SOFTWARE ENGINEERING

Chapter 2 – Software Processes

WEEK 2

Topics covered

- Software process models
- Process activities
- Coping with change
- The Rational Unified Process
  - An example of a modern software process.

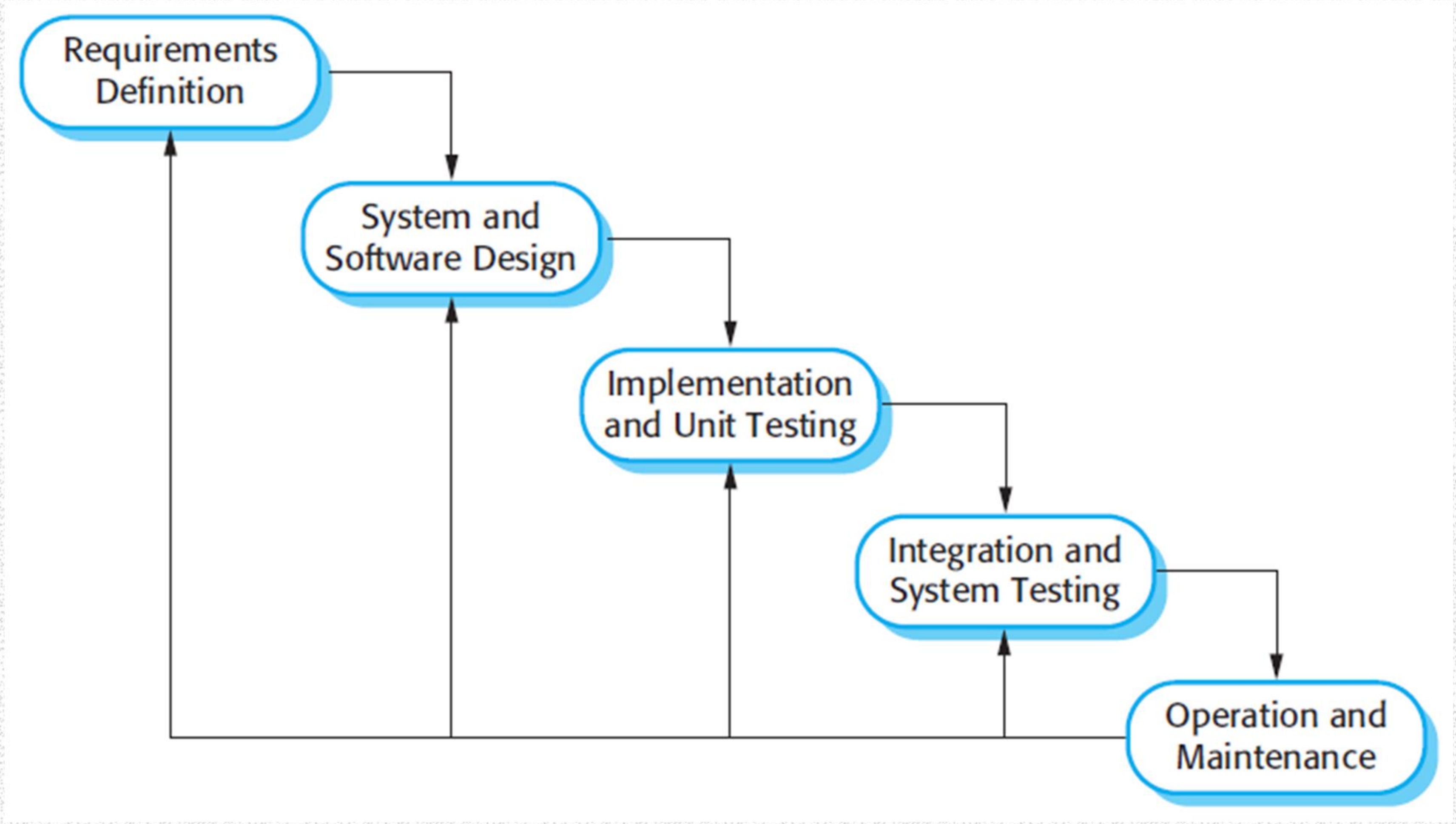
The software process

- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
  - Specification
  - Design and implementation
  - Validation
  - Evolution.
- A software process model
  - an abstract representation of a process

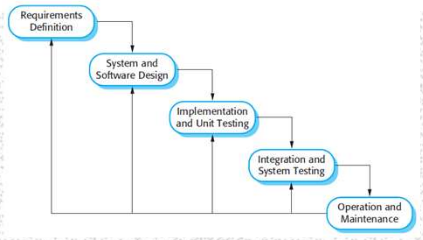
Some software process models

- 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*.
- Reuse-oriented software engineering
  - The system is assembled from existing components.
  - May be 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 problem

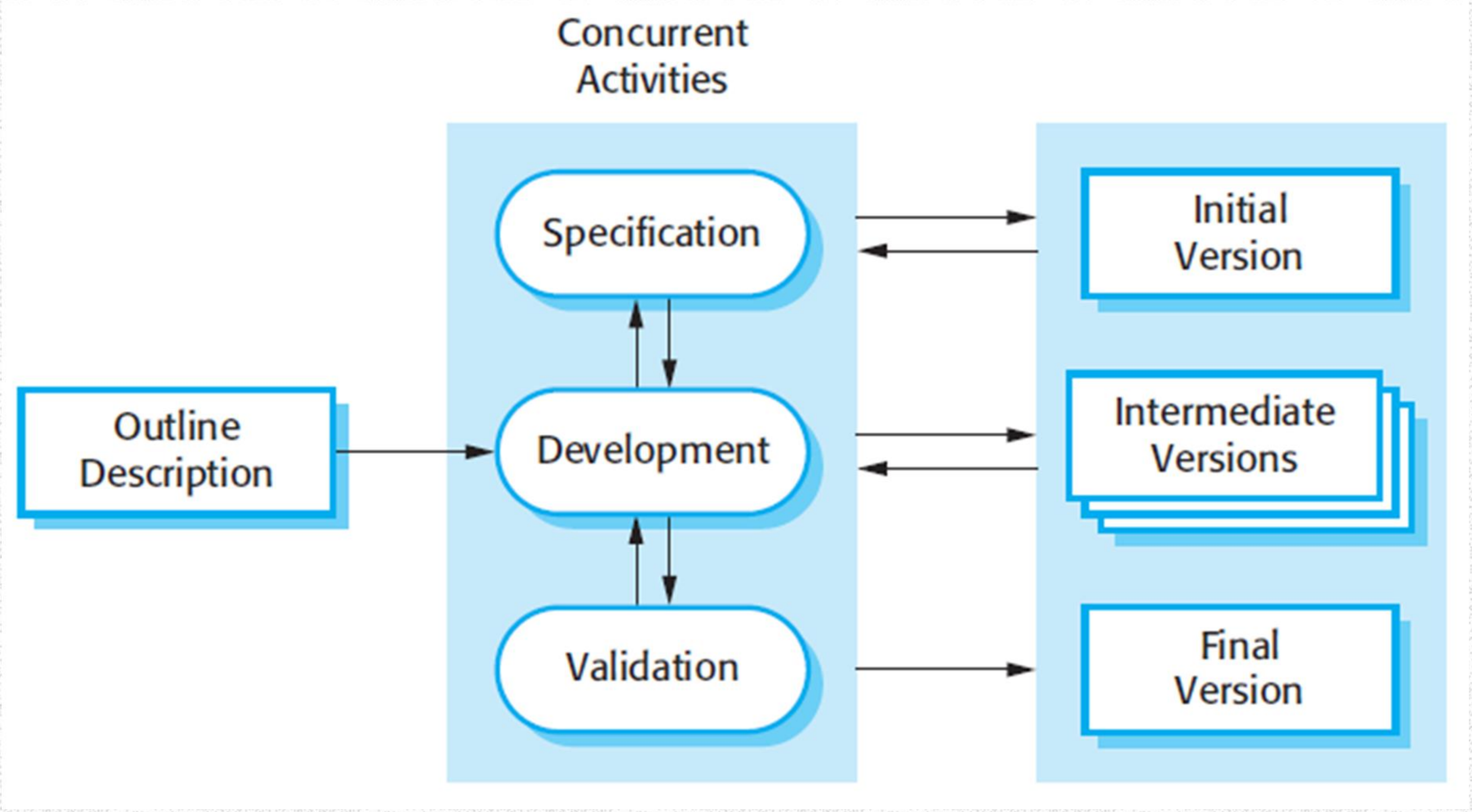

Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

- Difficult to accommodate changes after the process is underway.
  - In principle, a phase has to be complete before moving onto the next phase.

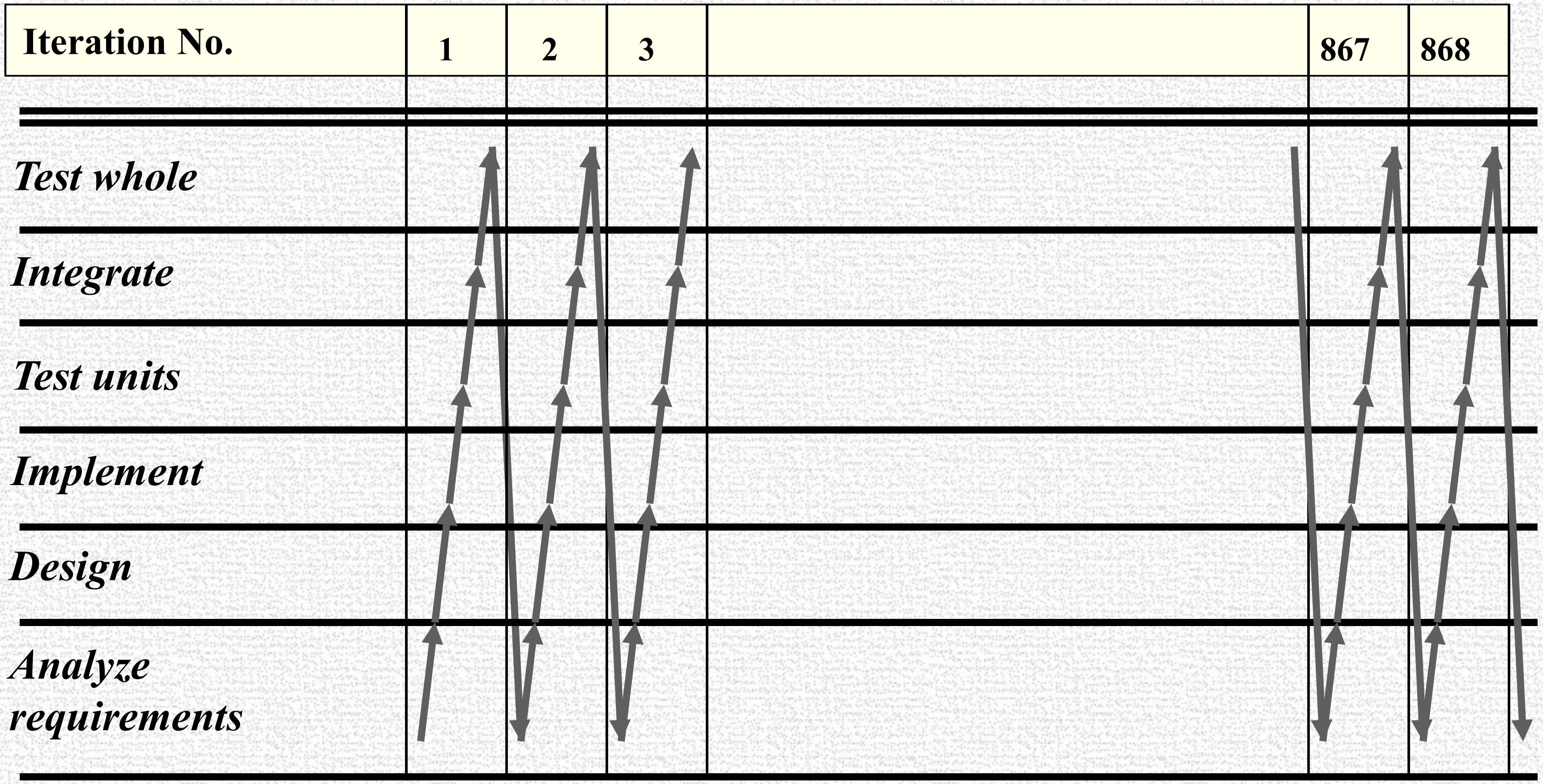
Waterfall model usages

- Mostly used for large systems engineering projects
  - a system is developed at several sites.
  - the plan-driven nature of the waterfall model helps coordinate the work.
- When the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.

Incremental development



Incremental development – Another view



Incremental development benefits

- Reduce the cost of accommodating changing customer requirements
- Easier to get customer feedback on the development work that has been done.
- More rapid delivery and deployment of useful software to the customer

Incremental development problems

- The process is not visible.
  - Managers need regular deliverables
  - Not cost-effective to produce documents for every product version
- System structure tends to degrade as new increments are added.
  - Need time and money on refactoring to improve the software
  - Regular change tends to corrupt the structure.
  - Incorporating further software changes becomes increasingly difficult and costly.

Reuse-oriented software engineering

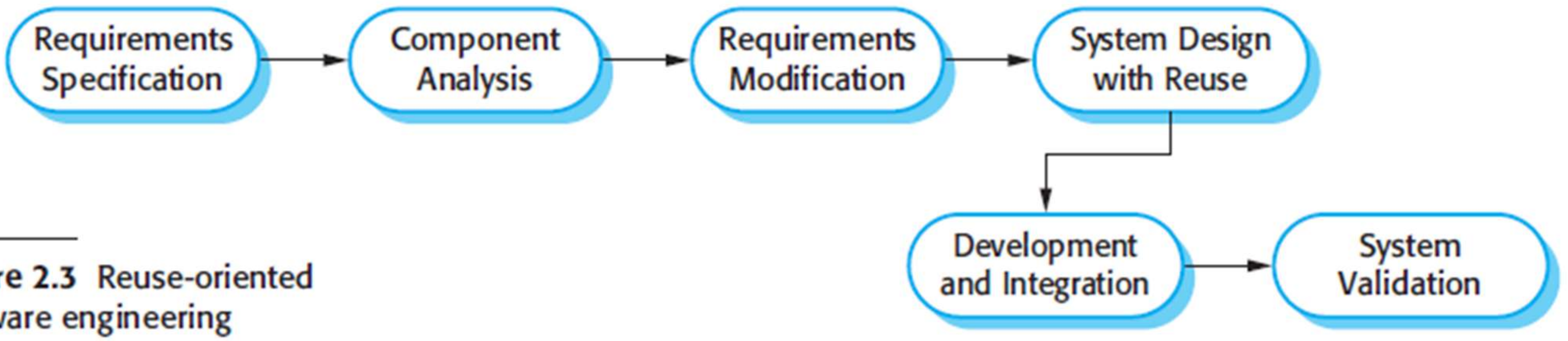

Requirements Specification
Component Analysis
Requirements Modification
System Design with Reuse
Development and Integration
System Validation

Figure 2.3 Reuse-oriented software engineering

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

Reuse-oriented software engineering Types of software component

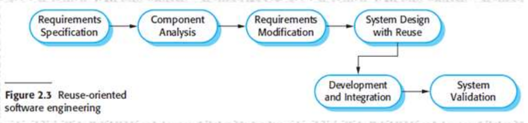

Requirements Specification
Component Analysis
Requirements Modification
System Design with Reuse
Development and Integration
System Validation

Figure 2.3 Reuse-oriented software engineering

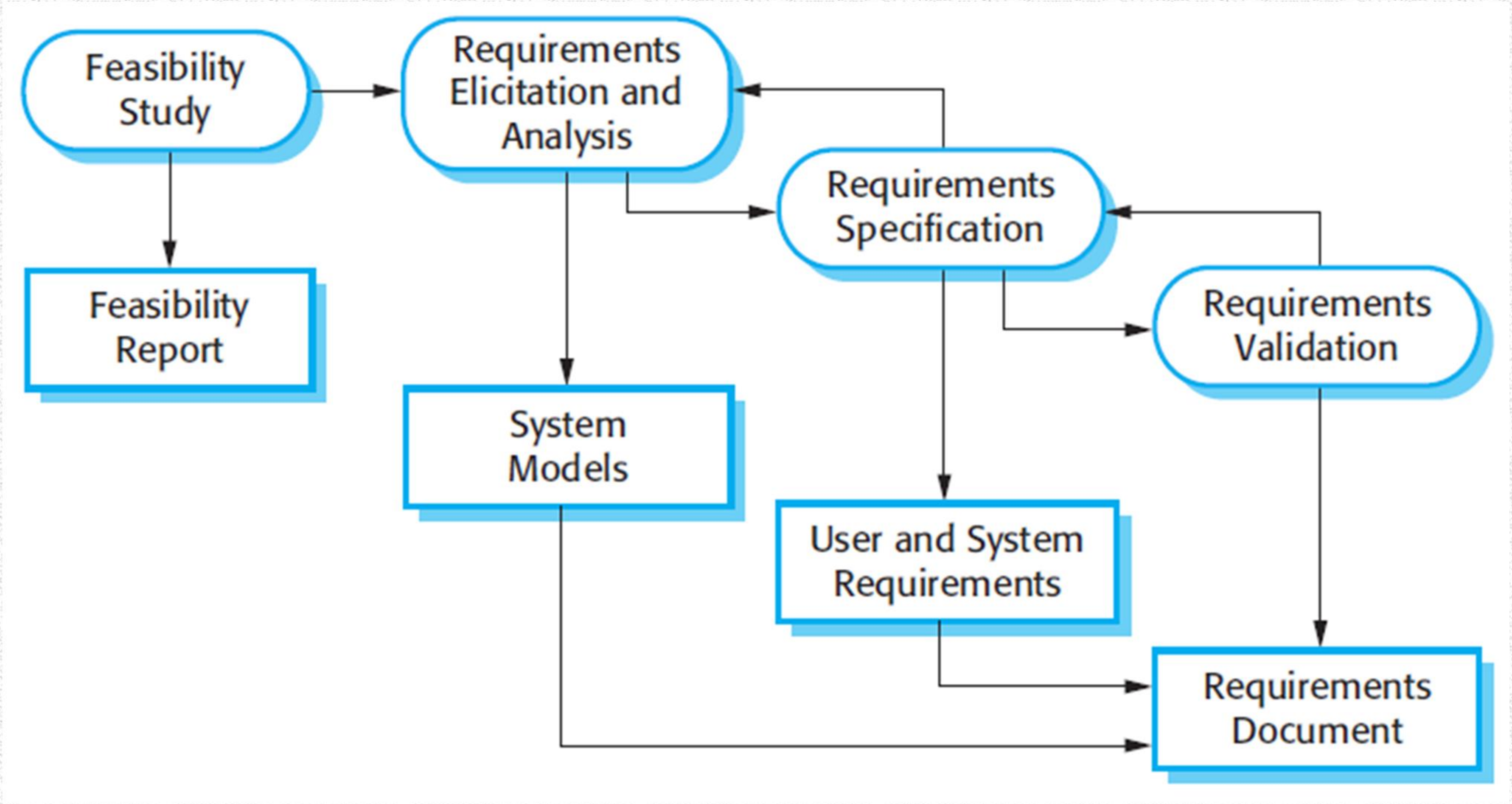
- Web services
  - developed according to service standards and which are available for remote invocation.
- Collections of objects
  - developed as a package to be integrated with a component framework such as .NET or J2EE.
- Stand-alone software systems (COTS)
  - configured for use in a particular environment.

PROCESS ACTIVITIES

Activity: Software specification

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

The requirements engineering process

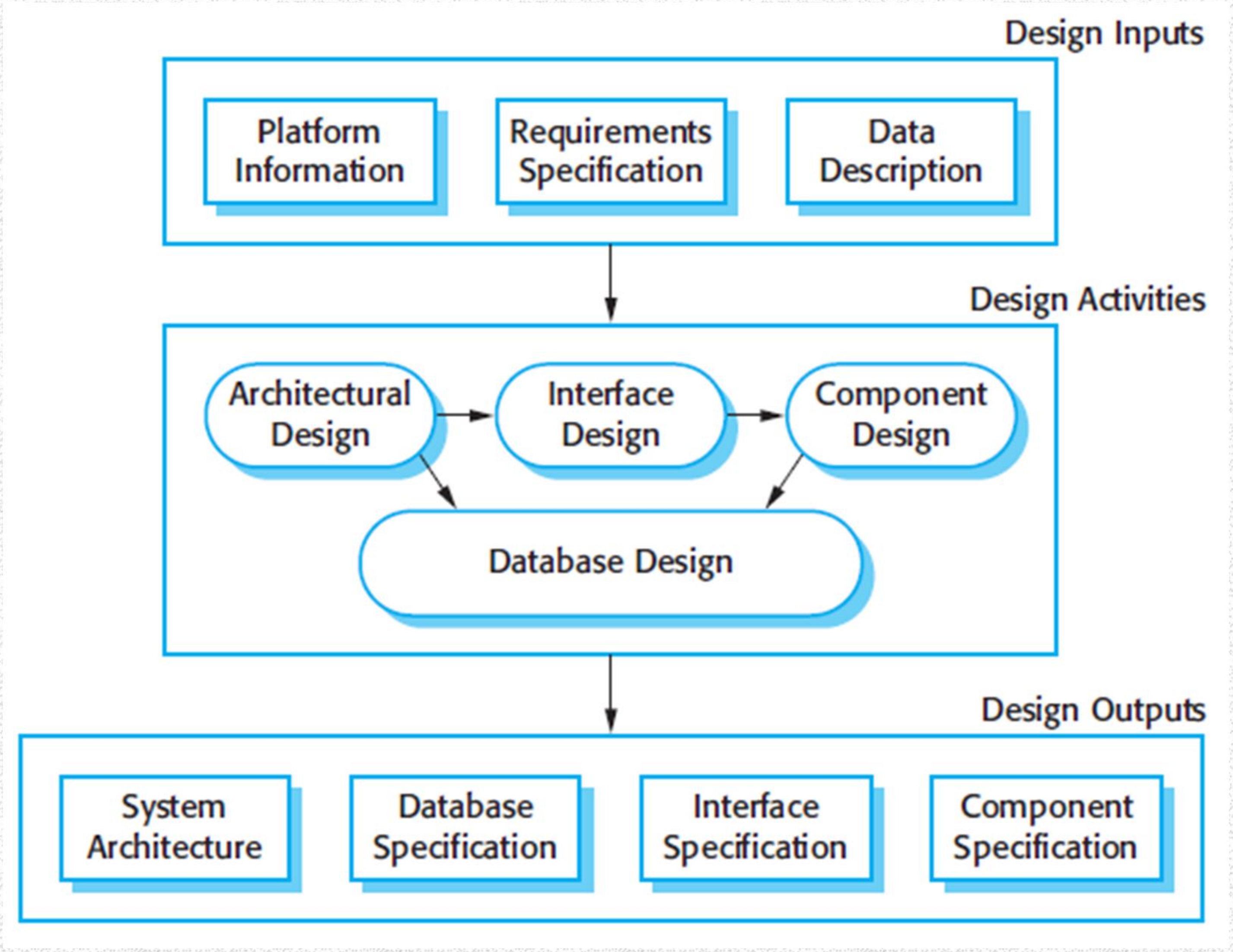


Activity: Software design and implementation ~ Software development

- The process of <u>converting the system specification into an executable system.</u>

- Two (sub) activities:
  - 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.

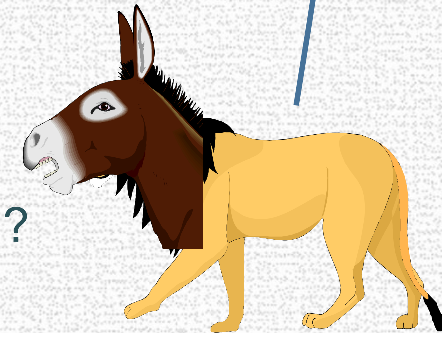
A general model of the design process



Activity: Software validation

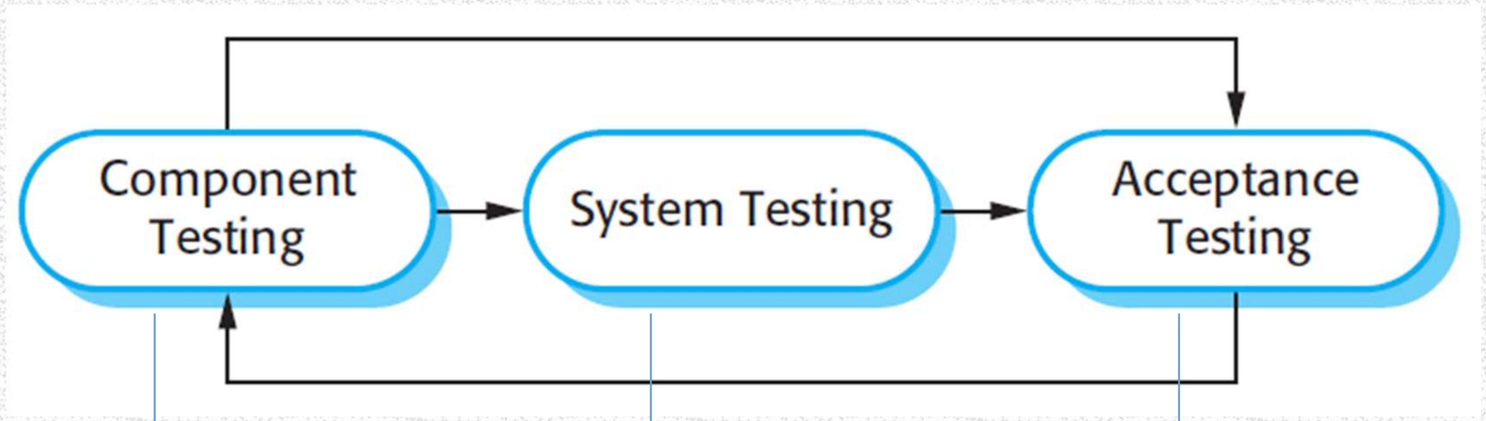
building the thing right?

- Verification and validation (V & V)
  - 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: executing the system with test cases
  - Testing: the most commonly used V & V activity.



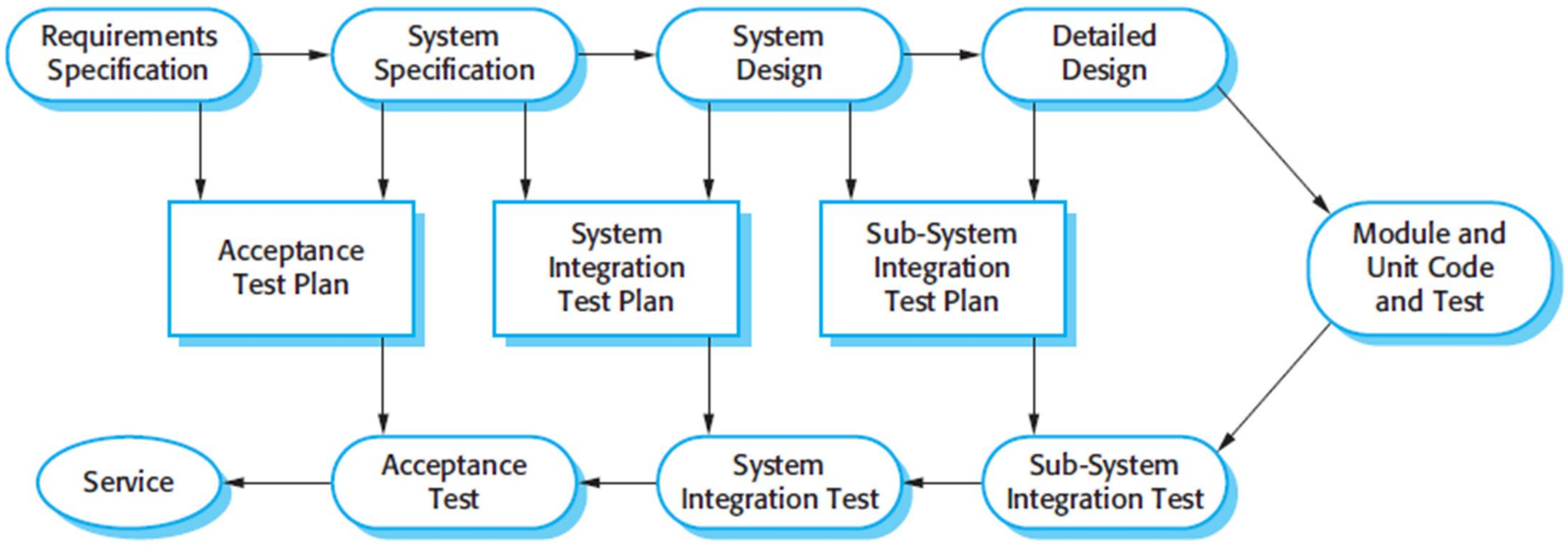
building the right thing?

Stages of testing



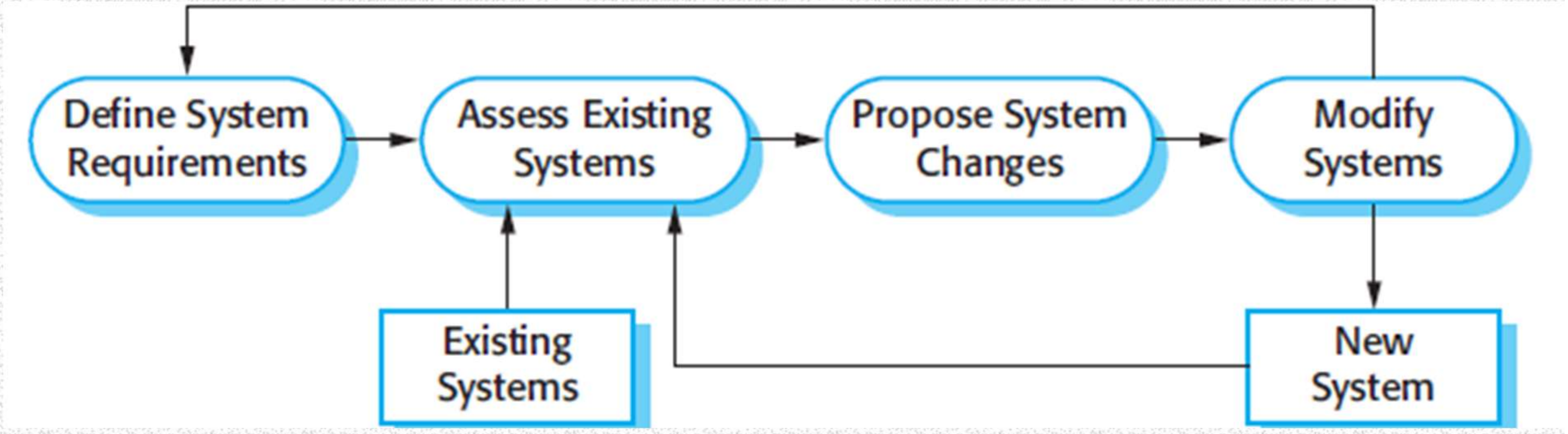
- Test individual components independently
- Testing of the system as a whole
- Testing with customer data

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

Activity: Software evolution

- Software is inherently flexible and can change.
- Requirements can change (changing business circumstances) ⇒ the software must also evolve and change.
- Process:



COPYING WITH CHANGE

Coping with change

- Change is inevitable in all large software projects.
  - Business changes
  - New technologies
  - Changing platforms
- Change leads to rework
  - costs include rework (re-analysing requirements) and implementing new functionality

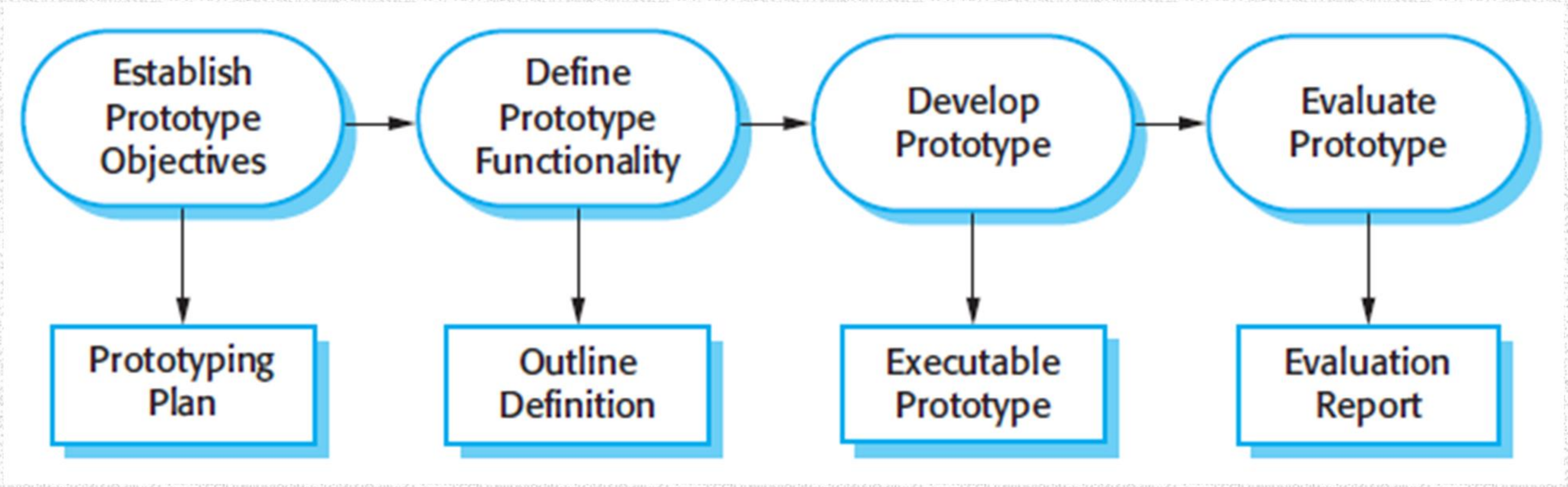
Software prototyping

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

Benefits:

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

The process of prototype development



<u>Prototype development:</u>

- May be based on rapid prototyping languages or tools
- May involve leaving out functionality

Incremental delivery

- The development and delivery is broken down into increments
  - each increment delivering part of the required functionality.
  - user requirements are prioritised and the highest priority requirements are included in early increments.
- Two approaches:
  - Incremental development: by developer
  - Incremental delivery: for end-user

Incremental delivery

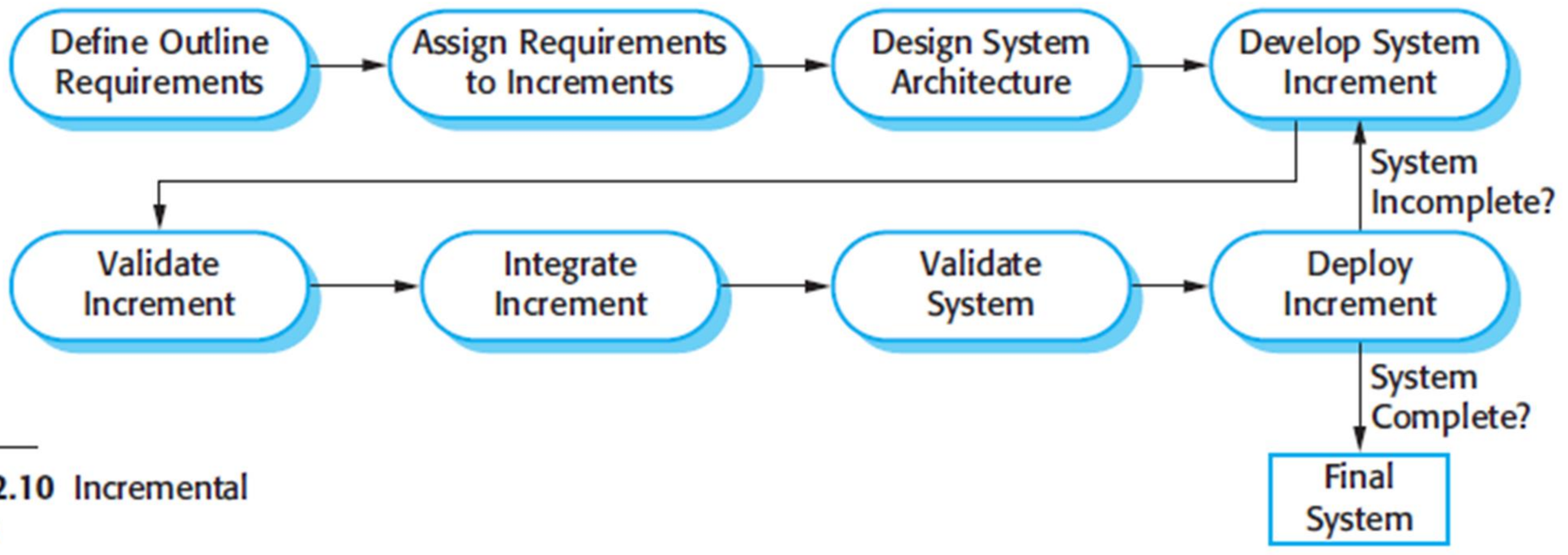


Figure 2.10 Incremental delivery

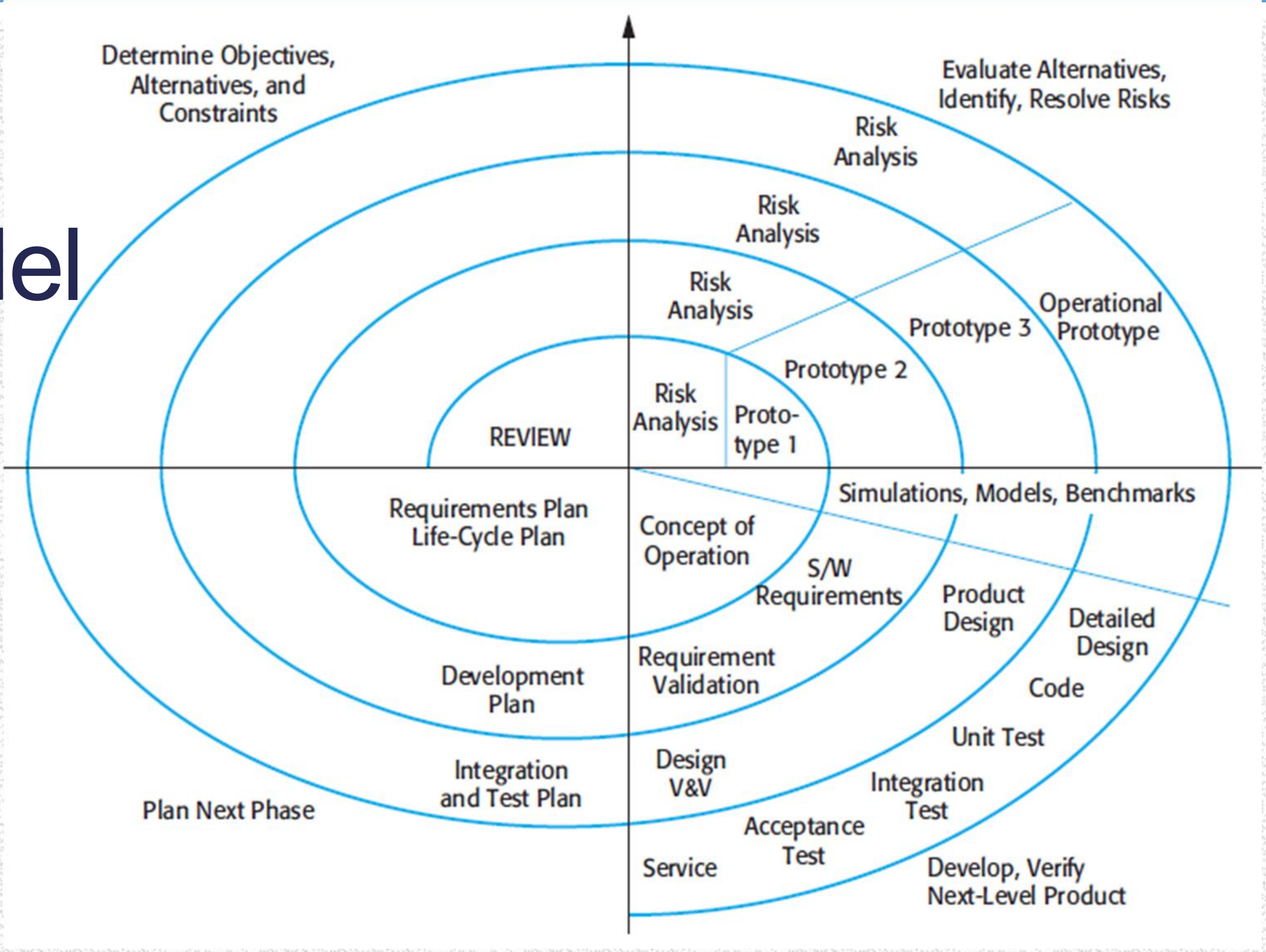
Advantages:

- system functionality is available earlier.
- early increments act as a prototype
- lower risk of overall project failure.
- highest priority system services receive most testing.

Problems:

- may require a set of basic facilities
- the specification is developed in conjunction with the software.

Boehm's spiral model



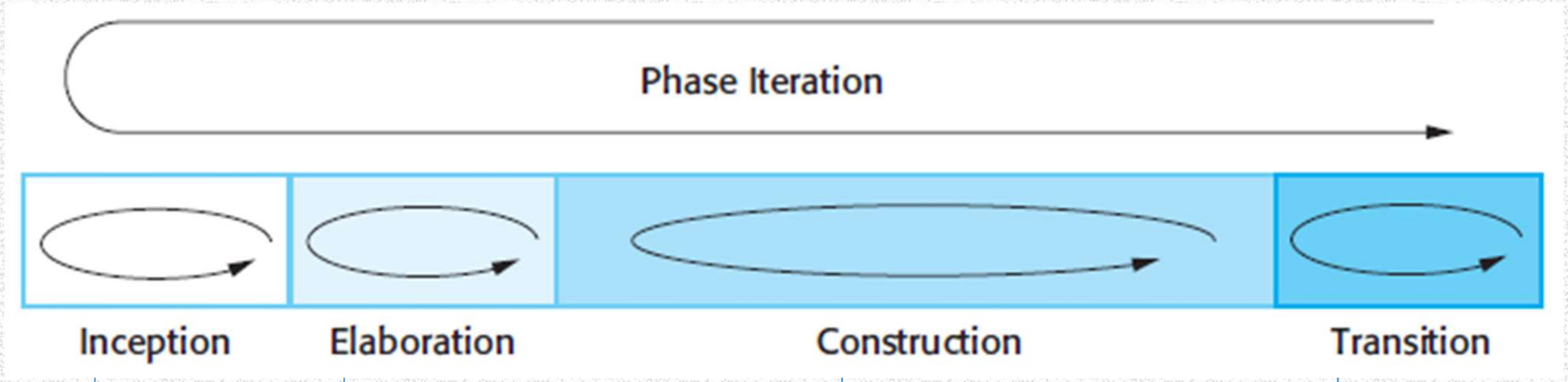
- Each loop = a phase in the process.
- No fixed phases such as specification or design
- Risks are explicitly assessed and resolved throughout the process.

RATIONAL UNIFIED PROCESS

The Rational Unified 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 proactive perspective that suggests good practice.

Phases in the Rational Unified Process



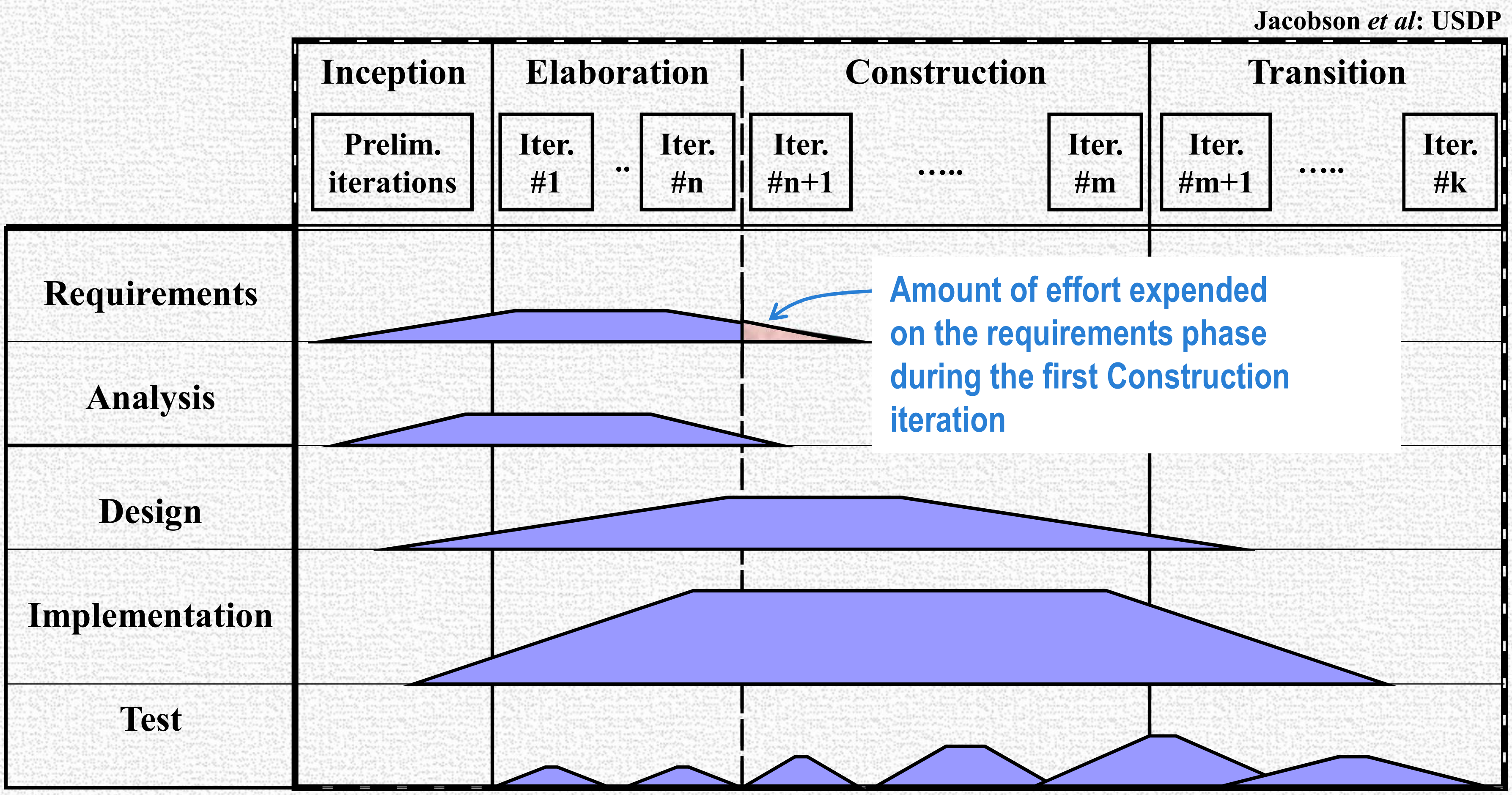
Establish the business case for the system.

Develop an understanding of the problem domain and the system architecture.

System design, programming and testing.

Deploy the system in its operating environment.

Unified Process Matrix



Static workflows in the 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.
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
Project management	This supporting workflow manages the system development.
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

RUP good practice

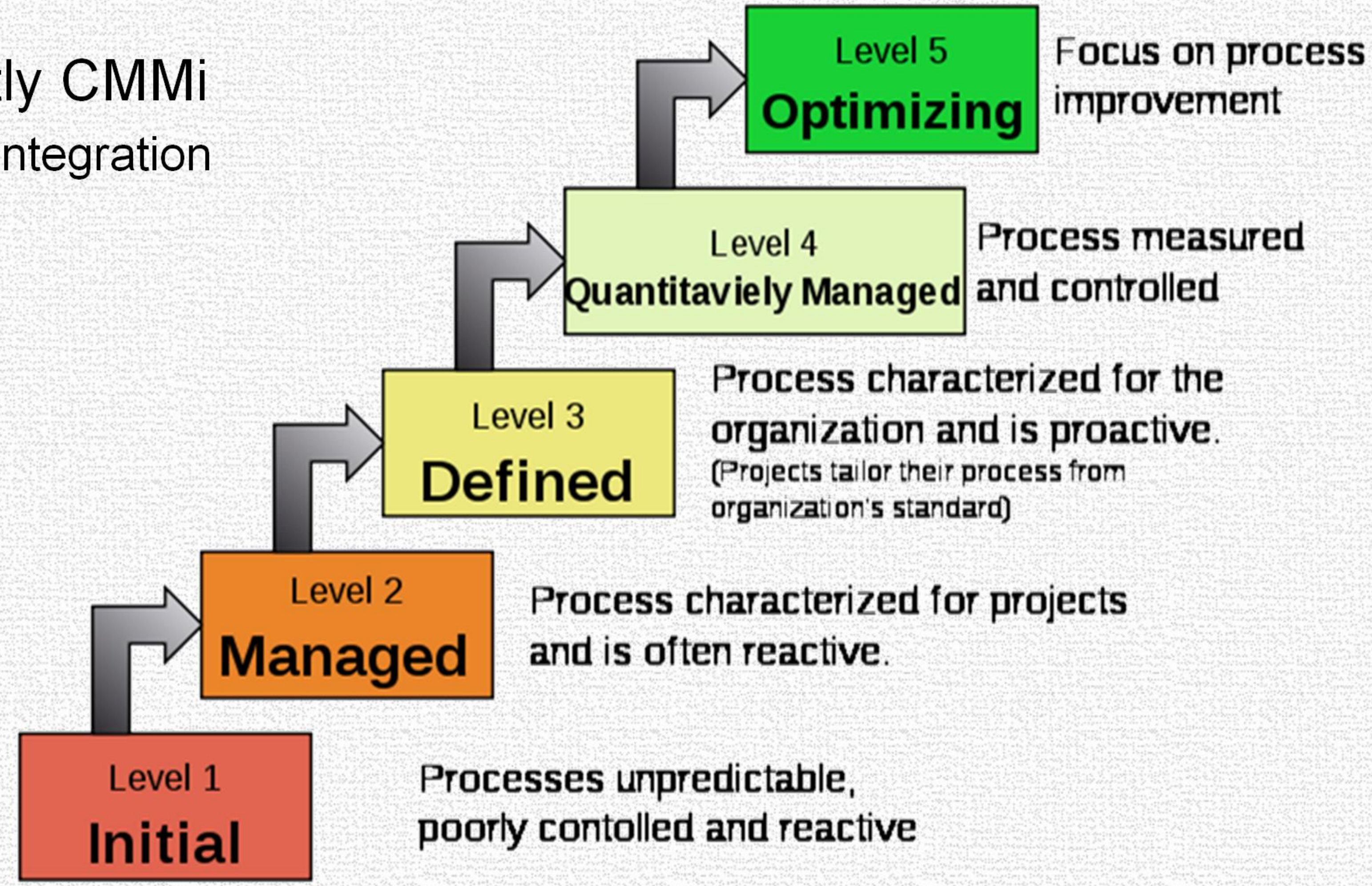
- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

Project Documentation

Verification & validation	**SVVP** software validation & verification plan
Quality assurance	**SQAP** software quality assurance plan
Configuration	**SCMP** software configuration management plan
Project status	**SPMP** software project management plan
Requirements	**SRS** software requirements specifications
Design	**SDD** software design document
Code	Source Code
Testing	**STD** software test document
Operation	User’s manual

The Capability Maturity Model (CMM)

- Currently CMMi
  - CMM integration


Level 5
Optimizing
Focus on process improvement
Level 4
Quantitaviely Managed
Process measured and controlled
Level 3
Defined
Process characterized for the organization and is proactive. (Projects tailor their process from organization's standard)
Level 2
Managed
Process characterized for projects and is often reactive.
Level 1
Initial
Processes unpredictable, poorly contolled and reactive

http://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration

Summary

- Software processes
- Software process models
  - waterfall, incremental development, reuse-oriented development.
- Fundamental activities:
  - Requirements engineering: developing specification.
  - Design and implementation: transforming a requirements specification into an executable software system
  - Software validation: checking that the system conforms to its specification.
  - Software evolution: change existing software systems to meet new requirements

Summary (cont.)

- Coping with change
  - prototyping
  - iterative development and delivery
- The Rational Unified Process – RUP
  - a modern generic process model
  - organized into phases (inception, elaboration, construction and transition) but separates activities (requirements, analysis and design, etc.) from these phases.