CMPT 475: Software Engineering II

# Software Development Processes

**Dr. Herbert H. Tsang**, P.Eng., Ph.D.

Summer 2014

School of Computing Science
Simon Fraser University, Canada

# References

- Ian Sommerville. Software Engineering (9th edition), Pearson, 2011
  - Ch. 2: Software Processes

# Outline

- Software Development Process Activities
  - A) Software specification
  - B) Software design and implementation
  - C) Software validation
  - D) Software evolution
- Software Development Paradigms

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

# Software Development Process Activities

# General Software Development Process Activities

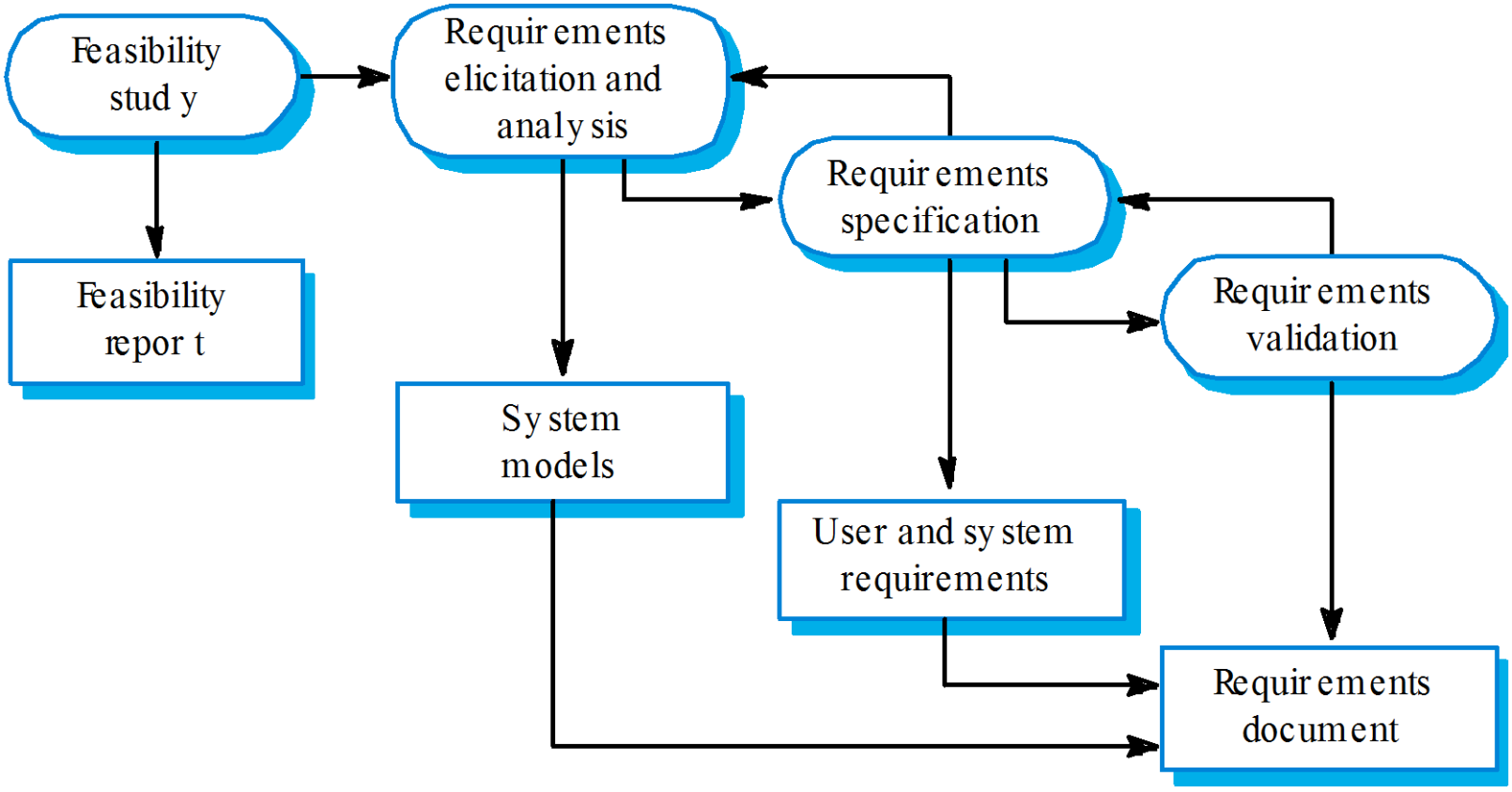
- A) Software specification
- B) Software design and implementation
- C) Software validation
- D) Software evolution

# A) SOFTWARE SPECIFICATION

# 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
    - Is it technically and financially feasible to build the system?
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
  - Requirements specification
    - Defining the requirements in detail
  - Requirements validation
    - Checking the validity of the requirements

# The requirements engineering process

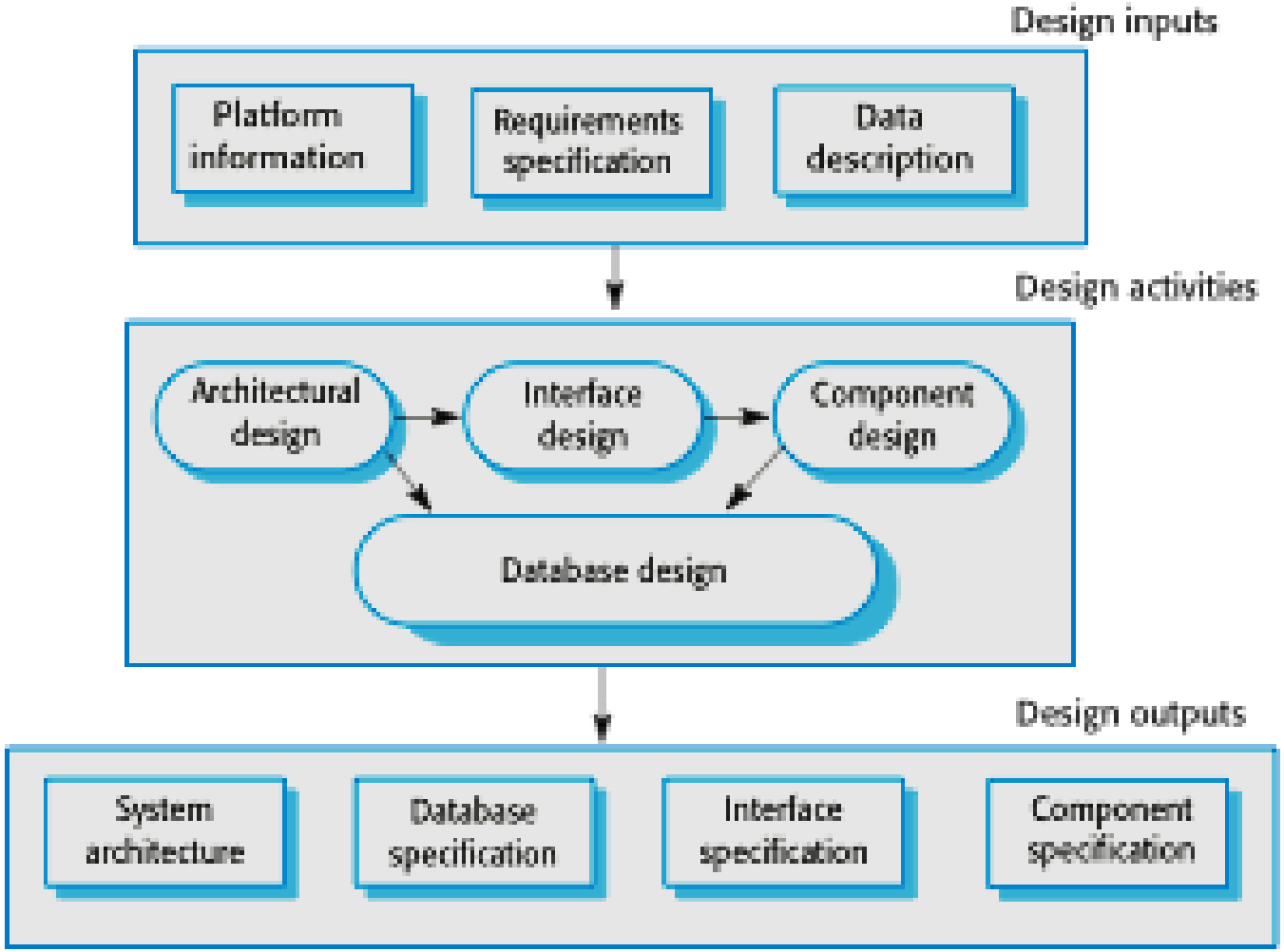


# B) SOFTWARE DESIGN AND IMPLEMENTATION

# Software design and implementation

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realizes the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.

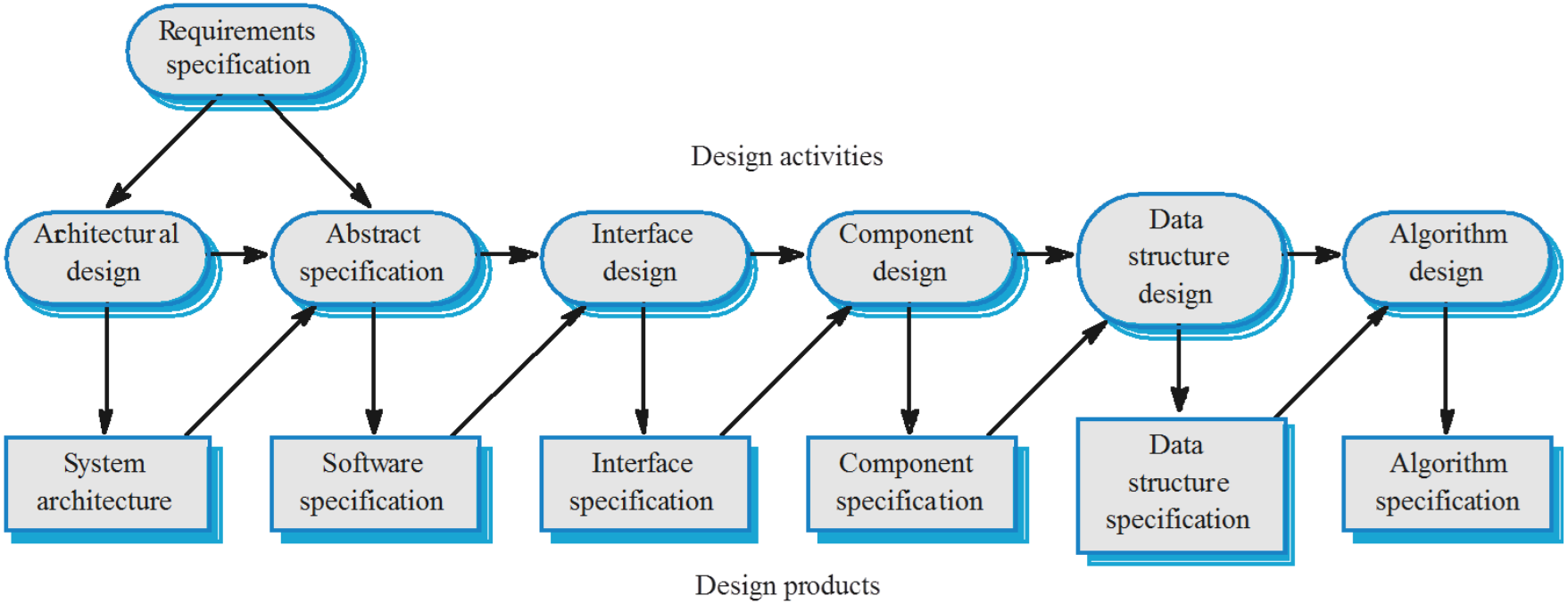
# A general model of the design process



# Design process activities

- 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.
- Interface design, where you define the interfaces between system components.
- Component design, where you take each system component and design how it will operate.
- Database design, where you design the system data structures and how these are to be represented in a database.

# The software design process



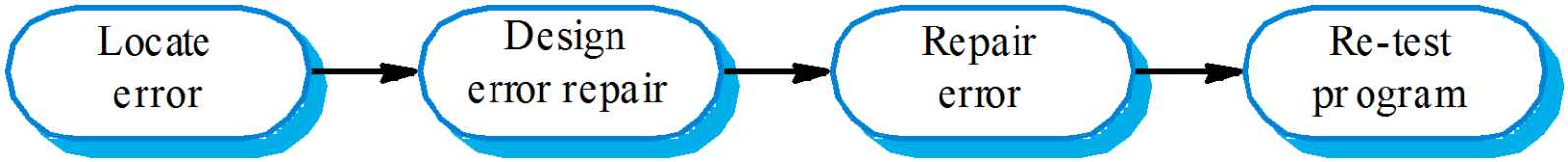
# Structured methods

- 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.

# The debugging process

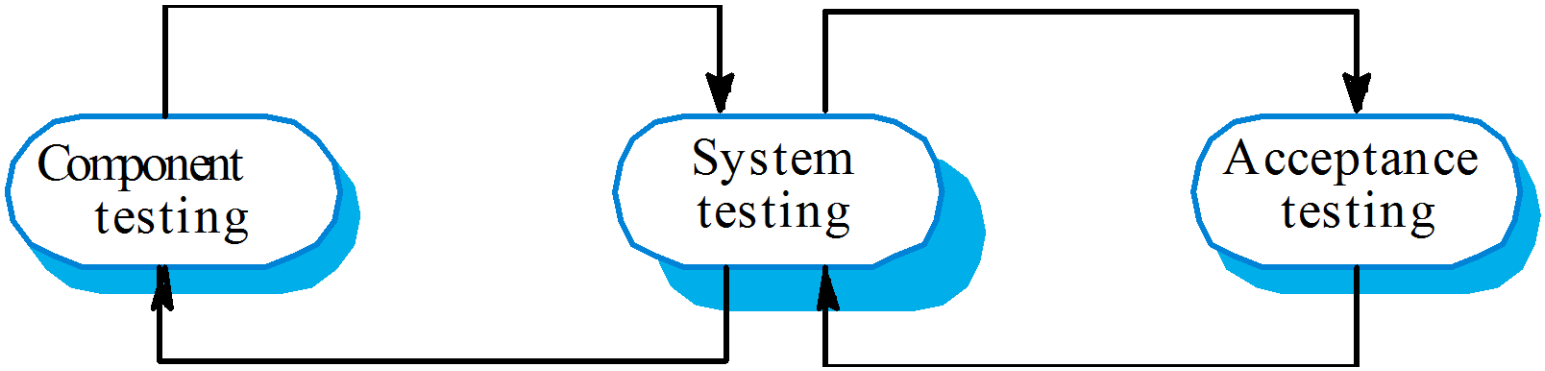


# C) SOFTWARE VALIDATION

# 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.

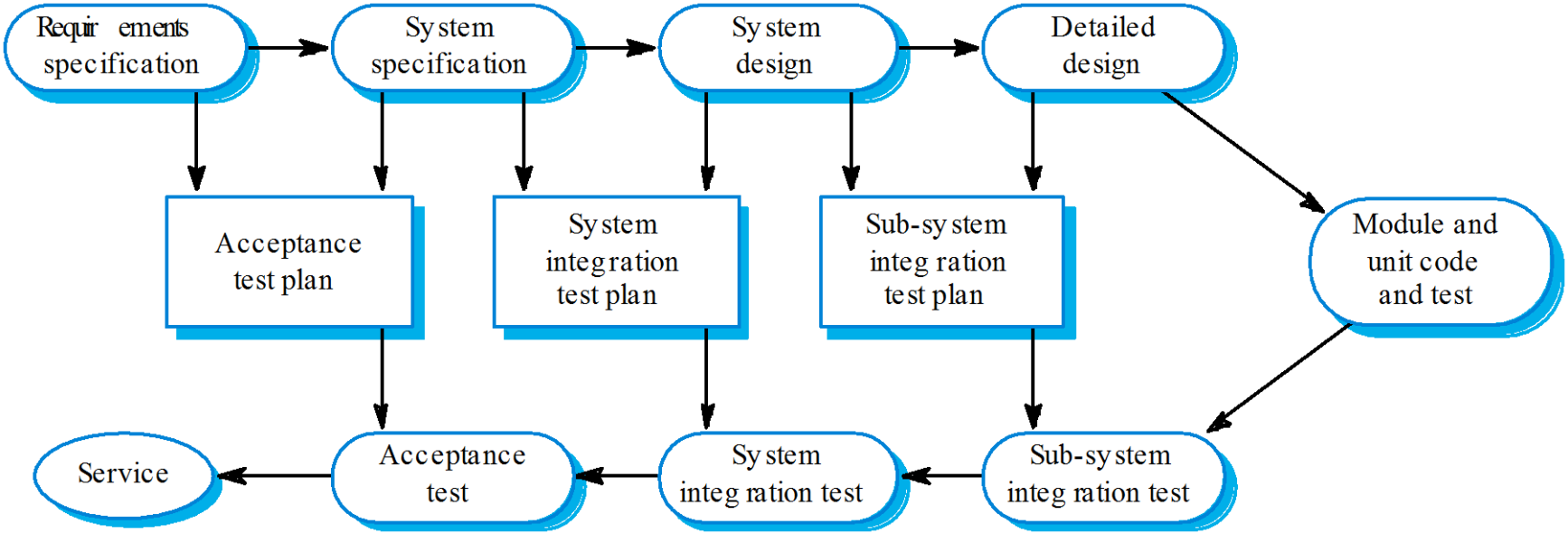
# The Stages of Testing



# Testing stages

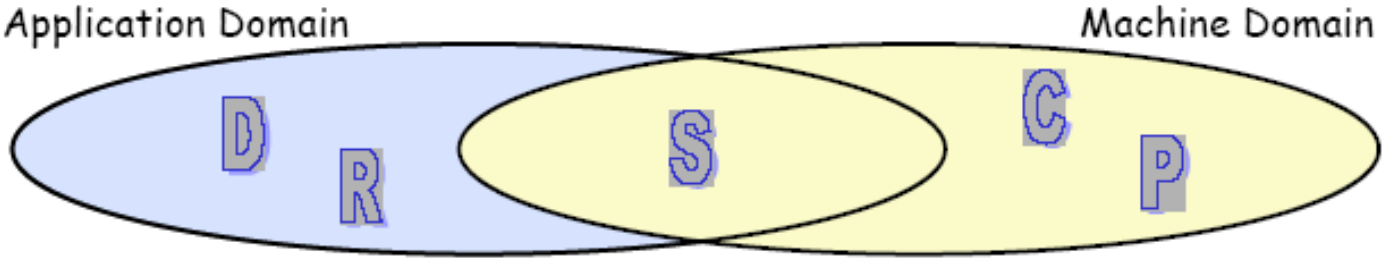
- Component or unit testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- Acceptance testing
  - Testing with customer data to check that the system meets the customer's needs.

# Testing phases



# Verification and Validation

*Source: Adapted from Jackson, 1995, p170-171*




- For V&V, we need to worry about:
  - The properties of the computer hardware (C)
  - The properties of the program (P)
  - The properties of the machine in the application domain (the specification, S)
  - The properties of the domain, independent of the machine (D)
  - The requirements for the machine (R)
- Demonstrating that P satisfies R is then a two step process:
  - Do C and P imply S? *(Verification)*
  - Do S and D imply R? *(Validation)*

# Validation Example

*Source: Adapted from Jackson, 1995, p172*

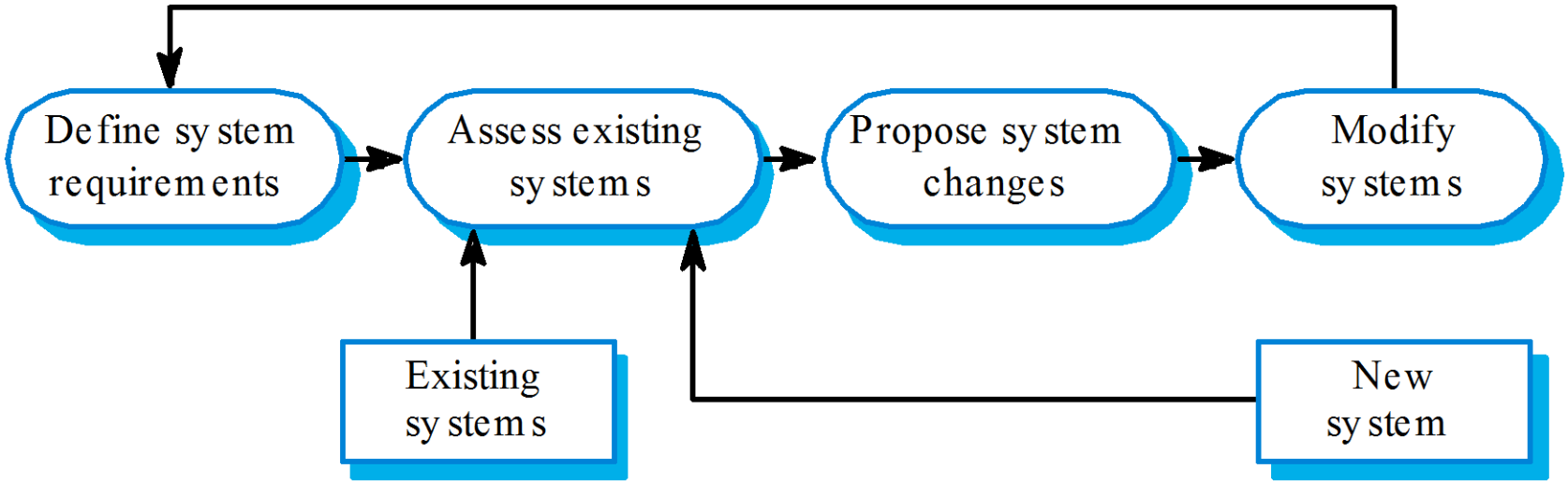
- **Requirement R:**
  - "Reverse thrust shall only be enabled when the aircraft is moving on the runway"
- **Domain Properties D:**
  - Wheel pulses on if and only if wheels turning
  - Wheels turning if and only if moving on runway
- **Specification S:**
  - Reverse thrust enabled if and only if wheel pulses on
- **S + D imply R**
  - But what if the domain model is wrong?

# D) SOFTWARE EVOLUTION

# 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.

# System evolution



# Key points

- 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.

# QUESTIONS?