

Application programming in engineering; Waterfall vs. Agile and Scrum approach

15.1.2019

Reading material:

- Book: Ian Sommerville, Software Engineering, 10th edition, chapters 2-3.
- Slides of the book:
<http://iansommerville.com/software-engineering-book/slides/>

Content

- Software process models
- Process activities
- Agile development

What is a software process?

A structured set of activities required to develop a software system.

What are the activities of software process and what is their purpose?

Many different software processes but all involve:

- Specification – defining what the system should do;
- Design and implementation – defining the organization of the system and implementing the system;
- Validation – checking that it does what the customer wants;
- Evolution – changing the system in response to changing customer needs.

- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.
- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- Process descriptions may also include:
 - Products, which are the outcomes of a process activity;
 - Roles, which reflect the responsibilities of the people involved in the process;
 - Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

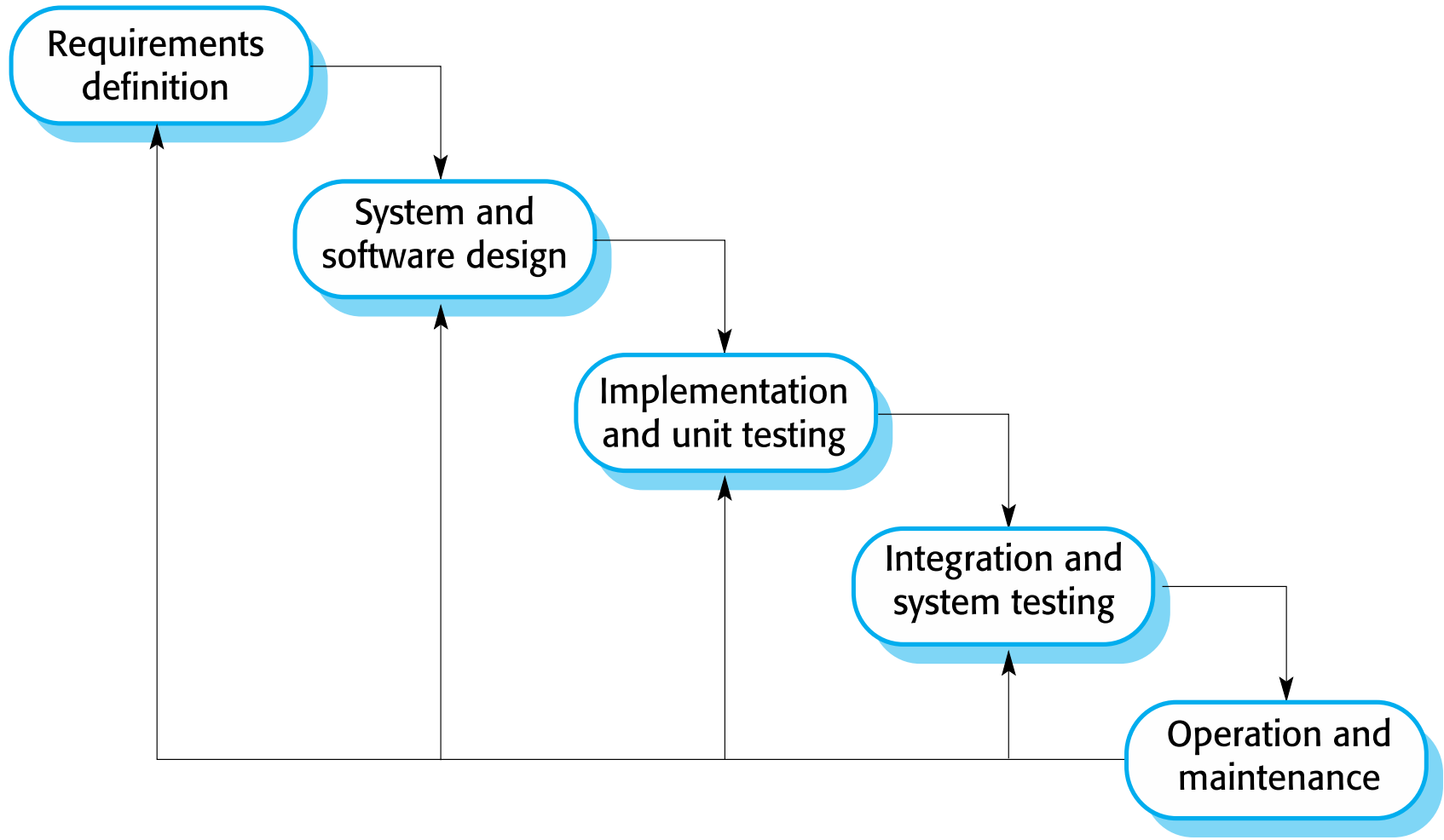
Plan-driven and agile processes

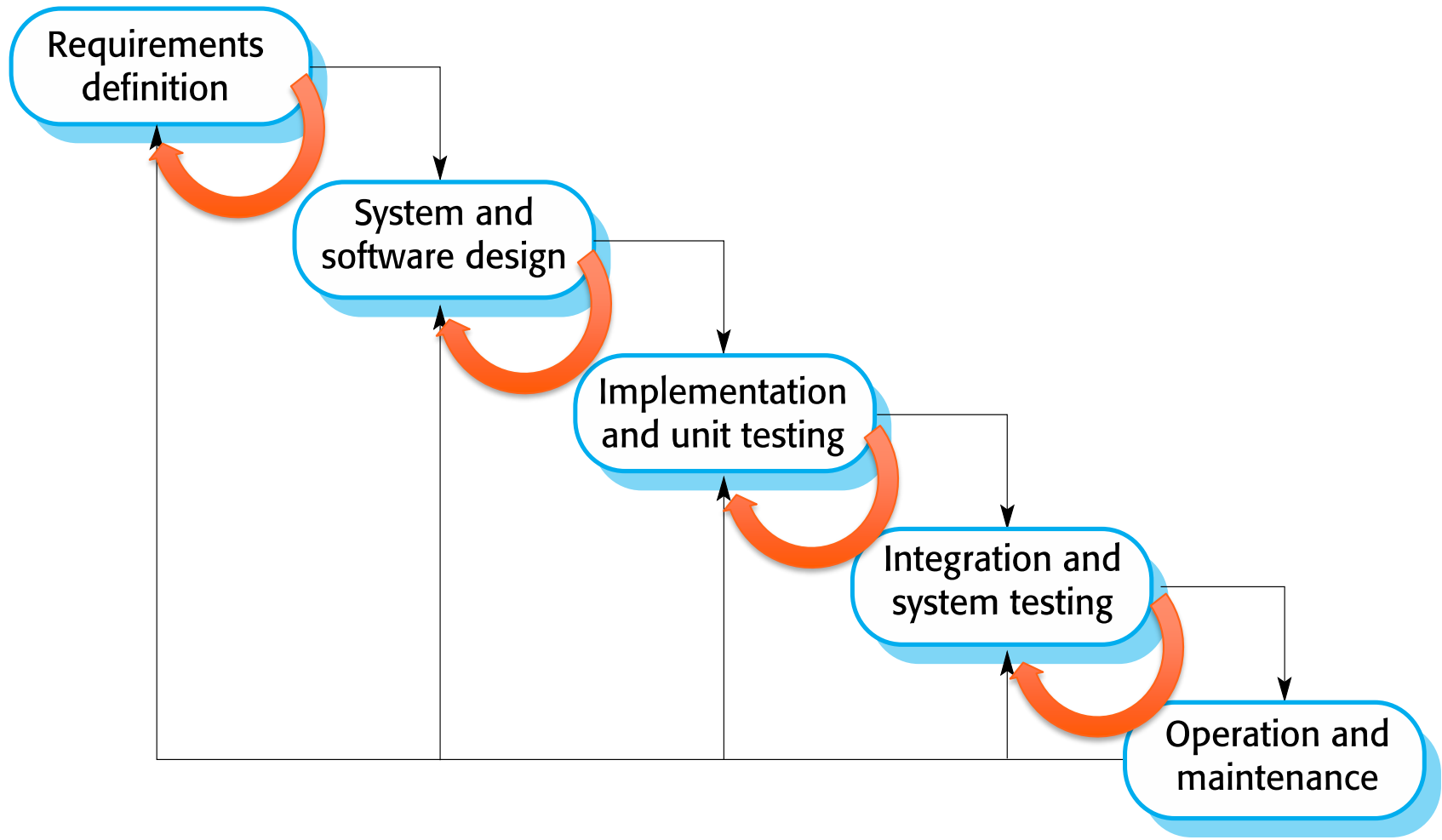
- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

Software process models

- The waterfall model
 - Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
 - Specification, development and validation are interleaved. May be plan-driven or agile.
- Integration and configuration
 - The system is assembled from existing configurable components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models

What is a waterfall model?





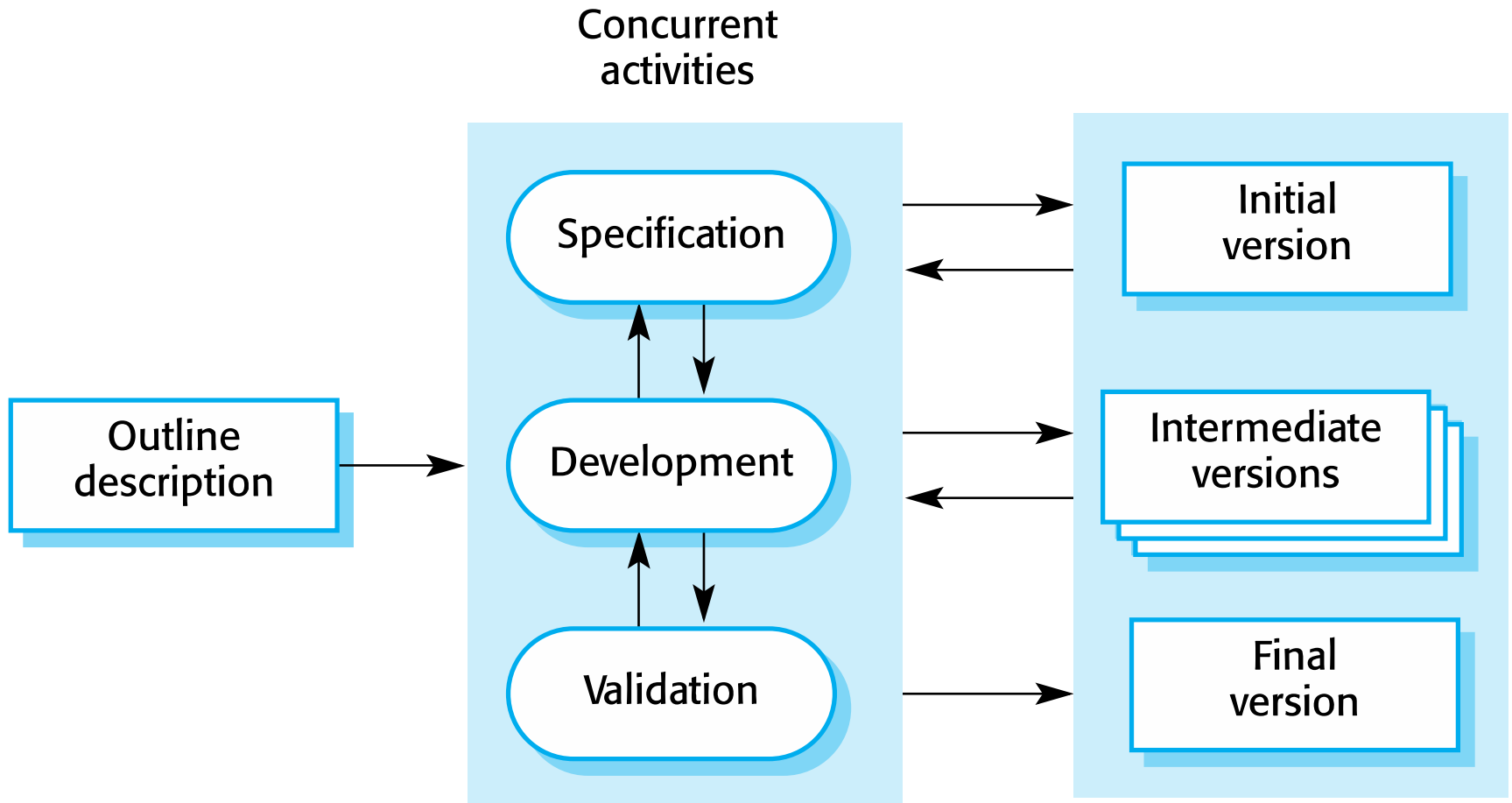
- There are separate identified phases in the waterfall model:
 - Requirements analysis and definition
 - System and software design
 - Implementation and unit testing
 - Integration and system testing
 - Operation and maintenance

What is the challenge of the waterfall model?

The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
 - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
 - Few business systems have stable requirements.
- The waterfall model is mostly used for large or safety critical systems engineering projects where a system is developed at several sites.
 - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

What is an incremental model?



Incremental model benefits:

- The cost of accommodating changing customer requirements is reduced.
 - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
 - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
 - Customers are able to use and gain value from the software earlier than is possible with a waterfall process

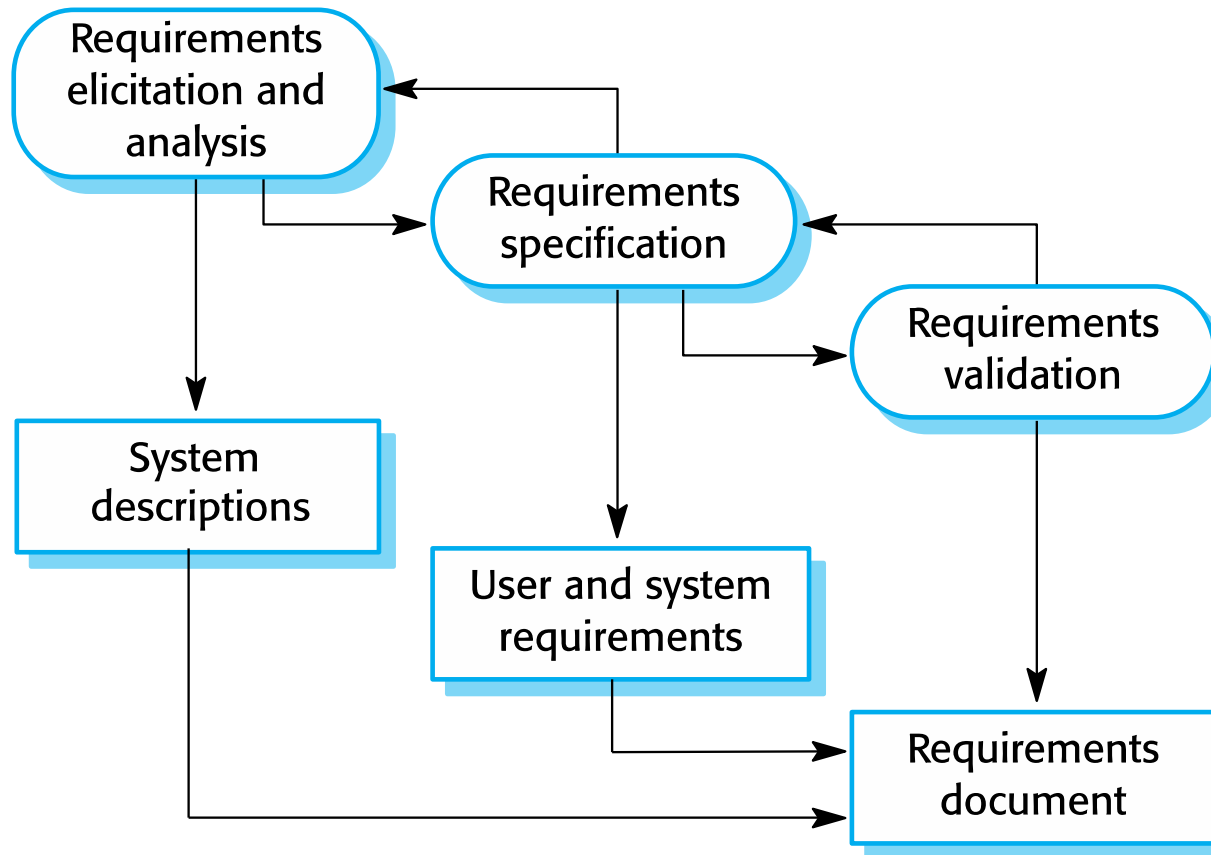
Incremental model challenges:

- The process is not visible.
 - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
 - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

Process activities

- Real software processes are inter-leaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.
- The four basic process activities of specification, development, validation and evolution are organized differently in different development processes.
- For example, in the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

Requirements engineering process

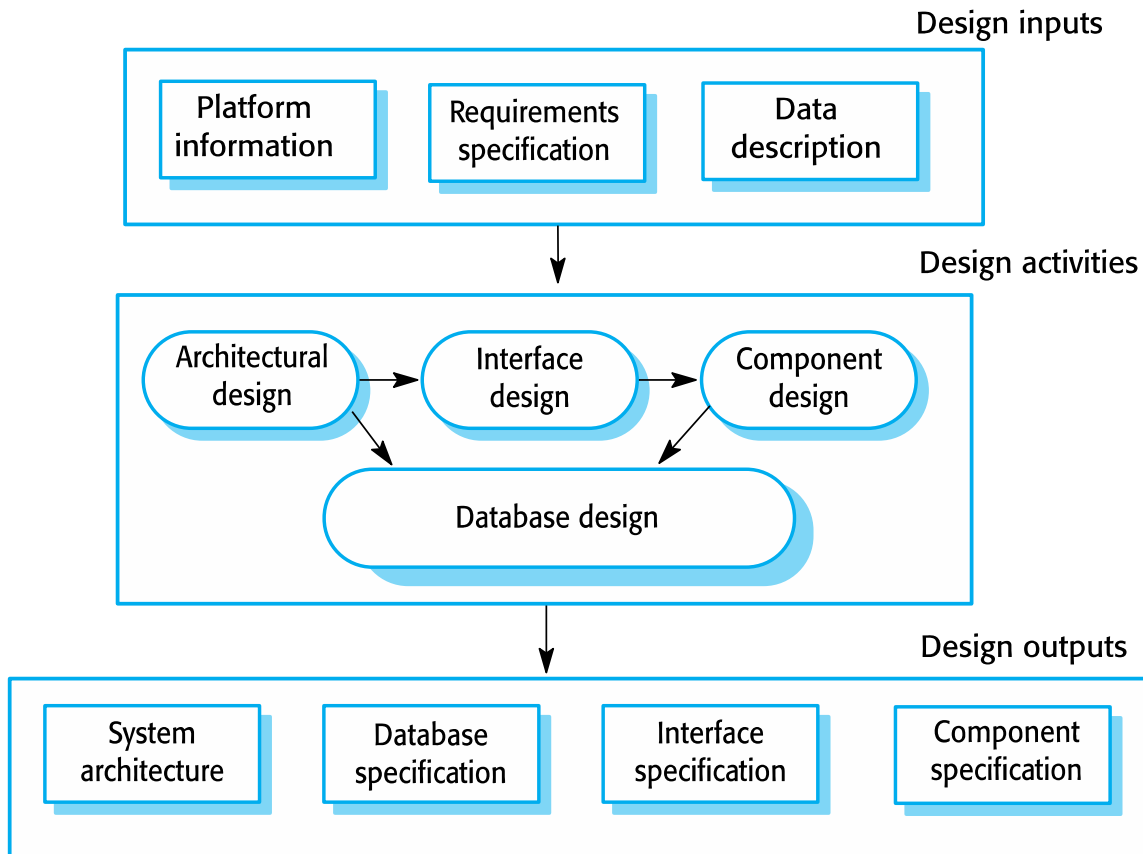


- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
 - Requirements elicitation and analysis
 - What do the system stakeholders require or expect from the system?
 - Requirements specification
 - Defining the requirements in detail
 - Requirements validation
 - Checking the validity of the requirements

Software design & implementation

- The process of converting the system specification into an executable system.
- Software design
 - Design a software structure that realises the specification;
- Implementation
 - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.

Design process model



Design activities

- *Architectural design*, where you identify the overall structure of the system, the principal components (subsystems or modules), their relationships and how they are distributed.
- *Database design*, where you design the system data structures and how these are to be represented in a database.
- *Interface design*, where you define the interfaces between system components.
- *Component selection and design*, where you search for reusable components. If unavailable, you design how it will operate.
- User experience design, where you design the user experience of the system

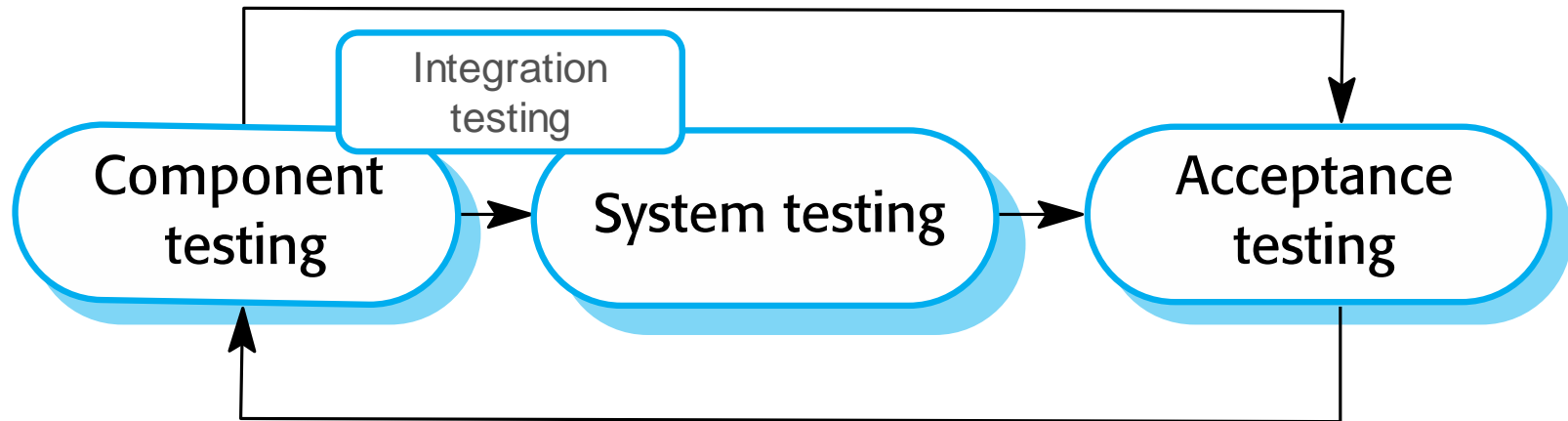
System implementation

- The software is implemented either by developing a program or programs or by configuring an application system.
- Design and implementation are interleaved activities for most types of software system.
- Programming is an individual activity with its own methods and processes.
- Debugging is the activity of finding program faults and correcting these faults.

Software validation

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- Testing is the most commonly used V & V activity.

Stages of testing

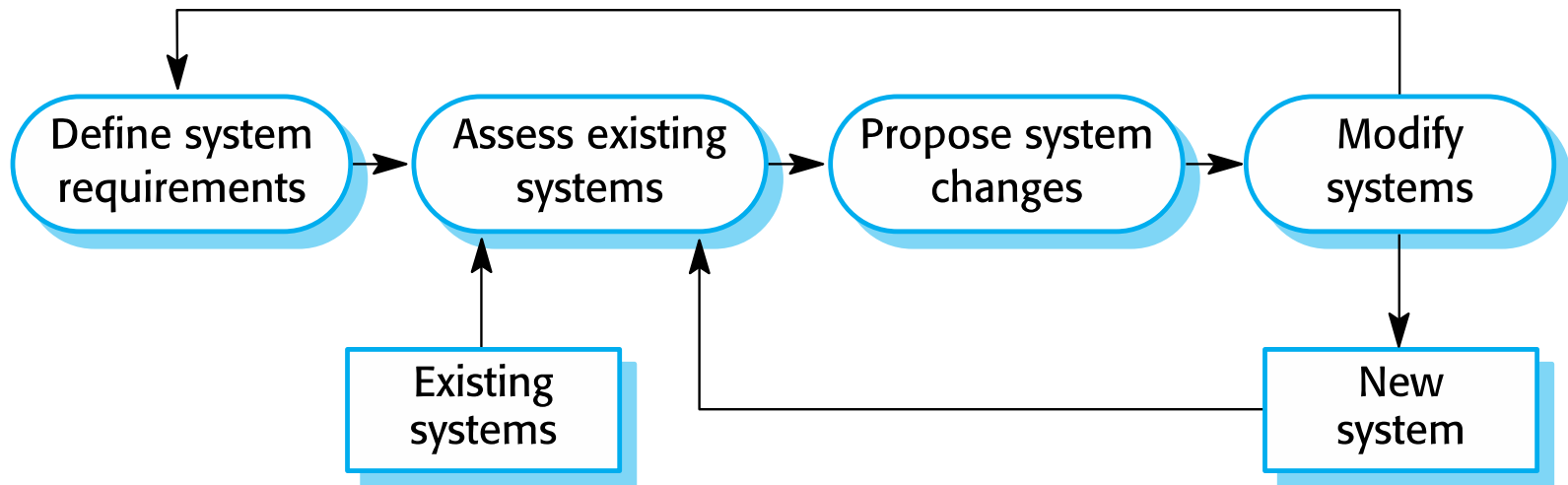


- Component testing
 - Individual components are tested independently;
 - Components may be functions or objects or coherent groupings of these entities.
- Integration testing
 - Two or more components are combined and tested
- System testing
 - Testing of the system as a whole. Testing of emergent properties is particularly important.
- Customer testing
 - Testing with customer data to check that the system meets the customer's needs.

Software evolution

- Software is inherently flexible and can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change.
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

System evolution



Agile development

Manifesto for Agile Software Development

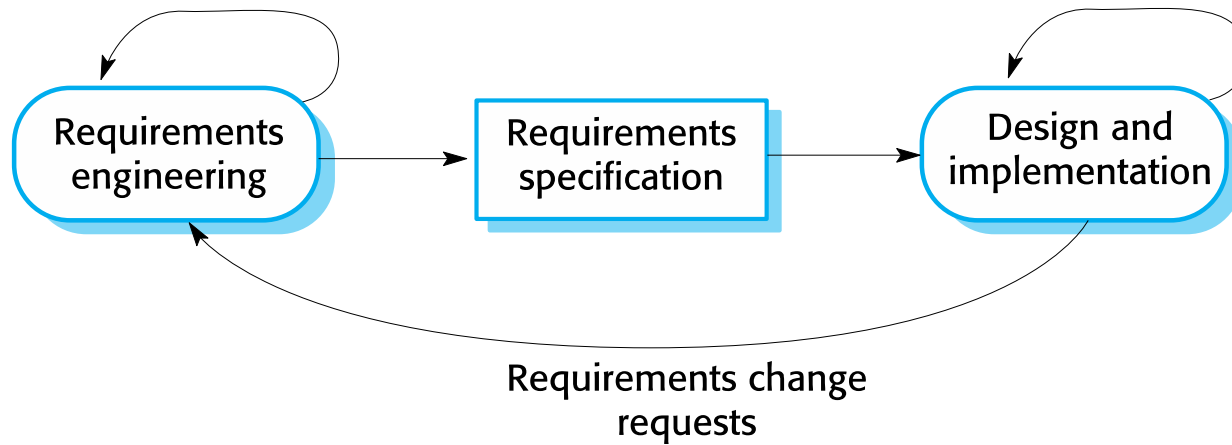
- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
 - **Individuals and interactions** over processes and tools
 - **Working software** over comprehensive documentation
 - **Customer collaboration** over contract negotiation
 - **Responding to change** over following a plan
- That is, while there is value in the items on the right, we value the items on the left more.

- Rapid development and delivery is now often the most important requirement for software systems
 - Businesses operate in a fast –changing requirement and it is practically impossible to produce a set of stable software requirements
 - Software has to evolve quickly to reflect changing business needs.
- Plan-driven development is essential for some types of system but does not meet these business needs.
- Agile development methods emerged in the late 1990s whose aim was to radically reduce the delivery time for working software systems

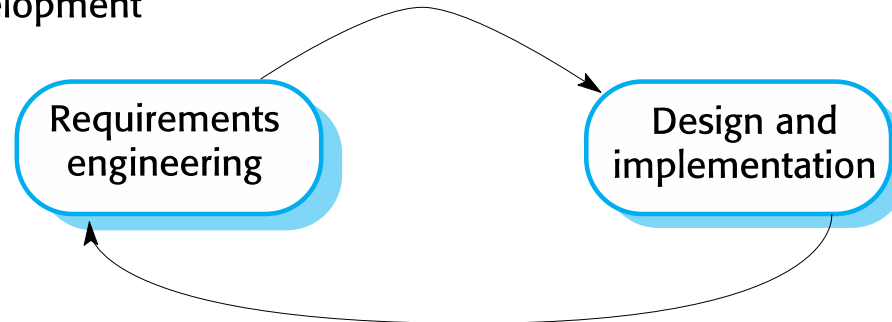
Agile development

- Program specification, design and implementation are inter-leaved
- The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
- Frequent delivery of new versions for evaluation
- Extensive tool support (e.g. automated testing tools) used to support development.
- Minimal documentation – focus on working code

Plan-based development



Agile development



- Plan-driven development
 - A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
 - Not necessarily waterfall model – plan-driven, incremental development is possible
 - Iteration occurs within activities.
- Agile development
 - Specification, design, implementation and testing are interleaved and the outputs from the development process are decided through a process of negotiation during the software development process.

The principles of agile methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Scrum

- Scrum is an agile method that focuses on managing iterative development rather than specific agile practices.
- There are three phases in Scrum.
 - **Sprint 0:** The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
 - **Sprint n:** This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
 - **Testing & delivery:** The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

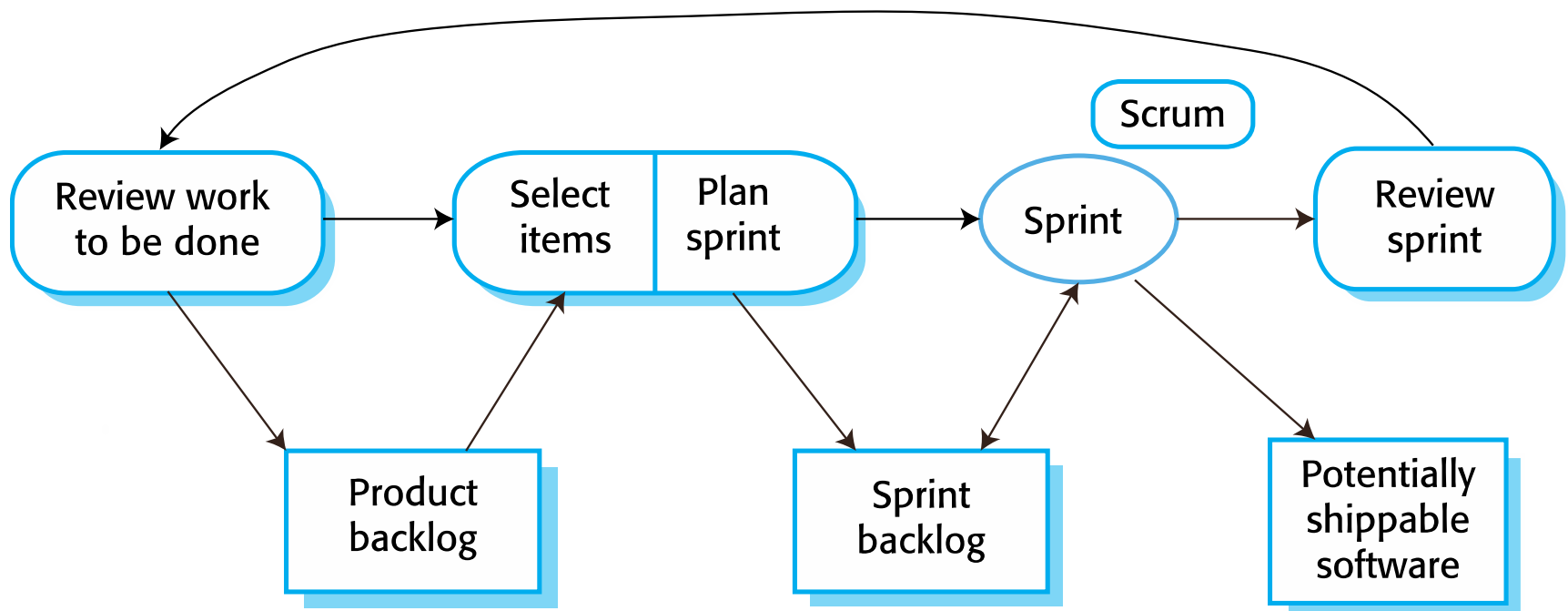
Scrum terminology

Scrum term	Definition
Development team	A self-organizing group of software developers, which should be no more than 7 people. They are responsible for developing the software and other essential project documents.
Potentially shippable product increment	The software increment that is delivered from a sprint. The idea is that this should be 'potentially shippable' which means that it is in a finished state and no further work, such as testing, is needed to incorporate it into the final product. In practice, this is not always achievable.
Product backlog	This is a list of 'to do' items which the Scrum team must tackle. They may be feature definitions for the software, software requirements, user stories or descriptions of supplementary tasks that are needed, such as architecture definition or user documentation.
Product owner	An individual (or possibly a small group) whose job is to identify product features or requirements, prioritize these for development and continuously review the product backlog to ensure that the project continues to meet critical business needs. The Product Owner can be a customer but might also be a product manager in a software company or other stakeholder representative.

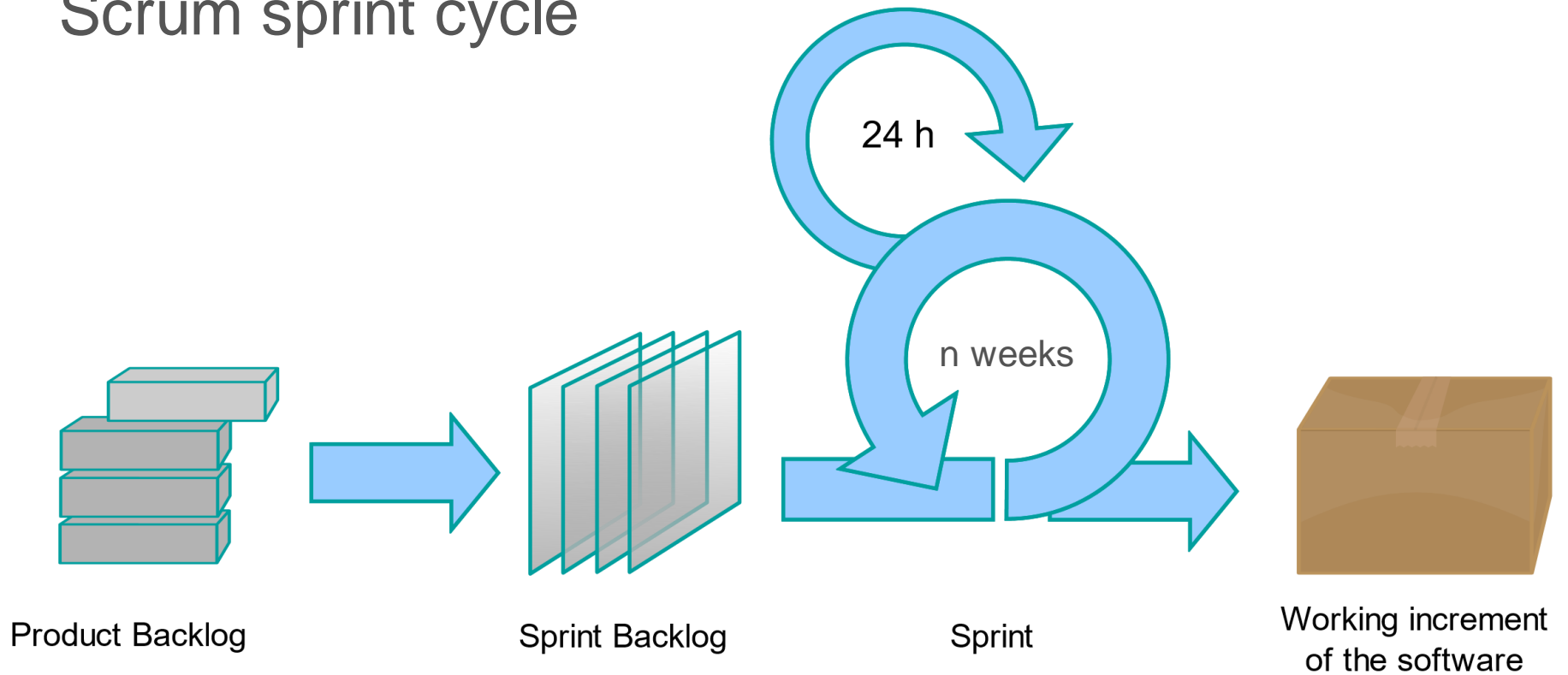
Scrum terminology

Scrum term	Definition
Scrum	A daily meeting of the Scrum team that reviews progress and prioritizes work to be done that day. Ideally, this should be a short face-to-face meeting that includes the whole team.
<u>ScrumMaster</u>	The <u>ScrumMaster</u> is responsible for ensuring that the Scrum process is followed and guides the team in the effective use of Scrum. He or she is responsible for interfacing with the rest of the company and for ensuring that the Scrum team is not diverted by outside interference. The Scrum developers are adamant that the <u>ScrumMaster</u> should not be thought of as a project manager. Others, however, may not always find it easy to see the difference.
Sprint	A development iteration. Sprints are usually 2-4 weeks long.
Velocity	An estimate of how much product backlog effort that a team can cover in a single sprint. Understanding a team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring improving performance.

Scrum sprint cycle



Scrum sprint cycle



Scrum sprint cycle

- Sprints are fixed length, normally 2–4 weeks.
- The starting point for planning is the product backlog, which is the list of work to be done on the project.
- The selection phase involves all of the project team who work with the customer to select the features and functionality from the product backlog to be developed during the sprint.

Scrum sprint cycle

- Once these are agreed, the team organize themselves to develop the software.
- During this stage the team is isolated from the customer and the organization, with all communications channelled through the so-called 'Scrum master'.
- The role of the Scrum master is to protect the development team from external distractions.
- At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

Teamwork in scrum

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

Scrum benefits

- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team have visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.
- The whole team, including the customer, has well defined common goal and they all work to reach it