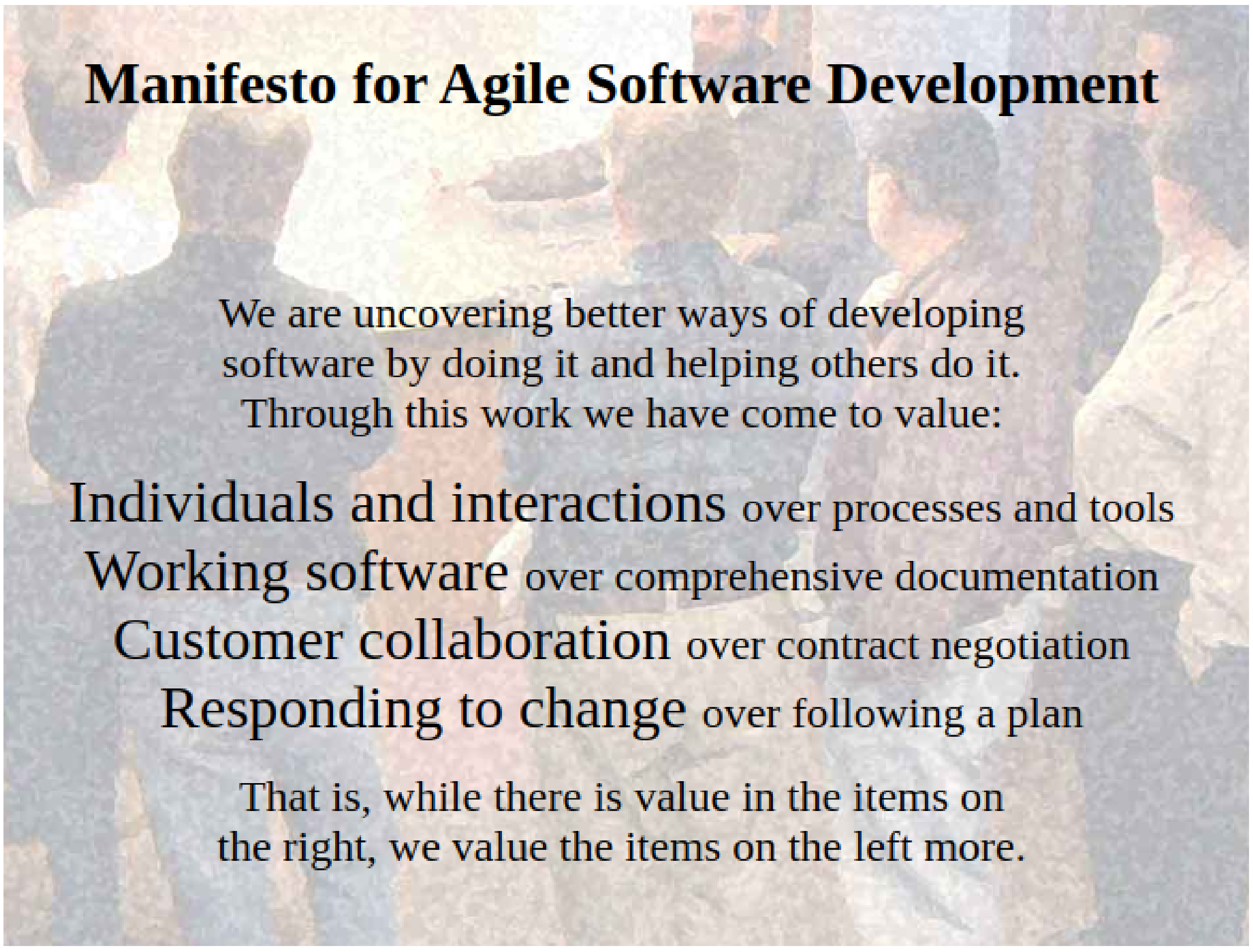
Agile
Agile is a flexible and iterative approach to project management and product development. It is more of an ideology containing methodologies. It involves iterative development where requirements to solutions evolve through collaboration between self-organising and cross-functional teams.

It involves collaboration, adaptability, and customer feedback throughout the entire process. Instead of rigid planning upfront, Agile involves breaking tasks into smaller, manageable chunks called "sprints" and continually adjusting plans based on real-time insights. This methodology enables teams to respond to changes quickly, deliver incremental value, and improve products based on evolving needs.

- There is no single variante or flavour of agile
    - Kanban - The main purpose of representing work as a card on the kanban board is to allow team members to track the progress of work through its workflow in a highly visual manner.
    - Agile is not one particular tool, practice or method
    - Agile is the mindset built by agile manifesto

Methodologies that fall under agile
TDD - test driven development
BDD - behaviour driven development
XP - extreme programming
LEAN - Lean programing
FDD - functional driven development

Origins of Agile
- Software developers in 80s and 90s experienced high levels of failed projects and noticed too much analysis upfront, restrictive change control, uncertainty



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.

https://agilemanifesto.org/

Principles of Agile

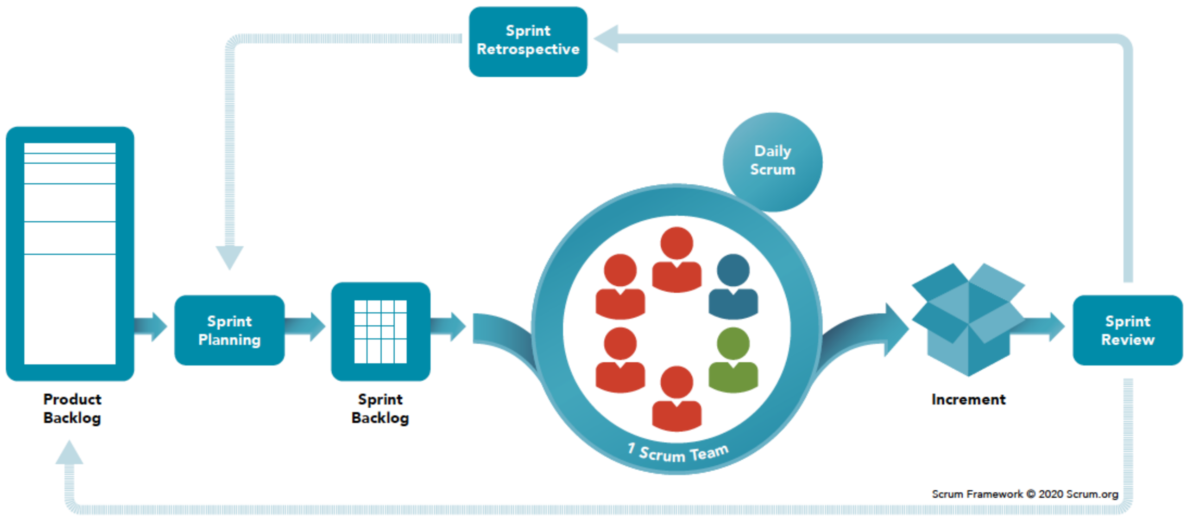
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks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

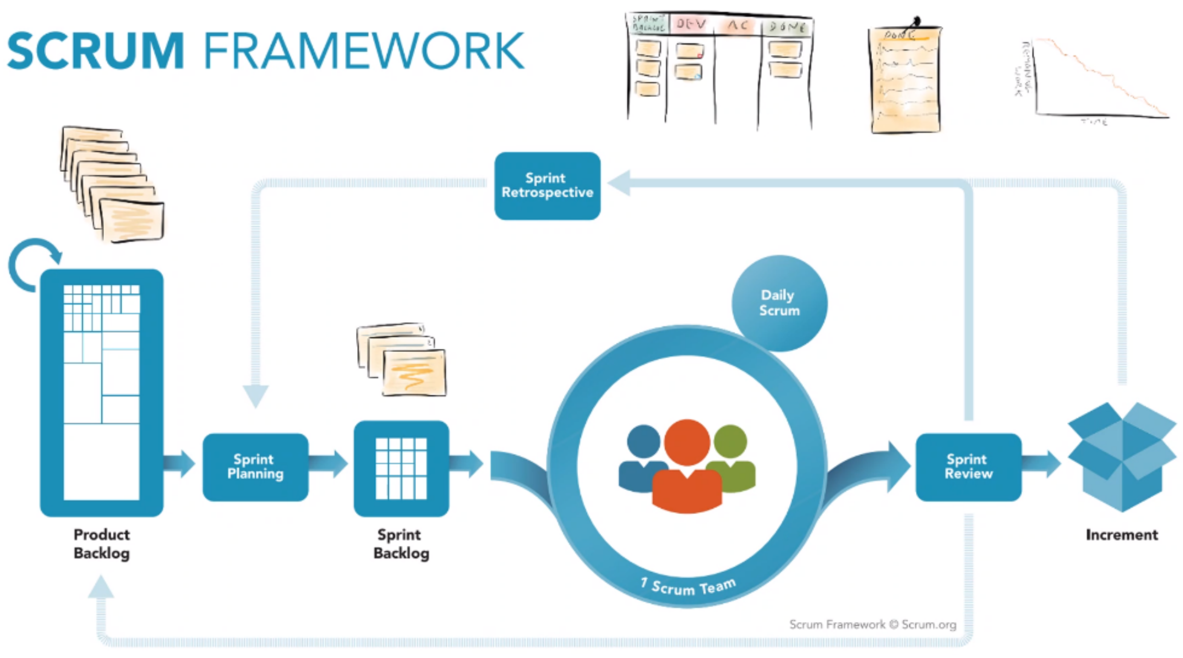
- Continuous attention to technical excellence and good design enhances agility.
- **Simplicity--the art of maximising the amount of work not done--is essential.**
- **The best architectures, requirements, and designs emerge from self-organising teams.**
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Scrum Framework

Scrum is an agile project management framework that emphasises collaboration, iterative progress, and flexibility in software development and other complex projects. It involves cross-functional teams working in short cycles called sprints to deliver incremental and valuable results.

“SCRUM is a team based framework to develop complex systems & products





Scrum Artefacts represent work or value and are human made
Product Backlog - An ordered or prioritised list of tasks or items for the whole project
Sprint Backlog - A lists of tasks or items from the product backlog that the team want to complete during one iteration or sprint of the process
Increment - The sum of all the completed backlog items at the end of a sprint with any tasks that have already been completed

Increments may be delivered to stakeholders prior to the end of the Sprint. The Sprint Review should never be considered a gate to releasing value

Events Each event in Scrum is a formal opportunity to inspect and adapt Scrum artefacts. These events are specifically designed to enable the transparency required.

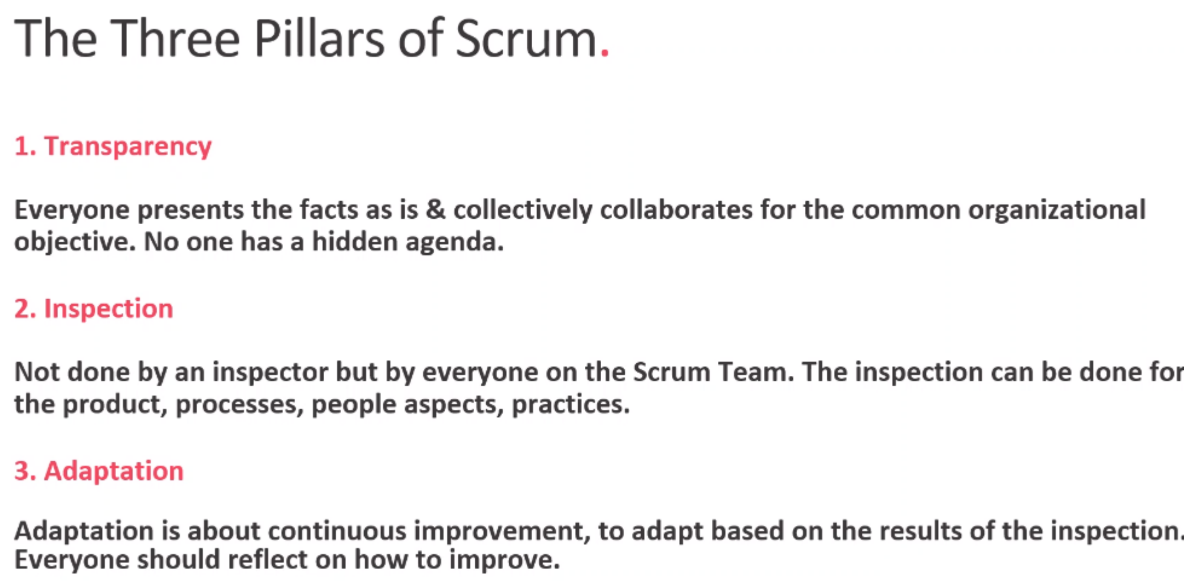
Sprint - fixed length event (usually 1 to 4 weeks) during which the team works to complete the items in the sprint backlog. It's where ideas are turned into value.

Sprint Planning - meeting held at the beginning of each sprint where the Scrum Team selects items from the product backlog to work on and moves them to the sprint backlog, and then plans how to accomplish each item and collaborate to decide roles for each task.

Daily Scrum - A daily stand-up meeting where the development team synchronises their activities, discussing progress towards the Sprint Goal. Daily Scrums improve communication, identify impediments and promote quick decision making, and often eliminate the need for further meetings.

Sprint Review - Scrum team present their work to key stakeholders and discuss their progress to the Product Goal. Review what was accomplished and commit incremented items(merge and pull), and can also receive some feedback from stakeholders. The scrum review is a working session, once the team has presented professionally, the team can celebrate when successful.

Sprint Retrospective - Internal meeting held after the sprint review where the team reflects on the sprint process and identifies ways to improve in the next sprint. Discuss what went well, what problems were encountered and provide helpful changes to improve effectiveness of the next sprint.



The Three Pillars of Scrum.

1. Transparency

Everyone presents the facts as is & collectively collaborates for the common organizational objective. No one has a hidden agenda.

2. Inspection

Not done by an inspector but by everyone on the Scrum Team. The inspection can be done for the product, processes, people aspects, practices.

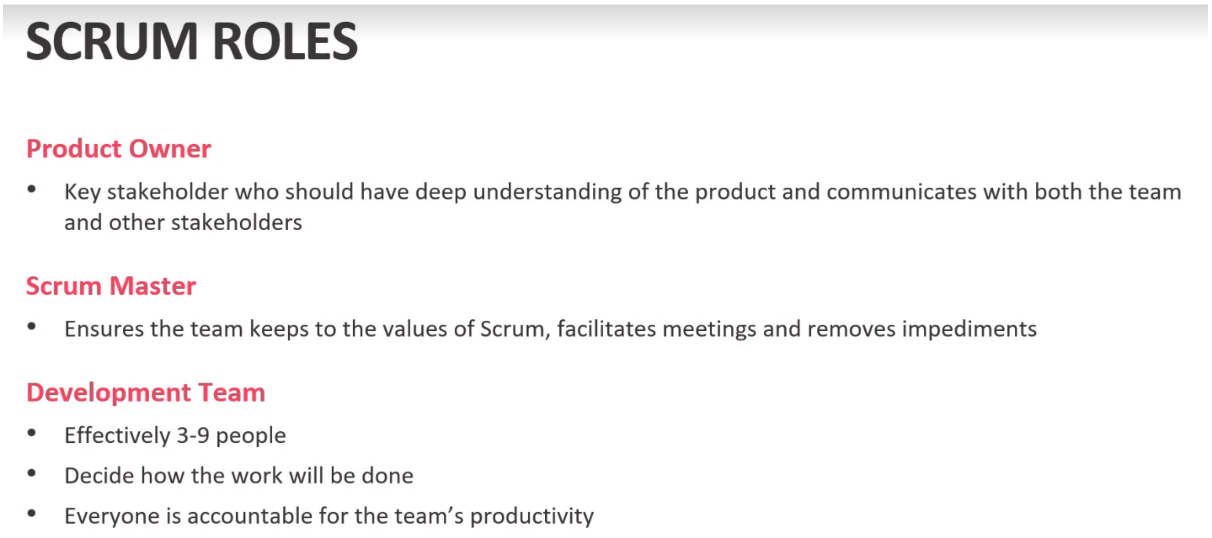
3. Adaptation

Adaptation is about continuous improvement, to adapt based on the results of the inspection. Everyone should reflect on how to improve.

Transparency - Each team member presents the facts as is, so there's no hiding unproductivity and collectively collaborating to a common objective. It means having a sense of openness where work and progress is visible to everyone including team members and stakeholders such as clients or ceo. No one has a hidden agenda. Trust individuals to deliver good news as well as bad news with the aim of reaching a common organisational goal.

Inspection - Done by everyone on the Scrum team. The inspection can be done for the product, processes, people aspects, and practice. Examples include inspecting whether each task in the sprint backlog has an assigned person, inspecting a team member's completed code that was discussed in the daily sprint to ensure it meets standard. Also links with transparency as it helps create an environment where you can ask other developers for clarity on their code. Good for feedback and team morale

Adaptation - continuous improvement, to adapt based on the results of the inspection. Everyone should reflect on how to improve. Involves adapting to changes in requirements during sprint reviews. Adapt roles and tasks to enhance productivity and improve the balance of workloads in a team.



SCRUM ROLES

Product Owner

- Key stakeholder who should have deep understanding of the product and communicates with both the team and other stakeholders

Scrum Master

- Ensures the team keeps to the values of Scrum, facilitates meetings and removes impediments

Development Team

- Effectively 3-9 people
- Decide how the work will be done
- Everyone is accountable for the team's productivity

Product owner - key stakeholder who has a deep understanding of the product and communicates with both the team and other stakeholders. They need to have the skills to explain concepts to technical and non-technical members. Act like the primary user of the product. They are accountable for communicating the Product Goal and incharge of the product backlog and have to ensure it remains transparent and visible.

Scrum master - also known as a scrum servant ensures the Scrum process is understood, followed, and continuously improved. They coach the team on self-management and cross-functionality. They serve as a facilitator, ensuring the team follows the best Scrum practices while also removing impediments to progress. For example, if a specialist developer is required for a specific task, they will find the right person and ensure the team progress continues.

Development team: effectively 3-9 people, cross-functional team that is self-organising. The team decides how the work will be done and essentially delivers the increments at the end of a sprint. They commit to completing items in the sprint backlog, and participate in daily stand ups to share progress or impediments. They work to continuously improve their skills, processes, and tools. Everyone is accountable for the team's productivity and creating value. It's better to create multiple cohesive teams rather than one large team as it slows progress and uses time during daily scrum meetings.

"Blockers" are critical issues that stop work, while "impediments" slow down progress in agile projects.

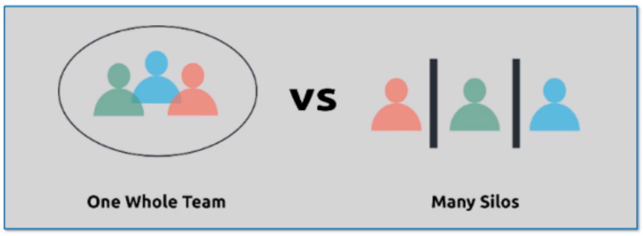
Agile is not appropriate

- Well-defined and predictable requirements
- Small and simple projects
- Critical path dependencies, if tasks must be completed in sequence
- Compliance or regulatory constraints
- Fixed deadline and fixed scope projects
- Limited user engagement
- Highly specialised and skilled teams

- Geographically diverse teams
- Real world example - construction - cannot iteratively improve foundation blueprints after they have been constructed

The Scrum Team.

- Co-located
- Engaged with the customer(s)
- Self-Organising
- Accountable & Empowered
- Cross Functional



Co-located - Agile teams are situated in the same physical space or communicate closely online, ensuring efficient collaboration and information exchange, often within a single office or through dedicated team channels.

Engaged with customers - Agile teams maintain active involvement with customers, gathering feedback and aligning their work to customer needs, ensuring the delivered product meets expectations.

Self-organising - trust teams to complete their tasks and meet expectations. Agile trusts teams to manage their own tasks and meet objectives, fostering responsibility and empowerment within the team.

Accountability - Team members take responsibility for each other's success by conducting inspections, code reviews, and openly discussing challenges. A positive environment encourages mutual help and sharing information openly.

Cross-functional - no predefined roles. Agile teams comprise diverse skills and expertise, allowing them to handle various aspects of a project. Learn new skills and processes from each other.

Agile teams cannot work across departments

Requirements

Requirements are descriptions of what a product, system, or project needs to achieve in order to meet the desired goals and objectives.

Functional requirements describe the specific functionalities, features, and interactions that the product or system should have, such as user log in

Non-functional requirements, also known as quality attributes or constraints, describe the characteristics that the product or system should exhibit such as responsiveness or response time

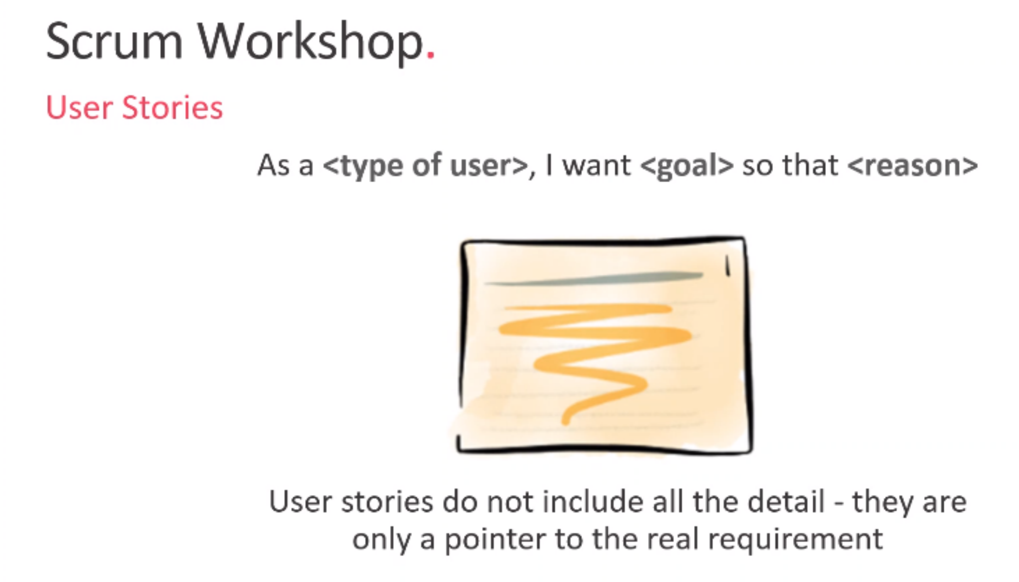
Product Owner facilitates communication between legal teams, investors, customers, owners, etc. Requirements can bring legal aspects such a data protection and copyright but also data collection where for existing software or architecture, we must consider how we gather information from customers, collect data, and web scrape data

- Interviewing and investigating the right people
- Documenting the product specifications that team and stakeholders have decided on
- Ask the right questions to the owner
- Communicate often and early
- Present requirements list to the stakeholder and ensure the teams is updated or clarified on feedback
- Do not make assumptions on requirements and ask specific questions that balance the value you are providing with the estimated cost of the project

In a scrum team the entire team is responsible for gathering requirements but is overlooked by the Product Owner who represents the interests of the stakeholders such as customers. The Product Owner:

- Defining and prioritising the product backlog
- Refining and clarifying requirements
- Communicating requirements to the team
- Making decision about requirements
- Accepting or rejecting work
- Promoting feedback on adapting requirements

The primary goal of requirements gathering is to create a comprehensive and accurate understanding of what stakeholders want to achieve with the project. To effectively gather requirements identify the relevant stakeholders including end-users, customers, or anyone invested in the project's outcome. Use techniques to gather information from stakeholders including interviews, ask the right questions to the owner and communicate often and early. Present requirements list to the stakeholder and ensure the team is updated or clarified on feedback. Do not make assumptions on requirements and ask specific questions that balance the value you are providing with the estimated cost of the project



Scrum Workshop.

User Stories

As a **<type of user>**, I want **<goal>** so that **<reason>**

User stories do not include all the detail - they are only a pointer to the real requirement

In Agile methodologies, including Scrum and Kanban, solution requirements are captured as "user stories." User stories are concise descriptions of a feature or functionality from the perspective of an end-user or customer. They help to communicate the needs and expectations for the software being developed in a way that is easily understandable and actionable by the development team.

As a [role]: This part identifies the type of user or stakeholder who will benefit from the feature. It could be an end-user, customer, admin, etc.

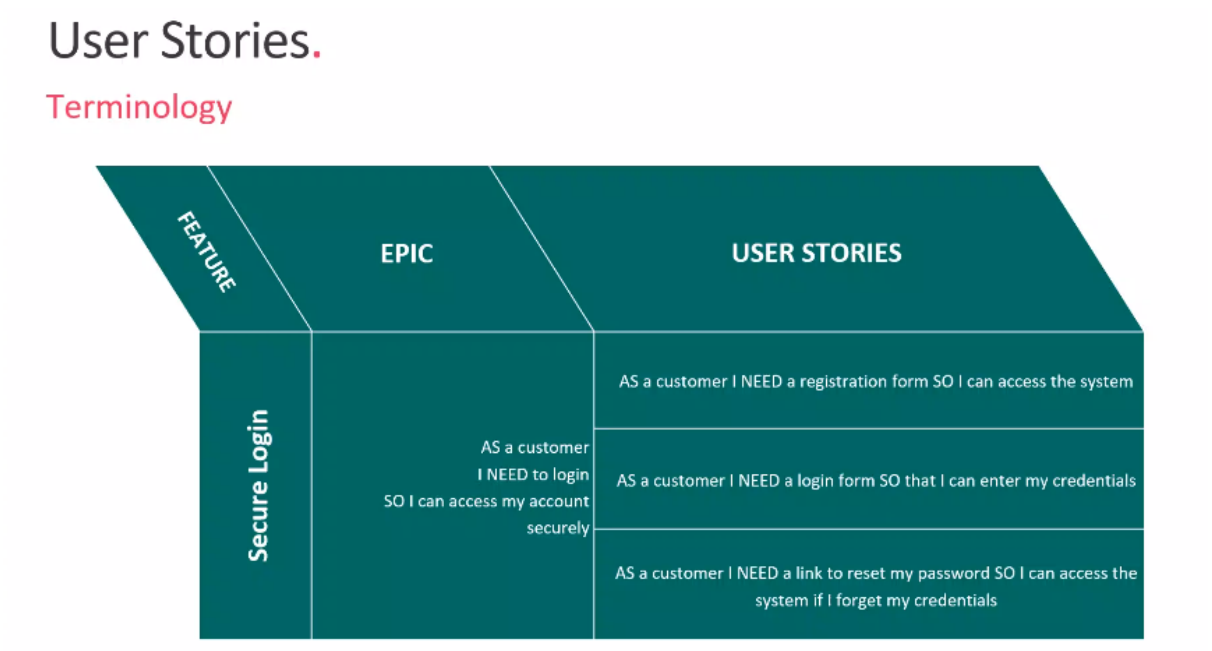
I want [feature]: This part describes the specific functionality or feature that the user desires.

So that [benefit]: This part explains the reason or benefit that the user will gain from having this feature.

"As a customer, I want to be able to save items to my wishlist so that I can easily keep track of products I'm interested in purchasing.

User stories are concise descriptions of features from a user's perspective, helping Agile teams prioritise and develop functionality that delivers value to customer

- A way of creating tasks from the users perspective such as an end-user or owner
- Clear, user-centred and easily understood structure allows new team mates understand he tasks
- Do not include all the details, they are only a pointer to the real requirement (no implementation details) Pointer could be conversations or more requirements gathering along the way
- A Kanban board is a visual tool used in Agile to display work stages as columns and tasks as cards. It's good for discussing features on the features integrated into the project, and facilitates collaboration. Example is Trello
- User stories are incomplete until a discussion takes place around them and are only a pointer to the real requirement
- Engage the team early to collecting details, high level ideas and then delve into details later



AS a customer I NEED to login SO I can access my account securely

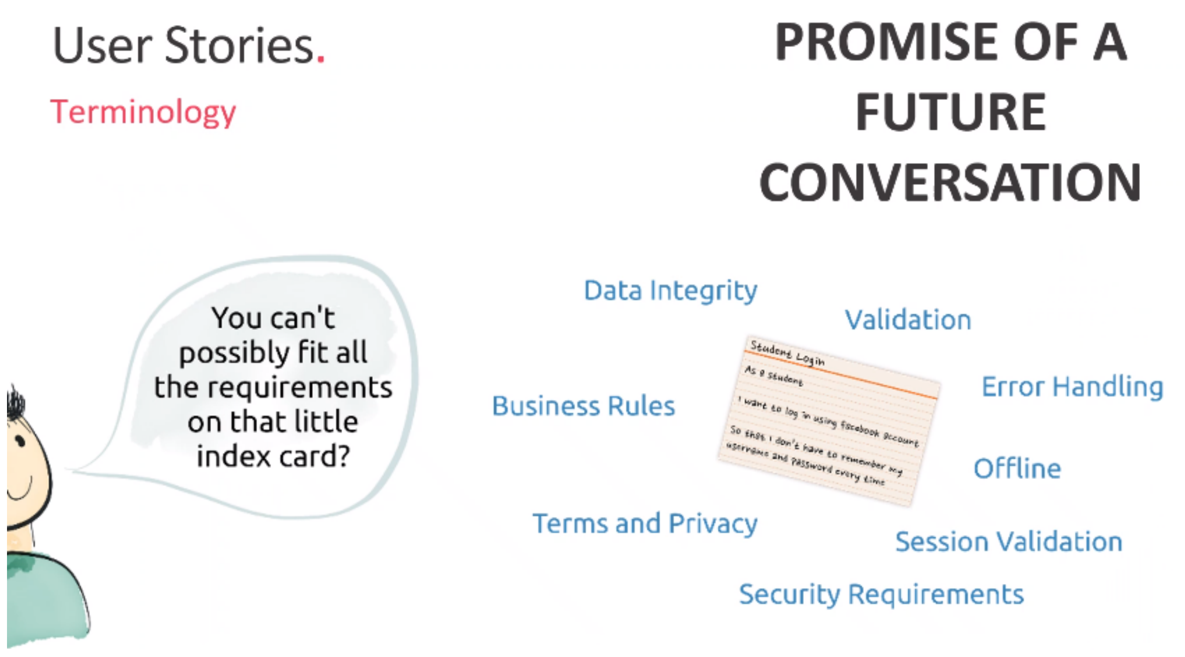
AS a customer I NEED a registration form SO I can access the system

AS a customer I NEED a login form SO that I can enter my credentials

AS a customer I NEED a link to reset my password SO I can access the system if I forget my credentials

Epic: Represents a large high-level user story or requirement that needs to be broken down into the exact tasks or user stories. It's a container for related user stories. Add feature labels for organisation.

User Stories: User stories are concise descriptions of specific functionality or requirements from an end-user perspective. They break down the features described in an epic into smaller, actionable units, allowing them to be planned, developed, and tested within a single iteration.



User Stories.

Terminology

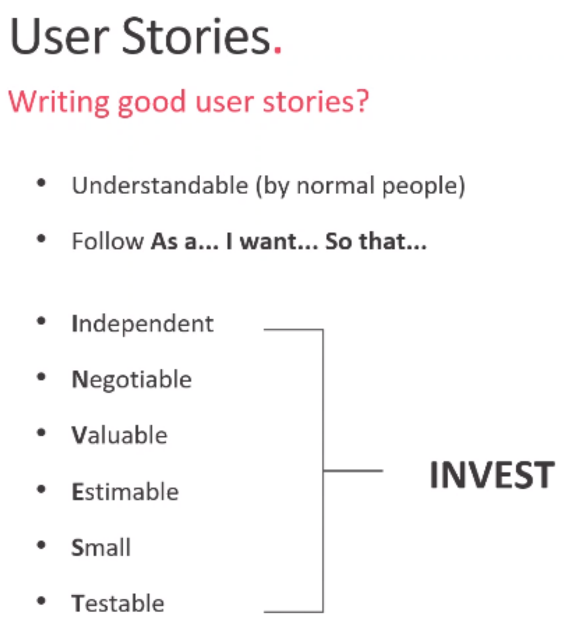
PROMISE OF A FUTURE CONVERSATION

You can't possibly fit all the requirements on that little index card?
Data Integrity
Validation
Business Rules
Student Login
As a student
I want to log in using facebook account
So that I don't have to remember my username and password every time
Error Handling
Offline
Terms and Privacy
Session Validation
Security Requirements

- User stories are a promise of a future conversation
- User stories can be prioritised and left in the product backlog
- Detailed user stories where implementation details are discussed are only necessary prior to their use
- User stories can be written by every member of the scrum team, the product owners responsibility is ensuring the product backlog exists and user stories are included. Team cannot proceed until the product backlog is created to make a sprint backlog. Beneficial for cross-functional teams to get involved where members have different skill sets, one member may provide insights on data security whilst another on data storage.

Before sprint, three amigos should agree the user story is ready for development ensures that user stories or requirements are well-understood, well-defined, and effectively implemented.

- Business Analysts represents whether the story aligns with the business goals
- Developer represents technical perspective such as discussing feasibility and technical solutions
- Tester considers how the user story can be effectively tested to ensure quality and the proper functionally
- But important to note there are no predefined roles so all scrum team members can contribute to any area.



User Stories.

Writing good user stories?

- Understandable (by normal people)
- Follow **As a… I want… So that…**

- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimable
- **S**mall
- **T**estable

INVEST

- Understandable non-technical people - Language non-technical and non-specific to technologies
- Independent - story should be end to end
- Negotiable - multiple ways to deliver any given story - no specific technology as a client may change technology choice. Methods of achieving end goal should be negotiable between product owner, development team, customers to prevent unrealistic constraints on the feature - encourage conversation and scope negotiation
- Valuable - valuable to business or user - discard if it doesn't provide value
- Estimable - estimate how big the user story is, how many resources and cost, time
- Small - small as possible, if an epic make smaller
- Testable - write a measurable test to say whether the test has passed or failed. This could be a unit test.

User Stories.

Writing good user stories?

- User comes first
- Use personas
- Create collaboratively
- Keep your stories simple
- Start with Epics
- Refine stories until they are ready
- Add acceptance criteria
- Use paper cards
- Keep them visible
- They are not everything

User Stories.

The Three C’s?

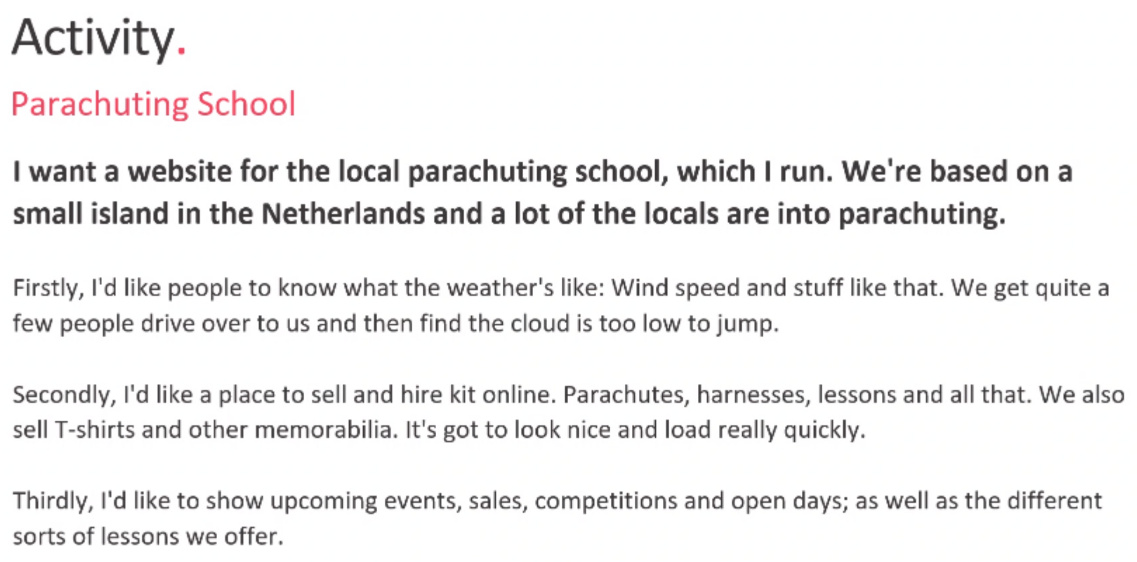
CARD

CONVERSATION

CONFIRMATION

- Card represents a card on trello for example to represent a user story
- Conversation is the interactive discussion that takes place between stakeholders
- Confirmation is the set of clear and formal acceptance criteria that define how the team will know when the user story is successfully implemented

https://chat.openai.com/share/f22c809b-cd68-4174-9f21-89a809050c9a



Activity.

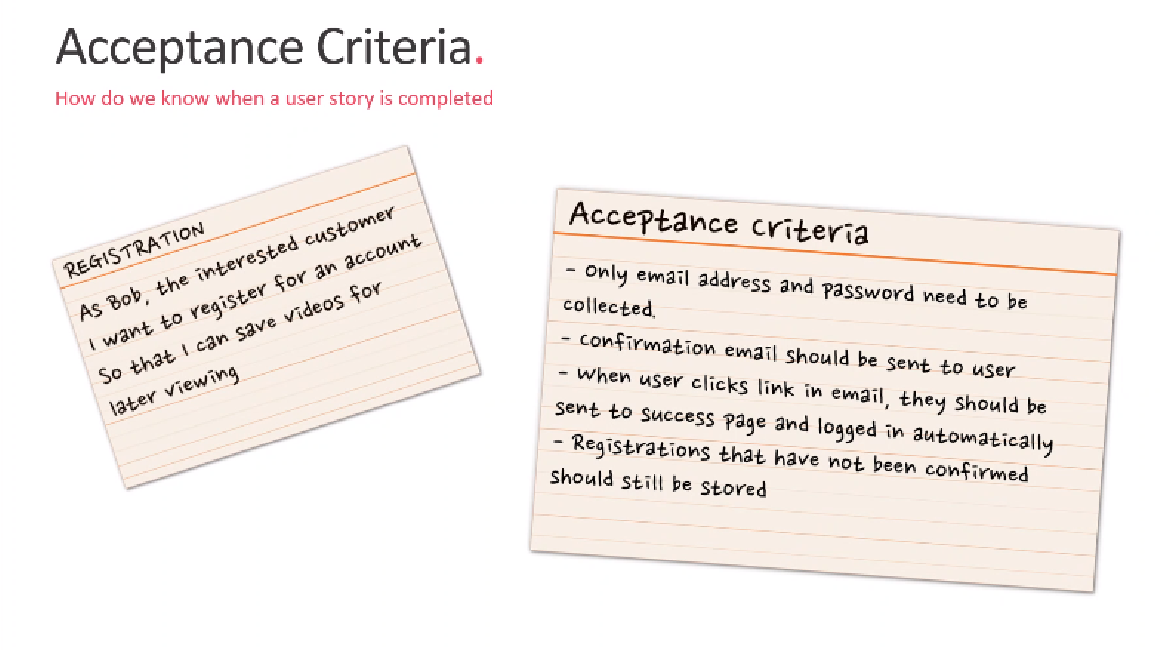
Parachuting School

I want a website for the local parachuting school, which I run. We're based on a small island in the Netherlands and a lot of the locals are into parachuting.

Firstly, I'd like people to know what the weather's like: Wind speed and stuff like that. We get quite a few people drive over to us and then find the cloud is too low to jump.

Secondly, I'd like a place to sell and hire kit online. Parachutes, harnesses, lessons and all that. We also sell T-shirts and other memorabilia. It's got to look nice and load really quickly.

Thirdly, I'd like to show upcoming events, sales, competitions and open days; as well as the different sorts of lessons we offer.



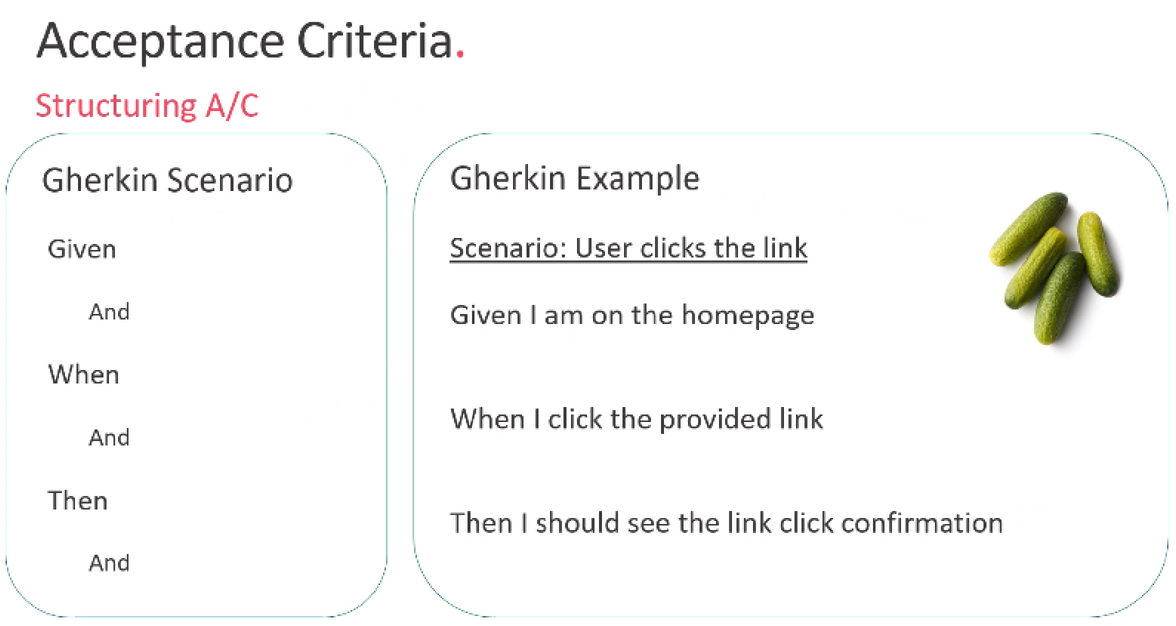
Acceptance Criteria.

How do we know when a user story is completed

REGISTRATION
As Bob, the interested customer
I want to register for an account
So that I can save videos for
later viewing

Acceptance criteria
- only email address and password need to be collected.
- confirmation email should be sent to user
- when user clicks link in email, they should be sent to success page and logged in automatically
- Registrations that have not been confirmed should still be stored

- Acceptance criteria are a set of conditions or requirements that must be satisfied in order for a user story or a feature to be considered complete
- helping both the development team and stakeholders have a shared understanding of the desired outcome
- Agrees with the owners acceptable solution
- Gherkin syntax



Acceptance Criteria.

Structuring A/C

Gherkin Scenario

Given

And

When

And

Then

And

Gherkin Example

Scenario: User clicks the link

Given I am on the homepage

When I click the provided link

Then I should see the link click confirmation

Given I am on the homepage and logged in and in the UK, when i clicked the provided link, then i should see the link confirmation
Scenario is the title of the gherkin script
Feature is the title of epics or user stories

Estimation and prioritising

Estimation is done by the entire team during a sprint planning meeting. A sprint estimation shows how much effort a series of tasks require.

- Product Owner ensures that the User Stories are clear, can be subjected to estimation
- One way estimation is done is to coordinate dependencies as its useful to know when one task is dependent on another task to be started
- Establish user stories and priorities them
- Convert story points into hours to estimate how long it would take to complete a task
- Teams can assign story points based on complexity, amount of work, and risk of uncertainty and over time they will better understand how much they can complete in a period of time.

It's based on assumptions, requirements, and dependencies of a project. A dependency is any relationship between two or more tasks where one task is dependent on the output of another task

- Establish and prioritise user stories

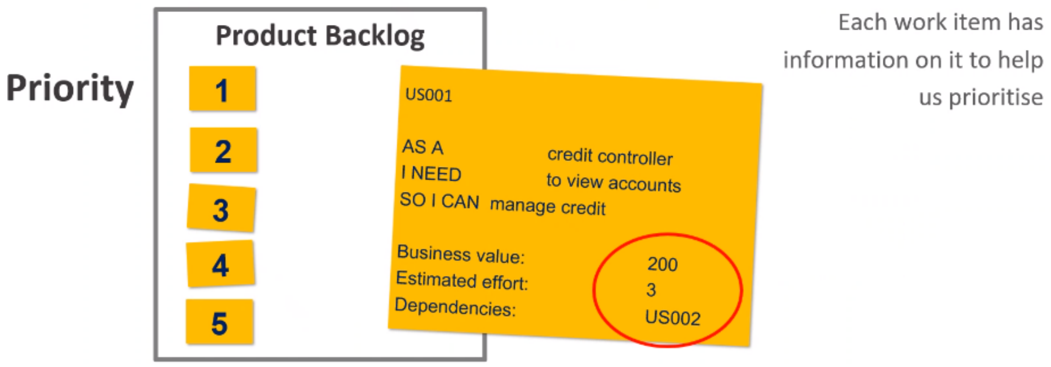
Product and Sprint Backlog

- Everyone is responsible for the product backlog but the Product owner responsible for content, availability and ordering of product backlog
- Dynamic and never complete as they tasks constantly adapt to change in requirements
- Sprint backlog set of product backlog items that have been moved to be ready for the upcoming sprint. Includes a plan for delivering a product, incrementing and realising the sprint goal

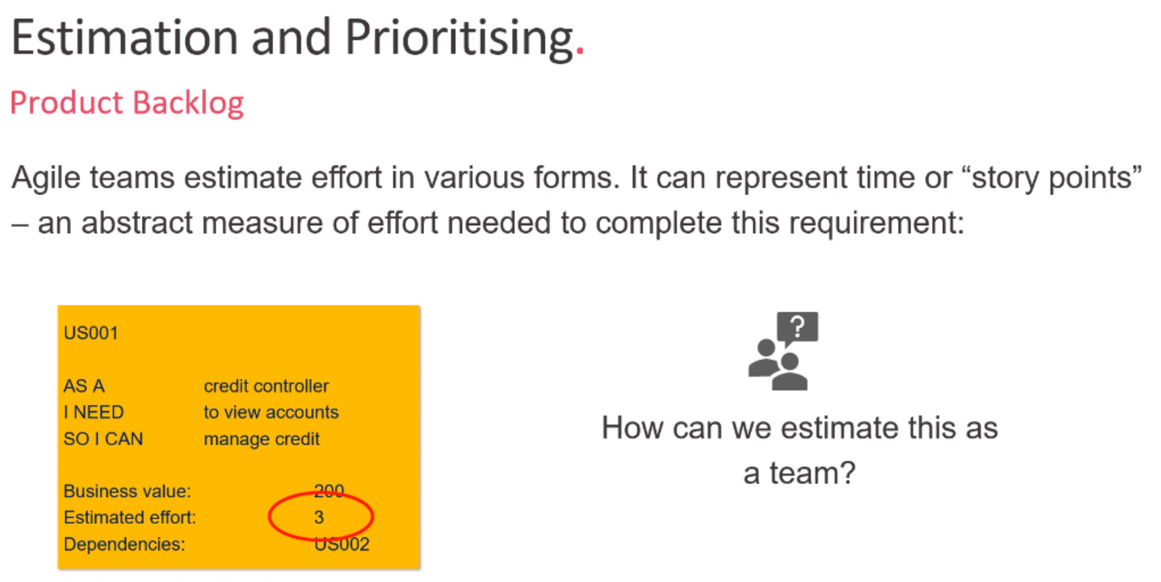
Estimation and Prioritising.

Product Backlog

User Stories are placed in the Product Backlog in priority order:



- Ordered list of all tasks needed to achieve a completed product
- Single source of requirements for any changes to be made to a product
- User stories are place in the product backlog in priority order
- Each work item has information on it to help us prioritise
    - Business value - how important is a customer being able to log in vs colour theme
    - Estimated effort: how many developers or engineers will it take
    - Dependencies - which tasks need to be done prior to being able to start this task. IN reality it may be difficult to make all user stories independent.



Estimation and Prioritising.

Product Backlog

Agile teams estimate effort in various forms. It can represent time or “story points” – an abstract measure of effort needed to complete this requirement:

US001

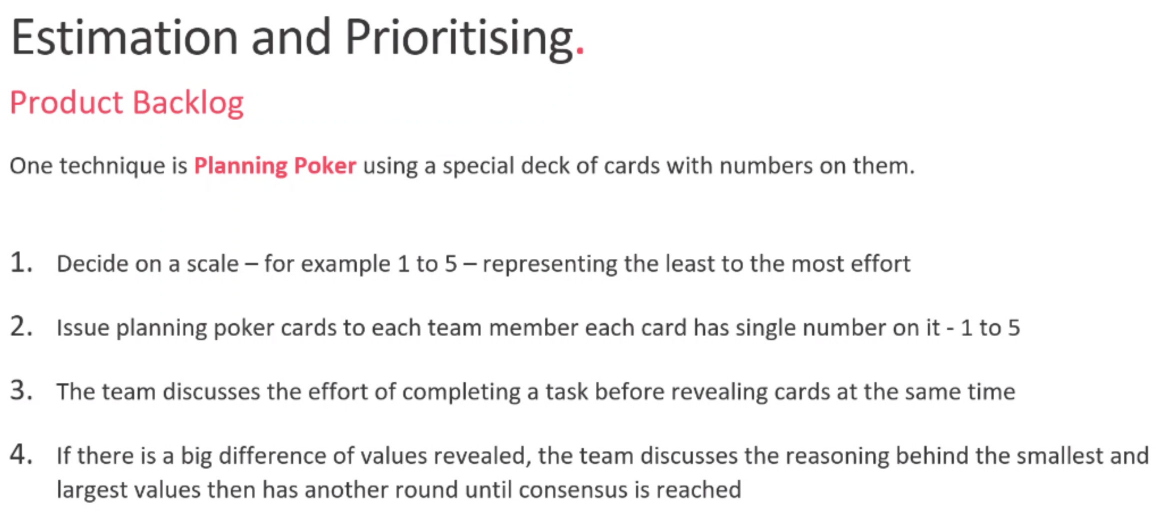
AS A	credit controller
I NEED	to view accounts
SO I CAN	manage credit
Business value:	200
Estimated effort:	3
Dependencies:	US002

How can we estimate this as a team?

- Agile teams estimate effort in various forms. It can represent time or story points - an abstract measure of effort needed to complete this requirement

To estimate effort:

- Planning Poker
- Deck of cards with numbers 1 - 5 representing least to most effort
- Issue the cards to each team member, discuss the reasons for effort involved and the team reveals the cards, if there is a big difference in the values revealed, and continue discussion until a consensus is reached. Works well for cross-functional teams of diverse skill sets



Estimation and Prioritising.

Product Backlog

One technique is **Planning Poker** using a special deck of cards with numbers on them.

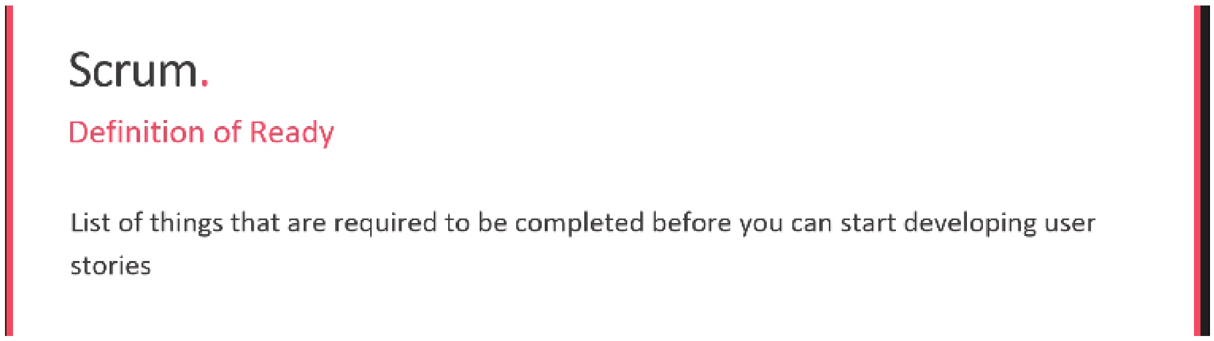
1. Decide on a scale – for example 1 to 5 – representing the least to the most effort
2. Issue planning poker cards to each team member each card has single number on it - 1 to 5
3. The team discusses the effort of completing a task before revealing cards at the same time
4. If there is a big difference of values revealed, the team discusses the reasoning behind the smallest and largest values then has another round until consensus is reached

The bucket system - estimating a large number of items with large number of participants

- Place a copy of a user story in a bucket from 1 - 5

Task management

KANBAN - TRELLO | JIRA | GITHUB PROJECTS
Cards on trello board for various columns such as product backlog, sprint backlog, and to do, ready, done, testing



Scrum.

Definition of Ready

List of things that are required to be completed before you can start developing user stories

"Definition of Ready" for user stories is a checklist of requirements that need to be met before the development team can start working on the story

- List of things required to be completed before you can start developing user stories
- Team must determine what needs to be done and the work needed to complete the user story (checklist of requirements that need to be met before the development team can start working on the story)
- Example case for being ready for a sprint - tasks in product backlog being confirmed as ready, product sprint backlog exists, user stories in sprint backlog are in priority order, backlog describes tasks, sufficient time to complete user stories, user stories in the backlog individually are ready in terms of prerequisites. As a whole the sprint backlog is ready
- Example case for a user story being ready - everyone must agree on what the user story says and mean, agree on the three c's, acceptance criteria must be done including happy and sad paths completed etc

Have a web page, decide on authentication for credentials, purchased subscriptions and downloaded dependencies/plug-ins. Then tasks is ready to be started

Happy and sad path
Scenario - Add item to basket
Happy path - given i am on the page, when i click the item, then it's added to my basket (successful)
Sad path - given i am on the page when i click the item, then i get a notification the item is out of stock (not successful or what but works correctly - unintended encounter for user)
Alternative paths non-happy and non-sad paths

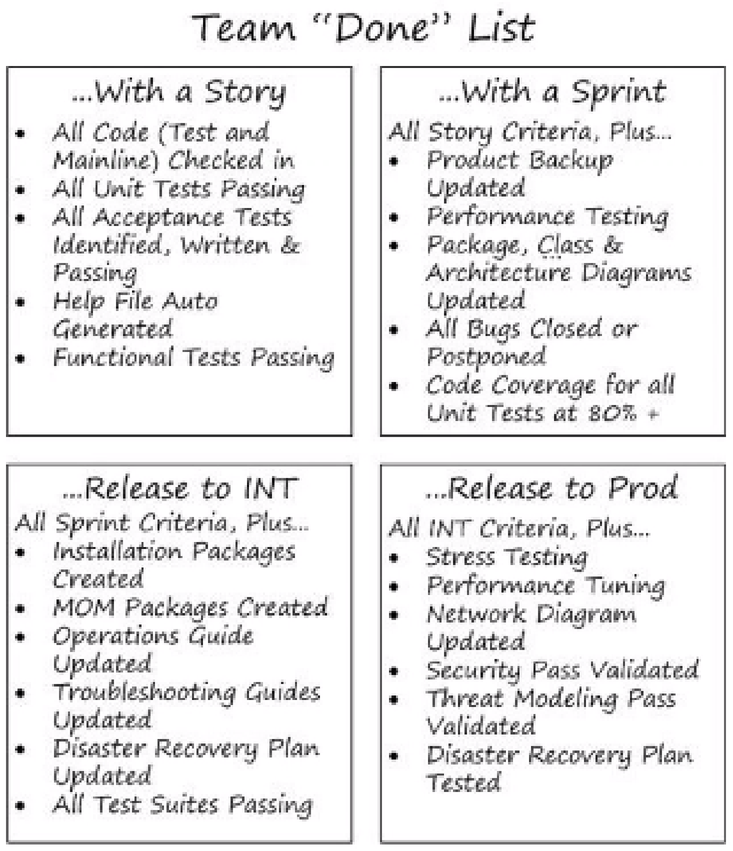
Definition of Ready
The "Definition of Ready" is a set of prerequisites that a user story should meet before it's considered ready for development
"Definition of Ready" for user stories is a checklist of requirements that need to be met before the development team can start working on the story

- The value, priority and size of the story is estimated
- Dependencies or blocker identified and solved
- Definitions of ready are generic, one definition of each for whole project
- Act as checklist on whether to proceed with a sprint or user story
- Dependencies, estimation, priority, and acceptance criteria, if followed, we can determine if a user story or sprint backlog is ready

Definition of Done
When all conditions including acceptance criteria, that a product backlog item must satisfy are met and ready to be accepted as done

When a product backlog item or increment is described as done, everyone must understand what done means. All team members need to have a shared understanding of completion which relates to transparency.

When a Product Backlog item fulfils this definition, it becomes part of an Increment



Team "Done" List

...With a Story
- All Code (Test and Mainline) Checked in
- All Unit Tests Passing
- All Acceptance Tests Identified, Written & Passing
- Help File Auto Generated
- Functional Tests Passing

...With a Sprint
All Story Criteria, Plus...
- Product Backup Updated
- Performance Testing
- Package, Class & Architecture Diagrams Updated
- All Bugs Closed or Postponed
- Code Coverage for all Unit Tests at 80% +

...Release to INT
All Sprint Criteria, Plus...
- Installation Packages Created
- MOM Packages Created
- Operations Guide Updated
- Troubleshooting Guides Updated
- Disaster Recovery Plan Updated
- All Test Suites Passing

...Release to Prod
All INT Criteria, Plus...
- Stress Testing
- Performance Tuning
- Network Diagram Updated
- Security Pass Validated
- Threat Modeling Pass Validated
- Disaster Recovery Plan Tested

With a story
- All code check in
- All unit test passed
- All acceptance test which are test written according to acceptance criteria are passed
- Functional tests passing - user tests functions and all pass
- Writing unit tests and passing acceptance criteria being met, proven by test

WIth a sprint
- Product backlog updated and removed any tasks and moved to sprint backlog
- Performance testing to ensure no crashes and expected response times
- Package, class and architecture diagrams updated - update ERD to new database
- All bugs closed or postponed
- Code coverage for all unit tests at 80%
- All user stories in the sprint are defined as done

Release to INT (testing stage not ready for deployment)
- All sprint criteria PLUS
- Installation packages created
- Operations guide updated
- Troubleshooting guides
- Disaster recovery plan updated
- All test suites passed

Release to Production (Production ready)

- All INT criteria PLUS
- Stress testing
- Performance testing
- Network diagram updated
- Security pass validated
- Threat modelling pass validated
- Disaster recovery plan tested

- Definition of done for the parachuting school
  - Acceptance criteria met due to test running and passing
  - Happy and sad and alternative paths being met
  - Performance tests passed

- Definition of done for sprint
  - All user stories being marked as done
  - Items being moved out of the product backlog
  - Items being moved into the increment
  - Documentation updated to reflect changes and additions

Sprint Retrospectives

- sprint retrospective is a meeting in Agile methodology where the team reflects on the completed sprint. It focuses on what went well, what could be improved, and actions to enhance the next sprint's effectiveness.

The agenda

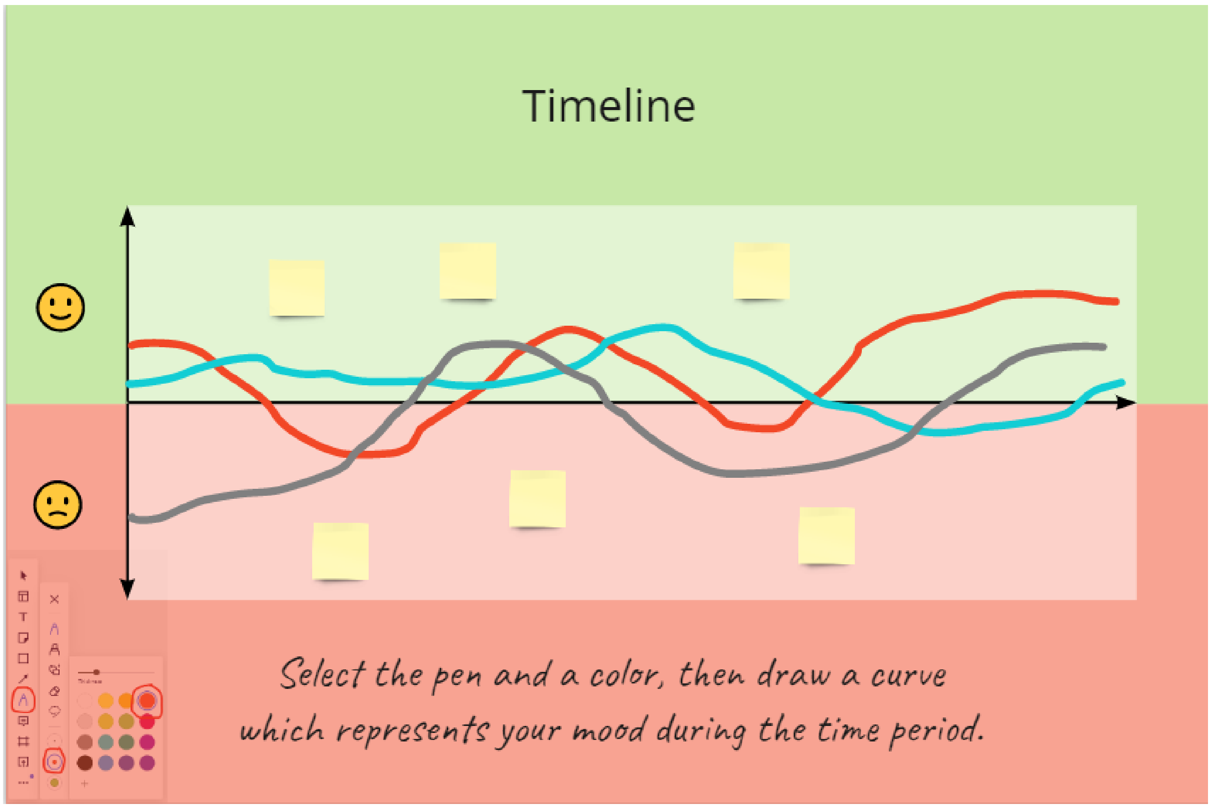
- Set the stage - Internal discussions on improving work processes, efficiency and welfare
- Gather data - gather data on the sprint, achievements, user stories that we did not complete, analyse mood graphs,
- Generate insights - insights and patterns, new technologies for future, causes behind problems, discuss and identify effective resolutions and improvements including new ways of working learned.
- Decide what to do - make a decision on the actual changes in the next sprint, pick a few specific and easily attainable goals on what to change and implement them
- Close the retro - summarise the action plan, celebrate achievements and motivate the team for the next sprint

1-1 tracker is a personal retrospective - continuously improve method of work
Starfish retrospective - start - stop - continue version

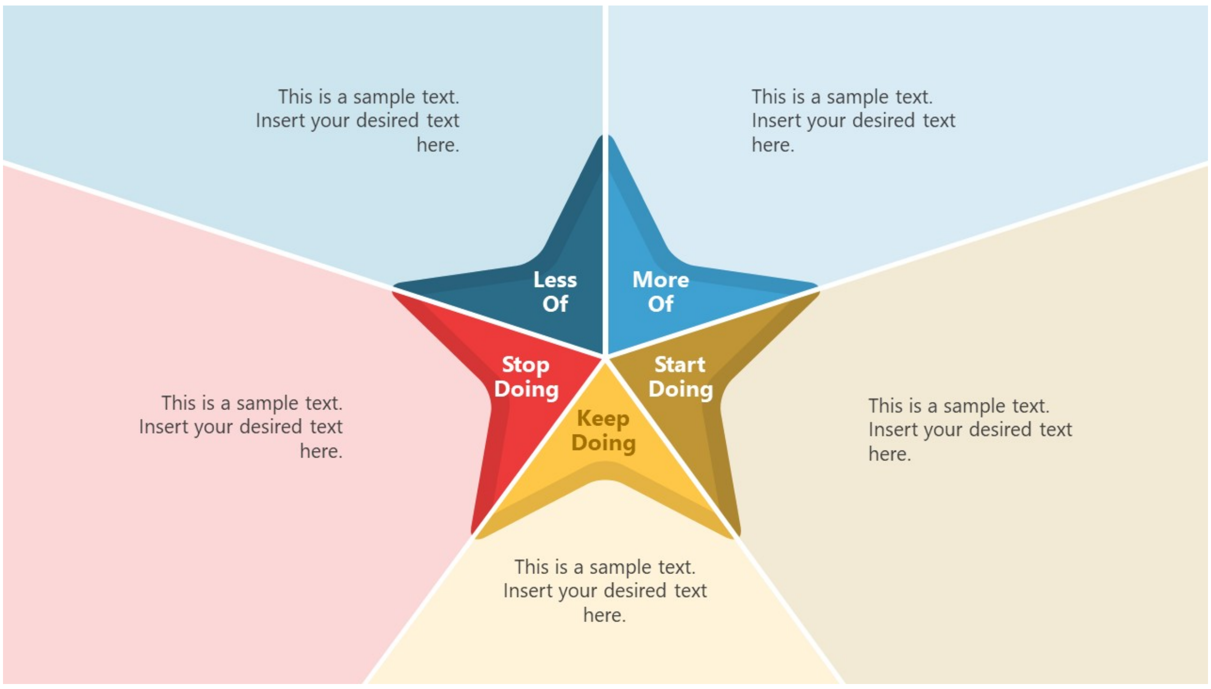
Team Retro Board

What went well
What could be improved
Actions

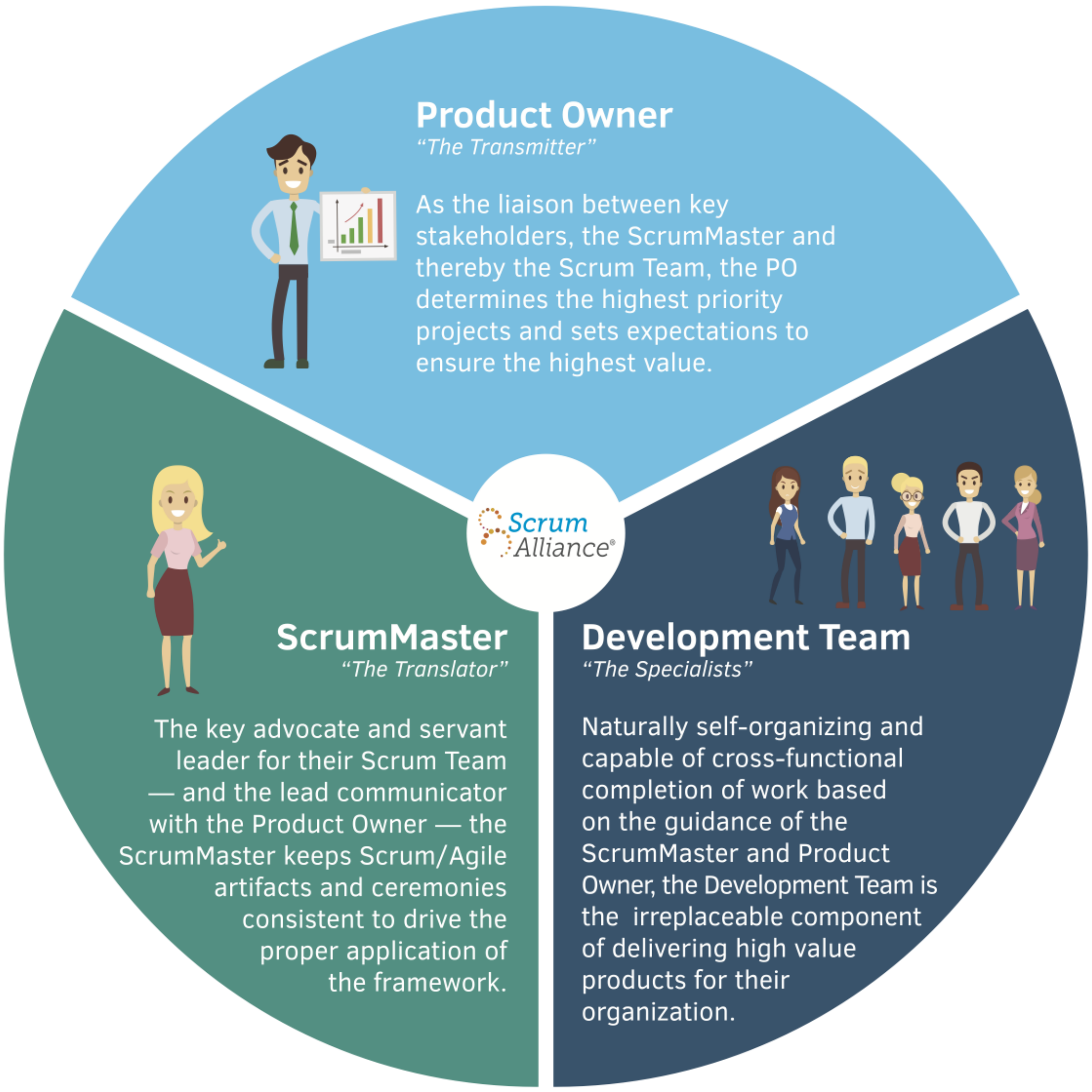
Mood graph retrospective



Starfish retrospective


This is a sample text. Insert your desired text here.
This is a sample text. Insert your desired text here.
Less Of
More Of
Stop Doing
Start Doing
Keep Doing
This is a sample text. Insert your desired text here.
This is a sample text. Insert your desired text here.
This is a sample text. Insert your desired text here.

https://trello.com/b/X4OIHkrQ/parachuting-school-requirements



Product Owner

"The Transmitter"

As the liaison between key stakeholders, the ScrumMaster and thereby the Scrum Team, the PO determines the highest priority projects and sets expectations to ensure the highest value.

ScrumMaster

"The Translator"

The key advocate and servant leader for their Scrum Team — and the lead communicator with the Product Owner — the ScrumMaster keeps Scrum/Agile artifacts and ceremonies consistent to drive the proper application of the framework.

Development Team

"The Specialists"

Naturally self-organizing and capable of cross-functional completion of work based on the guidance of the ScrumMaster and Product Owner, the Development Team is the irreplaceable component of delivering high value products for their organization.