Chapter 4
Contemporary Software Processes

Objectives

- To explain the Rational Unified Process model
- To introduce CASE technology to support software process activities

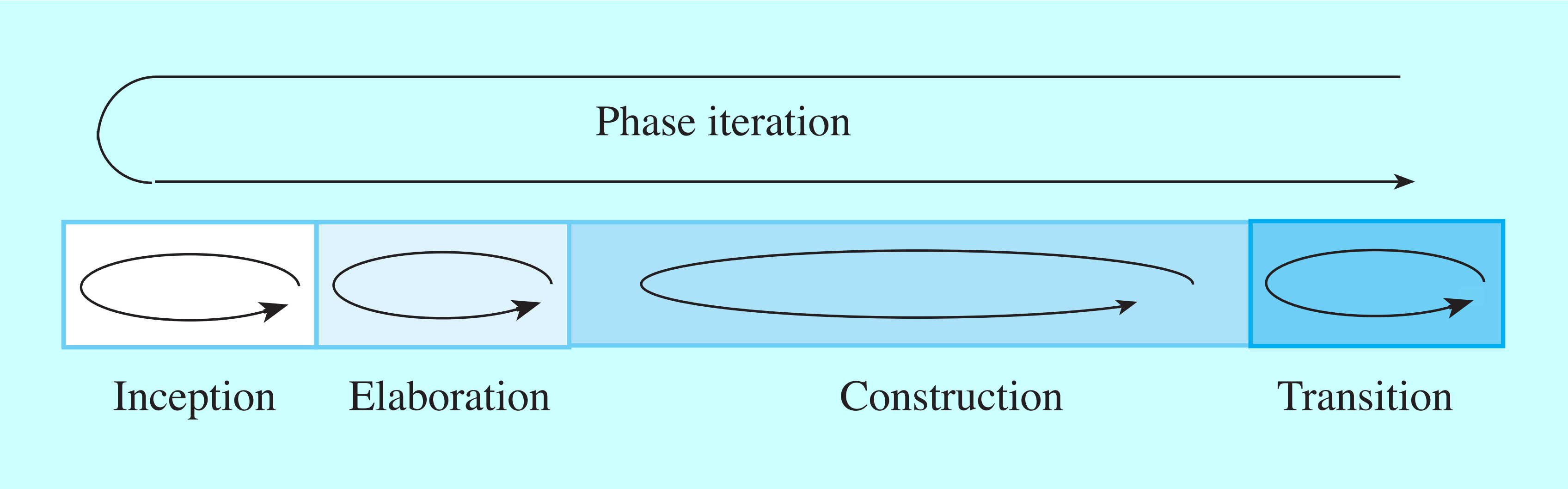
Topics covered

- The Rational Unified Process
- Computer-aided software engineering

The Rational Unified Process

- 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.

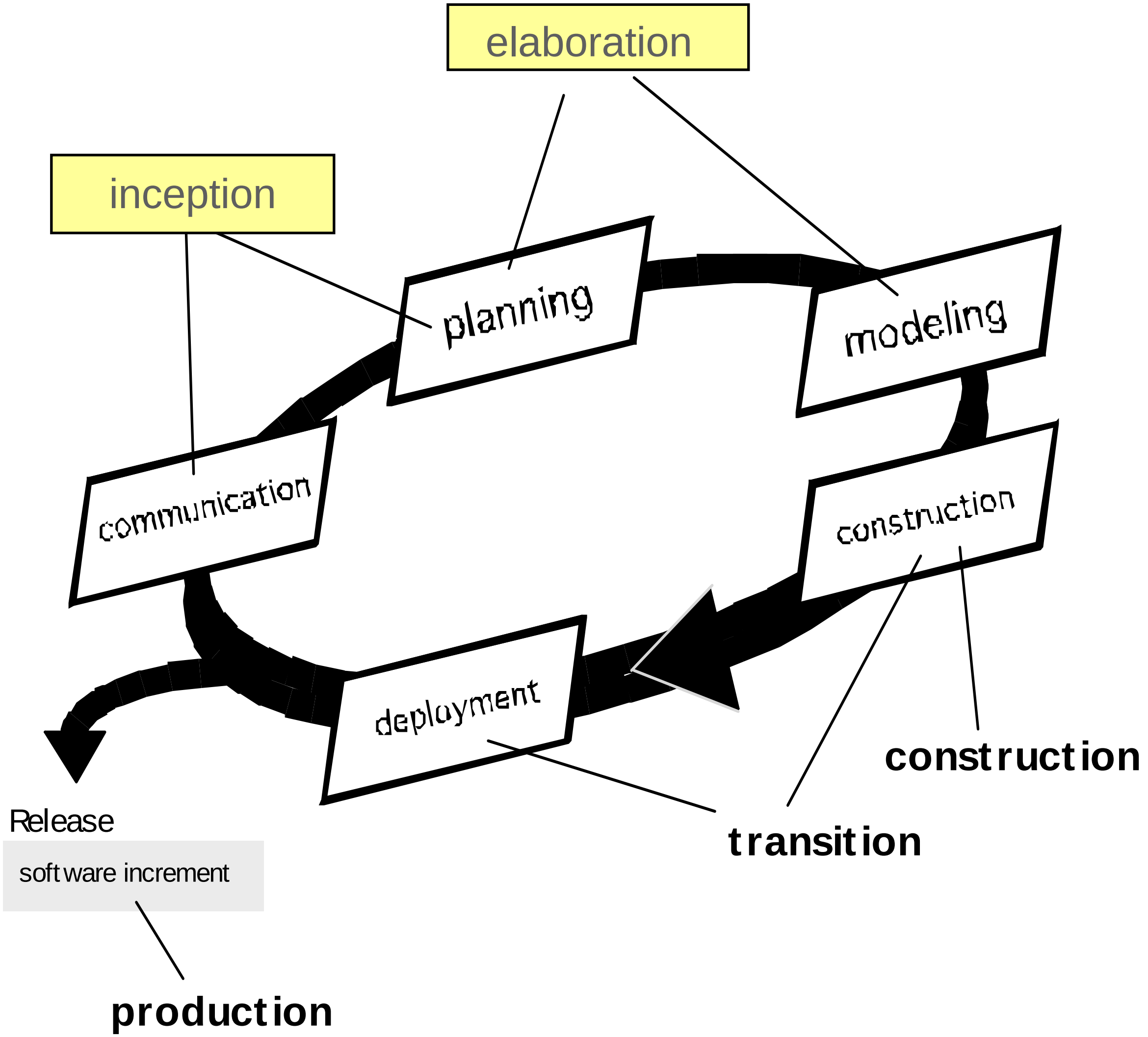
RUP phase model



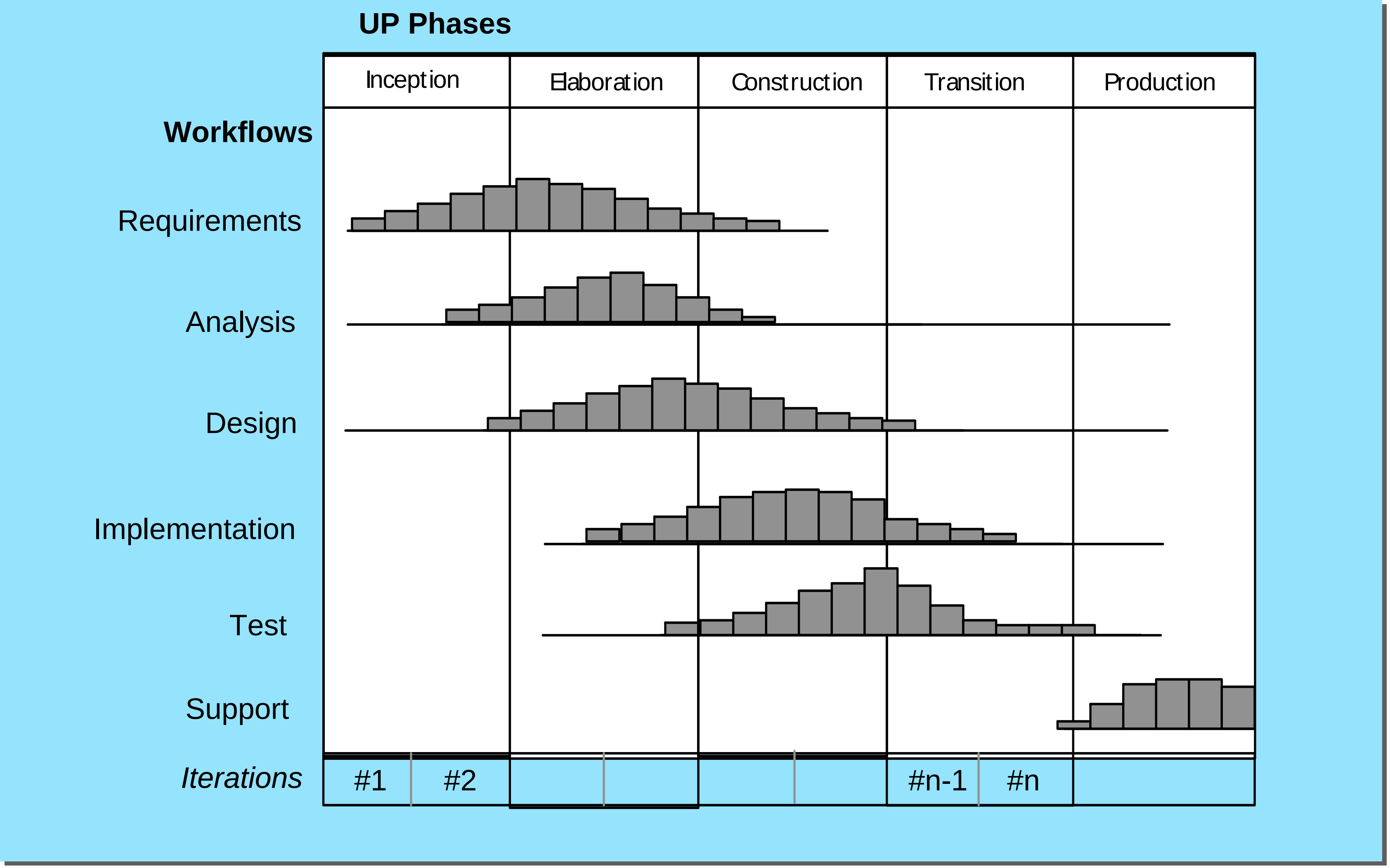
RUP phases

- 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

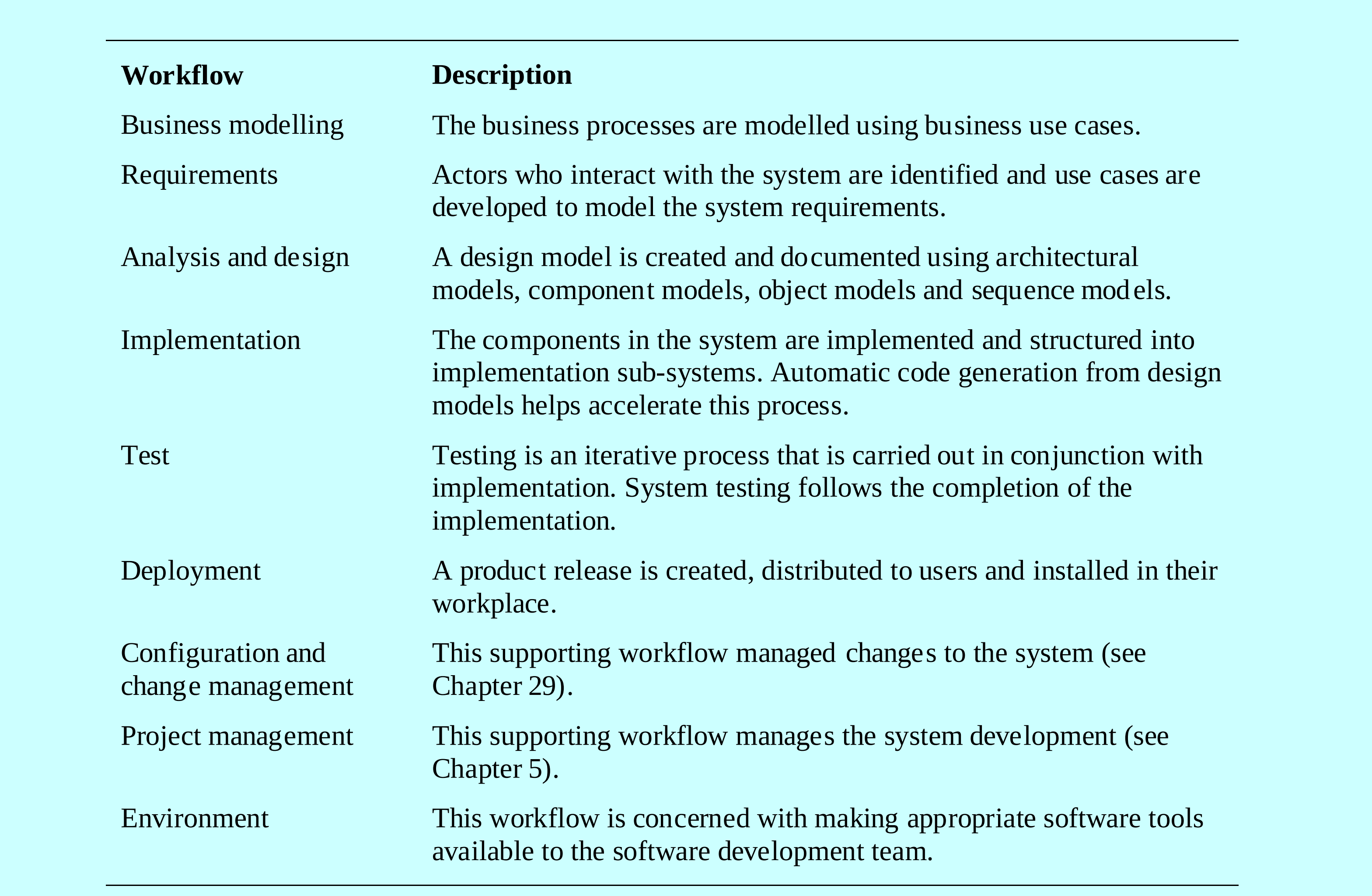
Transition phase

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

RUP good practice

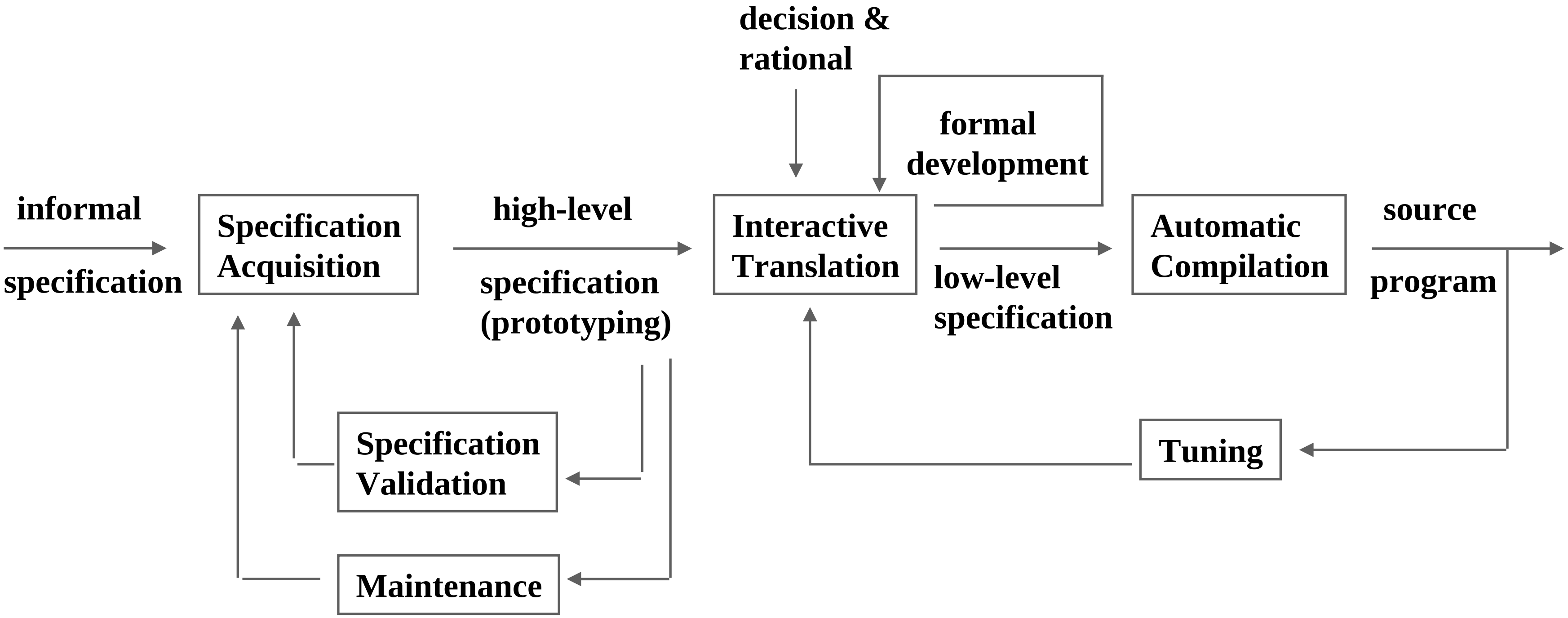
- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

Static workflows

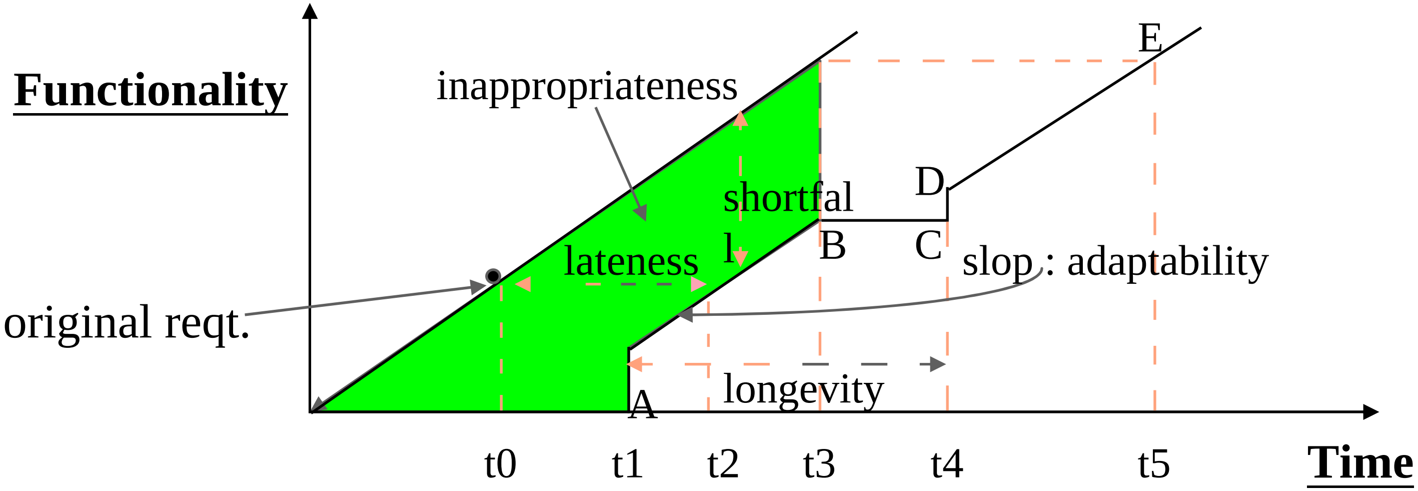


Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural 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



Comparing Various Process Models



waterfall
model

O : (t_0) original reqt.

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

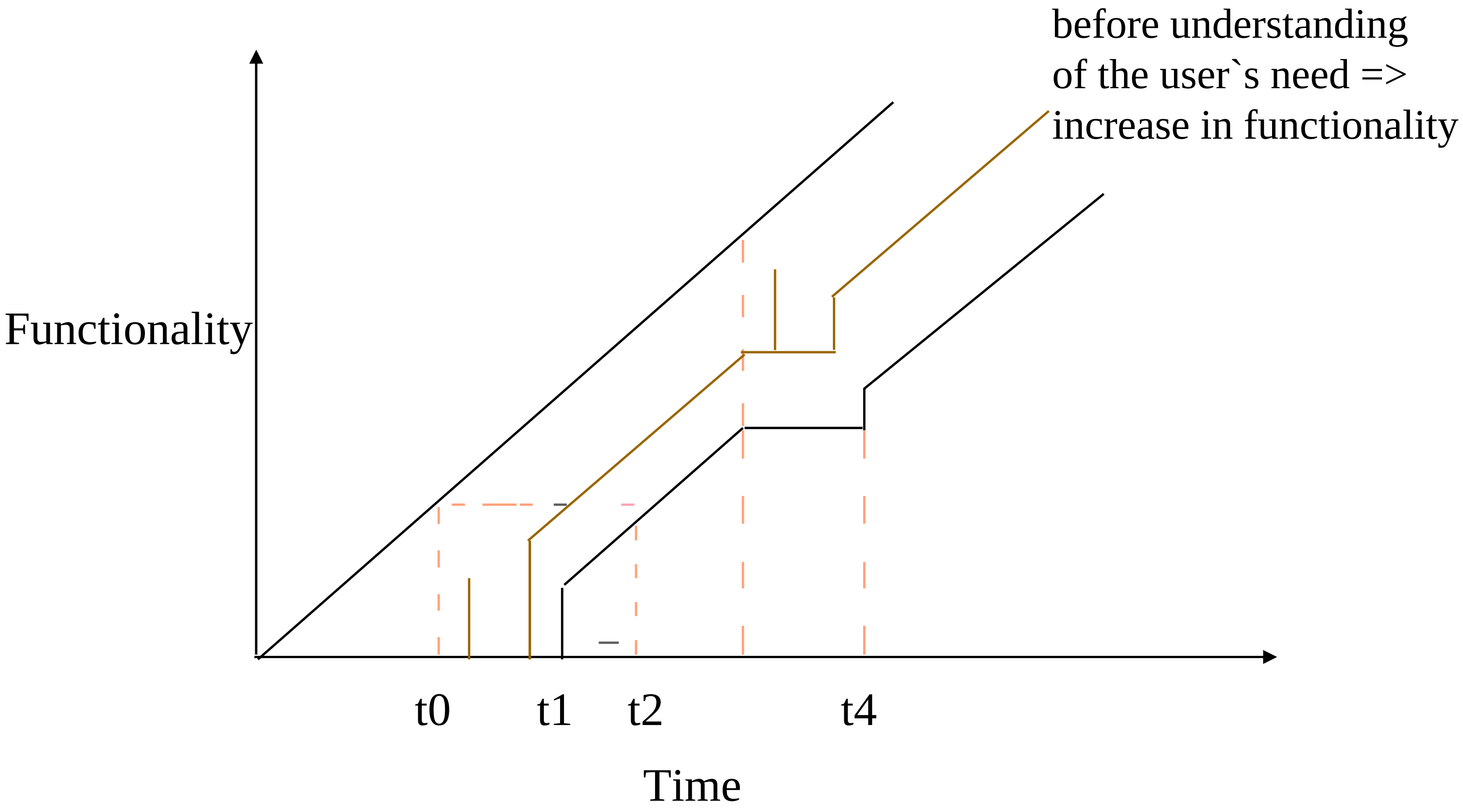
A - B : undergo a series of enhancements.

B - D : the cost of enhancements increase, to build a new system.

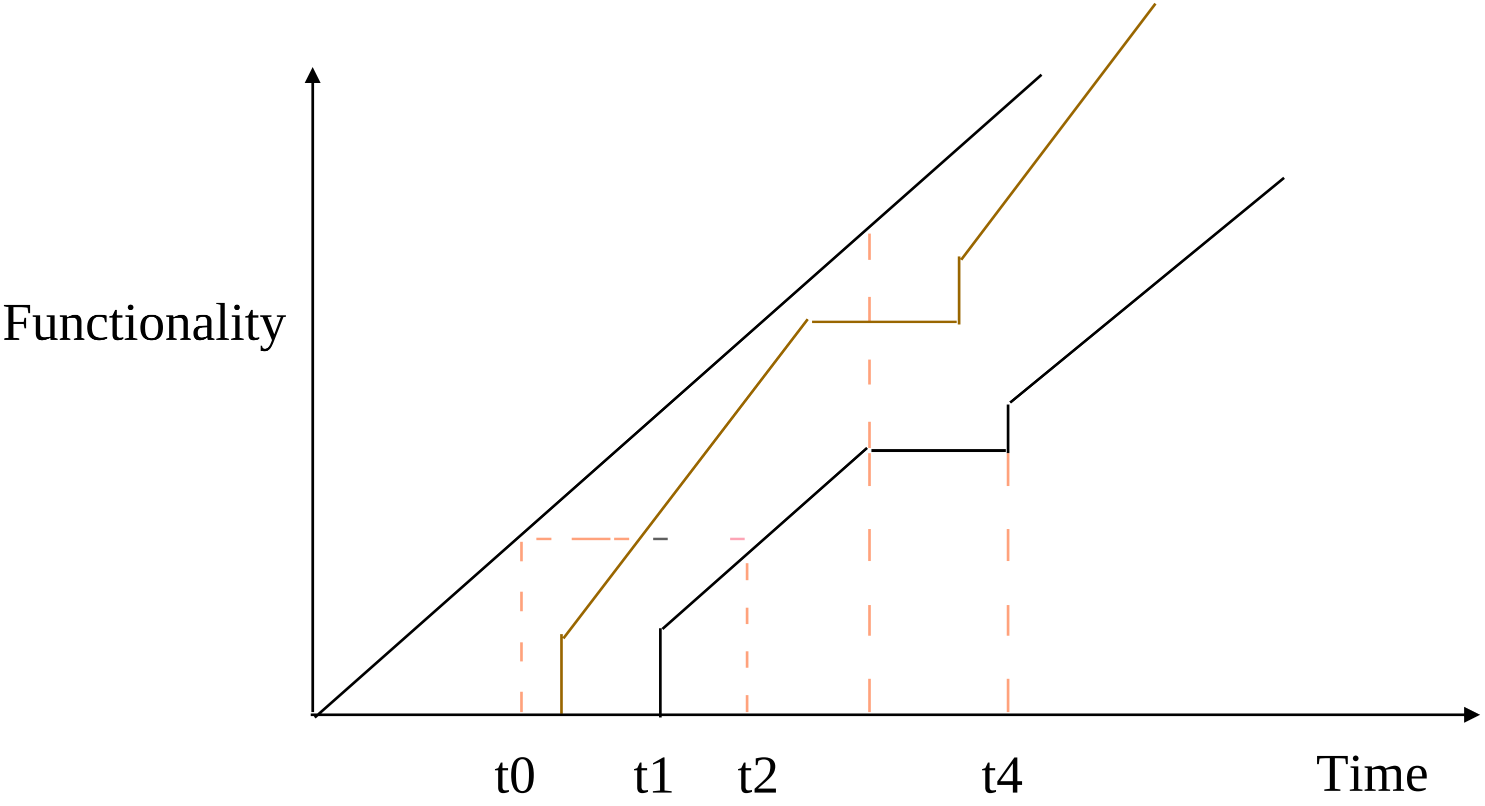
stop at t4.

* cycle repeat itself

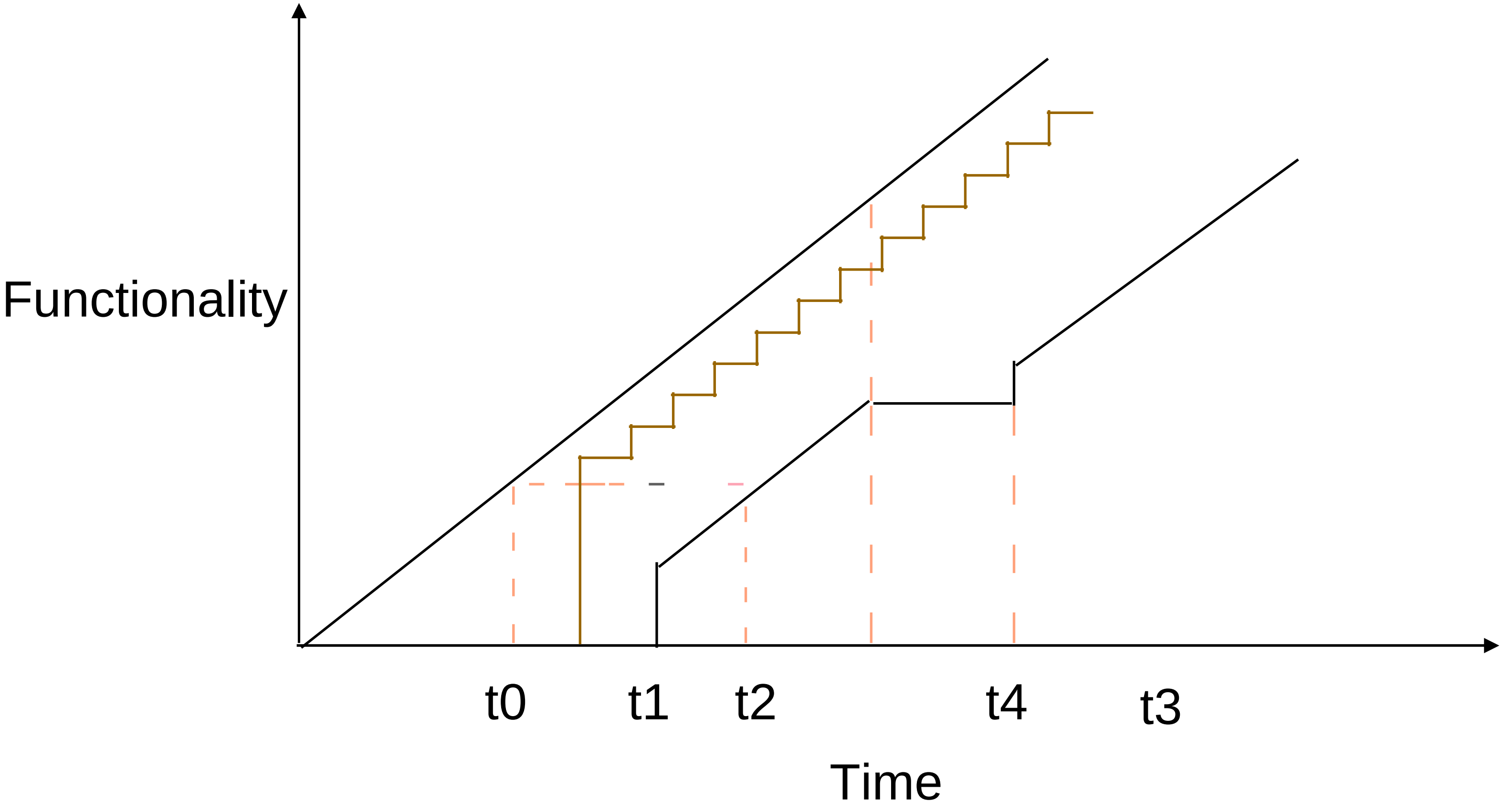
Throwaway Prototyping and Spiral Model



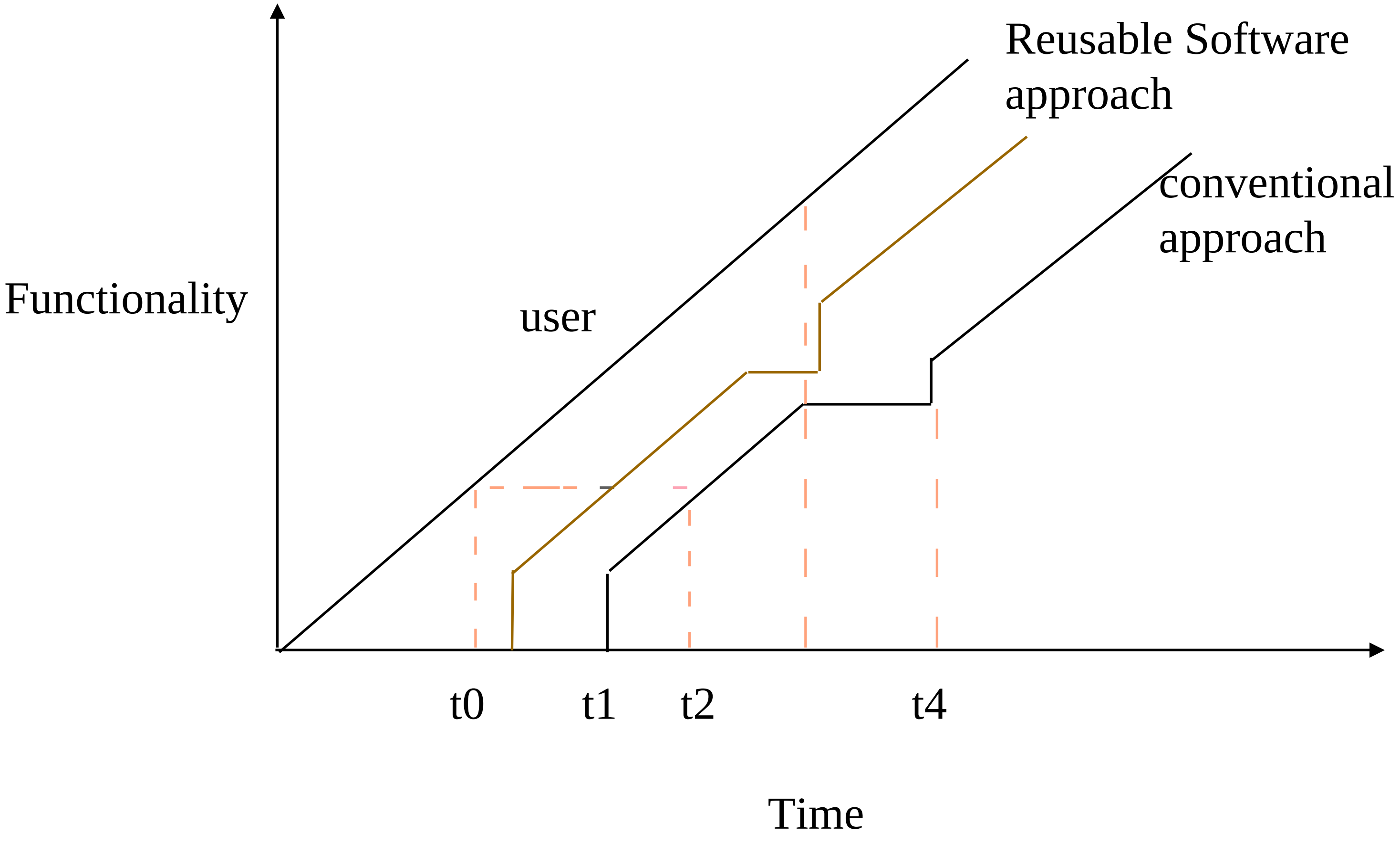
Evolutionary Prototyping



Automated Software Synthesis



Reusable Software versus Conventional



Computer-Aided Software Engineering

- 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.

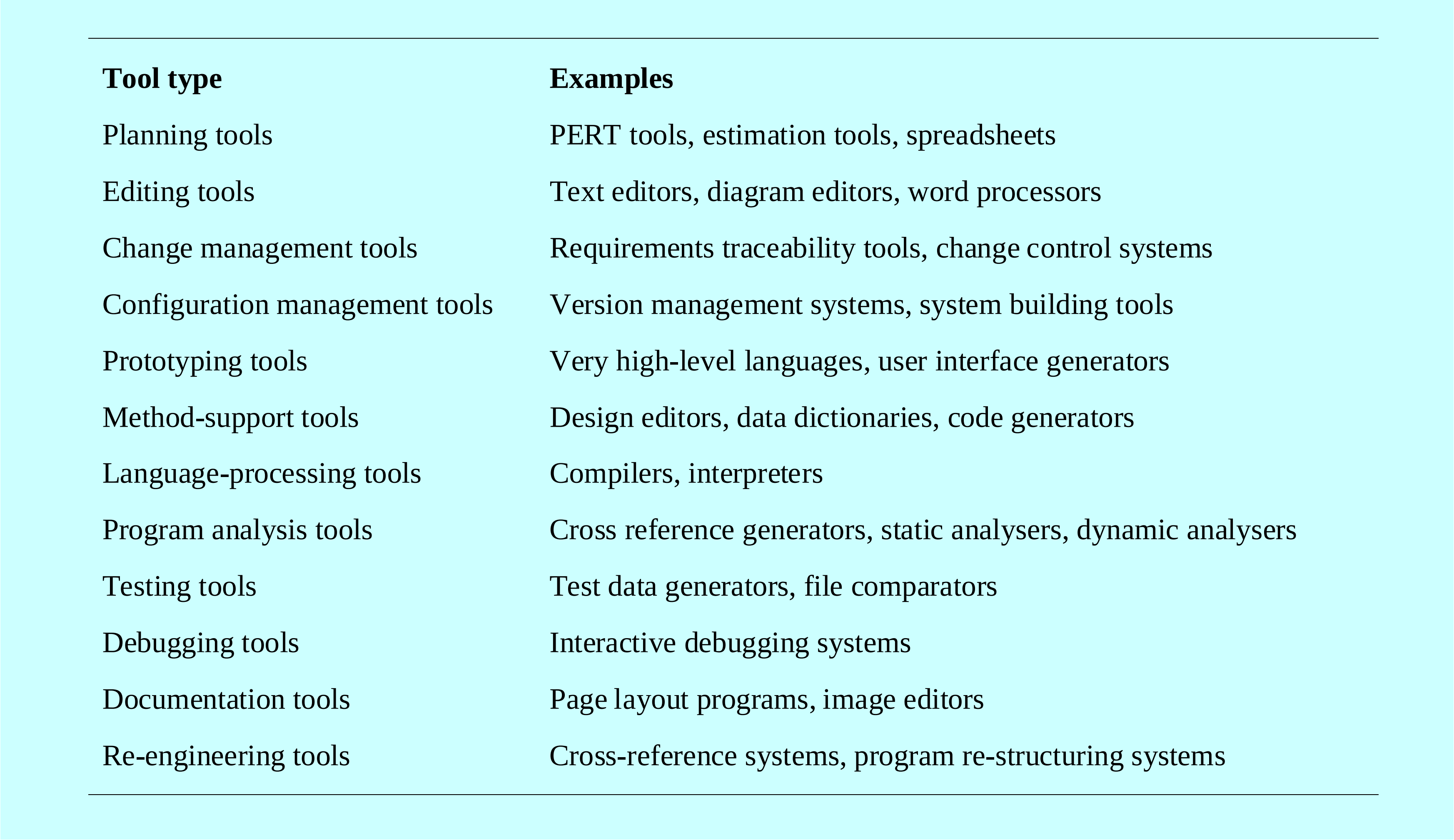
CASE technology

- CASE technology has led to significant improvements in the software process. However, these are not the order of magnitude improvements that were once predicted
  - Software engineering requires creative thought - this is not readily automated;
  - 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

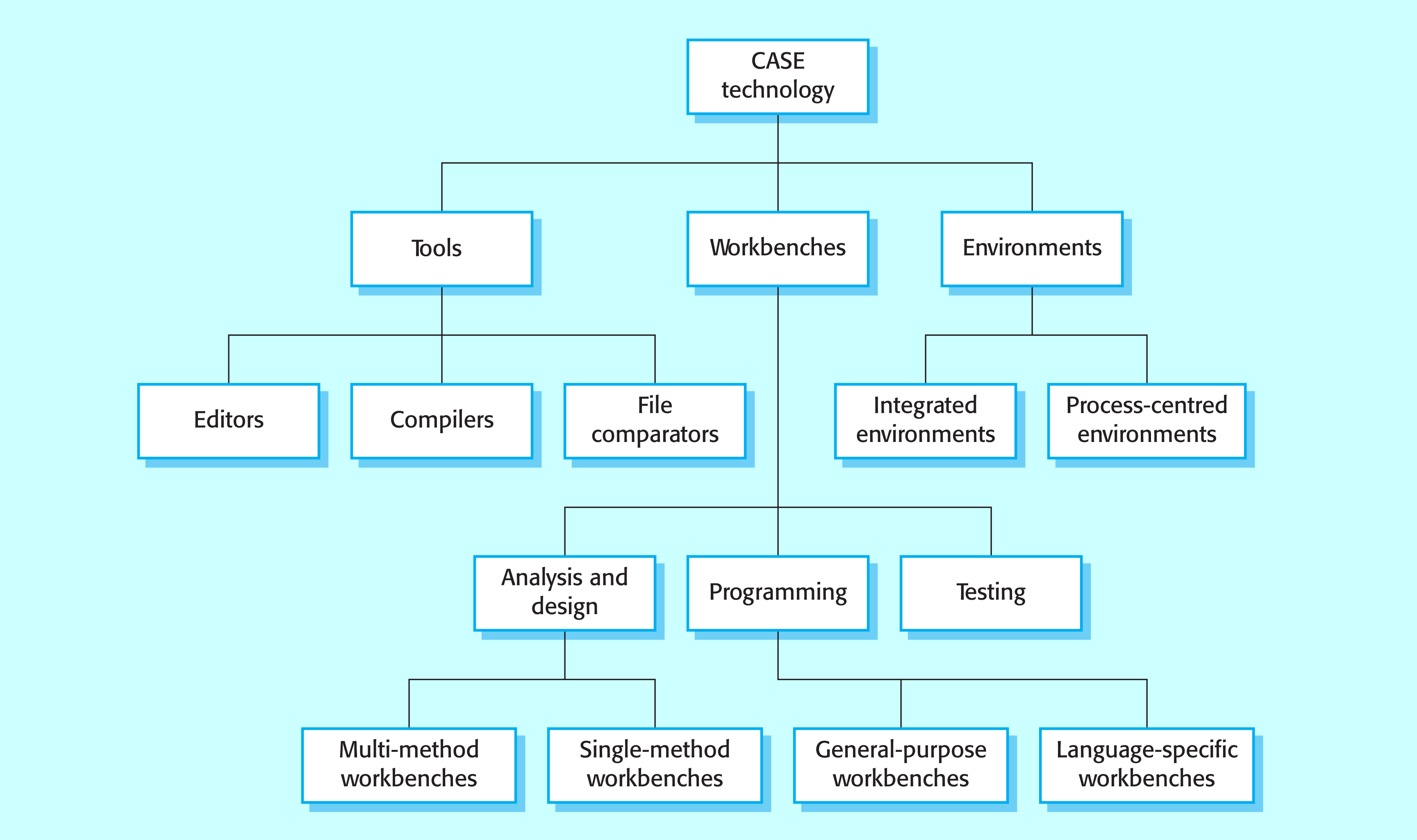
Activity-based tool classification

	Specification	Design	Implementation	Verification and Validation
Re-engineering 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.

Tools, workbenches, environments


CASE technology
Tools
Workbenches
Environments
Editors
Compilers
File comparators
Integrated environments
Process-centred environments
Analysis and design
Programming
Testing
Multi-method workbenches
Single-method workbenches
General-purpose workbenches
Language-specific workbenches