Module 1

Introduction

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable, and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation, and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software, and process engineering. Software engineering is part of this more general process.
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times, and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs; 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.
What differences has the Web made to software engineering?	The Web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

There are two kinds of software products:

1. Generic products These are stand-alone systems that are produced by a development organization and sold on the open market to any customer who is able to buy them. Examples of this type of product include software for PCs such as databases, word processors, drawing packages, and project-management tools.

2. Customized (or bespoke) products These are systems that are commissioned by a particular customer. A software contractor develops the software especially for that customer. Examples of this type of software include control systems for electronic devices, systems written to support a particular business process, and air traffic control systems.

Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

Product characteristics	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable, and compatible with other systems that they use.

Fig: Essential attributes of good software

There are four fundamental activities that are common

to all software processes. These activities are:

1. **Software specification,** where customers and engineers define the software that is to be produced and the constraints on its operation.
2. **Software development,** where the software is designed and programmed.
3. **Software validation,** where the software is checked to ensure that it is what the customer requires.
4. **Software evolution**, where the software is modified to reflect changing customer and market requirements.

Software engineering ethics

1. **Confidentiality** You should normally respect the confidentiality of your employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
2. **Competence** You should not misrepresent your level of competence. You should not knowingly accept work that is outside your competence.
3. **Intellectual property rights** You should be aware of local laws governing the use of intellectual property such as patents and copyright. You should be careful to ensure that the intellectual property of employers and clients is protected.
4. **Computer misuse** You should not use your technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses or other malware).

Case studies

The three types of systems that are uses as case studies are:

1. An embedded system This is a system where the software controls a hardware device and is embedded In that device. Issues in embedded systems typically include physical size, responsiveness, power management, etc. The example of an embedded system that I use is a software system to control a medical device.

2. An information system This is a system whose primary purpose is to manage and provide access to a database of information. Issues in information systems include security, usability, privacy, and maintaining data integrity. The example of an information system that I use is a medical records system.

3. A sensor-based data collection system This is a system whose primary purpose is to collect data from a set of sensors and process that data in some way. The key requirements of such systems are reliability, even in hostile environmental conditions, and maintainability

An insulin pump control system

An insulin pump is a medical system that simulates the operation of the pancreas (an internal organ). The software controlling this system is an embedded system, which collects information from a sensor and controls a pump that delivers a controlled dose of insulin to a user

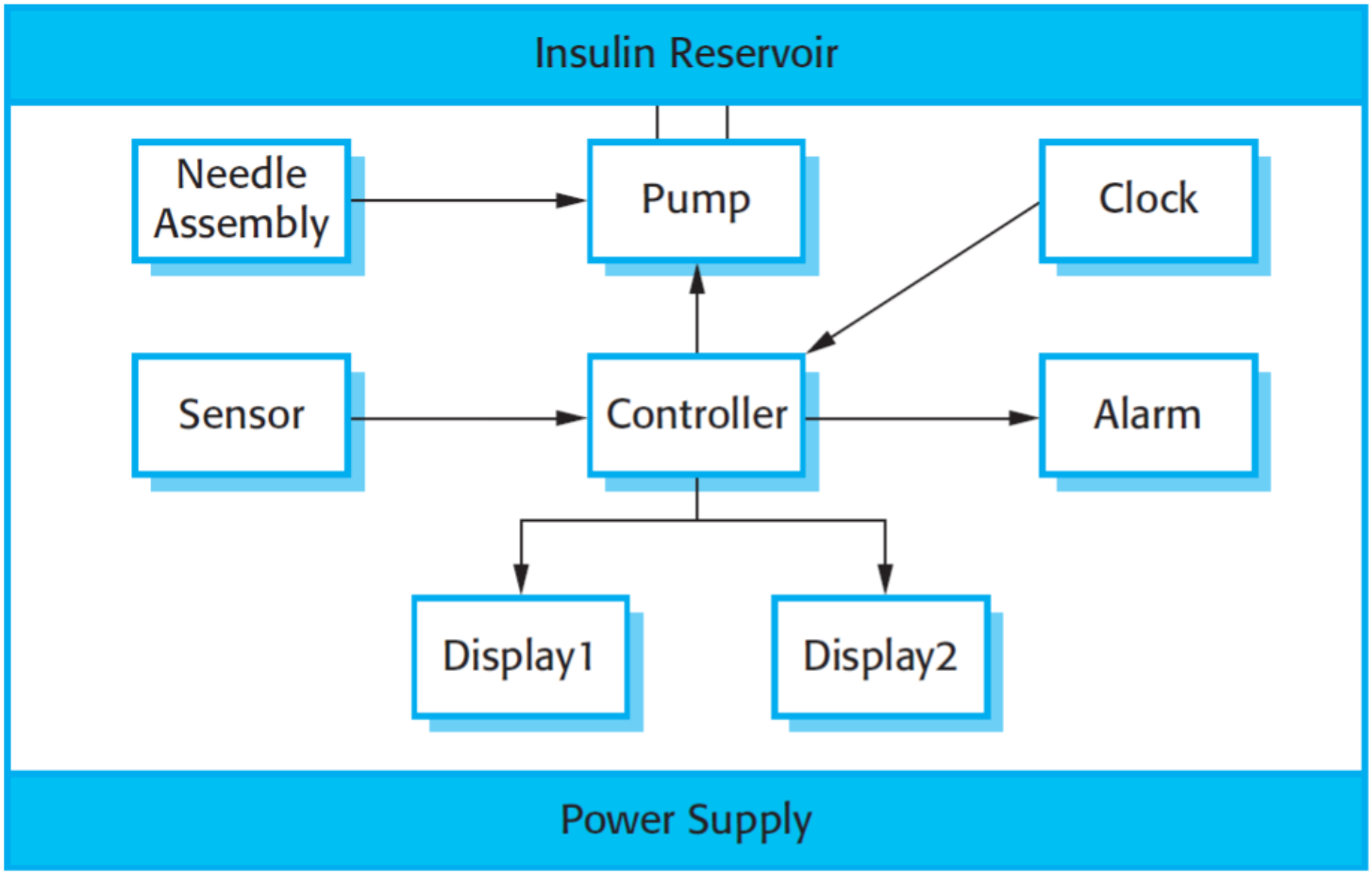

Fig: Insulin pump hardware

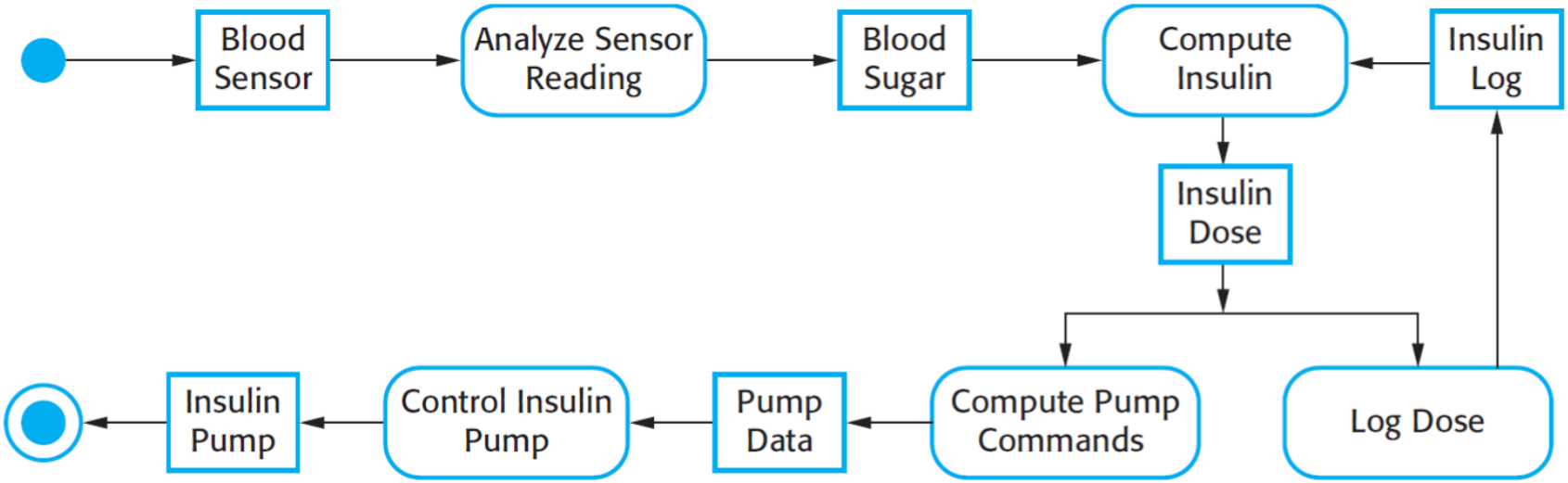

Fig: Activity model of the insulin pump

Two essential high-level requirements that this system must meet:

1. The system shall be available to deliver insulin when required.

2. The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.

The system must therefore be designed and implemented to ensure that the system always meets these requirements

A patient information system for mental health care

- A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received.
- Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems
- The MHC-PMS (Mental Health Care-Patient Management System) is an information system that is intended for use in clinics. It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity

The MHC-PMS has two overall goals:
1. To generate management information that allows health service managers to assess performance against local and government targets.
2. To provide medical staff with timely information to support the treatment of patients.

The key features of the system are:

1. **Individual care management** Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors who have not previously met a patient can quickly learn about the key problems and treatments that have been prescribed.

2. **Patient monitoring** The system regularly monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected. Therefore, if a patient has not seen a doctor for some time, a warning may be issued. One of the most important elements of the monitoring system is to keep track of patients who have been sectioned and to ensure that the legally required checks are carried out at the right time.

3. **Administrative reporting** The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

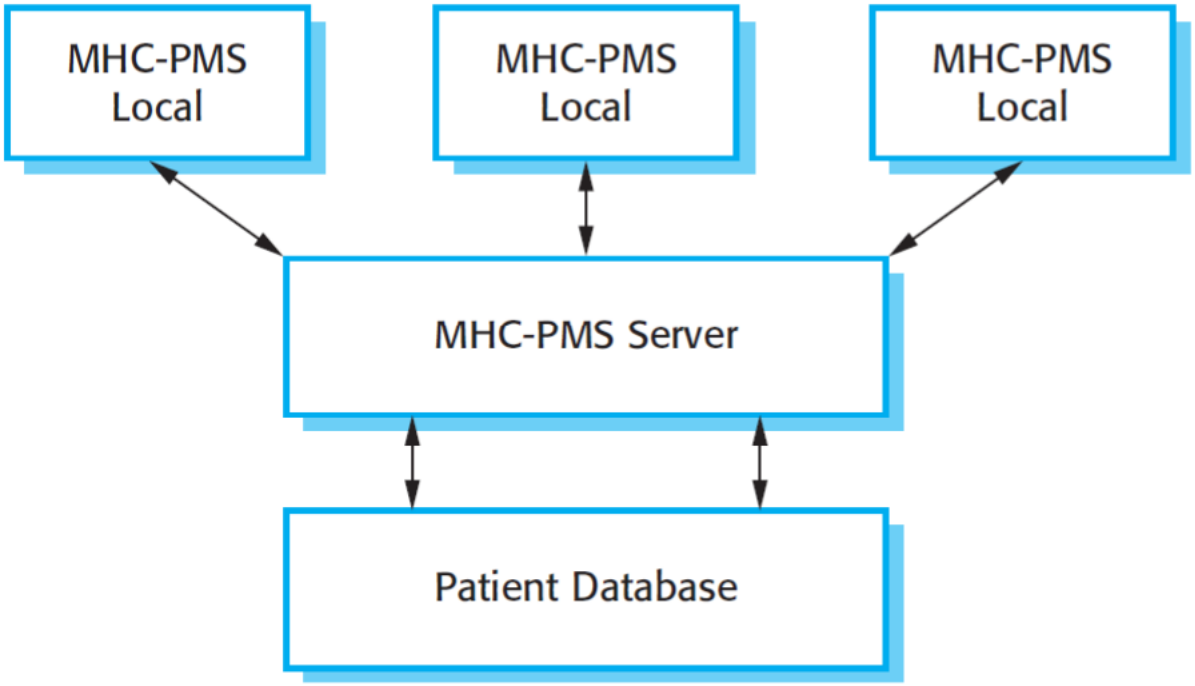

Fig: The organization of the MHC-PMS

A wilderness weather station

To help monitor climate change and to improve the accuracy of weather forecasts in

remote areas, the government of a country with large areas of wilderness decides to

deploy several hundred weather stations in remote areas. These weather stations collect

data from a set of instruments that measure temperature and pressure, sunshine,

rainfall, wind speed, and wind direction

The systems consists of:

1**.The weather station system** This is responsible for collecting weather data, carrying out some initial data processing, and transmitting it to the data management system.

2. **The data management and archiving system** This system collects the data from all of the wilderness weather stations, carries out data processing and analysis, and archives the data in a form that can be retrieved by other systems, such as weather forecasting systems.

3. **The station maintenance system** This system can communicate by satellite with all wilderness weather stations to monitor the health of these systems and provide reports of problems. It can update the embedded software in these systems. In the event of system problems, this system can also be used to remotely control a wilderness weather system.

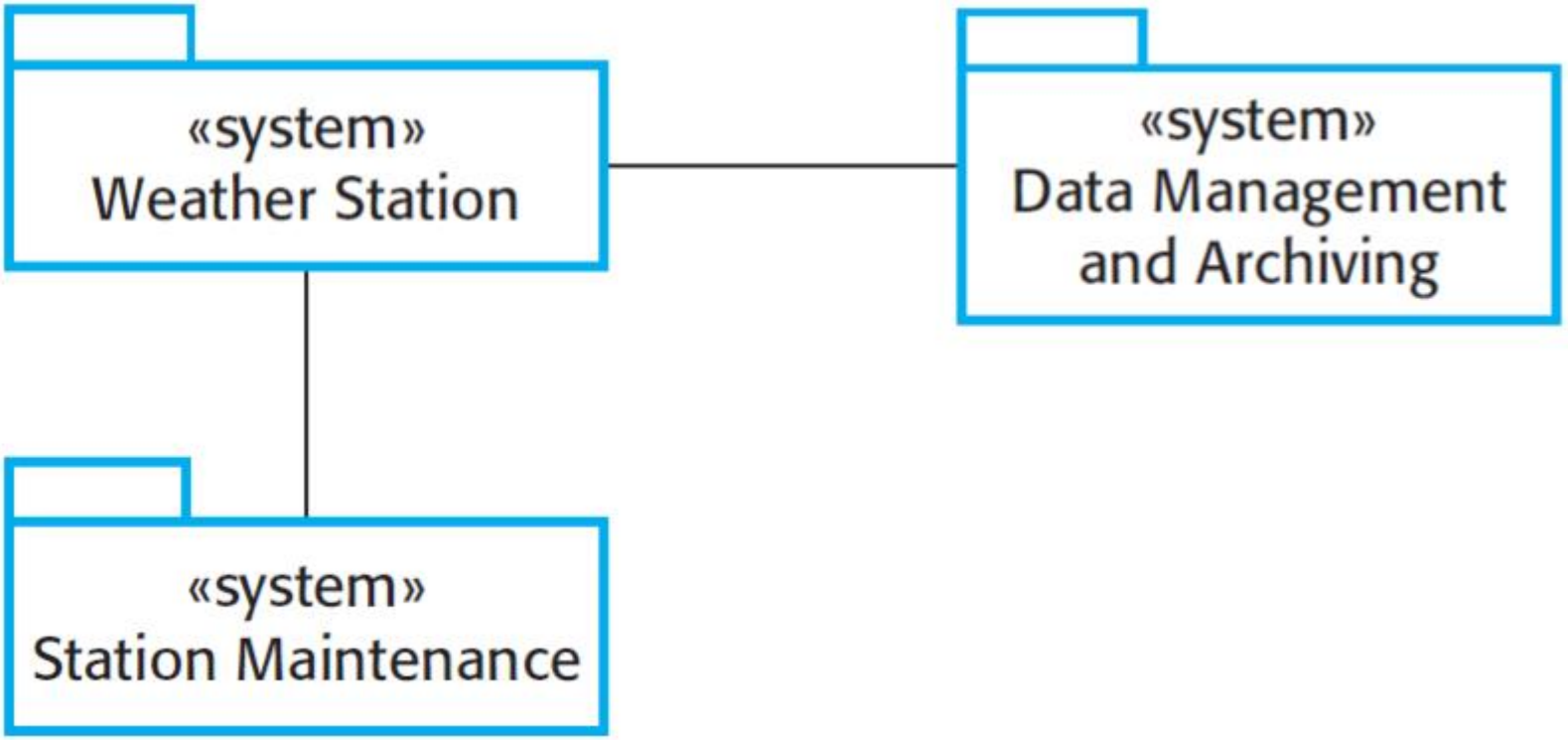

Fig: The weather station's environment

- Each weather station includes a number of instruments that measure weather parameters such as the wind speed and direction, the ground and air temperatures, the barometric pressure, and the rainfall over a 24-hour period.

- Each of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from the instruments.

- The weather station system operates by collecting weather observations at frequent intervals—for example, temperatures are measured every minute.

- However, because the bandwidth to the satellite is relatively narrow, the weather station carries out some local processing and aggregation of the data.

- It then transmits this aggregated data when requested by the data collection system.

- If, for whatever reason, it is impossible to make a connection, then the weather station maintains the data locally until communication can be resumed

The weather station must also:

1. Monitor the instruments, power, and communication hardware and report faults to the management system.

2. Manage the system power, ensuring that batteries are charged whenever the environmental conditions permit but also that generators are shut down in potentially damaging weather conditions, such as high wind.

3. Allow for dynamic reconfiguration where parts of the software are replaced with new versions and where backup instruments are switched into the system in the event of system failure

Software process models

A software process model is a simplified representation

of a software process. Each process model represents a process from a particular perspective,

and thus provides only partial information about that process.

For example,

a process activity model shows the activities and their sequence but may not show the roles of the people involved in these activities.

The waterfall model

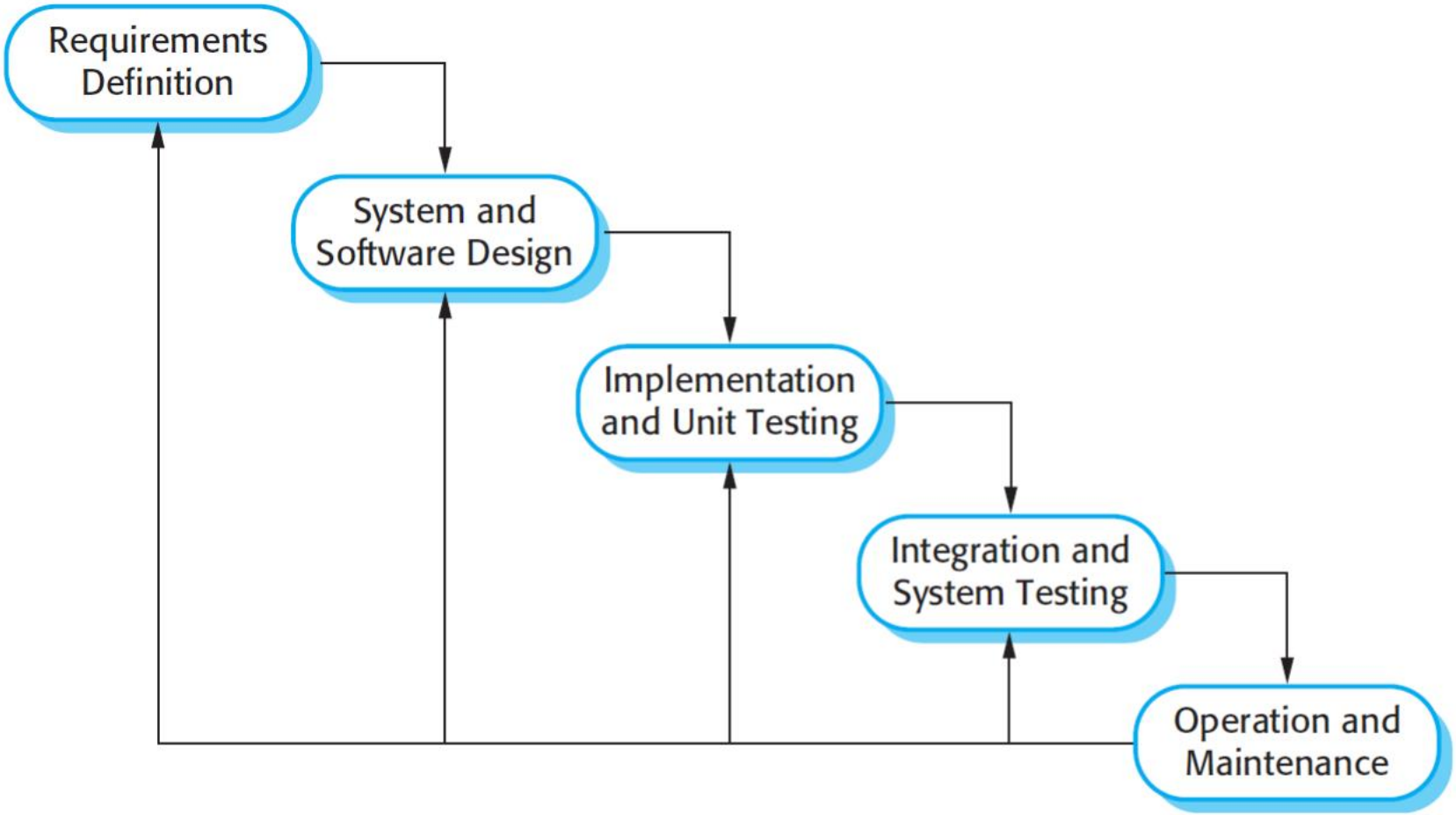

Fig: waterfall model

The waterfall model

The principal stages of the waterfall model directly reflect the fundamental development activities:

1. **Requirements analysis and definition** The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

2. **System and software design** The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

3. **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

4. 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.

5. **Operation and maintenance Normally** (although not necessarily), this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle

Incremental development

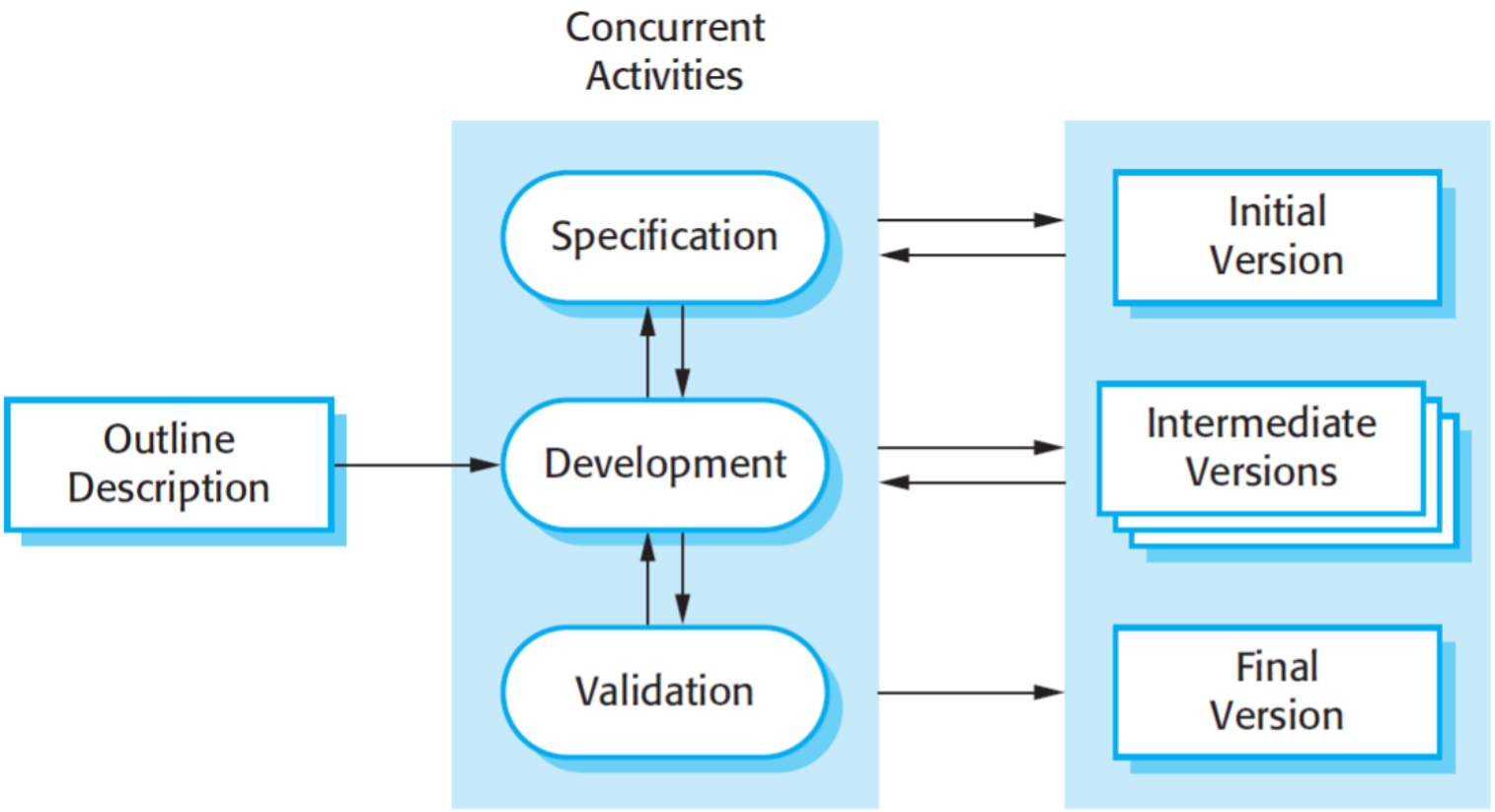

Fig: Incremental development

Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed

Incremental development has three important benefits, compared to the waterfall model:

1. The cost of accommodating changing customer requirements is reduced.

2. It is easier to get customer feedback on the development work that has been done.
3. More rapid delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included.

From a management perspective, the incremental approach has two problems:

1. 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.

2. 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.

Reuse-oriented software engineering

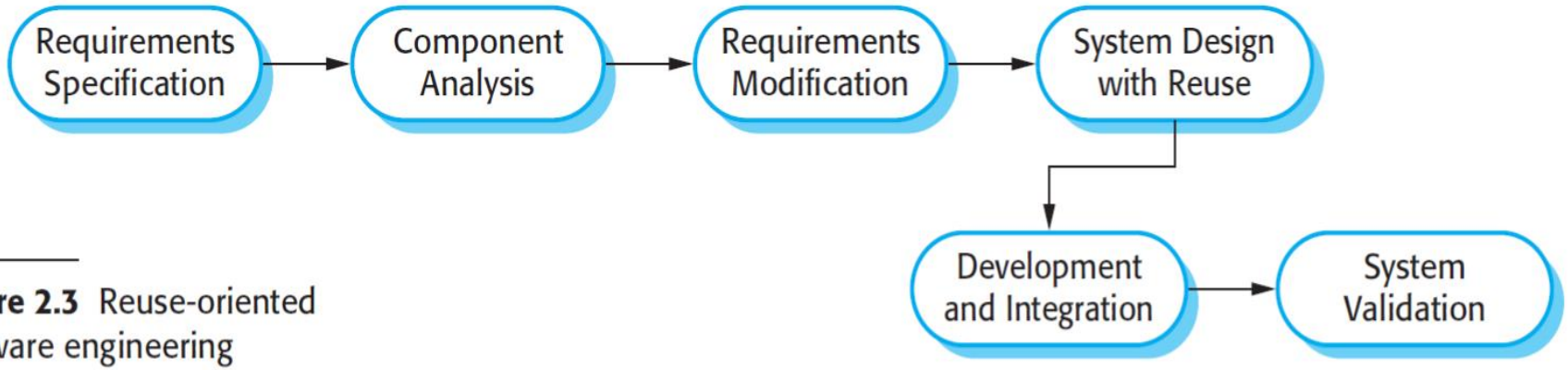

Figure 2.3 Reuse-oriented software engineering

In the majority of software projects, there is some software reuse. This often happens informally when people working on the project know of designs or code that are similar to what is required. They look for these, modify them as needed, and incorporate them into their system.

These stages are:

1. **Component analysis** Given the requirements specification, a search is made for components to implement that specification.

2. **Requirements modification** During this stage, the requirements are analyzed using information about the components that have been discovered. They are then modified to reflect the available components.

3. **System design with reuse** During this phase, the framework of the system is designed or an existing framework is reused. The designers take into account the components that are reused and organize the framework to cater for this.

4. **Development and integration** Software that cannot be externally procured is developed, and the components and COTS systems are integrated to create the new system.

There are three types of software component that may be used in a reuse-oriented process:

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

Boehm's spiral model

- A risk-driven software process framework (the spiral model) was proposed by Boehm.
- Here, the software process is represented as a spiral, rather than a sequence of activities with some backtracking from one activity to another.
- Each loop in the spiral represents a phase of the software process.
- Thus, the innermost loop might be concerned with system feasibility, the next loop with requirements definition, the next loop with system design, and so on

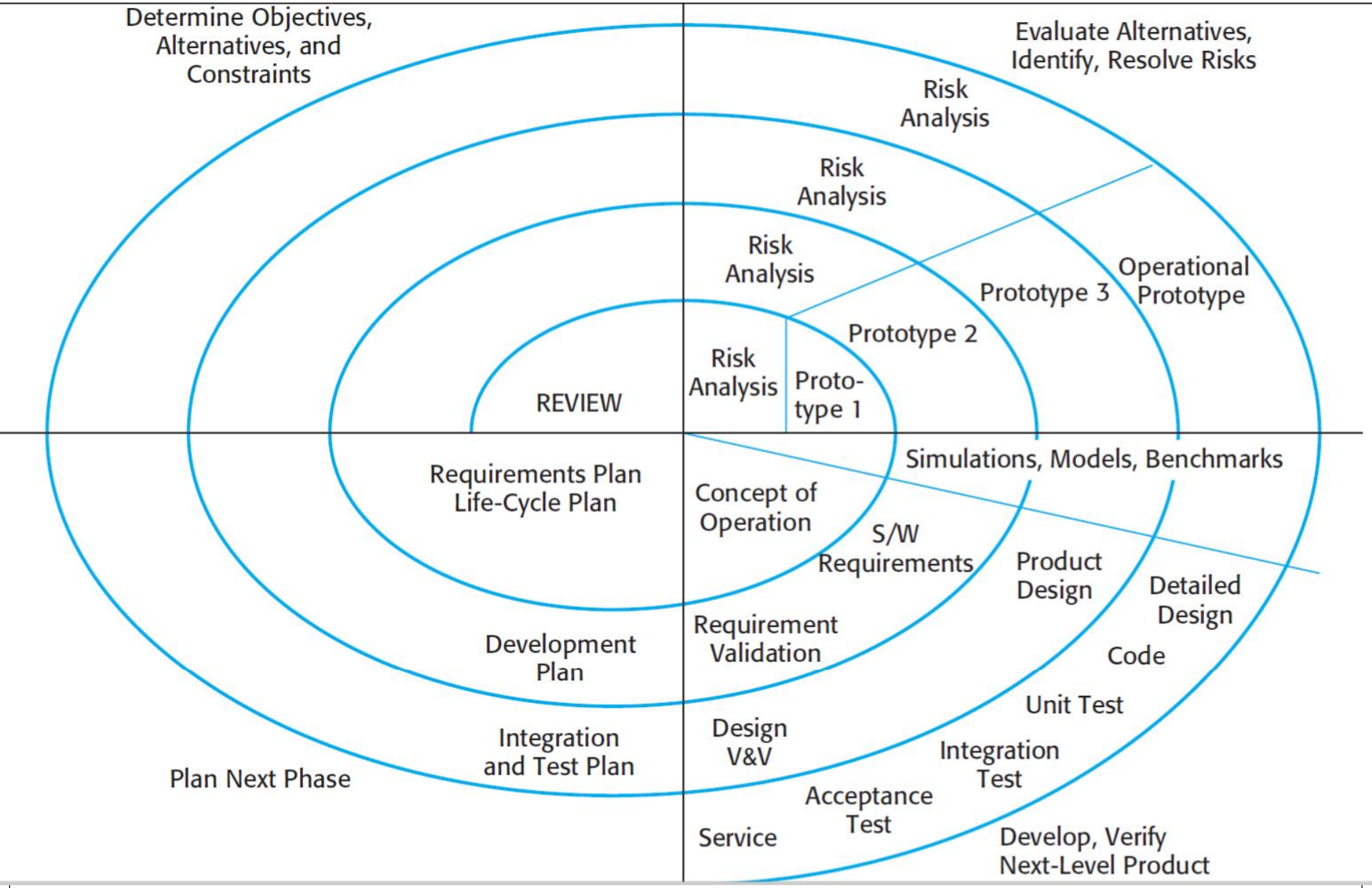

Fig: Boehm's spiral model of the software process

Process activities:
1. Software specification

Software specification or requirements engineering is the process of understanding and defining what services are required from the system and identifying the constraints on the syste

There are four main activities in the requirements engineering process:

1. Feasibility study
2. Requirements elicitation and analysis
3. Requirements specification
4. Requirements validation

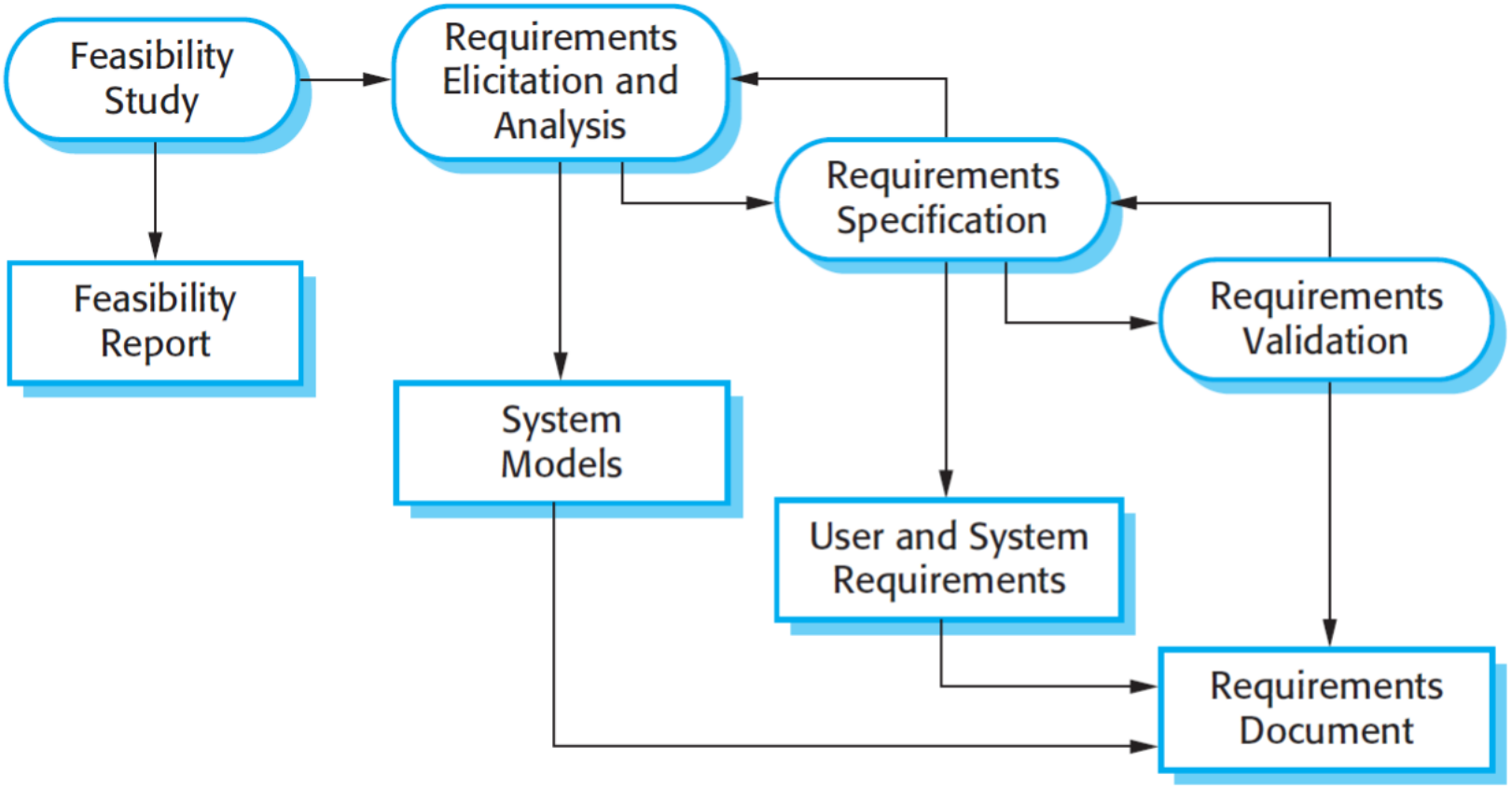

Fig: The requirements engineering process

2. Software design and implementation

- A software design is a description of the structure of the software to be implemented, the data models and structures used by the system, the interfaces between system components and, sometimes, the algorithms used
- The implementation stage of software development is the process of converting a system specification into an executable system.

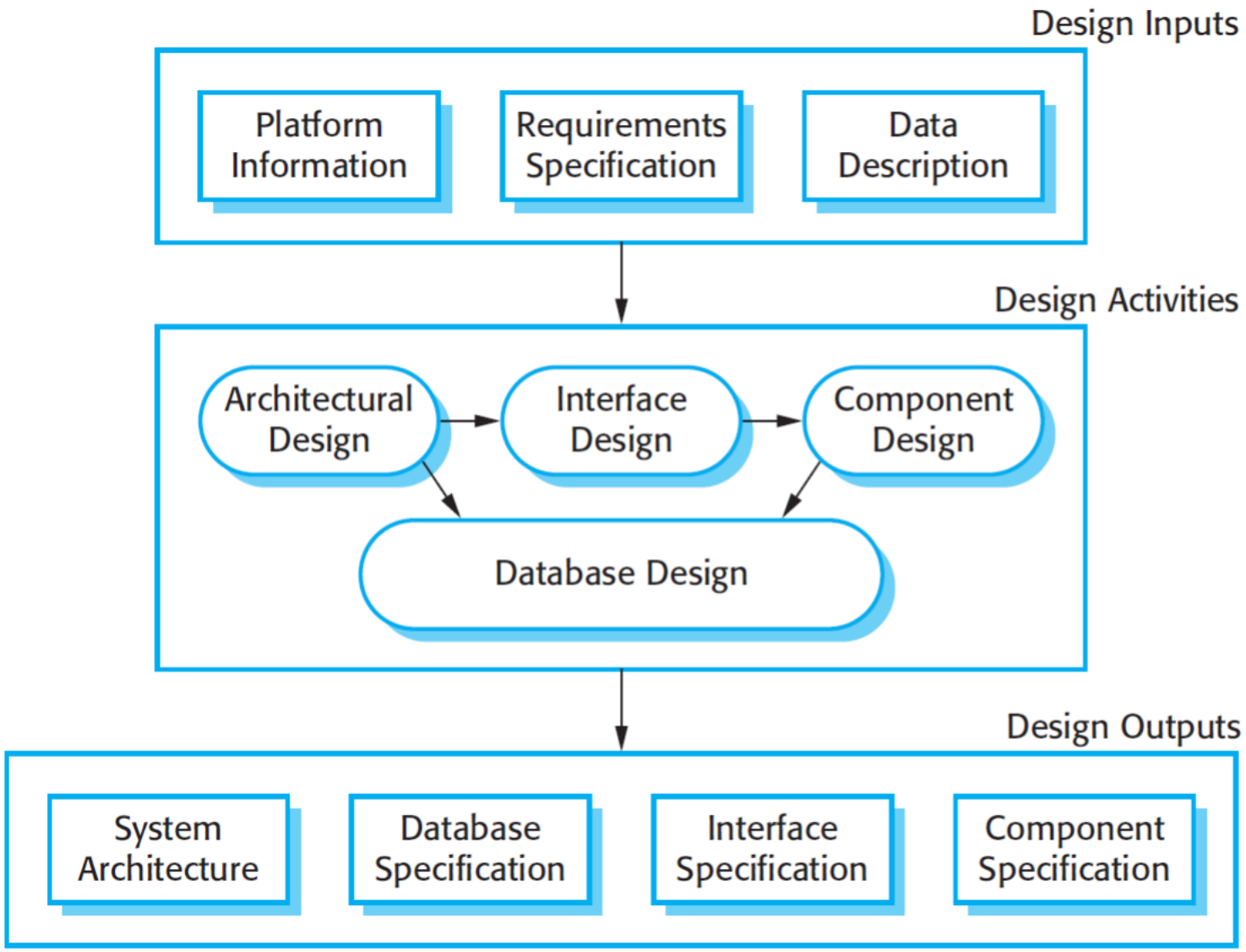

Fig: A general model of the design process

3. Software validation [verification and validation (V&V)]

- Is intended to show that a system both conforms to its specification and that it meets the expectations of the system customer.

- The stages in the testing process are:

1. 1. Development testing
2. System testing
3. Acceptance testing

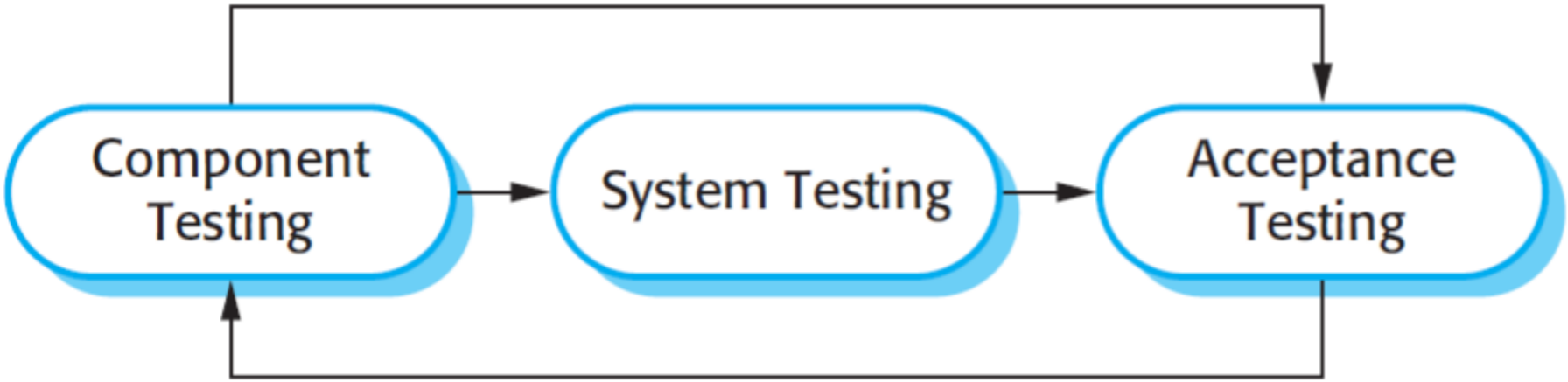

Fig: Stages of testing

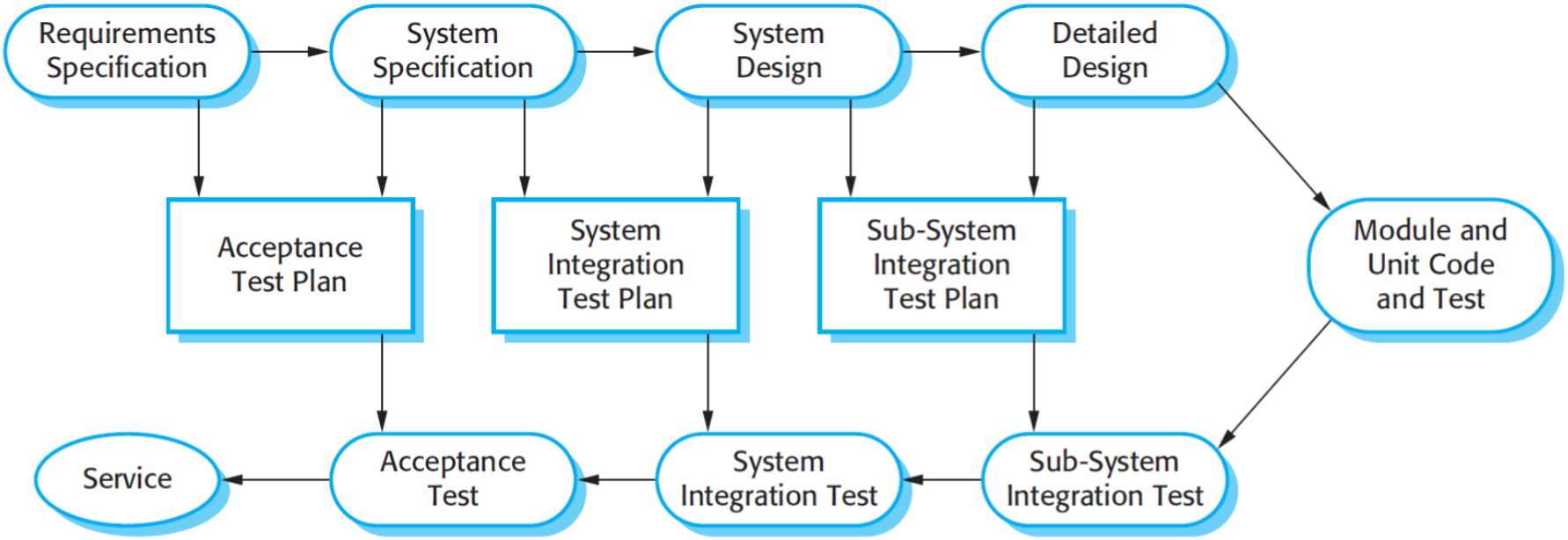

Fig: Testing phases in a plan-driven software process

4. Software evolution

The flexibility of software systems is one of the main reasons why more and more software is being incorporated in large, complex systems. Once a decision has been made to manufacture hardware, it is very expensive to make changes to the hardware design

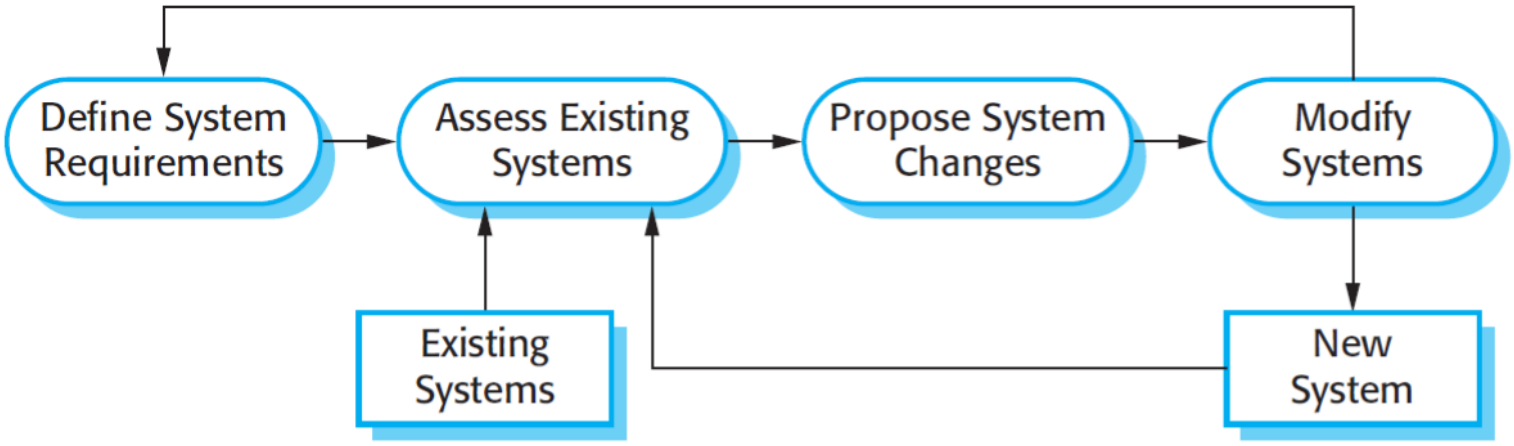

Fig: System evolution

End of Chapter 1