CSc 131
Computer Software Engineering

Chapter 5
SW Processes

Herbert G. Mayer, CSU CSC

Status 8/18/2019

Syllabus

- SWE Models
- SW Design Process
- Process Activities
- Structured Methods
- Software Validation
- Testing Phases
- Summary
- References

Objectives

Objectives

- To introduce software process models
- To describe three generic process models and when they may be used
- To describe outline process models for requirements engineering, software development, testing and evolution
- To explain Rational Unified Process model
- To introduce CASE technology to support software process activities

Topics Covered

- Software process models
- Process iteration
- Process activities
- Rational Unified Process
- Computer-aided software engineering

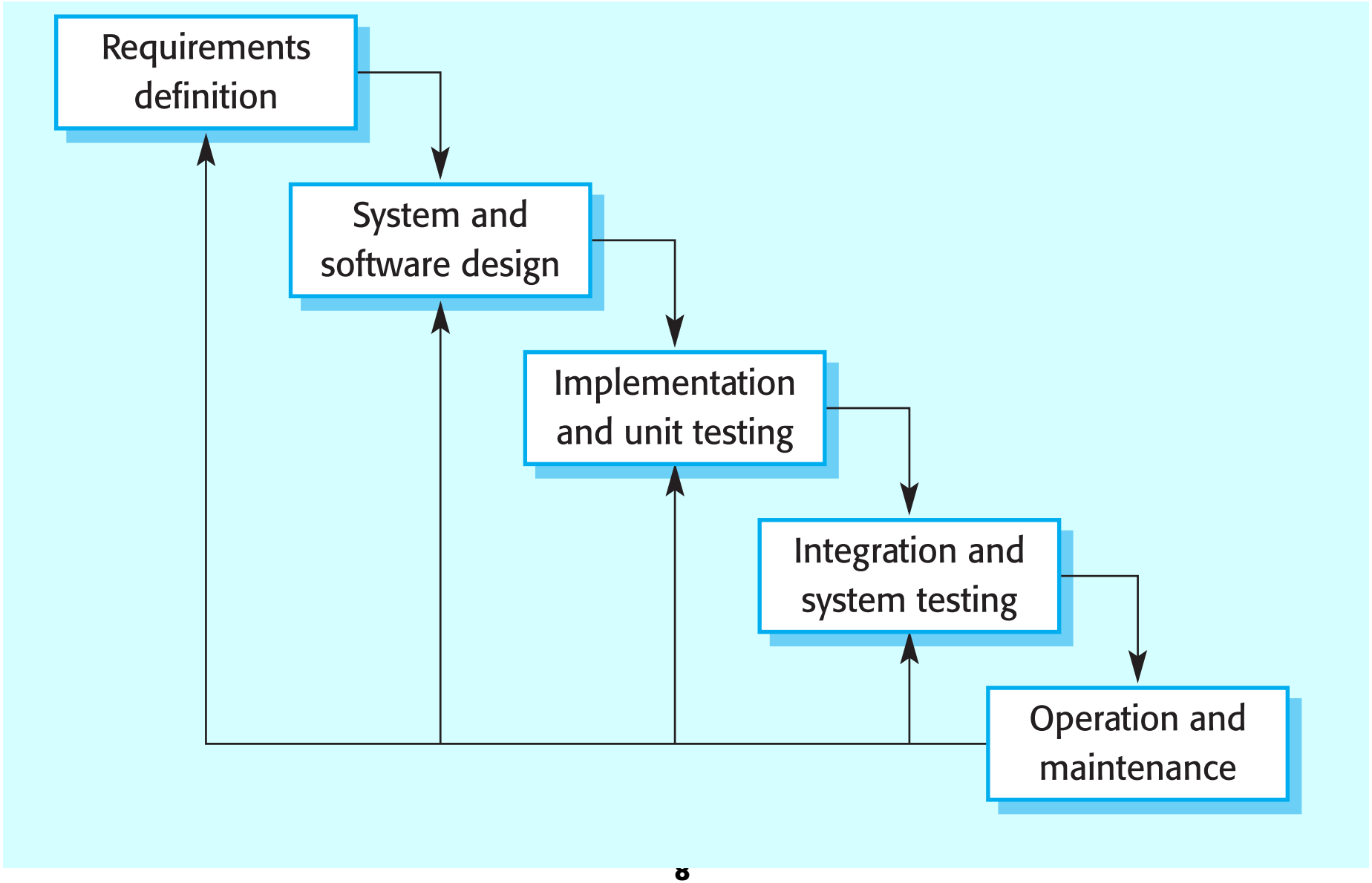
The SW Process

- A structured set of activities required to develop a software system
  - Specification
  - Design
  - Validation
  - Evolution
- A software process model is an abstract representation of a process
- It presents a description of a process from some particular perspective

Generic SW Process Models

- The Waterfall Model
  - Separate and distinct phases of specification and development
- Evolutionary development
  - Specification, development and validation are interleaved
- Component-based software engineering
  - System assembled from existing components
- There exists numerous variants of these models
- E.g. formal development where a waterfall-like process is used but the specification is quite formal, and refined through several stages to an implementable design
- Each stage to be formally reviewed!

Waterfall Model



Waterfall Model Phases

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

Main drawback of waterfall model is the difficulty of accommodating change after the process is underway

Any one phase has to be complete before moving onto the next phase

Waterfall Model Problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited and unlikely during design process
- Few business systems have stable requirements
- Waterfall model mostly used for large systems engineering projects where a system is developed at several sites

Evolutionary Development

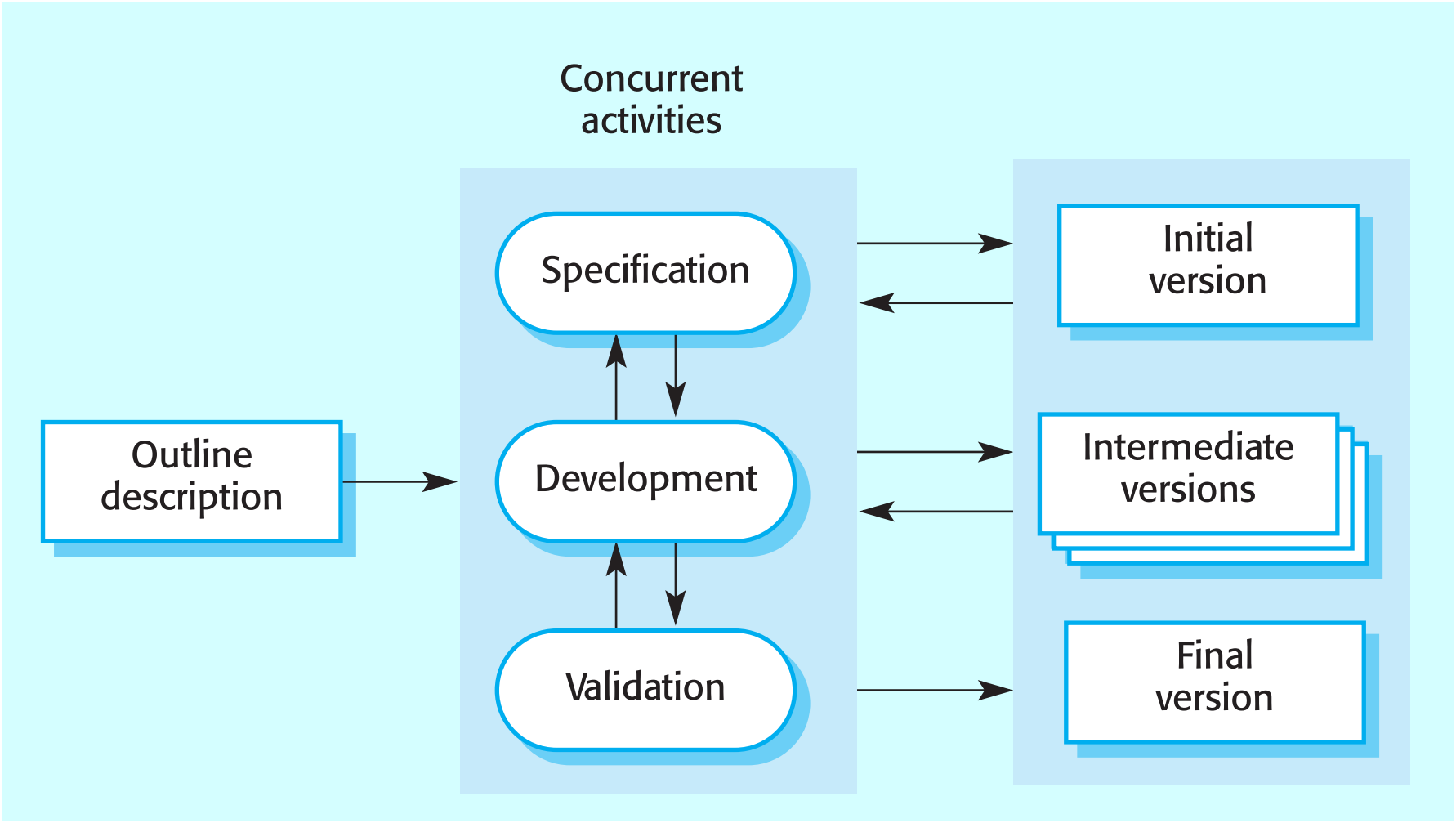
Exploratory development

- Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements and add new features as proposed by the customer

Throw-away prototyping

- Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed

Evolutionary Development



Evolutionary Development

Problems

- Lack of process visibility;
- Systems are often poorly structured;
- Special skills (e.g. in languages for rapid prototyping) may be required

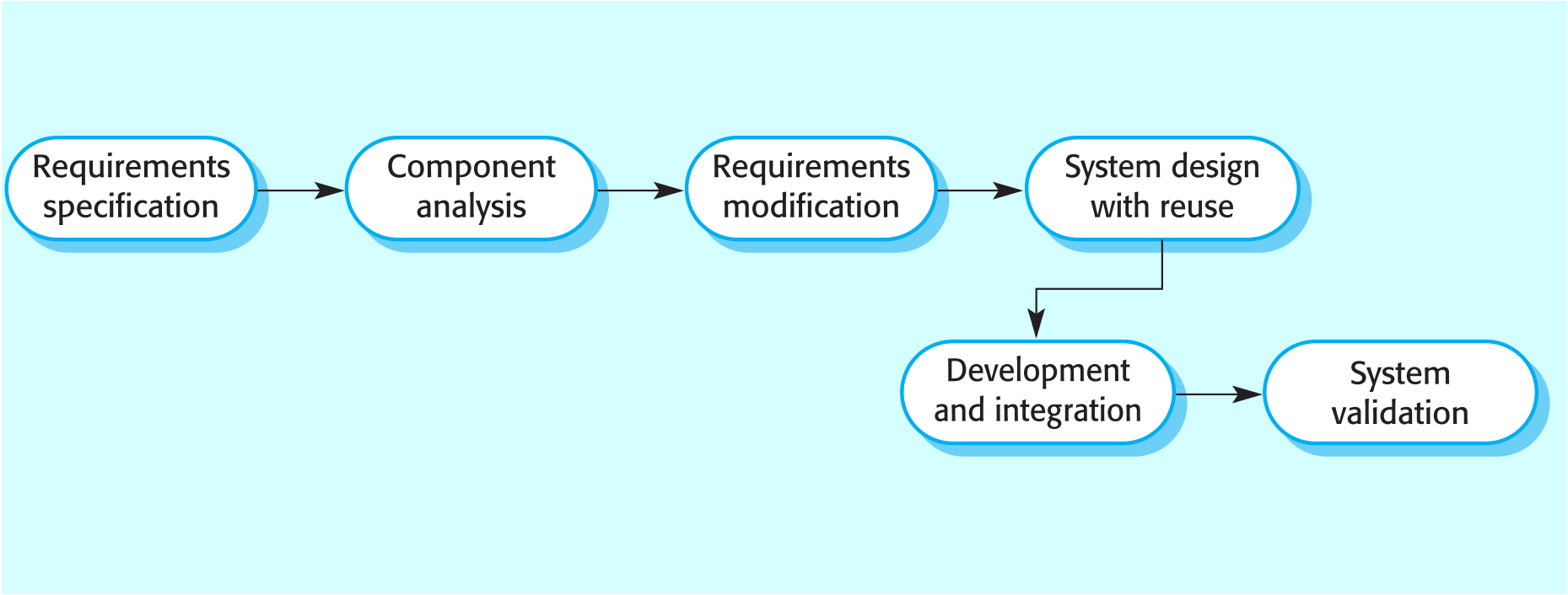
Applicability

- For small or medium-size interactive systems;
- For parts of large systems (e.g. the user interface);
- For short-lifetime systems

Component-Based SWE

- 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 increasingly used as component standards have emerged

Reuse-Oriented Development



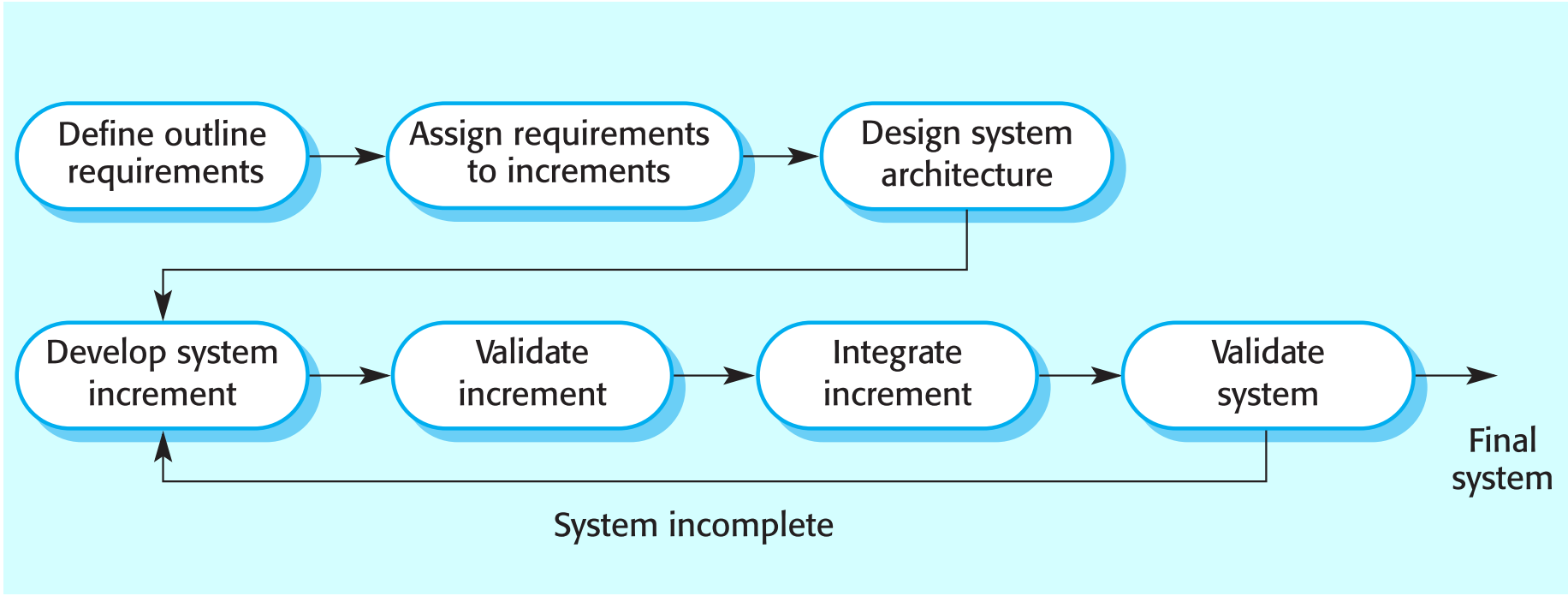
Process Iteration

- System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems
- Iteration can be applied to any of the generic process models
- Two related approaches
  - Incremental delivery
  - Spiral development

Incremental Delivery

- Rather than delivering a system in a single step, development and delivery are broken into increments with each component delivering part of the required functionality
- 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



Incremental Development Pros

- Customer value can be delivered with each increment so 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 tend to receive the most testing

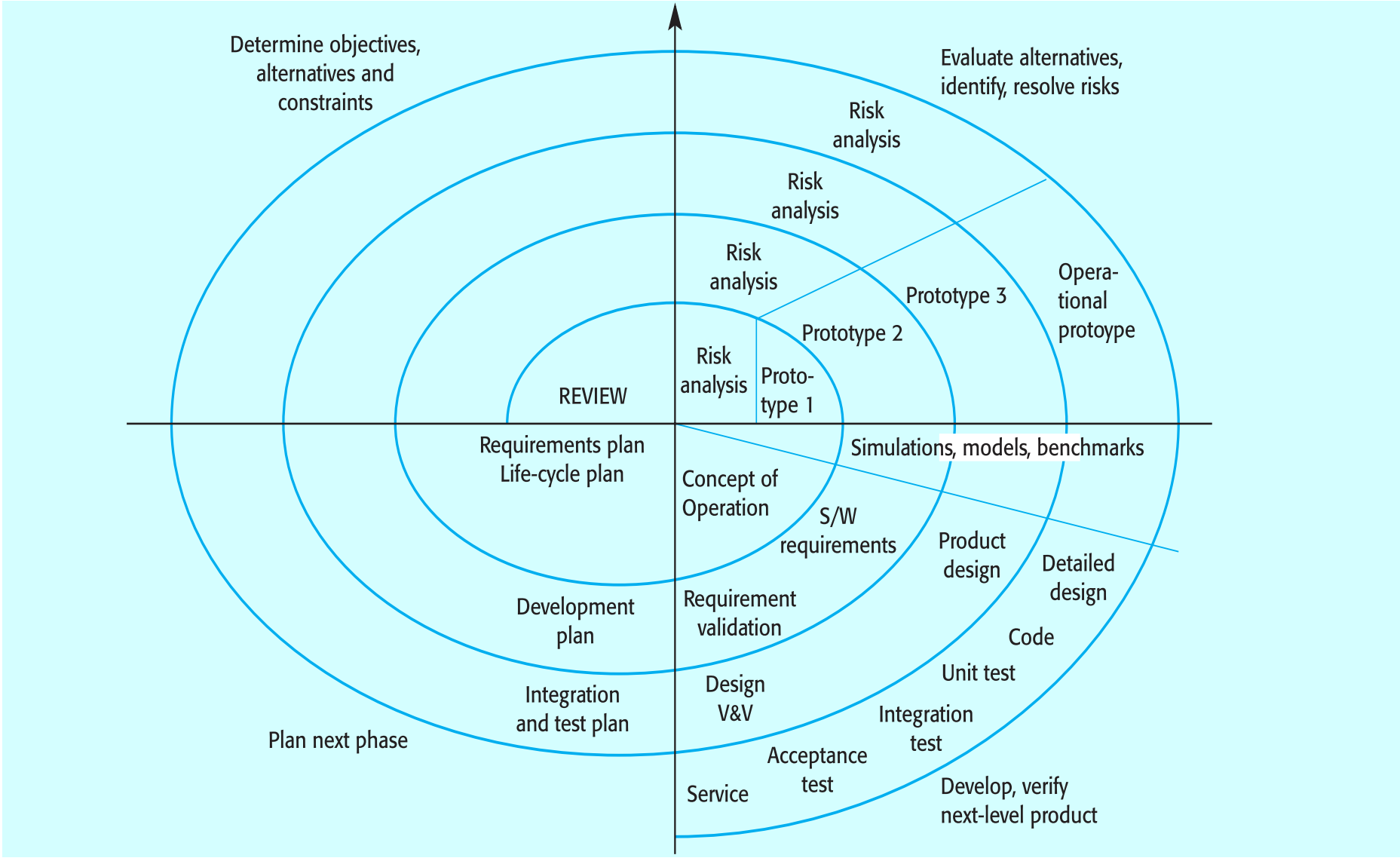
Extreme Programming

- An approach to development based on the creation and delivery of very small increments of functionality
- Relies on constant code improvement, user involvement in the development team
- Usually pairwise programming

Spiral Development

- Process is represented as a spiral rather than as a sequence of activities with backtracking
- Each loop in the spiral represents a phase in the process
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
- Risks are explicitly assessed and resolved throughout the process

Spiral Model of SW 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

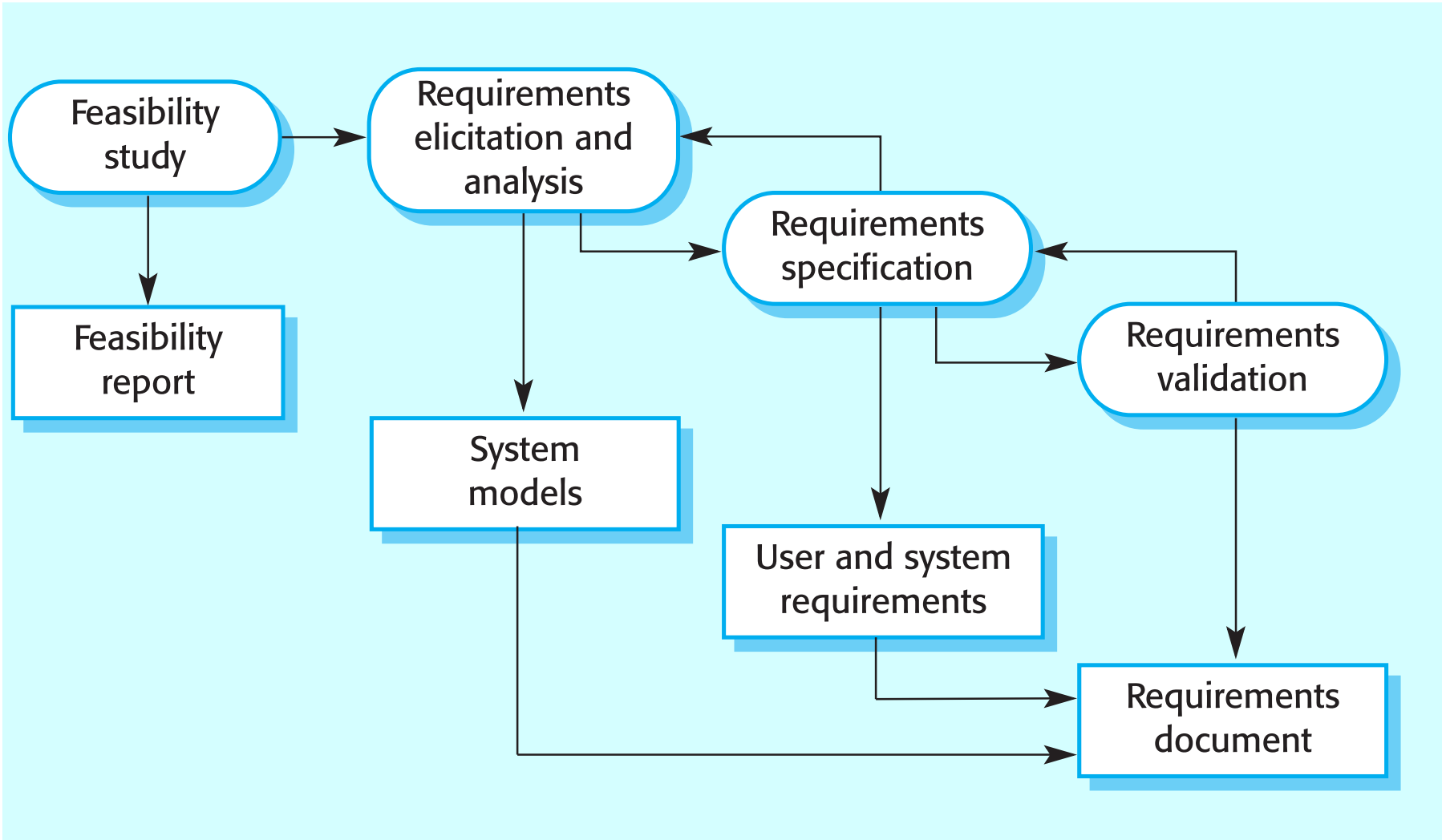
Process Activities

- Software specification
- Software design and implementation
- Software validation
- Software evolution

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;
  - Requirements elicitation and analysis;
  - Requirements specification;
  - Requirements validation.

Requirements Engineering Process



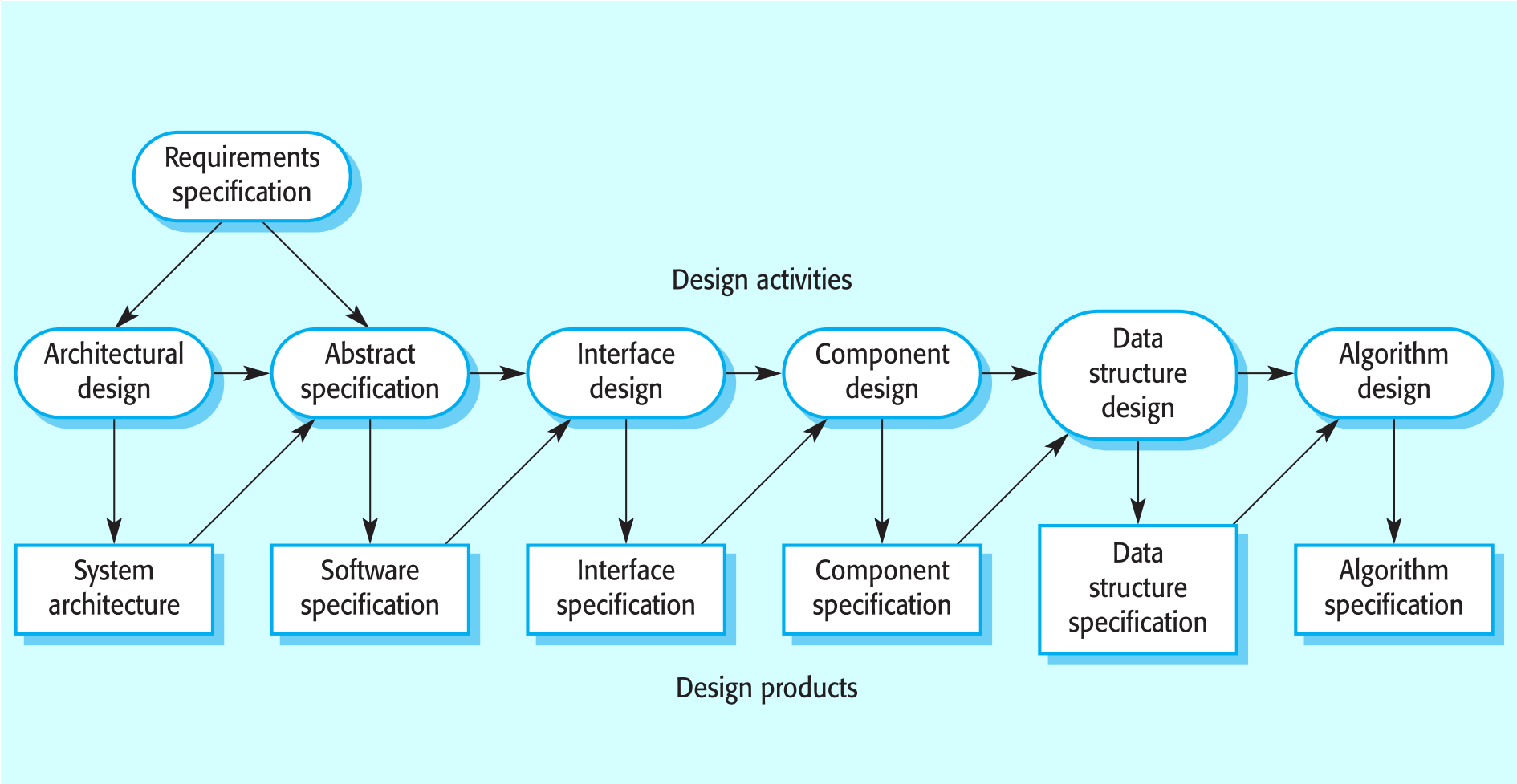
SW Design & Implementation

- 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

Design Process Activities

- Architectural design
- Abstract specification
- Interface design
- Component design
- Data structure design
- Algorithm design

Software Design Process



Structured Methods

Systematic approaches to developing software design

Design typically documented by graphical models

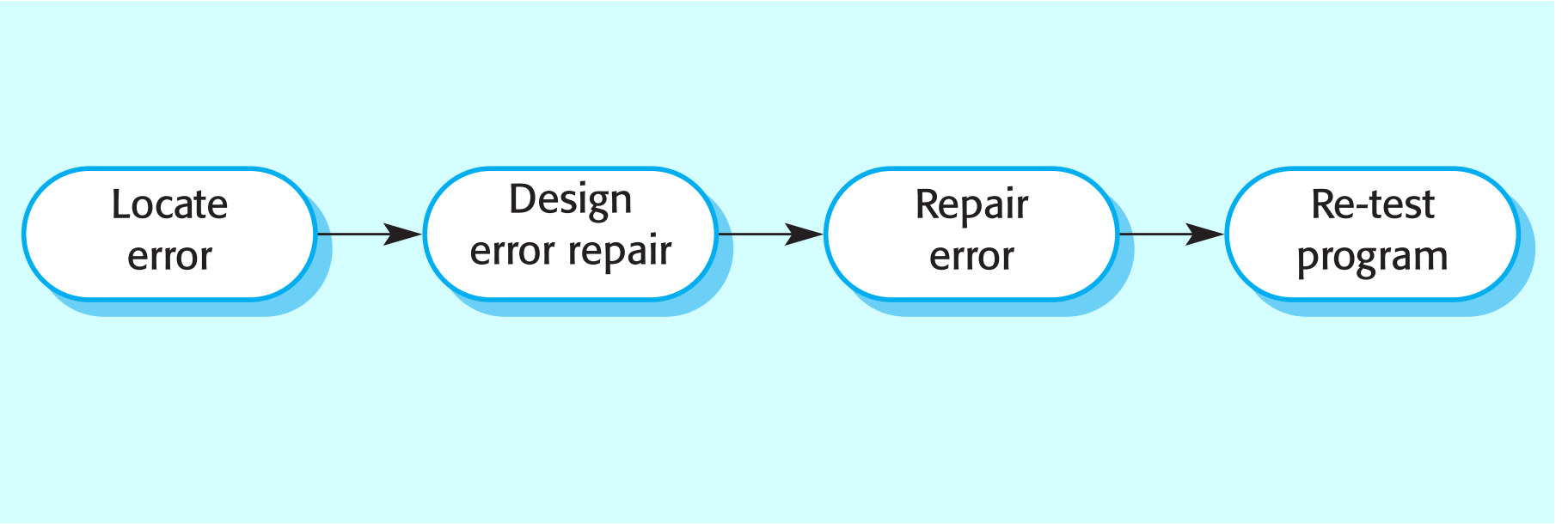
Possible models:

- **Object model**
- **Sequence model**
- **State transition model**
- **Structural model**
- **Data-flow model**

Programming & 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

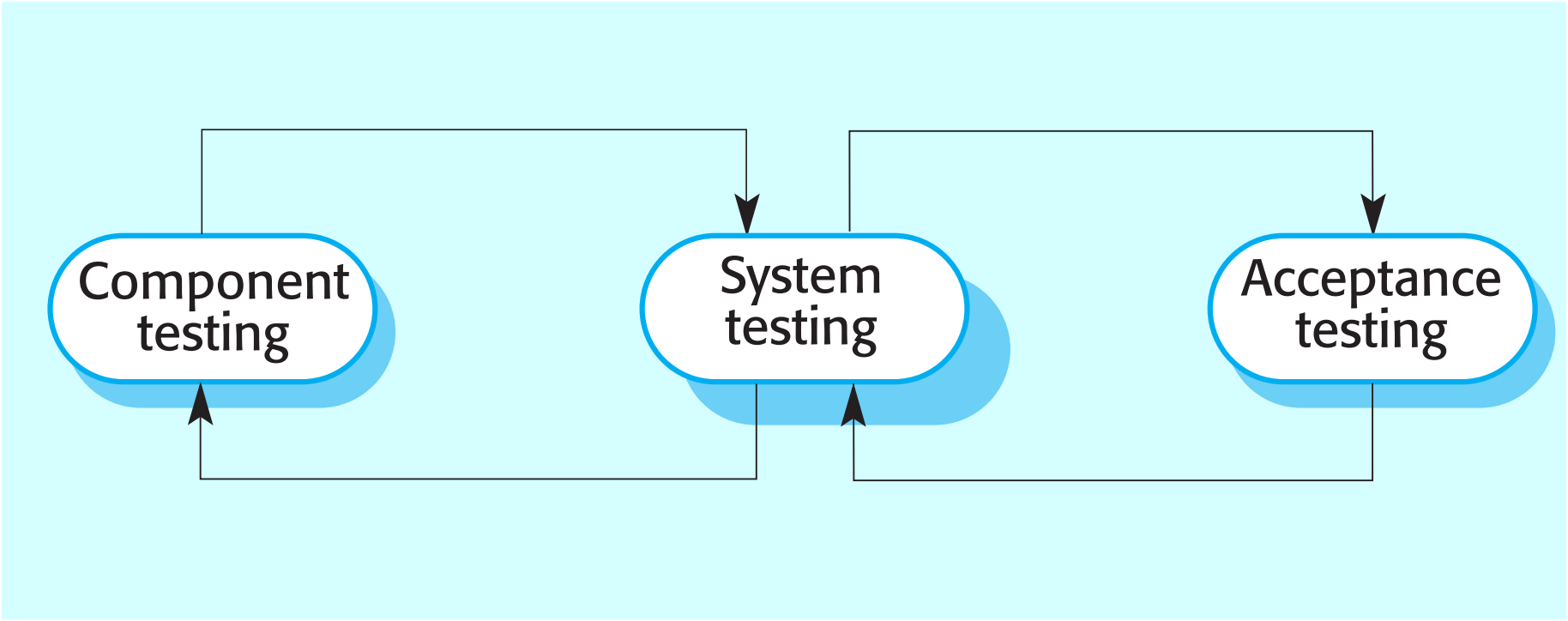
Debugging Process



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 Process



Testing Stages

Component or unit testing

- Individual components are tested independently
- Components may be functions or objects or coherent groupings of these entities

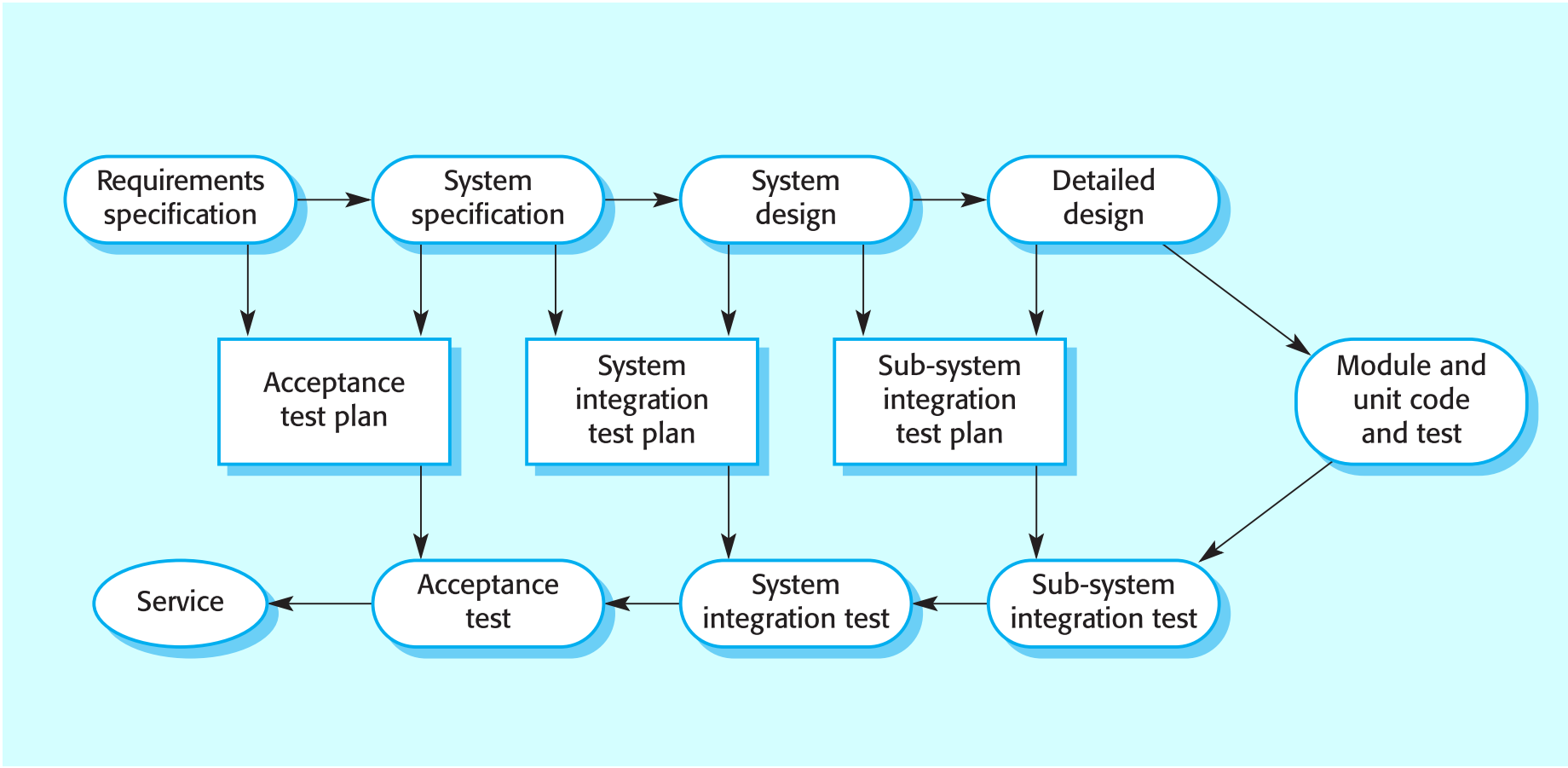
System testing

- 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

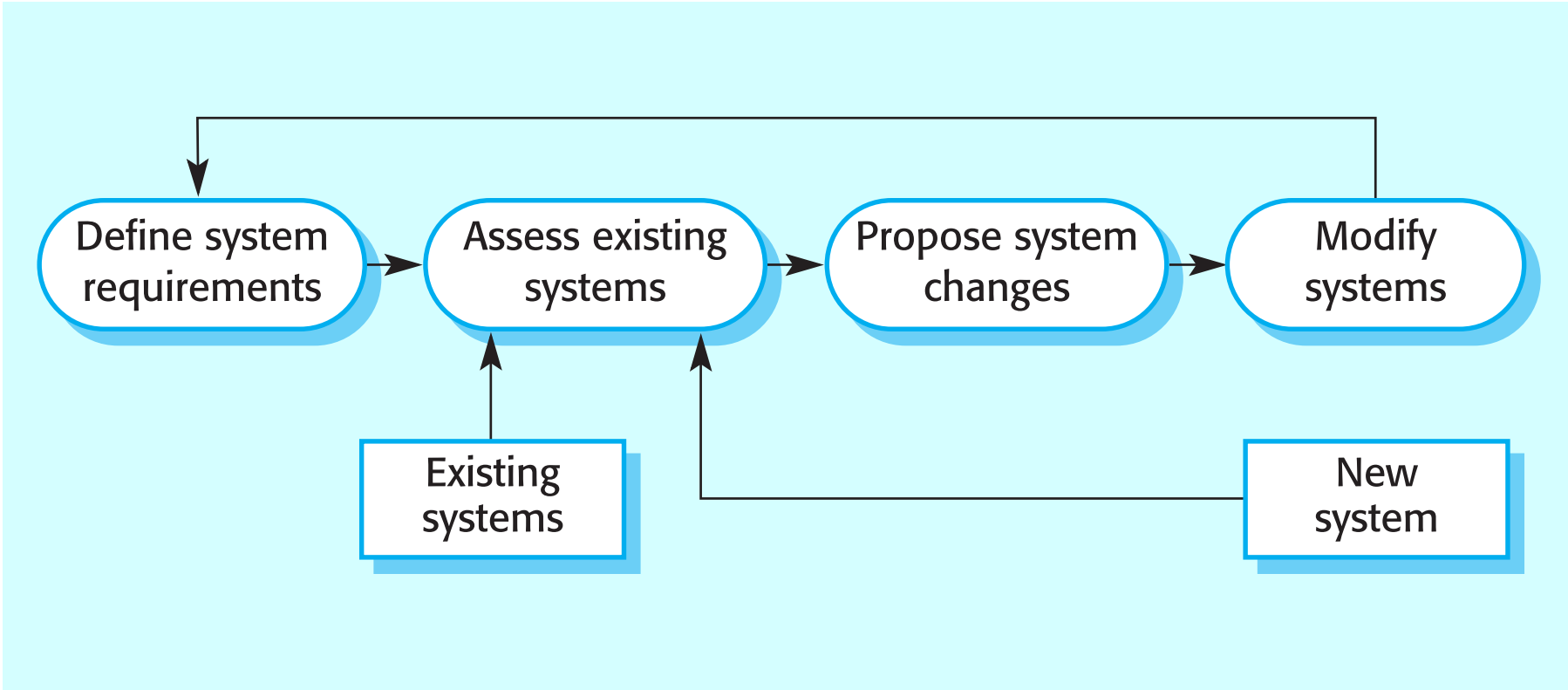
Testing Phases



Software Evolution

- **Software is inherently flexible and can change, is meant to be changeable!**
- **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 now**
- **Since fewer and fewer systems are completely new**

System Evolution



Rational Unified Process

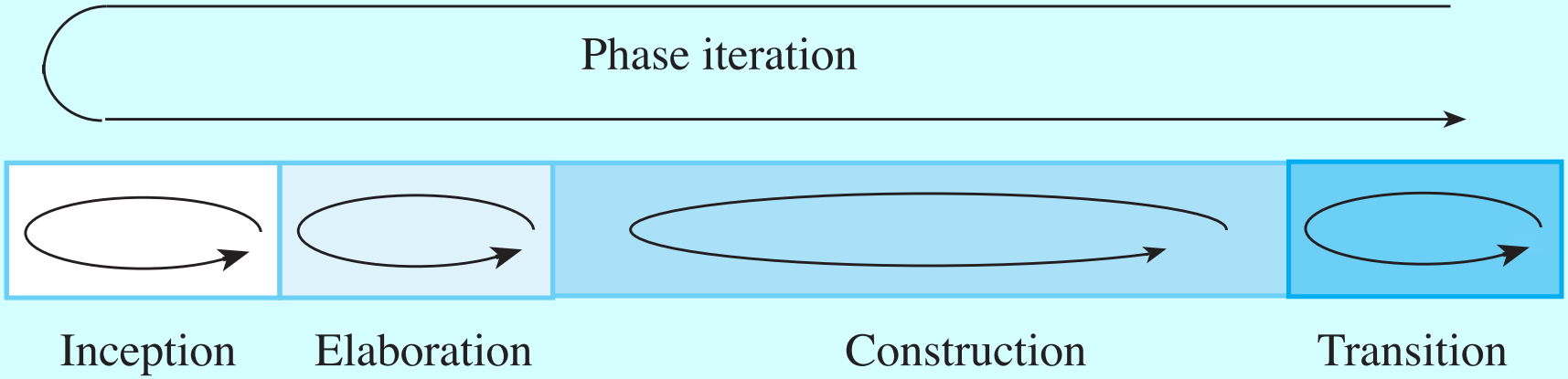
Rational Unified Process = RUP:

A modern process model derived from the work on the UML and associated process

Normally described from 3 perspectives

- **A dynamic perspective that shows phases over time**
- **A static perspective that shows process activities**
- **A practive perspective that suggests good practice**

RUP Phase Model



RUP phases

Inception

- Establish the business case for the system

Elaboration

- Develop an understanding of the problem domain and the system architecture

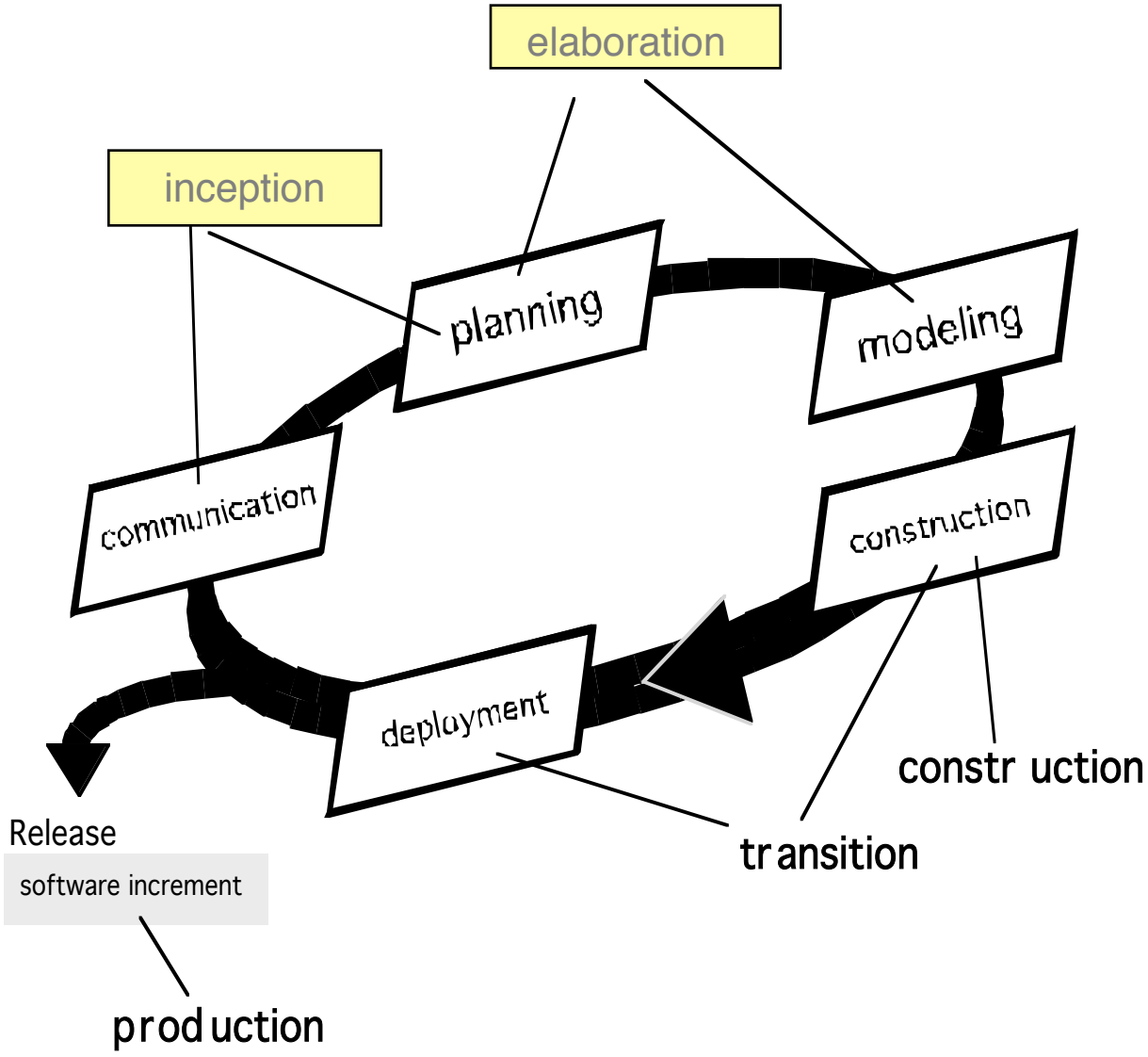
Construction

- System design, programming and testing

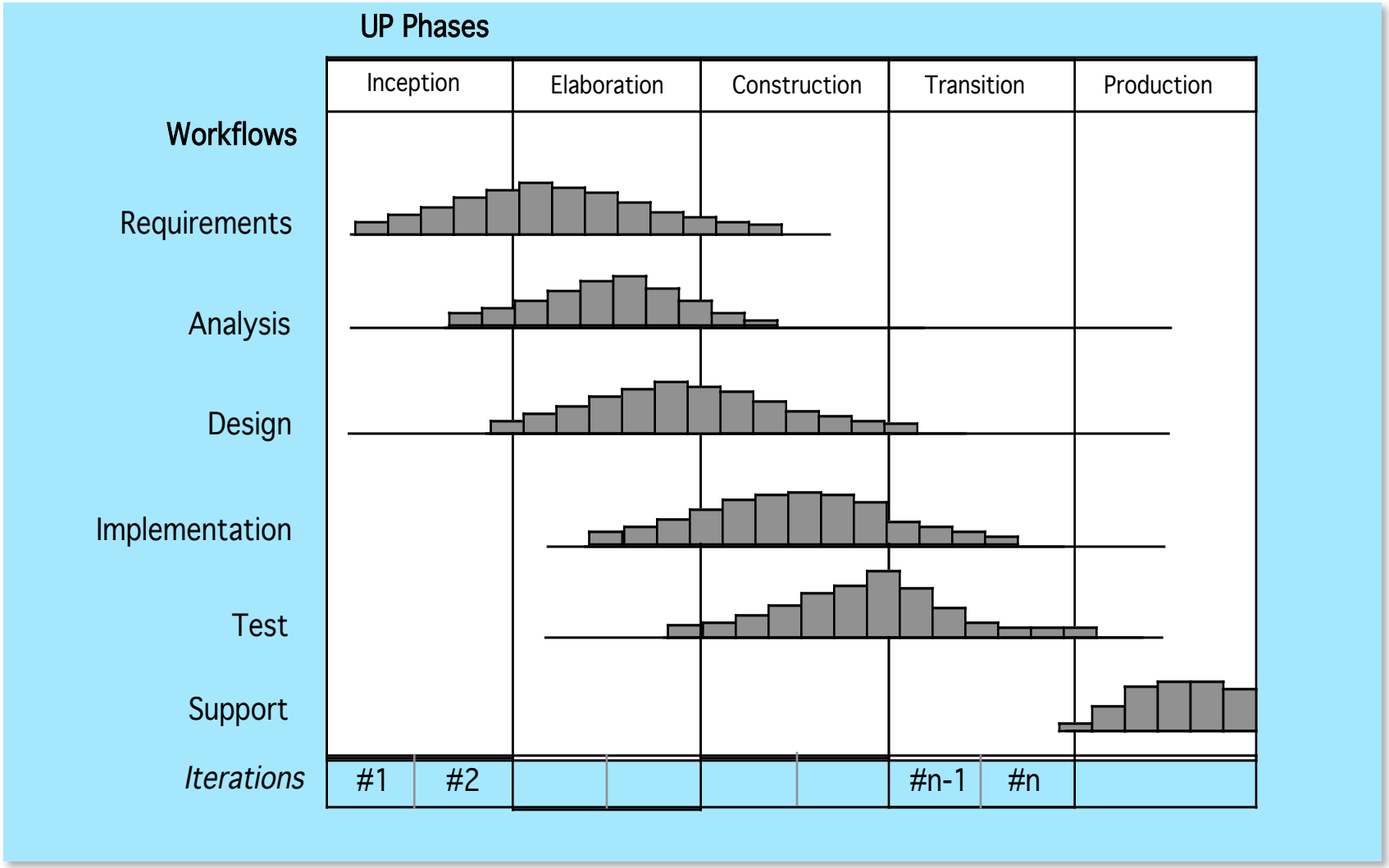
Transition

- Deploy the system in its operating environment

Rational Unified Process



RUP Phases


UP Phases
Inception
Elaboration
Construction
Transition
Production
Workflows
Requirements
Analysis
Design
Implementation
Test
Support
Iterations
#1
#2
#n-1
#n

RUP Work Products

Inception phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment.
- Project plan,
  - phases and iterations.
- Business model,
  - if necessary.
- One or more prototypes

Elaboration phase

- Use-case model
- Supplementary requirements
  - including non-functional
- Analysis model
- Software architecture
  - Description.
- Executable architectural
  - prototype.
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adapted workflows
  - milestones
  - technical work products
- Preliminary user manual

Construction phase

- Design model
- Software components
- Integrated software
  - increment
- Test plan and procedure
- Test cases
- Support documentation
  - user manuals
  - installation manuals
  - description of current
    - increment

Transition phase

- Delivered software increment
- Beta test reports
- General user feedback

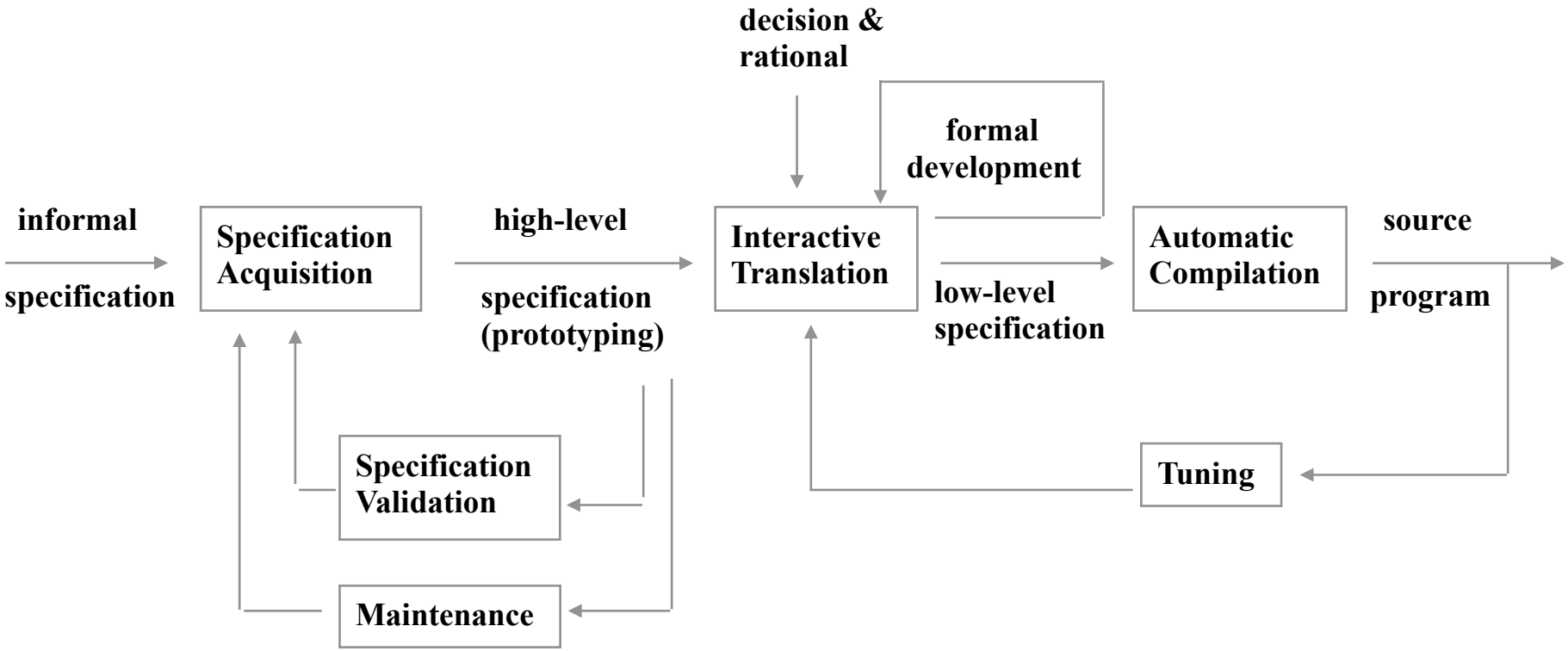
RUP Good Practice

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

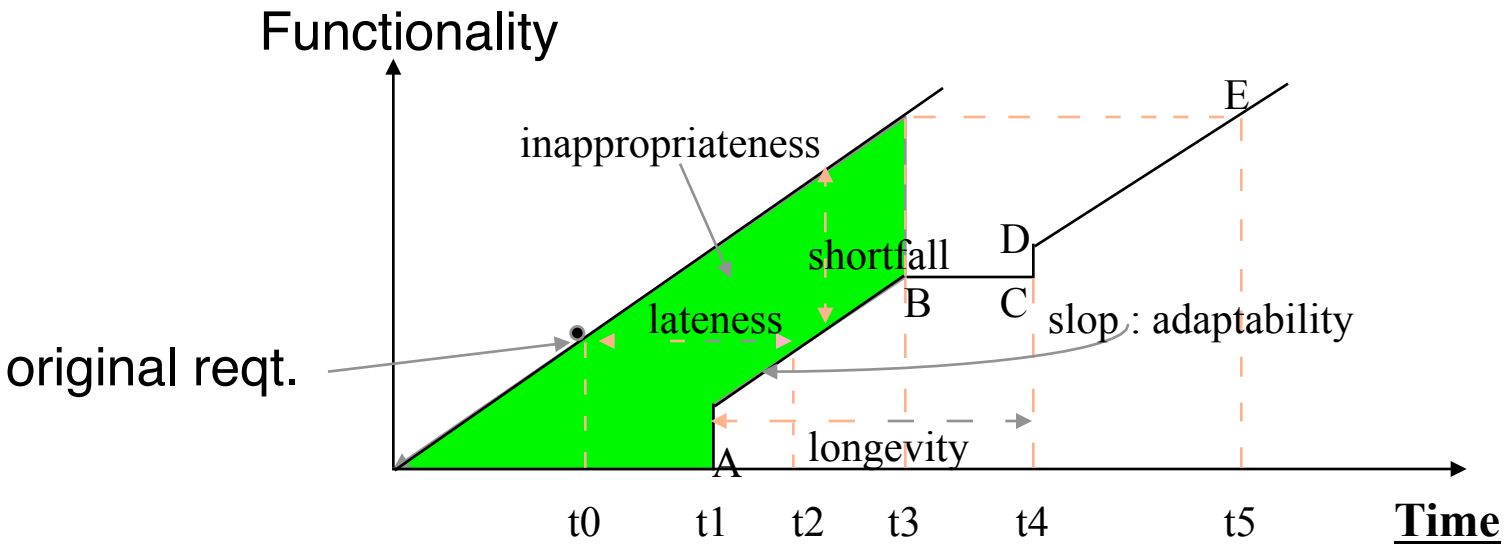
Static Workflows

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Test	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system (see Chapter 29).
Project management	This supporting workflow manages the system development (see Chapter 5).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Automated Synthesis Model



Various Process Models



waterfall model

O : (t_0) original reqt.

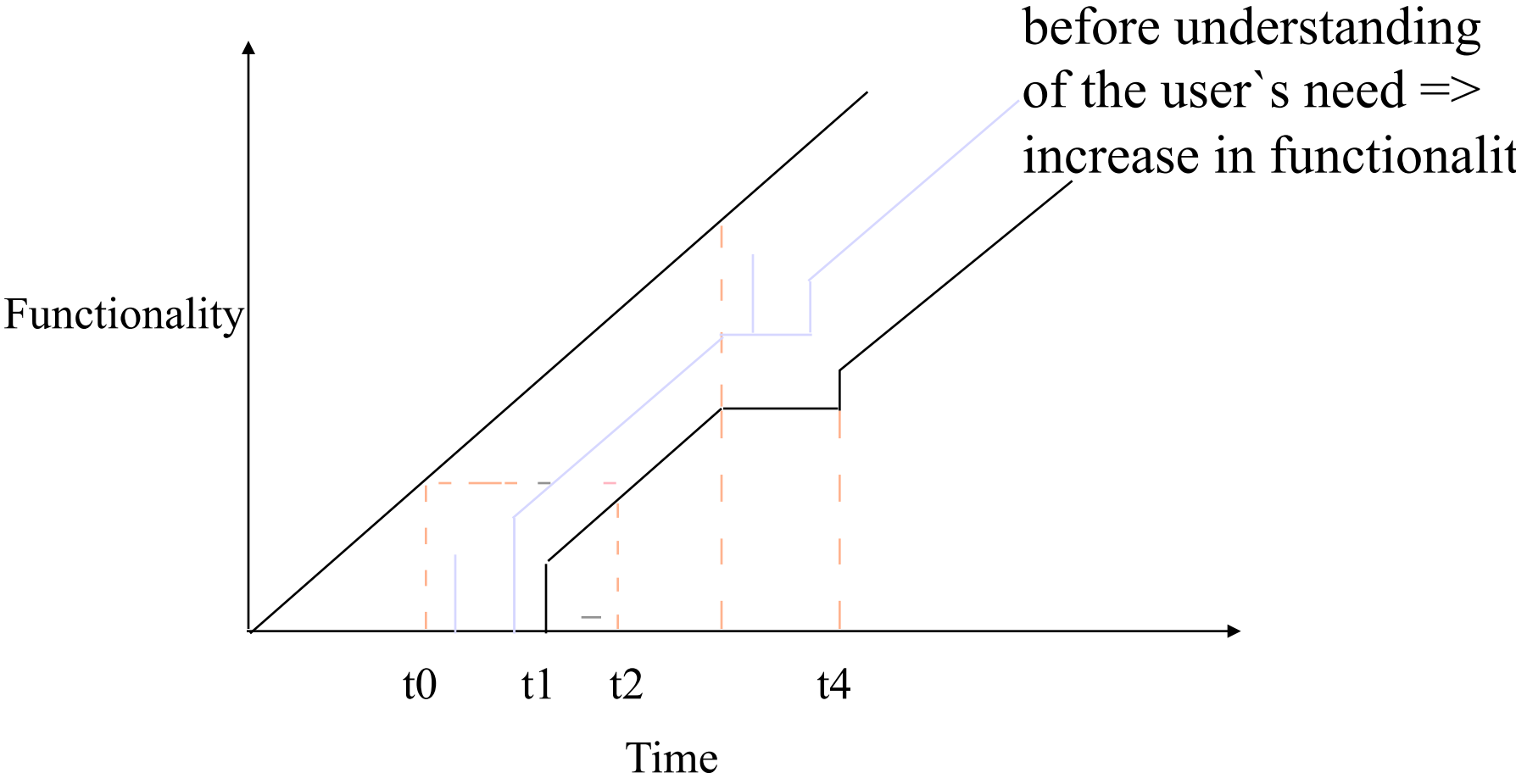
A : (at t1) operational product, not satisfying old to needs because poor understanding of needs

A - B : undergo a series of enhancements

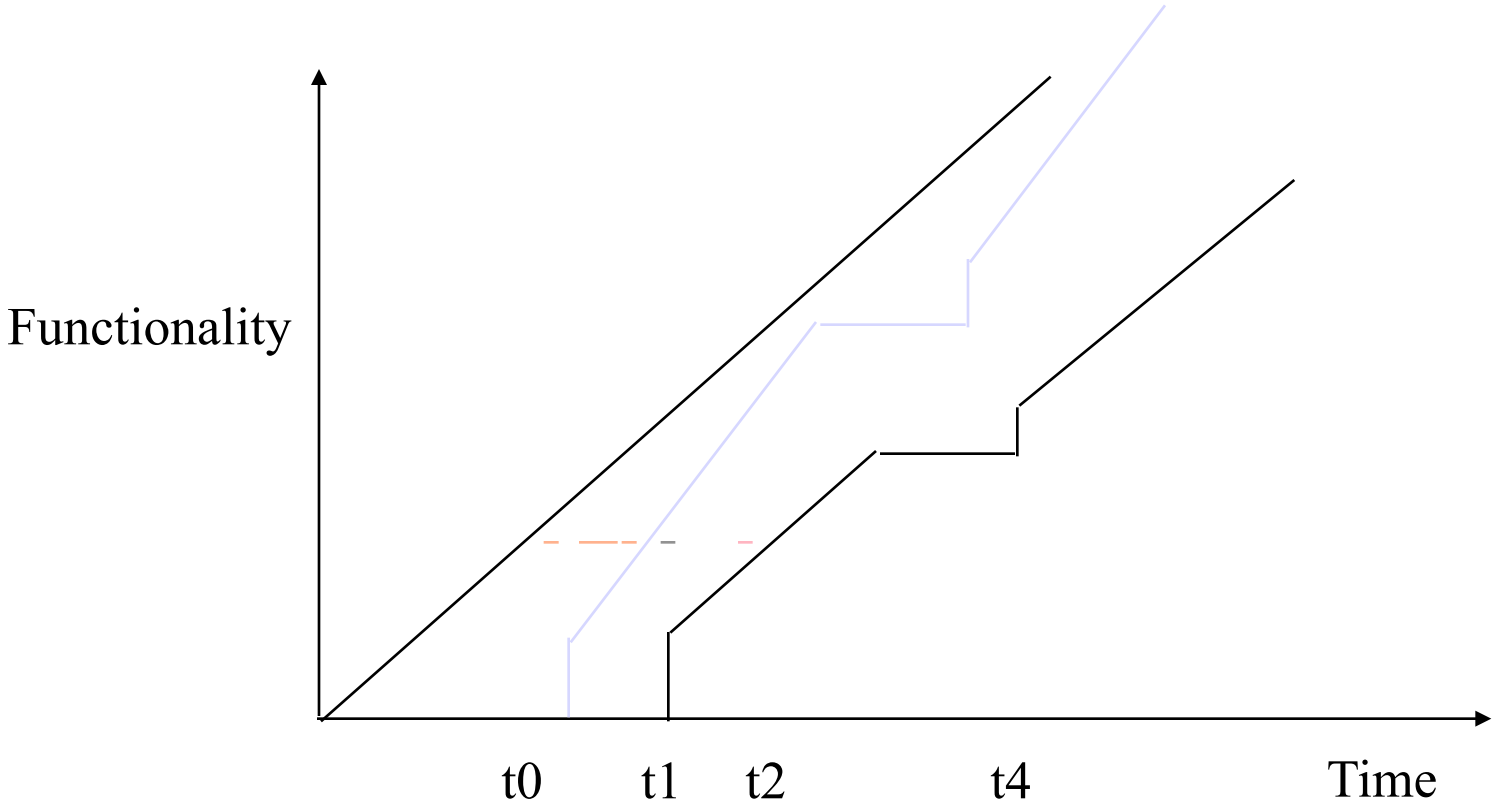
B - D : cost of enhancement increase, to build a new system stop at t4.

*** cycle repeat itself**

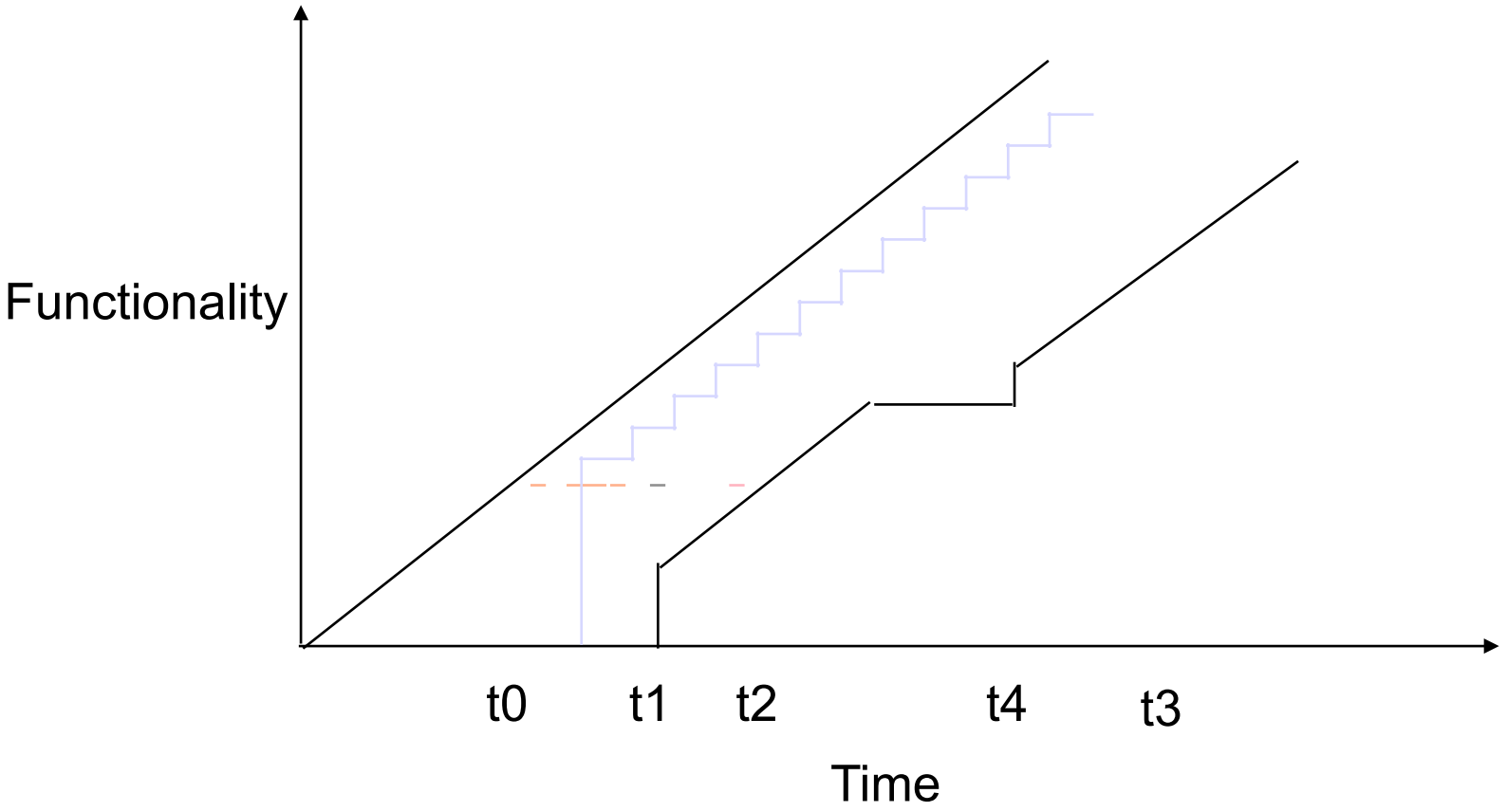
Throwaway Prototyping & Spiral Model



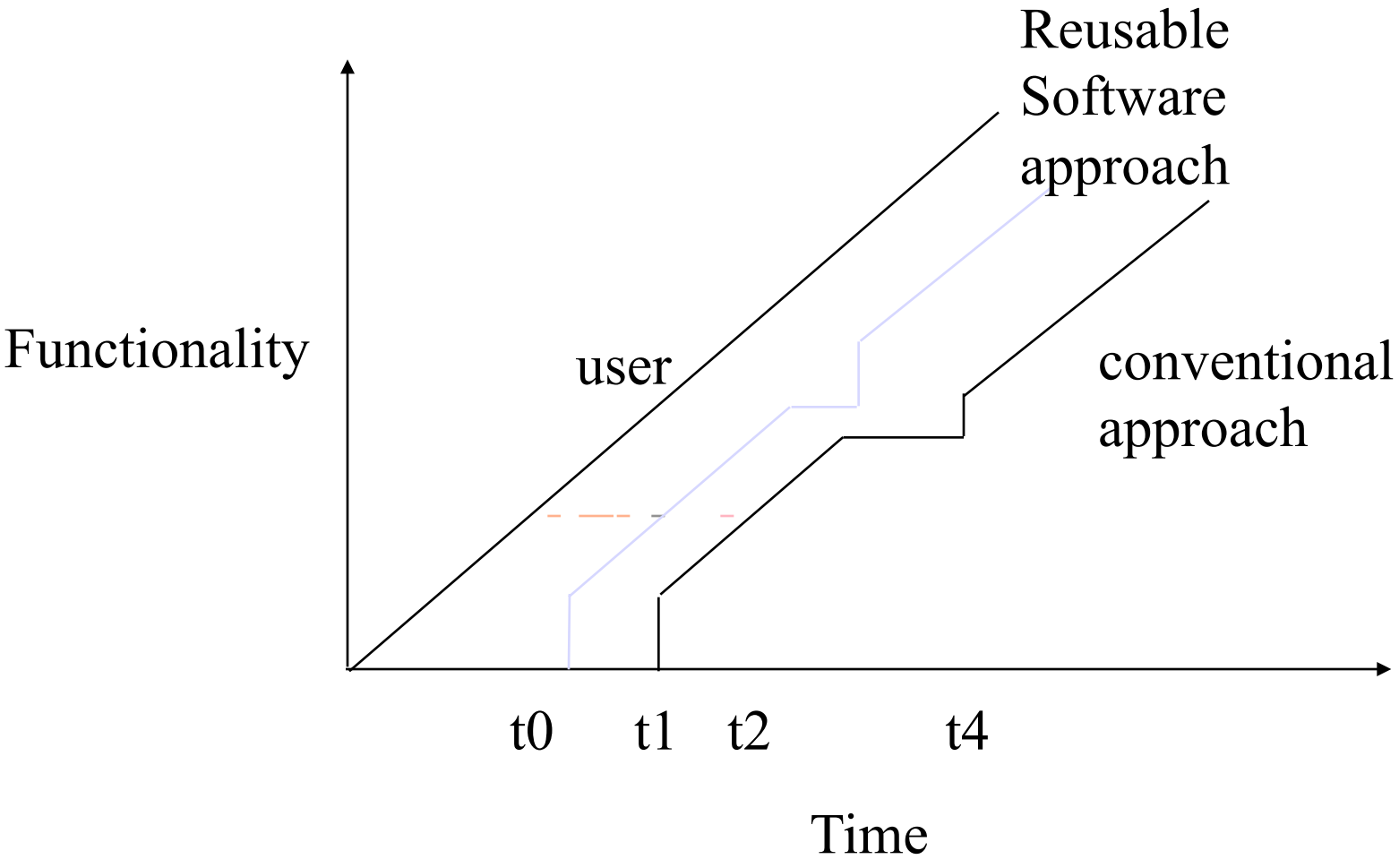
Evolutionary Prototyping



Automated Software Synthesis



Reusable Software vs. Conventional



Computer-Aided SWE

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 translator generates new program version**

CASE Technology

CASE technology has led to significant improvements in the software process. However, these are not the order of magnitude improvements that were once predicted:

- **SWE requires creative thought - this is not readily automated**
- **SWE is team activity; for large projects much time is spent in team interactions**
- **CASE technology does not naturally support these**

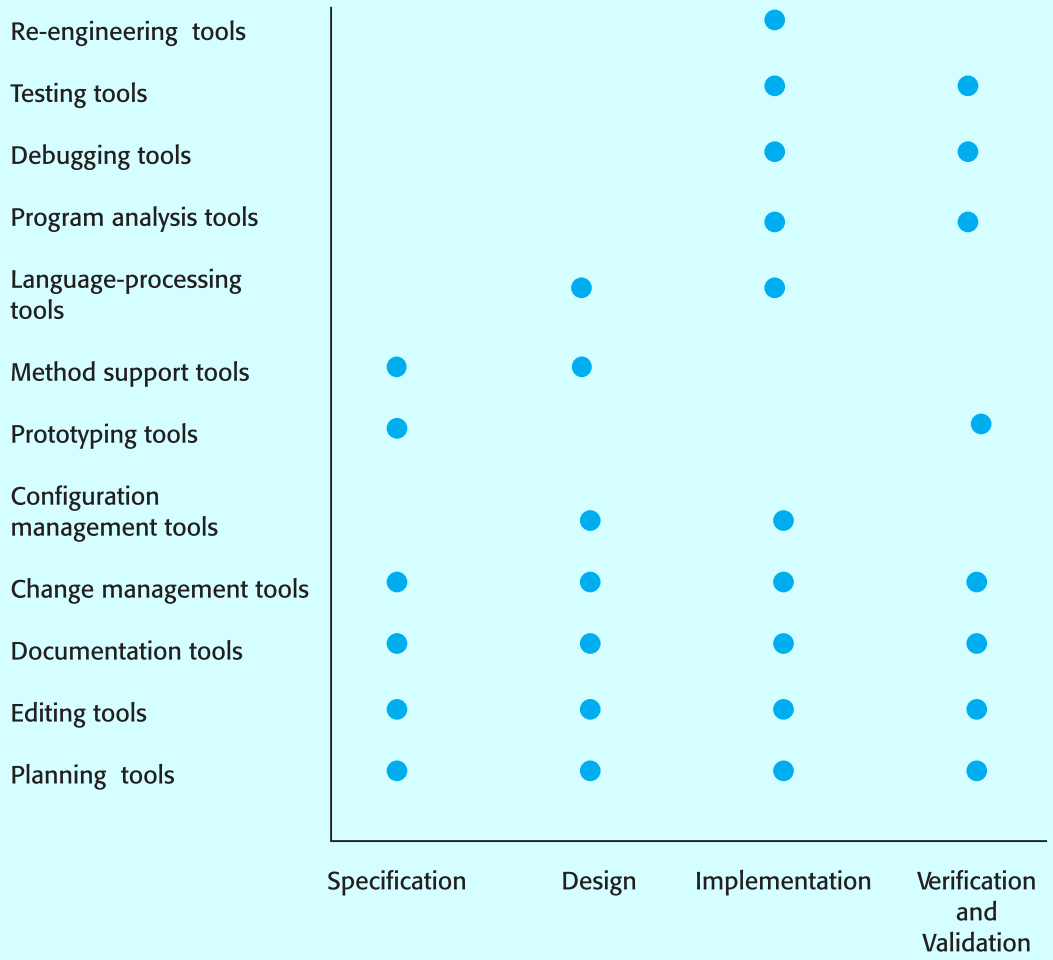
CASE Classification

- 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

Activity-Based Tool Classification



CASE Integration

Tools

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

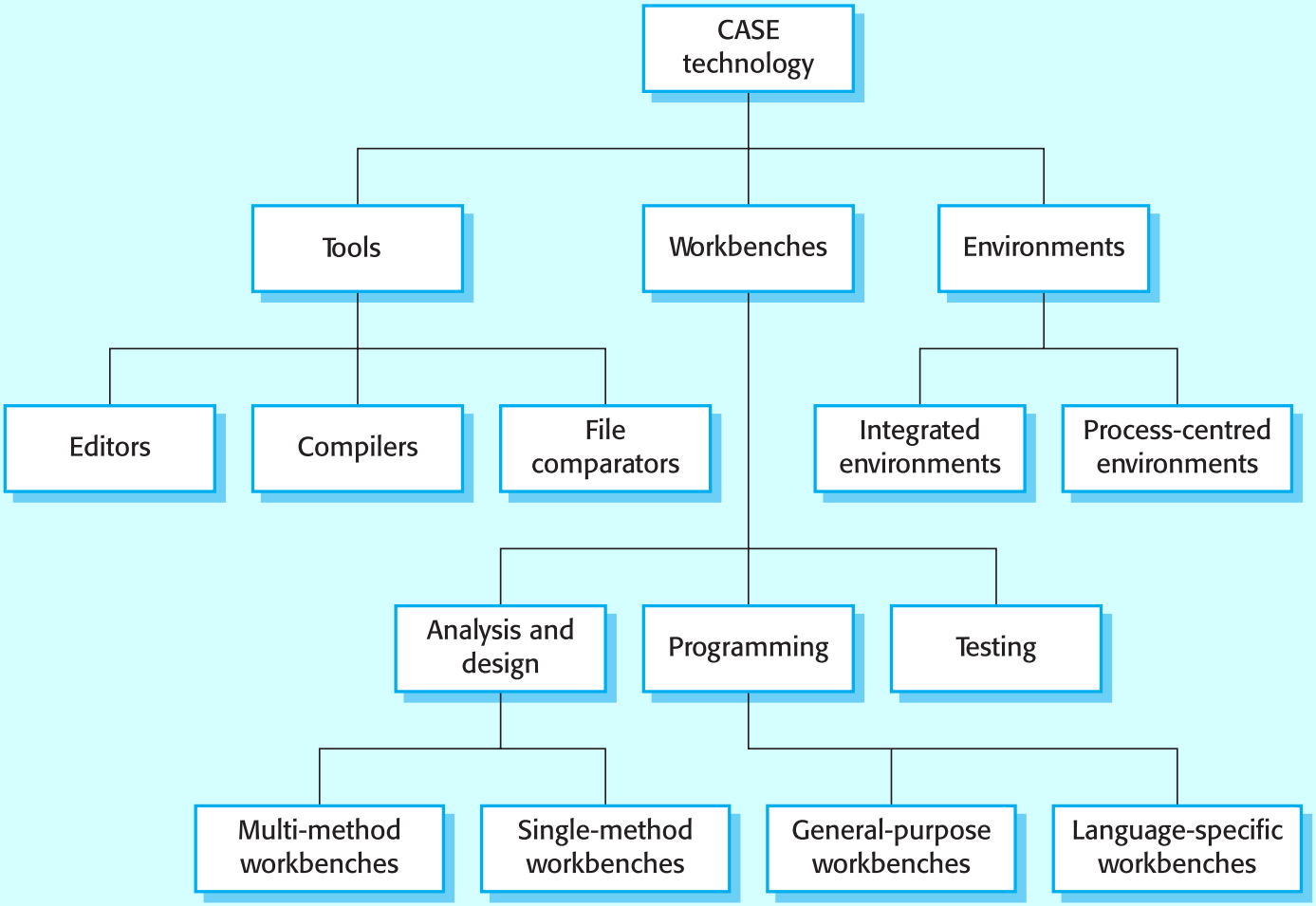
Workbenches

- 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.

Tools, Workbenches, Environments



Key Points

- Software processes are the activities involved in producing and evolving a software system
- Software process models are abstract representations of these processes
- General activities are specification, design and implementation, validation and evolution
- Generic process models describe the organisation of software processes. Examples include the waterfall model, evolutionary development and component-based software engineering
- 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
- Rational Unified Process is a generic process model that separates activities from phases
- CASE technology supports software process activities

Summary

- Successful SW process reuses what is possible, develops incrementally to yield early feedback and experience; reduces risk
- Specify, document requirements, goal, priorities to maximize SW development success
- Use automated tools, and develop throw-away and lomg-range tools
- Code of SW Ethics published, with guide for SWE to adhere to the stated principles

References

1. SW development process: https://en.wikipedia.org/wiki/Software_development_process
2. Good illustrations: https://medium.com/omarelgabrys-blog/software-engineering-software-process-and-software-process-models-part-2-4a9d06213fdc