

25 YEARS ANNIVERSARY

SOKT

The graphic consists of a large, bold number "25" in white. A curved banner arches over the top of the "2", containing the text "YEARS ANNIVERSARY" in a smaller, sans-serif font. Below the "25" is the acronym "SOKT" in a large, bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

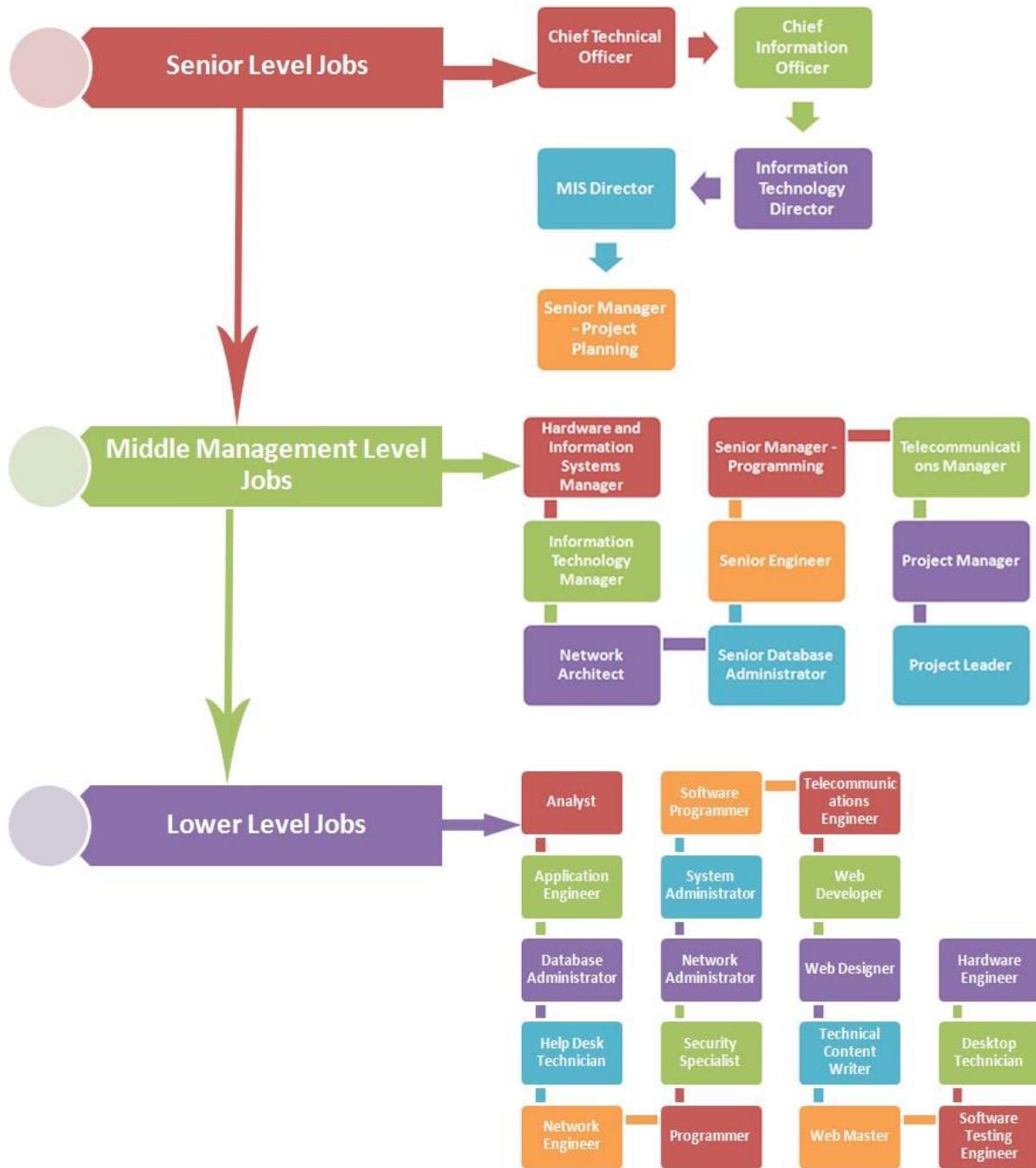
Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

Nội dung

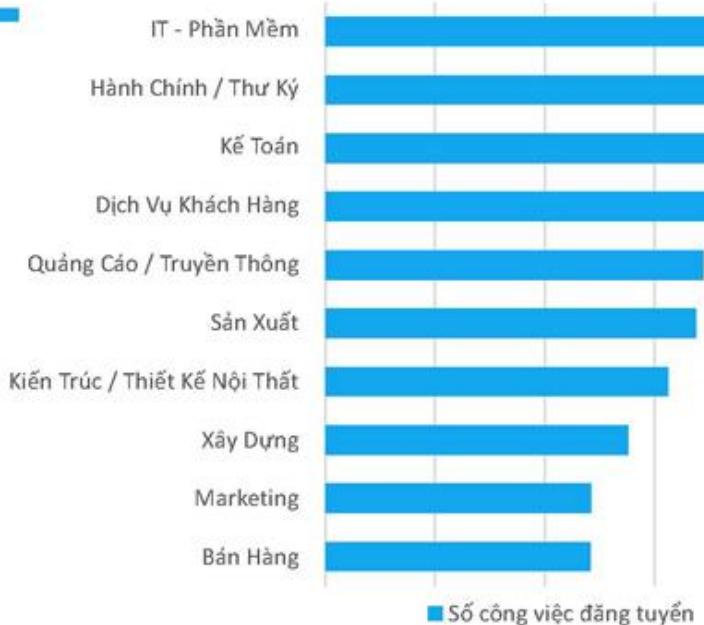
- Tại sao học môn này ?
- Mô tả môn học
 - Mục tiêu
 - Nội dung của môn học
 - Tài liệu học tập
 - Kế hoạch học tập
 - Danh sách bài tập/labs
 - Yêu cầu, đánh giá

Các vị trí công việc CNTT



Nhu cầu tuyển dụng 2019 – Việt Nam

TOP 10 NGÀNH CÓ NHU CẦU TUYỂN DỤNG CAO NHẤT



Nhóm công việc Phát triển phần mềm (Software Development) vẫn được các công ty săn tìm nhiều nhất, chiếm 57.4% số công việc ngành CNTT được đăng tuyển trên VietnamWorks.

* Biểu đồ thể hiện trên dữ liệu gốc nên không công bố con số chính xác nhằm đảm bảo tính bảo mật.

Nhu cầu tuyển dụng 2019 – Mỹ

Top 10 Best Jobs in America in 2019

Rank	Job Title	Median Base Salary	Job Satisfaction	Job Openings
1	Data Scientist	\$108,000	4.3	6,510
2	Nursing Manager	\$83,000	4.0	13,931
3	Marketing Manager	\$82,000	4.2	7,395
4	Occupational Therapist	\$74,000	4.0	17,701
5	Product Manager	\$115,000	3.8	11,884
6	Devops Engineer	\$106,000	4.1	4,657
7	Program Manager	\$87,000	3.9	14,753
8	Data Engineer	\$100,000	3.9	4,739
9	HR Manager	\$85,000	4.2	3,908
10	Software Engineer	\$104,000	3.6	49,007

SO ARE YOU A NEWBIE, PROGRAMMER, DEVELOPER, OR SOFTWARE ENGINEER?



NEWBIE

IN THE LEARNING STAGES, ABLE TO WRITE SIMPLE HELLO WORLD APPS



PROGRAMMER

ABLE TO WRITE SIMPLE ALGORITHMS TO SOLVE SIMPLE PROBLEMS, MORE THEORETICAL THAN PRACTICAL



DEVELOPER

ABLE TO CREATE APPLICATIONS THAT ARE USABLE, OFTEN MAKES A LIVING SELLING WRITTEN APPLICATIONS



SOFTWARE ENGINEER

ARCHITECT SOLUTIONS, BUILD SYSTEMS AS WELL AS WRITE CODE TO BUILD SCALABLE APPLICATIONS

Mô tả môn học

- **Kiến thức:**
 - Các hoạt động chính trong quy trình phát triển phần mềm
 - Vòng đời phần mềm, quy trình phát triển phần mềm
 - Các mô hình phần mềm
 - Quản lý dự án phần mềm, quản lý cấu hình – phiên bản
 - Phân tích thiết kế, xây dựng và đảm bảo chất lượng phần mềm.
- **Kỹ năng:**
 - Kỹ năng phát triển một phần mềm theo quy trình trong thực tiễn
 - Kỹ năng làm việc nhóm
 - Kỹ năng thuyết trình
- **Thái độ:**
 - Độc lập, chủ động, kiên trì và linh hoạt trong công việc.
 - Thể hiện tính trung thực, có trách nhiệm và tin cậy trong công việc.

Lộ trình học tập

Các khái niệm cơ bản về phần mềm và quy trình phát triển phần mềm

1

Tổng quan về Công nghệ phần mềm

2

Vòng đời phần mềm

3

Phương pháp Agile

Quản lý dự án phần mềm

4

Quản lý dự án phần mềm

5

Quản lý cấu hình phần mềm



Các pha chính phát triển phần mềm



Final Test

9

Đảm bảo chất lượng phần mềm + Bảo trì phần mềm

8

Xây dựng phần mềm

7

Thiết kế phần mềm

6

Kỹ nghệ yêu cầu phần mềm



Xem chi tiết tại đây

Tài liệu học tập

- Giáo trình:
 - R. Pressman, Software Engineering: A practitioner's approach, 8th Edition, McGraw Hill 2016
- Bài giảng:
 - Roadmap
 - Slides
 - Lab guides
 - Videos
- Tài liệu tham khảo:
 1. I. Sommerville, Software Engineering 10th Edition, Addison Wesley 2017

Yêu cầu với sinh viên

- Dự lớp đầy đủ
- Làm bài tập cá nhân, bài tập nhóm đầy đủ
- Đọc tài liệu trước khi đến lớp
- Tích cực tham gia thảo luận, phát biểu ý kiến trên lớp

Đánh giá

Điểm	Phương pháp đánh giá	Tỷ trọng
Quá trình	Thảo luận	10%
	Trắc nghiệm (LMS)	20%
	Bài tập nhóm	20%
Cuối kỳ	Thi tự luận / trắc nghiệm	60%

Q&A



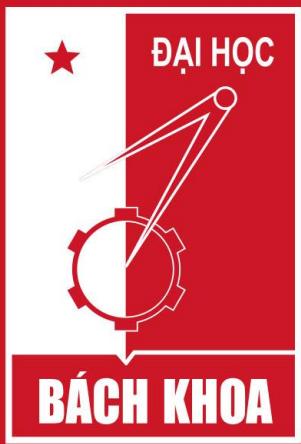


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you
for your
attentions!





25 YEARS ANNIVERSARY

SOKT

The graphic consists of the number "25" in large, bold, white font. A curved banner arches over the top of the "2" containing the text "YEARS ANNIVERSARY". Below the "25" is the acronym "SOKT" in a large, bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 1

Tổng quan về Công nghệ phần mềm

Nội dung

1. Phần mềm là gì?
2. Phân loại phần mềm
3. Công nghệ phần mềm là gì?
4. Các vấn đề trong công nghệ phần mềm

Mục tiêu của bài học

- Hiểu được các khái niệm cơ bản như: phần mềm và công nghệ phần mềm
- Biết được các pha trong phát triển phần mềm
- Biết được những khó khăn, thách thức gặp phải trong quá trình phát triển phần mềm

1. Phần mềm là gì?

- **Định nghĩa (*):**
 - **Các lệnh** (chương trình máy tính) khi được thực hiện thì cung cấp những chức năng và kết quả mong muốn
 - **Các cấu trúc dữ liệu** làm cho chương trình thao tác thông tin thích hợp
 - **Các tài liệu mô tả** thao tác và cách sử dụng chương trình

(*) Roger Pressman (McGraw-Hill 2014). *Software Engineering: A Practitioner's Approach, 8/e*

1. Phần mềm là gì?

- Các đặc trưng của phần mềm:
 - Là hàng hóa **vô hình**, không nhìn thấy được
 - Chất lượng phần mềm: không mòn đi mà có **xu hướng tốt lên** sau mỗi lần có lỗi (error) được phát hiện và sửa
 - Phần mềm vốn **chứa lỗi tiềm tàng**, theo quy mô càng lớn thì khả năng chứa lỗi càng cao
 - **Lỗi phần mềm** dễ được phát hiện bởi người ngoài
 - Chức năng của phần mềm **thường biến hóa**, thay đổi theo thời gian (theo nơi sử dụng)

2. Phân loại phần mềm

- Phần mềm **hệ thống** (System SW)
- Phần mềm **thời gian thực** (Real-time SW)
- Phần mềm **nghiệp vụ** (Business SW)
- Phần mềm **KH&KT** (Engineering & Science SW)
- Phần mềm **nhúng** (Embedded SW)
- Phần mềm máy **cá nhân** (Personal computer SW)
- Phần mềm trên **Web** (Web-based SW)
- Phần mềm **trí tuệ nhân tạo** (Artificial Intelligent SW)

Câu hỏi

- Phân biệt các khái niệm sau:
 1. Hệ thống, phần mềm, ứng dụng
 2. Lập trình, phát triển phần mềm
 3. Lập trình viên và kỹ sư phần mềm

Nội dung

1. Phần mềm là gì?
2. Phân loại phần mềm
3. Công nghệ phần mềm là gì?
4. Các vấn đề trong công nghệ phần mềm

2. Công nghệ phần mềm (Software Engineering)

Định nghĩa

- Bauer [1969]: CNPM là việc **thiết lập và sử dụng** các **nguyên tắc công nghệ** học đúng đắn dùng để thu được phần mềm một cách **kinh tế** vừa **tin cậy** vừa làm việc **hiệu quả** trên các máy thực
- Parnas [1987]: CNPM là việc xây dựng phần mềm nhiều phiên bản bởi nhiều người
- Ghezzi [1991]: CNPM là một lĩnh vực của khoa học máy tính, liên quan đến xây dựng các hệ thống phần mềm vừa lớn vừa phức tạp bởi một hay một số nhóm kỹ sư

Định nghĩa

- IEEE [1993]: CNPM là
 - (1) việc áp dụng phương pháp tiếp cận có hệ thống, bài bản và được lượng hóa trong phát triển, vận hành và bảo trì phần mềm;
 - (2) nghiên cứu các phương pháp tiếp cận được dùng trong (1)
- Pressman [1995]: CNPM là bộ môn tích hợp cả quy trình, các phương pháp, các công cụ để phát triển phần mềm máy tính

Định nghĩa

- Sommerville [1995]: CNPM là lĩnh vực liên quan đến lý thuyết, phương pháp và công cụ dùng cho phát triển phần mềm
- K. Kawamura [1995]: CNPM là lĩnh vực học vấn về các kỹ thuật, phương pháp luận công nghệ học (lý luận và kỹ thuật được hiện thực hóa trên những nguyên tắc, nguyên lý nào đó) trong toàn bộ quy trình phát triển phần mềm nhằm nâng cao cả chất và lượng của sản xuất phần mềm

Định nghĩa

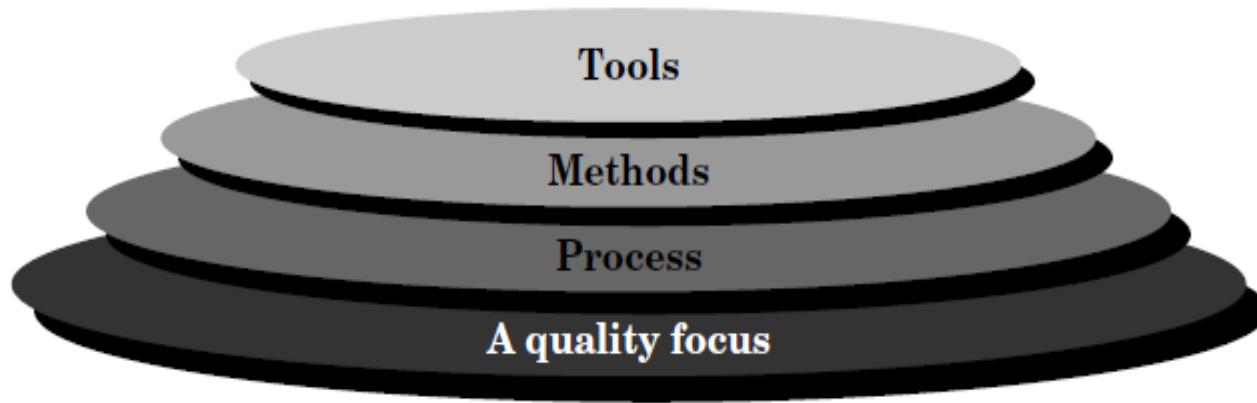
- Công nghệ phần mềm là lĩnh vực **khoa học về** các **phương pháp luận, kỹ thuật** và **công cụ** tích hợp trong quy trình sản xuất và vận hành phần mềm nhằm **tạo ra phần mềm** với những chất lượng mong muốn

[Software Engineering is a scientific field to deal with methodologies, techniques and tools integrated in software production-maintenance process to obtain software with desired qualities]

Các mục tiêu chính

- Tăng **năng suất** và **chất lượng** phần mềm
- Quản lý **lập lịch** hiệu quả
- Giảm **chi phí** phát triển phần mềm
- Đáp ứng **yêu cầu** và **nhu cầu** của khách hàng
- Tăng cường **quy trình** kỹ nghệ phần mềm
- Tăng cường **thực hành** kỹ thuật phần mềm
- **Hỗ trợ** hiệu quả và có hệ thống các hoạt động của kĩ sư phát triển

CNPM là công nghệ phân lớp



- Quy trình (Process)
- Các phương pháp (Methods)
- Các công cụ (Tools)

(*) Roger Pressman (McGraw-Hill 2014). *Software Engineering: A Practitioner's Approach, 8/e*

Quy trình - Process

- **Gắn kết các lớp** với nhau
- **Nền tảng** cho kỹ thuật phần mềm
- **Đảm bảo thời gian** phát triển
- **Tạo cơ sở cho** việc kiểm soát, quản lý dự án phần mềm
- Thiết lập bối cảnh mà các phương pháp kỹ thuật được sử dụng
- Tạo sản phẩm
- Thiết lập các cột mốc
- Đảm bảo chất lượng
- Quản lý thay đổi

Các phương pháp - Methods

- Cung cấp **kỹ thuật** cho xây dựng phần mềm
- Các tác vụ: giao tiếp, phân tích yêu cầu, mô hình thiết kế, xây dựng chương trình, kiểm thử và hỗ trợ.
- Dựa trên các nguyên tắc cơ bản
 - Để chi phối từng lĩnh vực công nghệ
 - Bao gồm các hoạt động mô hình hóa

Công cụ - Tools

- Tự động hoặc bán tự động hỗ trợ cho quy trình và các phương pháp
- Hướng đến chất lượng - A quality focus
 - Nền tảng
 - Bất kỳ cách tiếp cận kỹ thuật nào đều phải dựa trên cam kết về chất lượng
 - Thúc đẩy liên tục việc cải tiến quy trình

SE các pha

- Được phân thành ba giai đoạn chung
 - Pha **định nghĩa** (Definition phase)
 - Pha **phát triển** (Development phase)
 - Pha **hỗ trợ** (Support phase)

Pha định nghĩa

- Xác định **cái gì** “WHAT”.
 - Thông tin nào được xử lý,
 - Chức năng và hiệu quả mong muốn,
 - Hành vi mong đợi của hệ thống,
 - Các giao diện cần thiết lập,
 - Những ràng buộc về thiết kế,
 - Và những tiêu chí cần thẩm định.
- Các **yêu cầu chính** của hệ thống và phần mềm được xác định.

Pha phát triển

- Xác định **như thế nào** “HOW”.
 - Cách thức dữ liệu được cấu trúc,
 - Chức năng được triển khai trong kiến trúc phần mềm,
 - Các chi tiết thủ tục được cài đặt,
 - Cách xác định các đặc điểm của giao diện,
 - Cách chuyển từ thiết kế sang lập trình,
 - Và cách thức kiểm thử.

Pha hỗ trợ

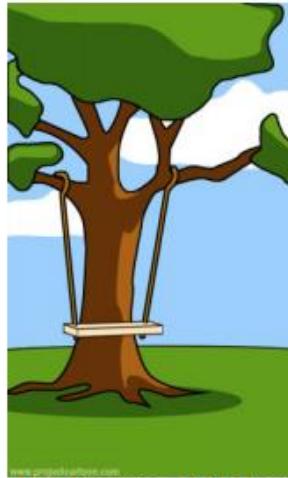
- Liên kết với **các thay đổi “CHANGE”**
 - Sửa lỗi,
 - Thích nghi với yêu cầu của môi trường,
 - Và các thay đổi bởi yêu cầu của khách hàng.
- 4 loại thay đổi: Sửa chữa, Thích ứng, Nâng cao, và Phòng ngừa (Correction, Adaptation, Enhancement, and Prevention).

Nội dung

1. Phần mềm là gì?
2. Phân loại phần mềm
3. Công nghệ phần mềm là gì?
4. Các vấn đề trong công nghệ phần mềm



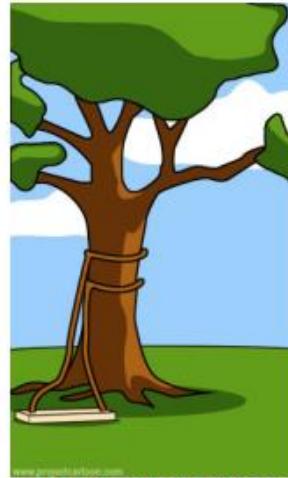
How the customer explained it



How the project leader understood it



How the analyst designed it



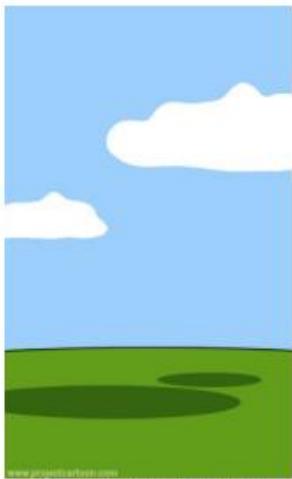
How the programmer wrote it



What the beta testers received



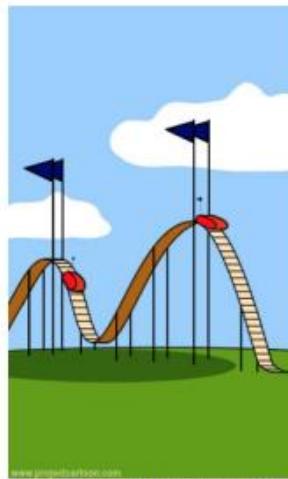
How the business consultant described it



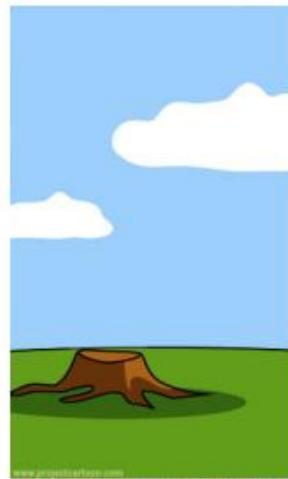
How the project was documented



What operations installed



How the customer was billed



How it was supported



iSwing



What the customer really needed

4. Các vấn đề

- Nhu cầu kinh doanh không được đáp ứng
- Yêu cầu không được giải quyết
- Các module không tích hợp
- Khó khăn khi bảo trì
- Phát hiện muộn về sai sót
- Chất lượng trải nghiệm kém
- Hiệu suất kém
- Không có nỗ lực phối hợp của nhóm
- Các vấn đề về xây dựng và phát hành

4. Các vấn đề

- Không có phương pháp mô tả rõ ràng yêu cầu của khách hàng
→ Sau khi bàn giao sản phẩm dễ phát sinh những trực trặc
- Với những phần mềm quy mô lớn, **tư liệu đặc tả** cố định
→ Khó đáp ứng nhu cầu thay đổi của người dùng
- Phương pháp luận thiết kế không nhất quán
→ Thiết kế theo cách riêng dẫn đến giảm chất lượng phần mềm
- Không có chuẩn về việc tạo **tư liệu** quy trình sản xuất phần mềm
→ Đặc tả không rõ ràng sẽ làm giảm chất lượng phần mềm

4. Các vấn đề

- Không có phương pháp mô tả rõ ràng yêu cầu của khách hàng
→ Sau khi bàn giao sản phẩm dễ phát sinh những trực trặc (troubles)
- Với những phần mềm quy mô lớn, **tư liệu đặc tả** cố định
→ Khó đáp ứng nhu cầu thay đổi của người dùng
- Phương pháp luận thiết kế không nhất quán
→ Thiết kế theo cách riêng dẫn đến giảm chất lượng phần mềm
- Không có chuẩn về việc tạo **tư liệu** quy trình sản xuất phần mềm
→ Đặc tả không rõ ràng sẽ làm giảm chất lượng phần mềm

4. Các vấn đề

- Không kiểm thử tính đúng đắn của phần mềm ở từng giai đoạn mà chỉ kiểm ở giai đoạn cuối và phát hiện ra lỗi
→ thường bàn giao sản phẩm không đúng hạn
- Coi trọng việc lập trình hơn khâu thiết kế
→ giảm chất lượng phần mềm
- Coi thường việc tái sử dụng phần mềm (software reuse)
→ giảm năng suất lao động
- Phần lớn các thao tác trong quy trình phát triển phần mềm do con người thực hiện
→ giảm năng suất lao động

4. Các vấn đề

- Không chứng minh được **tính đúng đắn** của phần mềm
→ *giảm độ tin cậy của phần mềm*
- Chuẩn về một phần mềm tốt không thể **đo được** một cách **định lượng**
→ *Không thể đánh giá được một hệ thống đúng đắn hay không*
- Đầu tư nhân lực lớn vào bảo trì
→ *giảm hiệu suất lao động của nhân viên*

4. Các vấn đề

- Công việc **bảo trì** kéo dài
 - giảm chất lượng của tư liệu và ảnh hưởng xấu đến những việc khác
- Quản lý dự án lỏng lẻo
 - quản lý lịch trình sản xuất phần mềm không rõ ràng
- Không có **tiêu chuẩn** để ước lượng nhân lực và dự toán
 - làm kéo dài thời hạn và vượt kinh phí của dự án

Tổng kết

- Phần mềm gồm: **chương trình, dữ liệu, tài liệu**
- Công nghệ phần mềm:
 - khoa học về các **phương pháp luận, kỹ thuật và công cụ** trong quy trình sản xuất và vận hành phần mềm nhằm **tạo ra phần mềm** với những chất lượng mong muốn
 - Các pha: **định nghĩa, phát triển, hỗ trợ**
- Một số khó khăn:
 - Không đáp ứng được nhu cầu
 - Khó khăn khi bảo trì
 - Phát hiện muộn về sai sót

Q&A



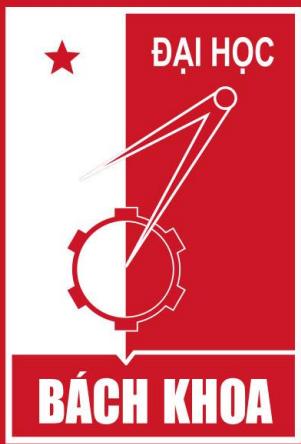


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The graphic consists of the number "25" in large, bold, white font. A curved banner arches over the top of the "2" containing the text "YEARS ANNIVERSARY". Below the "25" is the acronym "SOKT" in a large, bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 2

Vòng đời phần mềm

Nội dung

1. Hệ thống/phần mềm
2. Vòng đời hệ thống/phần mềm
3. Quy trình phát triển phần mềm
4. Các mô hình quy trình phần mềm

Mục tiêu của bài học

- Hiểu được thế nào là vòng đời phần mềm
- Biết được quy trình phát triển phần mềm
- Các mô hình phát triển phần mềm phổ biến

Nội dung

1. Hệ thống / phần mềm
2. Vòng đời hệ thống/phần mềm
3. Quy trình phát triển phần mềm
4. Các mô hình quy trình phần mềm

Hệ thống

- Một hệ thống, ví dụ hệ thống kinh doanh, bao gồm:
 - phần cứng, hệ thống mạng, phần mềm*, tài liệu
 - phần mềm* (software): bao gồm phần mềm nền tảng (như hệ điều hành), phần mềm trung gian (middle software), và ứng dụng doanh nghiệp (Business Application Software)

Nội dung

1. Hệ thống và phần mềm
2. Vòng đời hệ thống/phần mềm
3. Quy trình phát triển phần mềm
4. Các mô hình quy trình phần mềm

2. Vòng đời phần mềm

- Vòng đời phần mềm là thời kỳ tính từ khi phần mềm được **sinh (tạo) ra** cho đến khi **chết đi** (từ lúc hình thành đáp ứng yêu cầu, vận hành, bảo dưỡng cho đến khi loại bỏ không đâu dùng)
- Quy trình phần mềm (vòng đời phần mềm) được phân chia thành **các pha chính: phân tích, thiết kế, chế tạo, kiểm thử, bảo trì**. Biểu diễn các pha có thể khác nhau theo từng mô hình

Vòng đời phần mềm

- Mọi sản phẩm phần mềm đều có vòng đời.
- Vòng đời thường khá dài — một số sản phẩm phần mềm đã “tồn tại” được 30 năm.
- Vòng đời có thể được rút ngắn do tiến bộ công nghệ

Các pha trong vòng đời PM

- Một cách rõ ràng hoặc rõ ràng, tất cả các sản phẩm phần mềm đều trải qua ít nhất các giai đoạn sau:
 - **Yêu cầu** — xác định nhu cầu của khách hàng và các ràng buộc của sản phẩm
 - **Thiết kế** — xác định cấu trúc/tổ chức của hệ thống phần mềm
 - **Mã hóa** — viết phần mềm
 - **Kiểm thử** — vận hành hệ thống để tìm và loại bỏ các khiếm khuyết
 - **Bảo trì** — sửa chữa và nâng cao sản phẩm sau khi khách hàng triển khai

Các mô hình vòng đời phần mềm

- Quá trình là **một tập hợp các hoạt động**, với các đầu vào và đầu ra được xác định rõ ràng, để hoàn thành một số nhiệm vụ.
- Mô hình vòng đời là một mô tả về **một quá trình thực hiện** một sản phẩm phần mềm trong toàn bộ hoặc một phần vòng đời của nó.
 - Các mô hình vòng đời có xu hướng **tập trung vào các pha chính** của chu kỳ và mối quan hệ của chúng với các pha khác.
 - Các nghiên cứu gần đây về quy trình phần mềm đã xem xét chi tiết nhiều **khía cạnh** của việc phát triển và bảo trì.
 - Mô hình vòng đời là một **mô tả quy trình phần mềm**

Nội dung

1. Hệ thống và phần mềm
2. Vòng đời hệ thống/phần mềm
3. Quy trình phát triển phần mềm
4. Các mô hình quy trình phần mềm

3. Quy trình phát triển phần mềm

Khung quy trình chung (Common process framework)

Hoạt động khung (Framework activities)

Tập tác vụ (Task sets)

Tác vụ (Tasks)

Điểm quan trọng
(milestones), sản phẩm chuyển giao (deliverables)

Điểm Kiểm Tra Chất Lượng
(SQA points)

Các hoạt động giám sát, đánh giá kỹ thuật, đảm bảo chất lượng phần mềm, quản lý cấu hình, quản lý rủi ro, ...
(Umbrella activities)

Nội dung

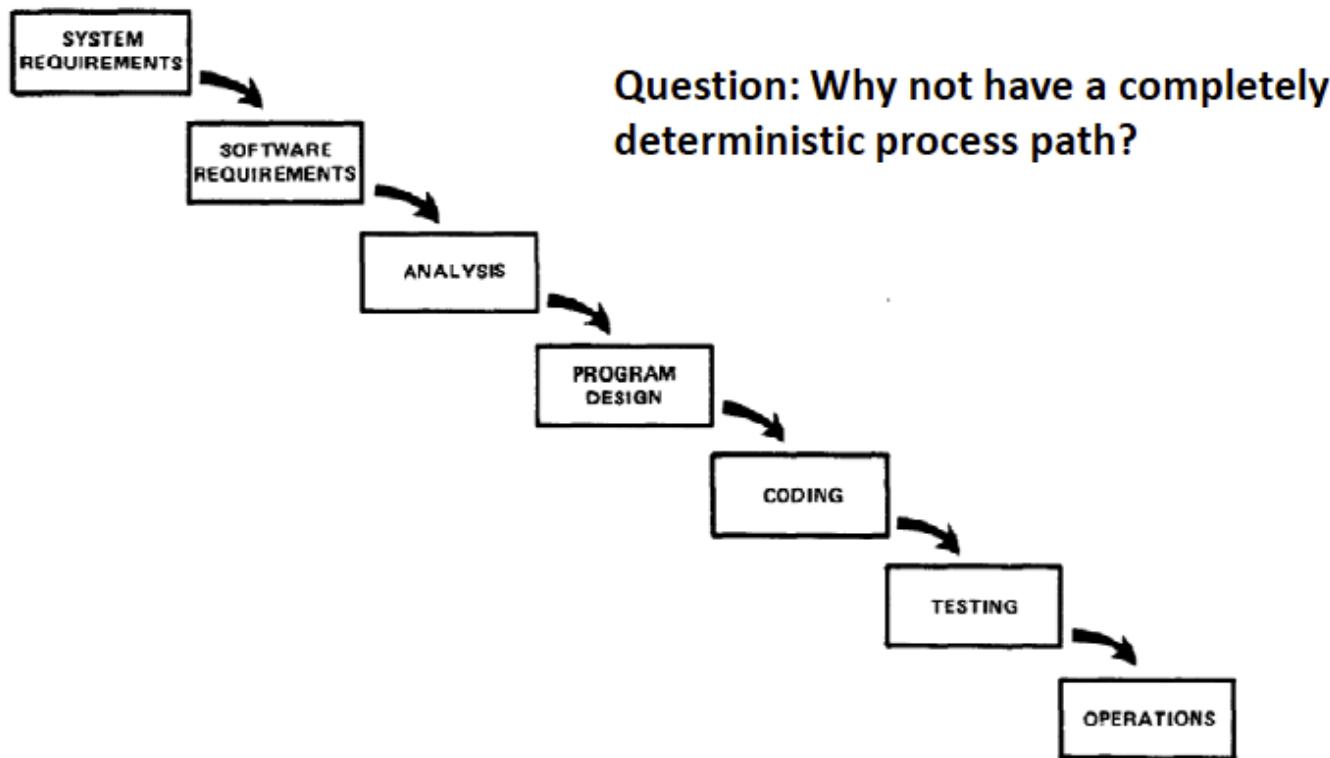
1. Hệ thống và phần mềm
2. Vòng đời hệ thống/phần mềm
3. Quy trình phát triển phần mềm
4. Các mô hình quy trình phần mềm

4.1. Mô hình thác nước

- Mô hình thác nước là mô hình vòng đời lâu đời nhất; được đề xuất bởi Winston Royce vào năm 1970.
- Mô hình này được gọi là thác nước vì nó thường được vẽ với một **chuỗi các hoạt động qua các giai đoạn** của vòng đời “xuống dốc” từ trái sang phải:
 - phân tích, yêu cầu, đặc tả, thiết kế, cài đặt, kiểm thử, bảo trì
- Có nhiều phiên bản của mô hình thác nước:
 - các giai đoạn / hoạt động có thể được cấu trúc theo các mức độ chi tiết khác nhau
 - phản hồi có thể linh hoạt hơn hoặc ít hơn

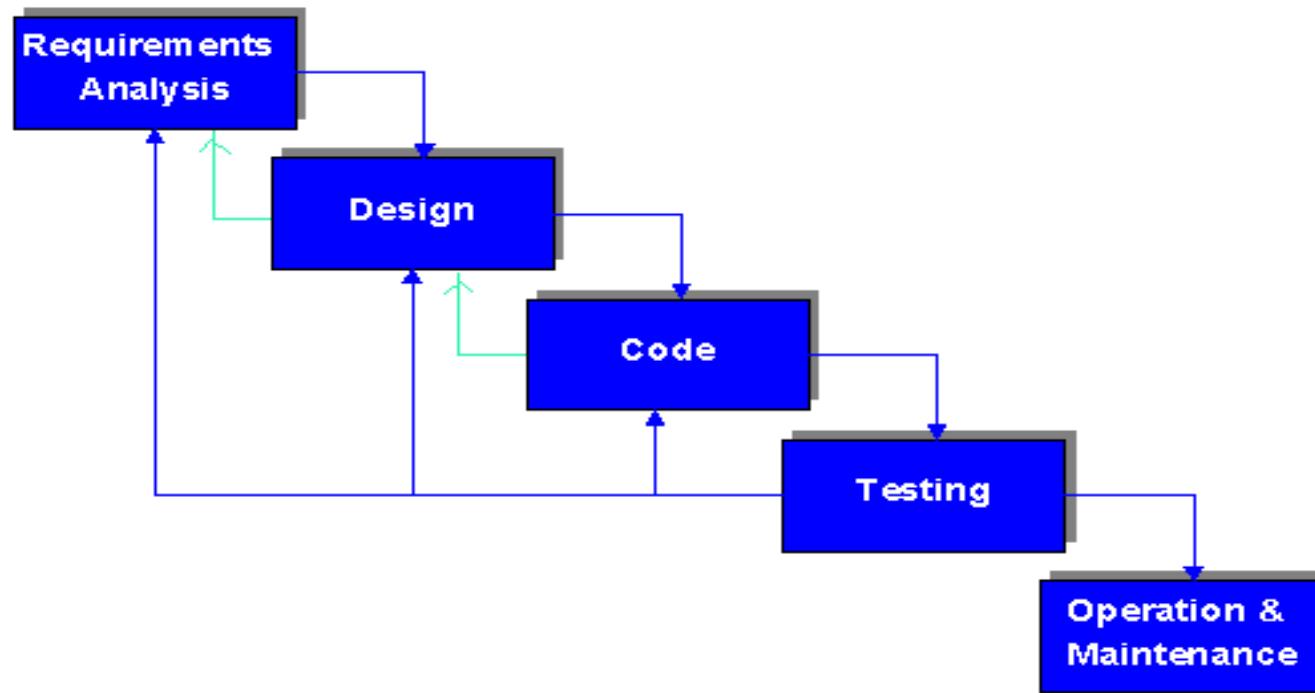
4.1. Vòng đời lý tưởng - Thác nước (Nghiêm ngặt) không có phản hồi

Life Cycle Ideal - (Strict) Waterfall With No Feedback



4.1. Mô hình thác nước (Non-stric)

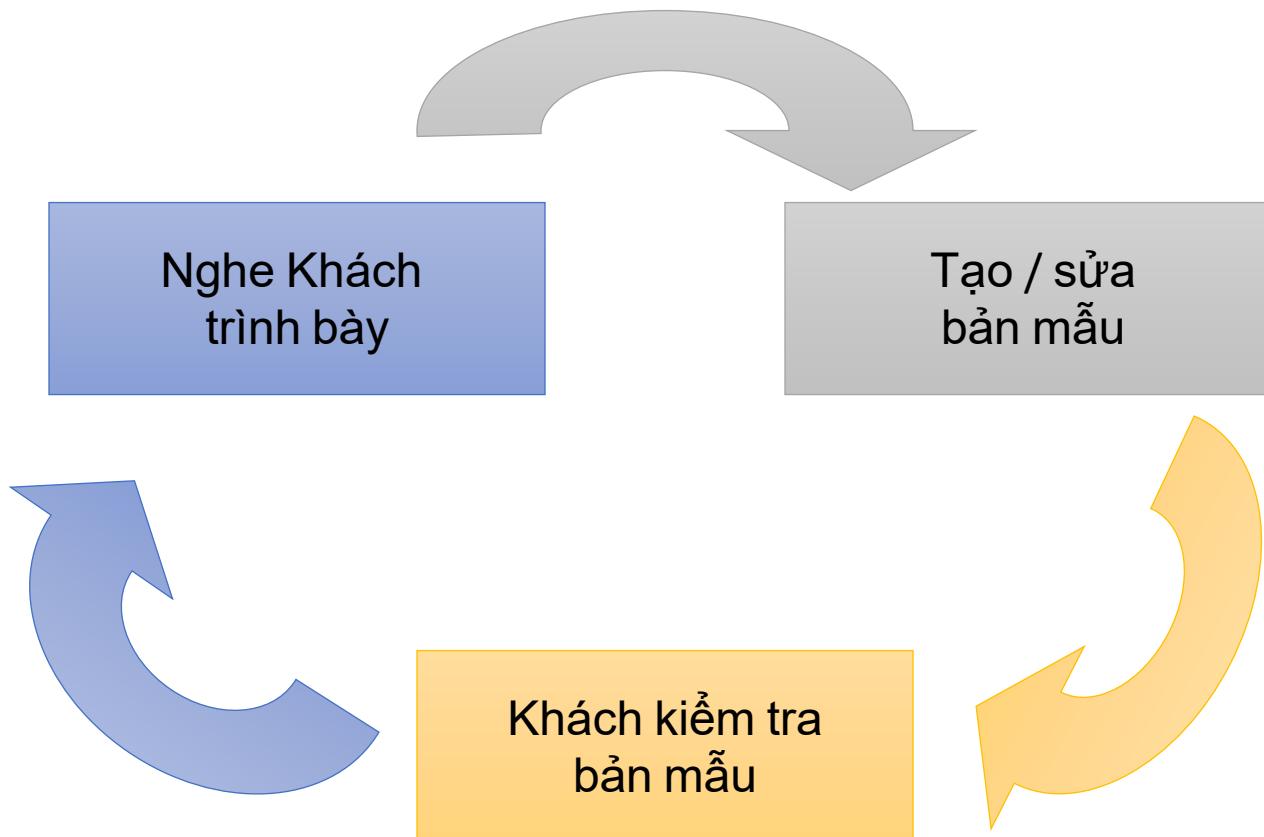
- Mặc dù mô hình thác nước nhấn mạnh một **chuỗi tuyến tính** của các pha, trên thực tế, trong thực tế luôn có một lượng lớn **sự lặp lại các pha** trước đó



4.1. Mô hình thác nước

- **Điểm mạnh:**
 - Hoàn thành một **giai đoạn** trước khi tiếp tục giai đoạn tiếp
 - Nhấn mạnh việc lập kế hoạch sớm, đầu vào của khách hàng và thiết kế
 - Nhấn mạnh kiểm tra như một phần không thể thiếu của vòng đời
 - Cung cấp các chất lượng ở mỗi giai đoạn vòng đời
- **Điểm yếu:**
 - Phụ thuộc vào các **yêu cầu** được xác định sớm từ đầu
 - Phụ thuộc vào việc **tách các yêu cầu** khỏi thiết kế
 - Không khả thi trong một số trường hợp đòi hỏi có nhiều **thay đổi**
 - Nhấn mạnh vào **sản phẩm hơn là quy trình**

4.2. Mô hình mẪu thử (Prototyping model)



4.2. Mô hình mẫu thử: Khi nào ?

- Khi mới rõ mục đích **chung chung** của phần mềm, chưa rõ chi tiết đầu vào hay xử lý ra sao hoặc chưa rõ yêu cầu đầu ra
- Dùng để thu thập yêu cầu qua các thiết kế nhanh
- Các giải thuật, kỹ thuật dùng làm bản mẫu có thể chưa nhanh, chưa tốt, miễn là **có mẫu để thảo luận gợi yêu cầu của người dùng**

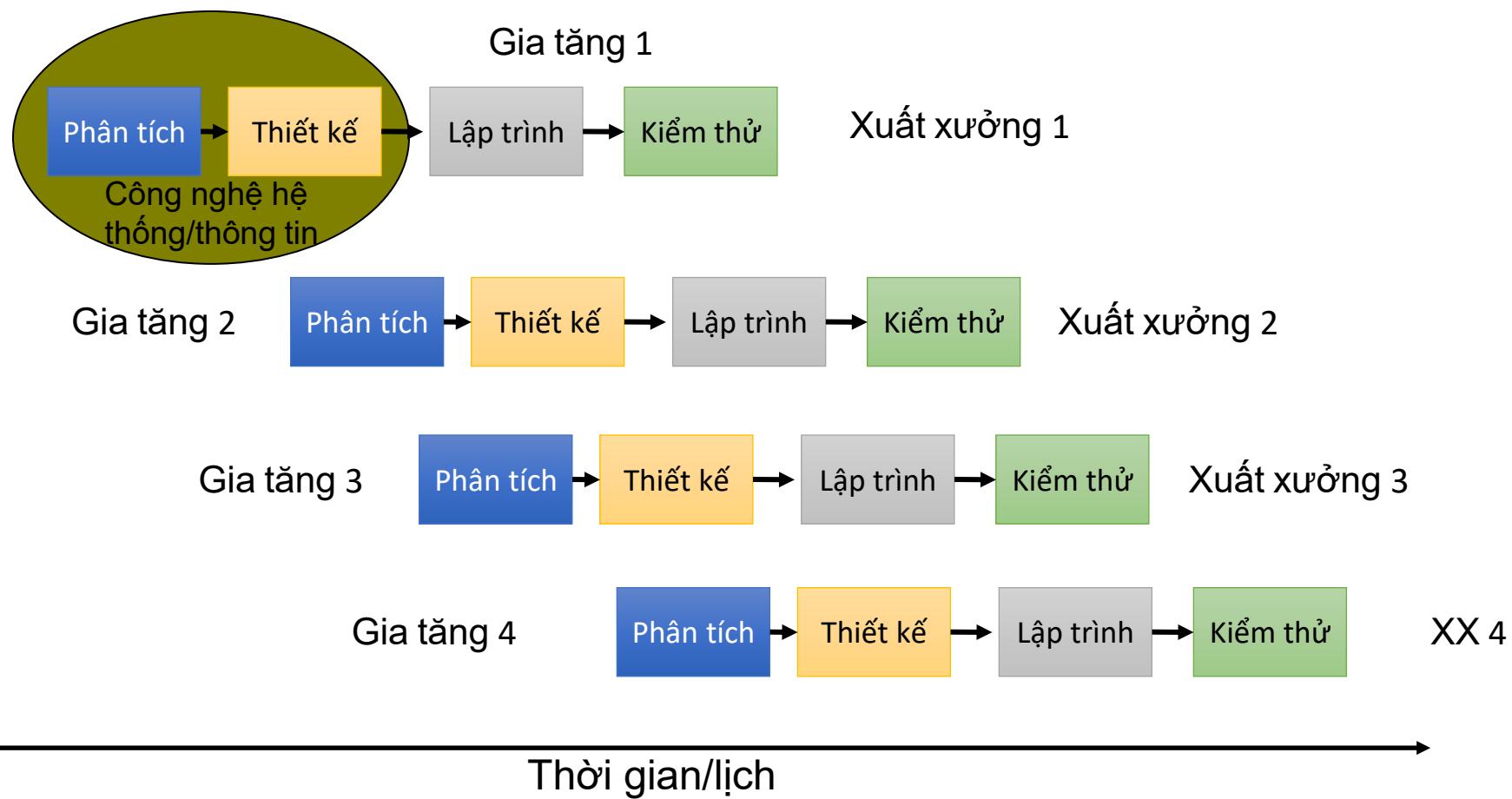
4.3. Các mô hình tăng dần

- Phần lớn các hệ phần mềm phức tạp đều **tiến hóa theo thời gian**: môi trường thay đổi, yêu cầu phát sinh thêm, hoàn thiện thêm chức năng, tính năng
- Các mô hình tiến hóa (evolutionary models) có tính **lặp lại**. Kỹ sư phần mềm tạo ra các phiên bản (versions) ngày càng hoàn thiện hơn, phức tạp hơn
- Các mô hình tiêu biểu:
 - Gia tăng (Incremental)
 - Xoắn ốc (Spiral)
 - Xoắn ốc WINWIN (WINWIN spiral)
 - Phát triển đồng thời (Concurrent development)

4.4. Mô hình gia tăng (The incremental model)

- Kết hợp mô hình **tuần tự** và ý tưởng **lắp lại** của chế bản mẫu
- Sản phẩm với những yêu cầu cơ bản nhất của hệ thống được phát triển
- Các chức năng với những yêu cầu khác được **phát triển thêm sau (gia tăng)**
- **Lắp lại quy trình** để hoàn thiện dần

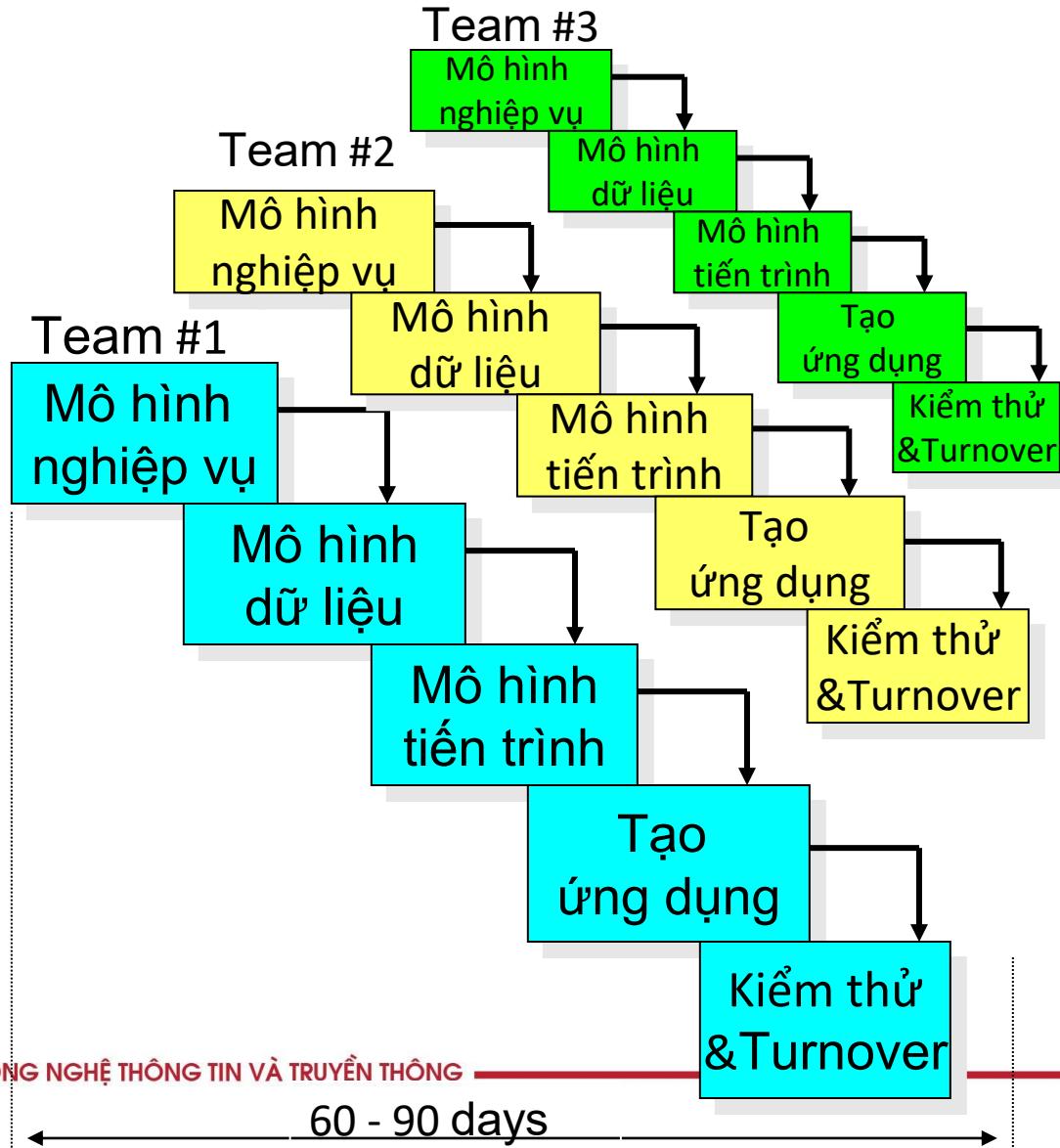
4.4. Mô hình gia tăng



4.5. Mô hình phát triển ứng dụng nhanh (Rapid Application Development: RAD)

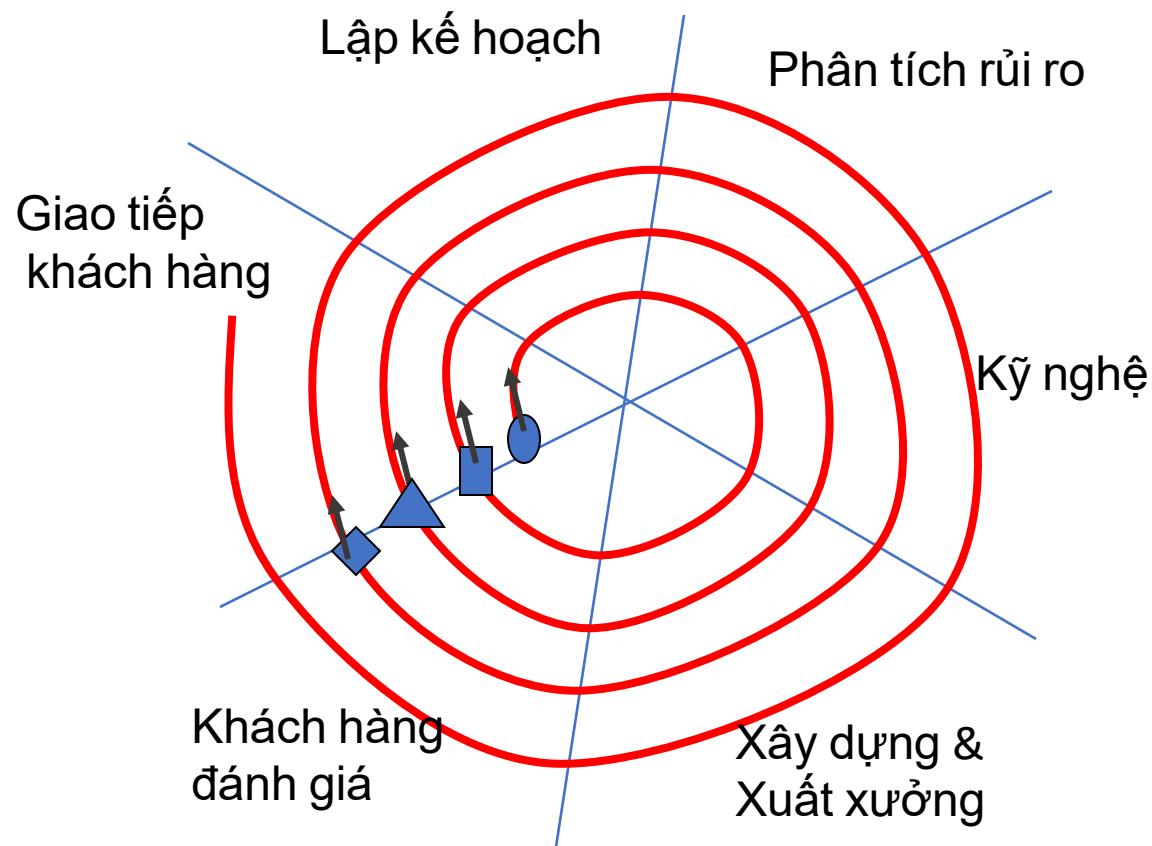
- Là quy trình phát triển phần mềm **gia tăng**, tăng dần từng bước (Incremental software development) với mỗi chu trình phát triển rất ngắn (60-90 ngày)
- Xây dựng dựa trên **hướng thành phần** (Component-based construction) với khả năng **tái sử dụng** (reuse)
- Gồm một số nhóm (teams), mỗi nhóm làm 1 RAD theo các pha: Mô hình nghiệp vụ, Mô hình dữ liệu, Mô hình xử lý, Tạo ứng dụng, Kiểm thử và đánh giá (Business, Data, Process, Appl. Generation, Test)

4.5. Mô hình phát triển ứng dụng nhanh



4.6. Mô hình xoắn ốc (spiral)

- Khái niệm
- Làm mới
- ▲ Nâng cấp
- ◆ Bảo trì



4.6. Mô hình xoắn ốc (tiếp)

- **Giao tiếp khách hàng:** giữa người phát triển và khách hàng để tìm hiểu yêu cầu, ý kiến
- **Lập kế hoạch:** Xác lập tài nguyên, thời hạn và những thông tin khác
- **Phân tích rủi ro:** Xem xét mạo hiểm kỹ thuật và mạo hiểm quản lý
- **Kỹ nghệ:** Xây dựng một hay một số biểu diễn của ứng dụng

4.6. Mô hình xoắn ốc (tiếp)

- **Xây dựng và xuất xưởng:** xây dựng, kiểm thử, cài đặt và cung cấp hỗ trợ người dùng (tư liệu, huấn luyện, . . .)
- **Đánh giá của khách hàng:** Nhận các phản hồi của người sử dụng về biểu diễn phần mềm trong giai đoạn kỹ nghệ và cài đặt

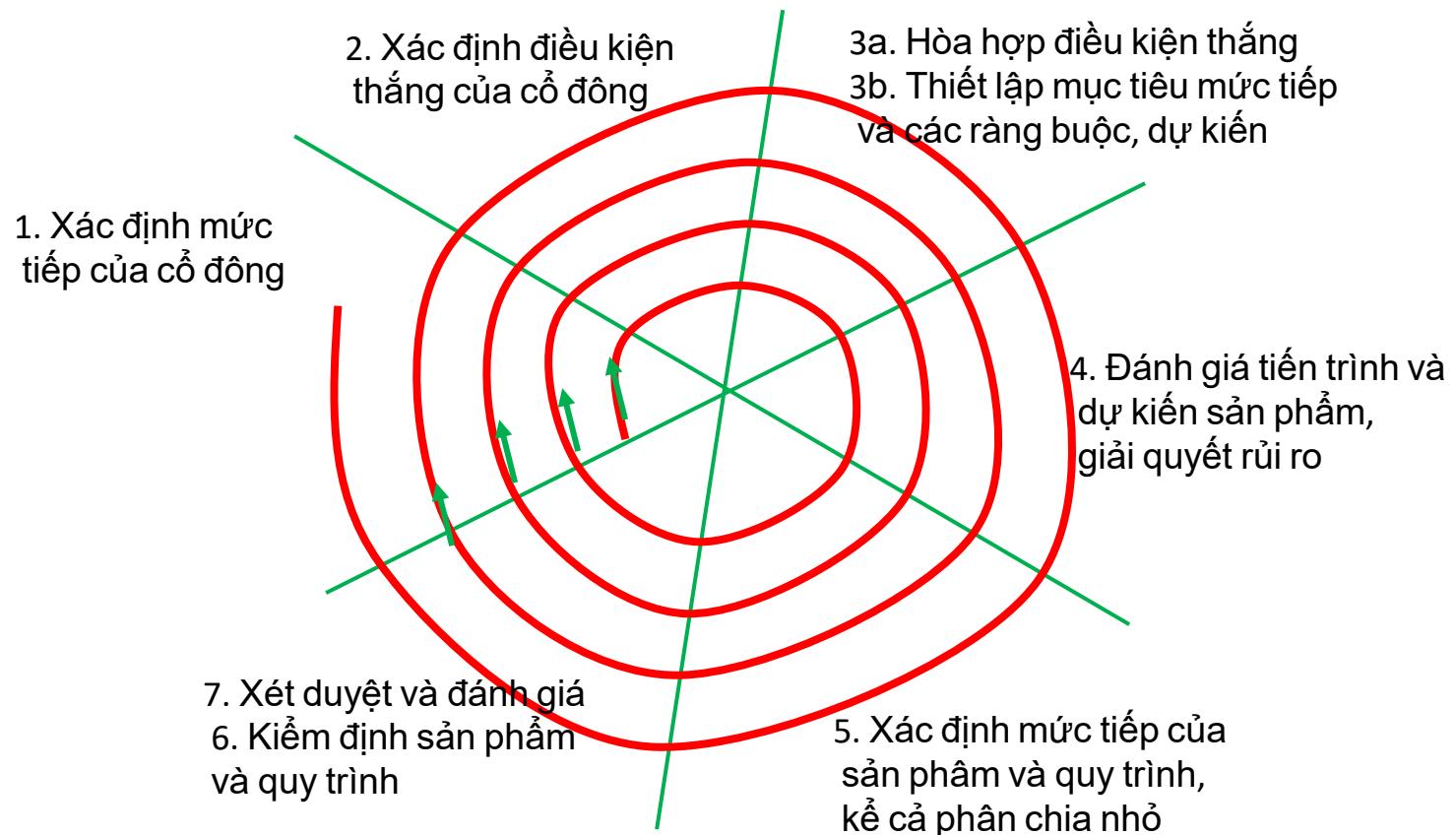
4.6. Mô hình xoắn ốc: Mạnh và yếu?

- Tốt cho các hệ phần mềm **quy mô lớn**
- Dễ **kiểm soát các mạo hiểm** ở từng mức tiến hóa
- **Khó thuyết phục** khách hàng là phương pháp tiến hóa xoắn ốc có thể kiểm soát được
- **Chưa được dùng rộng rãi** như các mô hình tuyến tính hoặc chế thử

4.7. Mô hình xoắn ốc WINWIN

- Nhằm **thỏa hiệp** giữa người phát triển và khách hàng, cả hai cùng “Thắng” (win-win)
 - Khách thì có phần mềm thỏa mãn yêu cầu chính
 - Người phát triển thì có kinh phí thỏa đáng và thời gian hợp lý
- Các hoạt động chính trong xác định hệ thống:
 - Xác định cổ đông (stakeholders)
 - Xác định điều kiện thắng của cổ đông
 - Thỏa hiệp điều kiện thắng của các bên liên quan

4.7. Mô hình xoắn ốc WINWIN



Tổng kết các mô hình

- **Thác nước:** mô hình **tuyến tính**
- **Mẫu thử:** mô hình **lặp đi lặp lại**
- **Gia tăng:** kết hợp giữa mô hình **tuyến tính** và **lặp đi lặp lại**
- **Xoắn ốc:** kết hợp giữa mô hình **tuyến tính** và **lặp đi lặp lại**
- **Phát triển nhanh:** mô hình **lặp đi lặp lại**

Tổng kết

- Vòng đời phần mềm tính **từ khi sinh ra đến khi chết đi.**
- Mô hình vòng đời là một mô tả về **một quá trình thực hiện** một sản phẩm phần mềm trong toàn bộ hoặc một phần vòng đời của nó.
- Các mô hình: **thác nước, mẫu thử, gia tăng, xoắn ốc, phát triển nhanh.**

Q&A



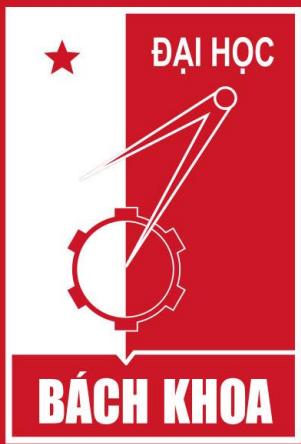


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The graphic consists of the number "25" in large, bold, white font. A curved banner arches over the top of the "2" containing the text "YEARS ANNIVERSARY". Below the "25" is the acronym "SOKT" in a large, bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 3

Phương pháp Agile

Nội dung

1. Khái niệm
2. Các nguyên lý cơ bản của phương pháp Agile
3. Ưu, nhược điểm của phương pháp Agile
4. Extreme Programming
5. Scrum
6. Các phương pháp Agile khác

Mục tiêu của bài học

- Biết phương pháp phát triển phần mềm nhanh (Agile)
- Hiểu các nguyên lý cơ bản của phương pháp agile
- Biết một số phương pháp phát triển phần mềm nhanh

1. Khái niệm

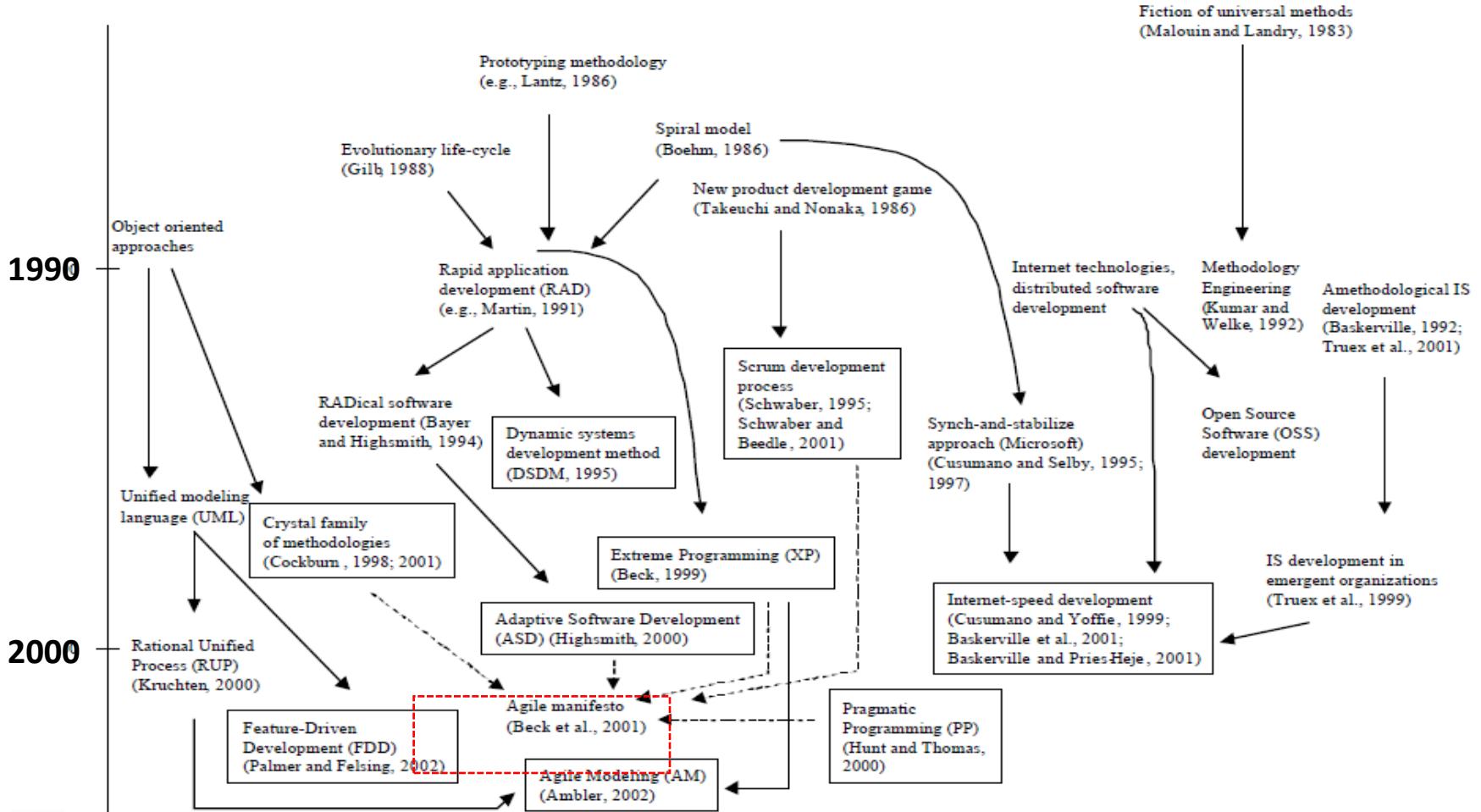
- Phương pháp Agile:
 - phản ứng hiệu quả (nhanh chóng và thích ứng) với sự thay đổi
 - kết hợp quá trình tạo mẫu và dựa trên thử nghiệm
 - có sự phát triển liên tục của lập trình
 - liên tục xác nhận các yêu cầu của khách hàng
 - mục tiêu: nhanh chóng phân phối phần mềm

Lịch sử của Agile

- Agile không phải là một bộ công cụ hay một phương pháp duy nhất, mà là một **triết lý** được đưa ra vào năm 2001.
- Agile **thay đổi** đáng kể từ **cách tiếp cận** phát triển phần mềm nặng **theo hướng tài liệu** (như mô hình thác nước).

Lịch sử của Agile

Pekka Abrahamsson, Juhani Warsta, Mikko T. Siponen, and Jussi Ronkainen. 2003. New directions on agile methods: a comparative analysis. In *Proceedings of the 25th International Conference on Software Engineering (ICSE '03)*. IEEE Computer Society, Washington, DC, USA, 244-254.



Nội dung

1. Khái niệm
2. Các nguyên lý cơ bản của phương pháp Agile
3. Ưu, nhược điểm của phương pháp Agile
4. Extreme Programming
5. Scrum
6. Các phương pháp Agile khác

2. Các nguyên lý cơ bản của phương pháp Agile

- Cách **tốt hơn** để phát triển phần mềm là **làm và giúp** người khác làm. Từ đó, các giá trị sau sẽ được nhận ra:
 - Các cá nhân và tương tác qua các quy trình và công cụ
 - Phần mềm làm việc dựa trên tài liệu toàn diện
 - Sự hợp tác của khách hàng trong quá trình đàm phán
 - Đáp ứng sự thay đổi so với việc tuân theo kế hoạch
- Các mục **bên trái** được coi trọng hơn **bên phải**.

<http://agilemanifesto.org/>

2. Agile: 12 nguyên lý

1. Ưu tiên cao nhất của là làm **hài lòng khách hàng** thông qua việc **phân phối sớm** và liên tục các phần mềm có giá trị.
2. Hoan nghênh các **yêu cầu thay đổi**, ngay cả trong giai đoạn muộn của việc phát triển. Các quy trình nhanh khai thác sự thay đổi vì lợi thế cạnh tranh của khách hàng.
3. **Cung cấp sản phẩm phần mềm thường xuyên**, từ vài tuần đến vài tháng, ưu tiên khoảng thời gian ngắn hơn.
4. Người kinh doanh và nhà phát triển phải **làm việc cùng nhau** hàng ngày trong suốt dự án.
5. Xây dựng các dự án xung quanh những **cá nhân có động lực**. Cung cấp cho họ môi trường và sự hỗ trợ mà họ cần, và tin tưởng để họ hoàn thành công việc.
6. Phương pháp hiệu quả nhất để truyền tải thông tin đến và trong nhóm phát triển là **trò chuyện trực tiếp**.

2. Agile: 12 nguyên lý

7. Phần mềm làm việc là thước đo chính của **sự tiến bộ**.
8. Các quy trình Agile thúc **đẩy sự phát triển** bền vững. Các nhà tài trợ, nhà phát triển và người dùng sẽ có thể duy trì một tốc độ phát triển liên tục.
9. **Sự quan tâm** liên tục, sự xuất sắc về kỹ thuật và thiết kế tốt giúp tăng cường sự nhanh nhẹn.
10. **Sự đơn giản** - nghệ thuật tối đa hóa khối lượng công việc chưa hoàn thành - là điều cần thiết.
11. Các **kiến trúc, yêu cầu** và **thiết kế** tốt nhất **xuất hiện** từ các **nhóm dự án**.
12. Theo định kỳ, các nhóm **phản ánh** về cách trở nên hiệu quả hơn, sau đó **điều chỉnh** hoạt động của mình sao cho cho phù hợp.

Nội dung

1. Khái niệm
2. Các nguyên lý cơ bản của phương pháp Agile
3. **Ưu, nhược điểm của phương pháp Agile**
4. Extreme Programming
5. Scrum
6. Các phương pháp Agile khác

Ưu, nhược điểm của phương pháp Agile

- Phù hợp với những **dự án nhỏ** thường có những yêu cầu không được xác định rõ ràng (có thể thay đổi thường xuyên).
- Khách hàng có thể được **xem trước từng phần** dự án trong quá trình phát triển, luôn **sẵn sàng cho bất kỳ thay đổi** từ phía khách hàng.
- Agile chia dự án thành những phần nhỏ và giao cho nhóm phát triển, hàng ngày tất cả mọi người phải họp trong khoảng thời gian ngắn **để thảo luận về tiến độ và giải quyết những vấn đề này sinh** nếu có nhằm đảm bảo đúng quy trình phát triển dự án.
- **Tỉ lệ thành công** của các dự án sử dụng Agile thường cao hơn các quy trình khác.

Ưu, nhược điểm của phương pháp Agile

- **Khó xác định** về loại dự án phần mềm nào phù hợp nhất cho cách tiếp cận Agile.
- Nhiều tổ chức lớn gặp **khó khăn** trong việc chuyển từ **phương pháp truyền thống** sang một phương pháp Agile (linh hoạt).
- Khi **Agile** có rủi ro:
 - Phát triển **quy mô lớn** (> 20 nhà phát triển)
 - Phát triển **phân tán** (các nhóm không nằm chung)
 - Khách hàng hoặc **người liên hệ** không đáng tin cậy
 - Bắt buộc **quy trình nhanh** trong nhóm phát triển
 - Nhà phát triển **thiếu kinh nghiệm**

Nội dung

1. Khái niệm
2. Các nguyên lý cơ bản của phương pháp Agile
3. Ưu, nhược điểm của phương pháp Agile
4. Extreme Programming
5. Scrum
6. Các phương pháp Agile khác

Extreme Programming (XP)



XP is not this *extreme*!

Extreme Programming (XP)

- Không thể lập trình cho đến khi biết mình đang làm gì.
- Cần khắc phục ở giai đoạn đầu tiên khi hệ thống đi vào sản xuất càng nhanh càng tốt. Tuy nhiên, mọi dự án đều phải bắt đầu từ điểm nào đó.
- Kết hợp phân tích tổng thể dưới dạng các vấn đề. Mỗi vấn đề phải theo định hướng kinh doanh, có thể kiểm tra và ước tính được.
- Một tháng là một khoảng thời gian dài để đưa ra những vấn đề cho một dự án.

Embracing Change with Extreme Programming, Beck, 1999

Extreme Programming (XP)

- Xác định thời điểm lập trình?
 - Khách hàng chọn bản phát hành tiếp theo tính năng có giá trị nhất (được gọi là các vấn đề trong XP) trong số tất cả các vấn đề có thể có, xác định bởi chi phí của các vấn đề và tốc độ của nhóm trong việc cài đặt chúng.
 - Khách hàng xác định vấn đề của **lần lắp tiếp theo** bằng cách **chọn những vấn đề có giá trị nhất còn lại** trong bản phát hành, được thông báo lại bằng chi phí của và tốc độ của nhóm.

Extreme Programming (XP)

- Xác định thời điểm lập trình?
 - Các lập trình viên nhận trách nhiệm cho các nhiệm vụ chi tiết.
 - Xác định các trường hợp kiểm thử để chứng minh rằng nhiệm vụ đã hoàn thành.
 - Làm việc với đối tác để chạy các trường hợp kiểm thử, đồng thời cải tiến để duy trì một thiết kế đơn giản nhất có thể cho toàn bộ hệ thống.

Extreme Programming (XP)

- Để XP thành công, **cần có kiến thức chuyên môn.**
 - Xây dựng, mô tả các vấn đề cần giải quyết của hệ thống đang được xây dựng.
 - Không mô tả một giải pháp, sử dụng ngôn ngữ kỹ thuật hoặc chứa các yêu cầu truyền thống.
 - Chuyển vấn đề thành mã.

Extreme Programming (XP)

- Biến thành mã thử nghiệm
- **Kiểm thử đơn vị là trọng tâm** của XP, trong đó hai điểm xoay quanh các chiến lược kiểm thử thông thường giúp kiểm tra hiệu quả hơn nhiều:
- Các lập trình viên viết các test riêng và cần viết trước khi viết mã.
- Thiết kế phát triển - Không được phá vỡ các thử nghiệm hiện có và cũng phải hỗ trợ phát triển gia tăng có thể mở rộng

Ưu điểm của XP

- Nếu được hoàn thành tốt, XP **cải thiện tính đồng đội**.
- **xây dựng năng lực** thực sự ở tất cả các thành viên trong nhóm.
- làm cho công việc thú vị.
- TDD (Test-driven development) hướng dẫn các nhà phát triển về cách viết mã chất lượng và cách cải thiện quan niệm về thiết kế; giúp cải thiện việc ước lượng.

Ưu điểm của XP

- Cung cấp **nhiều công cụ** cho quản lý , bao gồm khả năng dự đoán, tính linh hoạt của các nguồn lực, tính nhất quán và khả năng hiển thị về những gì thực sự đang diễn ra.
- Cung cấp cho khách hàng **khả năng xem xét** công ty có thể thực hiện công việc của mình hay không.
- **Không lãng phí thời gian** và không tạo ra nhiều tài liệu vô ích.

Nhược điểm của XP

- Thiết kế trở nên tiềm ẩn thay vì rõ ràng
- Dựa vào thiết kế là **rủi ro**
- Rất khó để viết các kiểm thử tốt
- Lặp lại quá thường xuyên có thể làm giảm chất lượng
- Để thực hiện tốt, cần phải **làm thường xuyên**

Nội dung

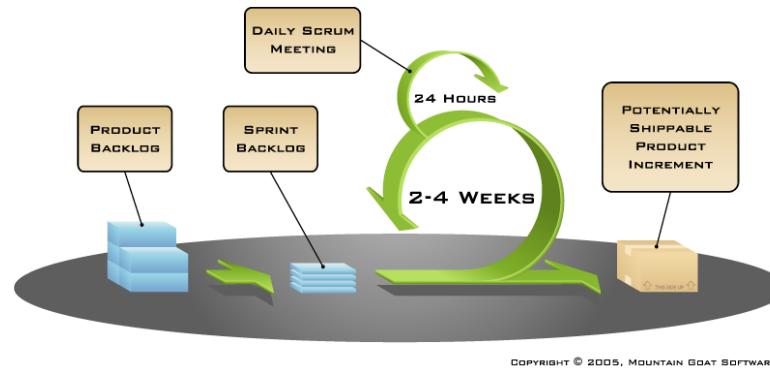
1. Khái niệm
2. Các nguyên lý cơ bản của phương pháp Agile
3. Ưu, nhược điểm của phương pháp Agile
4. Extreme Programming
5. Scrum
6. Các phương pháp Agile khác

Scrum

- ‘Scrum’ là thuật ngữ chỉ một **nhóm người chơi với nhau** trong môn bóng bầu dục.
- Trong phát triển phần mềm, **công việc là đưa ra một bản phát hành**.
- Nhiệm vụ: **tăng tốc** việc phát hành sản phẩm

Scrum

- Quá trình phát triển được chia thành một loạt **các lần lặp** được gọi là **sprint**.
- Trước mỗi sprint, các thành viên trong nhóm **xác định** các công việc còn tồn.
- Khi kết thúc sprint, nhóm sẽ xem xét để **nêu rõ** các kinh nghiệm và kiểm tra tiến độ.

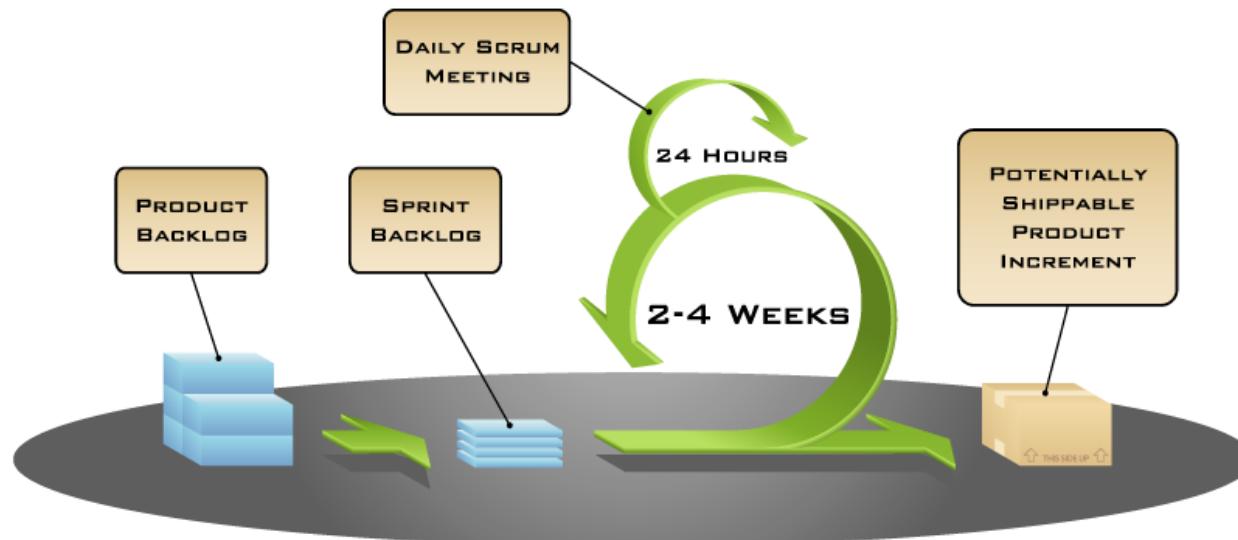


Các khái niệm chính

- **Burndown chart** biểu đồ cho thấy **công việc còn tồn** trong sprint (được cập nhật mỗi ngày) sử dụng để theo dõi tiến trình và quyết định khi nào các mục phải được loại bỏ khỏi sprint backlog và hoãn lại sprint tiếp theo.
- **Product backlog** là **danh sách đầy đủ các yêu cầu** - bao gồm lỗi, yêu cầu nâng cao, cải tiến khả năng sử dụng và hiệu suất - hiện không có trong bản phát hành sản phẩm.
- **ScrumMaster** là **người chịu trách nhiệm quản lý dự án**.
- **Sprint backlog** là **danh sách các công việc** được chỉ định cho một sprint, nhưng chưa hoàn thành.
 - Trong thực tế (phổ biến), không có công việc tồn nào của sprint phải mất hơn hai ngày để hoàn thành.
 - Sprint backlog giúp nhóm dự đoán mức độ nỗ lực cần thiết để hoàn thành một sprint.

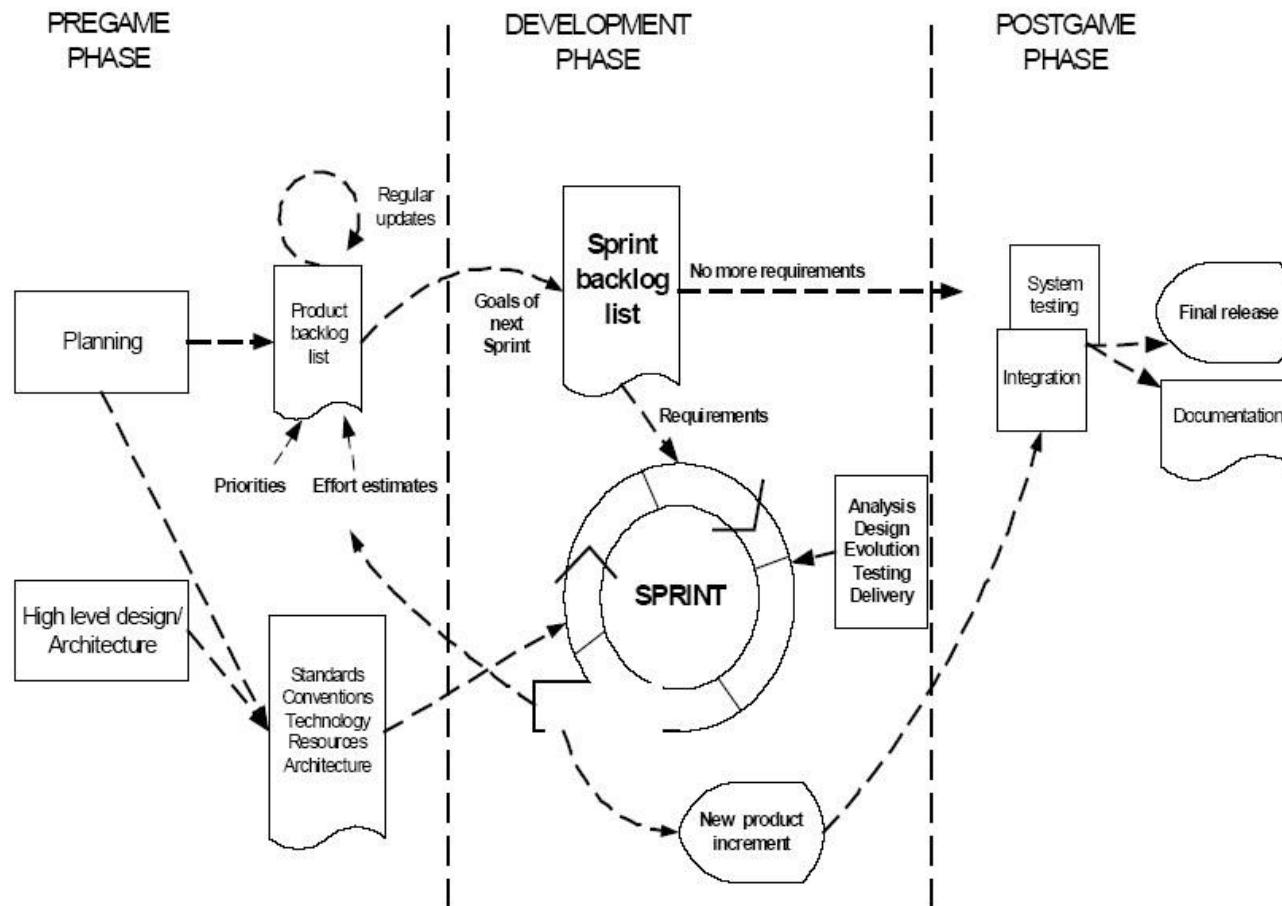
Scrum

- Các công việc tồn (là điểm mấu chốt): được điền vào trong pha lập kế hoạch của lần phát hành sản phẩm và xác định phạm vi của bản phát hành



COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE

Các pha của Scrums



Scrum Meetings

- Trong một sprint, nhóm có một **cuộc họp hàng ngày**.
 - Mỗi thành viên mô tả công việc sẽ được thực hiện trong ngày hôm đó, tiến độ từ ngày hôm trước.
 - Để giữ cho các cuộc họp ngắn gọn, tất cả mọi người đều tham dự (họp đứng trong cùng một phòng).
- Khi các backlog đã được thực hiện để người dùng tin rằng bản phát hành đáng được đưa vào sản xuất, kết thúc quá trình phát triển, nhóm thực hiện kiểm tra tích hợp, đào tạo và làm tài liệu khi cần thiết để phát hành sản phẩm.

Scrum

Ưu điểm (Pros)	Nhược điểm (Cons)
Scrum sử dụng Sprint nhỏ để chia hệ thống thành các thành phần nhỏ hơn một cách hiệu quả và phân chia cho các nhóm.	Scrum chỉ hoạt động khi người quản lý “tin tưởng các nhóm phát triển”. Scrum phụ thuộc vào khả năng kiểm soát , vì vậy, nếu đội ngũ phát triển còn trẻ và chưa trưởng thành thì Scrum trở nên rất rủi ro.
Một hoạt động chính trong scrum là “cuộc họp hàng ngày” , giúp các thành viên trong nhóm đưa ra bằng chứng về việc hoàn thành nhiệm vụ và cho phép cải tiến liên tục, do đó cho phép áp dụng kỹ thuật từ dưới lên một cách nhanh chóng	Scrum được thiết kế lý tưởng cho công ty có “các phương pháp nhanh hiện có”. Do đó, một công ty phải có kiến thức về phương pháp làm việc trước khi sử dụng Scrum.

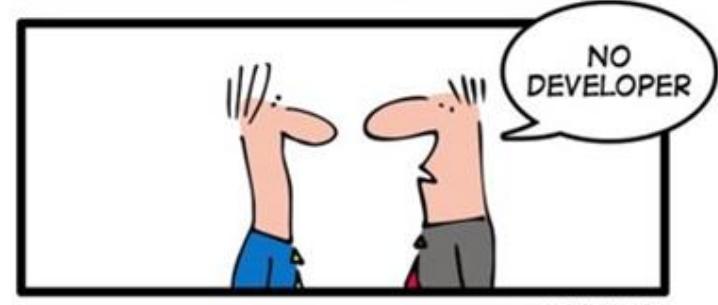
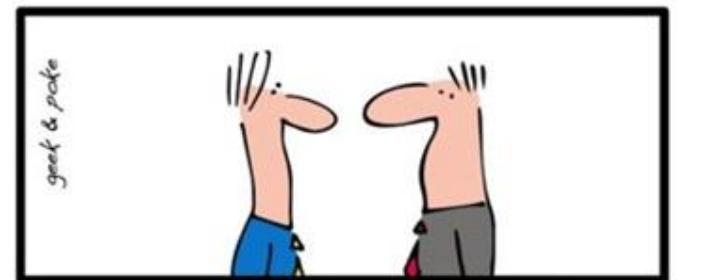
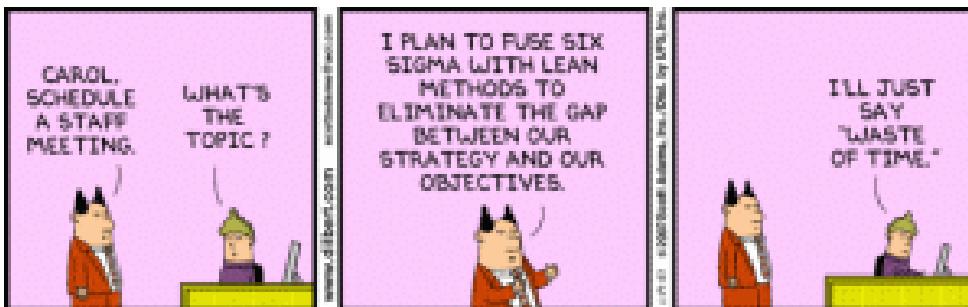
Nội dung

1. Khái niệm
2. Các nguyên lý cơ bản của phương pháp Agile
3. Ưu, nhược điểm của phương pháp Agile
4. Extreme Programming
5. Scrum
6. Các phương pháp Agile khác

Lean development (phát triển tinh gọn)

J Paul Gibson: Agile Methods

October 2011



LEAN IT

Lean Software Development

- Mọi thứ sẽ dễ dàng hơn nếu khách hàng ngừng thay đổi suy nghĩ của họ.
- Hãy để khách hàng trì hoãn quyết định về những gì họ muốn càng lâu càng tốt và khi họ yêu cầu hãy để họ không có thời gian để thay đổi.
- Các thiết kế xuất hiện khi các nhà thiết kế phát triển sự hiểu biết về vấn đề, chứ không phải thu thập số lượng lớn các yêu cầu.
- Cung cấp hệ thống nhanh nhất có thể.

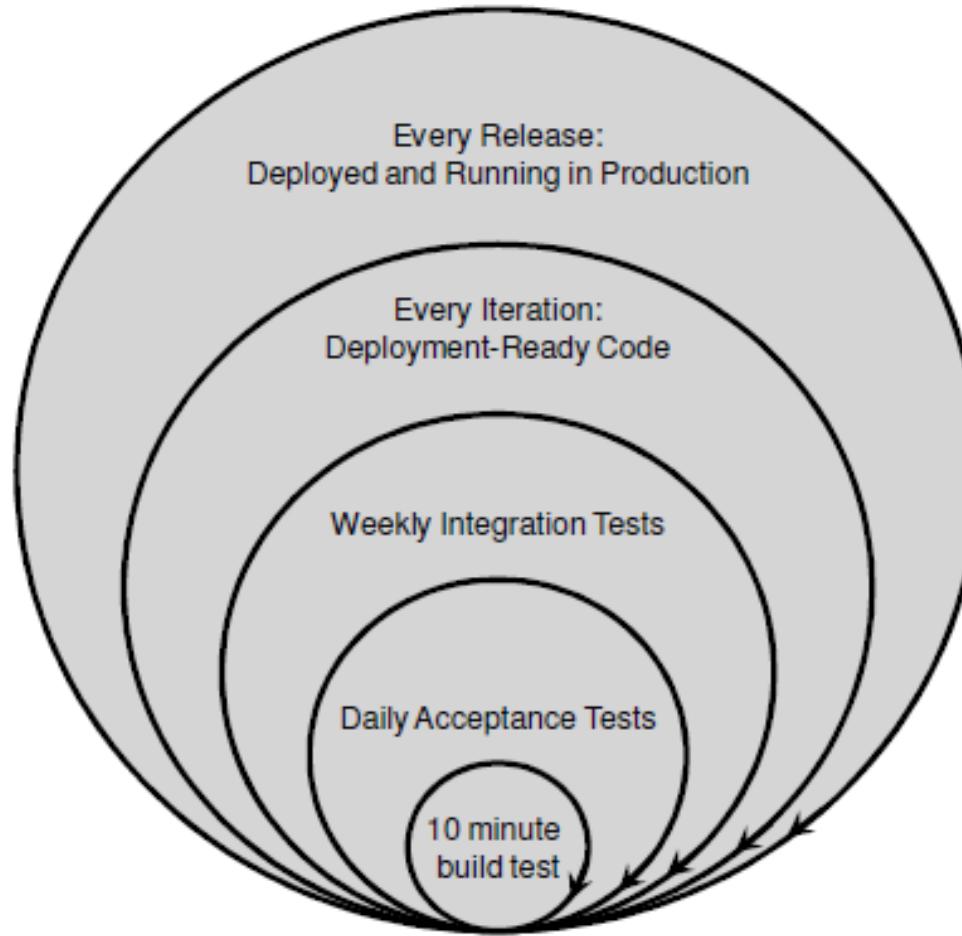
Lean Software Development

- **Eliminate waste.** Giảm thiểu sự lãng phí: Bất kỳ hoạt động nào không “tự trả giá” trong nỗ lực giảm thiểu ở những nơi khác trong hệ thống và cần được loại bỏ.
- **Amplify learning.** Nhấn mạnh vào việc học: Các nhà phát triển luôn cần học các phương pháp mới để tạo ra một hệ thống mạnh mẽ nhất.
- **Đưa ra quyết định muộn nhất có thể.** Tránh đưa ra quyết định sai lầm sớm và sau đó phải sửa chữa nó sau này.
- **Cung cấp nhanh nhất có thể.** Làm giảm đi việc thay đổi yêu cầu. Điều này có thể phải trả giá lớn nếu sự thay đổi đến muộn trong quá trình phát triển.

Lean Software Development

- **Để mọi người chịu trách nhiệm:** có động lực và gắn bó với nhau như một đội, cần phải chịu trách nhiệm về kết quả và được ủy quyền để biến nó thành hiện thực.
- **Duy trì tính toàn vẹn:** phải kiểm thử tích hợp, kiểm thử đơn vị và kiểm thử chung, đặc biệt là từ khách hàng.
- **Xem xét toàn bộ hệ thống:** đừng chia nhỏ hệ thống thành nhiều phần.

Tập trung vào kiểm thử



Lean Software Development

Ưu điểm	Nhược điểm
Tư duy toàn bộ hệ thống, mặc dù khó đối với hệ thống phức tạp, giúp đảm bảo tính nhất quán và toàn vẹn của hệ thống. Giảm thời gian tích hợp kể từ khi được phát triển.	Khó thực hiện việc hình dung cấu trúc của một hệ thống phức tạp là thông qua việc phân vùng hệ thống.
Nhóm làm việc thiết kế các quy trình của riêng mình, đưa ra các cam kết riêng, thu thập thông tin cần thiết để đạt được mục tiêu và tự lập chính sách để đạt được các mốc quan trọng của mình.	Lịch trình sẽ bị ảnh hưởng xấu bởi các quyết định muộn. Điều này có thể làm tổn hại đến sự phát triển song song và làm tăng thời gian thực hiện.

Tổng kết

- Phương pháp phát triển phần mềm nhanh: **phát triển liên tục, nhanh chóng phân phối sản phẩm.**
- Một số mô hình phát triển phần mềm nhanh:
 1. Extreme Programming
 2. Scrum
 3. Lean development

Q&A



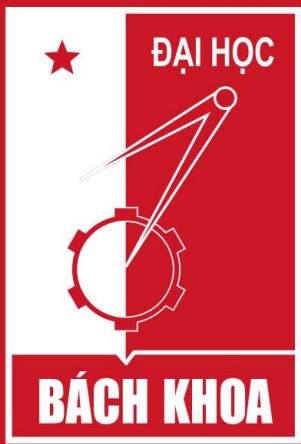


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The central focus of the banner is a large, white, stylized number "25" with a circular arc above it containing the text "YEARS ANNIVERSARY". To the right of the "25" is the acronym "SOKT" in a bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 4

Quản lý dự án PM

Mục tiêu của bài học

Sinh viên sẽ được trang bị các kiến thức sau:

- Các khái niệm liên quan tới QLDA
- Tổng quan các quy trình cần thực hiện trong quá trình QLDA
- Một số quy trình quan trọng như: Lập kế hoạch, quản lý rủi ro

Nội dung

1. Khái niệm chung

- a. Giới thiệu – định nghĩa
- b. Yếu tố con người
- c. Yếu tố sản phẩm

2. Quy trình quản lý dự án

- a. Tổng quan
- b. Ước lượng dự án
- c. Lập kế hoạch dự án
- d. Quản lý rủi ro dự án

1. Khái niệm chung

a. Giới thiệu - định nghĩa:

- **Dự án (project)**: Một dự án là một **công việc có thời hạn** nhằm **tạo ra một sản phẩm**, dịch vụ hay kết quả duy nhất.
 - Tính thời hạn (Temporariness) : có điểm bắt đầu và điểm kết thúc
 - Tính duy nhất (Uniqueness) :
 - Dự án là riêng biệt, độc lập
 - Có sản phẩm cụ thể cuối cùng
 - Sản phẩm hoặc môi trường dự án là duy nhất
 - Mang lại yếu tố mới cho đội ngũ thực hiện

→ Dự án cần được quản lý với giả định sẽ xảy ra thay đổi.

1. Khái niệm chung

a. Giới thiệu - định nghĩa:

Dự án phần mềm:

- Do **đội ngũ thành viên** gồm ít nhất 2 người thực hiện
- **Giới hạn** về thời gian, ngân sách, và nhân lực
- Sản phẩm là **phần mềm mới** hoặc **phần mềm có sẵn được cải tiến**
- Sản phẩm phải góp phần tạo dựng quy trình nghiệp vụ mới, hữu ích, hoặc mang lại lợi ích đáng kể cho quy trình nghiệp vụ hiện có.

1. Khái niệm chung

a. Giới thiệu - định nghĩa:

- Quản lý dự án là áp dụng kiến thức, kỹ năng, công cụ và kỹ thuật vào các hoạt động của dự án nhằm đáp ứng yêu cầu của dự án.
 - Đạt mục tiêu dự án
 - Đạt hoặc vượt các yêu cầu hay kỳ vọng của những người có quyền lợi và nghĩa vụ liên quan (stakeholders)
 - Cân bằng giữa các yếu tố: thời gian, chi phí, chất lượng sản phẩm

1. Khái niệm chung

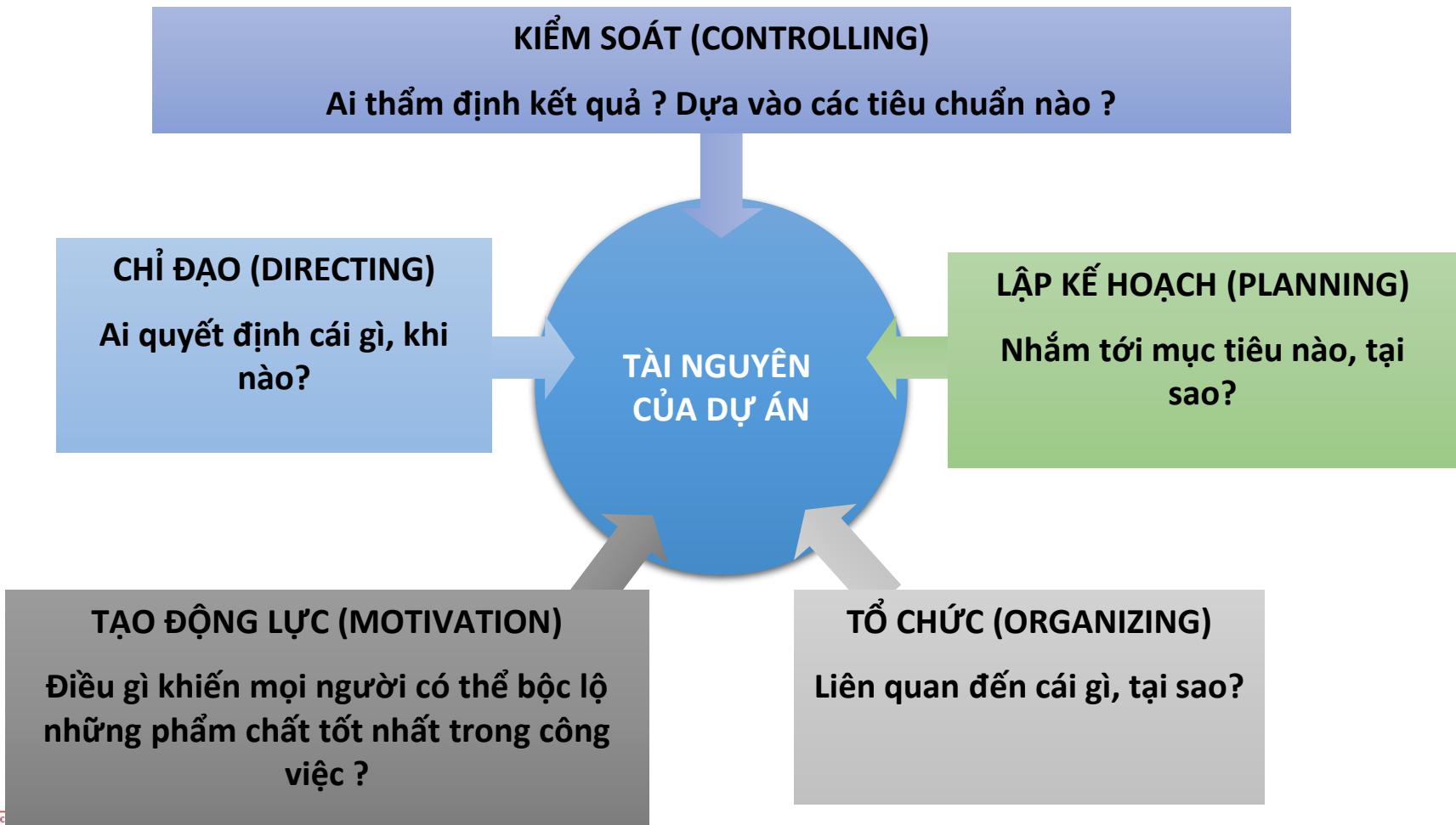
a. Giới thiệu - định nghĩa:

Một số yếu tố: Giá thành + Thời gian + Chất lượng

- Quản lý dự án là để **đưa ra một sản phẩm cuối cùng**:
 - trong phạm vi ngân sách hay nguồn tài chính cho phép
 - đúng hạn
 - với nguồn lực cho phép
 - phù hợp với đặc tả
 - chất lượng đủ để phục vụ các nhu cầu kinh doanh và đáp ứng các tiêu chuẩn chuyên môn và kỳ vọng của công tác quản lý

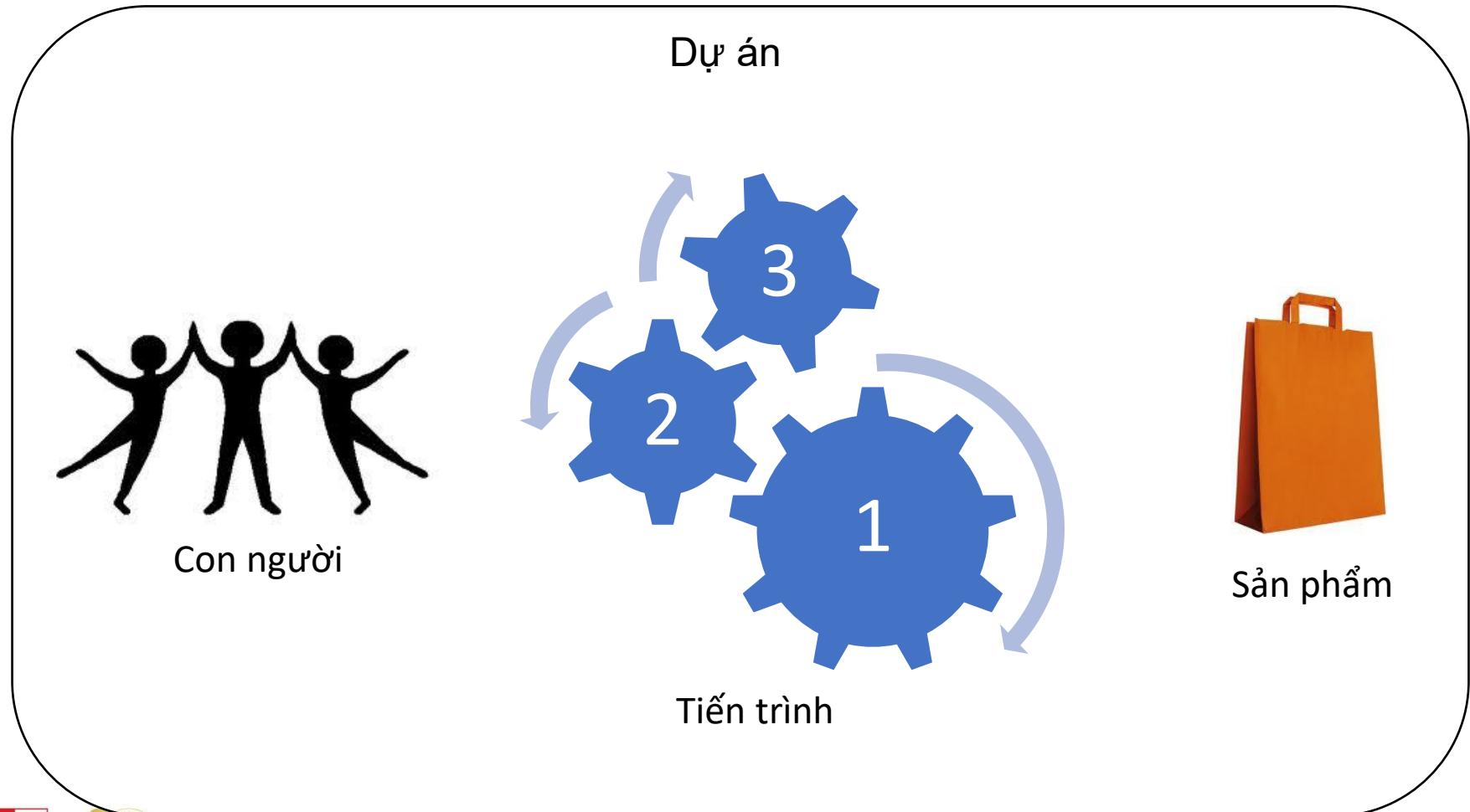
1. Khái niệm chung

a. **Giới thiệu - định nghĩa:** Các nhiệm vụ trong quản lý dự án



1. Khái niệm chung

a. **Giới thiệu - định nghĩa:** Các lĩnh vực trong quản lý dự án



Nội dung

1. Khái niệm chung

- a. Giới thiệu – định nghĩa
- b. Yếu tố con người**
- c. Yếu tố sản phẩm

2. Quy trình quản lý dự án

- a. Tổng quan
- b. Ước lượng dự án
- c. Lập kế hoạch dự án
- d. Quản lý rủi ro dự án

1. Khái niệm chung- yếu tố con người

4 yếu tố liên quan đến quản lý dự án

- **Con người** – yếu tố quan trọng nhất của dự án
- **Sản phẩm** – phần mềm được xây dựng
- **Quy trình** – tập hợp các hoạt động chương trình khung và nhiệm vụ cài phần mềm để thực hiện công việc
- **Dự án** – tất cả các công việc để sản phẩm trở thành hiện thực

1. Khái niệm chung - yếu tố con người

Các thành phần liên quan (Stakeholder)

- **Quản lý cấp cao:** người xác định vấn đề nghiệp vụ, thường có ảnh hưởng lớn tới dự án.
- **Quản lý dự án(kỹ thuật):** người lên kế hoạch, động viên, tổ chức và kiểm soát những người thực hiện phần mềm.
- **Người thực hiện:** có kĩ năng kỹ thuật cần thiết để thiết kế sản phẩm hoặc ứng dụng.
- **Khách hàng:** người đưa ra các yêu cầu cho phần mềm được thiết kế.
- **Người sử dụng cuối cùng:** người tương tác với phần mềm khi nó được phát hành.

1. Khái niệm chung - yếu tố con người

Nhóm phần mềm (Software Team)



1. Khái niệm chung - yếu tố con người

Team Leader

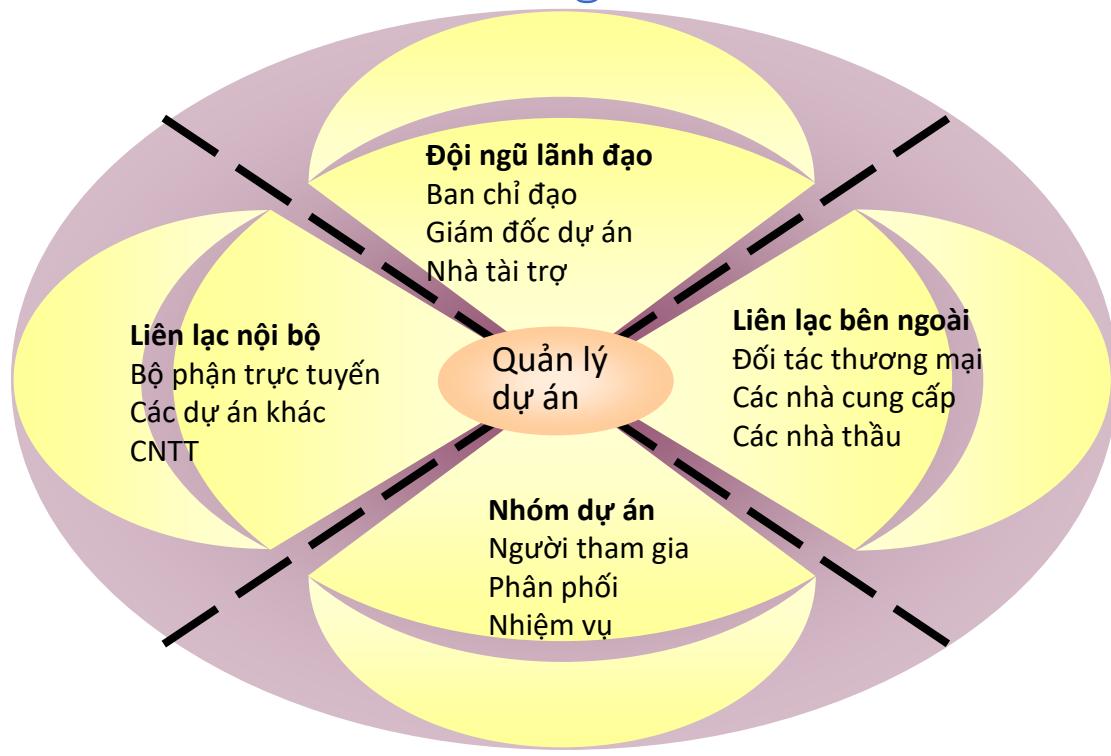
- Mô hình MOI (Motivation- Organization- Ideas or innovation)
 - **Động viên.** Khả năng khích lệ (bằng cách kéo hoặc đẩy) nhân viên kỹ thuật để tạo ra sản phẩm với khả năng tốt nhất
 - **Tổ chức.** Khả năng định hình quá trình hiện có (hoặc phát minh ra cái mới) để có thể từ ban đầu tạo ra sản phẩm cuối cùng
 - **Ý tưởng và cải tiến.** Khả năng khuyến khích mọi người tạo ra và cảm thấy sáng tạo ngay cả khi họ làm việc trong giới hạn đã được thiết lập của sản phẩm hay ứng dụng

1. Khái niệm chung - yếu tố con người

Người quản lý dự án (PM – Project manager)

- Làm thế nào để tăng khả năng:
 - Tạo ra sản phẩm có chất lượng
 - Tôn trọng lịch trình thực hiện
 - Thỏa mãn yêu cầu của khách hàng
 - Tạo khả năng kinh doanh
 - Đạt được thành công

Việc quản lý dự án chịu trách nhiệm về tất cả mọi thứ là cần thiết để thực hiện các dự án thành công



PM = Tâm điểm giao tiếp

- Không phải là công việc bán thời gian
- Phải biết chu kỳ sống của dự án, các tiến trình của dự án và vai trò của các tiến trình này trong việc thực hiện các công việc ở các pha khác nhau trong chu kỳ sống của dự án
- Nhận biết được sự phức tạp của môi trường thực hiện dự án
- Phải được chuẩn bị để đối phó với các mối xung đột khác nhau



Hầu hết các dự án thất bại vì thiếu quản lý dự án và quản lý con người, không phải vì lý do kỹ thuật

1. Khái niệm chung - yếu tố con người

Các mô hình tổ chức/giao tiếp (Communication issues)

- **Mô hình đóng**—cấu trúc một nhóm với sự phân quyền truyền thống
- **Mô hình ngẫu hứng**—cấu trúc một nhóm lỏng lẻo và phụ thuộc vào sáng kiến cá nhân của mỗi thành viên
- **Mô hình mở**—cố gắng cấu trúc một nhóm mà cách thi hành kết hợp giữa mô hình chặt chẽ và mô hình ngẫu hứng
- **Mô hình đồng bộ**—dựa trên sự phân chia tự nhiên của một vấn đề và các thành viên trong nhóm sẽ làm việc trên các thành phần của vấn đề đó với ít sự giao tiếp lẫn nhau giữa các thành phần của vấn đề

1. Khái niệm chung - yếu tố con người

Phối hợp và giao tiếp nhóm (Communication issues)

- **Chính thức, cách tiếp cận cá nhân:** bao gồm các tài liệu kĩ thuật phần mềm và các sản phẩm công việc (bao gồm cả mã nguồn), bản ghi nhớ kĩ thuật, sự kiện quan trọng của dự án, lịch trình và các công cụ kiểm soát dự án.(chương 23), yêu cầu thay đổi và các tài liệu liên quan, báo cáo theo dõi lỗi, và dữ liệu lưu trữ
- **Chính thức, thủ tục giao tiếp cá nhân:** tập trung vào các hoạt động đảm bảo chất lượng (Chương 25) áp dụng cho các sản phẩm công việc kĩ thuật phần mềm. Chúng bao gồm các cuộc họp đánh giá tình trạng và kiểm tra thiết kế, code.
- **Không chính thức, thủ tục giao tiếp cá nhân:** bao gồm các cuộc họp nhóm để phổ biến thông tin và giải quyết vấn đề và sắp xếp các yêu cầu của thành viên
- **Giao tiếp điện tử:** thông qua thư điện tử và họp qua truyền hình.
- **Mạng lưới giao tiếp cá nhân:** bao gồm các cuộc tranh luận không chính thức với thành viên trong nhóm hoặc người ngoài nhóm để trợ trong công việc

Nội dung

1. Khái niệm chung

- a. Giới thiệu – định nghĩa
- b. Yếu tố con người
- c. **Yếu tố sản phẩm**

2. Quy trình quản lý dự án

- a. Tổng quan
- b. Ước lượng dự án
- c. Lập kế hoạch dự án
- d. Quản lý rủi ro dự án

1. Khái niệm chung- yếu tố sản phẩm

Phạm vi sản phẩm

- Phạm vi
 - **Hoàn cảnh:** Làm thế nào để phần mềm được xây dựng phù hợp với hệ thống, sản phẩm hoặc hoàn cảnh kinh doanh lớn hơn, và các ràng buộc nào của hoàn cảnh
 - **Mục tiêu Thông tin:** khách hàng thấy được sản phẩm đầu ra của phần mềm là gì? Dữ liệu yêu cầu của đầu vào là gì?
 - **Chức năng và hiệu suất:** chức năng nào của phần mềm thực hiện biến đổi dữ liệu đầu vào thành đầu ra? Tất cả trường hợp riêng đặc biệt đều được giải quyết?
- Phạm vi dự án phần mềm phải rõ ràng và dễ hiểu tại mỗi mức quản lý và chuyên môn

1. Khái niệm chung- yếu tố sản phẩm

Phân tách vấn đề

- Đôi khi được gọi là **sự phân hoạch** hoặc **giải thích vấn đề**
- Khi phạm vi được xác định...
 - Nó được phân rã thành các chức năng cấu thành
 - Nó được phân rã thành đối tượng dữ liệu người dùng có thể thấy
 - Nó được chia ra thành một tập các lớp vấn đề
- Quá trình phân hoạch tiếp tục cho đến khi tất cả các chức năng hoặc các lớp vấn đề đã được xác định

Nội dung

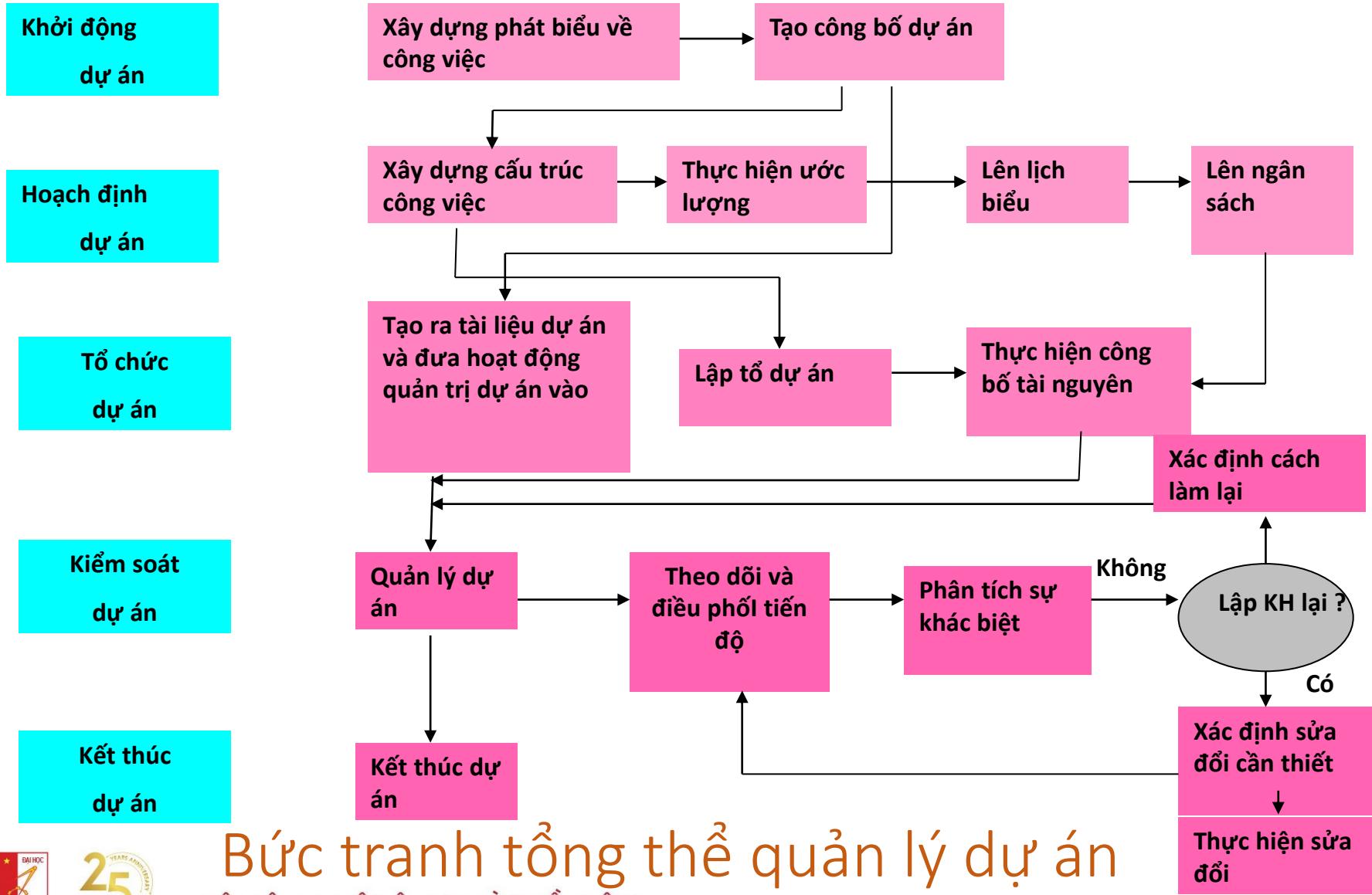
1. Khái niệm chung

- a. Giới thiệu – định nghĩa
- b. Yếu tố con người
- c. Yếu tố sản phẩm

2. Quy trình quản lý dự án

- a. **Tổng quan**
- b. Ước lượng dự án
- c. Lập kế hoạch dự án
- d. Quản lý rủi ro dự án

2. Quy trình quản lý dự án - **Tổng quan**



2. Quy trình quản lý dự án - **Tổng quan**

Giải quyết bài toán quản lý dự án

ĐẦU VÀO TIẾN TRÌNH (Đặc trưng)

Thông tin

Công nghệ

Các công cụ về chất lượng và số lượng

Đầu ra của các tiến trình khác

Đầu vào vật chất

Tương tác các bên liên quan

Các yêu cầu, chỉ dẫn

Chất lượng đầu vào của quá trình, Kiến thức, Năng lực, Kinh nghiệm, Sự rõ ràng, Khả năng, Truyền thông, Hợp tác, Phối hợp

Tính thuần thực của quá trình, Phương pháp luận, Đánh giá và tối ưu, Ràng buộc, khuôn mẫu, cơ sở hạ tầng, Nền tảng văn hoá và chính sách

ĐẦU RA TIẾN TRÌNH (Ví dụ chọn lọc)

Project Business Case

Danh mục đầu tư dự án tối ưu

Báo cáo khả thi dự án

Quy hoạch tổng thể dự án (kế hoạch lệ thuộc)

Yêu cầu thay đổi của khách hàng

Chi phí sửa đổi và lịch trình cơ bản

Báo cáo trạng thái dự án

CÁC BƯỚC TIẾN TRÌNH

1

2

3

N

Chuyển đổi đầu vào thành đầu ra



Thời gian & Giá thành



Hiệu quả của tiến trình

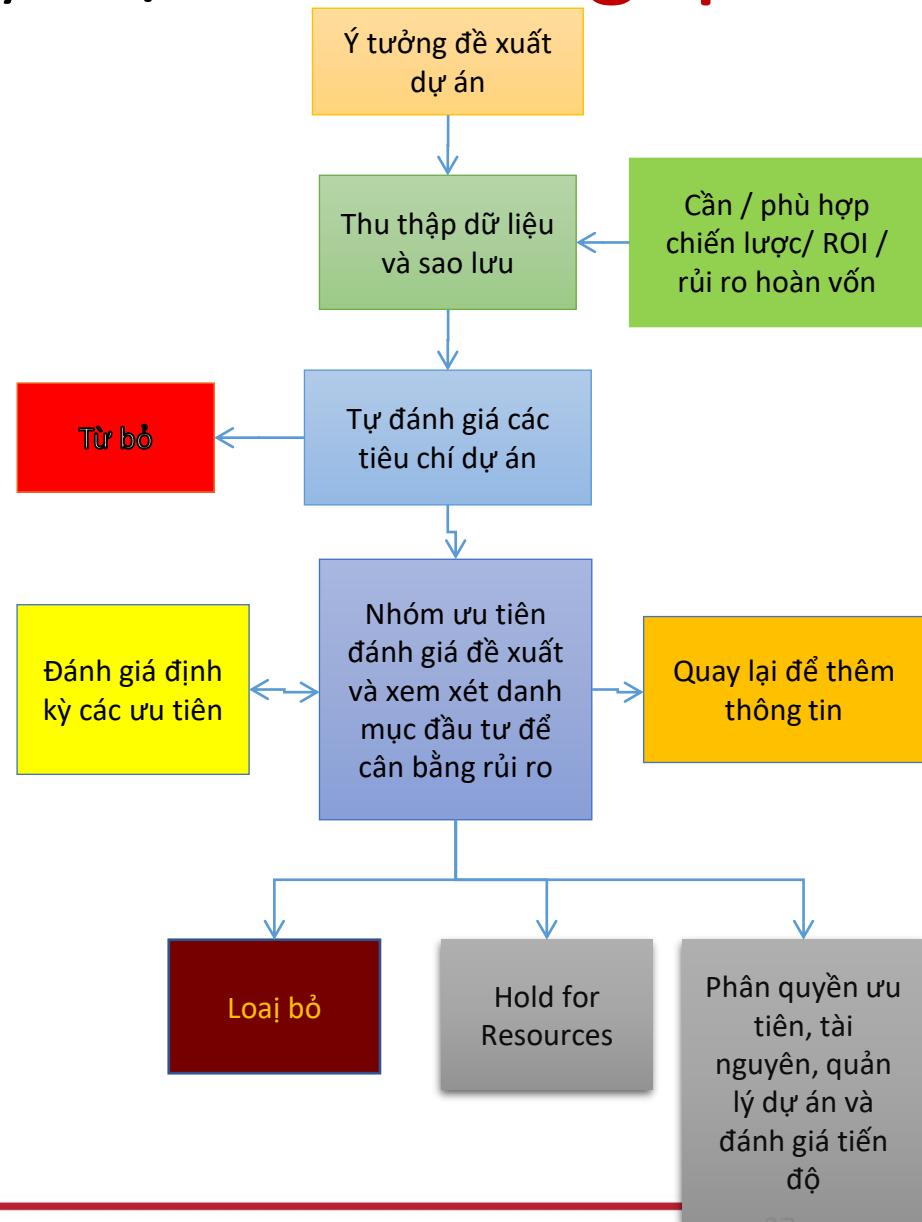
Do kết quả của một quá trình quản lý dự án thường là đầu vào của một quá trình khác, sai sót trong một hoặc nhiều quá trình sẽ kéo theo sai sót trên toàn chuỗi toàn bộ quá trình

Quản lý dự án sử dụng rộng rãi các quy trình để sản xuất "sản phẩm" (xem mẫu ở trên). Một số quy trình khá phức tạp và có nguy cơ lỗi cao.

2. Quy trình quản lý dự án - **Tổng quan**

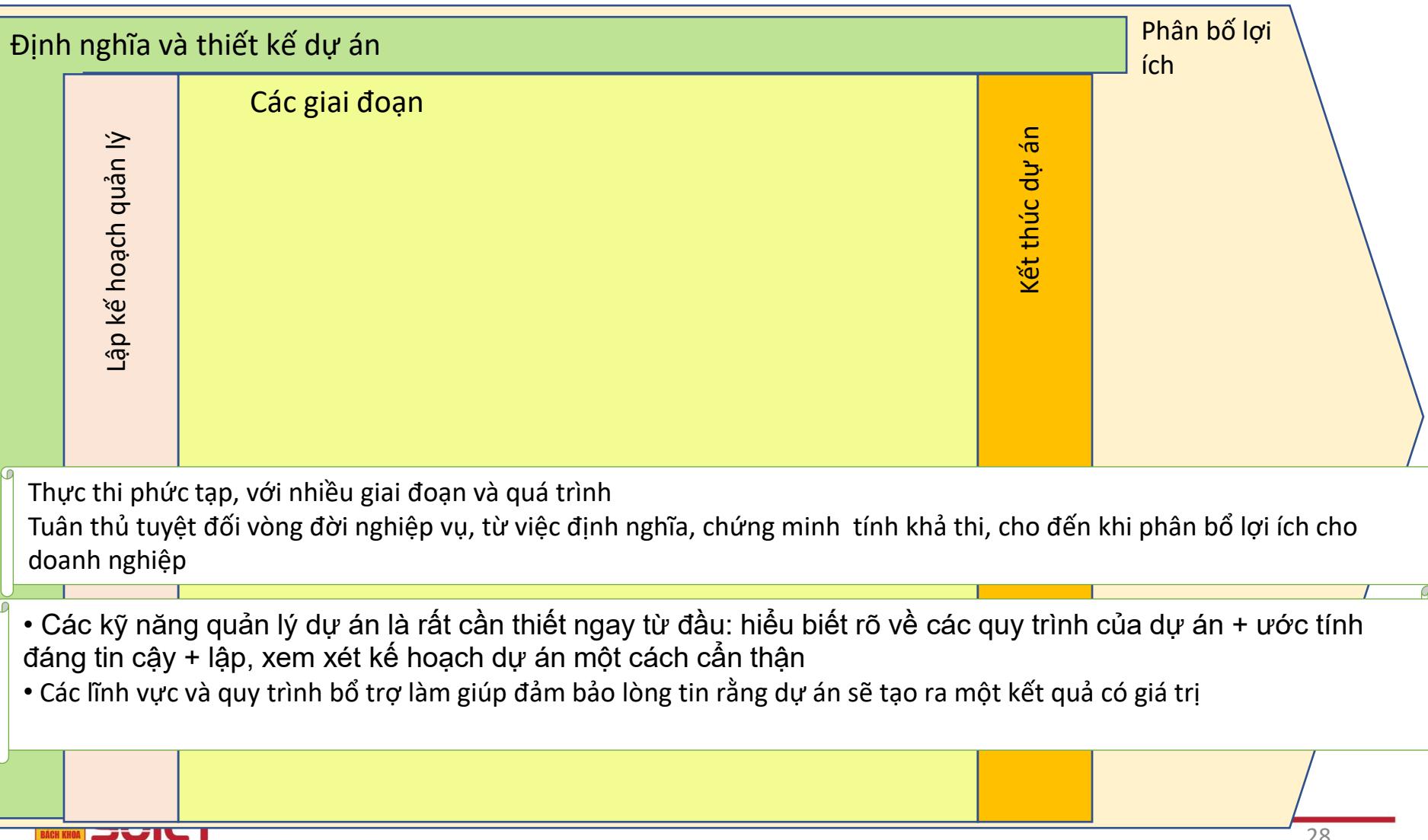
Sàng lọc dự án

- Tập trung vào sự đa dạng của các thành viên trong nhóm và độ phức tạp của công việc:
 - Xem xét các sự việc khác nhau ở các góc độ khác nhau, xuất phát từ thành viên và các công việc cần làm
 - Sử dụng quy trình “Plan - Do - Check - Act”
 - Người quản lý dự án giỏi phải tìm ra các năng lực tiềm ẩn của từng thành viên và sử dụng đầy đủ các năng lực đó.



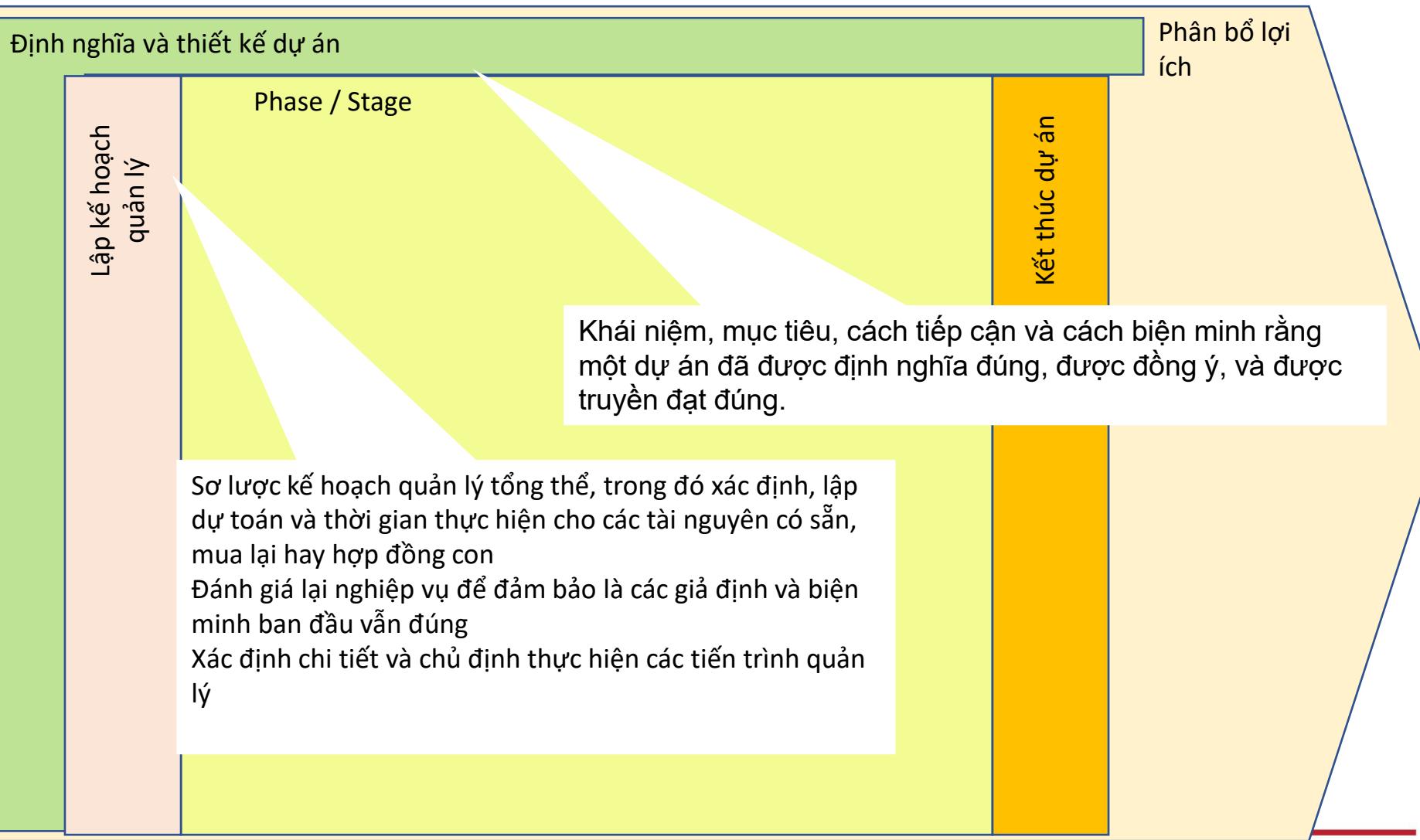
2. Quy trình quản lý dự án - **Tổng quan**

Các pha quản lý dự án (1)



2. Quy trình quản lý dự án - **Tổng quan**

Các pha quản lý dự án (2)



2. Quy trình quản lý dự án - **Tổng quan**

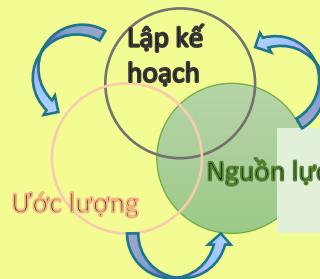
Các pha quản lý dự án (3)

Định nghĩa và thiết kế dự án

Phân bổ lợi ích

Lập kế hoạch quản lý

Pha / Giai đoạn



Sự huy động

Báo cáo kiểm soát quản lý

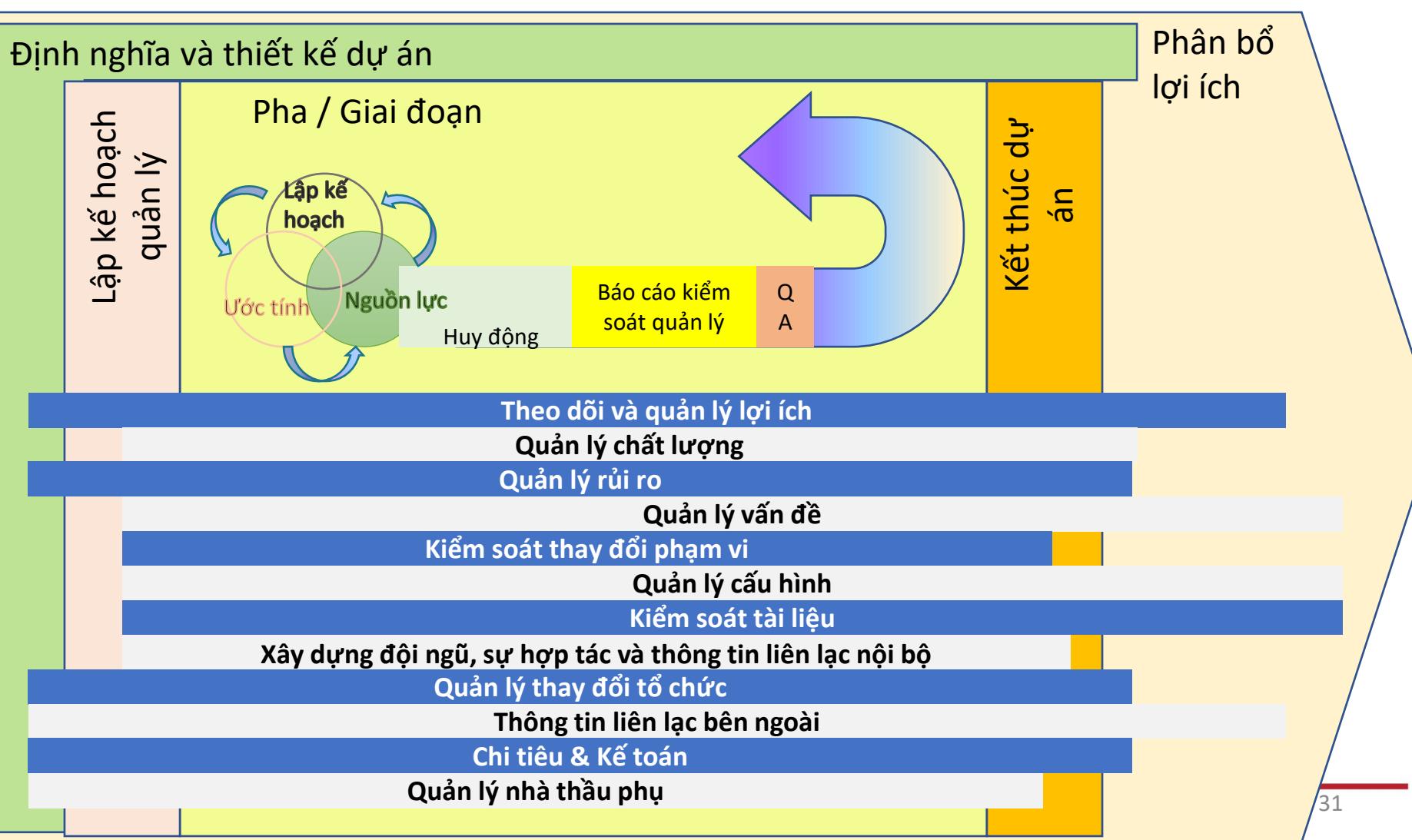
Q A

Kết thúc dự án

Một dự án có thể trải qua nhiều giai đoạn, mỗi giai đoạn có mục tiêu và kết quả cần đạt khác nhau. Các giai đoạn thường yêu cầu các kỹ năng, cấu trúc và mức độ tài nguyên khác nhau. Việc lập kế hoạch, ước lượng chi phí và phân bổ tài nguyên riêng cho từng giai đoạn là bình thường.

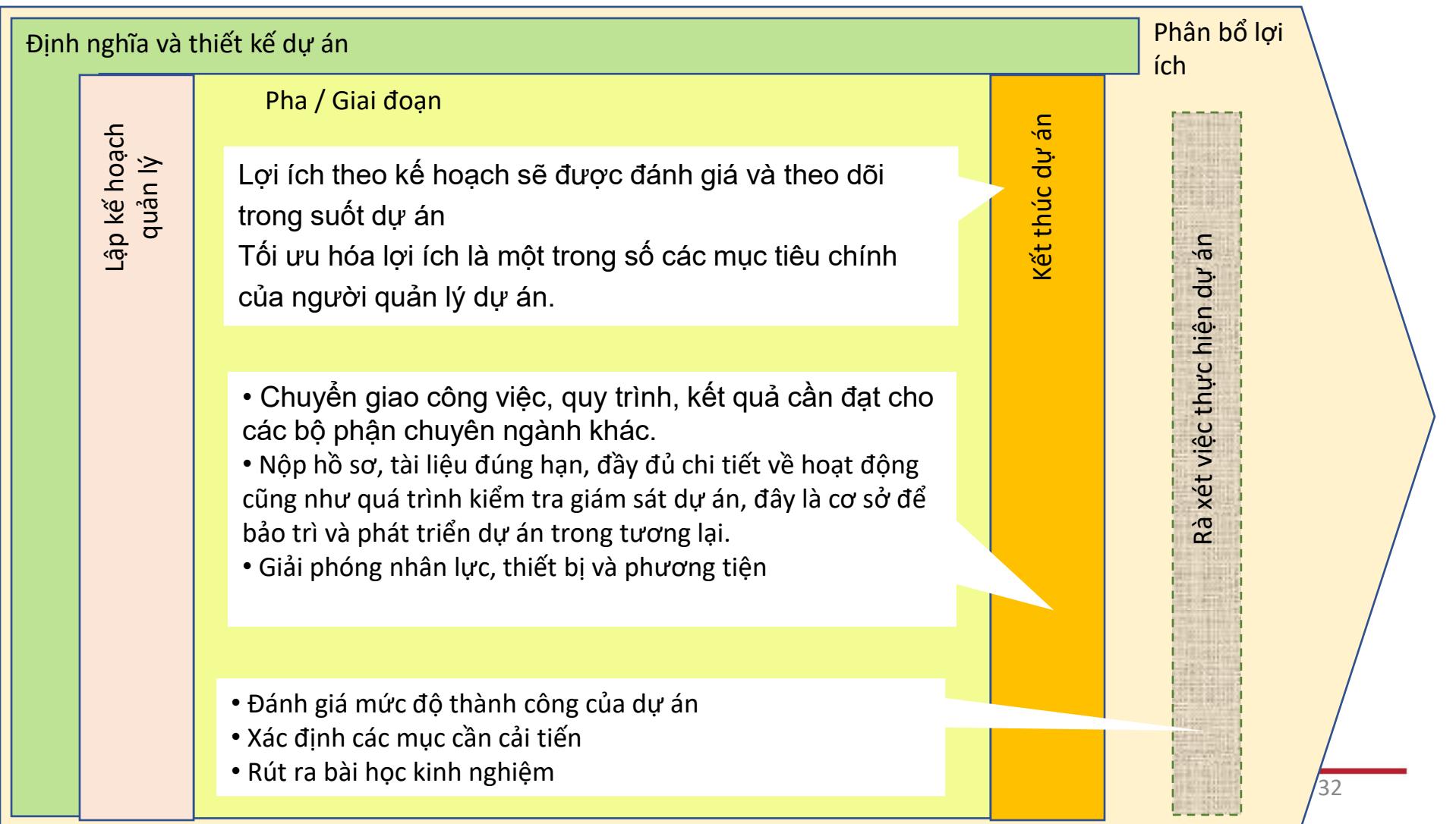
2. Quy trình quản lý dự án - **Tổng quan**

Các pha quản lý dự án (4)



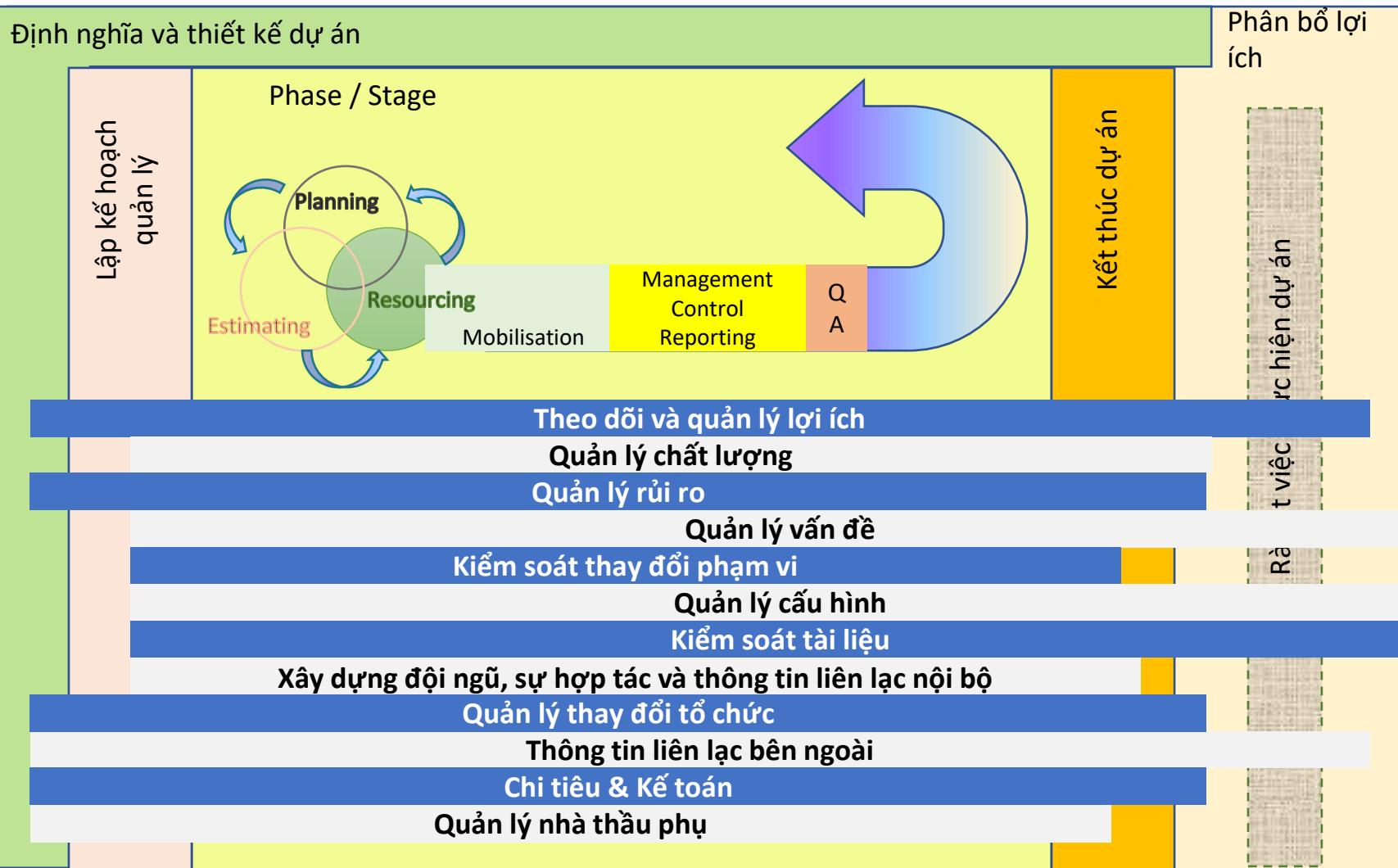
2. Quy trình quản lý dự án - **Tổng quan**

Các pha quản lý dự án (5)



2. Quy trình quản lý dự án - **Tổng quan**

Các pha quản lý dự án (6)



Nội dung

1. Khái niệm chung

- a. Giới thiệu – định nghĩa
- b. Yếu tố con người
- c. Yếu tố sản phẩm

2. Quy trình quản lý dự án

- a. Tổng quan
- b. Ước lượng dự án**
- c. Lập kế hoạch dự án
- d. Quản lý rủi ro dự án

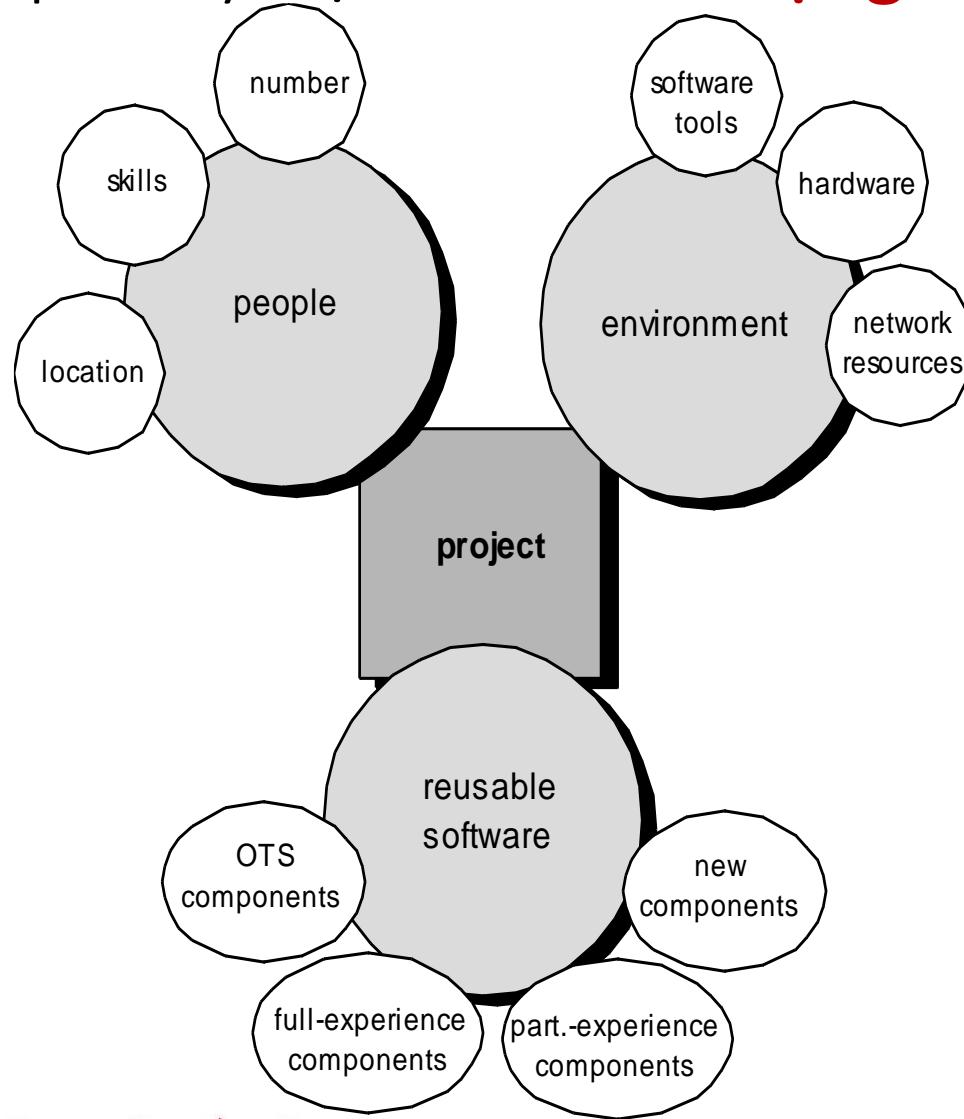
2. Quy trình quản lý dự án – **Ước lượng dự án**

Định nghĩa

- Việc ước lượng các nguồn lực, chi phí và lịch trình cho một nỗ lực kỹ thuật phần mềm yêu cầu
 - Kinh nghiệm
 - Khả năng tiếp cận tốt với thông tin lịch sử(các số liệu)
 - Đủ can đảm để cam kết với dự đoán định lượng khi chỉ tồn tại các thông tin định.
- Ước lượng mang nguy cơ tiềm tàng và rủi ro này dẫn đến sự không chắc chắn.

2. Quy trình quản lý dự án – **Ước lượng dự án**

Các nguồn lực



2. Quy trình quản lý dự án – **Ước lượng dự án**

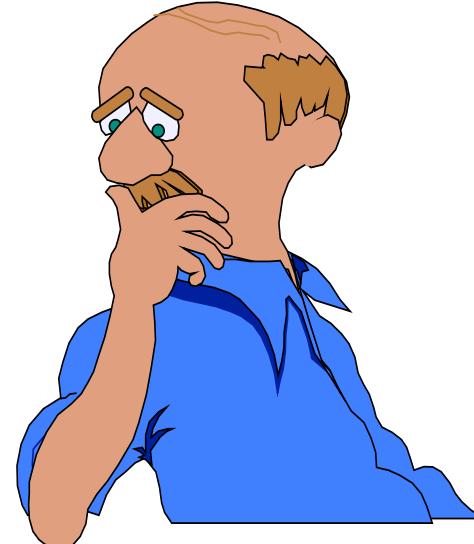
Phạm vi dự án

- Phạm vi phần mềm được mô tả là:
 - Các tính năng sẽ được chuyển giao cho người dùng đầu cuối.
 - Các dữ liệu là input và output
 - Các “nội dung” được trình bày cho người sử dụng như một hệ quả của việc sử dụng phần mềm.
 - Hiệu quả hoạt động, ràng buộc, giao diện, và độ tin cậy gắn với hệ thống.
- Phạm vi được mô tả bằng một trong 2 kĩ thuật:
 - Một mô tả tường thuật của mô mềm được phát triển sau khi giao tiếp với tất cả các bên liên quan..
 - Một tập hợp các use-case sử dụng được phát triển bởi người dùng đầu cuối.

2. Quy trình quản lý dự án – **Ước lượng dự án**

Các kĩ thuật ước lượng

- Kinh nghiệm từ các dự án tương tự
- Các kĩ thuật thông thường
 - Phân tích công việc and ước tính công sức.
 - Mô hình thực nghiệm
 - Đánh giá kích thước(vd: FP)
- Các công cụ tự động



2. Quy trình quản lý dự án – **Ước lượng dự án**

Độ chính xác ước lượng

- Dự đoán dựa trên...
 - mức độ mà các kế hoạch đã ước tính đúng đắn các kích thước của sản phẩm được xây dựng
 - khả năng chuyển đổi các ước tính kích thước vào công sức con người, thời gian, và tiền của(một chức năng có được nhờ sự sẵn có của số liệu phần mềm đáng tin cậy từ các dự án trước)
 - mức độ mà các kế hoạch dự án phản ánh khả năng của nhóm phần mềm
 - sự ổn định của yêu cầu sản phẩm và môi trường hỗ trợ kỹ thuật phần mềm.

Nội dung

1. Khái niệm chung

- a. Giới thiệu – định nghĩa
- b. Yếu tố con người
- c. Yếu tố sản phẩm

2. Quy trình quản lý dự án

- a. Tổng quan
- b. Ước lượng dự án
- c. Lập kế hoạch dự án**
- d. Quản lý rủi ro dự án

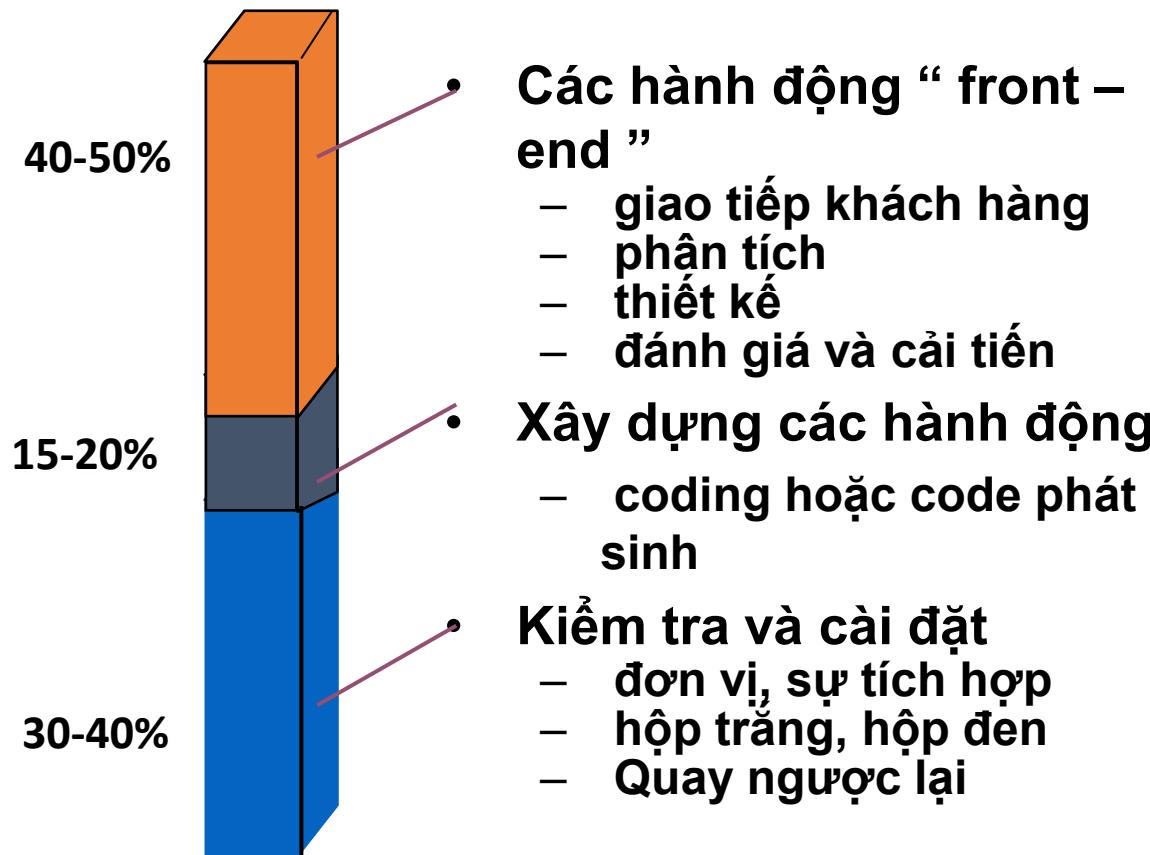
2. Quy trình quản lý dự án – **Lập kế hoạch dự án**

Các nguyên tắc lên kế hoạch

- **Sự phân chia** – Xác định các nhiệm vụ khác biệt
- **Tính phụ thuộc** – Chỉ ra các mối quan hệ tương tác
- **Xác định nhân lực** – Chắc chắn các nguồn lực có sẵn
- **Xác định trách nhiệm** – Mọi người phải được chỉ định
- **Xác định đầu ra** – Mỗi nhiệm vụ phải có đầu ra
- **Xác định mốc thời gian** – đánh giá cho chất lượng

2. Quy trình quản lý dự án – **Lập kế hoạch dự án**

Phân phối nhân lực



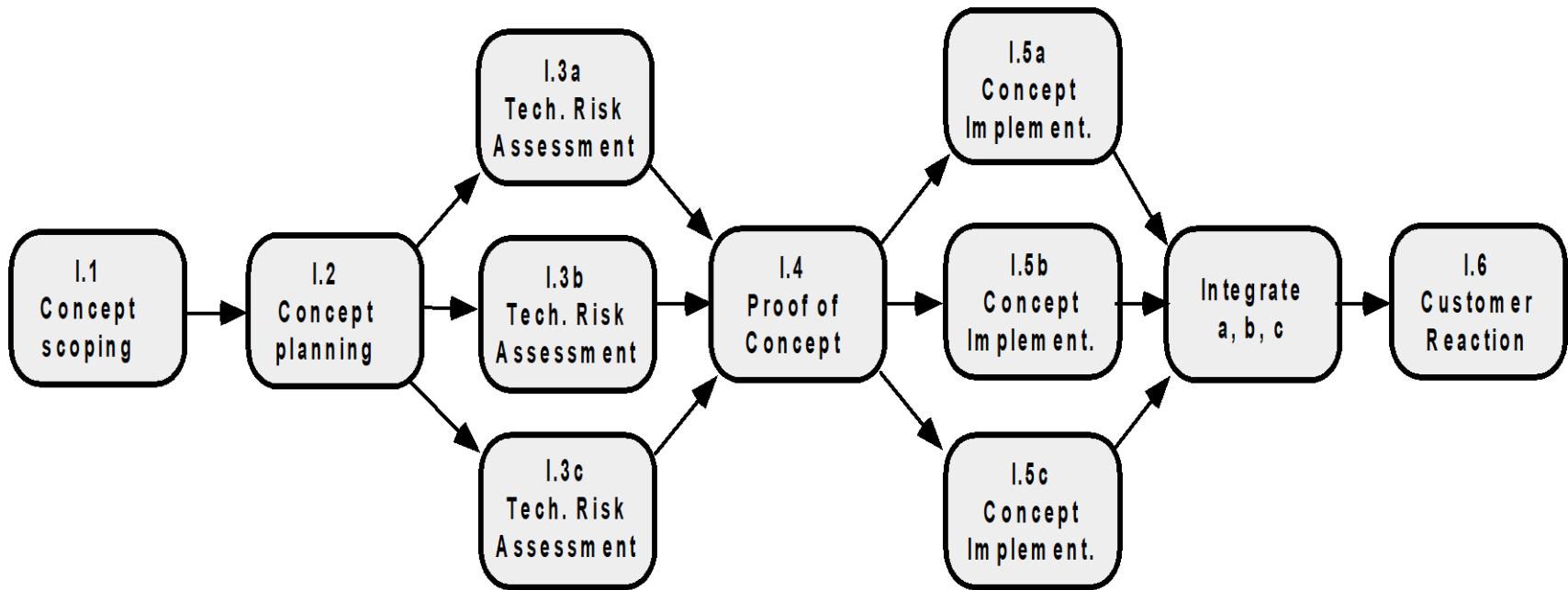
2. Quy trình quản lý dự án – **Lập kế hoạch dự án**

Xác định các nhiệm vụ

- Xác định loại project
- Đánh giá mức độ yêu cầu
- Xác định các tiêu chí tương ứng
- Chọn phần mềm thích hợp và các nhiệm vụ kĩ thuật

2. Quy trình quản lý dự án – Lập kế hoạch dự án

Xác định một mạng lưới các nhiệm vụ



Three L3 tasks are applied in parallel to 3 different concept functions

Three L3 tasks are applied in parallel to 3 different concept functions

2. Quy trình quản lý dự án – Lập kế hoạch dự án

Biểu đồ thời gian

Nhiệm vụ	Tuần 1	Tuần 2	Tuần 3	Tuần 4		Tuần n
Task 1	■					
Task 2		■				
Task 3						
Task 4			■			
Task 5			■			
Task 6		■				
Task 7				■		
Task 8					■	
Task 9			■			
Task 10					■	
Task 11						
Task 12		■				

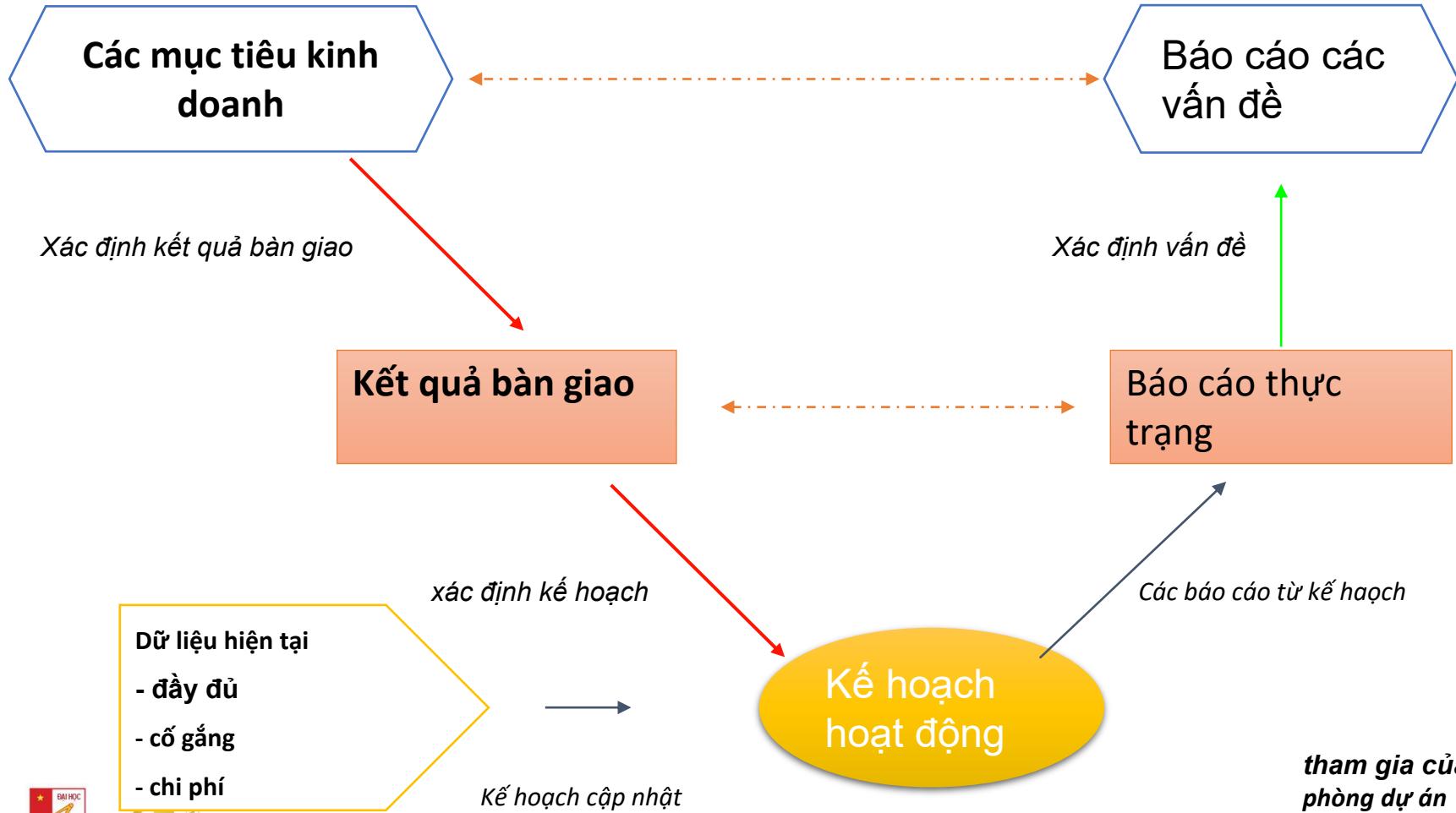
2. Quy trình quản lý dự án – **Lập kế hoạch dự án**

Theo dõi lịch trình thực hiện

- Tiến hành các cuộc họp định kì trong dự án trong đó mỗi thành viên trong nhóm báo cáo tiến độ và các vấn đề.
- Đánh giá kết quả của tất cả các báo cáo trong suốt quá trình xây dựng phần mềm.
- Xác định các mốc quan trọng của Project (Figure 27.3) đã được hoàn thành đúng tiến độ chưa.
- So sánh ngày bắt đầu thực tế để lên kế hoạch cho mỗi nhiệm vụ được sắp xếp trong bảng tài nguyên (Figure 27.4).
- Gặp trực tiếp những người thực hiện để có đánh giá chủ quan của họ về tiến độ và các vấn đề.
- Sử dụng giá trị thu được, phân tích để đánh giá về tiến độ.

2. Quy trình quản lý dự án – Lập kế hoạch dự án

Lập kế hoạch, theo dõi, báo cáo



Nội dung

1. Khái niệm chung

- a. Giới thiệu – định nghĩa
- b. Yếu tố con người
- c. Yếu tố sản phẩm

2. Quy trình quản lý dự án

- a. Tổng quan
- b. Ước lượng dự án
- c. Lập kế hoạch dự án
- d. Quản lý rủi ro dự án**

2. Quy trình quản lý dự án – Quản lý rủi ro

Giới thiệu

• Rủi ro là gì ?

- Những sự kiện có thể làm phá vỡ một dự án
- Những điều không chắc chắn, những khoản nợ hay những điểm yếu có thể làm cho dự án không đi theo đúng kế hoạch đã định

Có thể quản lý
được rủi ro

• Tại sao cần quản lý rủi ro ?

- Tất cả các dự án đều phụ thuộc vào rủi ro
- Tiến trình sẽ không đúng theo kế hoạch trong một số giai đoạn của dự án

Không thể loại trừ
hết rủi ro

• Khi nào cần quản lý rủi ro ?

- Khi lập kế hoạch quản lý
- Khi dự án sẵn sàng thực thi
- Khi khôi phục một dự án đã bỏ dở
- Khi rà xét dự án
- Khi có sự sai lệch lớn so với kế hoạch xảy ra

- Giảm thiểu ảnh hưởng của các sự cố không biết trước cho dự án
- Nâng cao xác suất thực hiện thành công dự án
- Tạo ra ý thức kiểm soát
- Có được các giải pháp hiệu quả và kịp thời

2. Quy trình quản lý dự án – Quản lý rủi ro

Xác định rủi ro

- **Kích thước sản phẩm** - rủi ro gắn liền với kích thước tổng thể của phần mềm được xây dựng hoặc chỉnh sửa.
- **Tác động kinh doanh** - rủi ro gắn liền với sự cưỡng ép bị áp đặt bởi quản lý hoặc các thị trường.
- **Đặc điểm của khách hàng** - rủi ro gắn liền với trình độ sử dụng của khách hàng và khả năng của nhà phát triển trong việc giao tiếp với khách hàng một cách kịp thời.
- **Định nghĩa quá trình** - rủi ro gắn liền với mức độ mà các quá trình phần mềm đã được định nghĩa và được theo dõi bởi tổ chức phát triển.
- **Môi trường phát triển** - rủi ro gắn liền với sự tiện lợi và chất lượng của các công cụ để xây dựng sản phẩm.
- **Công nghệ sử dụng** - rủi ro gắn liền với sự phức tạp của hệ thống và "sự mới mẻ" của công nghệ được đóng gói trong hệ thống.
- **Số lượng nhân viên và kinh nghiệm** - rủi ro gắn liền với kinh nghiệm kỹ thuật và kinh nghiệm trong dự án của các kỹ sư phần mềm.

2. Quy trình quản lý dự án – Quản lý rủi ro

Đánh giá rủi ro dự án (1)

- Người quản lý phần mềm và khách hàng cấp cao có chính thức cam kết hỗ trợ dự án hay không?
- Người dùng cuối có được cam kết rằng các dự án và hệ thống / sản phẩm sẽ được xây dựng hay không?
- Các yêu cầu có được hiểu cặn kẽ bởi đội ngũ kỹ sư phần mềm và khách hàng của họ?
- Khách hàng có được tham gia hoàn toàn trong việc định ra các yêu cầu?
- Người dùng cuối có những mong muốn mang tính thực tế hay không?

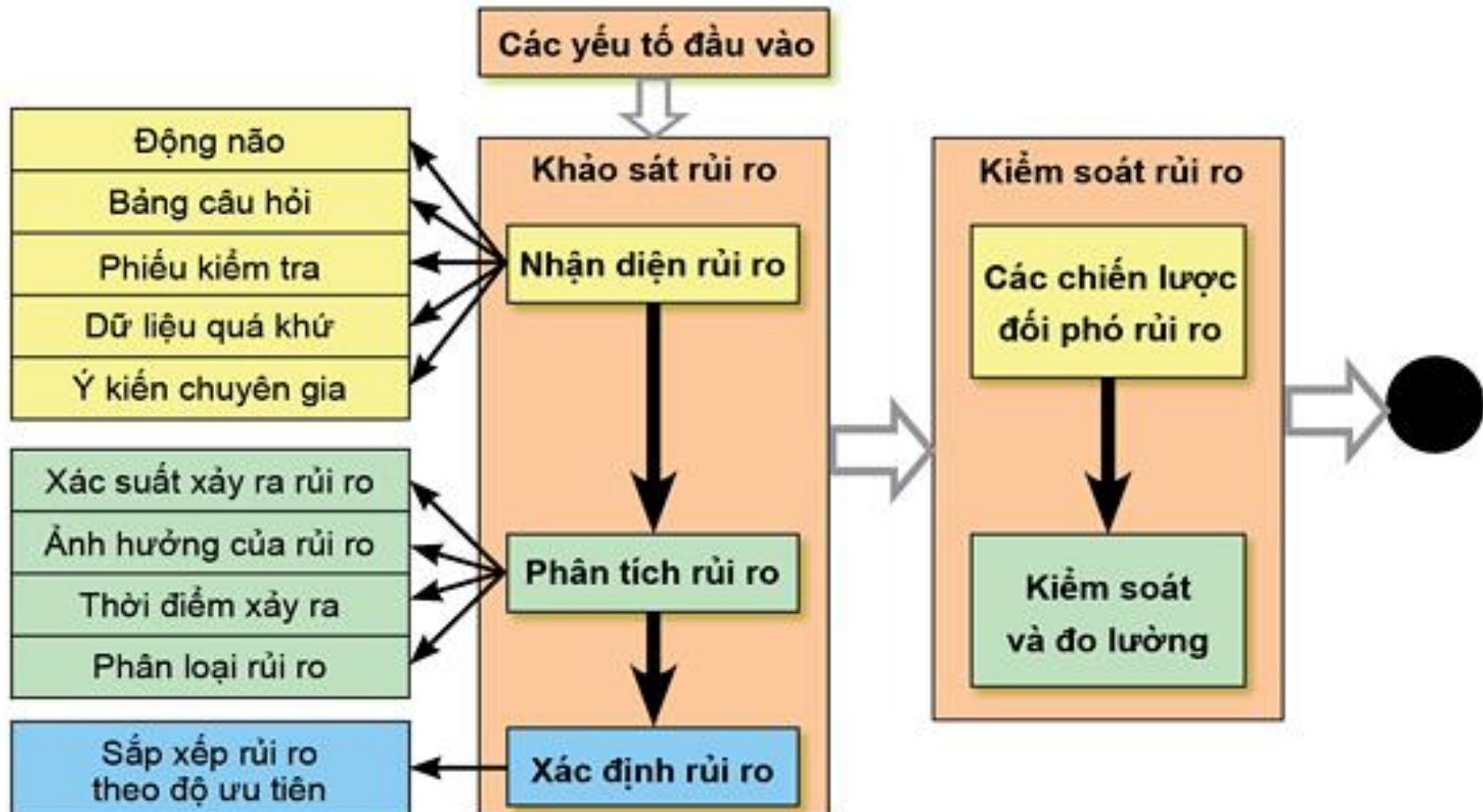
2. Quy trình quản lý dự án – Quản lý rủi ro

Đánh giá rủi ro dự án (2)

- Phạm vi dự án có ổn định hay không?
- Đội ngũ kỹ sư phần mềm có sự hòa hợp thích hợp trong kỹ năng hay không?
- Yêu cầu dự án có ổn định hay không?
- Đội ngũ dự án có kinh nghiệm với công nghệ được thực hiện hay không?
- Số người trong nhóm dự án là đủ để thực hiện công việc?
- Tất cả tập thể cử tri khách hàng/người sử dụng đều đồng ý về tầm quan trọng của dự án và yêu cầu đối với hệ thống/sản phẩm được xây dựng?

2. Quy trình quản lý dự án – Quản lý rủi ro

Các thành phần rủi ro



2. Quy trình quản lý dự án – Quản lý rủi ro

Các thành phần rủi ro

- **Rủi ro hiệu năng** - mức độ không chắc chắn rằng sản phẩm sẽ đáp ứng yêu cầu và phù hợp với mục đích sử dụng của nó.
- **Rủi ro chi phí** - mức độ không chắc chắn rằng ngân sách cho dự án sẽ được duy trì.
- **Rủi ro hỗ trợ** - mức độ không chắc chắn rằng phần mềm có được sẽ có thể dễ dàng sửa chữa, hiệu chỉnh và nâng cao.
- **Rủi ro tiến độ** - mức độ không chắc chắn rằng tiến độ dự án sẽ được duy trì và sản phẩm sẽ được giao đúng hẹn.

2. Quy trình quản lý dự án – Quản lý rủi ro

Dự phòng rủi ro

- Phép chiếu rủi ro, có tên khác là ước lượng rủi ro, đánh giá từng rủi ro bằng hai cách:
 - khả năng hoặc xác suất rằng rủi ro xảy ra.
 - hậu quả có thể xảy ra của các vấn đề liên quan với rủi ro .
- Bốn bước dự phòng rủi ro :
 - Thiết lập một phạm vi phản ánh khả năng của một rủi ro mà ta nhận thấy được.
 - Vạch ra các hậu quả của rủi ro
 - Ước tính tác động của rủi ro lên dự án và sản phẩm
 - Ghi nhớ về tính chính xác toàn bộ của dự phòng rủi ro để không có hiểu lầm xảy ra.

2. Quy trình quản lý dự án – Quản lý rủi ro

Xây dựng bảng rủi ro (1)

Tên Rủi ro	Xác suất	Tác động	RMMM
			<p>Giám sát & Quản lý Giảm thiểu Rủi ro</p>

2. Quy trình quản lý dự án – Quản lý rủi ro

Xây dựng bảng rủi ro (2) : Xác suất

- Ước tính **xác suất** xảy ra
- Ước tính **tác động** của dự án trên thang điểm từ 1-5, với:
 - 1 = ít ảnh hưởng đến thành công của dự án
 - 5 = tác động thảm khốc đến thành công của dự án
- sắp xếp bảng theo xác suất và tác động

2. Quy trình quản lý dự án – Quản lý rủi ro

Xây dựng bảng rủi ro (3) : Mức độ/tác động

- Mức độ rủi ro tổng cộng (**risk exposure**), **RE**, được xác định bằng công thức [Hal98]:

$$RE = P \times C$$

Với:

P là xác suất xảy ra đối với 1 rủi ro, và

C là chi phí cho dự án nếu rủi ro xảy ra.

2. Quy trình quản lý dự án – Quản lý rủi ro

Ví dụ về mức độ rủi ro

- Xác định rủi ro. Trên thực tế, chỉ có 70% các thành phần phần mềm được lên kế hoạch để tái sử dụng sẽ được tích hợp vào trong ứng dụng. Các chức năng còn lại sẽ phải được phát triển theo ý khách hàng.
- Xác suất rủi ro. 80% (xấp xỉ).
- Tác động rủi ro. Có 60 thành phần phần mềm tái sử dụng được đã được lên kế hoạch. Nếu chỉ có 70% có thể được sử dụng => 18 thành phần sẽ phải được phát triển từ đầu (bên cạnh đó phần mềm theo ý khách hàng khác mà đã được lên kế hoạch phát triển). Bởi các thành phần trung bình là 100 LOC và dữ liệu địa phương chỉ ra rằng chi phí kỹ thuật phần mềm cho từng LOC là 14,00 \$, chi phí tổng cộng (tác động) để phát triển các thành phần sẽ là $18 \times 100 \times 14 = 25.200 \$$.
- Mức độ rủi ro. $RE = 0.80 \times 25,200 \sim \$20,200$.

2. Quy trình quản lý dự án – Quản lý rủi ro

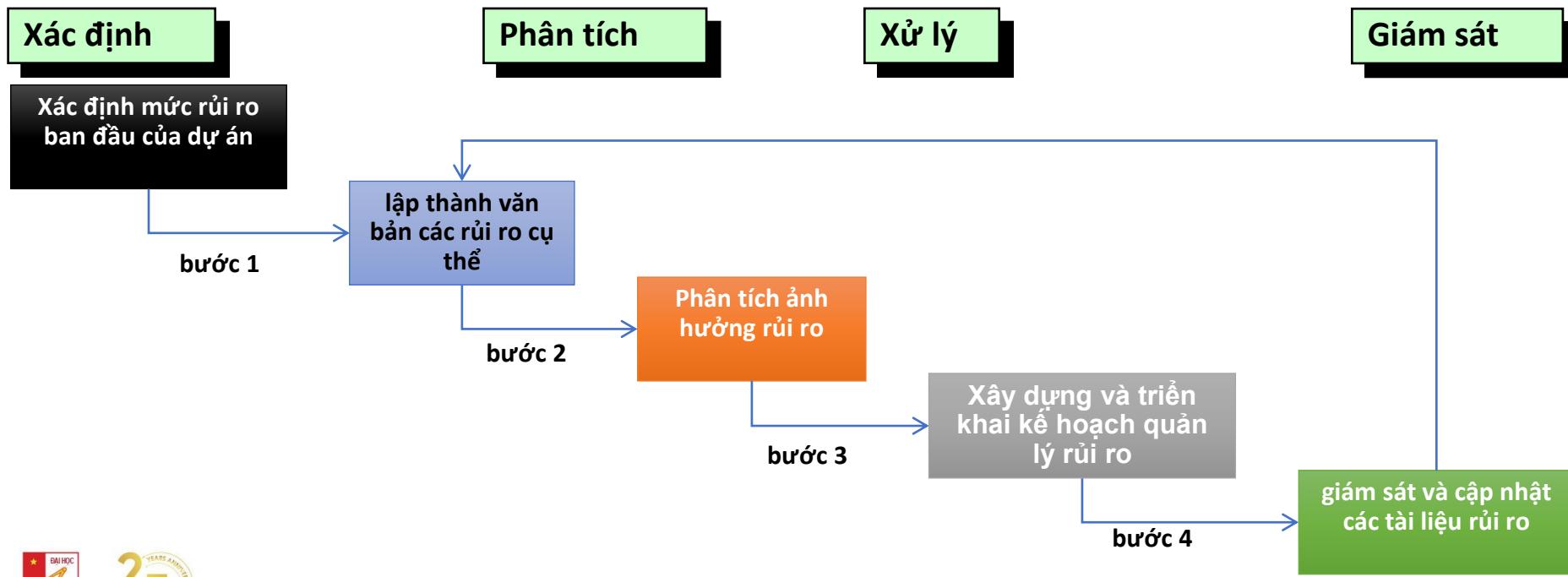
Giảm nhẹ rủi ro, giám sát, quản lý

- **Giảm nhẹ** - Làm thế nào để ngăn ngừa rủi ro ?
- **Giám sát** - Những yếu tố mà ta có thể theo dõi, cho phép ta xác định khả năng của rủi ro tăng lên hay giảm xuống?
- **Quản lý** - Ta có kế hoạch dự phòng gì nếu rủi ro trở thành hiện thực?

2. Quy trình quản lý dự án – Quản lý rủi ro

Quy trình quản lý rủi ro

- Giảm tối thiểu ảnh hưởng của những sự cố không biết trước cho dự án bằng cách xác định và đưa ra những giải pháp tình huống trước khi có những hậu quả xấu xảy ra



2. Quy trình quản lý dự án – **Quản lý rủi ro**

Ví dụ

- Chậm tiến độ xây dựng phần mềm vì các LTV gấp phải nhiều khó khăn trong giai đoạn lập trình hơn dự đoán.
- VỚI tiến độ hiện tại, xác suất các LTV không thể đáp ứng các sự kiện sắp tới đúng hạn là khoảng 30 %.
- Hành động ngăn ngừa có thể gồm:
 - Giảm thiểu rủi ro: đào tạo huấn luyện bổ sung cho các LTV
 - Loại bỏ rủi ro: hợp đồng thuê khoán chuyên môn với các LTV giàu kinh nghiệm

Tổng kết

Sau bài học, sinh viên đã nắm được các kĩ năng:

- Hiểu rõ quá trình QLDA: QLDA phần mềm là hoạt động bao trùm các hoạt động sản xuất phần mềm.
- Hiểu rõ các nhân tố ảnh hưởng QLDA: Nhân tố chính là Con người. Các kỹ thuật khác nhau về giao tiếp và phối hợp được dùng để hỗ trợ công tác nhân sự.
- Hiểu rõ các nghiệp vụ QLDA: nhấn mạnh công tác đánh giá, lượng hóa, kế hoạch và kiểm soát rủi ro.

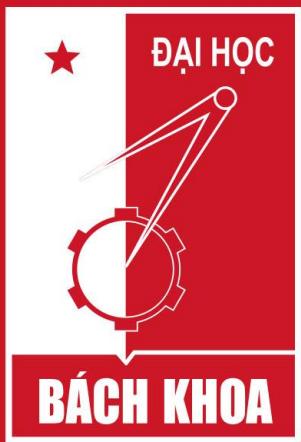


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The graphic consists of the number "25" in large white digits. A curved banner arches over the top of the "2", containing the text "YEARS ANNIVERSARY". Below the "25" is the acronym "SOKT" in a large, bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 5

Quản lý cấu hình phần mềm

Mục tiêu của bài học

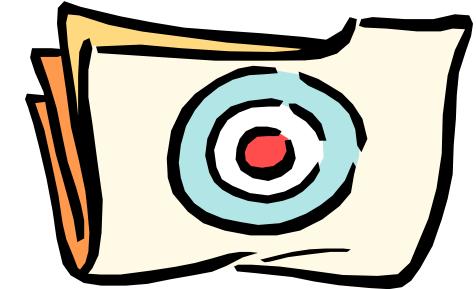
Sinh viên sẽ được trang bị các kiến thức sau:

- Các khái niệm liên quan tới Quản lý cấu hình phần mềm
- Giới thiệu về quy trình Quản lý cấu hình phần mềm
- Một số hoạt động quan trọng như: quản lý phiên bản, quản lý thay đổi

Nội dung

- 1. Khái niệm quản lý cấu hình phần mềm**
2. Quy trình quản lý cấu hình phần mềm
3. Quản lý phiên bản
4. Quản lý thay đổi

1.1 Đặt vấn đề

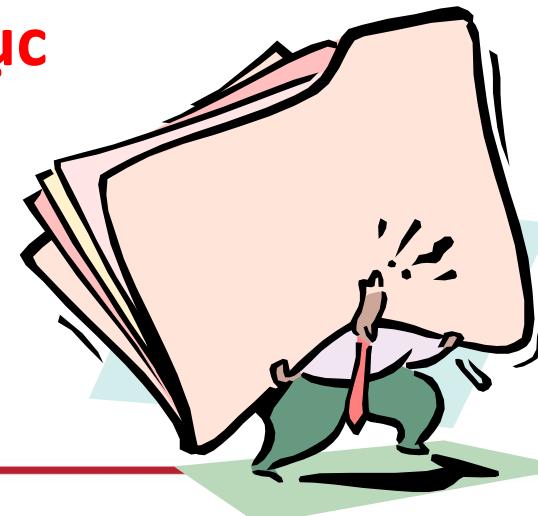


Quá trình phát triển phần mềm

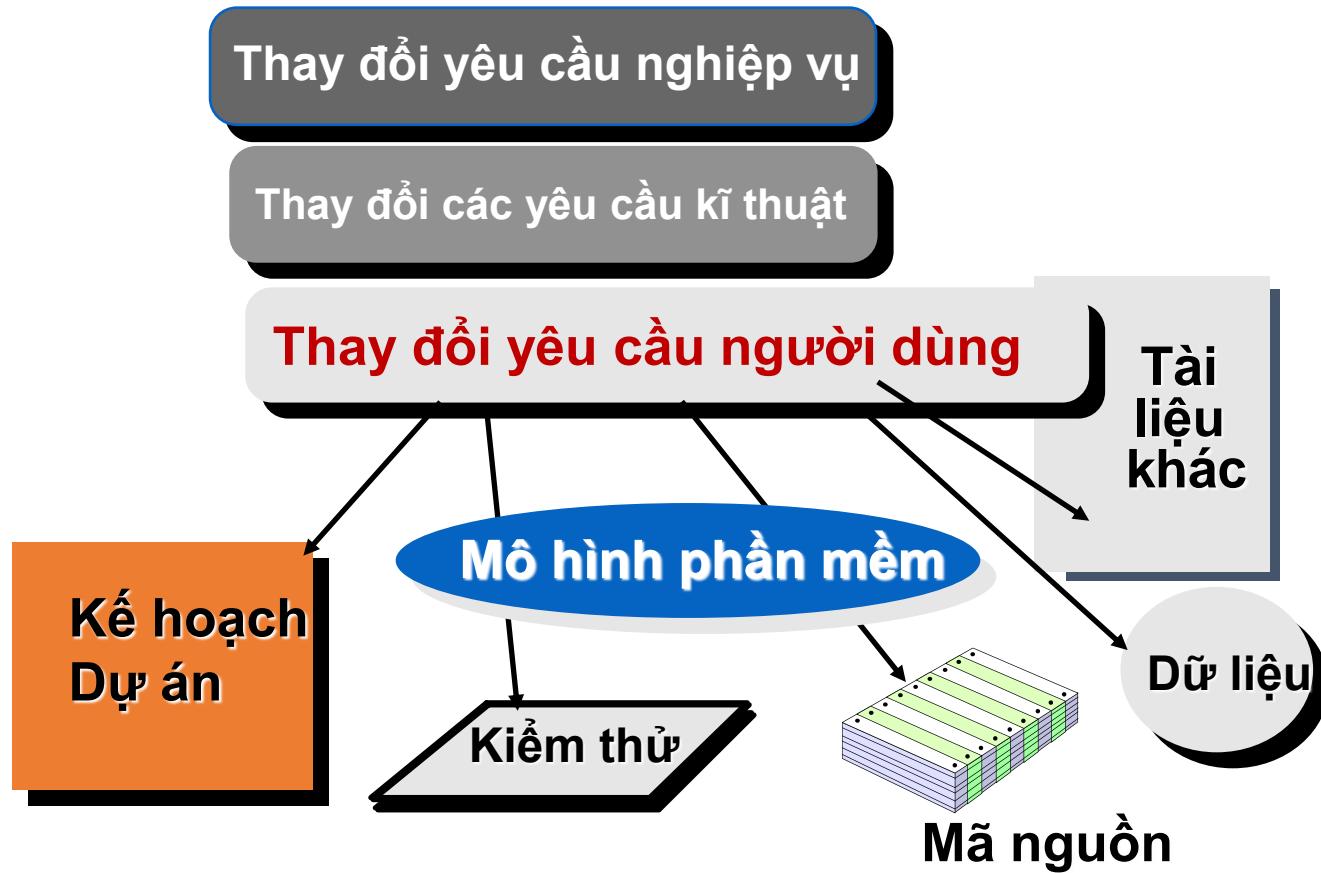
- Lý tưởng:
 - Phần mềm được phát triển từ các yêu cầu ổn định
 - (do việc hướng đến mục tiêu cố định luôn dễ dàng hơn mục tiêu bị thay đổi)
- Thực tế:
 - Các yêu cầu ổn định luôn không tồn tại cho hầu hết các hệ thống thực tế
- Do đó:
 - Một dự án phần mềm hiệu quả cần phải có chiến lược để **giải quyết vấn đề “THAY ĐỔI”**

Sự tiến hóa của phần mềm

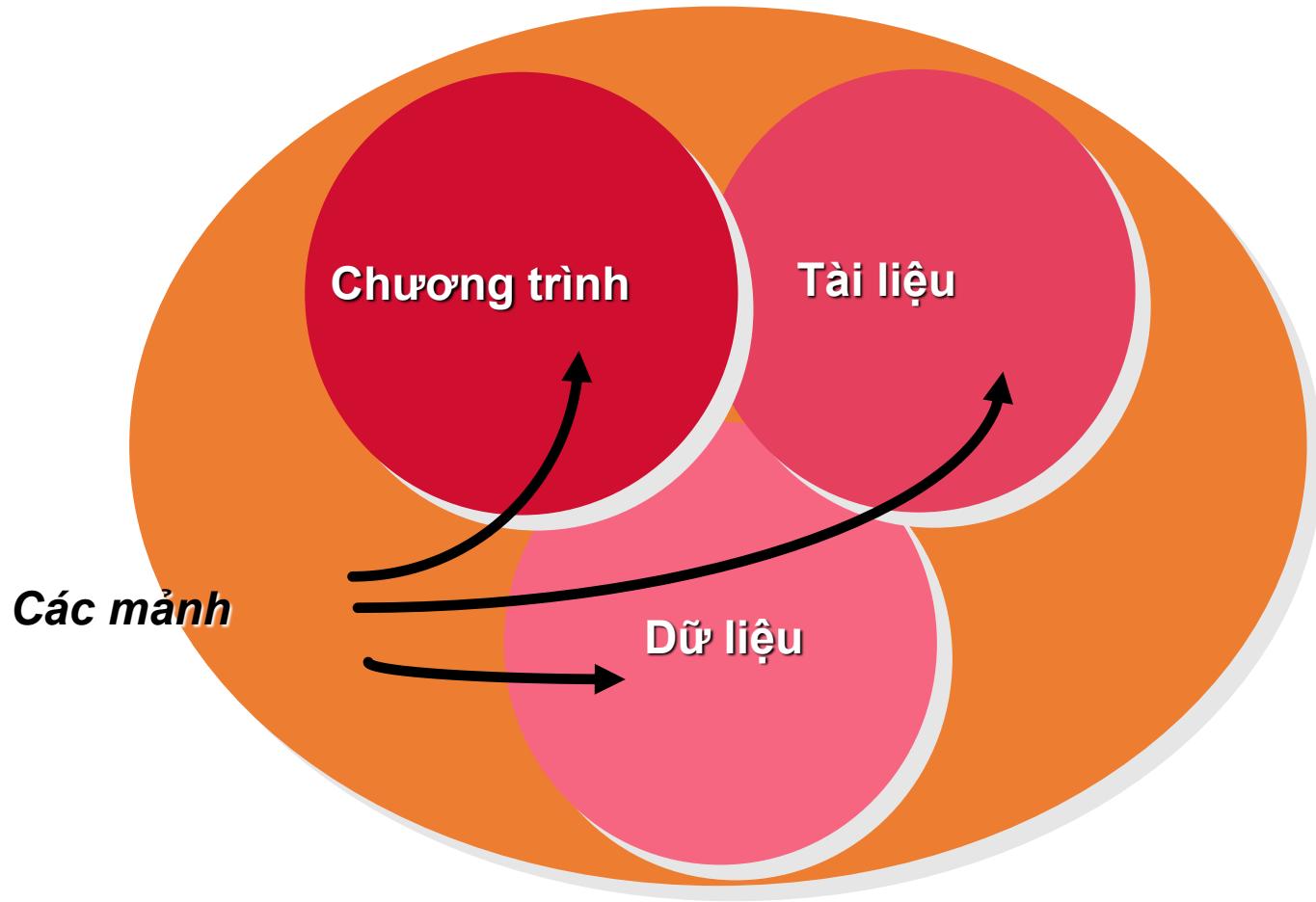
- Phần mềm được phát triển theo thời gian
 - Nhiều yếu tố khác nhau được tạo ra trong suốt thời gian của dự án
 - Có nhiều phiên bản khác nhau
 - Các nhóm làm việc song song để đưa ra sản phẩm cuối cùng
- Hệ thống có thể **thay đổi liên tục**



Những thay đổi là gì?



Cấu hình phần mềm



Vấn đề

- **Nhiều người** phải làm việc trên phần mềm đang thay đổi
- **Nhiều phiên bản** của phần mềm phải được hỗ trợ:
 - Hệ thống đã phát hành
 - Hệ thống được cấu hình tùy chỉnh (các chức năng khác nhau)
 - Hệ thống đang được phát triển
- Phần mềm phải chạy trên **các máy và hệ điều hành khác nhau**

- Do đó cần phải có sự quản lý và phối hợp với nhau
- Quản lý cấu hình phần mềm (Software Configuration Management (SCM))
 - Quản lý các hệ thống phần mềm đang phát triển
 - Kiểm soát chi phí liên quan đến việc thực hiện các thay đổi đối với hệ thống

Thay đổi và Kiểm soát

- Nếu những **thay đổi** không được **kiểm soát** - mọi thứ có thể và sẽ vượt khỏi tầm tay
- Vấn đề quản lý thay đổi thậm chí là cần thiết khi nhiều người cùng làm việc trong một dự án
- Nếu không có các chiến lược và cơ chế thích hợp để kiểm soát các thay đổi - người ta không bao giờ có thể khôi phục về bản sao cũ ổn định hơn của phần mềm
 - Do bởi **mọi thay đổi đều dẫn đến rủi ro**

Câu trả lời

- **Sự thật:**
 - Những thay đổi là không thể tránh khỏi
 - Các thay đổi cần được kiểm soát
 - Các thay đổi cần được quản lý
- **Giải pháp**
 - Quản lý cấu hình phần mềm
Software Configuration Management (SCM)



1.2 Quản lý cấu hình phần mềm

- Mô tả:
 - Quản lý cấu hình phần mềm - Software Configuration Management (SCM) bao gồm các nguyên tắc và kỹ thuật đánh giá và kiểm soát sự thay đổi đối với các sản phẩm phần mềm trong và sau quá trình kỹ thuật phần mềm.
- Các tiêu chuẩn (được ANSI phê duyệt)
 - IEEE 828: Software Configuration Management Plans
 - IEEE 1042: Guide to Software Configuration Management

1.2 Quản lý cấu hình phần mềm (2)

- Áp dụng một cách tiếp cận nghiêm ngặt để đảm bảo
 - Các chi tiết trong hệ thống phần mềm đều được xác định và theo dõi
 - Các thay đổi với các mục khác nhau được ghi lại và theo dõi
 - Tích hợp thích hợp tất cả các mô-đun khác nhau

1.2 Quản lý cấu hình phần mềm (3)

- SCM có thể giúp xác định tác động của thay đổi cũng như kiểm soát sự phát triển
- Nó có thể theo dõi và kiểm soát các thay đổi trong tất cả các khía cạnh của phát triển phần mềm
 - Yêu cầu
 - Thiết kế
 - Mã hóa
 - Kiểm thử
 - Làm tài liệu

Sự cần thiết của SCM...

- Khi phần mềm phát triển - nhiều tài nguyên hệ thống thay đổi
 - SCM ngăn ngừa các lỗi có thể tránh được phát sinh từ các thay đổi xung đột
- Thông thường nhiều phiên bản của phần mềm được phát hành và cần đến sự hỗ trợ
 - SCM cho phép một nhóm hỗ trợ nhiều phiên bản.
 - SCM cho phép các thay đổi trong các phiên bản tuần tự được truyền bá
- SCM cho phép các nhà phát triển theo dõi các thay đổi và khôi phục bất kỳ thay đổi nào để đưa hệ thống phần mềm trở lại trạng thái an toàn đã biết gần đây nhất

1.3 Các khái niệm trong SCM

- Mục cấu hình (Configuration Item - CI) / Mục cấu hình phần mềm (Software Configuration Item - SCI)
- Đường cơ sở (Baseline)
- Kho lưu trữ SCM (SCM Repository)
- Thư mục SCM (SCM Directory)
- Phiên bản, bản sửa đổi và bản phát hành (Version, Revision and Release)

Mục cấu hình (CI)

“Tập hợp phần cứng, phần mềm hoặc cả hai, được chỉ định để quản lý cấu hình và được coi như một thực thể duy nhất trong quy trình quản lý cấu hình”.

“An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process.”

Mục cấu hình (CI) (2)

- Các mục cấu hình phần mềm không chỉ là các đoạn mã chương trình mà là tất cả các loại tài liệu cho sự phát triển phần mềm, ví dụ:
 - các tệp mã
 - trình điều khiển cho các trường hợp kiểm thử
 - tài liệu phân tích hoặc thiết kế
 - tài liệu hướng dẫn người dùng
 - cấu hình hệ thống (ví dụ: phiên bản trình biên dịch được sử dụng)
- ❖ Trong một số hệ thống, không chỉ phần mềm mà còn tồn tại các mục cấu hình phần cứng (CPU, tần số tốc độ bus)!

Tìm kiếm các mục cấu hình

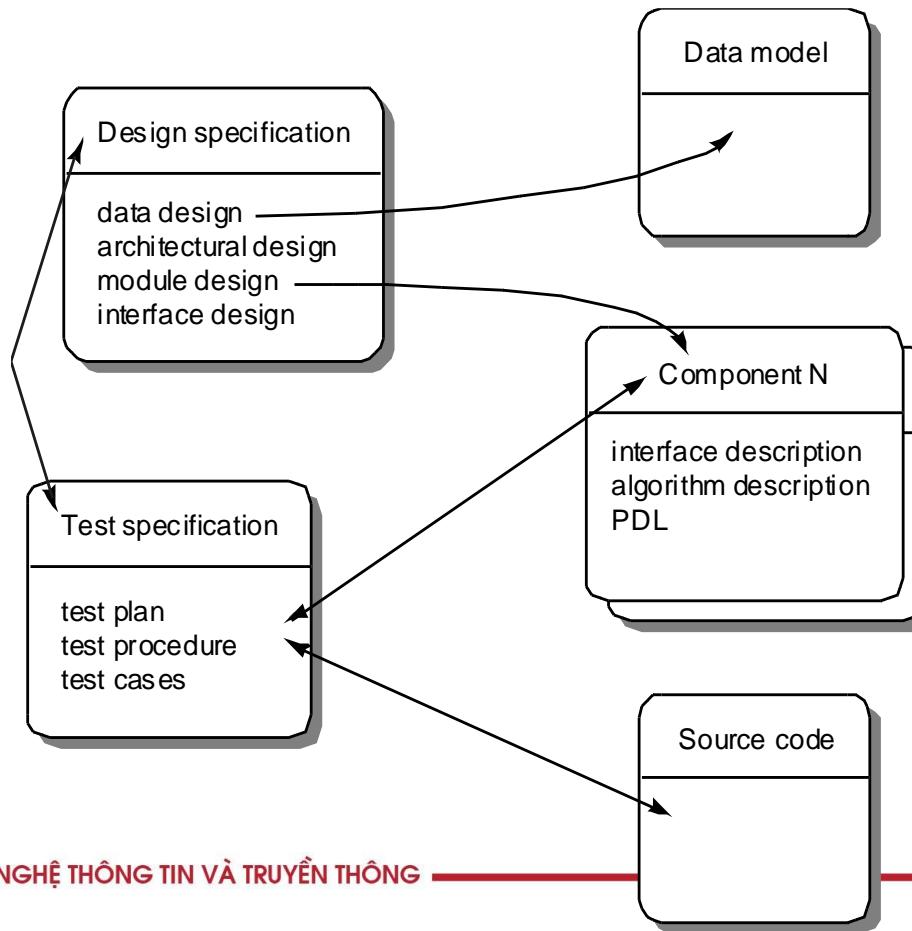
- Các dự án lớn thường tạo ra hàng nghìn thực thể (tệp, tài liệu, ...) phải được **xác định duy nhất**.
- Nhưng không phải tất cả các thực thể đều cần được định cấu hình. Vấn đề:
 - Cái gì: Lựa chọn CI (Nên quản lý những gì?)
 - Khi nào: Khi nào bạn bắt đầu đặt một thực thể dưới sự kiểm soát cấu hình?
- Bắt đầu quá sớm dẫn đến quá sự “áp đặt”
- Bắt đầu quá muộn dẫn đến hỗn loạn

Tìm kiếm các mục cấu hình (2)

- Một số thực thể này phải được duy trì trong suốt thời gian tồn tại của phần mềm. Điều này cũng bao gồm giai đoạn khi phần mềm không còn được phát triển nhưng vẫn được sử dụng bởi khách hàng mong đợi sự hỗ trợ thích hợp trong nhiều năm.
- **Một lược đồ đặt tên** thực thể nên được xác định để định danh cho các tài liệu có tên liên quan.
- Lựa chọn các mục cấu hình phù hợp là một kỹ năng cần thực hành
 - Rất giống với mô hình đối tượng
 - Sử dụng các kỹ thuật tương tự như mô hình hóa đối tượng để tìm các CI

Tìm kiếm các mục cấu hình (3)

- Các đối tượng cấu hình phần mềm



Đường cơ sở (Baseline)

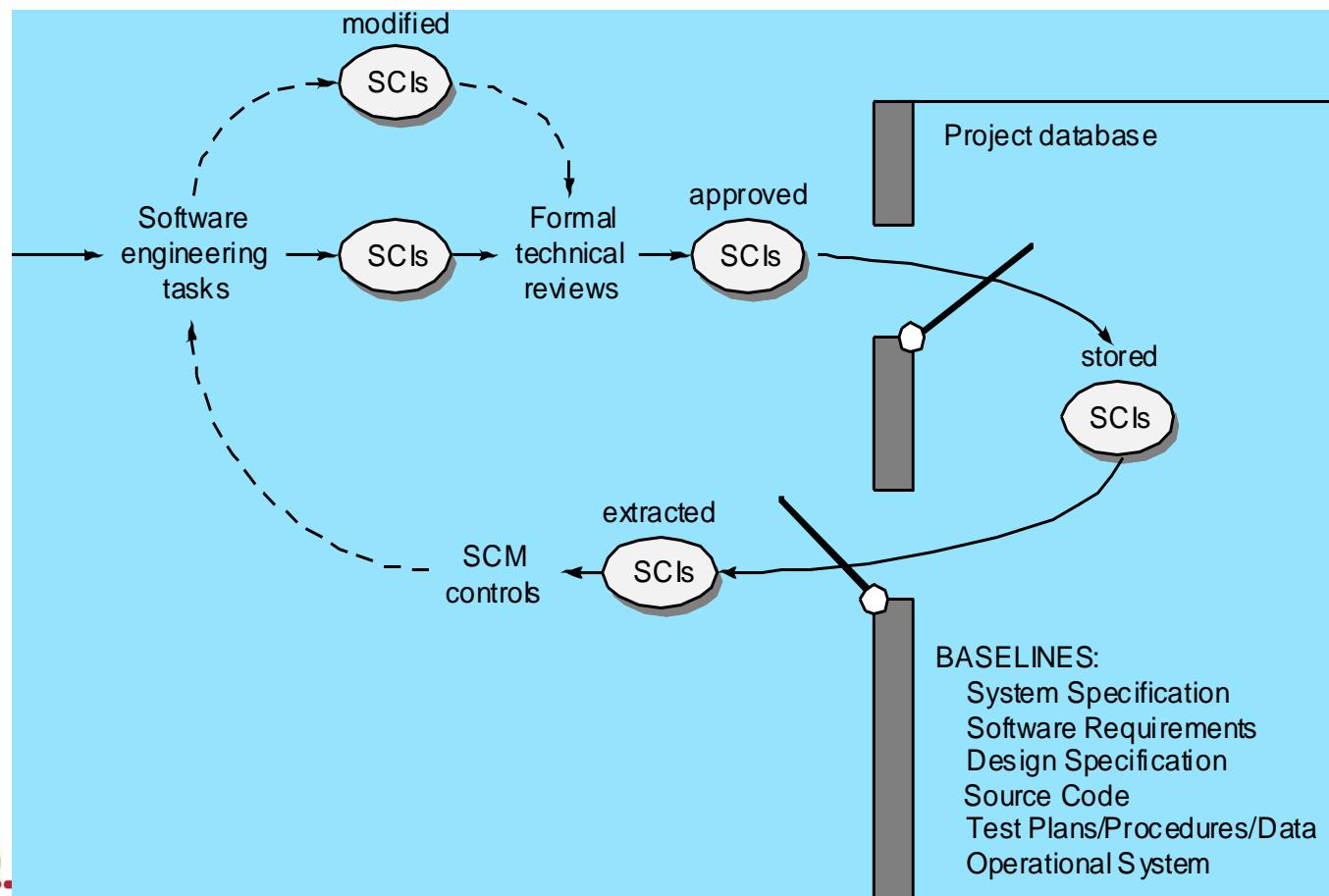
- Định nghĩa:
 - Chuẩn IEEE (IEEE Std. No. 610.12-1990) định nghĩa đường cơ sở (baseline) như sau:
 - Đặc tả kỹ thuật hoặc sản phẩm đã được xem xét và thống nhất chính thức, sau đó được dùng như là một cơ sở để tiếp tục phát triển, và có thể thay đổi chỉ thông qua thủ tục kiểm soát thay đổi chính thức.
 - Một baseline là một mốc quan trọng trong sự phát triển của phần mềm được đánh dấu bằng việc cung cấp một hoặc nhiều mục cấu hình phần mềm và sự chấp thuận của các SCIs - **software configuration items** thu được thông qua đánh giá kỹ thuật chính thức.

Đường cơ sở (Baseline) (2)

- Baseline có thể được định nghĩa ở bất kỳ mức chi tiết nào
- Ví dụ:
 - Baseline A: API của một chương trình được xác định hoàn toàn; phần thân của các phương thức trống.
 - Baseline B: Tất cả các phương thức truy cập dữ liệu được thực hiện và kiểm thử; lập trình GUI có thể bắt đầu.
 - Baseline C: GUI được triển khai, giai đoạn thử nghiệm có thể bắt đầu.

Đường cơ sở (Baseline) (3)

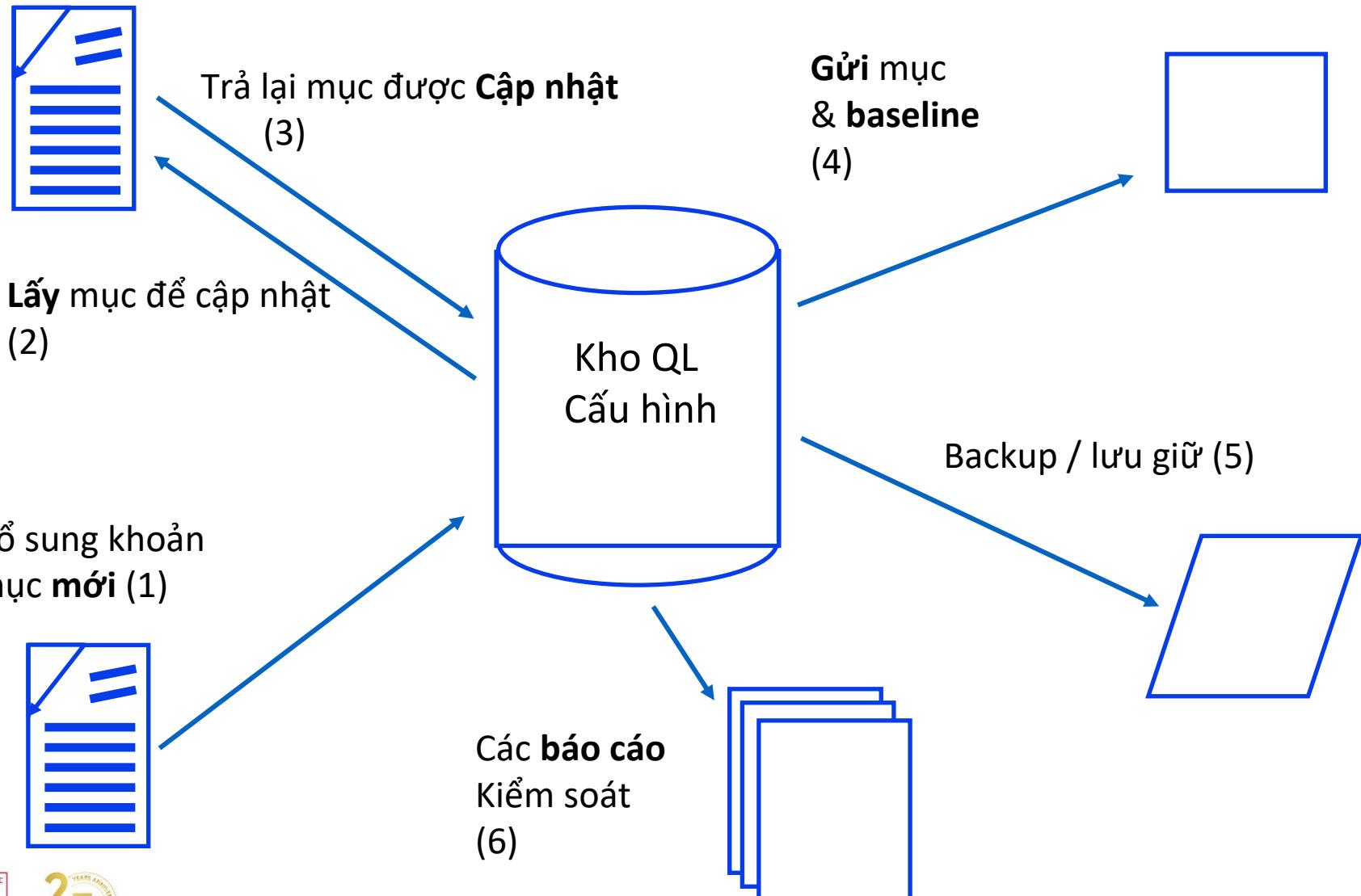
- Các SCI đã xác định đường cơ sở và cơ sở dữ liệu dự án:



Kho lưu trữ SCM (SCM Repository)

- Kho lưu trữ SCM là một tập các cơ chế hoạt động và cấu trúc dữ liệu cho phép một nhóm phát triển phần mềm có thể quản lý thay đổi, phát triển, bảo trì phần mềm một cách hiệu quả.
- Một kho lưu trữ (repository) dữ liệu có các chức năng sau đây:
 - Toàn vẹn dữ liệu
 - Chia sẻ thông tin
 - Tích hợp công cụ
 - Tích hợp dữ liệu
 - Thực thi phương pháp luận
 - Tiêu chuẩn hóa tài liệu

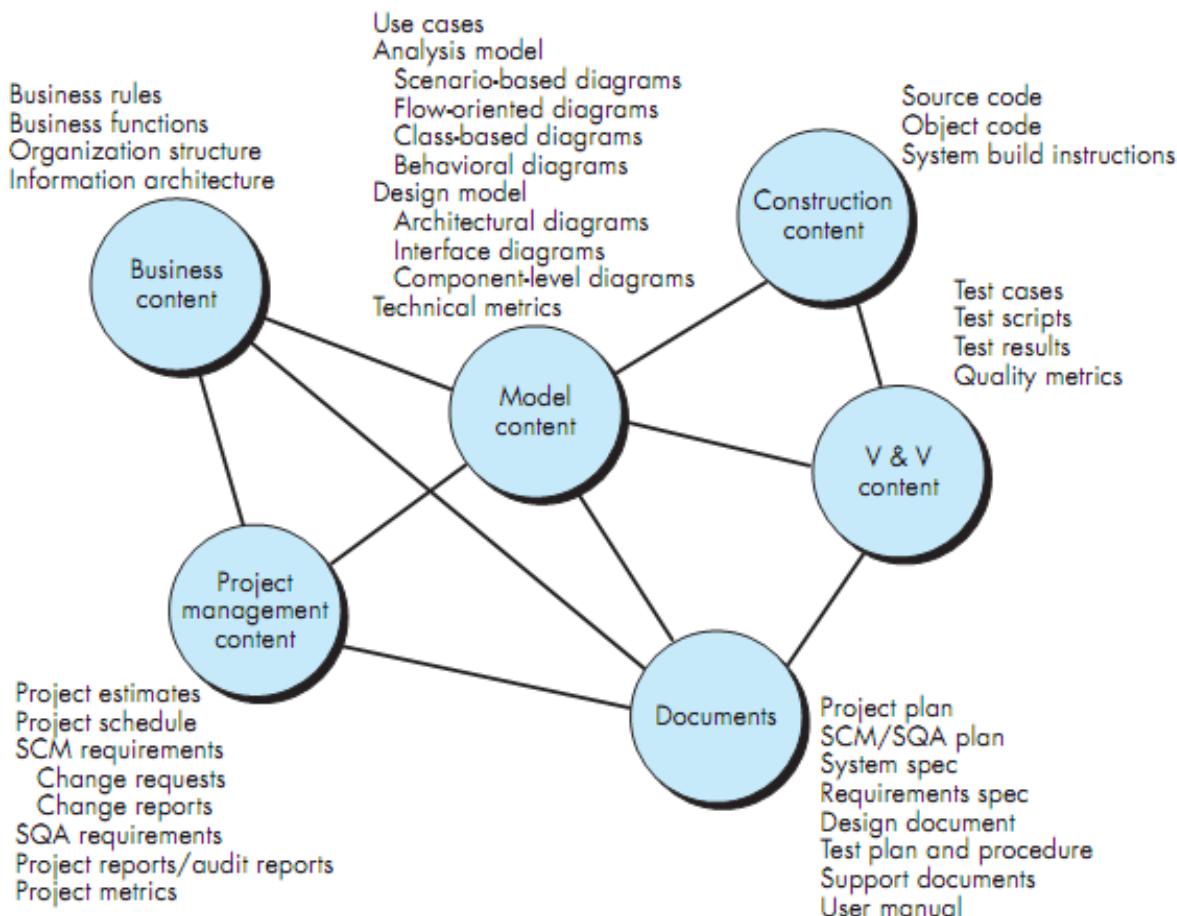
Kho lưu trữ SCM (SCM Repository) (2)



Kho lưu trữ SCM (SCM Repository)

(3)

- Nội dung repository:



Kho lưu trữ SCM (SCM Repository)

(4)

- Các đặc trưng của repository:

- Phiên bản

- Lưu trữ tất cả các phiên bản để cho phép quản lý các phiên bản đã đóng gói của sản phẩm và cho phép nhà phát triển có thể dùng các phiên bản này để phát triển, bảo trì.

- Theo dõi sự phụ thuộc và quản lý thay đổi

- Các kho quản lý lưu trữ mối quan hệ giữa các yếu tố dữ liệu được lưu trữ trong đó.

- Yêu cầu tìm kiếm

- Cung cấp để có thể theo dõi tất cả các bản thiết kế và các thành phần xây dựng và các sản phẩm mà kết quả từ một đặc tả cụ thể

- Quản lý cấu hình

- Lưu giữ tất cả những cấu hình cụ thể đại diện cho sự quan trọng và các sản phẩm đã đóng gói. Quản lý phiên bản các phiên bản cần thiết, và quản lý liên kết theo dõi phụ thuộc lẫn nhau.

- Thông tin tác giả

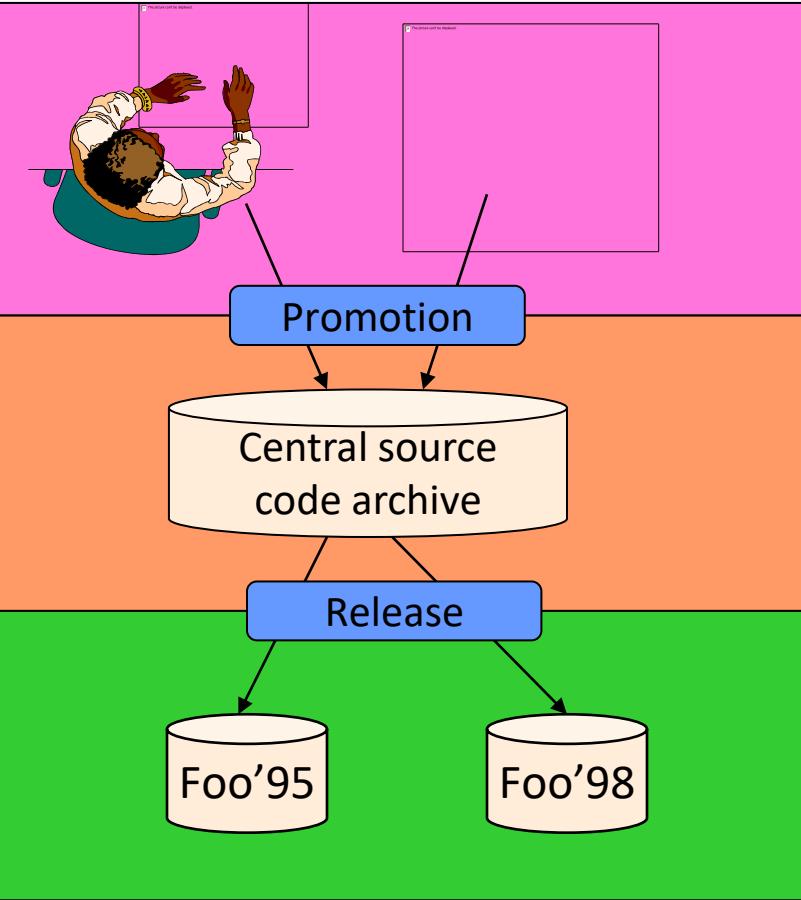
- Thiết lập đầy đủ các thông tin về thời gian bắt đầu và kết thúc, người thực hiện theo từng phiên bản.

Thư mục SCM (SCM Directory)

- Programmer's Directory (IEEE: Dynamic Library)
 - Thư viện để chứa các thực thể phần mềm mới được tạo hoặc sửa đổi. Không gian làm việc của lập trình viên chỉ do lập trình viên kiểm soát.
- Master Directory (IEEE: Controlled Library)
 - Quản lý (các) baseline và kiểm soát các thay đổi được thực hiện đối với chúng. Mục nhập được kiểm soát, thường sau khi được xác minh. Các thay đổi phải được cho phép.
- Software Repository (IEEE: Static Library)
 - Lưu trữ cho các baseline khác nhau được phát hành để sử dụng chung. Các bản sao của các baseline này có thể được cung cấp cho các tổ chức yêu cầu.

Thư mục SCM (SCM Directory) (2)

- Programmer's Directory
 - (IEEE Std: “Dynamic Library”)
 - Completely under control of one programmer.
- Master Directory
 - (IEEE Std: “Controlled Library”)
 - Central directory of all promotions.
- Software Repository
 - (IEEE Std: “Static Library”)
 - Externally released baselines.



Version vs. Revision vs. Release

- Phiên bản (Version):

- Một bản phát hành *ban đầu* hoặc tái phát hành một mục cấu hình được liên kết với một bản biên dịch hoặc biên dịch lại *hoàn chỉnh* của mục đó. Các phiên bản khác nhau có chức năng khác nhau.

- Bản sửa đổi (Revision):

- *Thay đổi* thành phiên bản chỉ sửa các lỗi trong thiết kế / mã, nhưng không ảnh hưởng đến chức năng đã được lập thành tài liệu.

- Bản phát hành (Release):

- Việc *phân phối chính thức* phiên bản đã được phê duyệt.

Các thành phần của một hệ thống quản lý cấu hình

- *Các yếu tố Cấu phần*—Một tập công cụ trong một hệ thống quản lý tập tin (ví dụ như dữ liệu) cho phép truy cập và quản lý mỗi mục cấu hình phần mềm.
- *Các yếu tố Xử lý*—Một tập các thủ tục và nhiệm vụ xác định một phương pháp hiệu quả quản lý sự thay đổi (và các hoạt động liên quan) cho tất cả các cử tri liên quan về quản lý , kĩ thuật và sử dụng phần mềm máy tính.
- *Các yếu tố Xây dựng* —Một tập hợp các công cụ tự động hóa việc xây dựng các phần mềm bằng cách đảm bảo rằng các thiết lập thích hợp của các thành phần được phê duyệt (tức là, đúng phiên bản) đã được lắp ráp.
- *Các yếu tố Con người* —để thực thi SCM hiệu quả, nhóm nghiên cứu phần mềm sử dụng một tập hợp các công cụ và tính năng quá trình (bao gồm các yếu tố khác SCM).

Nội dung

1. Khái niệm quản lý cấu hình phần mềm
- 2. Quy trình quản lý cấu hình phần mềm**
3. Quản lý phiên bản
4. Quản lý thay đổi

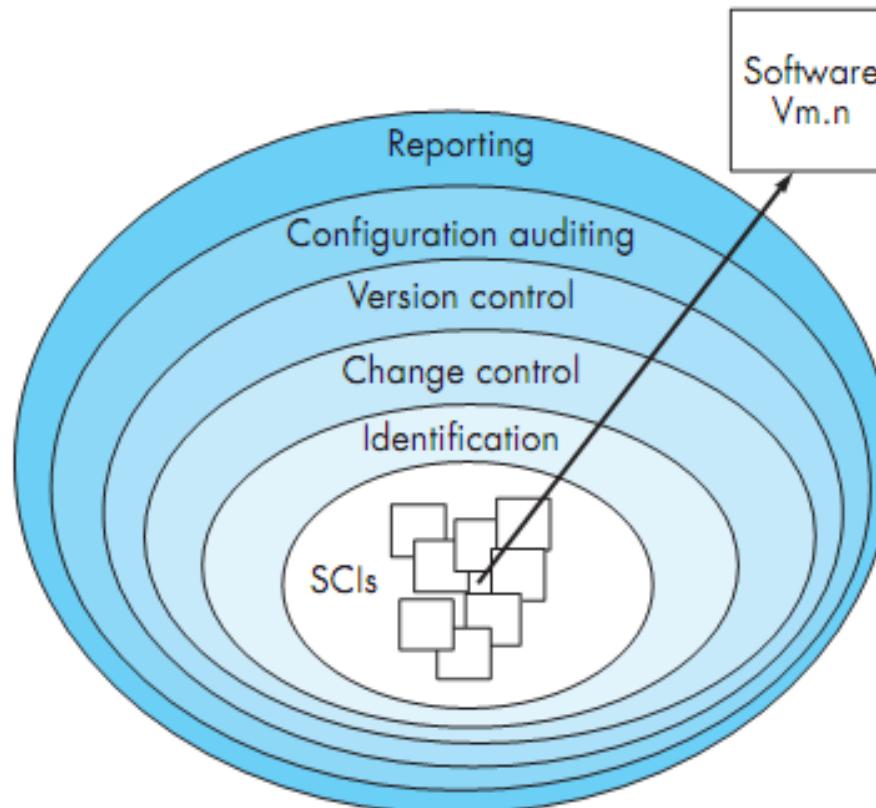
2.1 Quy trình SCM

Giải quyết các câu hỏi sau đây ...

- Một nhóm phần mềm xác định các yếu tố rủi ro của cấu hình phần mềm như thế nào?
- Một tổ chức quản lý nhiều phiên bản tồn tại của một chương trình (và tài liệu của nó) theo cách mà sẽ cho phép thay đổi một cách hiệu quả thế nào?
- Một tổ chức điều khiển những thay đổi trước và sau khi phần mềm được đóng gói bàn giao cho khách hàng như thế nào?
- Ai có trách nhiệm phê duyệt và xếp hạng những thay đổi?
- Làm thế nào chúng ta có thể đảm bảo rằng những thay đổi đã được thực hiện đúng cách?
- Cơ chế nào được sử dụng để đánh giá những thay đổi được thực hiện bởi người khác?

2.1 Quy trình SCM (2)

- 5 nhiệm vụ của SCM: identification, version control, change control, configuration auditing, reporting



2.1 Quy trình SCM (3)

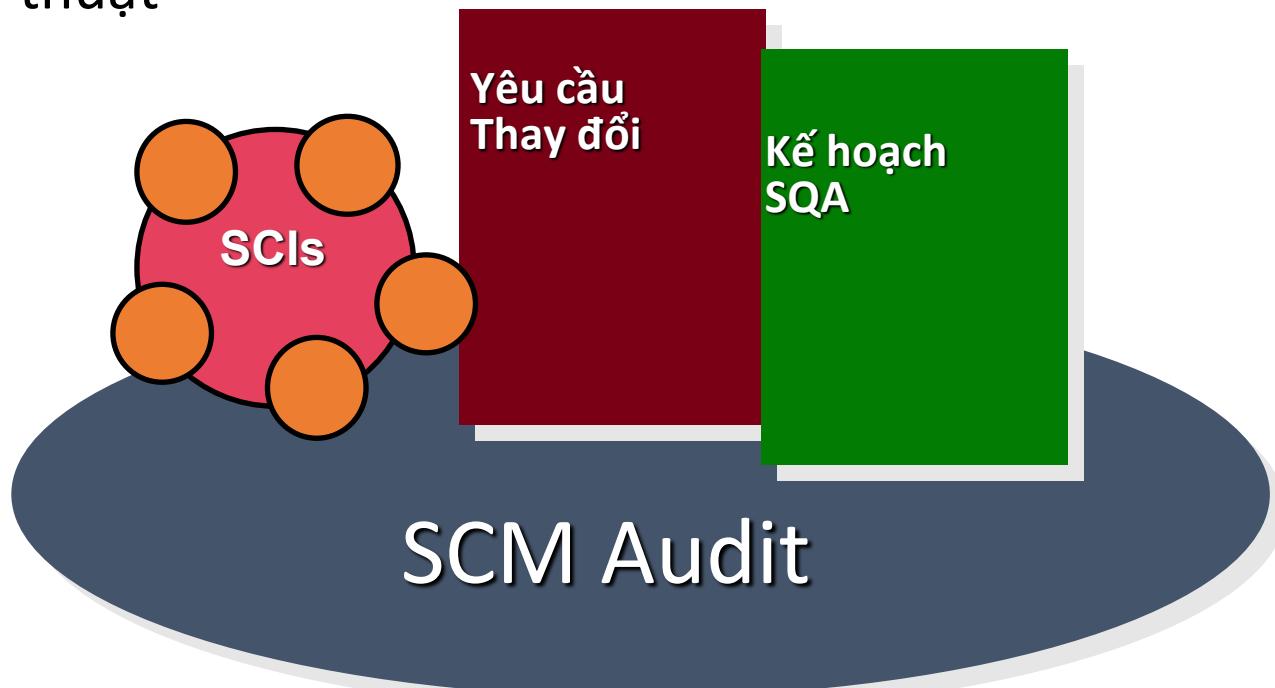
- Cung cấp một kho chứa an toàn đối với các kết quả bàn giao
- Cho phép việc kiểm soát và tiết lộ có nguyên tắc các kết quả bàn giao thông qua vòng đời của nó, với đầy đủ các dấu tích lịch sử, đảm bảo phiên bản đúng và cập nhật, đã được kiểm tra và phát hành
- Kiểm soát thay đổi của các kết quả bàn giao, đảm bảo các kết quả này được lưu theo đúng thứ tự
- Cung cấp việc lập báo cáo về hiện trạng của các kết quả bàn giao và những thay đổi của chúng

2.2 Các hoạt động trong SCM

- Nhận dạng mục cấu hình (Configuration item identification)
 - mô hình hóa hệ thống như một tập hợp các thành phần đang phát triển
- Quản lý tăng trưởng (Promotion management)
 - là việc tạo ra các phiên bản cho các nhà phát triển khác
- Quản lý phát hành (Release management)
 - là việc tạo ra các phiên bản cho khách hàng và người dùng
- Quản lý nhánh (Branch management)
 - là quản lý của sự phát triển đồng thời
- Quản lý biến thể (Variant management)
 - là việc quản lý các phiên bản dự định cùng tồn tại
- Quản lý thay đổi (Change management)
 - là việc xử lý, phê duyệt và theo dõi các yêu cầu thay đổi

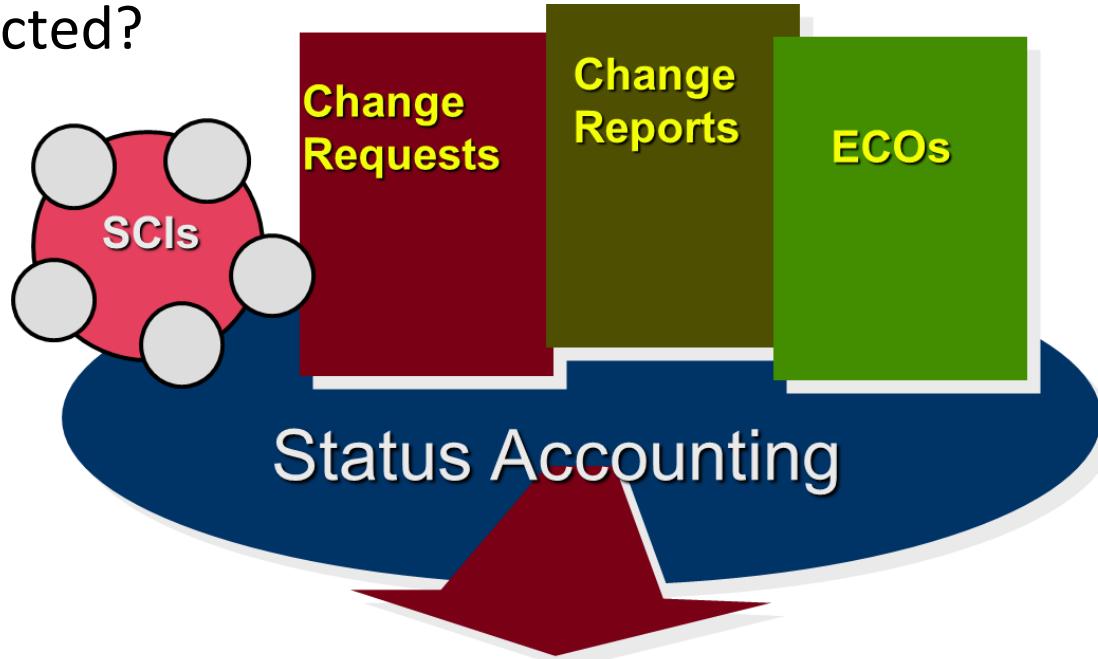
2.2 Các hoạt động trong SCM (2)

- Kiểm toán cấu hình (Configuration audit)
 - Đánh giá cấu hình phần mềm bổ sung cho việc xét duyệt kỹ thuật



2.2 Các hoạt động trong SCM (2)

- Báo cáo trạng thái (Status reporting)
 - Trả lời cho các câu hỏi: (1) What happened? (2) Who did it? (3) When did it happen? (4) What else will be affected?



2.3 Các vai trò trong SCM

- Người quản lý cấu hình
 - Chịu trách nhiệm xác định các mục cấu hình (configuration items – CI). Người quản lý cấu hình cũng có thể chịu trách nhiệm xác định các thủ tục để tạo các sự tăng trưởng và các bản phát hành.
- Thành viên ban kiểm soát thay đổi
 - Chịu trách nhiệm phê duyệt hoặc từ chối các yêu cầu thay đổi
- Lập trình viên
 - Tạo các thay đổi được kích hoạt bởi các yêu cầu. Nhà phát triển kiểm tra các thay đổi và giải quyết xung đột
- Kiểm soát viên
 - Chịu trách nhiệm về việc lựa chọn và đánh giá các thay đổi để phát hành và đảm bảo tính nhất quán và đầy đủ của bản phát hành này

Nội dung

1. Khái niệm quản lý cấu hình phần mềm
2. Quy trình quản lý cấu hình phần mềm
- 3. Quản lý phiên bản**
4. Quản lý thay đổi

3.1 Khái niệm quản lý phiên bản

- **Quản lý phiên bản** (Version management hay kiểm soát phiên bản – Version control) là quá trình theo dõi các phiên bản khác nhau của các thành phần phần mềm hoặc các mục cấu hình và hệ thống mà các thành phần này được sử dụng.
- Nó cũng liên quan đến việc đảm bảo rằng các thay đổi do các nhà phát triển khác nhau thực hiện đối với các phiên bản này không ảnh hưởng lẫn nhau.
- Quản lý phiên bản gắn liền với quản lý baselines, chỉ định các phiên bản thành phần được bao gồm trong hệ thống cụ thể

3.1 Khái niệm quản lý phiên bản (2)

- Ví dụ: Codeline (A)



Codeline (B)



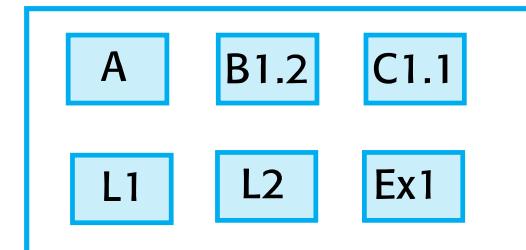
Codeline (C)



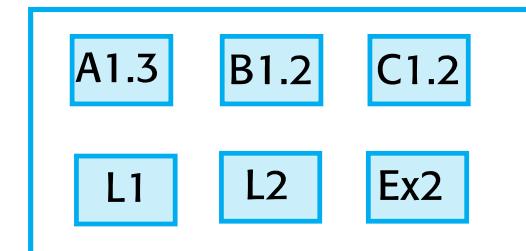
Libraries and external components



Baseline - V1



Baseline - V2



Mainline

- Codeline là một chuỗi các phiên bản mã nguồn với các phiên bản mới hơn trong chuỗi bắt nguồn từ các phiên bản trước đó.

3.2 Hệ thống kiểm soát phiên bản

- Hệ thống quản lý phiên bản kết hợp các thủ tục và các công cụ để quản lý các phiên bản khác nhau của các đối tượng cấu hình được tạo ra trong quá trình phần mềm
- Một hệ thống kiểm soát phiên bản thực hiện hoặc được tích hợp trực tiếp với bốn tính năng chính:
 - **Một cơ sở dữ liệu dự án (kho)** lưu trữ tất cả các đối tượng cấu hình có liên quan.
 - **Quản lý phiên bản** cho phép lưu trữ tất cả các phiên bản của đối tượng cấu hình (hoặc cho phép bất kỳ phiên bản được xây dựng bằng cách sử dụng sự khác biệt so với phiên bản trước đây).
 - **Make facility** cho phép các kỹ sư phần mềm để thu thập tất cả các đối tượng cấu hình có liên quan và xây dựng một phiên bản đặc biệt của phần mềm.
 - **Theo dõi vấn đề** (còn được gọi là *bug tracking*) cho phép các nhóm ghi lại và theo dõi tình trạng của tất cả các vấn đề nổi bật liên quan đến từng đối tượng cấu hình.

3.2 Hệ thống kiểm soát phiên bản (2)

- Có hai loại hệ thống điều khiển phiên bản hiện đại
 - **Hệ thống tập trung**, nơi có một kho lưu trữ chính duy nhất duy trì tất cả các phiên bản của các thành phần phần mềm đang được phát triển. **Subversion** là một ví dụ về hệ thống kiểm soát phiên bản tập trung.
 - **Hệ thống phân tán**, nơi tồn tại nhiều phiên bản của kho thành phần cùng một lúc. **Git** là một ví dụ về hệ thống kiểm soát phiên bản phân tán.

3.2 Hệ thống kiểm soát phiên bản (3)

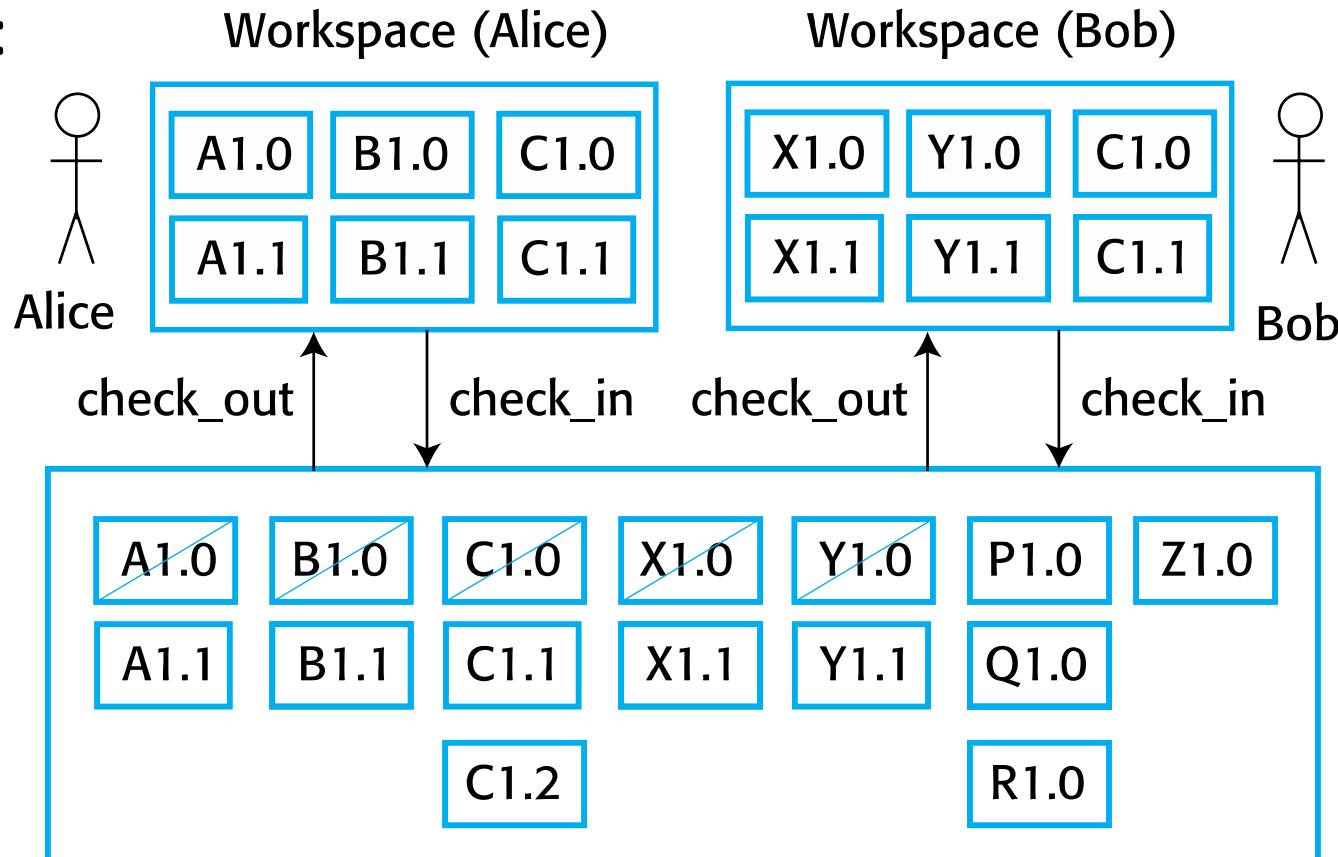
- Khái niệm về *public repository* và *private workspaces*
 - Để hỗ trợ phát triển độc lập mà không bị can thiệp, hệ thống kiểm soát phiên bản sử dụng khái niệm kho lưu trữ dự án và không gian làm việc riêng.
 - **Kho lưu trữ dự án duy trì phiên bản ‘chính’** của tất cả các thành phần. Nó được sử dụng để tạo đường cơ sở cho việc xây dựng hệ thống.
 - Khi sửa đổi các thành phần, nhà phát triển sao chép (**check-out**) những thành phần này từ kho lưu trữ vào không gian làm việc của họ và làm việc trên các bản sao này.
 - Khi họ hoàn thành các thay đổi của mình, các thành phần đã thay đổi sẽ được trả lại (**checked-in**) vào kho lưu trữ.

Kiểm soát phiên bản tập trung

- Các nhà phát triển *check-out* các thành phần hoặc thư mục của các thành phần từ kho lưu trữ dự án vào không gian làm việc riêng tư của họ và làm việc trên các bản sao này.
- Khi các thay đổi của họ hoàn tất, họ *check-in* các thành phần trở lại kho lưu trữ.
- Nếu nhiều người đang làm việc trên một thành phần cùng một lúc, mỗi người *check-out* nó từ kho lưu trữ → hệ thống sẽ cảnh báo rằng nó đã được người khác *check-*

Kiểm soát phiên bản tập trung (2)

- Ví dụ:



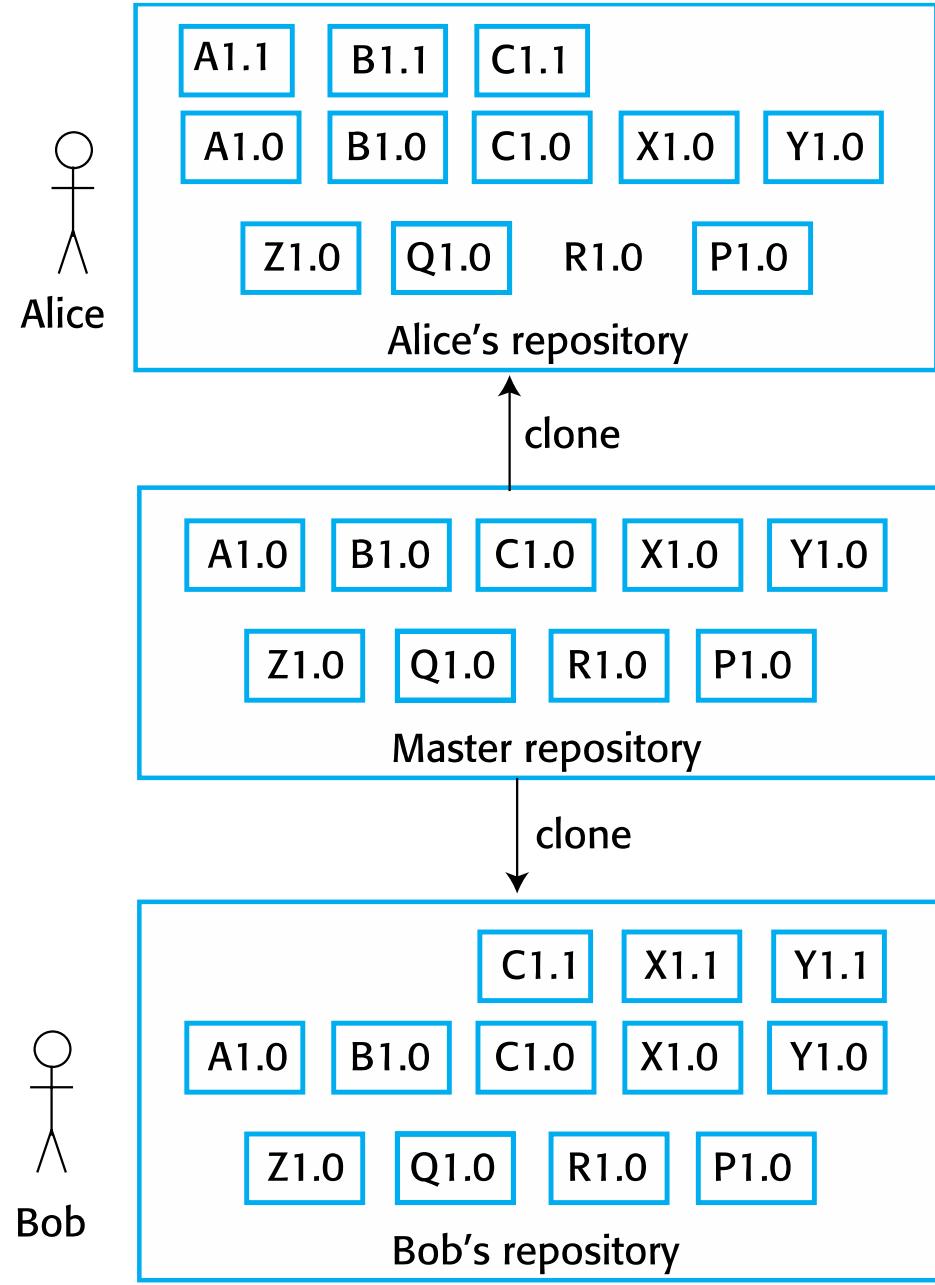
Kiểm soát phiên bản phân tán

- Kho lưu trữ ‘**master**’ được tạo trên máy chủ duy trì mã nguồn do nhóm phát triển tạo ra.
- Thay vì *check-out* các tệp cần dùng, một nhà phát triển tạo một bản sao của kho lưu trữ dự án được tải xuống và cài đặt trên máy tính.
- Các nhà phát triển làm việc trên các tệp được yêu cầu và duy trì các phiên bản mới trên kho lưu trữ riêng trên máy tính của mình.
- Khi các thay đổi được thực hiện, nhà phát triển ‘**commit**’ những thay đổi này và cập nhật kho lưu trữ riêng. Sau đó, có thể ‘**push**’ những thay đổi này vào kho dự án.

Kiểm soát phiên bản phân tán (2)

- Ví dụ:

Repository cloning

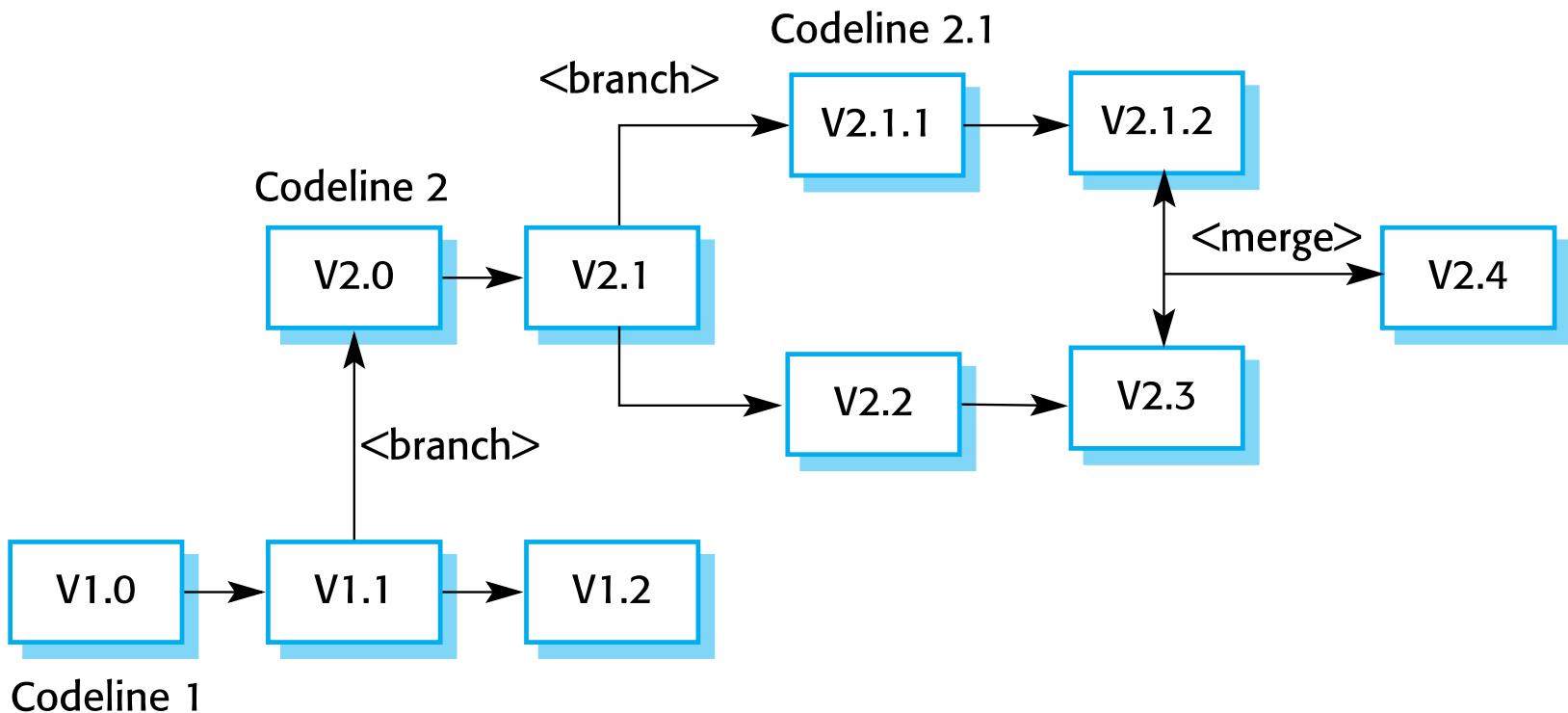


Kiểm soát phiên bản phân tán (3)

- Phân nhánh và hợp nhất (branching & merging):
 - Thay vì một chuỗi phiên bản tuyến tính phản ánh các thay đổi đối với thành phần theo thời gian, có thể có một số chuỗi độc lập.
 - Điều này là bình thường trong quá trình phát triển hệ thống, nơi các nhà phát triển khác nhau làm việc độc lập trên các phiên bản khác nhau của mã nguồn và vì vậy hãy thay đổi nó theo những cách khác nhau.
 - Ở một số giai đoạn, có thể cần hợp nhất các nhánh dòng mã để tạo phiên bản mới của thành phần bao gồm tất cả các thay đổi đã được thực hiện.
 - Nếu các thay đổi được thực hiện liên quan đến các phần khác nhau của mã, các phiên bản thành phần có thể được hợp nhất tự động bằng cách kết hợp các delta áp dụng cho mã.

Kiểm soát phiên bản phân tán (4)

- Phân nhánh và hợp nhất (branching & merging):



Kiểm soát phiên bản phân tán (5)

- Lợi ích:
 - Cung cấp một cơ chế sao lưu cho kho lưu trữ.
 - Nếu kho lưu trữ bị hỏng, công việc có thể tiếp tục và kho lưu trữ dự án có thể được khôi phục từ các bản sao cục bộ.
 - Cho phép hoạt động ngoại tuyến để các nhà phát triển có thể thực hiện các thay đổi nếu không có kết nối mạng.
 - Các nhà phát triển có thể biên dịch và kiểm tra toàn bộ hệ thống trên máy cục bộ và kiểm tra những thay đổi đã thực hiện.

3.3 Đánh số phiên bản

- Định danh/đánh số phiên bản
 - Một trong những hoạt động nền tảng của quản lý cấu hình.
 - Mục đích của định danh là để xác định tính duy nhất của một mục cấu hình (CI), cũng như mối quan hệ của nó với các CI khác.
 - Nó bao gồm việc mô tả tên, đánh số, đánh dấu đặc trưng, giúp nhận biết và phân biệt một CI với các CI hay thành phần khác.
- Một số phương pháp gán số hiệu phiên bản được dùng phổ biến:
 - Đánh số phiên bản bằng các con số (Sequence-based identifiers)
 - Đánh số hiệu phiên bản dựa theo mức độ ổn định của sản phẩm (Stage-based identifiers)

3.3 Đánh số phiên bản (2)

- Đánh số phiên bản bằng các con số
 - Sử dụng các con số (đôi khi kết hợp thêm các chữ cái) để gán số hiệu cho các phiên bản
 - Công thức đánh số hiệu phiên bản:
major.minor.[build [.revision]] hoặc
major.minor [maintenance[.build]]
 - Ý nghĩa các số major, minor, build, revision
 - major: Chuỗi phiên bản chính
 - minor: Chuỗi phiên bản phụ
 - build: Chuỗi phiên bản cấu tạo. Đánh dấu sự khác nhau trong cùng 1 phiên bản phụ, 2 chữ số
 - revision: Lần sửa đổi, đánh dấu lần sửa đổi của mã nguồn

Ví dụ: mô hình định danh 3 chữ số

MUE.0.0.1:Release

Alpha test release

MUE.1.0.0:Release

First major release

MUE.1.2.1:Release

Second minor release
with bug fixes

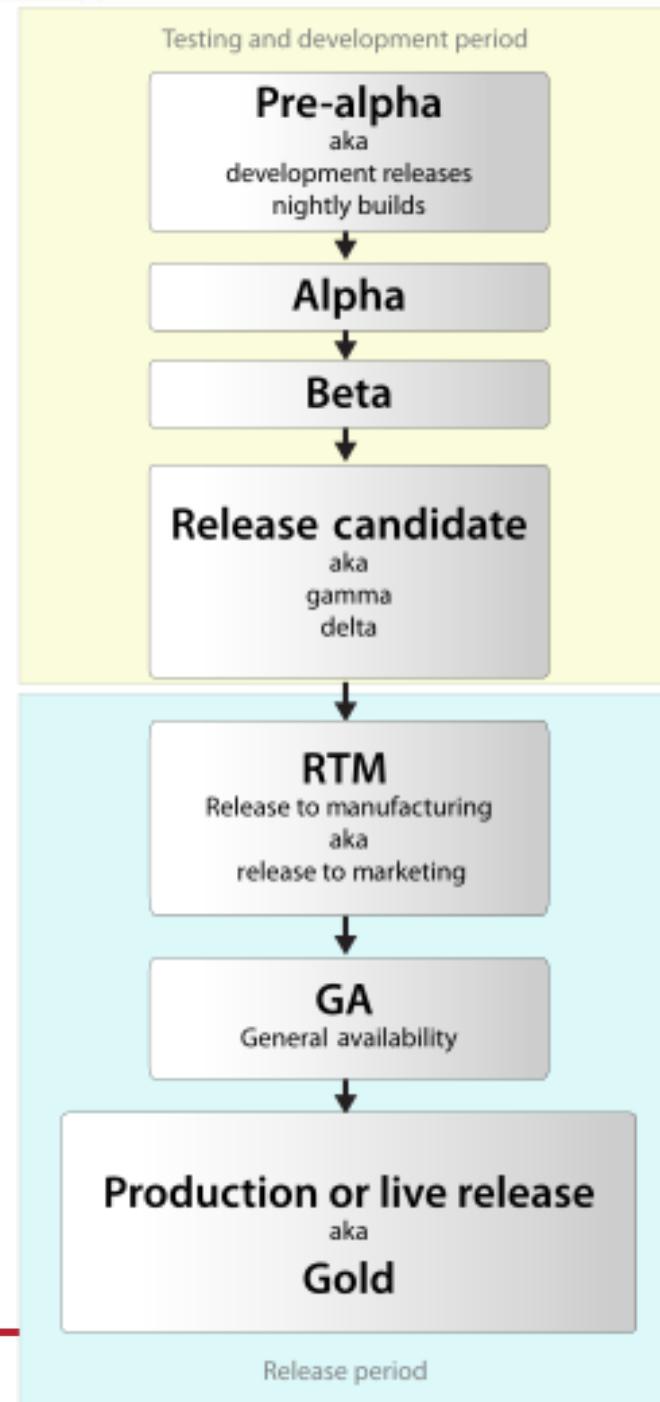
MUE.2.0.3:Release

Second major
release with three
series of bug fixes

3.3 Đánh số phiên bản (3)

- Đánh số hiệu phiên bản dựa theo mức độ ổn định của sản phẩm, sử dụng các tên gọi:
 - Closebeta: Phiên bản thử nghiệm hạn chế
 - Openbeta: Phiên bản thử nghiệm diện rộng
 - Release Candidate (RC): phiên bản ứng viên
 - Official version: phiên bản chính thức
 - ...

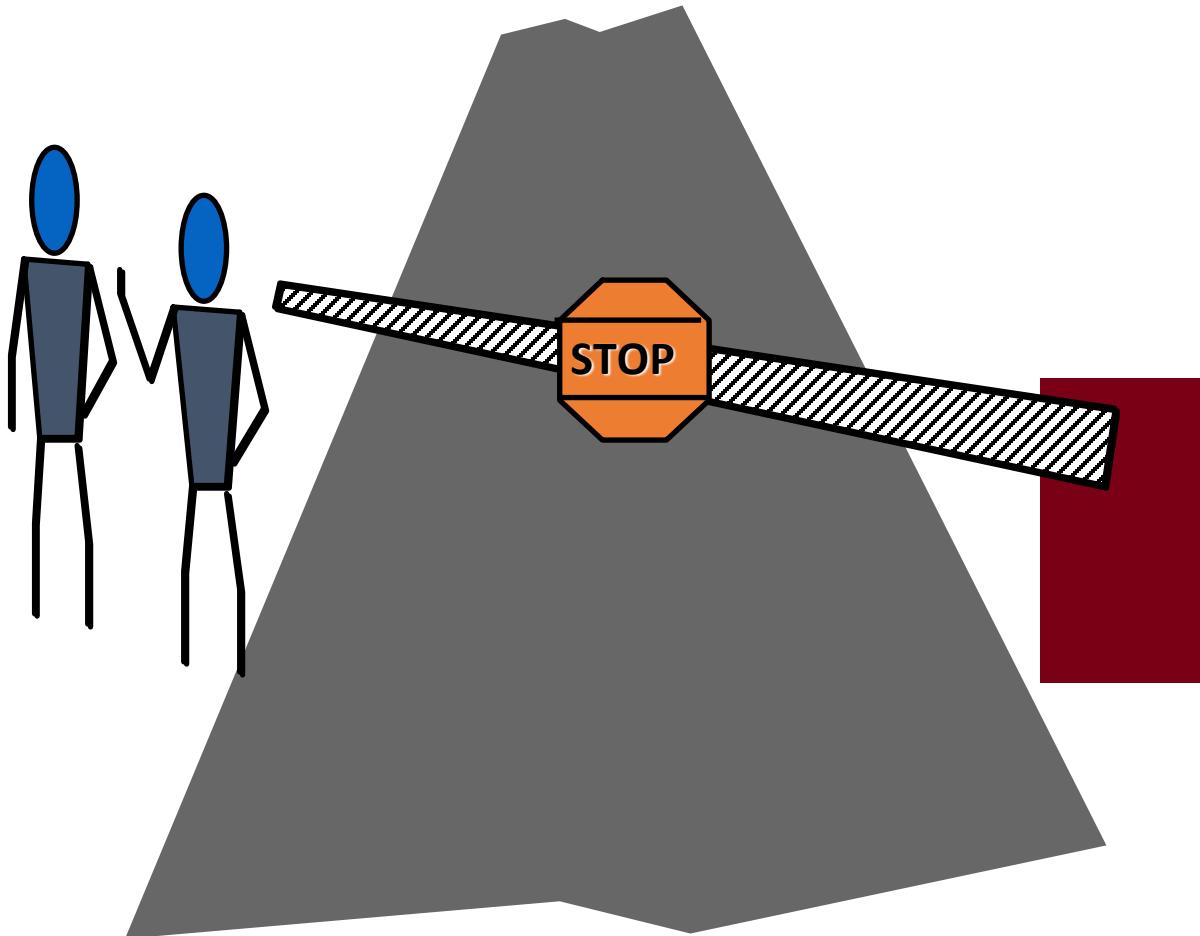
Ví dụ: mô hình tên phiên bản theo độ ổn định của sản phẩm



Nội dung

1. Khái niệm quản lý cấu hình phần mềm
2. Quy trình quản lý cấu hình phần mềm
3. Quản lý phiên bản
- 4. Quản lý thay đổi**

Quản lý thay đổi



Quản lý thay đổi (2)

- Các yêu cầu của tổ chức thay đổi trong suốt thời gian tồn tại của hệ thống, các lỗi phải được sửa chữa và các hệ thống phải thích ứng với những thay đổi trong môi trường của chúng.
- **Quản lý thay đổi** nhằm đảm bảo rằng sự phát triển của hệ thống là một quá trình được quản lý và ưu tiên dành cho những thay đổi cấp bách nhất và tiết kiệm chi phí.
- Quá trình quản lý thay đổi liên quan đến việc **phân tích chi phí và lợi ích** của những thay đổi được đề xuất, **phê duyệt** những thay đổi đáng giá và **theo dõi** những thành phần nào trong hệ thống đã được thay đổi.

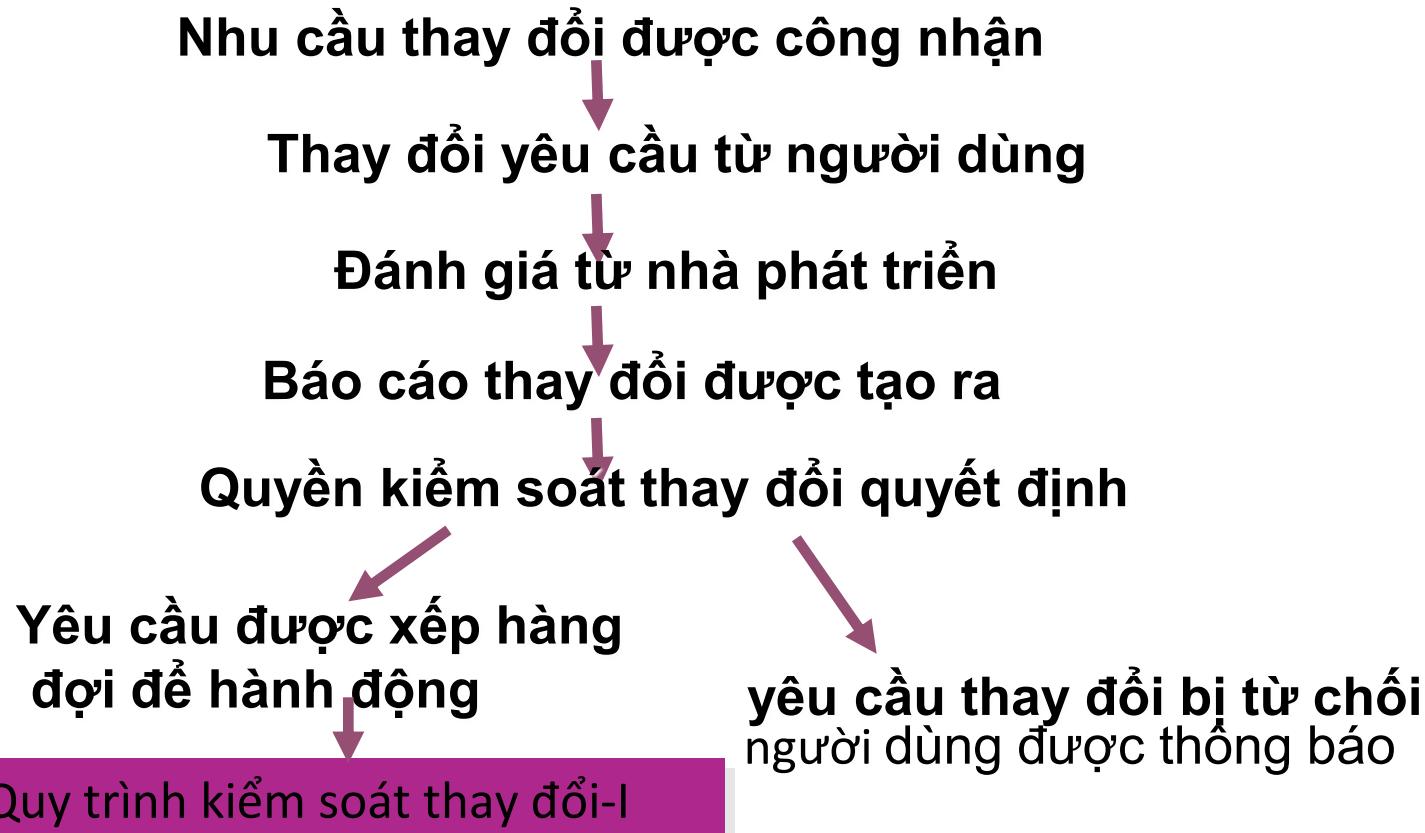
Quản lý thay đổi (3)

- Mức độ phức tạp của quy trình quản lý thay đổi thay đổi theo dự án.
- Các dự án nhỏ có thể thực hiện các yêu cầu thay đổi một cách không chính thức và nhanh chóng trong khi các dự án phức tạp yêu cầu các biểu mẫu yêu cầu thay đổi chi tiết và sự chấp thuận chính thức của một người quản lý khác.

Quy trình quản lý thay đổi

- Thay đổi được yêu cầu (có thể được thực hiện bởi bất kỳ ai bao gồm cả người dùng và nhà phát triển)
- Yêu cầu thay đổi được đánh giá dựa trên các mục tiêu của dự án
- Sau khi đánh giá, thay đổi được chấp nhận hoặc bị từ chối
- Nếu nó được chấp nhận, thay đổi được chỉ định cho người phát triển và được triển khai
- Thay đổi đã thực hiện được kiểm tra.

Quy trình quản lý thay đổi (2)

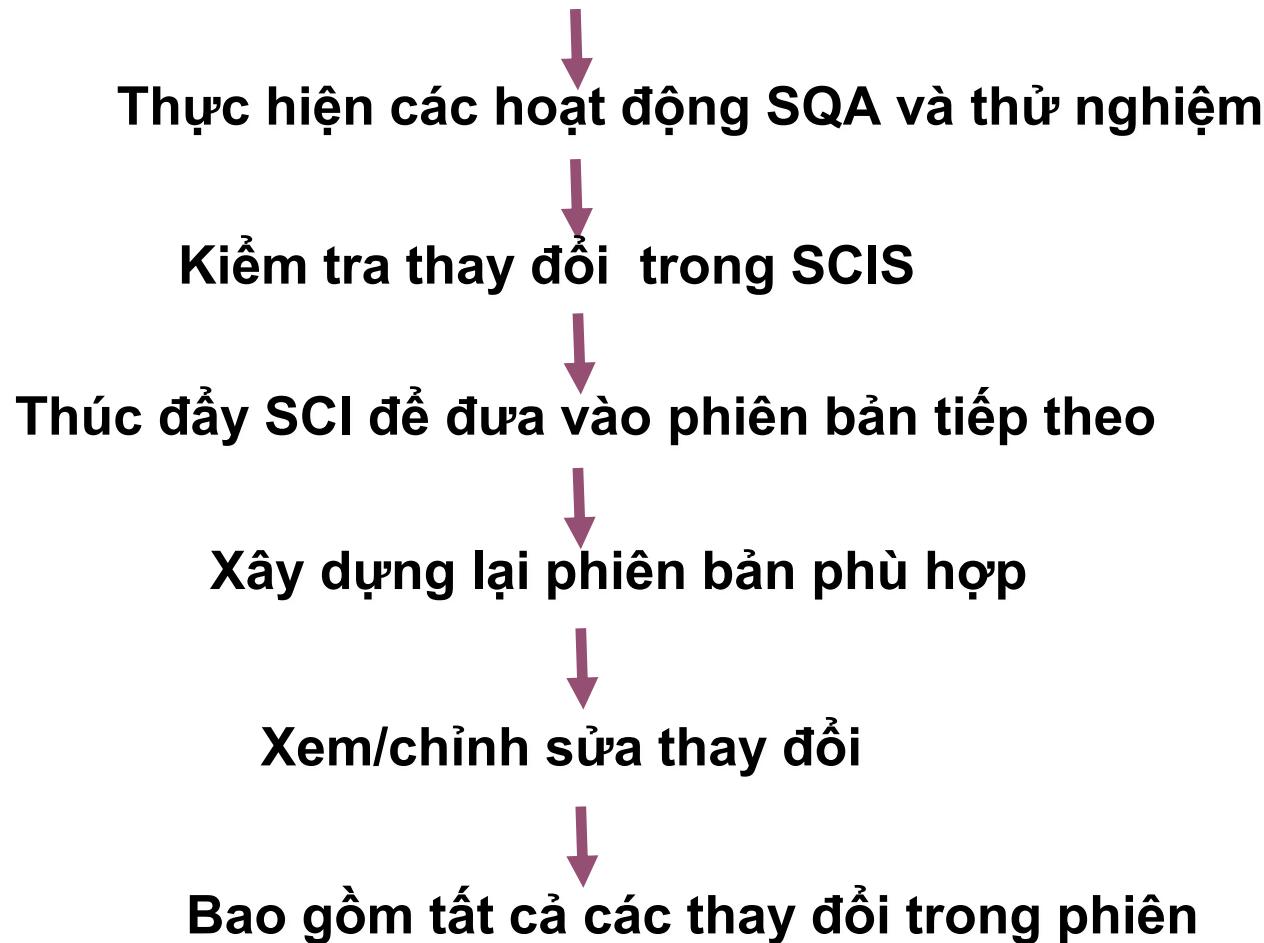


Quy trình quản lý thay đổi (3)



Quy trình kiểm soát thay đổi-III

Quy trình quản lý thay đổi (3)



Tổng kết

- Hệ thống phần mềm liên tục thay đổi trong quá trình phát triển và sử dụng.
- Quản lý cấu hình phần mềm là một phần cơ bản của kế hoạch quản lý dự án để quản lý các hệ thống phần mềm đang phát triển và điều phối các thay đổi đối với chúng.
- Quản lý cấu hình liên quan đến các chính sách, quy trình và công cụ để quản lý các hệ thống phần mềm đang thay đổi.
- SCM cần thiết vì rất dễ mất dấu những thay đổi và các phiên bản thành phần đã được tích hợp vào mỗi phiên bản hệ thống.
- SCM rất cần thiết cho các dự án nhóm để kiểm soát các thay đổi do các nhà phát triển khác nhau thực hiện

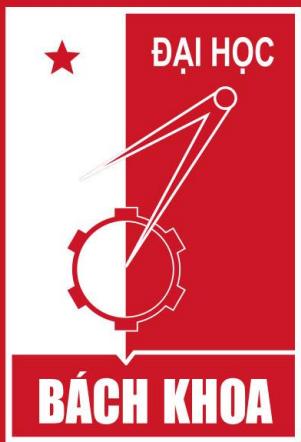


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The graphic consists of the number "25" in large, bold, white font. A curved banner arches over the top of the "2" containing the text "YEARS ANNIVERSARY". Below the "25" is the acronym "SOKT" in a large, bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 6

Kỹ nghệ yêu cầu phần mềm
(Requirement Engineering)

Mục tiêu của bài học

Sinh viên sẽ được trang bị các kiến thức sau:

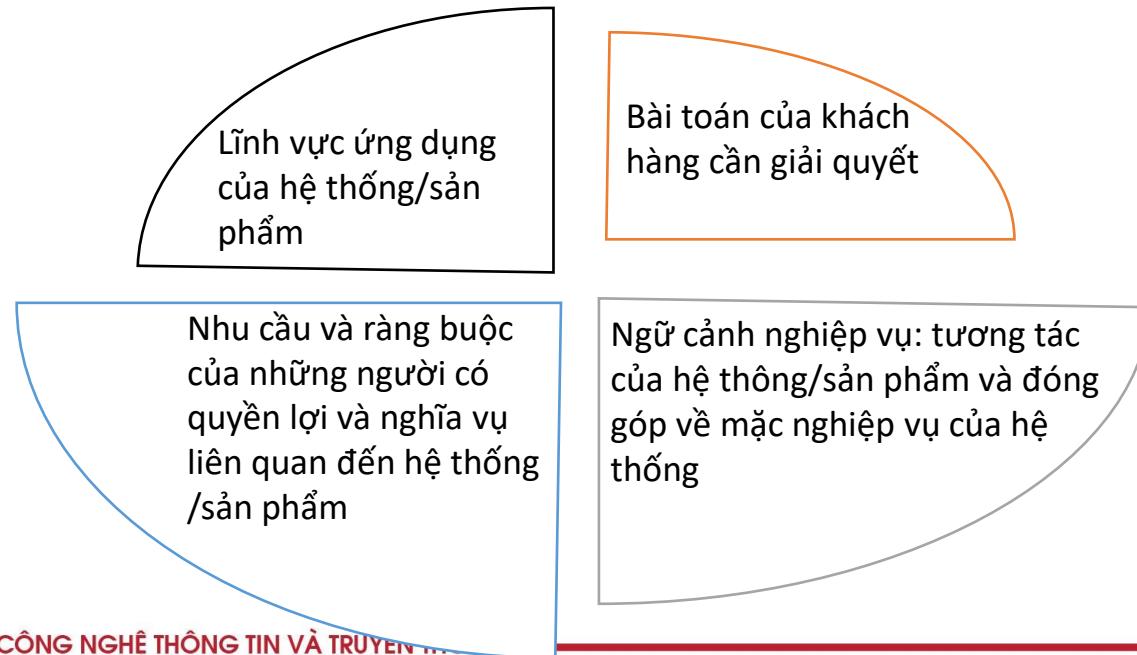
- Hiểu rõ các khái niệm liên quan tới kĩ nghệ yêu cầu phần mềm
- Biết, nắm vững vị trí, vai trò của kĩ nghệ YCPM
- Có khái niệm về một số yếu tố liên quan tới các yêu cầu chức năng và phi chức năng
- Nắm vững các hoạt động chính của kĩ nghệ YCPM

Nội dung

1. Khái niệm
2. Tầm quan trọng của yêu cầu phần mềm
3. Yêu cầu chức năng và yêu cầu phi chức năng
4. Các hoạt động chính trong kỹ nghệ yêu cầu phần mềm

1. Khái niệm (1)

- Các đặc tính của hệ thống hay sản phẩm do khách hàng - người sử dụng PM - đặt ra → Xác định được phần mềm đáp ứng được các yêu cầu và mong muốn của khách hàng - người sử dụng phần mềm



1. Khái niệm (2)

- **Khởi đầu (Inception):** Hỏi một loạt các câu hỏi để xác định:
 - Hiểu biết căn bản về vấn đề cần giải quyết.
 - Người đang cần giải pháp
 - Loại giải pháp mong muốn
 - Mức độ hiệu quả ban đầu của việc trao đổi thông tin giữa khách hàng và nhà phát triển
- **Khám phá (Elicitation):** tìm ra yêu cầu của tất cả khách hàng.
- **Xây dựng (Elaboration):** tạo ra mô hình phân tích xác định dữ liệu, chức năng và hành vi được yêu cầu.
- **Đàm phán (Negotiation):** đồng ý với một hệ thống có thể bàn giao một cách thực tế đối với cả 2 bên.

1. Khái niệm (3)

- **Đặc tả (Specification):** có thể là một/ nhiều những thứ sau:
 - Một Tài liệu được viết
 - Một Tập hợp các mô hình
 - Một hình thức biểu diễn toán học
 - Một tập các kịch bản người dùng (use-case)
 - Một Nguyên mẫu
- **Đánh giá (Validation):** tạo cơ chế xem xét các vấn đề:
 - Sai sót trong nội dung hoặc giải thích.
 - Phần được yêu cầu làm rõ.
 - Thông tin bị thiếu
 - Mâu thuẫn
 - Yêu cầu không thực tế, không thể đạt được.
- **Quản lý các Yêu cầu (Requirements management)**

Nội dung

1. Khái niệm
- 2. Tâm quan trọng của yêu cầu phần mềm**
3. Yêu cầu chức năng và yêu cầu phi chức năng
4. Các hoạt động chính trong kỹ nghệ yêu cầu phần mềm

Nguồn gốc yêu cầu phần mềm

- ❑ **Người sử dụng (Khách hàng):** theo mô hình phân lớp của yêu cầu phần mềm Khách hàng được chia làm hai loại:
 - Khách hàng cung cấp các business requirement: cung cấp các thông tin về công ty, về các đặc điểm ở mức độ cao, về mô hình và phạm vi của hệ thống
 - Khách hàng cung cấp các user requirement: cung cấp các công tin về từng nhiệm vụ cụ thể mà họ sẽ làm việc với phần mềm
- ❑ Cần phải phối hợp, kết hợp chặt chẽ với hai phân loại khách hàng trên

Đặc điểm của khách hàng phần mềm

- Khách hàng chỉ có những **ý tưởng còn mơ hồ** về phần mềm cần phải xây dựng để phục vụ công việc của họ.
- Cho nên chúng ta phải sẵn sàng, kiên trì theo đuổi để đi từ các ý tưởng mơ hồ đó đến “Phần mềm có đầy đủ các tính năng cần thiết”
- Khách hàng rất **hay thay đổi** các đòi hỏi của mình, chúng ta nắm bắt được các thay đổi đó và sửa đổi các mô tả một cách hợp lý

Vấn đề YCPM giải quyết (1)

❑ Sự tham gia quá mức của NSD

- ✓ Thông thường người sử dụng không hiểu rõ về quá trình xây dựng các yêu cầu phần mềm và các đặc điểm của phần mềm.
- ✓ Họ sẽ đưa những đòi hỏi quá cao hoặc chẳng liên quan đến quá trình phát triển phần mềm như viết code, ...
- ✓ Họ đưa ra những yêu cầu và đề nghị rất khó chấp nhận và gây khó khăn cho các PTV

❑ Có quá nhiều yêu cầu trong yêu cầu phần mềm

- ✓ Thông thường các yêu cầu phần mềm được phát hiện trong quá trình khảo sát và rất có thể các yêu cầu về phần mềm sẽ lớn hơn khả năng của đội ngũ phát triển về: nhân lực, thời gian, tài chính.
- ✓ Cần hạn chế không để các yêu cầu phần mềm phát sinh đi quá phạm vi và giới hạn của phần mềm
- ✓ Cần quản lý các thay đổi về yêu cầu phần mềm một cách hợp lý và xem xét ảnh hưởng của nó tới kiến trúc hệ thống, ... trong quá trình phát triển

Vấn đề YCPM giải quyết (2)

❑ Các yêu cầu phần mềm mơ hồ nhập nhằng

- ✓ Đây là một vấn đề rất hay xảy ra trong quá trình phát triển các yêu cầu phần mềm
- ✓ Các yêu cầu phần mềm cần phải rõ ràng, không được phép hiểu theo nhiều cách
- ✓ Phương pháp sửa các yêu cầu mơ hồ nhập nhằng là làm lại, đặc tả lại các yêu cầu này.
- ✓ Theo đánh giá của các nhà phân tích: làm lại yêu cầu phần mềm thường chiếm khoảng 40% quá trình xây dựng nó và 70-80% các đặc tính xây dựng lại có thể dẫn đến các lỗi
- ✓ Lưu ý tới từng yêu cầu phần mềm và không để sót những yêu cầu mơ hồ, không rõ ràng

❑ Không lưu ý tới người sử dụng phần mềm

- ✓ Thông thường phần mềm làm ra cho một tập hợp các đối tượng nào đó sử dụng
- ✓ Cần quan tâm tới đặc điểm của đối tượng này

Vấn đề YCPM giải quyết (3)

❑ Các đặc tính thừa:

- ✓ Thông thường người phát triển theo các thói quen nghề nghiệp thêm vào các yêu cầu phần mềm các chức năng không cần thiết cho phần mềm
- ✓ Tương tự như vậy người sử dụng có thể đưa ra một số yêu cầu phụ cho phần mềm. Các yêu cầu này có thể đòi hỏi các tốn kém về mặt xây dựng mà trên thực tế hoàn toàn không cần thiết
- ✓ Cần đánh giá các đặc tính và tính cần thiết của nó đối với các phần mềm. Những đặc tính phụ có thể xem xét kỹ hơn xem khả năng đáp ứng nó về mặt kỹ thuật có đáng giá hay không.

❑ Đặc tả quá ít

- ✓ NSD là chuyên viên trong một lĩnh vực nào đó có thói quen nghĩ rằng tất cả các LTV đều là các chuyên viên trong lĩnh vực đó.
- ✓ NSD đưa ra những yêu cầu quá ngắn gọn mà không miêu tả kỹ lưỡng chúng là gì
- ✓ Cần hỏi rõ NSD và tranh thủ các kiến thức của họ

Vấn đề YCPM giải quyết (4)

❑ Kế hoạch sai:

- ✓ Các LTV đánh giá sai về mức độ phức tạp của vấn đề và dẫn tới lập kế hoạch sai về mặt thời gian hoàn thành.
- ✓ Cần lưu ý đánh giá thật chắc chắn các đặc tính của phần mềm khi lập kế hoạch xây dựng phần mềm
- ✓ Nên trả lời cho khách hàng các câu trả lời dạng “gần chính xác”: trong trường hợp xấu nhất..., nếu.... Thì.

Nội dung

1. Khái niệm
2. Tầm quan trọng của yêu cầu phần mềm
- 3. Yêu cầu chức năng và yêu cầu phi chức năng**
4. Các hoạt động chính trong kỹ nghệ yêu cầu phần mềm

Phân loại yêu cầu

- Theo 4 thành phần của phần mềm:
 - Các yêu cầu về phần mềm (Software)
 - Các yêu cầu về phần cứng (Hardware)
 - Các yêu cầu về dữ liệu (Data)
 - Các yêu cầu về con người (People, Users)
- Theo cách đặc tả phần mềm
 - Các yêu cầu chức năng
 - Các yêu cầu ngoài chức năng
 - Các ràng buộc khác

Yêu cầu chức năng

- Miêu tả các chức năng của hệ thống, phụ thuộc vào kiểu phần mềm và mong đợi của người dùng
 - Tương tác giữa phần mềm và môi trường, độc lập với việc cài đặt
 - Ví dụ: Hệ thống đồng hồ phải hiển thị thời gian dựa trên vị trí của nó
- Các công cụ đặc tả yêu cầu chức năng tiêu biểu:
 - Biểu đồ luồng dữ liệu (Data Flow Diagrams)
 - Máy trạng thái hữu hạn (Finite State Machines)
 - Mạng Petri (Petri nets),...
 - Tuy nhiên không bắt buộc và có thể dùng ngôn ngữ tự nhiên.

Yêu cầu phi chức năng và ràng buộc

- Yêu cầu phi chức năng: Định nghĩa các khía cạnh sử dụng phần mềm, không liên quan trực tiếp tới các hành vi chức năng:
 - Các tính chất của hệ thống như độ tin cậy, thời gian trả lời, dung lượng bộ nhớ, ...
 - Thời gian trả lời phải nhỏ hơn 1 giây
- Ràng buộc: do khách hàng hay môi trường thực thi phần mềm đặt ra
 - Các yêu cầu do tổ chức qui định như qui định chuẩn về quá trình tiến hành, chuẩn tài liệu, ...
 - Ngôn ngữ cài đặt phải là COBOL
 - Các yêu cầu từ bên ngoài
 - Phải giao tiếp với hệ thống điều phối được viết vào năm 1956.
- Thường sử dụng các công cụ
 - Biểu đồ thực thể liên kết (Entity-Relationship Diagrams)
 - Đặc tả Logic (Logic Specifications)
 - Đặc tả đại số (Algebraic Specifications)

→ Khó phát biểu chính xác, Rất khó kiểm tra

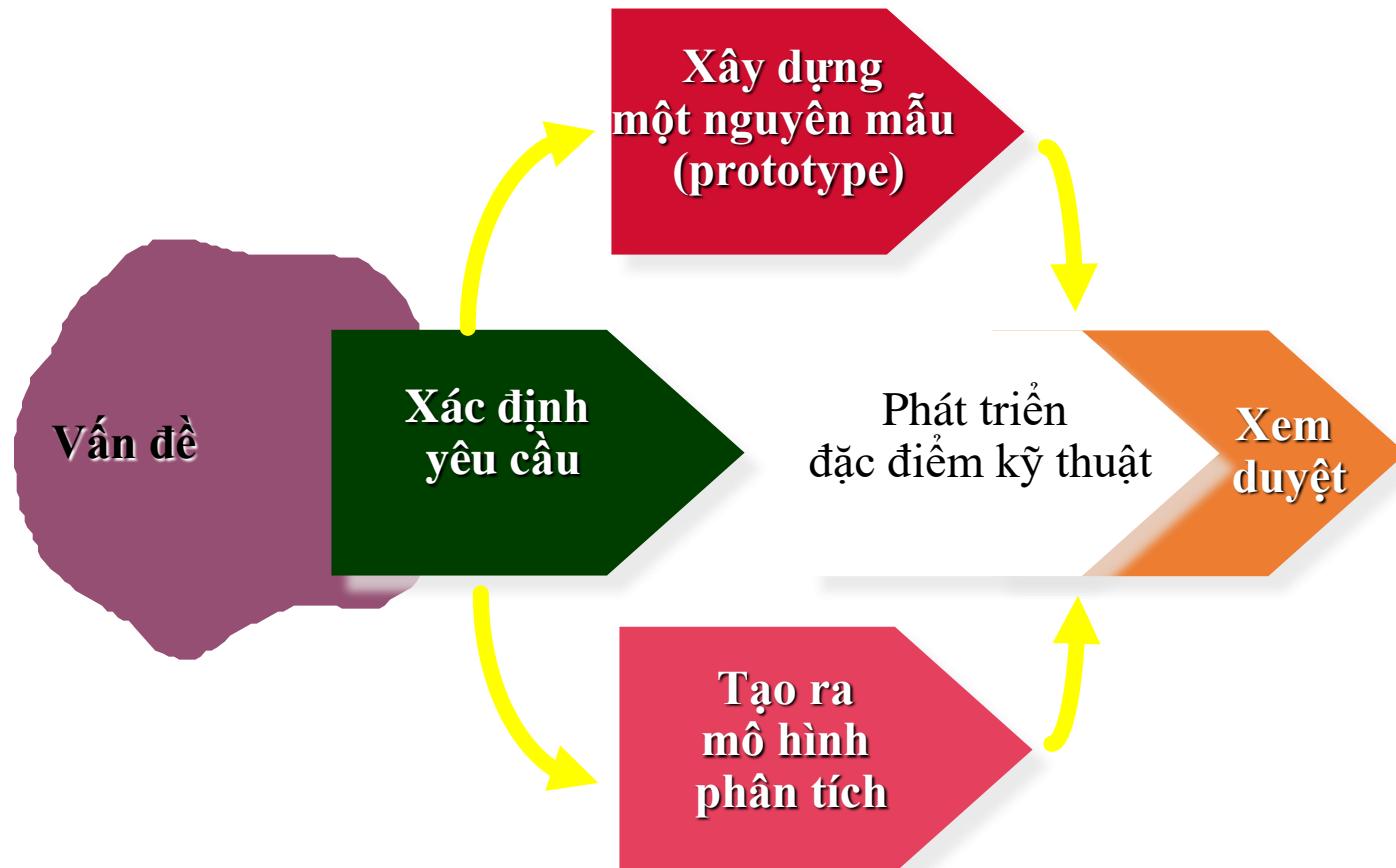
Nội dung

1. Khái niệm
2. Tầm quan trọng của yêu cầu phần mềm
3. Yêu cầu chức năng và yêu cầu phi chức năng
- 4. Các hoạt động chính trong kỹ nghệ yêu cầu phần mềm**

4. Các hoạt động chính trong kỹ nghệ YCPM

- **Phát hiện** các yêu cầu phần mềm (Requirements elicitation)
- **Phân tích** các yêu cầu phần mềm và thương lượng với khách hàng (Requirements analysis and negotiation)
- **Đặc tả** các yêu cầu phần mềm (Requirements specification)
- **Mô hình hóa** hệ thống (System modeling)
- **Kiểm tra tính hợp lý** của các yêu cầu phần mềm (Requirements validation)
- **Quản trị** các yêu cầu phần mềm (Requirements management)

4. Các hoạt động chính (tiếp)



4a. Phát hiện yêu cầu phần mềm

- **Đánh giá tính khả thi** về kỹ thuật và nghiệp vụ của phần mềm định phát triển
- Tìm kiếm các nhân sự (chuyên gia, người sử dụng) có những hiểu biết sâu sắc nhất, chi tiết nhất về hệ thống giúp chúng ta xác định yêu cầu phần mềm
- **Xác định môi trường** kỹ thuật trong đó sẽ triển khai phần mềm
- **Xác định các ràng buộc** về lĩnh vực ứng dụng của phần mềm (giới hạn về chức năng/hiệu năng phần mềm)

4a.Phát hiện yêu cầu phần mềm (tiếp)

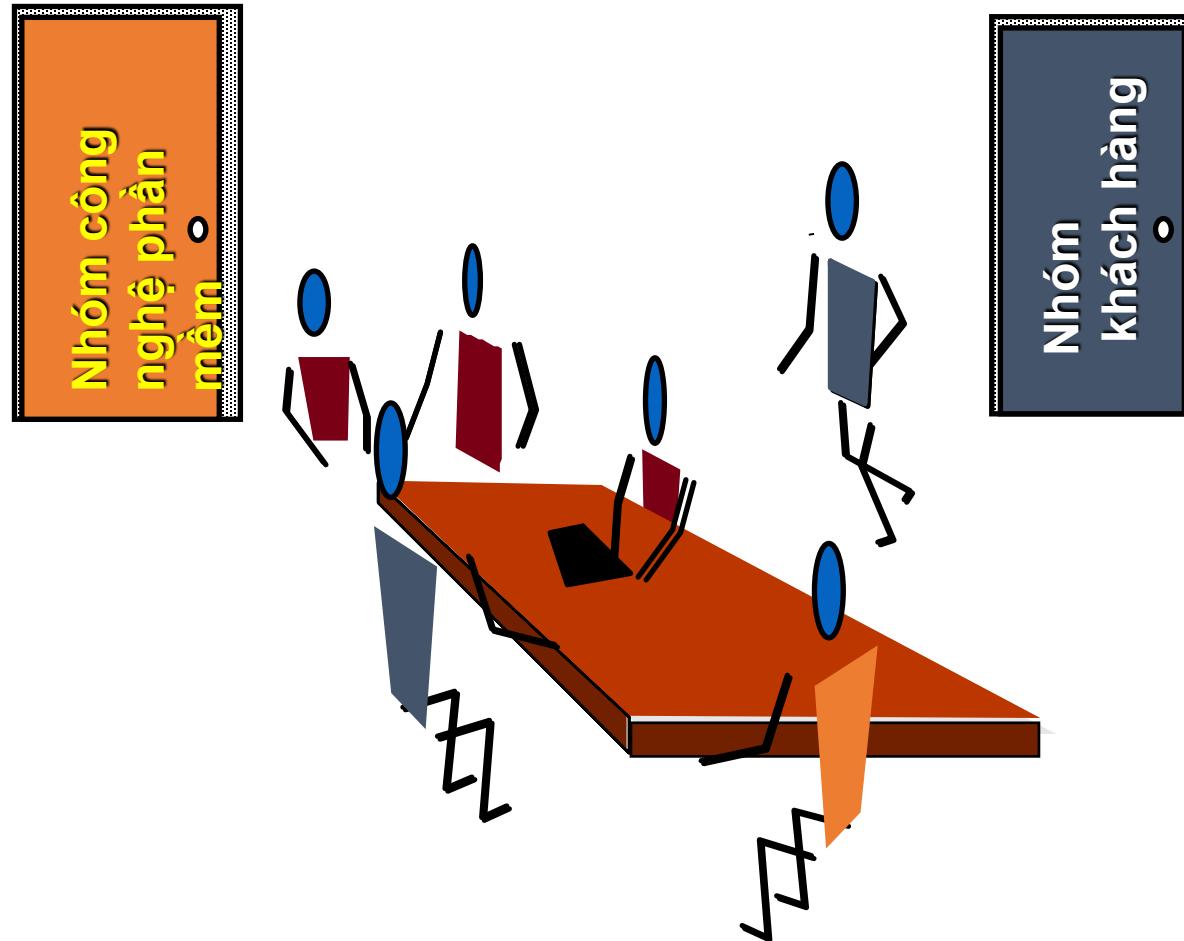
- Xác định các **phương pháp** sử dụng để phát hiện các yêu cầu phần mềm: phỏng vấn, làm việc nhóm, các buổi họp, gấp gỡ đối tác, v.v.
- Thu hút sự tham gia của nhiều chuyên gia, khách hàng để chúng ta có được các quan điểm xem xét phần mềm khác nhau từ phía khách hàng
- Xác định các yêu cầu còn nhập nhằng để làm mẫu thử
- **Thiết kế các kịch bản** sử dụng của phần mềm để giúp khách hàng định rõ các yêu cầu chính.

4a. Phát hiện yêu cầu phần mềm (tiếp)

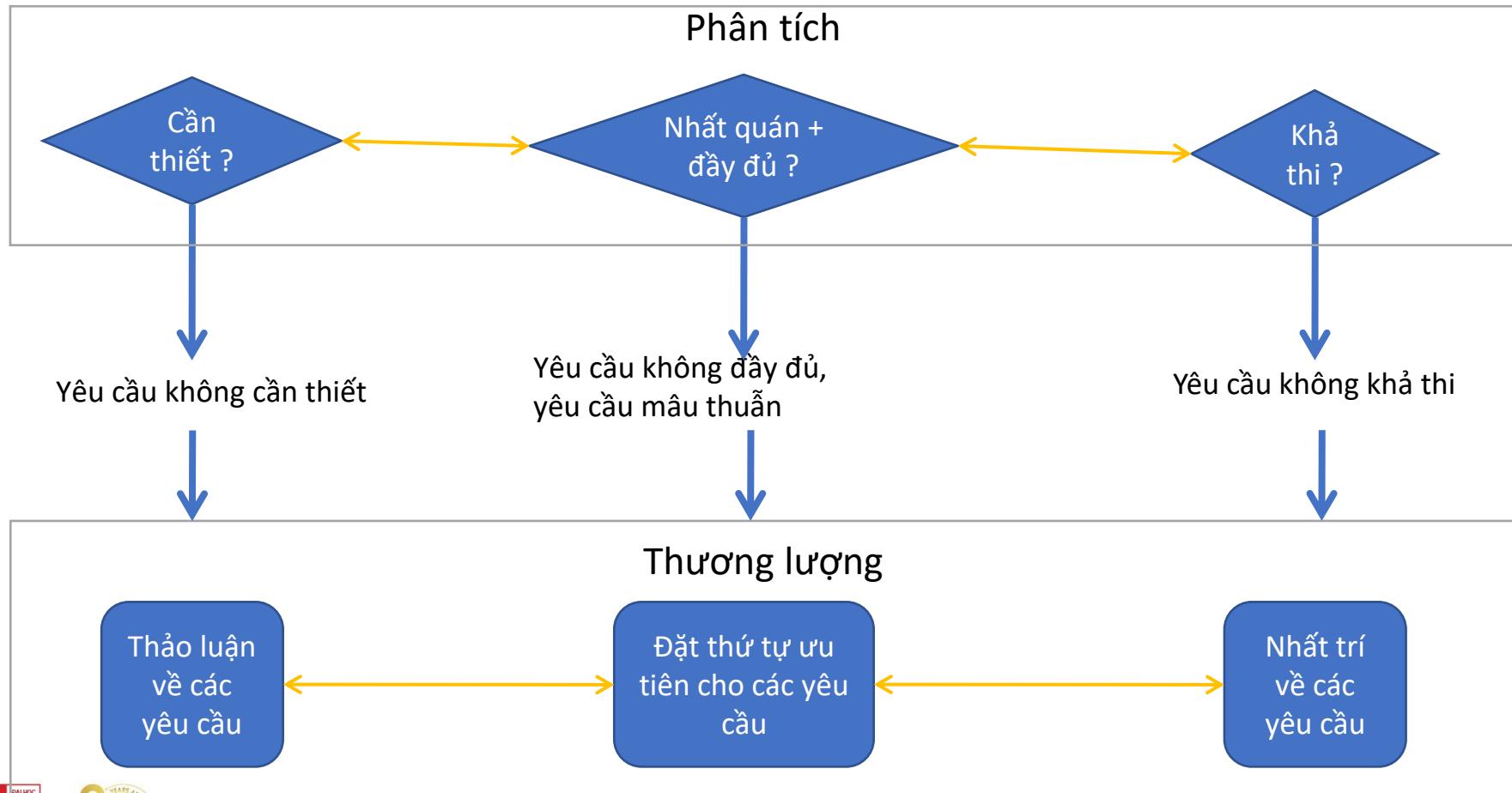
Đầu ra của bước phát hiện yêu cầu phần mềm

- Bảng kê (statement) các **đòi hỏi** và chức năng khả thi của phần mềm
- Bảng kê **phạm vi ứng dụng** của phần mềm
- Mô tả **môi trường kỹ thuật** của phần mềm
- Bảng kê tập hợp các **kịch bản sử dụng** của phần mềm
- Các **nguyên mẫu** xây dựng, phát triển hay sử dụng trong phần mềm (nếu có)
- Danh sách nhân sự tham gia vào quá trình phát hiện các yêu cầu phần mềm - kể cả các nhân sự từ phía công ty- khách hàng

4b. Phân tích các yêu cầu và thương lượng với khách hàng



4b. Phân tích các yêu cầu phần mềm và thương lượng với khách hàng (tiếp)



b. Phân tích các yêu cầu phần mềm và thương lượng với khách hàng (tiếp)

- **Phân loại** các yêu cầu phần mềm và sắp xếp chúng theo các nhóm liên quan
- **Khảo sát** tỉ mỉ từng yêu cầu phần mềm trong mối quan hệ của nó với các yêu cầu phần mềm khác
- **Thẩm định** từng yêu cầu phần mềm theo các tính chất: phù hợp, đầy đủ, rõ ràng, không trùng lặp
- **Phân cấp** các yêu cầu phần mềm theo dựa trên nhu cầu và đòi hỏi khách hàng / người sử dụng
- Thẩm định từng yêu cầu phần mềm để xác định:
 - Các yêu cầu PM có khả năng thực hiện được trong môi trường kỹ thuật hay không
 - Có khả năng kiểm định các yêu cầu phần mềm hay không

b.Phân tích các yêu cầu phần mềm và thương lượng với khách hàng (tiếp)

- **Thẩm định các rủi ro** có thể xảy ra với từng yêu cầu phần mềm
- **Đánh giá thô (tương đối)** về **giá thành** và **thời gian** thực hiện của từng yêu cầu phần mềm trong giá thành sản phẩm phần mềm và thời gian thực hiện phần mềm
- Giải quyết tất cả các bất đồng về yêu cầu phần mềm với khách hàng/người sử dụng trên cơ sở thảo luận và thương lượng các yêu cầu đề ra

4c. Đặc tả yêu cầu phần mềm

- **Đặc tả các yêu cầu phần mềm:** xây dựng các tài liệu đặc tả, trong đó có thể sử dụng tới các công cụ như: mô hình hóa, mô hình toán học hình thức (a formal mathematical model), tập hợp các kịch bản sử dụng, các nguyên mẫu hoặc bất kỳ một tổ hợp các công cụ nói trên
- Phương pháp đặc tả:
 - Đặc tả phi hình thức (Informal specifications): viết bằng ngôn ngữ tự nhiên
 - Đặc tả hình thức (Formal specifications): viết bằng tập các ký pháp có các quy định về cú pháp (syntax) và ngữ nghĩa (sematic) rất chặt chẽ, thí dụ ký pháp đồ họa dùng các lưu đồ.
- Tiêu chí đánh giá chất lượng của hồ sơ đặc tả:
 - Tính rõ ràng, chính xác
 - Tính phù hợp
 - Tính đầy đủ, hoàn thiện

Ví dụ: Các yêu cầu về hồ sơ đặc tả

- Đặc tả hành vi bên ngoài của HT
- Đặc tả các ràng buộc về cài đặt
- Dễ thay đổi
- Dùng như công cụ tham khảo cho bảo trì
- Sự ghi chép cẩn thận về vòng đời của HT, nghĩa là dự đoán các thay đổi
- Các đáp ứng với các sự cố không mong đợi

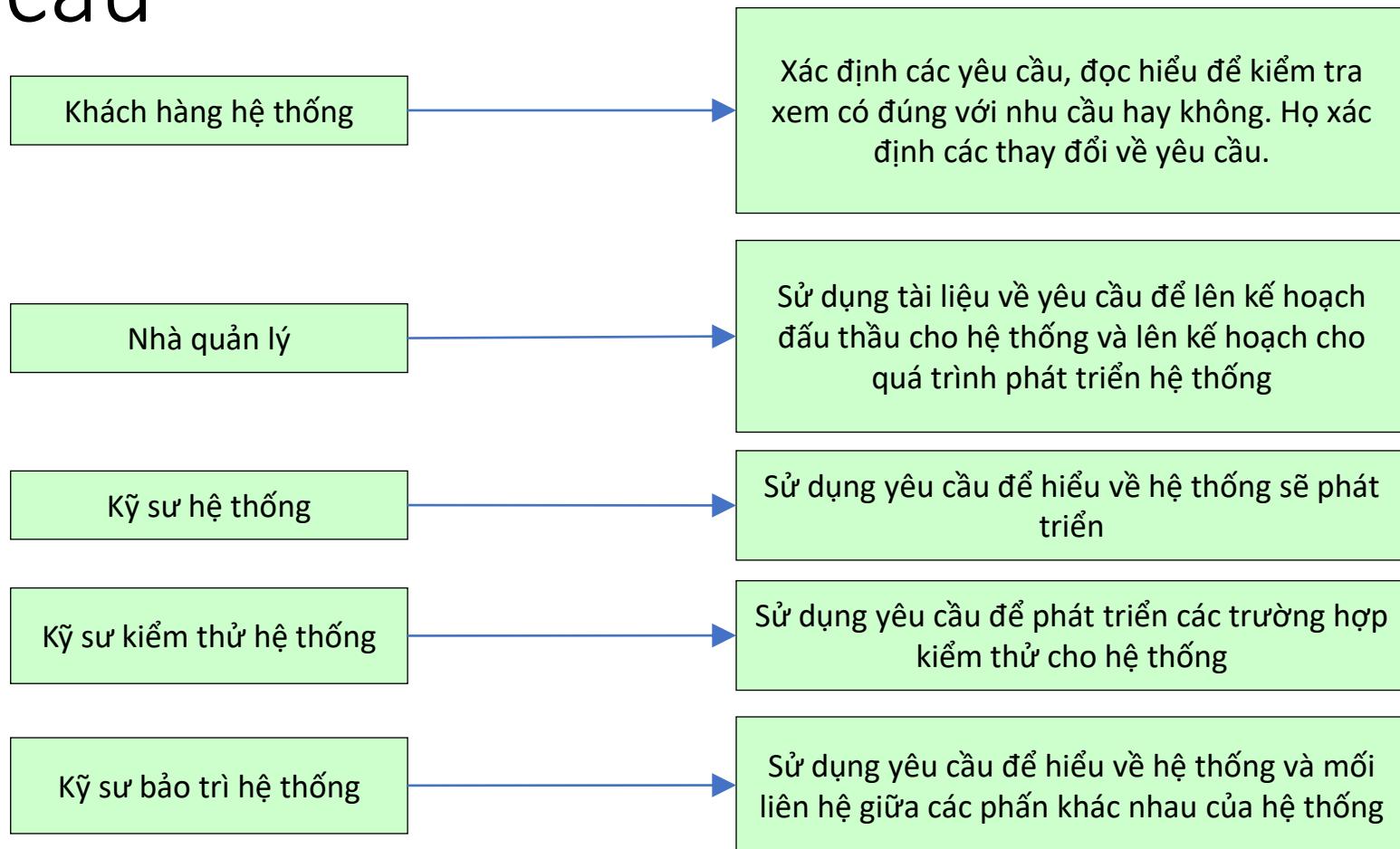
Các thành phần của hồ sơ đặc tả

- Đặc tả vận hành hay **đặc tả chức năng** (Operational specifications): mô tả các hoạt động của hệ thống phần mềm sẽ xây dựng:
 - Các dịch vụ mà hệ thống phải cung cấp
 - Hệ thống sẽ phản ứng với đầu vào cụ thể ra sao
 - Hành vi của hệ thống trong các tình huống đặc biệt.
- Đặc tả mô tả hay **đặc tả phi chức năng** (Descriptive specifications): đặc tả các đặc tính, đặc trưng của phần mềm:
 - Các ràng buộc về các dịch vụ hay các chức năng hệ thống, cung cấp như thời gian, ràng buộc về các quá trình phát triển, các chuẩn,...
- Ngoài ra còn có yêu cầu về lĩnh vực, bắt nguồn từ lĩnh vực của ứng dụng hệ thống và các đặc trưng của lĩnh vực này.

Tài liệu yêu cầu

- Tài liệu về yêu cầu là các phát biểu chính thức về cái được yêu cầu bởi các nhà phát triển hệ thống
- Nó bao gồm cả 2 phần:
 - **định nghĩa và đặc tả yêu cầu**
- Nó không phải là tài liệu thiết kế. Tốt hơn có thể nó chỉ là 1 tập các cái mà hệ thống phải làm hơn là hệ thống phải làm thế nào (phân tích chứ không phải là thiết kế)

Nội dung cần có của tài liệu yêu cầu

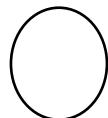


4d. Một số mô hình hóa hệ thống

- Biểu đồ phân cấp chức năng - WBS (work break down structure)
- Biểu đồ luồng dữ liệu – DFD (data flow diagram)
- Máy trạng thái – FSM (Finite state machine)
- Sơ đồ thực thể liên kết – ERD (entity relation diagram)

Ví dụ: ĐẶC TẢ CHỨC NĂNG VỚI DFD

- Hệ thống (System): tập hợp các dữ liệu (data) được xử lý bằng các chức năng tương ứng (functions)
- Các ký pháp sử dụng:



Thể hiện các chức năng (functions)



Thể hiện luồng dữ liệu



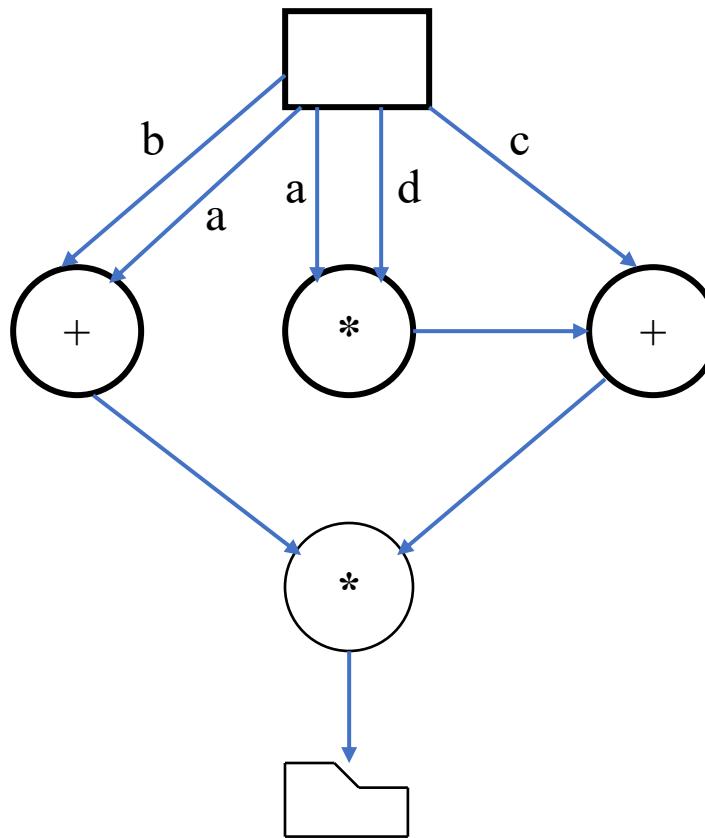
Kho dữ liệu



Vào ra dữ liệu và tương tác giữa
hệ thống và người sử dụng

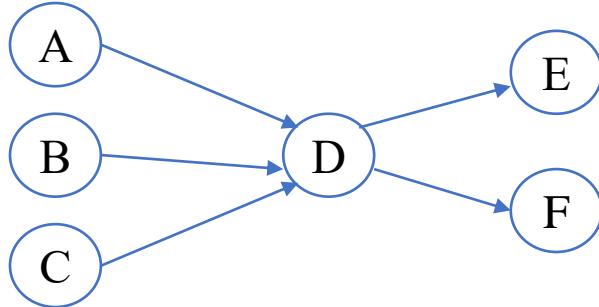
Ví dụ: mô tả biểu thức toán học bằng DFD

$$(a+b)^*(c+a*d)-e^*(a+b)$$



Các hạn chế của DFD

- Trong DFD không xác định rõ các hướng thực hiện (control aspects)

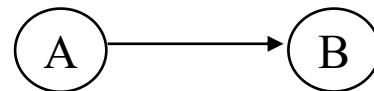


- Biểu đồ DFD này không chỉ rõ đầu vào là gì để thực hiện chức năng D và đầu ra là gì sau khi thực hiện chức năng D.

- Chức năng D có thể cần cả A, B và C
- Chức năng D có thể chỉ cần một trong A, B và C để thực hiện
- Chức năng D có thể kết xuất kết quả cho một trong E và F
- Chức năng D có thể kết xuất kết quả chung cho cả E và F
- Chức năng D có thể kết xuất kết quả riêng cho cả E và F

Các hạn chế của DFD

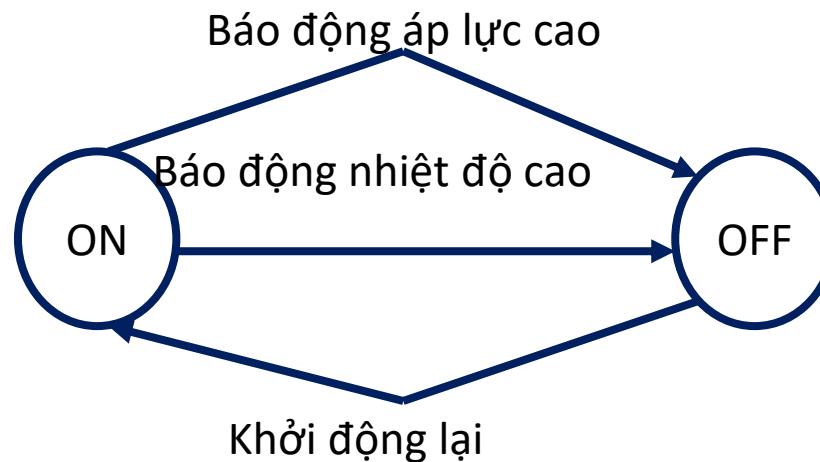
- DFD không xác định sự đồng bộ giữa các chức năng / mô-đun
 - A xử lý dữ liệu và B được hưởng (nhận) các kết quả được xử lý từ A
 - A và B là các chức năng không đồng bộ (asynchronous activities) vì thế cần có buffer để ngăn chặn tình trạng mất dữ liệu



Ví dụ: ĐẶC TẢ TRẠNG THÁI VỚI FSM - Finite State Machines

- FSM chứa
 - Tập hữu hạn các trạng thái Q
 - Tập hữu hạn các đầu vào I
 - Các chức năng chuyển tiếp

$$\delta : Q \times I \rightarrow Q$$



Ví dụ: Đặc tả dữ liệu với Mô hình thực thể liên kết -ERD

- Mô hình khái niệm cho phép đặc tả các yêu cầu logic của hệ thống, thường được sử dụng trong các hệ thống dữ liệu lớn
- ER Model
 - Thực thể
 - Quan hệ
 - Thuộc tính
- Biểu đồ thực thể

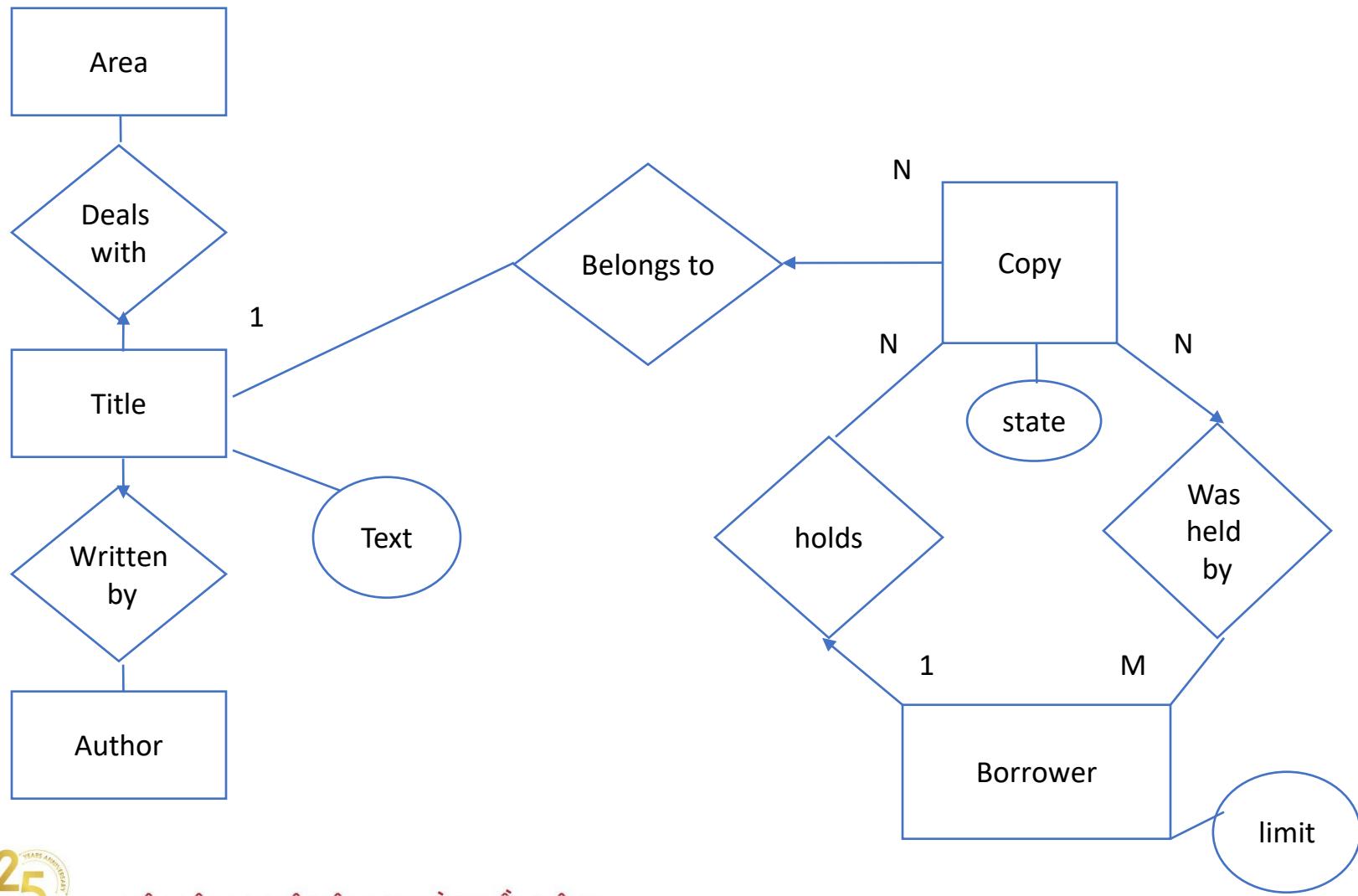
Thực thể

- Thực thể : tập hợp các thông tin liên quan cần được xử lý trong phần mềm
- Thực thể có thể có mối quan hệ:
 - người sở hữu ôtô



- Thực thể có các thuộc tính

Ví dụ: Erd mô tả thư viện



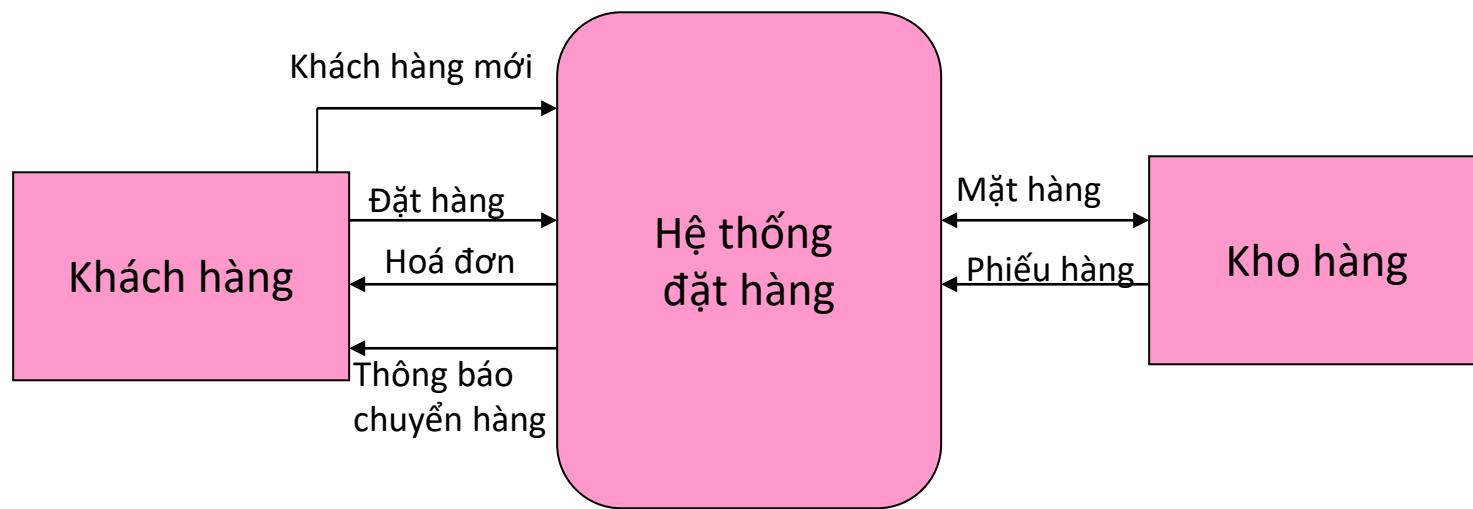
So sánh

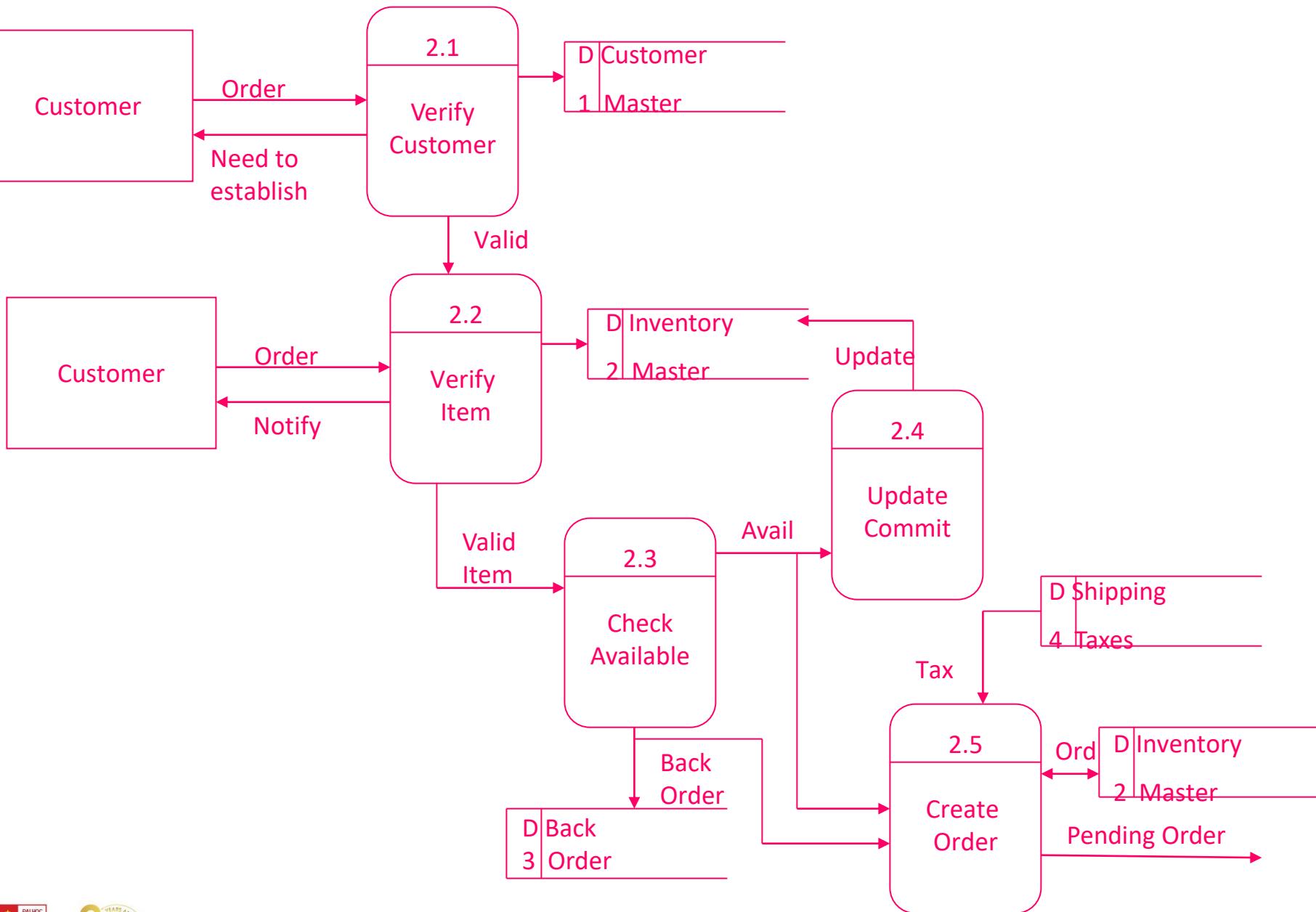
DFD	FSM	ERD
Đơn giản, dễ hiểu.	Có thể phức tạp với số lượng trạng thái lớn	Đơn giản, dễ hiểu
Mô tả luồng dữ liệu	Mô tả trạng thái của thực thể	Mô tả trừu tượng cơ sở dữ liệu
Không xác định rõ hướng thực hiện	Xác định rõ hướng thực hiện	Không xác định rõ hướng thực hiện
Không thể hiện tính tuần tự hay song song của tiến trình	Thể hiện tốt tính song song và tuần tự	Không thể hiện tính tuần tự hay song song

Thế nào là một đặc tả tốt?

- Dễ hiểu với người dùng
- Có ít điều nhập nhằng
- Có ít quy ước khi mô tả, có thể tạo đơn giản
- Với phong cách từ trên xuống (topdown)
- Dễ triển khai cho những pha sau của vòng đời:
 - thiết kế hệ thống, thiết kế chương trình và giao diện dễ làm, đảm bảo tính nhất quán, . . .

Thế nào là một đặc tả tốt?





4e. Quản trị các yêu cầu phần mềm

❑ Các công việc liên quan

- ✓ Quản lý thay đổi và vấn đề phát sinh
- ✓ Kiểm soát nguồn thay đổi tiềm năng

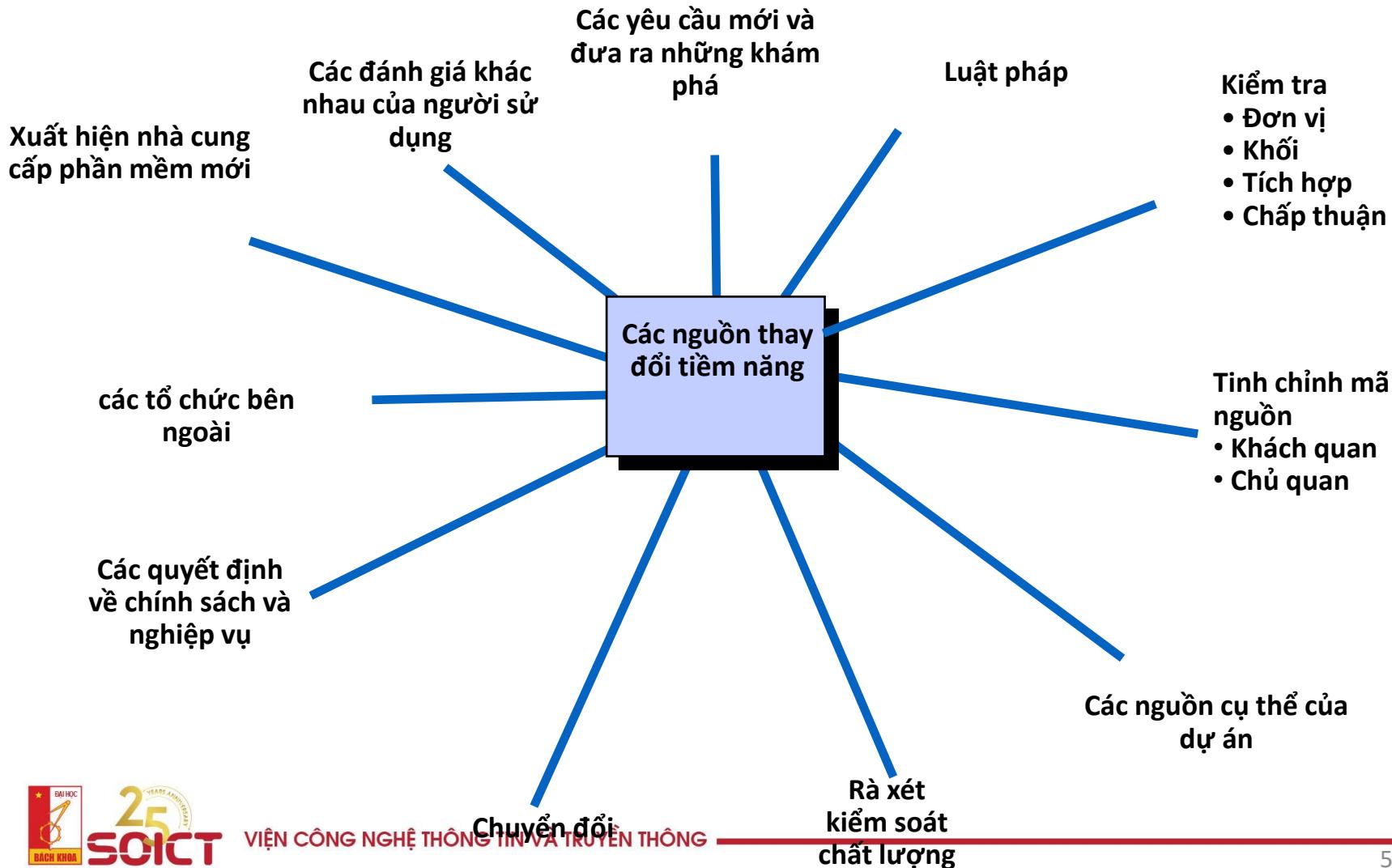
❑ Lợi ích:

- ✓ Chi phí phát triển thấp hơn trong suốt vòng đời phát triển phần mềm
- ✓ Ít sai sót hơn
- ✓ Giảm thiểu rủi ro đối với các sản phẩm quan trọng về an toàn
- ✓ Giao hàng nhanh hơn
- ✓ Khả năng tái sử dụng
- ✓ Truy xuất nguồn gốc
- ✓ Các yêu cầu gắn liền với các trường hợp thử nghiệm

Quản lý thay đổi và vấn đề phát sinh

- Thay đổi là gì ?
 - Bất cứ hoạt động nào thay đổi phạm vi, kết quả bàn giao, kiến trúc cơ bản, chi phí, lịch trình của một dự án
- Tại sao cần phải quản lý thay đổi và vấn đề phát sinh ?
 - Thay đổi và vấn đề phát sinh là 2 lý do thường làm dự án thất bại
- Làm thế nào để kiểm soát thay đổi và giải quyết các vấn đề phát sinh ?
 - Giảm rủi ro dự án nhờ quy trình hiệu quả quản lý thay đổi và vấn đề
 - Các thành viên nhóm hiểu được quy trình quản lý thay đổi và vấn đề
 - Ghi chép đầy đủ về các yêu cầu thay đổi/ vấn đề

Kiểm soát nguồn thay đổi tiềm năng



Tổng kết

1. Sau bài học, sinh viên đã nắm được các kỹ năng:
 - Các khái niệm liên quan tới kỹ nghệ YCPM
 - Các hoạt động chính trong kỹ nghệ YCPM
 - Một số mô hình thường dùng trong quá trình đặc tả YCPM
2. Trong bài tiếp theo sẽ giới thiệu tổng quan về các mô hình trong kỹ nghệ YCPM

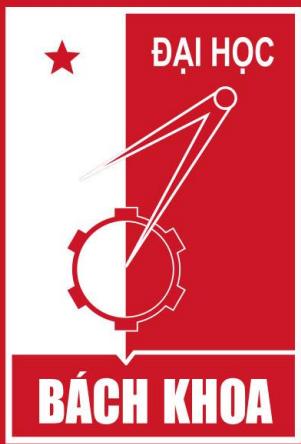


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The central focus of the banner is a large, white, stylized number "25" with a circular arc above it containing the text "YEARS ANNIVERSARY". To the right of the "25" is the acronym "SOKT" in a bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 6_2

Mô hình các yêu cầu phần mềm

Mục tiêu của bài học

Sinh viên sẽ được trang bị các kiến thức sau:

- Các khái niệm liên quan tới mô hình yêu cầu phần mềm
- Giới thiệu về một số mô hình yêu cầu phần mềm
- Áp dụng mô hình yêu cầu phần mềm trong ứng dụng WebApps

Nội dung

1. Tổng quan – phân loại
2. Một số mô hình thực tế
3. Mô hình cho mô hình các yêu cầu
4. Mô hình yêu cầu cho ứng dụng WebApps

1. Tổng quan - phân loại

- Mô hình yêu cầu phần mềm được gọi là *structured analysis (phân tích có cấu trúc)*: xem xét, phân tích yêu cầu theo dữ liệu giữa các tiến trình và sự thay đổi dữ liệu như các thực thể riêng biệt.
 - Data objects (dữ liệu đối tượng) được mô hình bằng cách định nghĩa các trạng thái và quan hệ giữa chúng.
 - Các tiến trình thao tác đối với dữ liệu đối tượng được mô hình để cho thấy làm thế nào biến đổi dữ liệu như dữ liệu đối tượng thông qua hệ thống.
- Mô hình yêu cầu phần mềm được gọi là object-oriented analysis (*phân tích hướng đối tượng*), tập trung vào:
 - Định nghĩa các lớp
 - Cách thức mà chúng liên kết với nhau để thực hiện các yêu cầu của khách hàng.

2. Một số mô hình thực tế

a. Flow-Oriented Modeling – Biểu đồ luồng

- Trình diễn làm thế nào dữ liệu đối tượng được biến đổi thông qua hệ thống **data flow diagram** (biểu đồ luồng dữ liệu) (DFD)
- Được nhiều người nhìn nhận là một cách tiếp cận theo trường phái cũ, tuy nhiên nó vẫn tiếp cung cấp một khía cạnh của hệ thống nên vẫn được sử dụng để bổ sung cho các mô hình phân tích khác.

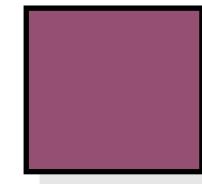
The Flow Model

- Mỗi hệ thống máy tính cơ bản là một hệ thống biến đổi thông tin

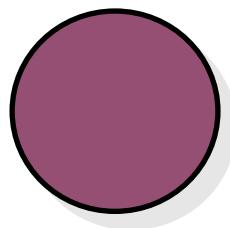


Các ký hiệu mô hình luồng Dữ Liệu

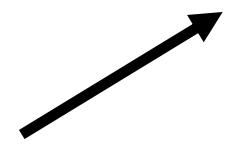
- Mỗi hệ thống máy tính cơ bản là một hệ thống biến đổi thông tin



Thực thể bên ngoài



Tiến trình



Luồng dữ liệu



Kho dữ liệu

Thực thể ngoài

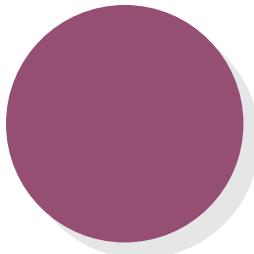
Một sản xuất hoặc tiêu thụ dữ liệu

Ví dụ: một người, một thiết bị, một cảm biến...

Ví dụ khác: hệ thống máy tính cơ bản

*Dữ liệu luôn luôn phải xuất phát từ một nơi nào đó
và luôn luôn phải được gửi đến một cái gì đó.*

Tiến trình

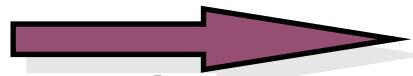


Một sự biến đổi dữ liệu (thay đổi đầu vào qua đầu ra)

Ví dụ: tính thuế, xác định diện tích,
Định dạng báo cáo, hiển thị đồ thị

Dữ liệu phải luôn luôn được xử lý bằng cách nào đó để đạt được chức năng của hệ thống

Luồng dữ liệu

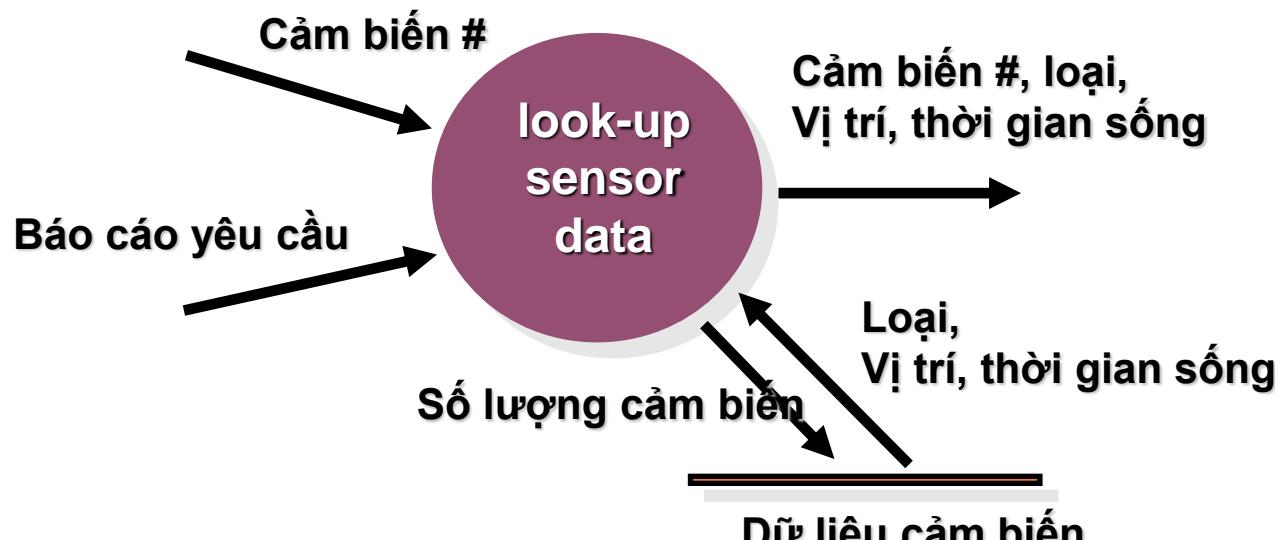


Luồng dữ liệu thông qua một hệ thống, bắt đầu như đầu vào và biến đổi thành đầu ra.



Các kho dữ liệu

Dữ liệu thường được lưu cho lần sử dụng sau



Biểu đồ luồng dữ liệu: hướng dẫn

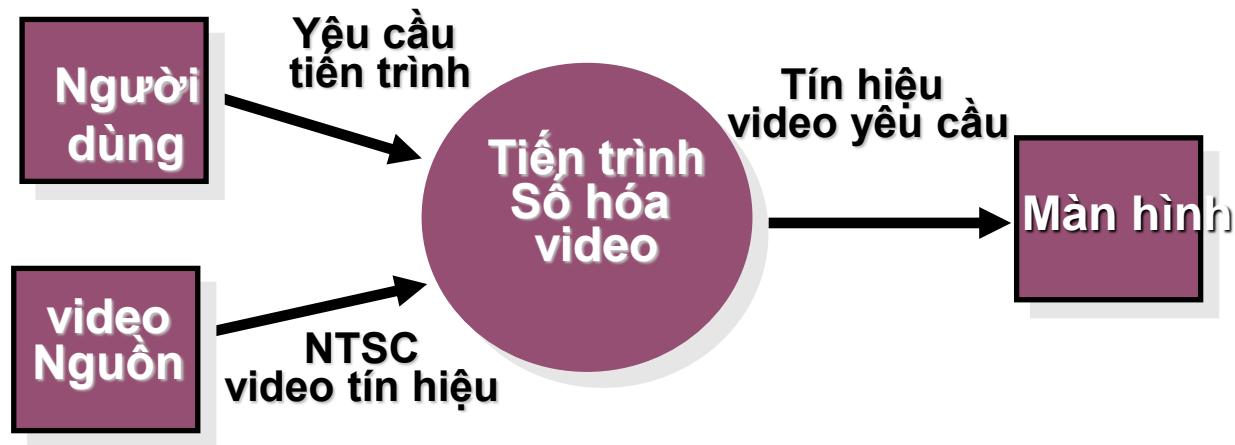
- Tất cả các biểu tượng phải được gán nhãn có nghĩa đầy đủ
- Biểu đồ DFD phát triển thông qua số lượng mức các chi tiết
- Luôn luôn bắt đầu với một mức nội dung (gọi là level 0)
- Luôn luôn chỉ ra các thực thể ngoài tại mức 0
- Luôn luôn gán nhãn cho hướng luồng dữ liệu
- Không thể hiện các thủ tục logic

Ví dụ: hướng dẫn xây dựng một DFD

- Xem xét kịch bản người dùng và/hoặc để tách biệt các dữ liệu đối tượng và sử dụng một phân tích ngũ pháp để xác định “hoạt động”
- Xác định thực thể ngoài (các sản xuất và tiêu thụ của dữ liệu)
- Tạo mức 0 DFD



Ví dụ mức 0 DFD

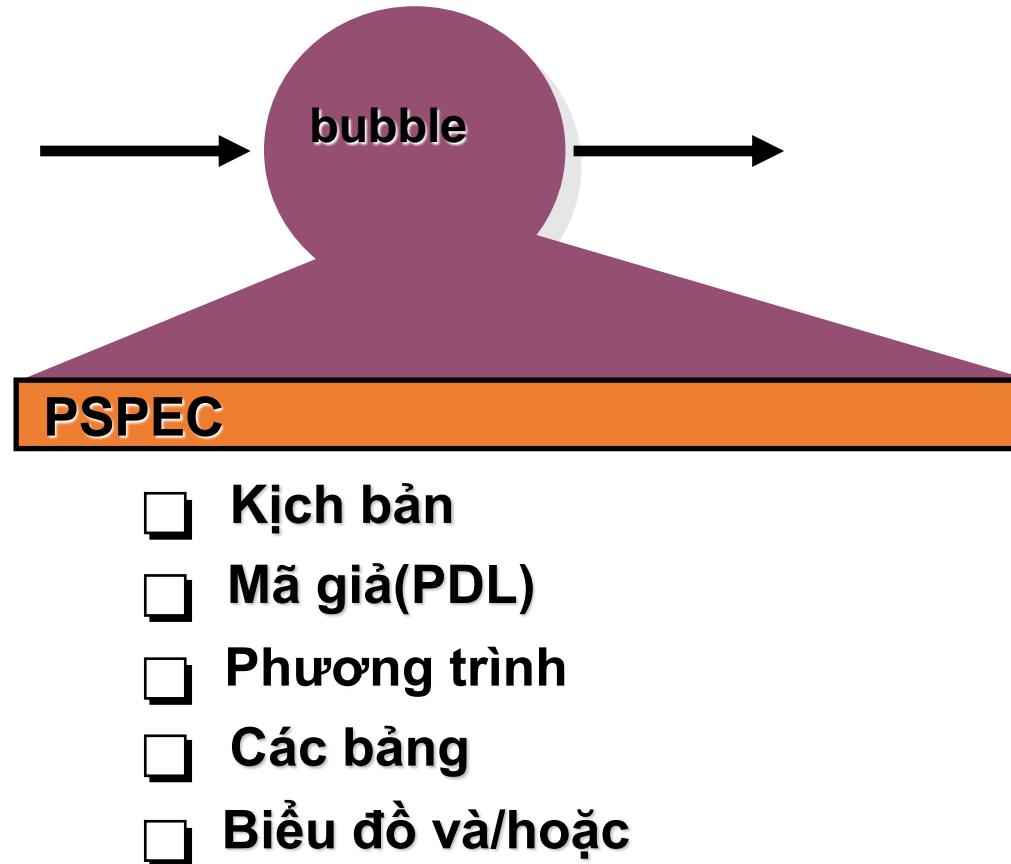


Các ghi chú

- Mỗi ô tròn là đơn giá trị cho đến khi nó làm gì đấy
- Tỷ lệ mở rộng giảm khi số lượng mức tăng
- Phần lớn hệ thống yêu cầu giữa 3 và 7 cấp cho một mô hình đầy đủ luồng
- Một mục lưu lượng dữ liệu duy nhất (mũi tên) có thể được mở rộng như cấp độ tăng (dữ liệu từ điển cung cấp thông tin)

2. Một số mô hình thực tế

b. Đặc tả tiến trình (PSPEC)



Mô hình luồng Điều khiển

- Biểu diễn các “**sự kiện**” và các tiến trình mà quản lý các sự kiện
- Một “**sự kiện**” là một điều kiện đúng/sai mà có thể chắc chắn bởi:
 - Nghe tất cả các cảm biến mà “đọc” bởi phần mềm
 - Nghe tất cả các ngắt điều kiện.
 - Nghe tất cả các “ngắt” mà được dẫn động bởi hệ thống.
 - Nghe tất cả các dữ liệu điều kiện.
 - Nhắc lại phân tích danh từ/động từ được áp dụng cho kịch bản sản xuất, xem xét tất cả "kiểm soát mục" như có thể CSPEC đầu vào/đầu ra..

Đặc tả Điều khiển (CSPEC)

CSPEC có thể là:

- Biểu đồ trạng thái
(biểu đồ tuần tự)
 - Bảng chuyển đổi
trạng thái
 - Bảng quyết định
 - Bảng kích hoạt
- 
- Tổ hợp*

2. Một số mô hình thực tế

c. Mô hình hành vi

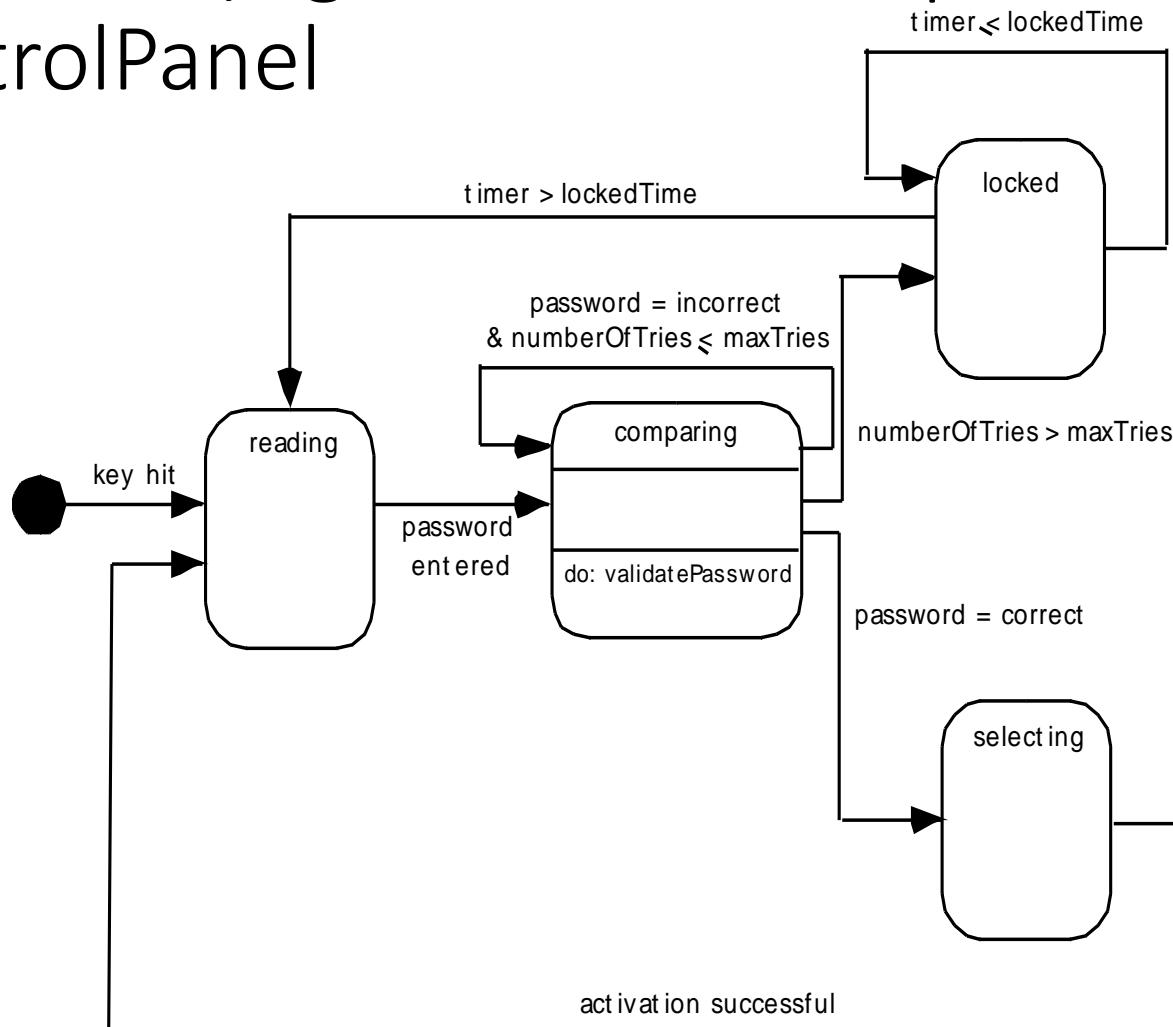
- Mô hình hành vi chỉ ra cách phần mềm sẽ phản ứng với các sự kiện hoặc kích thích bên ngoài (ví dụ biểu đồ User-case). Để tạo ra các mô hình, phải thực hiện theo các bước sau:
 - Đánh giá tất cả users-case để hiểu đầy đủ trình tự của các tương tác trong hệ thống.
 - Xác định sự kiện điều khiển dây tương tác và hiểu làm thế nào những sự kiện liên quan đến các đối tượng cụ thể.
 - Tạo ra chuỗi cho mỗi use-case.
 - Xây dựng một biểu đồ trạng thái cho hệ thống.
 - Xem lại các mô hình hành vi để xác minh tính chính xác và nhất quán.

2. Một số mô hình thực tế

Biểu diễn trạng thái

- Trong bối cảnh của mô hình hành vi, 2 đặc điểm khác nhau của trạng thái phải được xem xét:
 - Trạng thái của mỗi lớp
 - trạng thái của hệ thống (quan sát từ bên ngoài hệ thống thực hiện chức năng của nó)
- Trạng thái của một lớp có cả hai đặc tính chủ động và thụ động.
 - Một trạng thái thụ động chỉ đơn giản là tình trạng hiện tại của tất cả các thuộc tính của một đối tượng.
 - Các trạng thái hoạt động của một đối tượng cho biết tình trạng hiện tại của đối tượng như nó trải qua một sự biến đổi hoặc qua hoạt động của 1 tiến trình.

Biểu đồ trạng thái cho các lớp ControlPanel



Các trạng thái của một hệ thống

- **Trạng thái**—1 tập của các tình huống mà đặc trưng cho hành vi của 1 hệ thống tại một thời điểm nhất định
- **Chuyển đổi trạng thái**—sự di chuyển từ một trạng thái tới trạng thái khác.
- **Sự kiện**—Một biến cố mà nhờ đó hệ thống có thể dự đoán được hành vi
- **Hành động**—tiến trình mà xuất hiện như một chuỗi các biến đổi

Ví dụ: Mô hình hành vi

- Tạo ra một danh sách của các trạng thái khác nhau của hệ thống (làm thế nào để hệ thống xử lý các hành vi?)
- Chỉ ra cách hệ thống làm cho một trạng thái chuyển đổi từ một này sang một trạng thái khác (Làm thế nào hệ thống chuyển đổi trạng thái?)
 - Chỉ ra sự kiện
 - Chỉ ra hành động
- Vẽ một biểu đồ trạng thái hoặc một biểu đồ trình tự

Biểu đồ trình tự

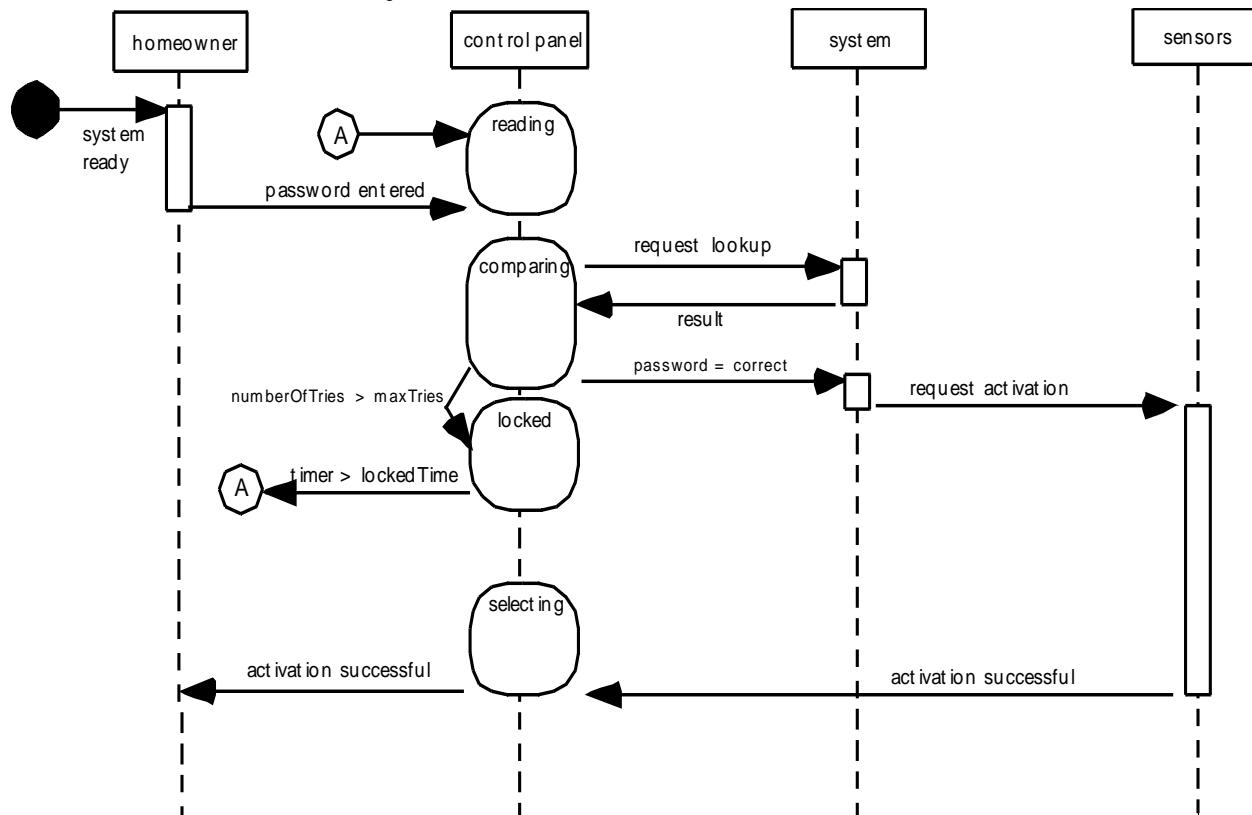


Figure 8.27 Sequence diagram (partial) for SafeHome security function

Viết các thông số phần mềm



3. Mô hình cho mô hình các yêu cầu

- Mô hình phần mềm là một cơ chế để thu nhận tri thức mà cho phép nó được áp dụng lại khi xuất hiện một vấn đề mới
 - Miền kiến thức có thể được áp dụng cho một vấn đề mới trong cùng miền ứng dụng .
 - các miền kiến thức được lưu lại bởi một mô hình có thể được áp dụng bằng cách tương tự vào một miền ứng dụng khác mà hoàn toàn khác nhau.
- Ban đầu, các tác giả của một mô hình phân tích không "tạo ra" các mô hình, nhưng sau đó, họ phát hiện ra đó là công việc yêu cầu kỹ thuật cần được tiến hành.
- Khi mô hình đã được khảo sát, nó đã được chứng thực

Khảo sát mô hình phân tích

- Các yếu tố cơ bản nhất trong mô tả của một mô hình yêu cầu là các use-case.
- Một tập hợp chặt chẽ các use-case có thể phục vụ là cơ sở cho việc phát hiện một hoặc nhiều các mẫu phân tích thêm.
- Một mô hình phân tích ngữ nghĩa (SAP) "là một mô hình mô tả một tập hợp nhỏ các use-case mạch lạc theo cùng mô tả một ứng dụng cơ bản."

Một ví dụ

- Hãy xem xét các use-case sơ bộ sau đây cho phần mềm cần thiết để kiểm soát và giám sát một máy ảnh thực sự xem và cảm biến khoảng cách cho một chiếc điện thoại tự động :

Use case: Theo dõi chuyển động ngược

Mô tả: Khi chiếc xe được đặt tại số lùi, các phần mềm điều khiển cho phép một nguồn cấp dữ liệu video từ một máy quay phim phía sau được đặt để hiển thị bảng điều khiển. Phần mềm điều khiển tính toán các khoảng cách và định hướng trên màn hình hiển thị để các nhà điều hành xe có thể duy trì định hướng khi xe di chuyển theo chiều ngược lại. Phần mềm điều khiển cũng theo dõi một cảm biến để xác định xem một đối tượng nằm trong 10 feet phía sau của chiếc xe. Nó sẽ tự động dừng chiếc xe nếu các cảm biến khoảng cách chỉ ra một đối tượng trong 3 feet phía sau của xe.

Một ví dụ

- Use-case này với một loạt các chức năng đó sẽ được phân tích và xây dựng (thành một bộ mạch lạc của use-case) trong quá trình thu thập yêu cầu và mô hình hóa.
- Bất kể bao nhiêu xây dựng được thực hiện, use-case(s) đề nghị một SAP-giám sát dựa trên phần mềm đơn giản, chưa áp dụng rộng rãi và kiểm soát các cảm biến và cơ cấu chấp hành trong một hệ thống vật lý.
- Trong trường hợp này, các cảm biến cung cấp thông tin xung quanh khoảng cách và thông tin video. Các "thiết bị truyền động" là hệ thống dừng của xe (gọi nếu một đối tượng là rất gần với xe).
- Nhưng trong trường hợp tổng quát hơn, một mô hình được áp dụng rộng rãi được phát hiện -> (Thiết bị truyền động-cảm biến) **Actuator-Sensor**

4. Mô hình yêu cầu cho ứng dụng WebApps

- **Phân tích nội dung:** đầy đủ các nội dung được cung cấp bởi các WebApp (ứng dụng Web) được xác định, bao gồm cả văn bản, đồ họa và hình ảnh, video, âm thanh và dữ liệu. Mô hình hóa dữ liệu có thể được sử dụng để xác định và mô tả một trong các dữ liệu đối tượng.
- **Phân tích Tương tác:** Cách thức mà người dùng tương tác với các WebApp được mô tả chi tiết. Biểu đồ ca sử dụng(Use-Case) có thể được phát triển để cung cấp mô tả chi tiết của tương tác này.
- **Phân tích chức năng:** Các use-case được tạo ra như một phần của phân tích tương tác xác định các hoạt động mà sẽ được áp dụng cho nội dung WebApp và bao hàm các chức năng xử lý khác. Tất cả các hoạt động và chức năng được mô tả chi tiết.
- **Phân tích cấu hình:** Môi trường và cơ sở hạ tầng, trong đó vị trí WebApp được mô tả chi tiết.

Khi nào chúng ta tiến hành phân tích?

- Trong một số tình huống phân tích và thiết kế hợp nhất. Tuy nhiên, **một phân tích hoạt động rõ ràng xuất hiện khi:**
 - các WebApp sẽ được xây dựng nhiều và/hoặc phức tạp.
 - số lượng các bên liên quan là lớn
 - số lượng kỹ sư Web và đóng góp khác quá lớn
 - các mục tiêu và đối tượng (được xác định trong quá trình lập) cho các WebApp sẽ ảnh hưởng đến “kinh doanh”
 - sự thành công của các WebApp sẽ có một ảnh hưởng mạnh mẽ đến sự thành công của doanh nghiệp

Mô hình nội dung

- **Đối tượng nội dung** được lấy từ biểu đồ ca sử dụng
 - kiểm tra các mô tả kịch bản để tham khảo trực tiếp và gián tiếp đến nội dung
- **Các trạng thái** của một đối tượng nội dung được định nghĩa
- Các mối quan hệ giữa các đối tượng nội dung và/hoặc, hệ thống phân cấp của nội dung được duy trì bởi một WebApp
 - Sơ đồ mối quan hệ thực thể-mối quan hệ hoặc UML
 - Cây phân cấp dữ liệu hoặc UML

Cây dữ liệu

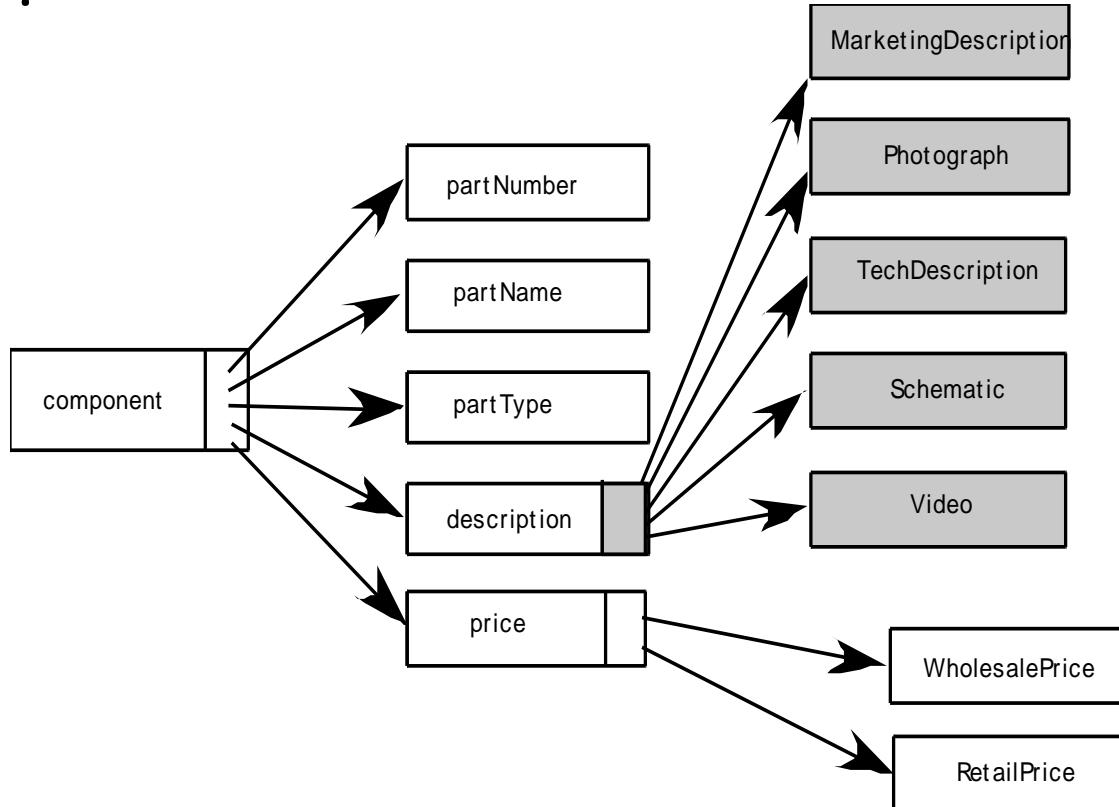


Figure 18.3 Data tree for aSafeHome component

Mô hình tương tác

- Bốn nguyên tố cấu thành:
 - use-cases(biểu đồ ca sử dụng)
 - sequence diagrams(biểu đồ trình tự)
 - state diagrams(biểu đồ trạng thái)
 - a user interface prototype(giao diện người dùng)
- Mỗi trong số này là một nguyên tố UML quan trọng và được mô tả trong Phụ lục I

Biểu đồ trình tự

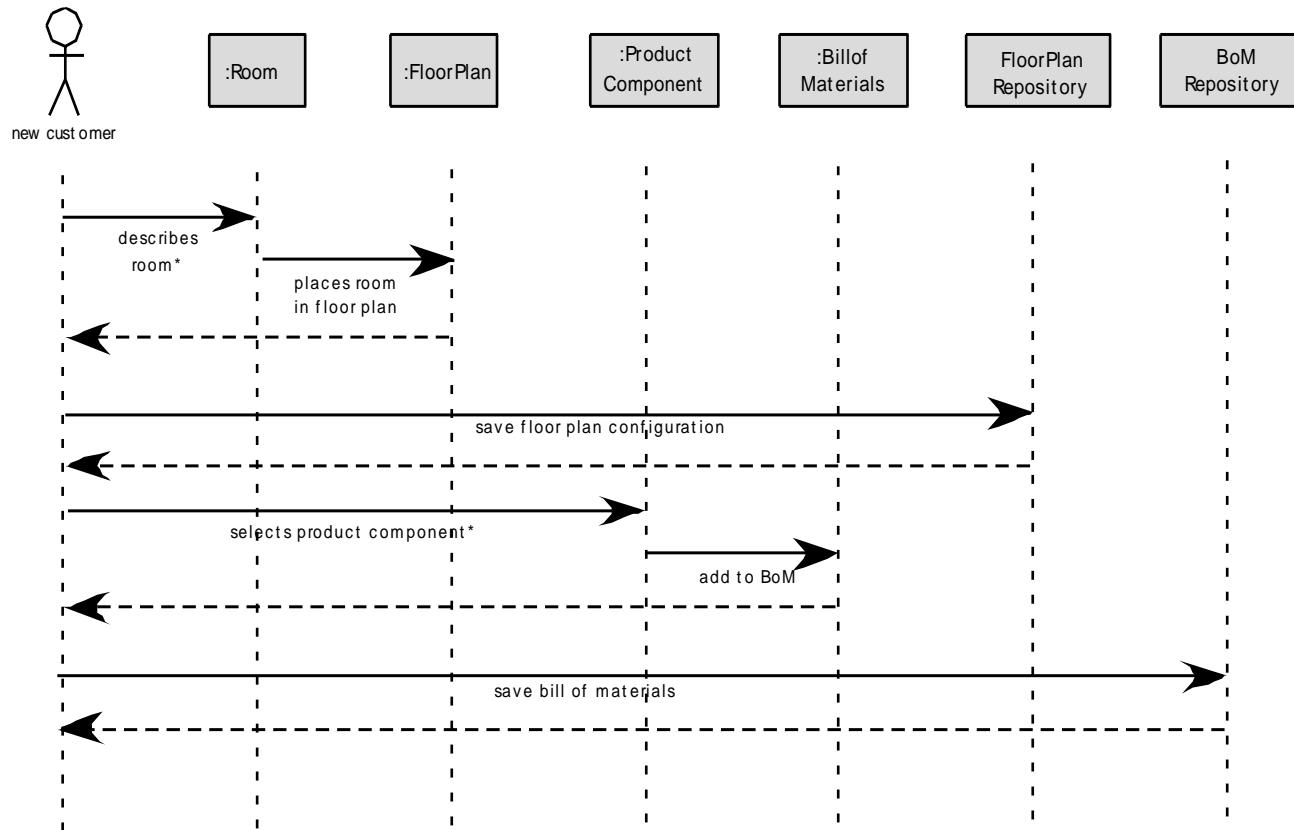


Figure 18.5 Sequence diagram for use-case:select SafeHome components

Biểu đồ trạng thái

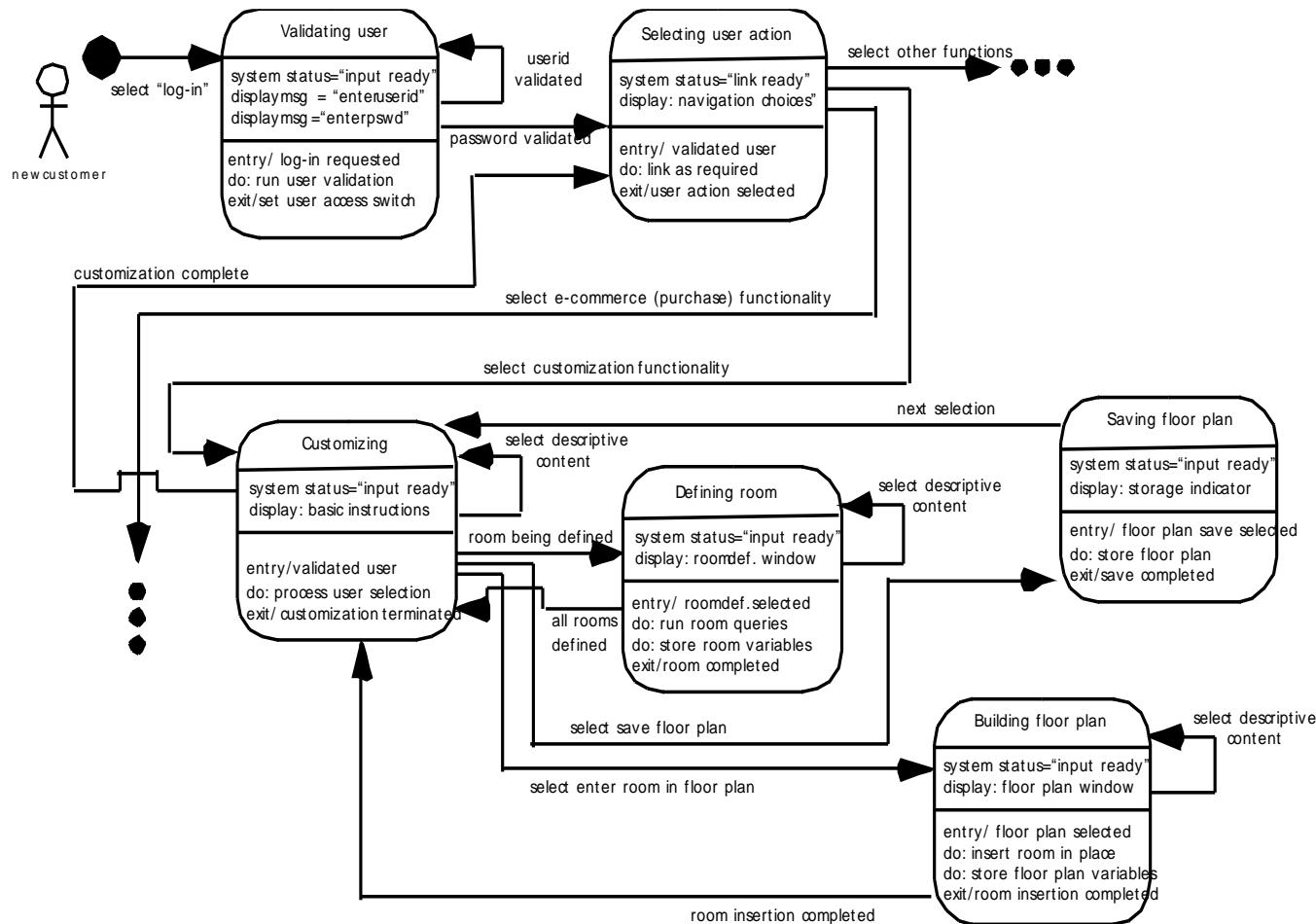


Figure 18.6 Partial state diagram for **new customer** interaction

Mô hình chức năng

- Các mô hình chức năng giải quyết hai yếu tố xử lý của WebApp
 - Khảo sát chức năng người dùng được phân phối bởi các WebApp cho người dùng cuối
 - các hàm trong lớp thực hiện hành vi kết hợp với lớp.
- Một biểu đồ hoạt động có thể được sử dụng để biểu diễn luồng tiến trình



Biểu đồ hoạt động

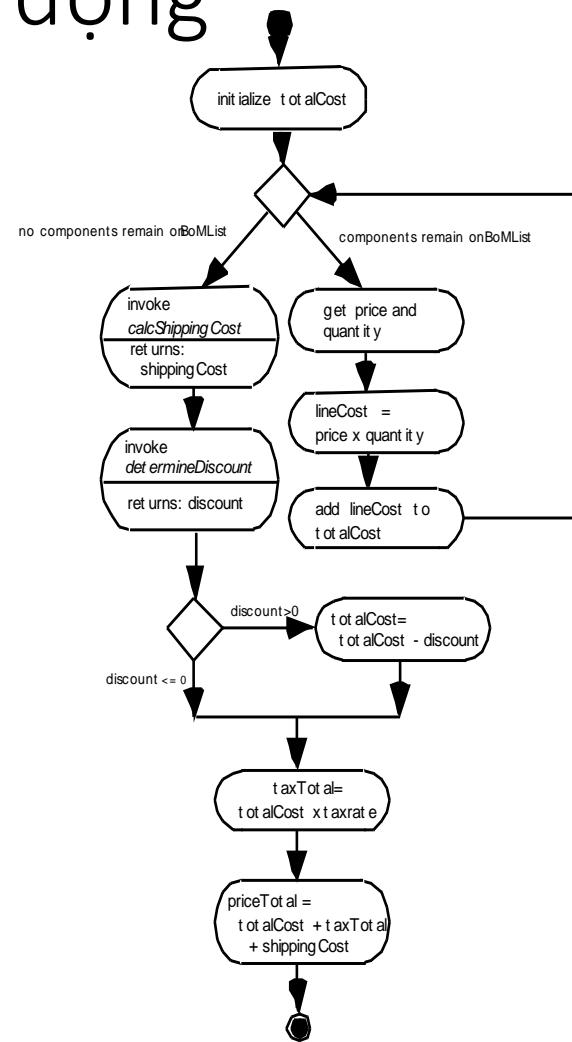


Figure 18.7 Activity diagram for `computePriceOperative`

Mô hình cấu hình

- Máy chủ
 - Phần cứng máy chủ và môi trường hệ điều hành phải được xác định
 - Cân nhắc khả năng tương tác trên phía máy chủ phải được xem xét
 - Giao diện thích hợp, giao thức truyền thông và thông tin hợp tác liên quan phải được quy định
- Khách hàng(client)
 - Vấn đề cấu hình trình duyệt phải được xác định
 - Yêu cầu thử nghiệm cần được xác định

Mô hình điều hướng

- Đối với các yếu tố/trạng thái dễ dàng để đạt được (yêu cầu các bước chuyển hướng ít hơn) những người khác? Ưu tiên cho các trình diễn là gì?
- Một số yếu tố cần được nhấn mạnh để buộc người dùng điều hướng theo hướng của họ?
- Làm thế nào lỗi điều hướng cần được xử lý?
- Chuyển hướng tới các nhóm có liên quan cần được ưu tiên hơn hướng đến một yếu tố cụ thể.
- Chuyển hướng nên được thực hiện thông qua các liên kết, thông qua truy cập dựa trên tìm kiếm, hoặc bằng một số phương tiện khác?
- Các yếu tố có nhất định nên giới thiệu đến người dùng dựa vào bối cảnh của hành động chuyển hướng trước?



Một bản ghi chuyển hướng nên được duy trì cho người sử dụng?

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Mô hình điều hướng

- Nên một bản đồ chuyển hướng đầy đủ hoặc thực đơn có sẵn ở mọi điểm trong sự tương tác của người dùng?
- Thiết kế chuyển hướng nên được thúc đẩy bởi các hành vi sử dụng dự kiến phổ biến nhất hoặc bởi tầm quan trọng của các yếu tố nhận thức WebApp ?
- Có thể sử dụng một "lưu trữ" chuyển hướng trước đó thông qua các WebApp để tiến hành sử dụng trong tương lai?
- Nhóm người dùng điều hướng tối ưu nên được thiết kế?
- Làm thế nào kết nối các đường link bên ngoài để các WebApp được xử lý? Chồng lên các cửa sổ trình duyệt hiện hành? Mở một cửa sổ trình duyệt mới? Mở một khung riêng biệt?

Tài liệu tham khảo

- Slide Set to accompany Software Engineering: A Practitioner's Approach, 7/e by Roger S. Pressman
- <http://bit.ly/2ihOLB4>

Tổng kết

Sau buổi học, sinh viên đã hiểu được:

- Các mô hình đặc tả yêu cầu phần mềm.
- Biết cách sử dụng các mô hình trong quá trình đặc tả yêu cầu
- Hiểu được mô hình thông qua một ví dụ về xác định yêu cầu phần mềm của hệ thống WebApps

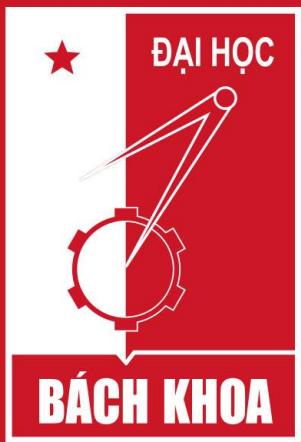


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The graphic consists of the number "25" in large white digits. A curved banner arches over the top of the "2", containing the text "YEARS ANNIVERSARY". Below the "25" is the acronym "SOKT" in a large, bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 7

Thiết kế phần mềm

Mục tiêu của bài học

Sinh viên sẽ được trang bị các kiến thức sau:

- Các khái niệm liên quan tới Thiết kế phần mềm
- Giới thiệu về hai giai đoạn thiết kế phần mềm: thiết kế kiến trúc và thiết kế chi tiết
- Phân biệt tính móc nối (Coupling) và tính kết dính (Cohesion) trong thiết kế phần mềm

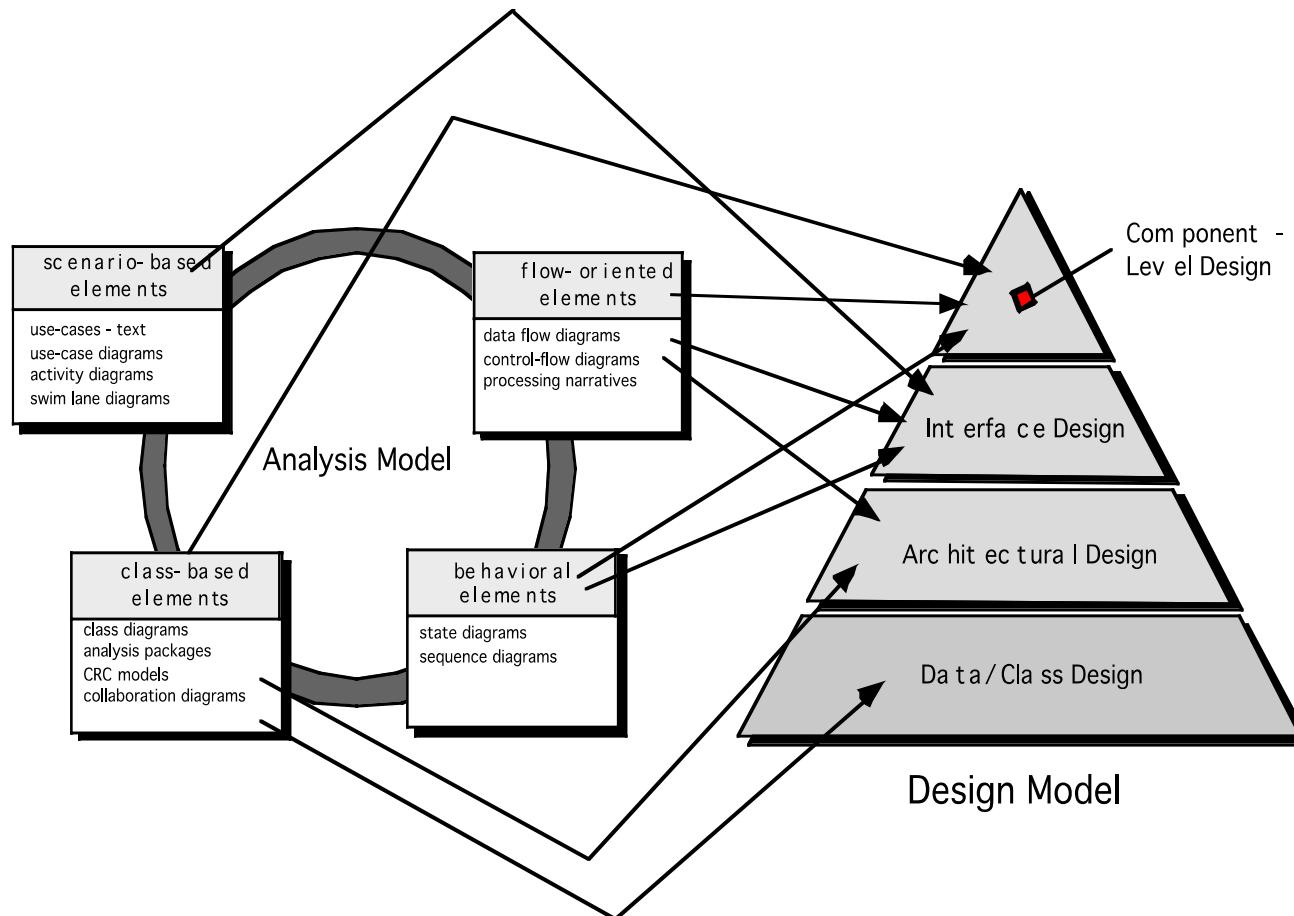
Nội dung

- 1. Tổng quan thiết kế phần mềm**
2. Các khái niệm trong thiết kế phần mềm
3. Thiết kế kiến trúc phần mềm
4. Thiết kế chi tiết phần mềm
5. Tính móc nối (Coupling) và tính kết dính (Cohesion)

1.1 Khái niệm về thiết kế phần mềm

- Thiết kế tạo ra các biểu diễn và dữ kiện của hệ thống phần mềm cần xây dựng từ kết quả phân tích yêu cầu để có thể tiếp tục thực hiện giai đoạn tiếp theo trong tiến trình phát triển phần mềm.
- Kết quả của pha thiết kế là mô hình thiết kế của phần mềm
 - Mô hình thiết kế đủ chi tiết để giai đoạn lập trình có thể thực hiện
 - Là phương tiện để trao đổi thông tin và đảm bảo chất lượng
 - Mô hình thiết kế dễ sửa đổi hơn mã chương trình, cung cấp cái nhìn tổng thể đồng thời có nhiều mức chi tiết

Mô hình phân tích-> Mô hình thiết kế



These slides are designed to accompany Software Engineering:
A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

1.1 Khái niệm về thiết kế phần mềm

- Thiết kế xuất phát từ kết quả phân tích yêu cầu, giúp trả lời câu hỏi “Như thế nào?”
 - Mô tả một hoặc nhiều giải pháp, giúp đánh giá các giải pháp, lựa chọn giải pháp tốt nhất
- Thiết kế thường mô tả ở một mức trừu tượng nhất định, sử dụng các mô hình (khác với cài đặt chi tiết khi lập trình)
- Nếu không có thiết kế hoặc thiết kế tồi:
→ làm tăng công sức viết mã chương trình, tăng công sức bảo trì, khó khăn khi cần sửa đổi, mở rộng

1.1 Khái niệm về thiết kế phần mềm

- Theo Mitch Kapor, người đã tạo ra Lotus 1-2-3, giới thiệu trong “Tuyên ngôn về thiết kế phần mềm” trên Dr. Dobbs Journal. Thiết kế phần mềm tốt nên thể hiện:
 - **Sự ổn định (Firmness):** Một chương trình không nên có bất cứ lỗi nào làm hạn chế chức năng của nó.
 - **Tiện nghi (Commodity):** Một chương trình nên phù hợp với mục đích đã định của nó .
 - **Sự hài lòng (Delight):** Trải nghiệm sử dụng chương trình nên làm hài lòng người dùng.

1.1 Khái niệm về thiết kế phần mềm

- Các giai đoạn thiết kế: hoạt động thiết kế xuất hiện trong các mô hình phát triển phần mềm khác nhau, gồm hai giai đoạn thiết kế chính:
 - Thiết kế kiến trúc / Thiết kế mức cao (high level design)
 - Mô hình tổng thể của hệ thống
 - Cách thức hệ thống được phân rã thành các mô đun
 - Mối quan hệ giữa các môđun
 - Cách thức trao đổi thông tin giữa các môđun
 - Thực hiện bởi nhiều mức trừu tượng
 - Thiết kế chi tiết / Thiết kế mức thấp (low level design)
 - Thiết kế chi tiết lớp, module, thiết kế thuật toán, thiết kế dữ liệu, thiết kế giao diện,...

1.2 Quan hệ giữa thiết kế và chất lượng

- Thiết kế phải thực hiện tất cả các yêu cầu rõ ràng chứa trong mô hình phân tích, và nó phải đáp ứng tất cả các yêu cầu tiềm ẩn khách hàng mong muốn.
- Thiết kế phải là một hướng dẫn dễ hiểu dễ đọc cho những người tạo ra code và cho những người kiểm thử và sau đó hỗ trợ cho phần mềm.
- Thiết kế nên cung cấp một bức tranh hoàn chỉnh của phần mềm, giải quyết vấn đề dữ liệu, chức năng, và hành vi từ một quan điểm thực thi.

1.2 Quan hệ giữa thiết kế và chất lượng

- **Một thiết kế nên thể hiện một kiến trúc mà** (1) đã được tạo ra bằng cách sử dụng phong cách kiến trúc hoặc các pattern được công nhận , (2) bao gồm các thành phần mang những đặc tính thiết kế tốt và (3) có thể được thực hiện một cách tiến hóa
 - Đối với các hệ thống nhỏ hơn, thiết kế đôi khi có thể được phát triển tuyến tính.
- **Một thiết kế nên môđun hoá:** đó là, phần mềm nên được phân chia thành các thành phần hợp lý hoặc hệ thống con
- **Một thiết kế cần có biểu diễn riêng biệt** của dữ liệu, kiến trúc, giao diện, và các thành phần.

1.3 Nguyên tắc Chất lượng

- Một thiết kế nên dẫn đến các cấu trúc dữ liệu thích hợp cho các lớp sẽ được thực thi và được rút ra từ mô hình dữ liệu có thể nhận biết.
- Một thiết kế nên dẫn đến các thành phần mang những đặc tính chức năng độc lập.
- Một thiết kế nên dẫn đến giao diện mà giảm sự phức tạp của các kết nối giữa các thành phần và với môi trường bên ngoài
- Một thiết kế nên được chuyển hóa bằng cách sử dụng một phương pháp lặp lại được dẫn dắt bởi các thông tin thu được trong quá trình phân tích các yêu cầu phần mềm.
- Một thiết kế nên được đại diện bằng một ký hiệu truyền đạt hiệu quả ý nghĩa của nó.

1.4 Nguyên tắc Thiết kế

- Quá trình thiết kế không nên mắc phải 'tunnel vision.'
- Việc thiết kế nên có thể truy ngược về mô hình phân tích.
- Việc thiết kế không nên 'phát minh lại bánh xe'.
- Việc thiết kế nên "giảm thiểu khoảng cách trí tuệ" [DAV95] giữa phần mềm và bài toán như nó tồn tại trong thế giới thực.
- Việc thiết kế nên biểu lộ tính đồng nhất và tích hợp.

1.4 Nguyên tắc Thiết kế

- Việc thiết kế nên được cấu trúc để thích ứng với thay đổi.
- Việc thiết kế nên được cấu trúc để làm suy thoái (degrade) nhẹ nhàng, ngay cả khi đang gấp phải dữ liệu bất thường, các sự kiện, hoặc điều kiện hoạt động .
- Thiết kế không phải là coding, coding không phải là thiết kế.
- Việc thiết kế nên được đánh giá về chất lượng khi nó được tạo ra, chứ không phải sau thực tế.
- Việc thiết kế cần được xem xét để giảm thiểu lỗi khái niệm (ngữ nghĩa).

Nội dung

1. Tổng quan thiết kế phần mềm
- 2. Các khái niệm trong thiết kế phần mềm**
3. Thiết kế kiến trúc phần mềm
4. Thiết kế chi tiết phần mềm
5. Tính móc nối (Coupling) và tính kết dính (Cohesion)

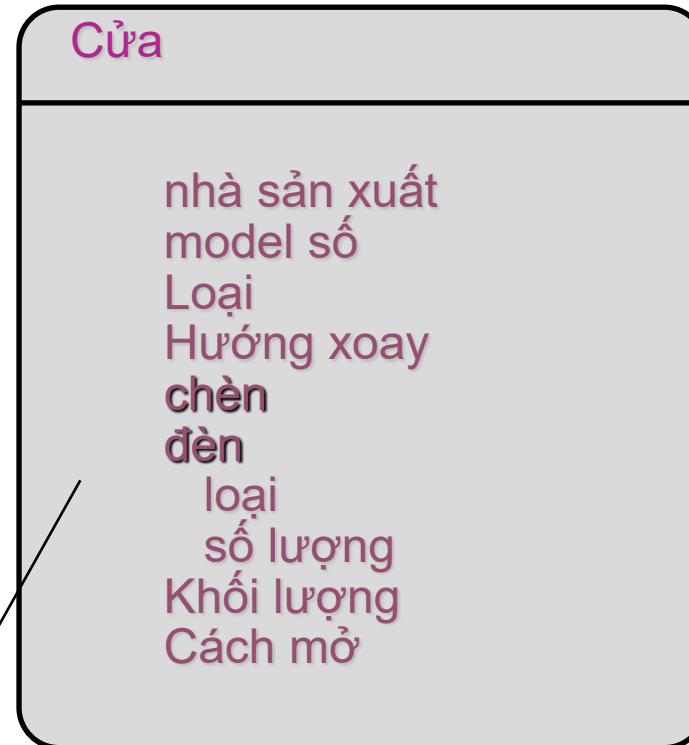
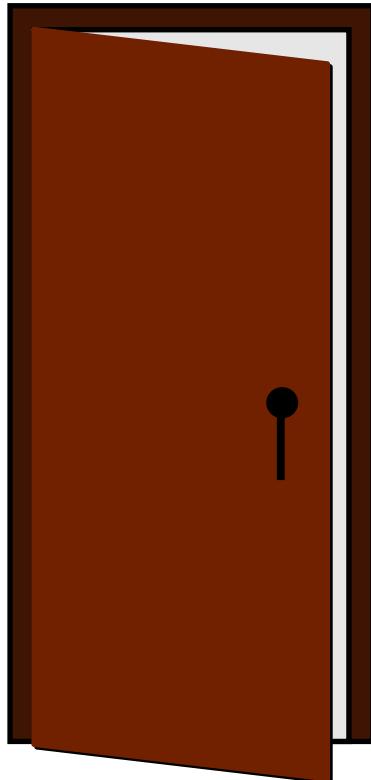
2.1 Các khái niệm cơ bản trong thiết kế phần mềm

- **Trừu tượng hoá**—dữ liệu, thủ tục, kiểm soát
- **Kiến trúc**—cấu trúc tổng thể của phần mềm
- **Mẫu thiết kế (Design Patterns)**—"chuyển tải những tinh túy" của một giải pháp thiết kế đã được chứng minh
- **Phân tách mối quan tâm (Separation of concerns)**—bất kỳ vấn đề phức tạp có thể được xử lý dễ dàng hơn nếu nó được chia thành nhiều mảnh
 - Bằng cách tách mối quan tâm ra nhỏ hơn, và các mảnh dễ quản lý hơn, một vấn đề mất ít hơn công sức và thời gian để giải quyết.
- **Mô đun hoá (Modularity)**—mô đun hoá các dữ liệu và chức năng
- **Tính ẩn**—giao diện được điều khiển

2.1 Các khái niệm cơ bản trong thiết kế phần mềm (2)

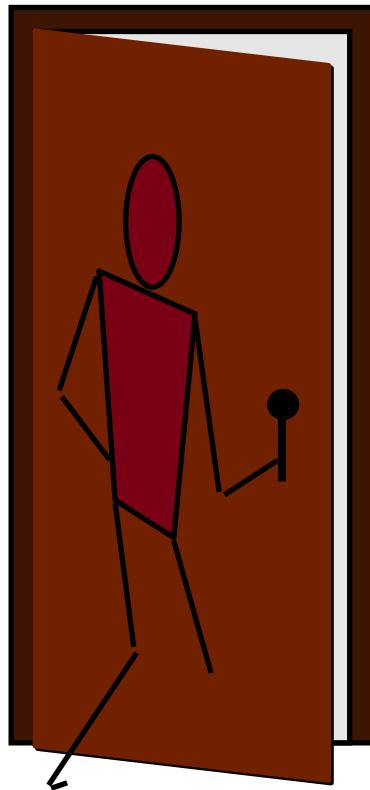
- **Độc lập chức năng** — Các hàm đơn mục đích và khớp nối thấp
- **Sàng lọc**—xây dựng chi tiết cho tất cả các khái niệm trùu tượng
- **Các khía cạnh**—một cơ chế cho sự hiểu biết các yêu cầu tổng thể ảnh hưởng đến thiết kế như thế nào
- **Refactoring**— một kỹ thuật tái tổ chức nhằm đơn giản hóa thiết kế
- **OO design concepts**—các khái niệm về thiết kế hướng đối tượng
- **Thiết kế Lớp**—cung cấp chi tiết thiết kế mà sẽ cho phép các lớp đã phân tích được thực thi

Ví dụ: Trừu tượng dữ liệu



Thực thi như một cấu trúc dữ liệu

Ví dụ: Trừu tượng thủ tục



Mở

chi tiết về
Thuật toán vào cửa

thực hiện với "kiến thức" về các
đối tượng được kết hợp với vào cửa

Khuôn mẫu thiết kế (Design Patterns)

- Mỗi chúng ta đều đã trải qua vấn đề thiết kế và từng nghĩ: **Liệu
đã có ai phát triển một lời giải cho vấn đề này chưa?**
 - Điều gì sẽ xảy ra nếu đã có một cách tiêu chuẩn để mô tả một vấn
đề (để bạn có thể nhìn nhận nó), và một phương pháp có tổ chức
để trình bày lời giải cho vấn đề đó?
- **Khuôn mẫu thiết kế** là một phương pháp có hệ thống để mô tả
các vấn đề và giải pháp, cho phép các cộng đồng công nghệ
phần mềm có thể nắm bắt kiến thức thiết kế theo cách có thể
tái sử dụng

Khuôn mẫu thiết kế (Design Patterns)

- *Mỗi khuôn mẫu mô tả một vấn đề xảy ra hết lần này đến lần khác trong môi trường của chúng ta và sau đó mô tả cốt lõi của giải pháp cho vấn đề đó theo một cách mà bạn có thể sử dụng nó hàng triệu lần mà không bao giờ phải làm lại một việc lần thứ hai.*

Christopher Alexander, 1977

- “một quy tắc ba phần diễn tả một mối quan hệ giữa một ngũ cảnh, một vấn đề, và một giải pháp.”

Các loại khuôn mẫu

- **Architectural patterns** Mẫu kiến trúc mô tả các vấn đề thiết kế trên diện rộng được giải quyết bằng cách tiếp cận cấu trúc.
- **Data patterns** Mẫu dữ liệu mô tả vấn đề dữ liệu và các giải pháp mô hình dữ liệu có thể được dùng để giải quyết vấn đề trên
- **Component patterns** Mẫu thành phần (còn gọi là các mẫu thiết kế) nhằm đến các vấn đề liên quan với việc phát triển các hệ thống con và các thành phần, cách thức chúng giao tiếp với nhau, và vị trí của chúng trong một kiến trúc lớn hơn
- **Interface design patterns**. Mẫu thiết kế giao diện mô tả các vấn đề giao diện người dùng thông thường và giải pháp bao gồm các đặc trưng cụ thể của người dùng cuối.
- **WebApp patterns**. Mẫu WebApp giải quyết tập hợp vấn đề có thể gặp phải khi xây dựng Ứng dụng web và thường kết hợp nhiều mô hình khác nhau đến ở trên.

Các loại khuôn mẫu (2)

- **Khuôn mẫu khởi tạo (Creational patterns)** tập trung vào việc khởi tạo, sáng tác và biểu diễn của các đối tượng, ví dụ:
 - Abstract factory pattern
 - Factory method pattern
- **Khuôn mẫu cấu trúc (Structural patterns)** tập trung vào các vấn đề và các giải pháp liên quan đến cách các lớp và các đối tượng được tổ chức và tích hợp để xây dựng một cấu trúc lớn hơn, ví dụ như:
 - Adapter pattern
 - Aggregate pattern
- **Khuôn mẫu hành vi (Behavioral patterns)** giải quyết các vấn đề liên quan đến việc phân công trách nhiệm giữa các đối tượng và cách thức mà giao tiếp được thực hiện giữa các đối tượng, ví dụ như:
 - Chain of responsibility pattern:
 - Command pattern:

Ví dụ: thông tin của một mẫu thiết kế

Bản mẫu thiết kế

Tên Pattern—mô tả bản chất của mô hình trong một tên ngắn nhưng ý nghĩa

Intent (ý định)—mô tả các mô hình và những gì nó làm

Also-known-as—liệt kê các từ đồng nghĩa cho các pattern

Motivation(Động lực)—cung cấp một ví dụ về vấn đề

Applicability (Khả năng áp dụng)—lưu ý tình huống thiết kế cụ thể, trong đó mô hình được áp dụng

Structure(cấu trúc)—mô tả các lớp được yêu cầu để thực hiện mô hình

Participants (thành phần tham gia)—mô tả trách nhiệm của các lớp được yêu cầu để thực hiện pattern

Collaborations (sự cộng tác)—mô tả cách những thành phần tham gia cộng tác để thực hiện trách nhiệm của mình

Consequences(điều sau)—mô tả các "lực lượng thiết kế" có ảnh hưởng đến các mô hình và các đánh đổi tiềm năng phải được xem xét khi mô hình được thực hiện

Related patterns(patterns liên quan)—tham khảo chéo liên quan đến các mẫu thiết kế

Khung (frameworks)

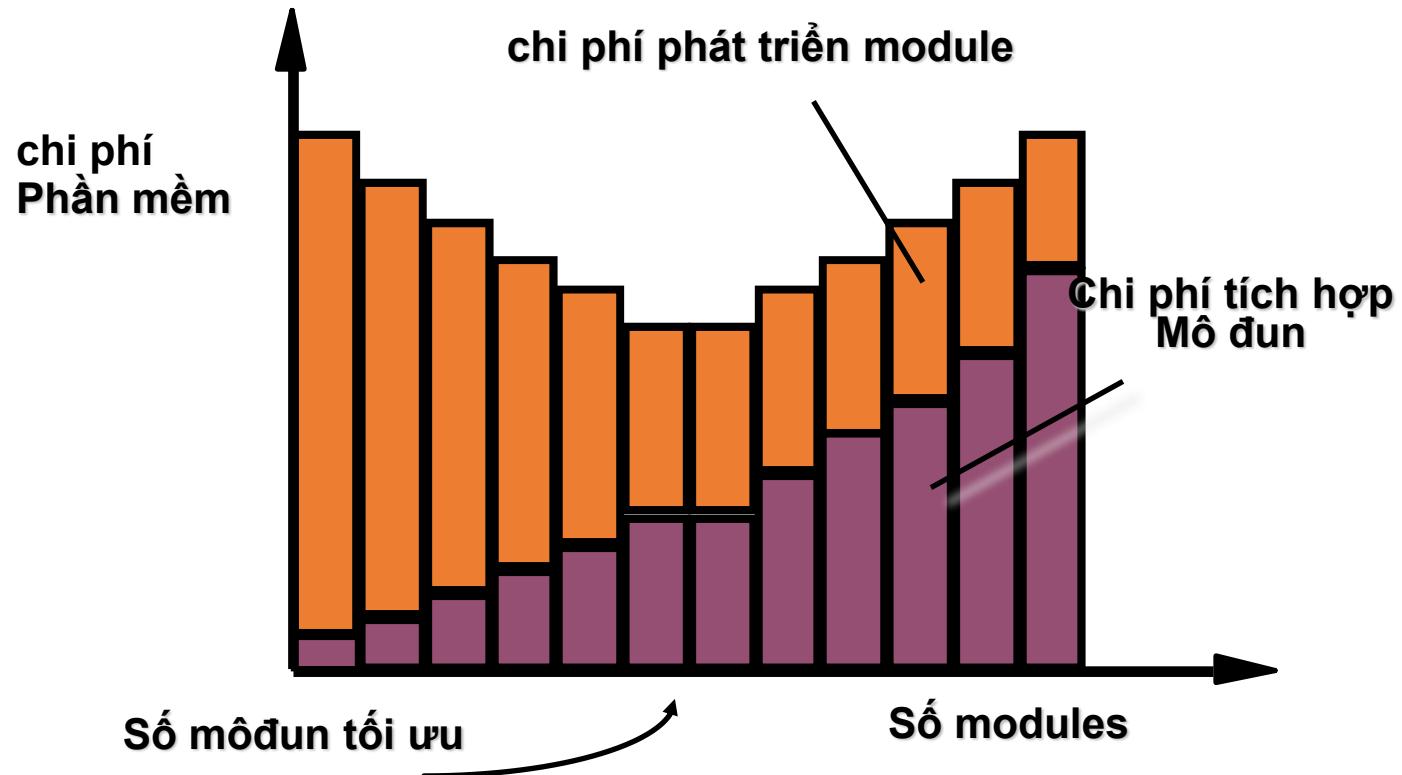
- Bản thân khuôn mẫu có thể không đủ để phát triển một thiết kế đầy đủ
 - Trong một số trường hợp, cần thiết phải cung cấp một cơ sở hạ tầng thực hiện cụ thể, được gọi là một khung (**framework**) cho công việc thiết kế.
- Một framework không phải là một khuôn mẫu kiến trúc, mà là một bộ khung với một tập hợp các “**plug points**” (còn được gọi là hooks hay slots) cho phép nó thích ứng với một miền vấn đề xác định.
 - Những “**plug points**” cho phép bạn tích hợp các lớp vấn đề cụ thể hoặc chức năng trong các bộ khung.

Mô đun hoá (Modularity)

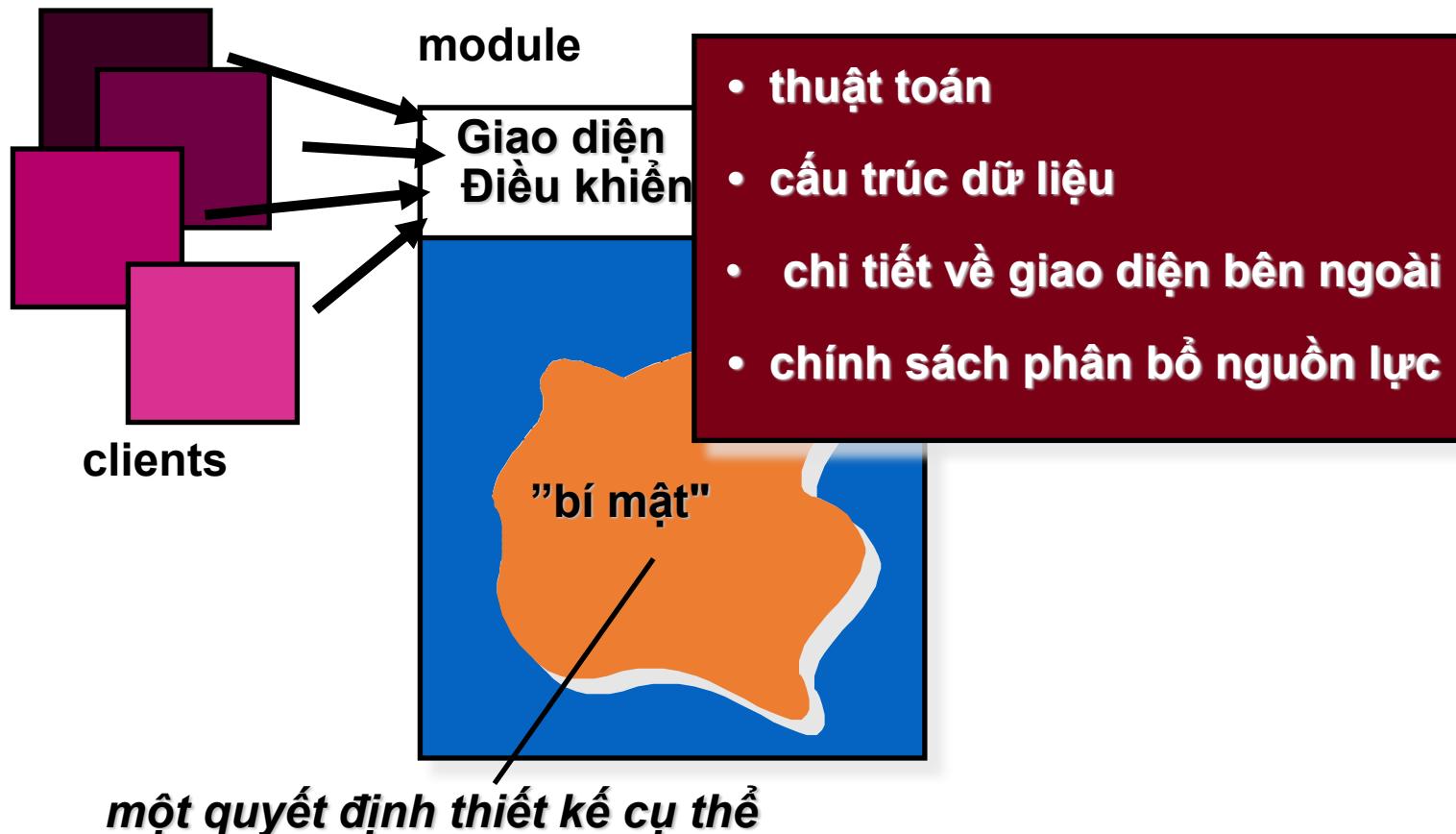
- "Mô đun là thuộc tính duy nhất của phần mềm cho phép một chương trình có thể quản lý một cách thông minh" [Mye78].
- Phần mềm nguyên khối (ví dụ, một chương trình lớn gồm một mô-đun duy nhất) có thể không được dễ dàng nắm bắt được bởi một kỹ sư phần mềm.
 - Số lượng các đường dẫn điều khiển, khoảng thời gian tham khảo, số lượng các biến, và độ phức tạp tổng thể sẽ làm cho việc hiểu được gần như không thể.
- Trong hầu hết các trường hợp, bạn nên phá vỡ thiết kế thành nhiều module, hy vọng sẽ làm cho việc hiểu biết dễ dàng hơn và như một hệ quả, giảm chi phí cần thiết để xây dựng các phần mềm.

Modularity: sự đánh đổi

**Số "chính xác" các mô đun
cho một thiết kế phần mềm cụ thể?**



Ân thông tin



Tại sao lại Ân thông tin?

- Làm giảm khả năng "tác dụng phụ"
- Hạn chế ảnh hưởng chung của quyết định thiết kế cục bộ
- Nhấn mạnh truyền thông qua giao diện điều khiển
- Không khuyến khích việc sử dụng các dữ liệu toàn cục
- Dẫn đến đóng gói, một thuộc tính của thiết kế chất lượng cao
- Kết quả tạo ra phần mềm chất lượng cao

Tái cấu trúc (Refactoring)

- Fowler [FOW99] định nghĩa cấu trúc lại theo cách sau đây:
 - "Tái cấu trúc là quá trình thay đổi một hệ thống phần mềm trong một cách mà nó không làm thay đổi hành vi bên ngoài của mã [thiết kế] nhưng cải thiện cấu trúc bên trong của nó."
 - Khi phần mềm được refactored, thiết kế hiện có được kiểm tra về sự
 - Dư thừa
 - Yếu tố thiết kế không sử dụng
 - Các thuật toán không hiệu quả hoặc không cần thiết
 - Cấu trúc dữ liệu xây dựng kém hoặc không phù hợp
 - Hoặc bất kỳ sự thất bại thiết kế khác có thể được điều chỉnh để mang lại một thiết kế tốt hơn.

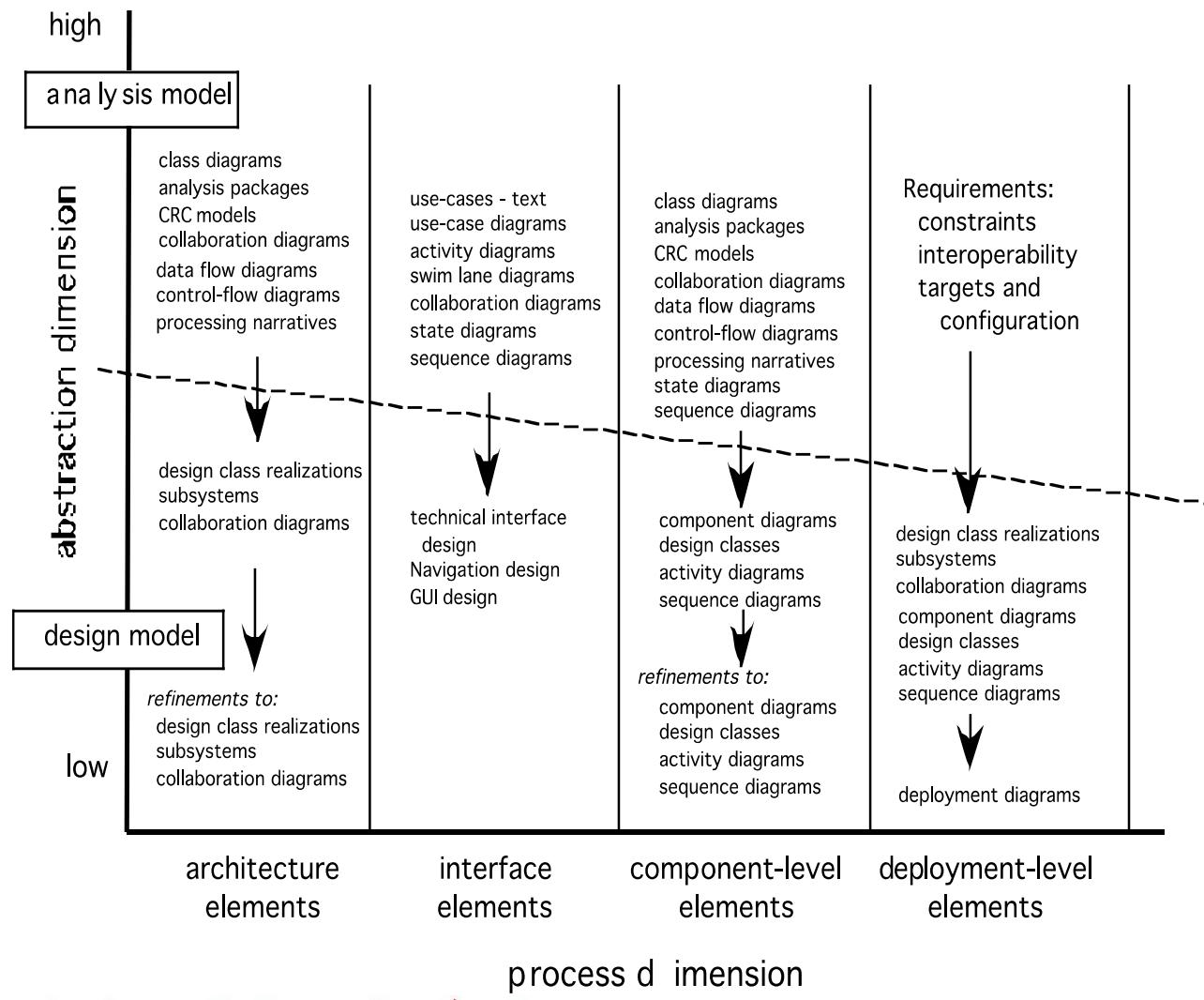
2.2 Các khái niệm thiết kế Hướng đối tượng

- **Các lớp thiết kế**
 - Các lớp thực thể
 - Các lớp biên
 - Các lớp điều khiển
- **Sự thừa kế**—tất cả các trách nhiệm của một lớp cha được ngay lập tức được thừa kế bởi tất cả các lớp con
- **Thông điệp**—khuyến khích một số hành vi xảy ra trong đối tượng nhận
- **Đa hình**—một đặc tính mà làm giảm đáng kể nỗ lực cần thiết để mở rộng thiết kế.

Các lớp thiết kế

- Các lớp phân tích được tinh chỉnh trong quá trình thiết kế để trở thành **các lớp thực thể**
- **Các lớp biên** phát triển trong thiết kế để tạo ra giao diện (ví dụ, màn hình tương tác hoặc báo cáo) mà người dùng thấy và tương tác với phần mềm.
 - Các lớp biên được thiết kế với trách nhiệm quản lý các đối tượng cách thực thể được đại diện cho người sử dụng.
- **Các lớp điều khiển** được thiết kế để quản lý
 - Việc tạo ra hoặc cập nhật các đối tượng thực thể;
 - Sự tức thời của các đối tượng biên khi họ có được thông tin từ các đối tượng thực thể;
 - Truyền thông phức tạp giữa các tập của các đối tượng;
 - Xác nhận của dữ liệu trao đổi giữa các đối tượng hoặc giữa người sử dụng và với ứng dụng.

2.3 Mô hình thiết kế



Các thành phần trong Mô hình thiết kế

- **Các thành phần dữ liệu**

- Mô hình dữ liệu -> cấu trúc dữ liệu
- Mô hình dữ liệu -> kiến trúc cơ sở dữ liệu

- **Các thành phần kiến trúc**

- Miền ứng dụng
- Các lớp phân tích, mối quan hệ, hợp tác và hành vi của chúng được chuyển thành chứng ngộ thiết kế
- Có thể áp dụng các mẫu thiết kế và “kiểu”

- **Các thành phần giao diện**

- Giao diện người dùng (UI)
- Giao diện bên ngoài đến các hệ thống khác, các thiết bị, mạng, hoặc các nhà sản xuất khác hoặc người dùng thông tin
- Giao diện nội bộ giữa các thành phần thiết kế khác nhau.

- **Các yếu tố cấu thành**

- **Các thành phần triển khai**

Các thành phần kiến trúc

- Các mô hình kiến trúc [Sha96] có nguồn gốc từ ba nguồn:
 - Thông tin về miền ứng dụng cho xây dựng phần mềm;
 - Các thành phần mô hình yêu cầu cụ thể như sơ đồ luồng dữ liệu và phân tích các lớp, các mối quan hệ và sự hợp tác của chúng, và
 - Sự sẵn có của mô hình kiến trúc và các kiểu

Các thành phần giao diện

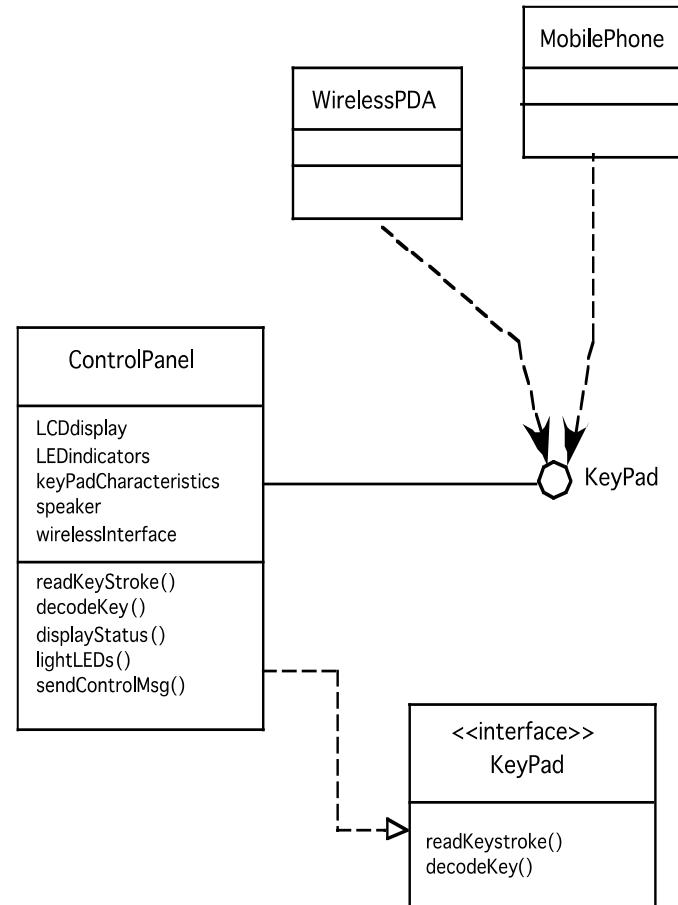


Figure 9.6 UML interface representation for ControlPanel

Các thành phần triển khai

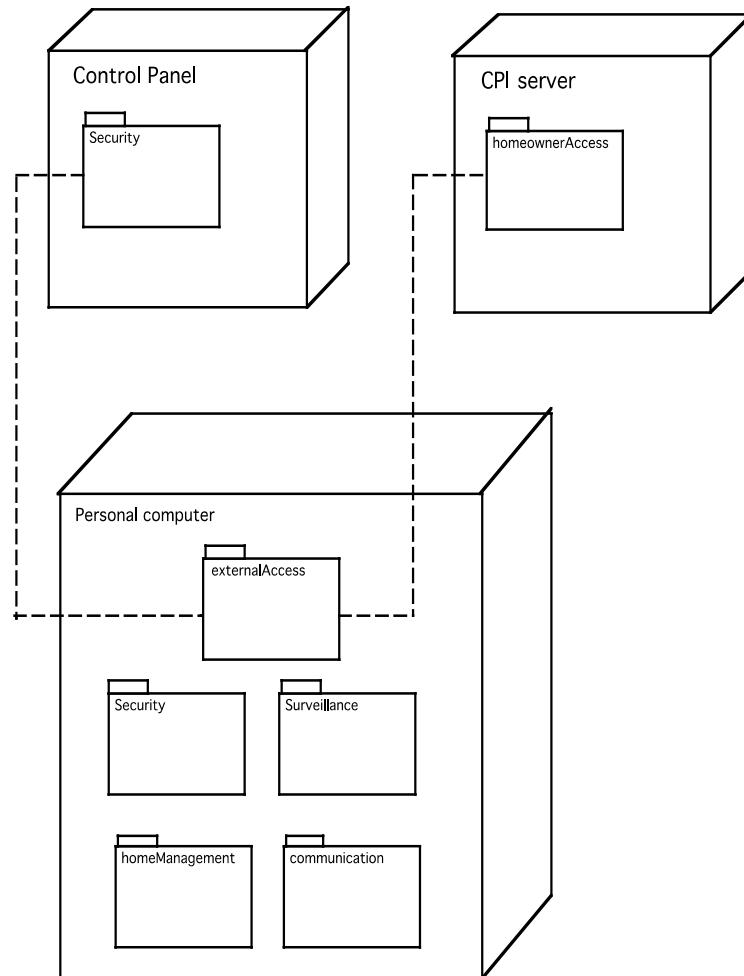


Figure 9.8 UML deployment diagram for *SafeHome*

Nội dung

1. Tổng quan thiết kế phần mềm
2. Các khái niệm trong thiết kế phần mềm
- 3. Thiết kế kiến trúc phần mềm**
4. Thiết kế chi tiết phần mềm
5. Tính móc nối (Coupling) và tính kết dính (Cohesion)

3.1 Tại sao cần thiết kế kiến trúc?

- Các kiến trúc không phải là phần mềm hoạt động. Thay vào đó, nó là một đại diện cho phép một kỹ sư phần mềm:
 1. Phân tích hiệu quả của thiết kế trong việc đáp ứng các yêu cầu đề ra,
 2. Xem xét lựa chọn thay thế kiến trúc khi thay đổi thiết kế vẫn tương đối dễ dàng, và
 3. Giảm thiểu rủi ro gắn liền với việc xây dựng các phần mềm.

3.1 Tại sao cần thiết kế kiến trúc?

- **Đại diện của kiến trúc phần mềm là một tạo khả năng cho truyền thông giữa tất cả các bên (các bên liên quan) quan tâm đến sự phát triển của một hệ thống dựa trên máy tính.**
- **Những kiến trúc làm nổi bật thiết kế quyết định ban đầu mà sẽ có một tác động sâu sắc trên tất cả các công việc kỹ thuật phần mềm sau và, quan trọng hơn, vào sự thành công cuối cùng của hệ thống như là một thực thể hoạt động.**
- **Kiến trúc "tạo thành một mode minh bạch tương đối nhỏ về cách hệ thống được cấu trúc và cách các thành phần của nó làm việc cùng nhau"**

3.2 Mô tả kiến trúc

- The IEEE Standard định nghĩa một mô tả kiến trúc (AD) là một "một tập hợp các sản phẩm để tài liệu hoá một kiến trúc"
 - Các mô tả chính nó được đại diện bằng cách sử dụng nhiều quan điểm, nơi từng xem là "một đại diện của cả một hệ thống từ quan điểm của một tập hợp có liên quan của [các bên liên quan] các mối quan tâm"

Thể loại kiến trúc

- **Thể loại** ngũ ý một phân loại cụ thể trong lĩnh vực phần mềm tổng thể.
- Trong mỗi thể loại, bạn gấp phải một số tiểu phân loại.
 - Ví dụ, trong các thể loại của các tòa nhà, bạn sẽ gấp phải những phong cách chung sau đây: nhà ở, căn hộ, chung cư, cao ốc văn phòng, tòa nhà công nghiệp, nhà kho, vv.
 - Trong mỗi phong cách chung, phong cách cụ thể hơn có thể được áp dụng. Mỗi phong cách sẽ có một cấu trúc có thể được mô tả bằng một tập các mô hình dự đoán được.

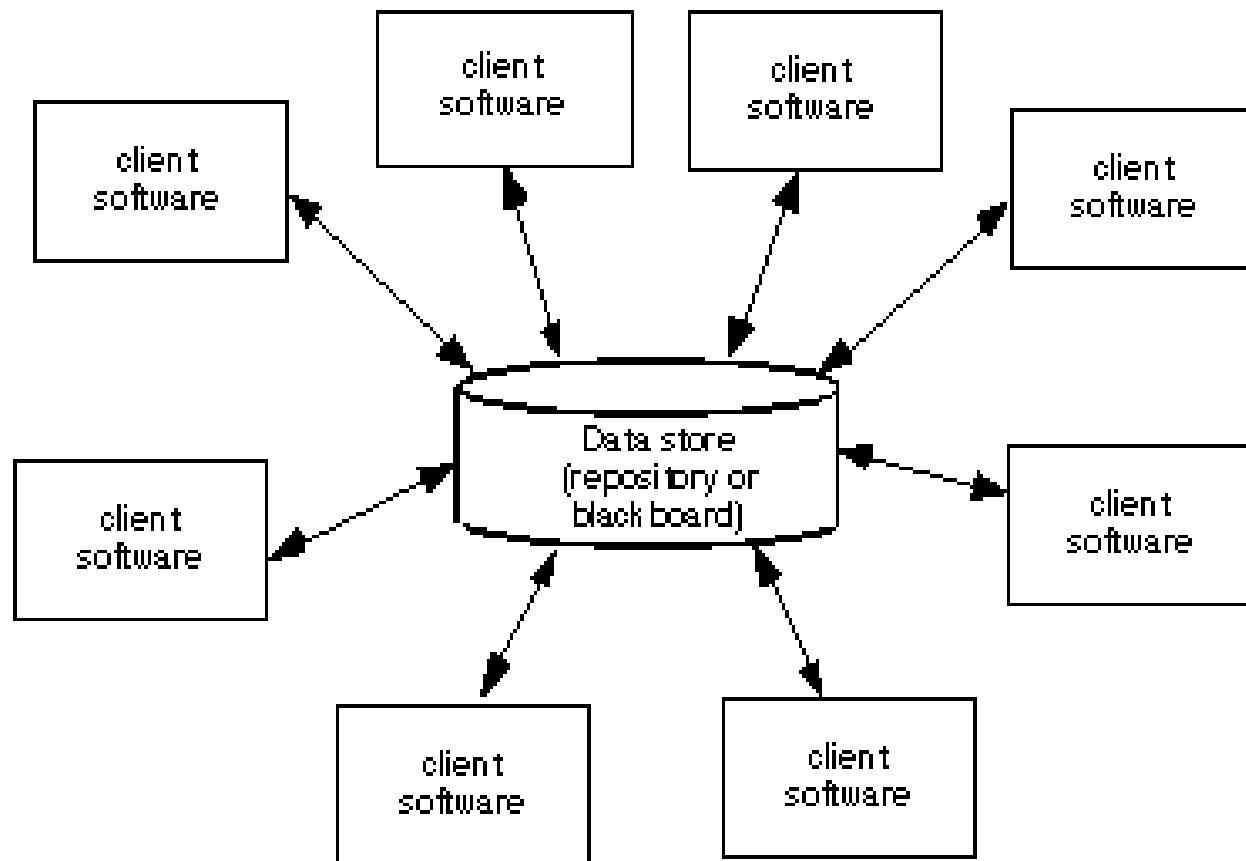
3.3 Kiểu kiến trúc

- **Mỗi phong cách mô tả một loại hệ thống bao gồm:**
 - (1) một tập hợp các thành phần (ví dụ, một cơ sở dữ liệu, mô đun tính toán) thực hiện một chức năng cần thiết của một hệ thống,
 - (2) một tập hợp các kết nối cho phép "truyền thông, phối hợp và hợp tác" giữa các thành phần,
 - (3) khó khăn để xác định cách các thành phần có thể được tích hợp để tạo thành hệ thống, và
 - (4) các mô hình ngữ nghĩa cho phép một nhà thiết kế phải hiểu được tính chất tổng thể của hệ thống bằng cách phân tích các đặc tính được biết đến của các bộ phận cấu thành của nó.

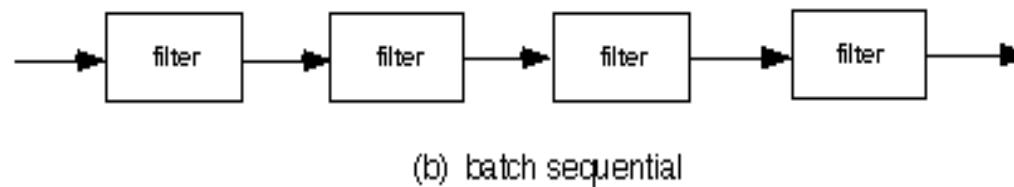
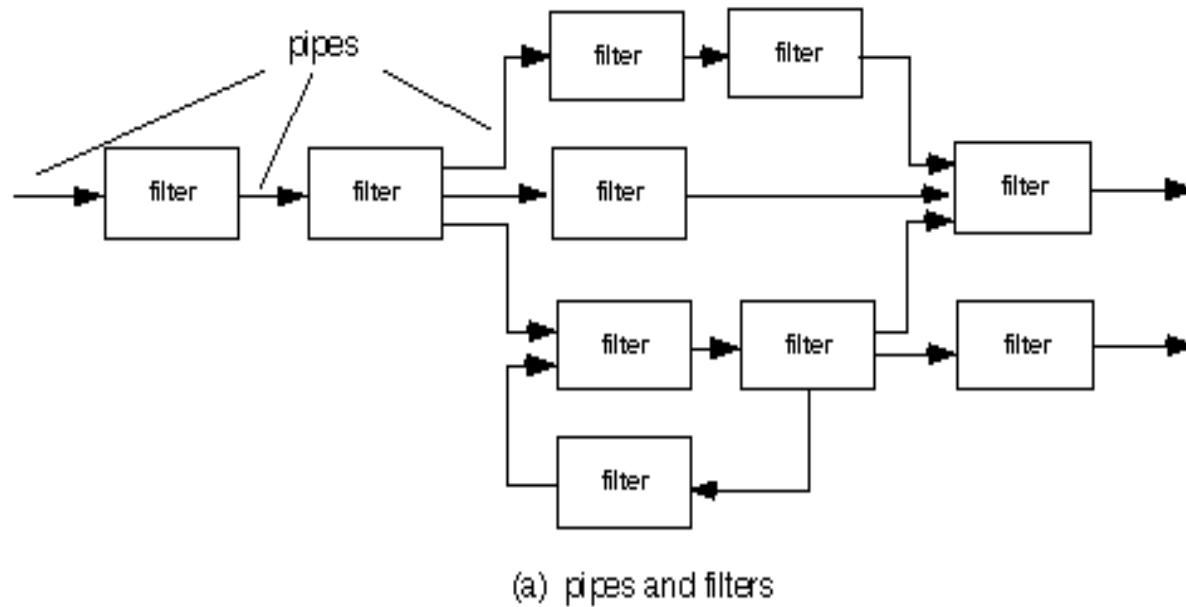
3.3 Kiểu kiến trúc

- Một số kiểu kiến trúc hệ thống:
 - Kiến trúc lấy dữ liệu làm trung tâm
 - Kiến trúc luồng dữ liệu
 - Kiến trúc gọi và trả về
 - Kiến trúc phân lớp
 - Kiến trúc hướng đối tượng
 - ...

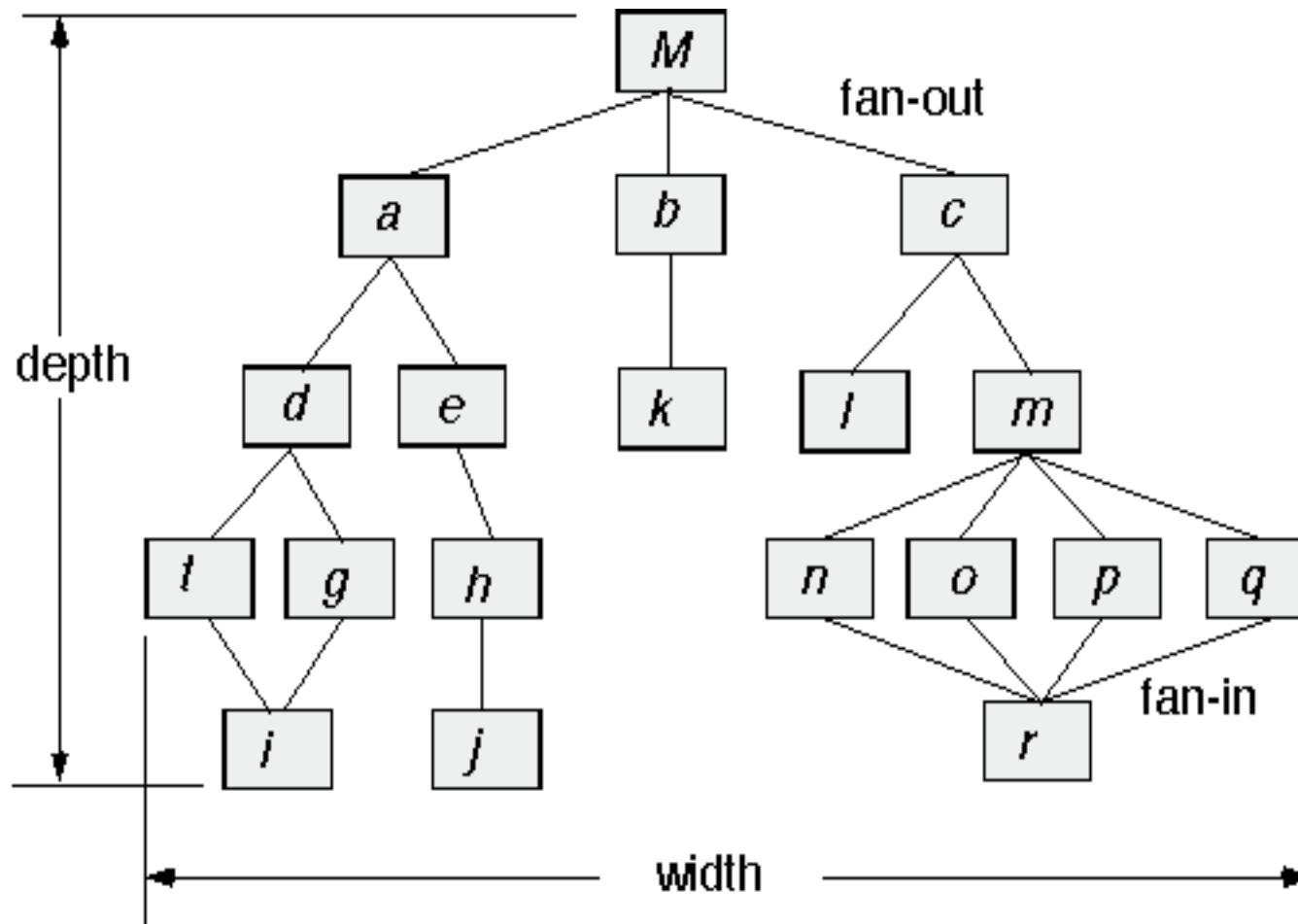
Kiến trúc lấy dữ liệu làm trung tâm



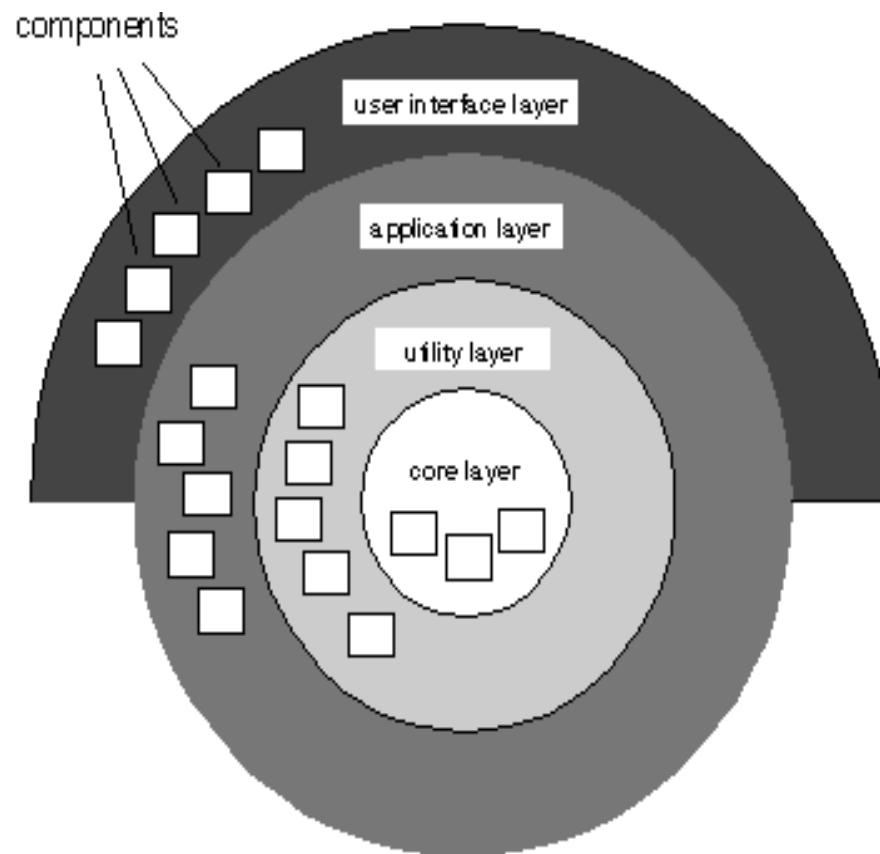
Kiến trúc luồng dữ liệu



Kiến trúc gọi và trả về



Kiến trúc phân lớp



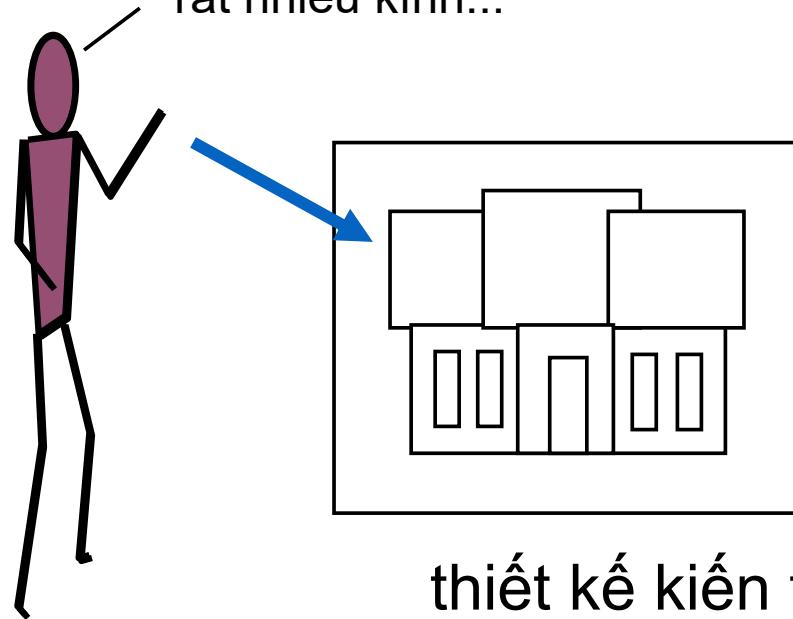
3.4 Thiết kế kiến trúc

- Phần mềm phải được đặt trong bối cảnh
 - Thiết kế cần xác định các thực thể bên ngoài (các hệ thống khác, thiết bị, con người) mà phần mềm tương tác với và bản chất của sự tương tác
- Một tập hợp các nguyên mẫu kiến trúc cần được xác định
 - Một nguyên mẫu là một trừu tượng (tương tự như một lớp) đại diện cho một phần tử của hệ thống hành vi
 - Các nhà thiết kế xác định cấu trúc của hệ thống bằng cách xác định và tinh chỉnh các thành phần phần mềm thực hiện từng nguyên mẫu

Ví dụ

yêu cầu khách hàng

"bốn phòng ngủ, ba phòng tắm,
rất nhiều kính..."



thiết kế kiến trúc

Các bước thiết kế kiến trúc

1. Thu thập các kịch bản.
2. Gợi ý các yêu cầu, ràng buộc, và đặc tả môi trường.
3. Mô tả các phong cách / mô hình kiến trúc đã được lựa chọn để giải quyết các tình huống và yêu cầu:
 - quan điểm mô-đun
 - góc nhìn quá trình
 - quan điểm dòng dữ liệu
4. Đánh giá chất lượng thuộc tính bằng cách coi mỗi thuộc tính trong sự cô lập.
5. Xác định mức độ nhạy cảm của các thuộc tính chất lượng với các thuộc tính kiến trúc khác nhau cho một phong cách kiến trúc cụ thể.
6. Kiến trúc ứng cử viên phê bình (được phát triển trong bước 3) bằng cách sử dụng phân tích độ nhạy được thực hiện ở bước 5.

Nội dung

1. Tổng quan thiết kế phần mềm
2. Các khái niệm trong thiết kế phần mềm
3. Thiết kế kiến trúc phần mềm
- 4. Thiết kế chi tiết phần mềm**
5. Tính móc nối (Coupling) và tính kết dính (Cohesion)

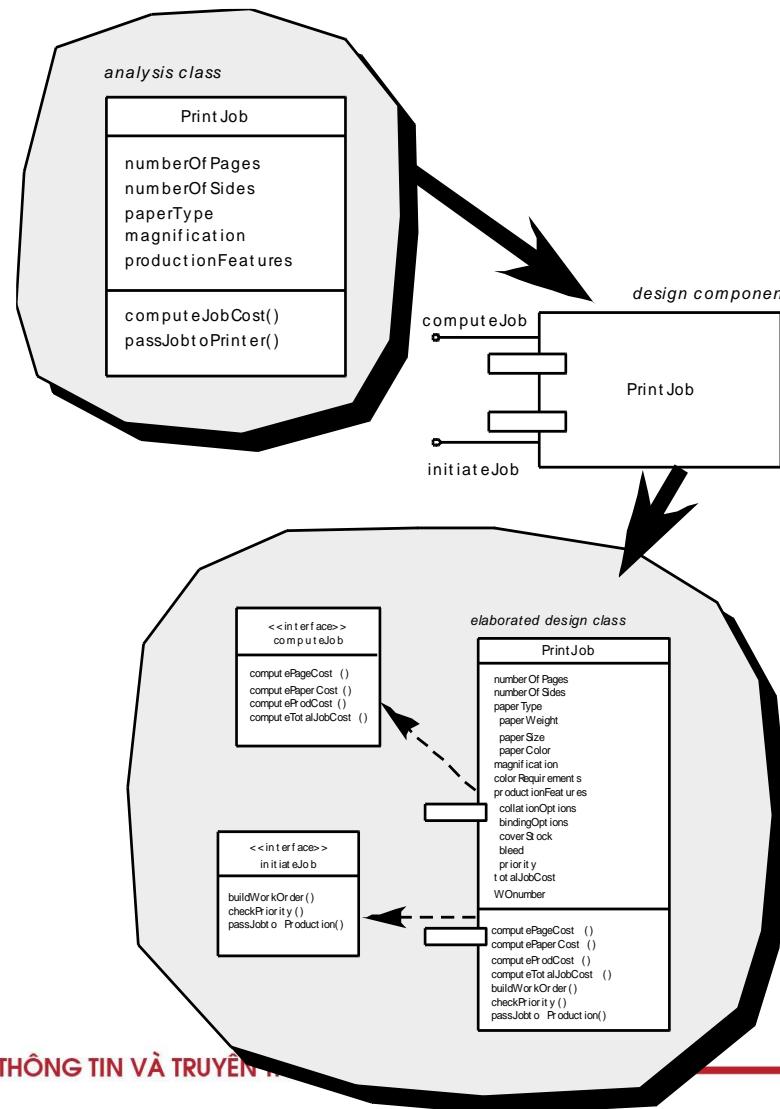
4.1 Thiết kế chi tiết

- Thiết kế chi tiết: chỉ ra chi tiết và đầy đủ về *thành phần*, tạo điều kiện xây dựng phần mềm trong pha sau đó → thiết kế mức thành phần
- Các hoạt động thiết kế chi tiết:
 - Thiết kế chi tiết lớp, module,
 - Thiết kế thuật toán,
 - Thiết kế dữ liệu,
 - Thiết kế giao diện,
 - ...

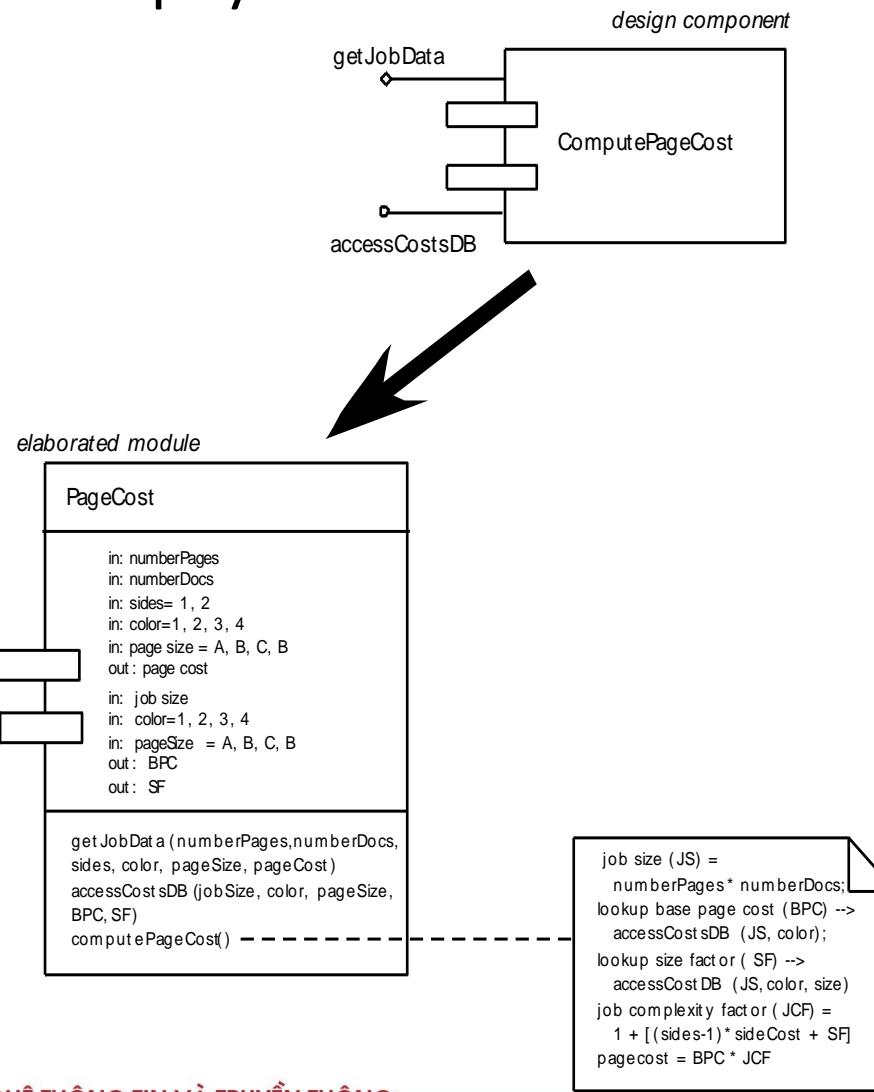
Thế nào là một thành phần?

- *OMG Unified Modeling Language Specification* [OMG01] định nghĩa một thành phần (component) là:
 - Một phần có tính mô-đun, có thể triển khai và thay thế của một hệ thống mà đóng gói việc thực thi và đưa ra một tập các giao diện.
- **Góc nhìn hướng đối tượng:** Một thành phần bao gồm một tập các lớp cộng tác.
- **Góc nhìn quy ước:** Một thành phần bao gồm logic xử lý, cấu trúc dữ liệu bên trong cần thiết để triển khai logic đó và một giao diện cho phép thành phần được gọi và dữ liệu được truyền đến nó.

Thành phần hướng đối tượng



Thành phần quy ước



Nguyên lý thiết kế cơ bản

- **Nguyên lý mở-đóng (OCP)**: Một mô-đun (thành phần) nên được mở cho việc mở rộng nhưng nên đóng cho việc hiệu chỉnh.
- **Nguyên lý thay thế Liskov (LSP)**: Các lớp con nên thay thế được các lớp chs.
- **Nguyên lý tráo đổi phụ thuộc (DIP)**: Phụ thuộc vào trừu tượng hóa, không phụ thuộc vào kết cấu.
- **Nguyên lý tách biệt giao diện (ISP)**: Nhiều giao diện client cụ thể thì tốt hơn một giao diện mục đích chung.
- **Nguyên lý phát hành và tái sử dụng tương đương (REP)**: "Nguyên tắc tái sử dụng là nguyên tắc phát hành". Nói cách khác, nếu một thành phần được coi là có thể tái sử dụng thì nó phải là một đơn vị có thể thay thế được.
- **Nguyên lý đóng chung (CCP)**: Các lớp thay đổi cùng nhau thì thuộc về nhau
- **Nguyên lý tái sử dụng chung (CRP)**. Các lớp không được tái sử dụng cùng nhau thì không nên nhóm lại với nhau"

Source: Martin, R., "Design Principles and Design Patterns," downloaded from <http://www.objectmentor.com>, 2000.

Hướng dẫn thiết kế

- **Thành phần:**

- Quy ước đặt tên nên được thiết lập cho các thành phần được xác định như là một phần của mô hình kiến trúc rồi sau đó tinh chỉnh và xây dựng như là một phần của mô hình mức thành phần

- **Giao diện:**

- Giao diện cung cấp thông tin quan trọng về sự giao tiếp và cộng tác (đồng thời cũng giúp chúng ta đạt được OPC)

- **Phụ thuộc và kế thừa:**

- Việc mô hình hóa sự phụ thuộc từ trái sang phải và sự kế thừa từ dưới lên trên.

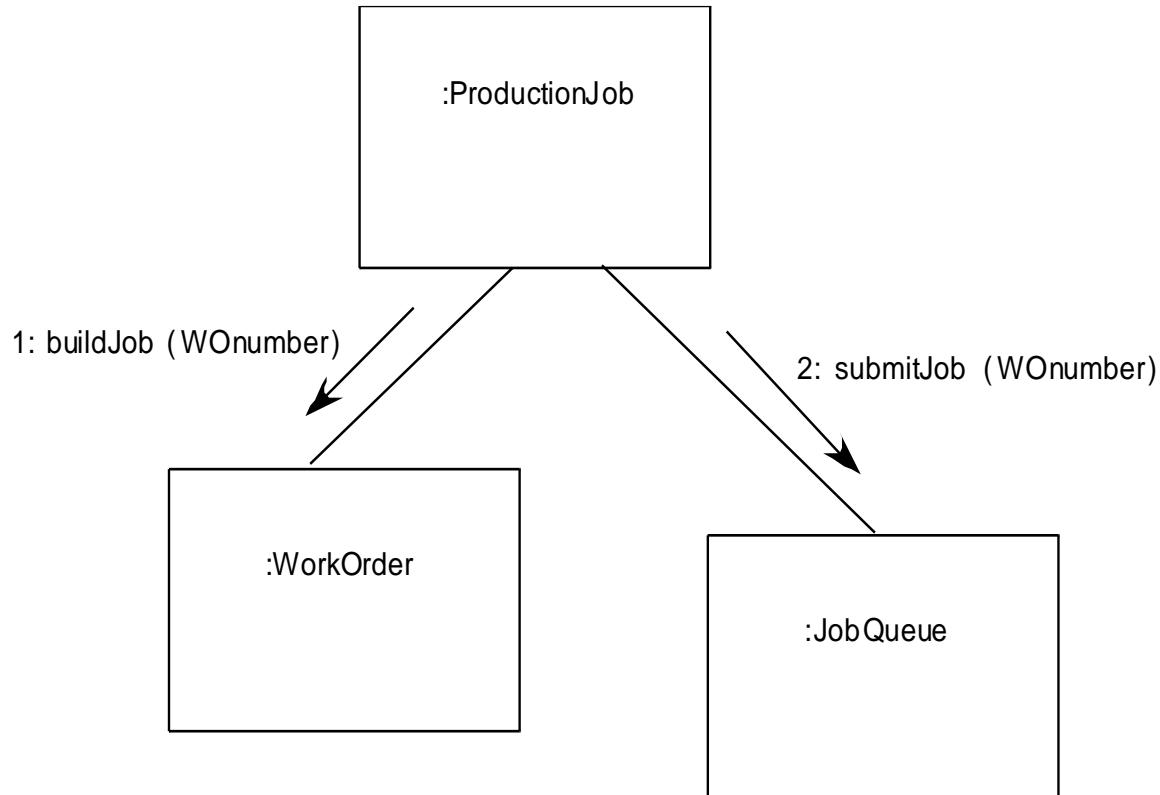
Thiết kế mức thành phần - I

- Bước 1. Xác định tất cả các lớp thiết kế tương ứng với miền vấn đề.
- Bước 2. Xác định tất cả các lớp thiết kế tương ứng với kết cấu hạ tầng.
- Bước 3. Xây dựng tất cả các lớp không có được như là thành phần tái sử dụng.
- Bước 3a. Xác định chi tiết thông điệp khi các lớp hoặc các thành phần cộng tác.
- Bước 3b. Xác định các giao diện thích hợp cho mỗi thành phần.

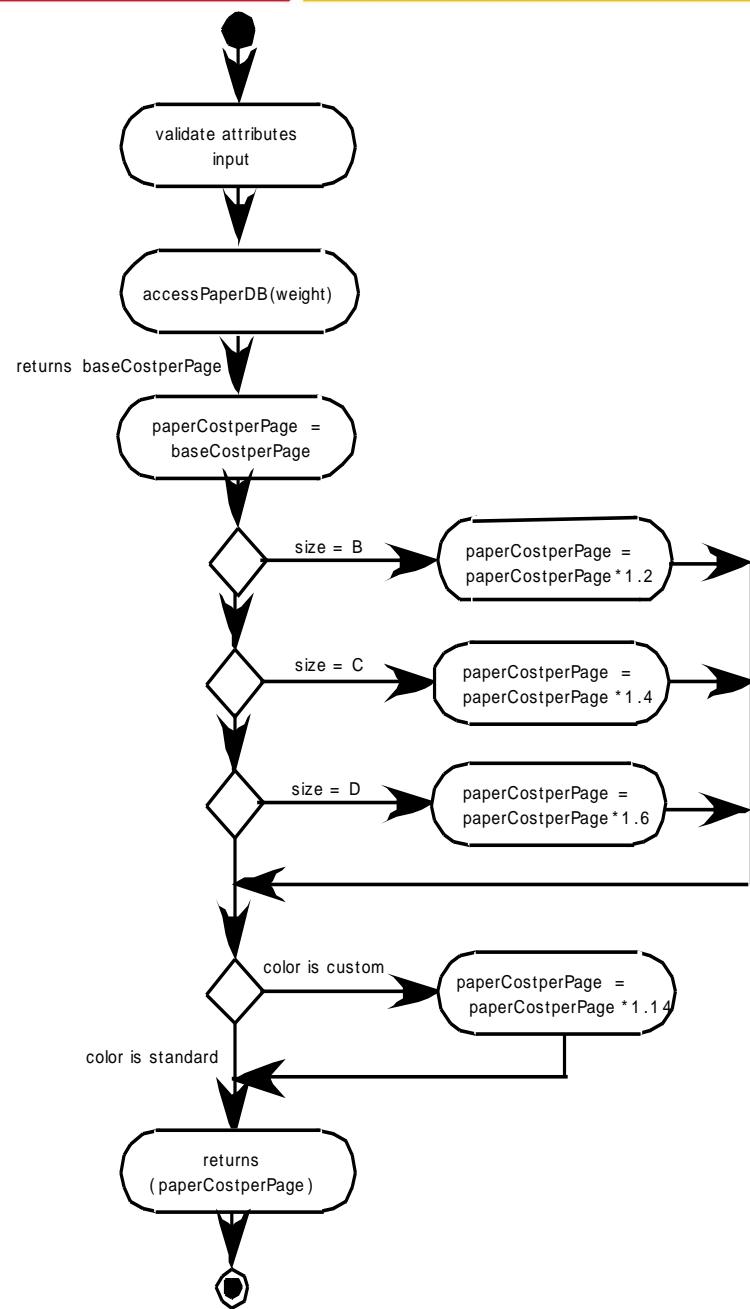
Thiết kế mức thành phần - II

- Step 3c. Xây dựng các thuộc tính và xác định kiểu dữ liệu và cấu trúc dữ liệu cần thiết để thực hiện chúng.
- Step 3d. Mô tả luồng xử lý trong từng hoạt động cụ thể.
- Step 4. Mô tả các nguồn dữ liệu bền vững (cơ sở dữ liệu và các tập tin) và xác định các lớp cần thiết để quản lý chúng.
- Step 5. Phát triển và xây dựng biểu diễn hành vi cho một lớp hoặc một thành phần.
- Step 6. Xây dựng sơ đồ triển khai để cung cấp thêm thông tin về chi tiết thực hiện.
- Step 7. Nhân tố hóa mỗi thể hiện thiết kế mức thành phần và luôn luôn xem xét phương án thay thế.

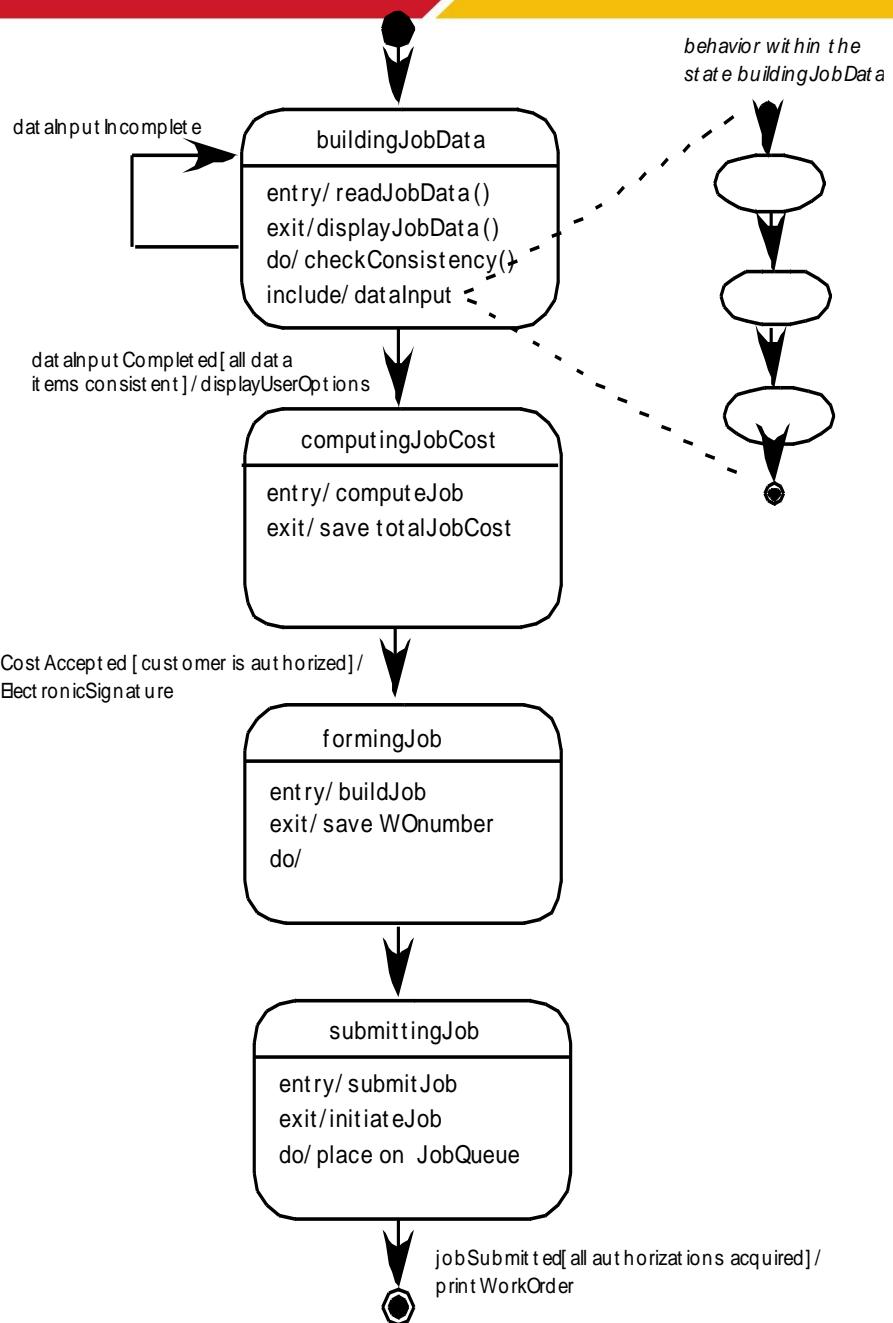
Sơ đồ cộng tác



Sơ đồ hoạt động



Sơ đồ trạng thái



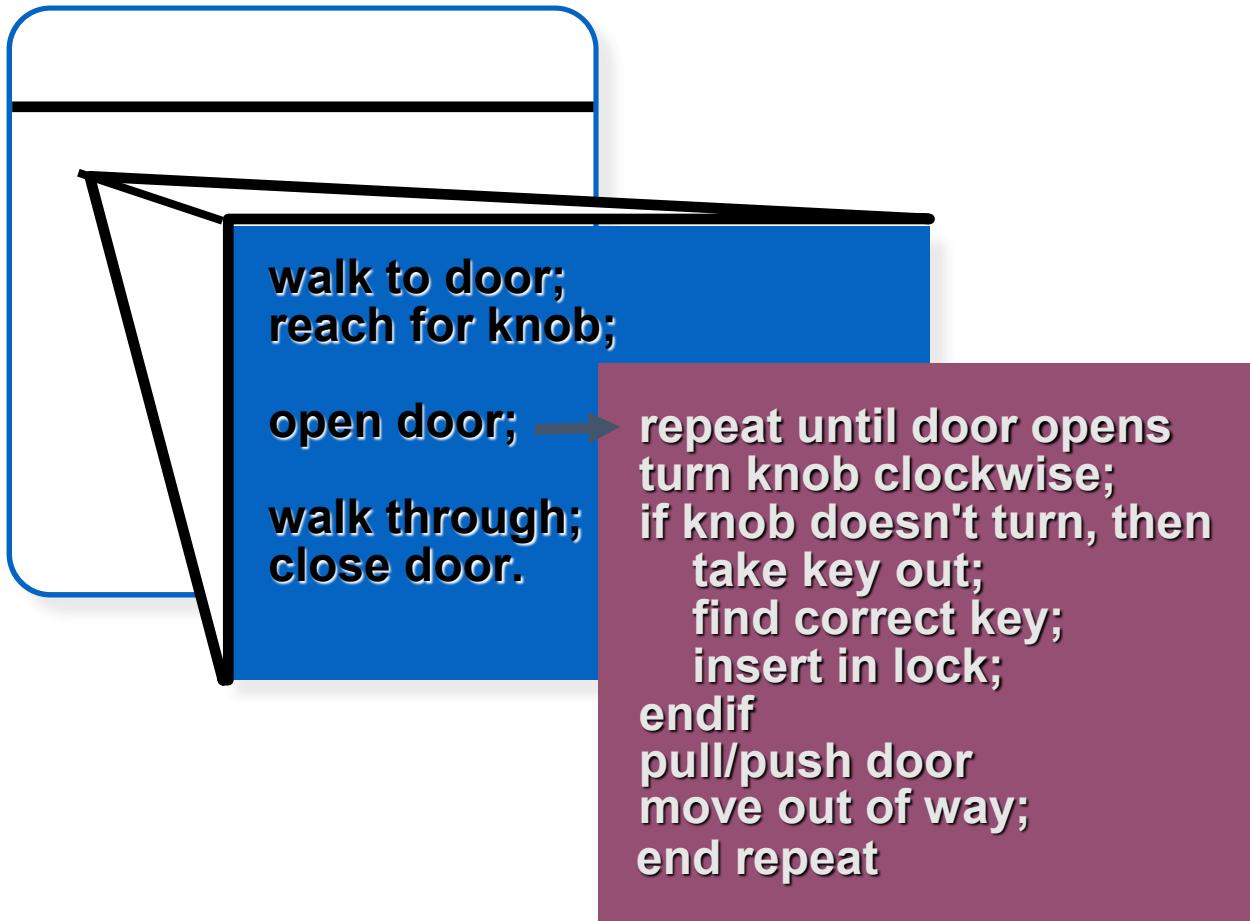
Thiết kế thành phần quy ước

- Việc thiết kế các xử lý logic được quy định bởi các nguyên tắc cơ bản của thiết kế thuật toán và lập trình có cấu trúc.
- Việc thiết kế các cấu trúc dữ liệu được định nghĩa bởi các mô hình dữ liệu được phát triển cho hệ thống.
- Việc thiết kế các giao diện được quy định bởi sự hợp tác mà một thành phần phải có ảnh hưởng.

Thiết kế thuật toán

- Là hoạt động thiết kế sát nhất với việc coding.
- Cách tiếp cận:
 - Xem xét mô tả thiết kế cho các thành phần
 - Sử dụng tinh chỉnh từng bước để phát triển thuật toán
 - Sử dụng lập trình có cấu trúc để thực hiện logic có tính thủ tục
 - Sử dụng ‘phương pháp chính thống’ để chứng minh logic.

Tinh chỉnh từng bước



Mô hình biểu diễn thiết kế thuật toán

- Trình bày các thuật toán ở mức độ chi tiết mà có thể được xem xét lại chất lượng.
- Tùy chọn:
 - Đồ họa (e.g. flowchart, box diagram)
 - Mã giả (e.g., PDL)
 - Ngôn ngữ lập trình
 - Bảng quyết định

Thiết kế dữ liệu

- Tìm kiếm biểu diễn logic cho các phần tử dữ liệu đã được nhận diện trong giai đoạn phân tích yêu cầu.
- Thiết kế các cấu trúc dữ liệu của chương trình và cơ sở dữ liệu
- Các tiêu chí thiết kế dữ liệu
 - Không dư thừa dữ liệu
 - Tối ưu hóa không gian lưu trữ
- Được biểu diễn ở các mức trừu tượng khác nhau
 - Mô hình dữ liệu quan niệm, mô hình dữ liệu logic, mô hình dữ liệu vật lý

Một số nguyên tắc thiết kế dữ liệu

- Nhận diện cả cấu trúc dữ liệu và tác vụ truy xuất
- Chú ý sử dụng từ điển dữ liệu
- Trì hoãn thiết kế dữ liệu mức thấp cho đến cuối giai đoạn này
- Che giấu biểu diễn bên trong của cấu trúc dữ liệu
- Nên áp dụng kiểu ADT trong thiết kế cũng như trong lập trình

Nội dung

1. Tổng quan thiết kế phần mềm
2. Các khái niệm trong thiết kế phần mềm
3. Thiết kế kiến trúc phần mềm
4. Thiết kế chi tiết phần mềm
5. **Tính móc nối (Coupling) và tính kết dính (Cohesion)**

Chất lượng của thiết kế

- Thế nào là một thiết kế tốt?
 - Easy for Developing, Reading & Understanding
 - Easy for Communication
 - Easy for Extending (add new features)
 - Easy for Maintenance

Chất lượng của thiết kế

- Mỗi module có tính cố kết cao (high cohesion)
 - Mỗi module là một đơn vị logic
 - Toàn bộ module cùng đóng góp thực hiện một mục tiêu
- Liên kết lỏng lẻo (loose coupling) giữa các module
 - Ít ràng buộc, ít phục thuộc lẫn nhau
- Dễ hiểu
- Định nghĩa rõ ràng
 - Các module và quan hệ giữa chúng

Cohesion và Coupling

- Cohesion

- Gắn kết đề cập đến mức độ mà các phần tử của một mô-đun thuộc về nhau. Tính gắn kết là thước đo mức độ liên quan hoặc tập trung của các trách nhiệm của một mô-đun đơn lẻ.

- Coupling

- Ghép nối / Khớp nối hoặc phụ thuộc là mức độ mà mỗi chương trình mô-đun phụ thuộc vào từng mô-đun khác.

“Loose coupling and high cohesion”

Sự gắn kết (cohesion)

- Góc nhìn quy ước:
 - “Tư duy đơn lẻ” của một mô-đun: mục tiêu thiết kế chung của việc tách các mối quan tâm cho thấy một mô-đun nên giải quyết một nhóm mối quan tâm duy nhất
- Góc nhìn hướng đối tượng:
 - Sự gắn kết nghĩa là một thành phần hoặc một lớp chỉ đóng gói các thuộc tính và hoạt động liên quan chặt chẽ với nhau và với bản thân thành phần hoặc lớp đó.

Sự gắn kết (cohesion)

- Các mức của sự gắn kết:

- Chức năng
- Tầng
- Truyền thông
- Liên tục
- Thủ tục
- Phụ thuộc thời gian
- Lợi ích



Ghép nối (coupling)

- Góc nhìn quy ước:
 - Là mức độ mà một thành phần kết nối với các thành phần khác và với thế giới bên ngoài.
- Góc nhìn hướng đối tượng:
 - Một thước đo chất lượng mức độ kết nối của các lớp với lớp khác.
- Các mức của ghép nối:
 - Nội dung
 - Chung
 - Điều khiển
 - Đánh dấu
 - Dữ liệu
 - Lời gọi công thức
 - Loại sử dụng
 - Sự lồng ghép và bao hàm



Tổng kết

- Tầm quan trọng của thiết kế phần mềm có thể được phát biểu bằng một từ “**chất lượng**”.
- *Thiết kế là nơi chất lượng phần mềm được nuôi dưỡng trong quá trình phát triển*: cung cấp cách biểu diễn phần mềm có thể được xác nhận về chất lượng, là cách duy nhất mà chúng ta có thể chuyển hóa một cách chính xác các yêu cầu của khách hàng thành sản phẩm hay hệ thống phần mềm cuối cùng.
- Thiết kế phần mềm là **công cụ giao tiếp** làm cơ sở để có thể mô tả một cách đầy đủ các dịch vụ của hệ thống, để quản lý các rủi ro và lựa chọn giải pháp thích hợp.

Tổng kết (2)

- Thiết kế phần mềm phục vụ như một **nền tảng cho mọi bước kĩ nghệ phần mềm và bảo trì**.
- Không có thiết kế có nguy cơ sản sinh một hệ thống không ổn định - một hệ thống sẽ thất bại.
- Một hệ thống phần mềm rất khó xác định được chất lượng chừng nào chưa đến bước kiểm thử. **Thiết kế tốt là bước quan trọng đầu tiên để đảm bảo chất lượng phần mềm.**

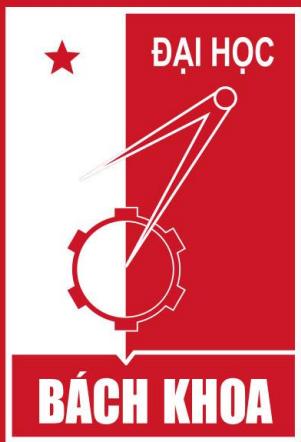


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The graphic consists of the number "25" in large, bold, white font. A curved banner arches over the top of the "2" containing the text "YEARS ANNIVERSARY". Below the "25" is the acronym "SOKT" in a large, bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 7

Thiết kế phần mềm
(Thiết kế giao diện người dùng)

Mục tiêu của bài học

Sinh viên sẽ được trang bị các kiến thức sau:

- Các khái niệm liên quan tới Thiết kế giao diện người dùng
- Quy trình thiết kế giao diện và các vấn đề liên quan
- Đánh giá thiết kế giao diện

Nội dung

Thiết kế giao diện người dùng

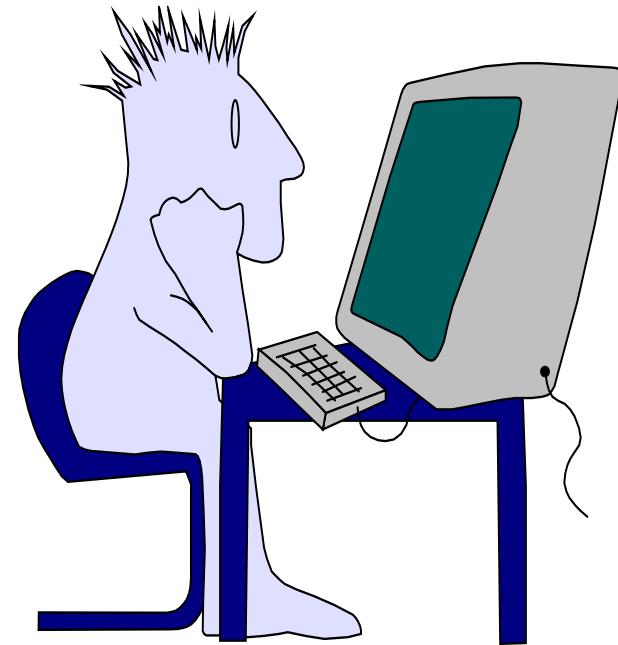
- 1. Các vấn đề thiết kế**
2. Quy trình thiết kế UI
3. Phân tích người dùng
4. Tạo mẫu thử giao diện, mẫu thử tương tác
5. Đánh giá UI
6. Các công cụ thiết kế UI

Thiết kế giao diện

Dễ học?

Dễ sử dụng?

Dễ hiểu?



Thiết kế giao diện

Lỗi thiết kế thông thường

- Thiếu nhất quán
- Quá nhiều ghi nhớ
- Không có hướng dẫn / giúp đỡ
- Không nhạy cảm với ngữ cảnh
- Đáp ứng kém
- Phức tạp / không thân thiện



Nguyên tắc cơ bản trong thiết kế giao diện

- *Dễ học:* Phần mềm cần phải dễ học cách sử dụng, do đó người dùng có thể nhanh chóng bắt đầu làm việc sử dụng phần mềm đó
- *Quen thuộc với người sử dụng:* Giao diện nên dùng các thuật ngữ và khái niệm rút ra từ kinh nghiệm của những người sẽ dùng hệ thống nhiều nhất
- *Tính nhất quán:* giao diện cần nhất quán sao cho các thao tác gần giống nhau có thể được kích hoạt theo cùng kiểu.
- *Ngạc nhiên tối thiểu:* Người dùng không bao giờ bị bất ngờ về hành vi của hệ thống

Nguyên tắc cơ bản trong thiết kế giao diện

- *Khôi phục được:* Giao diện nên có các cơ chế cho phép người dùng khôi phục lại tình trạng hoạt động bình thường sau khi gặp lỗi
- *Hướng dẫn người dùng:* Giao diện nên có phản hồi có nghĩa khi xảy ra lỗi và cung cấp các tiện ích trợ giúp theo ngữ cảnh
- *Người dùng đa dạng:* Giao diện nên cung cấp các tiện ích tương tác thích hợp cho các loại người dùng hệ thống khác nhau

Quy tắc vàng

- Đặt người dùng trong sự kiểm soát
- Giảm tải bộ nhớ cho người dùng
- Làm cho giao diện nhất quán



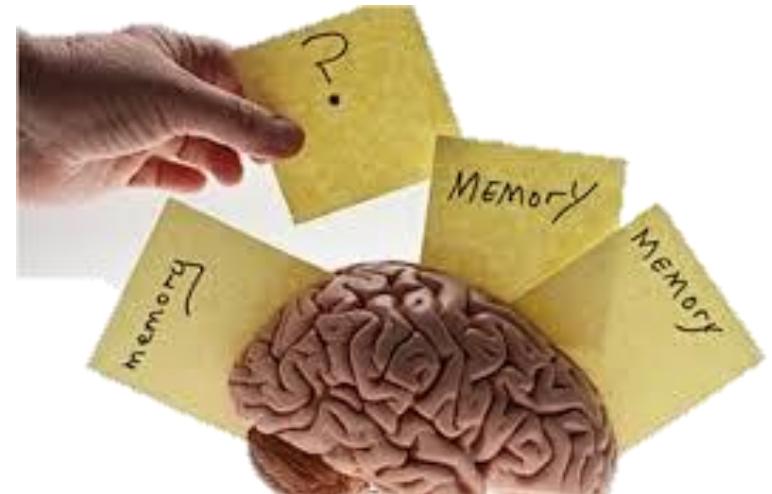
Source: Martin, R., "Design Principles and Design Patterns," downloaded from <http://www.objectmentor.com>, 2000.

Đặt người dùng trong sự kiểm soát

- Xác định phương thức tương tác theo một cách mà không ép người dùng tới những hành động không cần thiết hoặc không mong muốn.
- Cung cấp sự tương tác linh hoạt.
- Cho phép tương tác người dùng được ngắt và hoàn tác .
- Hợp lý hóa tương tác như trình độ kỹ năng cao và cho phép tùy chỉnh tương tác.
- Ẩn kỹ thuật bên trong với người sử dụng bình thường.
- Thiết kế cho tương tác trực tiếp với các đối tượng xuất hiện trên màn hình.

Giảm tải bộ nhớ cho người dùng

- Giảm nhu cầu về bộ nhớ ngắn hạn.
- Thiết lập các mặc định có ý nghĩa.
- Xác định các phím tắt trực quan.
- Các thiết kế trực quan của giao diện phải được dựa trên một phép ẩn dụ thế giới thực.
- Tiết lộ thông tin theo kiểu lũy tiến.



Làm cho giao diện nhất quán

- Cho phép người sử dụng đưa các tác vụ hiện hành vào một ngữ cảnh có ý nghĩa.
- Duy trì tính nhất quán giữa một họ các ứng dụng.
- Nếu mô hình tương tác cũ đã tạo ra những kỳ vọng của người dùng, không làm thay đổi trừ khi có một lý do thuyết phục để làm như vậy.



Mô hình thiết kế giao diện người dùng

- **Mô hình người dùng** — một hồ sơ của tất cả người dùng cuối của hệ thống
- **Mô hình thiết kế** — một nhận thức thiết kế của các mô hình sử dụng
- **Mô hình về tinh thần (nhận thức hệ thống)** — hình ảnh nhận thức của người dùng về giao diện
- **Mô hình triển khai** — giao diện "nhìn và cảm nhận" cùng với thông tin hỗ trợ mô tả cú pháp và ngữ nghĩa giao diện

Nội dung

Thiết kế giao diện người dùng

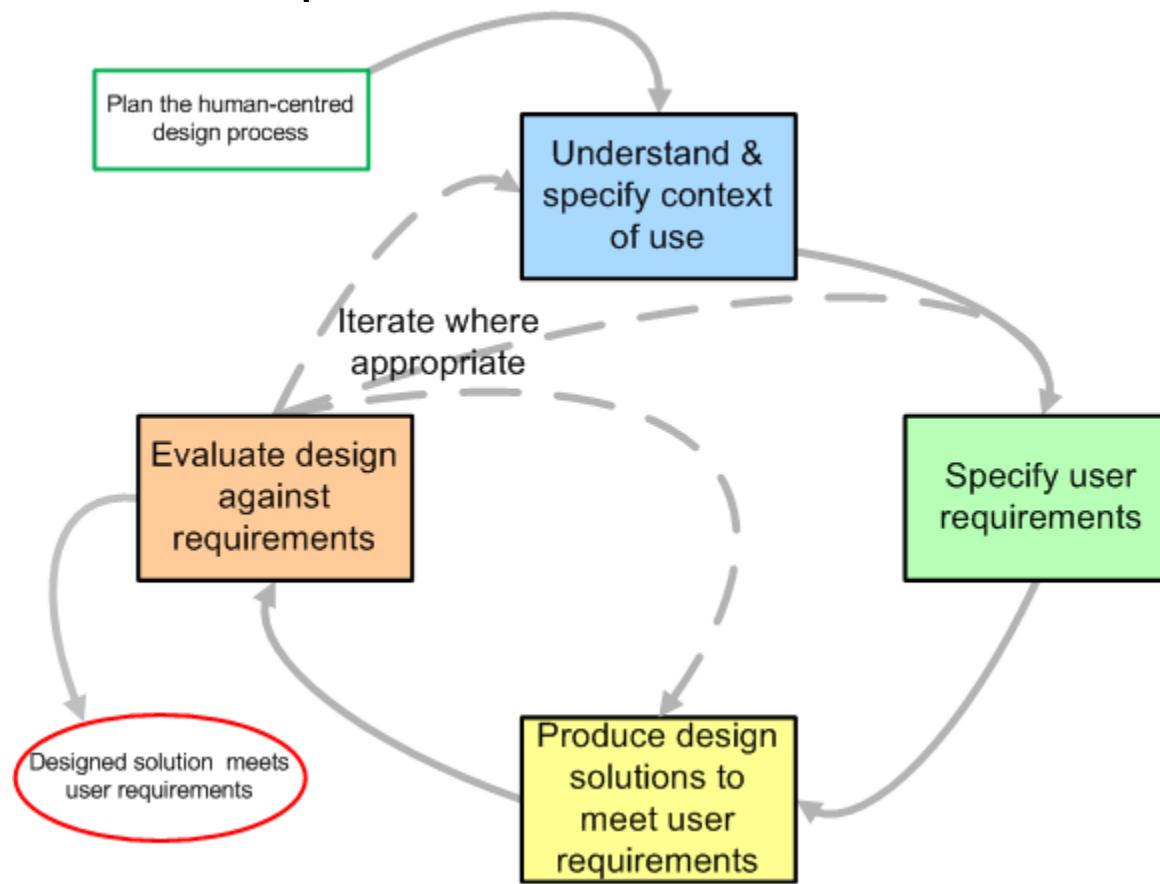
1. Các vấn đề thiết kế
- 2. Quy trình thiết kế UI**
3. Phân tích người dùng
4. Tạo mẫu thử giao diện, mẫu thử tương tác
5. Đánh giá UI
6. Các công cụ thiết kế UI

Quy trình thiết kế giao diện người dùng

- Thiết kế giao diện là một quy trình lặp đi lặp lại với sự liên lạc chặt chẽ giữa người dùng và người thiết kế. Ba hoạt động chính trong quy trình:
 - User analysis:** Tìm hiểu người dùng sẽ làm gì với hệ thống;
 - System prototyping:** phát triển một loạt các bản mẫu để thử nghiệm
 - Interface evaluation:** thử nghiệm các bản mẫu cùng với người dùng

Quy trình thiết kế giao diện người dùng

- Quy trình thiết kế lặp theo ISO 9241 - 210



Nội dung

Thiết kế giao diện người dùng

1. Các vấn đề thiết kế
2. Quy trình thiết kế UI
- 3. Phân tích người dùng**
4. Tạo mẫu thử giao diện, mẫu thử tương tác
5. Đánh giá UI
6. Các công cụ thiết kế UI

Phân tích giao diện

- Phân tích giao diện nghĩa là hiểu được:
 - (1) những người (người dùng cuối), người sẽ tương tác với các hệ thống thông qua giao diện;
 - (2) các tác vụ mà người dùng phải thực hiện để làm công việc của họ,
 - (3) các nội dung được trình bày như là một phần của giao diện
 - (4) môi trường trong đó những công việc này sẽ được tiến hành.

Phân tích người dùng

- Liệu người dùng là các chuyên gia đã qua đào tạo, kỹ thuật viên, văn thư hay là công nhân sản xuất?
- Mức độ giáo dục trung bình của người dùng là gì?
- Liệu người dùng có thể học từ những tài liệu viết hay họ bày tỏ mong muốn một chương trình luyện tập kiểu lớp học?
- Người dùng là những chuyên gia đánh máy hay họ sơ bàn phím?
- Độ tuổi của cộng đồng người dùng là bao nhiêu?
- Liệu người dùng sẽ được đại diện chủ yếu bởi một giới tính?
- Người dùng sẽ trả cho công việc họ thực hiện như thế nào?

Phân tích người dùng

- Liệu người dùng chỉ làm việc trong giờ công sở hay họ sẽ làm việc đến khi công việc hoàn thành?
- Liệu phần mềm sẽ trở thành một phần quan trọng trong công việc của người dùng hay nó sẽ chỉ thỉnh thoảng được sử dụng?
- Ngôn ngữ chính giữa các người dùng là gì?
- Sẽ có những hệ quả nào nếu người dùng gây ra lỗi khi sử dụng hệ thống?
- Liệu người dùng có phải là chuyên gia trong lĩnh vực liên quan được xử lý bởi hệ thống?
- Liệu người dùng có muốn biết về công nghệ phía sau giao diện?

Phân tích tác vụ và mô hình hóa

- Trả lời những câu hỏi sau...
 - Công việc gì mà người dùng sẽ thực hiện trong những trường hợp cụ thể?
 - Tác vụ hay tác vụ con nào sẽ được thực hiện khi người dùng làm việc?
 - Những vấn đề miền đối tượng cụ thể nào mà người dùng sẽ thao tác khi công việc được thực hiện?
 - Chuỗi các nhiệm vụ-quy trình là gì?
 - Hệ thống cấp bậc của tác vụ là gì?
- Use-cases xác định tương tác cơ bản.
- Xây dựng tác vụ điều chỉnh các nhiệm vụ tương tác.
- Xây dựng đối tượng xác định đối tượng giao diện (lớp)
- Phân tích quy trình làm việc xác định cách một quá trình làm việc được hoàn thành khi một số người (và vai trò) đều tham gia

Sơ đồ hoạt động (Swimlane)

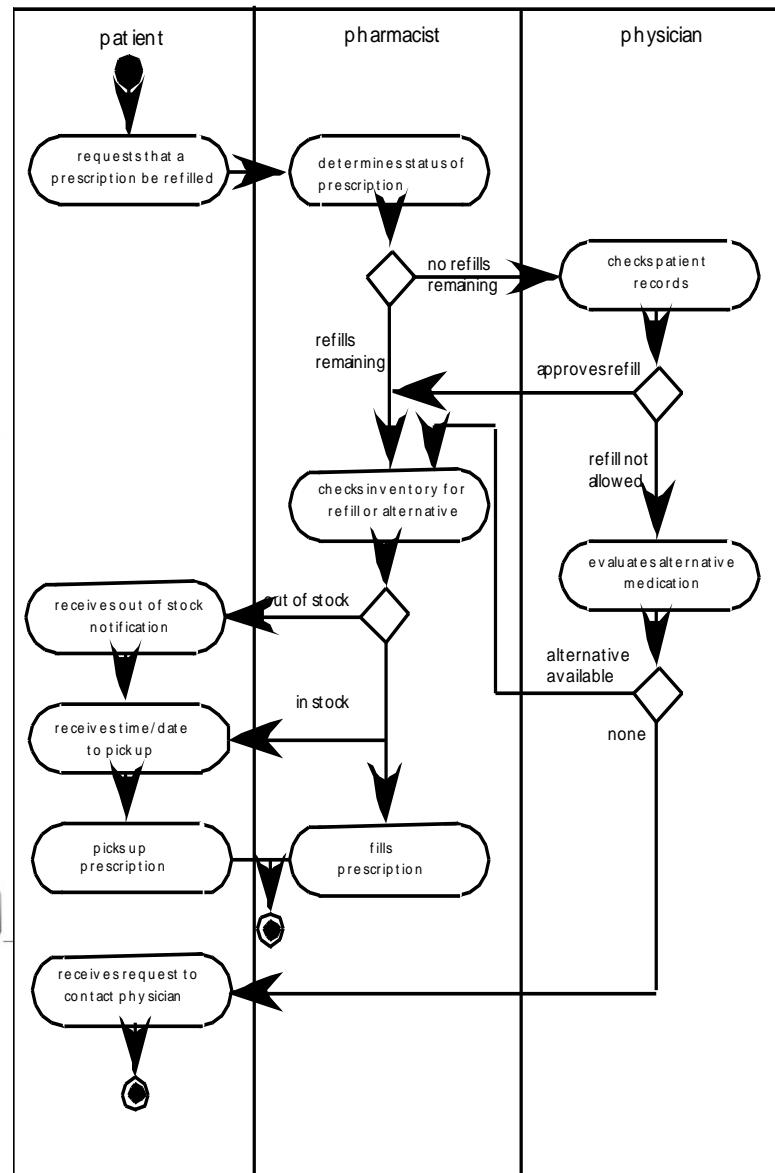
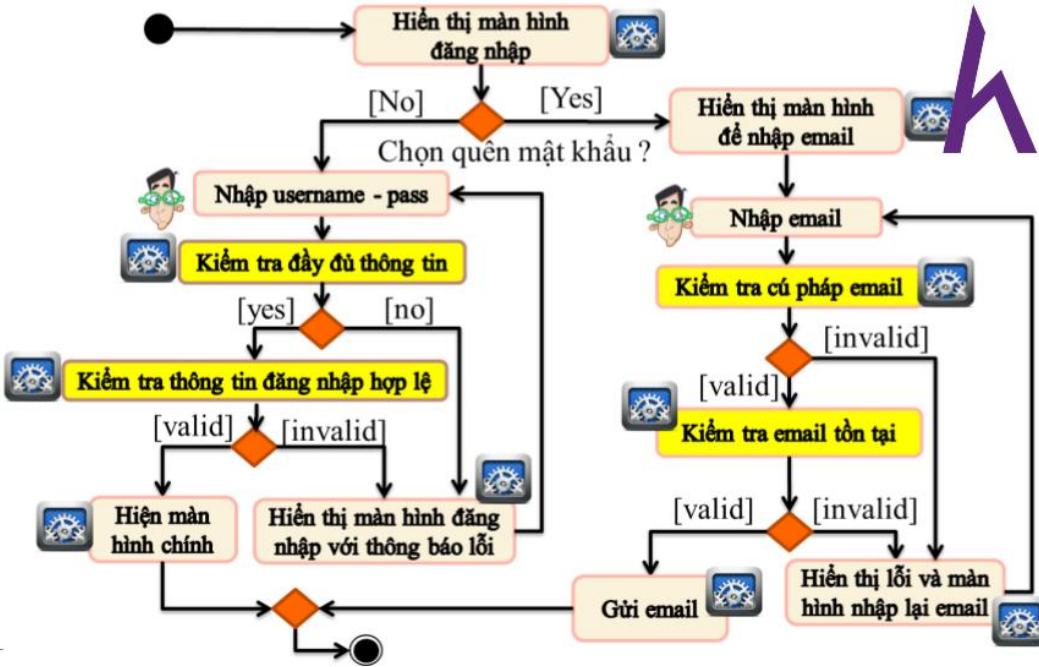


Figure 12.2 Swimlane diagram for prescription refill function

Phân tích nội dung hiển thị

- Liệu các loại dữ liệu khác nhau có được gán cho vị trí địa lý nhất định trên màn hình (ví dụ, hình ảnh luôn luôn xuất hiện ở góc trên bên phải)?
- Liệu người dùng có thể tùy chỉnh vị trí màn hình cho nội dung?
- Liệu các nhận dạng phù hợp có được gán cho tất cả nội dung?
- Nếu một báo cáo lớn được trình bày, nó sẽ được phân chia như thế nào cho dễ hiểu?
- Liệu cơ chế có sẵn sàng để di chuyển trực tiếp tới thông tin tóm tắt cho lượng dữ liệu lớn?
- Liệu các đầu ra đồ họa có được căn chỉnh để vừa vặn với các giới hạn của thiết bị hiển thị được sử dụng?
- Màu sắc sẽ được sử dụng như thế nào để tăng tính dễ hiểu?
- Thông báo lỗi và cảnh báo sẽ được trình bày tới người dùng như thế nào?

Nội dung

Thiết kế giao diện người dùng

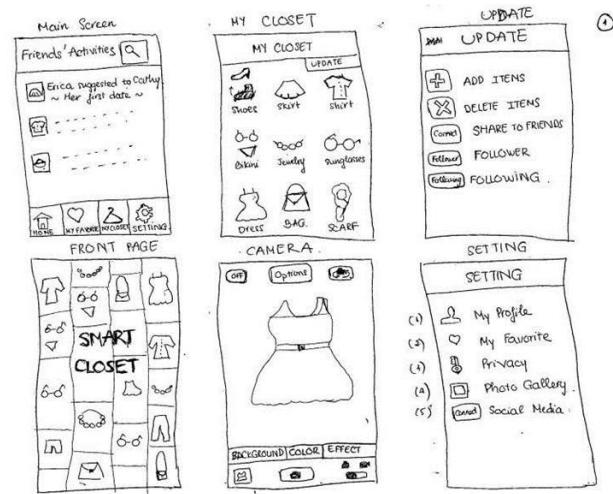
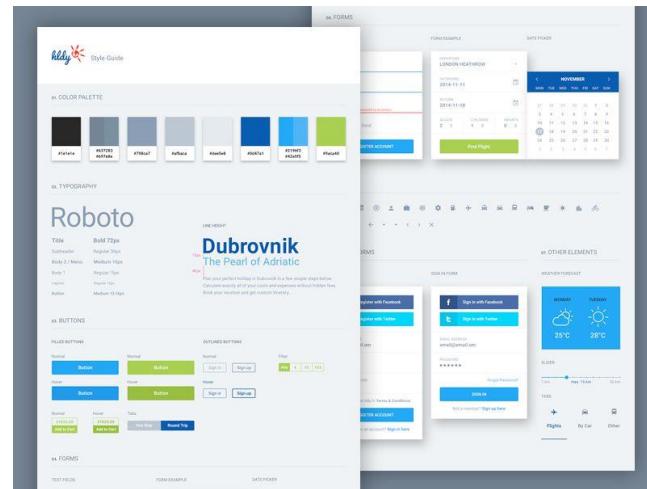
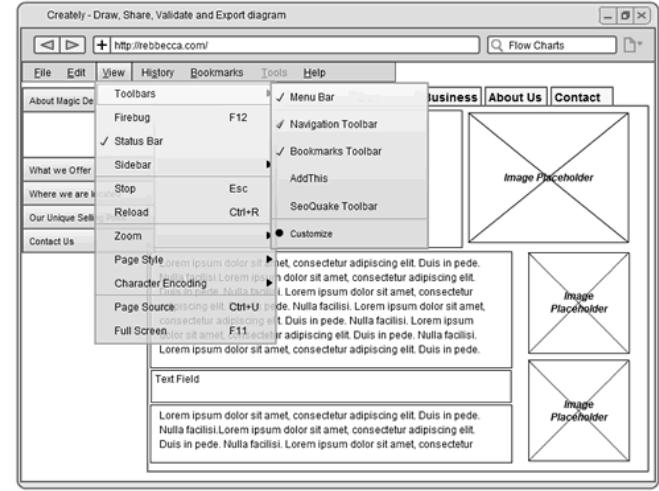
1. Các vấn đề thiết kế
2. Quy trình thiết kế UI
3. Phân tích người dùng
- 4. Tạo mẫu thử giao diện, mẫu thử tương tác**
5. Đánh giá UI
6. Các công cụ thiết kế UI

Tạo mẫu thử giao diện

- Sử dụng thông tin được phát triển trong quá trình phân tích giao diện, xác định đối tượng giao diện và hành động (hoạt động).
- Xác định các sự kiện (các hành động người dùng) sẽ gây ra sự thay đổi trạng thái của giao diện người dùng. Mô hình hóa hành vi này.
- Miêu tả mỗi trạng thái giao diện như thể nó sẽ thực sự tìm đến người dùng cuối.
- Xác định cách người dùng diễn giải các trạng thái của hệ thống từ thông tin được cung cấp thông qua giao diện.

Tạo mẫu thử giao diện

- Wire-frames
 - Bản phác thảo bút chì ban đầu
 - Làm sạch Wire-frames (công cụ phần mềm)
- Lo-Fi Prototype
- Application Style-Guide



Các vấn đề thiết kế

- Thời gian trả lời
- Trợ giúp các tiện nghi
- Xử lý lỗi
- Gắn nhãn menu và câu lệnh
- Khả năng tiếp cận ứng dụng
- Quốc tế hóa



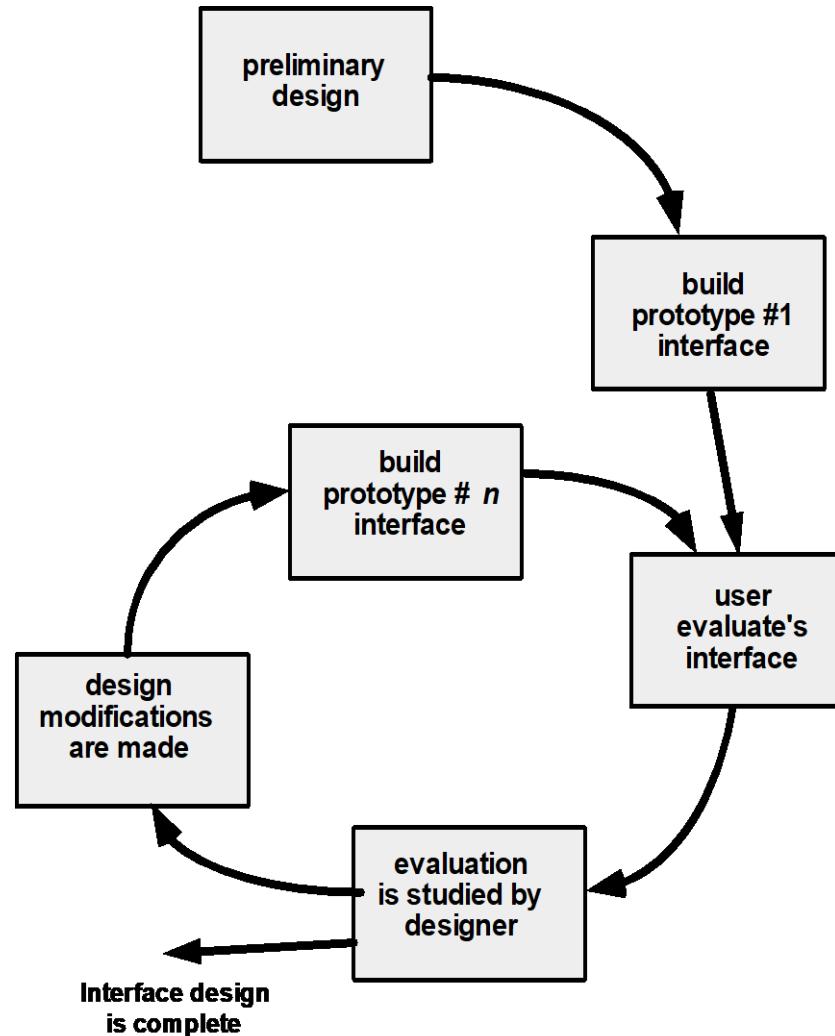
ĐĂNG KÝ NHÃN HIỆU QUỐC TẾ

Nội dung

Thiết kế giao diện người dùng

1. Các vấn đề thiết kế
2. Quy trình thiết kế UI
3. Phân tích người dùng
4. Tạo mẫu thử giao diện, mẫu thử tương tác
- 5. Đánh giá UI**
6. Các công cụ thiết kế UI

Quy trình đánh giá thiết kế UI



Quy trình đánh giá thiết kế UI

- Cần thực hiện một số đánh giá UI để đánh giá mức độ thích hợp
- Đánh giá đầy đủ và toàn bộ thì quá đắt và không thực tế cho hầu hết các hệ thống
- Một giao diện cần được đánh giá theo một đặc tả về tính sử dụng.

Quy trình đánh giá thiết kế UI

- Các thuộc tính về tính sử dụng

Thuộc tính	Mô tả
Khả năng học	Người dùng mới cần bao lâu để có thể hoạt động hiệu quả với hệ thống?
Tốc độ vận hành	Tốc độ phản ứng của hệ thống có đáp ứng tốt công việc của người dùng?
Chịu lỗi	Mức độ dung thứ lỗi của hệ thống đối với lỗi người dùng.
Khả năng khôi phục	Khả năng hệ thống khôi phục từ lỗi của người dùng.
Tương thích	Hệ thống gắn bó chặt chẽ với một kiểu làm việc đến đâu?

Quy trình đánh giá thiết kế UI

- Kỹ thuật đánh giá và phân tích:
 - Câu hỏi điều tra để lấy phản hồi của người dùng.
 - Quay video về việc sử dụng hệ thống rồi sau đó đánh giá nội dung.
 - Cài các đoạn mã thu thập thông tin về các tiện ích được sử dụng và lỗi của người dùng.
 - Phần mềm có chức năng thu thập phản hồi trực tuyến của người dùng.

Khuôn mẫu giao diện người dùng (UI Patterns)

- **Giao diện người dùng tổng thể (Whole UI).** Cung cấp hướng dẫn thiết kế cho các cấu trúc cấp cao nhất và điều hướng trong suốt toàn bộ giao diện.
- **Bố cục trang.** Giải quyết các tổ chức chung của trang (cho các trang web) hay hiển thị màn hình riêng biệt (cho các ứng dụng tương tác)
- **Biểu mẫu và đầu vào.** Xem xét nhiều kỹ thuật thiết kế cho việc hoàn thành đầu vào dạng biểu mẫu.
- **Bảng.** Cung cấp hướng dẫn thiết kế cho việc tạo và xử lý dữ liệu dạng bảng dưới mọi dạng.
- **Xử lý dữ liệu trực tiếp.** Address data editing, modification, and transformation.

Khuôn mẫu giao diện người dùng (UI Patterns)

- **Điều hướng.** Hỗ trợ người dùng định hướng xuyên suốt menu phân cấp, các trang web và màn hình hiển thị tương tác.
- **Tìm kiếm.** Cho phép tính năng tìm kiếm nội dung cụ thể thông qua các thông tin được duy trì trong một trang Web hoặc được lưu trữ trong cơ sở dữ liệu có thể truy cập thông qua một ứng dụng tương tác..
- **Các phần tử trang.** Thực hiện các phần tử cụ thể của một trang web hoặc màn hình hiển thị.
- **Thương mại điện tử.** Tùy vào các trang web, các mô hình triển khai các yếu tố định kỳ của các ứng dụng thương mại điện tử.

Nội dung

Thiết kế giao diện người dùng

1. Các vấn đề thiết kế
2. Quy trình thiết kế UI
3. Phân tích người dùng
4. Tạo mẫu thử giao diện, mẫu thử tương tác
5. Đánh giá UI
- 6. Các công cụ thiết kế UI**

Các công cụ hỗ trợ thiết kế giao diện

Traditional	Desktop	Mobile
Paper and Pen	Sketch App	Pop
Post it Notes	Balsamiq	App Cooker
UI Stencils (uistencils.com)	Visio	Blueprint
...	Omnigraffle	Interface HD
	Axure	Adobe Proto
	Just in Mind	iMockups
	Easel	SketchyPad
	Skeleton (HTML Prototyping)	Livewires
	App Sketcher (HTML Prototyping)	Launch
	Adobe Brackets (Coding)	Briefscase

Ví dụ: Thiết kế giao diện WebApp

- **Giao diện cần**
 - Cung cấp dấu hiệu truy cập
 - Thông báo cho người sử dụng vị trí của họ trong nội dung phân cấp
- **Giao diện luôn giúp người dùng hiểu tùy chọn hiện tại của họ**
 - Những chức năng nào khả dụng
 - Những liên kết nào còn sử dụng được
 - Những nội dung nào có liên quan
- **Giao diện cần tạo điều kiện chuyển hướng**
 - Cung cấp một “bản đồ” dễ hiểu để người dùng có thể chuyển hướng trong ứng dụng

Ví dụ: Thiết kế giao diện WebApp

- Bruce Tognazzi [TOG01] đề nghị Giao diện WebApp hiệu quả:
 - **Giao diện hiệu quả cần trực quan**, người sử dụng có thể nhanh chóng xem các lựa chọn, nắm bắt xem làm thế nào để đạt mục tiêu và làm việc
 - **Người sử dụng không liên quan đến hoạt động bên trong của hệ thống**. Công việc thực hiện xuyên suốt và thường xuyên được lưu, với các tùy chọn đầy đủ để có thể hoàn tác.
 - **Thực hiện công việc một cách tối đa**, trong khi nhận thông tin ít nhất từ người dùng

Nguyên lý thiết kế giao diện WebApp I

- **Tiên đoán** — WebApp nên thiết kế để dự kiến được hành động tiếp theo của người dùng
- **Truyền thông** — Giao diện cần truyền thông các trạng thái của các hoạt động của người dùng
- **Nhất quán** — Sử dụng công cụ điều hướng, menu, biểu tượng
- **Kiểm soát quyền tự trị** — Giao diện tạo điều kiện cho người dùng di chuyển trong ứng dụng, theo các quy ước điều hướng được quy định trong ứng dụng
- **Hiệu quả** — Thiết kế của WebApp và giao diện nên được tối ưu hóa hiệu quả làm việc của người dùng, chứ không nên tối ưu hóa theo công việc của kỹ sư hay môi trường thực thi

Nguyên lý thiết kế giao diện WebApp II

- **Tập trung** – Giao diện và nội dung nên tập trung vào các tác vụ người dùng đang sử dụng
- **Luật Fitt** – “Thời gian đạt được mục tiêu là hàm của khoảng cách và kích thước mục tiêu”
- **Đối tượng giao diện người dùng** – Một thư viện lớn của các đối tượng giao diện tái sử dụng, phát triển cho WebApp
- **Giảm độ trễ** – Ứng dụng thực hiện đa nhiệm tuy nhiên vẫn cho phép người dùng thực hiện công việc liên tục
- **Thời gian học** – Giao diện nên thiết kế giảm thiểu thời gian học, và sau khi học để giảm thiểu thời gian học lại khi được truy cập

Nguyên lý thiết kế giao diện WebApp III

- **Duy trì kết quả công việc** — Công việc của người dùng phải được lưu lại tự động, tránh mất mát nếu có lỗi xảy ra
- **Dễ đọc** — Mọi thông tin được trình bày phải dễ đọc với mọi người
- **Theo dõi trạng thái** — Khi thích hợp, các trạng thái tương tác của người dùng nên được theo dõi và ghi lại, để người dùng có thể đăng xuất và sau đó quay lại trang thái làm việc cuối của họ.
- **Điều hướng** — Một giao diện tốt cung cấp các điều hướng rõ ràng, bố trí hợp lý

Tính thẩm mỹ

- Không ngại các khoảng trắng
- Nhấn mạnh nội dung
- Tổ chức các thành phần từ góc trên bên trái xuống góc dưới bên phải
- Nhóm các điều hướng, nội dung và các chức năng di chuyển trong trang
- Không mở rộng khung trang với thanh cuộn
- Xem xét độ phân giải và kích thước cửa sổ khi thiết kế

Bước 1. Xác định yêu cầu khách hàng

- Các nội dung
 - Khái quát chức năng, cấu trúc nội dung, về giao diện, đối tượng sử dụng.
 - Xác định các quy trình nghiệp vụ trong hệ thống
 - Sắp xếp độ ưu tiên của các yêu cầu
 - Đánh giá khách quan các chức năng và hiệu năng

Bước 1. Xác định yêu cầu khách hàng

- Câu hỏi

- Sau 3 năm nữa website sẽ phục vụ mục đích gì?
- Liệt kê các tính năng mà bạn (khách hàng) nghĩ ra theo nhóm: bắt buộc, mong muốn, tùy chọn
- Cho biết 3 website mà bạn ưa thích và cảm nhận của bạn về những website đó.

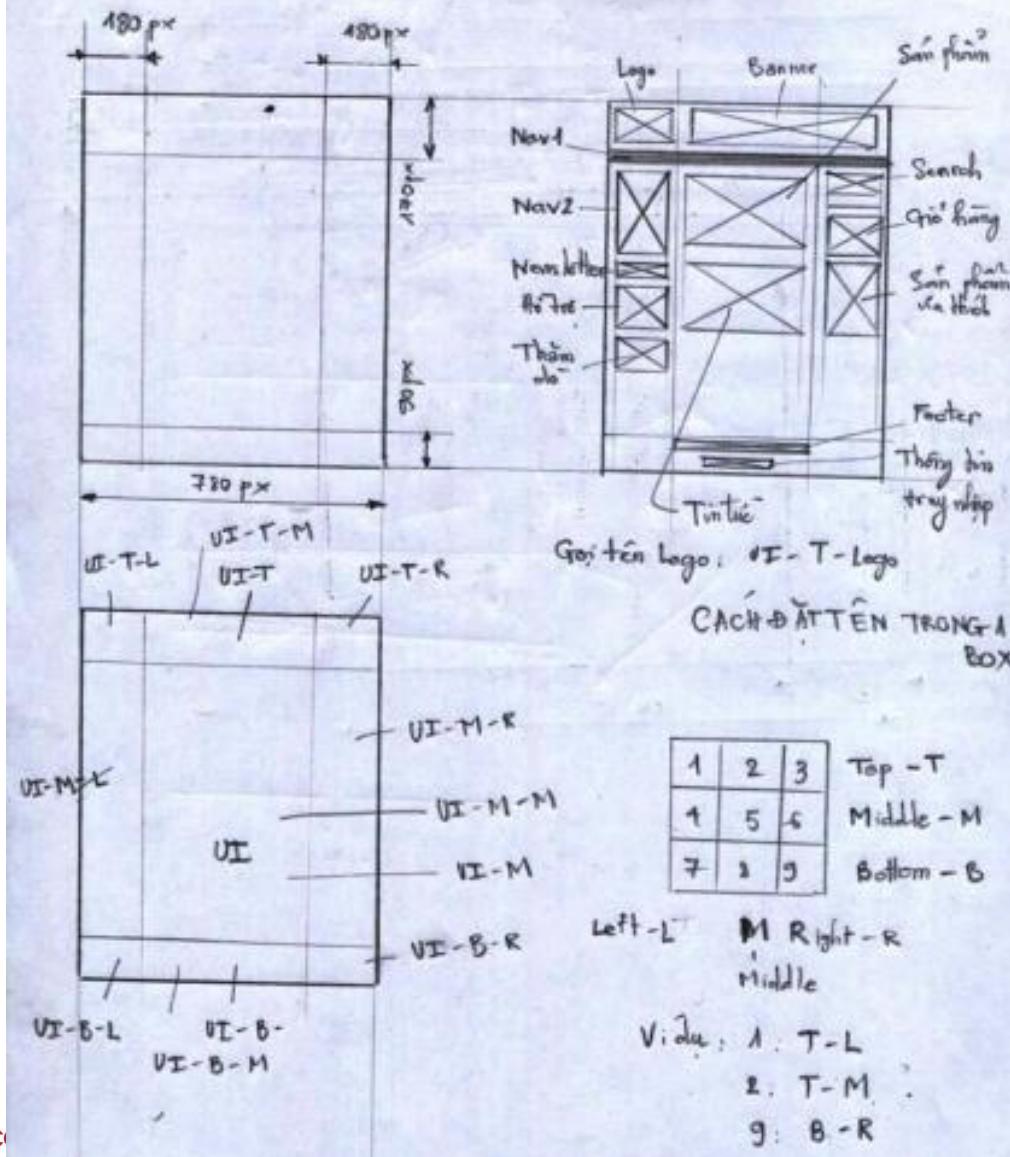
Bước 2. Phác thảo ý tưởng trên giấy

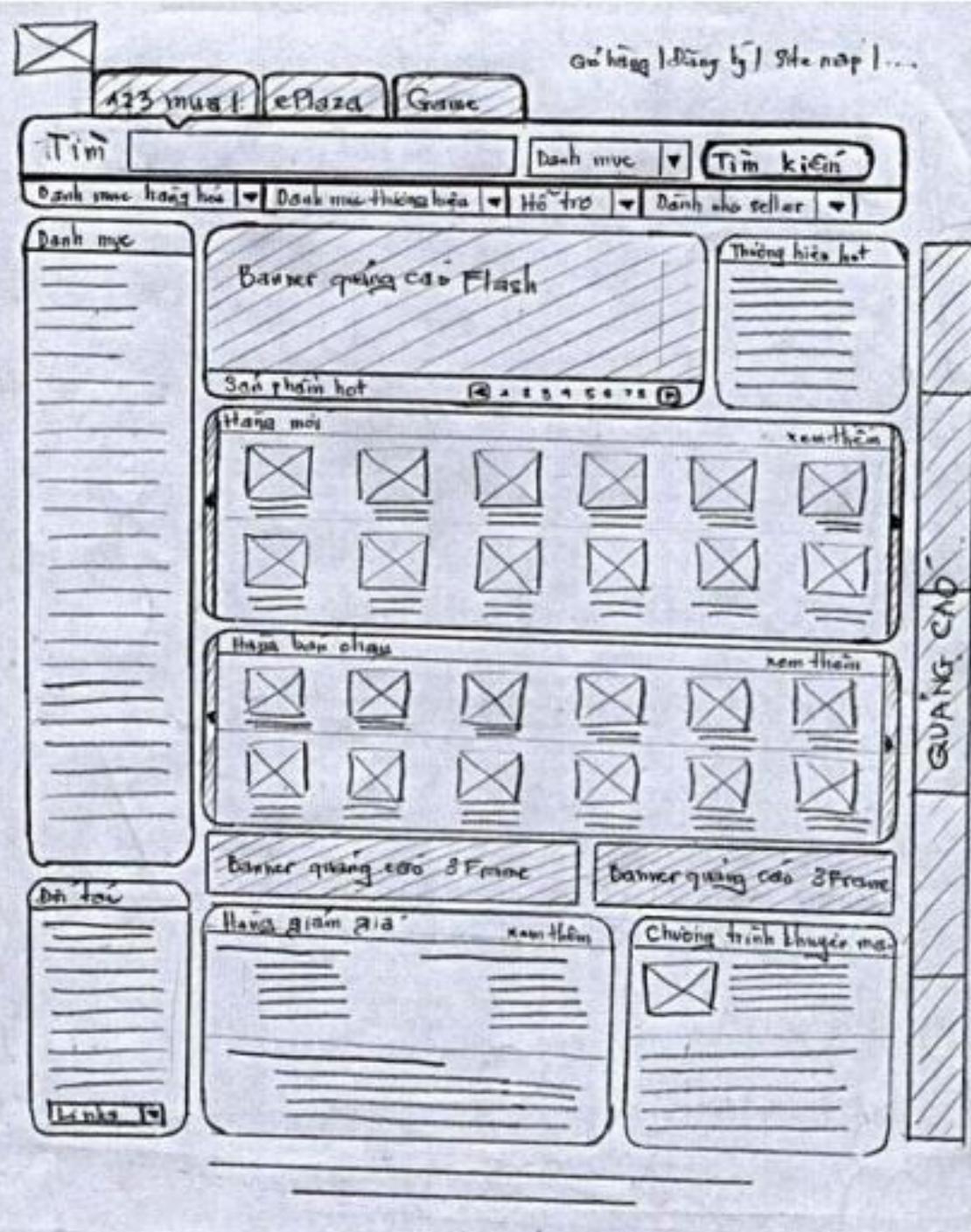
- Mục đích
 - Định hình bố cục trang web
- Thực hiện
 - Sử dụng bút chì, tẩy, thước kẻ để vẽ phác thảo ý tưởng của mình
 - Dựa vào kinh nghiệm của bản thân đưa ra các tiêu chuẩn nên có.
 - Chia trang web làm 2 phần: vùng template theo chuẩn và vùng hiệu chỉnh

Bước 2. Phác thảo ý tưởng trên giấy

- Vùng template: là vùng thay đổi rất ít trong các trang web
- Vùng hiệu chỉnh: là vùng thay đổi nội dung giữa các trang web.
- Quy chuẩn các đối tượng để dễ trao đổi và lập trình
 - Ảnh là hình chữ nhật: đánh dấu X
 - Dòng văn bản: đường kẻ ngang

CÁCH ĐẶT TÊN & TRUY XUẤT GIAO DIỄN





Bước 3. Đánh giá mẫu phác thảo

- Mục đích
 - Đánh giá mẫu phác thảo nào phù hợp với yêu cầu, mong muốn của khách hàng
- Thực hiện
 - Đưa ra tối thiểu 3 mẫu thiết kế
 - Dán lên tường, mời mọi người cùng xem và góp ý
 - Đáp ứng yêu cầu khách hàng?
 - Thông tin, chức năng dễ tìm?
 - Bố cục gắn kết, thẩm mỹ?

Bước 4. Thiết kế đồ họa bản đơn sắc

- Mục đích

Đánh giá bản phác thảo trên giấy khi chuyển sang đồ họa vi tính có phù hợp với yêu cầu của khách hàng hay không

- Thực hiện

- Sử dụng các công cụ thiết kế đồ họa như: PhotoShop, VS 2010... để thiết kế mẫu giao diện web
- Chưa thực hiện phối màu cho các thành phần, để ở màu xám
- Tuyệt đối không sử dụng hai màu đen, trắng ở vùng muốn phối màu

TÌM MUA

Trang chủ

Danh mục hàng hóa

Danh mục thương hiệu

Hỗ trợ

Dành

Danh mục hàng hóa

Công nghệ số

Nghe nhìn

Thiết bị Viễn thông

Đồ chơi giải trí

Hoa

Quà tặng

May mặc

Thời trang

Mỹ phẩm

Làm đẹp

Trang sức

Đồng hồ

Mắt kính

Phim

Nhạc

Phần mềm

Thể thao

Chăm sóc sức khỏe

Văn phòng phẩm

Tranh

Ảnh nghệ thuật



Sản phẩm được ưa chuộng

< 1 2 3 4

Hàng mới



Cặp đi học MU (đen)

GIÁ: 80.000 VND



Nón bảo hiểm Haly Evo

GIÁ: 128.000 VND



TÌM HIỂU

Tìm kiếm...

Danh mục hàng hóa

Công nghệ số

Nghe nhìn

Thiết bị Viễn thông

Đồ chơi giải trí

Hoa

Quà tặng

May mặc

Thời trang

Hộ phẩm

Làm đẹp

Trang sức

Đồng hồ

Hút kim

Phim

Nhạc

Phần mềm

Thể thao

Chăm sóc sức khỏe

Văn phòng phẩm

Tranh

Ảnh nghệ thuật

Thực phẩm

Túi đồng

Dịch vụ nổi bật

Giao hàng trực tuyến

+ Quản lý giao hàng

+ Quản lý tài khoản

Danh mục quảng cáo

+ Quảng cáo với 123mua!

+ Hướng dẫn quảng cáo

Thứ bắc của 123mua!



- + iGURU - Web designer
- + Võ Lâm Truyền Kỳ
- + PC World Việt Nam
- + Báo điện tử Vietnamnet
- + Báo điện tử Dân Trí
- + Vina Game

VIỆN CÔ

In Theaters Now
SAAWARIYA
www.sawariya.com

Sản phẩm đang xác định

10 thương hiệu hot

- + iGURU - Web designer
- + Võ Lâm Truyền Kỳ
- + PC World Việt Nam
- + Báo điện tử Vietnamnet
- + Báo điện tử Dân Trí

Hàng mới

Xem thêm

Cặp đi học M&M (Đen)
Giá: 100.000 VNĐHộp bảo hành Holy Eve
Giá: 120.000 VNĐBộ drap giường và gối vải bay
Giá: 1.300.000 VNĐCặp đi học M&M (Đen)
Giá: 100.000 VNĐHộp bảo hành Holy Eve
Giá: 120.000 VNĐBộ drap giường và gối vải bay
Giá: 1.300.000 VNĐ

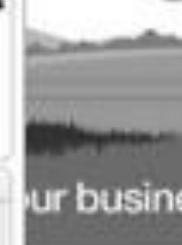
Xem hàng mới tại: Công nghệ số | Nghe nhìn | Đồ chơi giải trí | Phim | Nhạc

Hàng bán chạy

Xem thêm

Cặp đi học M&M (Đen)
Giá: 100.000 VNĐHộp bảo hành Holy Eve
Giá: 120.000 VNĐBộ drap giường và gối vải bay
Giá: 1.300.000 VNĐCặp đi học M&M (Đen)
Giá: 100.000 VNĐHộp bảo hành Holy Eve
Giá: 120.000 VNĐBộ drap giường và gối vải bay
Giá: 1.300.000 VNĐ

Xem hàng bán chạy tại: Văn phòng phẩm | Tranh | Ảnh nghệ thuật | Thực phẩm | Túi đồng



Holiday Gift Guide

Our Ultimate Gift Guide to Inspire & Surprise Everyone on Your List



Hàng giảm giá

Ngày 11 tháng 11 năm 2007

Cặp đi học MU (đen)
GIÁ: 80.000.000đ
chỉ còn: 75.000 VND

Khung hình kỹ thuật số
GIÁ: 2.000.000 VND
chỉ còn: 75.000 VND

Thẻ VisaVoice 60.000 VND
GIÁ: 80.000.000đ
chỉ còn: 75.000 VND

Bộ drap giường và gối vẽ tay
GIÁ: 2.000.000 VND
chỉ còn: 75.000 VND

Xem thêm

Khuyến mại lớn

Xem thêm



Sản phẩm thuộc
chương trình: Điện
Tử Thiên Đạt
Giá sản phẩm:
1.725.000 VND

* CÔNG TY CP Tập Đoàn Dược Phẩm và

Bước 4. Thiết kế đồ họa bản đơn sắc

- Thực hiện (tiếp)

- In và dán lên tường. Sau đó mời mọi người đến nhận xét.
- Thông tin quan trọng có dễ tìm với màn hình thực không?
- Giao diện có dễ đọc, dễ thực hiện với người dùng không?
- Giao diện có thể hiện ra tính cách riêng không?

Bước 5. Phối màu cho giao diện web

- Mục đích

Phối màu cho các thành phần đơn sắc

- Thực hiện

- Tuân thủ các phương pháp như sau
- Từ yêu cầu khách hàng đưa ra 1 màu chủ đạo, 1 màu thứ cấp và một mảng các màu hỗ trợ để tăng tính sinh động
- Với text nên có tối đa 3 màu, 3 font
- Giai đoạn phối màu rất dễ bị ảnh hưởng bởi màu của ảnh như banner...
- Chọn ảnh truyền đạt chính xác thông điệp của trang Web.

TÌM MUA



Trang chủ

Danh mục hàng hóa

Danh mục thương hiệu

Hỗ trợ

Dành

Danh mục hàng hóa

Công nghệ số

Nghe nhìn

Thiết bị viễn thông

Đồ chơi giải trí

Hoa

Quà tặng

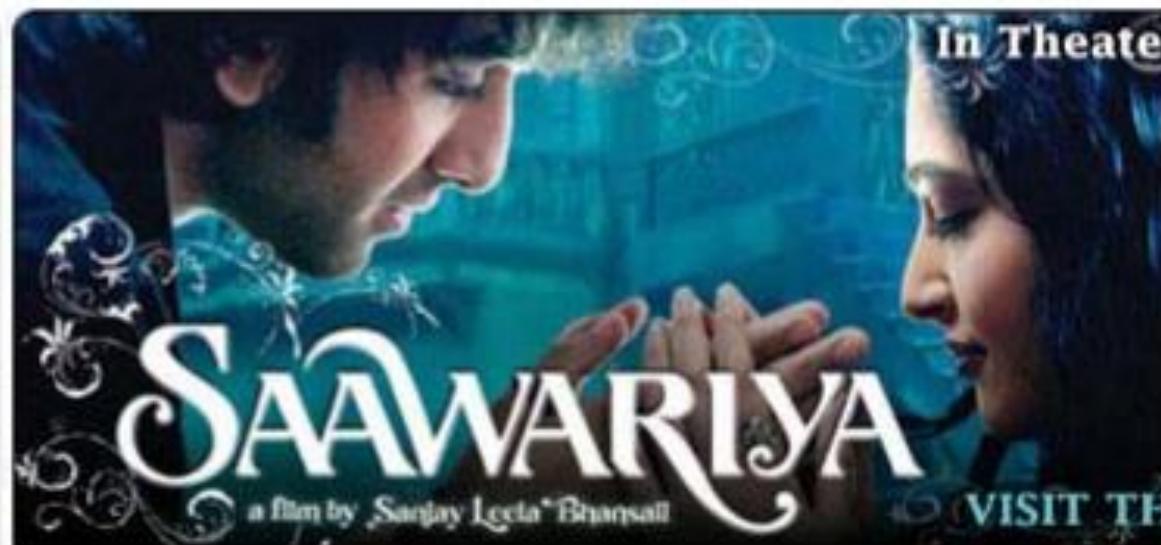
May mặc

Thời trang

Mỹ phẩm

Làm đẹp

Trang sức



Sản phẩm được ưa chuộng

1 2 3 4

Hàng mới



Phần mềm

Thể thao

Chăm sóc sức khỏe

Văn phòng phẩm

Tranh

Ảnh nghệ thuật

Thực phẩm

Tiêu dùng

Dịch vụ nổi bật

Gian hàng trực tuyến

▶ Quản lý gian hàng

▶ Quản lý tài khoản

Dành cho quảng cáo

▶ Quảng cáo với iGURU!mua

▶ Hướng dẫn quảng cáo

Đối tác của iGURU!mua



- ▶ iGURU - Web đẹp
- ▶ Võ lâm truyền kỳ
- ▶ PC World Việt Nam
- ▶ Báo điện tử Vietnamnet
- ▶ Báo điện tử Dân Trí
- ▶ Vina Game

Giá: 80.000 VNĐ

Giá: 128.000 VNĐ



Cặp đi học MU (đen)

Giá: 80.000 VNĐ

Nón bảo hiểm Haly Evo

Giá: 128.000 VNĐ

Xem hàng mới tại: [Công nghệ số](#) | [Nghe nhìn](#) | [Đồ chơi giải trí](#) ||

Hàng bán chạy



Cặp đi học MU (đen)

Giá: 80.000 VNĐ

Nón bảo hiểm Haly Evo

Giá: 128.000 VNĐ



Cặp đi học MU (đen)

Giá: 80.000 VNĐ

Nón bảo hiểm Haly Evo

Giá: 128.000 VNĐ

Xem hàng bán chạy tại: [Văn phòng phẩm](#) | [Tranh](#) | [Ảnh nghệ thuật](#)

Holiday *Gift* Guide

Our Ultimate Gift Guide to Inspire & Surprise Everyone on Your List

Hàng giảm giá

Xem thêm

Ngày 11 tháng 11 năm 2007

Cặp đi học MU (đen)

Giá: 80.000 VND

chỉ còn: **75.000 VND**

Khung hình kỹ thuật số

Giá: 2.053.000 VND

chỉ còn: **75.000 VND**

Mùa Cưới 07-08

Giá: 55.000 VND

chỉ còn: **75.000 VND**

Thẻ VinaVoice 60.000 VND

Giá: 60.000 VND

chỉ còn: **75.000 VND**

Bộ drap giường và gối vẽ tay

Giá: 2.053.000 VND

chỉ còn: **75.000 VND**

Mùa Cưới 07-08

Giá: 55.000 VND

chỉ còn: **75.000 VND**

Chương trình khuyến mại

- CN Cty CP Tập Đoàn Dược Phẩm và Thương mại SOHACO
- Cửa hàng Bách hóa Lưu Gia
- Thẻ cào điện thoại - Game Online

Bước 6. Xây dựng chuẩn CSS, Script, Ảnh, Folder cho trang Web

- Mục đích

Giúp quy trình xây dựng, triển khai, bảo trì ít rủi ro hơn.

- Các chuẩn

- Định nghĩa các vùng của site, các vùng trong một thẻ DIV
- Chuẩn đặt tên lớp, id trong CSS
- Chuẩn đặt tên thư mục chứa các thành phần của Web
- Chuẩn đặt tên cho các file HTML, CSS, Script...

Bước 7. Sử dụng các ngôn ngữ để thiết kế giao diện

- Mục đích

Thiết kế Web bằng HTML, CSS (JavaScript, Ajax, Flash...nếu có)

- Thực hiện

- Chỉ hiển thị dữ liệu: HTML
- Định dạng bố cục trang web: CSS
- Người thiết kế giờ thực hiện công việc lập trình. Cần hiểu rõ về ngôn ngữ sử dụng.

Bước 8. Kiểm thử trên các trình duyệt

- Mục đích

Kiểm soát việc hiển thị chính xác trang web trên các trình duyệt khác nhau như thiết kế ở bước 5.

Bước 9. Chuyển mã nguồn tới bộ phận phát triển Web

- Mục đích

Chuyển thiết kế trang Web hiển thị tốt trên các trình duyệt chính sang bộ phận lập trình

- Thực hiện

- Chuyển các yêu cầu bắt buộc và chuẩn thiết kế tới các bộ phận liên quan

Tổng kết

- Qua bài này các bạn đã nắm được cách thiết kế giao diện người dùng tốt.
- Thiết kế giao diện người dùng là một phần quan trọng quá trình thiết kế phần mềm.
- Thiết kế giao diện và xử lý tương tác với người sử dụng là một yếu tố quan trọng trong việc sử dụng phần mềm.
- Người thiết kế phải làm sao để phù hợp với kĩ năng, kinh nghiệm và mong đợi từ phía người sử dụng phần mềm.

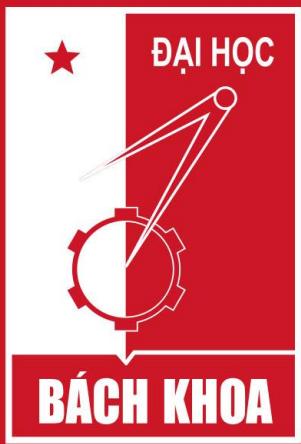


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The central focus of the banner is a large, white, stylized number "25" with a circular arc above it containing the text "YEARS ANNIVERSARY". To the right of the "25" is the acronym "SOKT" in a bold, white, sans-serif font.

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 8

Xây dựng phần mềm Software Construction



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Nội dung

1. Khái niệm
2. Quy trình xây dựng phần mềm
3. Quy ước viết mã nguồn
4. Tái cấu trúc mã nguồn
5. Rà soát mã nguồn

1. Khái niệm

- **Coding:** viết mã nguồn
 - Chuyển đổi **thiết kế chi tiết** của một hệ thống thành mã nguồn
 - Việc chuyển đổi mã này sẽ phải được thực hiện theo các cấp độ từng unit/modul/component riêng lẻ, sau đó các thành phần riêng lẻ này phải được tích hợp lại (unit/modul/component integration)
- **Documenting:** biên soạn tài liệu đi kèm với mã nguồn
 - giúp xác minh sự phù hợp giữa mã nguồn với bản đặc tả của unit/modul/component, đảm bảo chất lượng của mã nguồn
 - giúp chia sẻ các công việc có thể bị lặp lại giữa các project
- **Unit Testing:** kiểm thử đơn vị phải được thực hiện ở giai đoạn này cho từng unit/module/component một cách độc lập

Mục tiêu của việc xây dựng phần mềm

1. Thực hiện các tác vụ theo **chỉ định của thiết kế**.
2. Để **giảm chi phí** của các giai đoạn: kiểm tra và bảo trì (mã hóa hiệu quả).
3. Làm cho **mã nguồn dễ đọc, dễ hiểu** hơn: Việc mã hóa cần đảm bảo mục tiêu làm tăng khả năng hiểu mã và đọc mã trong quá trình tạo ra phần mềm dễ bảo trì.

Từ chương trình tới hệ thống

From programs to systems

- Chương trình = (cấu trúc dữ liệu + thuật toán)
- Hệ thống / Phần mềm:
 - Cấu trúc dữ liệu, giải thuật và tài liệu đặc tả
- Hệ thống có thể bao gồm nhiều chương trình, module, thành phần có kết nối hoặc tương tác với nhau
- Quy mô của hệ thống lớn hơn chương trình
- Thường được xây dựng sử dụng các thư viện, các framework có sẵn để tận dụng được nhiều thành phần xây dựng sẵn

Mô thức lập trình

Programming paradigm

- Imperative paradigm
- Logical paradigm
- Functional paradigm
- Object-oriented paradigm
- Visual paradigm
- Parallel paradigm
- Concurrent paradigm
- Distributed paradigm
- Service-oriented paradigm

Mô thức LT hướng chức năng

Functional Programming

- Nguồn gốc: lý thuyết hàm số
- Ngôn ngữ lập trình hướng chức năng miêu tả
 - Tập hợp các kiểu dữ liệu có cấu trúc
 - Tập hợp các hàm định nghĩa trên các kiểu dữ liệu đó
- Thành phần:
 - Tập hợp các cấu trúc dữ liệu và các hàm có liên quan
 - Tập hợp các hàm cơ sở
 - Tập hợp các toán tử

Mô thức LT hướng chức năng Functional Programming

- Đặc trưng cơ bản: tính module hoá chương trình - Modularity
 - Chương trình là tập hợp của các hàm (function)
 - Tính toán trong chương trình được thực hiện bằng các lời gọi hàm
 - Các chức năng khác nhau trong chương trình có thể gọi chéo hàm lẫn nhau (function sharing)
 - Khi xây dựng chương trình theo mô thức lập trình hướng chức năng, thông thường lập trình viên sẽ tiếp cận vấn đề theo hướng top-down

Mô thức LT hướng đối tượng

Object Oriented Programming

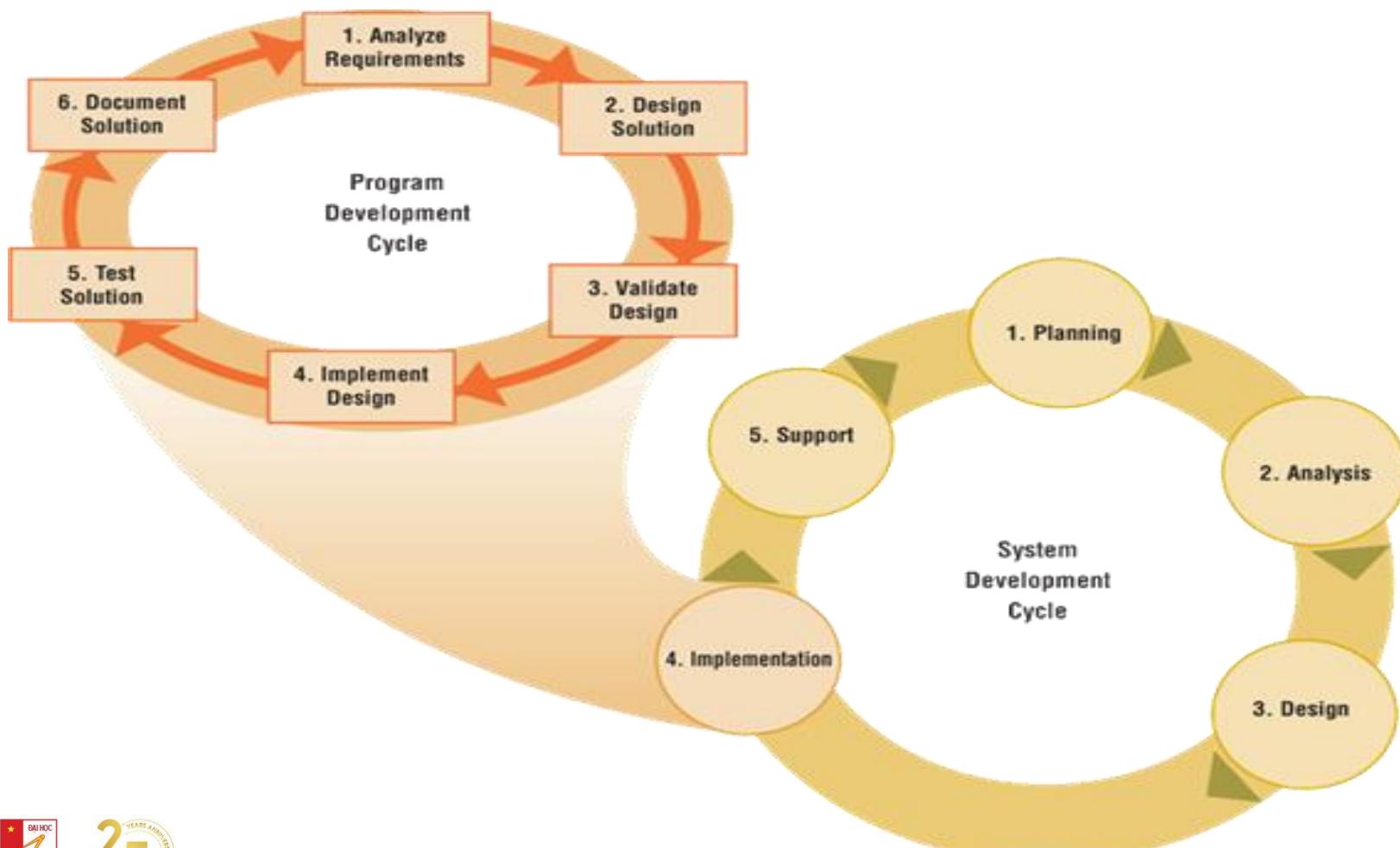
- History:
 - Simulation – Simula 67 là ngôn ngữ lập trình OOP đầu tiên
 - Interactive graphics – Smalltalk-76, lấy ý tưởng từ Simula
- Đặc trưng của mô thức lập trình HĐT:
 - Tập trung vào các khái niệm (concepts) thay vì các thao tác, hàm chức năng(operations/functions)
 - Mỗi khái niệm có thể biểu diễn một đối tượng dữ liệu trong chương trình (Book, Category, Product etc.)

Mô thức LT hướng đối tượng

Object Oriented Programming

- Một số ưu điểm của OOP so với các ngôn ngữ lập trình cấu trúc
 - Dễ dàng tái sử dụng các khái niệm trong các chương trình khác
 - Ví dụ Book trong chương trình quản lý thư viện và Book trong BookStore có thể được tái sử dụng dễ dàng
 - Dễ dàng mở rộng các khái niệm cũ (old concepts) thành các khái niệm mới hơn cho các chương trình khác
 - Ví dụ: MusicPlayer → MP3Player, WinampPlayer, DVDPlayer etc.
 - Dễ dàng hơn trong việc định vị các khu vực thay đổi khi các đối tượng dữ liệu liên quan có thay đổi (changes are localized in some code base of changed concepts)

2. Quy trình xây dựng phần mềm



Bước 1: Phân tích yêu cầu

Analyse requirements

Phân tích hệ thống

- *Dựa trên các hệ thống có thực (do con người vận hành hoặc hệ thống tự động)*
- *Do các nhà phân tích hệ thống tiến hành, sẽ hiệu quả hơn nếu phỏng vấn người dùng*

Mục tiêu

- *Xác định xem hệ thống hiện tại đã làm được những gì, làm như thế nào, còn tồn tại các vấn đề gì*

→ Quyết định xem có nên thực hiện bước tiếp theo hay không (Return-on-Investment – ROI estimation)

Bước 1: Phân tích yêu cầu

Analyse requirements

Các công việc chính

- Thiết lập các requirements
- Gặp các nhà phân tích hệ thống và users
- Xác định input, output, processing, và các thành phần dữ liệu

IPO CHART

Input	Processing	Output
Regular Time Hours Worked Overtime Hours Worked Hourly Pay Rate	Read regular time hours worked, overtime hours worked, hourly pay rate. Calculate regular time pay. If employee worked overtime, calculate overtime pay. Calculate gross pay. Print gross pay.	Gross Pay

Ví dụ

IPO chart module Register for Course



- Register for Course:

- Sinh viên mỗi đầu kì học sẽ phải đăng ký các khoá học
- Module sẽ căn cứ vào học kỳ mấy, mã số sinh viên, lịch sử học tập của sinh viên và danh sách các khoá học của học kỳ mới để có thể output ra một danh sách các môn học sinh viên có thể đăng ký
- Yêu cầu: xây dựng IPO chart cho module

Bước 2: Thiết kế giải pháp

Design solution



Object-oriented
design, thường thực hiện
theo bottom-up

Structured
design, còn gọi là
top-down design

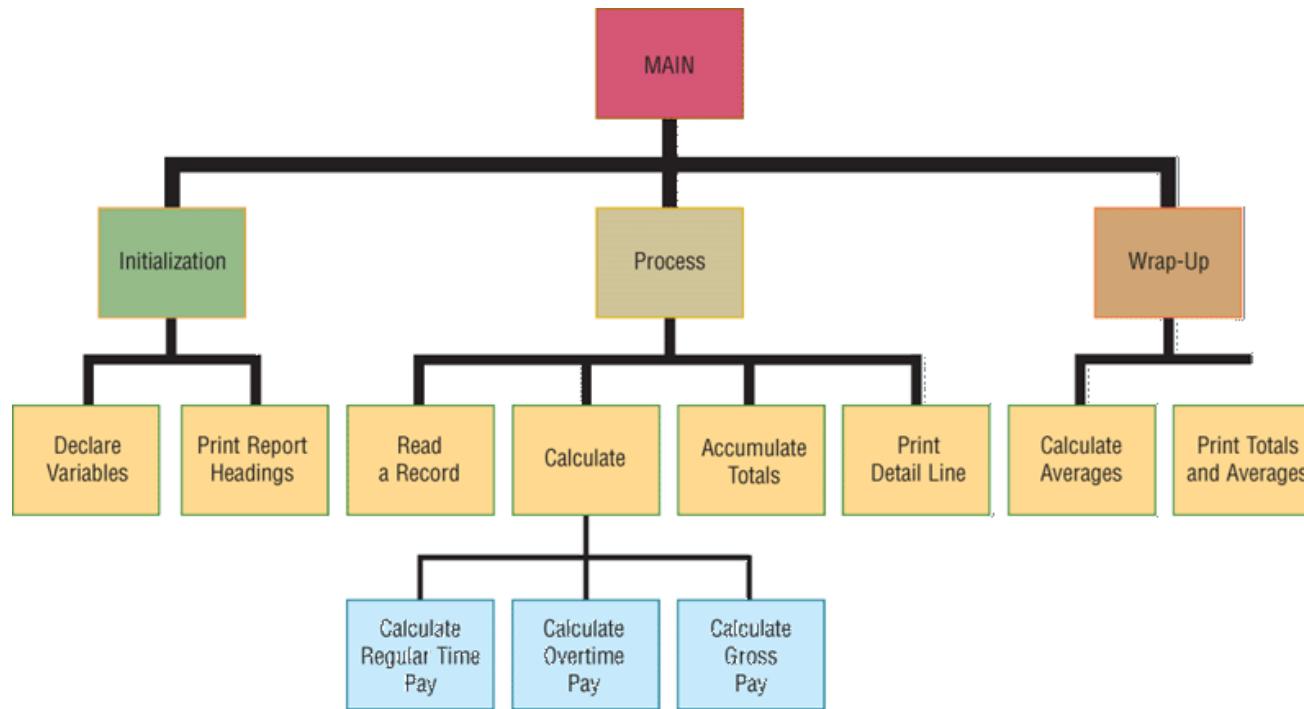
LTV bắt đầu với
thiết kế tổng thể
rồi đi đến
thiết kế chi tiết

Bước 2: Thiết kế giải pháp

Design solution

Thiết kế **Sơ đồ phân cấp chức năng** (hierarchy chart)

- Còn gọi là sơ đồ cấu trúc
- Trực quan hóa các module chương trình

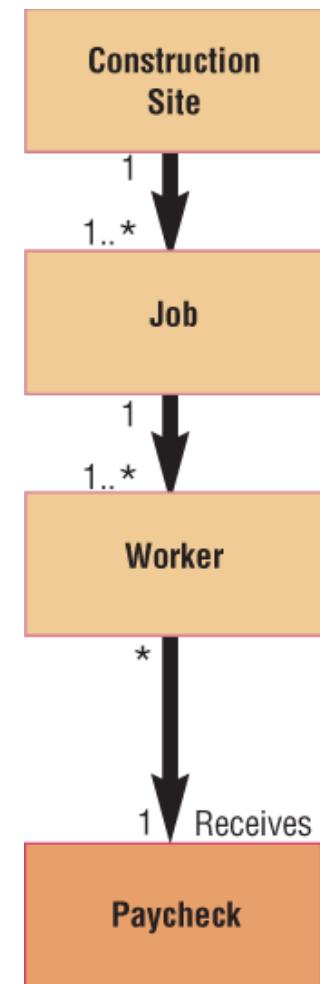


Bước 2: Thiết kế giải pháp

Design solution

Thiết kế hướng đối tượng

- LTV đóng gói dữ liệu và các thủ tục xử lý dữ liệu trong đối tượng (object)
- Các đối tượng được phân loại thành các lớp (classes)
- Thiết kế các biểu đồ lớp thể hiện trực quan các quan hệ phân cấp quan hệ của các lớp



Bước 2: Thiết kế giải pháp

Design solution

Thiết kế giải thuật

- Máy tính không thể tự nghĩ ra hay tự quyết định một sơ đồ hoạt động
- Máy tính chỉ có thể làm chính xác những gì được yêu cầu, theo cách được yêu cầu, chứ không phải làm những gì con người muốn máy tính làm
- Giải thuật là một tập các chỉ thị miêu tả cho máy tính nhiệm vụ cần làm và thứ tự thực hiện các nhiệm vụ đó.

Bước 3: Chứng thực thiết kế

Validate design

Kiểm tra độ chính xác của chương trình

LTV kiểm tra tính đúng đắn bằng cách tìm các lỗi logic (Logic Error)

Desk check
LTV dùng các dữ liệu thử nghiệm (Test data) để kiểm tra chương trình

Logic error
các sai sót khi thiết kế gây ra những kết quả không chính xác

Test data
các dữ liệu thử nghiệm giống như số liệu thực mà chương trình sẽ thực hiện

Structured Walkthrough
LTV mô tả logic của thuật toán trong khi đội lập trình duyệt theo logic chương trình

Bước 4: Cài đặt thiết kế

Implement design

- Viết mã nguồn chương trình (coding): dịch từ thiết kế thành chương trình
 - Cú pháp (Syntax): Quy tắc xác định cách viết các lệnh
 - Chú thích (Comments): tài liệu chương trình (tài liệu trong)
- Lập trình nhanh (Extreme programming - XP): viết mã nguồn và kiểm thử ngay sau khi các yêu cầu được xác định

Bước 5: Kiểm thử giải pháp

Test solution

Đảm bảo chương trình chạy thông
và cho kết quả chính xác

Debugging - Tìm và sửa các lỗi
syntax và logic errors

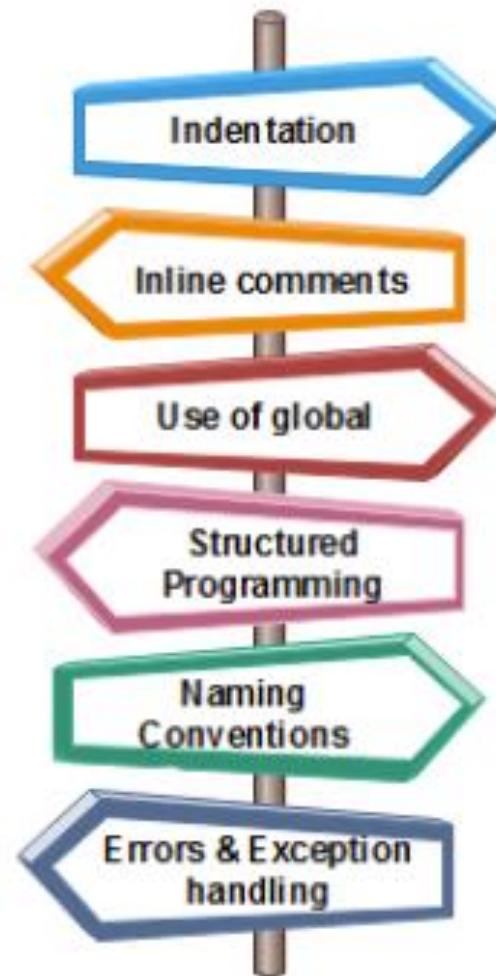
Kiểm tra phiên bản
beta, giao cho Users
dùng thử và thu thập
phản hồi

Bước 6: Viết tài liệu cho giải pháp Document solution

Rà soát lại program code -
loại bỏ các **dead code**, tức
các lệnh mà chương trình
không bao giờ gọi đến

Rà soát, hoàn thiện
documentation

3. Quy ước viết mã nguồn



Tại sao cần có phong cách lập trình tốt

- Lỗi logic đôi khi đến từ việc nhầm lẫn tên biến, nhầm lẫn mục đích của hàm do cách đặt tên...
- Mã nguồn tốt là mã nguồn dễ đọc
 - *Cấu trúc chương trình rõ ràng, dễ hiểu, khúc triết*
 - *Sử dụng thành ngữ phổ biến*
 - *Chọn tên phù hợp, gợi nhớ*
 - *Viết chú thích rõ ràng*

Một số quy tắc cơ bản

Nhất quán

- Tuân thủ quy tắc đặt tên trong toàn bộ chương trình
- Nhất quán trong việc dùng các biến cục bộ.

Khúc triết

- Mỗi function/method phải có một nhiệm vụ rõ ràng.
- Đủ ngắn để có thể nắm bắt được
- Số tham số của chương trình con là tối thiểu (dưới 6)

Một số quy tắc cơ bản

Rõ ràng

- Chú thích rõ ràng, vd. đầu mỗi chương trình con

Bao đóng

- Hàm chỉ nên tác động tới duy nhất 1 giá trị - giá trị trả về của hàm
- Không nên thay đổi giá trị của biến chạy trong thân của vòng lặp, ví dụ

```
for(i=1;i<=10;i++) i++;
```

Khoảng trắng – Cách lề Spacing - Indentation

- Sử dụng khoảng trắng dễ đọc và nhất quán
- Cách lề hợp lý để tránh nhầm lẫn cấu trúc các khối lệnh
- Cách đoạn (paragraph) để tạo các khối lệnh thực hiện các mục đích khác nhau giúp dễ đọc dễ theo dõi

Đặt tên Naming

- Dùng tên gợi nhớ, có tính miêu tả cho các biến và hàm
 - VD : hovaten, **CONTROL**, **CAPACITY**
- Dùng tên nhất quán cho các biến cục bộ
 - VD : **i** (not **arrayIndex**) cho biến chạy vòng lặp
- Dùng chữ hoa, chữ thường nhất quán
 - VD : Buffer_Insert (Tên hàm)
 CAPACITY (hằng số)
 buf (biến cục bộ)
- Dùng phong cách nhất quán khi ghép từ
 - VD : **frontsize**, **frontSize**, **front_size**

Đặt tên Naming

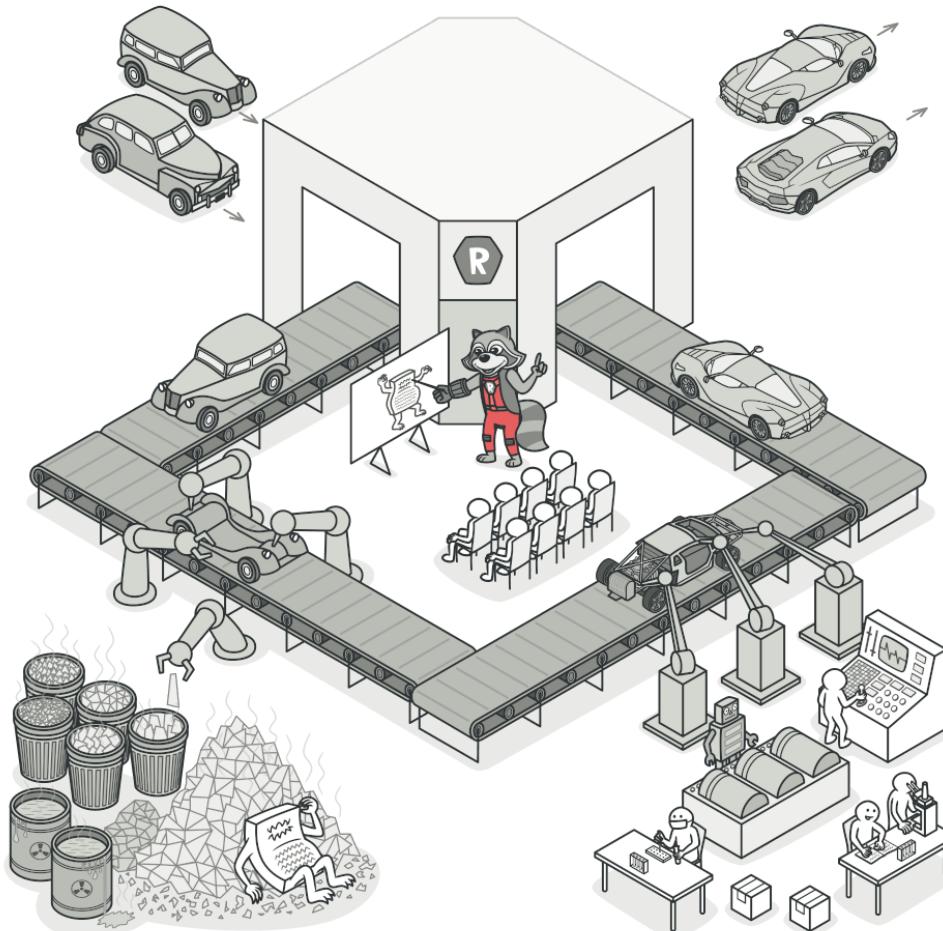
- Tên của các biến, các class thường là danh từ
- Tên class viết hoa chữ cái đầu
- Tên các phương thức, hàm thường là các động từ, chữ cái đầu viết thường

Chú thích Comments

- Chú thích cho từng hàm, mô tả mục đích của hàm, inputs, outputs
- Chú thích cho từng khối lệnh, không nên chú thích cho từng dòng lệnh
- Một số framework hỗ trợ generate tự động tài liệu mã nguồn (e.g., Javadoc)

4. Tái cấu trúc mã nguồn

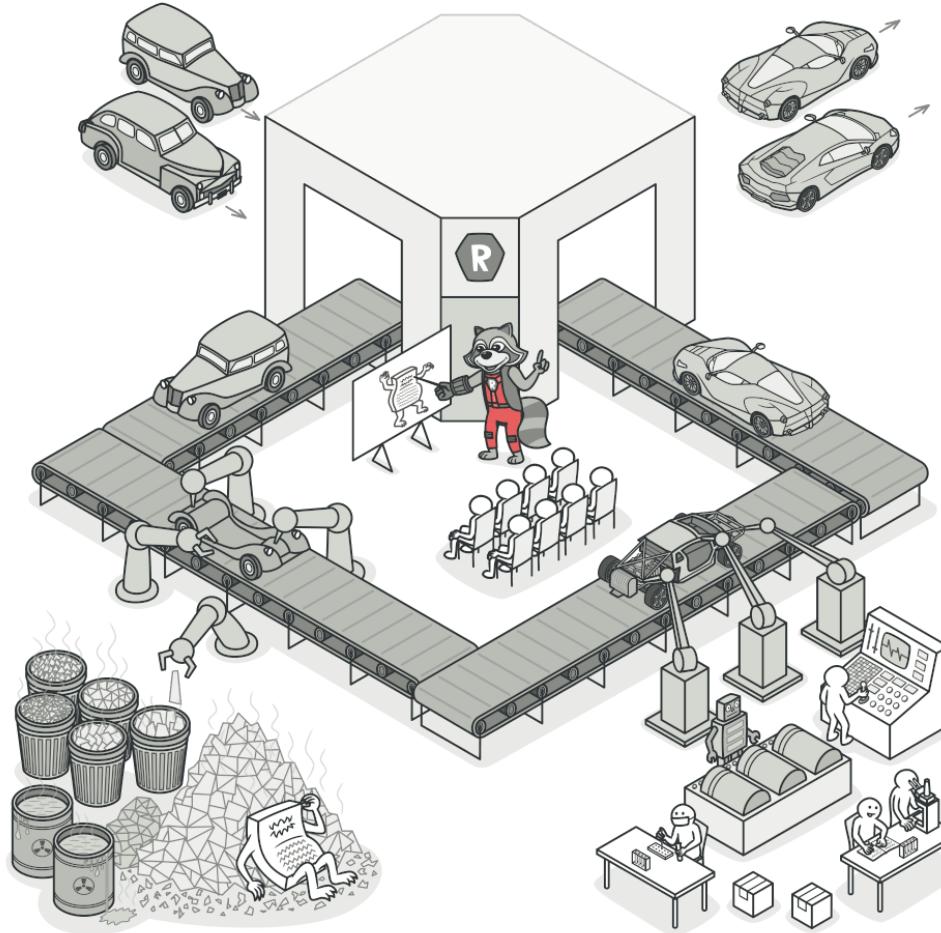
Code Restructuring / Refactoring



Refactoring Code: là một quá trình cải thiện mã nguồn mà không cần viết thêm các chức năng mới, làm cho mã nguồn sạch hơn (clean code) và chuyển đổi thiết kế phức tạp khó hiểu thành đơn giản

4. Tái cấu trúc mã nguồn

Code Restructuring / Refactoring



Clean Code: Mã sạch là mã dễ đọc, dễ hiểu, dễ bảo trì. Làm sạch mã giúp quá trình phát triển phần mềm dễ dự đoán được và làm tăng chất lượng sản phẩm phần mềm

4. Tái cấu trúc mã nguồn

Code Restructuring / Refactoring

Dirty Code: Mã nguồn “bẩn” thường là kết quả của việc thiếu kinh nghiệm lập trình, quản lý yếu kém, thời gian thực hiện quá ngắn hoặc các quyết định có tính chất tạm thời trong quá trình phát triển phần mềm



4. Tái cấu trúc mã nguồn

Code Restructuring / Refactoring



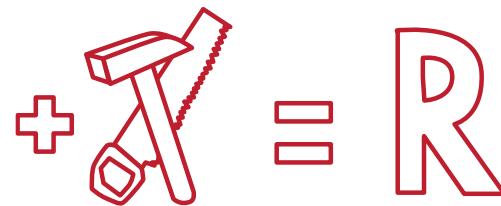
Mục tiêu của tái cấu trúc

Refactoring's goal

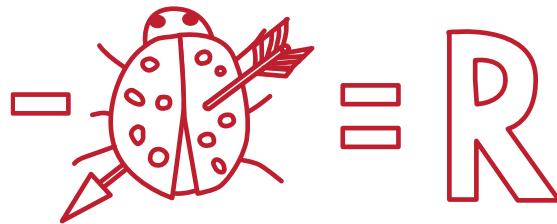
- Mục tiêu của tái cấu trúc mã nguồn là để tạo ra mã nguồn sạch (clean code), loại bỏ các lỗi tiềm ẩn, và làm cho thiết kế đơn giản hơn
- Mã nguồn trước khi đem tái cấu trúc là mã nguồn đã chạy, đã được test và đảm bảo các chức năng của chương trình
- Việc tái cấu trúc mã nguồn không làm thay đổi các chức năng của chương trình mà chỉ giúp cho mã nguồn sạch hơn và dễ đọc, dễ hiểu, dễ bảo trì

Khi nào cần thực hiện? When to refactor?

- When adding a feature
 - Khi cần thêm 1 tính năng mới vào một code sẵn có, có thể do người khác phát triển
 - Chúng ta cần hiểu code đó → Hãy thử refactoring lại code để giúp chúng ta dễ hiểu nó hơn
 - “It’s much easier to make changes in clean code”



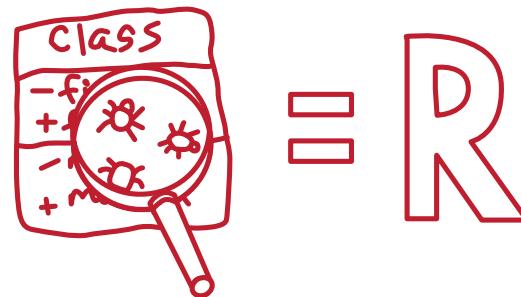
Khi nào cần thực hiện? When to refactor?



- When fixing a bug
 - Bug trong code thường tập trung ở những chỗ rối rắm khó hiểu của mã nguồn
 - Đôi khi chỉ cần làm sạch mã là các bug sẽ “**tự xuất hiện**”

Khi nào cần thực hiện? When to refactor?

- During a code review
 - Rà soát mã nguồn là cơ hội cuối để làm sạch mã trước khi nó được public cho toàn bộ đội phát triển
 - Việc rà soát thường được thực hiện bởi 2 người trong đó có tác giả của mã nguồn



Refactoring checklist

- ✓ Kiểm tra xem code đã dễ đọc, dễ hiểu hơn chưa, các method quá dài đã được tách nhỏ hơn để dễ hiểu cấu trúc code?
- ✓ Trong quá trình refactoring, kiểm tra để đảm bảo không có một chức năng mới nào được thêm vào
- ✓ Tất cả các tests đều passed sau khi refactoring
- **Các kỹ thuật refactoring code [đọc thêm]**
 - <https://refactoring.guru/refactoring/techniques>

5. Rà soát mã nguồn Code Review / Inspection

- Là quá trình mà mã nguồn (và cả thiết kế) được xem xét lại bởi một người (hoặc 1 nhóm người) khác thay vì chính tác giả
- Lợi ích - Benefits:
 - Góc nhìn khác với tác giả
 - Lỗi đôi khi dễ dàng được phát hiện bởi người khác hơn là tác giả
 - Chia sẻ tri thức
 - Xem xét các thiết kế và ý tưởng
 - Phát hiện lỗi sớm
 - Giảm thiểu công sức phải làm lại

Inspection vs Test

- Một số vấn đề chỉ được phát hiện trong quá trình review code thay vì kiểm thử
 - Xem xét một số tính chất của mã nguồn
 - Tính bảo trì được - Maintainability
 - Tính tái sử dụng được - Reusability
 - Tính dễ dàng mở rộng được – Extensibility
 - Xem xét độ tin cậy và tính tối ưu của thiết kế, tài liệu

Một số kĩ thuật rà soát

Different kinds of inspection

- Inspection / Formal technical review – Rà soát theo quy trình chính thống
 - Người tham gia: QA leaders, QA testers, developpers, project manager, team leaders etc.
 - Cần được chuẩn bị trước
 - Review checklist
 - Formal meeting:
 - Được chủ trì bởi một moderator không phải tác giả
 - Quá trình thực hiện phải được ghi chép lại đầy đủ
 - Theo danh sách checklist đã được chuẩn bị sẵn
 - Formal follow-up
 - Kết quả review sẽ được gửi cho tất cả các bên liên quan

Một số kĩ thuật rà soát

Different kinds of inspection

- Walkthroughs:

- Không cần chuẩn bị trước
- Tác giả sẽ là người dẫn dắt cuộc họp
- Chi phí tốn ít hơn so với Formal Technical Review
- Có tính chất đào tạo hơn

Q&A



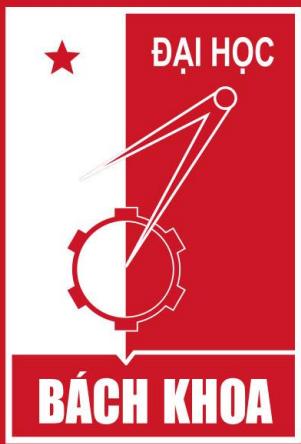


25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**





25 YEARS ANNIVERSARY

SOKT

The graphic consists of the number "25" in large, bold, white font. A curved banner arches over the top of the "2" containing the text "YEARS ANNIVERSARY". Below the "25" is the acronym "SOKT" in a large, bold, white, sans-serif font.

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 9

Đảm bảo chất lượng phần mềm
Software Quality Assurance

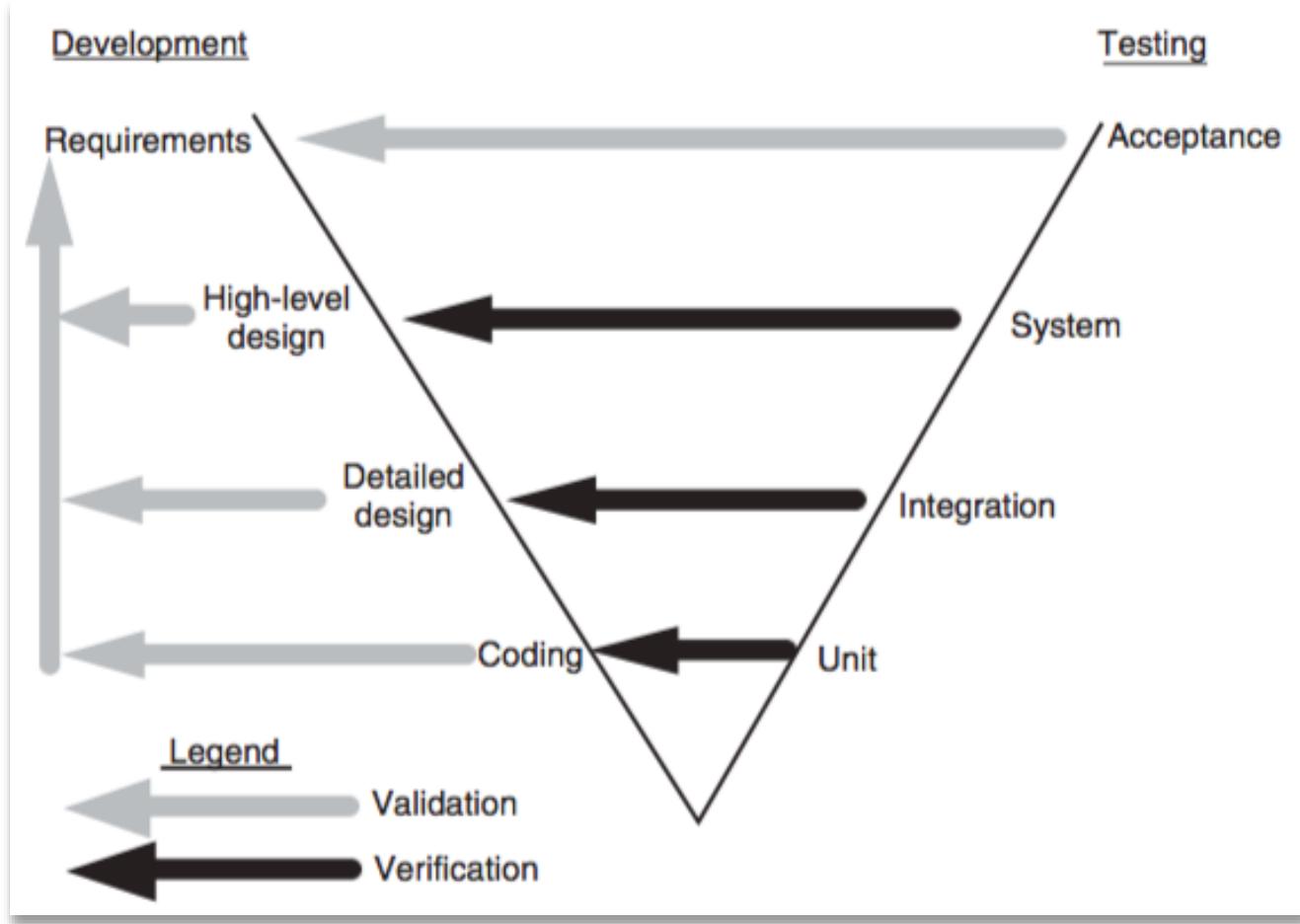
Nội dung

1. Mô hình V&V
2. Các thuật ngữ
3. Phương pháp kiểm thử hộp trắng
4. Phương pháp kiểm thử hộp đen
5. Quản lý chất lượng phần mềm
6. Bảo trì phần mềm

1. Mô hình V & V

- Mô hình V&V: Verification&Validation
- Verification : xác minh
 - Verify if the system is built right
- Validation: kiểm chứng
 - Validate if we built a right system
- Verification là hoạt động đánh giá hệ thống phần mềm bằng cách xác định xem hệ thống có thoả mãn các yêu cầu nêu ra ban đầu hay không (from perspective of development team)
- Validation là hoạt động kiểm chứng xem sản phẩm phần mềm có thoả mãn các yêu cầu của người dùng hay không (from perspective of users)

1. Mô hình V & V



Mô hình V & V

- Kiểm thử được thực hiện ở các mức khác nhau tương ứng với các phase trong quá trình phát triển phần mềm
- Kiểm thử đơn vị
- Kiểm thử tích hợp
- Kiểm thử hệ thống
- Kiểm thử xác nhận

2. Một số thuật ngữ cơ bản

- Failure: là một sự kiện xảy ra bất cứ khi nào hệ thống có một hành vi không tương ứng với đặc tả
- Fault: hay Bug là lỗi hay những bất thường trong phần mềm
- Error: là một trạng thái của hệ thống gây ra bởi một Fault
- Software Testing: là một quy trình phân tích và đánh giá một thành phần phần mềm (software item) để phát hiện sự khác biệt giữa những gì tồn tại và những yêu cầu đặt ra

2. Một số thuật ngữ cơ bản

- Test case: ca kiểm thử là một cặp <input, expected outcome>
 - Khi thực hiện một ca kiểm thử người ta sẽ đưa input tương ứng vào chương trình và kiểm tra kết quả trả về của hệ thống
 - Passed test: là trạng thái sau khi thực hiện một ca kiểm thử thu được kết quả trả về tương ứng với kết quả mong đợi
 - Failed test: ngược lại
- Test suite: là một tập hợp các test case được xây dựng để kiểm thử cho 1 chức năng hoặc 1 nhóm chức năng của phần mềm

Khó khăn

- Nâng cao chất lượng phần mềm nhưng không vượt quá chất lượng khi thiết kế: **chỉ phát hiện các lỗi tiềm tàng và sửa chúng**
- Phát hiện lỗi bị hạn chế do thủ công là chính
- Dễ bị ảnh hưởng tâm lý khi kiểm thử
- Khó đảm bảo tính đầy đủ của kiểm thử

Lưu ý khi kiểm thử

1. Chất lượng phần mềm do khâu thiết kế quyết định là chủ yếu, chứ không phải khâu kiểm thử
2. Tính dễ kiểm thử phụ thuộc vào cấu trúc chương trình
3. Người kiểm thử và người phát triển nên khác nhau
4. Dữ liệu thử cho kết quả bình thường thì không có ý nghĩa nhiều, cần có những dữ liệu kiểm thử mà phát hiện ra lỗi
5. Khi thiết kế trường hợp thử, không chỉ dữ liệu kiểm thử nhập vào, mà phải thiết kế trước cả dữ liệu kết quả sẽ có
6. Khi phát sinh thêm trường hợp thử thì nên thử lại những trường hợp thử trước đó để tránh ảnh hưởng lan truyền sóng

Các kỹ thuật kiểm thử

- Có 2 kỹ thuật kiểm thử cơ bản:
 - Kiểm thử hộp đen – black box testing
 - Kiểm thử hộp trắng – white box testing
- Cả hai kỹ thuật này đều yêu cầu phải thực thi chương trình để xem xét kết quả
- Giúp dễ dàng thiết kế các ca kiểm thử để có thể phát hiện nhiều nhất các lỗi có thể trong phần mềm
- Sử dụng kỹ thuật nào phụ thuộc vào các giai đoạn (các mức kiểm thử khác nhau) trong quá trình phát triển phần mềm

3. Kỹ thuật kiểm thử hộp trắng

- Kiểm thử hộp trắng là 1 chiến lược kiểm thử dựa trên cấu trúc của chức năng cần kiểm thử để thiết kế các ca kiểm thử
- Structural Testing
- Thường được thực hiện bởi lập trình viên và có thể được triển khai bởi các kiểm thử viên
- Thực hiện chủ yếu ở giai đoạn kiểm thử đơn vị và kiểm thử tích hợp

Các phương pháp thiết kế ca kiểm thử

- Kiểm thử luồng điều khiển
 - Bao phủ tất cả đường dẫn chương trình
 - Bao phủ lệnh
 - Bao phủ nhánh
- Kiểm thử luồng dữ liệu
 - Bao phủ tất cả các điểm định nghĩa của biến
 - Bao phủ tất cả các điểm sử dụng của biến
 - ...

Các phương pháp thiết kế ca kiểm thử

- **Kiểm thử luồng điều khiển**
 - Bao phủ tất cả đường dẫn chương trình
 - Bao phủ lệnh
 - Bao phủ nhánh
- **Kiểm thử luồng dữ liệu**
 - Bao phủ tất cả các điểm định nghĩa của biến
 - Bao phủ tất cả các điểm sử dụng của biến
 - ...

Đồ thị luồng điều khiển Control Flow Graph

- Biểu diễn đồ thị cấu trúc của một đơn vị chương trình
- 1 đường dẫn (path) của chương trình
 - Là một chuỗi các câu lệnh (statements) từ điểm bắt đầu cho đến điểm kết thúc của chương trình
 - Với mỗi một input data, chương trình có thể thực thi theo các đường dẫn khác nhau (execution path)
- Mục đích của việc thiết kế các ca kiểm thử trong kiểm thử cấu trúc:
 - Tìm ra những đường dẫn (path) tại đó lỗi hay xảy ra hoặc tiềm ẩn nhiều lỗi nhất có thể mà không cần phải bao phủ tất cả các path của chương trình

Đồ thị luồng điều khiển Control Flow Graph

- Có 3 loại statements:
 - tuần tự
 - rẽ nhánh
 - lặp
- Các tiêu chí lựa chọn đường dẫn chương trình
 - Bao phủ toàn bộ path → vét cạn
 - Chương trình có 10 lệnh rẽ nhánh true/false → Số đường dẫn tối đa là $2^{10} = 1024$ đường dẫn → 1024 test case!!!
 - Bao phủ các lệnh
 - Đảm bảo mọi câu lệnh trong chương trình đều được test
 - Bao phủ các nhánh
 - Đảm bảo mọi nhánh rẽ true/false trong chương trình đều được test

Example

- Chương trình có source code là hàm AccClient với input là tuổi và giới tính như sau:

Life Insurance Example

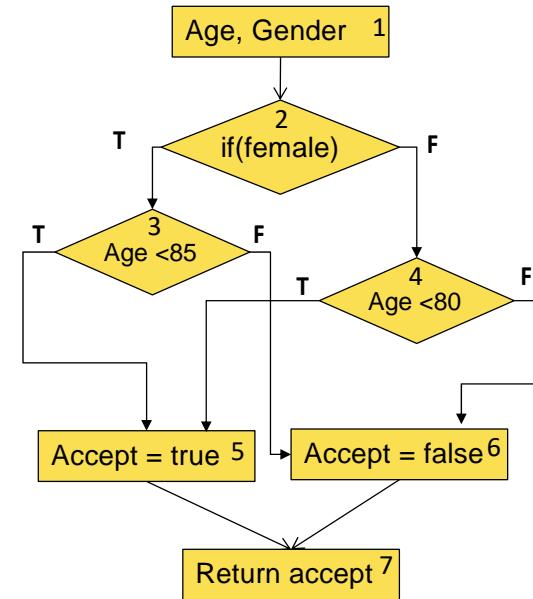
```
bool AccClient (agetype
    age; gndrtype gender)
bool accept
    if(gender=female)
        accept := age < 85;
    else
        accept := age < 80;
return accept
```

Example

- Vẽ sơ đồ luồng điều khiển

Life Insurance Example

```
bool AccClient (ageType  
                age; genderType gender)  
bool accept  
if (gender==female)  
    accept := age < 85;  
else  
    accept := age < 80;  
return accept
```



All path coverage

- Lựa chọn tất cả các path có thể trong chương trình
- Mỗi path sẽ tương ứng với 1 test case

All paths

Female	Age < 85	Age < 80
Yes	Yes	Yes
Yes	Yes	No
Yes	No	Yes
No	Yes	Yes
No	Yes	No
No	No	Yes
No	No	No

<Yes, Yes, *> 1-2(T)-3(T)-5-7
<Yes, No, No> 1-2(T)-3(F)-6-7
<No, Yes, Yes> 1-2(F)-4(T)-5-7
<No, *, No> 1-2(F)-4(F)-6-7

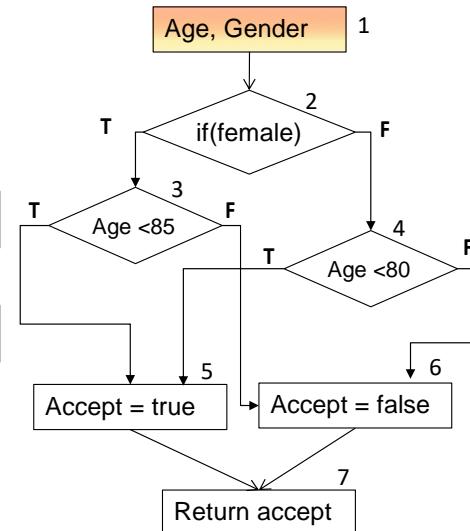
```
graph TD; A[Age, Gender 1] --> B{if(female)}; B -- T --> C{Age < 85}; B -- F --> D{Age < 80}; C -- T --> E[Accept = true 5]; C -- F --> D; D -- T --> E; D -- F --> F[Accept = false 6]; E --> G[Return accept 7]; F --> G;
```

Statement Coverage

- Đảm bảo mỗi một câu lệnh đều được thực thi ít nhất 1 lần → Đảm bảo mọi đinh của đồ thị đều được xuất hiện ít nhất 1 lần trong các đường dẫn

```
bool AccClient (ageType  
    age; genderType gender)  
bool accept  
if (gender==female)  
    accept := age < 85;  
else  
    accept := age < 80;  
return accept
```

AccClient(83, female)->accept
AccClient(83, male) ->reject



Branch coverage

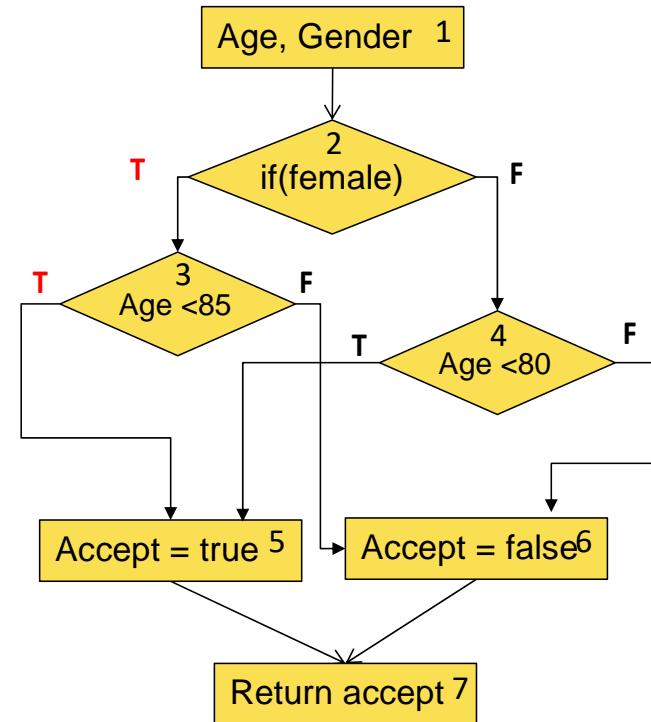
- Mục đích: đảm bảo mỗi một điểm quyết định trên đồ thị (điểm tại đó có nhánh rẽ) đều được kiểm thử ít nhất 1 lần với mỗi nhánh true/false

Example

Branch Coverage /1

```
bool AccClient (agetype
    age; gndrtype gender)
bool accept
if(gender=female)
    accept := age < 85;
else
    accept := age < 80;
return accept
```

AccClient(83,
female)->accept

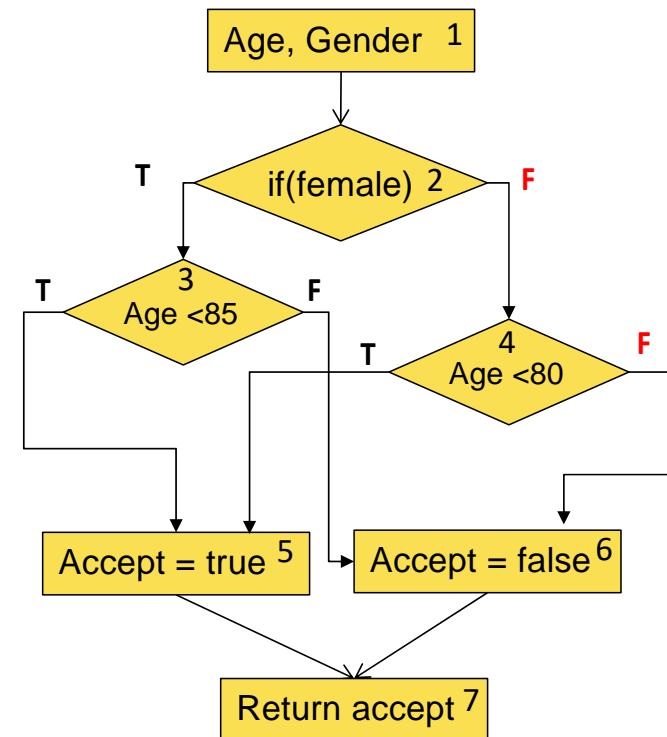


Example

Branch Coverage /2

```
bool AccClient (ageType  
    age; genderType gender)  
bool accept  
if(gender==female)  
    accept := age < 85;  
else  
    accept := age < 80;  
return accept
```

AccClient(83, male)
->reject

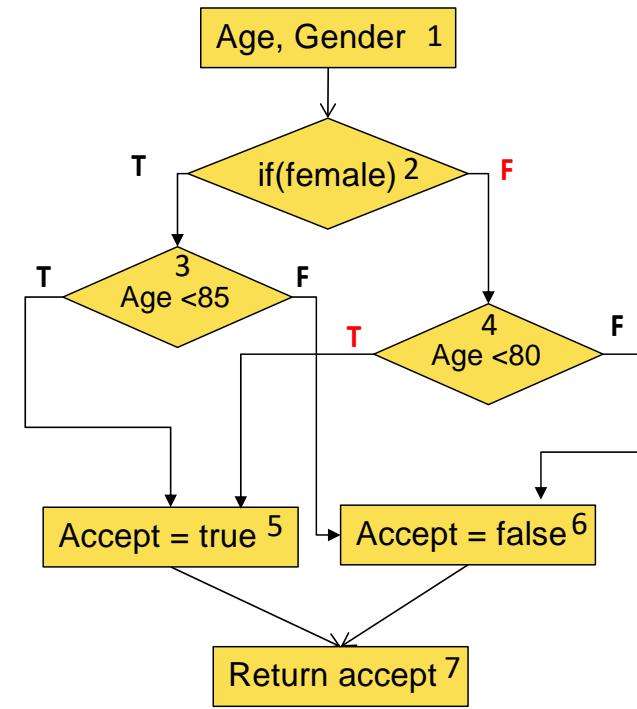


Example

Branch Coverage /3

```
bool AccClient (ageType  
    age; genderType gender)  
bool accept  
if (gender==female)  
    accept := age < 85;  
else  
    accept := age < 80;  
return accept
```

AccClient(78, male)-
>accept

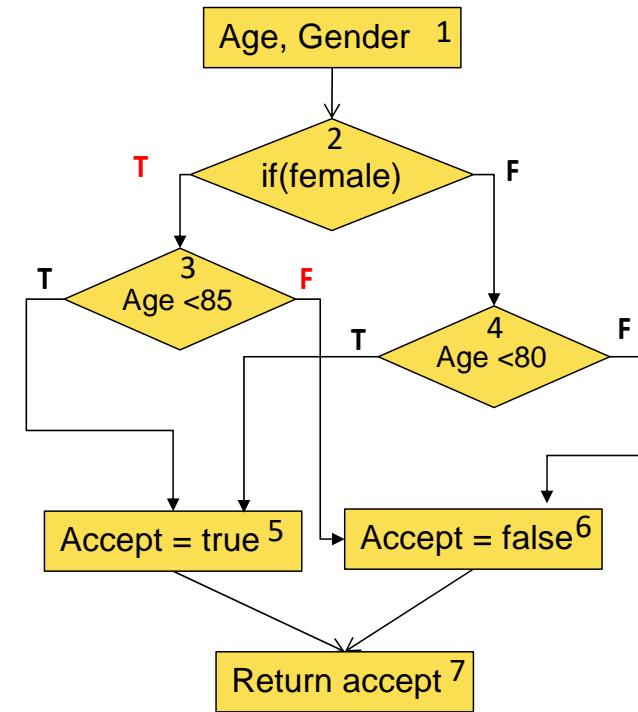


Example

Branch Coverage /4

AccClient(88,
female) ->reject

```
bool AccClient (ageType  
    age; genderType gender)  
bool accept  
if (gender==female)  
    accept := age < 85;  
else  
    accept := age < 80;  
return accept
```



So sánh các tiêu chí

- Tiêu chí mạnh nhất: bao phủ mọi đường dẫn
- Tiêu chí yếu nhất: bao phủ mọi câu lệnh
 - Các câu lệnh rẽ nhánh có thể chỉ cần kiểm thử 1 nhánh true/false
- Tiêu chí bao phủ điểm quyết định thường được sử dụng.

4. Kỹ thuật kiểm thử hộp đen

- Kiểm thử hộp đen là 1 chiến lược kiểm thử
 - Thực hiện dựa trên yêu cầu và đặc tả chức năng
 - Xác minh đầu ra của chức năng mà không quan tâm tới cấu trúc bên trong
- Còn được gọi với tên khác là kiểm thử chức năng (functional testing)
- Thường được thực hiện bởi các kiểm thử viên, kỹ sư đảm bảo chất lượng, và chủ yếu được sử dụng ở giai đoạn kiểm thử hệ thống và kiểm thử chấp nhận

Các phương pháp thiết kế ca kiểm thử

- Inputs của kiểm thử hộp đen là những đặc tả chi tiết của chức năng
- Outputs là các hành xử của hệ thống tương ứng với từng loại dữ liệu đầu vào và kịch bản kiểm thử
- Các phương pháp thiết kế ca kiểm thử:
 - Phân chia lớp tương đương
 - Phân tích giá trị biên
 - Sử dụng bảng quyết định
 - Kiểm thử trạng thái

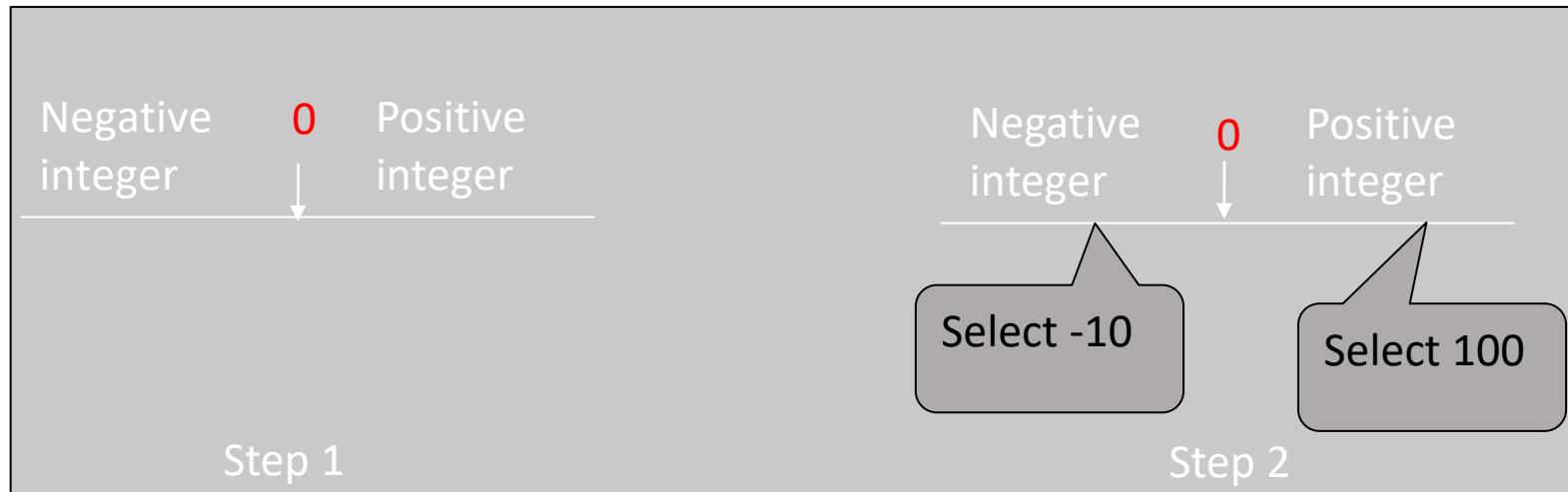
Các phương pháp thiết kế ca kiểm thử

- Inputs của kiểm thử hộp đen là những đặc tả chi tiết của chức năng
- Outputs là các hành xử của hệ thống tương ứng với từng loại dữ liệu đầu vào và kịch bản kiểm thử
- Các phương pháp thiết kế ca kiểm thử:
 - **Phân chia lớp tương đương**
 - Phân tích giá trị biên
 - Sử dụng bảng quyết định
 - Kiểm thử trạng thái

Kĩ thuật phân chia lớp tương đương

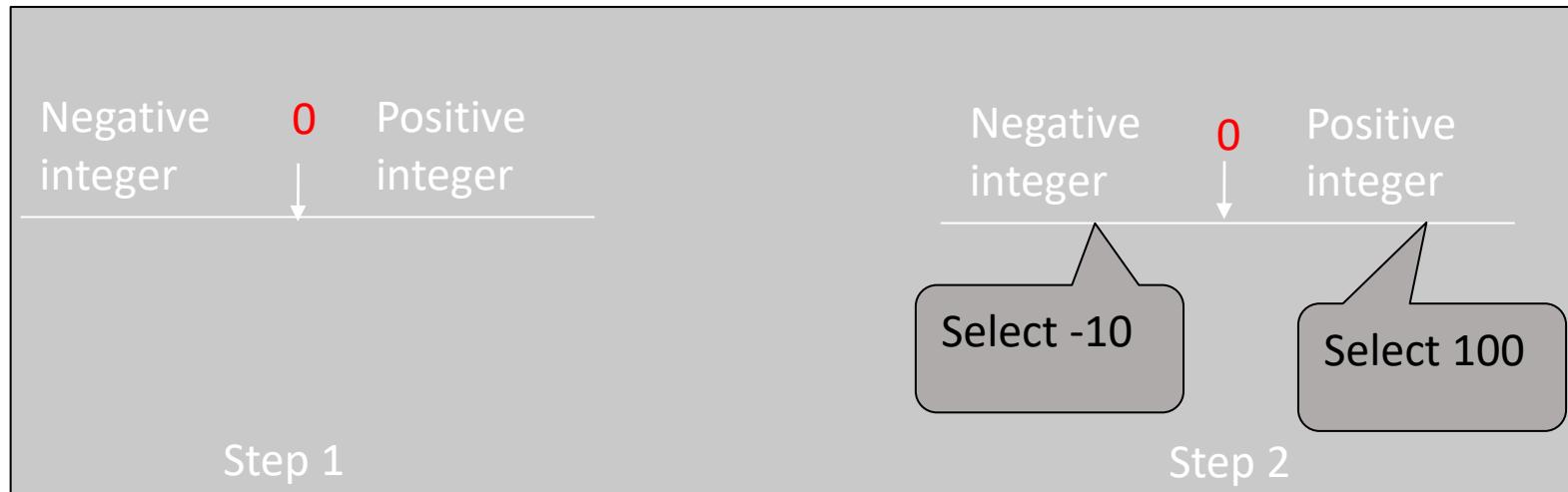
- Dựa vào miền dữ liệu trong đặc tả của inputs và outputs để xác định các phân lớp tương đương
- 1 phân lớp tương đương được định nghĩa là 1 miền dữ liệu (input hoặc output) trong đó với mọi điểm dữ liệu thuộc miền, chức năng được test thực hiện cùng một hành vi

Kĩ thuật phân chia lớp tương đương



- Một hàm thực hiện kiểm tra xem dữ liệu số nguyên truyền vào là âm hay dương.
- Input: số nguyên Output: true/false

Kỹ thuật phân chia lớp tương đương

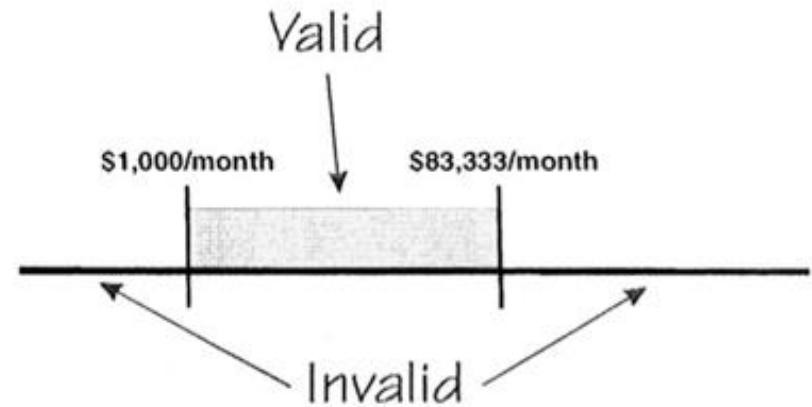


- Step 1: Xác định được 2 lớp tương đương ứng với $\text{input} \geq 0$ và $\text{input} < 0$
- Step 2: Ở mỗi phân lớp tương đương lấy 1 giá trị đại diện để xây dựng ca kiểm thử → 2 test cases

Ưu điểm



Giảm số lượng test case, thay vì phải quét toàn bộ miền dữ liệu của input



Đảm bảo mỗi một lớp đều được test

Nhược điểm

- Việc phân chia lớp tương đương như thế nào phụ thuộc vào kinh nghiệm của kiểm thử viên
 - Một số kiểm thử viên có thể bỏ qua các phân lớp dữ liệu không hợp lệ (invalid data)
- Không xem xét các giá trị tại biên của các lớp
 - Kết hợp với phân tích giá trị biên để xây dựng các ca kiểm thử
- Khi dữ liệu có mối quan hệ với nhau không còn độc lập nữa thì việc phân chia rõ ràng các lớp tương đương sẽ khó
 - Kết hợp với bảng quyết định để xây dựng các ca kiểm thử

5. Quy trình đảm bảo chất lượng phần mềm

- Chất lượng phần mềm được định nghĩa là
 - Mức độ một hệ thống hay một thành phần đạt được các yêu cầu đặt ra
 - Mức độ một hệ thống hay một thành phần đạt được những mong đợi của khách hàng hay người dùng hệ thống đó
- Chất lượng được nhìn nhận theo 2 quan điểm
 - Quan điểm của người phát triển
 - Quan điểm của người dùng

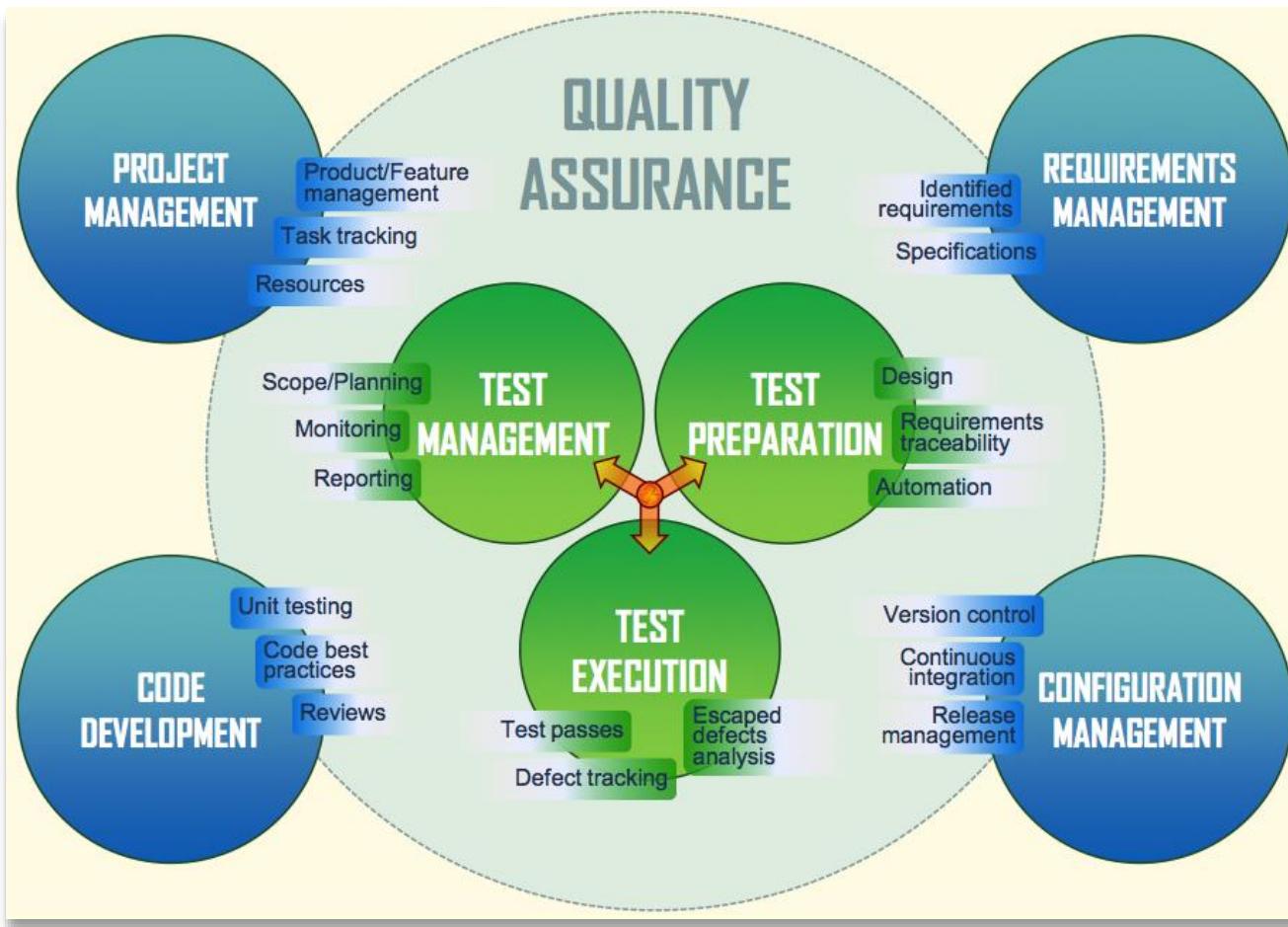
Các vấn đề trong việc đảm bảo chất lượng

- Quan điểm về chất lượng giữa người dùng và đội phát triển khác nhau
 - Đội phát triển quan tâm đến khía cạnh phát triển hệ thống: tái sử dụng, dễ kiểm thử, dễ bảo trì
 - Người dùng quan tâm đến hiệu quả, hiệu suất và độ tin cậy của phần mềm
 - Đôi khi để đảm bảo cho code sạch, dễ tái sử dụng, dễ bảo trì mà các tiêu chí chất lượng khác như hiệu quả, độ tin cậy có thể giảm đi
- Đặc tả phần mềm không bao giờ là đầy đủ và nhất quán ngay từ đầu → Rất khó để quy chuẩn được chất lượng phần mềm ngay từ đầu

Quy trình đảm bảo chất lượng

- Quy trình đảm bảo chất lượng là một quy trình mang tính hệ thống được thực hiện để kiểm chứng xem sản phẩm hoặc dịch vụ đang được phát triển có đạt được các yêu cầu đặc tả ban đầu hay không?

Quy trình đảm bảo chất lượng



Quy trình đảm bảo chất lượng

- Tích hợp với tất cả các quy trình khác trong toàn bộ vòng đời phát triển của phần mềm
 - Quản lý dự án
 - Quản lý yêu cầu
 - Quản lý cấu hình
 - Quản lý mã nguồn
- Quy trình kiểm thử là nền tảng của quy trình QA: 3 hoạt động chính
 - Quản lý kiểm thử
 - Chuẩn bị kiểm thử / Kế hoạch kiểm thử
 - Thực thi kiểm thử

6. Bảo trì phần mềm

- Bảo trì là công việc **tu sửa, thay đổi phần mềm đã được phát triển** (chương trình, dữ liệu, JCL, các loại tư liệu đặc tả, . . .) theo những lý do nào đó.
- Các hình thái bảo trì: bảo trì để
 - Tu chỉnh
 - Thích nghi
 - Cải tiến
 - Phòng ngừa

a. Bảo trì để tu sửa

- Là bảo trì khắc phục những **khiếm khuyết** có trong phần mềm.
- Một số nguyên nhân điển hình
 - Kỹ sư phần mềm và khách hiểu nhầm nhau.
 - Lỗi tiềm ẩn của phần mềm do sơ ý của lập trình hoặc khi kiểm thử chưa bao quát hết.
 - Vấn đề tính năng của phần mềm: không đáp ứng được yêu cầu về bộ nhớ, tệp, ... Thiết kế sai, biên tập sai ...
 - Thiếu chuẩn hóa trong phát triển phần mềm (trước đó).
- Kỹ nghệ ngược (Reverse Engineering): dò lại thiết kế để tu sửa.
- Những lưu ý
 - Mức trừu tượng
 - Tính đầy đủ
 - Tính tương tác
 - Tính định hướng

b. Bảo trì để thích hợp

- Là **tu chỉnh phần mềm theo thay đổi** của môi trường bên ngoài nhằm duy trì và quản lý phần mềm theo vòng đời của nó.
- Thay đổi phần mềm thích nghi với môi trường: công nghệ phần cứng, môi trường phần mềm.
- Những nguyên nhân chính:
 - Thay đổi về phần cứng (nguyên vi, máy chủ,...)
 - Thay đổi về phần mềm (môi trường): đổi OS
 - Thay đổi cấu trúc tệp hoặc mở rộng CSDL

c. Bảo trì để cải tiến

- Là việc tu chỉnh hệ phần mềm theo các yêu cầu ngày càng **hoàn thiện hơn, đầy đủ hơn, hợp lý hơn.**
- Những nguyên nhân chính:
 - Do muốn nâng cao hiệu suất nên thường hay cải tiến phương thức truy cập tệp.
 - Mở rộng thêm chức năng mới cho hệ thống.
 - Cải tiến quản lý kéo theo cải tiến tư liệu vận hành và trình tự công việc.
 - Thay đổi người dùng hoặc thay đổi thao tác.
- Còn gọi là tái kỹ nghệ (re-engineering)
- Mục đích: đưa ra một thiết kế cùng chức năng nhưng có chất lượng cao hơn.
- Các bước thực hiện:
 - Xây dựng lưu đồ phần mềm
 - Suy diễn ra biểu thức BNF cho từng dãy xử lý
 - Biên dịch bảng chân lí
 - Tái cấu trúc phần mềm

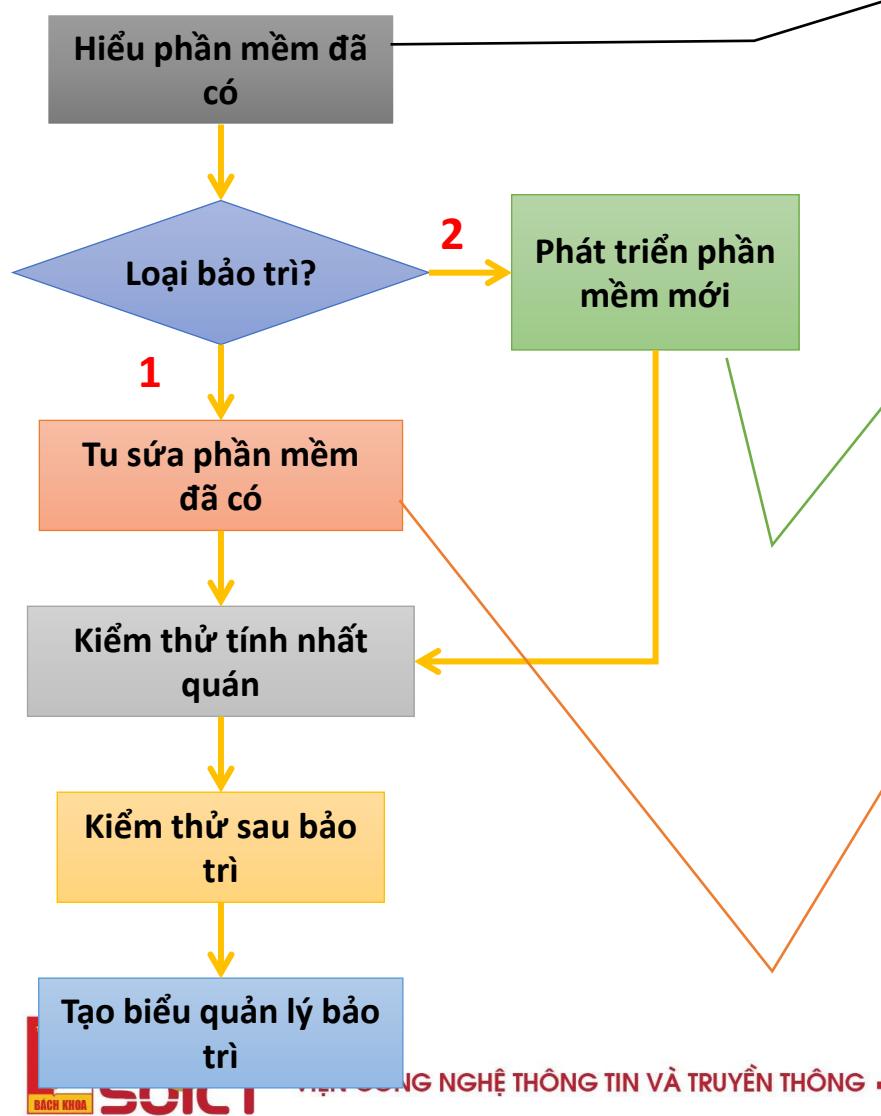
d. Bảo trì để phòng ngừa

- Là công việc tu chỉnh chương trình có **tính đến tương lai** của phần mềm đó sẽ mở rộng và thay đổi như thế nào.
- Thực ra trong khi thiết kế phần mềm đã phải tính đến tính mở rộng của nó, nên thực tế ít khi ta gặp bảo trì phòng ngừa nếu như phần mềm được thiết kế tốt.
- Mục đích: sửa đổi để thích hợp với yêu cầu thay đổi sẽ có của người dùng.
- Thực hiện những thay đổi trên thiết kế không tường minh.
- Hiểu hoạt động bên trong chương trình
 - Thiết kế / lập trình lại.
 - Sử dụng công cụ CASE

2. Quy trình nghiệp vụ

- Quy trình bảo trì: quá trình trong vòng đời của phần mềm, cũng **tuân theo các pha phân tích, thiết kế, phát triển và kiểm thử** từ khi phát sinh vấn đề cho đến khi giải quyết xong.
- Các nhiệm vụ bảo trì:
 - Phân tích/cô lập: phân tích tác động, phân tích những giá trị lợi ích, và cô lập các thành phần cần bảo trì
 - Thiết kế: thiết kế lại hệ thống (phải biết cách tu sửa, thay đổi).
 - Thực thi: thay thế mã nguồn và kiểm soát từng đơn vị thành phần hệ thống, có tính đến thời gian lập trình.
- Thao tác bảo trì: Gồm 2 loại
 - Tu chỉnh cải đã có (loại 1)
 - Thêm cái mới (loại 2)

Sơ đồ bảo trì



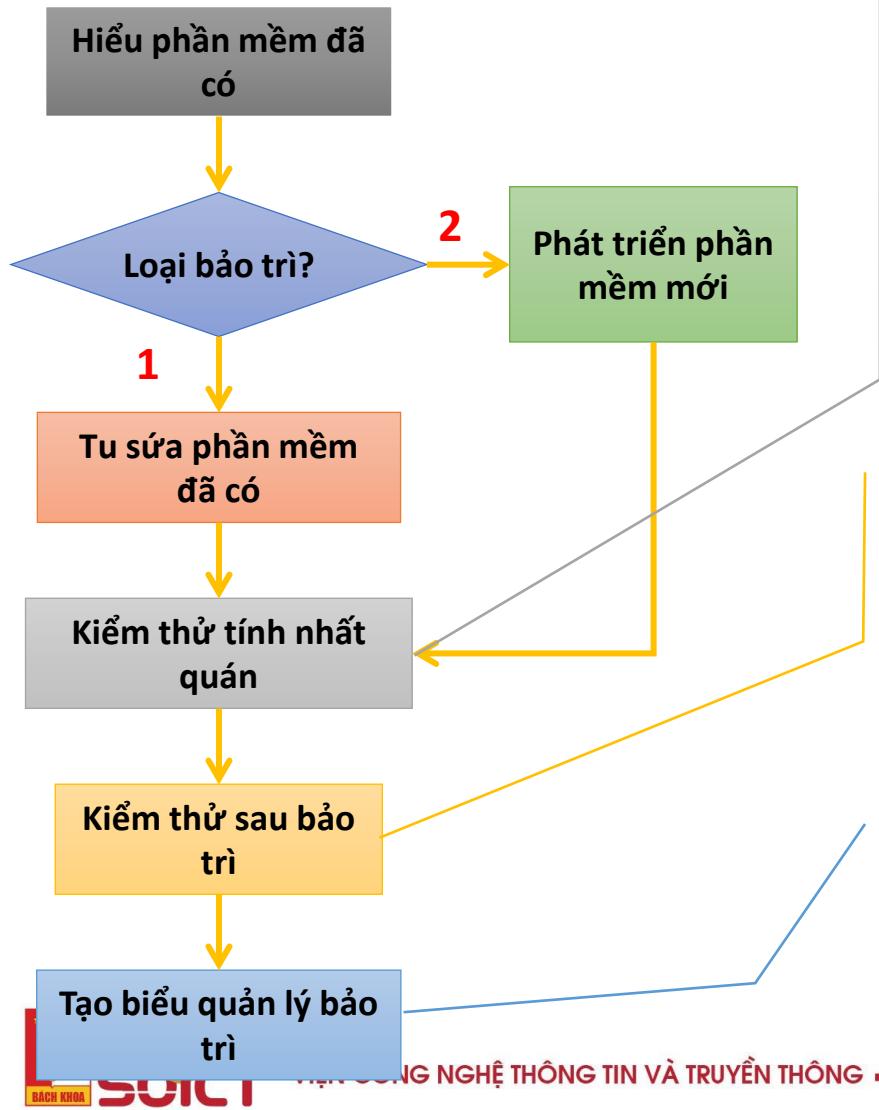
Thực thi “trên bàn”:

- Nắm vững các chức năng của hệ thống theo tài liệu
- Nắm vững đặc tả chi tiết, điều kiện kiểm thử, ... theo tài liệu
- Đọc chương trình nguồn, hiểu trình tự xử lý chi tiết của hệ thống

- Khi thêm chức năng mới phải phát triển chương trình cho phù hợp với yêu cầu
- Cần tiến hành từ thiết kế, lập trình, gỡ lỗi và kiểm thử unit
- Phản ảnh vào giao diện của phần mềm (thông báo, phiên bản, ...)

- Bảo trì chương trình nguồn, tạo các module mới và dịch lại.
- Thực hiện kiểm thử unit và tu chỉnh những mục liên quan có trong tư liệu đặc tả.
- Chú ý theo sát tác động của module được sửa đến các thành phần khác trong hệ thống.

Sơ đồ bảo trì



Bằng kiểm thử tích hợp

- Đưa đơn vị (unit) đã được kiểm thử vào hoạt động trong hệ thống
 - Điều chỉnh sự tương tích giữa các module
 - Dùng các dữ liệu trước đây khi kiểm thử để kiểm thử lại tính nhất quán
- ! Chú ý hiệu ứng làn sóng trong chỉnh sửa

Khi hoàn thành bảo trì:

- Kiểm tra nội dung mô tả có trong tư liệu đặc tả
- Cách ghi tư liệu có phù hợp với mô tả môi trường phần mềm mới hay không ?

Để quản lý tình trạng bảo trì, lập biểu:

- Ngày tháng, giờ
- Nguyên nhân
- Tóm tắt cách khắc phục
- Chi tiết khắc phục, hiệu ứng làn sóng
- Người làm bảo trì
- Số công

3. Các vấn đề còn tồn tại

- Phương pháp cải tiến thao tác bảo trì:
 - Sáng kiến trong quy trình phát triển phần mềm
 - Sáng kiến trong quy trình bảo trì phần mềm
 - Phát triển những kỹ thuật mới cho bảo trì

a. Sáng kiến trong quy trình phát triển phần mềm

- Chuẩn hóa mọi khâu trong phát triển phần mềm
- Người bảo trì chủ chốt tham gia vào giai đoạn phân tích và thiết kế
- Thiết kế để dễ bảo trì

b. Sáng kiến trong quy trình bảo trì phần mềm

- Sử dụng các công cụ hỗ trợ phát triển phần mềm
- Chuẩn hóa thao tác bảo trì và thiết bị môi trường bảo trì
- Lưu lại những thông tin sử bảo trì
- Dự án nên cử một người chủ chốt của mình làm công việc bảo trì sau khi dự án kết thúc giai đoạn phát triển.

c. Phát triển những kỹ thuật mới cho bảo trì

- Công cụ phần mềm hỗ trợ bảo trì
- Cơ sở dữ liệu cho bảo trì
- Quản lý tài liệu, quản lý dữ liệu, quản lý chương trình nguồn, quản lý dữ liệu thử, quản lý sử bảo trì
- Trạm bảo trì tính năng cao trong hệ thống mạng lưới bảo trì với máy chủ thông minh.

Kiểm thử và bảo trì

- Kiểm thử
 - Phát hiện lỗi, đánh giá phần mềm
 - Thực hiện trong và sau quá trình phát triển phần mềm
- Bảo trì
 - Thay đổi, cải tiến phần mềm đã được phát triển
 - Thực hiện sau khi phần mềm được đưa vào sử dụng

Q&A





25
YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**

