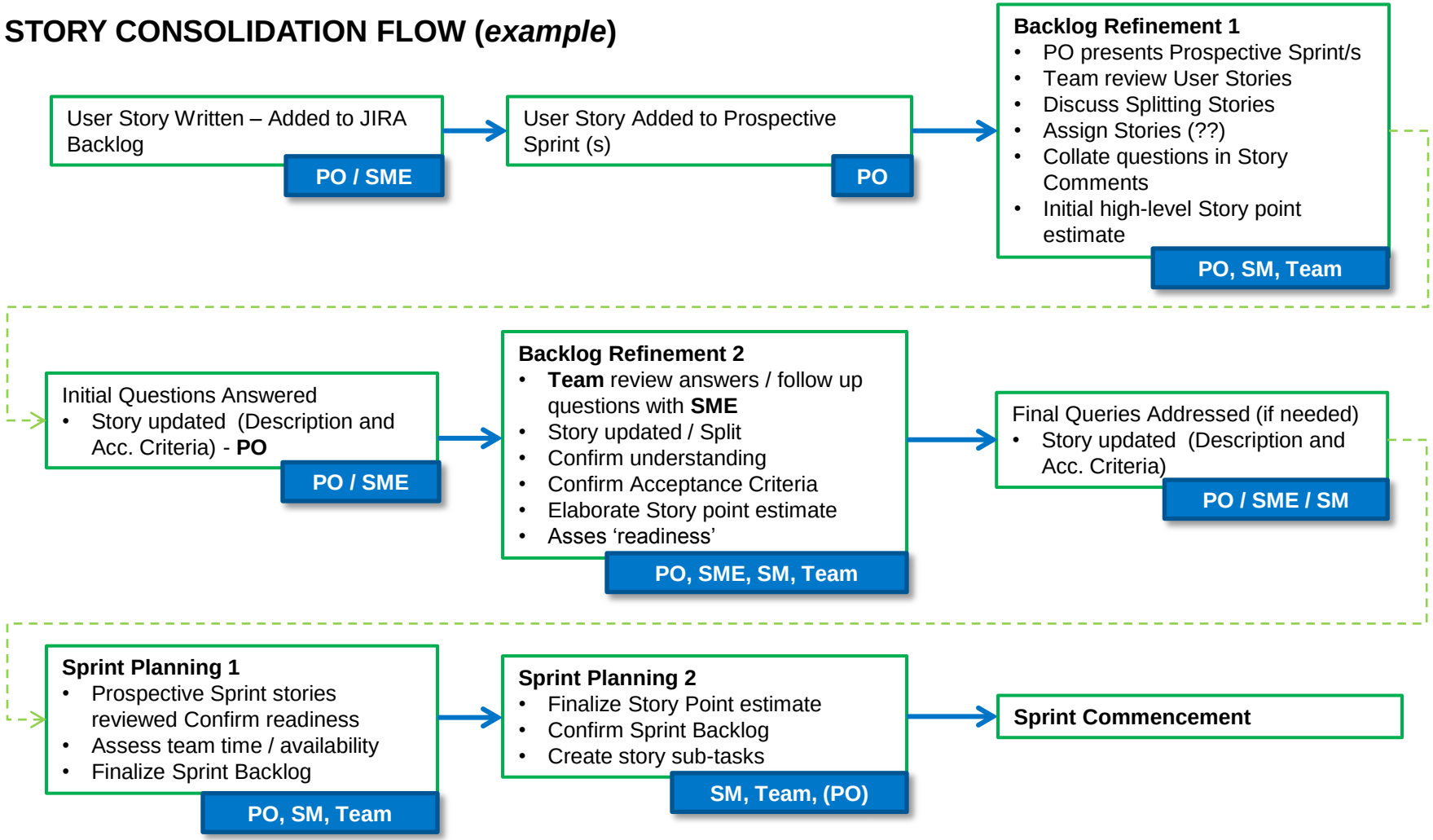
Product Owner – Participates when required
Scrum Master – Leads
Scrum Team - Participates

SPRINT SCHEDULE

Meetings should be held at the same time each Sprint to promote the process and maintain a steady cadence.

Meeting	*Tues*	Wed	Thur	Fri	Mon	Tues	Wed	Thu	Fri	Fri
	Day 1	*Day 2*	*Day 3*	*Day 4*	*Day 5*	*Day 6*	*Day 7*	*Day 8*	*Day 9*	*Day10*
Daily Scrum **15 Minutes**		✓	✓	✓	✓	✓	✓	✓	✓	✓
Backlog Refinement **1-3 Hours**			✓							
Backlog Refinement 2 **1-2 Hours**								✓		
Sprint Planning #1 **1-2 Hours**	✓									
Sprint Planning #2 **1-2 Hours**	✓									
Demo **30 - 45 min – As Required**										✓
Retrospective **1 hour**										✓

STORY CONSOLIDATION FLOW (*example*)



summary

SUMMARY

1. Active user involvement is imperative
2. The team must be empowered to make decisions
3. The team is accountable for the teams output meeting project expectations
4. Capture initial requirements at a high-level; lightweight and visual – elaborate as needed to deliver
5. Dedvelop small, incremental releases and iterate
6. Focus on frequent delivery of products
7. Complete each feature before moving on to the next
8. Apply the 80/20 rule
9. Testing is integrated throughout the project lifecycle - test early and often
10. A collaborative and cooperative approach between all stakeholders is essential

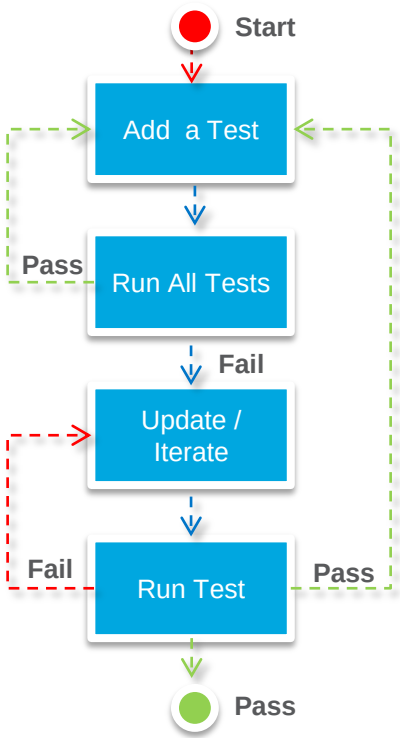
techniques

TEST DRIVEN DEVELOPMENT

TDD is a software development technique that involves repeatedly first writing a test case and then implementing only the code necessary to pass the test.

As an aspect of XP (Extreme Programming, TDD requires that an automated unit testis written before each aspect of the code itself.

1. Add a test
2. Run **ALL** tests and see new test *Fail*
3. Write code that satisfies test
4. Run **ALL** tests again to see them succeed
5. Refactor if necessary
6. Repeat – adding test to address all Story Test Cases / Acceptance Criteria



TEST DRIVEN DEVELOPMENT

Some of the benefits of TDD are:

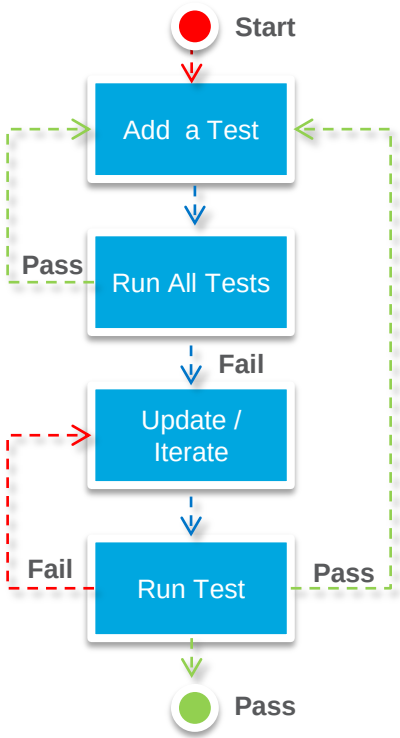
Faster Feedback: The team will have almost immediate feedback on components developed and tested. This shorter feedback loop allows for much faster turnaround on resolution of defects compared to traditional waterfall methodology where code is tested days or weeks after implementation.

Higher Acceptance: TDD implementations are more likely to match the product owners vision for the user story. Test cases are generated from the Acceptance Criteria.

Avoid Scope Creep: Prevents unwarranted design or components to sneak (think gold-platting) into the product. The test cases or unit test drivers define the exact set of required features. TDD makes it easy to identify redundant code, detect and terminate unnecessary engineering tasks.

Customer-Centric: The iteration can be defined as the implementation of functionality to execute against a pre-defined set of test cases, instead of the more traditional set of specifications or problem statement.

Modulation: Can lead to more modularized, flexible, and extensible code. The technique requires the team to think of the software in terms of small units that can be written and tested independently and integrated together later.




dev opps

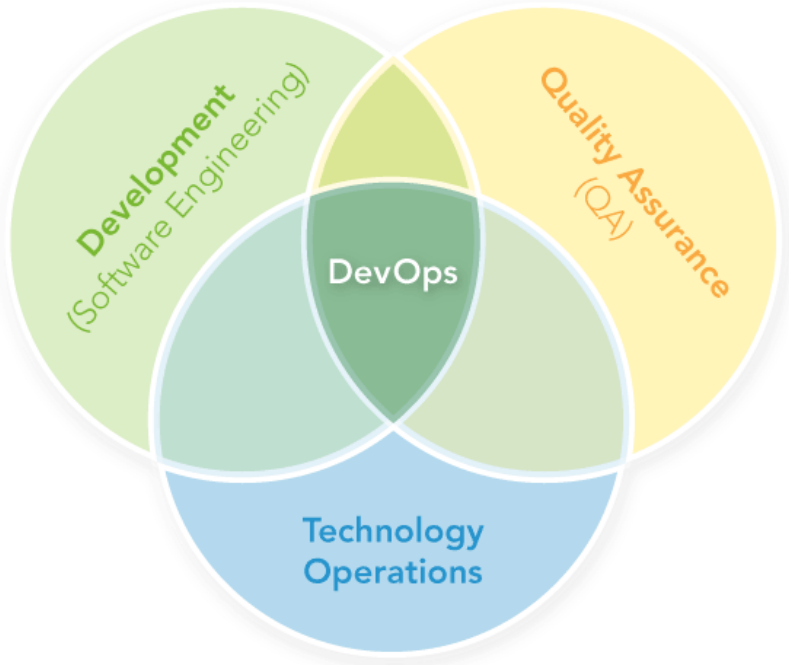
DEV OPPS

Is a term used to refer to a set of practices that emphasize the collaboration and communication of both software developers and information technology (IT) professionals while automating the process of software delivery.

At Micron the following is required from all Agile Teams:

Environment Management

- **Structured and Managed Environments**
- **Version Control** – Managing code in a repository (i.e. GIT)
- **Configuration Management** - Managing the environment around the code/solution, application/appserver/OS configuration, managing cfg. in production environments



DEV OPPS - CONTINUAL INTEGRATION

"…is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day.

Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly."

Martin Fowler

https://martinfowler.com/articles/continuousIntegration.html

At Micron the following is required from all Agile Teams:

- **Build Management & Automation** - Continuous Integration server in place and build automated at regular frequency
- **Automated Deployment** - Automated Deployment of code, configuration, binaries, etc to target systems
- **Release Management** - Scheduling, integration, coordination, release and deployment processes and tools
- **Automated Unit Testing** - Ability to automatically execute unit tests on build