5SENG007C

Software Engineering Principles and Practice

IIT ML - Cassim Farook

Cassim.f@iit.ac.lk

(notes are adopted from UoW ML - Dr Alexander Bolotov)

Please read the detailed slides with extra notes on BB

Mr. Cassim Farook

Senior Lecturer - Grade I

Good Morning!

I teach
@ IIT for
UoW BSc and BEng.

@ IIT for
RGU BSc AI & DS
RGU MSc Big Data Analytics

IIT Alumna 2004 MMU
11+ years Industry
8+ Years Academic

Wesley College
Rotary Club – Colombo Mid Town

The Panel

Lecturers

- Cassim Farook
- Ovini Seneviratne – Visiting Lecturer

Tutors

1. Cassim Farook
2. Lakshan Costa
3. Ovini Seneviratne
4. Dilani Lunugalage
5. Suvetha Suvendran
6. Adshayan Balachandran – Visiting Lecturer

Module Schedule

Week 1 25th Sep	**Introduction to the Module. Software Engineering Life Cycle. Introduction to Software Design and Software developmental methodologies;**
Week 2 02nd Oct	Requirements Engineering 1: elicitation methods, types of requirements, formalisation (UML), use case analysis
Week 3 09th Oct	Requirements Engineering 2: domain analysis and modelling, behaviour analysis and modelling, UML diagrams – sequence, class, activity, testing requirements
Week 4 16th Oct	Software developmental methodologies – detailed look
Week 5 23rd Oct	Design and development principles. Modules, interfaces, separation of concerns, and programming patterns. Practical aspects of abstractions, invariants.
Week 6 30th Oct	Engagement Week No Lecture
Week 7 06th Nov	Software Architecture – Part 1
Week 8 13th Nov	Software Architecture – Part 2
Week 9 20th Nov	Software Sustainability
Week 10 27th Nov	Software Quality, Verification, Validation and Testing – Part 1
Week 11 04th Dec	Software Quality, Verification, Validation and Testing – Part 1
Week 12 11th Dec	Concluding Lecture – modern SE, SE and AI, links to future study

Expectations of You

Attend Lectures and Tutorials

To Learn how to “Earn” Marks

This module does not “give” marks.

“Earning” (in commerce) is the process of getting money for genuine working done.

Using Generative AI for CW earns money to the AI, - not you.

This module would be useful during placement “in parts”, but you will apply it fully if you “decide” to do the Final Year project (honours degree)

What is a Process … ?

When we provide a service or create a product we always follow a sequence of steps to accomplish a set of tasks

- You do not usually bake a cake before all the ingredients are mixed together

Any process has the following characteristics

1. Prescribes all of the major activities
2. Resources and produces intermediate and final products
3. Sub-processes and has entry and exit criteria
4. The activities are organized in a sequence
5. Constrains or control may apply to activities (budget control, availability of resources)

Four Step Baking cake process (2023) *123RF*. Available at: https://www.123rf.com/photo_77491920_baking-cake-process-demonstrates-in-4-steps-till-you-get-beautiful-creamy-butter-cake.html (Accessed: 27 September 2023).

Involves building of a product - we refer to the process as a **life cycle**

Software development process – software life cycle

Software Life Cycle

- Specifying,
- Designing,
- Implementing and
- Testing software systems

The Software Process

A structured set of activities

1. Specification
2. Design
3. Validation
4. Evolution

A software process model is an abstract representation of a process

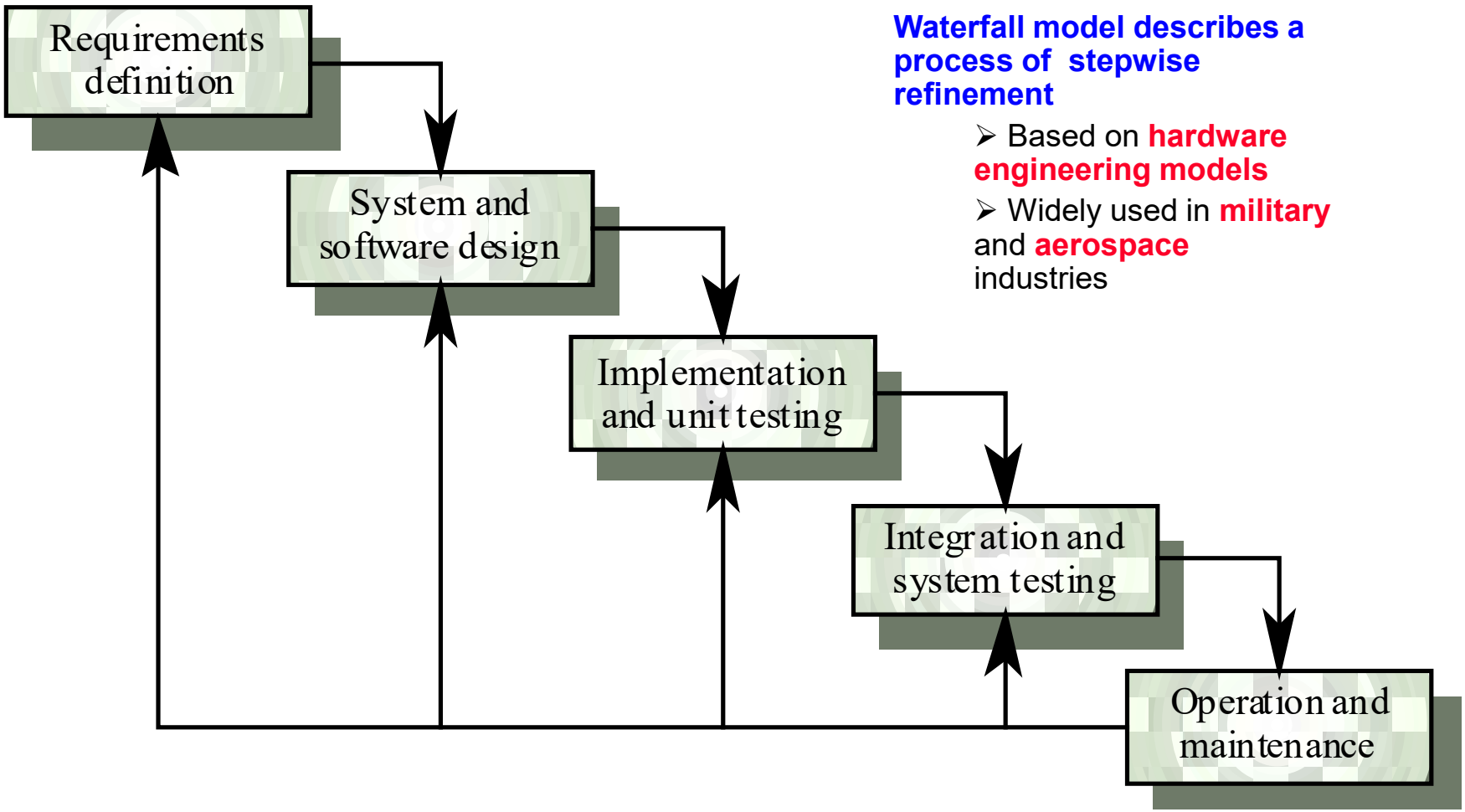
It presents a description of a process from some particular perspective

1. **Generic process models** describe the organisation of software processes
2. **Iterative process models** describe the software process as a cycle of activities

Generic (Software) Process Models

- **1. The waterfall model**
  - Separate and distinct phases of specification and development
- **2. Evolutionary development**
  - Specification and development are interleaved
- **3. Formal systems development**
  - A mathematical system model is formally transformed to an implementation
- **4. Reuse-based development**
  - The system is assembled from existing components

1. Waterfall Model



Waterfall model describes a process of stepwise refinement

- Based on **hardware engineering models**
- Widely used in **military** and **aerospace** industries

But software is different :

- **No fabrication step**
  - Program code is another design level
  - Hence, no “commit” step – software can always be changed…!
- **No body of experience for design analysis (yet)**
  - Most analysis (testing) is done on program code
  - Hence, problems not detected until late in the process
- **Waterfall model takes a static view of requirements**
  - Ignore changing needs
  - Lack of user involvement once specification is written
- **Unrealistic separation of specification from the design**
- **Doesn’t accommodate prototyping, reuse, etc**

2. Evolutionary development

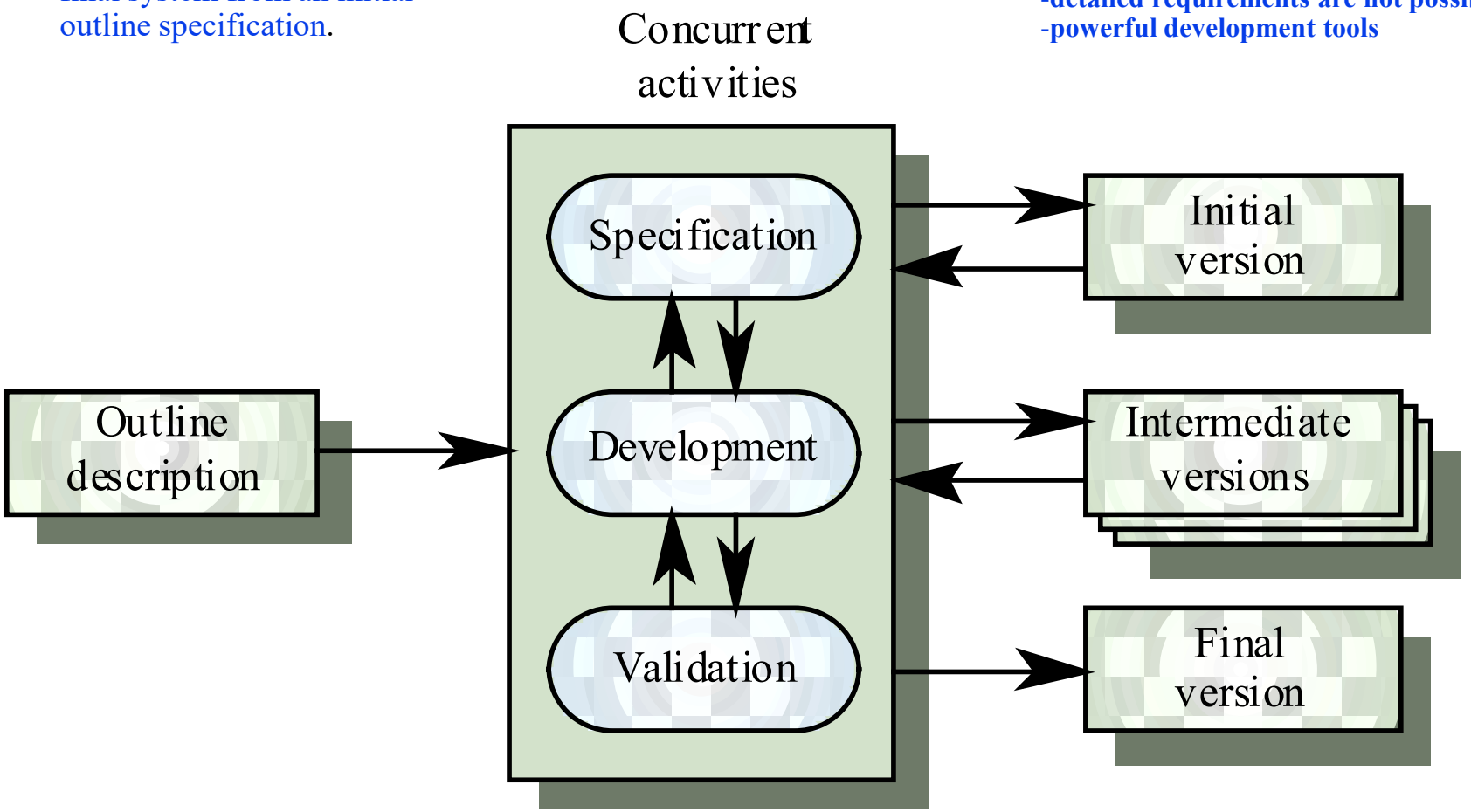
Exploratory development

-Objective is to work with customers and to evolve a final system from an initial outline specification.

Throw-away prototyping

Objective is to understand the system requirements.
We start with some rough idea of the requirements

-detailed requirements are not possible;
-powerful development tools



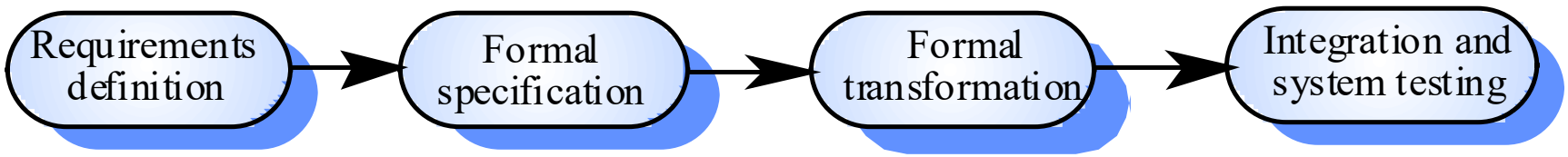
Evolutionary development -restrictions

- **Problems**
  - Lack of process visibility
  - Systems are often poorly structured
  - Special skills (e.g. in languages for rapid prototyping) may be required
- **Applicability**
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

3. Formal systems development

- **Based on the transformation of a mathematical specification**
- **Transformations are 'correctness-preserving'**

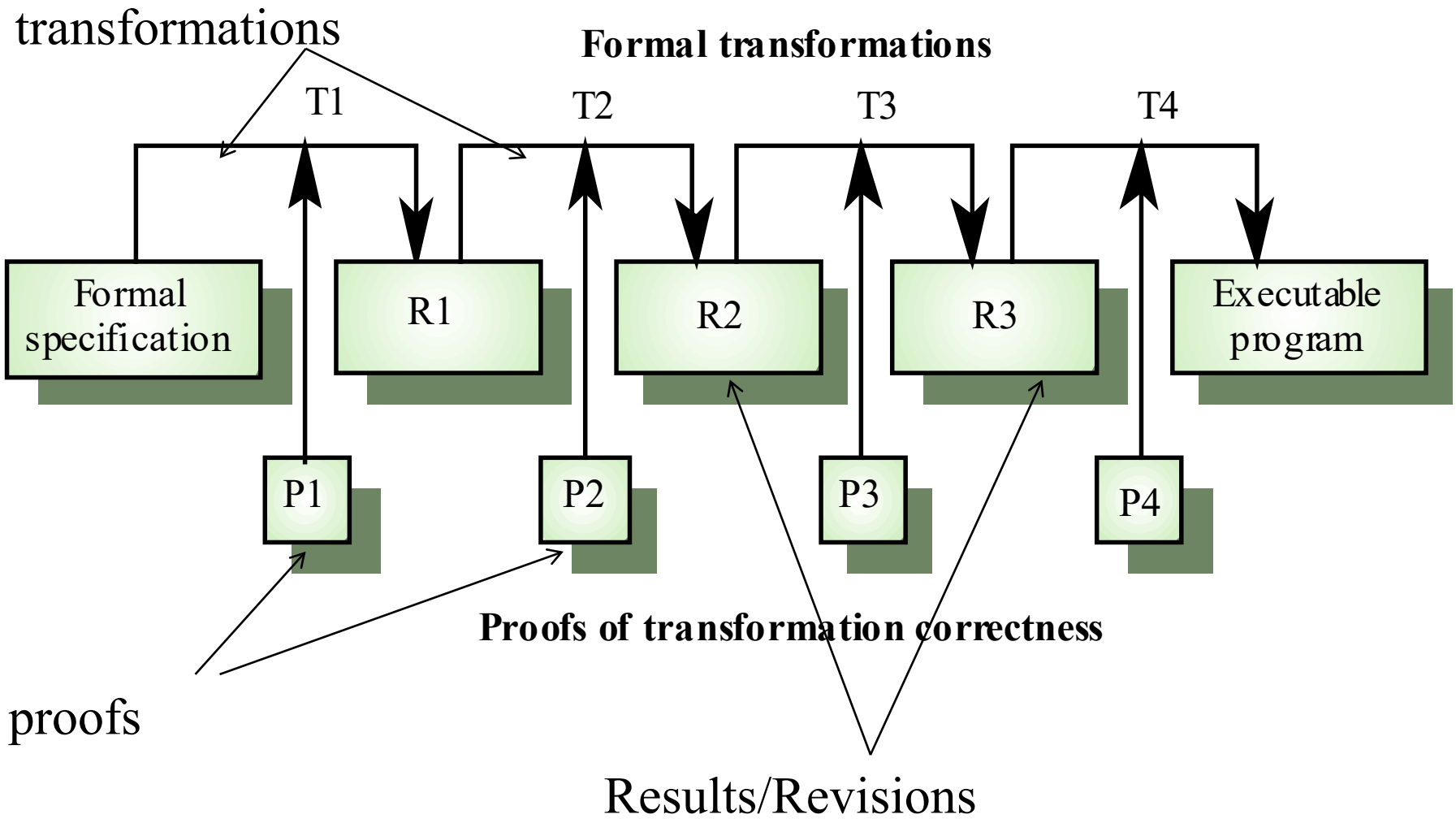
Embodied in the 'Cleanroom' approach *(which was originally developed by IBM)* **to software development**



Applicability

Critical systems especially those where a safety or security case must be made before the system is put into operation

Formal transformations



Formal transformations

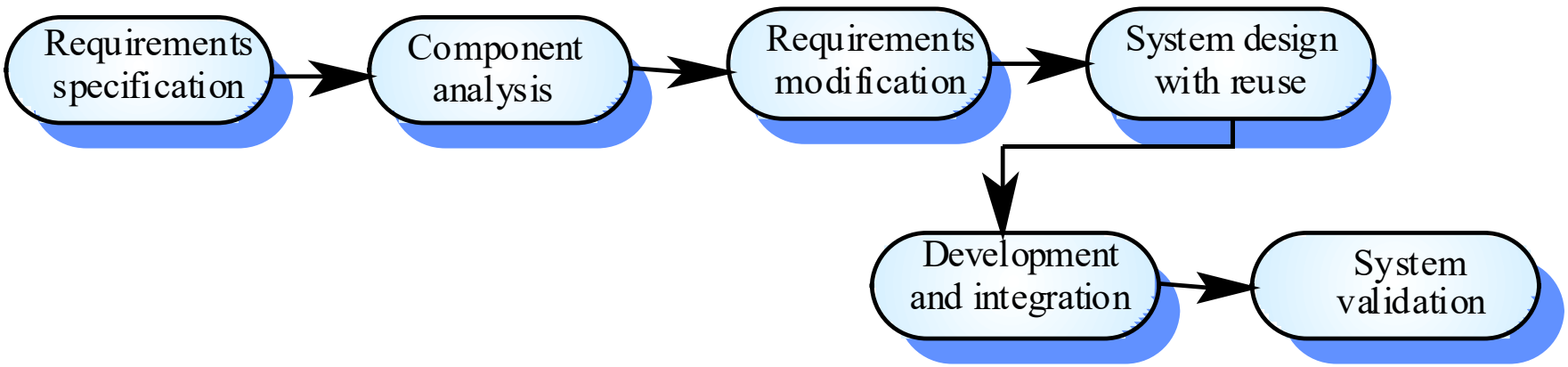
- What do we prove:
  - Program Terminates (if needed)
  - No Deadlock
  - Something Good will happen
  - Something Bad will never happen
  - Prove that the desired invariants of the program are preserved

  - Note: For SE students , Concurrent Programming in Level 6 will use the above properties to build safe concurrent programs.

4. Reuse-oriented development

- **Based on systematic reuse** where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems
- **Process stages**
  - Component analysis
  - Requirements modification
  - System design with reuse
  - Development and integration

This approach is becoming more important

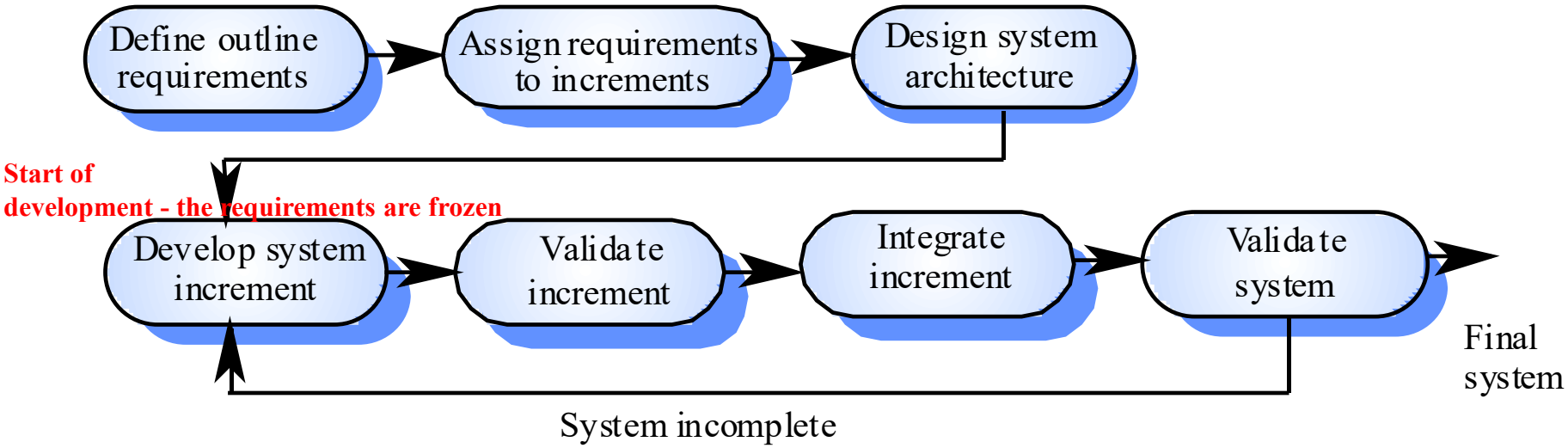


Iterative Process Models (aka Process Iteration)

- **System requirements ALWAYS evolve**
- **Iteration** can be applied to any of the generic process models
- **Two (related) approaches**

1. Incremental development
2. Spiral development

Process Iteration - Incremental development



Customer value

- can be delivered with each increment
- system functionality is available earlier

Early increments act as a prototype to help elicit requirements for later increments

Lower risk of overall project failure

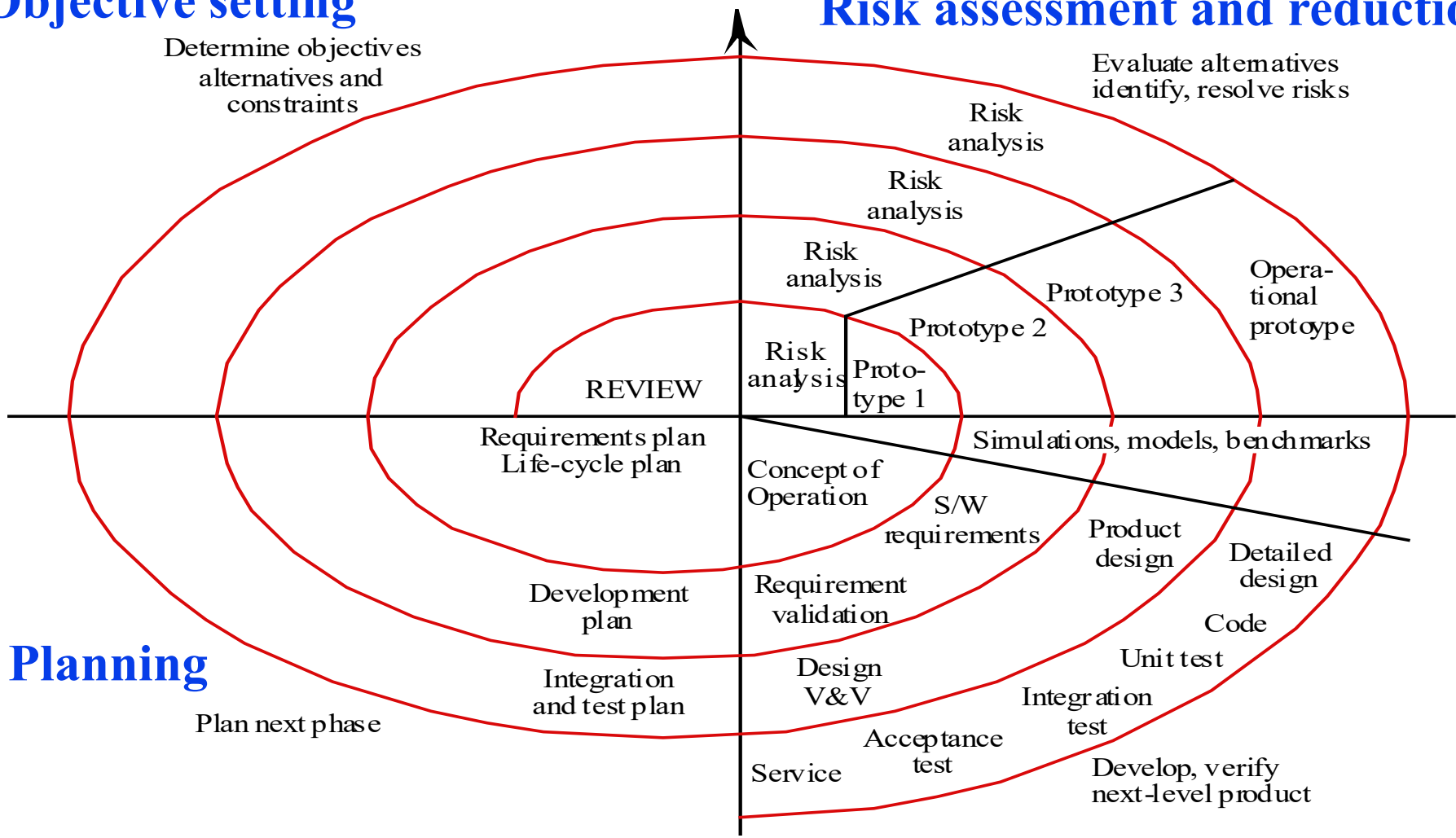
The highest priority system services

Process Iteration - Extreme programming

- **New approach** to development based on the development and delivery of **very small increments** of functionality
- **Relies on constant code improvement**, user involvement in the development team and pairwise programming
- **Design** of the test suits first !
  Then you perform **testing** of the system after each small increment

Process Iteration - Spiral development model

Process is represented as a spiral.
Each loop in the spiral represents a phase
No fixed phases - depending on what is required.
Risks are explicitly assessed and resolved

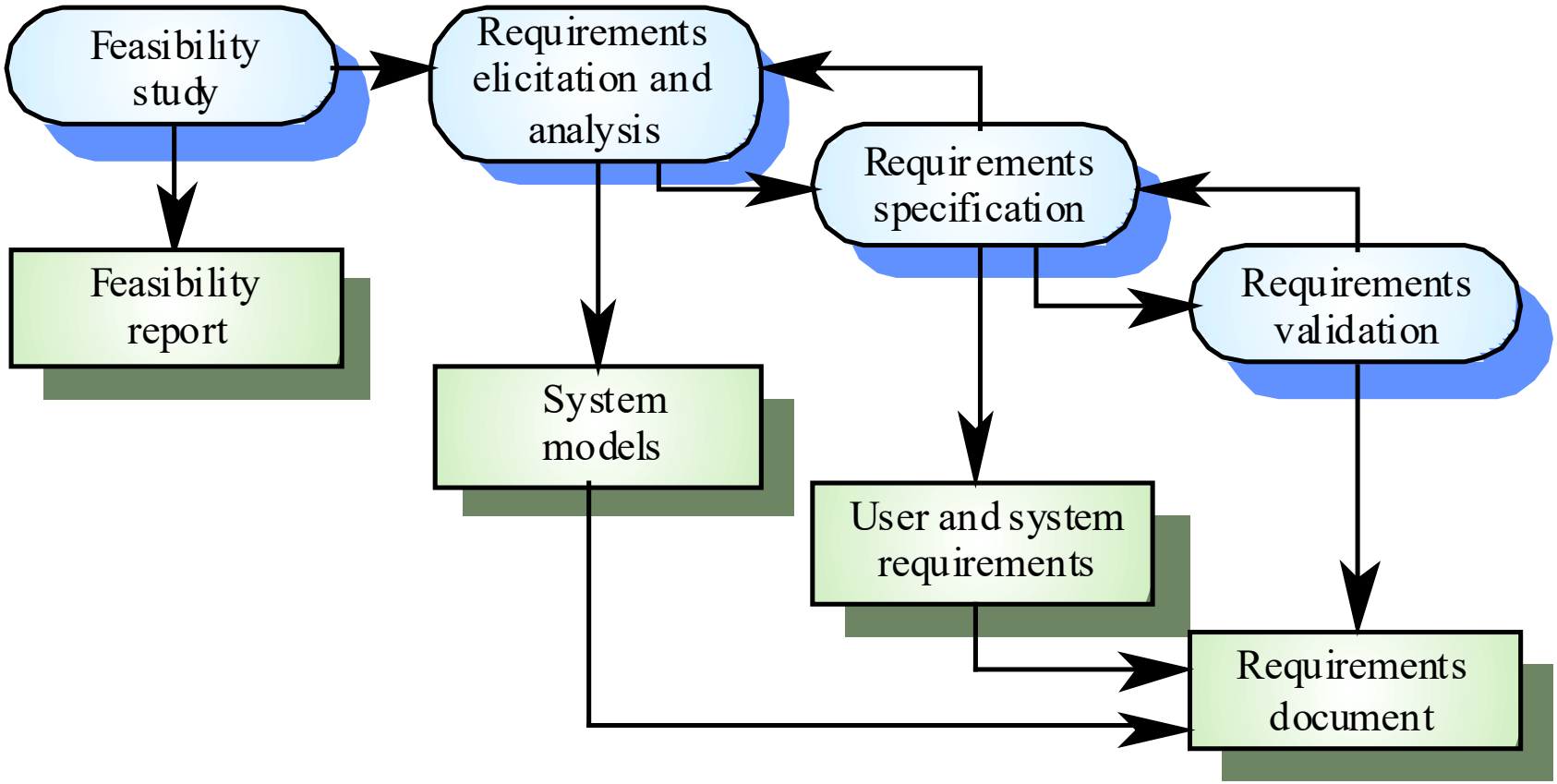


I. Software specification

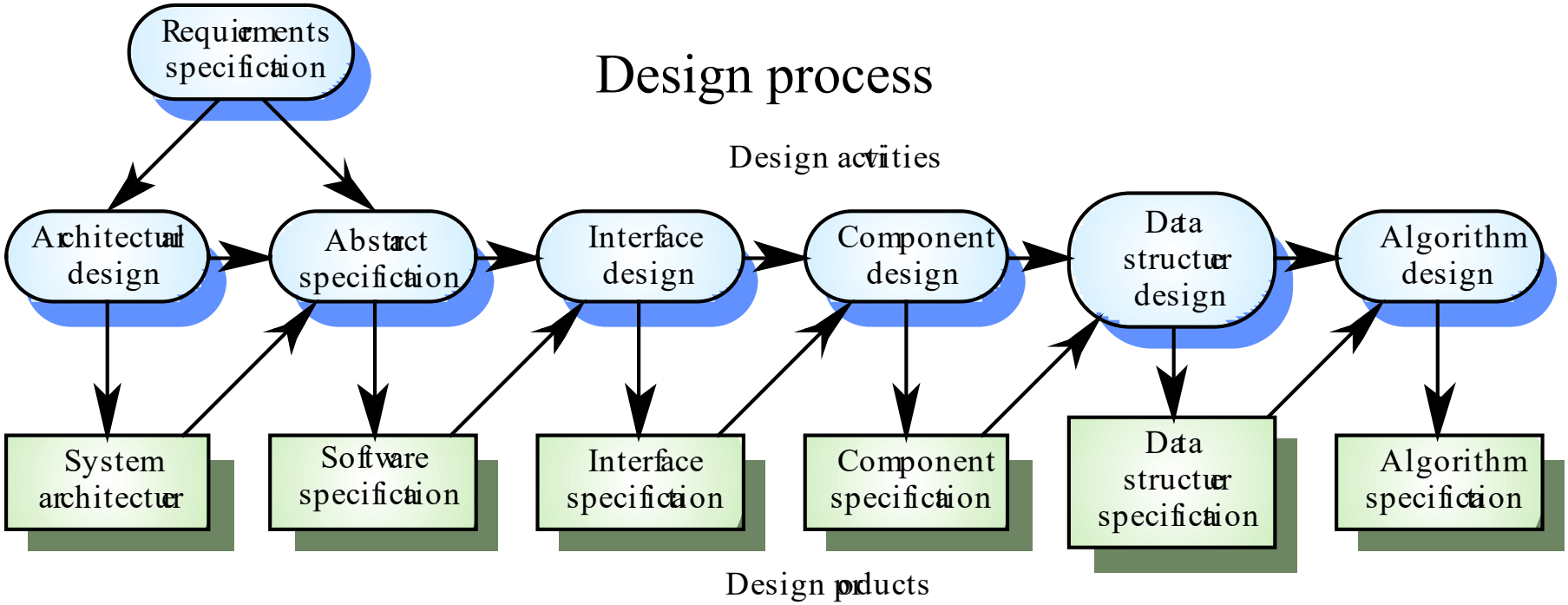
The process of establishing what services are required and the constraints on the system's operation and development

- **Requirements engineering process**
  1. Feasibility study
  2. Requirements elicitation and analysis
  3. Requirements specification
  4. Requirements validation

I. Software specification - The requirements engineering process



II. Software Design and Implementation



Systematic approaches to developing a software design

- **The design** is usually documented as a set of graphical models
- **Possible models**
  - Data-flow model
  - Entity-relation-attribute model
  - Structural model
  - Object models

Design Methods

II. Software Design and Implementation

The process of converting the system specification into an executable system

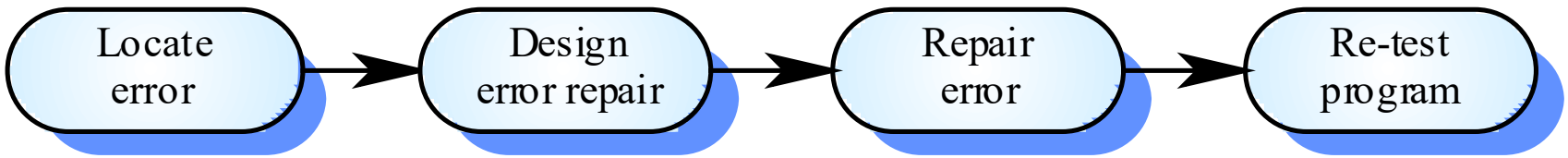
- **Software design**
  - Design a software structure that realises the specification
- **Implementation**
  - Translate this structure into an executable program

- The activities of **design** and **implementation** are closely related and **may be inter-leaved**

II. Software Design and Implementation - The Debugging Process

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



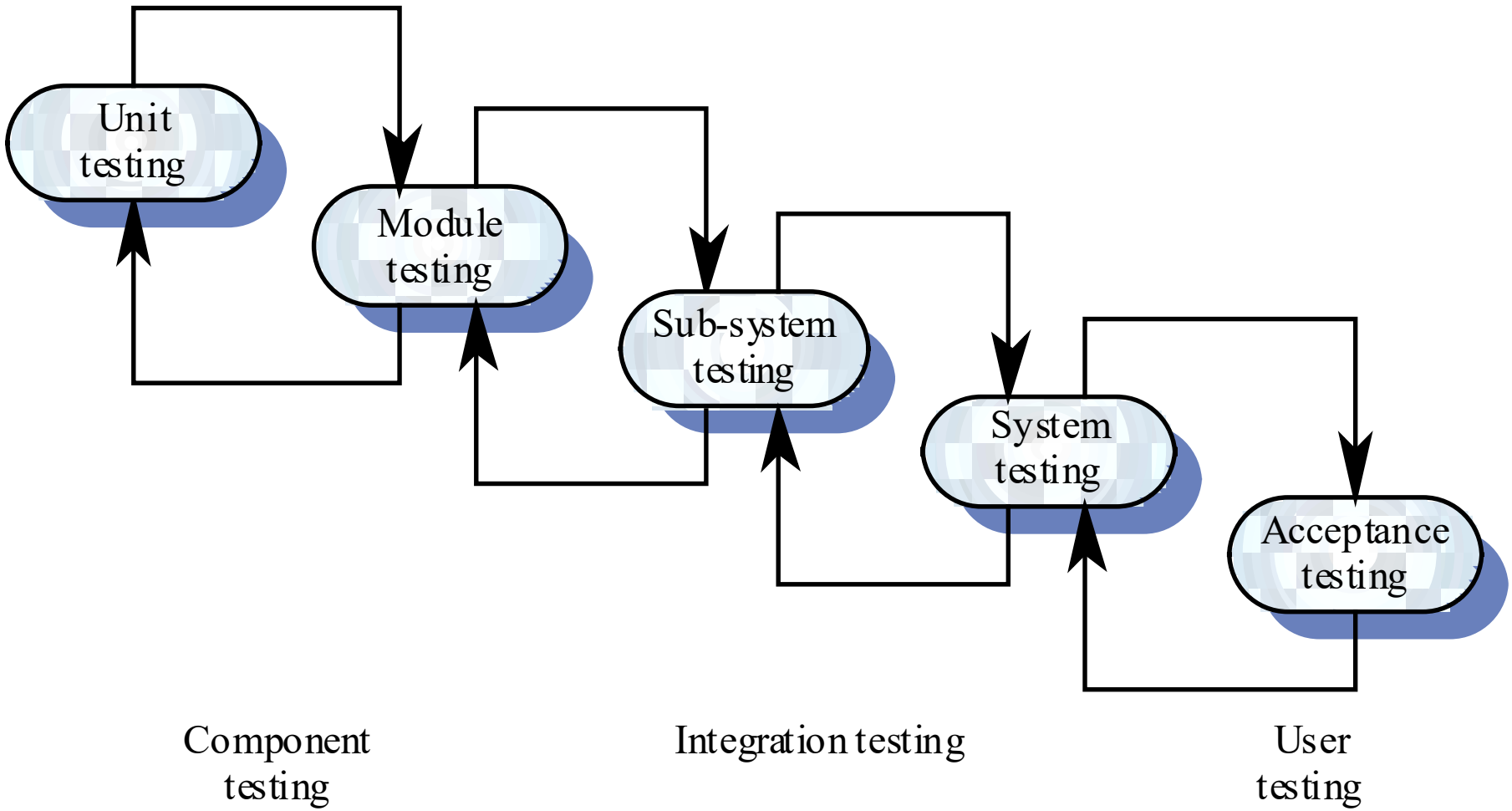
III Software validation

- **A) Verification and validation** 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**

- **B) 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

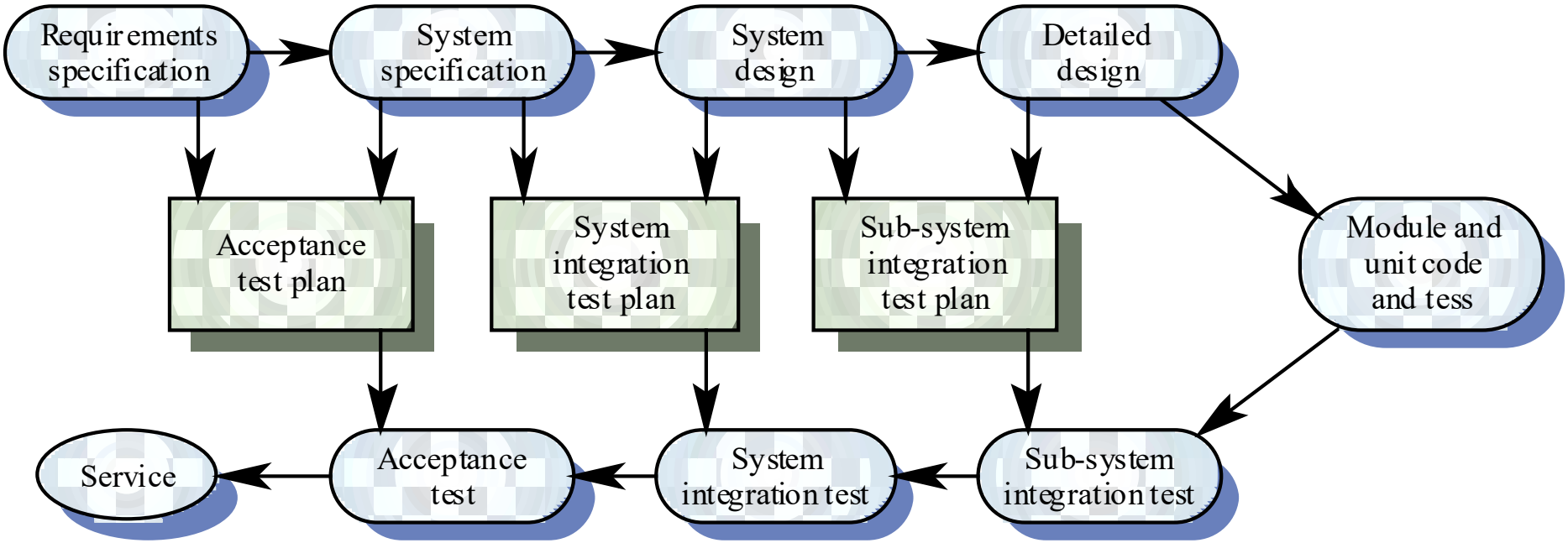
III Software validation - The Testing process



III Software validation - Testing stages

- **Unit testing**
  - Individual components are tested
- **Module testing**
  - Related collections of dependent components are tested
- **Sub-system testing**
  - Modules are integrated into sub-systems and tested. The focus here should be on interface testing
- **System testing**
  - Testing of the system as a whole. Testing of emergent properties
- **Acceptance testing**
  - Testing with customer data to check that it is acceptable

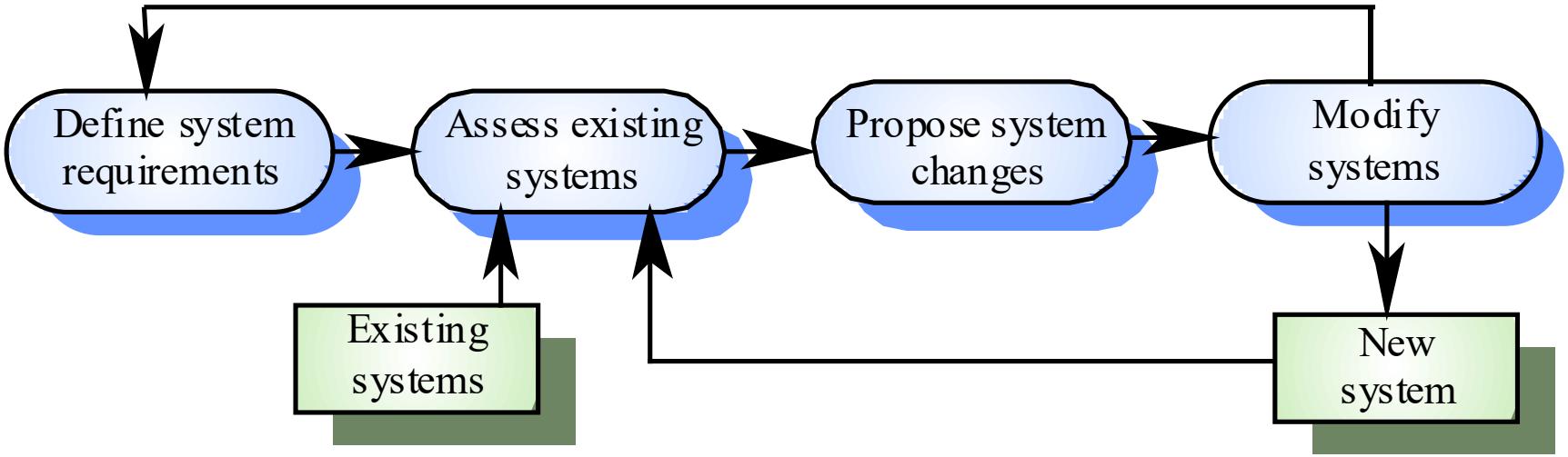
III Software validation - Testing phases



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

IV Software evolution System evolution

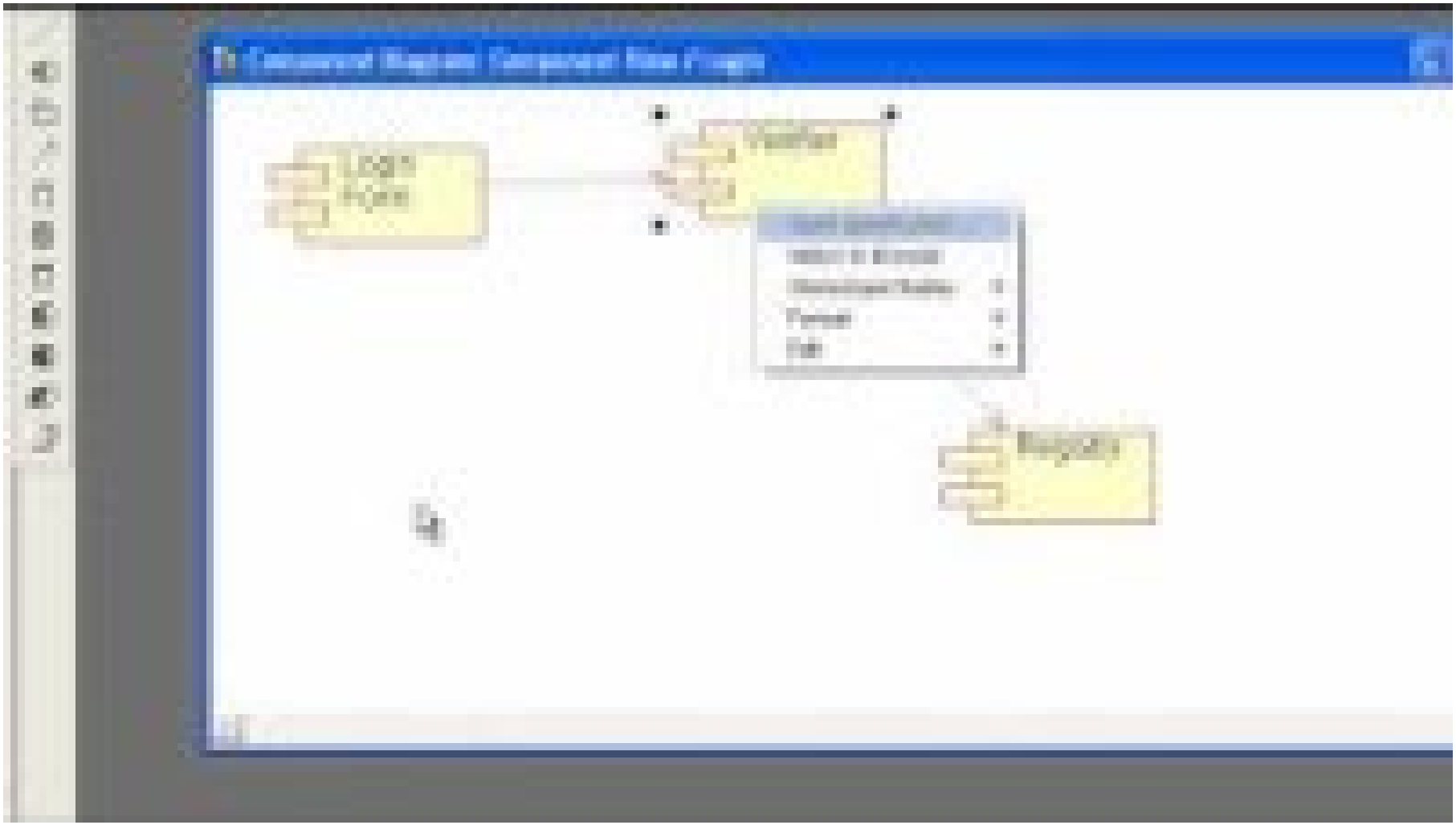


Automated process support (CASE)

- **Computer-aided software engineering (CASE)** is software to support software development and evolution processes

- **Activity automation**
  - **Graphical editors** for system model development
  - **Data dictionary** to manage design entities
  - **Graphical UI builder** for user interface construction
  - **Debuggers** to support program fault finding
  - **Automated translators** to generate new versions of a program

CASE tool Example: Rational Rose (Diagram to Code Generation)



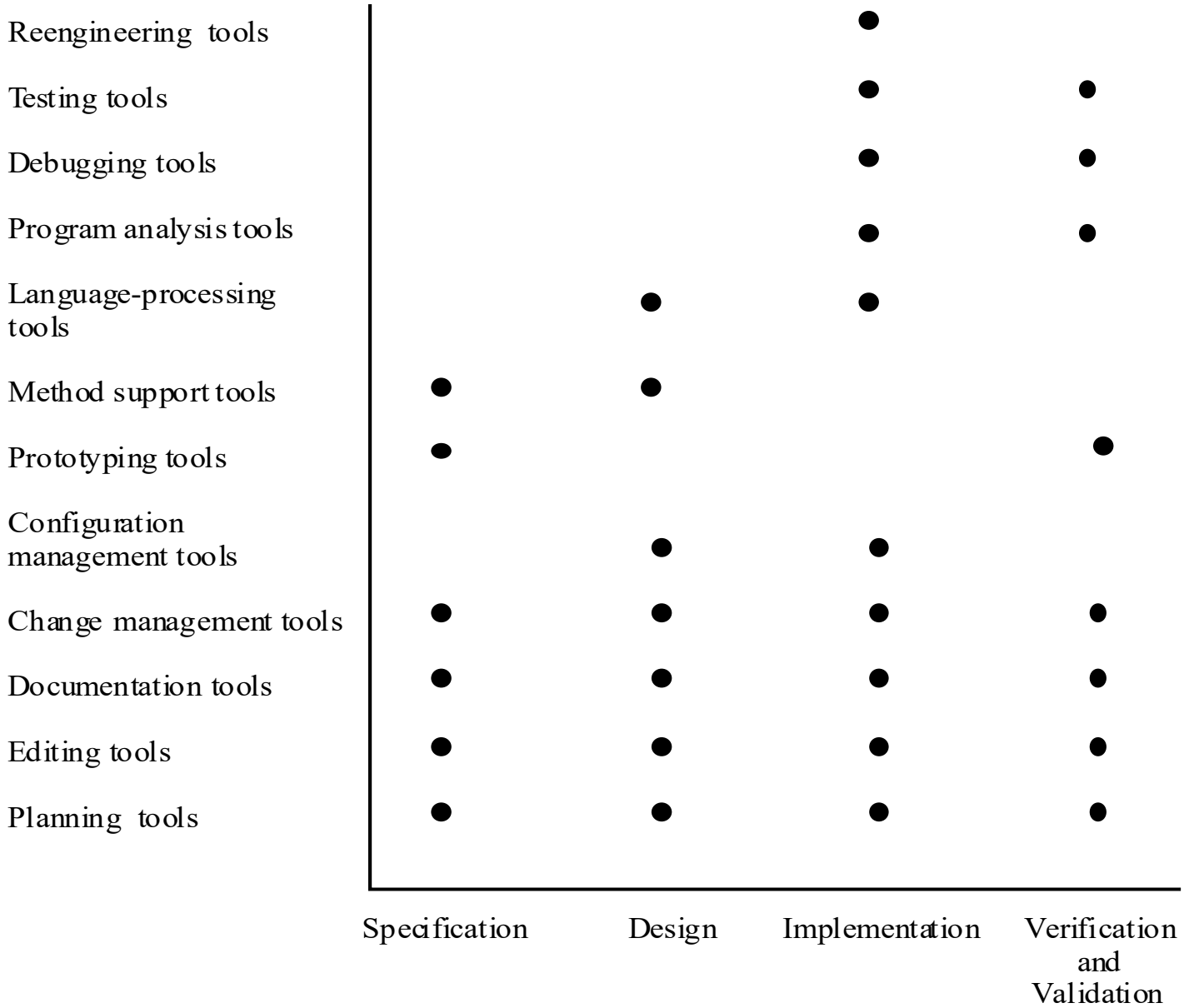
CASE classification

- **Functional perspective**
  - Tools are classified according to their specific function
- **Process perspective**
  - Tools are classified according to process activities that are supported
- **Integration perspective**
  - Tools are classified according to their organisation into integrated units

CASE classification - Functional perspective

Tool type	Examples
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

CASE classification - Process perspective

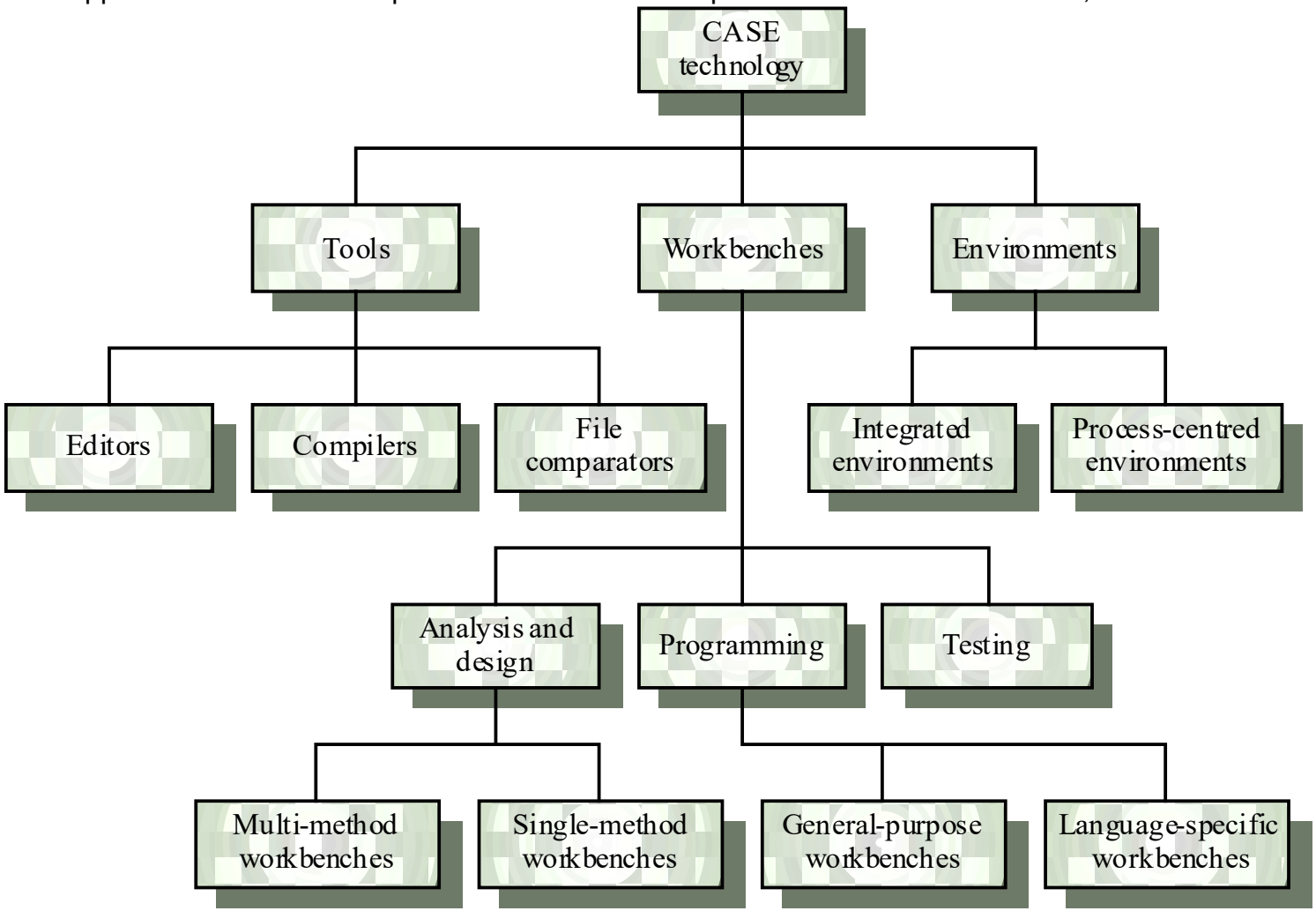


CASE classification – Integration Perspective

Tools - Support individual process tasks such as design consistency checking, text editing, etc.

Workbenches - Support a process phase such as specification or design, Normally include a number of integrated tools

Environments - Support all or a substantial part of an entire software process. Include several IDEs, Workbenches



Key points

- **Software processes** are the activities involved in producing and evolving a software system. They are represented in a software process model
- **General activities** are specification, design and implementation, validation and evolution
- **Generic process models** describe the organisation of software processes
- **Iterative process models** describe the software process as a cycle of activities

Key points

- **Requirements engineering** is the process of developing a software specification
- **Design** and **implementation** processes transform the specification to an executable program
- **Validation** involves checking that the system meets to its specification and user needs
- **Evolution** is concerned with modifying the system after it is in use
- **CASE** technology supports software process activities