TRƯỜNG ĐẠI HỌC KINH TẾ QUỐC DÂN
VIỆN CÔNG NGHỆ THÔNG TIN & KINH TẾ SỐ

CHƯƠNG I

TỔNG QUAN VỀ PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

MỤC TIÊU

Hiểu các khái niệm chung	Trình bày được mô hình hóa hướng đối tượng – UML	Trình bày được quy trình phát triển phần mềm - quy trình RUP
• HT và HTTT • Vòng đời phát triển hệ thống • Phương pháp PT HT • Công cụ PT HT • Đội ngũ PT HT	• Lịch sử • Góc nhìn • Các biểu đồ • Công cụ	• Khái niệm • Quy trình RUP

NỘI DUNG

1.1 Khái niệm Hệ thống và Hệ thống thông tin

1.2 Vòng đời phát triển hệ thống

1.3 Phương pháp phát triển hệ thống

1.4 Đội ngũ phát triển hệ thống

1.5 Ngôn ngữ mô hình hóa thống nhất UML

1.6 Quy trình phát triển phần mềm

1.1. MỘT SỐ KHÁI NIỆM CHUNG

1.1.1 Hệ thống

ĐN: Hệ thống là một tập hợp gồm nhiều ***phần tử*** có ***quan hệ*** ràng buộc lẫn nhau và cùng hoạt động hướng tới một ***mục đích*** chung

Ví dụ:

HT hành tinh thuộc hệ mặt trời

HT bán hàng trong siêu thị

HT quản lý sinh viên trường ĐH KTQD

HT quản lý giảng đường ĐH KTQD

1.1.1 HỆ THỐNG
PHẦN TỬ VÀ MỐI QUAN HỆ GIỮA CÁC PHẦN TỬ

KN: Phần tử là thành phần hợp thành hệ thống

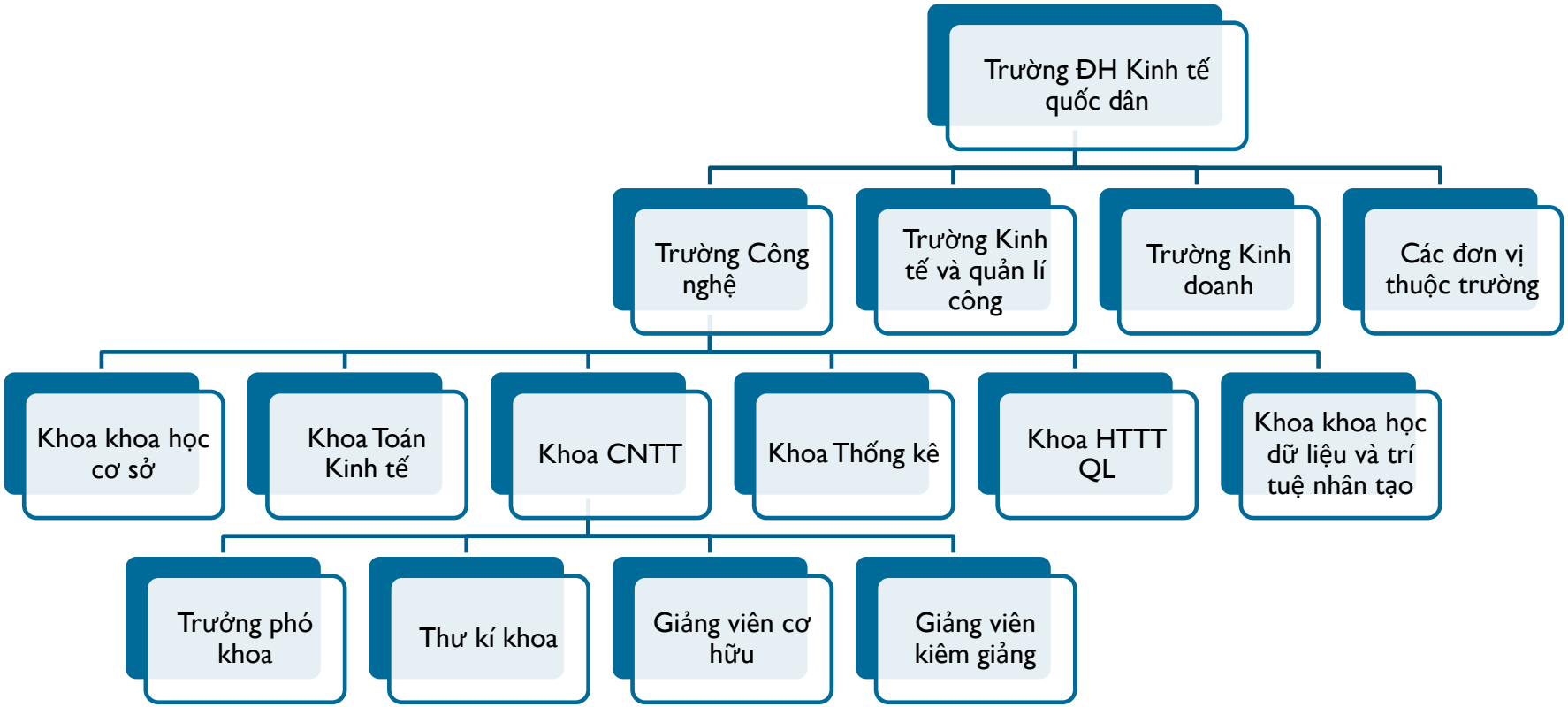
- Phần tử rất đa dạng
- Các phần tử khác biệt nhau về bản chất giữa các hệ thống và trong cùng hệ thống.
- Bản thân phần tử có thể rất phức tạp -> hệ thống khác
- Phân loại:
  - Phần tử cụ thể:
  - Phần tử trừu tượng:

1.1.1 HỆ THỐNG
PHẦN TỬ VÀ MỐI QUAN HỆ GIỮA CÁC PHẦN TỬ

Mối quan hệ giữa các phần tử:

- Các phần tử không tập hợp một cách ngẫu nhiên rời rạc mà giữa chúng luôn tồn tại mối quan hệ. Có thể tạo thành một cấu trúc hay tổ chức – Sơ đồ cơ cấu tổ chức.
- Phân loại quan hệ:
  - Quan hệ lâu dài:
  - Quan hệ tạm thời:
- Khi xem xét tổ chức của hệ thống phải xem các quan hệ lâu dài

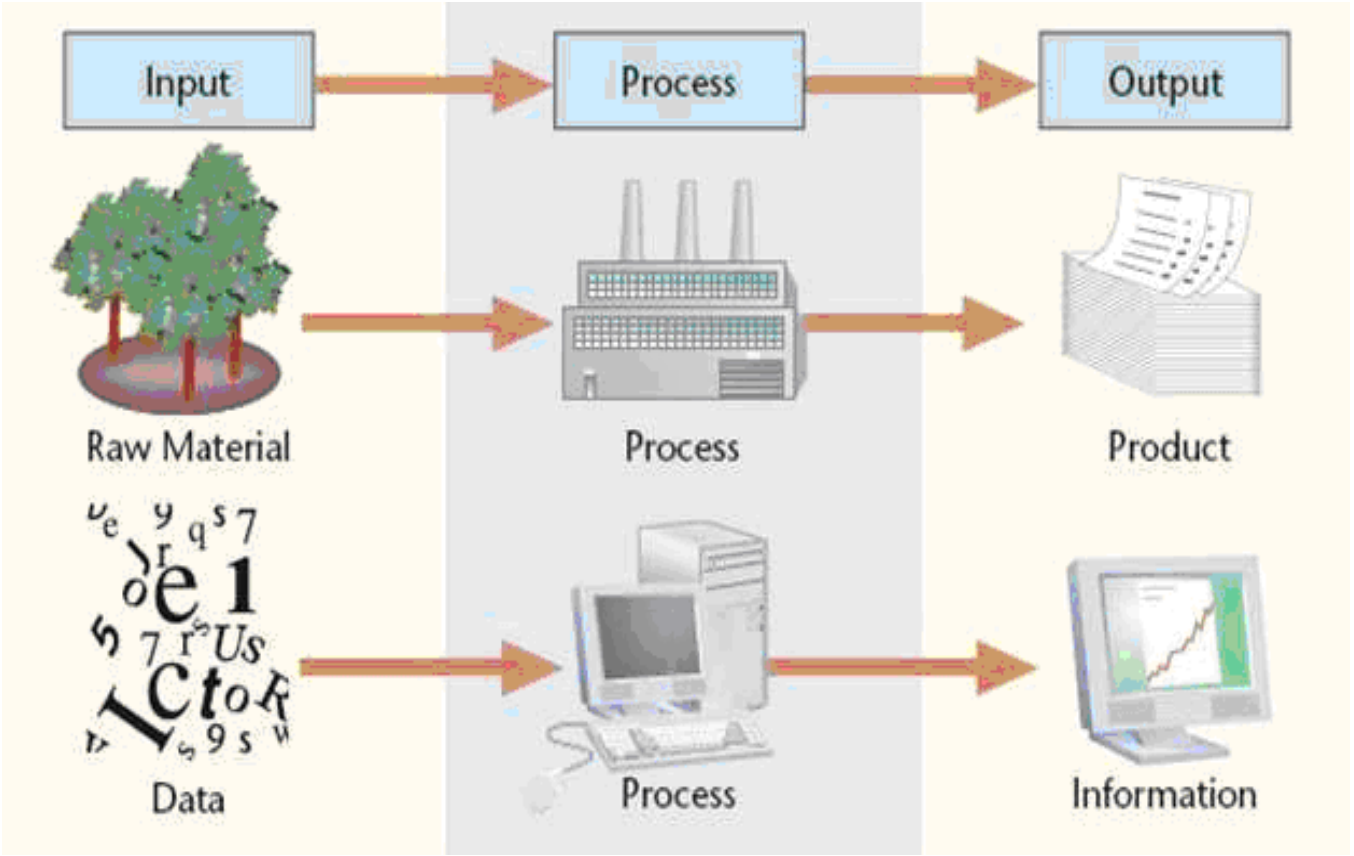
CƠ CẤU TỔ CHỨC NHÂN SỰ CỦA KHOA CNTT



1.1.1. HỆ THỐNG
SỰ HOẠT ĐỘNG VÀ MỤC ĐÍCH CỦA HT

- Hệ thống không ngừng biến động (thay đổi) thể hiện ở hai mặt:
  - Sự tiến triển: biến đổi về quy mô
  - Sự hoạt động: nghiệp vụ kinh doanh
- Mục đích chung của hệ thống thể hiện ở chỗ: Hệ thống nhận những đầu vào và biến đổi thành đầu ra nhất định.
- Hệ thống không hoạt động một cách độc lập mà còn quan hệ với hệ thống khác (hệ thống được đặt trong một môi trường là hệ thống khác)
- Môi trường hệ thống: Giữa HT và môi trường có đường biên

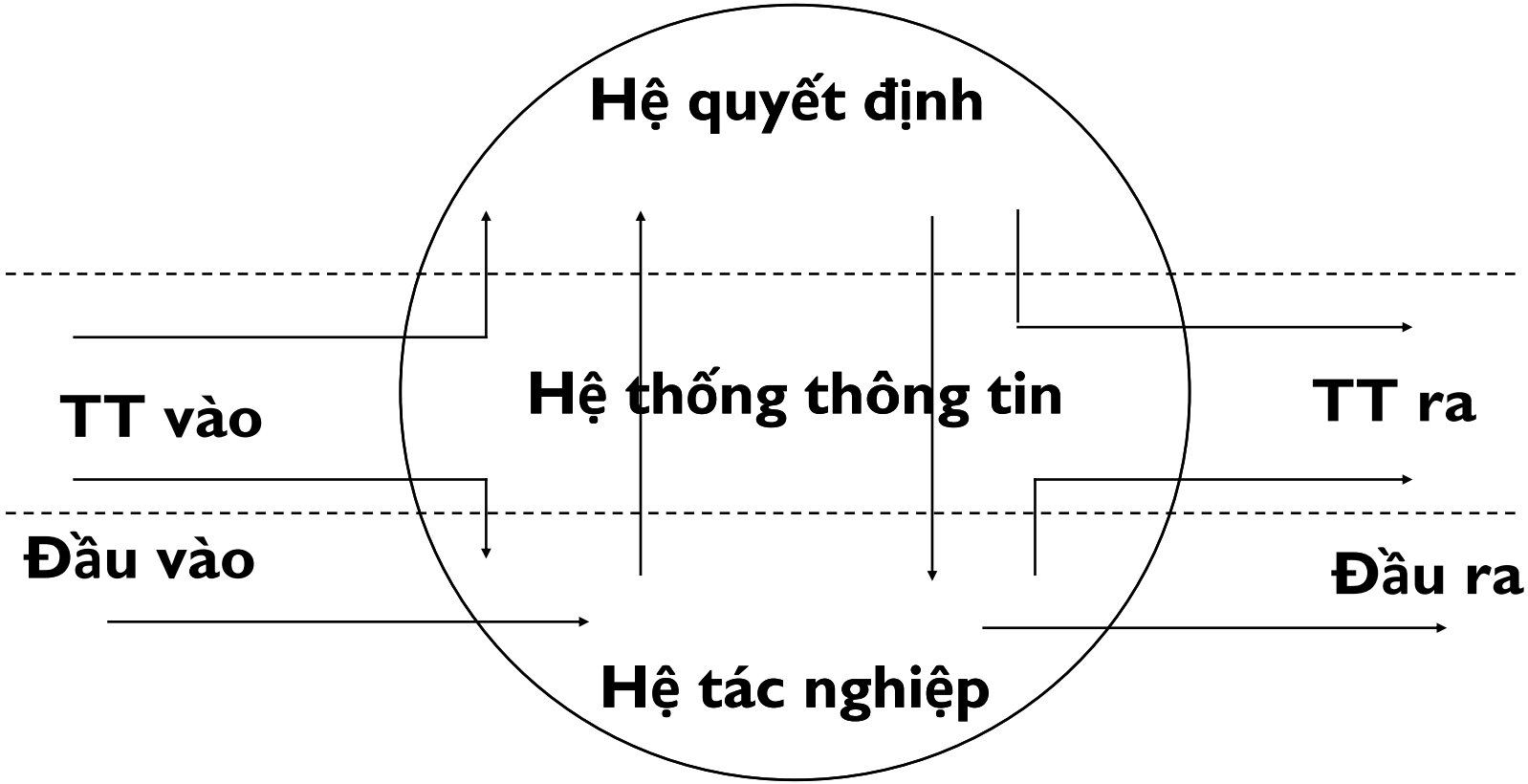
VÍ DỤ: MỤC ĐÍCH CỦA HT



1.1.2. HỆ THỐNG THÔNG TIN

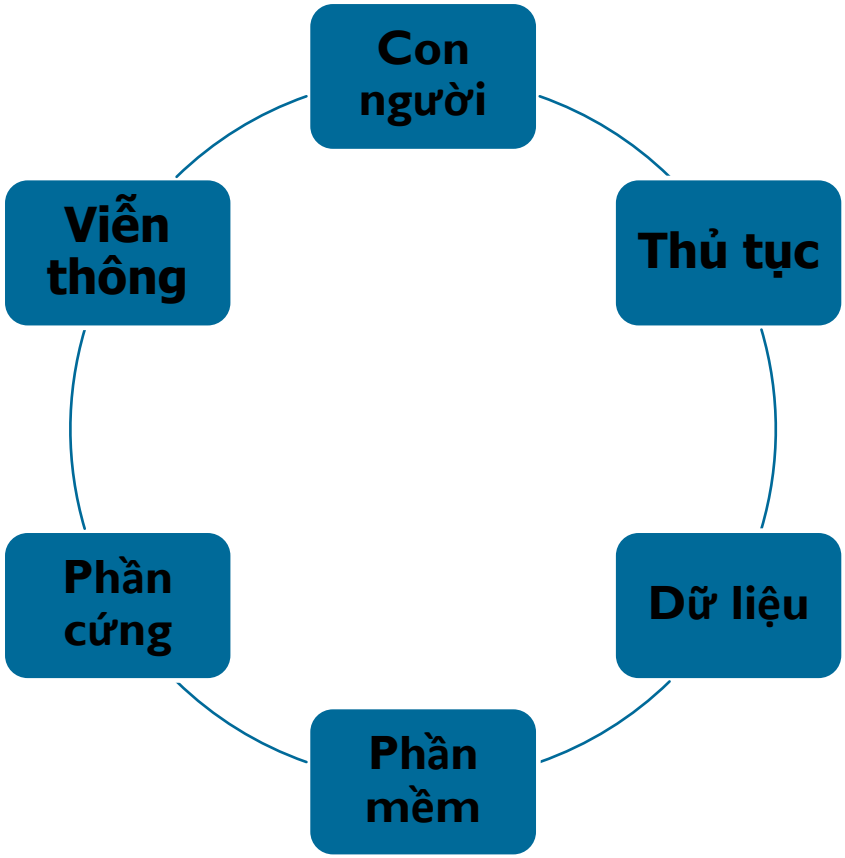
- Khái niệm:
  - Là HT làm nhiệm vụ xử lí thông tin
  - HTTT: là một HT bao gồm các yếu tố có mối quan hệ với nhau cùng làm nhiệm vụ, thu thập, xử lí, lưu trữ, phân phối dữ liệu và thông tin, cung cấp một cơ chế phản hồi để đạt được mục tiêu định trước
    - Đầu vào
    - Đầu ra
    - Xử lí
    - Thông tin phản hồi
  - HTTT dựa trên máy tính: là một HT tích hợp các yếu tố phần cứng, phần mềm, cơ sở dữ liệu, viễn thông, con người và các thủ tục cùng làm nhiệm vụ thu thập, xử lí, lưu trữ và biến đổi dữ liệu thành thông tin *(Giáo trình HTTT quản lí_KTQD)*

HỆ THỐNG VÀ HỆ THỐNG THÔNG TIN (HTTT)



Các hệ thống trong doanh nghiệp

THÀNH PHẦN CỦA HTTT



PHẦN CỨNG

- Bao gồm tất cả các ***tầng vật lí*** của HTTT: máy chủ, máy trạm, hệ thống mạng, các thiết bị truyền thông, cáp quang, các thiết bị di động, máy quét, các thiết bị chụp số, các cơ sở hạ tầng dựa trên công nghệ khác.
- Định luật Moore: "*Số lượng transistor trên mỗi đơn vị inch vuông sẽ tăng lên gấp đôi sau mỗi năm.*"

→ Phần cứng mạnh hơn, rẻ hơn.

PHẦN MỀM

- Phần mềm là tất cả các *chương trình* được dùng *điều khiển phần cứng* và *sản sinh ra các thông tin hay kết quả theo yêu cầu.*

→ Hệ thống phần mềm vận hành, quản lí các thành phần của phần cứng.

- ***Phần mềm hệ thống*** (System Software) gồm: phần mềm vận hành, phần mềm bảo mật, điều khiển các thiết bị. Thực hiện điều khiển dữ liệu, cung cấp bảo mật dữ liệu và quản lý hoạt động mạng máy tính.
- ***Phần mềm ứng dụng*** (Application Software): gồm các chương trình hỗ trợ các chức năng kinh doanh hàng ngày của doanh nghiệp, và cung cấp cho người dùng các thông tin họ yêu cầu → phần mềm mới || hệ thống phần mềm cũ.

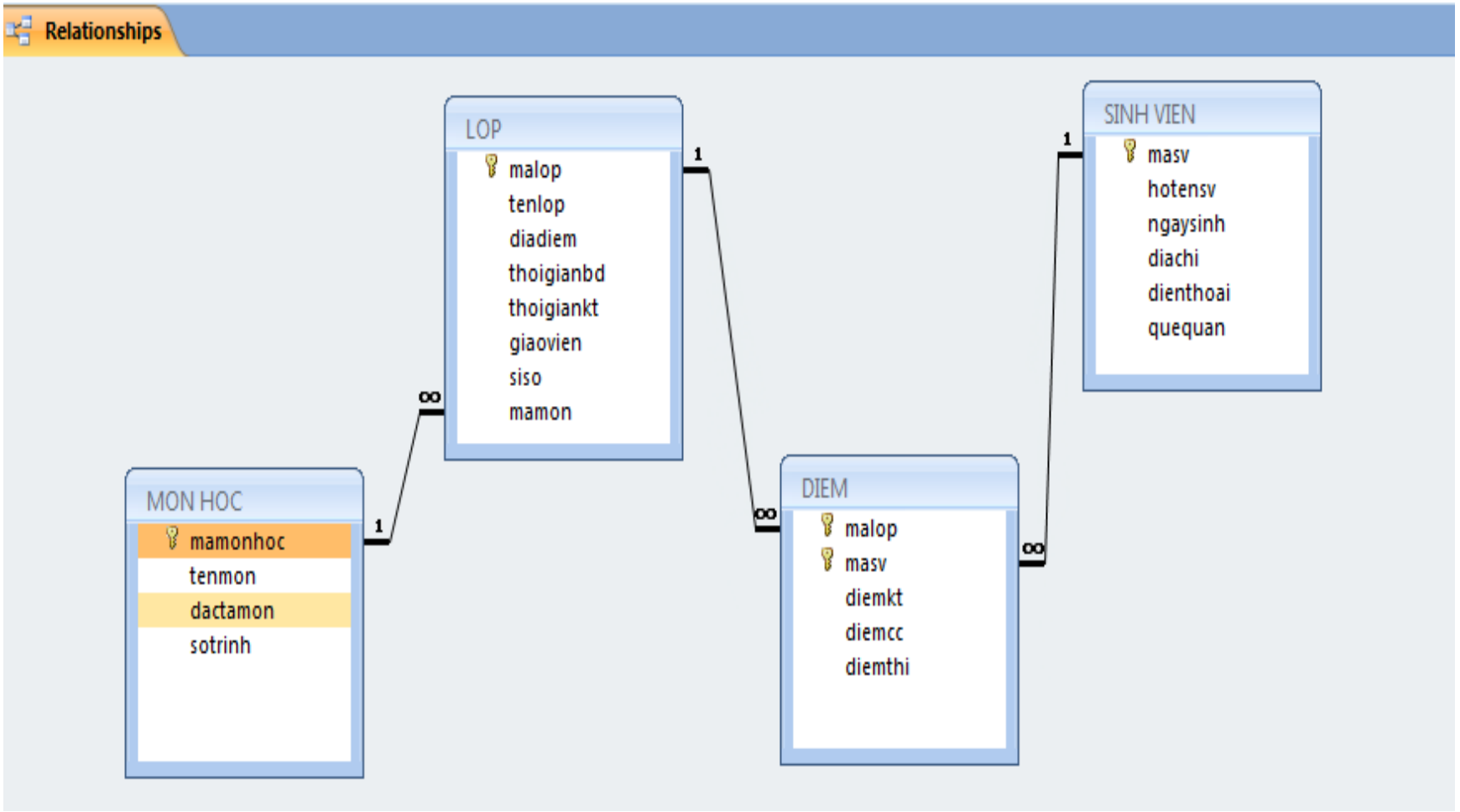
DỮ LIỆU

- Là các nguyên liệu thô của HTTT dùng để biến đổi thành các thông tin hữu ích.
- HTTT lưu trữ dữ liệu ở nhiều nơi (dạng bảng, document) bằng cách liên kết và trích rút ta có được các thông tin cần thiết.

Ví dụ: Quản lí điểm Sinh viên

hotensv	masv	tenmon	sotrinh	diemkt	diemcc	diemthi	diemhocphan

VÍ DỤ: QUẢN LÍ ĐIỂM SINH VIÊN



THỦ TỤC

- Mô tả tiến trình nghiệp vụ - hoạt động mà người dùng, nhà quản lý, và đội ngũ IT thực hiện để đạt được một kết quả cụ thể.
- Thủ tục - Quy trình được xây dựng theo thành khối nghiệp vụ trong HTTT vì chúng đại diện cho các hoạt động kinh doanh hàng ngày.
- Để phát triển HTTT thành công, người phát triển HT phải ***hiểu rõ quy trình nghiệp vụ*** này.

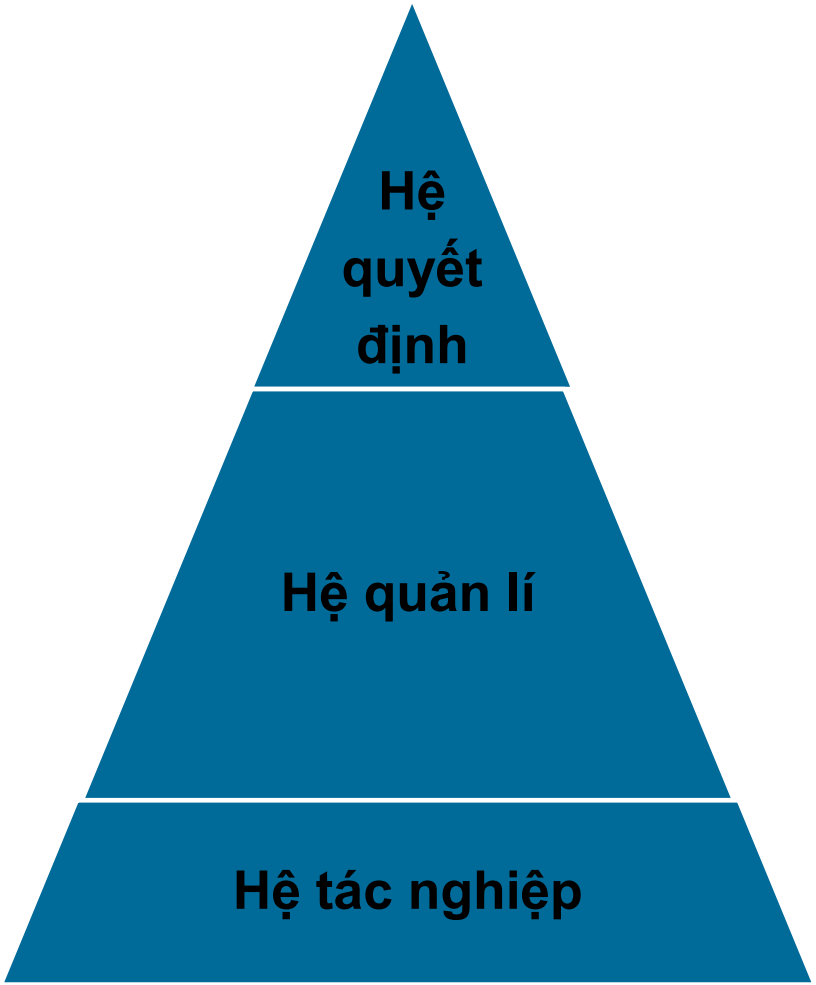
VIỄN THÔNG VÀ MẠNG MÁY TÍNH

- Viễn thông cho phép tổ chức liên kết các hệ thống máy tính thành mạng có hiệu quả
- Mạng máy tính có thể kết nối trong phạm vi toà nhà, thành phố, quốc gia, toàn cầu.
- Internet: mạng của các mạng cho phép trao đổi thông tin một cách tự do

CON NGƯỜI

- Con người bao gồm:
  - Người quản lý HT Administrator
  - Người dùng-Users
  - Các đối tượng trong và ngoài công ty có tương tác với HT
  - Đội ngũ IT (Phân tích viên, lập trình viên, kiểm thử viên,..)

NHÓM NGƯỜI DÙNG SỬ DỤNG THÔNG TIN



Hệ thống thông tin trong kinh doanh

1. Hệ hoạch định nguồn lực doanh nghiệp (***ERP-Enterprise Resource Planning***)
2. Hệ xử lý giao dịch (***TPS-Transaction Processing System***)
3. HTTT quản lý: (***MIS-Management Information System)***
4. Hệ hỗ trợ ra quyết định và hệ chuyên gia: ***(DSS - Decision Supported System, ES-Expert System***)
5. Hệ tăng hiệu suất người dùng: ***User Productivity Systems***
6. HTTT tích hợp: ***(ISI- Information System Intergration)***

1.2 VÒNG ĐỜI PHÁT TRIỂN HỆ THỐNG

- Vòng đời phát triển hệ thống (System Development Life Cycle - SDLC):
  - Là quá trình phát triển HT từ khi có ý tưởng phát triển đến khi hệ thống bị rỡ bỏ
  - Là khuôn mẫu xác định nhiệm vụ cần thực hiện ở mỗi giai đoạn trong quá trình phát triển hệ thống
- Các giai đoạn vòng đời phát triển *HT*:
  - Khảo sát hệ thống và xác lập dự án phát triển HT
  - Phân tích HT
  - Thiết kế HT
  - Xây dựng HT
  - Kiểm thử HT
  - Triển Khai HT
  - Bảo trì và Nâng cấp hệ thống

1.2.1 KHẢO SÁT VÀ XÁC LẬP DỰ ÁN PHÁT TRIỂN HT

- Nhiệm vụ:
  - Xem xét các nhu cầu của Doanh nghiệp, các nguồn tài nguyên có thể sử dụng, công nghệ và cộng đồng người dùng.
  - Tập hợp yêu cầu người dùng (khả thi, được mong muốn.. trên cả phương diện kĩ thuật và xã hội)
- Kết quả: Báo cáo nghiên cứu sơ bộ, báo cáo kết quả nghiên cứu tính khả thi. Thường tiến hành một phiên bản thô của lịch trình và kết hoạch sử dụng tài nguyên HT

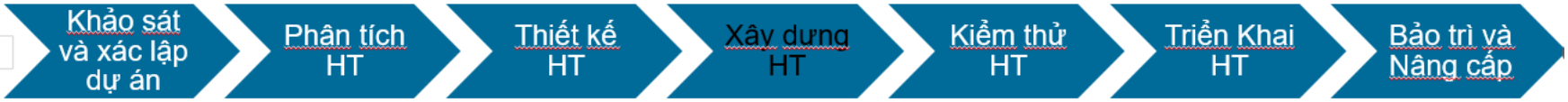
1.2.2 PHÂN TÍCH HT

- Nhiệm vụ:
  - Hiểu hệ thống cần xây dựng: HT làm gì? Dữ liệu sử dụng là gì?
  - Quy trình: nghiên cứu chi tiết HT hiện thời, nguyên lí hoạt động HT, nghiệp vụ HT có thể cải thiện, các chức năng HT và mối quan hệ giữa chúng, các tính năng mới, yêu cầu mới với HT tương lai, quy trình cũ → quy trình mới (khoảng cách và các bước để chuyển)
  - Yêu cầu nghiệp vụ → Yêu cầu hệ thống
- Kết quả: bản đặc tả phân tích yêu cầu HT

1.2.3 THIẾT KẾ HT

- Nhiệm vụ:
  - Trả lời câu hỏi: HT làm cách nào để thỏa mãn các yêu cầu đặt ra trong giai đoạn phân tích có tính đến các ràng buộc thực tế
  - Công việc thực hiện:
    - Thiết kế kiến trúc HT
    - Thiết kế giao diện
    - Thiết kế cơ sở dữ liệu
- Kết quả: bản đặc tả thiết kế HT mới

1.2.4 XÂY DỰNG HT



- Nhiệm vụ:
  - Xây dựng HT: thực hiện các yêu cầu được thiết kế sẵn
  - Công việc thực hiện:
    - Lập trình/ Code chương trình
    - Kiểm thử đơn vị:
- Kết quả: các chương trình của HT

1.2.5 KIỂM THỬ HT

- Nhiệm vụ: Kiểm thử đơn vị, tích hợp, hệ thống, chấp nhận người dùng
  - Mọi đơn vị chương trình con được tích hợp và chạy thử, sửa đổi nếu cần
  - Kiểm tra mọi chi tiết ghi trong đặc tả yêu cầu.
  - Dữ liệu cần được chọn lọc, kết quả chạy cần được phân tích để phát hiện sai lệch
  - Lập tài liệu hướng dẫn sử dụng HT
- Kết quả: Chương trình hoàn chỉnh, chạy tốt

1.2.6 VẬN HÀNH HỆ THỐNG

- Nhiệm vụ: Triển khai sang phía người dùng, đưa HT vào sử dụng
  - Triển khai thí điểm hoặc toàn bộ hệ thống: điều chỉnh
  - Đào tạo, hướng dẫn người dùng
- Kết quả: là một HT vận hành tốt trong thực tế

1.2.7 BẢO TRÌ VÀ NÂNG CẤP HT

- Nhiệm vụ: để sử dụng hệ thống hiệu quả
  - HT có thể trở nên lỗi thời khi môi trường sử dụng biến đổi nên việc sửa đổi và nâng cấp là cần thiết.
- Kết quả: Hệ thống vận hành tốt theo thời gian

CÂU HỎI ĐẶT RA

Có cần phương pháp để phát triển hệ thống không?

1.3 PHƯƠNG PHÁP PHÁT TRIỂN HT

- ***PP phát triển hệ thống là phương pháp luận để xây dựng và phát triển HT bao gồm: các lý thuyết, mô hình, phương thức, công cụ sử dụng trong quá trình xây dựng, phát triển HT.***
- Phương pháp PT HT là sự hợp thành của 3 yếu tố:
  - **Các khái niệm và mô hình:**
  - **Quy trình thực hiện**: Các bước đi lần lượt, các hoạt động cần làm
  - **Công cụ trợ giúp**: Phần mềm giúp (hỗ trợ) việc phân tích và thiết kế HT
- Nhiều tổ chức xây dựng phương pháp phát triển HT cho mình theo các giai đoạn trong vòng đời phát triển HT.

1.3.1 PHƯƠNG PHÁP HƯỚNG CHỨC NĂNG

- Functional System Analysis and Design
- Tập trung vào việc định nghĩa các chức năng (xử lí) của HT. Lấy chức năng làm trung tâm của việc định nghĩa HT
- Tập trung vào việc mô hình hóa hệ thống như là tập hợp các chức năng với các thông tin vào và ra của các chức năng đó → lấy chức năng làm đơn vị phân rã khi tiến hành PTTK HT
- Ra đời vào những năm 70, 80 của thế kỉ XX

1.3.1 PHƯƠNG PHÁP HƯỚNG CHỨC NĂNG

- Cài đặt hệ thống bằng các ngôn ngữ lập trình thủ tục (Pascal, C,...)
- Công cụ mô hình hóa hướng chức năng: công cụ đắc lực là biểu đồ phân cấp chức năng, biểu đồ luồng dữ liệu
- Nhược điểm: khó sửa chữa, khó nâng cấp, khó tái sử dụng

1.3.2 PHƯƠNG PHÁP HƯỚNG ĐỐI TƯỢNG

- Object Oriented System Analysis and Design
- Lấy đối tượng (gồm dữ liệu và chức năng) làm trung tâm, nguyên đơn cơ bản của hệ thống
- Ra đời vào những năm 90 của thế kỉ XX
- Cài đặt bằng ngôn ngữ lập trình hướng đối tượng: C++, Java, C#,...

1.3 PHƯƠNG PHÁP PHÁT TRIỂN HT

- Trình tự của các giai đoạn phát triển HT cũng như thời gian và nỗ lực dành cho từng giai đoạn là nhân tố quan trọng để phát triển HT thành công
- Lập kế hoạch tốt → phân tích tốt để thấu hiểu yêu cầu→ thiết kế tốt → xây dựng lên hệ thống tốt.

1.3 PHƯƠNG PHÁP PHÁT TRIỂN HT

- Mô hình và mô hình hóa:
  - **Mô hình**: là một dạng trừu tượng hóa bằng biểu đồ, hình ảnh... để biểu diễn của một hệ thống thực. Có thể được diễn tả ở:
    - Mức độ trừu tượng hóa nào đó
    - Theo một quan điểm, góc nhìn nào đó
    - Bởi một hình thức diễn tả hiểu được nào đó (văn bản, đồ thị, phương trình,...)
  - **Mô hình hóa**: là dùng mô hình để nhận thức và diễn tả hệ thống

??? *Quá trình phân tích thiết kế có được gọi là quá trình mô hình hóa hệ thống không?*

1.3 PHƯƠNG PHÁP PHÁT TRIỂN HT_MÔ HÌNH VÀ MÔ HÌNH HÓA

- Mục đích của mô hình hóa
  - Để hiểu
  - Để trao đổi
  - Để hoàn chỉnh
- Mô hình hóa tốt phải thỏa mãn các yêu cầu sau:
  - Dễ đọc
  - Dễ hiểu
  - Dễ trao đổi
  - Xác thực
  - Chặt chẽ
  - Đầy đủ
  - Dễ thực hiện cài đặt

1.3 PHƯƠNG PHÁP PHÁT TRIỂN HT_MÔ HÌNH VÀ MÔ HÌNH HÓA

- Thành phần
  - Hệ kí pháp (Notation) các khái niệm và mô hình
  - Một tiến trình (Process) các bước cần tiến hành, các sản phẩm (tài liệu, mô hình) qua từng giai đoạn, cách điều hành tiến trình, cách đánh giá chất lượng
  - Công cụ hỗ trợ (CASE) phần mềm hỗ trợ cho quá trình mô hình hóa, có khả năng:
    - Sản sinh ra các mô hình và biểu đồ
    - Biến đổi và điều chỉnh nhanh các mô hình và biểu đồ
    - Kiểm tra cú pháp, sự chặt chẽ, đầy đủ,...
    - Kiểm thử và đánh giá
    - Mô phỏng thực hiện mô hình

1.3 PHƯƠNG PHÁP PHÁT TRIỂN HT_MÔ HÌNH VÀ MÔ HÌNH HÓA

- Xu hướng chính của mô hình hóa:
  - Hướng chức năng
    - Từ những năm 1970
    - Lấy chức năng làm đơn vị phân rã HT
    - Phù hợp với phương pháp lập trình hướng thủ tục
  - Hướng đối tượng
    - Từ những năm 1990
    - Lấy đối tượng làm đơn vị phân rã
    - Phù hợp với các phương pháp lập trình hướng đối tượng

1.4 ĐỘI NGŨ PHÁT TRIỂN HT

- Dự án phát triển hệ thống
- Đội phát triển HT: những người xây dựng phương án phát triển hệ thống, phân tích thiết kế hệ thống, người xây dựng HT, người hỗ trợ, đào tạo, khuyến khích người dùng sử dụng HT.
- Đòi hỏi nhiều kỹ năng: *có kĩ thuật*, *có nghiệp vụ, có khả năng giao tiếp, có khả năng quản lí, có đạo đức nghề nghiệp,..*

1.4 ĐỘI NGŨ PHÁT TRIỂN HT

- Các vị trí trong đội phát triển HT
  - Phân tích viên nghiệp vụ: Business Analyst - BA
  - Phân tích viên HT: System Analyst
  - Thiết kế viên HT: System Designer
  - Lập trình viên: Coder/ Developer/ Programmer
  - Kiểm thử viên: Tester
  - Nhà quản trị HT: System Aministrator
  - Nhà quản lí dự án phát triển HT: Project Manager

1.4 ĐỘI NGŨ PHÁT TRIỂN HT

- ***Phân tích viên nghiệp vụ***: hiểu nghiệp vụ HT, phát triển ý tưởng và gợi ý nghiệp vụ mới, thiết kế các chức năng mới cũng các chính sách đi kèm, trình bày và thuyết phục các bên liên quan, thực hiện trong giai đoạn *phân tích* và định nghĩa thiết kế
- ***Phân tích viên HT***: có kinh nghiệm trong phân tích, thiết kế, lập trình HT, đảm bảo cơ sở hạ tầng kĩ thuật (phần cứng, phần mềm, mạng, cơ sở dữ liệu,…) của doanh nghiệp tương thích với HT trong tương lai.
- ***Thiết kế viên***: am hiểu nghiệp vụ, có tính sáng tạo, kĩ thuật thiết kế tốt, vận dụng tốt các công cụ thiết kế, trình bày ý tưởng tốt. (thiết kế viên giao diện, thiết kế CSDL, kiến trúc HT)

1.4 ĐỘI NGŨ PHÁT TRIỂN HT

- ***Lập trình viên-Coder***: thông thạo ngôn ngữ lập trình → xây dựng chương trình (test bước đầu).
- ***Kiểm thử viên-Tester***: kiểm thử các đơn vị-unit hệ thống, kiểm thử chương trình, kiểm thử tích hợp, kiểm thử chấp nhận người dùng, yêu cầu tính cẩn thận, chăm chỉ..
- ***Nhà quản trị mạng***: hiểu về phần cứng hệ thống, các hệ thống mạng không dây và có dây, an toàn mạng,...
- ***Hỗ trợ người dùng***: tập trung vào con người và các vấn đề quản lí khi cài đặt HT nhằm đảm bảo việc cung cấp tài liệu và hướng dẫn người dùng.
- ***Nhà quản lí dự án***: đảm bảo dự án phát triển HT hoàn thành đúng mục tiêu đề ra, theo đúng thời gian, ngân sách dự kiến.

1.4 ĐỘI NGŨ PHÁT TRIỂN HT

- Yêu cầu về kĩ năng:
  - Có khả năng phân tích
  - Có kiến thức kĩ thuật
  - Có khả năng quản lí
  - Có khả năng giao tiếp
- Có tư duy hệ thống, tư duy quy trình,...
- Có khả năng liên lạc và gắn kết với các bên liên quan: người dùng, lập trình viên, các chuyên gia hệ thống khác,…

ĐẶC ĐIỂM CỦA ĐỘI PHÁT TRIỂN HT THÀNH CÔNG

- Có nền tảng và kĩ năng đa dạng
- Có khả năng chịu đựng áp lực về công việc, thời gian
- Có khả năng giao tiếp
- Tin tưởng, kính trọng lẫn nhau
- Có chế độ lương thưởng, trách nhiệm, thăng chức rõ ràng
- Tinh thần làm việc nhóm, đặt sở thích cá nhân dưới lợi ích của nhóm

BÀI TẬP BUỔI 2_1

BT1. Hãy chọn một vị trí công việc trong đội phát triển hệ thống mà em quan tâm và nêu trách nhiệm và yêu cầu của vị trí đó.

1.5 NGÔN NGỮ MÔ HÌNH HÓA THỐNG NHẤT UML

- Giới thiệu về ngôn ngữ mô hình hóa thống nhất UML
- Các góc nhìn của UML
- Các biểu đồ được sử dụng trong UML
- Các công cụ mô hình hóa miễn phí

1.5.1 NGÔN NGỮ MÔ HÌNH HÓA THỐNG NHẤT UML

- Là ngôn ngữ đặc tả hình thức được dùng để đặc tả, trực quan hóa, và tư liệu hóa phần mềm hướng đối tượng.
  - Nó không phải là một phương pháp chỉ là ngôn ngữ mô hình hóa (tập các phần tử và tập các quy luật riêng dùng để mô hình hóa)
  - Sử dụng kết hợp với quy trình phát triển phần mềm – Tiến trình RUP (Rational Unified Process) được xem như là phương pháp luận phát triển HT có ngôn ngữ mô hình hóa là UML: cho biết mô hình nào được tạo ra và khi nào được tạo ra.

1.5.1 NGÔN NGỮ MÔ HÌNH HÓA THỐNG NHẤT UML

- UML là ngôn ngữ dùng để trực quan hóa - mô hình hóa trực quan (visualizing)
  - Xây dựng các mô hình với kí hiệu chuẩn, riêng, rõ ràng duy nhất, được trực quan hóa
  - Dùng làm tư liệu để hiểu, trao đổi, sữa chữa..
- ULM là ngôn ngữ dùng để chi tiết hóa – đặc tả (Specifying)
  - Để xây dựng các mô hình một cách chi tiết rõ ràng, đầy đủ dưới các góc độ khác nhau
  - Chi tiết hóa các quyết định quan trọng trong phân tích, thiết kế, thực thi một HT phần mềm

1.5.1 NGÔN NGỮ MÔ HÌNH HÓA THỐNG NHẤT UML

- UML là ngôn ngữ sinh ra mã ở dạng nguyên mẫu (constructing)
  - Ánh xạ tới một ngôn ngữ lập trình cụ thể: C++, java,..
  - Ánh xạ sang các bảng trong CSDL quan hệ hay hướng đối tượng
  - Yêu cầu – Mô hình – Mã lệnh: Có cơ chế đồng bộ giữa mô hình và mã lệnh
- UML là ngôn ngữ dùng để lập và cung cấp tài liệu (documenting)
  - Ghi chép về yêu cầu HT
  - Tài liệu về phân tích thiết kế
  - Các nguyên mẫu

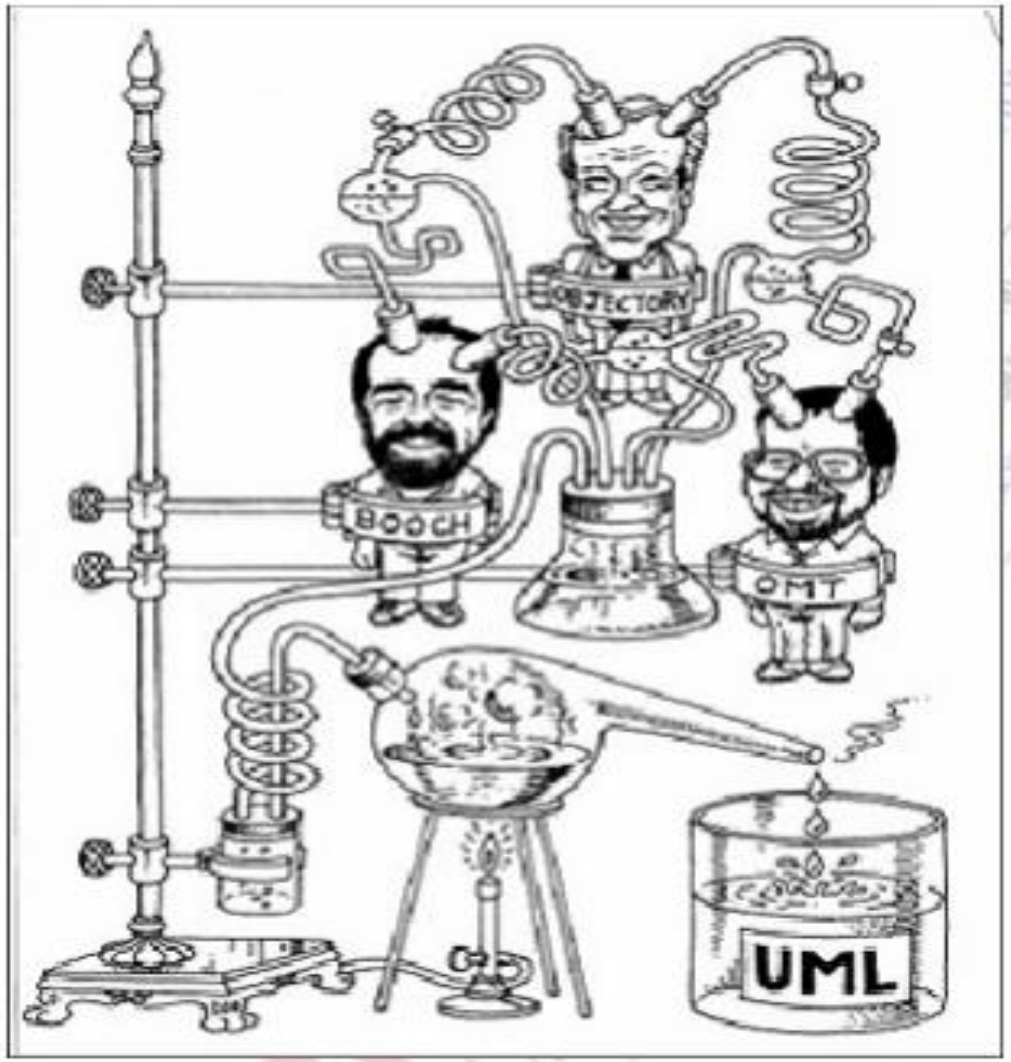
1.5.1 NGÔN NGỮ MÔ HÌNH HÓA THỐNG NHẤT UML

- Có thể sử dụng trong bất kì tiến trình phát triển HT
- Xuyên suốt vòng đời phát triển HT
- Được sử dụng bởi các công nghệ cài đặt khác nhau

1.5.1 NGÔN NGỮ MÔ HÌNH HÓA THỐNG NHẤT UML

- Lịch sử phát triển
  - 1975-1990: có nhiều Ngôn ngữ MHH hướng đối tượng được phát triển
  - 1990-1994: Hơn 50 phương pháp phát triển hướng đối tượng trong đó có 3 phương pháp nổi tiếng:
    - OOD - Object Oriented Design (Grady Booch)
    - OOSE - Object Oriented Software Engineering (Ivar Jacobson)
    - OMT - Object Modeling Technique (Jim Rumbaugh)

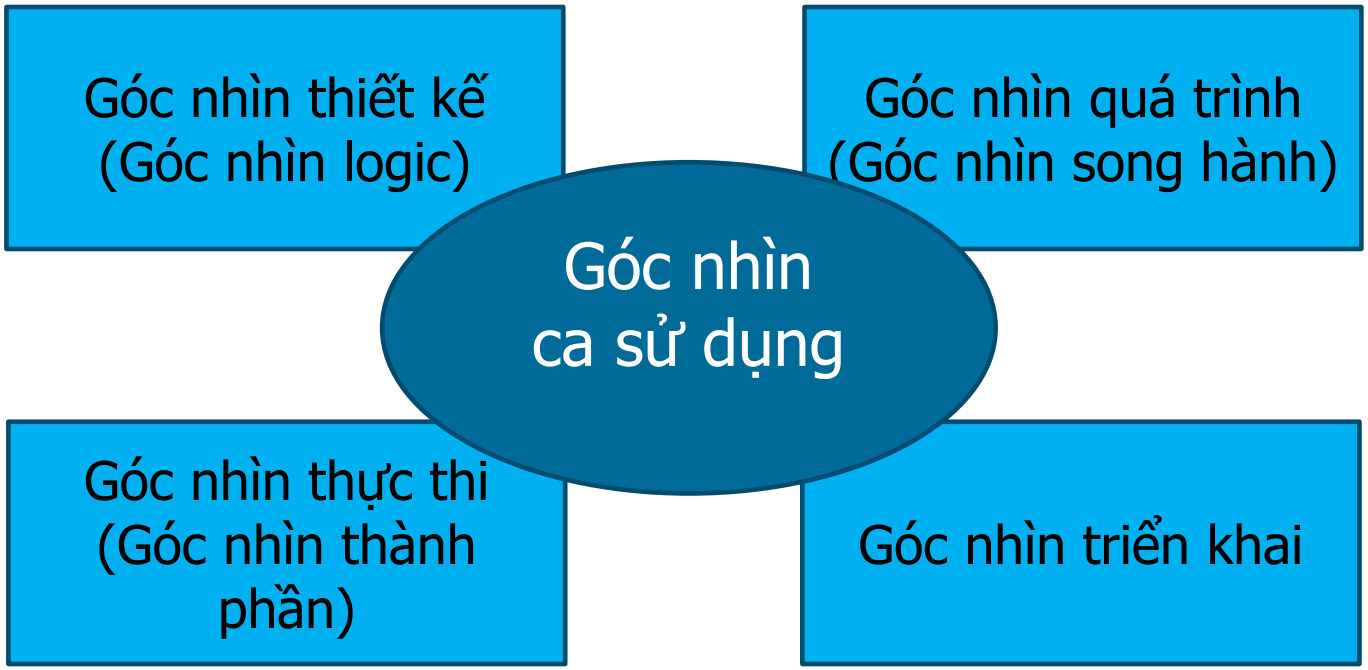
SỰ RA ĐỜI CỦA UML



1.5.1 NGÔN NGỮ MÔ HÌNH HÓA THỐNG NHẤT UML
LỊCH SỬ PHÁT TRIỂN UML

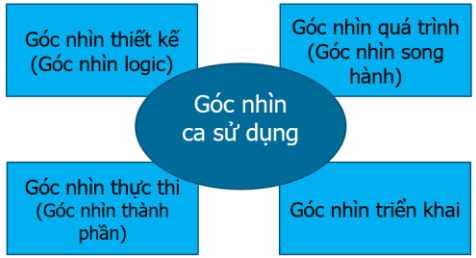
- 10/1994: Rumbaugh và Booch tiến hành dự án UML ở Rational, xây dựng một phương pháp hợp nhất trên cơ sở hai phương pháp Booch 93 và OMT-2
- 1995: Jacobson gia nhập dự án
- 10/1995: Phác thảo UML, phiên bản 0
- 6/1996: Phiên bản UML 0.9
- 1/1997: IBM và SoftTeam kết hợp với các thành viên => Phiên bản 1.1
- 14/11/1997: UML 1.1 được OMG (Object Management Group) công nhận là chuẩn
- 6/1998: UML 1.2
- 10/1998: UML 1.3
- 5/2001: UML 1.4
- 6/2003: UML 2.0

1.5.2 CÁC GÓC NHÌN CỦA UML



1.5.2 CÁC GÓC NHÌN CỦA UML

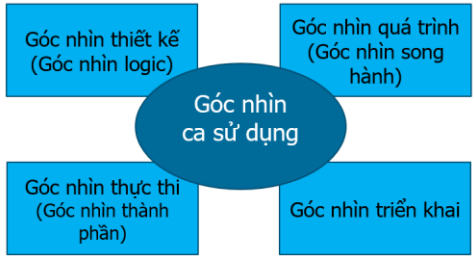
- **Góc nhìn ca sử dụng (Use case view)**



  - Là góc nhìn từ ngoài nhìn vào hệ thống
  - Là cách nhìn của *người dùng cuối*, *người phân tích*, *người kiểm thử*
  - Không phản ánh tổ chức bên trong, mà chỉ làm rõ các chức năng chính/quan trọng mà hệ thống phải đáp ứng cho người dùng
  - Sắc thái tĩnh: Biểu đồ ca sử dụng (Use case diagram)
  - Sắc thái động: Biểu đồ giao tiếp (Communication diagram), Biểu đồ máy trạng thái (State diagram), và Biểu đồ hoạt động (Activity diagram)

1.5.2 CÁC GÓC NHÌN CỦA UML

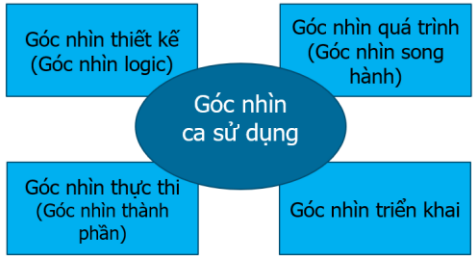
- **Góc nhìn thiết kế (Design view)**



  - Còn được gọi là góc nhìn logic (Logical view)
  - Là góc nhìn vào bên trong (cấu trúc) hệ thống, cho thấy các nhiệm vụ của hệ thống
  - Là cách nhìn của *người thiết kế hệ thống*
  - Sắc thái tĩnh: Biểu đồ lớp (Class diagram), Biểu đồ đối tượng (Object diagram)
  - Sắc thái động: Biểu đồ giao tiếp (Communication diagram), Biểu đồ máy trạng thái (State diagram), Biểu đồ hoạt động (Activity diagram)

1.5.2 CÁC GÓC NHÌN CỦA UML

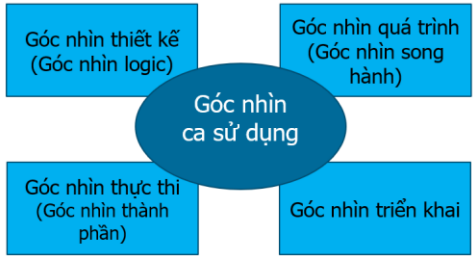
- **Góc nhìn quá trình (Process view)**



  - Còn được gọi là góc nhìn song hành
  - Phản ánh các quá trình điều khiển, các quá trình thực hiện, cho thấy sự hoạt động đồng bộ của hệ thống
  - Được thể hiện (sử dụng) với các biểu đồ như trong Góc nhìn thiết kế, tập trung vào các lớp chủ động
    - Lớp chủ động: Lớp biểu diễn cho các quá trình điều khiển và quá trình thực hiện

1.5.2 CÁC GÓC NHÌN CỦA UML

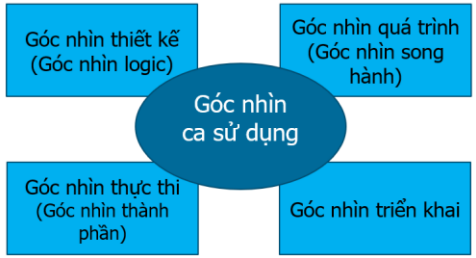
- **Góc nhìn thành phần (**Component view – góc nhìn thực thi - Implementation view)



  - Là góc nhìn đối với dạng phát hành của phần mềm
  - Cho thấy các thành phần và tập tin tương đối độc lập, có thể lắp ráp để hệ thống chạy được
  - Sắc thái tĩnh: Biểu đồ thành phần (Component diagram)
  - Sắc thái động: Biểu đồ giao tiếp (Communication diagram), Biểu đồ máy trạng thái (State diagram), Biểu đồ hoạt động (Activity diagram)

1.5.2 CÁC GÓC NHÌN CỦA UML

- **Góc nhìn triển khai (Deployment view)**



  - Là góc nhìn về kiến trúc phần cứng và nền tảng hạ tầng mà trên đó hệ thống được triển khai
  - Chỉ rõ sự phân bố, sắp đặt các thành phần của hệ thống trên các đơn vị phần cứng và nền tảng hạ tầng
  - Sắc thái tĩnh: Biểu đồ triển khai (Deployment diagram)
  - Sắc thái động: Biểu đồ giao tiếp (Communication diagram), Biểu đồ máy trạng thái (State diagram), Biểu đồ hoạt động (Activity diagram)

1.5.2 CÁC GÓC NHÌN CỦA UML

- Mỗi vai trò trong tiến trình phát triển hệ thống (vd: người phân tích, thiết kế, tích hợp, kiểm thử, người dùng cuối,…) thường chỉ quan tâm tới một góc nhìn nào đó của hệ thống
- 5 góc nhìn có sự liên hệ và bổ trợ lẫn nhau
- Góc nhìn ca sử dụng (Use case view) có ảnh hưởng (liên quan) đến 4 góc nhìn còn lại
- Mỗi biểu đồ có thể thuộc nhiều góc nhìn và mỗi góc nhìn có thể có nhiều biểu đồ mô tả.

1.5.3 CÁC BIỂU ĐỒ CỦA UML

- **Các biểu đồ về cấu trúc**:
  - *Biểu đồ lớp (Class diagram)*
  - *Biểu đồ đối tượng (Object diagram)*
  - *Biểu đồ triển khai (Deployment diagram)*
  - *Biểu đồ gói (Package diagram)*
  - *Biểu đồ thành phần (Component diagram)*
  - Biểu đồ cấu trúc đa hợp (Composite structure diagram)
- **Các biểu đồ về hành vi**:
  - *Biểu đồ ca sử dụng (Use case diagram)*
  - *Biểu đồ hoạt động (Activity diagram)*
  - *Biểu đồ trình tự (Sequence diagram)*
  - *Biểu đồ giao tiếp (Communication diagram)*
  - *Biểu đồ máy trạng thái (State diagram)*
  - Biểu đồ thời gian (Timing diagram)
  - Biểu đồ tổng quan tương tác (Interaction overview diagram)

1.5.4 CÁC CÔNG CỤ MÔ HÌNH HÓA MIỄN PHÍ

- **ArgoUML** (http://argouml.tigris.org/)
- **BOUML** (http://www.bouml.fr/)
- **NClass** (http://nclass.sourceforge.net/)
- **Umbrello UML Modeller** (https://umbrello.kde.org/)
- **WhiteStarUML** (https://sourceforge.net/projects/whitestaruml/)
- **Modelio** (https://www.modelio.org/)
- **Dia** (https://wiki.gnome.org/Apps/Dia/)
- **Papyrus** (http://www.eclipse.org/papyrus/)
- **Acceleo** (https://www.eclipse.org/acceleo/)

1.5.4 CÁC CÔNG CỤ MÔ HÌNH HÓA MIỄN PHÍ

- **UML Designer** (http://www.umldesigner.org/)
- **yEd** (http://www.yworks.com/)
- **UMLet** (http://www.umlet.com/)
- **Violet** (http://alexdp.free.fr/violetumleditor/page.php)
- **PlantUML** (http://plantuml.com/)
- **Astah** (http://astah.net/editions/community)
- **Visual Paradigm** (https://www.visual-paradigm.com/solution/freeumltool/)
- **MetaUML** (https://github.com/ogheorghies/MetaUML/wiki)
- **TinyUML** (https://sourceforge.net/projects/tinyuml/)
- **Eclipse UML2 Tools** (https://www.eclipse.org/)
- **NetBeans IDE** (https://netbeans.org/)
- **Oracle Jdeveloper** (http://www.oracle.com/technetwork/developertools/jdev/overview/index.html)

CÁC CÔNG CỤ THIẾT KẾ GUI MIỄN PHÍ

- **Pencil** (https://pencil.evolus.vn/)
- **Wireframe** (https://wireframe.cc/)
- **InVision** (https://www.invisionapp.com/studio)
- **Adobe XD** (https://www.adobe.com/in/products/xd.html)
- **Lunacy** (https://icons8.com/lunacy)
- **Fluid UI** (https://www.fluidui.com/)
- **Sketch UX Kit** (http://www.mikolajdobrucki.com/sketchuxkit/)

1.6 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM

- Giới thiệu về quy trình phát triển phần mềm
- Một số mô hình phát triển phần mềm thông dụng
- Quy trình RUP

1.6.1 GIỚI THIỆU VỀ QUY TRÌNH PHÁT TRIỂN PHẦN MỀM

- KN Quy trình PTPM (Software development process): Một tập có cấu trúc (có trật tự) các hoạt động cần thiết để phát triển một hệ thống phần mềm
- Có nhiều quy trình PTPM: Thác nước (Waterfall), Nguyên mẫu (Prototyping), Xoắn ốc (Spiral),…
- Không tồn tại một quy trình PTPM lý tưởng duy nhất phù hợp cho mọi bài toán, yêu cầu thực tế

1.6.1 GIỚI THIỆU VỀ QUY TRÌNH PHÁT TRIỂN PHẦN MỀM

Các yếu tố lựa chọn Quy trình PTPM:

- Kiểu của hệ thống phần mềm cần được xây dựng
  - Xây dựng mới từ đầu >< Nâng cấp, chỉnh sửa hệ thống có sẵn
  - Kiểu thông thường, phổ biến >< Kiểu tùy biến, đặc thù
  - Các yêu cầu phần mềm xác định >< Các yêu cầu phần mềm thay đổi (nhanh chóng)
  - Hệ thống trọng yếu (critical) >< Hệ thống nghiệp vụ, kinh doanh
- Quy mô của dự án PTPM, Quy mô (nguồn lực) của nhóm PTPM, Thời gian thực hiện dự án PTPM, Kinh phí thực hiện dự án PTPM
- Các đặc điểm của nhóm PTPM
  - Kinh nghiệm, Động cơ (+ sự khuyến khích), Thái độ làm việc (nỗ lực)

1.6.1 GIỚI THIỆU VỀ QUY TRÌNH PHÁT TRIỂN PHẦN MỀM

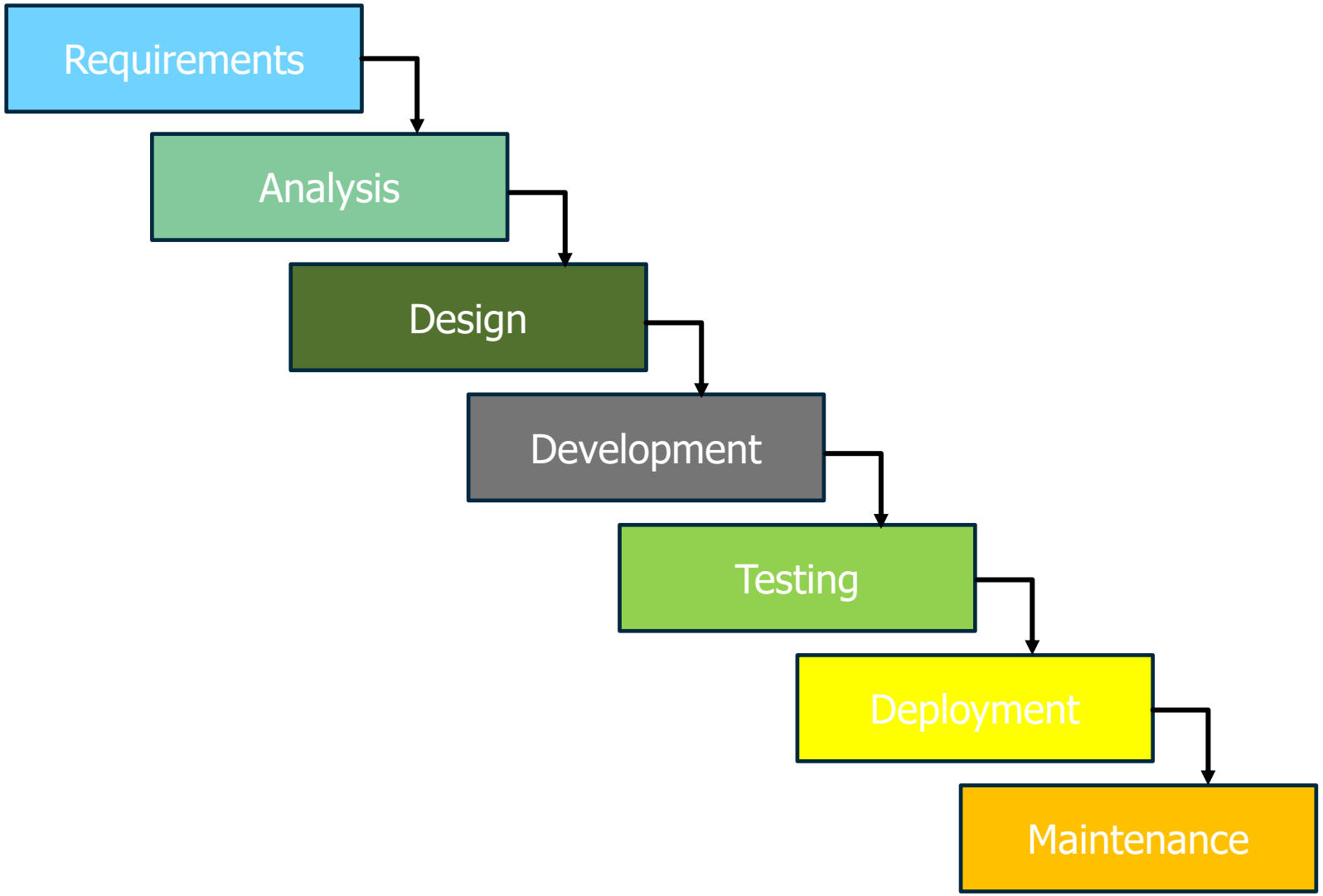
Các hoạt động cơ bản:

- Xác định yêu cầu (Requirement)
- Phân tích (Analysis)
- Thiết kế (Design)
- Thực hiện, Xây dựng (Development)
- Kiểm thử (Testing)
- Triển khai (Deployment)
- Bảo trì (Maintenance)

1.6.2 MỘT SỐ MÔ HÌNH PHÁT TRIỂN PM THÔNG DỤNG

- Mô hình thác nước (Waterfall model)
- Mô hình nguyên mẫu (Prototyping model)
- Mô hình xoắn ốc (Spiral model)
- Mô hình nhanh lẹ (Agile model): Scrum & RAD
- Mô hình hợp nhất (Unified model)

MÔ HÌNH THÁC NƯỚC



MÔ HÌNH THÁC NƯỚC

- Được giới thiệu bởi Winston Royce vào năm 1970, và hiện tại vẫn là mô hình được sử dụng phổ biến nhất trong các dự án PTPM
- Việc PTPM dựa trên một tập hợp các giai đoạn (phases) có thứ tự liên tiếp
  - Trật tự (thứ tự) của các giai đoạn là xác định, và các kết quả (đầu ra – out put) của một giai đoạn trước sẽ được sử dụng làm đầu vào (input) cho các giai đoạn sau
- Một khi tiến trình PTPM kết thúc và hệ thống phần mềm được bàn giao (signed off) cho khách hàng, thì hệ thống phần mềm sẽ không thể được thay đổi, điều chỉnh
  - Tiến trình PTPM chỉ có thể được mở lại (để đáp ứng các điều chỉnh, thay đổi) thông qua một quy trình thực hiện thay đổi chính thức (a formal change process)

MÔ HÌNH THÁC NƯỚC

- Đặc điểm quan trọng nhất của Quy trình thác nước: **các giai đoạn (phases) không giao nhau, không lặp lại** (trong một tiến trình PTPM)
  - Giai đoạn Thiết kế (Design) không thể bắt đầu cho đến khi giai đoạn Phân tích (Analysis) được hoàn thành, và Giai đoạn Kiểm thử (Testing) không thể bắt đầu cho đến khi giai đoạn Thực hiện, lập trình (Implementation) được hoàn thành

MÔ HÌNH THÁC NƯỚC

- Các ưu điểm
  - Là quy trình PTPM đơn giản, dễ hiểu, và dễ sử dụng
  - Các tài liệu được hoàn thành sau mỗi giai đoạn
  - Các yêu cầu phần mềm được cung cấp sớm cho các người kiểm thử (the testers)
  - Cho phép người quản lý dự án (Project Manager – PM) lập kế hoạch và kiểm soát thực hiện một cách chặt chẽ
  - Quy trình này cũng rất nổi tiếng và được biết bởi cả những người không chuyên về PTPM, giúp nó dễ dàng được dùng để trao đổi

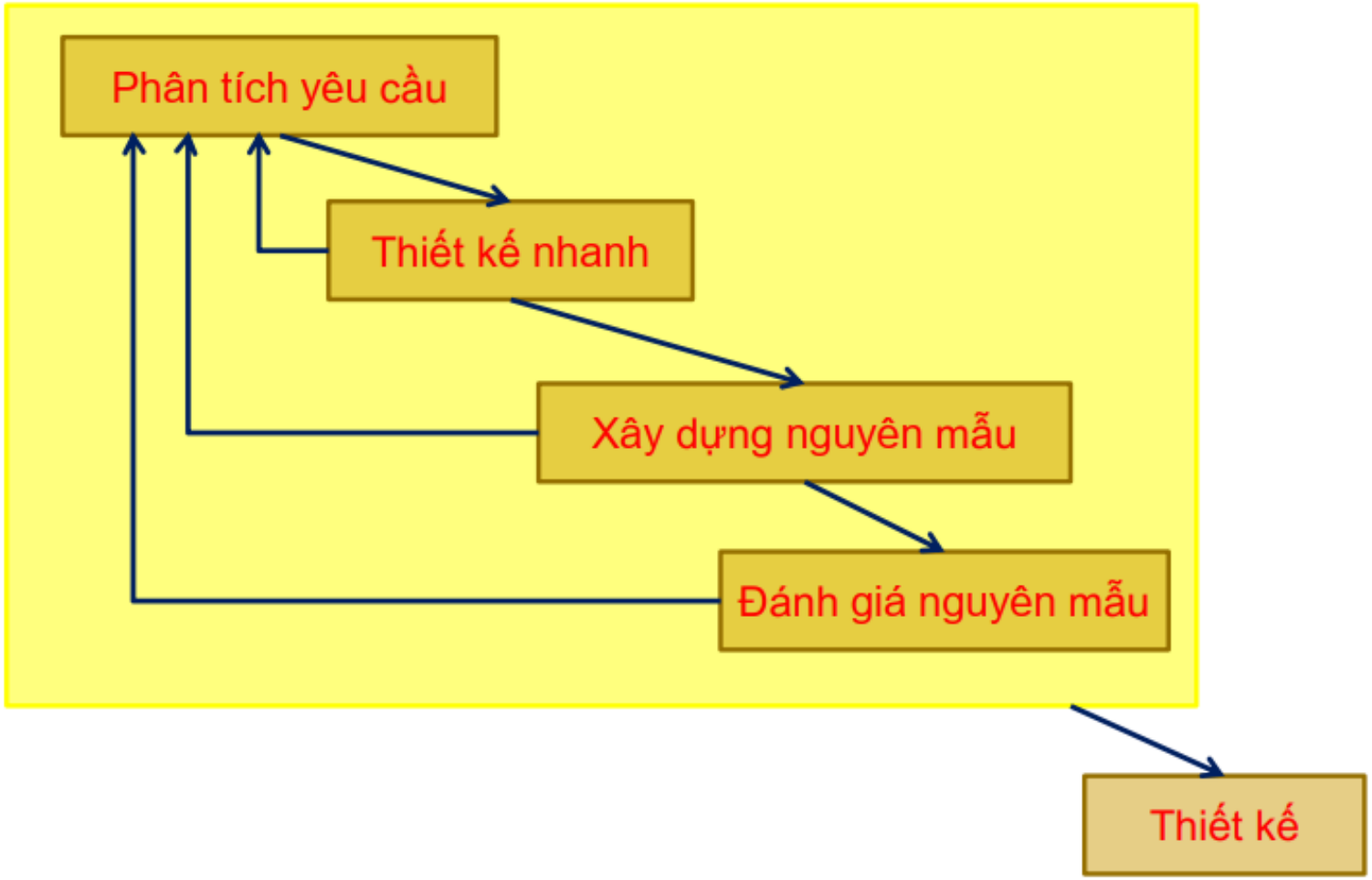
MÔ HÌNH THÁC NƯỚC

- Các nhược điểm
  - Chỉ phù hợp đối với các bài toán thực tế **khi mà các yêu cầu phần mềm được xác định rõ ràng, đầy đủ và cố định từ đầu** (trước giai đoạn Thiết kế
  - Không phù hợp đối với các dự án kéo dài và tiếp diễn lâu
  - Có thể có nhiều nguy cơ (risk) và không chắc chắn (uncertainty)
  - Khó (không thể) sớm có các kết quả (phiên bản) ban đầu của phần mềm

MÔ HÌNH THÁC NƯỚC

- Khi nào nên sử dụng mô hình thác nước?
  - **Khi các yêu cầu phần mềm được xác định rõ ràng, đầy đủ và cố định**
  - Định nghĩa về sản phẩm (hệ thống phần mềm) không thay đổi
  - Các công nghệ liên quan cần thiết được nắm vững
  - Các nguồn lực và kinh nghiệm của nhóm PTPM đủ đáp ứng
  - Thời gian thực hiện dự án ngắn (không kéo dài)

MÔ HÌNH NGUYÊN MẪU



MÔ HÌNH NGUYÊN MẪU

- Thay vì cố định các yêu cầu trước khi tiến hành thiết kế hoặc thực hiện (lập trình), **một (hoặc một số các) nguyên mẫu (prototype) được xây dựng để hiểu chính xác các yêu cầu phần mềm**
- Mỗi nguyên mẫu (prototype) được xây dựng dựa trên các yêu cầu phần mềm hiện thời (thu được từ đánh giá các nguyên mẫu trước)
- Nhờ việc sử dụng thử nguyên mẫu, khách hàng có thể có được "cảm nhận thực tế" về hệ thống phần mềm, bởi vì các tương tác với nguyên mẫu cho phép khách hàng hiểu rõ hơn, chính xác hơn về các yêu cầu của hệ thống phần mềm mong muốn

MÔ HÌNH NGUYÊN MẪU

- Sử dụng nguyên mẫu là hợp lý đối với việc phát triển các hệ thống phần mềm lớn và phức tạp (khi không có quy trình thu thập yêu cầu hoặc hệ thống sẵn có nào giúp xác định các yêu cầu phần mềm)
- Một nguyên mẫu thường không phải là một hệ thống phần mềm hoàn chỉnh/hoàn thiện, và rất nhiều các chi tiết không được xây dựng trong nguyên mẫu

MÔ HÌNH NGUYÊN MẪU

- Các ưu điểm
  - Người sử dụng được tham gia tích cực vào trong quá trình PTPM
  - Sử dụng nguyên mẫu là một mô hình hoạt động của hệ thống, những người sửdụng hiểu rõ hơn về hệ thống đang được xây dựng
  - Các lỗi, vấn đề có thể được phát hiện từ (rất) sớm
  - Sớm có được các phản hồi đánh giá từ người sử dụng, giúp có được các giảipháp PTPM tốt hơn
  - Các chức năng còn thiếu có thể được phát hiện sớm
  - Các chức năng không rõ ràng hoặc khó thao tác có thể được phát hiện

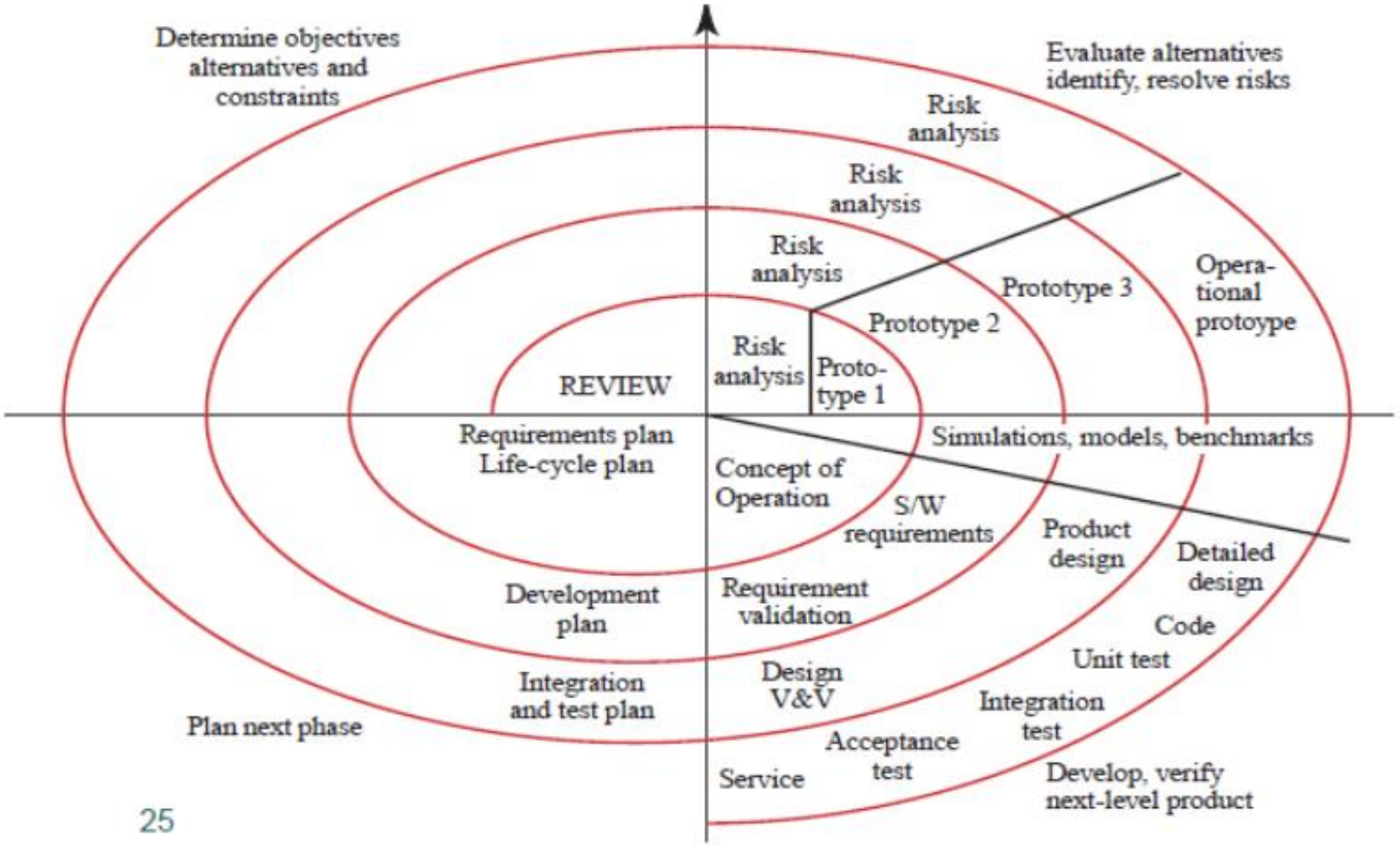
MÔ HÌNH NGUYÊN MẪU

- Các nhược điểm
  - Người sử dụng có thể nghĩ rằng việc phát triển phần mềm là dễ dàng, và vì vậytrở nên không nhất quán trong việc diễn đạt các yêu cầu
  - Không có việc lập kế hoạch ngay từ đầu, có thể dẫn đến các vấn đề về quản lýdự án: không xác định được thời hạn hoàn thành, ngân sách và các kết quả bàn giao
  - Mô hình này thường dẫn đến kéo dài quá trình PTPM
  - Các người phát triển có xu hướng bàn giao một nguyên mẫu hoạt động cơ bản, thay vì bàn giao một sản phẩm hoàn thiện thực sự

MÔ HÌNH NGUYÊN MẪU

- Khi nào nên dùng mô hình nguyên mẫu?
  - Khi các yêu cầu phần mềm không thể được xác định tại thời điểm bắt đầu dự án
  - **Khi những người sử dụng (vì các lý do khác nhau) không thể diễn đạt các yêu cầu của họ một cách rõ ràng**
  - Mô hình PTPM này rất phù hợp để phát triển "cảm nhận" (look and feel) hoặc giao diện sử dụng của hệ thống, bởi vì các đặc điểm này *rất khó để miêu tả bằng tài liệu*, mà thường thu được thông qua việc dùng thử nghiệm
  - Khi khách hàng yêu cầu chứng minh tính khả thi
  - Khi cần có các demos cho các cấp quản lý ở mức cao
  - Khi các vấn đề về công nghệ cần được thử nghiệm, kiểm tra

MÔ HÌNH XOẮN ỐC



25

MÔ HÌNH XOẮN ỐC

- Được đề xuất bởi Barry Boehm
- Là một mô hình phát triển tiến hóa, dựa trên sự kết hợp lai ghép của đặc điểm phát triển lặp (iterative) của Mô hình nguyên mẫu (Prototyping model) và phát triển theo các bước tuần tự (sequential) của Mô hình thác nước (Waterfall model)
  - **Chú trọng vào việc phân tích nguy cơ (risk analysis)**

MÔ HÌNH XOẮN ỐC

- Trong mô hình xoắn ốc, hệ thống phần mềm được phát triển qua một chuỗi các phiên bản tăng cường (incremental releases)
  - Trong các bước phát triển lặp ban đầu, thì các phiên bản của hệ thống phần mềm có thể chỉ là các mô hình được phác thảo trên giấy hoặc là các nguyên mẫu (prototypes)
  - Trong các bước phát triển lặp về sau, thì các phiên bản ngày càng hoàn thiện của hệ thống phần mềm sẽ được tạo ra

MÔ HÌNH XOẮN ỐC

- Các ưu điểm
  - Chú trọng vào phân tích rủi ro (risk analysis), nhờ đó giúp giảm thiểu rủi ro trong dự án PTPM
  - Phù hợp đối với các dự án lớn và quan trọng đặc biệt
  - Các chức năng mới có thể được bổ sung vào sau
  - Các phiên bản đầu của hệ thống phần mềm được tạo ra sớm

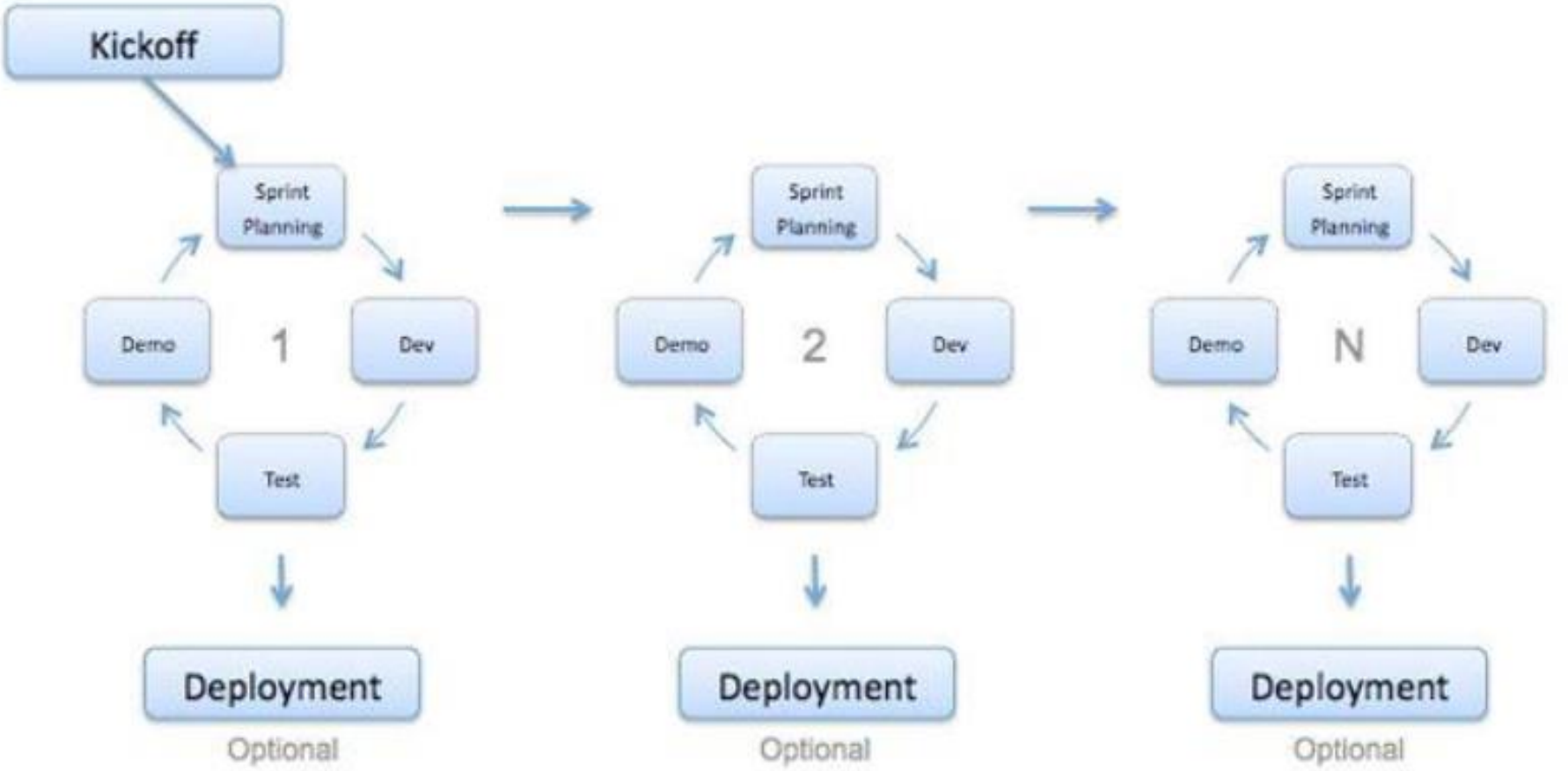
MÔ HÌNH XOẮN ỐC

- Các nhược điểm
  - Chi phí cao (thời gian, nguồn lực, tiền bạc) để áp dụng
  - Việc phân tích rủi ro (risk analysis) đòi hỏi kỹ năng và kinh nghiệm cao
  - Thành công của dự án phụ thuộc rất nhiều vào giai đoạn phân tích rủi ro (risk analysis)
  - Không phù hợp cho các dự án nhỏ

MÔ HÌNH XOẮN ỐC

- Khi nào nên dùng mô hình xoắn ốc?
  - **Khi việc đánh giá (phân tích) các chi phí và các rủi ro là quan trọng**
  - Đối với các dự án có độ rủi ro trung bình đến cao
  - Các người sử dụng không chắc chắn về các nhu cầu của họ
  - Các yêu cầu phần mềm phức tạp và lớn
  - Cần phát triển một dòng sản phẩm mới (New product line)
  - Mong muốn có các thay đổi quan trọng (cần nghiên cứu và khảo sát cẩn thận)

MÔ HÌNH NHANH LẸ - AGILE MODEL



MÔ HÌNH NHANH LẸ - AGILE MODEL

- Là một kiểu mô hình tăng cường (incremental) và lặp lại (iterative)
- Hệ thống phần mềm được phát triển thông qua các vòng phát triển nhanh, tăng cường (incremental and rapid cycles)
- Giúp tạo ra các phiên bản tăng cường ở mức nhỏ, trong đó mỗi phiên bản tiếp theo được xây dựng dựa trên các tính năng của phiên bản liền trước đó
- Mỗi phiên bản tăng cường được kiểm thử cẩn thận để đảm bảo chất lượng phần mềm
- **Được sử dụng cho các dự án PTPM đòi hỏi thời gian hoàn thành nhanh chóng**
  - Lập trình nhanh (Extreme Programming – XP) là một trong những phương pháp phát triển phần mềm nổi tiếng thuộc nhóm mô hình nhanh lẹ

MÔ HÌNH NHANH LẸ - AGILE MODEL

- Các ưu điểm
  - Thỏa mãn khách hàng với các phiên bản phần mềm tăng cường mau lẹ
  - Nhấn mạnh đến sự tương tác giữa các tác nhân hơn là quá trình và các công cụ (Khách hàng, người phát triển, và các người kiểm thử liên tục tương tác trao đổi với nhau
  - Trao đổi thường xuyên giữa nhóm phân tích nghiệp vụ và nhóm lập trình
  - **Thích nghi (đáp ứng) nhanh với các yêu cầu thay đổi**
  - Thậm chí cho phép các thay đổi yêu cầu phần mềm được bổ sung sau

MÔ HÌNH NHANH LẸ - AGILE MODEL

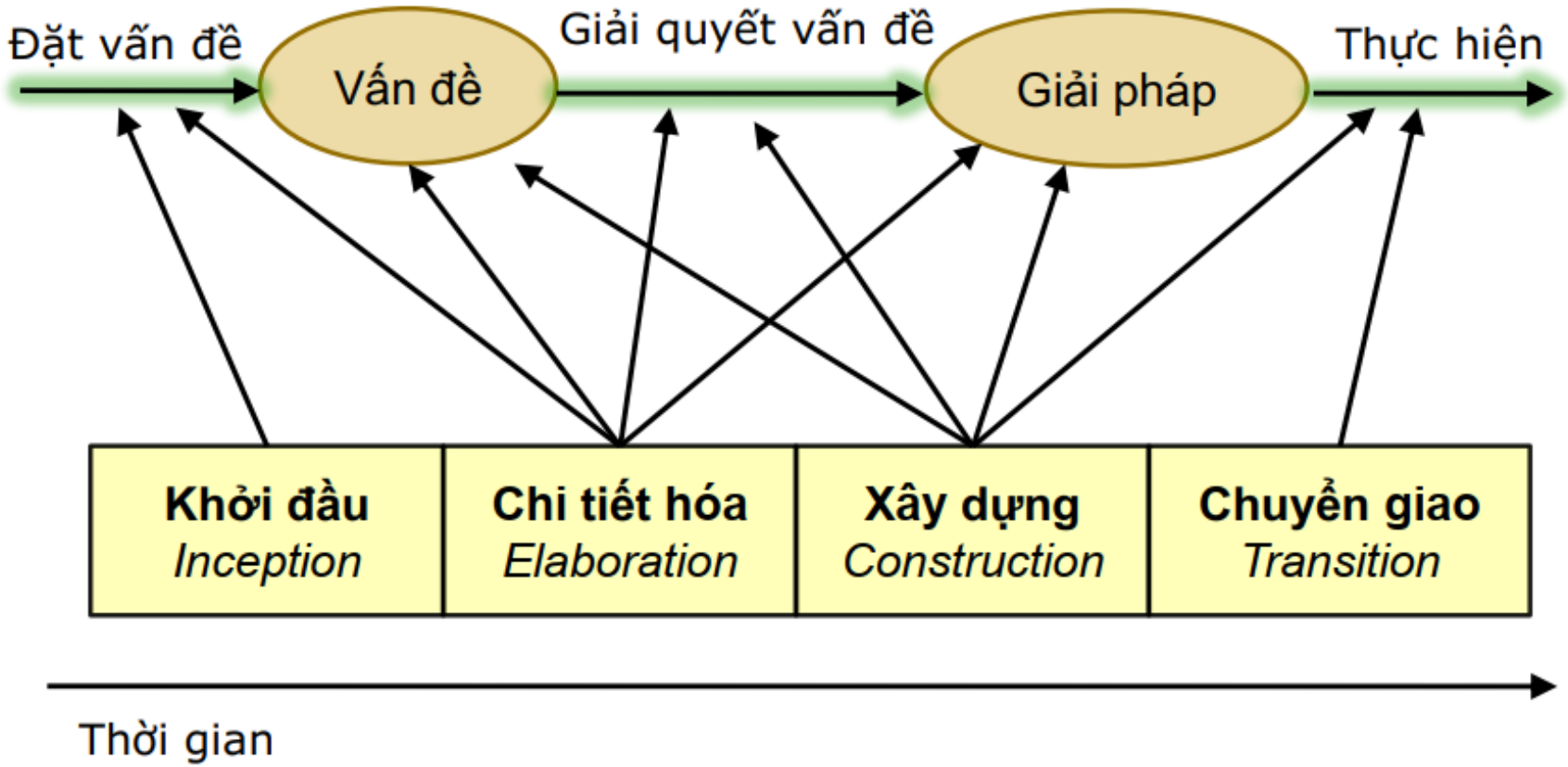
- Các nhược điểm
  - Đối với các hệ thống phần mềm lớn, mô hình này khó đánh giá được các chi phí (effort) cần thiết tại thời điểm bắt đầu tiến trình PTPM
  - Ít chú trọng đến các thiết kế và tài liệu cần thiết
  - Thường chỉ các người lập trình có kinh nghiệm (senior programmers) mới có khả năng đưa ra các quyết định cần thiết trong quá trình phát triển. (Không phù hợp cho các người lập trình ít kinh nghiệm, trừ khi phải làm việc kết hợp với các người có kinh nghiệm)

MÔ HÌNH NHANH LẸ - AGILE MODEL

- Khi nào nên dùng mô hình nhanh lẹ?
  - Khi các thay đổi cần được bổ sung và thực hiện. Các thay đổi mới có thể được thực hiện với chi phí thấp, nhờ việc thường xuyên tạo ra các phiên bản tăng cường
  - Để thực hiện một tính năng mới (a new feature), những người lập trình chỉ cần tốn thời gian vài ngày, hoặc thậm chí là chỉ vài giờ để thực hiệ
  - Không như mô hình thác nước, trong mô hình nhanh lẹ, thì việc lập kế hoạch tốn chi phí thấp hơn (nhiều). **Mô hình nhanh lẹ giả sử rằng các nhu cầu của người dùng sẽ có thể (thường xuyên) thay đổi. Các thay đổi luôn luôn có thể được yêu cầu, và các tính năng luôn luôn có thể được bổ sung hoặc loại bỏ dựa trên các phản hồi**. Điều này giúp đem lại cho khách hàng hệ thống phần mềm mà họ cần và muốn dùng
  - Cả người phát triển và người sử dụng hệ thống đều thấy họ được tự do về thời gian và các lựa chọn hơn là so với các mô hình tuân thủ chặt chẽ theo trình tự các bước (vd: mô hình thác nước)

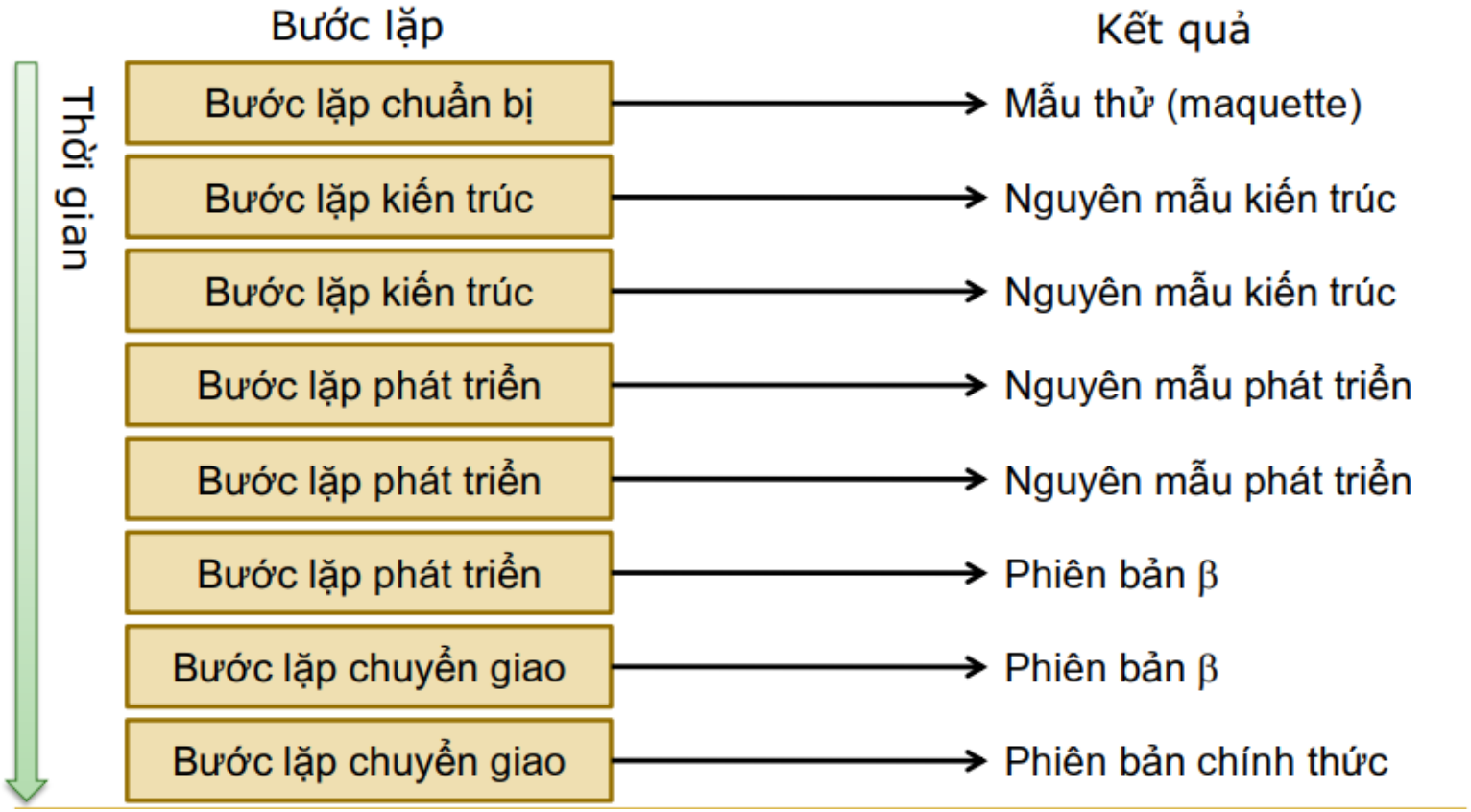
MÔ HÌNH HỢP NHẤT

- Góc nhìn quản lí



MÔ HÌNH HỢP NHẤT

- Góc nhìn kĩ thuật



BÀI TẬP BUỔI 2_2

BT2. Trình bày mô hình Agile và mô hình Scrum trong Agile

1.6.3 QUY TRÌNH RUP

- Các nguyên tắc cơ bản
- Các giai đoạn chính
- Các bước chính

CÁC NGUYÊN TẮC CƠ BẢN

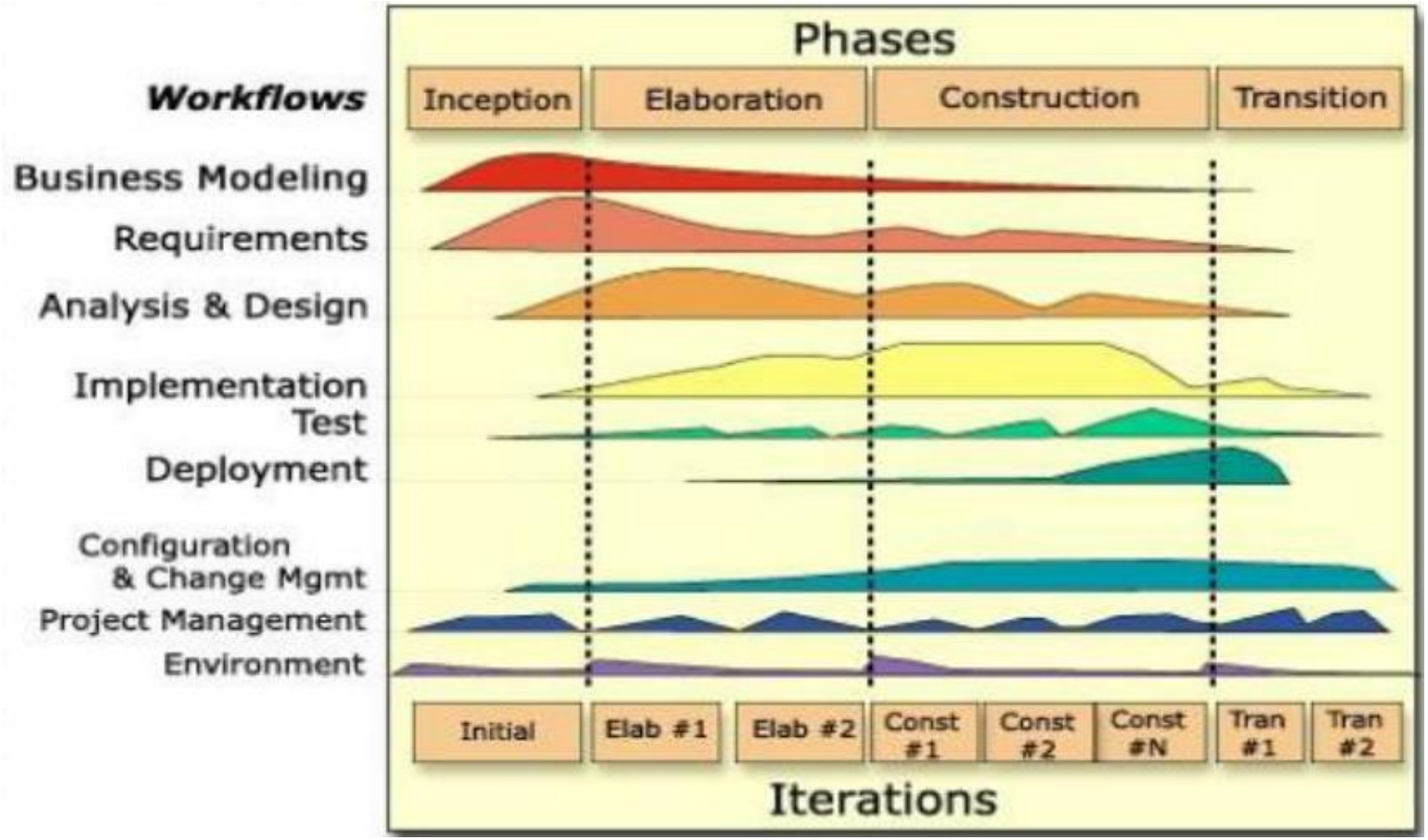
- Lặp và tăng trưởng
  - Dự án được chia thành những vòng lặp hoặc giai đoạn ngắn để dễ dàng kiểm soát
  - Cuối mỗi vòng lặp, phần thi hành được của hệ thống phần mềm được sản sinh theo cách thêm vào dần dần
- Tập trung vào kiến trúc
  - Hệ thống phức tạp được chia thành các mô-đun để dễ dàng triển khai và bảo trì
  - Kiến trúc này được trình bày theo 5 góc nhìn khác nhau

CÁC NGUYÊN TẮC CƠ BẢN

- Dẫn dắt theo các ca sử dụng (use cases)
  - Nhu cầu người dùng thể hiện bởi các ca sử dụng. Các ca sử dụng ảnh hưởng xuyên suốt cho mọi giai đoạn phát triển hệ thống, là cơ sở xác định vòng lặp và tăng trưởng, là căn cứ để phân chia công việc trong nhóm
  - **Nắm bắt nhu cầu**: Phát hiện các ca sử dụng
  - **Phân tích**: Đi sâu vào mô tả các ca sử dụng
  - **Thiết kế và cài đặt**: Xây dựng hệ thống theo các ca sử dụng
  - **Kiểm thử và nghiệm thu hệ thống**: Thực hiện theo các ca sử dụng
- Khống chế các nguy cơ (risks)
  - Phát hiện sớm và loại bỏ các nguy cơ đối với dự án PTPM

CÁC GIAI ĐOẠN CỦA RUP

- RUP được tổ chức thành 4 giai đoạn: Khởi đầu (Inception), Chi tiết hóa (Elaboration), Xây dựng (Construction), và Chuyển giao (Transition)



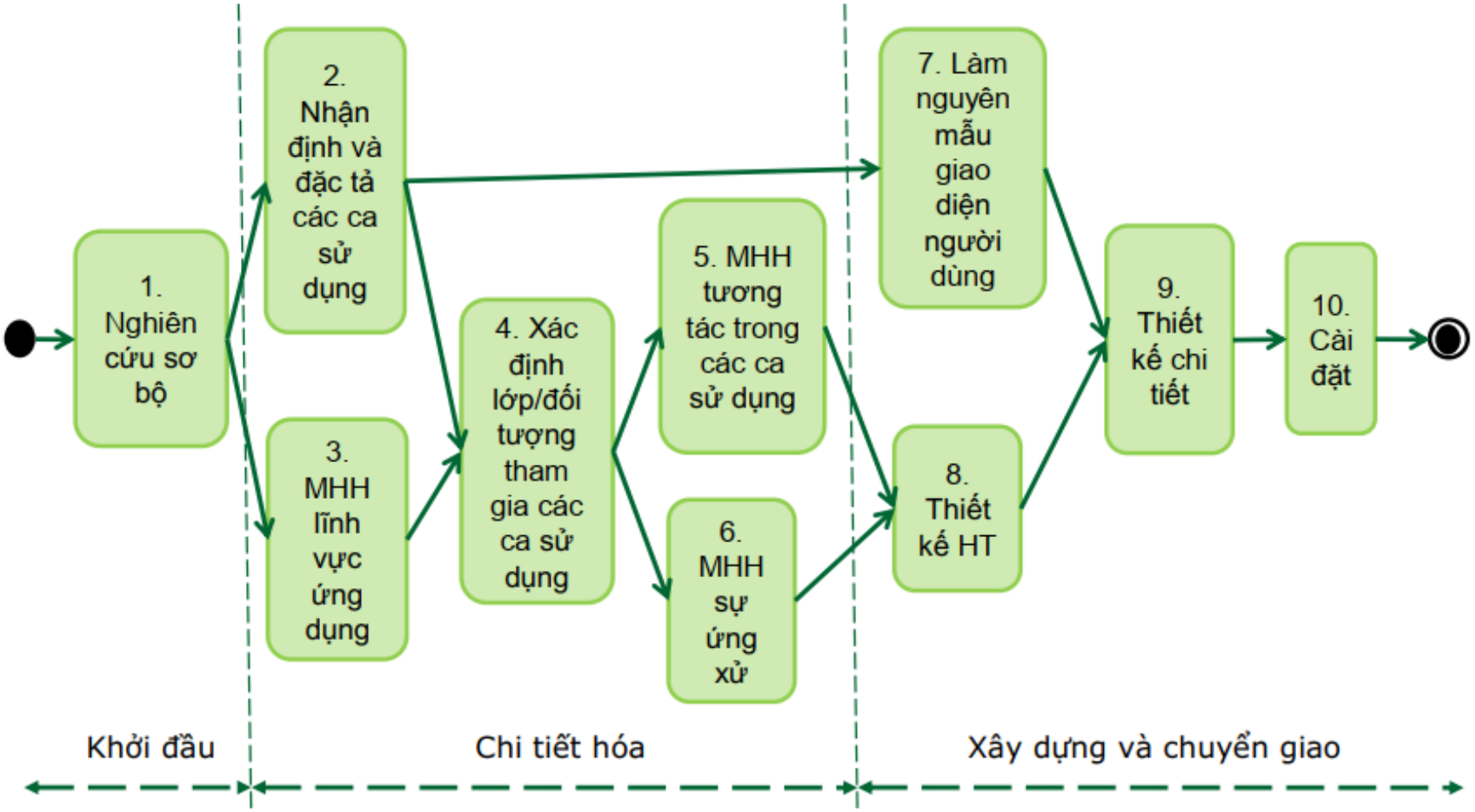
CÁC GIAI ĐOẠN CỦA RUP

- Giai đoạn khởi đầu (Inception)
  - Cho một cái nhìn tổng quát về hệ thống phần mềm (chức năng, hiệu năng, công nghệ,…) và về dự án PTPM sẽ triển khai (phạm vi, mục tiêu, tính khả thi,…) => Đưa ra kết luận nên phát triển dự án hay loại bỏ?
- Giai đoạn chi tiết hóa (Elaboration)
  - Phân tích chi tiết hơn về hệ thống (chức năng và cấu trúc tĩnh)
  - Đề xuất một kiến trúc hệ thống (nguyên mẫu)

CÁC GIAI ĐOẠN CỦA RUP

- Giai đoạn xây dựng (Construction)
  - Tập trung vào thiết kế và cài đặt hệ thống
  - Kiến trúc hệ thống được chi tiết hóa, chỉnh sửa
  - Kết thúc khi đã cho ra được 1 hệ thống hoàn chỉnh với tài liệu kỹ thuật đi kèm
  - Là giai đoạn tốn nhiều thời gian, công sức nhất
- Giai đoạn chuyển giao (Transition)
  - Chuyển giao hệ thống cho người dùng cuối: chuyển đổi dữ liệu, lắp đặt, kiểm tra, đào tạo,…

CÁC BƯỚC CHÍNH CỦA RUP



CÁC BƯỚC CHÍNH CỦA RUP

1. Nghiên cứu sơ bộ

- Đưa ra cái nhìn khái quát về hệ thống phần mềm và dự án PTPM
- Đưa ra kết luận: nên/không nên triển khai dự án?

2. Nhận định và đặc tả các ca sử dụng

- Nắm bắt nhu cầu của người dùng, phát hiện các ca sử dụng
- Mỗi ca sử dụng phải được đặc tả (được mô tả) dưới dạng kịch bản và/hoặc một biểu đồ trình tự

3. MHH lĩnh vực ứng dụng

- Đưa ra mô hình biểu đồ lớp phản ánh mọi khái niệm, nghiệp vụ
- Các lớp ở đây là các lớp lĩnh vực (không phải là các lớp đối tượng)

CÁC BƯỚC CHÍNH CỦA RUP

4. Xác định các đối tượng/lớp tham gia ca sử dụng

- Với mỗi ca sử dụng, phát hiện các lớp lĩnh vực, lớp điều khiển, lớp biên

5. MHH tương tác trong các ca sử dụng

- Các đối tượng tương tác bằng cách trao đổi thông điệp
- Tạo kịch bản của ca sử dụng: biểu đồ trình tự, biểu đồ giao tiếp

6. MHH sự ứng xử

- Các đối tượng điều khiển có khả năng ứng xử đối với các sự kiện đến từ bên ngoài để điều khiển
- Sử dụng biểu đồ máy trạng thái để mô tả hành vi ứng xử của các đối tượng điều khiển

CÁC BƯỚC CHÍNH CỦA RUP

7. Làm nguyên mẫu giao diện người dùng

- Sử dụng các bộ tạo lập giao diện người dùng (graphical user interface – GUI) để làm sớm nguyên mẫu giao diện, giúp cho việc mô hình hóa và cài đặt hệ thống dễ dàng hơn

8. Thiết kế hệ thống

- Thiết kế kiến trúc tổng thể của hệ thống
- Chia thành các hệ thống con, chọn lựa loại hình điều khiển thích hợp
- Dùng biểu đồ thành phần mô tả các thành phần vật lý
- Dùng biểu đồ triển khai mô tả cách bố trí, triển khai các thành phần thực thi của hệ thống vào các phần cứng và nền tảng hạ tầng
- Kiến trúc khách/chủ (client/server) là một kiến trúc hệ thống hay được sử dụng

CÁC BƯỚC CHÍNH CỦA RUP

9. Thiết kế chi tiết

- Thiết kế chi tiết đối với các lớp, các liên kết, các thuộc tính, các phương thức
- Xác định các giải pháp cài đặt HT

10. Cài đặt

- Lập trình và kiểm thử
- Hệ thống được nghiệm thu dựa theo các ca sử dụng

TÓM TẮT CHƯƠNG 1

- Hiểu các khái niệm chung
  - Vòng đời phát triển hệ thống
  - Phương pháp PT HT
  - Công cụ PT HT
  - Đội ngũ PT HT
- Mô hình hóa hướng đối tượng – UML
  - Lịch sử
  - Góc nhìn
  - Các biểu đồ
  - Công cụ
- Quy trình phát triển phần mềm - quy trình RUP
  - Khái niệm
  - Các mô hình phát triển phần mềm
  - Quy trình RUP

THANKS FOR LISTENING