MCSE 652 Software Quality Assurance

Chapter 2: Software Process

Dr. Mehedi Hasan
Stamford University Bangladesh

Outline

- Software process
- Process activities
- Software process models
- CASE tools

Software Process

Fundamental Assumption:

Good processes lead to good software

Good processes reduce risk

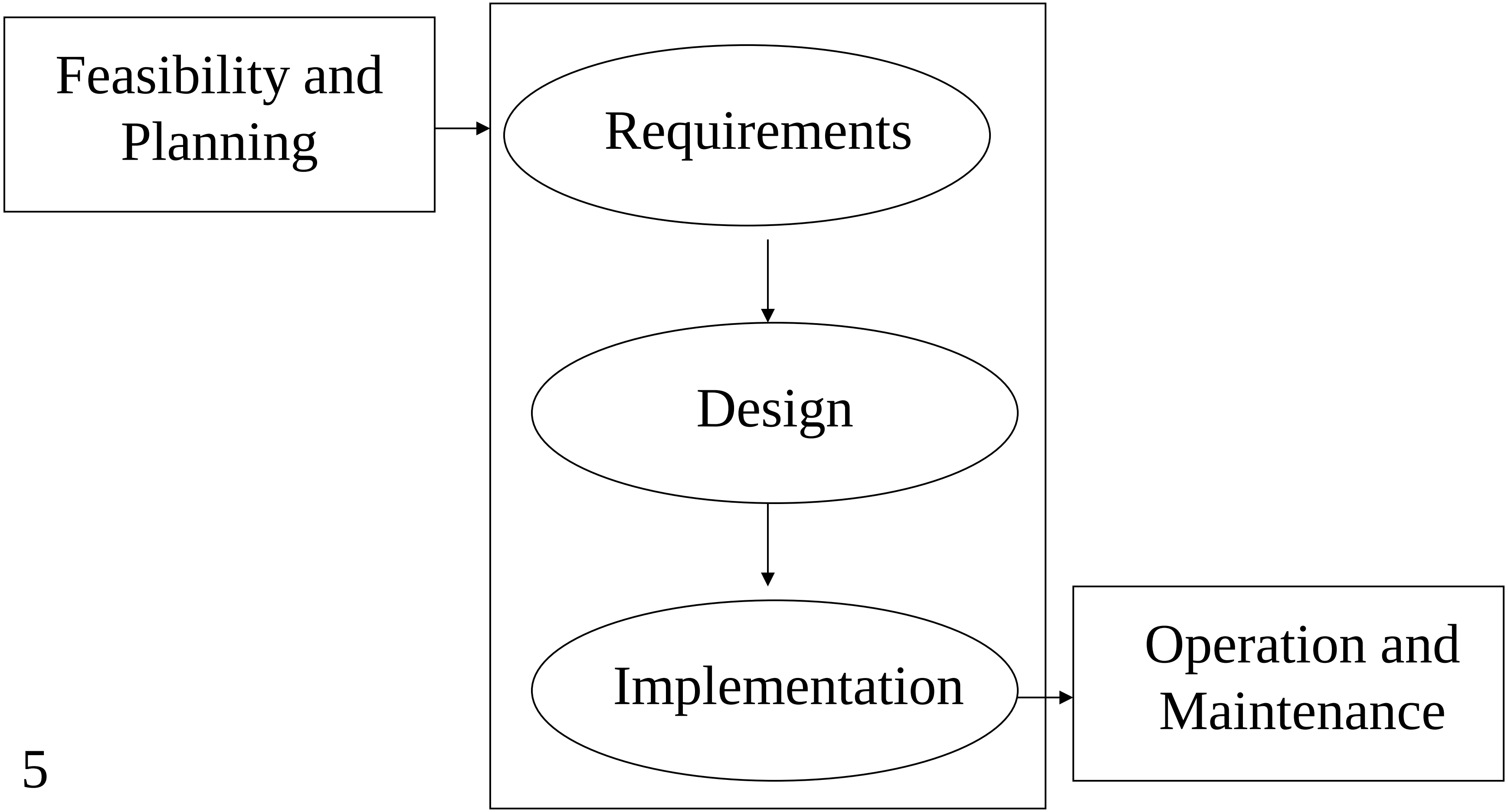
The software process

-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 (Simplified)



Plan-driven and agile processes

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

Process activities

- Real software processes are inter-leaved sequences of **technical, collaborative and managerial activities** with the overall goal of specifying, designing, implementing and testing a software system.

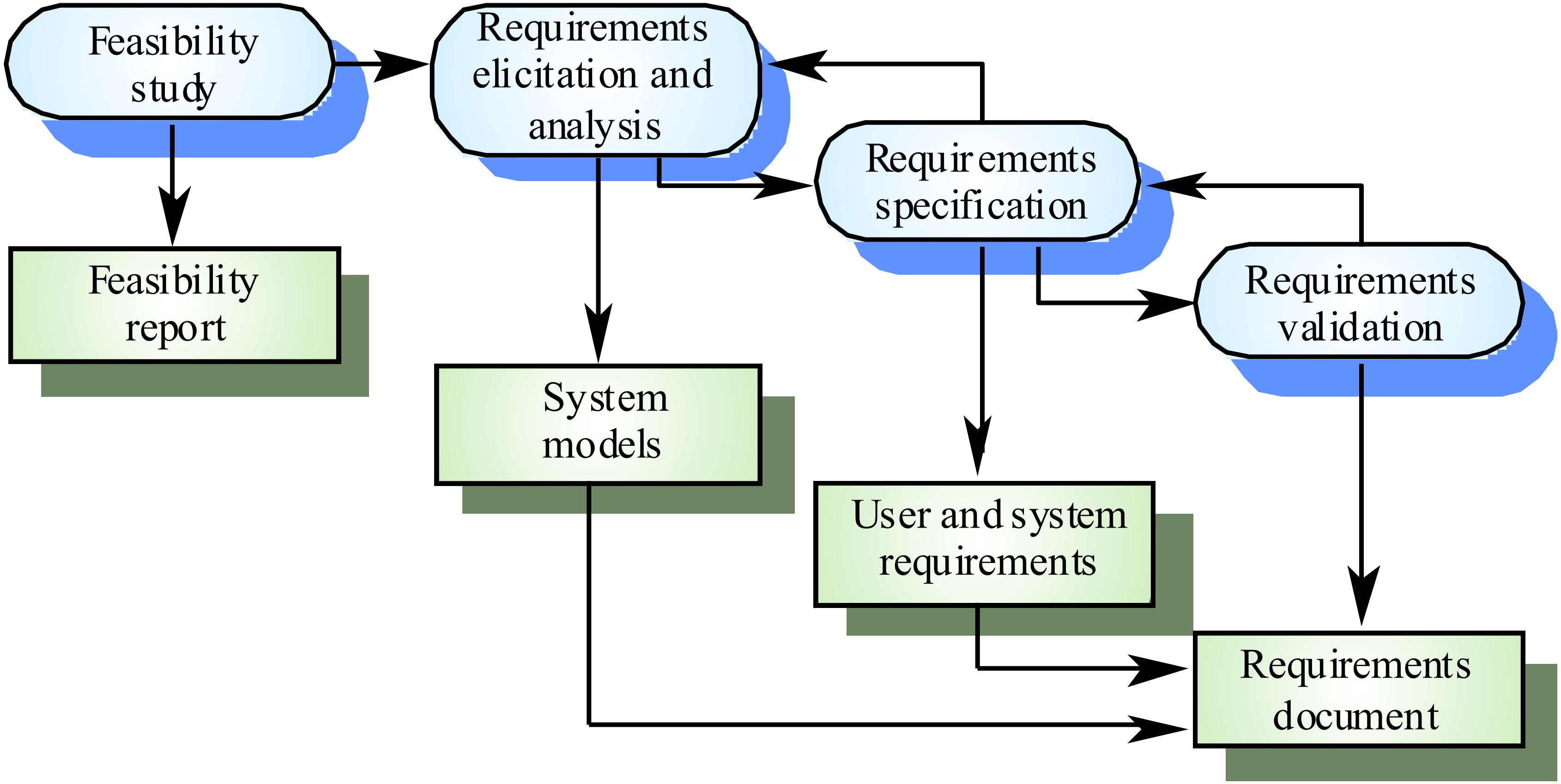
- The four basic process activities of **specification, development, validation and evolution** are organized differently in different development processes.
  - In **the waterfall model, they are organized in sequence**, whereas in **incremental development they are inter-leaved.**

I. 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 gathering 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



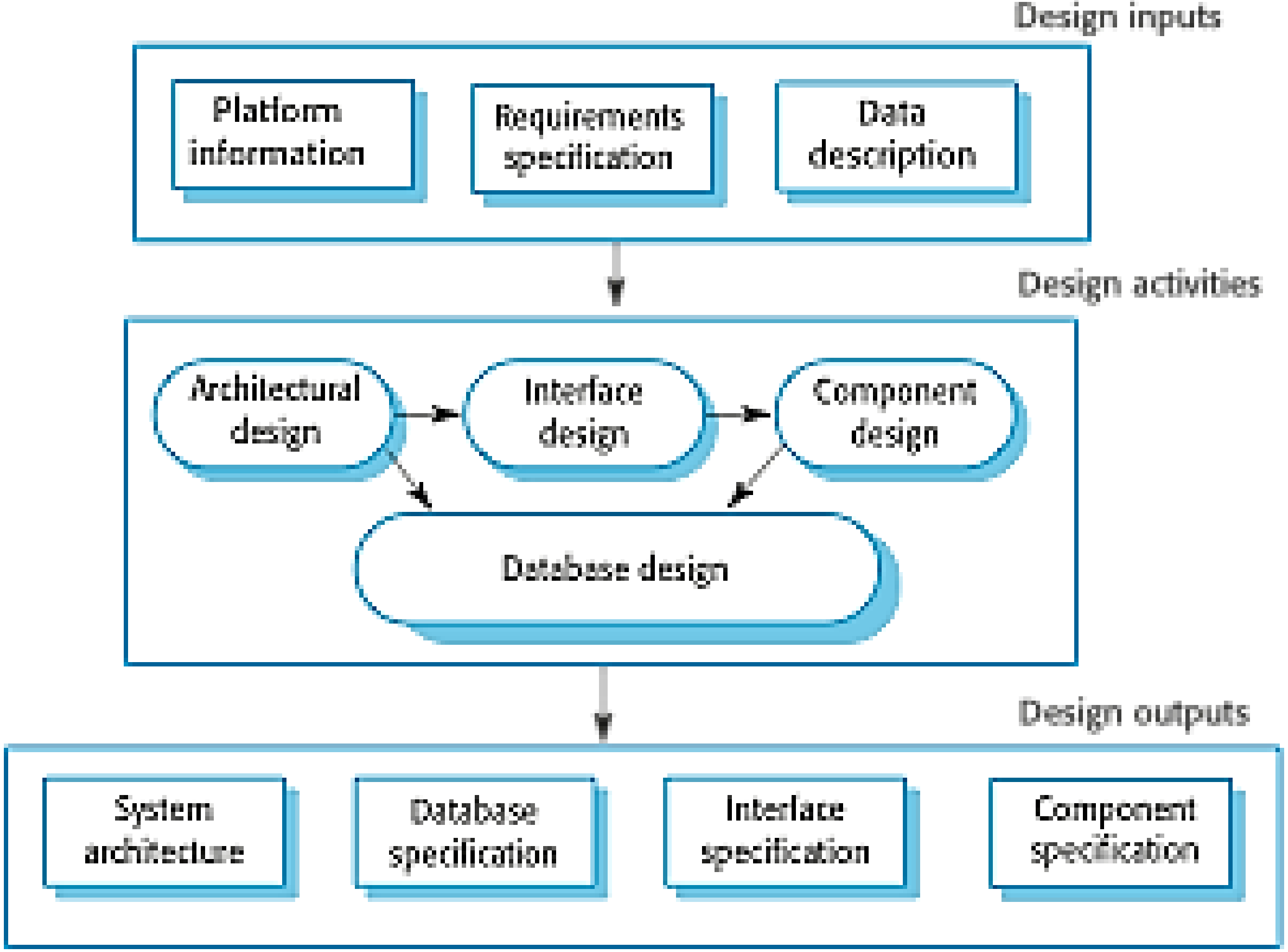
II. Software design and implementation

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**

A general model of the design process



Design 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.

Design methods

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
  - Structured methods such as object-oriented designs - UML

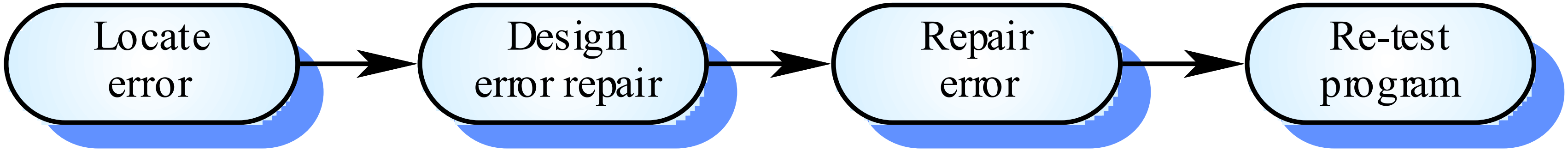
Programming and debugging

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



III Software validation

- **Verification and validation** is intended to show that a system conforms to its specification and meets the requirements of the system customer

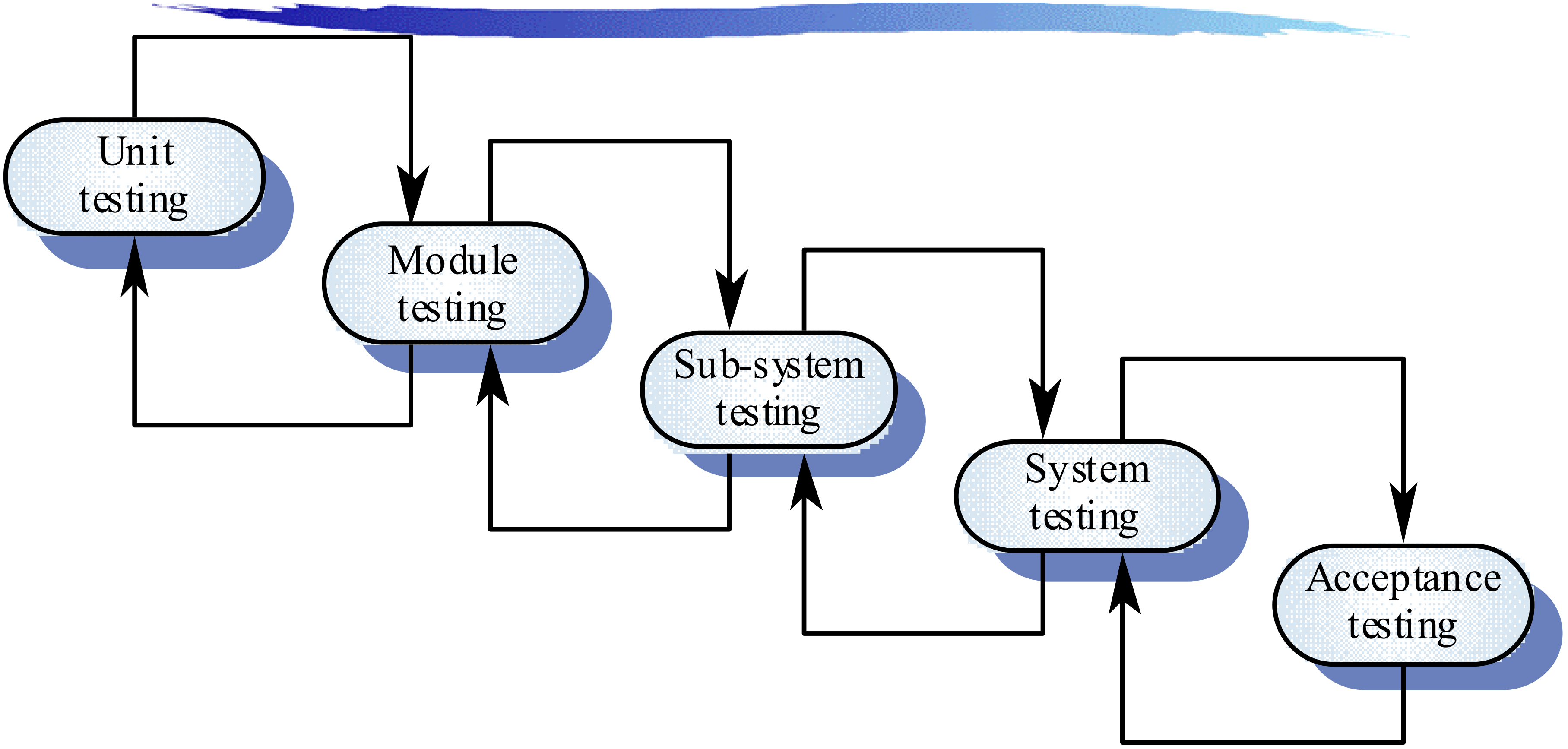
- **Verification** ensures that the right product is designed to deliver all functionality to the customer. Involves **reviews and meetings to evaluate requirements and spec.**

- **Validation** ensures that the functionalities are the intended behaviour of the product. **Involves actual testing with test cases.**

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

The testing process

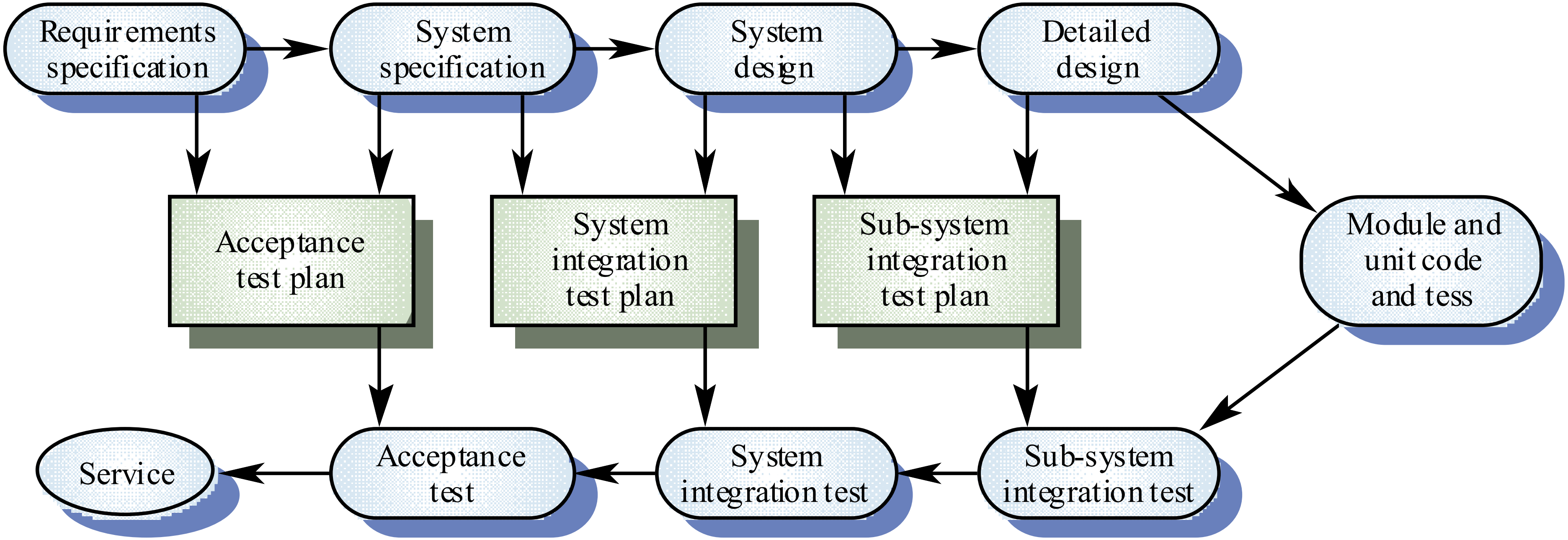


Component testing

Integration testing

User testing

Testing phases in a plan-driven software process


Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and tess
Service
Acceptance test
System integration test
Sub-system integration test

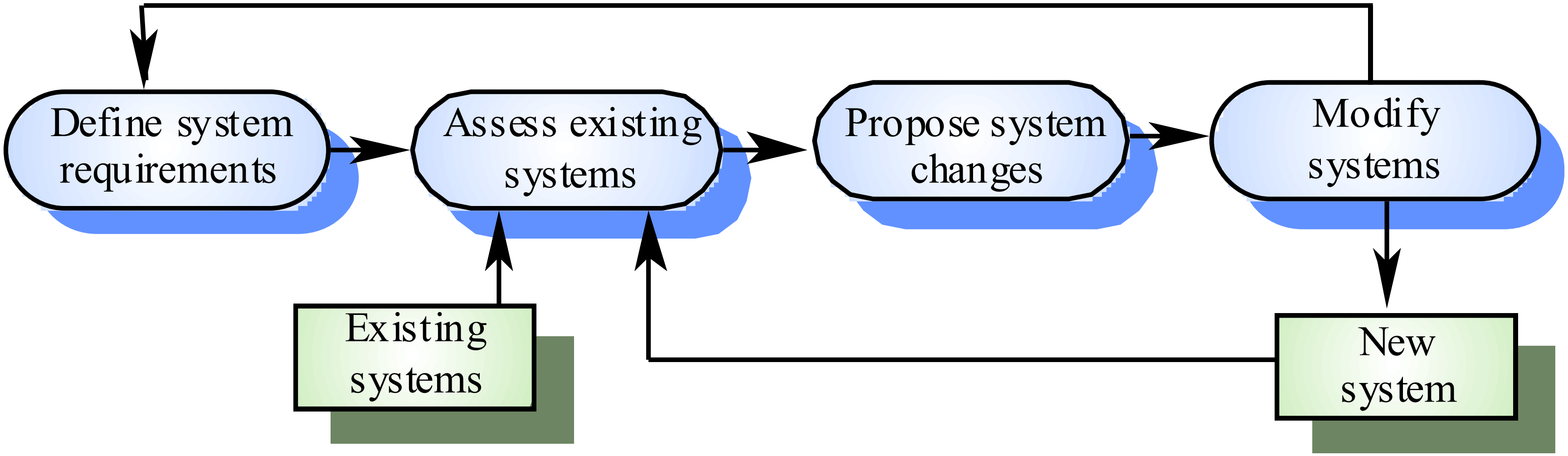
IV Software evolution

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 about software process

Software processes are the activities involved in producing a software system

- **Requirements engineering** is the process of developing a software specification.
- **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.

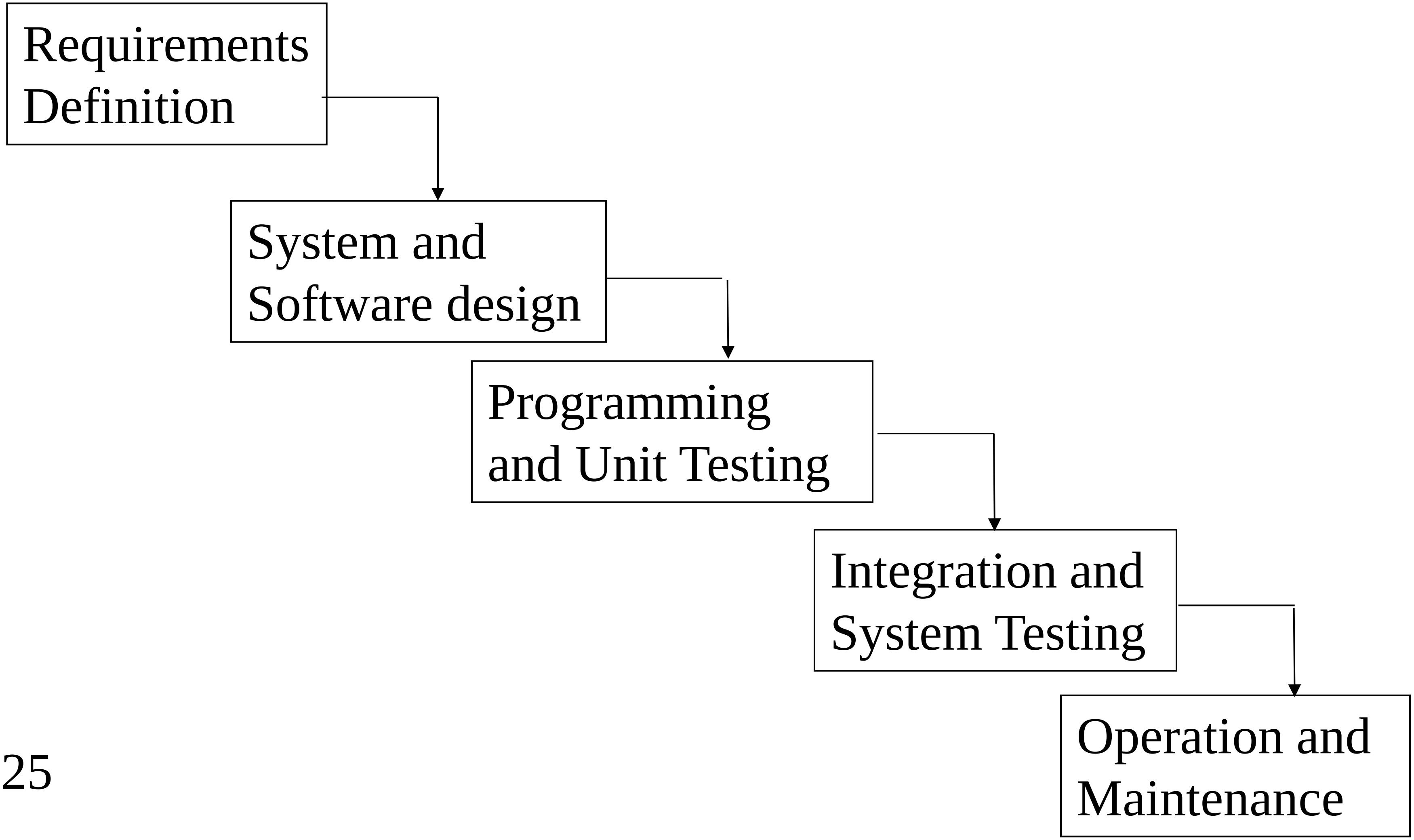
Software process models

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

Software process models

1. The waterfall model
2. Incremental Development model
   - RAD process
3. Evolutionary development
   - V-Model
   - Prototyping
   - The spiral model
4. Component based/reuse-oriented software engineering

The Waterfall Model


Requirements Definition
System and Software design
Programming 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. They are then defined in a manner that is understandable by both users and development staff.

This phase can be divided into:

- Feasibility study (often carried out separately)
- Requirements analysis
- Requirements definition
- Requirements specification

System and Software Design

System design: Partition the requirements to hardware or software systems. Establishes an overall system architecture

Software design: Represent the software system functions in a form that can be transformed into one or more executable programs

- Unified Modeling Language (UML)
- Data modeling and function modeling (ERD, DFD)

Programming and Unit Testing

The software design is realized as a set of programs or program units. (Written specifically, acquired from elsewhere, or modified.)

Individual components are tested against specifications.

Integration and System Testing

The individual program units are:

- integrated and tested as a complete system
- tested against the requirements as specified
- delivered to the client

Operation and Maintenance

- Operation: The system is put into practical use.
- Maintenance: Errors and problems are identified and fixed.
- Evolution: The system evolves over time as requirements change, to add new functions or adapt the technical environment.
- Phase out: The system is withdrawn from service.

Discussion of the Waterfall Model

Advantages:

- Process visibility
- Dependence on individuals
- Quality control
- Cost control

Disadvantages:

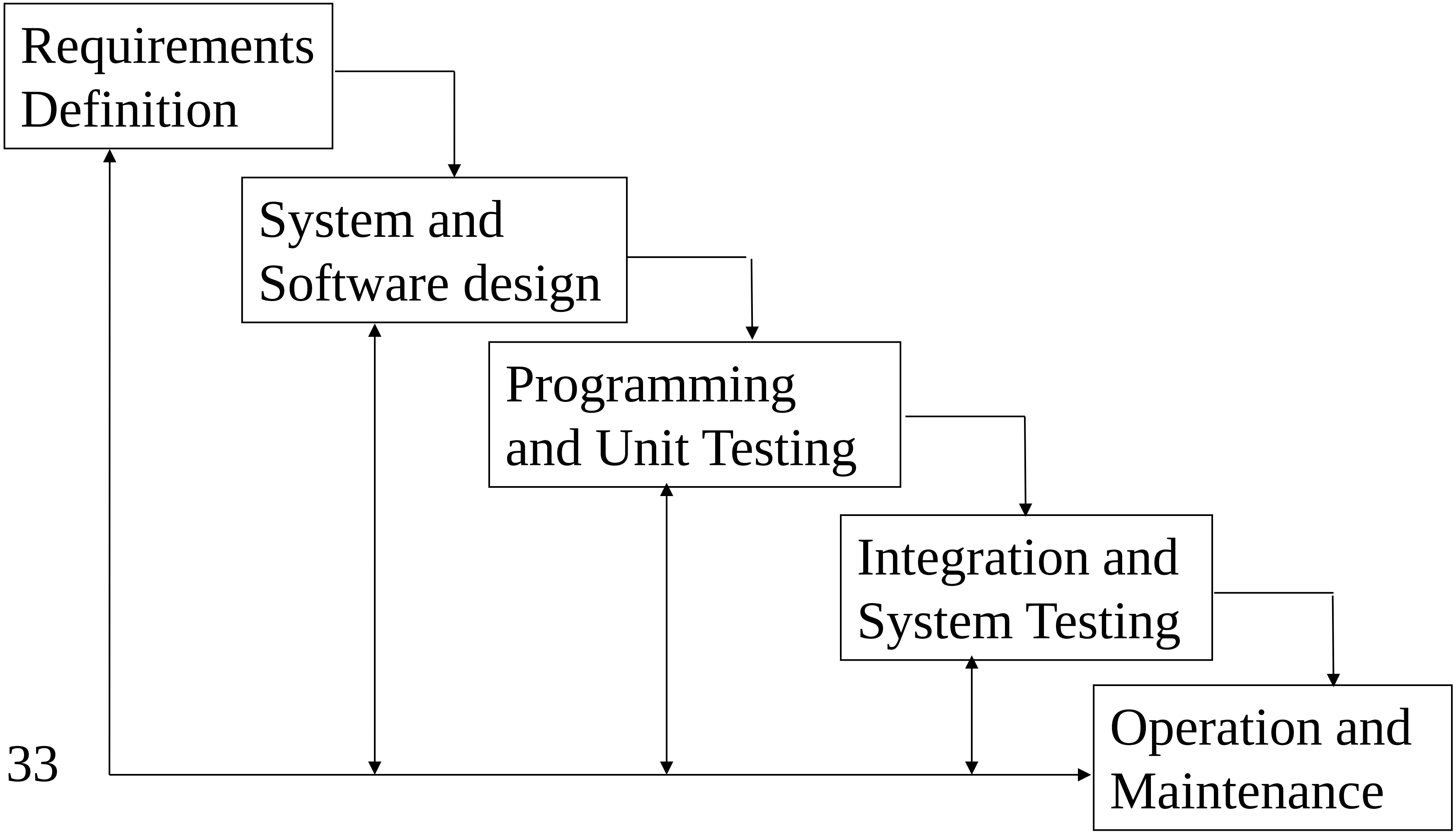
In principle, a phase has to be complete before moving onto the next phase. **Inflexible partitioning** of the project into **distinct stages** makes it **difficult to respond to changing customer requirements.**

Waterfall model

- Therefore, this model is **only appropriate when the requirements are well-understood and changes will be fairly limited** during the design process.
    - Few business systems have stable requirements.

- The waterfall model is **mostly used for large systems engineering projects** where a system is developed at several sites.
    - In those circumstances, the **plan-driven nature** of the waterfall model helps coordinate the work.

Feedback in the Waterfall Model


Requirements Definition
System and Software design
Programming and Unit Testing
Integration and System Testing
Operation and Maintenance

Incremental development

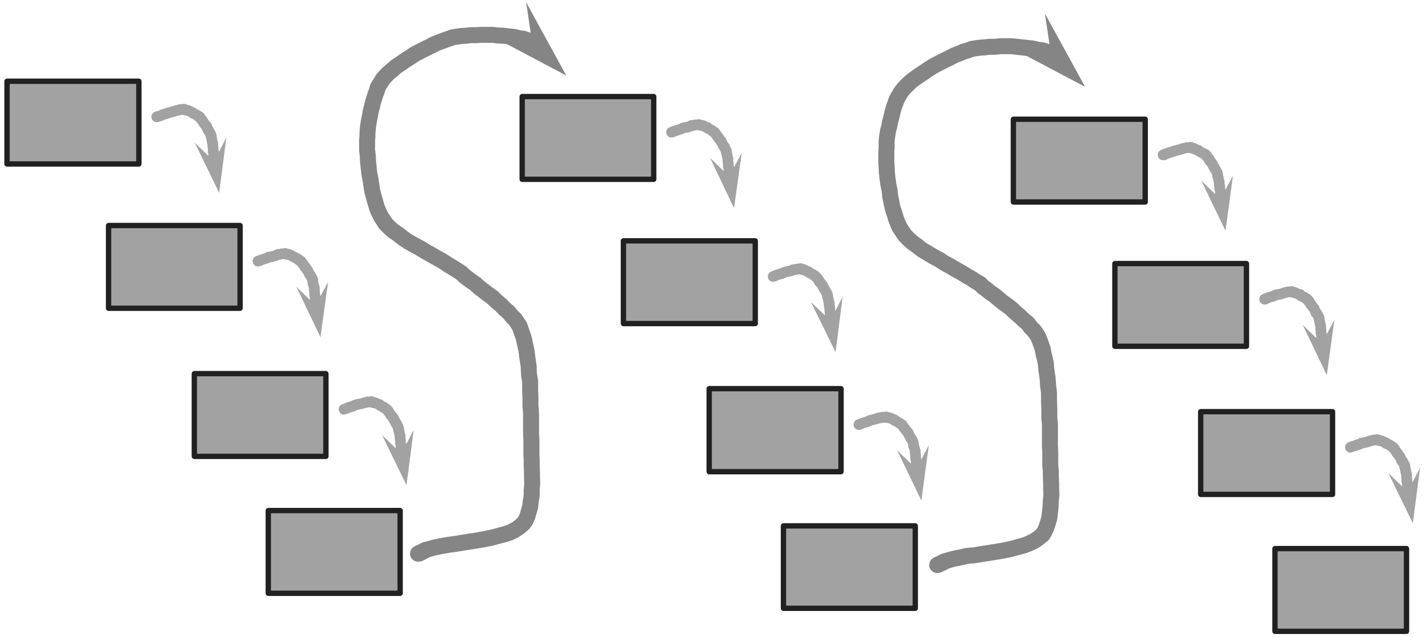
- Rather than deliver the system as a single delivery, **the development and delivery is broken down into increments** with each increment delivering part of the required functionality. -involve prototyping at each increment

- **User requirements are prioritised** and the highest priority requirements are included in early increments

- **Once the development of an increment is started, the requirements are frozen** though requirements for later increments can continue to evolve

Incremental development-Comparison

Compare waterfall model: - Each release is a mini-waterfall



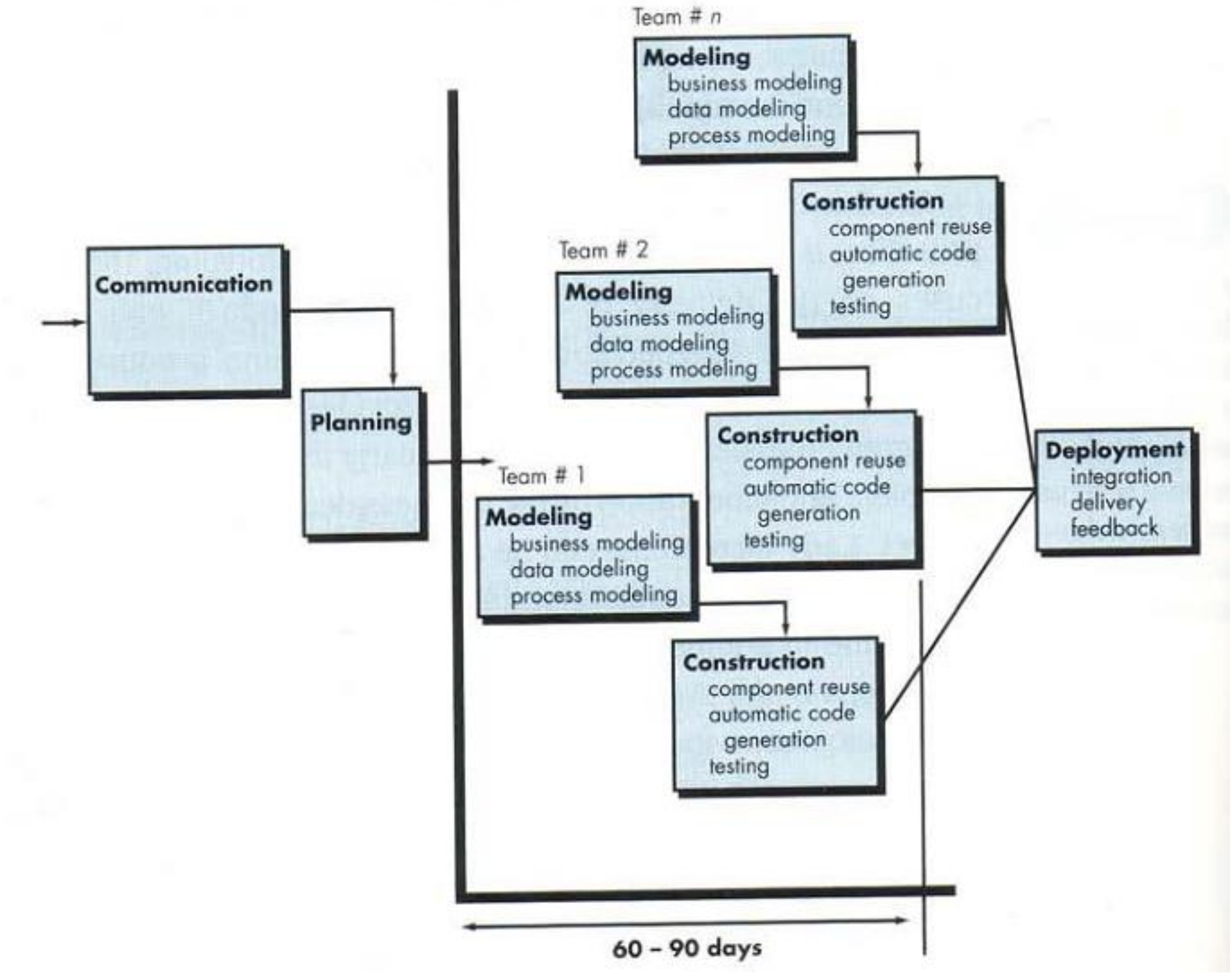
Planning of Incremental development

- 1st Increment is the core product (basic requirements are addressed) and at the end of the first increment, it is used by customers for evaluation
- A plan is developed for the next increment. The plan include the modifications in the 1st increments to better meet customer needs and the delivery of additional functionality and features in the new increment.
- The increments are continued until the product is complete and up to customers' satisfaction

An example -The Rapid Application Development

- An incremental sequential process model
- Emphasizes on short development cycle by using component-based construction approach
- High-speed adaptation of waterfall model
- Multiple software teams work in parallel on different system functions

The Rapid Application Development



Incremental development advantages

- **Customer requirements** can be delivered with each increment so system functionality is available earlier
- **Early increments** act as a prototype to help bring out requirements for later increments
- Lower risk of overall project failure
- **The highest priority system services** tend to receive the most testing
- Fewer staff in the team

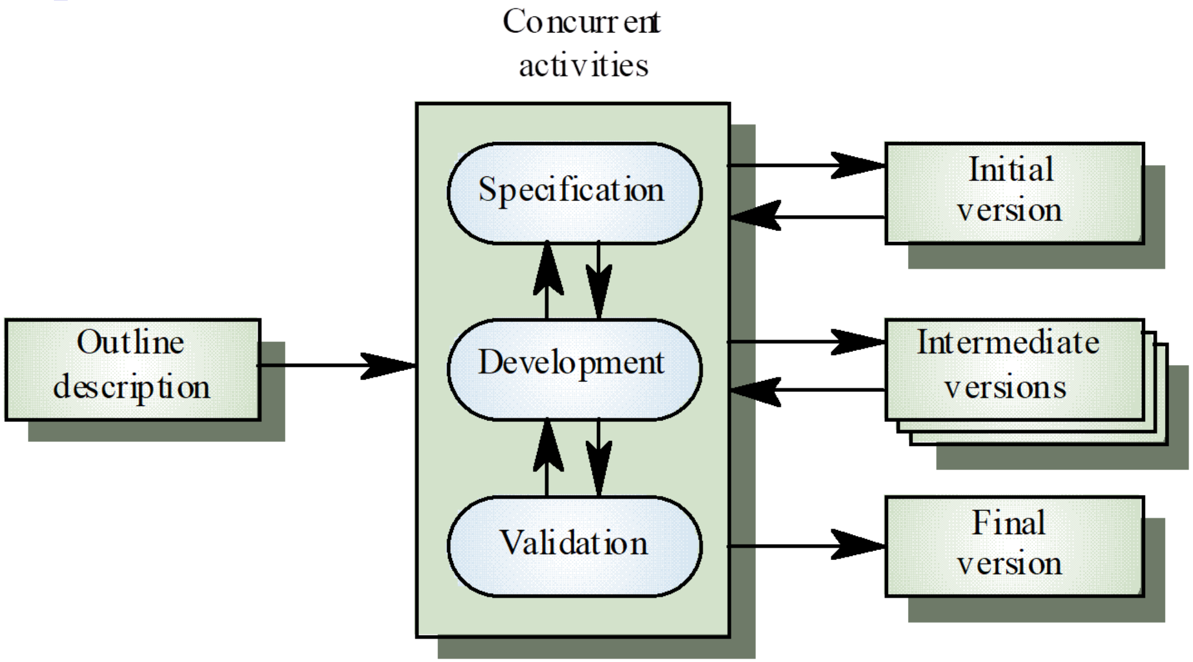
Incremental development problems

- **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.

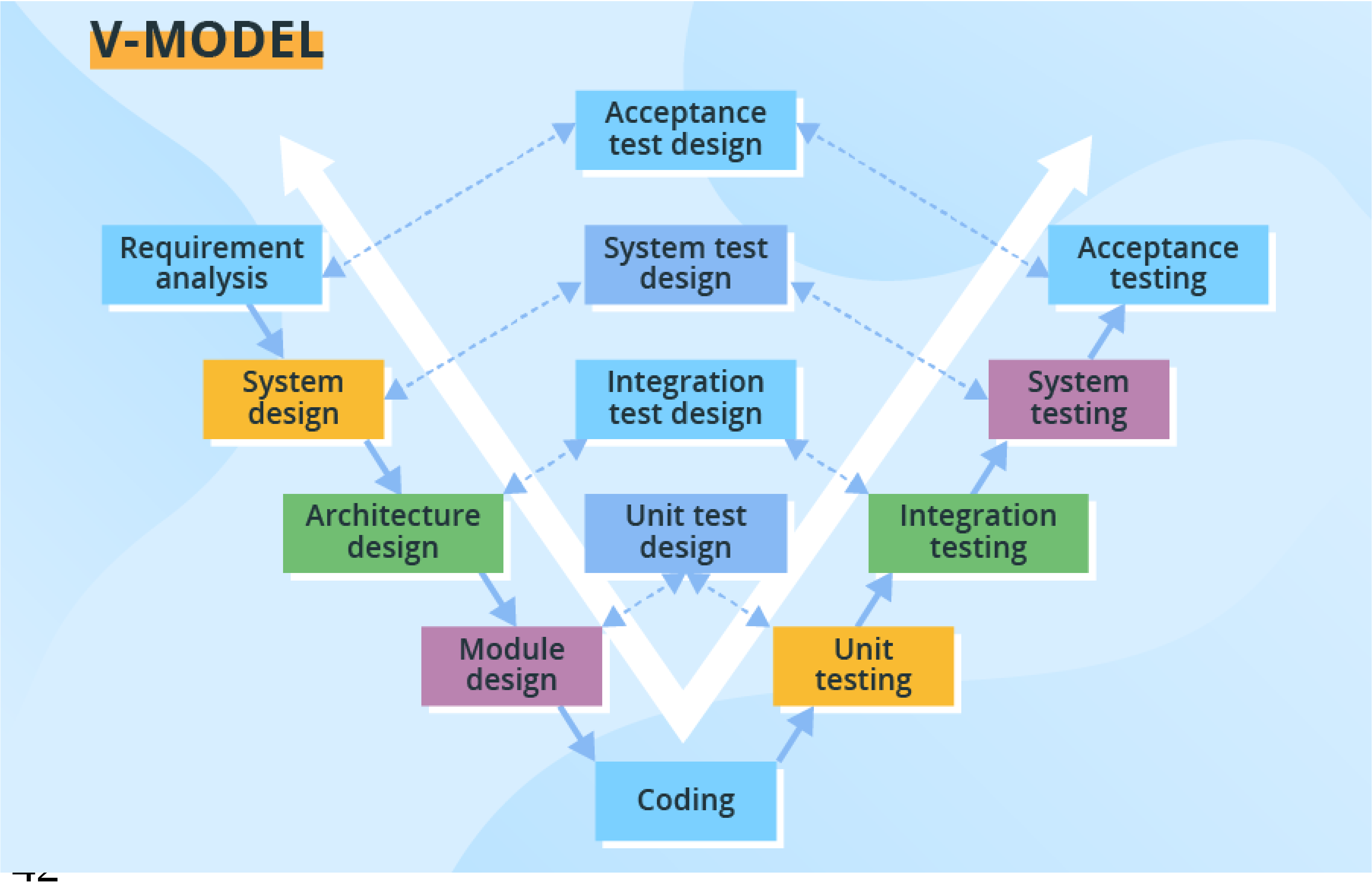
3. Evolutionary Development -iterative refinement

Concept: Initial implementation for user comment, followed by refinement until system is complete.

1. **The V-Model**
2. **Prototyping**
3. **The Spiral model**



3 i) V-Model


V-MODEL
Acceptance test design
Requirement analysis
System test design
Acceptance testing
System design
Integration test design
System testing
Architecture design
Unit test design
Integration testing
Module design
Unit testing
Coding

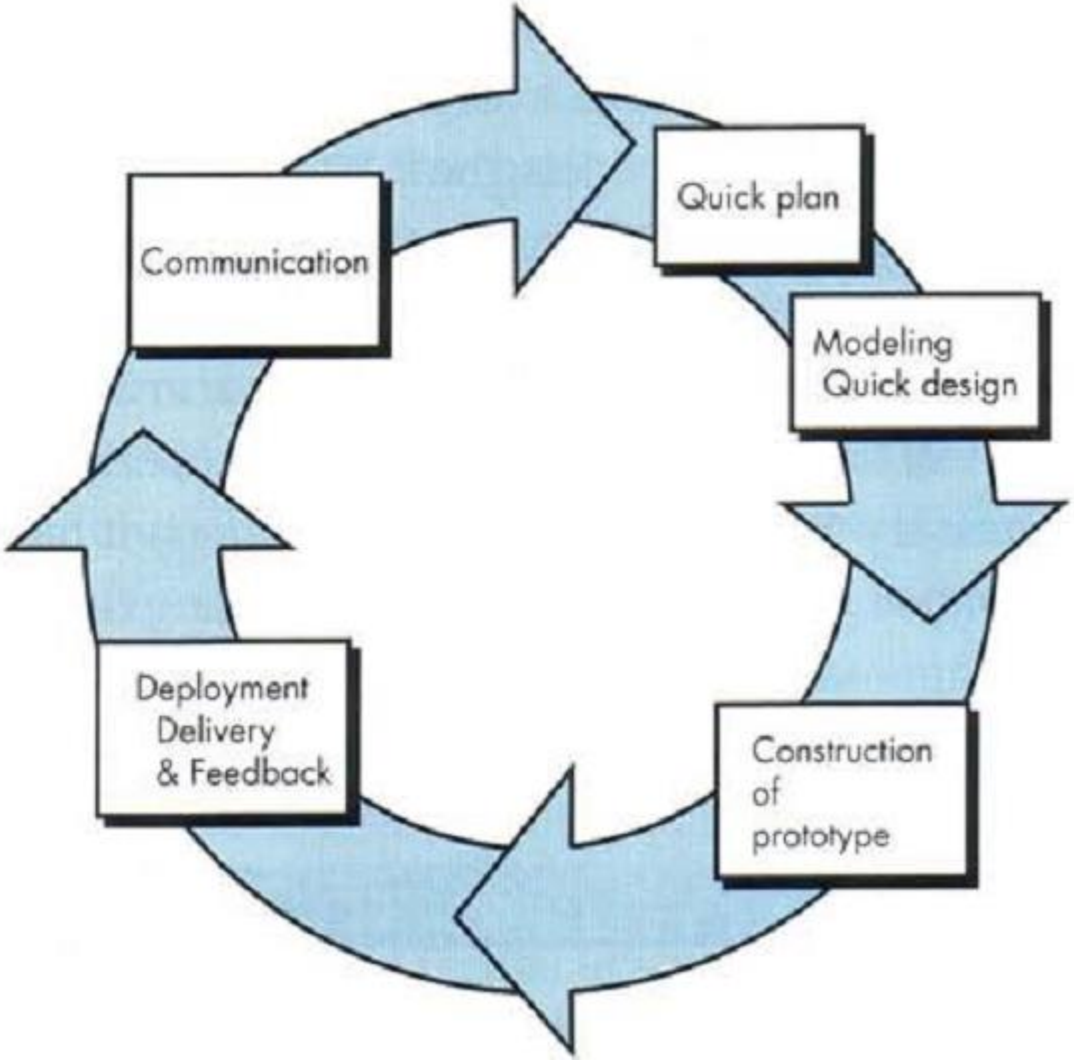
3 ii) Prototyping

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

- Should start with **a set of general objectives** for software
- Develop “quick and dirty” system quickly
- The system evolves by adding new features as they are proposed by customer until adequate system is developed

- Particularly suitable where: **detailed requirements is not possible;**
- **Assists the software engineers and the customers to better understand what is to be built**

3 ii) Prototyping



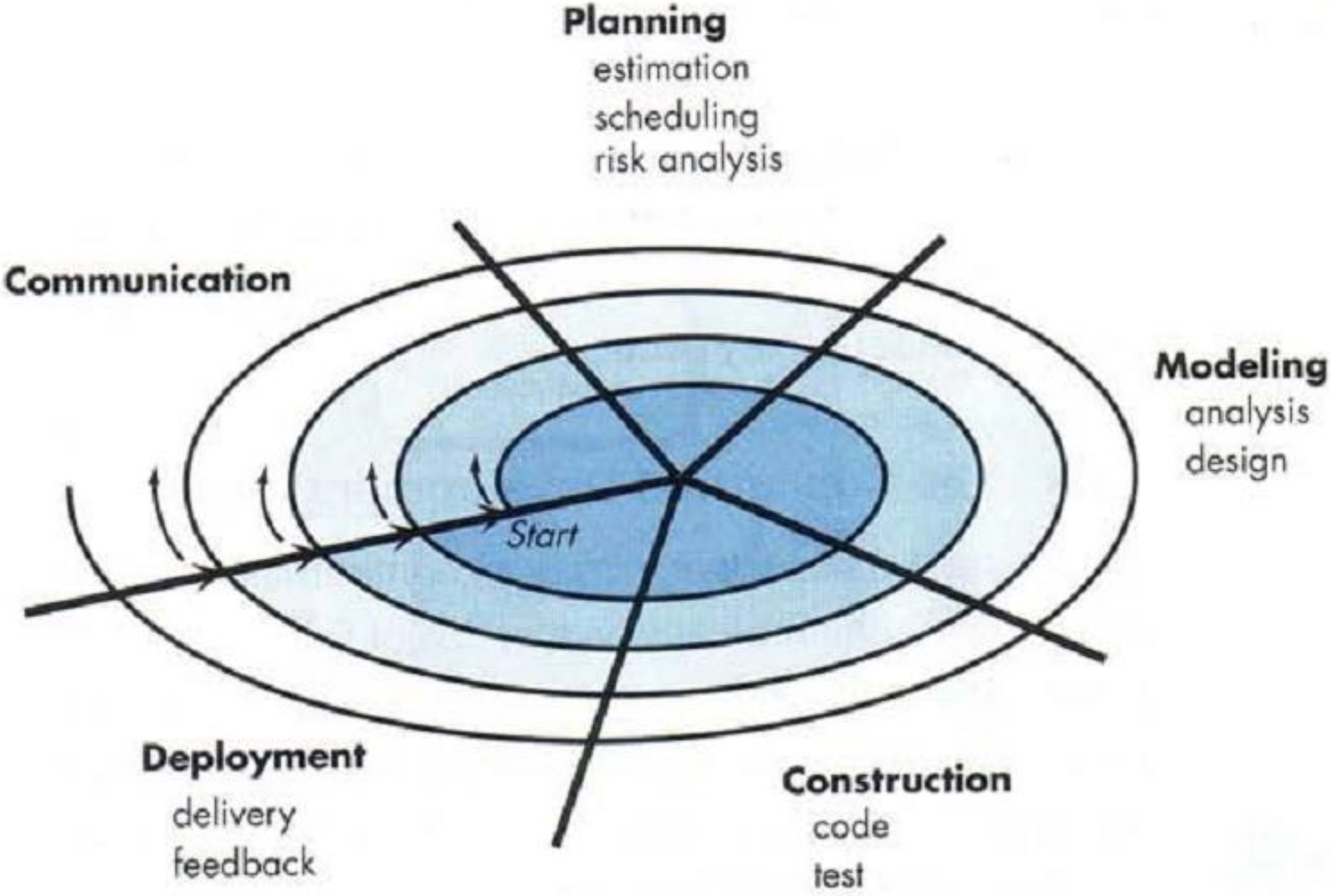
Evaluation of prototyping

- **Advantages:**
  - The model can be used when the requirements cannot or will not be specified.
  - The user can experiment with the system to improve the requirements
- **Disadvantages:**
  - Use of the method is exploratory in nature and therefore constitutes a high-risk endeavour. Strong management is required.

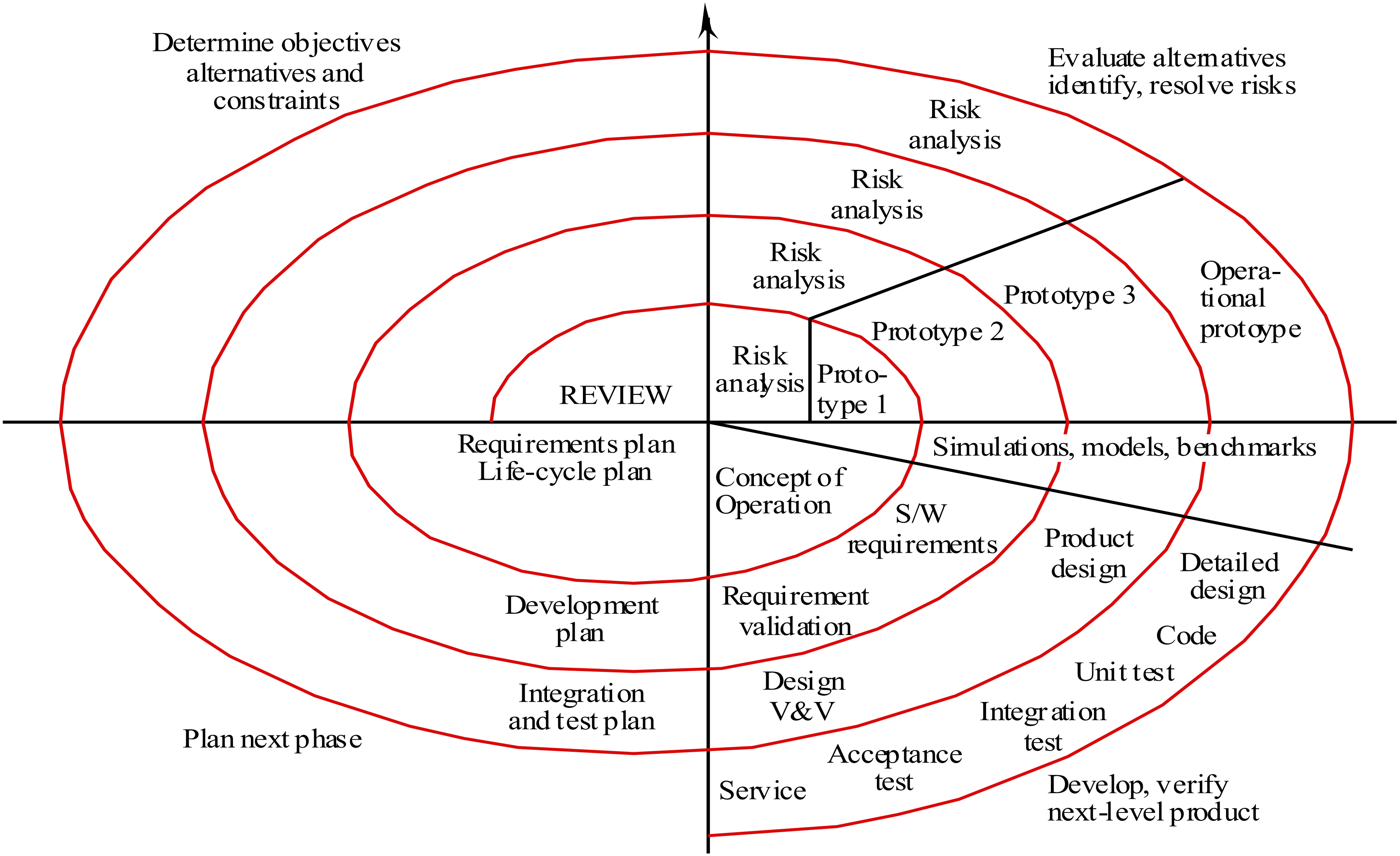
3 iii) The Spiral model

- Couples the **iterative nature of prototyping** with the **controlled and systematic aspects** of the **waterfall model**
- Process is represented as a spiral and each loop in the spiral represents a phase in the process.
- Provides the potential for rapid development of increasingly more complete versions of the software
- Risks are explicitly assessed and resolved throughout the process

The spiral model



Spiral model of the software process



Spiral model sectors

- **Objective setting**
  - Specific objectives for the phase are identified
- **Risk assessment and reduction**
  - Risks are assessed and activities put in place to reduce the key risks
- **Development and validation**
  - A development model for the system is chosen which can be any of the generic models
- **Planning**
  - The project is reviewed and the next phase of the spiral is planned

An evaluation of spiral model

- **Integration of technical development and risk management**
- **Rational incorporation of prototypes**
  - **All prototypes serve to reduce risk, e.g.**
    - **Risk of misunderstanding customer requirements**
    - **Risk of unfamiliarity with implementation tools**
    - **Risk of unfeasible architecture**
- **Applicability**
  - For development of large-scale systems and software
  - risk analysis is primarily important

Pros and cons of Evolutionary development

Benefits:

Flexibililty and extensiblity

Challenges:

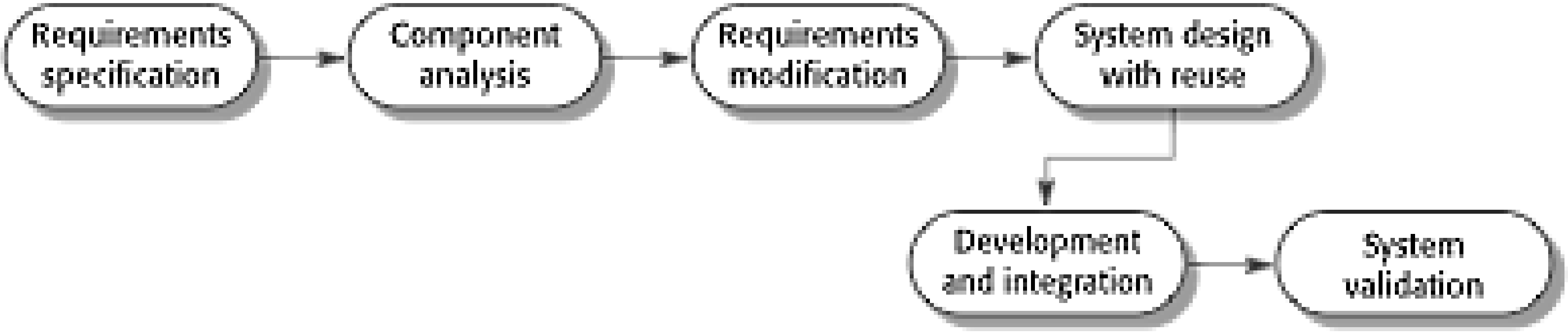
- prototyping poses a problem on project planning because of the uncertain number of cycles to complete the product

- Speed of evolution
  - too fast -> fail!
  - too slow -> affects productivity

4. Component-based/reuse-oriented software engineering

- 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**.
- Reuse is now the standard approach for building many types of business system

Reuse-oriented software engineering


Requirements specification
Component analysis
Requirements modification
System design with reuse
Development and integration
System validation

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

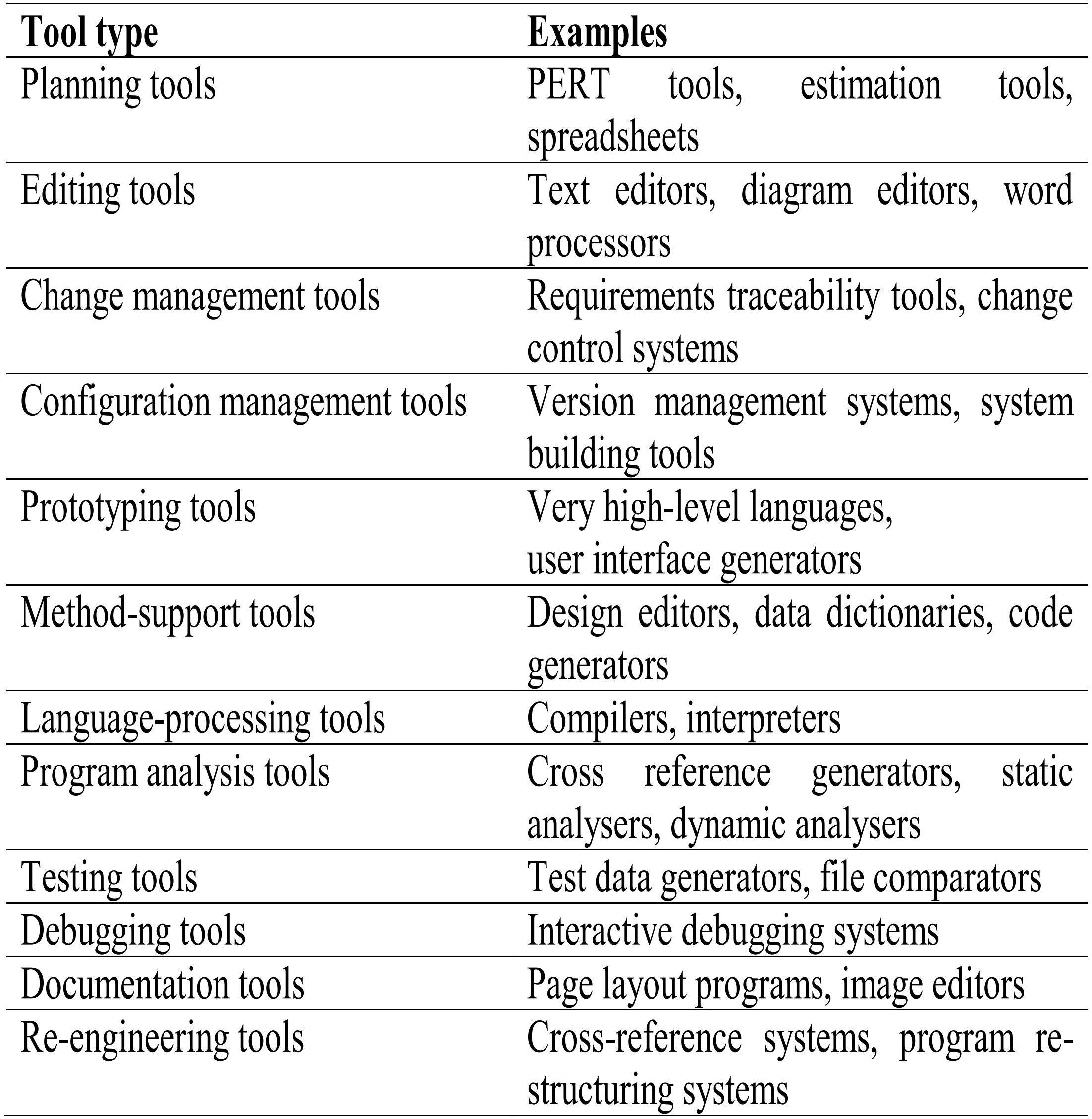
Case technology

- Case technology has led to significant improvements in the software process though not the order of magnitude improvements that were once predicted
  - Software engineering requires creative thought - this is not readily automatable
  - Software engineering is a team activity and, for large projects, much time is spent in team interactions. CASE technology does not really support these

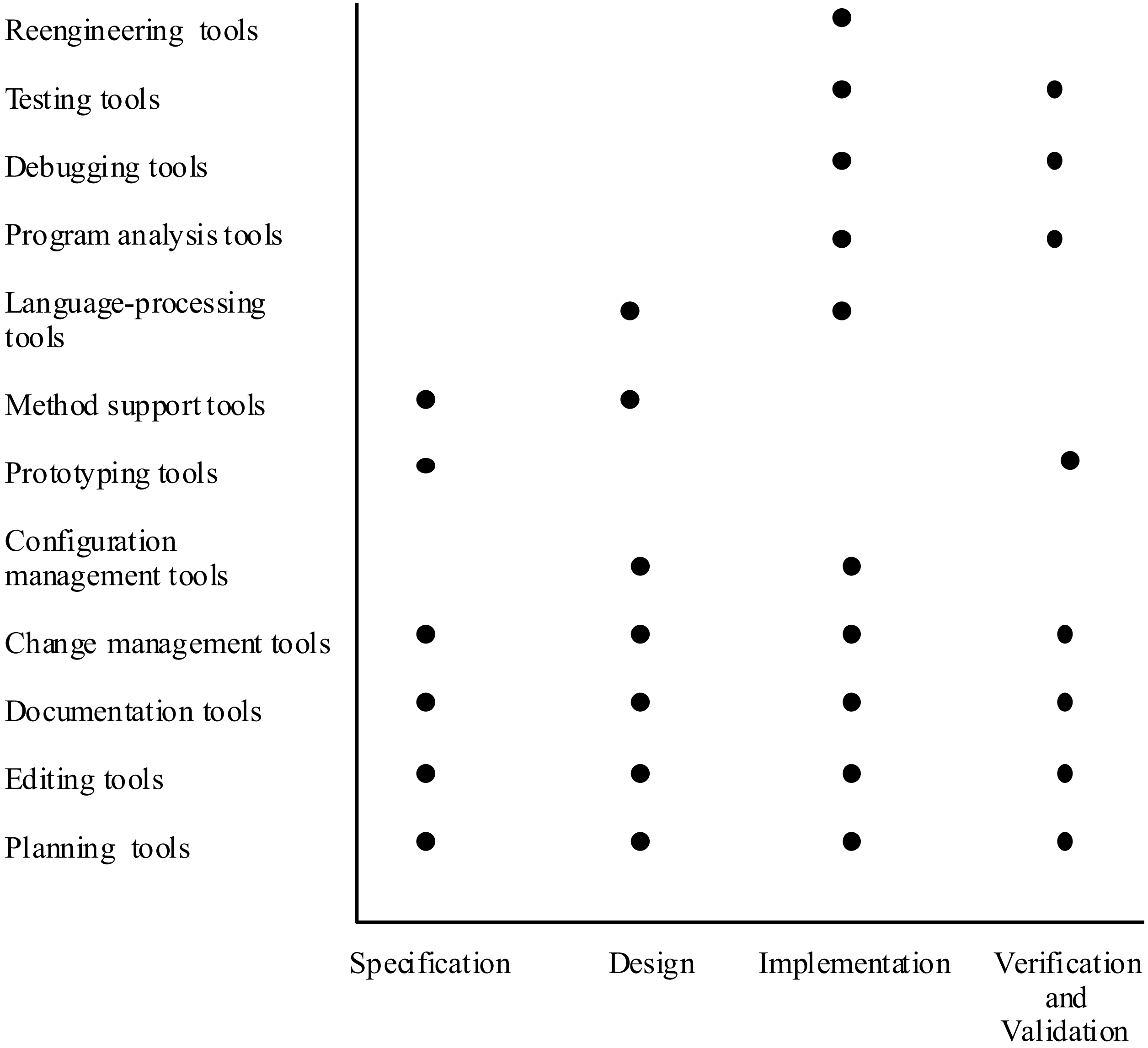
CASE classification

- Classification helps us understand the different types of CASE tools and their support for process activities
- 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

Functional tool classification



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



	Specification	Design	Implementation	Verification and Validation
Reengineering tools			●	
Testing tools			●	●
Debugging tools			●	●
Program analysis tools			●	●
Language-processing tools		●	●	
Method support tools	●	●		
Prototyping tools	●			●
Configuration management tools		●	●	
Change management tools	●	●	●	●
Documentation tools	●	●	●	●
Editing tools	●	●	●	●
Planning tools	●	●	●	●

CASE integration

- 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. Normally include several integrated 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
- CASE technology supports software process activities