ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

502045
Software Engineering

Chapter 02
Lesson 02: Software Processes

Topics covered

- ✧ Software process models
- ✧ Process activities
- ✧ Coping with change

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.

Software process descriptions

- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- Process descriptions may also include:
  - Products, which are the outcomes of a process activity;
  - Roles, which reflect the responsibilities of the people involved in the process;
  - Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

Software process descriptions

- Khi chúng ta mô tả và thảo luận về các quy trình, thường chúng ta nói về các hoạt động trong các quy trình này như **đặc tả một mô hình dữ liệu, thiết kế giao diện người dùng**, v.v. và **thứ tự của các hoạt động này.**
- Mô tả quy trình cũng có thể bao gồm:
  - **Sản phẩm:** Là các kết quả của một hoạt động quy trình;
  - **Vai trò:** Phản ánh các trách nhiệm của những người tham gia vào quy trình;
  - **Điều kiện tiên quyết và hậu quả**: Là các câu lệnh đúng trước và sau khi một hoạt động quy trình đã được thực hiện hoặc một sản phẩm được tạo ra.

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.

Plan-driven and agile processes

- Các quy trình dựa trên kế hoạch là các quy trình trong đó **tất cả các hoạt động quy trình được lập kế hoạch trước và tiến độ được đo lường dựa trên kế hoạch này**.
- Trong các quy trình linh hoạt, kế hoạch hóa được thực hiện theo cách tăng dần và dễ dàng thay đổi quy trình để phản ánh các yêu cầu của khách hàng đang thay đổi.
- Trong thực tế, hầu hết các quy trình thực tế đều bao gồm các yếu tố của cả hai phương pháp dựa trên kế hoạch và phương pháp linh hoạt.
- Không có quy trình phần mềm nào là đúng hoặc sai.

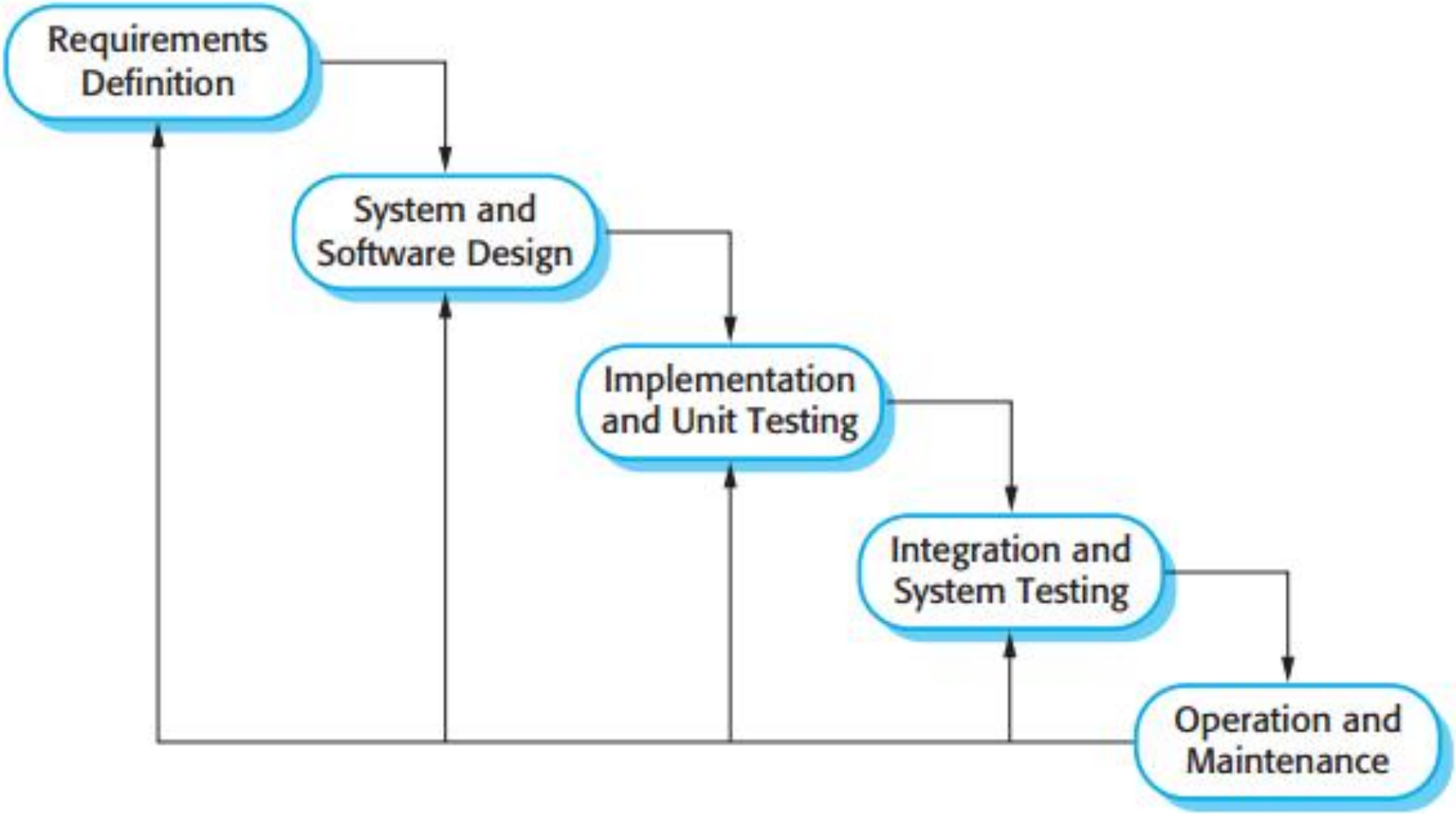
Software process models

- ✧ The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- ✧ Incremental development
  - Specification, development and validation are interleaved[xen-kẽ]. May be plan-driven or agile.
- ✧ Reuse-oriented software engineering
  - The system is assembled[lắp-ráp] from existing components. May be plan-driven or agile.
- ✧ In practice, most large systems are developed using a process that incorporates elements from all of these models.

Software process models

- ✧ Mô hình thác nước (Waterfall model)
  - - Mô hình dựa trên kế hoạch. Có các giai đoạn riêng biệt và khác biệt của việc đặc tả và phát triển.
- ✧ Phát triển tăng dần (Incremental development)
  - - Đặc tả, phát triển và kiểm thử được xen kẽ. Có thể dựa trên kế hoạch hoặc linh hoạt.
- ✧ Kỹ thuật phần mềm hướng tái sử dụng (Reuse-oriented software engineering)
  - - Hệ thống được lắp ráp từ các thành phần hiện có. Có thể dựa trên kế hoạch hoặc linh hoạt.
- ✧ Trong thực tế, hầu hết các hệ thống lớn được phát triển bằng một quy trình tích hợp các yếu tố từ tất cả các mô hình này.

The waterfall model



Waterfall model phases

- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

Waterfall model phases

- Trong mô hình thác nước, có các giai đoạn riêng biệt được xác định:
  1. Phân tích và xác định yêu cầu
  2. Thiết kế hệ thống và phần mềm
  3. Thực hiện và kiểm thử đơn vị
  4. Tích hợp và kiểm thử hệ thống
  5. Vận hành và bảo trì
- Nhược điểm chính của mô hình thác nước là **khó khăn trong việc điều chỉnh** khi có thay đổi sau khi quy trình đã bắt đầu. Theo nguyên tắc, một giai đoạn phải hoàn thành trước khi di chuyển sang giai đoạn tiếp theo.

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 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.

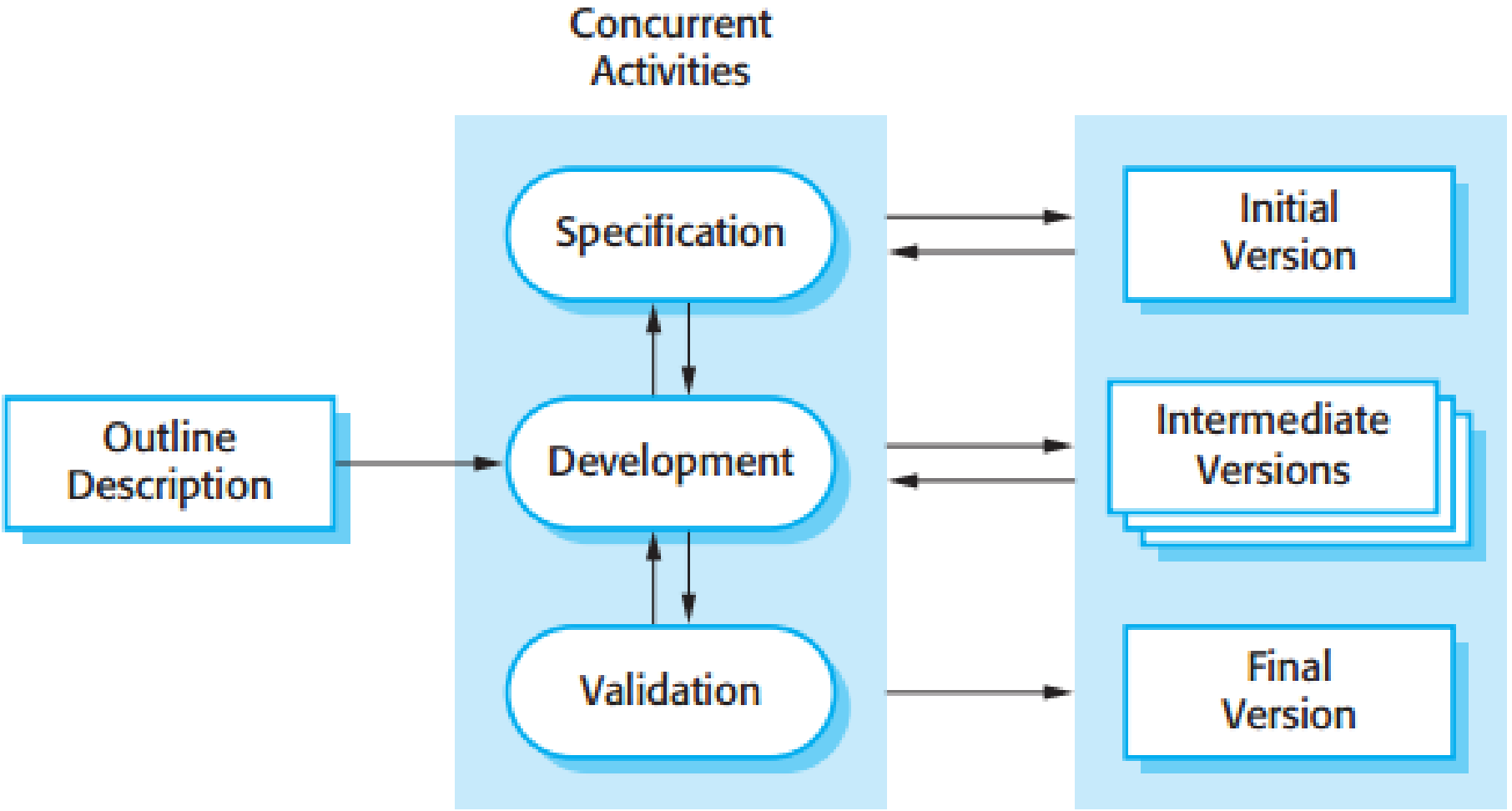
Waterfall model problems

- Việc chia dự án thành các giai đoạn cố định làm cho việc phản ứng lại các yêu cầu thay đổi của khách hàng trở nên khó khăn.
  - Do đó, mô hình này chỉ thích hợp khi các yêu cầu đã được hiểu rõ và các thay đổi trong quá trình thiết kế sẽ khá hạn chế.
  - Ít hệ thống doanh nghiệp có yêu cầu ổn định, nghĩa là yêu cầu thường thay đổi theo thời gian.
- Mô hình thác nước thường được sử dụng cho các dự án kỹ thuật hệ thống lớn nơi một hệ thống được phát triển tại nhiều địa điểm.
  - Trong các hoàn cảnh đó, tính chất dựa trên kế hoạch của mô hình thác nước giúp điều phối công việc một cách hiệu quả.

Discussion (10')

- Describe water model's phases for a specific software product.

Incremental development



Incremental development benefits

- The cost of accommodating changing customer requirements is reduced.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development benefits

- Việc chấp nhận các yêu cầu thay đổi của khách hàng trở nên dễ dàng hơn.
  - Lượng phân tích và tài liệu cần phải làm lại ít hơn so với mô hình thác nước.
- Việc thu thập phản hồi từ khách hàng về công việc phát triển đã thực hiện trở nên dễ dàng hơn.
  - Khách hàng có thể đưa ra ý kiến về các phiên trình diễn của phần mềm và nhìn thấy mức độ đã triển khai.
- Việc giao hàng và triển khai phần mềm hữu ích cho khách hàng trở nên nhanh chóng hơn.
  - Khách hàng có thể sử dụng và nhận giá trị từ phần mềm sớm hơn so với quá trình thác nước.

Incremental development problems

- 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.
- Cấu trúc hệ thống có xu hướng suy giảm khi các phần mới được thêm vào.
  - Trừ khi thời gian và tiền bạc được tiêu vào việc tái cấu trúc để cải thiện phần mềm, sự thay đổi định kỳ có xu hướng làm hỏng cấu trúc của nó. Việc tích hợp thêm các thay đổi phần mềm trở nên ngày càng khó khăn và tốn kém hơn.

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

- Dựa trên việc tái sử dụng hệ thống một cách có hệ thống trong đó các hệ thống được tích hợp từ các thành phần hiện có hoặc hệ thống COTS (Commercial-off-the-shelf).
- Các giai đoạn quy trình:
  1. Phân tích thành phần;
  2. Sửa đổi yêu cầu;
  3. Thiết kế hệ thống với sự tái sử dụng;
  4. Phát triển và tích hợp.
- Tái sử dụng hiện đã trở thành phương pháp tiêu chuẩn cho việc xây dựng nhiều loại hệ thống doanh nghiệp.

Reuse-oriented software engineering

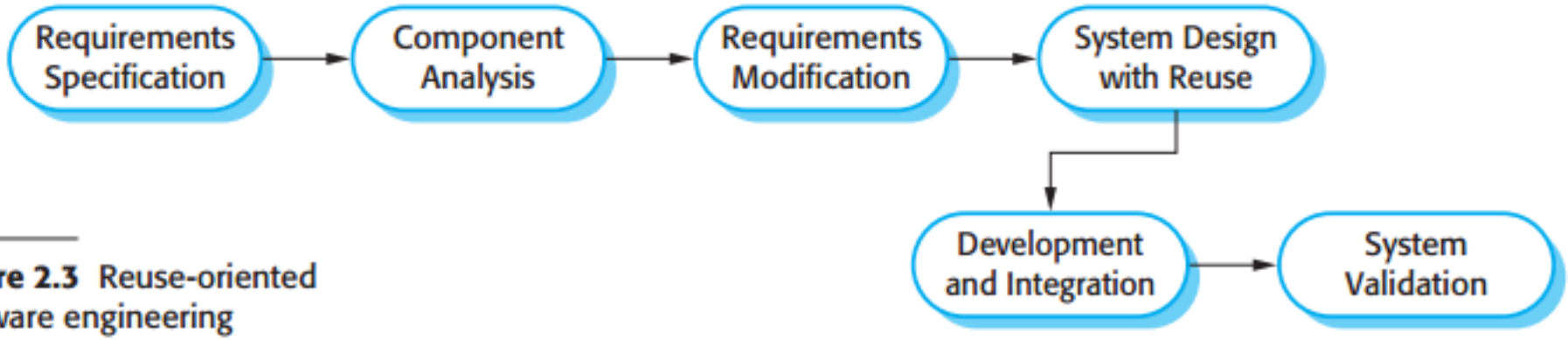


Figure 2.3 Reuse-oriented software engineering

Types of software component

- Web services that are developed according to service standards and which are available for remote invocation[call].
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Stand-alone software systems (COTS) that are configured for use in a particular environment.

Types of software component

1. Dịch vụ web (Web services): Đây là các dịch vụ được phát triển theo các tiêu chuẩn dịch vụ và có sẵn để gọi từ xa. Các dịch vụ web thường được truy cập thông qua giao thức HTTP và có thể được sử dụng cho việc truyền thông giữa các ứng dụng khác nhau trên Internet.

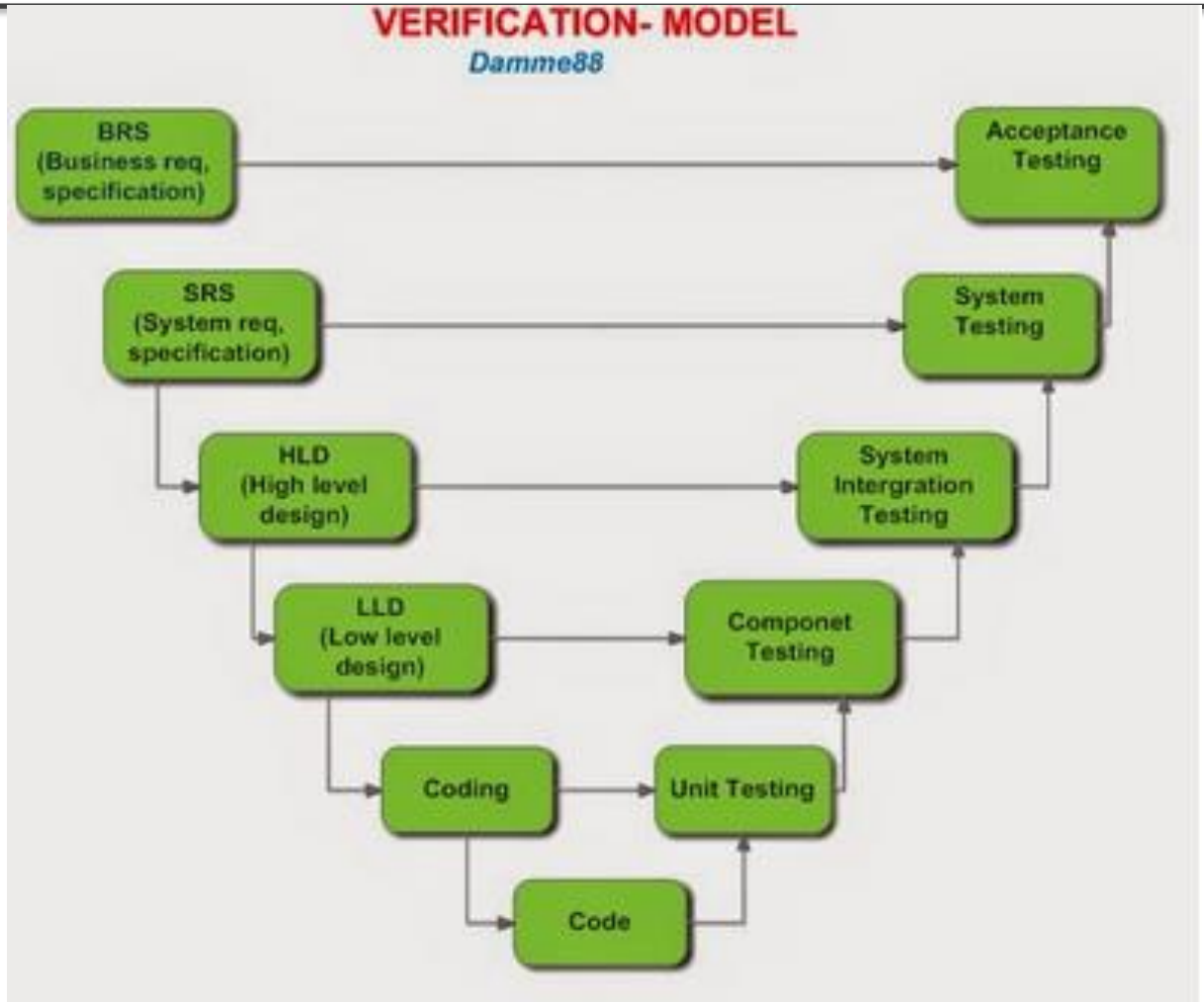
2. Bộ sưu tập các đối tượng (Collections of objects): Đây là các bộ sưu tập các đối tượng được phát triển như một gói để tích hợp với một framework thành phần như .NET hoặc J2EE. Các đối tượng này có thể được sử dụng để xây dựng các ứng dụng phần mềm và dễ dàng tích hợp vào các môi trường phát triển đã có sẵn.

3. Hệ thống phần mềm độc lập (COTS): Đây là các hệ thống phần mềm đứng một mình (stand-alone) đã được cấu hình để sử dụng trong một môi trường cụ thể. Các hệ thống này thường được mua sẵn từ các nhà cung cấp và được cấu hình lại để phù hợp với nhu cầu cụ thể của môi trường và doanh nghiệp sử dụng.

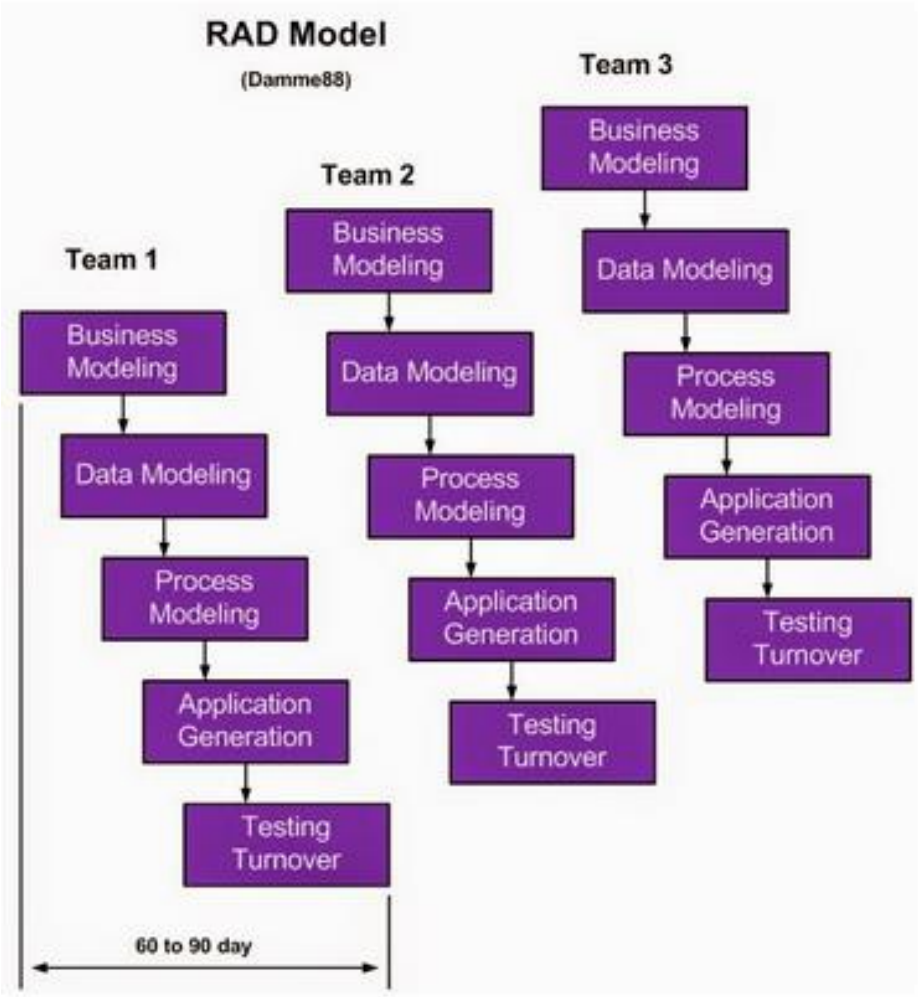
Others (self study)

- ✧ V Software model
- ✧ Rapid application development: RAD

V Software model



RAD model


RAD Model
(Damme88)
Team 1
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing Turnover
Team 2
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing Turnover
Team 3
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing Turnover
60 to 90 day

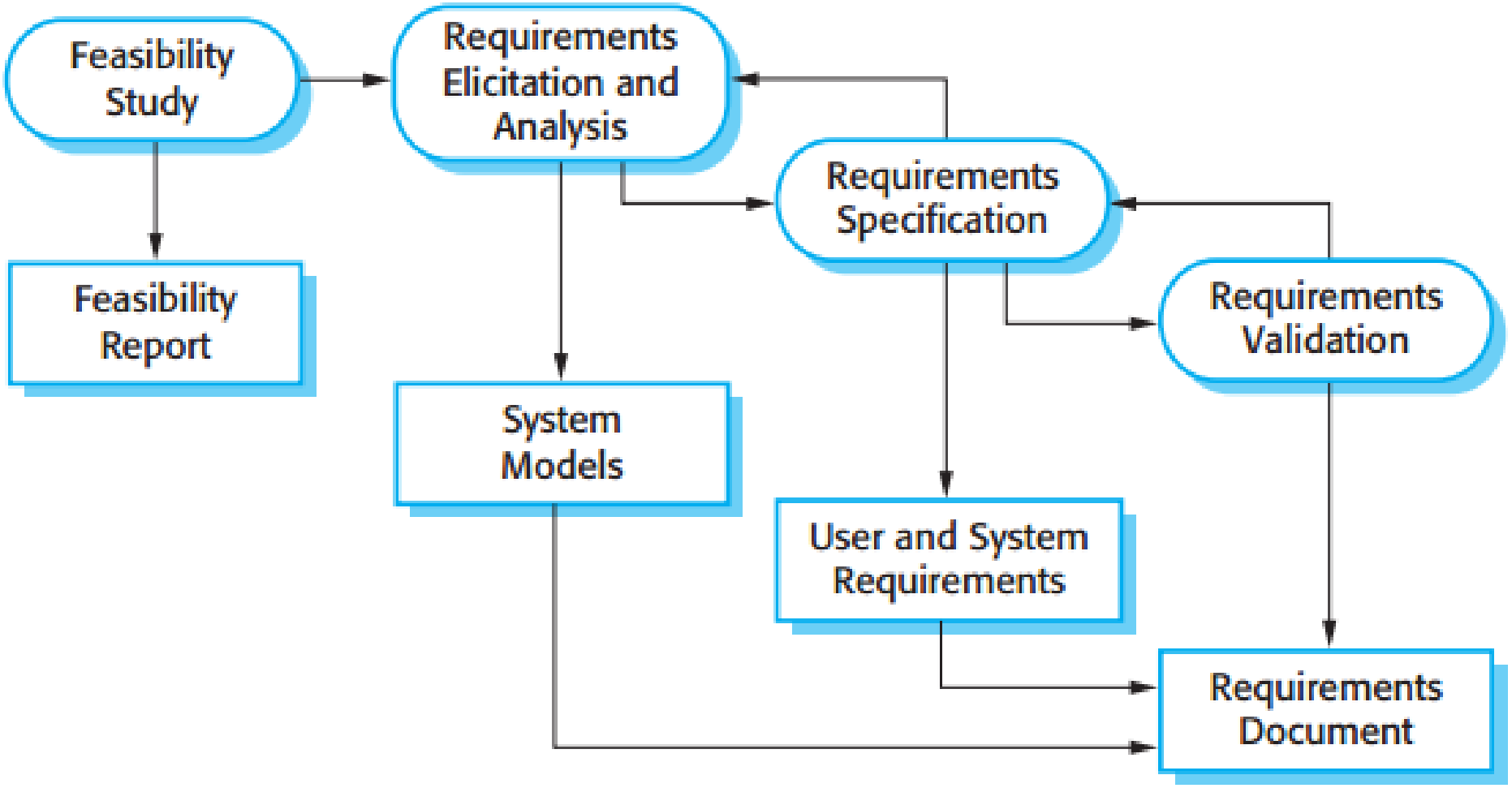
Process activities (self study)

- Real software processes are inter-leaved (đan xen) 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.

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 elicitation 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



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.

Software design and implementation

✧ Quá trình chuyển đổi đặc tả **hệ thống thành một hệ thống thực thi.**

1. Thiết kế phần mềm (Software design): Thiết kế một cấu trúc phần mềm để thực hiện đặc tả.

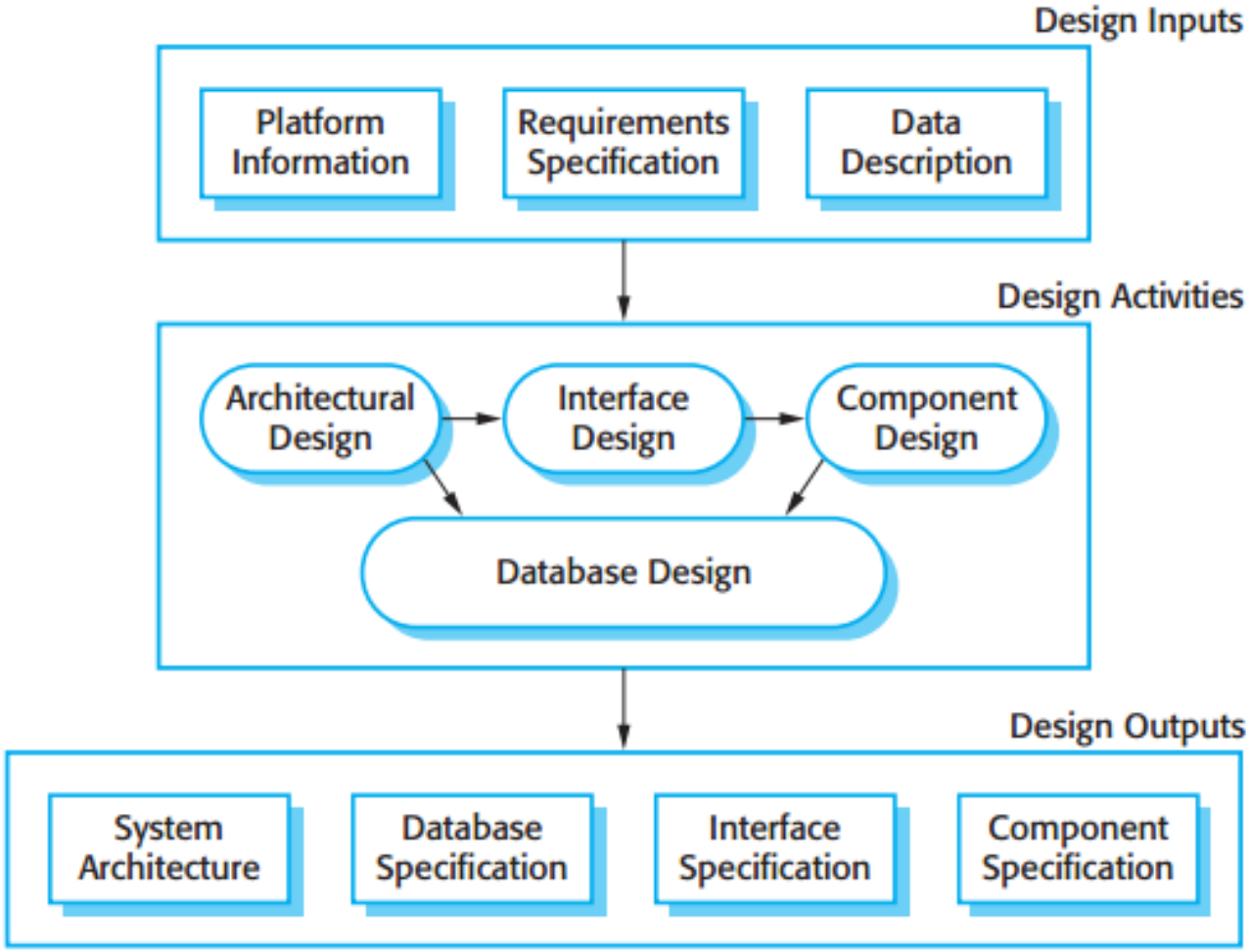
- Trong giai đoạn này, các quyết định về cách thức triển khai các yêu cầu cụ thể và cách tổ chức các thành phần phần mềm được đưa ra.

2. Thực thi (Implementation): Dịch cấu trúc này thành một chương trình thực thi.

- Các nhà phát triển sẽ viết mã và triển khai các thành phần của phần mềm dựa trên thiết kế đã được xác định trong giai đoạn thiết kế.

✧ Các hoạt động của thiết kế và thực thi có mối liên hệ chặt chẽ và có thể được **xen kẽ với nhau**. Điều này có nghĩa là trong quá trình phát triển, các nhà phát triển có thể thực hiện các hoạt động thiết kế và thực thi đồng thời hoặc xen kẽ để tối ưu hóa quá trình phát triển và đáp ứng nhanh chóng các yêu cầu thay đổi của khách hàng.

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 activities

1. Thiết kế kiến trúc (Architectural design): Trong giai đoạn này, bạn xác định cấu trúc tổng thể của hệ thống, các thành phần chính (đôi khi được gọi là các hệ thống phụ hoặc module), mối quan hệ giữa chúng và cách chúng được phân phối. Mục tiêu là tạo ra một bản thiết kế tổng thể cho hệ thống, bao gồm sơ đồ kiến trúc và cấu trúc chung của hệ thống.

2. Thiết kế giao diện (Interface design): Ở giai đoạn này, bạn xác định các giao diện giữa các thành phần của hệ thống. Điều này bao gồm xác định các giao diện người dùng (user interfaces) và các giao diện giữa các thành phần phần mềm khác nhau.

3. Thiết kế thành phần (Component design): Trong giai đoạn này, mỗi thành phần của hệ thống được thiết kế chi tiết, bao gồm cách hoạt động của nó và cách nó liên kết với các thành phần khác trong hệ thống.

4. Thiết kế cơ sở dữ liệu (Database design): Ở giai đoạn này, bạn thiết kế cấu trúc dữ liệu của hệ thống và cách dữ liệu được biểu diễn trong cơ sở dữ liệu. Các yếu tố như bảng dữ liệu, quan hệ giữa các bảng, và các quy tắc hợp lý hóa cơ sở dữ liệu được xác định và thiết kế.

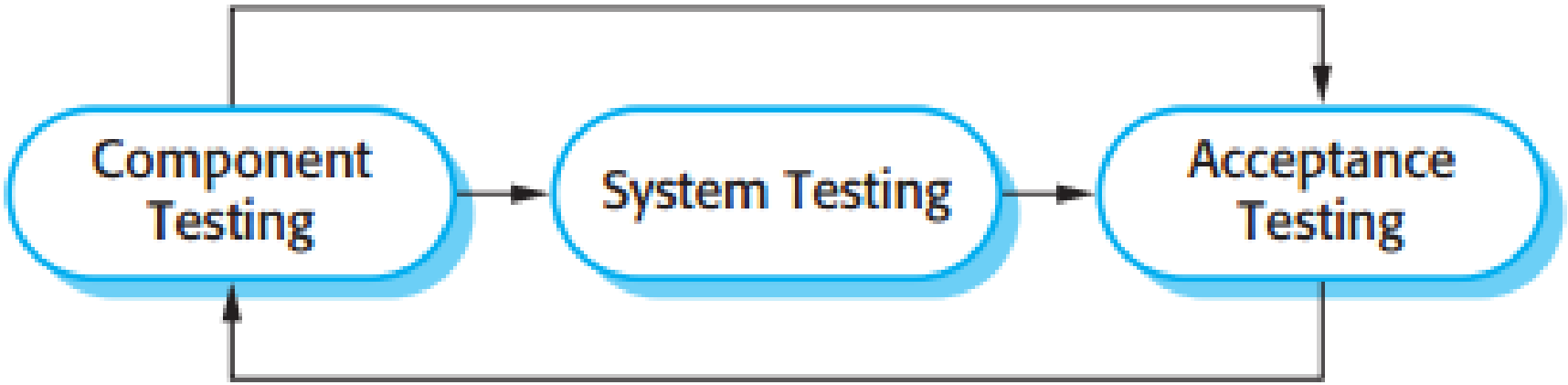
Software validation

- 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 is the most commonly used V & V activity.

Software validation

- Xác nhận và xác thực (Verification and validation V & V) được dùng để chỉ ra rằng một hệ thống tuân thủ theo đặc tả của nó và đáp ứng các yêu cầu của khách hàng hệ thống.
- Nó bao gồm các quy trình kiểm tra và xem xét cũng như kiểm thử hệ thống.
- Kiểm thử hệ thống bao gồm việc thực thi hệ thống với các trường hợp kiểm thử được rút ra từ đặc tả của dữ liệu thực sự cần được xử lý bởi hệ thống.
- Kiểm thử là hoạt động V & V phổ biến nhất.

Stages of testing



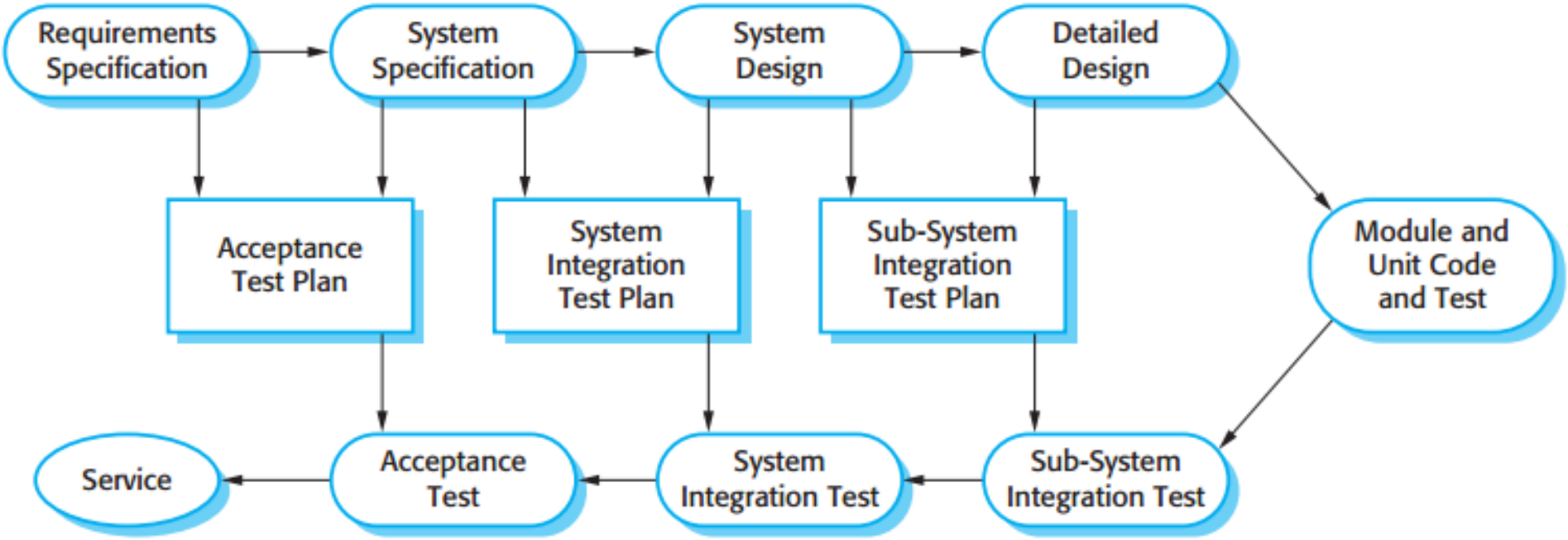
Testing stages

- ✧ Development or component 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 stages

- ✧ Kiểm thử phần mềm hoặc kiểm thử thành phần (Development or component testing):
  - - Các thành phần cá nhân được kiểm thử độc lập.
  - - Các thành phần có thể là các hàm hoặc đối tượng hoặc nhóm các thực thể nhóm lại một cách hợp lý.
- ✧ Kiểm thử hệ thống (System testing):
  - - Kiểm thử toàn bộ hệ thống. Kiểm thử các thuộc tính xuất hiện là đặc biệt quan trọng.
- ✧ Kiểm thử chấp nhận (Acceptance testing):
  - - Kiểm thử với dữ liệu của khách hàng để kiểm tra xem hệ thống có đáp ứng được nhu cầu của khách hàng hay không.

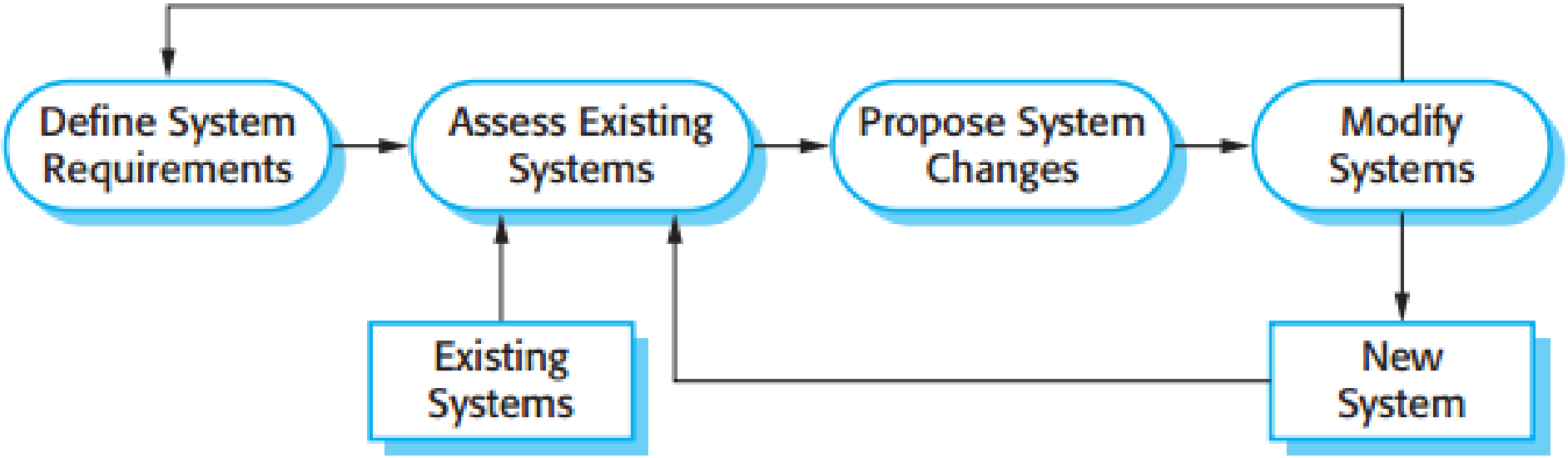
Testing phases in a plan-driven software process



Software evolution

- Software is inherently[nature] flexible and can change.
- As requirements change through changing business circumstances[trường-hợp], the software that supports the business must also evolve[phát-triển] and change.

System evolution



Key points

- Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.
- General process models describe the organization of software processes. Examples of these general models include the ‘waterfall’ model, incremental development, and reuse-oriented development.

Key points

- Các quy trình phần mềm là các hoạt động liên quan đến việc sản xuất một hệ thống phần mềm. Các mô hình quy trình phần mềm là các biểu diễn trừu tượng của các quy trình này.
- Các mô hình quy trình tổng quát mô tả tổ chức của các quy trình phần mềm. Ví dụ về các mô hình tổng quát này bao gồm mô hình 'thác nước' (waterfall), phát triển tăng dần (incremental development) và phát triển hướng tái sử dụng (reuse-oriented development).

Key points

- 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.

Key points

Kỹ thuật yêu cầu (Requirements engineering) là quá trình phát triển một đặc tả phần mềm.

Các quy trình thiết kế và thực thi liên quan đến việc chuyển đổi một đặc tả yêu cầu thành một hệ thống phần mềm có thể thực thi.

Kiểm thử phần mềm (Software validation) là quá trình kiểm tra hệ thống xem nó tuân thủ đặc tả của nó và liệu nó có đáp ứng được nhu cầu thực sự của người dùng hệ thống hay không.

Tiến hóa phần mềm (Software evolution) xảy ra khi bạn thay đổi các hệ thống phần mềm hiện có để đáp ứng các yêu cầu mới. Phần mềm phải tiến hóa để vẫn có ích.

Topics covered

- ✧ Software process models
- ✧ Process activities
- ✧ Coping with change

Coping with change [self study]

- Change is inevitable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- Change leads to rework so the costs of change include both rework (e.g. re-analysing requirements) as well as the costs of implementing new functionality

Coping with change [self study]

- Thay đổi là không thể tránh khỏi trong tất cả các dự án phần mềm lớn.
  - Sự thay đổi trong doanh nghiệp dẫn đến yêu cầu hệ thống mới và thay đổi.
  - Các công nghệ mới mở ra những khả năng mới để cải thiện cài đặt.
  - Việc thay đổi nền tảng yêu cầu thay đổi ứng dụng.
- Thay đổi dẫn đến việc làm lại, vì vậy các chi phí của thay đổi bao gồm cả làm lại (ví dụ: phân tích yêu cầu lại) cũng như chi phí triển khai chức năng mới.

Reducing the costs of rework

- Change avoidance, where the software process includes activities that can anticipate possible changes before significant rework is required.
  - For example, a prototype system may be developed to show some key features of the system to customers.
- Change tolerance, where the process is designed so that changes can be accommodated at relatively low cost.
  - This normally involves some form of incremental development. Proposed changes may be implemented in increments that have not yet been developed. If this is impossible, then only a single increment (a small part of the system) may have be altered to incorporate the change.

Reducing the costs of rework

- Tránh thay đổi (Change avoidance) là khi quy trình phần mềm bao gồm các hoạt động có thể dự đoán được các thay đổi có thể xảy ra trước khi cần phải làm lại một cách đáng kể.
  - Ví dụ, một hệ thống nguyên mẫu có thể được phát triển để hiển thị một số tính năng chính của hệ thống cho khách hàng.
- Chấp nhận thay đổi (Change tolerance) là khi quy trình được thiết kế để có thể chứa đựng các thay đổi với chi phí tương đối thấp.
  - Thông thường, điều này bao gồm một dạng phát triển tăng dần. Những thay đổi đề xuất có thể được triển khai theo các giai đoạn chưa được phát triển. Nếu điều này không thể thực hiện được, thì chỉ một giai đoạn đơn lẻ (một phần nhỏ của hệ thống) có thể được thay đổi để tích hợp thay đổi.

Software prototyping

- A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation;
  - In design processes to explore options and develop a UI design;
  - In the testing process to run back-to-back tests.

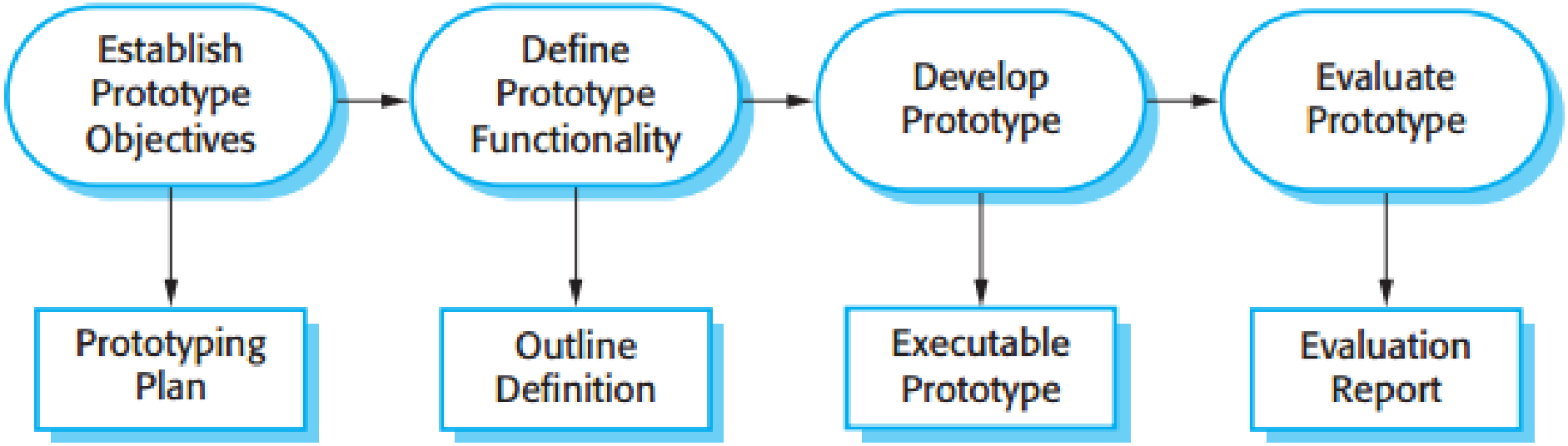
Software prototyping

- Một nguyên mẫu (prototype) là một phiên bản ban đầu của một hệ thống được sử dụng để trình bày các khái niệm và thử nghiệm các tùy chọn thiết kế.
- Một nguyên mẫu có thể được sử dụng trong:
  - - Quy trình kỹ thuật yêu cầu để hỗ trợ việc thu thập và xác thực yêu cầu;
  - - Trong quy trình thiết kế để khám phá các lựa chọn và phát triển thiết kế giao diện người dùng;
  - - Trong quy trình kiểm thử để chạy các kiểm thử back-to-back (liên tục).

Benefits of prototyping

- ✧ Improved system usability.
- ✧ A closer match to users' real needs.
- ✧ Improved design quality.
- ✧ Improved maintainability.
- ✧ Reduced development effort.

The process of prototype development



Prototype development

- ✧ May be based on rapid prototyping languages or tools
- ✧ May involve leaving out[exclude] functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

Prototype development

- Có thể dựa trên ngôn ngữ hoặc công cụ nguyên mẫu nhanh chóng.
- Có thể bao gồm việc loại bỏ các chức năng.
  - Nguyên mẫu nên tập trung vào các lĩnh vực của sản phẩm không được hiểu rõ;
  - Kiểm tra lỗi và khôi phục có thể không được bao gồm trong nguyên mẫu;
  - Tập trung vào các yêu cầu chức năng thay vì các yêu cầu không chức năng như độ tin cậy và bảo mật.

Throw-away prototypes

- Prototypes should be discarded after development as they are not a good basis for a production system:
  - It may be impossible to tune the system to meet non-functional requirements;
  - Prototypes are normally undocumented;
  - The prototype structure is usually degraded through rapid change;
  - The prototype probably will not meet normal organisational quality standards.

Throw-away prototypes

- Nguyên mẫu nên được loại bỏ sau quá trình phát triển vì chúng không phải là một cơ sở tốt cho hệ thống sản xuất:
  - - Có thể không thể điều chỉnh hệ thống để đáp ứng các yêu cầu không chức năng;
  - - Nguyên mẫu thường không được tài liệu hóa;
  - - Cấu trúc của nguyên mẫu thường bị suy giảm thông qua các thay đổi nhanh chóng;
  - - Nguyên mẫu có thể sẽ không đáp ứng được các tiêu chuẩn chất lượng tổ chức thông thường.

Incremental delivery

- 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.
- User requirements are prioritised[uutien] 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 delivery

- Thay vì giao hệ thống như một lần giao hàng duy nhất, quá trình phát triển và giao hàng được phân chia thành các phần nhỏ (increments) với mỗi phần nhỏ cung cấp một phần của chức năng cần thiết.
- Yêu cầu của người dùng được ưu tiên và các yêu cầu ưu tiên cao được bao gồm trong các phần nhỏ ban đầu.
- Một khi việc phát triển của một phần nhỏ đã bắt đầu, các yêu cầu sẽ được đóng băng mặc dù yêu cầu cho các phần nhỏ sau có thể tiếp tục phát triển.

Incremental development and delivery

ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

✧ Incremental development

- Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
- Normal approach used in agile methods;
- Evaluation done by user/customer proxy.

✧ Incremental delivery

- Deploy an increment for use by end-users;
- More realistic evaluation about practical use of software;
- Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Incremental development and delivery

✧ Phát triển tăng dần (Incremental development):

- - Phát triển hệ thống thành từng phần nhỏ và đánh giá mỗi phần nhỏ trước khi tiếp tục phát triển phần nhỏ tiếp theo;
- - Phương pháp thường được sử dụng trong các phương pháp linh hoạt (agile);
- - Đánh giá được thực hiện bởi người đại diện của người dùng/khách hàng.

✧ Giao hàng tăng dần (Incremental delivery):

- - Triển khai một phần nhỏ để sử dụng bởi người dùng cuối;
- - Đánh giá được thực hiện dựa trên việc sử dụng thực tế của phần mềm;
- - Khó thực hiện cho các hệ thống thay thế vì các phần nhỏ có ít chức năng hơn so với hệ thống được thay thế.

Incremental delivery

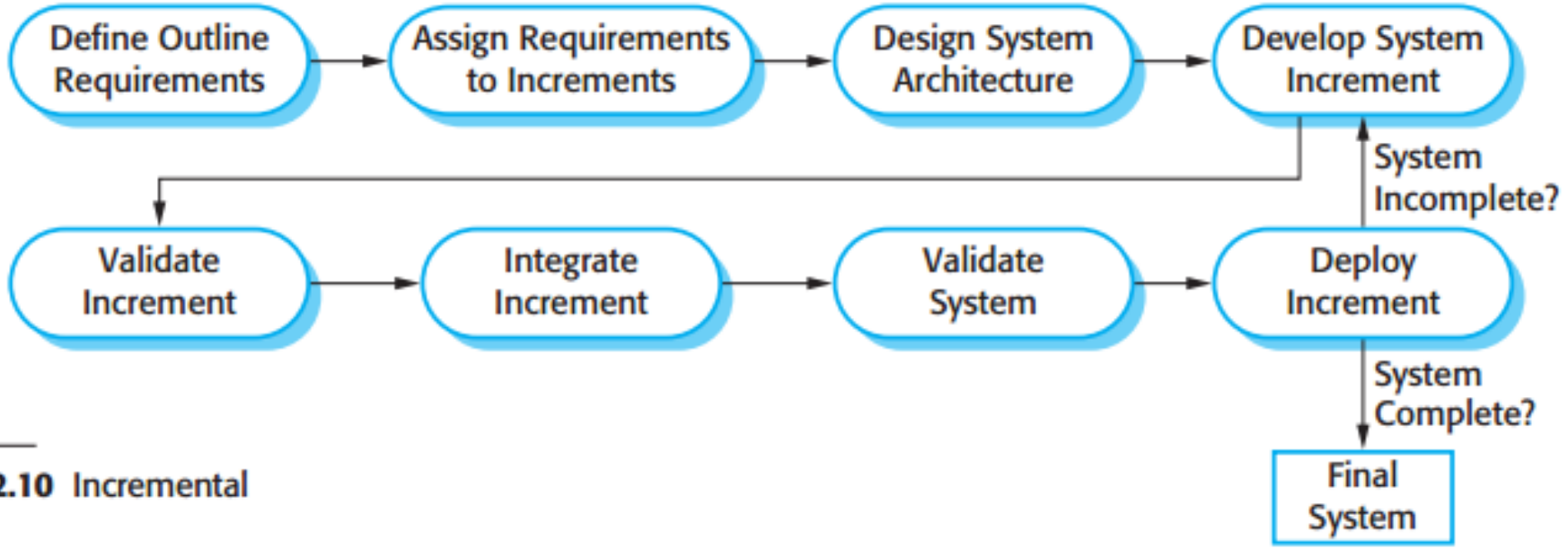


Figure 2.10 Incremental delivery

Incremental delivery advantages

- 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.

Incremental delivery advantages

- - Giá trị cho khách hàng có thể được giao hàng cùng với mỗi phần nhỏ, vì vậy chức năng của hệ thống sẵn có sớm hơn.
- - Các phần nhỏ ban đầu hoạt động như một nguyên mẫu để giúp thu thập yêu cầu cho các phần nhỏ sau.
- - Rủi ro thất bại của dự án tổng thể thấp hơn.
- - Các dịch vụ hệ thống ưu tiên cao nhất thường nhận được nhiều kiểm thử nhất.

Incremental delivery problems

- Most systems require a set of basic facilities that are used by different parts of the system.
  - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
- The essence of iterative processes is that the specification is developed in conjunction with the software.
  - However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

Incremental delivery problems

- Hầu hết các hệ thống đòi hỏi một tập hợp các cơ sở cơ bản được sử dụng bởi các phần khác nhau của hệ thống.
  - Khi yêu cầu chưa được xác định chi tiết cho đến khi một phần nhỏ được triển khai, việc xác định các cơ sở chung cần thiết cho tất cả các phần nhỏ có thể khó khăn.
- Bản chất của các quy trình lặp lại là đặc tả được phát triển song song với phần mềm.
  - Tuy nhiên, điều này xung đột với mô hình mua sắm của nhiều tổ chức, trong đó đặc tả hệ thống hoàn chỉnh là một phần của hợp đồng phát triển hệ thống.

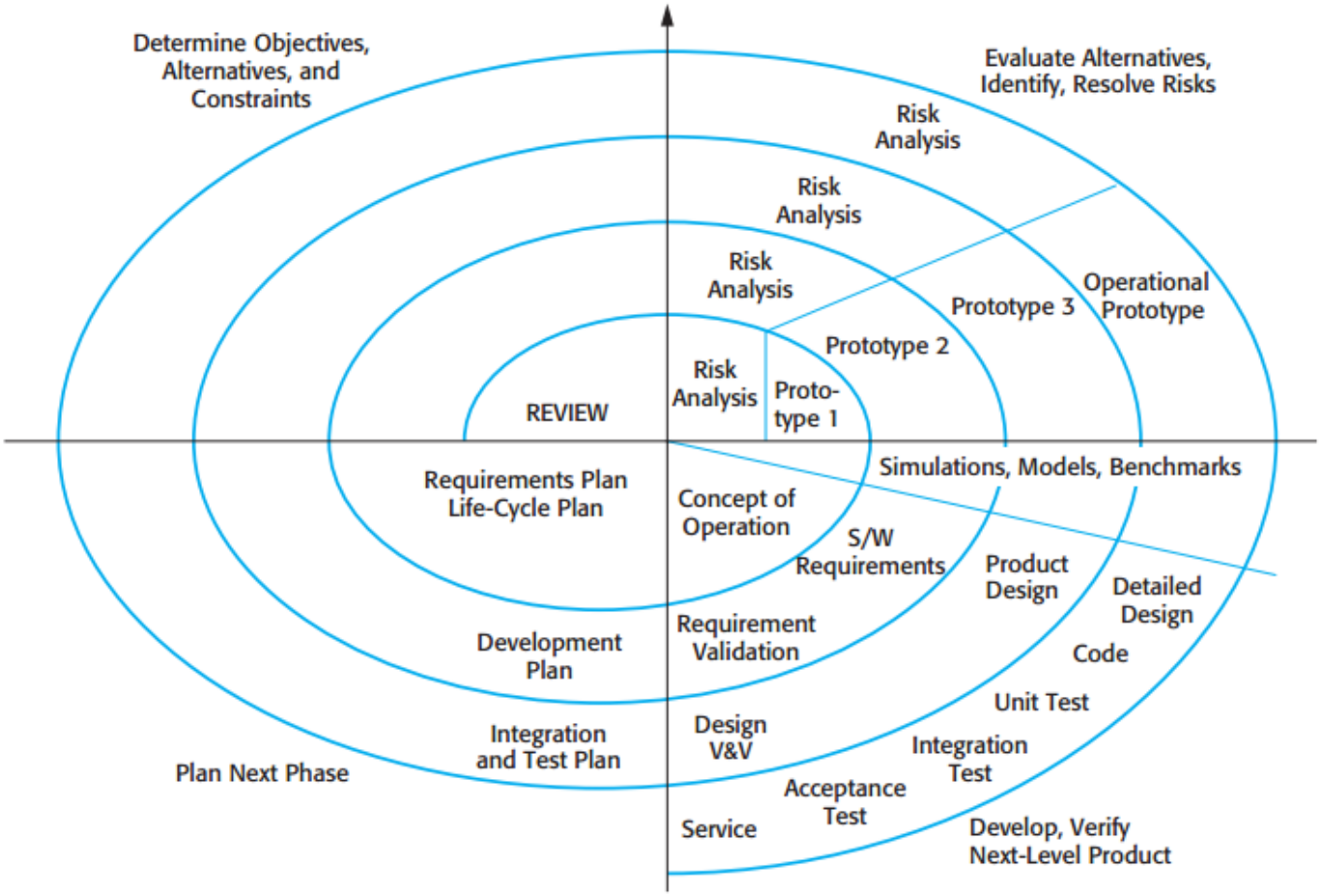
Boehm's spiral model

- 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.

Boehm's spiral model

- Quy trình được biểu diễn dưới dạng một vòng xoắn thay vì là một chuỗi các hoạt động với việc quay lại.
- Mỗi vòng trong vòng xoắn đại diện cho một giai đoạn trong quy trình.
- Không có các giai đoạn cố định như đặc tả hoặc thiết kế - các vòng trong vòng xoắn được chọn tùy thuộc vào những gì được yêu cầu.
- Rủi ro được đánh giá rõ ràng và giải quyết xuyên suốt quy trình.

Boehm's 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.

Spiral model sectors

- **1. Đặt mục tiêu:** Xác định các mục tiêu cụ thể cho giai đoạn.
- **2. Đánh giá và giảm thiểu rủi ro:** Đánh giá các rủi ro và thiết lập các hoạt động để giảm thiểu các rủi ro chính.
- **3. Phát triển và xác nhận:** Chọn một mô hình phát triển cho hệ thống, có thể là bất kỳ mô hình tổng quát nào.
- **4. Lập kế hoạch:** Đánh giá dự án và lập kế hoạch cho giai đoạn tiếp theo của vòng xoắn.

Spiral model usage

- Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- In practice, however, the model is rarely used as published for practical software development.
- Mô hình xoắn đã có ảnh hưởng rất lớn trong việc giúp mọi người suy nghĩ về việc lặp lại trong các quy trình phần mềm và giới thiệu phương pháp phát triển dựa trên rủi ro.
- Tuy nhiên, trong thực tế, mô hình hiếm khi được sử dụng như đã công bố cho việc phát triển phần mềm thực tế.

Key points

- Processes should include activities to cope with change. This may involve a prototyping phase that helps avoid poor decisions on requirements and design.
- Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole.

Key points

- Các quy trình nên bao gồm các hoạt động để xử lý thay đổi. Điều này có thể bao gồm một giai đoạn nguyên mẫu giúp tránh ra các quyết định kém chất lượng về yêu cầu và thiết kế.
- Các quy trình có thể được cấu trúc cho việc phát triển và giao hàng lặp lại để có thể thực hiện các thay đổi mà không làm ảnh hưởng đến hệ thống toàn bộ.