2. Software Processes

IT2206 - Fundamentals of Software Engineering

Level I - Semester 2

Learning Outcomes

- Describe different process models used for software development
- Identify the most appropriate software process model for a given problem
- Identify how CASE tools can be used to support software 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.

The Software Process Cont...

- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

Software Process Descriptions

When we describe and discuss processes, we usually talk about the <u>activities in these processes</u> such as <u>specifying a data model, designing a user interface, etc. and the ordering of these activities.</u>

Software Process Descriptions Cont...

Process descriptions may also include:

- **Products** - outcomes of a process activity;
- **Roles** - responsibilities of the people (Software Engineer, QA Engineer, Project Manager);
- **Pre- and post-conditions**, which are statements that are true before and after a product produced.

Plan-Driven & Agile Processes

- **Plan-driven processes** - all of the process activities are planned in advance and progress is measured against this plan.
- **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.

2.1 Software Process Models

IT2206 - Fundamentals of Software Engineering

Level I - Semester 2

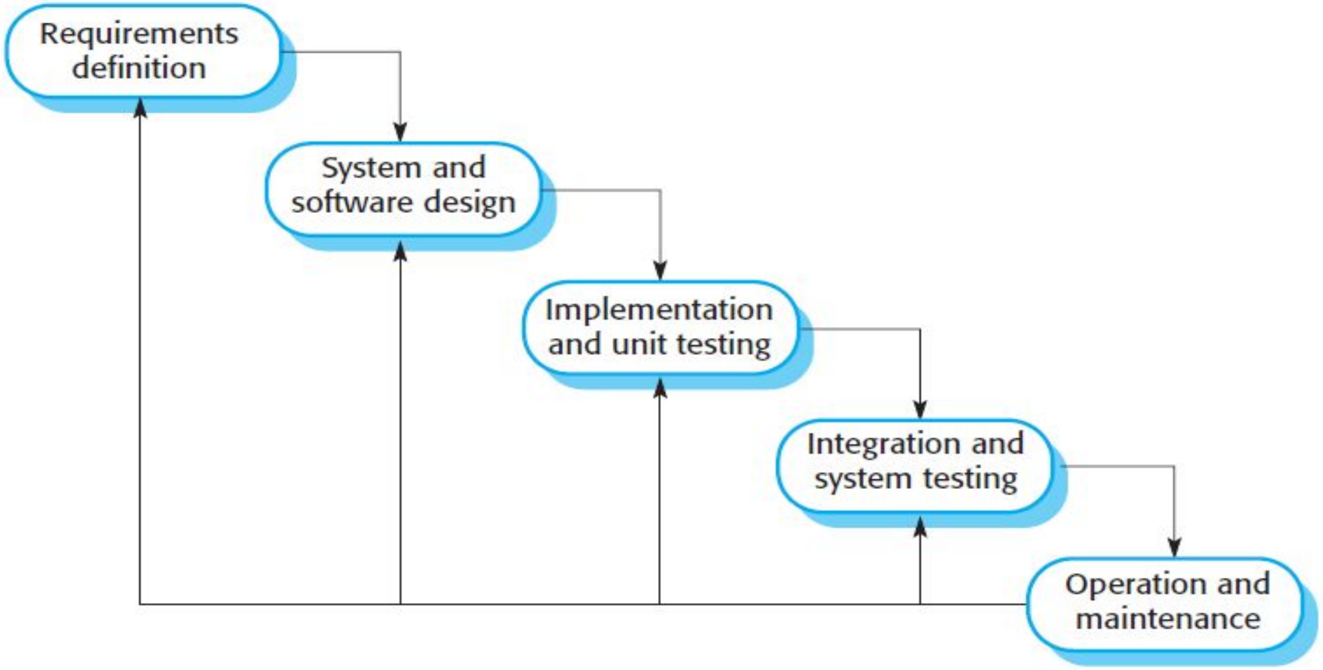
Software Process Models

- 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 or (*Integration and configuration)*
    - The system is assembled from existing components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

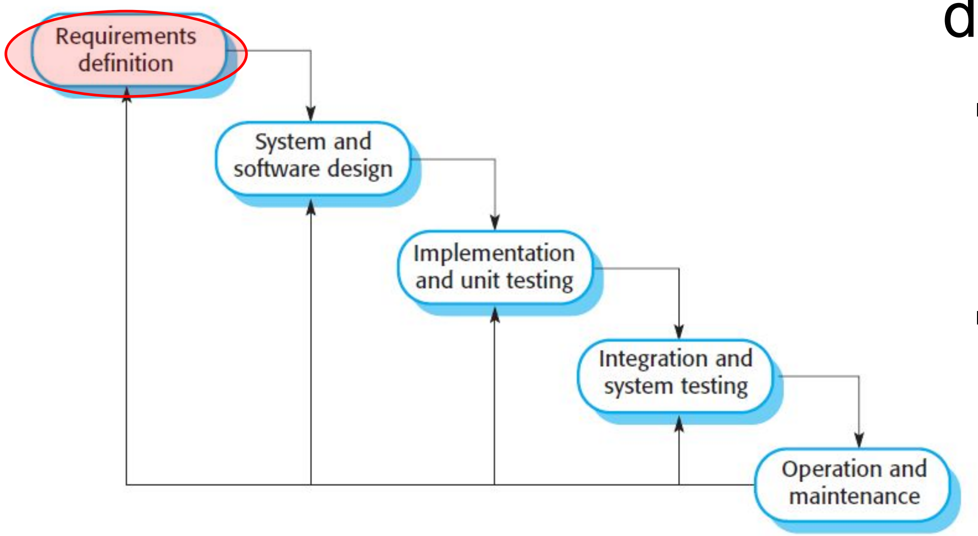
Waterfall Model

- The first published model of the software development process
- The waterfall model is an example of a plan-driven process.
- In principle at least, you plan and schedule all of the process activities before starting software development.

Waterfall Model

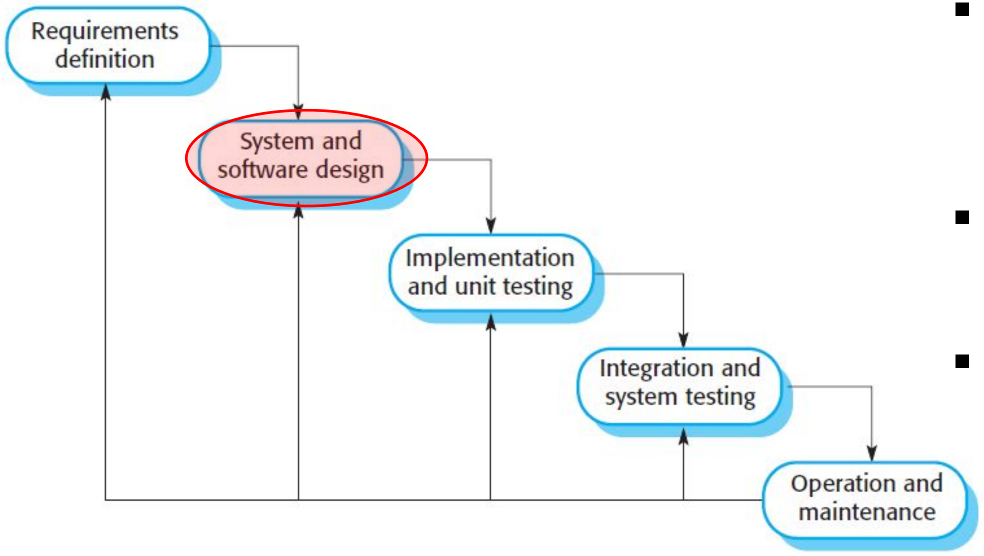

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Waterfall Model


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

- Requirements analysis and definition
  - The system's services, constraints, and goals are established by consultation with system users.
  - Need to define those in detail and serve as a system specification.

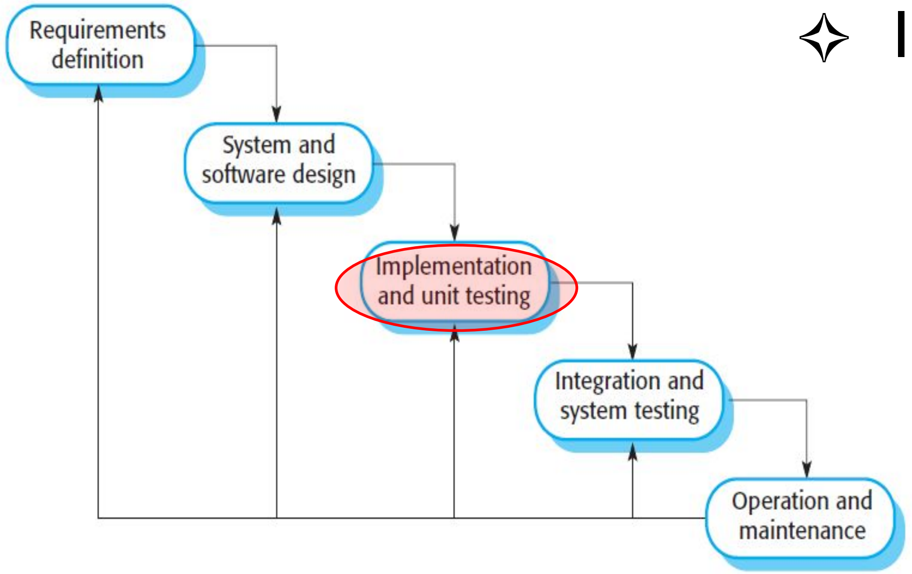
Waterfall Model



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

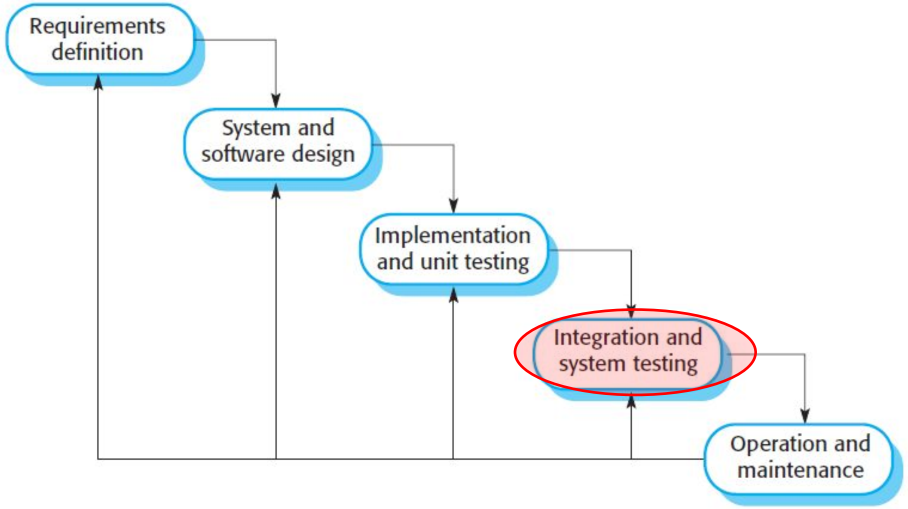
- System and software design
  - Allocates the requirements to either hardware or software systems
  - It establishes an overall system architecture.
  - Software design involves identifying and describing the fundamental software system abstractions and their relationships.

Waterfall Model


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

- Implementation and unit testing
  - During this stage, the software design is realized as a set of programs or program units.
  - Unit testing involves verifying that each unit meets its specification.

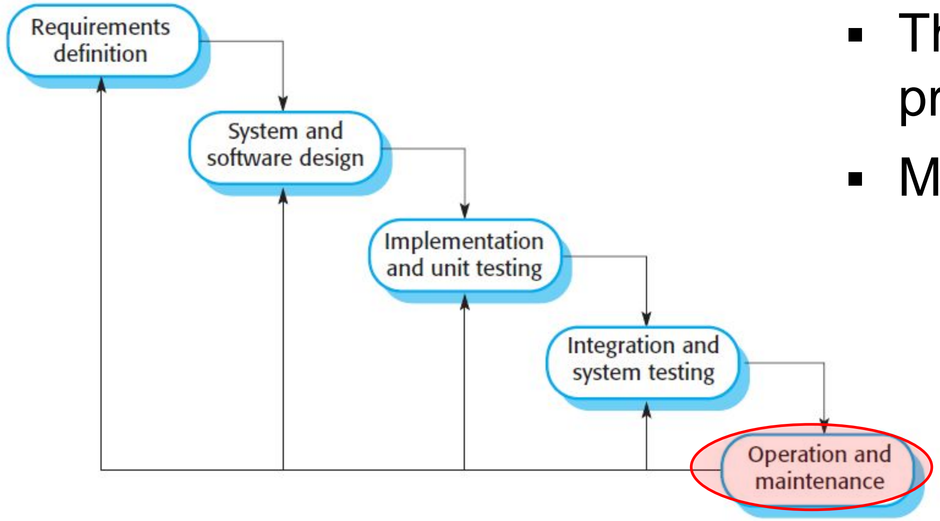
Waterfall Model



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

- ✧ Integration and system testing
  - The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met.
  - After testing, the software system is delivered to the customer.

Waterfall Model

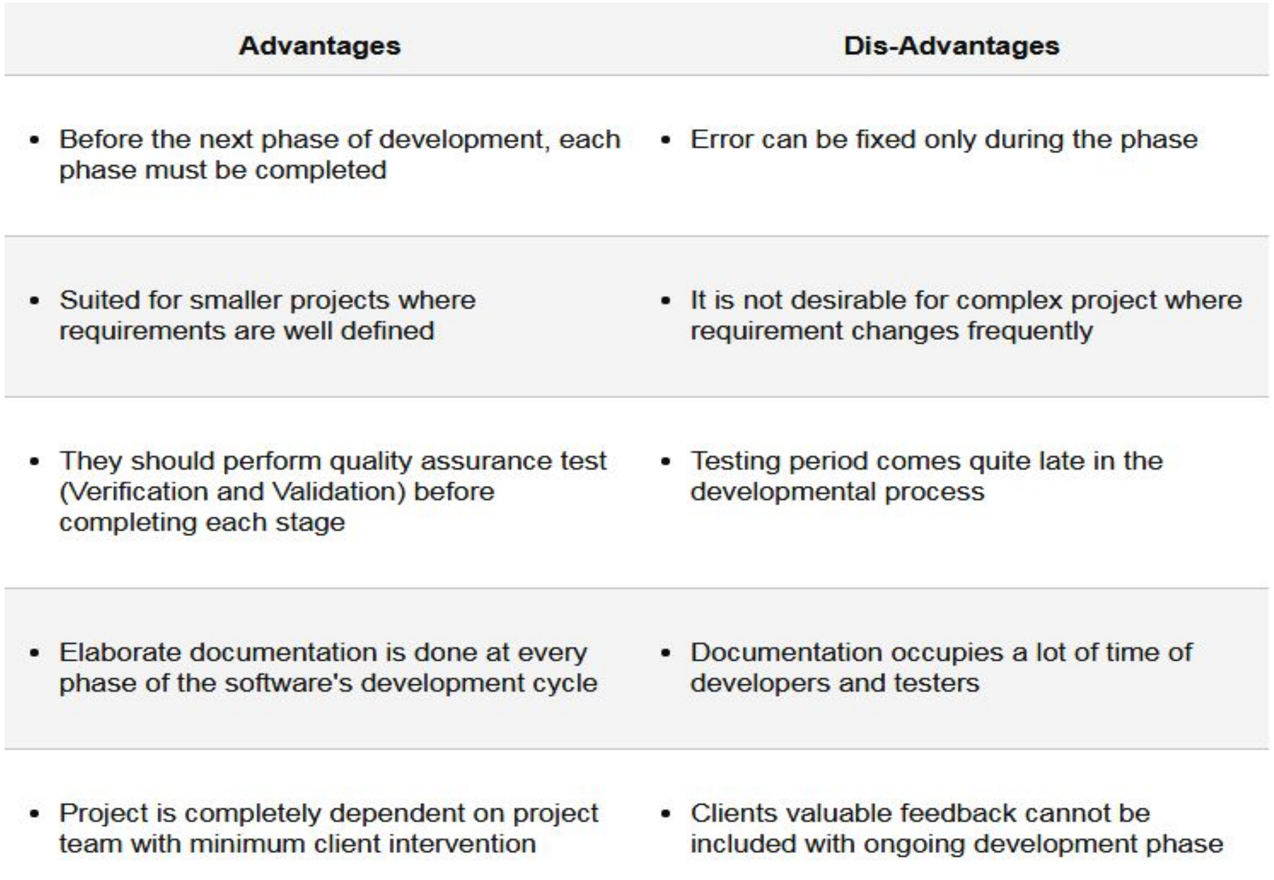

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

- Operation and maintenance
  - Normally, this is the longest life-cycle phase.
  - The system is installed and put into practical use.
  - Maintenance involves
    - correcting errors that were not discovered in earlier stages of the life cycle
    - improving the implementation of system units
    - enhancing the system's services as new requirements are discovered.

Waterfall Model

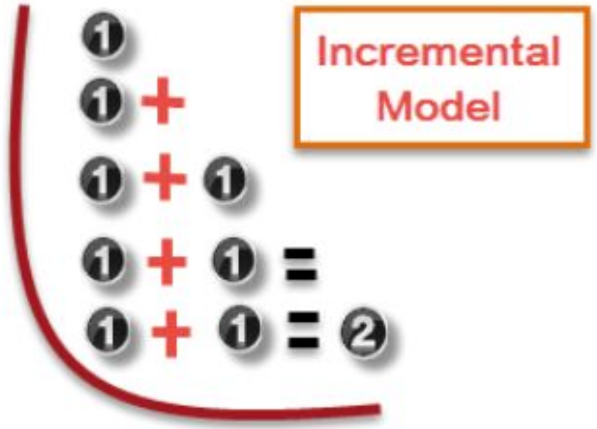
- Waterfall model can be used when
  - Requirements are not changing frequently
  - Application is not complicated and big
  - Project is short
  - Requirement is clear
  - Environment is stable
  - Technology and tools used are not dynamic and is stable
  - Resources are available and trained

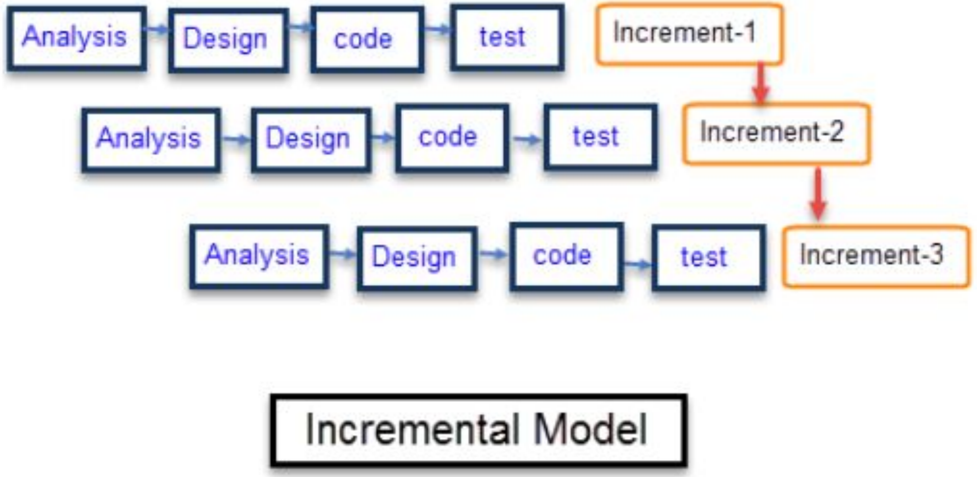
Advantages and Disadvantages of Waterfall Model



Advantages	Dis-Advantages
• Before the next phase of development, each phase must be completed	• Error can be fixed only during the phase
• Suited for smaller projects where requirements are well defined	• It is not desirable for complex project where requirement changes frequently
• They should perform quality assurance test (Verification and Validation) before completing each stage	• Testing period comes quite late in the developmental process
• Elaborate documentation is done at every phase of the software's development cycle	• Documentation occupies a lot of time of developers and testers
• Project is completely dependent on project team with minimum client intervention	• Clients valuable feedback cannot be included with ongoing development phase

Incremental Development

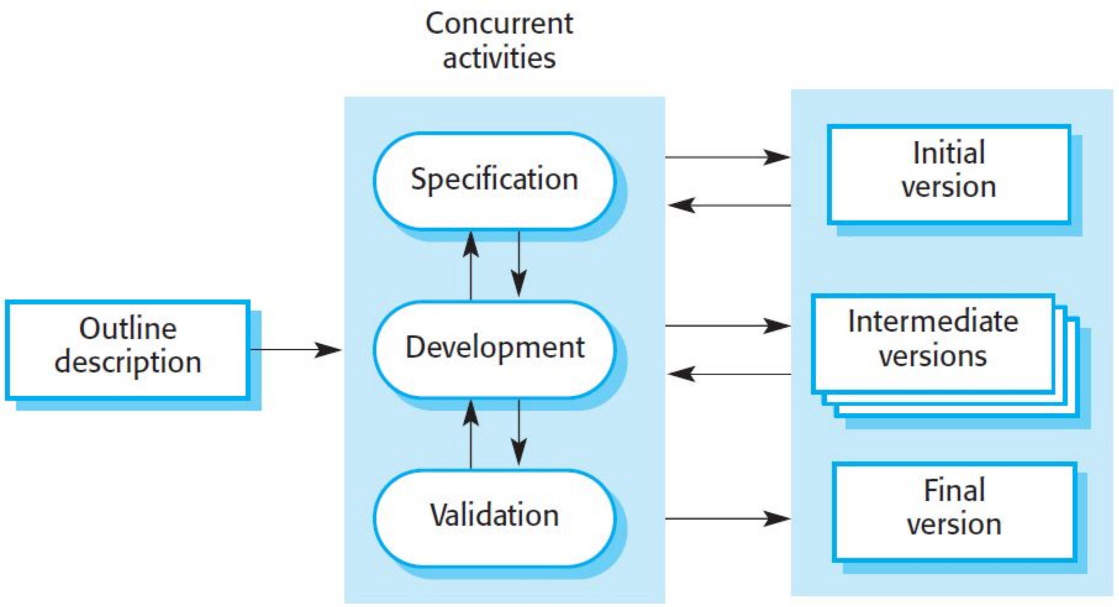




Incremental Development Cont...

- Incremental development is based on the idea of developing an initial implementation, getting feedback from users and evolving the software through several versions until the required system has been developed.
- System development is broken down into many mini development projects
- Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first

Incremental Development



Incremental Development: Benefits

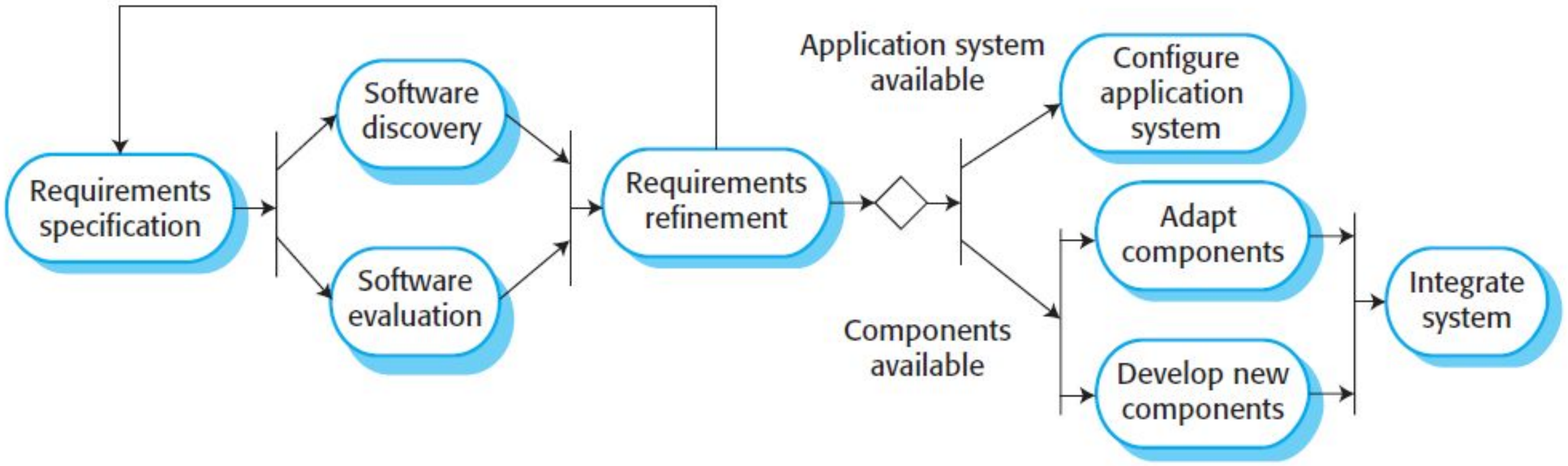
- The software will be generated quickly during the software life cycle
- It is flexible and less expensive to change requirements and scope
- Throughout the development stages, changes can be done
- This model is less costly compared to others
- A customer can respond to each building
- Errors are easy to be identified

Incremental Development: Problems

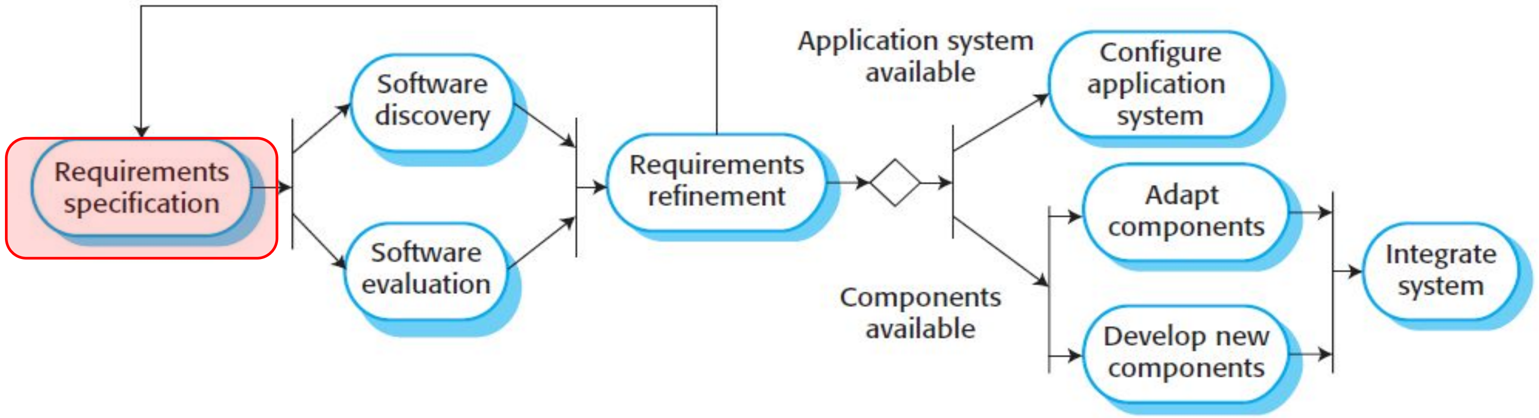
- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

Reuse-Oriented Software Engineering (*Integration and Configuration)*

Process Stages of Reuse-Based Development

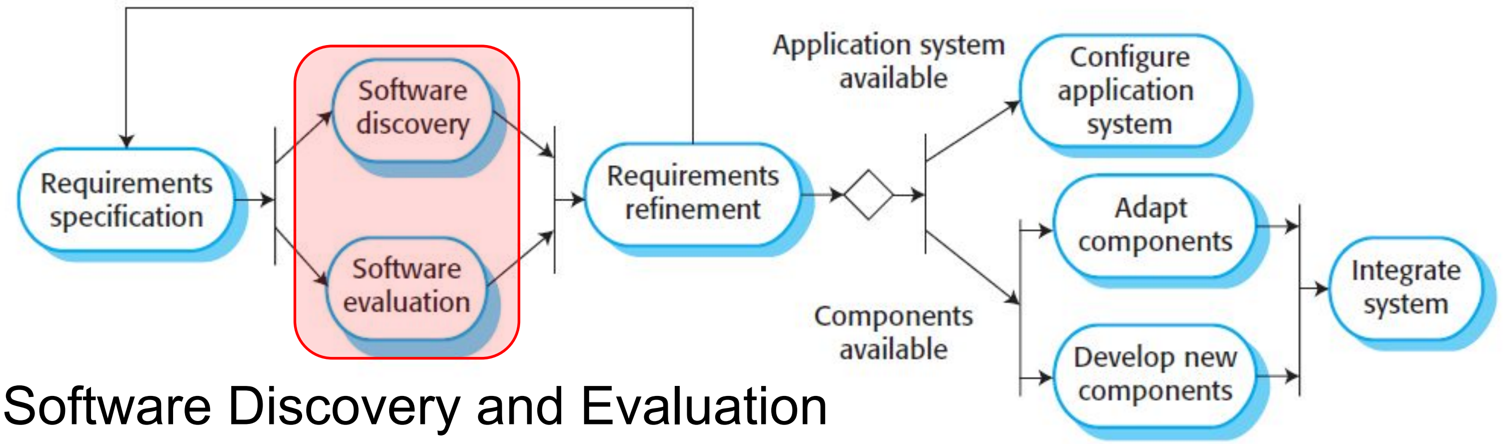


Process Stages of Reuse-Based Development Cont...



- Requirement specification;
  - initial requirements for the system are proposed.
  - These do not have to be elaborated in detail but should include brief descriptions of essential requirements and desirable system features.

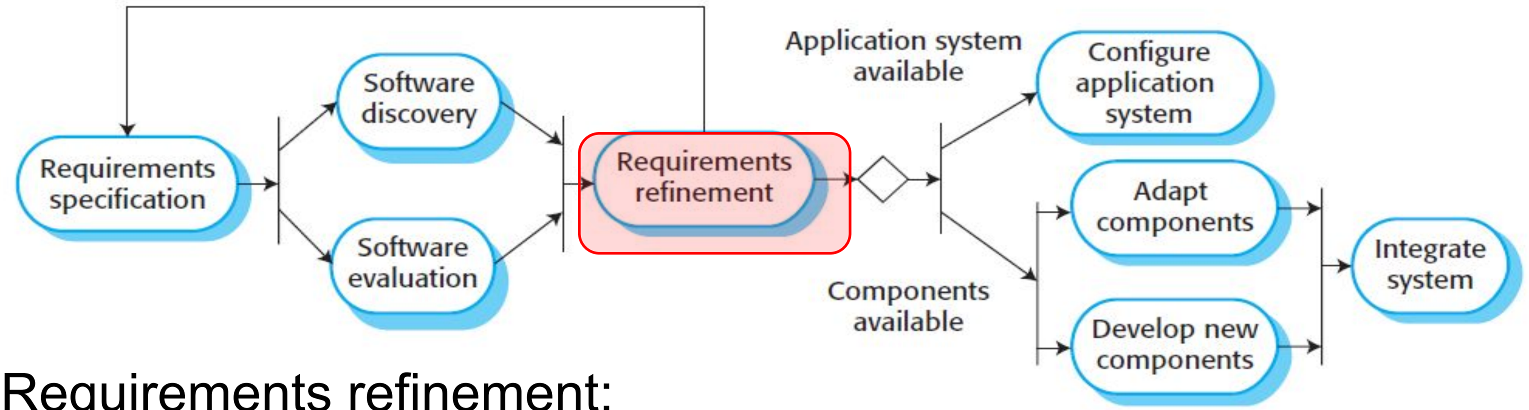
Process Stages of Reuse-Based Development Cont...



✧ Software Discovery and Evaluation

- Given an outline of the software requirements, a search is made for components and systems that provide the functionality required.
- Candidate components and systems are evaluated to see if they meet the essential requirements and if they are generally suitable for use in the system

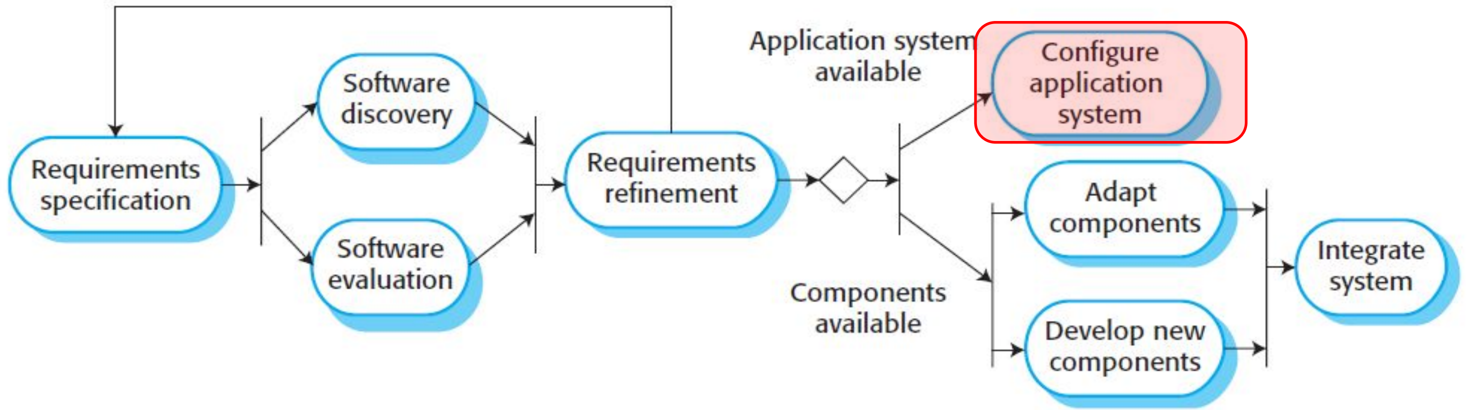
Process Stages of Reuse-Based Development Cont...



✧ Requirements refinement:

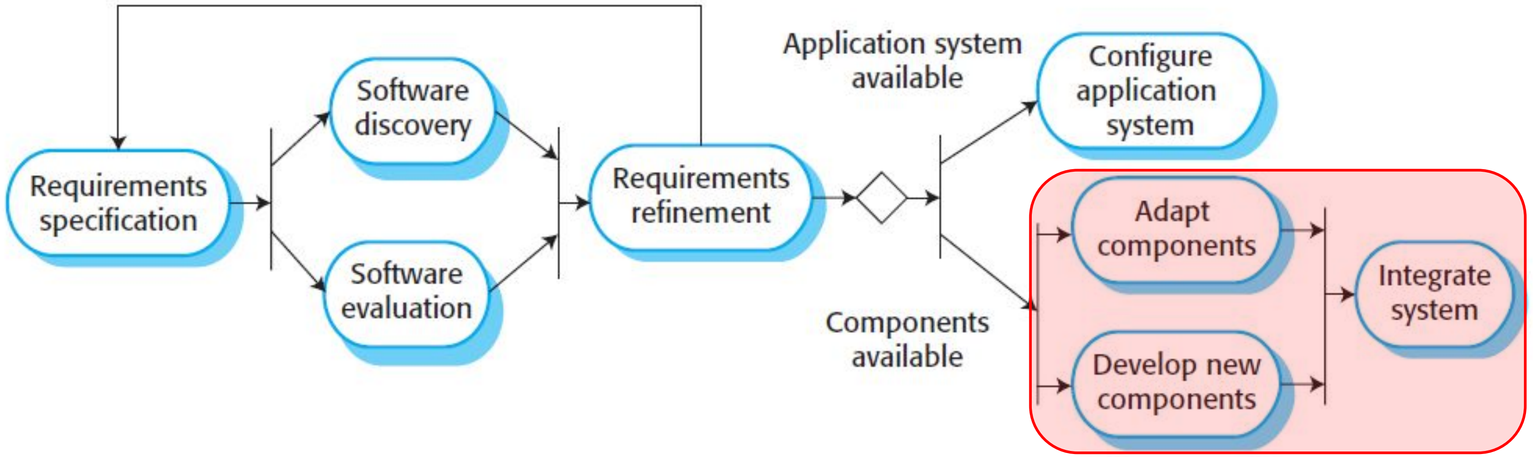
- During this stage, the requirements are refined using information about the reusable components and applications that have been discovered.
- The requirements are modified to reflect the available components, and the system specification is re-defined. Where modifications are impossible, the component analysis activity may be reentered to search for alternative solutions.

Process Stages of Reuse-Based Development



- ✧ Application system configuration:
  - If an off-the-shelf application system that meets the requirements is available, it may then be configured for use to create the new system.

Process Stages of Reuse-Based Development



- Component adaptation and integration:
  - If there is no off-the-shelf system, individual reusable components may be modified and new components developed.
  - These are then integrated to create the system.

Activity

- Identify the advantages of using reuse-oriented software engineering approach for software development.

2.2 Process Activities

IT2206 - Fundamentals of Software Engineering

Level I - Semester 2

Process Activities

- The four basic process activities:
  - 2.2.1. Software Specification
  - 2.2.2. Software Development
  - 2.2.3. Software Validation
  - 2.2.3. Software Evolution

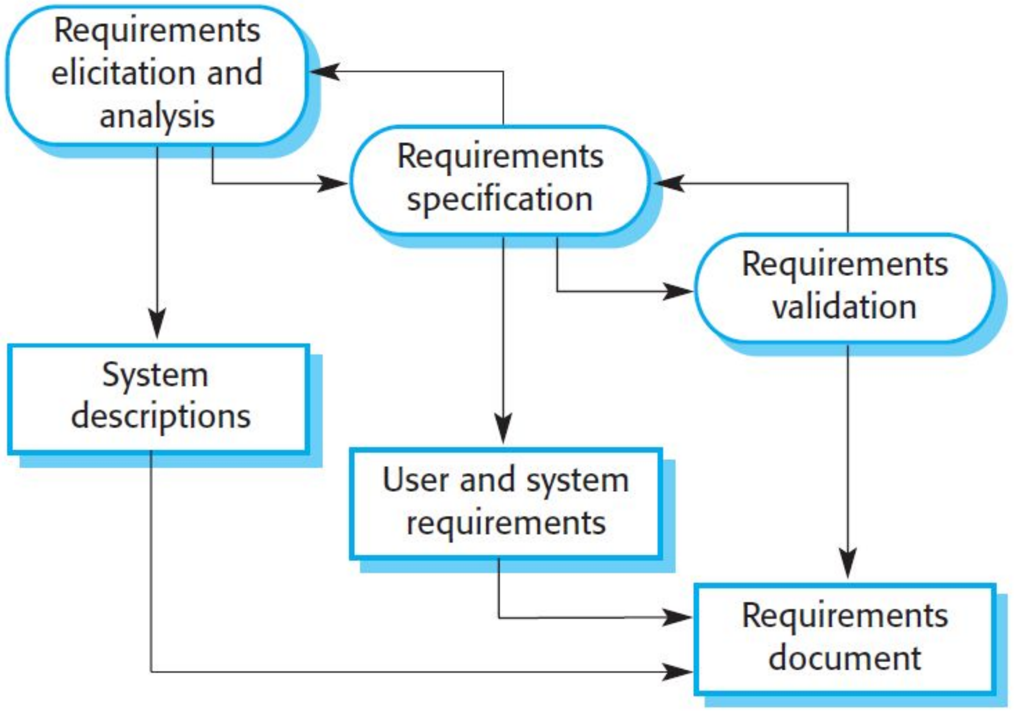
2.2.1. Software Specification

- Also known as Requirements Engineering
- The process of establishing what services are required and the constraints on the system's operation and development.
- Before the requirements engineering process starts, a company may carry out a feasibility or marketing study to assess whether or not there is a need or a market for the software and whether or not it is technically and financially realistic to develop the software required.

2.2.1. Software Specification Cont...

- Requirements engineering process
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
  - Requirements specification
    - Translate the information gathered during requirements analysis into a document that defines a set of requirements.
    - User requirements and System requirements
  - Requirements validation
    - Checking the validity of the requirements

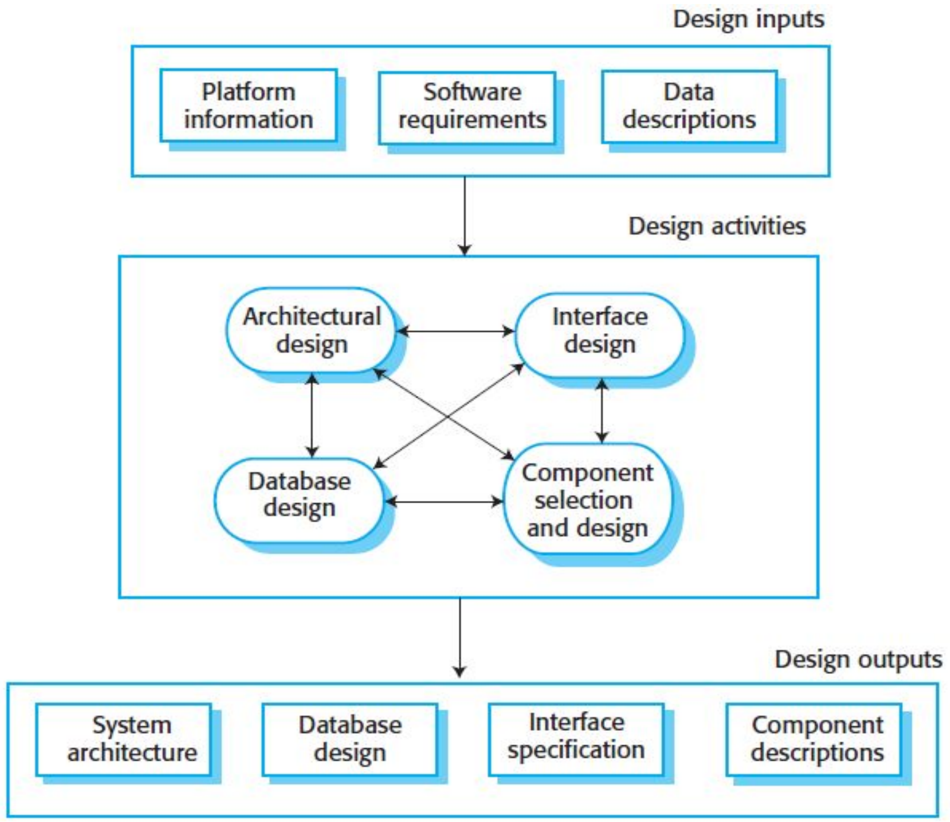
2.2.2. Software Development
The Requirements Engineering (Software Specification) Process


Requirements elicitation and analysis
Requirements specification
Requirements validation
System descriptions
User and system requirements
Requirements document

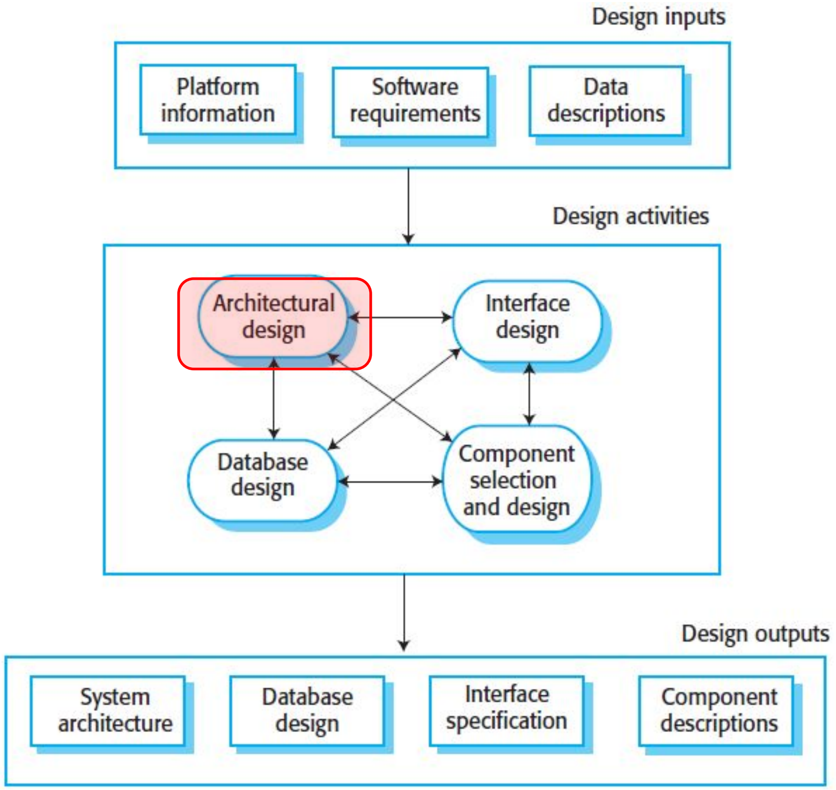
Software Design and Implementation

- 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.

A General Model of the Design Process

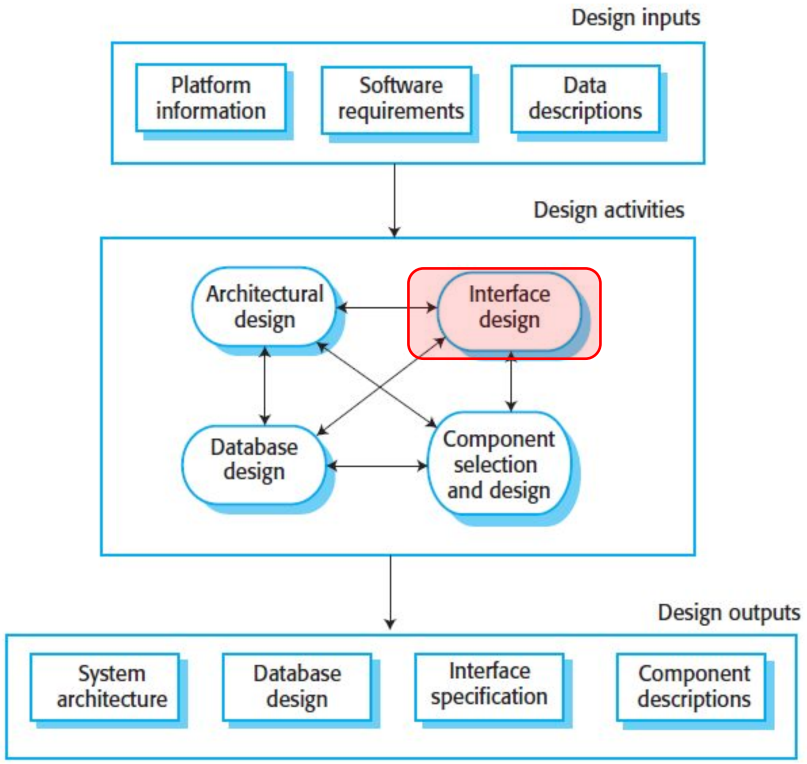


A General Model of the Design Process



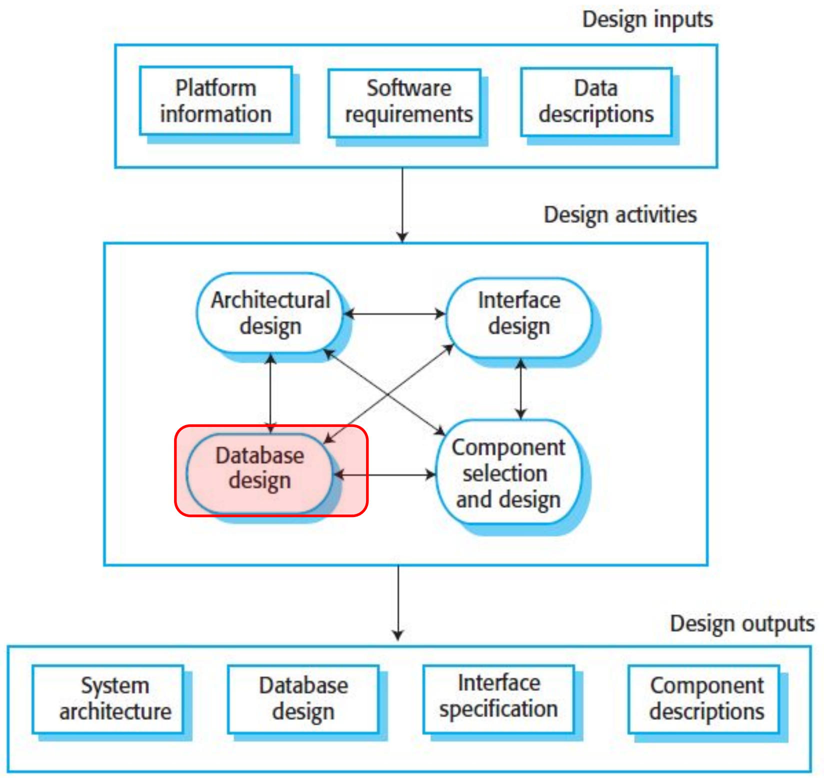
1. *Architectural design:*
   - where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.

A General Model of the Design Process



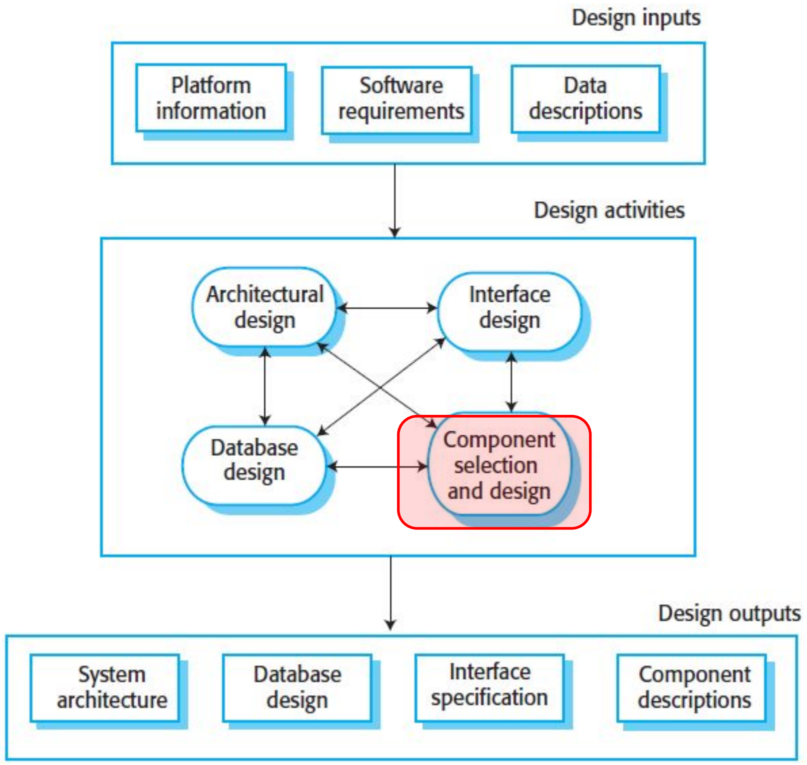
2. *Interface design:*
   - where you define the interfaces between system components.

A General Model of the Design Process



3. *Database design:*
   - where you design the system data structures and how these are to be represented in a database
   - work here depends on whether an existing database is to be reused or a new database is to be created

A General Model of the Design Process



4. Component selection and design:

- where you search for reusable components and, if no suitable components are available, design new software components

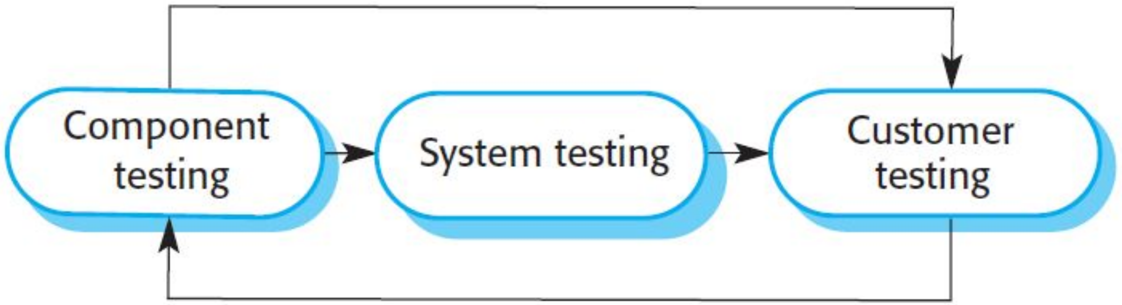
2.2.3. Software Validation

- 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 is the most commonly used V & V activity.

Verification & Validation

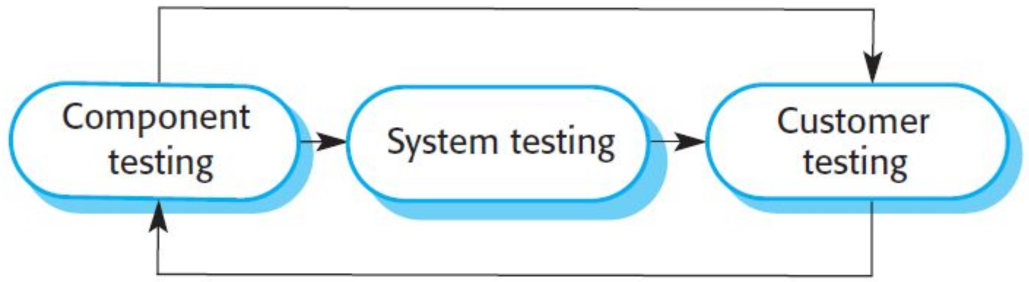
- **Verification** is the process of checking that the software meets the specification
  - Walkthrough
  - Inspection
  - Review
- **Validation** is the process of checking whether the specification captures the customer's needs.
  - Testing

Stages of Testing



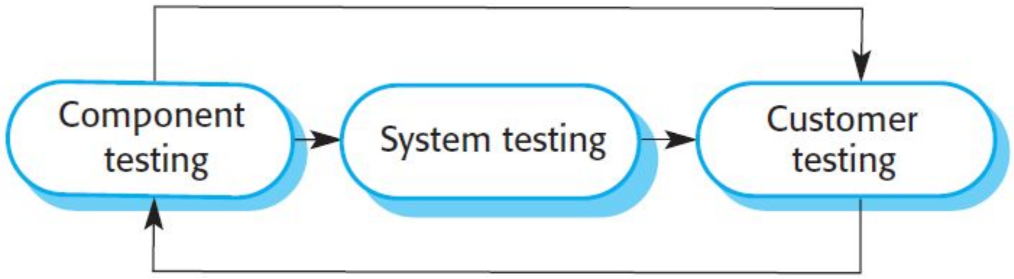
- Development or component testing
  - Individual components are tested independently;
  - Components may be functions or objects of these entities.

Stages of Testing



- System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.

Stages of Testing



- Acceptance testing
  - Testing with customer data to check that the system meets the customer’s needs.

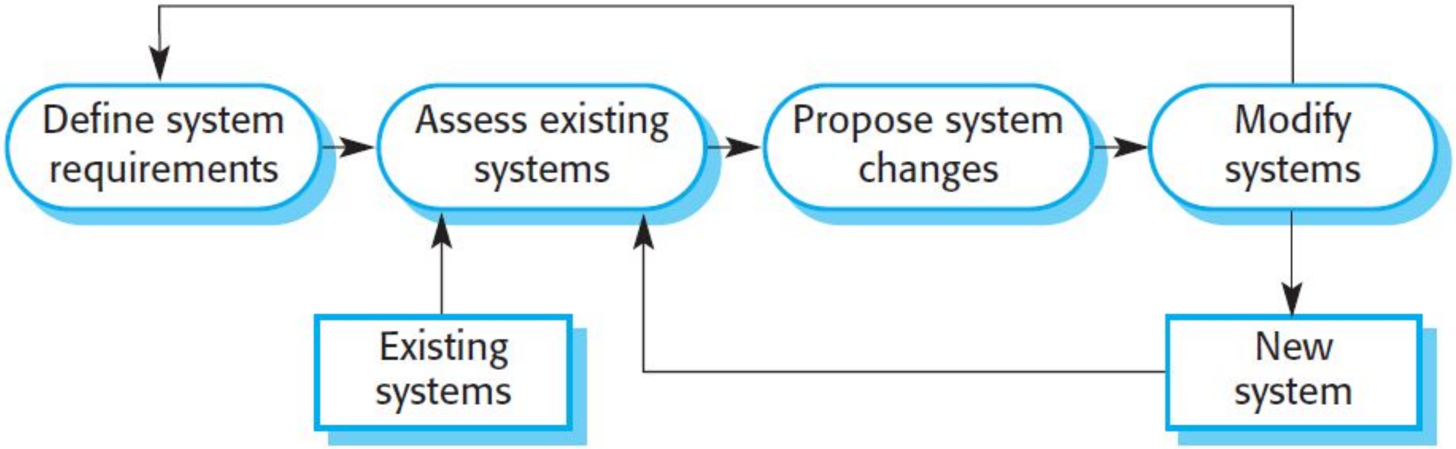
Testing Phases in a plan-driven Software Process

- When a plan-driven software process is used (e.g., for critical systems development), testing is driven by a set of test plans.
- An independent team of testers works from these test plans, which have been developed from the system specification and design.

2.2.4. Software Evolution (Maintenance)

- The flexibility of software is one of the main reasons why more and more software is being incorporated into large, complex systems.
- Once a decision has been made to manufacture hardware, it is very expensive to make changes to the hardware design.
- However, changes can be made to software at any time during or after the system development.
- Even extensive changes are still much cheaper than corresponding changes to system hardware.

System Evolution



2.3 Coping with Change

IT2206 - Fundamentals of Software Engineering

Level I - Semester 2

Coping with Change

- Change is unavoidable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- Change leads to rework so the costs of change include both rework (e.g. re-analyzing requirements) as well as the costs of implementing new functionality

Approaches to reduce the costs of rework

- **Change anticipation**, where the software process includes activities that can predict possible changes before significant rework is required.

  Example : a prototype system may be developed to show some key features of the system to customers.

- **Change tolerance**, where the process and software are designed so that changes can be easily made to the system.
  - This normally involves some form of incremental development. Proposed changes may be implemented in increments that have not yet been developed. If this is impossible, then only a single increment (a small part of the system) may have be altered to incorporate the change.

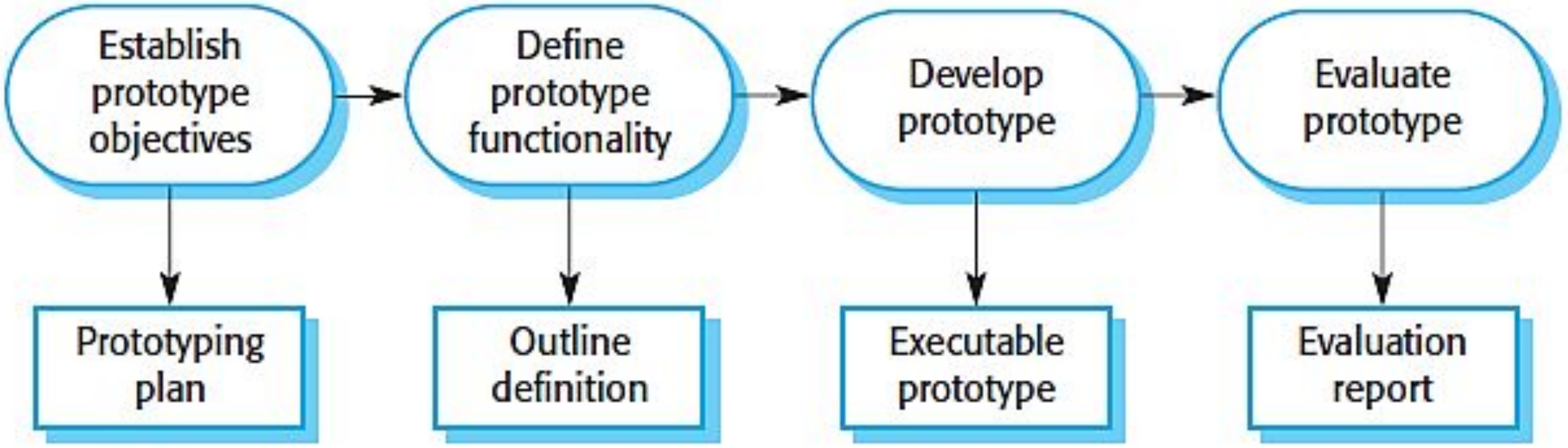
Software Prototyping

- 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;
  - System design processes to explore software solutions and develop a UI design;

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



Incremental Delivery

- Incremental delivery is an approach to software development where some of the developed increments are delivered to the customer and deployed for use in their working environment.
- In this process, customers define which of the services are most important and which are least important to them.
- Once the system increments have been identified, the requirements for the services to be delivered in the first increment are defined in detail and that increment is developed.

Incremental Delivery Cont...

- During development, further requirements analysis for later increments can take place, but requirements changes for the current increment are not accepted.
- Once an increment is completed and delivered, it is installed in the customer’s normal working environment.
- They can experiment with the system, and this helps them clarify their requirements for later system increments.
- As new increments are completed, they are integrated with existing increments so that system functionality improves with each delivered increment.

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

- Iterative delivery is problematic when the new system is intended to replace an existing system.
  - Users need all of the functionality of the old system and are usually unwilling to experiment with an incomplete new system.
  - It is often impractical to use the old and the new systems alongside each other as they are likely to have different databases and user interfaces.

- 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.

2.4 Process Improvement

IT2206 - Fundamentals of Software Engineering

Level I - Semester 2

Process Improvement

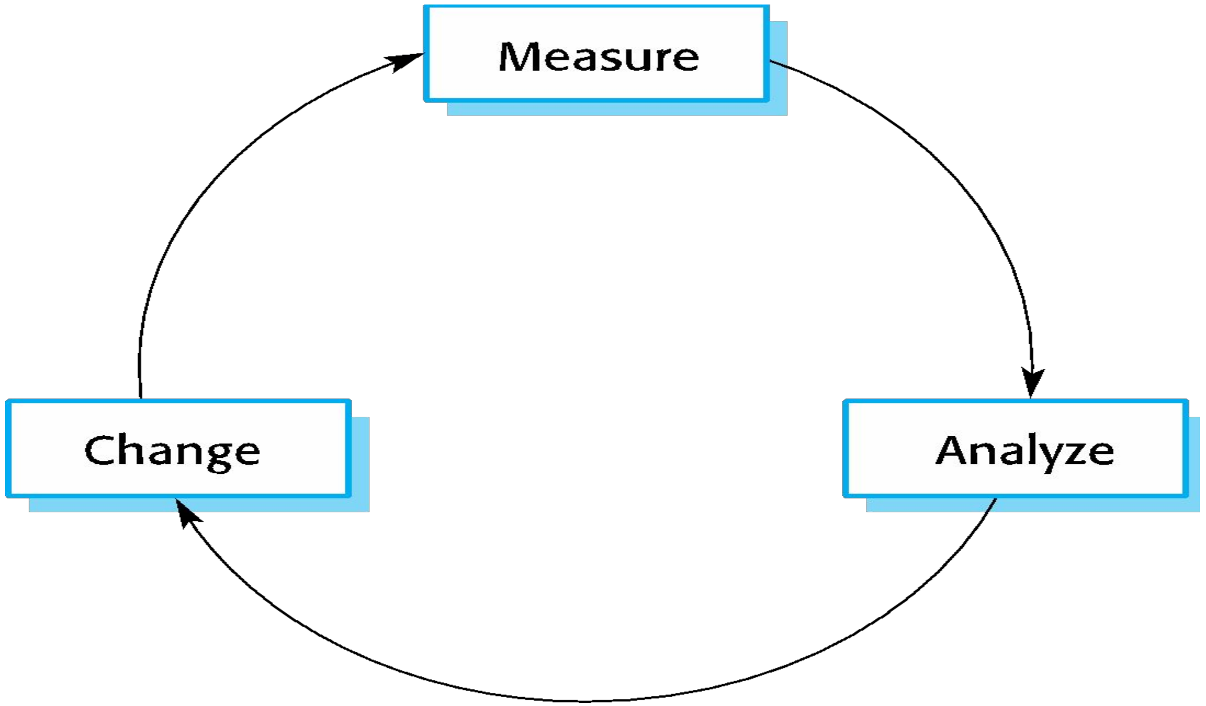
- Many software companies have turned to software process improvement as a way of enhancing the quality of their software, reducing costs or accelerating their development processes.
- Process improvement means understanding existing processes and changing these processes to increase product quality and/or reduce costs and development time.

Process Improvement Approaches

1. The process maturity approach, which focuses on improving process and project management and introducing good software engineering practice.
   - The level of process maturity reflects the extent to which good technical and management practice has been adopted in organizational software development processes.

2. The agile approach, which focuses on iterative development and the reduction of overheads in the software process.
   - The primary characteristics of agile methods are rapid delivery of functionality and responsiveness to changing customer requirements.

Process Improvement Cycle



Process Improvement Activities

- **Process measurement**
  - You measure one or more attributes of the software process or product. These measurements forms a baseline that helps you decide if process improvements have been effective.

- **Process analysis**
  - The current process is assessed, and process weaknesses and bottlenecks are identified. Process models (sometimes called process maps) that describe the process may be developed.

- **Process change**
  - Process changes are proposed to address some of the identified process weaknesses. These are introduced and the cycle resumes to collect data about the effectiveness of the changes.

Process Measurement

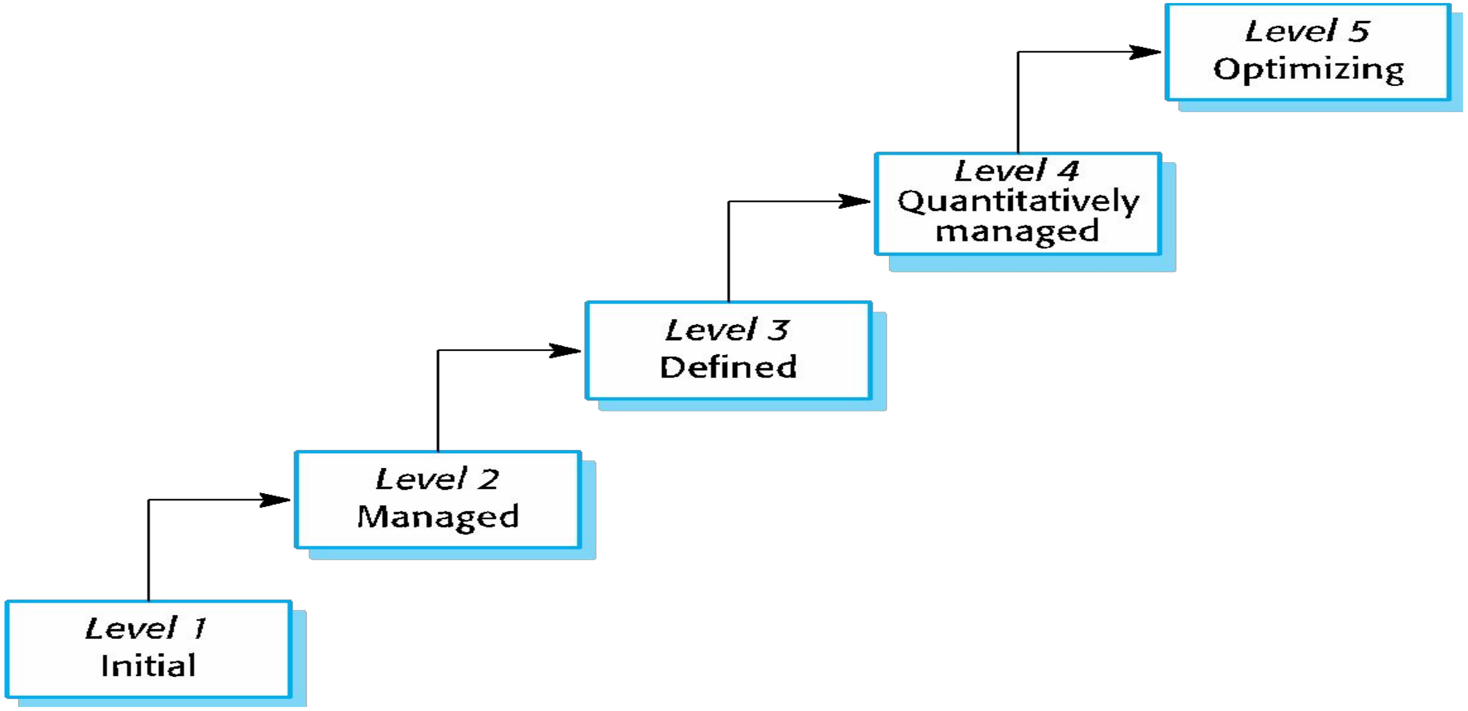
- Wherever possible, quantitative process data should be collected
  - However, where organisations do not have clearly defined process standards this is very difficult as you don’t know what to measure. A process may have to be defined before any measurement is possible.

- Process measurements should be used to assess process improvements
  - But this does not mean that measurements should drive the improvements. The improvement driver should be the organizational objectives.

Process Metrics

- Time taken for process activities to be completed
  - E.g. Calendar time or effort to complete an activity or process.
- Resources required for processes or activities
  - E.g. Total effort in person-days.
- Number of occurrences of a particular event
  - E.g. Number of defects discovered.

Capability Maturity Levels



The SEI Capability Maturity Model

- **Initial**
  - Essentially uncontrolled
- **Repeatable**
  - Product management procedures defined and used
- **Defined**
  - Process management procedures and strategies defined and used
- **Managed**
  - Quality management strategies defined and used
- **Optimising**
  - Process improvement strategies defined and used

Activity

- Suggest the most appropriate generic software process model that might be used as a basis for managing the development of the following systems. Explain your answer according to the type of system being developed:
  - A system to control anti lock braking in a car
  - A virtual reality system to support software maintenance
  - A university accounting system that replaces an existing system
  - An interactive travel planning system that helps users plan journeys with the lowest environmental impact
- Discuss why and how the incremental software development could be very effectively used for customers who do not have a clear idea about the systems needed for their operations.

Summary

- Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.
- General process models describe the organization of software processes.
  - Examples of these general models include the 'waterfall' model, incremental development, and reuse-oriented development.
- Requirements engineering is the process of developing a software specification.

Summary

- Design and implementation processes are concerned with transforming a requirements specification into an executable software system.
- Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.
- Software evolution takes place when you change existing software systems to meet new requirements. The software must evolve to remain useful.
- Processes should include activities such as prototyping and incremental delivery to cope with change.

Summary

- Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole.
- The principal approaches to process improvement are agile approaches, geared to reducing process overheads, and maturity-based approaches based on better process management and the use of good software engineering practice.
- The SEI process maturity framework identifies maturity levels that essentially correspond to the use of good software engineering practice.