Software Development Processes - Part 1

Ohjelmankehityspr., versionhallinta ja testaus – Chapter 5

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Deepak K.C. ; deepak.kc@kamk.fi

Software Development Process

Structured set of activities required to develop a software system

Software development process involve

- Specification
  - What the system should do
- Design & implementation
  - Defining the organization of the system
  - Implementing the system
- Validation
  - Checking it does what your customer wants
- Evolution
  - Maintaining (changing) system in response to changing customer needs

Plan-driven & agile process

Plan driven process

- Process activities are planned in advance
- Progress is measured against this plan

Agile process

- Planning is incremental
- Easier to change the process to reflect changing customer requirements

Note: Practical process include elements of both plan driven & agile process
No right or wrong software process

Software process models

The waterfall model

- Plan driven model
- Separate and distinct phases of specification and development

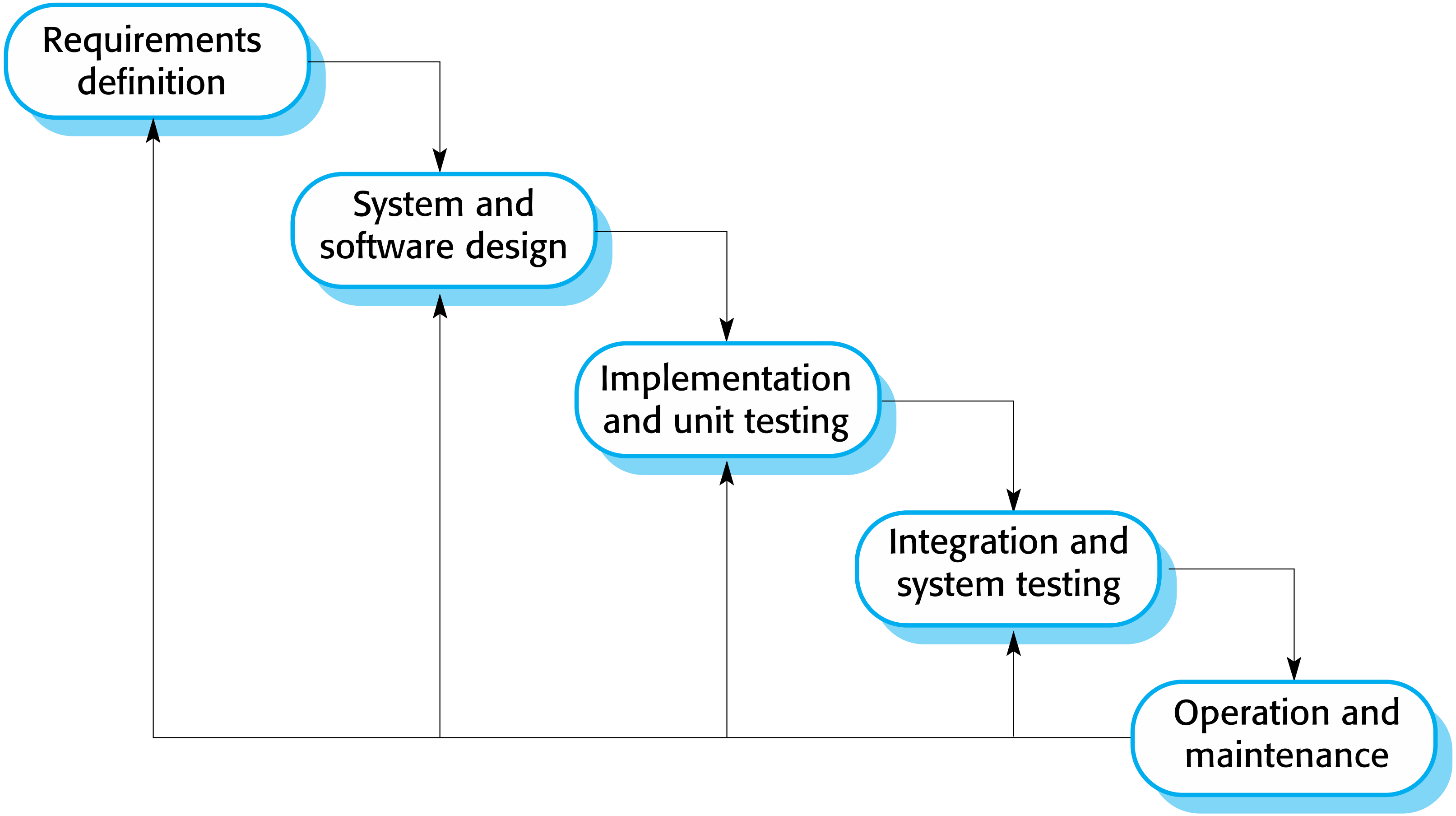
Incremental development model

- Specification, development and validation are interleaved
- Plan driven or agile

Reuse-oriented software engineering

- The system is assembled from existing components
- plan-driven or agile

The waterfall model



Requirements definition

System and software design

Implementation and unit testing

Integration and system testing

Operation and maintenance

Waterfall model phases

- Linear-sequential life cycle model
- Each phase must be completed before the next phase can begin
- Separate identified phases
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance

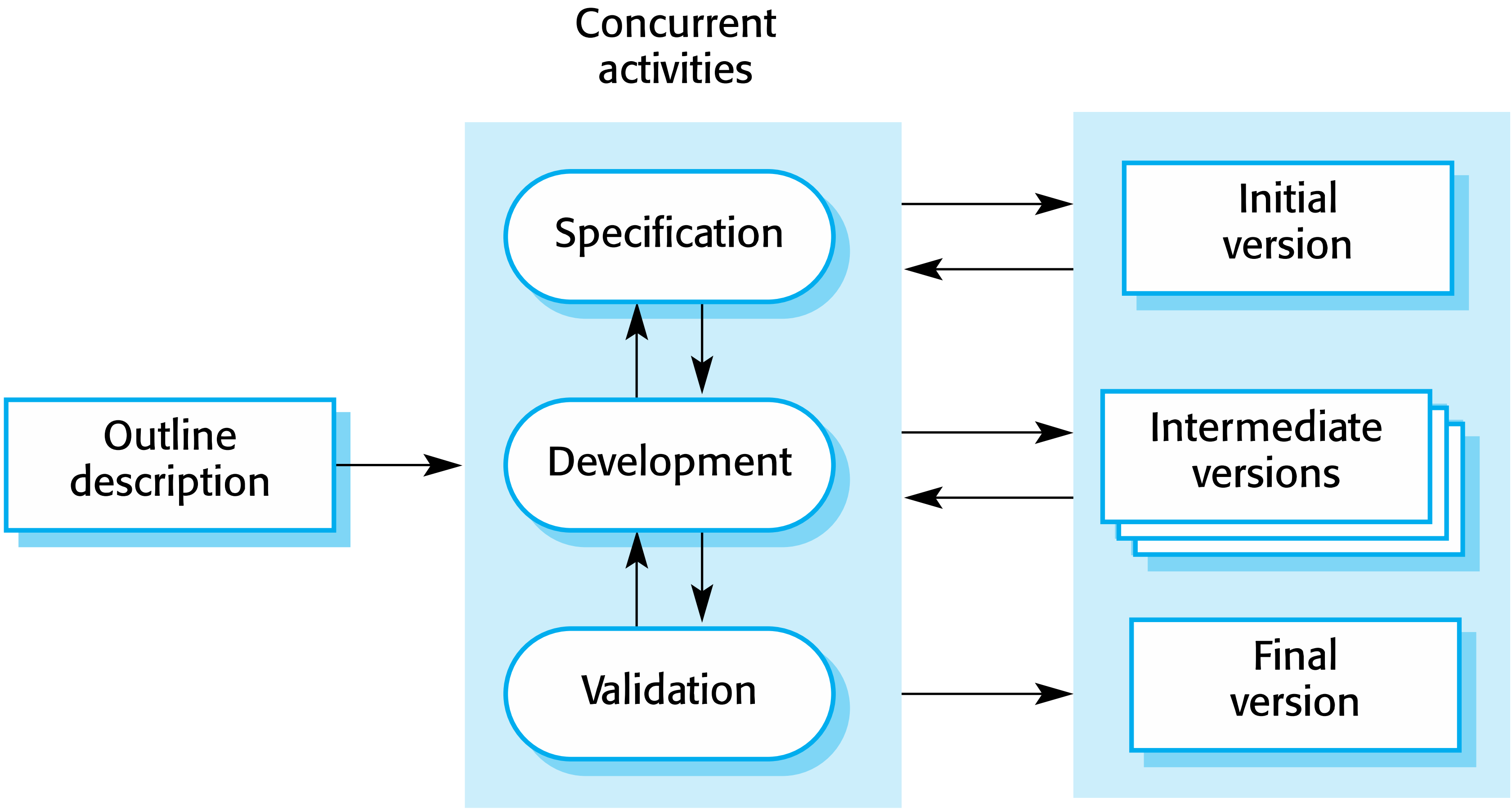
Drawbacks – Waterfall model

- Difficulty of accommodating change after the process is underway
- A phase has to completed before moving onto the next phase
- This model is mostly used for large systems engineering projects where a system is developed at several sites
  - In these situations, Plan driven nature of the waterfall model helps coordinate the work

When to use waterfall model

- Requirement are clear, fixed and very well known
- Product definition is stable
- Technology is understood
- No ambiguous requirements

Incremental development model



Incremental model

- Whole requirement divided into various builds
- Multiple development life cycle (multi-waterfall cycle)
- Cycles are divided into smaller, more easily managed modules
- Each module pass through - requirements, design, implementation & testing phases
- Initial version of the software is produced during the first module (i.e working software early on during the software life cycle)
- Subsequent releases add function to the previous release
- Process continues until the complete system is achieved

Advantages of Incremental model

- A working version of the software early during the software life cycle
- More rapid delivery and deployment of useful software to the customer is possible.
- Cost effective when accommodating customer changing requirements
- Quick comments and feedback on demonstrations of the software
- Lower initial delivery cost
- Easy to test and debug during a small iteration
- Difficult and risk parts can be left for the later iterations

Disadvantages of Incremental model

- Requires good planning & design
- Requires a clear and complete definition of the whole system before broking down into iterations
- Total cost is higher than waterfall
- Regular deliverables to measure progress
- When systems are developed quickly, it is not cost-effective to produce documents on every version of the system
- Regular changes might corrupt the overall system structure

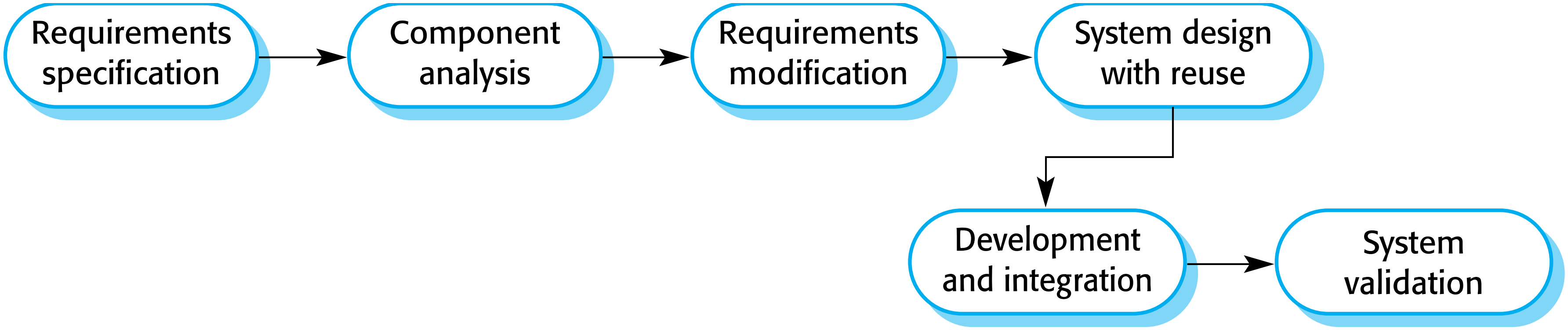
When to use incremental model?

- Requirements are clearly defined and understood
- Some details can evolve over time however major requirements should be well defined
- To get the product early to the market
- New technology is being used
- A system has some high risk features and goals

Reuse-oriented model

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Component analysis;
  - Requirements modification;
  - System design with reuse;
  - Development and integration.
- Reuse is now the standard approach for building many types of business system

Reuse-oriented model



Requirements specification

Component analysis

Requirements modification

System design with reuse

Development and integration

System validation

Exercise 2.1

- What software process model (waterfall or incremental)better suits for the development of the following systems:
  - E-learning system (like Moodle) for KAMK
  - A general purpose website
  - The control system for a nuclear plant
  - A simple game for an android phone

Please provide appropriate reasons for your answers.

Types of software component

- Web services
  - Develop according to service standard
  - Available for remote invocation
- Collection of objects developed as package with a component framework like .NET or J2EE
- Stand-alone software systems that are configured for use in particular environment

Process activities

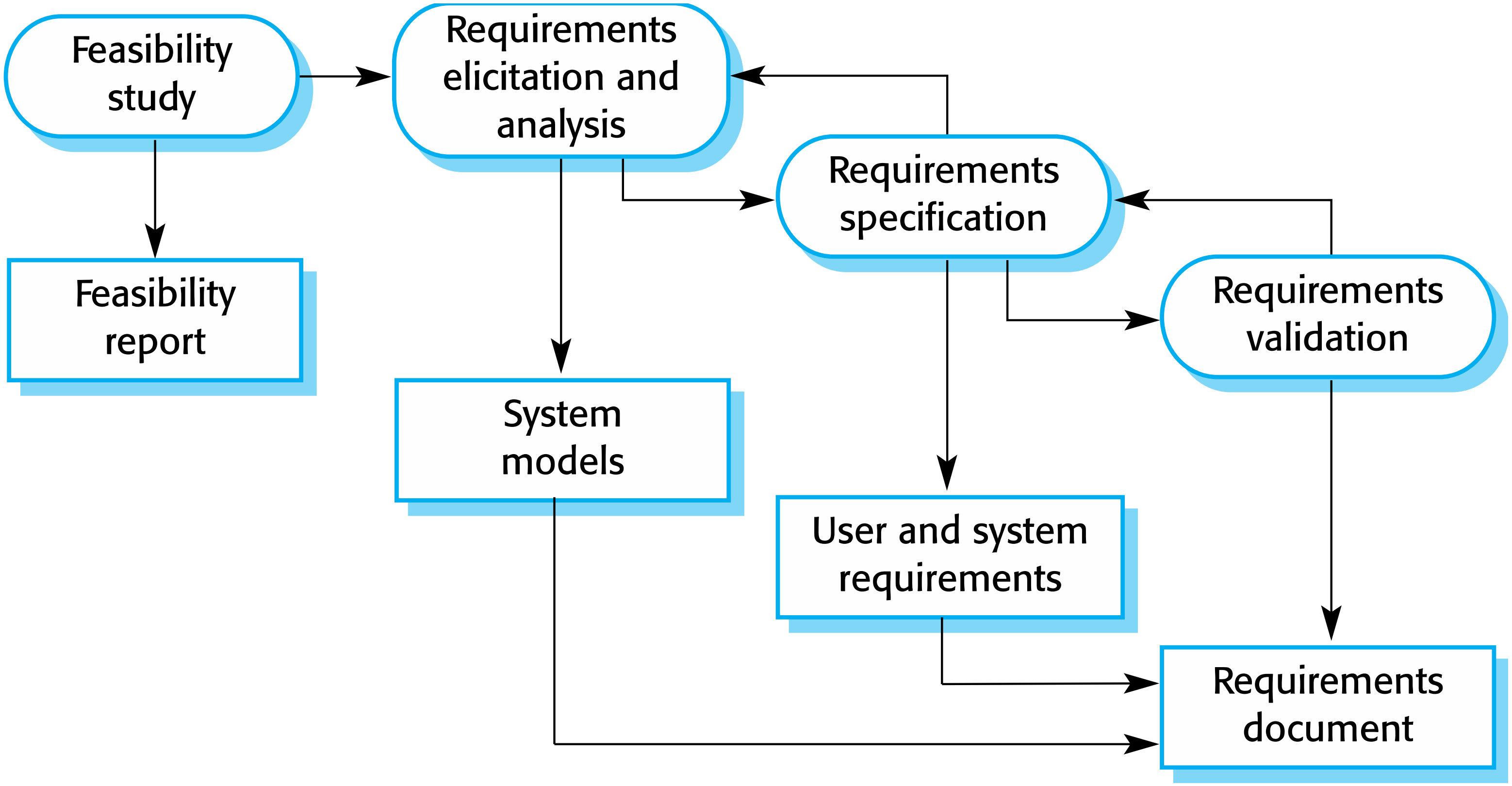
- Software processes are inter-leaved sequences of **technical, collaborative & managerial activities** with the overall goal of specifying, designing, implementing & testing a software system

- Process activities like specification, development, validation & evolution differ from one organization to other
  - In waterfall model, these activities are sequential
  - In incremental , they are interleaved

Software specification

- Process to establish
  - What services are required
  - What are the constraints on the system’s operation & development?
- Requirements engineering process
  - Feasibility study
    - Is it technically & financially feasible to build the system?
  - Requirements elicitation & 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

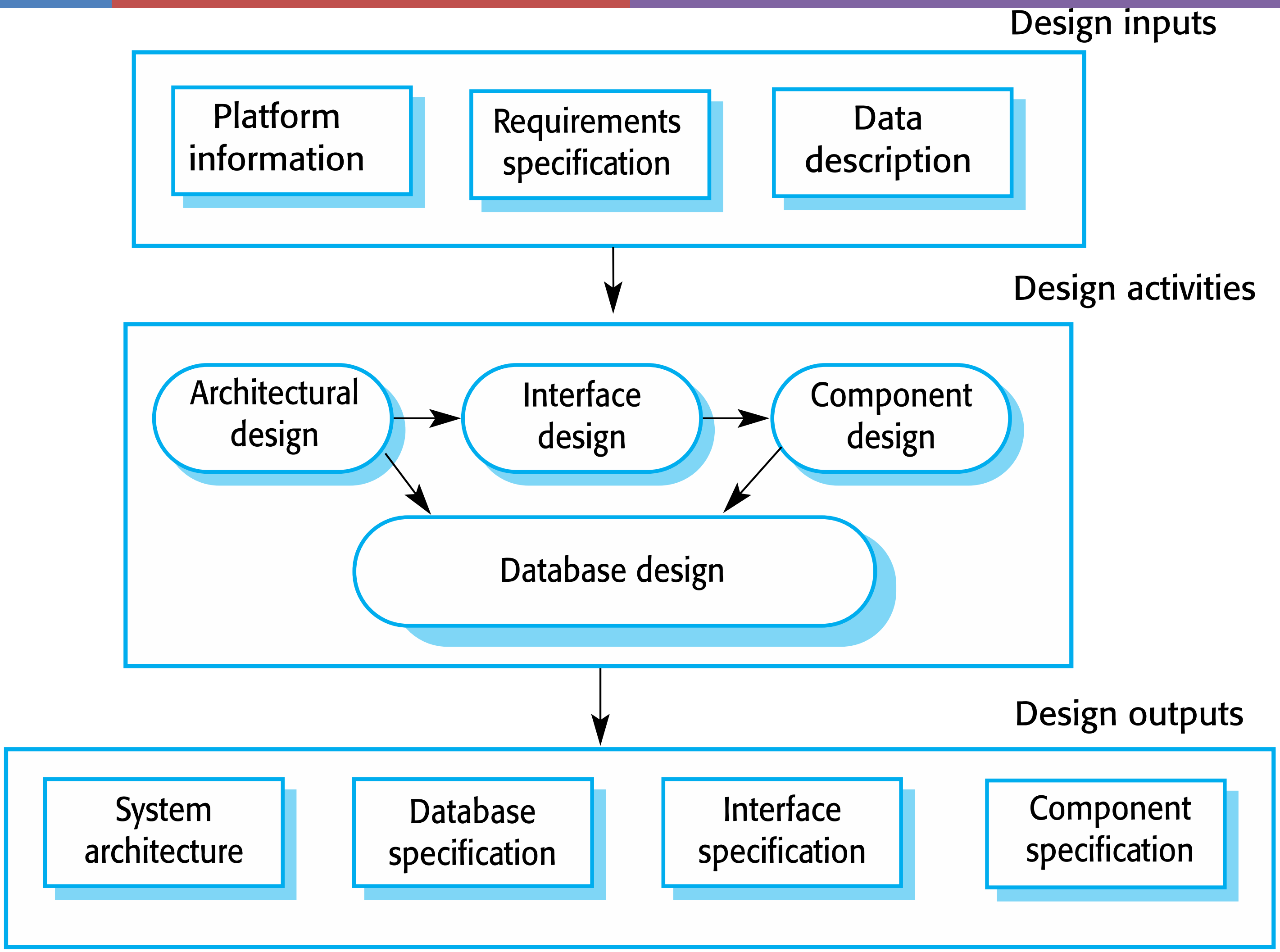
Requirements engineering process



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.

General model of the design process



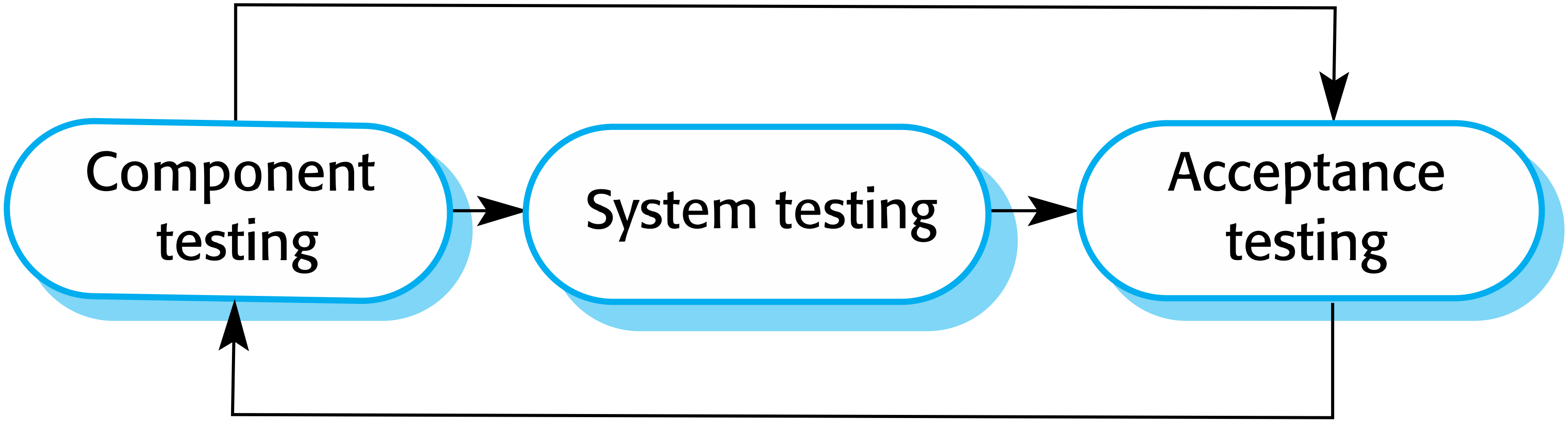
Design activities

- Architectual design
  - Identify the overall structure of the system, the principal components, subsystems or modules, their relationships and how they are distributed
- Interface design
  - Define the interfaces between system components
- Component design
  - Take each system component & design how it will operate
- Database design
  - Design the system data structure & how they are represented in a database

Software validation

- Verification and validation (V&V)
  - To ensure system conform to its specification & meets the requirements of the system consumer
- 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.

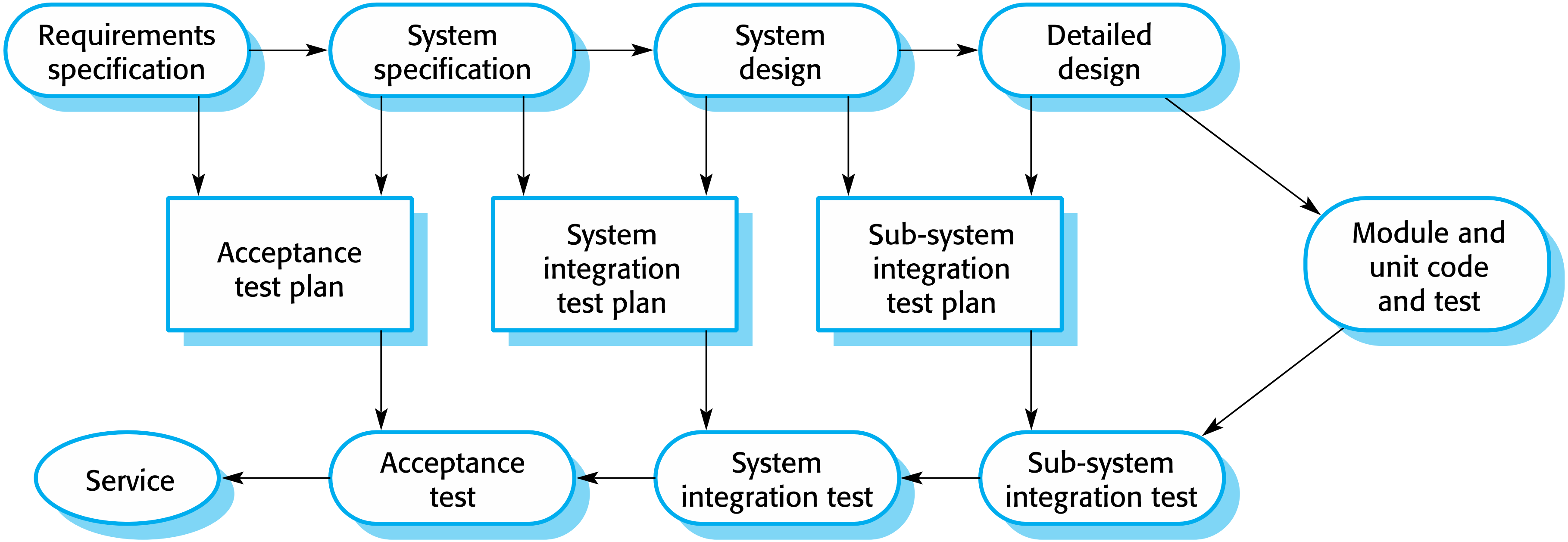
Stages of testing



Testing stages

- Development or component testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- Acceptance testing
  - Testing with customer data to check that the system meets the customer’s needs.

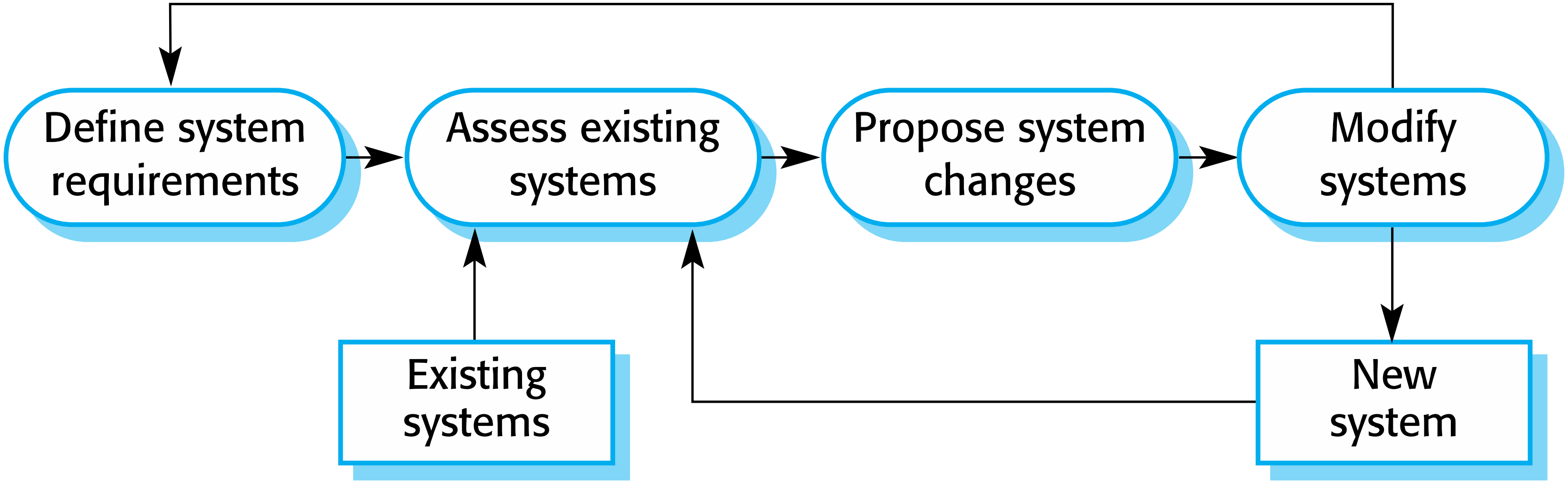
Testing phases in a plan-driven software process


Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and test
Service
Acceptance test
System integration test
Sub-system integration test

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



Summary

- Software processes
  - Activities involved in developing a software system
  - Process models are abstract representations of these processes
- General process models like
  - Waterfall model
  - Incremental Development
  - Reuse-oriented development
- Requirements engineering
  - The process of developing a software specification

Summary

- Design & implementation process
  - Concerned with transforming a requirements specification into an executable software system
- Software validation
  - Process of checking that the system conforms to its specification & that it meets the real needs of users of the system
- Software evolution
  - Takes place when changes are made to existing software systems to meet new requirements
  - Software must evolve to remain useful

Exercise 2.2

- Think of a simple software that you are planning to develop. List
  - What the system should do?
  - Is it technically and financially feasible to build the system?
  - Functional & non functional requirements
  - Validate your requirements
  - Draw the overall design of the system architectural design

Software Development Processes - Part 2

Ohjelmankehityspr., versionhallinta ja testaus – Chapter 5

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Deepak K.C. ; deepak.kc@kamk.fi ;

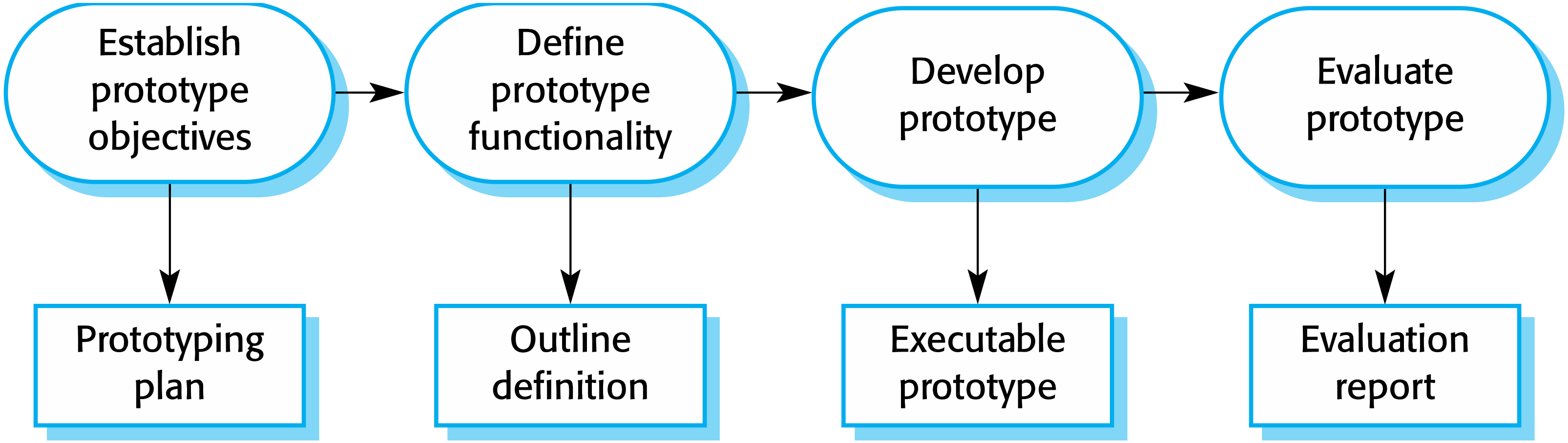
Software prototyping

- A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation;
  - In design processes to explore options and develop a UI design;
  - In the testing process to run back-to-back tests.

Benefits of prototyping

- Improved system usability.
- A closer match to users’ real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

The process of prototype development



Prototype development

- May be based on rapid prototyping languages or tools
- May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

Throw-away prototypes

- Prototypes should be discarded after development as they are not a good basis for a production system:
  - It may be impossible to tune the system to meet non-functional requirements;
  - Prototypes are normally undocumented;
  - The prototype structure is usually degraded through rapid change;
  - The prototype probably will not meet normal organisational quality standards.

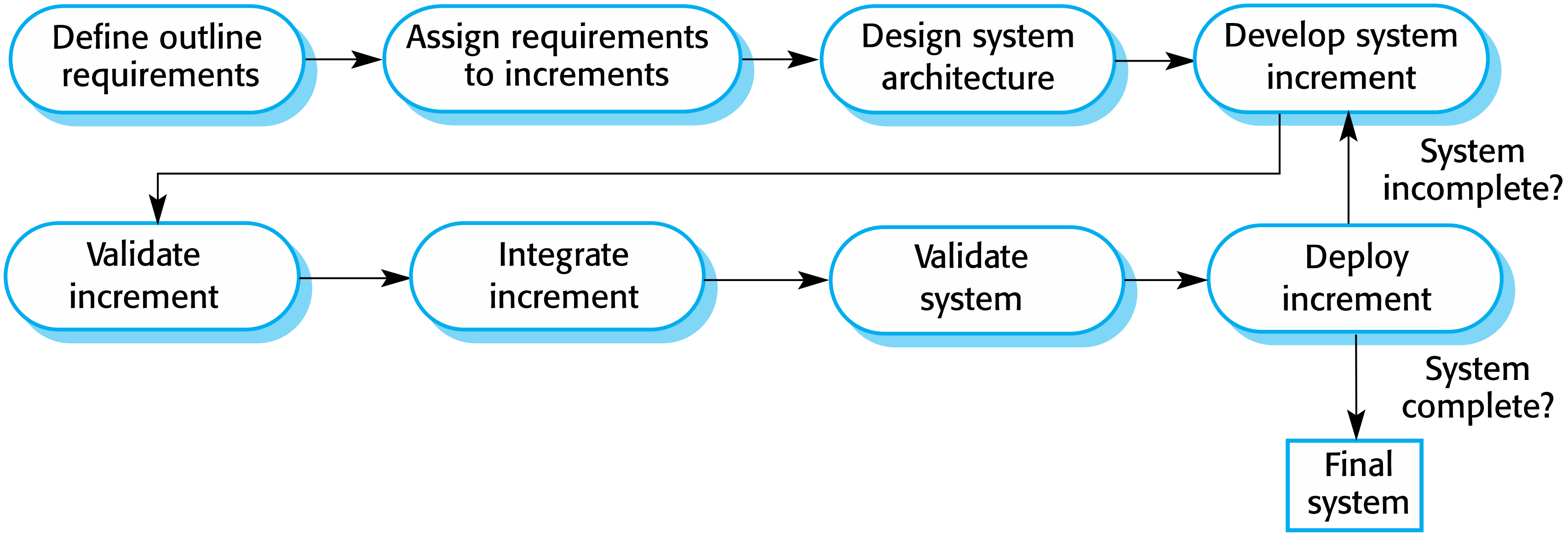
Incremental delivery

- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritised and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

Incremental development and delivery

- Incremental development
  - Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
  - Normal approach used in agile methods;
  - Evaluation done by user/customer proxy.
- Incremental delivery
  - Deploy an increment for use by end-users;
  - More realistic evaluation about practical use of software;
  - Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Incremental delivery



Incremental delivery advantages

- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

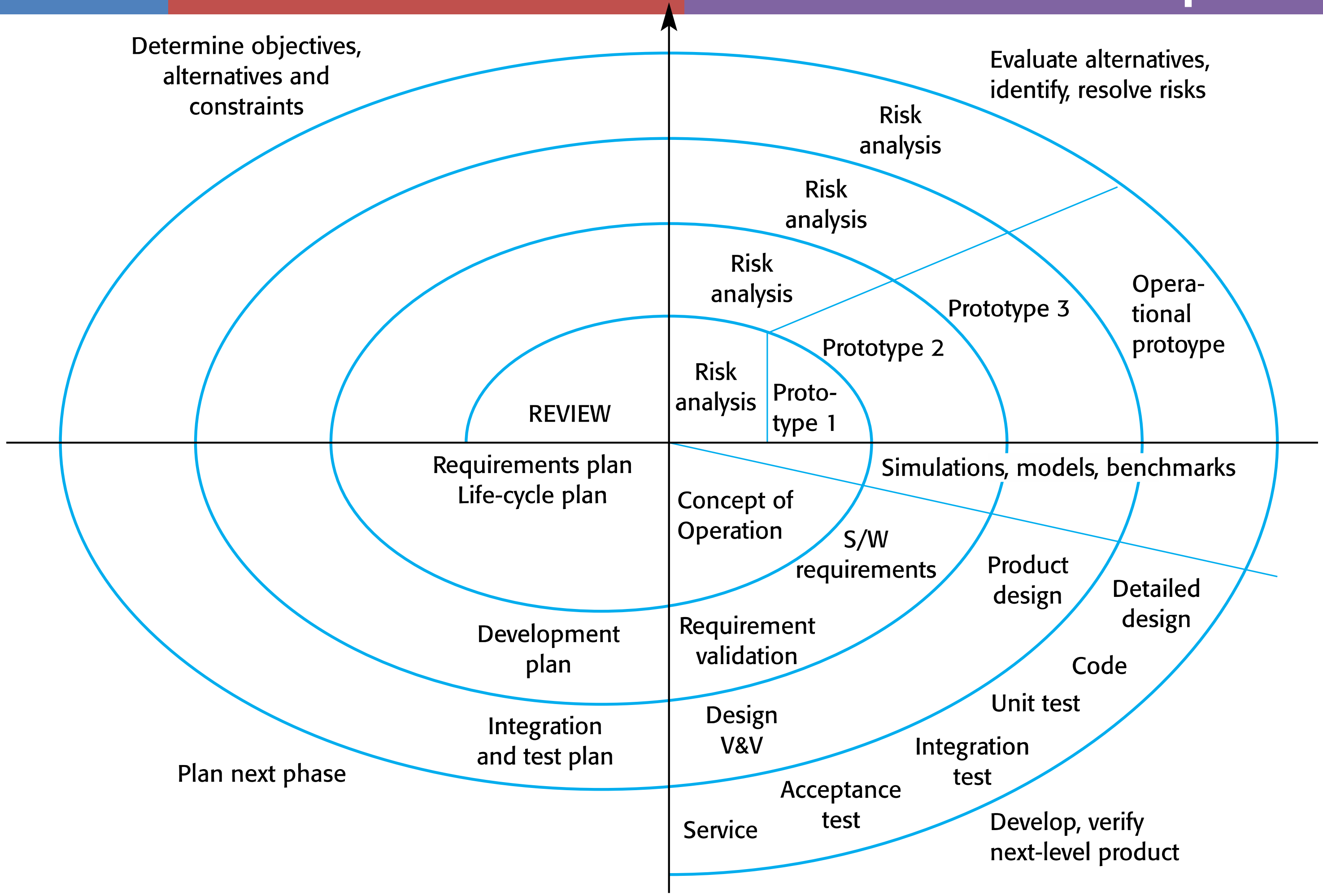
Incremental delivery problems

- Most systems require a set of basic facilities that are used by different parts of the system.
  - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
- The essence of iterative processes is that the specification is developed in conjunction with the software.
  - However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

Boehm’s spiral model

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

Boehm's spiral model of the software process



Spiral model sectors

- Objective setting
  - Specific objectives for the phase are identified.
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks.
- Development and validation
  - A development model for the system is chosen which can be any of the generic models.
- Planning
  - The project is reviewed and the next phase of the spiral is planned.

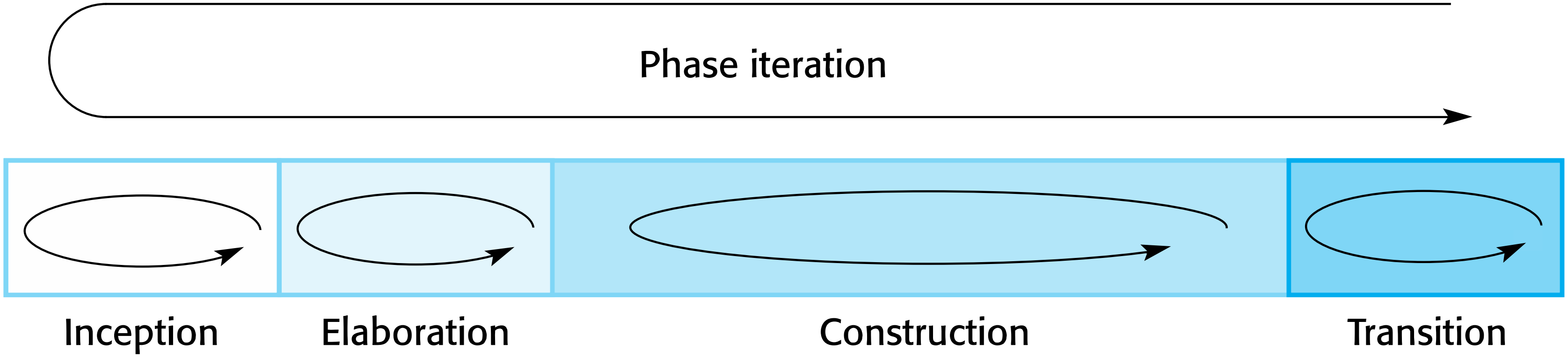
Spiral model usage

- Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- In practice, however, the model is rarely used as published for practical software development.

The Rational Unified Process

- A modern generic process derived from the work on the UML and associated process.
- Brings together aspects of the 3 generic process models discussed previously.
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A practice perspective that suggests good practice.

Phases in the Rational Unified Process



RUP phases

- Inception
  - Establish the business case for the system.
- Elaboration
  - Develop an understanding of the problem domain and the system architecture.
- Construction
  - System design, programming and testing.
- Transition
  - Deploy the system in its operating environment.

RUP iteration

- In-phase iteration
  - Each phase is iterative with results developed incrementally.
- Cross-phase iteration
  - As shown by the loop in the RUP model, the whole set of phases may be enacted incrementally.

Static workflows in the Rational Unified Process

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.

Static workflows in the Rational Unified Process

Workflow	Description
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system (see Chapter 25).
Project management	This supporting workflow manages the system development (see Chapters 22 and 23).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

RUP good practice

- Develop software iteratively
  - Plan increments based on customer priorities and deliver highest priority increments first.
- Manage requirements
  - Explicitly document customer requirements and keep track of changes to these requirements.
- Use component-based architectures
  - Organize the system architecture as a set of reusable components.

RUP good practice

- Visually model software
  - Use graphical UML models to present static and dynamic views of the software.
- Verify software quality
  - Ensure that the software meet’s organizational quality standards.
- Control changes to software
  - Manage software changes using a change management system and configuration management tools.

References

- Software Engineering, 9th Edition by Ian Sommerville