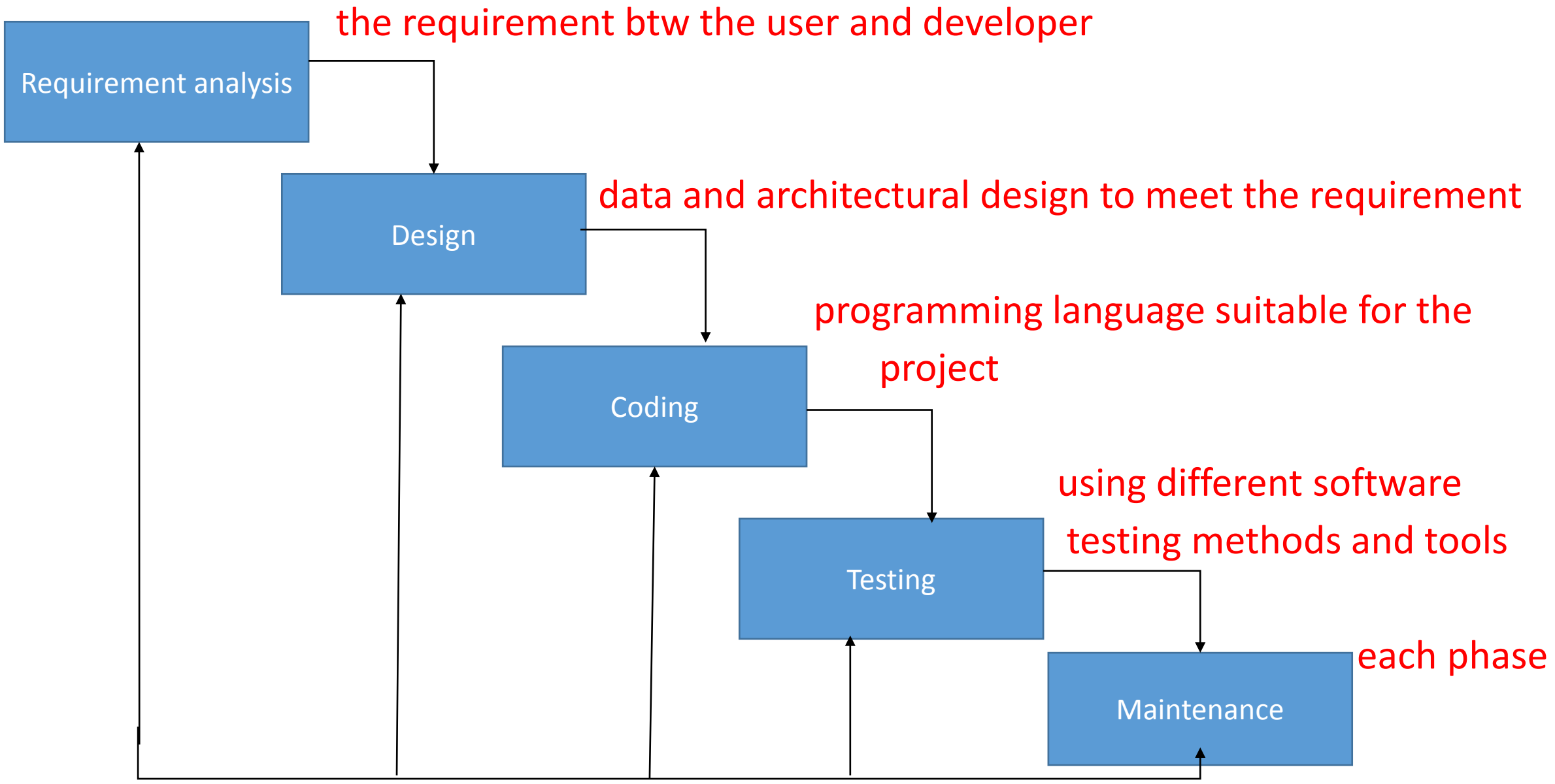
# SOFTWARE ENGINEERING : CHAPTER 1

- **SDLC Models**
- **Waterfall Model**
- **Spiral Model**
- **Iterative Model**
- **Professional And Ethical Responsibility**
- **8 PRINCIPLE Professional And Ethical.**
- **Software Engineering Challenges**

# SOFTWARE ENGINEERING PROCESS MODEL

## 1ST MODEL: WATERFALL MODEL



# When to use the waterfall model

- **The project is small**
- **Technology is understood.**
- **Product definition is stable.**
- **Requirements are very well known, clear and fixed.**

# Advantages

Easy to understand and implement

Its widely used and known
(in theory)

It allows for communication btw
customer n developer, to specify
What will be delivered and what cost

# Disadvantage

Its linear fashion, software process
flow stepwise from one phase to another.

Development of new system its not suitable
Bcoz it takes lot of effort and time.

It Requires the user to define system
requirements early in the project.

# SPIRAL MODEL

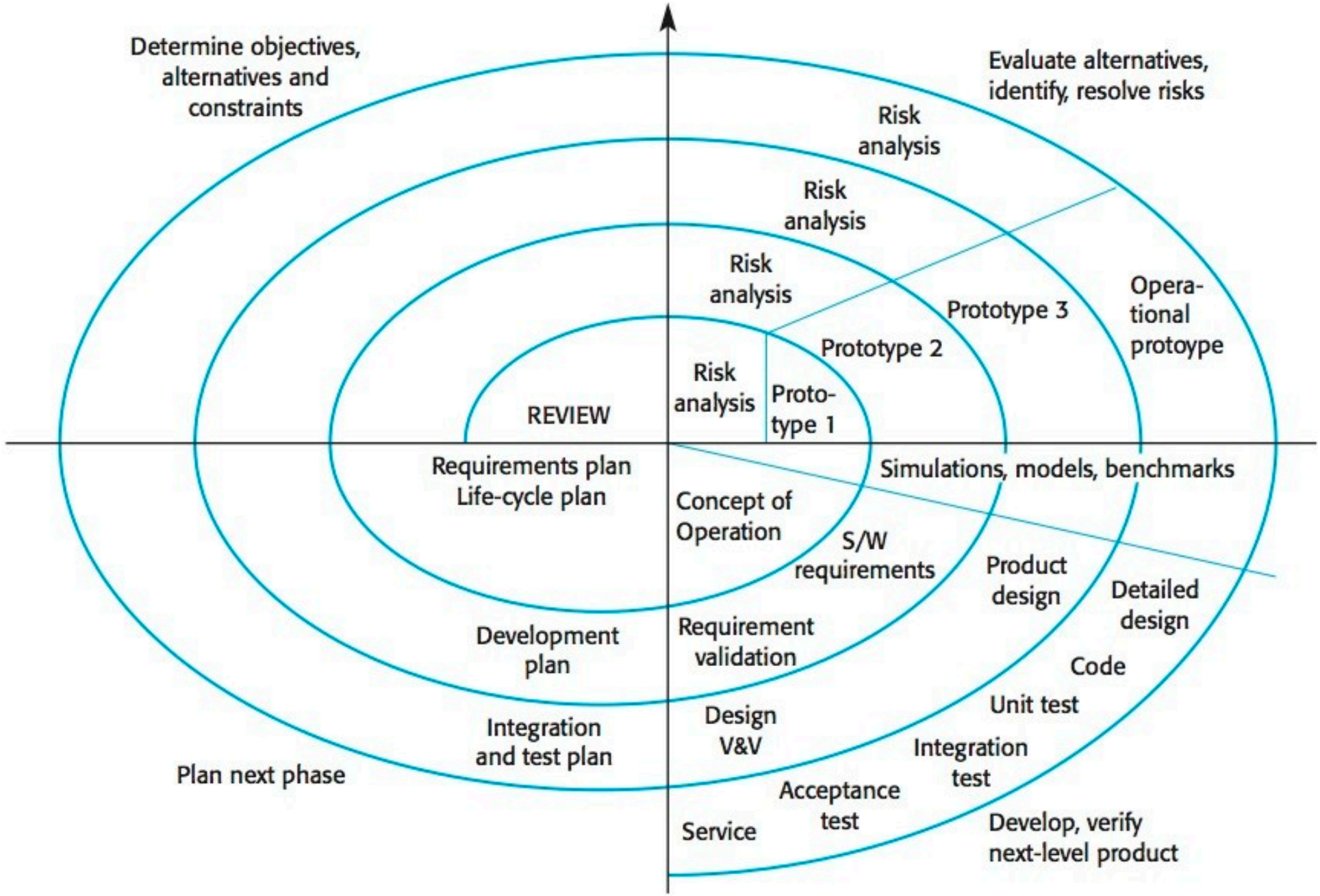
- The spiral model was proposed by Boehm, so its called Boehm spiral model.

- This model copies the iterative nature of prototyping with controlled and systematic aspects of waterfall model.

- It has 2 distinguishing features:

1. One is cyclic approach for increment process system's. Degree of implementation while decreasing the degree of risk.

2. Other is a set of anchor point milestone for ensuring satisfactory system solution.


Determine objectives, alternatives and constraints
Evaluate alternatives, identify, resolve risks
Risk analysis
Risk analysis
Risk analysis
Risk analysis
Prototype 3
Prototype 2
Operational prototype
Risk analysis
Prototype 1
REVIEW
Requirements plan
Life-cycle plan
Simulations, models, benchmarks
Concept of Operation
S/W requirements
Product design
Detailed design
Development plan
Requirement validation
Code
Unit test
Integration and test plan
Design V&V
Integration test
Acceptance test
Plan next phase
Service
Develop, verify next-level product

- Important Phases of Spiral Model are:

Phase 1 : Objective settings (s/w targets, restrictions, risk,plan alternation)
Phase 2 : Risk analysis (identify and consider how to reduce the risk)
Phase 3 : Development (develop s/w process model and test the s/w)
Phase 4 : Planning (connecting the phases to the next phase of the project)

# When to use spiral model

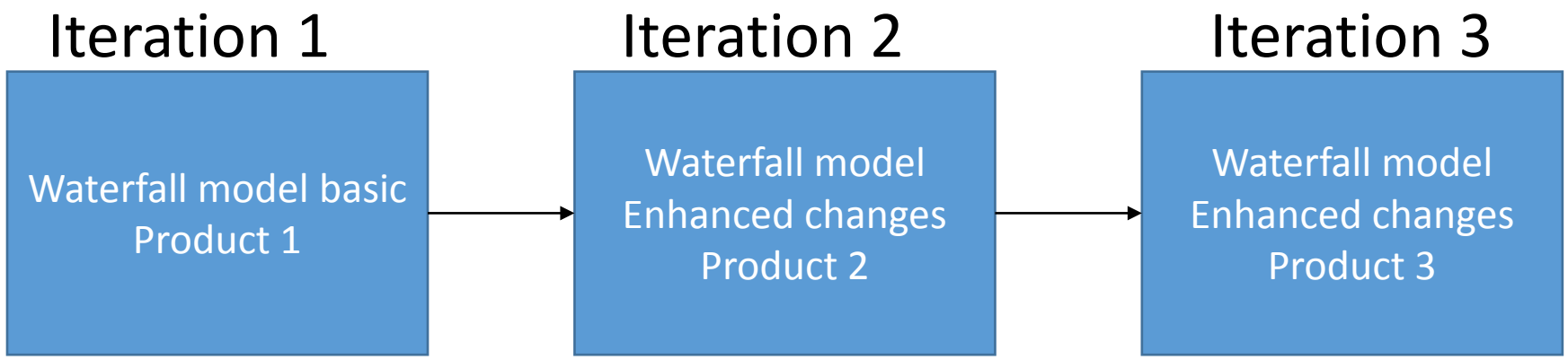
- When costs and risk evaluation is important.
- For medium to high-risk projects
- Long term project commitment
- Users are unsure of their needs
- Requirements are complex

| Advantage | Disadvantage |
| --- | --- |
| s/w developers actively look for possible risks and analyse it. | cost is high and risk analysis requires Highly specific expertise. |
| This model uses prototyping as risk Reduction mechanism | doesn’t work well for small projects and nt suitable for low risk projects. |
| It maintains systematic stepwise approach | skills required, reviewing project time to time , need expertise. |
| It reduce risks before they become Problematic | difficult to convince the customer |

# ITERATIVE ENHANCEMENT MODEL

- This model helps to remove the short comings of waterfall model.
- The process is done in an iterative (step by step) fashion.
- Every iteration consists of phases of waterfall model.

- In the first iteration, a less capable product are developed and delivered for use.
- Next iteration, with incremental features is developed.


Iteration 1
Iteration 2
Iteration 3
Waterfall model basic
Product 1
Waterfall model
Enhanced changes
Product 2
Waterfall model
Enhanced changes
Product 3

# Advantage

Useful when more manpower
Is available for development

Useful when the release
deadlines are tight

# Disadvantage

Iteration may never end, user have
to wait for final product.

cost estimation is so high

# Professional and Ethical Responsibility

- Software Engineering activities has to be carried out within legal and ethical frame work.

PROFESSIONAL RESPONSIBILITIES INCLUDE:

1. Confidentiality
2. Competence
3. Intellectual property rights
4. Computer misuse

# 8 Principles of software Engineering ethics and professional practice

1. Principle of public
2. Principle of client and employer
3. Principle of product
4. “ “ “ management
5. “ “ “ judgement
6. “ “ “ profession
7. “ “ “ colleagues
8. “ “ “ self

# Software Engineering Challenges

1. Legacy Challenges → huge s/w sys used today were developed many yrs ago, challenge is to maintaining and updating the s/w.
2. Heterogeneity Challenges → distributed sys across n/w. challenges of developing techniques to build different kinds of support sys.
3. Delivery Challenges → s/w project are time consuming to be manufactured and maintain quality, so Challenges is on time to deliver the project
4. Trust Challenges → Maintaining the trust.

Other Challenges are: -

Changing Requirements → adapting for hardware changes

Changing Optimism → Estimate how long it take to develop and future

- What is software engineering ? 2m
- Define system? 2m
- Differentiate system software and application  software? 3m
- Explain the characteristic of software? 5m
- Explain the different phases of SDLC? 5m
- Name different types of SDLC models? 2m
- Differentiate generic product and customized product? 3m
- Explain the different phases of spiral model with advantage and disadvantage?  8m
- Explain about Risk Management ? 6m
- Explain the challenges of software engineers? 5m
- What are the professional and ethical responsibilities of Software engineering? 5m
- Explain the different phases of Waterfall model with advantage and disadvantage?  8m
- Compare waterfall model and spiral model? 5m

# SOFTWARE ENGINEERING : CHAPTER 2

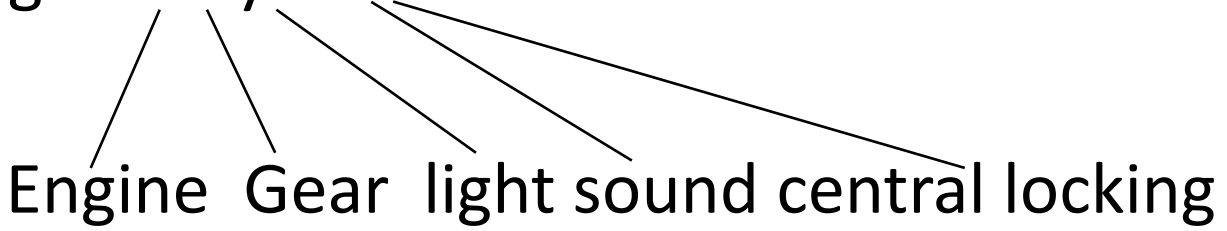
## ➢ SYSTEM ENGINEERING

- System, System Engineering
- Emergent Properties With Examples
- System Procurement
- Contractors And Sub-contractors
- The System  Design Processing
- Functional System Components

# What is a system?

- A system is the collection of inter-related components working together towards some common objective.
- A system may include software, mechanical, electrical and electronic hardware and be operated by people.
- System components are dependent on other system components ie subsystem

Eg : car system



# Systems Engineering

- Is the activity which provides the framework within which complex system can be defined, analysed, specified, manufactured, operated and support.

The system may include:

Software

Hardware

people

# Emergent properties

- Properties of the system as a whole rather than properties that can be derived from the properties of components of a system
- Emergent properties are a consequence of the relationships between system components
- They can therefore only be assessed and measured once the components have been integrated into a system

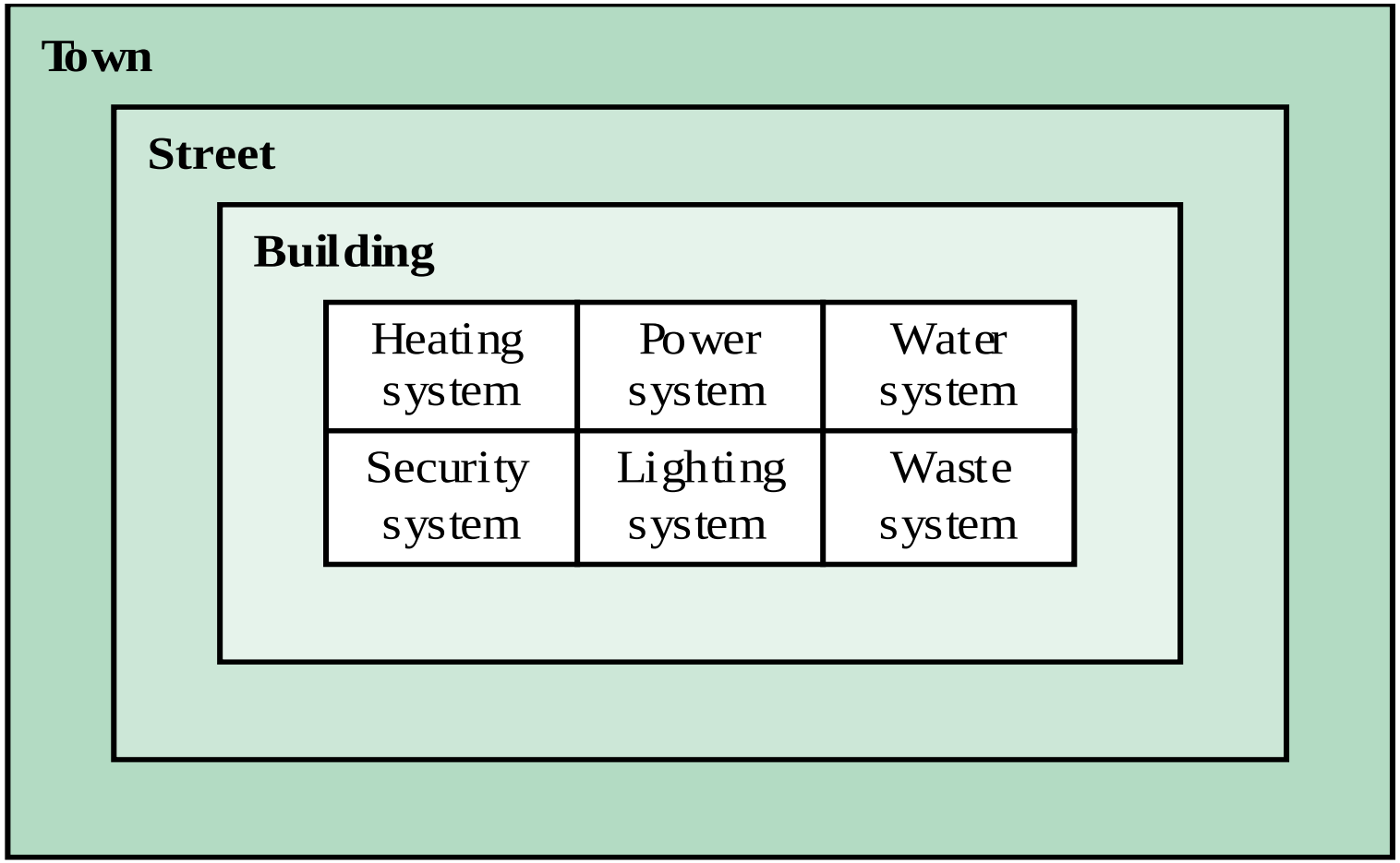
# Examples of emergent properties

- *The overall weight of the system*
  - This is an example of an emergent property that can be computed from individual component properties.
- *The reliability of the system*
  - This depends on the reliability of system components and the relationships between the components.
- *The usability of a system*
  - This is a complex property which is not simply dependent on the system hardware and software but also depends on the system operators and the environment where it is used.

# Systems and their environment

- Systems are not independent but exist in an environment
- System’s function may be to change its environment
- Environment affects the functioning of the system e.g. system may require electrical supply from its environment
- The organizational as well as the physical environment may be important

# System hierarchies



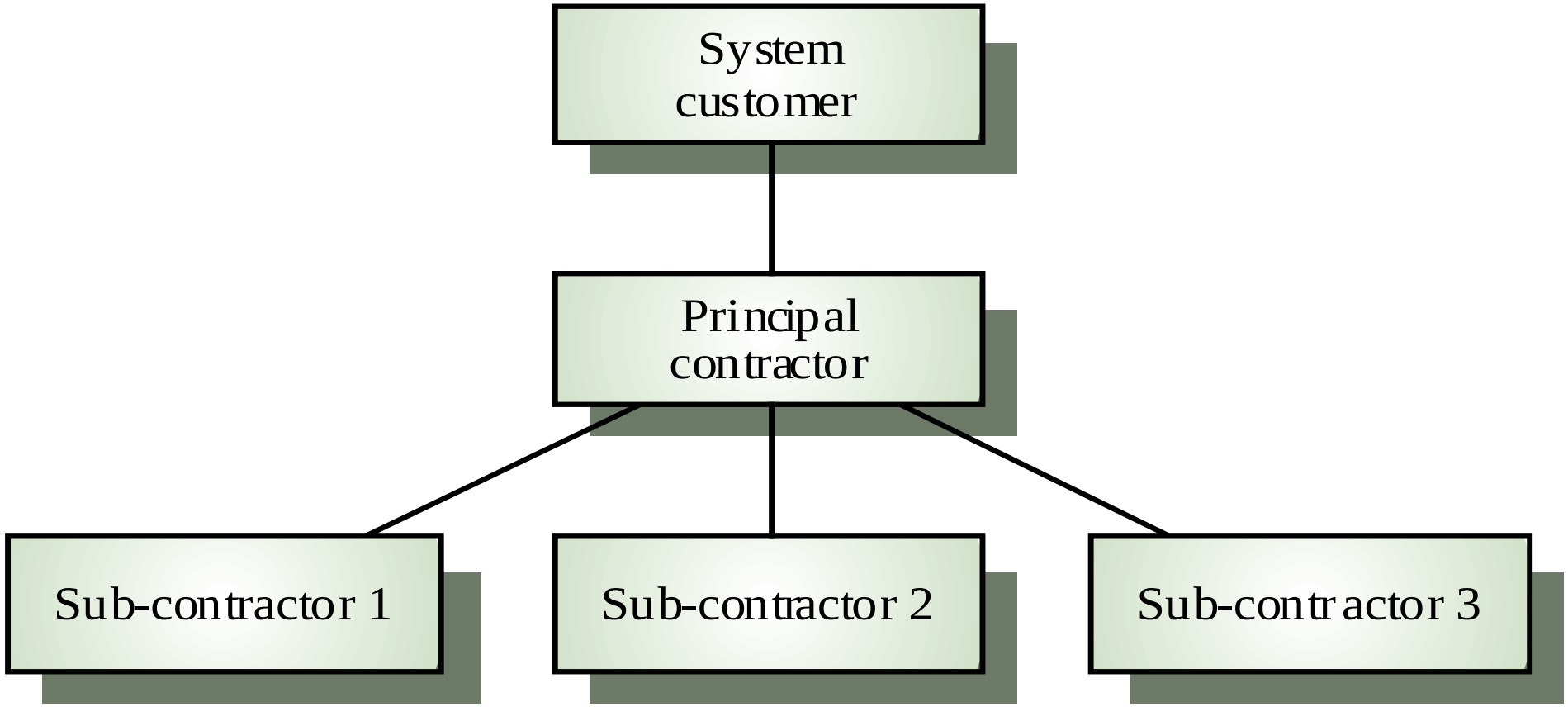
# System procurement

- Acquiring a system for an organization to meet some need
- Some system specification and architectural design is usually necessary before procurement
  - You need a specification to let a contract for system development
  - The specification may allow you to buy a commercial off-the-shelf (COTS) system. Almost always cheaper than developing a system from scratch

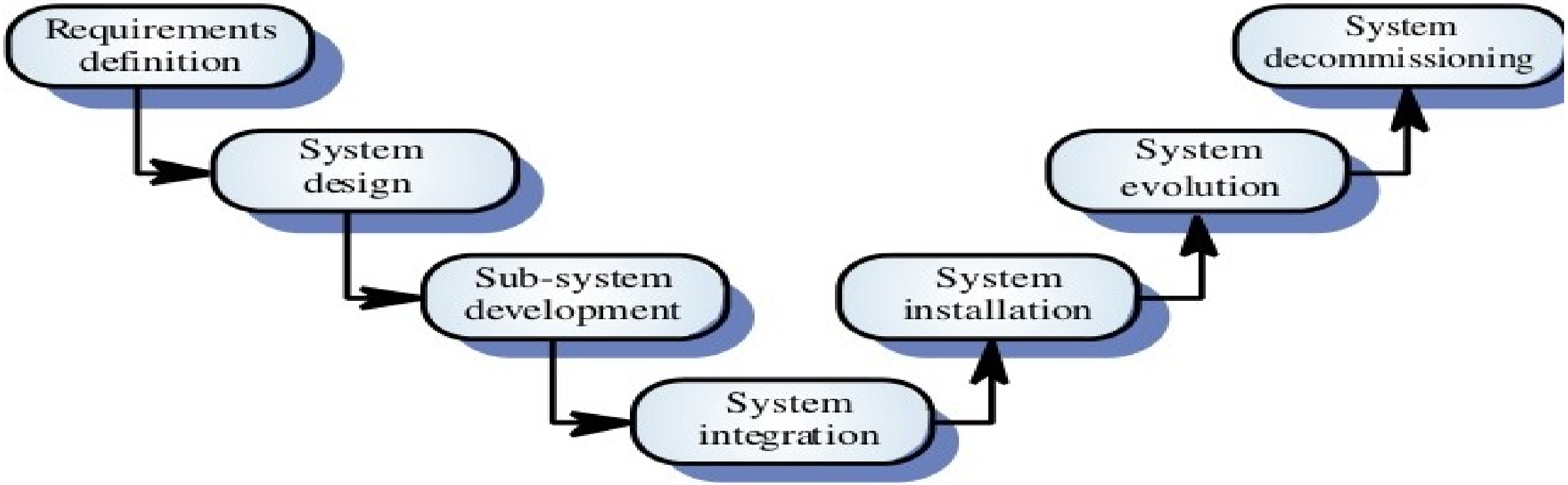
# Contractors and sub-contractors

- The procurement of large hardware/software systems is usually based around some principal contractor
- Sub-contracts are issued to other suppliers to supply parts of the system
- Customer does not deal directly with sub-contractors

# Contractor/Sub-contractor model



# The system engineering process



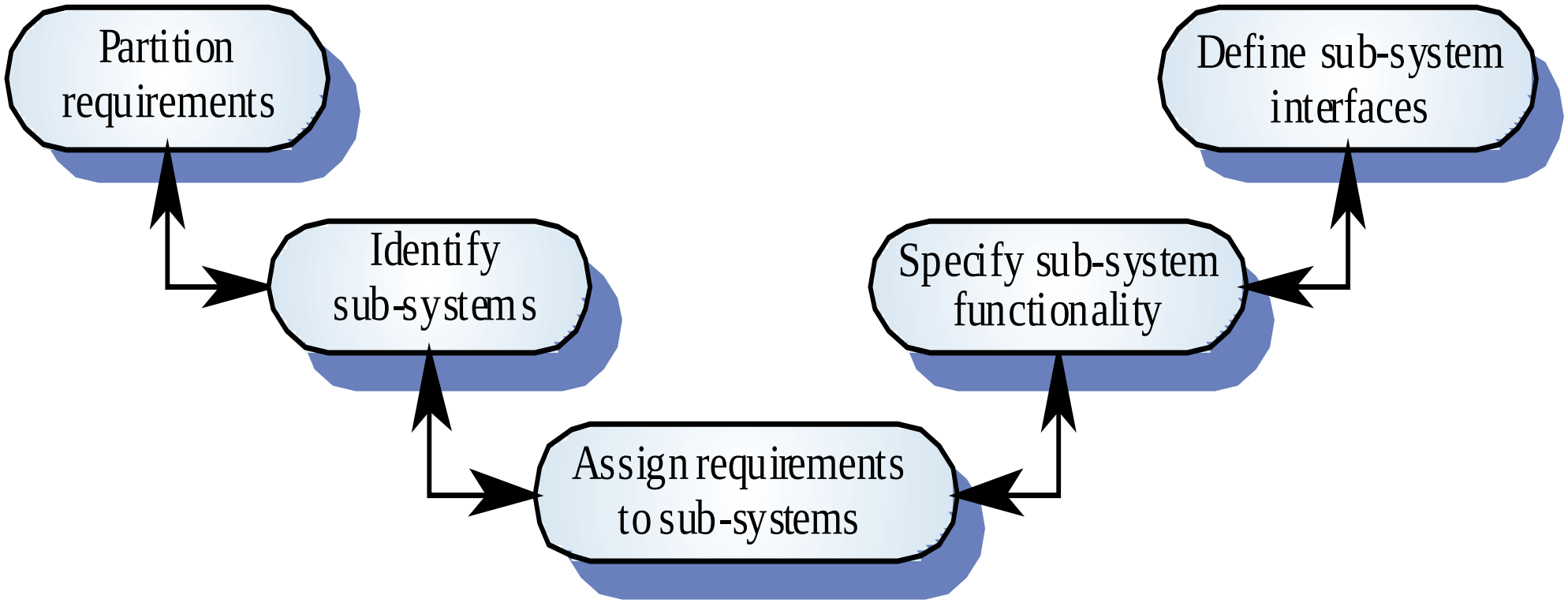
# System requirements definition

- Three types of requirement defined at this stage
  - Abstract functional requirements. System functions are defined in an abstract way
  - System properties. Non-functional requirements for the system in general are defined
  - Undesirable characteristics. Unacceptable system behaviour is specified
- Should also define overall organisational objectives for the system

# The system design process

- Partition requirements
  - Organise requirements into related groups
- Identify sub-systems
  - Identify a set of sub-systems which collectively can meet the system requirements
- Assign requirements to sub-systems
  - Causes particular problems when COTS are integrated
- Specify sub-system functionality
- Define sub-system interfaces
  - Critical activity for parallel sub-system development

# The system design process



# Sub-system development

- Typically parallel projects developing the hardware, software and communications
- May involve some COTS  (Commercial Off-the-Shelf) systems procurement
- Lack of communication across implementation teams
- slow mechanism for proposing system changes means that the development schedule may be extended because of the need for rework

# System integration

- The process of putting hardware, software and people together to make a system
- Should be incremental integration so that sub-systems are integrated one at a time
- Interface problems between sub-systems are usually found at this stage
- May be problems with uncoordinated deliveries of system components

# System installation

- Environmental assumptions may be incorrect
- May be human resistance to the introduction of a new system
- System may have to coexist with alternative systems for some time
- May be physical installation problems (e.g. cabling problems)
- Operator training has to be identified

# System evolution

- Large systems have a long lifetime. They must evolve to meet changing requirements
- Evolution is very costly for a number of reasons:
  - Changes must be analysed from a technical and business perspective
  - Sub-systems interact so unanticipated problems can arise
  - There is rarely a rational for original design decisions
  - System structure is corrupted as changes are made to it
- Existing systems which must be maintained are sometimes called legacy systems

# System decommissioning

- Taking the system out of service after its useful lifetime
- May require removal of materials (e.g. dangerous chemicals) which pollute the environment
  - Should be planned for in the system design by encapsulation
- May require data to be restructured and converted to be used in some other system
- Hardware system may involves de-assembling and recycling materials.

# System architecture modelling

- An architectural model presents an abstract view of the sub-systems making up a system
- May include major information flows between sub-systems
- Usually presented as a block diagram
- May identify different types of functional component in the model

# Functional system components

- Sensor components
- Actuator components
- Computation components
- Communication components
- Co-ordination components
- Interface components

# System components

- Sensor components
  - Collect information from the system’s environment e.g. radars in an air traffic control system
- Actuator components
  - Cause some change in the system’s environment e.g. valves in a process control system which increase or decrease material flow in a pipe
- Computation components
  - Carry out some computations on an input to produce an output e.g. a floating point processor in a computer system

# System components

- Communication components
  - Allow system components to communicate with each other e.g. network linking distributed computers
- Co-ordination components
  - Co-ordinate the interactions of other system components e.g. scheduler in a real-time system
- Interface components
  - Facilitate the interactions of other system components e.g. operator interface
- All components are now usually software controlled

# System reliability engineering

- Because of component inter-dependencies, faults can be propagated through the system
- System failures often occur because of unforeseen inter-relationships between components
- It is probably impossible to anticipate all possible component relationships
- Software reliability measures may give a false picture of the system reliability

# Influences on reliability

- *Hardware reliability*
  - What is the probability of a hardware component failing and how long does it take to repair that component?
- *Software reliability*
  - How likely is it that a software component will produce an incorrect output. Software failure is usually distinct from hardware failure in that software does not wear out.
- *Operator reliability*
  - How likely is it that the operator of a system will make an error?