
Software Processes

Dong-Ah Lee

Objectives

- To introduce software process models
- To describe three generic process models and when they may be used
- To describe outline process models for requirements engineering, software development, testing and evolution
- To explain the Rational Unified Process model
- To introduce CASE technology to support software process activities

Topics covered

- Software process models
- Process iteration
- Process activities
- The Rational Unified Process
- Computer-aided software engineering

The software process

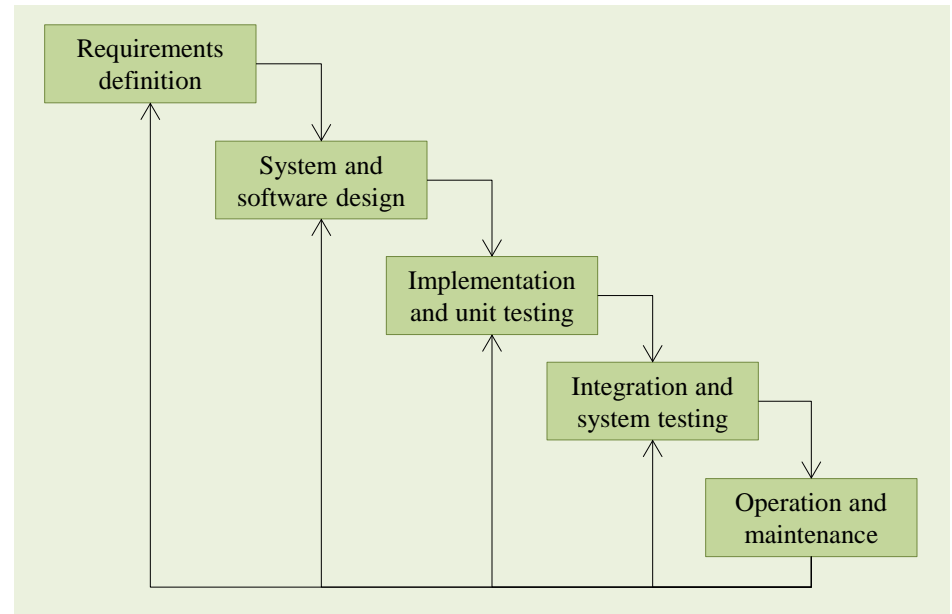
- A structured set of activities required to develop a software system
 - Specification;
 - Design;
 - Validation;
 - Evolution.
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

Generic software process models

- The waterfall model
 - Separate and distinct phases of specification and development.
- Evolutionary development
 - Specification, development and validation are interleaved.
- Component-based software engineering
 - The system is assembled from existing components.
- There are many variants of these models e.g. formal development where a waterfall-like process is used but the specification is a formal specification that is refined through several stages to an implementable design.

Waterfall model

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance



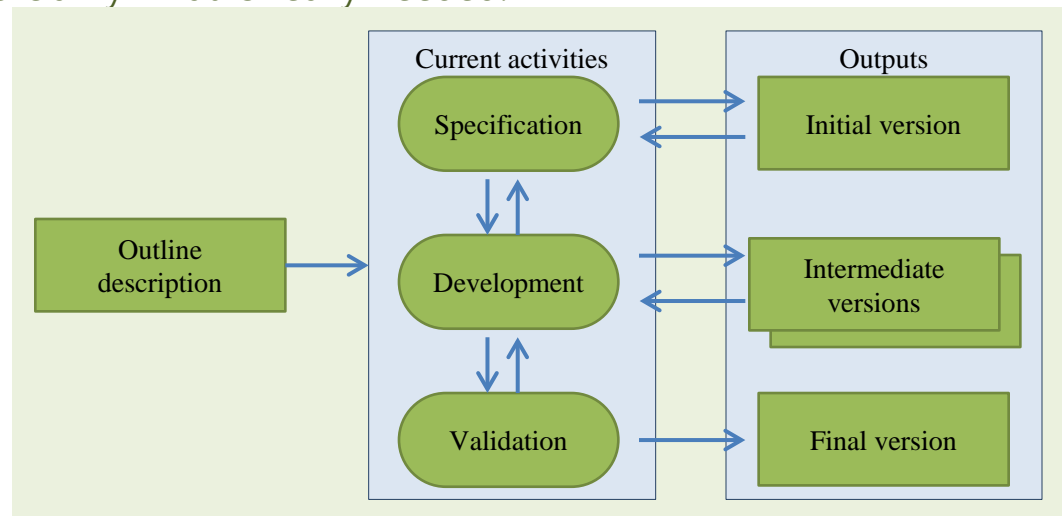
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. One phase has to be complete before moving onto the next phase.

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.
- 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.

Evolutionary development

- Exploratory development
 - Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements and add new features as proposed by the customer.
- Throw-away prototyping
 - Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

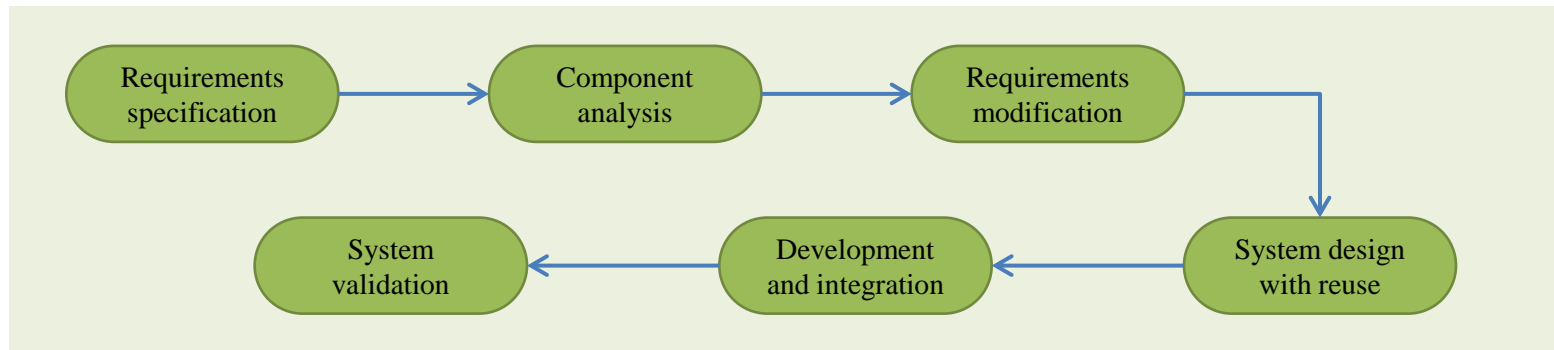


Evolutionary development

- Problems
 - Lack of process visibility;
 - Systems are often poorly structured;
 - Special skills (e.g. in languages for rapid prototyping) may be required.
- Applicability
 - For small or medium-size interactive systems;
 - For parts of large systems (e.g. the user interface);
 - For short-lifetime systems.

Component-based software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages



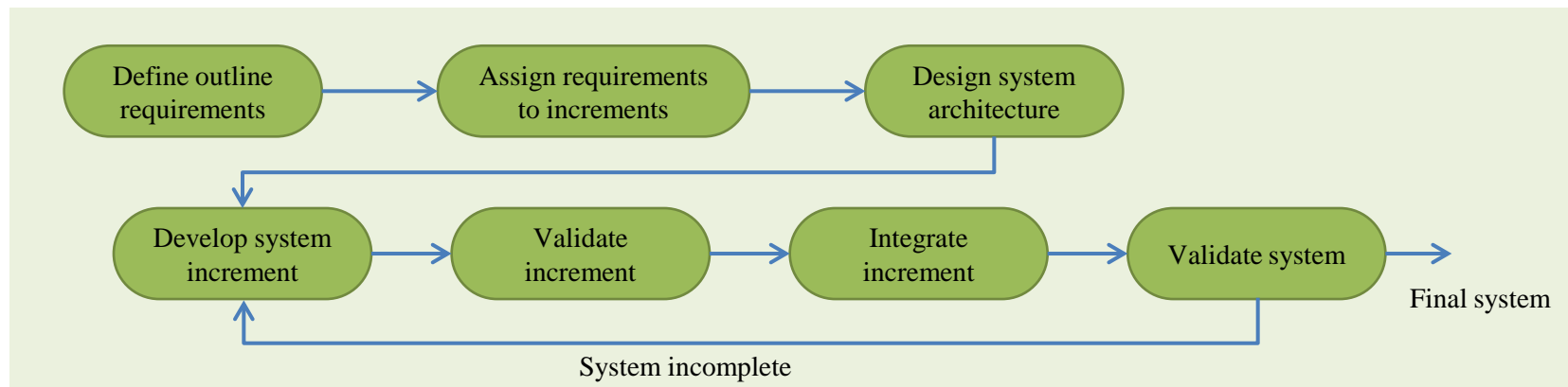
- This approach is becoming increasingly used as component standards have emerged.

Process iteration

- System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems.
- Iteration can be applied to any of the generic process models.
- Two (related) approaches
 - Incremental delivery;
 - Spiral development.

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.



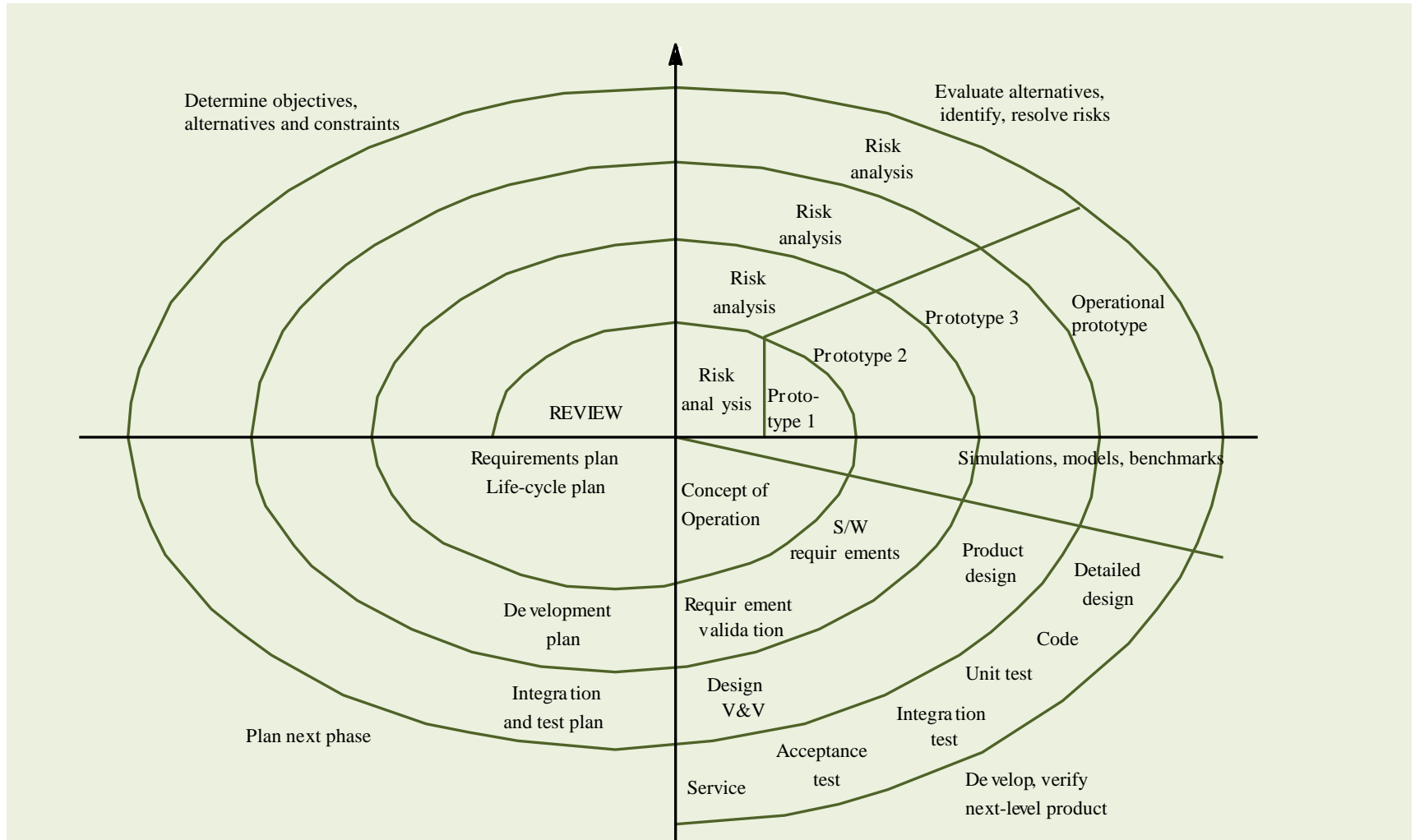
Incremental development advantages

- 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.

Spiral development

- 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.

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.

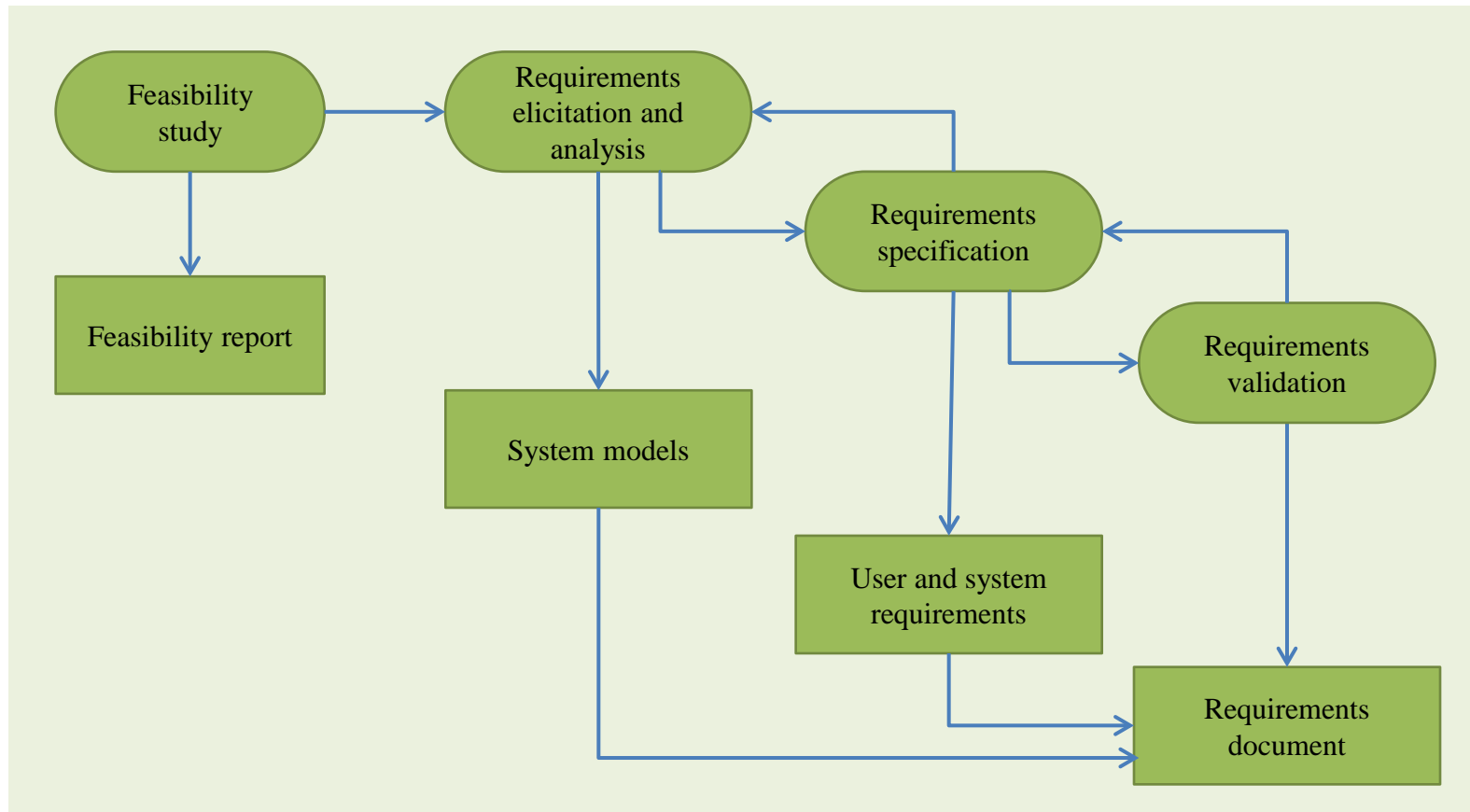
Process activities

- Software specification
- Software design and implementation
- Software validation
- Software evolution

Software specification

- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
 - Feasibility study;
 - Requirements elicitation and analysis;
 - Requirements specification;
 - Requirements validation.

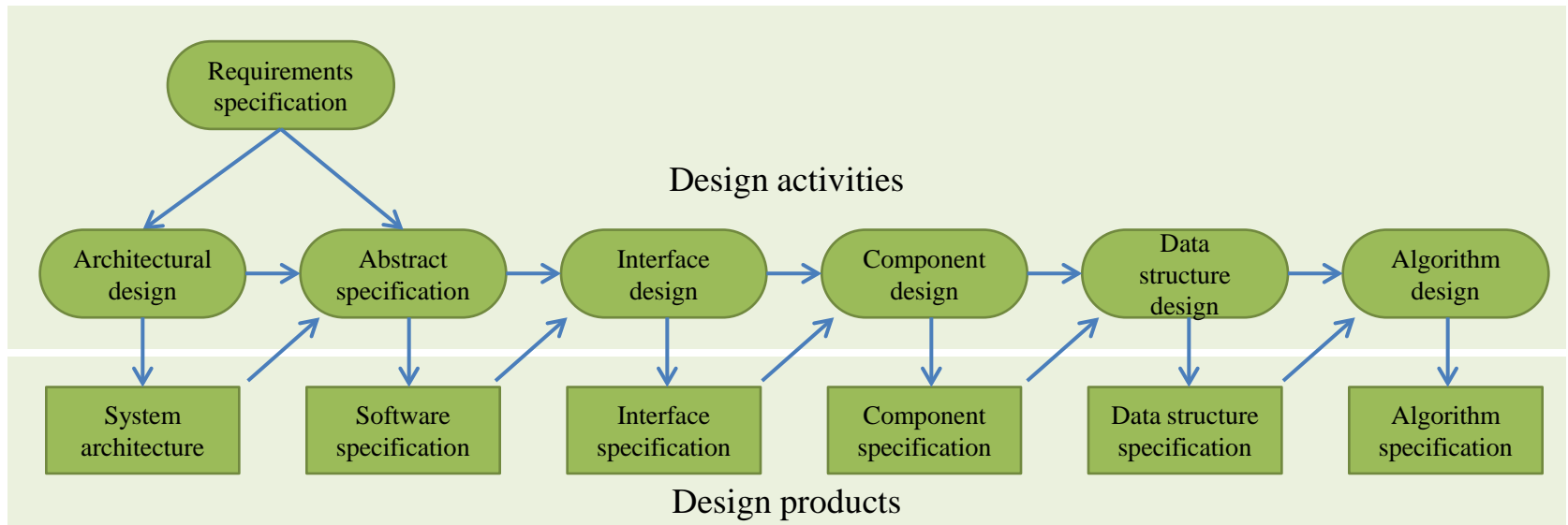
The requirements engineering process



Software design and 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 interleaved.

The software design process

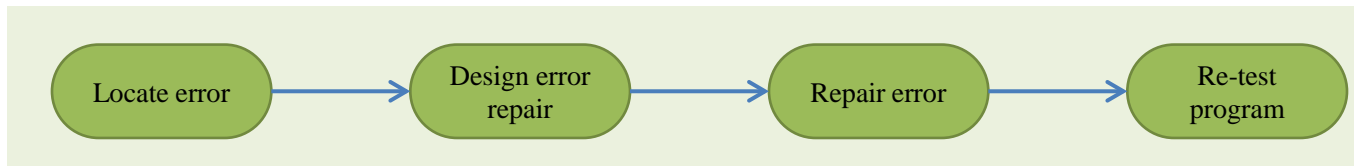


Structured methods

- Systematic approaches to developing a software design.
- The design is usually documented as a set of graphical models.
- Possible models
 - Object model;
 - Sequence model;
 - State transition model;
 - Structural model;
 - Data-flow model.

Programming and debugging

- Translating a design into a program and removing errors from that program.
- Programming is a personal activity - there is no generic programming process.
- Programmers carry out some program testing to discover faults in the program and remove these faults in the debugging process.

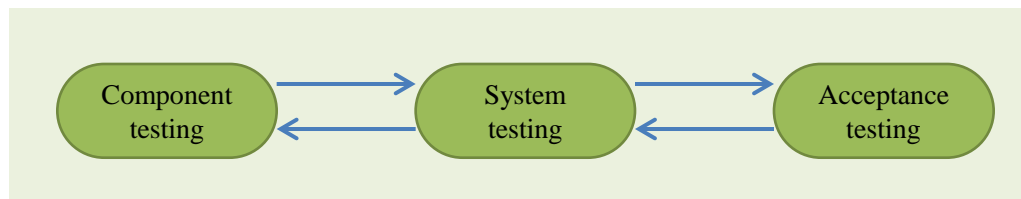


Software validation

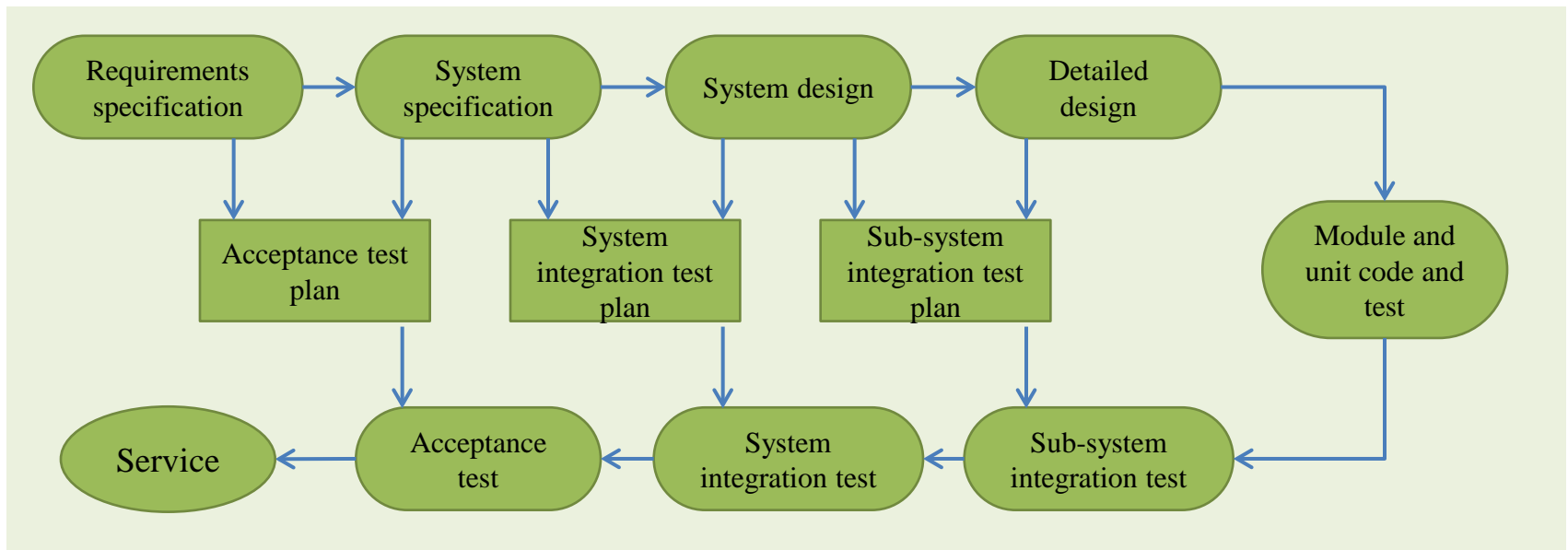
- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the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

Testing stages

- Component or unit 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.

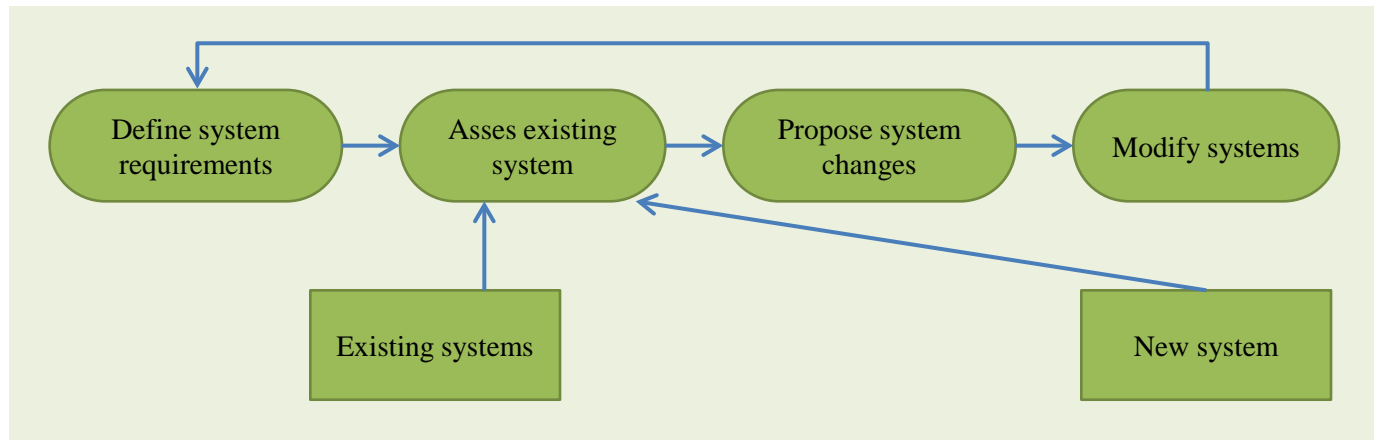


Testing phases



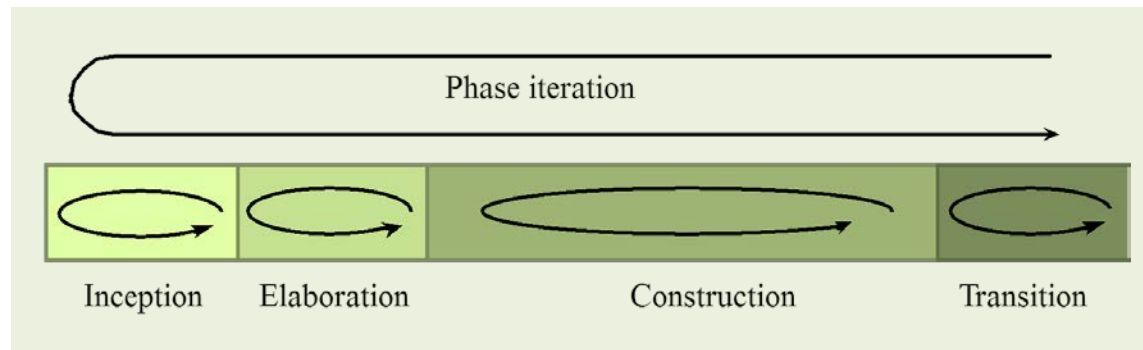
Software evolution

- 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.



The Rational Unified Process

- A modern process model derived from the work on the UML and associated process.
- Normally described from 3 perspectives
 - A dynamic perspective that shows phases over time;
 - A static perspective that shows process activities;
 - A practical perspective that suggests good practice.



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 good practice

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

Static workflows of RUP

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.
Test	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 29).
Project management	This supporting workflow manages the system development (see Chapter 5).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Key points

- Software processes are the activities involved in producing and evolving a software system.
- Software process models are abstract representations of these processes.
- General activities are specification, design and implementation, validation and evolution.
- Generic process models describe the organisation of software processes. Examples include the waterfall model, evolutionary development and component-based software engineering.
- Iterative process models describe the software process as a cycle of activities.

Key points

- Requirements engineering is the process of developing a software specification.
- Design and implementation processes transform the specification to an executable program.
- Validation involves checking that the system meets to its specification and user needs.
- Evolution is concerned with modifying the system after it is in use.
- The Rational Unified Process is a generic process model that separates activities from phases.