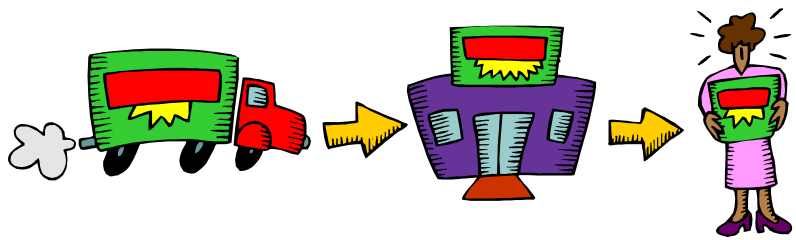
Kỹ thuật phần mềm

Các quy trình phần mềm



Các chủ đề

- Các mô hình quy trình phần mềm
- Process iteration (quá trình lặp lại)
- Các hoạt động quy trình
- Đọc thêm
  - Rational Unified Process
  - CASE – Computer-aided software engineering

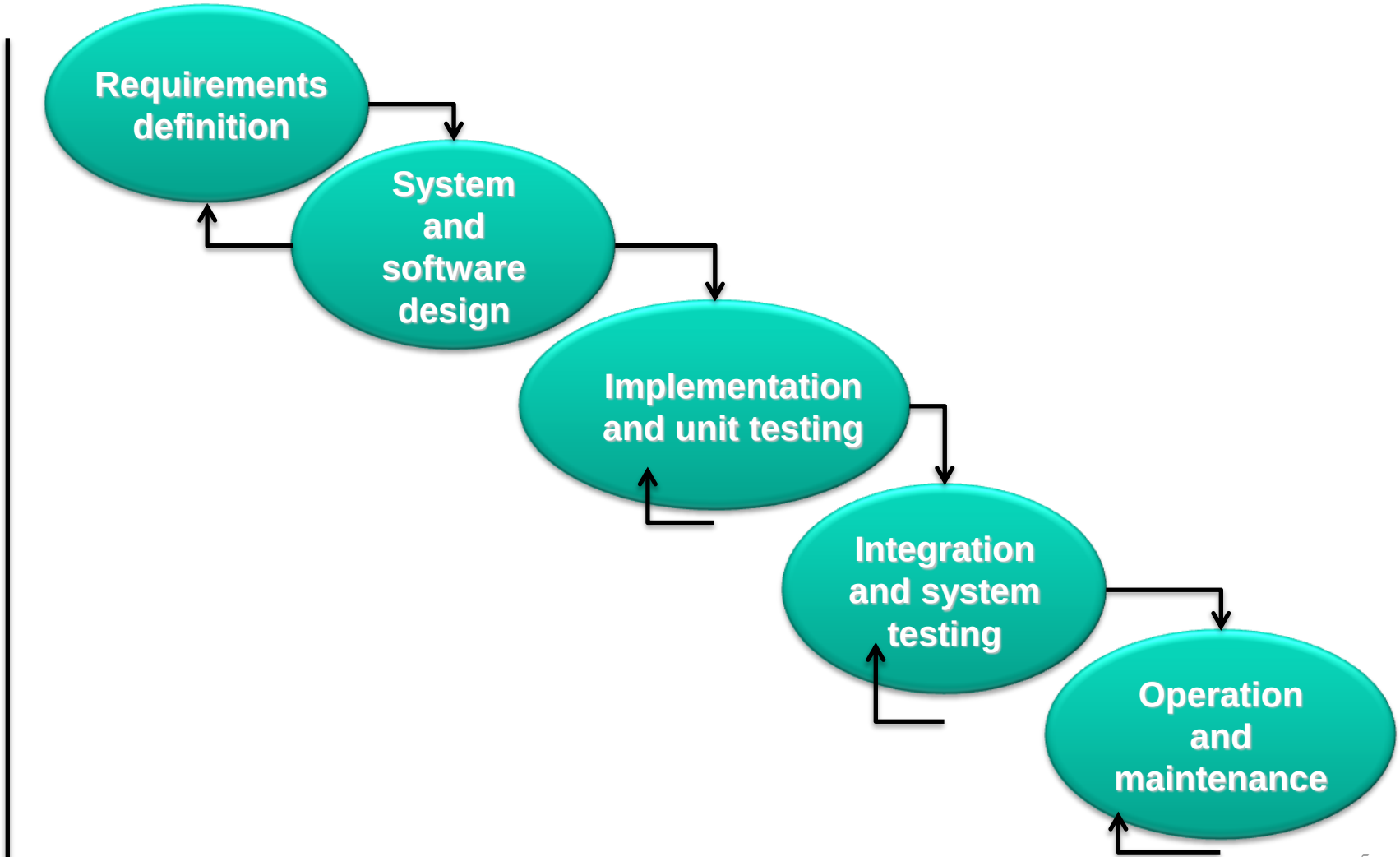
Quy trình phần mềm

- Quy trình phần mềm (software process) là một tập các hoạt động cần thiết để phát triển một hệ thống phần mềm:
  - Đặc tả - Specification;
  - Thiết kế - Design;
  - Phát triển – Development;
  - Kiểm thử - Testing;
  - Kiểm định – Verification;
  - Thẩm định - Validation;
  - Tiến hóa - Evolution.
- Một mô hình quy trình phần mềm là một biểu diễn trừu tượng của một quy trình.
  - Một mô tả về một quy trình từ một góc độ nào đó.

Các mô hình quy trình phần mềm tổng quát

- Mô hình thác nước – The waterfall model
  - Tách biệt các pha đặc tả và phát triển.
- Phát triển tiến hóa – Evolutionary development
  - Các hoạt động đặc tả, phát triển và thẩm định xen kẽ nhau.
- CNPM dựa thành phần – Component-based SE
  - Hệ thống được lắp ráp từ các thành phần sẵn có.
- Có nhiều biến thể của các mô hình này (kết hợp các mô hình khác nhau)
  - v.d. hoạt động phát triển dùng quy trình kiểu thác nước nhưng hoạt động đặc tả được làm mịn qua nhiều bước cho đến khi đạt được một thiết kế cài đặt được.

Mô hình thác nước



Các pha trong mô hình thác nước

- Phân tích và định nghĩa yêu cầu
- Thiết kế hệ thống và phần mềm
- Cài đặt và kiểm thử đơn vị
- Tích hợp và kiểm thử hệ thống
- Vận hành và bảo trì

Nhược điểm chính của mô hình thác nước là khó khăn của việc sửa đổi sau khi quy trình đã vào guồng. Pha này phải được hoàn tất trước khi bước vào pha tiếp theo.

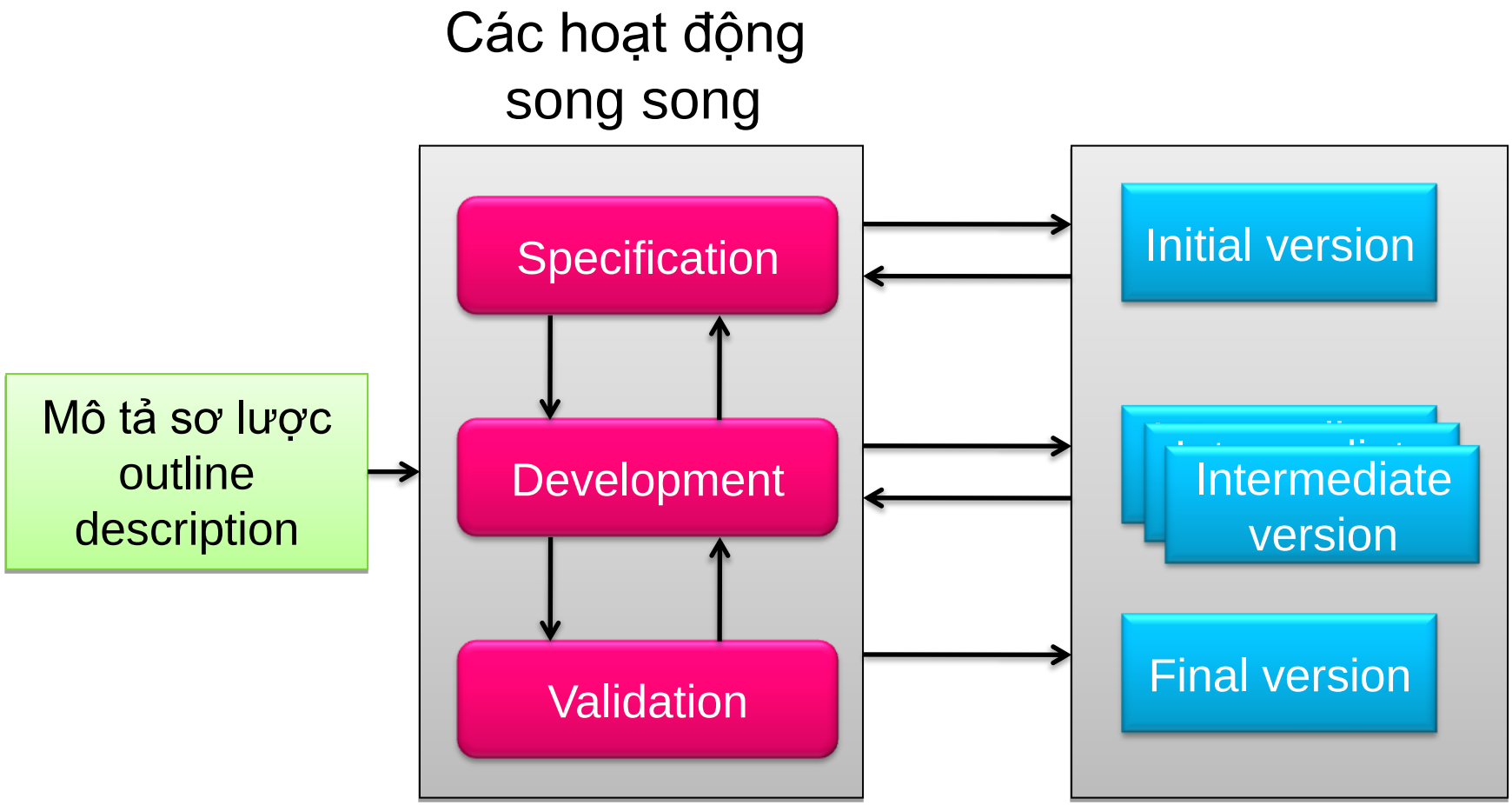
Các vấn đề của mô hình thác nước

- Khó đáp ứng việc khách hàng thay đổi yêu cầu.
  - do việc phân dự án thành các giai đoạn tách biệt
- Chỉ thích hợp khi các yêu cầu được hiểu rõ và ít có thay đổi trong quy trình phát triển.
  - Ít hệ thống doanh nghiệp có các yêu cầu ổn định ít thay đổi theo thời gian.
- Chủ yếu dùng cho các dự án hệ thống lớn, khi một hệ thống được phát triển tại các địa điểm khác nhau.

Phát triển tiến hóa

- Phát triển thăm dò (exploratory development)
  - Mục đích là làm việc với khách hàng và từng bước phát triển (evolve) từ một đặc tả sơ lược ban đầu tới một hệ thống là sản phẩm cuối cùng.
    - nên bắt đầu từ một bộ yêu cầu được hiểu rõ và bổ sung các tính năng mới khi khách hàng đề xuất.
- Các phiên bản thử nghiệm dùng tạm (throw-away prototyping)
  - Mục đích để hiểu các yêu cầu hệ thống.
    - nên bắt đầu từ bộ yêu cầu không được hiểu rõ để có thể làm rõ đâu là cái thực sự được yêu cầu.

Phát triển tiến hóa



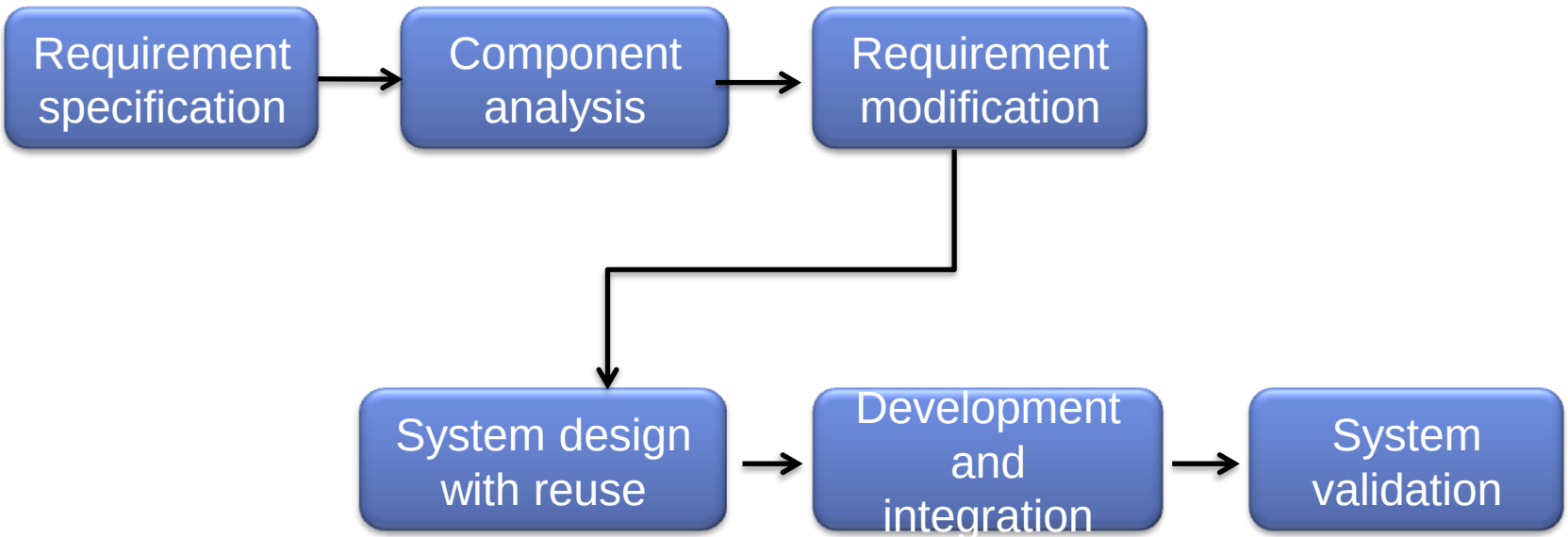
Phát triển tiến hóa

- Vấn đề
  - Tính quy trình không thể hiện rõ ràng;
  - Các hệ thống thường được cấu trúc tồi;
  - Đòi hỏi các kỹ năng đặc biệt
    - ví dụ kĩ năng dùng các ngôn ngữ cho việc xây dựng cấp tốc các phiên bản thử nghiệm (rapid prototyping)
- Ứng dụng
  - cho các hệ thống kích thước nhỏ và trung bình;
  - cho các phần của hệ thống lớn (chẳng hạn giao diện người dùng);
  - cho các hệ thống chỉ dùng trong thời gian ngắn.

CNPM dựa thành phần

- dựa trên việc tái sử dụng một cách có hệ thống
  - các hệ thống được tích hợp từ các thành phần có sẵn hoặc các hệ thống COTS (Commercial-off-the-shelf – sẵn sàng để người dùng mua về cài vào máy)
- Các pha trong quy trình
  - Phân tích thành phần (component analysis);
  - Sửa yêu cầu (requirements modification);
  - Thiết kế hệ thống trong đó tái sử dụng;
  - Phát triển và tích hợp.
- Cách tiếp cận này ngày càng được sử dụng nhiều, khi các chuẩn thành phần đã bắt đầu xuất hiện.

Phát triển hướng đến tái sử dụng



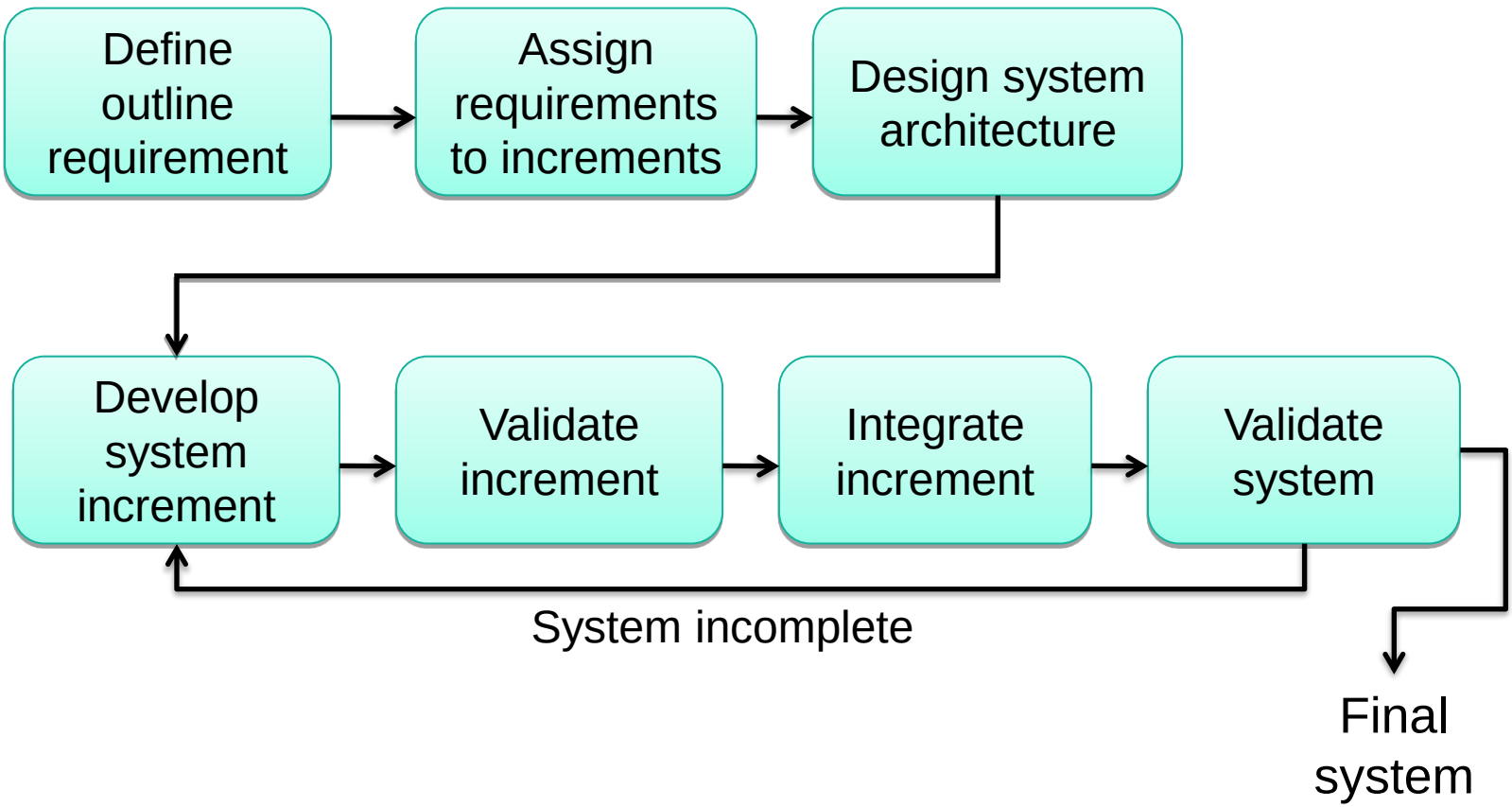
Process iteration

- Các yêu cầu hệ thống LUÔN LUÔN thay đổi trong quá trình thực hiện một dự án, do đó trong quy trình cho các hệ thống lớn luôn có việc lặp lại quy trình (process iteration) mà trong đó các giai đoạn đã qua được thực hiện lại.
- Việc lặp lại có thể được áp dụng cho bất kì mô hình quy trình tổng quát nào.
- Hai cách tiếp cận (có liên quan)
  - Chuyển giao tăng dần – Incremental delivery;
  - Phát triển kiểu xoắn ốc – Spiral development.

Chuyển giao tăng dần

- Thay vì giao cho khách hàng sản phẩm hệ thống trong một lần chuyển giao duy nhất, việc phát triển và chuyển giao được chia thành các đợt, mỗi đợt giao một phần của chức năng được yêu cầu.
- Các yêu cầu của người dùng là các yêu cầu được ưu tiên, và các yêu cầu có độ ưu tiên cao nhất cần được đáp ứng ngay từ các đợt đầu tiên.
- Một khi hoạt động phát triển cho một đợt đã bắt đầu, bộ yêu cầu dùng được đóng băng, các thay đổi đối với yêu cầu được dành cho đợt sau.

Phát triển tăng dần



Ưu điểm của phát triển tăng dần

- Customer value can be delivered with each increment so system functionality is available earlier.
- Các đợt đầu đóng vai trò phiên bản thử nghiệm (prototype) để hỗ trợ việc làm rõ bộ yêu cầu cho các đợt sau.
- Rủi ro thấp đối với thất bại toàn bộ dự án.
- Các dịch vụ hệ thống có ưu tiên cao nhất có xu hướng được kiểm thử nhiều nhất.

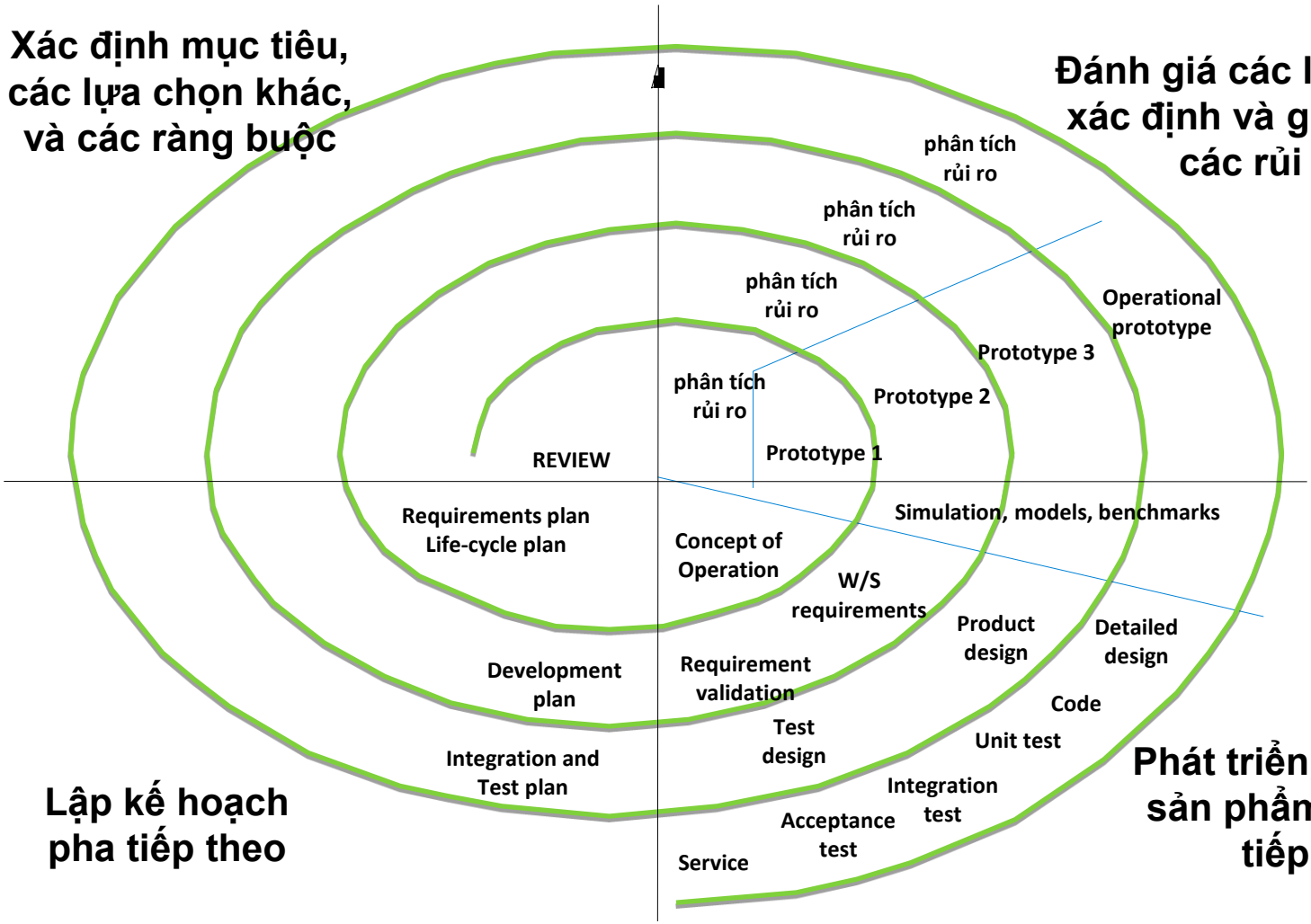
Lập trình cực đoan
Extreme programming

- Một cách tiếp cận đối với hoạt động phát triển dựa trên việc phát triển và chuyển giao các đợt với bổ sung rất nhỏ về chức năng (very small increments of functionality).
- Dựa vào việc liên tục cải tiến mã chương trình, người dùng tham gia nhóm phát triển, và việc lập trình theo cặp.
- *Chi tiết nằm trong bài sau*

Phát triển kiểu xoắn ốc

- Quy trình được biểu diễn dưới dạng xoắn ốc thay vì một chuỗi các hoạt động với các bước quay lui.
- Mỗi vòng trong đường xoắn ốc đại diện cho một pha trong quy trình.
- Không có các pha cố định như pha đặc tả hay pha thiết kế - các vòng xoắn được lựa chọn tùy theo nhu cầu.
- Các rủi ro được đánh giá một cách tường minh và được giải quyết trong suốt quy trình.

Mô hình xoắn ốc



Các phân khu trong mô hình xoắn ốc

- Đặt mục tiêu
  - xác định các mục tiêu cụ thể của pha.
- Đánh giá và giảm rủi ro
  - đánh giá các rủi ro và sắp xếp các hoạt động để giảm các rủi ro chính yếu.
- Phát triển và thẩm định
  - Chọn một mô hình phát triển cho hệ thống, đây có thể là một trong các mô hình tổng quát.
- Lập kế hoạch
  - Review dự án và lập kế hoạch cho pha tiếp theo của đường xoắn ốc.

Các hoạt động quy trình

Đặc tả

Thiết kế và cài đặt

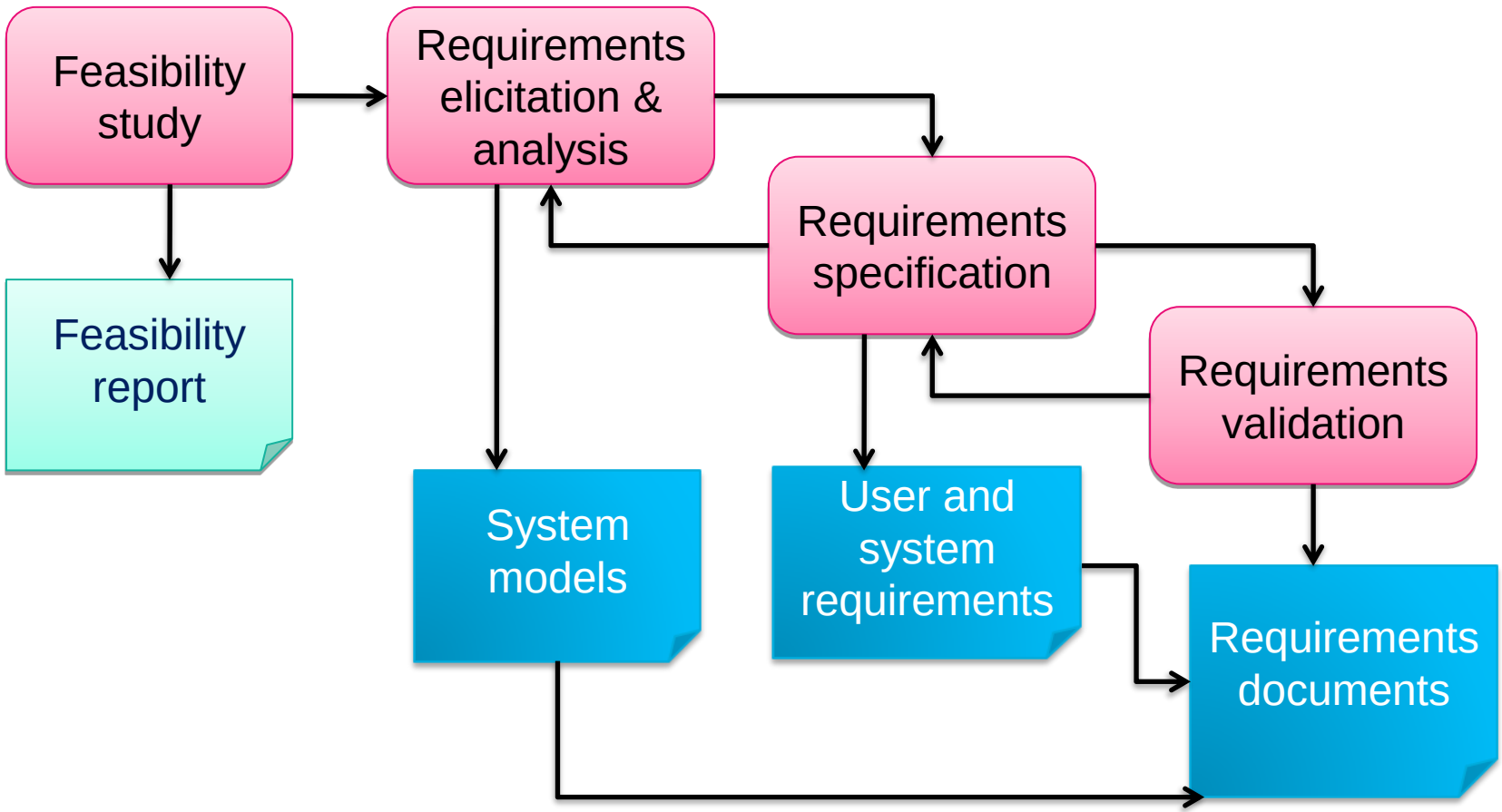
Thẩm định

Tiến hóa

Đặc tả phần mềm

- Quy trình thiết lập danh sách các dịch vụ được yêu cầu và các ràng buộc đối với hoạt động của hệ thống và việc phát triển hệ thống.
- Quy trình kĩ nghệ yêu cầu
  - Nghiên cứu tính khả thi – Feasibility study;
  - Thu thập và phân tích yêu cầu – Requirements elicitation and analysis;
  - Đặc tả yêu cầu – Requirements specification;
  - Thẩm định yêu cầu – Requirements validation.

Quy trình kĩ nghệ yêu cầu



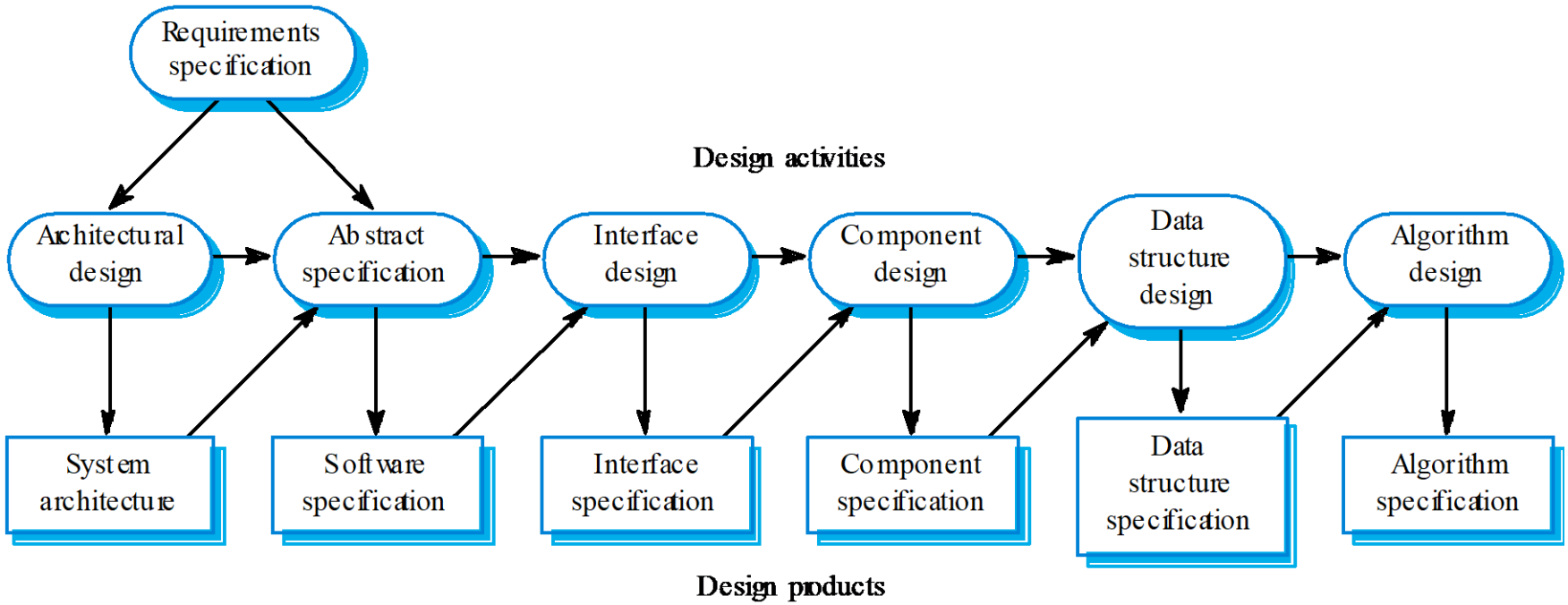
Thiết kế và cài đặt phần mềm

- Quy trình biến đổi từ đặc tả hệ thống thành một hệ thống chạy được.
- Thiết kế phần mềm
  - Thiết kế một cấu trúc phần mềm mà hiện thực hóa được đặc tả;
- Cài đặt
  - Dịch cấu trúc đó thành một chương trình chạy được;
- Các hoạt động thiết kế và cài đặt có quan hệ chặt chẽ với nhau và có thể xen kẽ.

Các hoạt động quy trình thiết kế

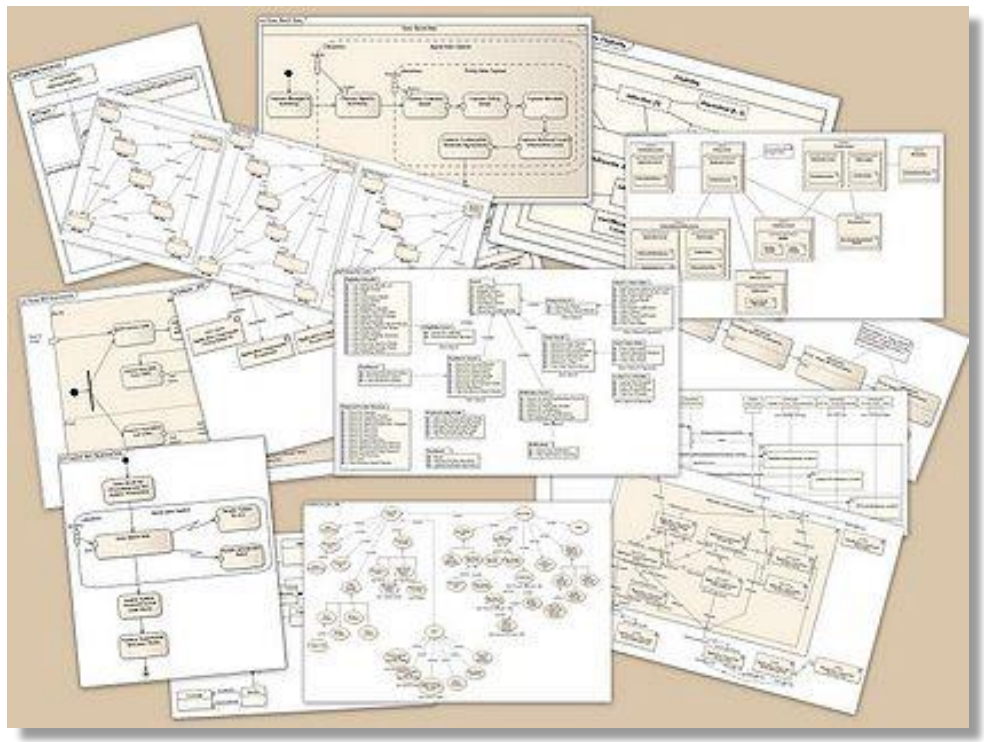
- Thiết kế kiến trúc – Architectural design
- Đặc tả trừu tượng – Abstract specification
- Thiết kế giao diện – Interface design
- Thiết kế thành phần – Component design
- Thiết kế cấu trúc dữ liệu – Data structure design
- Thiết kế thuật toán – Algorithm design

Quy trình thiết kế phần mềm



Các phương pháp có cấu trúc

- Các cách tiếp cận một cách có hệ thống đối với việc phát triển một thiết kế phần mềm.
- Thiết kế thường được ghi lại dưới dạng một tập các mô hình đồ họa.
- Các mô hình có thể có:
  - Object model;
  - Sequence model;
  - State transition model;
  - Structural model;
  - Data-flow model.

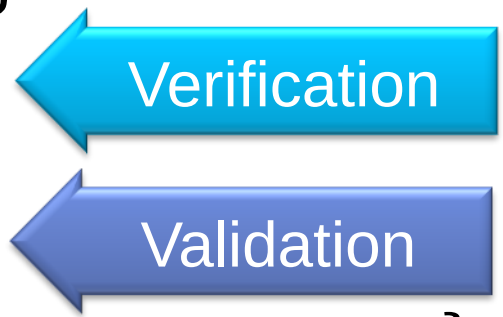


Lập trình và tìm lỗi

- là hoạt động dịch một thiết kế thành một chương trình và loại bỏ lỗi trong chương trình đó.
- lập trình là một hoạt động cá nhân – không có quy trình lập trình tổng quát.
- trong quy trình tìm lỗi (debugging process), lập trình viên thực hiện việc kiểm thử chương trình để phát hiện lỗi trong chương trình và loại bỏ các lỗi đó.

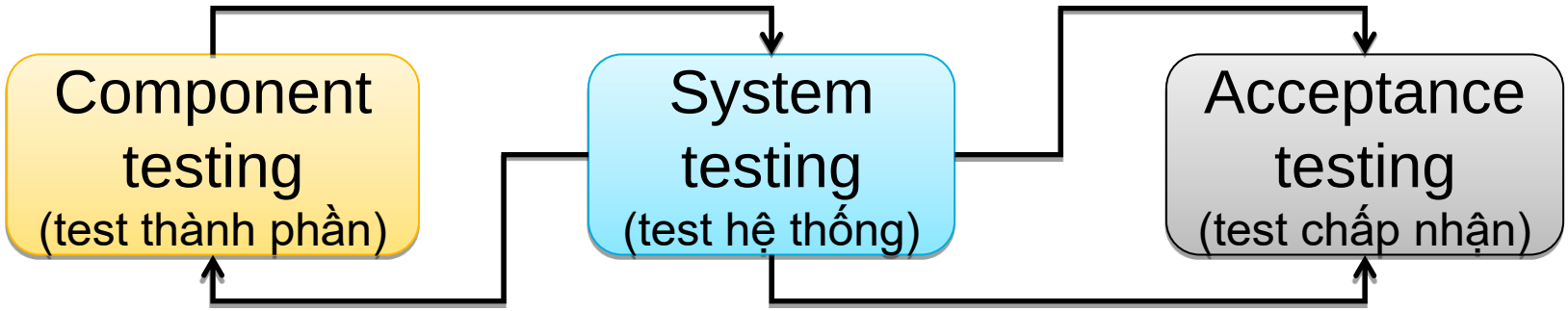
Thẩm định phần mềm

- Kiểm định (verification) và thẩm định (validation) nhằm chứng tỏ rằng một hệ thống
  - tuân theo **đặc tả** của nó, và
  - thỏa mãn các yêu cầu của **khách hàng** hệ thống.



- bao gồm các quy trình checking và review và kiểm thử hệ thống (system testing).
  - Việc kiểm thử bao gồm việc chạy hệ thống với các test case được dẫn xuất từ đặc tả.

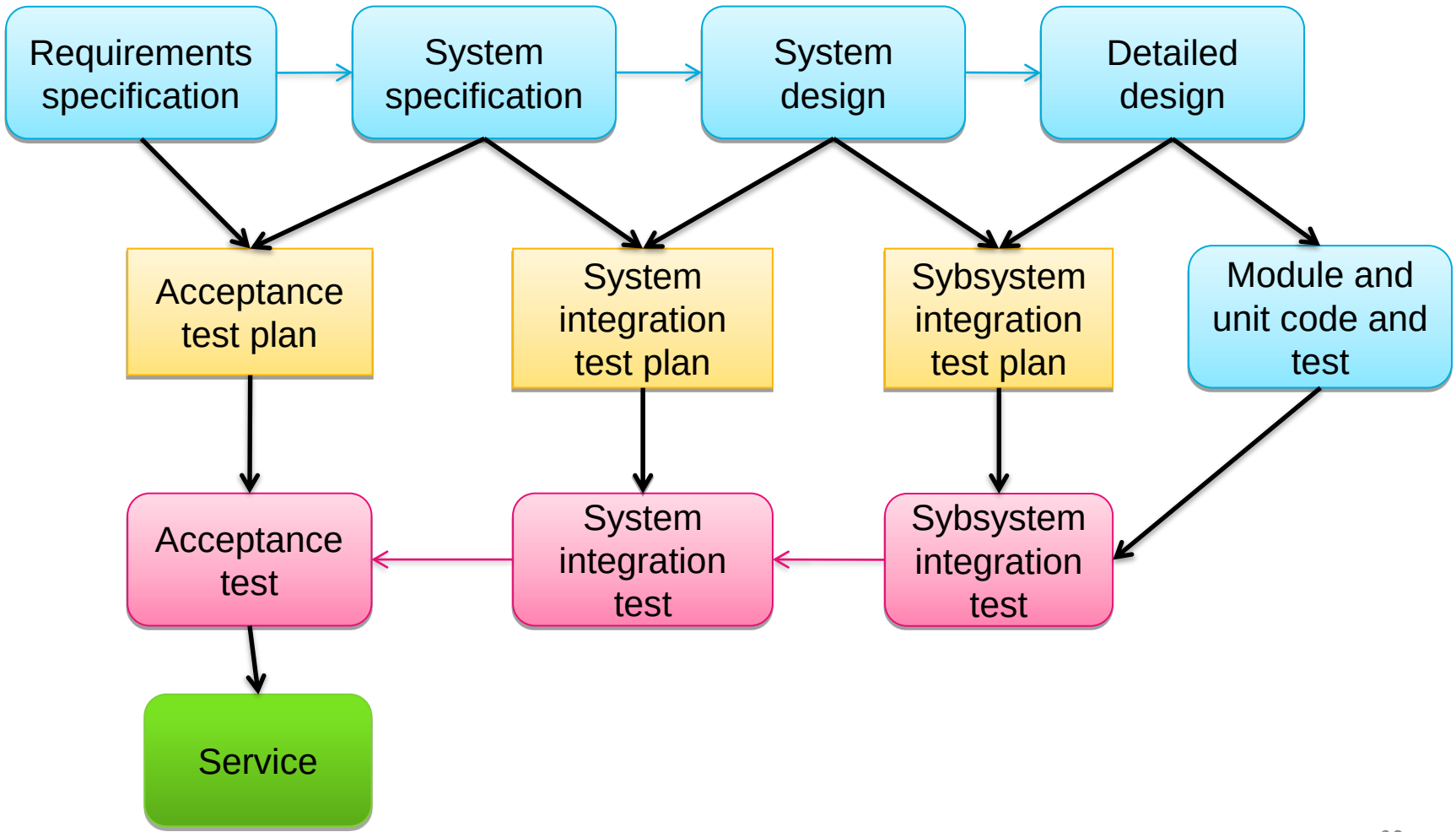
Quy trình kiểm thử



Các giai đoạn kiểm thử

- Kiểm thử thành phần hoặc đơn vị (component or unit testing)
  - Các thành phần được test một cách độc lập;
  - Thành phần: hàm hoặc đối tượng hoặc các nhóm hàm/đối tượng có tính gắn kết cao.
- Kiểm thử hệ thống (system testing)
  - Kiểm thử toàn bộ hệ thống – chạy cả chương trình.
- Kiểm thử chấp nhận (acceptance testing)
  - Test bằng dữ liệu của khách hàng để kiểm tra xem hệ thống có thỏa mãn nhu cầu của khách hàng hay không.

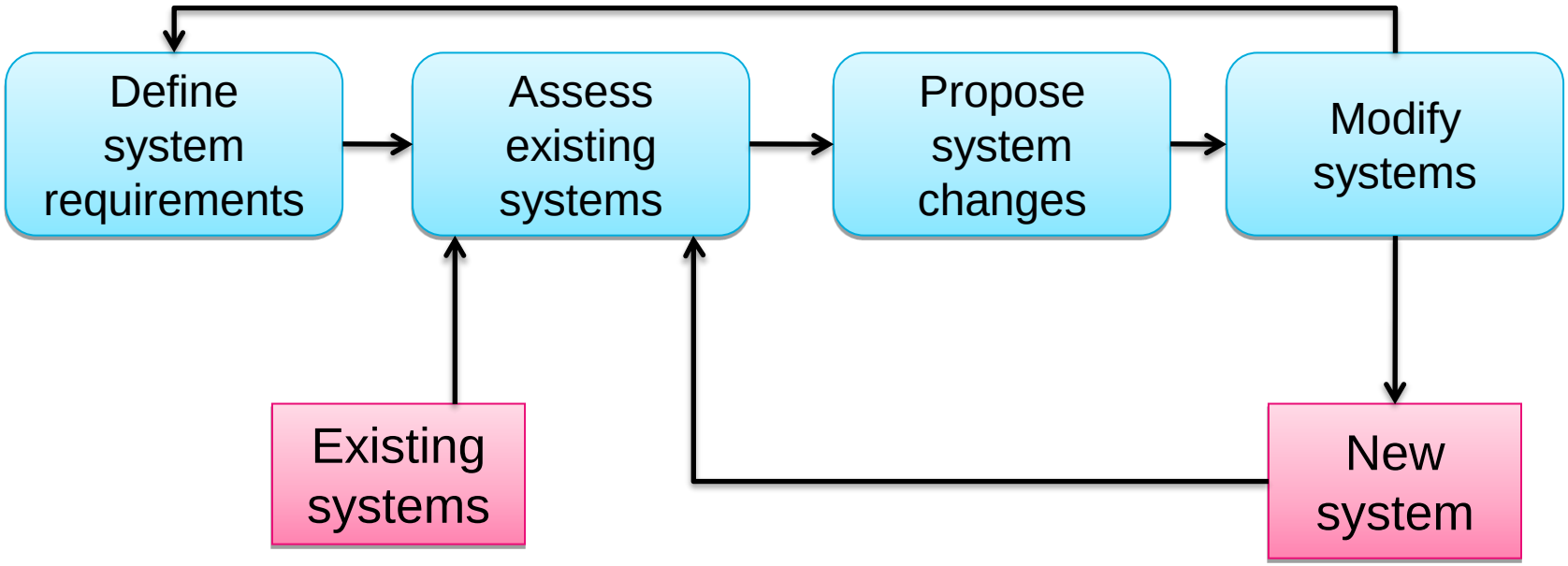
Các pha kiểm thử



Tiến hóa phần mềm

- Về mặt cố hữu, phần mềm có tính mềm dẻo và có thể thay đổi.
- Các yêu cầu thay đổi do sự biến đổi của các hoàn cảnh doanh nghiệp, phần mềm hỗ trợ doanh nghiệp cũng phải tiến hóa và thay đổi theo.
- Tuy đã có một ranh giới giữa phát triển và tiến hóa (bảo trì), ranh giới này ngày càng ít ý nghĩa khi ngày càng ít hệ thống hoàn toàn mới.

Tiến hóa hệ thống



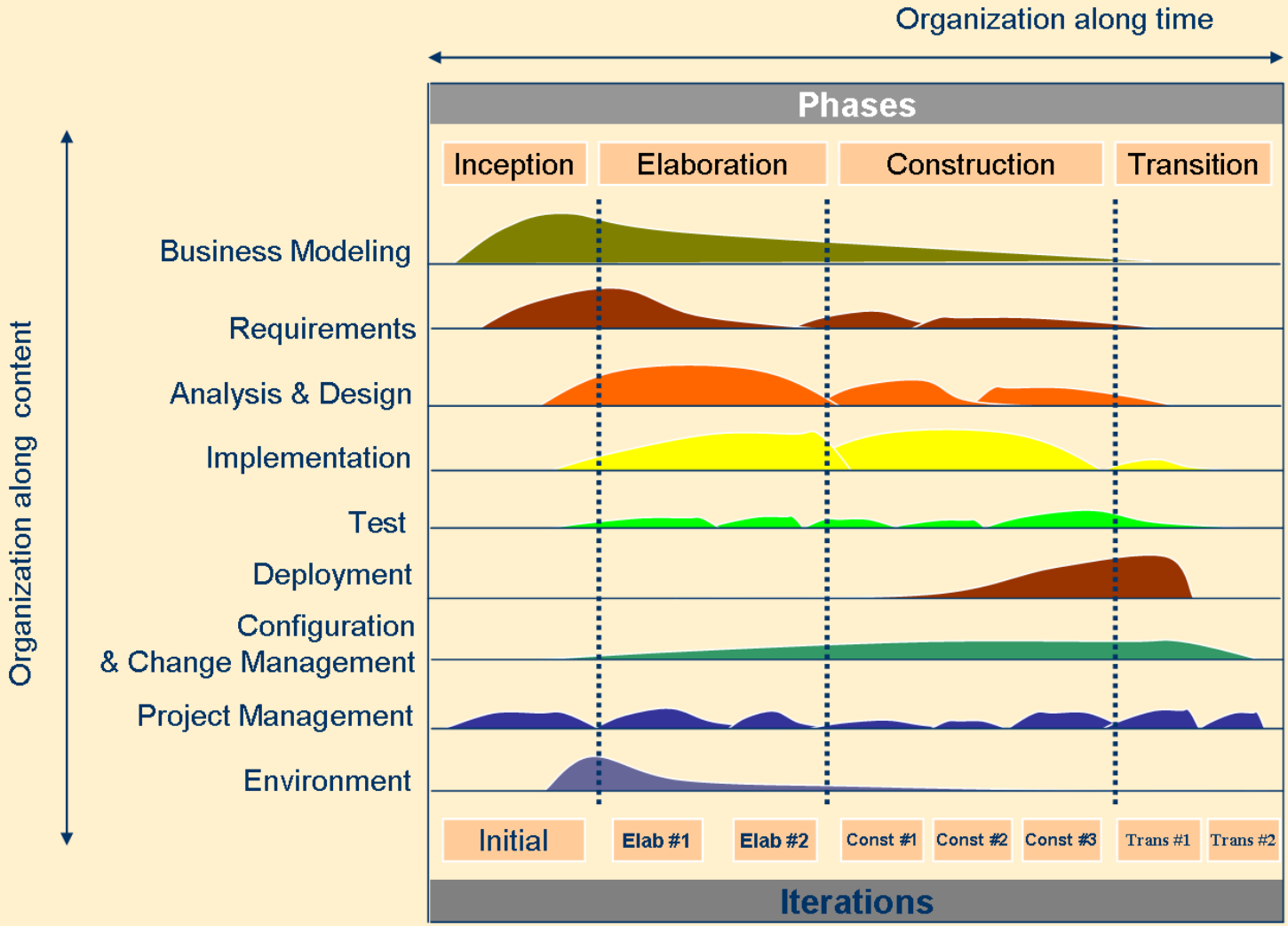
Đọc thêm

- Tài liệu tiếng Anh

The Rational Unified Process

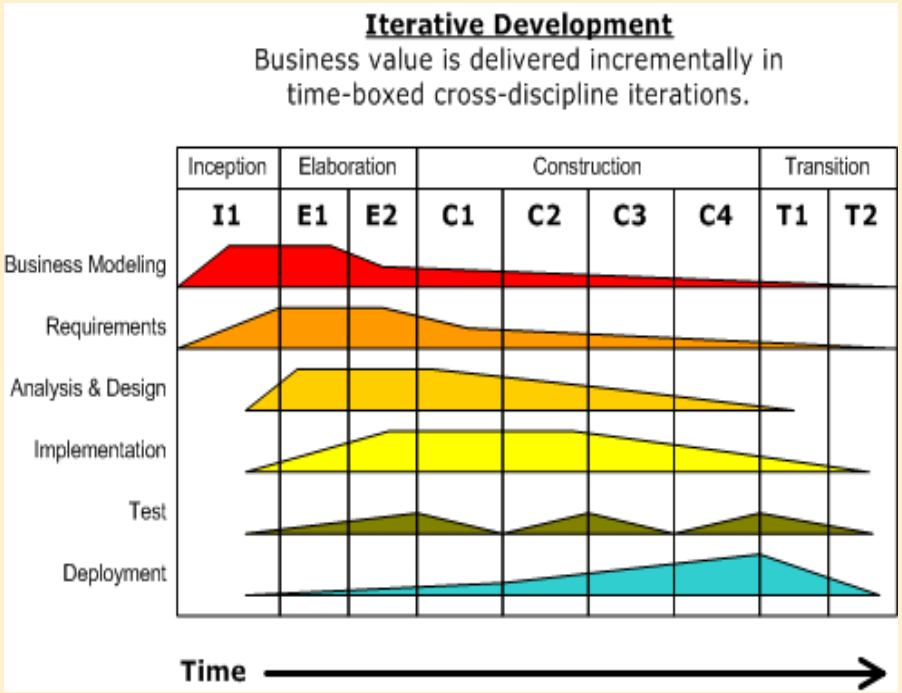
- A modern process model derived from the work on the UML and associated process.
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A practive perspective that suggests good practice.

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.



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 mgnt	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.

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
  - For large projects, much time is spent in team interactions.
  - CASE technology does not really support these.

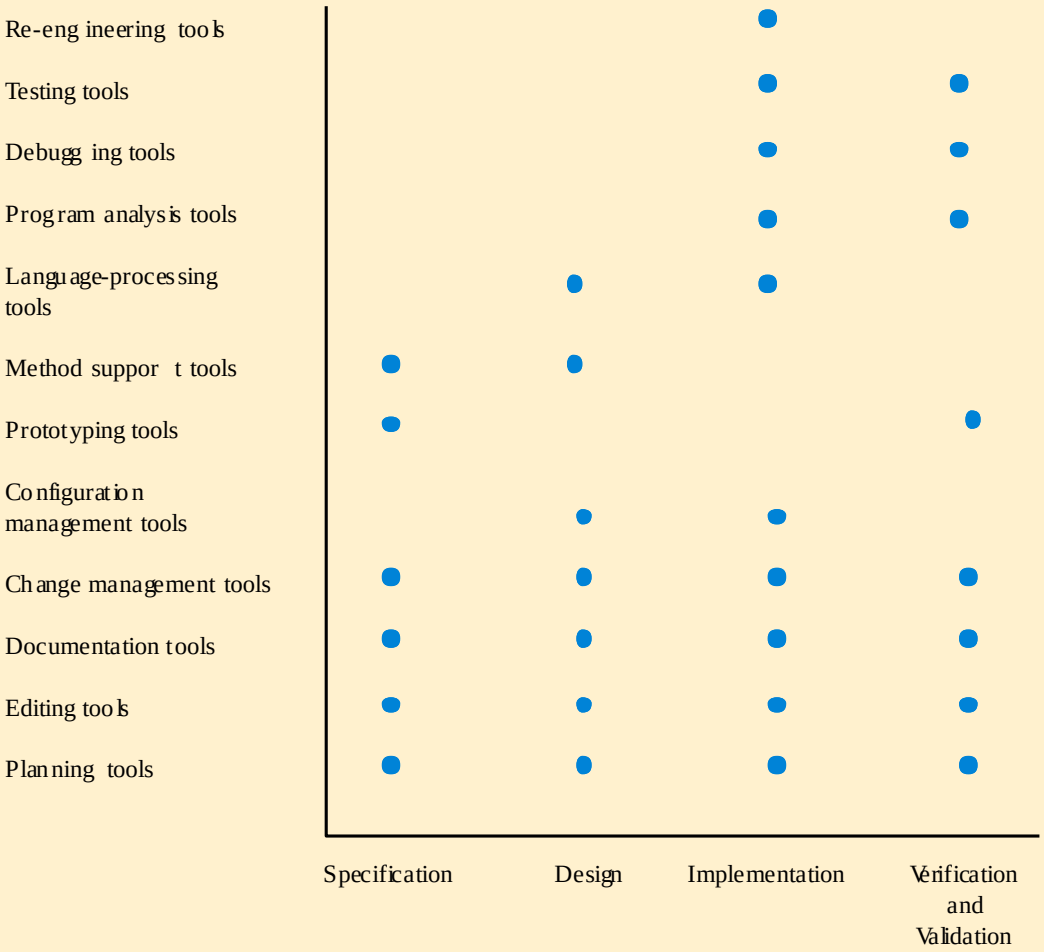
CASE classification

- 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

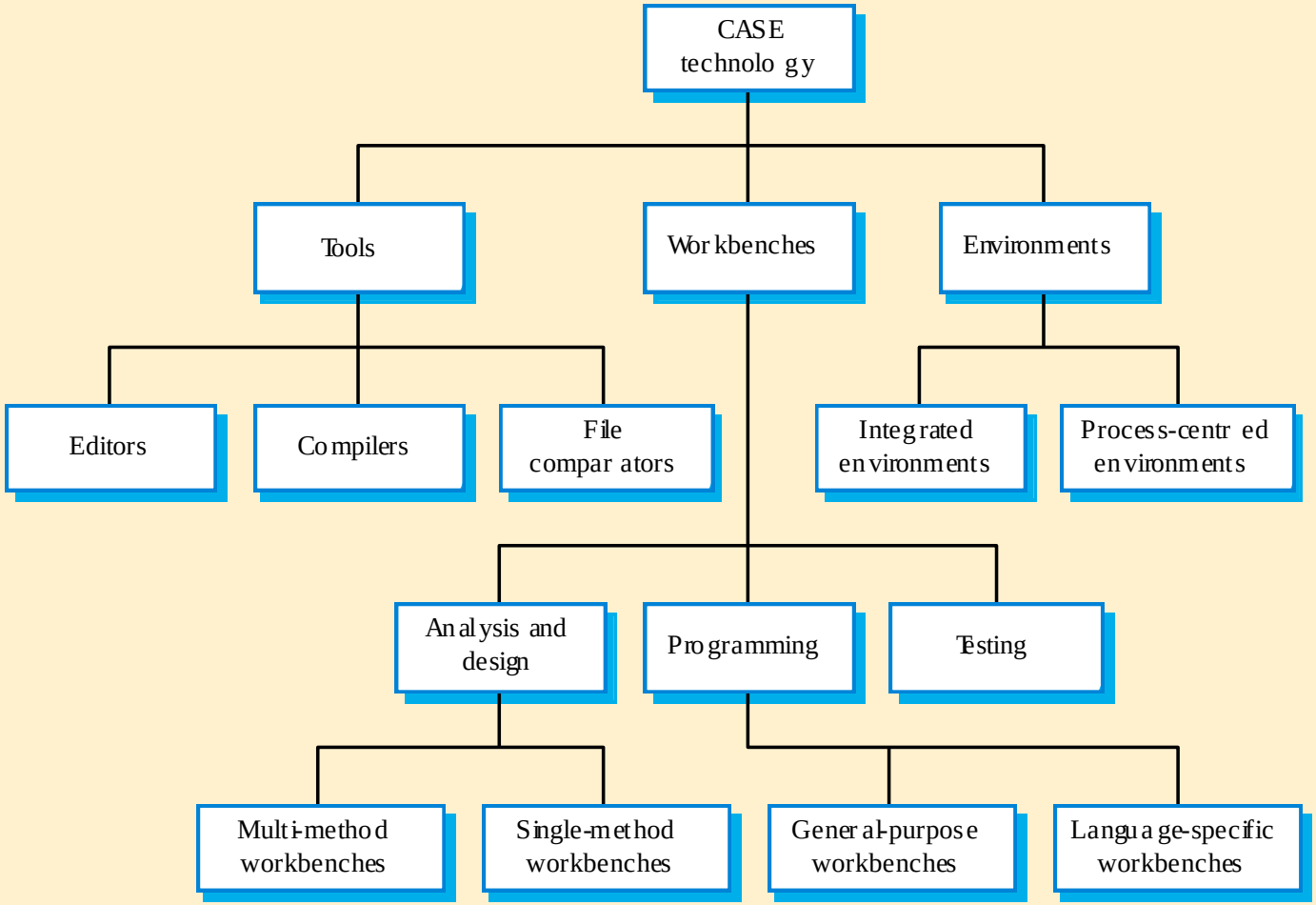
Activity-based tool classification



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.
- The Rational Unified Process is a generic process model that separates activities from phases.
- CASE technology supports software process activities.