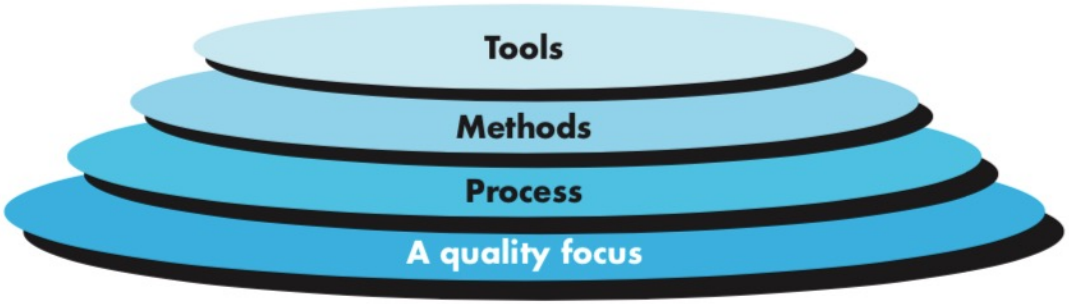
The Software Development Process

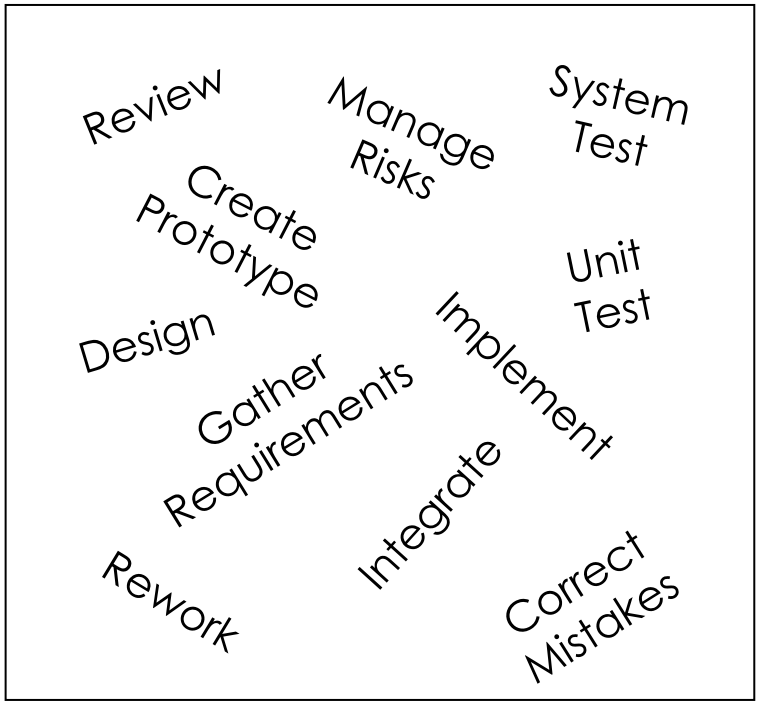
UA.DETI.IES

Process

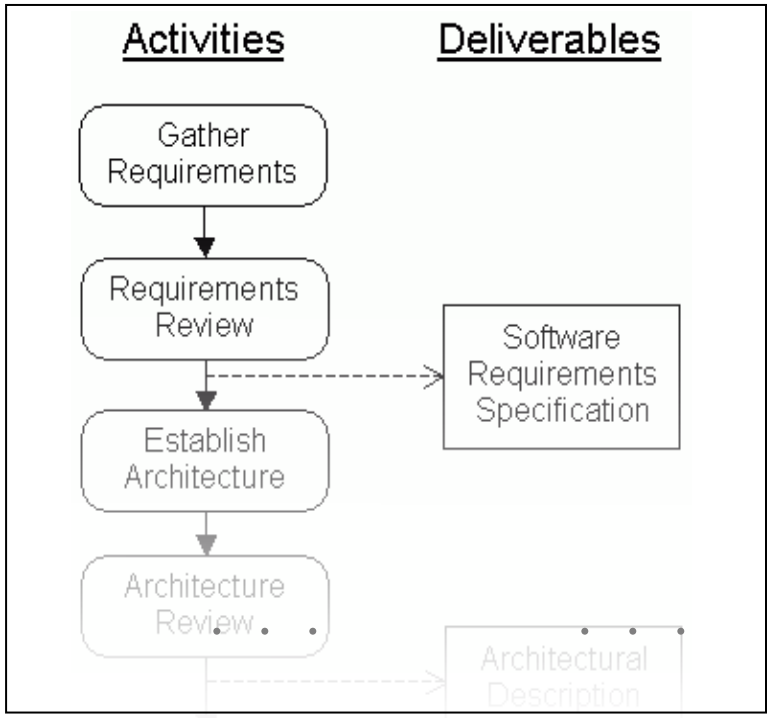
- The foundation for software engineering is the *process* layer.
- A *software process* is a framework for the activities, actions, and tasks that are required to build high-quality software.
- It establishes the technical and management framework for applying **methods**, **tools**, and **people** to the development task.



Why Software Process?



Developing software without a defined process is chaotic and inefficient



A process makes software development more orderly and manageable

"It is better not to proceed at all, than to proceed without method." -- Descartes

Software process

Many different software processes ... but all involve:

- Specification (communication and planning)
  - 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.

Software process description

- When we describe and discuss a software process, we usually talk about …
  - the **activities** 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.

Software process

- A software process specifies:
  - What?
  - Who?
  - How?
  - When?

- A software process includes:
  - Roles
  - Workflows
  - Procedures
  - Standards
  - Templates

Key points

- **Software process is a guide**

- There isn't one best process for writing software. The process that an individual or organization selects and follows depends on:
  - the specific characteristics of the project
  - the organization's culture
  - the abilities and preferences of the people involved

- A good process will raise the productivity of less experienced team members without impeding the work/progress of more experienced team members.

Resistance to Software Process

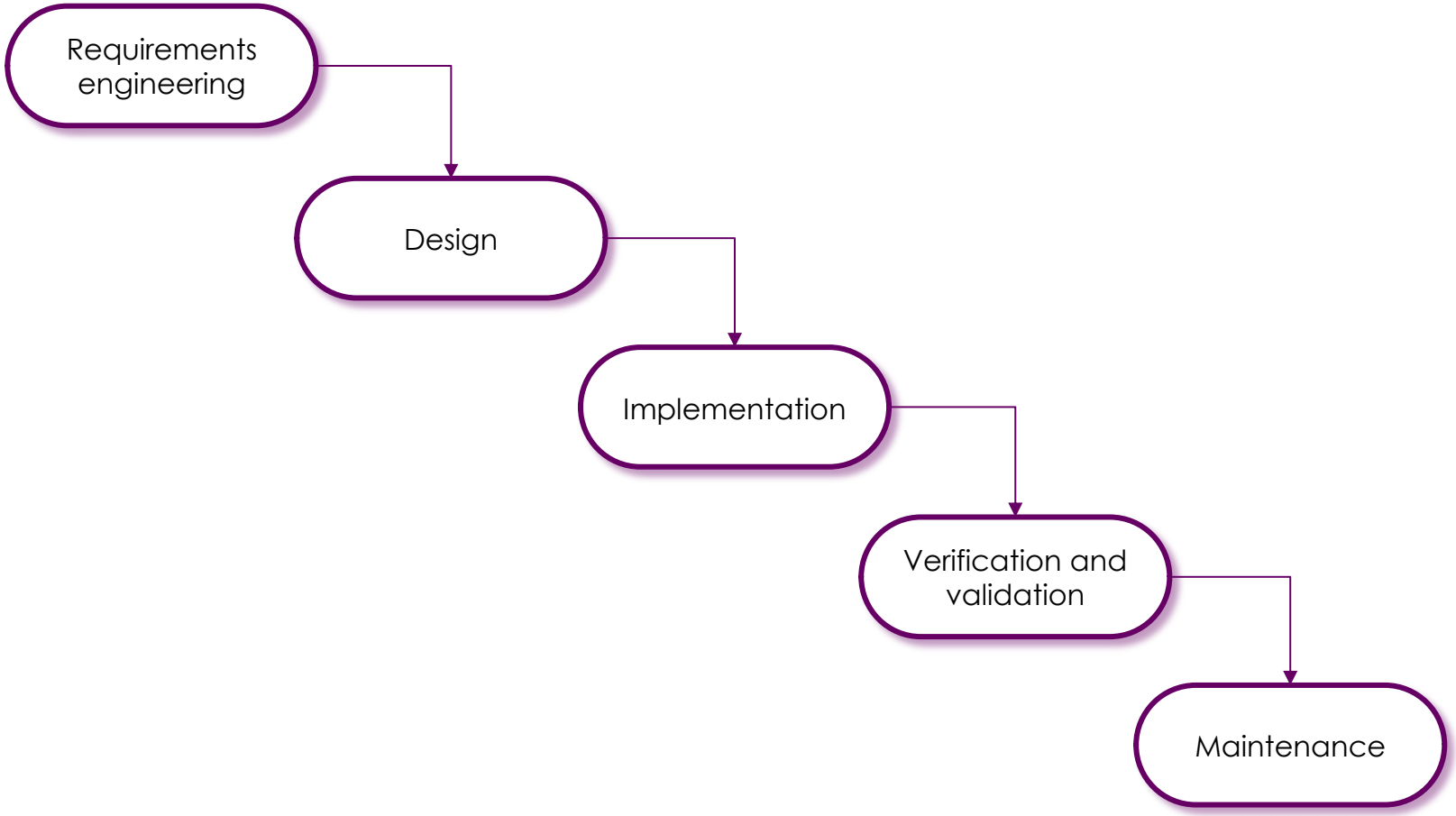
- **Perception**: Some people view following a process as an unnecessary overhead on productivity.
  - Interferes with creativity
  - Bureaucratic and regimented
  - Hinders agility in fast-moving markets

Resistance to Software Process

- **Perception**: Some people view following a process as an unnecessary overhead on productivity.
  - Interferes with creativity
  - Bureaucratic and regimented
  - Hinders agility in fast-moving markets

- **The reality**: Groups that don't start out following a defined process often find themselves adding process later in the project in reaction to problems.
  - As the size and complexity of a project grows, the importance of following a defined process grows proportionally.

Software phases



Software process models

- Abstract models that describe a class of development approaches with similar characteristics.

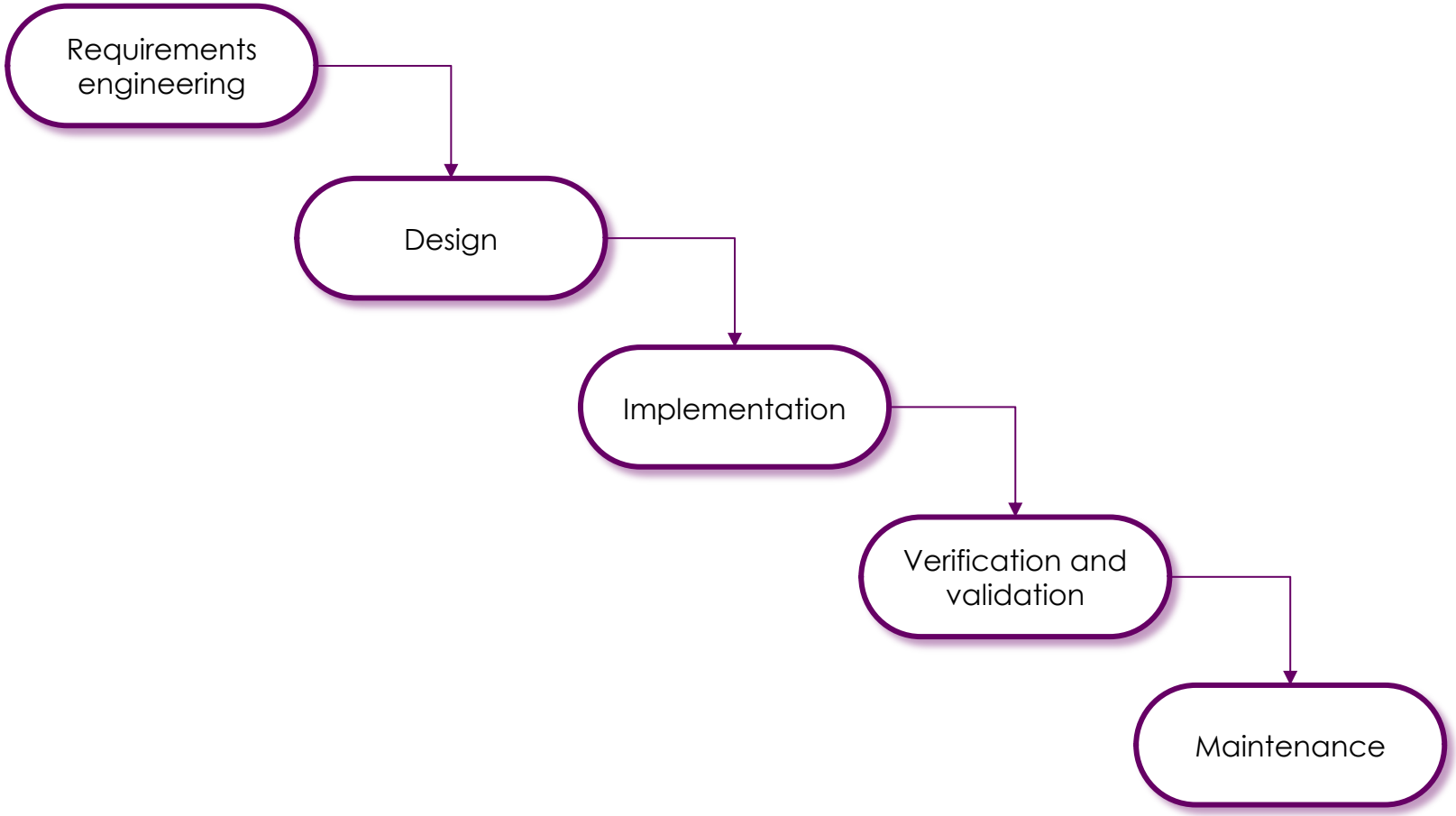
- Some of the criteria used to distinguish software process models are:
  - timing between phases,
  - entry and exit criteria between phases
  - the artifacts created during each phase.

- Examples include:
  - Waterfall, Spiral, Rapid Prototyping, Incremental Development, etc.

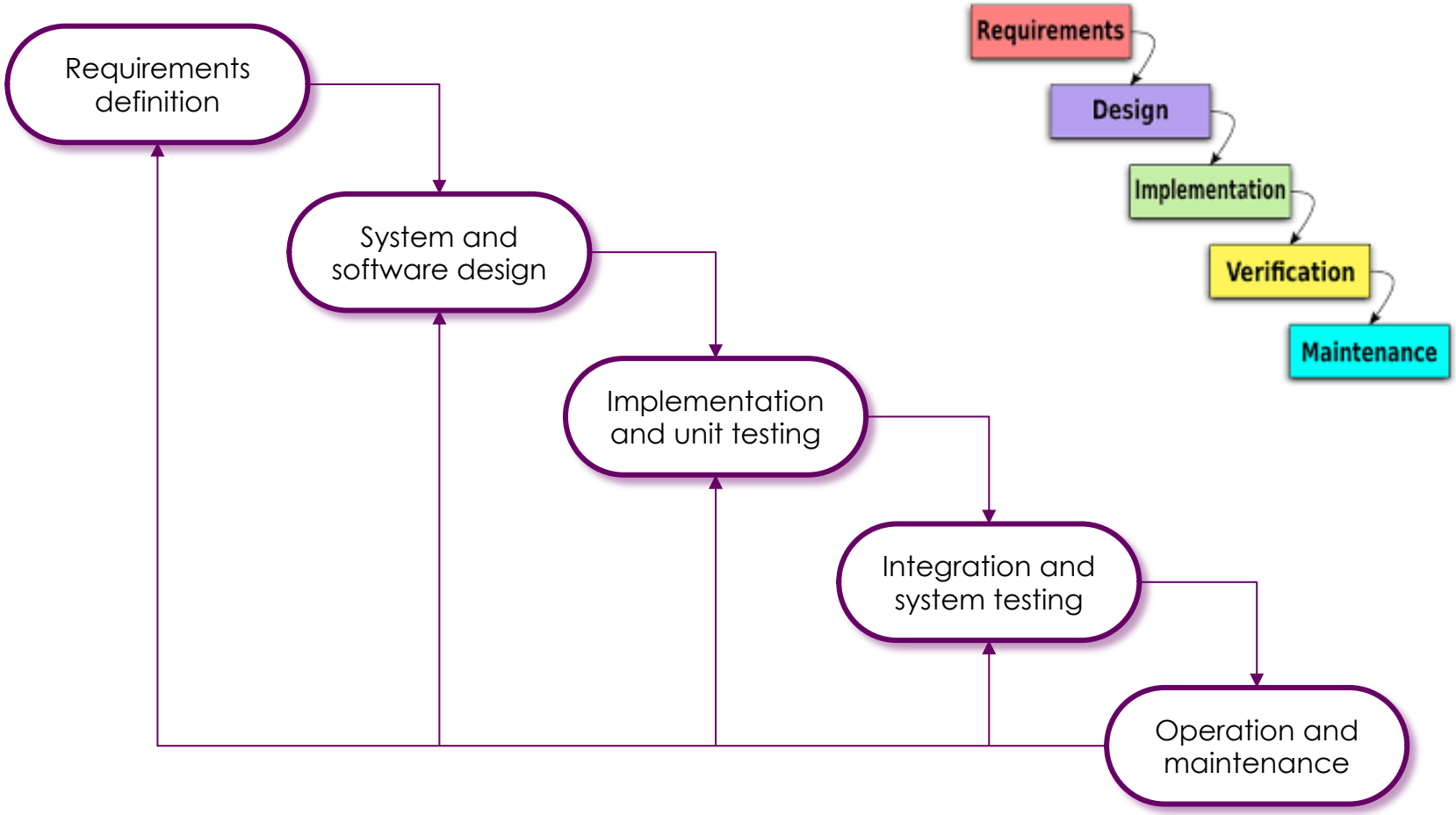
(Traditional) Software process models

- 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.
- **Evolutionary/Iterative** processes
  - The system is developed from start with very raw specification and modifying this according to the software needs.
- .. Many others
  - Most large systems are developed using a process that incorporates elements from different models.

Software phases



The Waterfall model



Waterfall model advantages

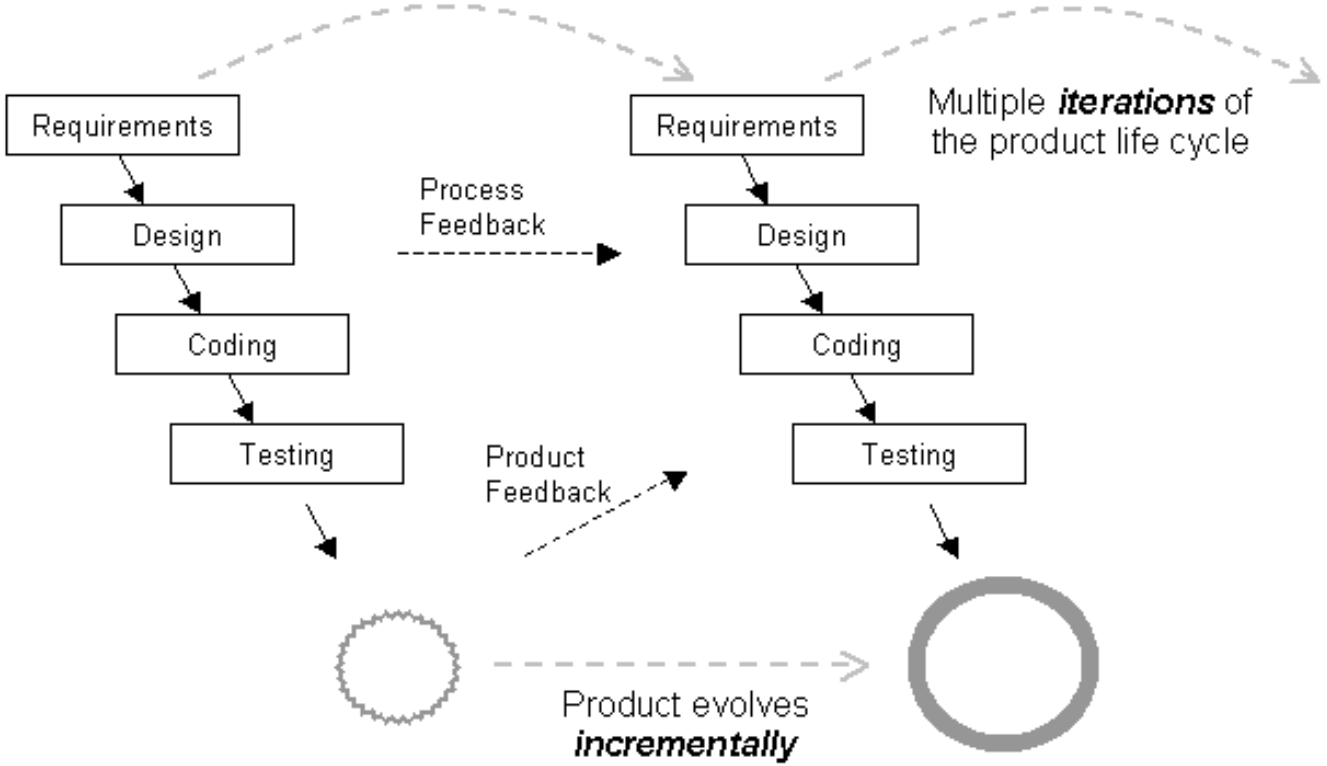
- Simple and easy to understand and use.
- Easy to plan
  - A schedule can be set with deadlines for each stage of development and a product can proceed through the development process like a car in a car-wash, and theoretically, be delivered on time.
- Easy to manage
  - each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well where requirements are very well understood.

Waterfall model disadvantages

- 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.
- Poor model for long and ongoing projects.
  - No working software is produced until late during the life cycle.
- Not suitable for the projects where requirements are uncertain or at the risk of changing.

The Incremental model

- A characteristic of modern life cycle models. The product evolves incrementally over a series of iterations.



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 can use and gain value from the software earlier than is possible with a waterfall process.

Incremental development problems

- **Each iteration phase is rigid** and does not overlap each other.
- The process is not visible.
  - Managers need regular deliverables to measure progress. But, 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.

Evolutionary/Iterative models

- **Prototyping**
  - Often, a customer defines a set of general objectives for software but does not identify detailed requirements for functions and features.
- **Spiral Model**
  - Using the spiral model, software is developed in a series of evolutionary releases. During early iterations, the release might be a model or prototype.
- **Concurrent Model**
  - allows a software team to represent iterative and concurrent elements of any of the process models.

Incremental vs Evolutionary/Iterative

Iterative

Incremental

Example #1

- **Scenario**
  - You are developing a web-based e-commerce platform for a client. The client has requested **several features**, including user authentication, product catalogue, shopping cart functionality, and payment processing. They want to launch the platform as soon as possible to start generating revenue but also want to **add new features and improvements over time**.

- Which approach do you propose?
  - Incremental
    1. Minimal viable product (MVP) – user authentication and catalogue
    2. Shopping cart
    3. Payment process

Example #2

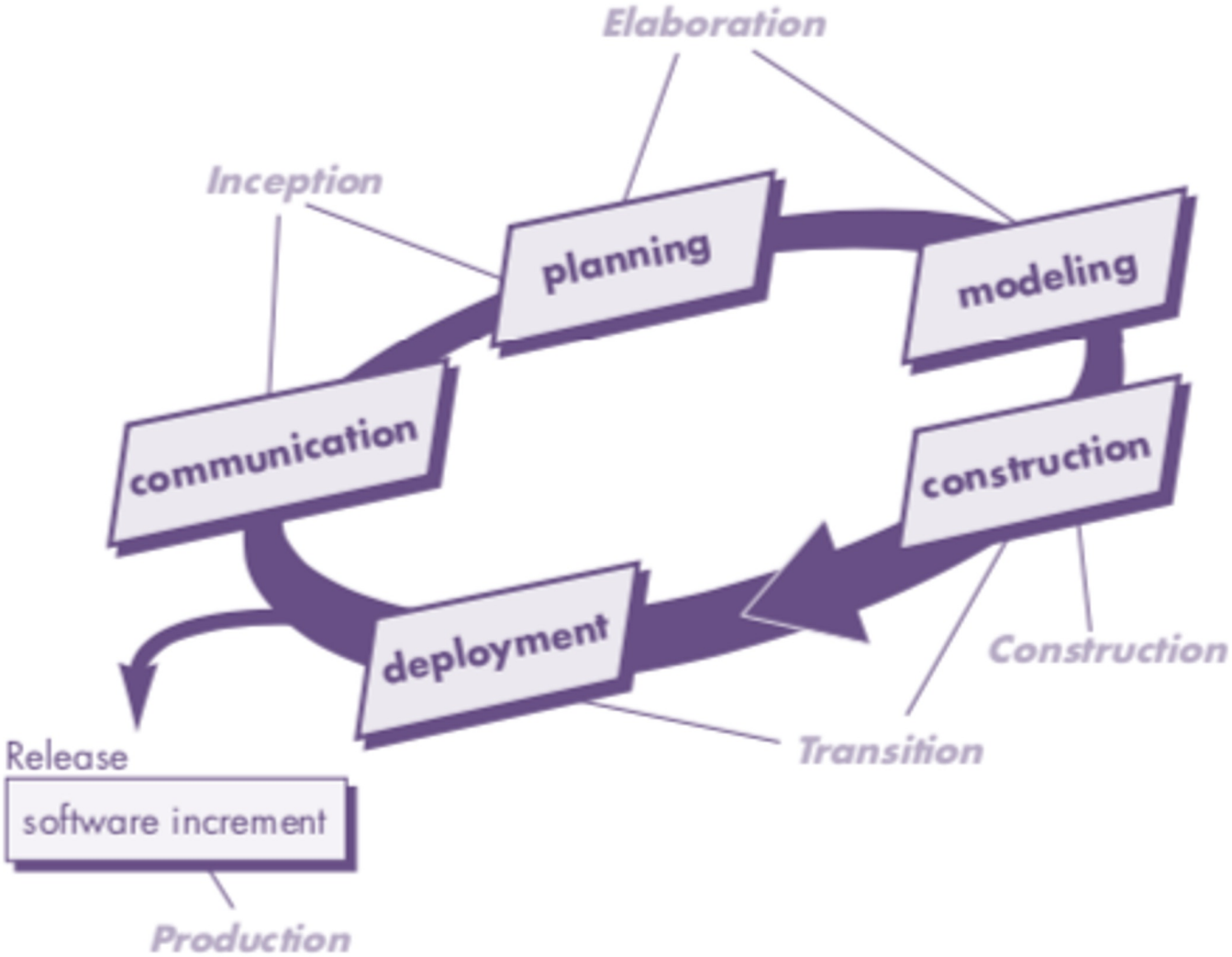
- **Scenario**
  - You need to develop a machine learning-based recommendation system for an online streaming service. The goal is to provide personalized content recommendations to users based on their viewing history and preferences. The **requirements are complex**, and it's **essential to continually refine the recommendation** algorithms for better accuracy and user satisfaction.

- Which approach do you propose?
  - Iterative
    1. Initial version of the system
    2. Simple recommendation algorithm
    3. Refine algorithms and models (in subsequent iterations)

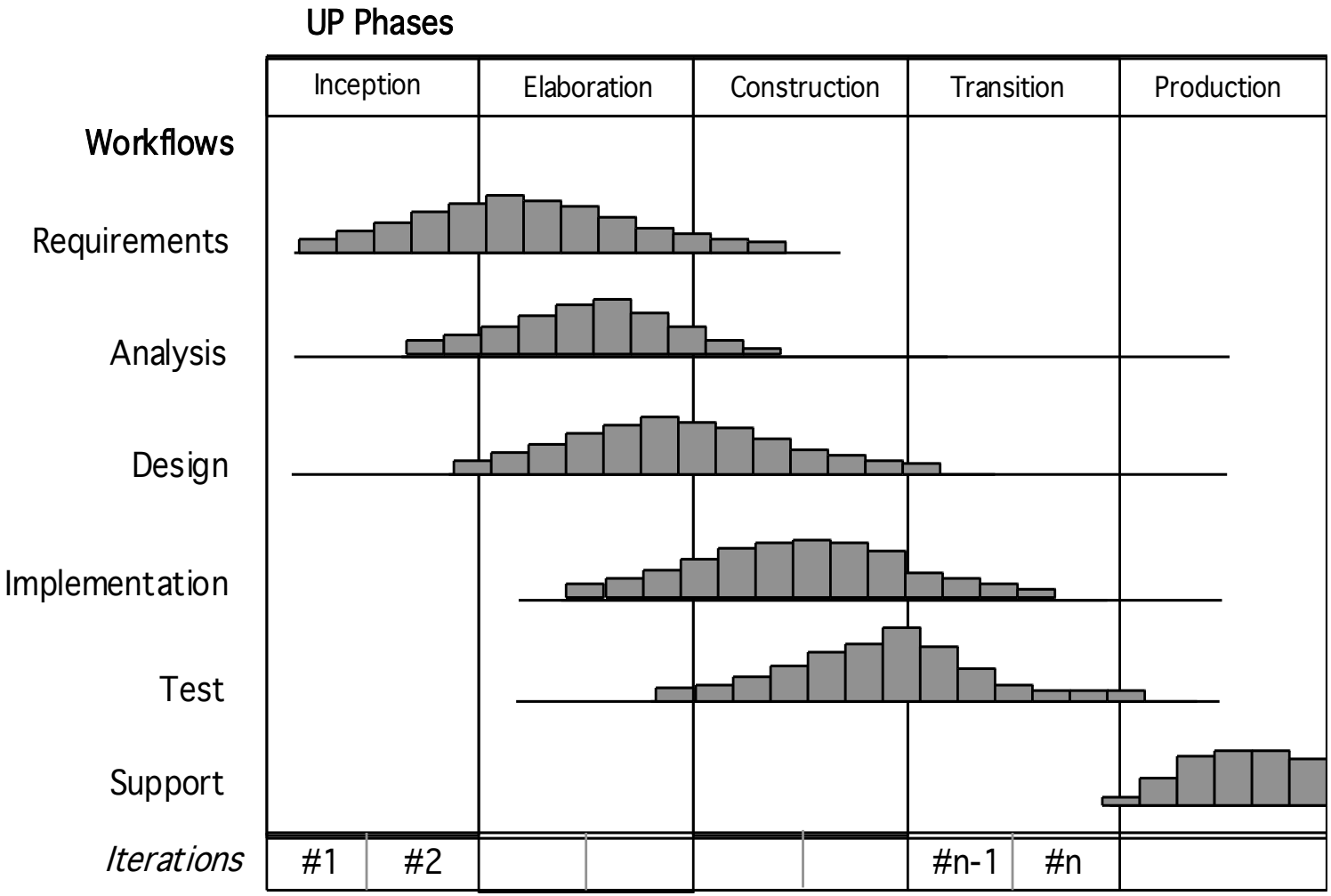
Other process models

- **Component-based** development (COTS)
  - the process to apply when reuse is a development objective
- **Formal methods**
  - emphasizes the mathematical specification of requirements
- **Unified Process**
  - a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML)

The Unified Process (UP)



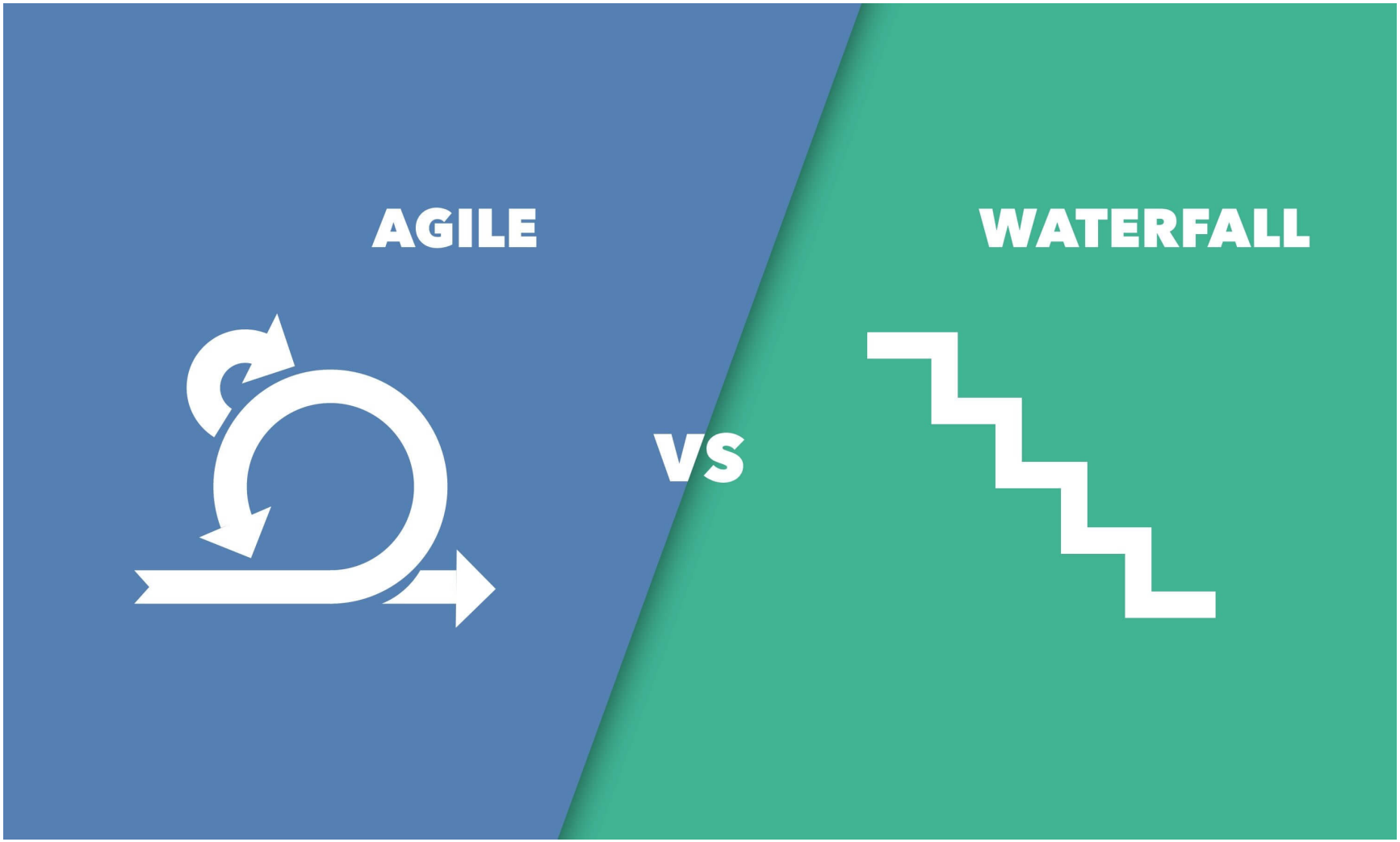
UP Phases


UP Phases
Inception
Elaboration
Construction
Transition
Production
Workflows
Requirements
Analysis
Design
Implementation
Test
Support
Iterations
#1
#2
#n-1
#n

Plan-driven and agile processes

- **Plan-driven processes** are processes where all 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.

Plan-driven and agile processes



Agile processes

- 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 must 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 methods

- Agile methods were developed in an effort to overcome perceived and actual weaknesses in conventional software engineering.

- **Focus on the code rather than the design**
- Are based on an **iterative approach** to software development
- Are intended to **deliver working software quickly** and evolve this quickly to meet changing requirements.

Origins: The Agile manifesto

"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*

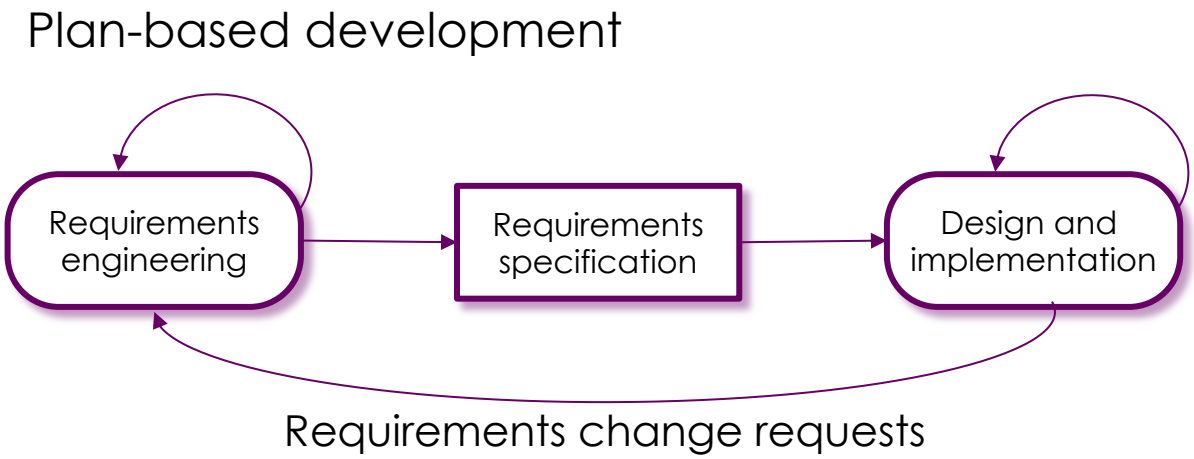
That is, while there is value in the items on the right, we value the items on the left more."

K. Beck et al, 2001

The principles of agile methods

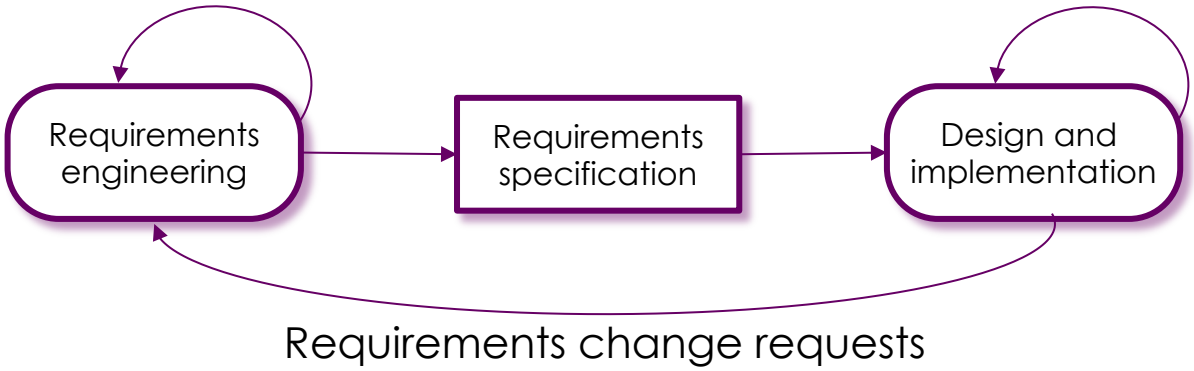
Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.

Plan-driven and agile development

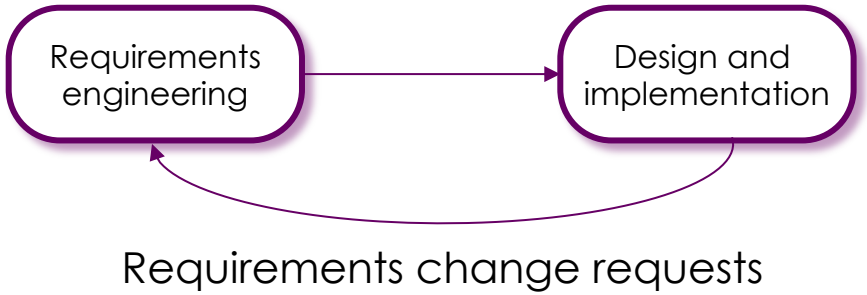


Plan-driven and agile development

Plan-based development



Agile development



Plan-driven and agile development

- **Plan-driven development**
  - 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

- **Agile development**
  - Specification, design, implementation and testing are inter-leaved
  - The outputs from the development process are decided through a process of negotiation during the software development process.

Agile methods benefits

- Requirements in Agile model can change as per the customer requirement.
  - Sometimes requirements are not very clear.
  - Changes in the requirements are accepted even in the later stages of the development.
- The delivery of software is continuous.
  - The customers can follow every Sprint working feature of the software.
- Refactoring is not costly.

Agile methods disadvantages

- The documentation is sparse.
- With unclear requirement it is difficult to predict the expected result.
  - More difficult to estimate the effort required.
- Some unknown/unpredicted risks which can affect the development of the project.

Agile method applicability

- Product development
  - where a software company is developing a small or medium-sized product for sale.
- Custom system development within an organization
  - where there is a clear commitment from the customer to become involved in the development process, and
  - where there are few external rules and regulations that affect the software.
- Virtually all software products and apps are now developed using an agile approach

Summary

- The software process

- Sequential model
  - Waterfall, V-Models
- Incremental model
- Evolutionary/Iterative models
  - Prototyping, Spiral
- Specialized models

- Plan-driven (predictive)
- Agile processes (adaptative)

Resources & Credits

- Roger S. Pressman, Bruce Maxim, Software Engineering: A Practitioner's Approach, 7th Edition, McGraw-Hill Education, 2015
  chapters 3 & 4

- Ian Sommerville, Software Engineering, 10th Edition, Pearson, 2016
  chapter 2