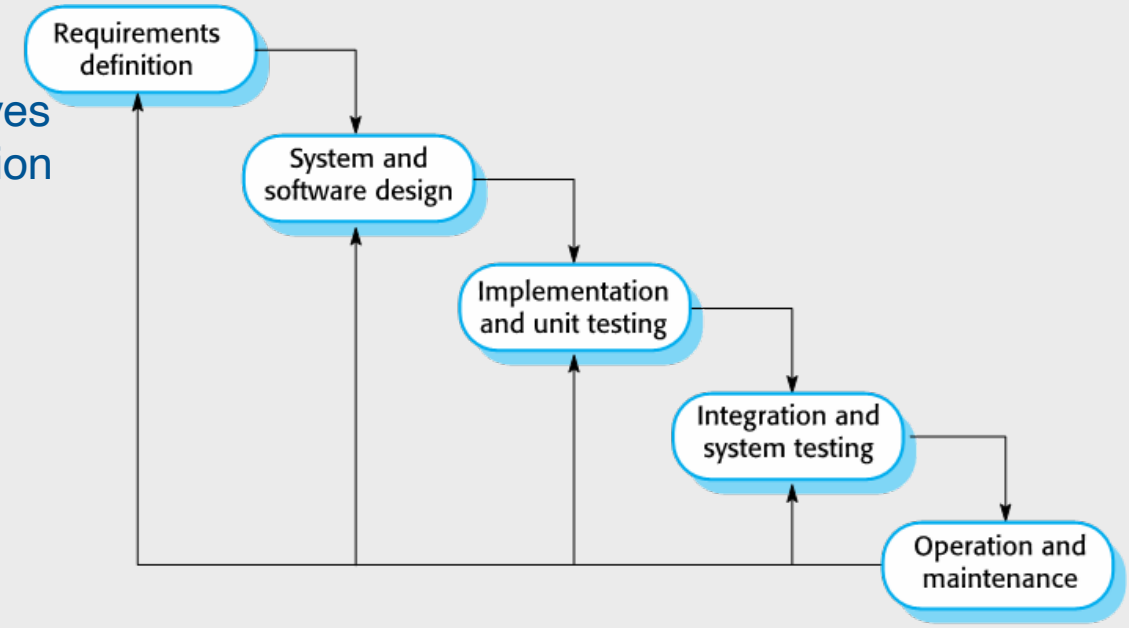
# Agile Software Engineering

# Agile software engineering

- Software products must be brought to market quickly so rapid software development and delivery is essential.
  - Virtually all software products are now developed using an agile approach.
- Agile software engineering is to:
  - delivering functionality quickly
  - responding to changing product specifications
  - minimizing development overheads.
- A large number of ‘agile methods’ have been developed.
  - Examples: XP, Scrum, Kanban, feature-driven development (FDD),…
  - There is no ‘best’ agile method or technique.
  - It depends on who is using the technique, the development team and the type of product being developed

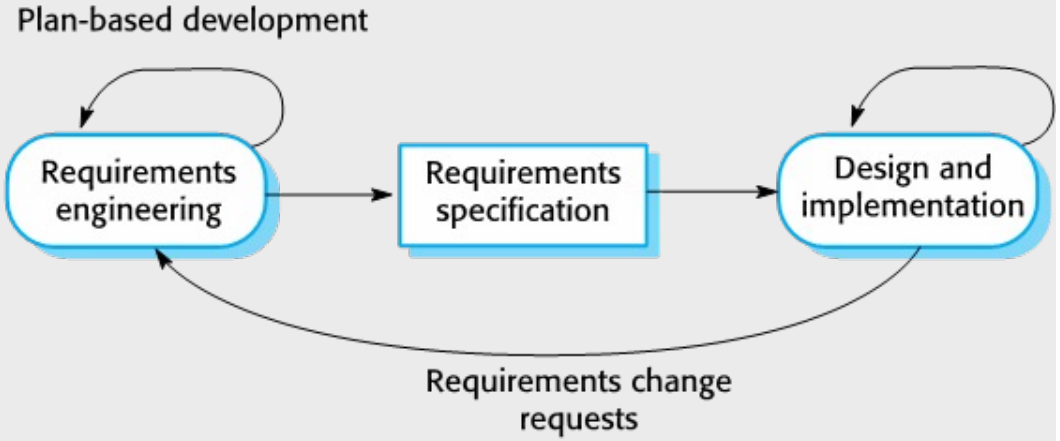
# Conventional Software Engineering: Waterfall Methodology!

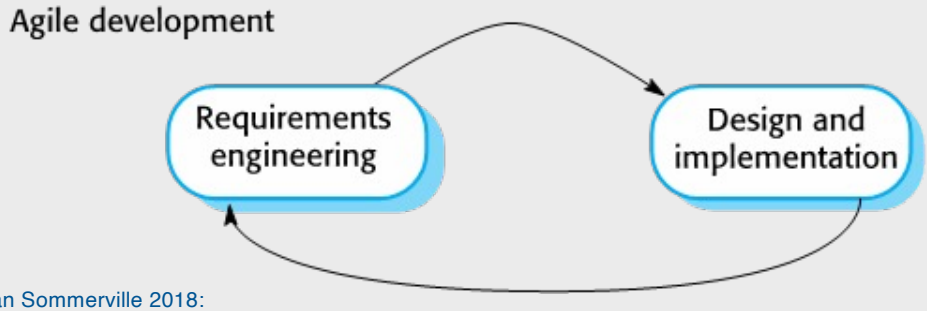
- Plan-driven development was to engineer large, long-lifetime systems
  - Examples: aircraft control systems, banking, insurance, etc.
  - These software are often used for several years (decades!).
  - This approach is based on “sequential” software development processes (aka Waterfall approach) for project planning, requirements specification and analysis and system modelling.
  - Plan-driven approach involves overheads and documentation and it is not rapid!



# Waterfall vs Agile Methodology

- Agile methods were developed in 1990s to address this problem of waterfall methodology.
  - these methods focus on the software rather than its documentation,
  - develop software in a series of increments
  - reduce process bureaucracy



# The agile manifesto

We are uncovering better ways of developing software by doing it and helping others to 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;

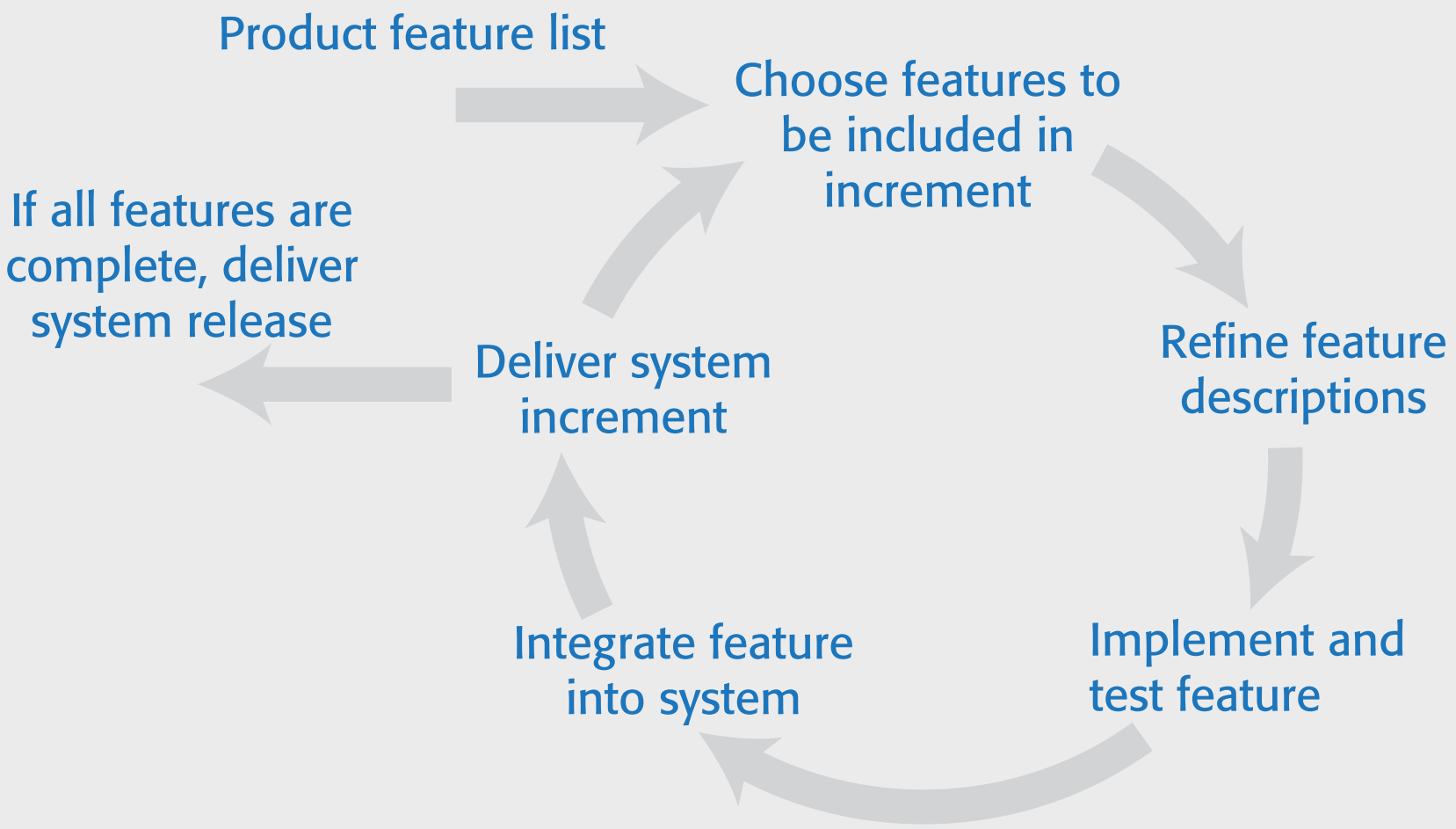
- **responding to change** over following a plan.

While there is value on the items on the right, we value the items on the left more!

# Incremental development

- Agile methods have one common theme: incremental development & delivery.
- Product development focuses on the software features,
  - Each feature **does something** for the software user.
- With incremental development, you start by prioritizing the features
  - The most important features are implemented first.
  - You only define the details of the feature being implemented in an increment.
  - That feature is then implemented and delivered.
- Users or surrogate users can try it out and provide feedback to the developers
- Next, the team will define and implement the next feature of the software

# Incremental development



# Incremental development activities

***Choose features to be included in an increment***

- From the list of features, select those that can be implemented in next increment

***Refine feature descriptions***

- Add detail to the feature descriptions so that the team have a common understanding of each feature and there is sufficient detail to implement it

***Implement and test***

- Implement the feature and develop automated tests for it to show that its behavior is consistent with its description

***Integrate feature and test***

Integrate the developed feature with the existing system and test it to check that it works in conjunction with other features

***Deliver system increment***

- Deliver the system increment to product manager for checking and comments. If enough features have been implemented, release a version of the system

# Agile development principles

***Involve the (surrogate) customer with the development team***

- Customers provide and prioritize features and evaluate each increment of the system.

***Embrace change***

- Expect the features or their details change as the development team and the product manager learn more about it.
- Adapt the software to cope with changes as they are made.

***Develop and deliver incrementally***

- Test and evaluate each increment as it is developed and feed back required changes to the development team.

# Agile development principles (2)

***Maintain simplicity in software development***

- Do what you can to eliminate complexity from the system

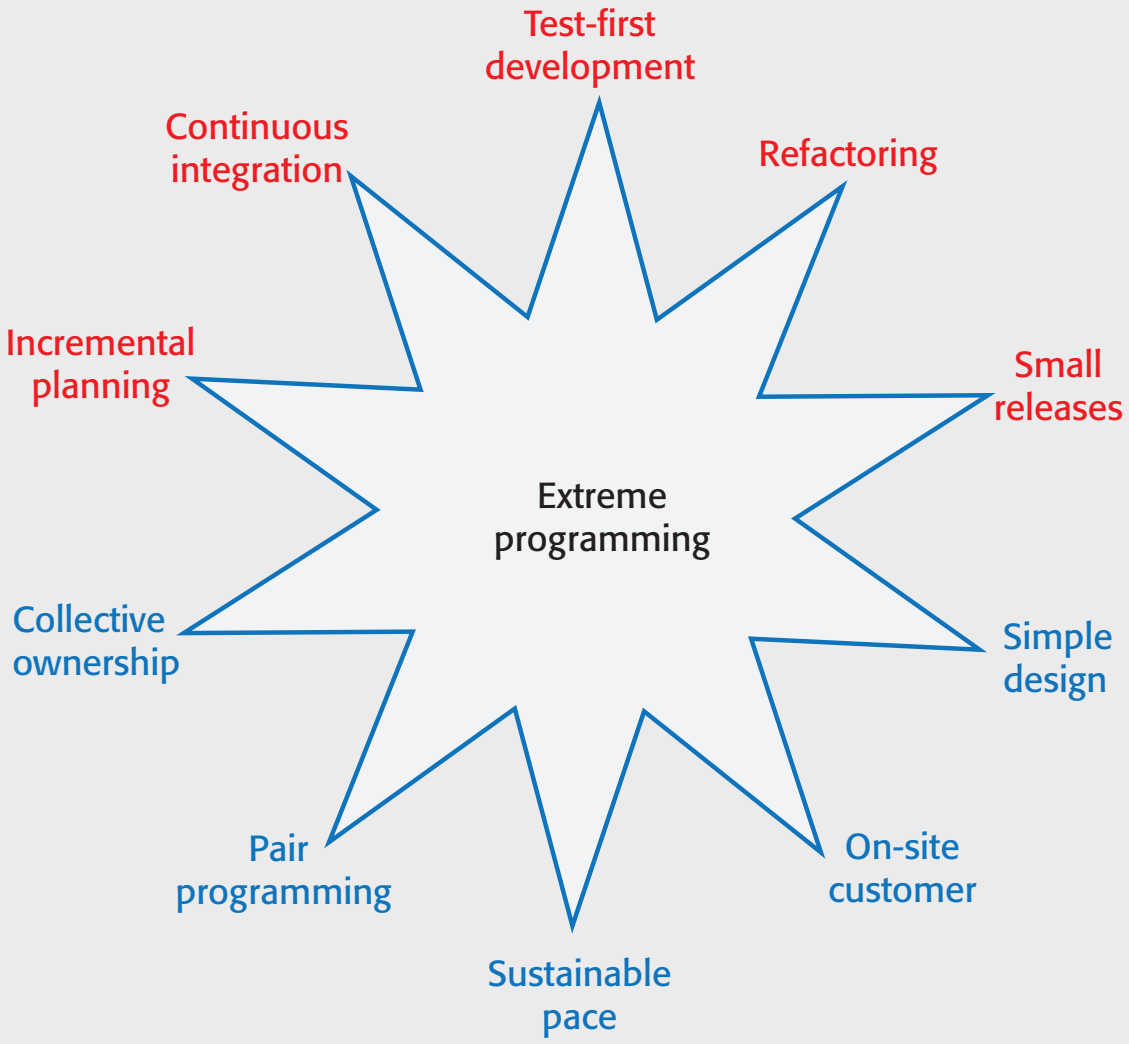
***Focus on people, not things***

- Trust the development team
- Do not expect everyone to do the development in the same way
- Team members should be left to develop their own ways of working without being limited by prescriptive software processes.

# Extreme programming (XP)

- The most influential work that has changed software development culture was the development of XP
- The name was coined by Kent Beck in 1998
- Nomenclature: practice iterative development to ‘extreme’ levels
- XP focused on 12 new development techniques for rapid, incremental software development, change, and delivery
  - Some of these techniques are now widely used; others have been less popular

# Extreme programming (XP) practices



# Widely adopted XP practices

***Incremental planning/user stories***

- There is no 'grand plan' for the system. Instead, what needs to be implemented (the requirements) in each increment are established in discussions with a customer representative. The requirements are written as user stories. The stories to be included in a release are determined by the time available and their relative priority.

***Small releases***

- The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the previous release.

***Test-driven development***

- Instead of writing code then test it, developers write the tests first!
- This helps clarify what the code should actually do and there is always a 'tested' version of the code available.
- An automated unit test framework is used to run the tests after every change. New code should not 'break' the existing code implemented.

# Widely adopted XP practices

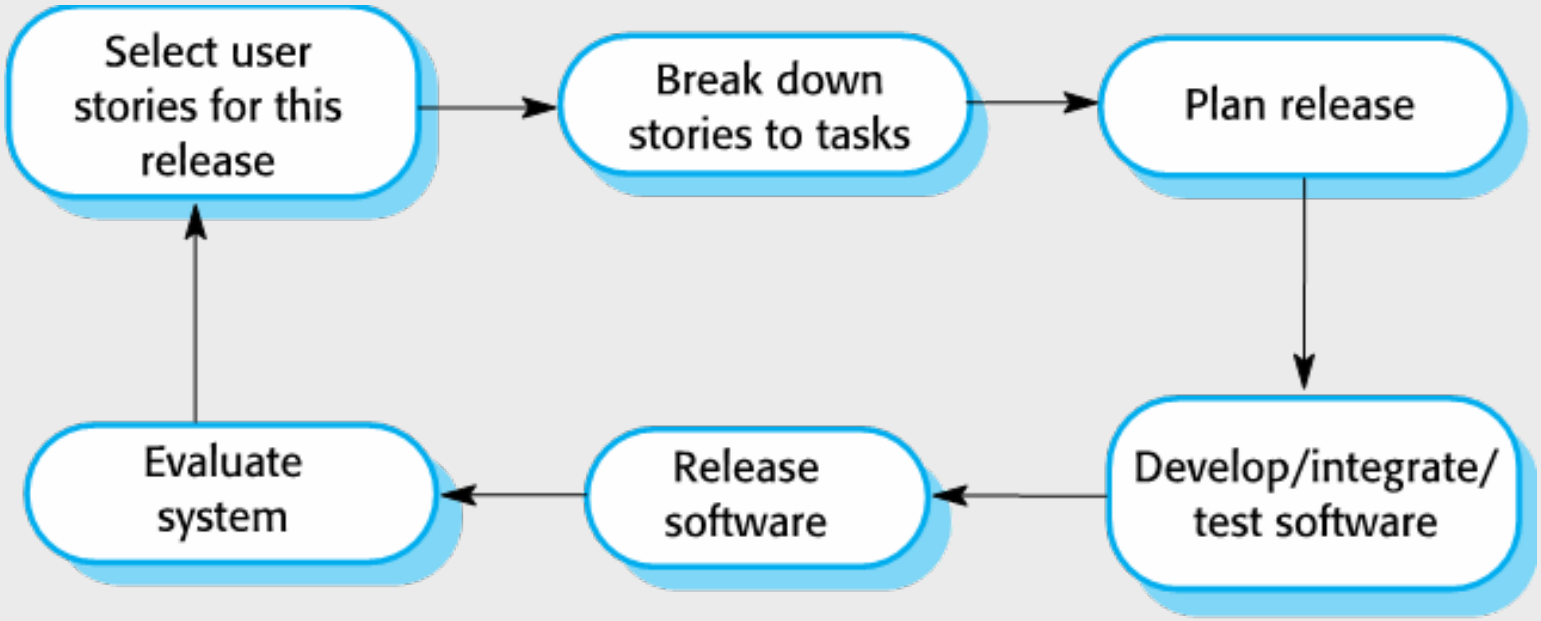
***Continuous integration***

- As soon as the work on a task is complete, it is integrated into the whole system and a new version of the system is created.
- All unit tests from all developers are run automatically and must be successful before the new version of the system is accepted.

***Refactoring***

- Refactoring means improving the structure, readability, efficiency and security of a program.
- All developers are expected to refactor the code as soon as potential code improvements are found.
- This keeps the code simple and maintainable.

# XP release cycle



# Scrum Agile Software Engineering Methodology

- Software company managers need to know the cost and development time to be brought to market.
  - Plan-driven development (waterfall) provides time and cost of s/w development through long-term development plans.
  - Plans always change so **anything apart from short-term plans are unreliable.**
- Scrum is an agile method that provides a framework for agile project organization and planning.
  - Scrum DOES NOT focus/mandate any specific technical practices.
- **Scrum vs XP**:
  Scrum focuses on project management and teamwork, whereas, XP focuses on code quality and individual programmers' work.

# Scrum terminology

***Product***
The software product that is being developed by the Scrum team.

***Product owner***
A team member who is responsible for identifying product features and attributes. They review work done and help to test the product.

***Product backlog***
A to-do list of items such as bugs, features and product improvements that the Scrum team have not yet completed.

***Development team***
A small self-organizing team of five to eight people who are responsible for developing the product.

***Sprint***
A short period, typically two to four weeks, when a product increment is developed.

# Scrum terminology

***Scrum***
A daily team meeting where progress is reviewed and work to be done that day as discussed and agreed.

***ScrumMaster***
A team coach who guides the team in the effective use of Scrum

***Potentially shippable product increment***
The output of a sprint which should be of high enough quality to be deployed for customer use.

***Velocity***
An estimate of how much work a team can do in a single sprint.

# Key roles in Scrum

- ***The Product Owner*** is responsible for ensuring that the development team is always focused on the product they are building rather than diverted into technically interesting but less relevant work.
  - The product manager normally takes on the Product Owner role.
- **The ScrumMaster** is a Scrum expert whose job is to guide the team in the effective use of the Scrum method.
  - ScrumMaster is not a conventional project manager, but is a coach for the team. They have authority within the team on how Scrum is used.
  - In many companies that use Scrum, the ScrumMaster also has some project management responsibilities.

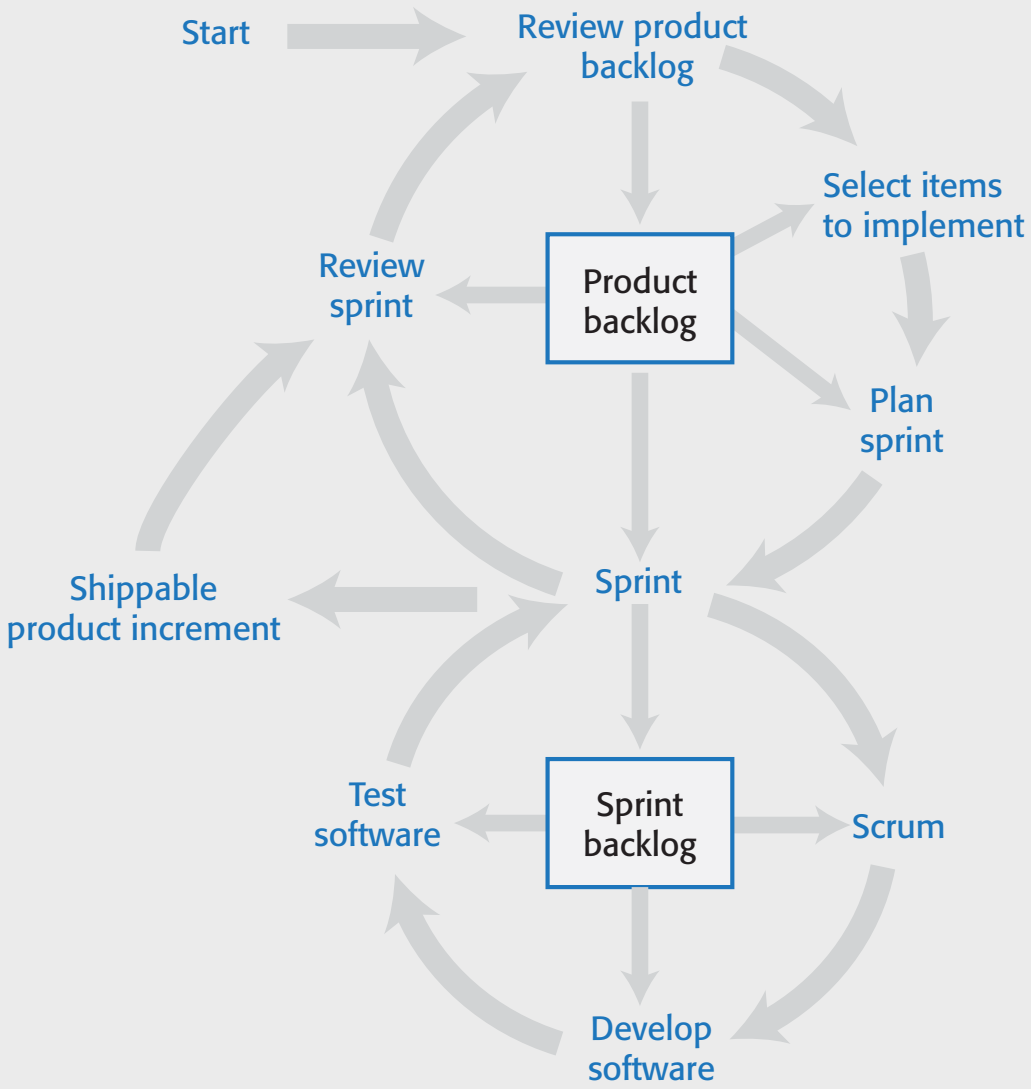
# Scrum and sprints

- In Scrum method, software is developed in sprints, which are fixed-length periods (2 - 4 weeks) in which software features are developed and delivered.

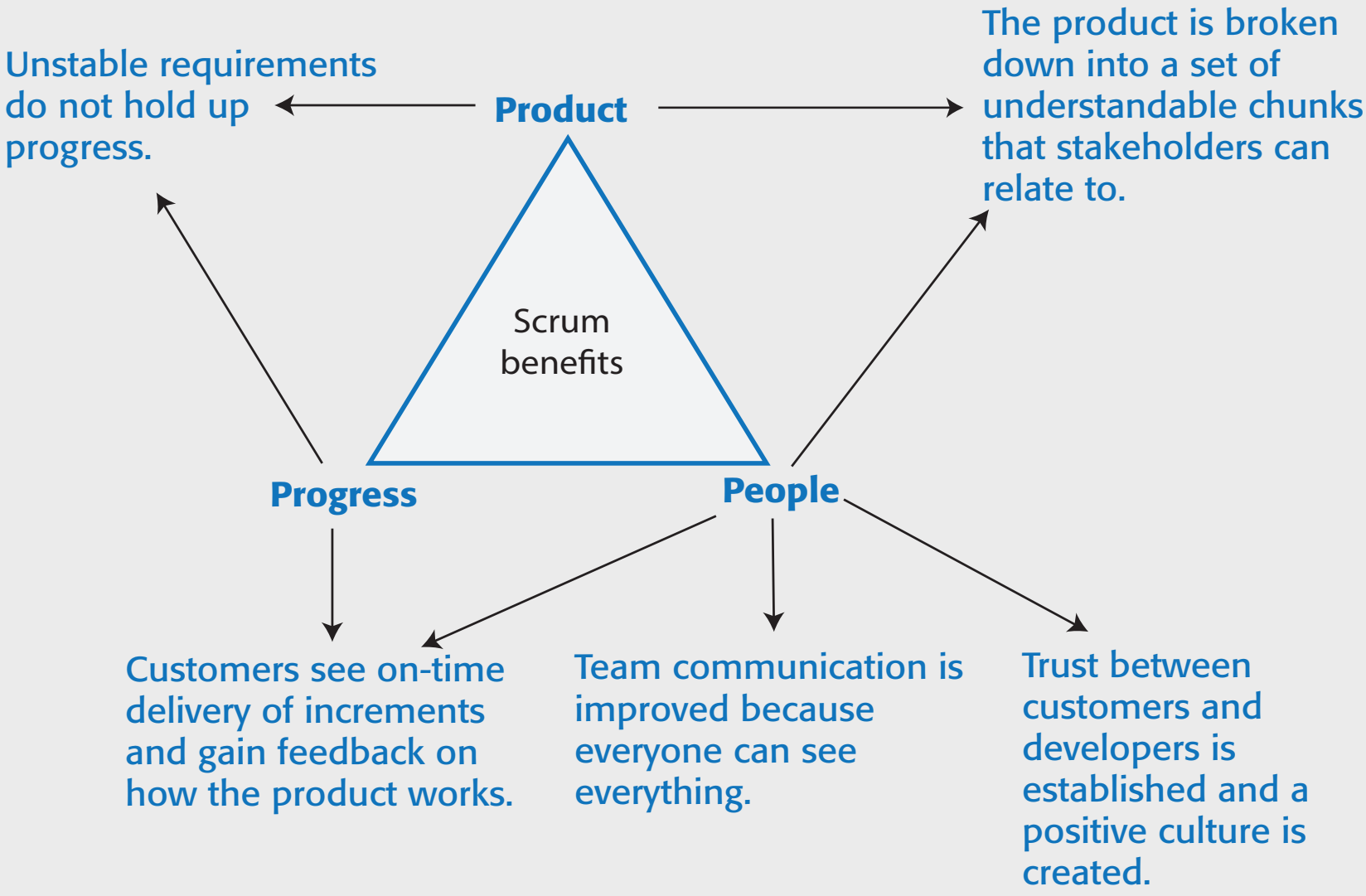
- During a sprint, the team has daily meetings (Scrums) to review progress and to update the list of work items that are incomplete.

- Sprints should produce a ‘shippable product increment’. This means that the developed software should be complete and ready to deploy.

# Scrum cycles



# Main benefits of using Scrum



# Key Scrum practices

- ***Product backlog***
  This is a to-do list of items to be implemented that is reviewed and updated before each sprint.

- ***Timeboxed sprints***
  Fixed-time (2-4 week) periods in which items from the product backlog are implemented,

- ***Self-organizing teams***
  Self-organizing teams make their own decisions and work by discussing issues and making decisions by consensus.

# Product backlogs (user stories)

- The product backlog is a list of what needs to be done to complete the development of the product.
- The items on this list are called product backlog items (PBIs).
- The product backlog may include a variety of different items
  - product features to be implemented
  - user requests,
  - essential development activities
  - desirable engineering improvements.
- The product backlog must be prioritized, so that the items that be implemented first are at the top of the list.

# How to Write User Stories (backlog)

- General format of user stories: ***As a ...., I want to ....., so that.....***
  - *Firt blank is filled with the role. Example: buyer, seller, platform (amazon) owner*
  - *Second blank is filled with the goal*
  - *The third blank is filled with a reason*
- User stories must be:
  - non-technical and from the user perspective
  - independent (from other stories) and self-sufficient
  - **Valuable** (meaningful) stories: add a value to the project
  - negotiable (ie, can be changed at any time upon need)
  - estimable- must be descriptive and small, so we can estimate the work for it.
  - User stories must be small and testable
- Each user story must have one or more **acceptance criteria**
  - Write one or more sentences explaining the minimum requirements that makes this story done
- **Spikes**: items in the backlog that are NOT user-facing.
  - Ex: setting up a development environment, setup servers and databases, etc.
  - These are NOT user stories!

# Examples of product backlog (user stories) items

1. As a teacher, I want to be able to configure the group of tools that are available to individual classes (feature)
#Estimation of efforts… #Acceptance criteria….

2. As a parent, I want to be able to view my children's work and the assessments made by their teachers. (feature)
#Estimation of efforts… #Acceptance criteria….

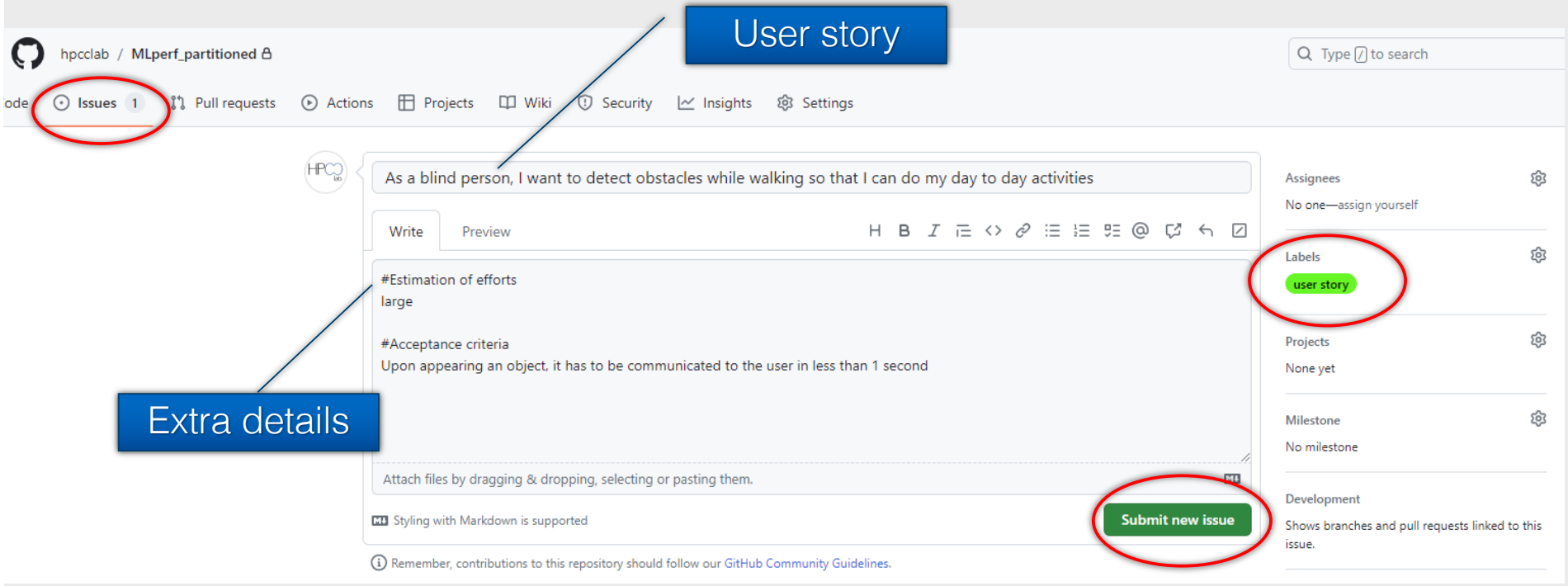
3. As a teacher of young children, I want a pictorial interface for children with limited reading ability. (user request)

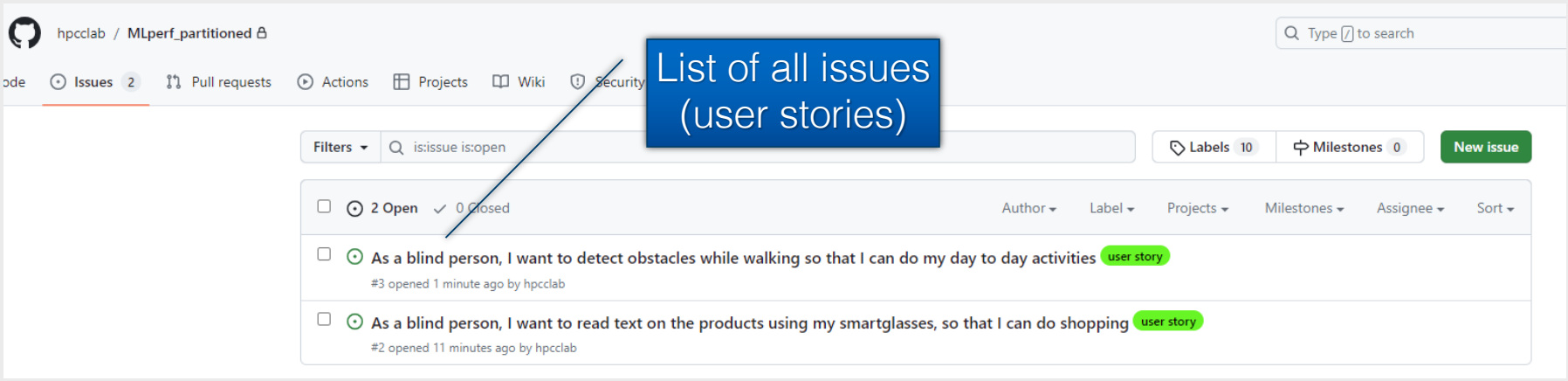
4. Establish criteria for the assessment of open-source software as a basis for parts of this system. (development activity) → spike

5. Refactor user interface code to improve understandability and performance. (engineering improvement) → spike

6. Implement encryption for all personal user data. (engineering improvement) → spike

# Defining User Stories in Github



# Product backlog item states

***Ready for consideration***

These are high-level ideas and feature descriptions that will be considered for inclusion in the product. They are tentative so may radically change or may not be included in the final product.

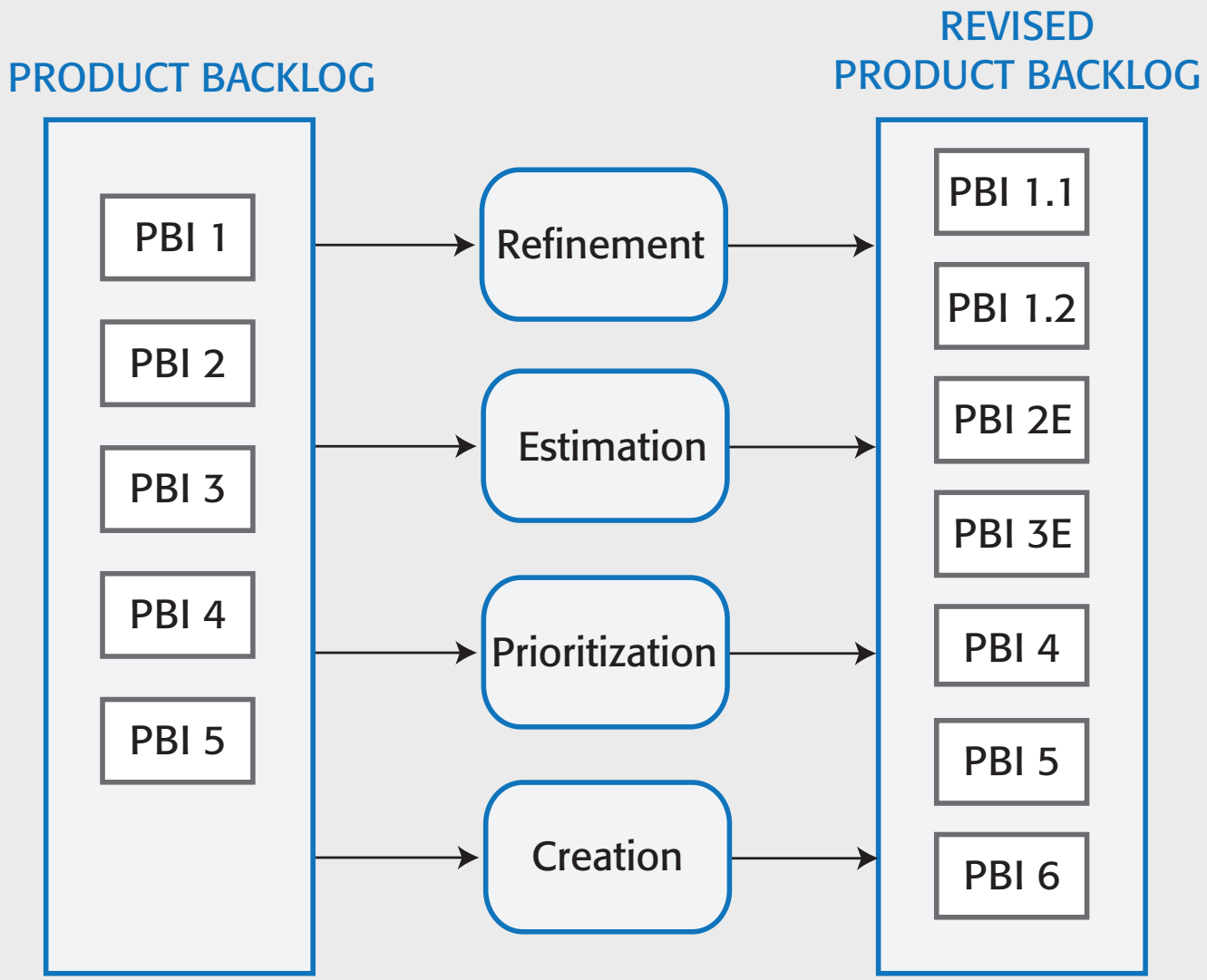
***Ready for refinement***

- The team has agreed that this is an important item that should be implemented as part of the current development.
- There is a reasonably clear definition of what is required. However, work is needed to understand and refine the item.

***Ready for implementation***

- The PBI has enough detail for the team to estimate the effort involved and to implement the item. Dependencies on other items have been identified.

# Product backlog activities



# Product backlog activities

- ***Refinement***
  Existing PBIs are analyzed and refined to create more detailed PBIs. This may lead to the creation of new product backlog items.

- **Estimation**
  The team estimate the amount of work required to implement a PBI and add this assessment to each analyzed PBI.

- **Creation**
  New items are added to the backlog. These may be new features suggested by the product manager, required feature changes, engineering improvements, or process activities such as the assessment of development tools that might be used.

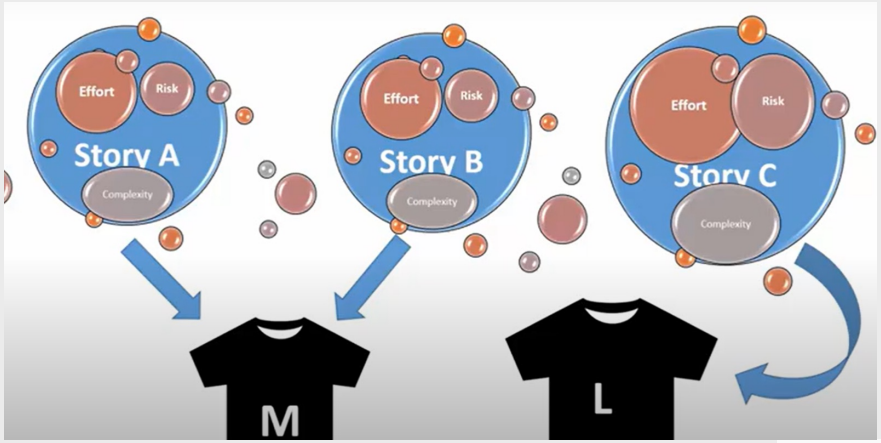
- **Prioritization**
  The product backlog items are reordered to take new information and changed circumstances into account.

# PBI estimation metrics

- Effort required
  - This may be expressed in person-hours or person-days i.e. the number of hours or days it would take one person to implement that PBI.
  - This is not the same as calendar time. Several people may work on an item, which may shorten the calendar time required.
- Story points
  - Story points are an arbitrary estimate of the effort involved in implementing a PBI, taking into account the size of the task, its complexity, the technology that may be required and the ‘unknown’ characteristics of the work.
  - Story points are estimated relatively. The team agrees on the story points for a baseline task and other tasks are estimated by comparison with this e.g. more/less complex, larger/smaller etc.

# How to estimate the work a user story takes?

- Amazon approach: **T-shirt sizing** (small, medium, large, xlarge,…)




- **Poker planner**:
  - use cards based on Fibonacci sequence across the team
  - read the user story and ask everyone to show cards at the same time
  - ask outliers to explain their position.
  - do this until the team reaches a consensus

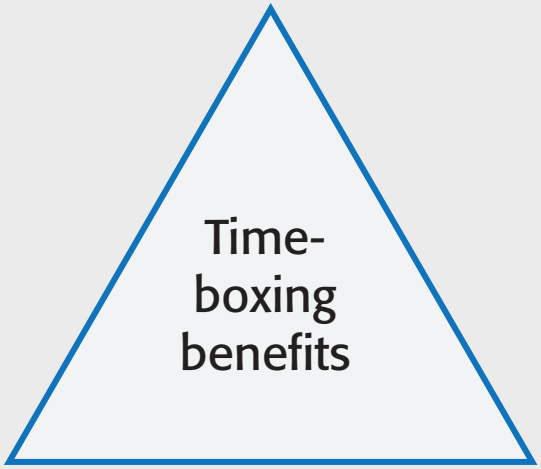
# Timeboxed sprints

- Products are developed in a series of sprints, each of which delivers an increment of the product or supporting software.
- Sprints are short duration activities (2-4 weeks or sometimes up to 6 weeks) and take place between a defined start and end date.
- Sprints are timeboxed, *which means that development stops at the end of a sprint whether or not the work has been completed*.
- During a sprint, the team work on the tasks of the certain backlog items.

# Benefits of using timeboxed sprints

There is a tangible output (usually a software demonstrator) that can be delivered at the end of every sprint.

**Demonstrable progress**




**Problem discovery**

If errors and omissions are discovered the rework required is limited to the duration of a sprint.

**Work planning**

The team develops an understanding of how much work they can do in a fixed time period.

# Sprint activities

***Sprint planning***

- Work items to be completed in that sprint are selected and refined to create a sprint backlog.
- This should not last more than a day at the beginning of the sprint.

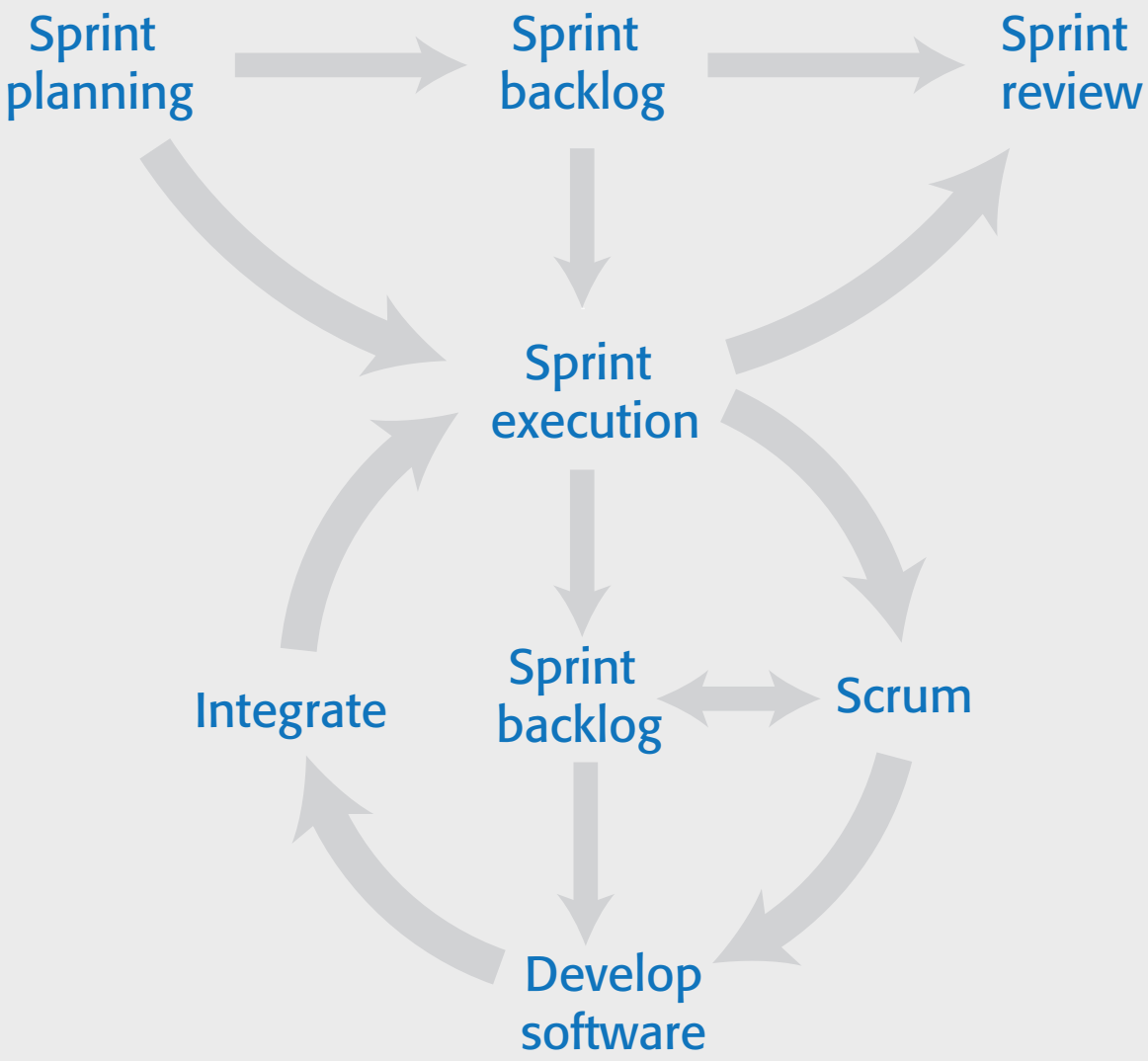
***Sprint execution***

- The team work to implement the sprint backlog items that have been chosen for that sprint.
- If it is impossible to complete all of the sprint backlog items, the sprint is not extended.
- The unfinished items are returned to the product backlog & queued for a future sprint.

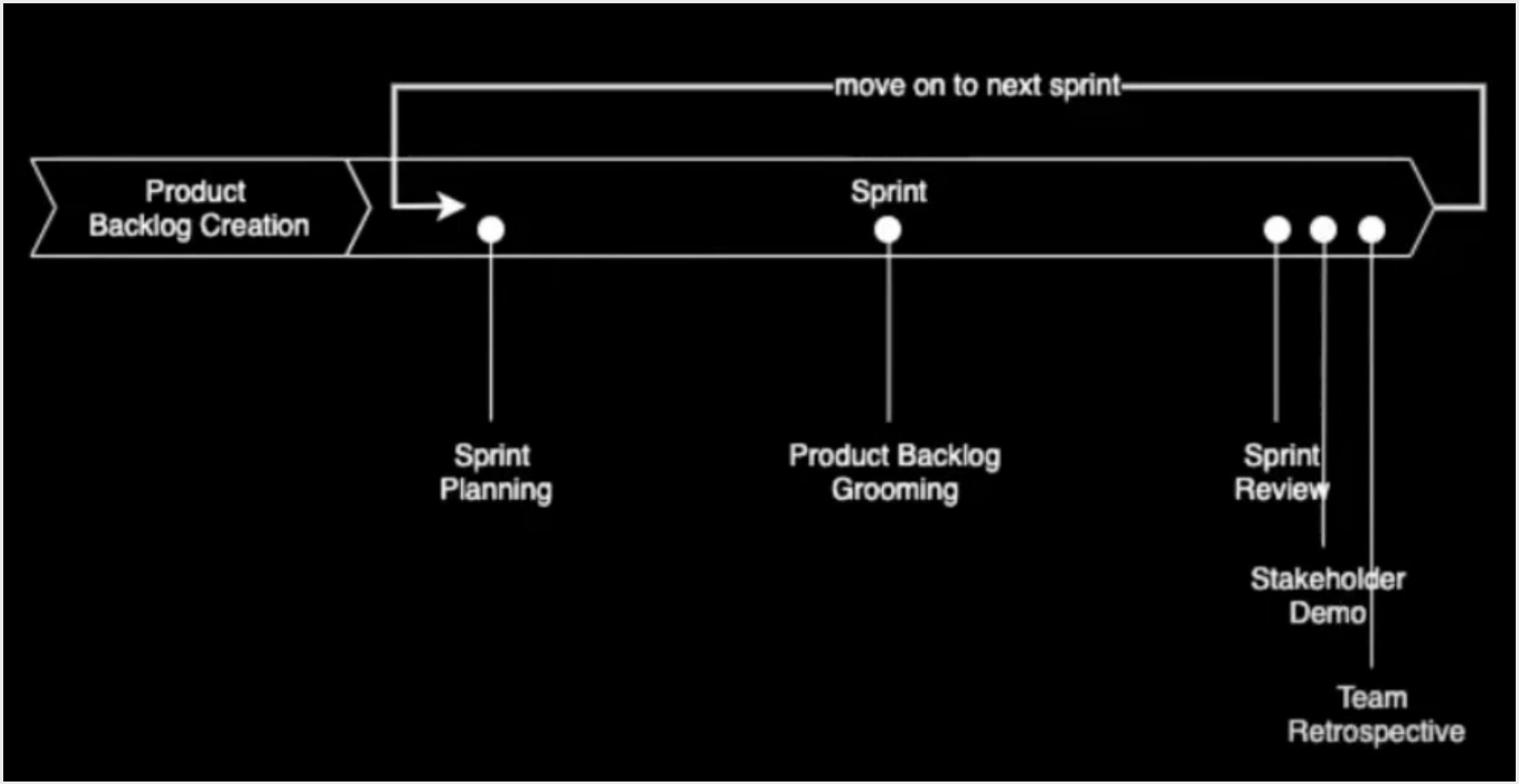
***Sprint reviewing***

- The work done in the sprint reviewed by the team and external stakeholders.
- The team reflect on what went well and what went wrong during the sprint with a view to improving their work process.

# Sprint activities



# Sprint Timeline



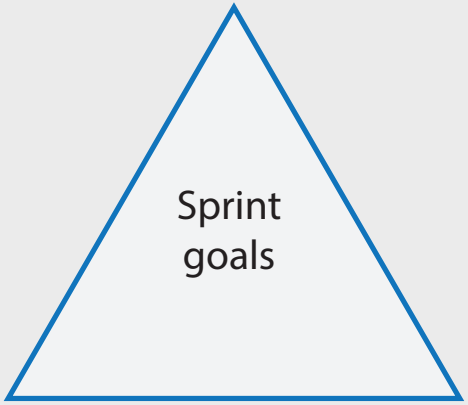
# Sprint planning

- Establish an agreed "sprint goal"
  - Sprint goals may be focused on software functionality, support or performance and reliability.
- Decide on the list of items from the product backlog that should be implemented
- Create a sprint backlog
  - This is a more detailed version of the product backlog that records the work to be done during the sprint in form of **tasks**

# Sprint goals: types and examples

Implement user roles so that a user can select their role when they login to the system

**Functional**




**Support**

Develop analytics that maintain information about the time users spend using each feature of the system.

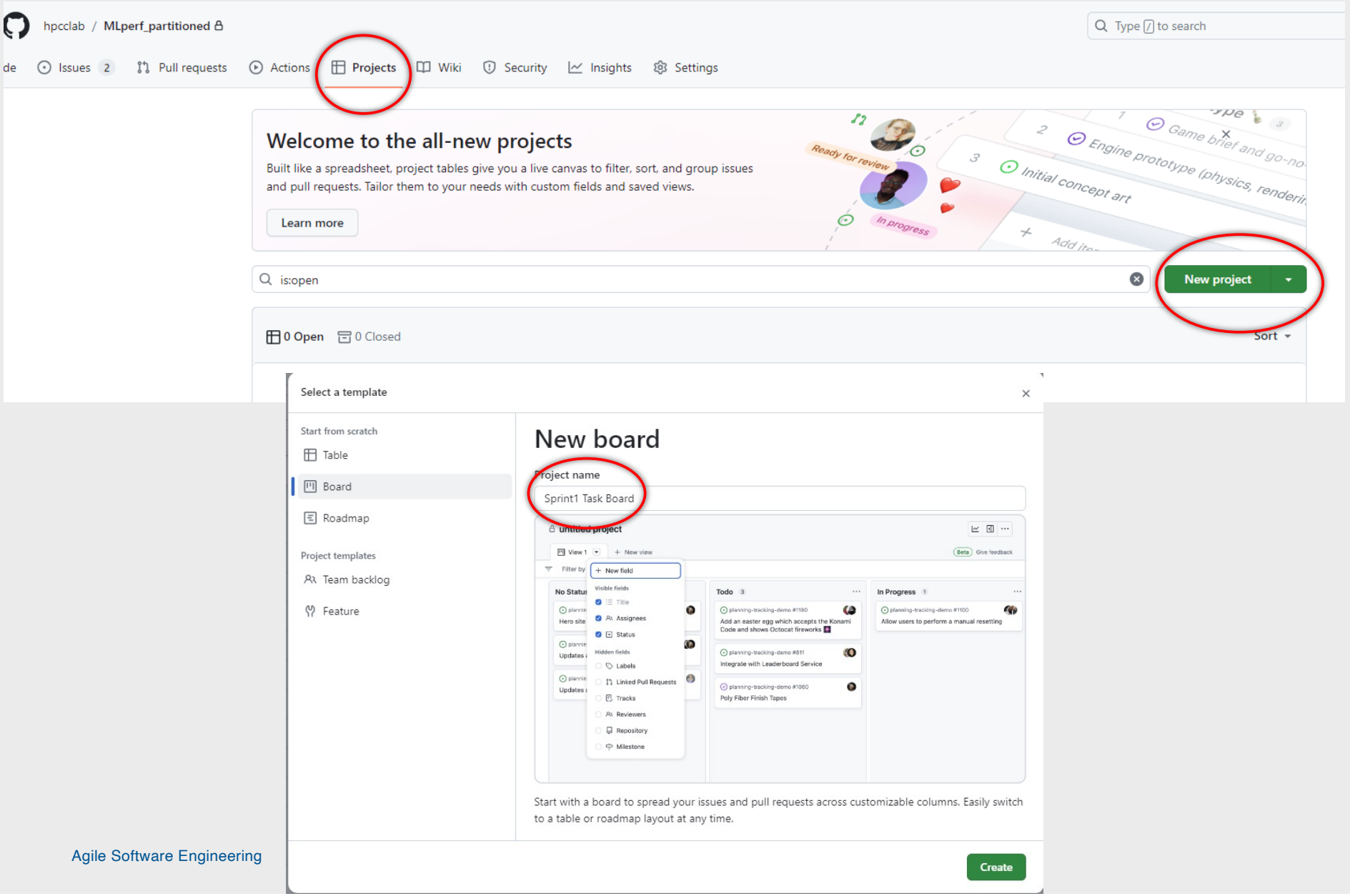
**Performance and reliability**

Ensure that the login response time is less than 10 seconds for all users where there are up to 2000 simultaneous login connections.

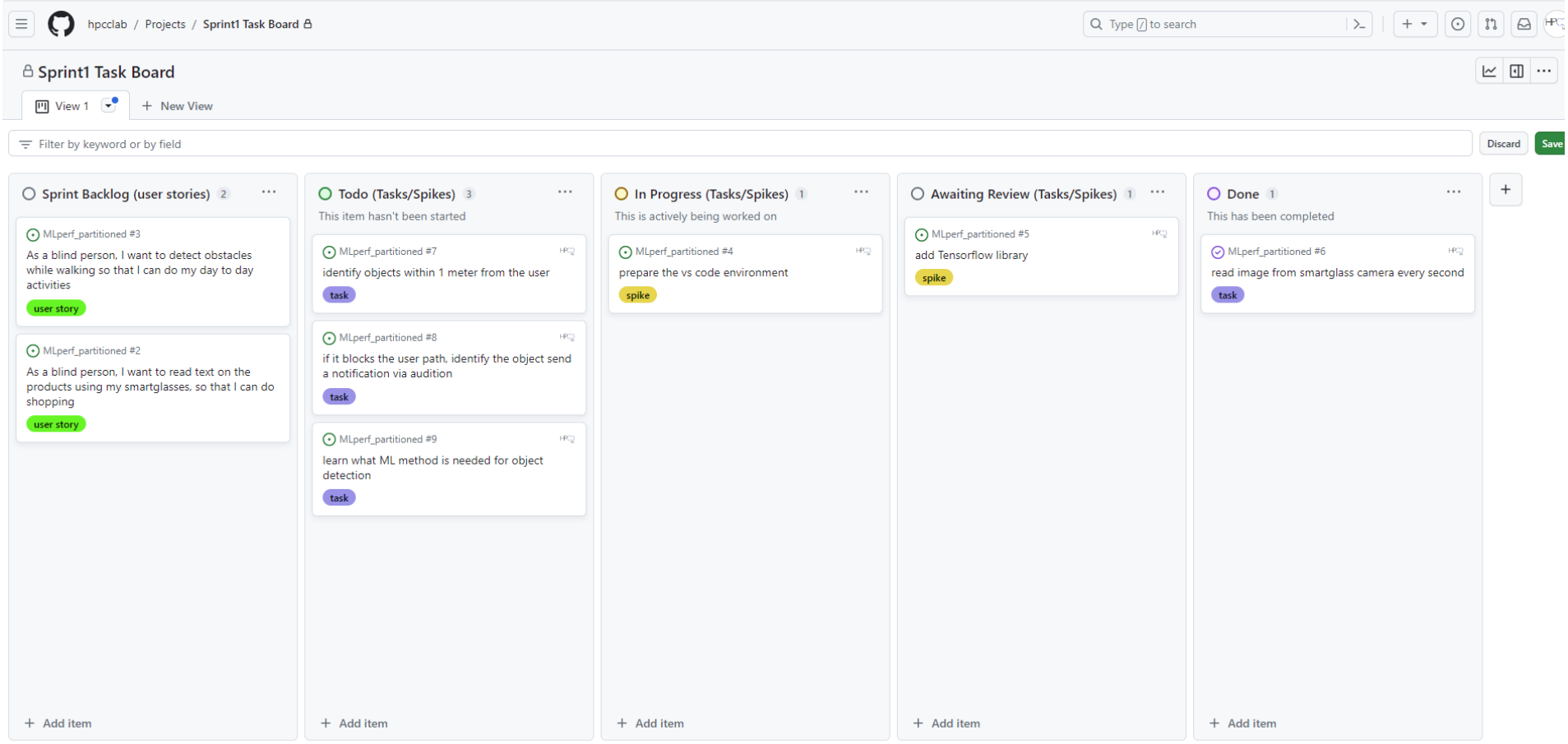
# Sprint planning

- In a sprint plan, the team decides which items in the product backlog should be implemented during that sprint
  - Key inputs are the effort estimates associated with PBIs and the team's velocity
- The output of the sprint planning process is a **sprint backlog**
  - The sprint backlog is a breakdown of a PBI to show what is involved in implementing it during that sprint. → PBI is broken to discrete tasks
  - A task is generally a unit of work that is doable by one developer in one day
  - If you think a task takes more than one day → break it to smaller tasks!
  - Every task **must be** assigned to a team-member! (no orphan tasks)
  - Many tools to manage sprint tasks: Github projects, Trello, etc.
- During a sprint, the team have daily meetings (scrums) to coordinate their work.

# Sprint Tasks in Github



# Sprint Tasks in Github



# Scrums

- A scrum is a short, daily meeting held at the beginning of the day. During a scrum,
  - All team members share information:
    - progress since the previous day's scrum,
    - problems that have arisen
    - plans for the coming day
  - The sprint backlog is reviewed: Completed items are removed from it. New items may be added to the backlog as new information emerges.
  - Then decide who should work on sprint backlog items that day.
- Thus, everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them.
- Scrum meetings should be short (5 minutes) and focused.
  - To dissuade team members from getting involved in long discussions, they are sometimes organized as 'stand-up' meetings where there are no chairs in the meeting room!

# Agile activities

- Scrum is not about technical agile activities that should be used.
- However, two practices that should always be used in a sprint:
  - ***Test automation***
    - As far as possible, product testing should be automated.
    - You should develop a suite of executable tests that can be run at any time.
  - ***Continuous integration***
    When someone changes to the software components, these components should be immediately integrated with other components to create a system.

    This system should then be tested to check for unanticipated component interaction problems.

# Code completeness checklist

***Reviewed***
The code has been reviewed by another team member who has checked that it meets agreed coding standards, is understandable, includes appropriate comments, and has been refactored if necessary.

***Unit tested***
All unit tests have been run automatically and all tests have executed successfully.

***Integrated***
The code has been integrated with the project codebase and no integration errors have been reported.

***Integration tested***
All integration tests have been run automatically and all tests have executed successfully.

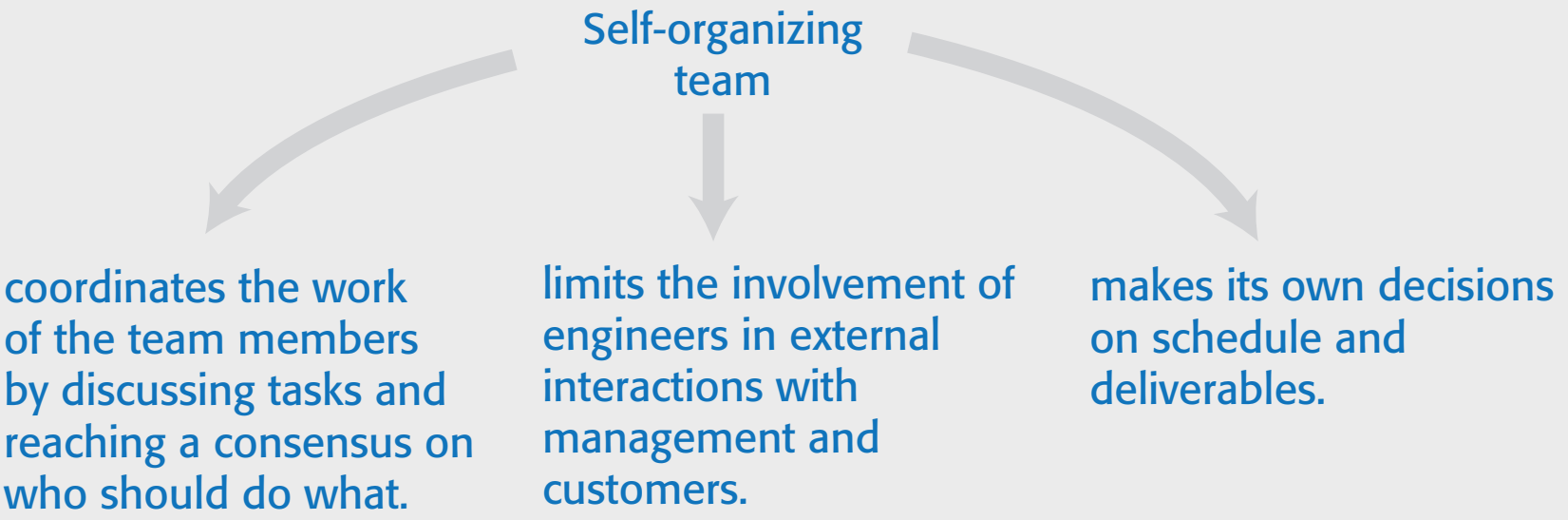
***Accepted***
Acceptance tests have been run if appropriate and the product owner or the development team have confirmed that the product backlog item has been completed.

# Sprint reviews

- At the end of each sprint, there is a review meeting, which involves the whole team. This meeting:
  - reviews whether or not the sprint has met its goal.
  - sets out any new problems and issues that have emerged during the sprint.
  - is a way for a team to reflect on how they can improve the way they work.

- The product owner/manger has the ultimate authority to decide whether or not the goal of the print has been achieved!
  - They should confirm that the implementation of the selected product backlog items is complete.

# Self-organizing teams



# Team size and composition

- The ideal Scrum team size is between 5 and 8 people with different levels of experience.
  - Teams have to tackle diverse tasks and so usually require people with different skills, such as networking, user experience, database design and so on.
  - A team of 5-8 people is large enough to be diverse yet small enough to communicate informally and effectively and to agree on the team priorities
- The advantage of a self-organizing team is that it can be a cohesive team that can adapt to change.
  - Because the team (rather than individuals) take responsibility for the work, they can cope with people leaving and joining the team.
  - Good team communication means that team members inevitably learn something about each other's areas

# Team coordination

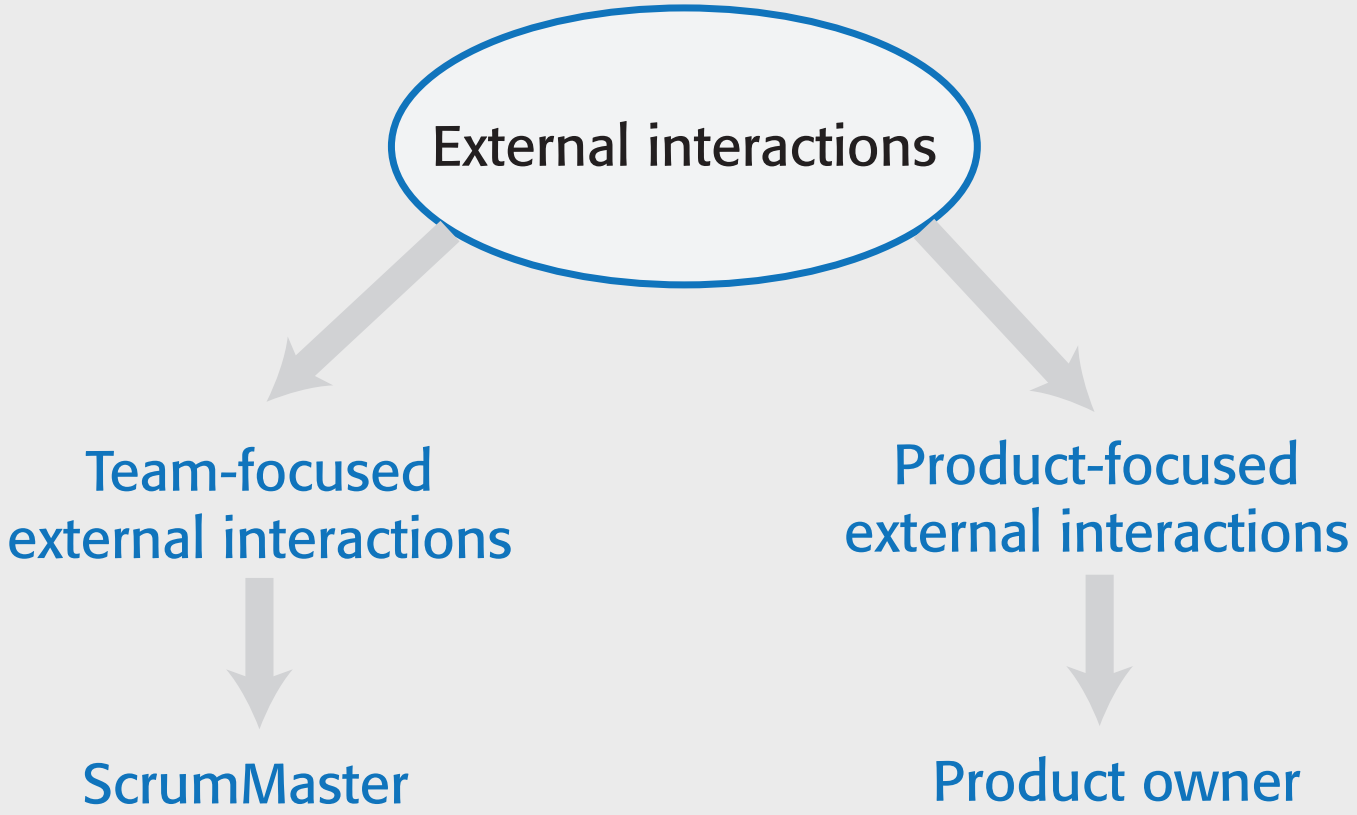
- The developers of Scrum assumed that teams would be co-located (in the same room) and could communicate informally.
  - Daily scrums mean that the team members know what's been done and what others are doing.

- However, the use of daily scrums as a coordination mechanism is based on two assumptions that are not always correct:
  - Team members may be part-time and may work in different places (remote).
  - Some team members may work flexible hours (e.g. because of childcare responsibilities) or may work on several projects at the same time.

# External interactions

- External interactions are interactions that team members have with people outside of the team.
- In Scrum, developers should focus on development and only the ScrumMaster and Product Owner should be involved in external interactions.
- The intention is that the team should be able to work on software development without external interference or distractions.

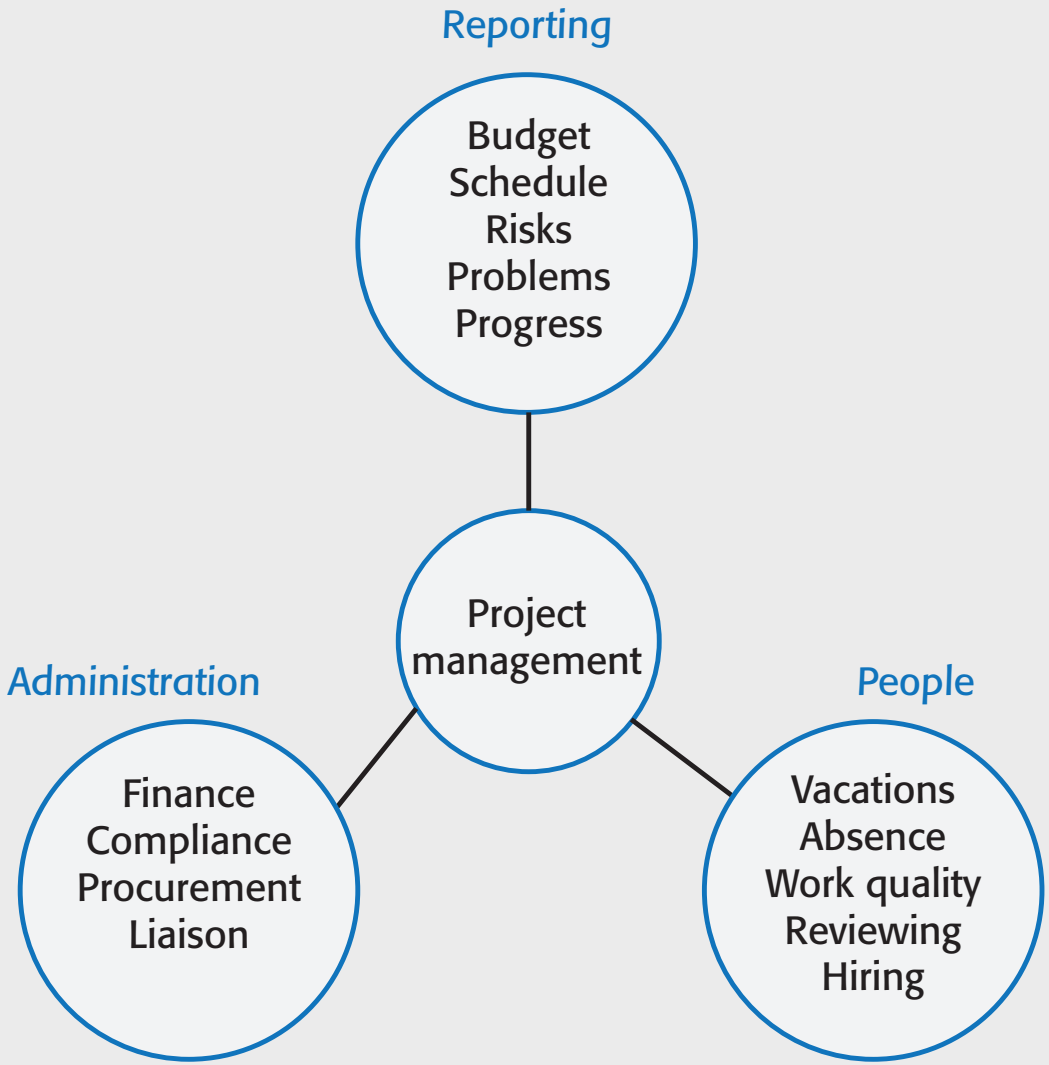
# Managing external interactions



# Project management

- There is a need for development teams to report on progress to company management.
- A self-organizing team has to appoint someone to take on these responsibilities.
  - Because of the need to maintain continuity of communication with people outside the group, rotating these activities around team members is not viable.
- The developers of Scrum did not envisage that the ScrumMaster should also have project management responsibilities.
  - In many companies, however, the ScrumMaster has to take on project management responsibilities.
  - They know the work going on and are in the best position to provide accurate information and project plans and progress.

# Project management responsibilities



# Key points 1

- The best way to develop software products is to use agile software engineering methods that are geared to rapid product development and delivery.
- Agile methods are based around iterative development and the minimization of overheads during the development process.
- Extreme programming (XP) is an influential agile method that introduced agile development practices such as user stories, test-first development and continuous integration. These are now mainstream software development activities.
- Scrum is an agile method that focuses on agile planning and management. Unlike XP, it does not define the engineering practices to be used. The development team may use any technical practices that they believe are appropriate for the product being developed.
- In Scrum, work to be done is maintained in a product backlog – a list of work items to be completed. Each increment of the software implements some of the work items from the product backlog.

# Key points 2

- Sprints are fixed-time activities (usually 2–4 weeks) where a product increment is developed. Increments should be ‘potentially shippable’ i.e. they should not need further work before they are delivered.
- A self-organizing team is a development team that organizes the work to be done by discussion and agreement amongst team members.
- Scrum practices such as the product backlog, sprints and self-organizing teams can be used in any agile development process, even if other aspects of Scrum are not used.