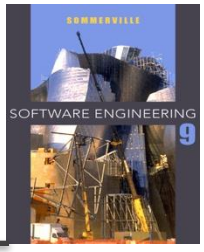
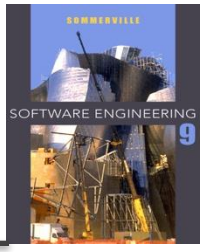


Software Processes: Waterfall & Agile

Youssef Ghatas

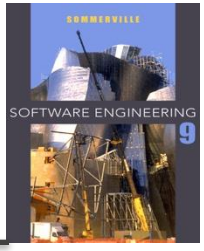
This material is a modified version of the slides provided by Ian Sommerville for his book “Software Engineering”, 9th edition.

Acknowledgement



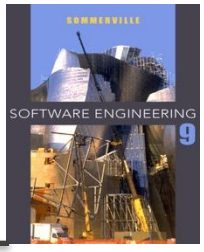
- Presentation is adapted from Ian Sommerville “Software Engineering 9th ed.”

Topics covered



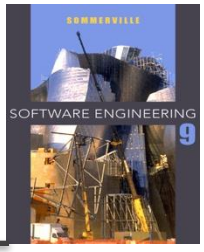
- ✧ Software process models
- ✧ Process activities

The software process



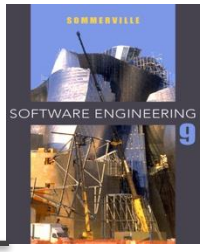
- A **structured set of activities** required to develop a software system.
- Many different software processes but all involve:
  - **Specification** – defining what the system should do;
  - **Design and implementation** – defining the organization of the system and implementing the system;
  - **Validation** – checking that it does what the customer wants;
  - **Evolution** – changing the system in response to changing customer needs.

Plan-driven vs. 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.
- Some practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

Software process models

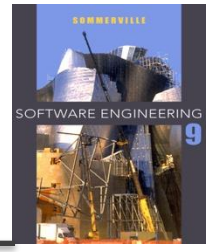


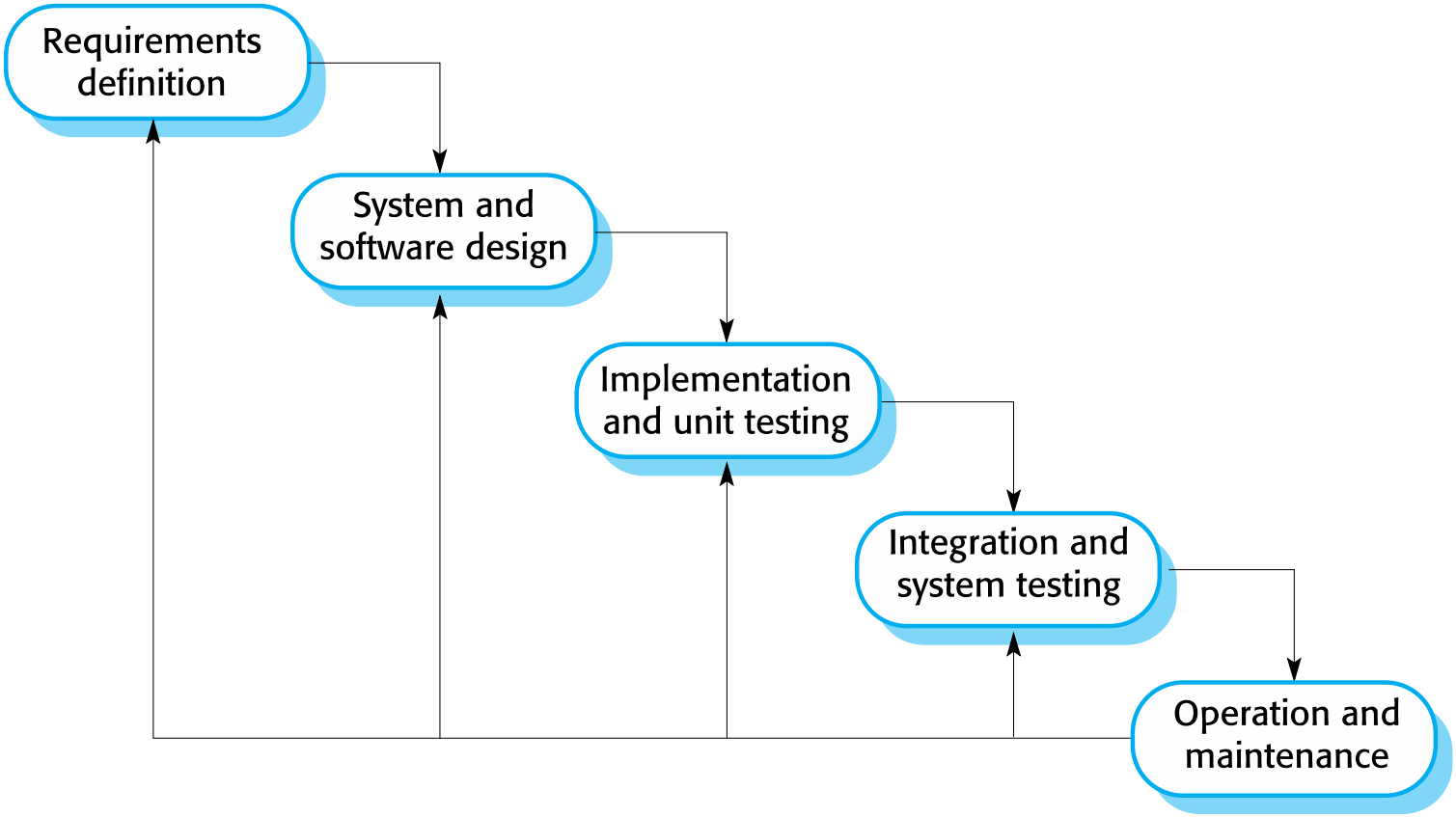
- The waterfall model
  - **Plan-driven** model. **Separate and distinct phases** of specification and development.
- Incremental development
  - Specification, development and validation are **interleaved**.

May be plan-driven or agile.

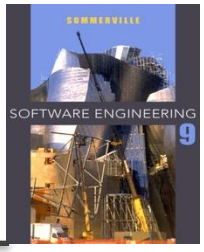
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

The waterfall model



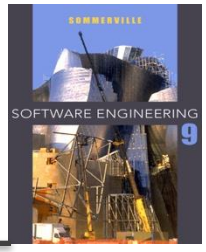


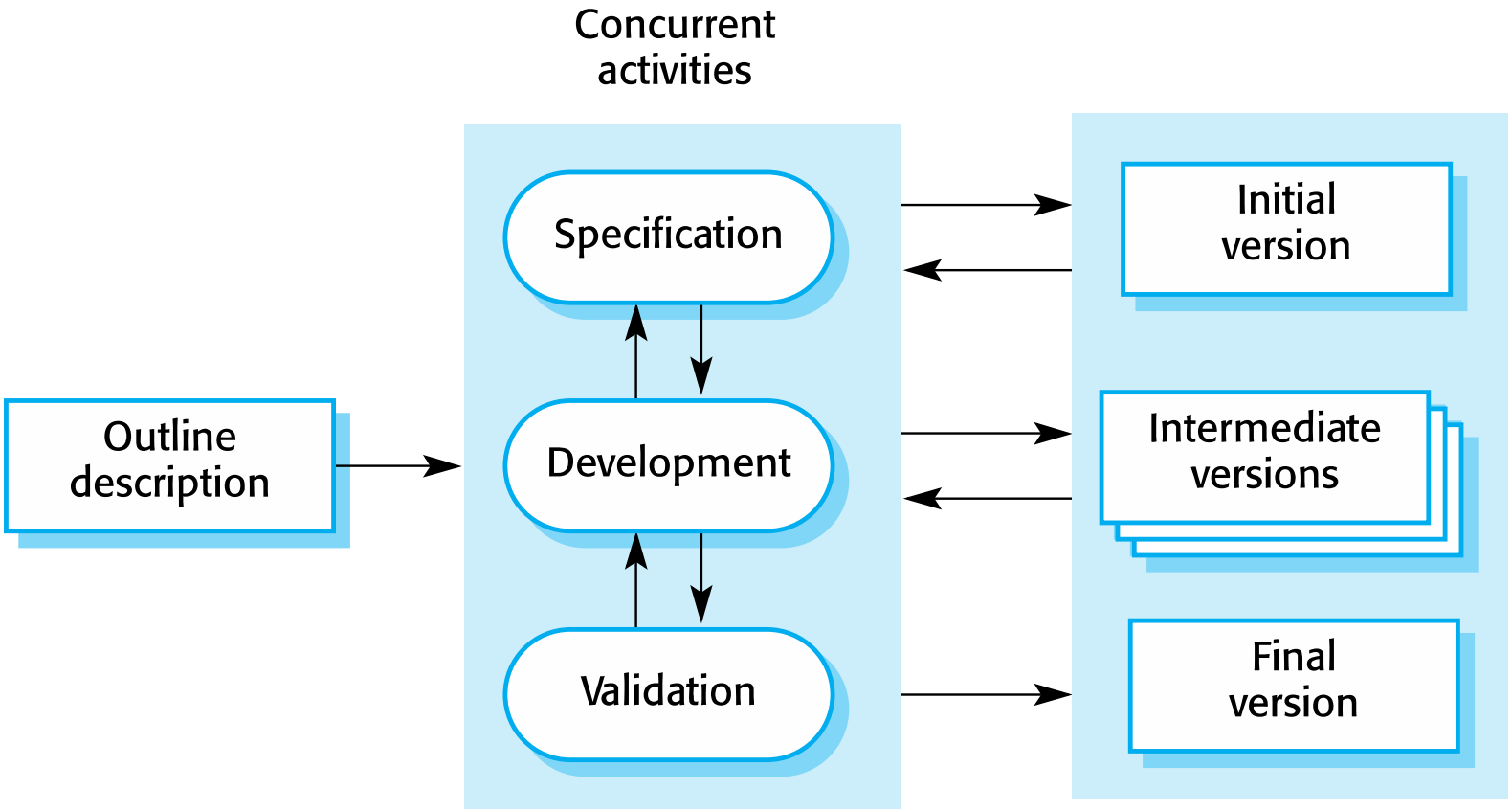
Waterfall model problems



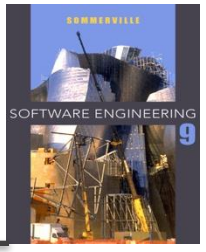
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the **requirements are well-understood** and changes will be fairly limited during the design process.
  - Very Few business systems have stable requirements.
- The waterfall model is mostly used for **large 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.

Incremental development



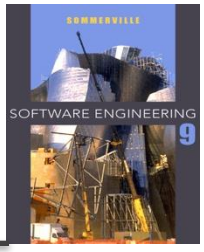


Incremental development benefits



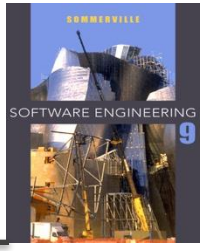
- The **cost of accommodating changing customer requirements is reduced**.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is **easier to get customer feedback** on the development work that has been done.
  - Customers can 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 development problems



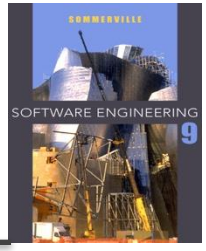
- **The process is not visible**.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- **System structure tends to degrade** as new increments are added.
  - Unless time 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



- The four basic process activities of specification, development, validation and evolution are organized differently in different development processes.
  - In the waterfall model, they are organized in sequence
  - In incremental development they are inter-leaved.

Why Agile?



https://oril.co/blog/intro-to-agile-methodology-scrum-kanban-frameworks/

Agile software engineering

- Software products must be brought to market quickly so rapid software development and delivery is essential.
- Virtually most software products are now developed using an agile approach.
- Agile software engineering focuses on delivering functionality quickly, responding to changing product specifications and minimizing development overheads.
- A large number of ‘agile methods’ have been developed.
  - 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

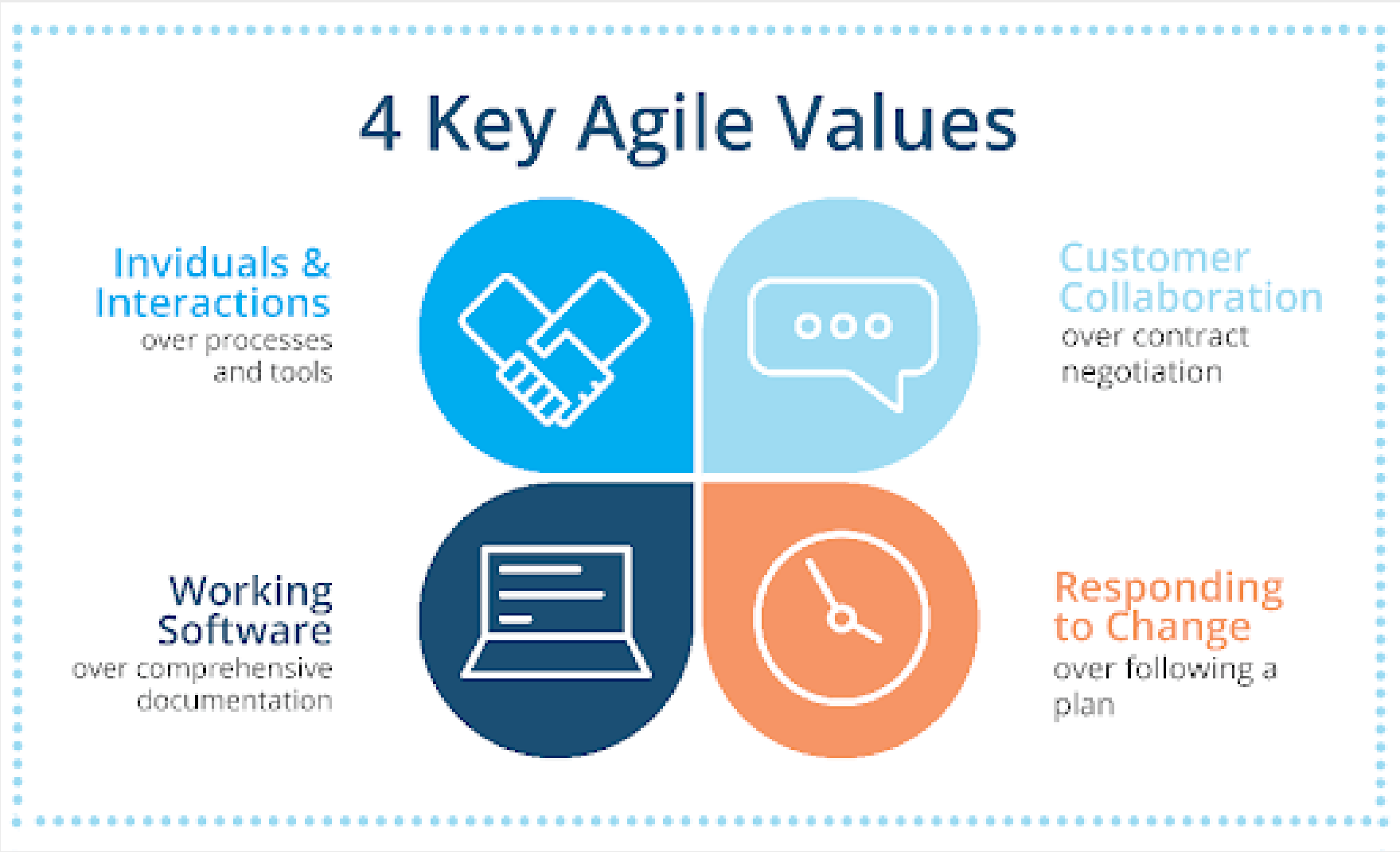
Agile methods

- Plan-driven development evolved to support the engineering of **large, long-lifetime systems** (such as aircraft control systems) where teams may be geographically dispersed and work on the software for several years.
    - This approach is based on controlled and rigorous software development processes that **include detailed project planning**, requirements specification and analysis and system modelling.
    - However, plan-driven development involves **significant overheads and documentation** and it does not support the rapid development and delivery of software.
- Agile methods were developed in the 1990s to address this problem.
    - These methods focus on the **software rather than its documentation**, develop software in a series of increments and aim to reduce process bureaucracy as much as possible.

Agile

- Agile Manifesto
- Incremental Development
- Agile Principles
    - XP
    - Scrum

The agile manifesto



Source: https://www.geneca.com/what-is-agile-software-development/

The agile manifesto

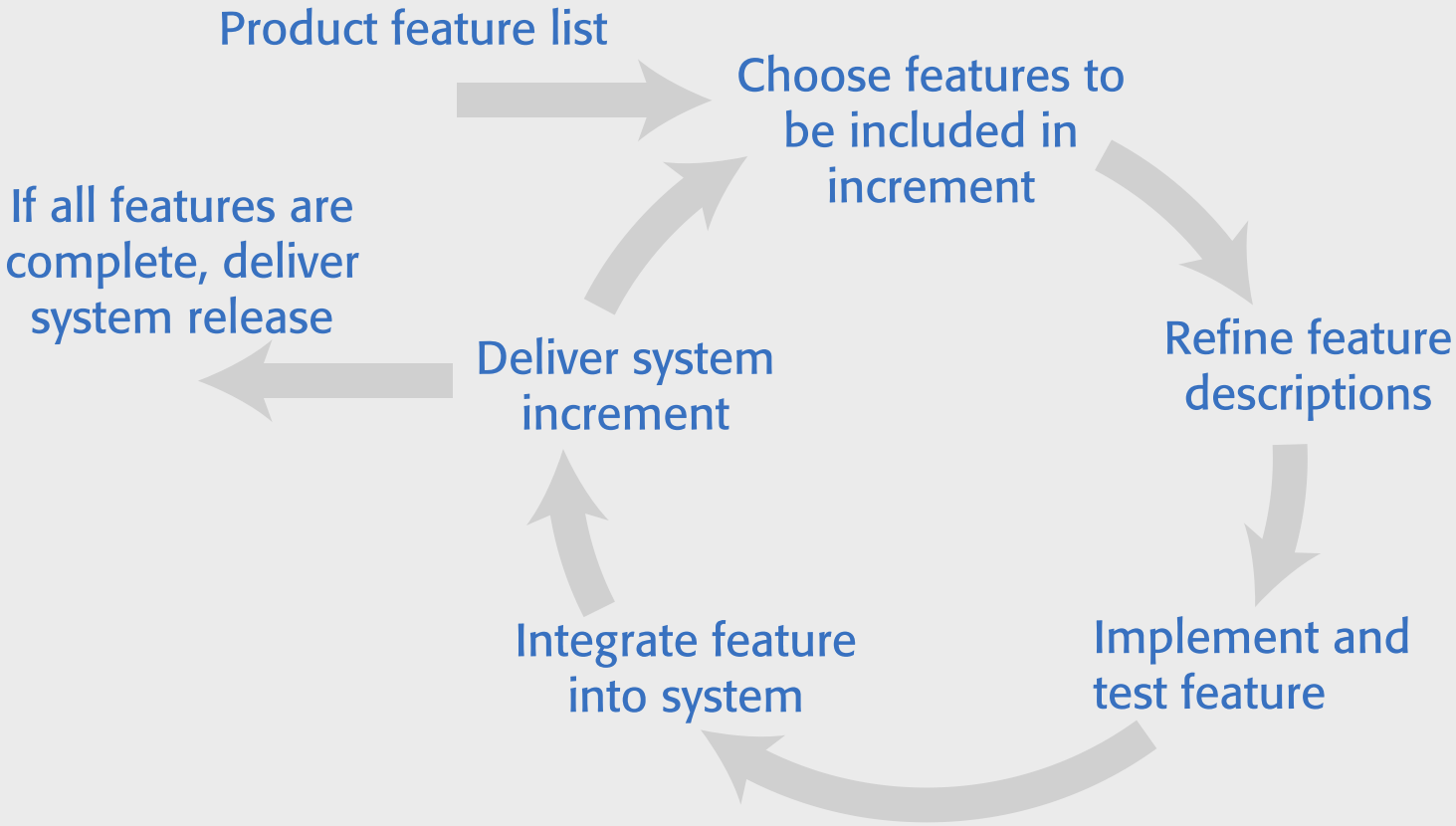
- We are uncovering better ways of developing software by doing it and helping others to do it. Through this work, we have come to value:
- - individuals and interactions >> processes and tools;
- - working software >> comprehensive documentation;
- - customer collaboration >> contract negotiation;
- - responding to change >> following a plan.

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

Incremental development

- All agile methods are based around incremental development and delivery.
- Product development focuses on the software features, where a feature does something for the software user.
- With incremental development, you start by **prioritizing the features** so that 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 development team. You then go on to define and implement the next feature of the system.

Incremental development



Agile development principles

- ***Involve the customer***
  Involve customers closely with the software development team. Their role is to provide and prioritize new system requirements and to evaluate each increment of the system.

- ***Embrace change***
  Expect the features of the product and the details of these features to 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***
  Always develop software products in increments. Test and evaluate each increment as it is developed and feed back required changes to the development team.

Agile development principles

- ***Maintain simplicity***
  Focus on simplicity in both the software being developed and in the development process. **Wherever possible, do what you can to eliminate complexity** from the system.

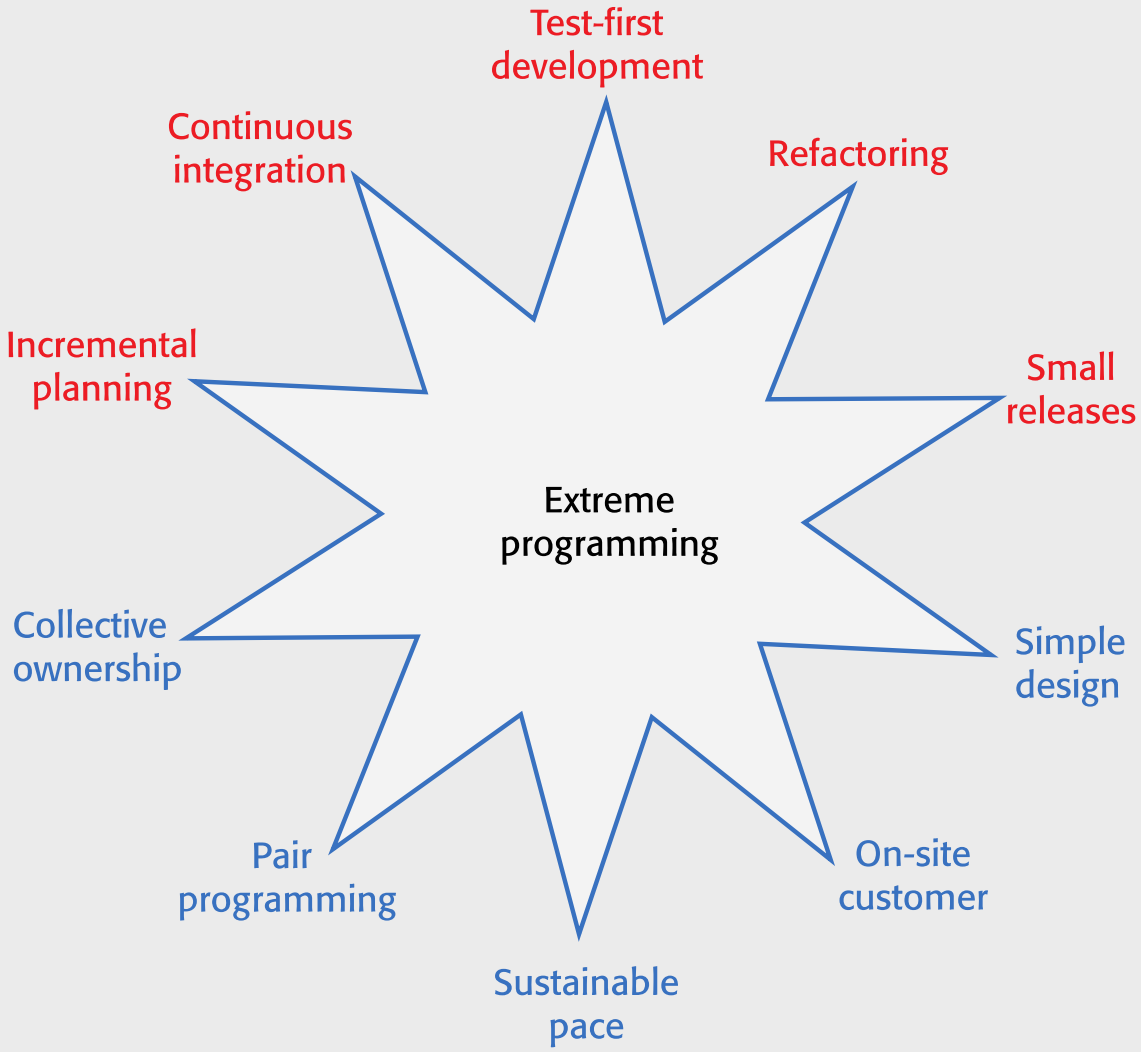
- ***Focus on people, not things***
  Trust the development team and do not expect everyone to always do the development process in the same way. **Team members should be left to develop their own ways** of working without being limited by prescriptive software processes.

Extreme programming

- The name was coined by Kent Beck in 1998 because the approach was developed by pushing recognized good practice, such as iterative development, to ‘extreme’ levels.
- Extreme programming focused on 12 new development techniques that were geared to rapid, incremental software development, change and delivery.
- Some of these techniques are now widely used; others have been less popular.

- The most widely used XP techniques (highlighted in red on the following slide) are explained elsewhere in the book.

Extreme programming 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 tests for that code, developers write the tests first. This helps clarify what the code should actually do and that 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' code that has already been 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 and readability 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.

Question

- You are leading a team to develop a critical and complex software system for a client in the finance sector. The requirements are well-defined and unlikely to change during the project.
    - Agile
    - Waterfall
    - Extreme Programming

Question

- You are working on a mobile app development project for a startup. The client is open to changes in requirements based on user feedback and market trends.
    - Agile
    - Waterfall
    - Extreme Programming

Question

- You are part of a team developing a software solution for a medical device that requires a high level of safety and reliability. The client emphasizes thorough documentation and rigorous testing.
    - Agile
    - Waterfall
    - Extreme Programming

Question

- You are leading a team developing a website for a creative agency. The client values rapid development, frequent iterations, and close collaboration throughout the project.

  - Agile
  - Waterfall
  - Extreme Programming

Question

- You are working on a project where the primary focus is on delivering a high-quality software product through continuous integration, test-driven development, and regular feedback loops.

  - Agile
  - Waterfall
  - Extreme Programming