NHẬP MÔN
CÔNG NGHỆ PHẦN MỀM
(INTRODUCTION TO SOFTWARE ENGINEERING)

Nội dung

I. Phần mềm
- 1. Định nghĩa
- 2. Đặc tính của phần mềm
- 3. Thế nào là phần mềm tốt?
- 4. Phân loại phần mềm

II. Công nghệ phần mềm
- 1. Định nghĩa
- 2. Công nghệ học trong CNPM
- 3. Mục tiêu của công nghệ học phần mềm
- 4. SE - công nghệ phân lớp
- 5. SE - các pha
- 6. Những khó khăn trong sản xuất phần mềm

I. Phần mềm

1. Định nghĩa về phần mềm
   - Phần mềm (Software - SW) như một khái niệm đối nghĩa với phần cứng (Hardware - HW), tuy nhiên, đây là 2 khái niệm tương đối
   - Từ xưa, SW như thứ được cho không hoặc bán kèm theo máy (HW)
   - Dần dần, giá thành SW ngày càng cao và nay cao hơn HW

Các đặc tính của SW và HW

Hardware

- Vật "cứng"
- Kim loại
- Vật chất
- Hữu hình
- Sản xuất công nghiệp bởi máy móc là chính
- Định lượng là chính
- Hỏng hóc, hao mòn

Software

- Vật "mềm"
- Kỹ thuật sử dụng
- Trừu tượng
- Vô hình
- Sản xuất bởi con người là chính
- Định tính là chính
- Không hao mòn

SW đối nghĩa với HW

- Vai trò SW ngày càng thể hiện trội
- Máy tính là . . . chiếc hộp không có SW
- Ngày nay, SW quyết định chất lượng một hệ thống máy tính (HTMT), là chủ đề cốt lõi, trung tâm của HTMT
- Hệ thống máy tính gồm HW và SW

Định nghĩa 1

- Phần mềm là
  - 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 tư liệu mô tả thao tác và cách sử dụng chương trình
- IEEE: Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.

Định nghĩa 2

- Trong một hệ thống máy tính, nếu trừ bỏ đi các thiết bị và các loại phụ kiện thì phần còn lại chính là phần mềm (SW)
- Nghĩa hẹp: SW là dịch vụ chương trình để tăng khả năng xử lý của phần cứng của máy tính (như hệ điều hành - OS)
- Nghĩa rộng: SW là tất cả các kỹ thuật ứng dụng để thực hiện những dịch vụ chức năng cho mục đích nào đó bằng phần cứng

SW theo nghĩa rộng

- Không chỉ SW cơ bản và SW ứng dụng
- Phải gồm cả khả năng, kinh nghiệm thực tiễn và kỹ năng của kỹ sư (người chế ra phần mềm): Kỹ năng của kỹ sư phầm mềm (Know-how of Software Engineer)
- Là tất cả các kỹ thuật làm cho sử dụng phần cứng máy tính đạt hiệu quả cao

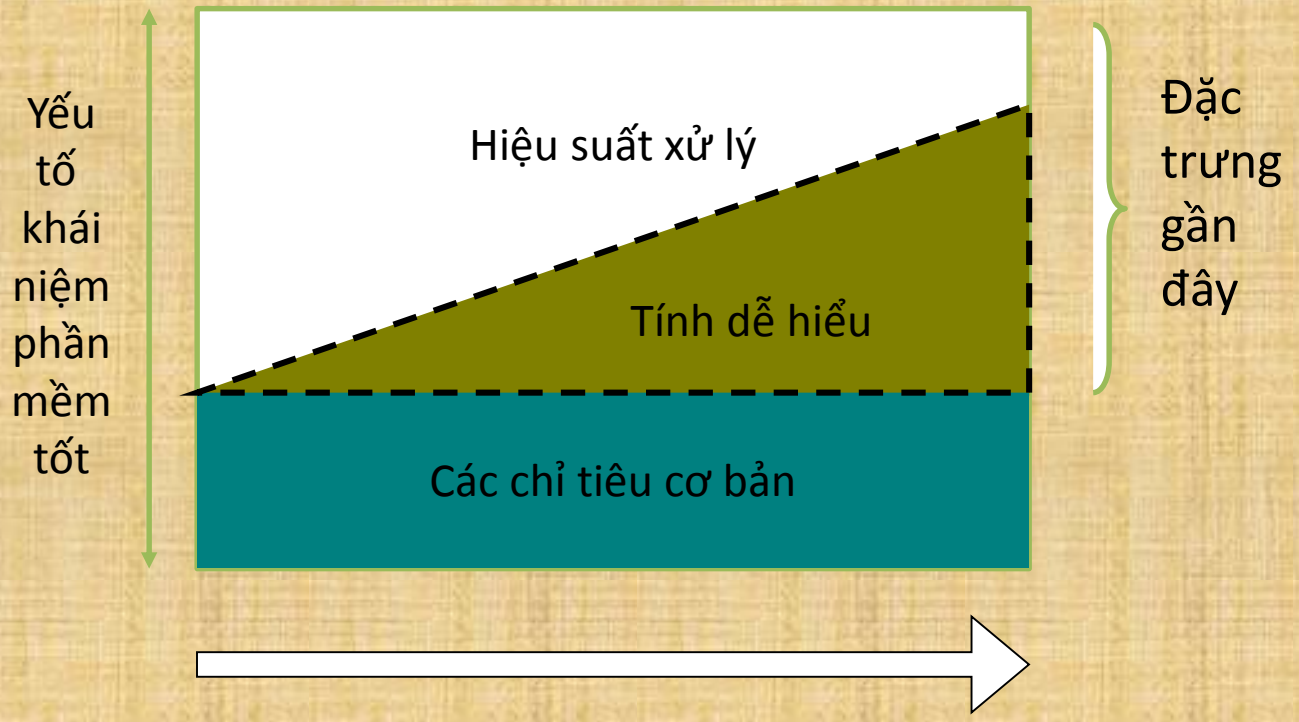
2. Đặc tính chung của phần mềm

- 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/bug) 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

2. Đặc tính chung của phần mềm

- Chức năng của phần mềm thường biến hóa, thay đổi theo thời gian (theo nơi sử dụng)
- Hiệu ứng làn sóng trong thay đổi phần mềm
- Phần mềm vốn chứa ý tưởng và sáng tạo của tác giả/nhóm làm ra nó
- Cần khả năng "tư duy nhị phân" trong xây dựng, phát triển phần mềm
- Có thể sao chép rất đơn giản

3. Thế nào là phần mềm tốt ?



3.1. Các chỉ tiêu cơ bản

- Phản ánh đúng yêu cầu người dùng (tính hiệu quả - effectiveness)
- Chứa ít lỗi tiềm tàng
- Giá thành không vượt quá giá ước lượng ban đầu
- Dễ vận hành, sử dụng
- Tính an toàn và độ tin cậy cao

3.2. Hiệu suất xử lý cao

- Hiệu suất thời gian tốt (efficiency):
  - Độ phức tạp tính toán thấp (Time complexity)
  - Thời gian quay vòng ngắn (Turn Around Time: TAT)
  - Thời gian hồi đáp nhanh (Response time)
- Sử dụng tài nguyên hữu hiệu: CPU, RAM, HDD, Internet resources, . . .

3.3. Dễ hiểu

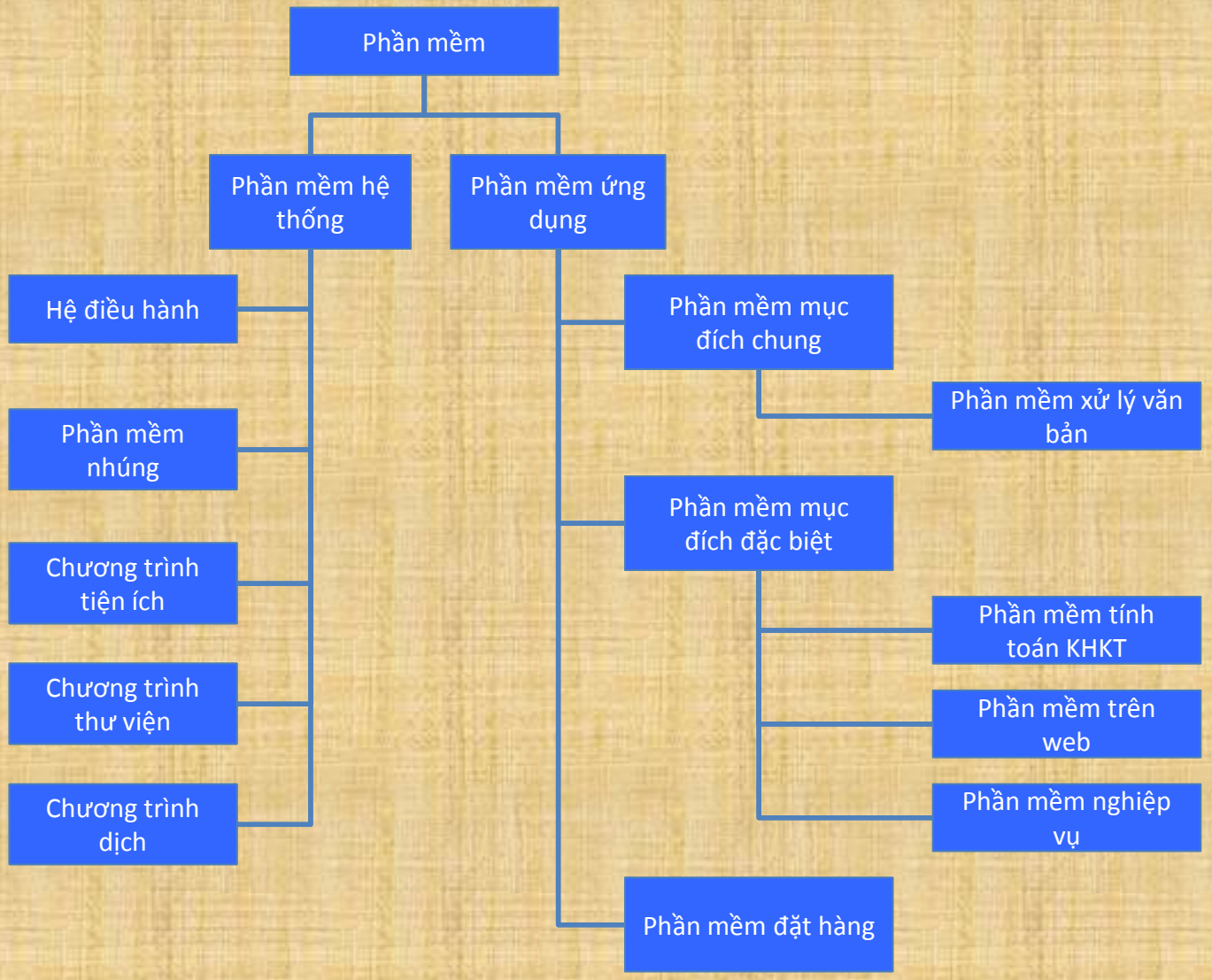
- Kiến trúc và cấu trúc thiết kế dễ hiểu
- Dễ kiểm tra, kiểm thử, kiểm chứng
- Dễ bảo trì
- Có tài liệu (mô tả yêu cầu, điều kiện kiểm thử, vận hành, bảo trì, FAQ, . . .) với chất lượng cao

Tính dễ hiểu: chỉ tiêu ngày càng quan trọng

4. 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 tính toán 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)

4. Phân loại phần mềm



Câu hỏi: Phân biệt các khái niệm sau

- Hệ thống, phần mềm, ứng dụng
- Lập trình, phát triển phần mềm
- Lập trình viên và kỹ sư phần mềm

II. Công nghệ phần mềm (Software Engineering)

1. 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

1. 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

1. 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

1. 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]

2. Công nghệ học trong CNPM

- Như các ngành công nghệ học khác, CNPM cũng lấy các phương pháp khoa học làm cơ sở
- Các kỹ thuật về thiết kế, chế tạo, kiểm thử và bảo trì phần mềm đã được hệ thống hóa thành phương pháp luận và hình thành nên CNPM
- Toàn bộ quy trình quản lý phát triển phần mềm gắn với khái niệm vòng đời phần mềm, được mô hình hóa với những kỹ thuật và phương pháp luận trở thành các chủ đề khác nhau trong CNPM

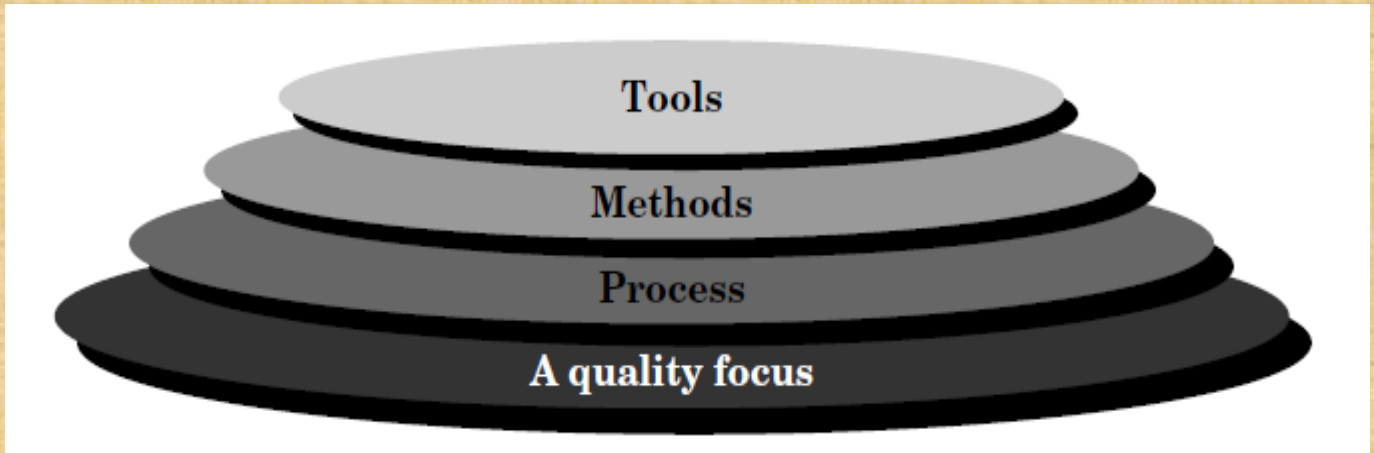
2. Công nghệ học trong CNPM

- Trong vòng đời phần mềm không chỉ có chế tạo mà bao gồm cả thiết kế, vận hành và bảo dưỡng (tính quan trọng của thiết kế và bảo dưỡng)
- Trong khái niệm phần mềm, không chỉ có chương trình mà cả tư liệu về phần mềm
- Cách tiếp cận công nghệ (khái niệm công nghiệp hóa) thể hiện ở chỗ nhằm nâng cao năng suất (tính năng suất) và độ tin cậy của phần mềm, đồng thời giảm chi phí giá thành

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

What is SE?



- SE là công nghệ phân lớp
  - Quy trình (Process)
  - Các phương pháp (Methods)
  - Các công cụ (Tools)

What is SE?

- 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

What is SE?

- 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

What is SE?

- 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 - phases

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

Definition phase

- 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 điện 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.

Development phase

- 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.

Support phase

- 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).

6. Những khó khăn trong sản xuất phần mềm

- Không có phương pháp mô tả rõ ràng định nghĩa yêu cầu của người dùng (khách hàng)

→ ***Sau khi bàn giao sản phẩm dễ phát sinh những trục trặc (troubles)***

- Với những phần mềm quy mô lớn, tư liệu đặc tả đã cố định thời gian dài

  → ***Khó đáp ứng nhu cầu thay đổi của người dùng một cách kịp thời trong thời gian đó***

- Phương pháp luận thiết kế không nhất quán

  → ***Thiết kế theo cách riêng (của công ty, nhóm), thì sẽ dẫn đến suy giảm chất lượng phần mềm (do phụ thuộc quá nhiều vào con người)***

- 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***

6. Những khó khăn trong sản xuất phần mềm

- 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***
- Không chứng minh được tính đúng đắn của phần mềm

  → ***giảm độ tin cậy của phần mềm***

6. Những khó khăn trong sản xuất phần mềm

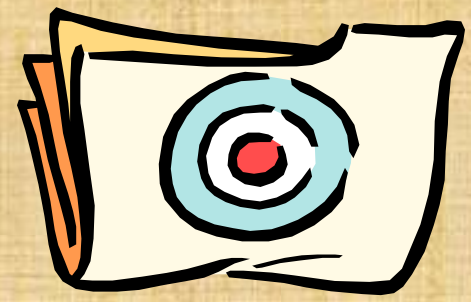
- Chuẩn về một phần mềm tốt không thể đo được một cách định lượng
  → ***Không thể đánh giá được một hệ thống đúng đắn hay không***
- Đầu tư nhân lực lớn vào bảo trì
  → ***giảm hiệu suất lao động của nhân viên***
- Công việc bảo trì kéo dài
  → ***giảm chất lượng của tư liệu và ảnh hưởng xấu đến những 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***

NHẬP MÔN
CÔNG NGHỆ PHẦN MỀM
(INTRODUCTION TO SOFTWARE ENGINEERING)

Chương 5: Quản lý cấu hình PM

- 1. Đặt vấn đề
- 2. Khái niệm quản lý cấu hình PM
- 3. Các hoạt động trong quản lý cấu hình PM
- 4. Vai trò của người quản lý cấu hình PM
- 5. Các khái niệm trong SCM
- 6. Lập kế hoạch quản lý cấu hình PM
- 7. Các công cụ quản lý cấu hình PM

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**"

Software Evolution

- 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

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
  - 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)

Configuration Management…

- 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

Configuration Management

- SCM có thể giúp xác định tác động của thay đổi cũng như kiểm soát sự phát triển song song
- 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 … triển - nhiều tài ngu… …ống thay đổi
  - CM ng… … các lỗi có thể tr… được phát sin… …ác thay đổi xung đột
- Thông th… …ng nhi… …iên bản của … …mềm … …ược phát hành và … …n đến … …ỗ trợ
  - CM cho… phép … …óm hỗ … …hiều phi…
  - CM cho phép … …y đổi t… …ng các phiê… …n tự được truyền bá
- CM cho phép … … triển th… …c thay đổi và khôi phục bất kỳ th… …ể đưa l… …hần mềm trở lại trạng thá… …an toà… …ần đầ…

2. Software Configuration Management

Forward Definition!

- Definition:
  - Một tập hợp các quy tắc quản lý trong quy trình kỹ thuật phần mềm để phát triển đường cơ sở (baseline).
  - Chuẩn IEEE (IEEE Std. No. 610.12-1990) định nghĩa một cơ sở 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 SCI - software configuration items thu được thông qua đánh giá kỹ thuật chính thức.

Software Configuration Management

- Description:
  - Quản lý cấu hình phần mềm 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.

- Standards (approved by ANSI)
  - IEEE 828: Software Configuration Management Plans
  - IEEE 1042: Guide to Software Configuration Management

2. SCM Các hoạt động

- Software Configuration Management (SCM) Activities:
  - Configuration item identification
  - Promotion management
  - Release management
  - Branch management
  - Variant management
  - Change management

- No fixed rules:
  - SCM functions are usually performed in different ways (formally, informally) depending on the project type and life-cycle phase (research, development, maintenance).

SCM Activities (continued)

- 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

4. SCM Roles

- 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

5. Các khái niệm trong SCM

- What are
  - Configuration Items
  - Baselines
  - SCM Directories
  - Versions, Revisions and Releases

⇨ *Các thuật ngữ được định nghĩa ở đây không nghiêm chặt và thay đổi đối với các hệ thống quản lý cấu hình khác nhau.*

Configuration Item

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

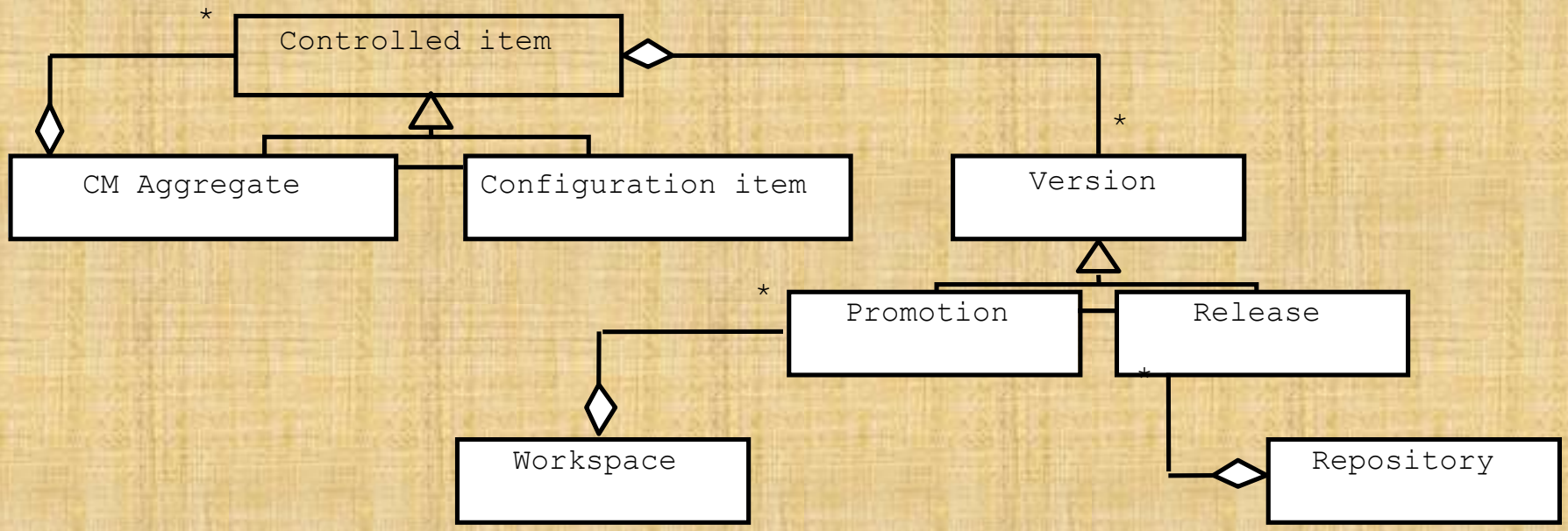
"Tập hợp phần cứng, phần mềm hoặc cả hai, được chỉ định để quản lý cấu hình và được coi như một thực thể duy nhất trong quy trình quản lý cấu hình".

Configuration Item

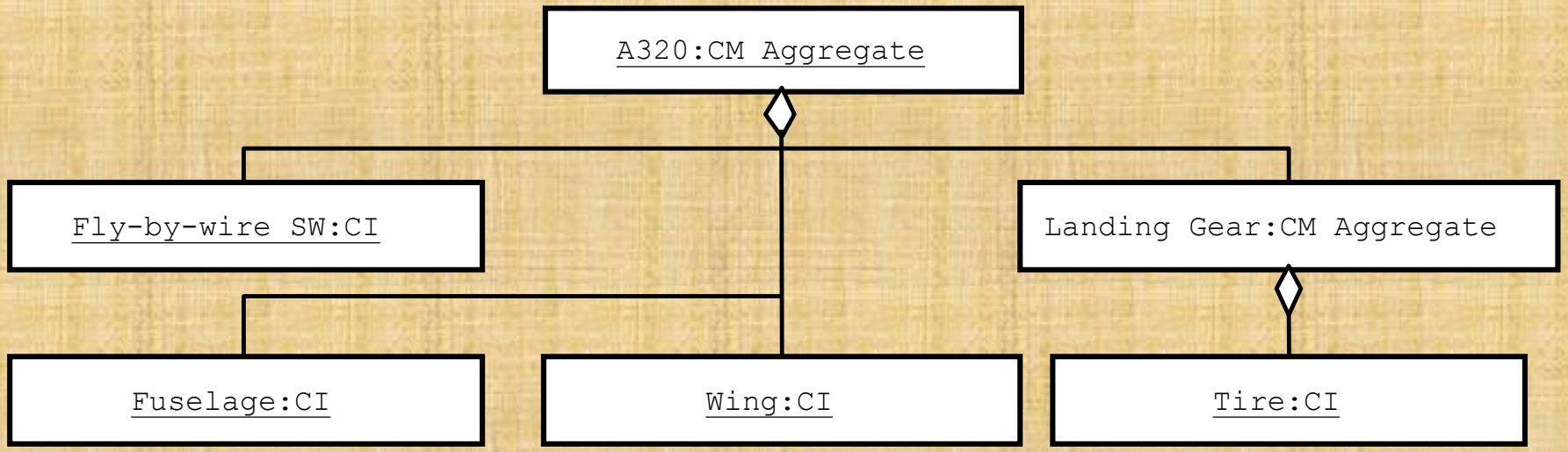
- Các mục cấu hình phần mềm không chỉ là các đoạn mã chương trình mà là tất cả các loại tài liệu cho sự phát triển phần mềm, ví dụ:
  - các tệp mã
  - trình điều khiển cho các trường hợp kiểm thử
  - tài liệu phân tích hoặc thiết kế
  - tài liệu hướng dẫn người dùng
  - cấu hình hệ thống (ví dụ: phiên 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)!

Configuration management concepts (UML class diagram).



An example of CM aggregates and configuration items



Finding Configuration Items (CIs)

- 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

Finding Configuration Items (continued)

- 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 vẫn mong đợi sự hỗ trợ thích hợp trong nhiều năm.
- Một lược đồ đặt tên thực thể nên được xác địnhđể các tài liệu liên quan 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

Terminology: Baseline

"A specification or product that has been formally reviewed and agreed to by responsible management, that thereafter serves as the basis for further development, and can be changed only through formal change control procedures."

"Một đặc tả hoặc sản phẩm đã được xem xét và chấp nhận, sau đó sẽ là cơ sở để phát triển thêm và chỉ có thể được thay đổi thông qua các thủ tục kiểm soát thay đổi chính thức."

Terminology: Baseline

Examples:

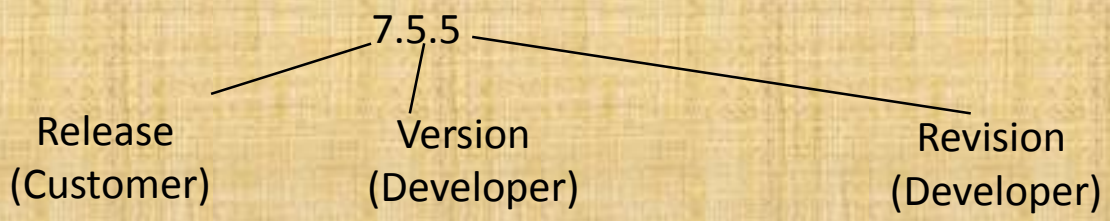
Baseline A: API của một chương trình được xác định hoàn toàn; phần thân của các phương thức trống.

Baseline B: Tất cả các phương pháp truy cập dữ liệu được thực hiện và thử nghiệm; lập trình GUI có thể bắt đầu.

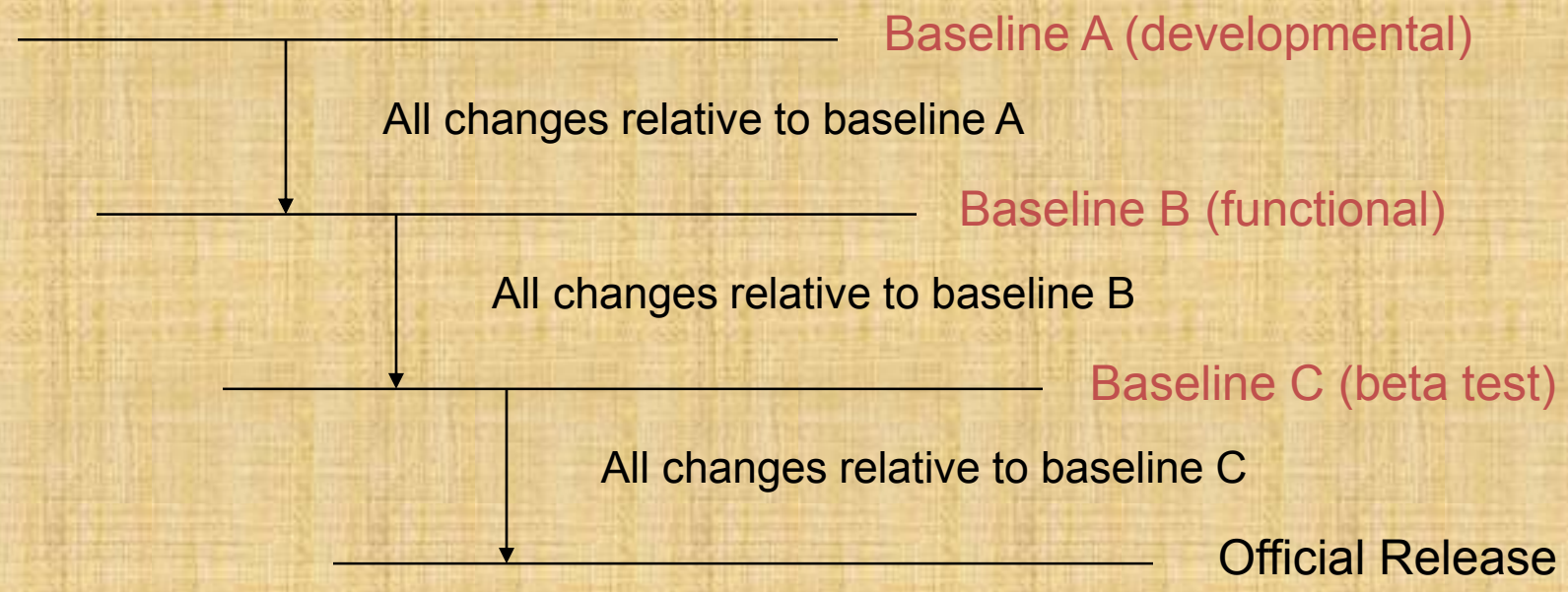
Baseline C: GUI được triển khai, giai đoạn thử nghiệm có thể bắt đầu.

More on Baselines

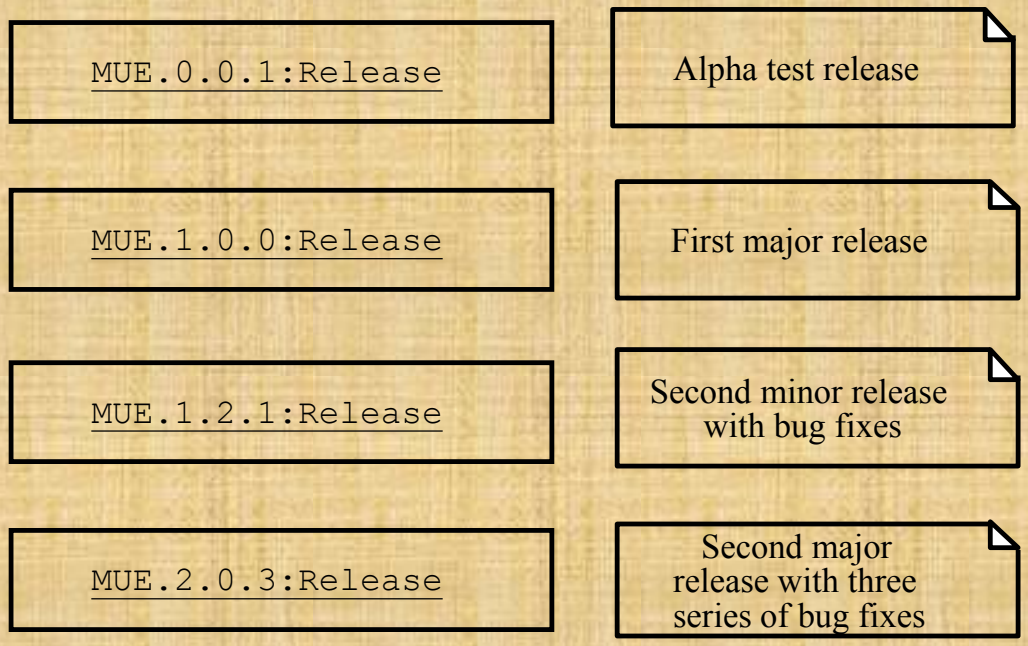
- Khi hệ thống được phát triển, một loạt baseline được phát triển, thường là sau khi xem xét (đánh giá phân tích, đánh giá thiết kế, xem xét mã, kiểm tra hệ thống, chấp nhận khách hàng, ...)
- Nhiều lược đồ đặt tên cho đường cơ sở tồn tại (1.0, 6.01a, ...)
- Lược đồ 3 chữ số:



Baselines in SCM



Three digit version identification scheme



Three-digit version identification scheme

<version> ::= <configuration item name>.<major>.<minor>.<revision>

<major> ::= <nonnegative integer>

<minor> ::= <nonnegative integer>

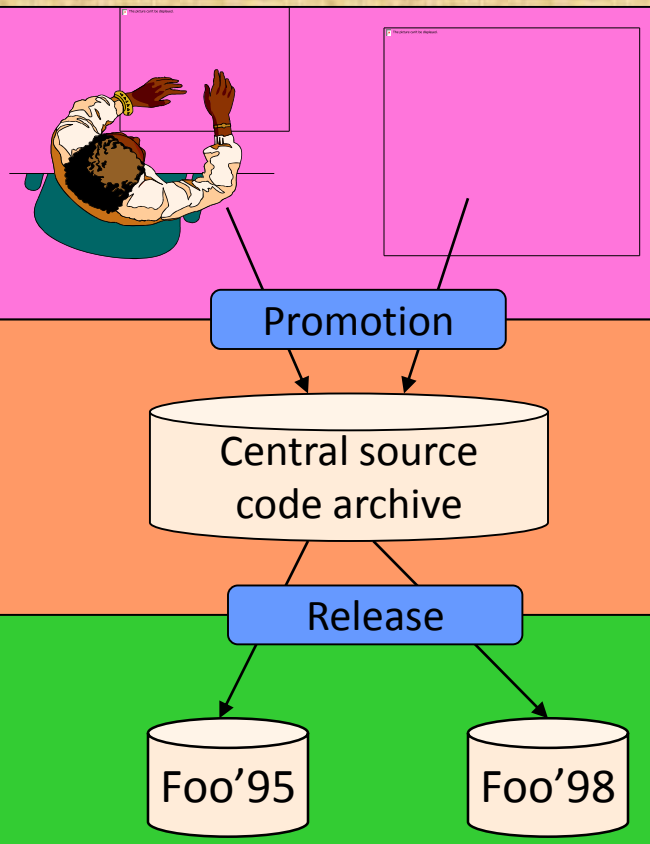
<revision> ::= <nonnegative integer>

SCM Directories

- 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.

Standard SCM Directories

- 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.



Change management

- Quản lý thay đổi là việc xử lý các yêu cầu thay đổi
  - Một yêu cầu thay đổi dẫn đến việc tạo ra một bản phát hành mới
- Quy trình 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.
- 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 nữa.

Version vs. Revision vs. Release

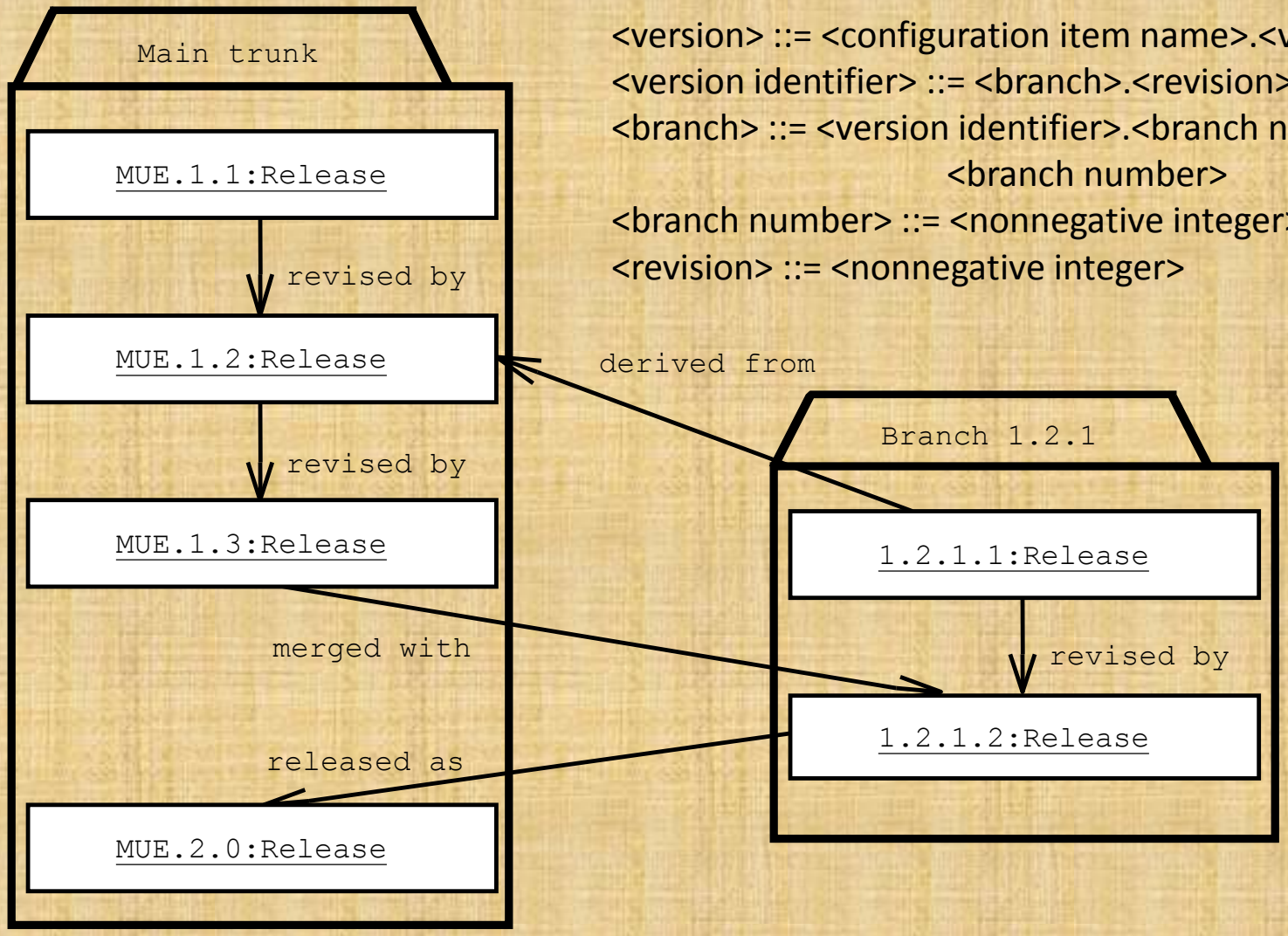
- Version:
  - An *initial* release or re-release of a configuration item associated with a *complete compilation* or recompilation of the item. Different versions have different functionality.
- Revision:
  - *Change* to a version that corrects only errors in the design/code, but does not affect the documented functionality.
- Release:
  - The *formal distribution* of an approved version.

Quiz: Is Windows98 a new version or a new revision compared to Windows95 ?

Managing Concurrent Development

- Trong các dự án lớn, các nhà phát triển thường muốn thay đổi cùng các hạng mục (giống nhau)
- Cần hỗ trợ đồng thời các phiên bản đã phát hành và phát triển mới.
- HOW?

Branches: CVS



<version> ::= <configuration item name>.<version identifier>
<version identifier> ::= <branch>.<revision>
<branch> ::= <version identifier>.<branch number> |
<branch number>
<branch number> ::= <nonnegative integer>
<revision> ::= <nonnegative integer>

6. SCM planning

- Lập kế hoạch quản lý cấu hình phần mềm bắt đầu trong giai đoạn đầu của dự án.
- Kết quả của giai đoạn lập kế hoạch SCM là Kế hoạch quản lý cấu hình phần mềm (SCMP) có thể được mở rộng hoặc sửa đổi trong phần còn lại của dự án.
- SCMP có thể tuân theo tiêu chuẩn công khai như IEEE 828 hoặc tiêu chuẩn nội bộ (ví dụ: của công ty).

The Software Configuration Management Plan

- Xác định các loại tài liệu được quản lý và sơ đồ đặt tên tài liệu.
- Xác định người chịu trách nhiệm về các thủ tục CM và việc tạo ra các baseline.
- Xác định các chính sách để kiểm soát thay đổi và quản lý phiên bản.
- Mô tả các công cụ nên được sử dụng để hỗ trợ quá trình CM và bất kỳ hạn chế nào trong việc sử dụng chúng.
- Xác định cơ sở dữ liệu quản lý cấu hình được sử dụng để ghi lại thông tin cấu hình.

Outline of a Software Configuration Management Plan (SCMP, IEEE 828-1990)

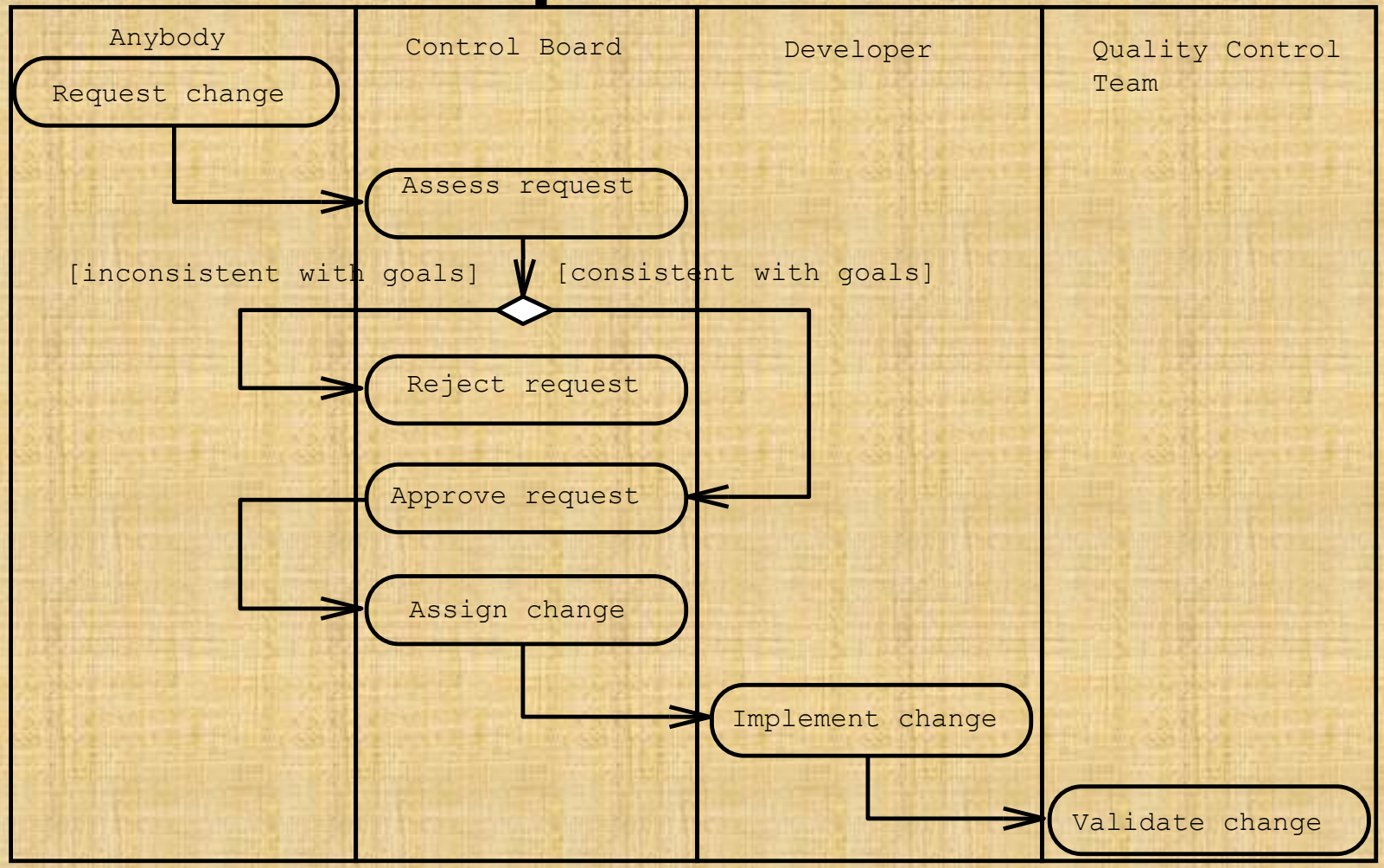
- 1. Introduction
  - Describes purpose, scope of application, key terms and references
- 2. Management (WHO?)
  - Identifies the responsibilities and authorities for accomplishing the planned configuration management activities
- 3. Activities (WHAT?)
  - Identifies the activities to be performed in applying to the project.
- 4. Schedule (WHEN?)
  - Establishes the sequence and coordination of the SCM activities with project mile stones.
- 5. Resources (HOW?)
  - Identifies tools and techniques required for the implementation of the SCMP
- 6. Maintenance
  - Identifies activities and responsibilities on how the SCMP will be kept current during the life-cycle of the project.

7. Tools for Software Configuration Management

- Quản lý cấu hình phần mềm thường được hỗ trợ bởi các công cụ với các chức năng khác nhau.

- Examples:
  - RCS
    - very old but still in use; only version control system
  - CVS
    - based on RCS, allows concurrent working without locking
  - Perforce
    - Repository server; keeps track of developer's activities
  - ClearCase
    - Multiple servers, process modeling, policy check mechanisms

An example of change management process



Summary

- 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.
- SCM được thực hiện theo kế hoạch SCM. Kế hoạch này có thể tuân theo tiêu chuẩn công khai (ví dụ IEEE 828) hoặc tiêu chuẩn nội bộ.
- Cần phải điều chỉnh một tiêu chuẩn cho một dự án cụ thể:
  - Các dự án lớn cần có kế hoạch chi tiết để thành công
  - Các dự án nhỏ không đủ khả năng gánh vác những kế hoạch như vậy
- SCM được hỗ trợ bởi các công cụ. Chức năng của chúng thay đổi từ các công cụ lưu trữ phiên bản đơn giản đến các hệ thống rất phức tạp với các quy trình tự động để kiểm tra chính sách và hỗ trợ tạo tài liệu SCM.

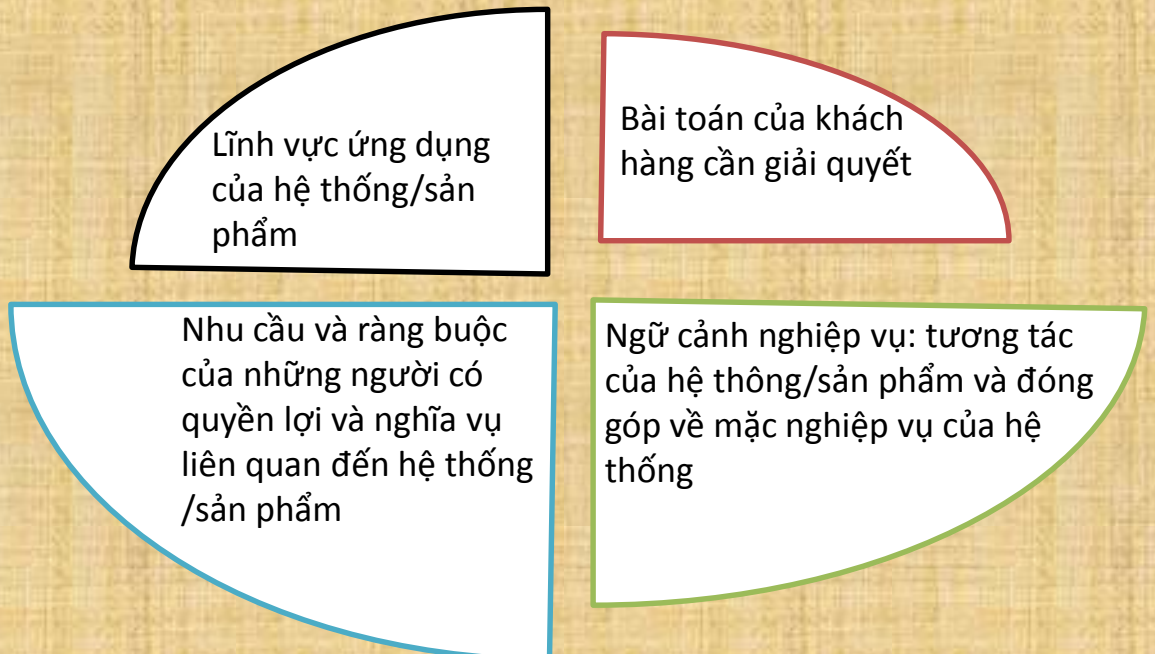
NHẬP MÔN
CÔNG NGHỆ PHẦN MỀM
(INTRODUCTION TO SOFTWARE ENGINEERING)

Chương 5: Kỹ nghệ yêu cầu phần mềm (Requirement Engineering)

- **1. Tổng quan về yêu cầu phần mềm**
- 2. Quy trình xác định yêu cầu phần mềm
- 3. Phương pháp và công cụ đặc tả yêu cầu phần mềm
- 4. Nguyên lý phân tích yêu cầu sử dụng

Khái niệm

- 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



Mục đích xác định yêu cầu 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

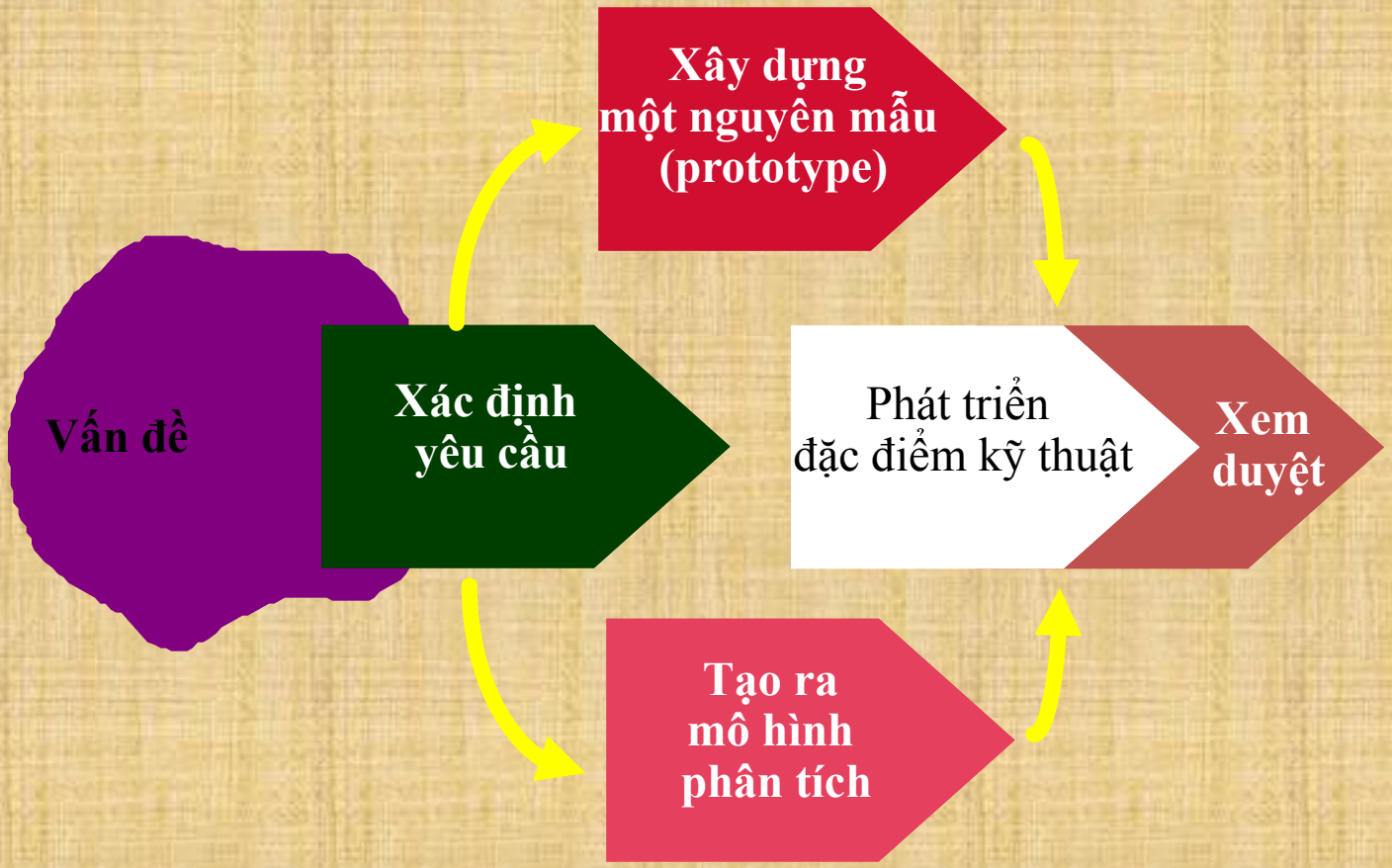
Phân loại yêu cầu

- 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

2. Quy trình xác định yêu cầu PM

- 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)

Quy trình xác định yêu cầu PM (tiếp)



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)

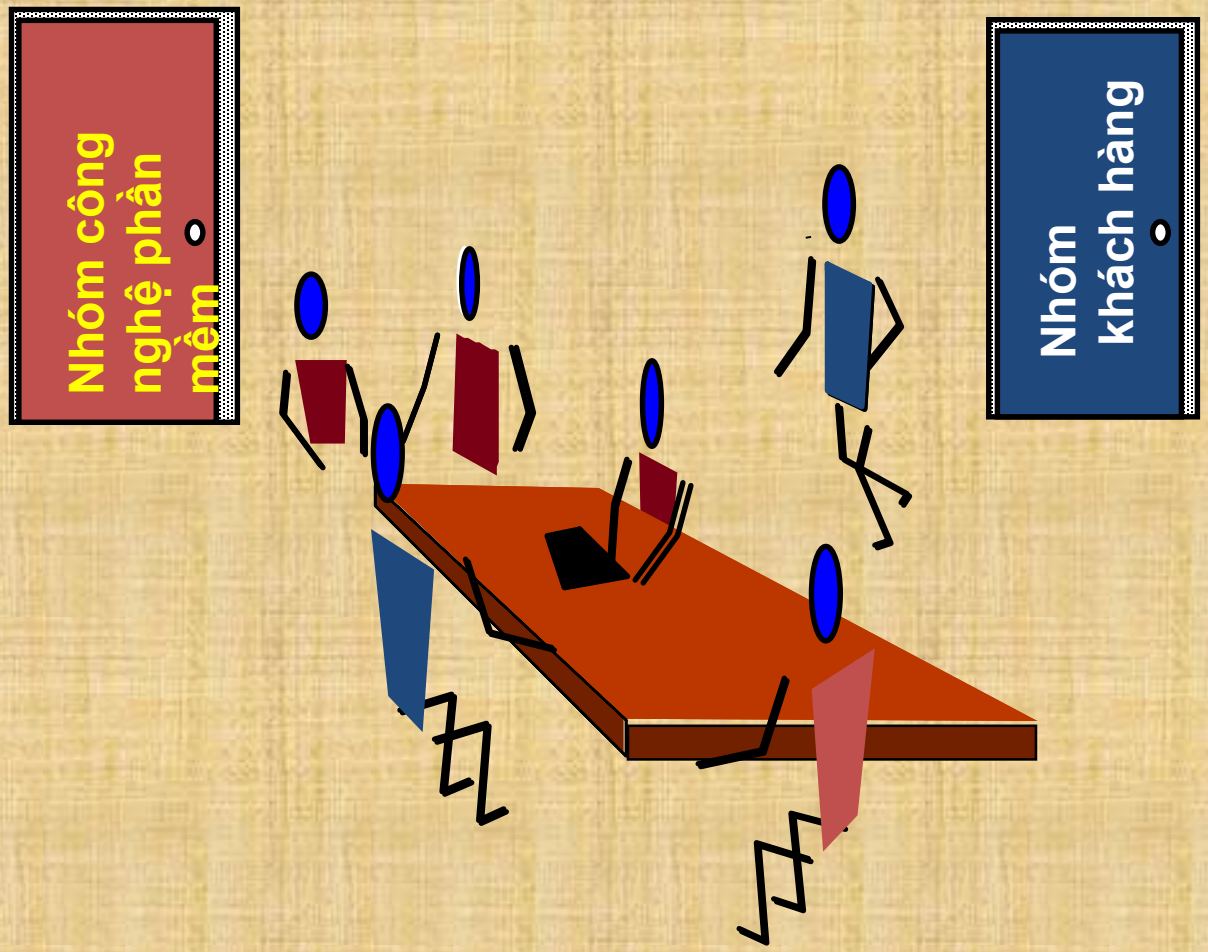
Phát hiện yêu cầu phần mềm

- Xác định các phương pháp sử dụng để phát hiện các yêu cầu phần mềm: phỏng vấn, làm việc nhóm, các buổi họp, gặp 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.

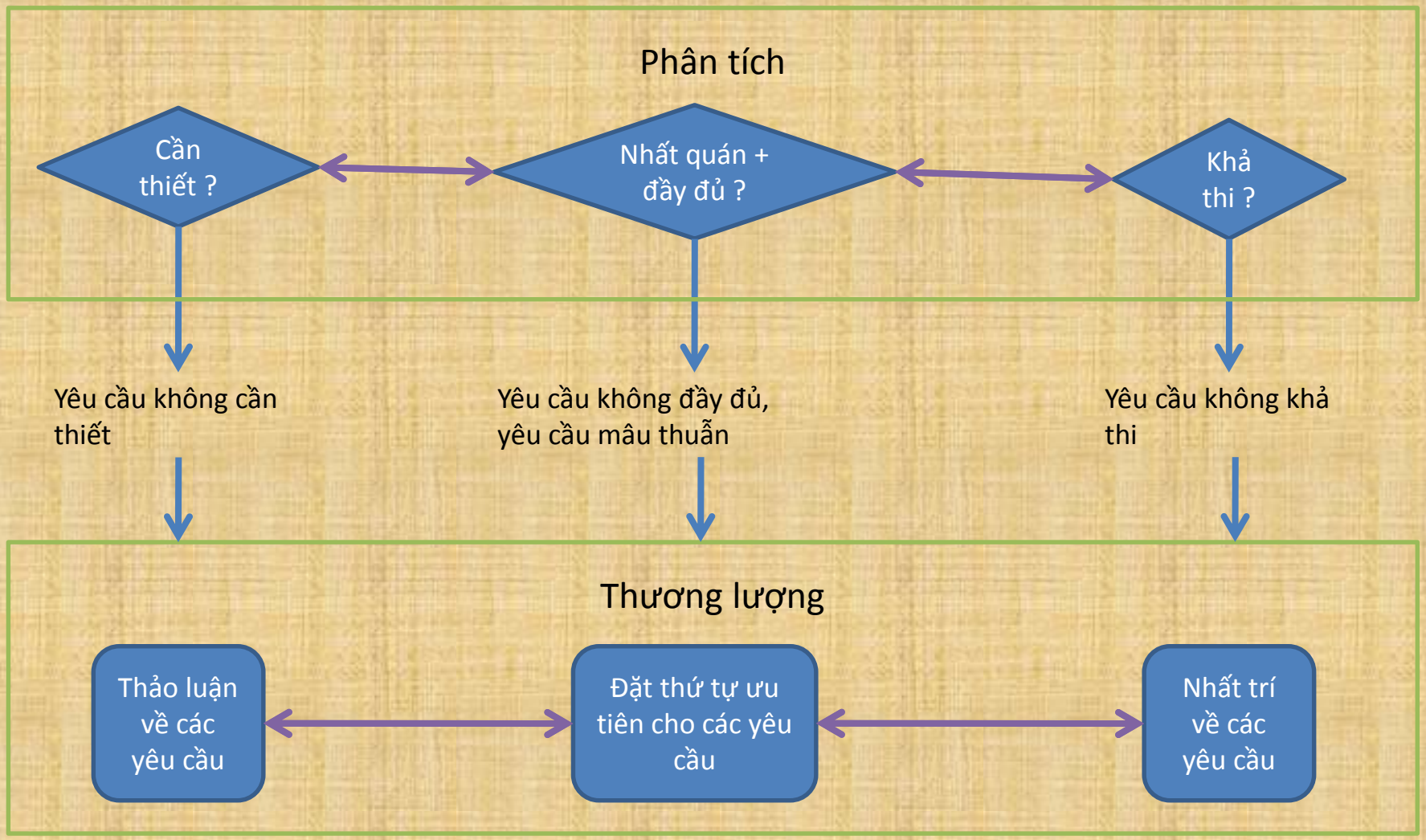
Đầu ra của bước phát hiện yêu cầu phần mềm

- 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

Phân tích các yêu cầu PM và thương lượng với khách hàng



Phân tích các yêu cầu phần mềm và thương lượng với khách hàng



Phân tích các yêu cầu phần mềm và thương lượng với khách hàng

- 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

Phân tích các yêu cầu phần mềm và thương lượng với khách hàng

- 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

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.

Đặc tả 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 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.

Đặc tả 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

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 HT
- 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 HT phải làm hơn là HT phải làm thế nào (PT chứ không phải là TK)

Nội dung cần có của tài liệu yêu cầu

3. Phương pháp và công cụ đặc tả yêu cầu phần mềm

- 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)

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)

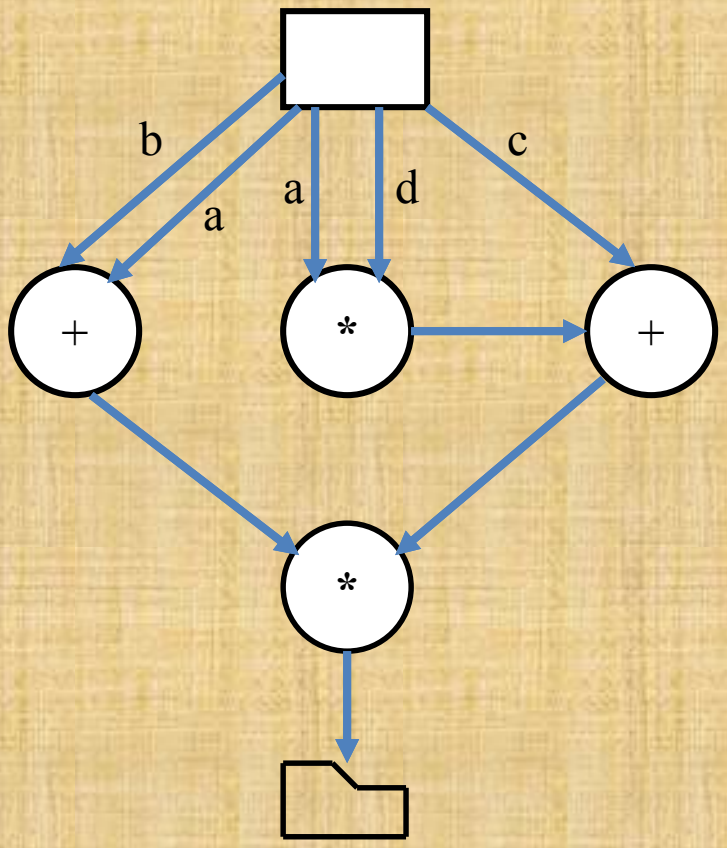
Thể hiện luồng dữ liệu

Kho dữ liệu

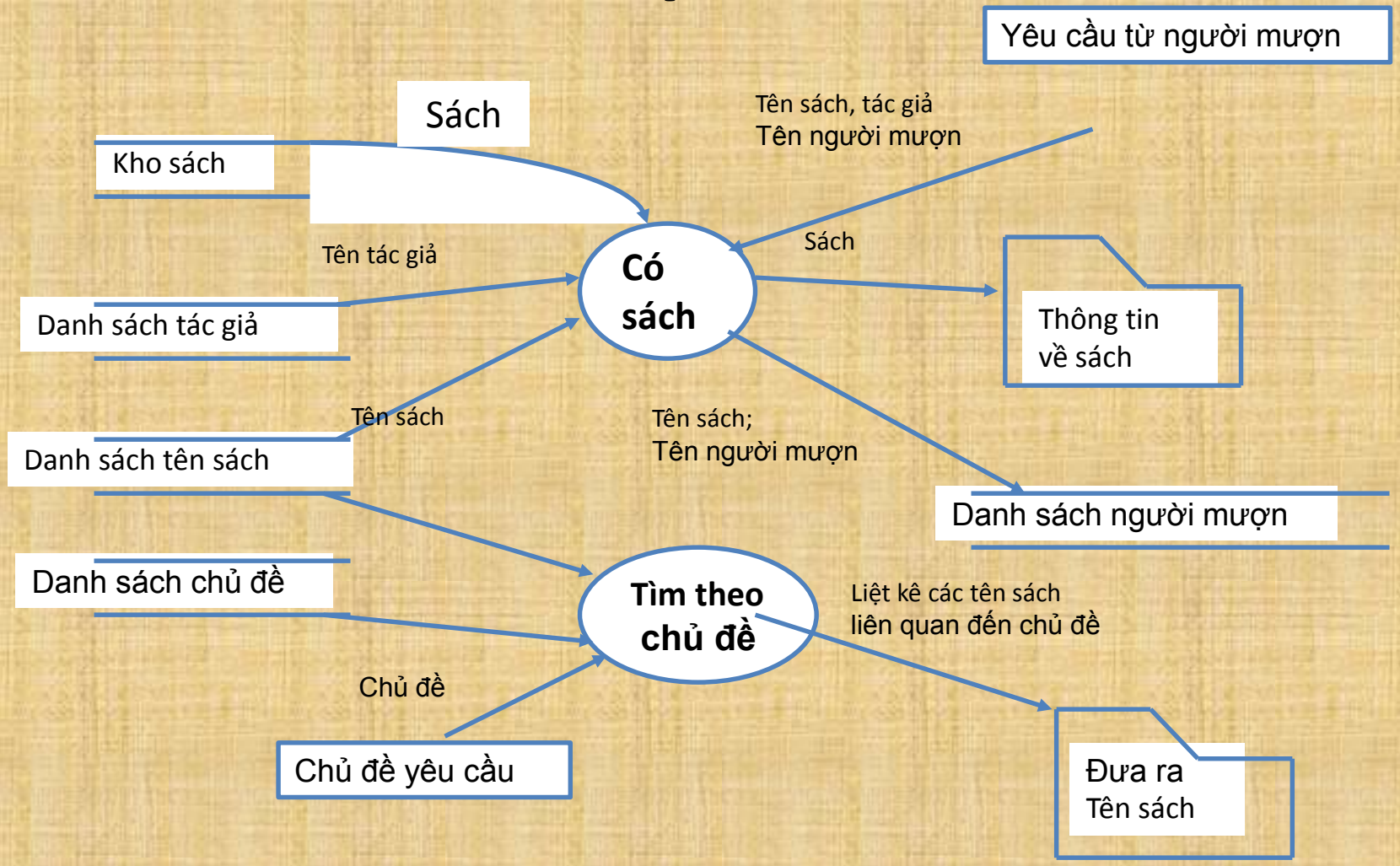
Vào ra dữ liệu và tương tác giữa hệ thống và người sử dụng

Ví dụ: mô tả biểu thức toán học bằng DFD

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



Ví dụ đặc tả các chức năng của thư viện qua DFD



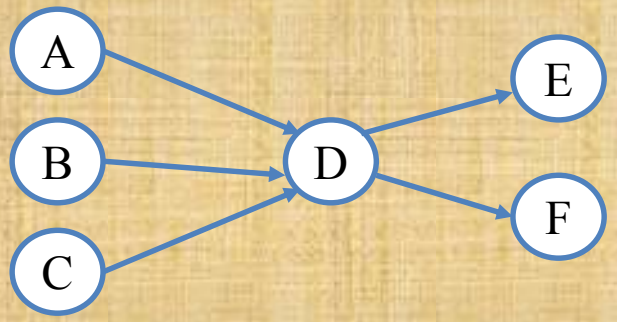
Các hạn chế của DFD

- Ý nghĩa của các ký pháp sử dụng được xác định bởi các định danh lựa chọn của NSD
- Ví dụ: DFD của chức năng tìm kiếm sách:

If NSD nhập vào cả tên tác giả và tiêu đề sách Then
tìm kiếm sách tương ứng, không có thì thông báo lỗi
Elseif chỉ nhập tên tác giả Then
hiển thị danh sách các sách tương ứng với
tên tác giả đã nhập và yêu cầu NSD lựa chọn sách
Elseif chỉ nhập tiêu đề sách Then
. . .
Endif

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 trang mất dữ liệu

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: quản lý thư viện

- Xét các giao dịch:
  - Mượn sách / Trả sách
  - Thêm đầu sách / Loại bỏ đầu sách
  - Liệt kê danh sách các đầu sách theo tên tác giả hay theo chủ đề
  - Tìm kiếm sách theo các yêu cầu của người mượn
  - Tìm kiếm sách quá hạn trả, . . .

Đặc tả yêu cầu

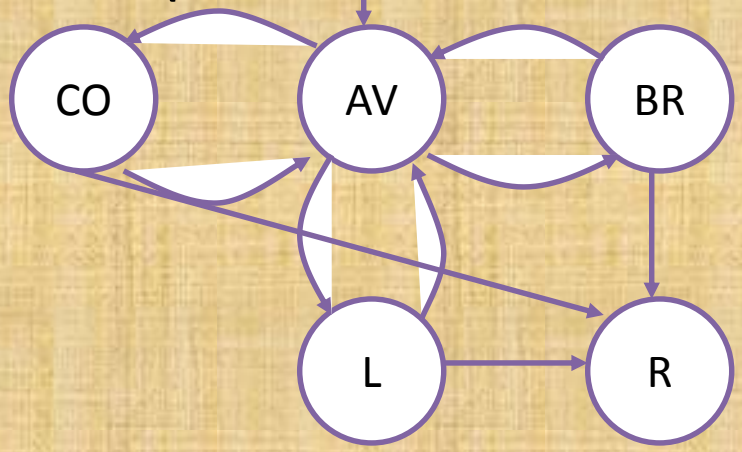
- Độc giả không được mượn quá một số lượng sách nhất định, trong một thời gian nhất định
- Một số sách không được mượn về
- Một số người không được mượn một số loại sách nào đó, . . .

Đặc tả các đối tượng

- Các đối tượng:
  - Tên sách
  - Mã quyển
  - Nhân viên phục vụ
  - Người mượn
- Cần có:
  - tập hợp (danh sách) các tiêu đề sách
  - danh sách các tác giả cho từng quyển sách,
  - danh sách các chủ đề liên quan của các quyển sách

FSM đặc tả trạng thái

- Ta có tập hợp các sách (mỗi đầu sách có thể có nhiều quyển sách trong thư viện).
- Mỗi quyển sách có thể có 1 trong 5 trạng thái sau:
  - (AV) Available: được phép mượn,
  - (CO) - (BR): đã mượn (Check Out; Borrow),
  - (L): Last,
  - (R): Remove



Có thể có hạn chế về số sách được mượn cho 1 nhóm độc giả hoặc mọi độc giả, . . .

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

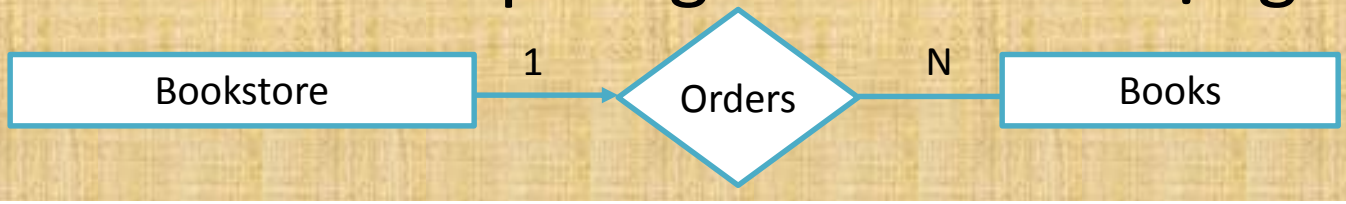
Thuộc tính

- Tính chất của một thực thể hoặc một đối tượng dữ liệu
  - đặt tên cho 1 mẫu (instance) của đối tượng dữ liệu
  - mô tả mẫu (instance)
  - tạo liên kết (reference) đến các mẫu khác

Tập các thuộc tính của 1 đối tượng dữ liệu được xác định thông qua ngữ cảnh của bài toán.

Quan hệ

- Chỉ ra mối liên quan gữa các đối tượng dữ liệu

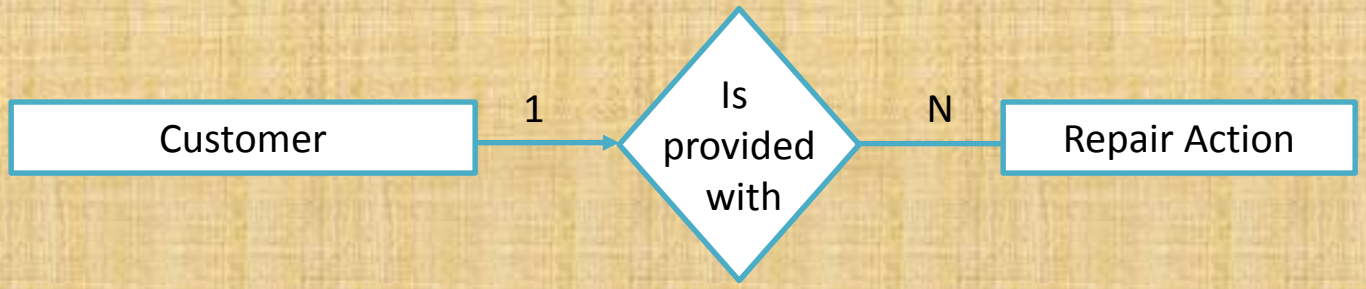


- Cardinality : chỉ ra định lượng của mối quan hệ

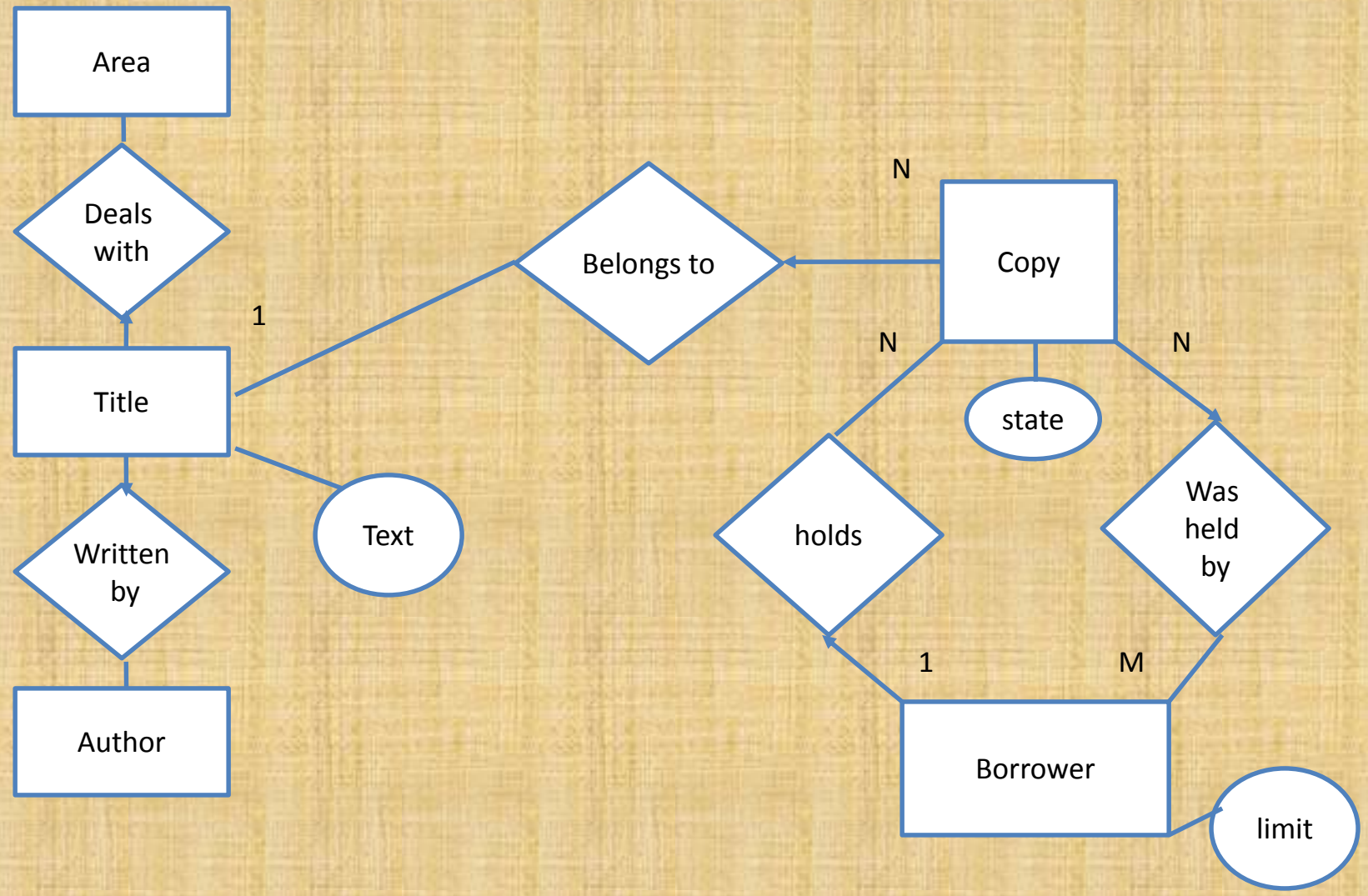
  1:1 one-to-one 1:N one-to-many M:N many-to-many

- Modality : 0 – có thể có, có thể không có quan hệ

  1 – bắt buộc có quan hệ



Ví dụ: ERD mô tả thư viện



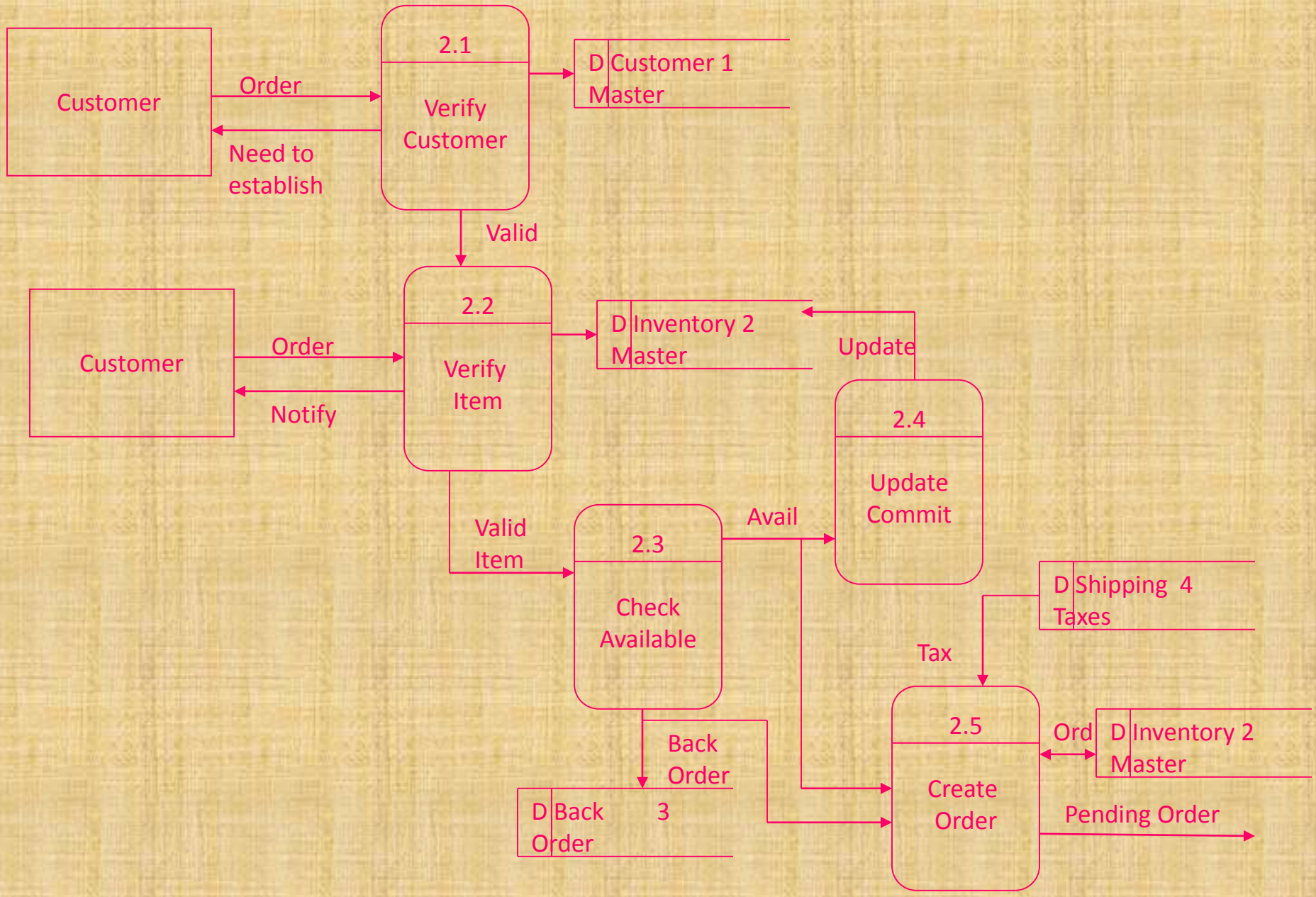
So sánh

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

Thế nào là một đặc tả tốt?


Customer
Order
Need to establish
2.1
Verify Customer
D Customer 1 Master
Valid
Customer
Order
Notify
2.2
Verify Item
D Inventory 2 Master
Update
2.4
Update Commit
Valid Item
2.3
Check Available
Avail
D Shipping 4 Taxes
Tax
Back Order
D Back Order 3
2.5
Create Order
Ord
D Inventory 2 Master
Pending Order

4. Nguyên lý phân tích yêu cầu sử dụng

Mô hình hóa dữ liệu

- Xác định các đối tượng dữ liệu
- Xác định các đặc tính của các đối tượng dữ liệu
- Thiết lập các mối quan hệ giữa các đối tượng dữ liệu

Mô hình hóa các chức năng

- Xác định các chức năng chuyển đổi đối tượng dữ liệu
- Chỉ ra luồng dữ liệu đi qua hệ thống như thế nào
- Biểu diễn bộ phận sản sinh dữ liệu và bộ phận tiêu thụ dữ liệu

Mô hình hóa hành vi

- Chỉ ra các trạng thái (states) khác nhau của hệ thống
- Đặc tả các hiện tượng (events) làm hệ thống thay đổi trạng thái

Phân mảnh các mô hình

- Tinh lọc từng mô hình để biểu diễn các mức trừu tượng thấp hơn
- Lọc đối tượng dữ liệu
- Tạo ra phân cấp chức năng
- Biểu diễn hành vi (behavior) ở các mức chi tiết khác nhau

Bản chất của việc phân tích

- Hãy bắt đầu bằng cách tập trung vào bản chất của vấn đề chứ không xem xét những chi tiết cài đặt

NHẬP MÔN
CÔNG NGHỆ PHẦN MỀM
(INTRODUCTION TO SOFTWARE ENGINEERING)

Chương 8: Xây dựng phần mềm

- 1. Khái niệm

1. Khái niệm

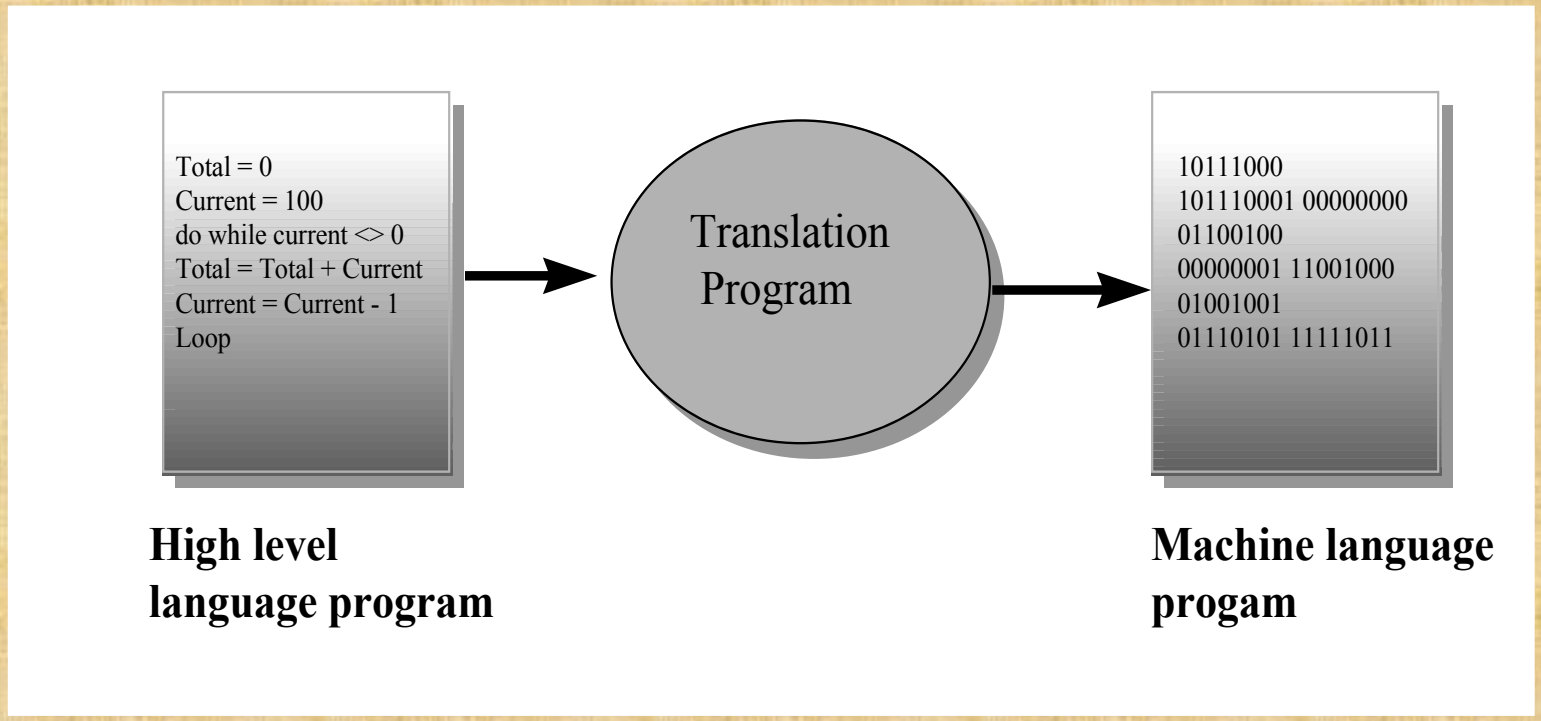
- Mã hóa là quá trình chuyển đổi thiết kế của một hệ thống sang một ngôn ngữ máy.
- Giai đoạn viết mã này liên quan đến việc chuyển đặc tả thiết kế thành mã nguồn.
- Việc biên soạn tài liệu đi kèm với mã nguồn là cần thiết để có thể dễ dàng xác minh sự phù hợp giữa mã với bản đặc tả của nó.
- Công việc mã hóa được thực hiện bởi lập trình viên là người độc lập với người thiết kế. Mục tiêu không phải là giảm nỗ lực và chi phí của giai đoạn mã hóa, mà là để cắt giảm chi phí của các giai đoạn sau.
- Chi phí kiểm thử và bảo trì có thể được giảm đáng kể với việc mã hóa hiệu quả.

Mục tiêu của lập trình

- 1. Để chuyển thiết kế của hệ thống sang ngôn ngữ máy, thực hiện các tác vụ theo chỉ định của thiết kế.
- 2. Để giảm chi phí của các giai đoạn sau: Chi phí kiểm tra và bảo trì có thể giảm đáng kể với việc mã hóa hiệu quả.
- 3. Làm cho chương trình dễ đọc hơn: Chương trình phải dễ đọc và dễ hiểu. 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.

Để tiến hành việc cài đặt thiết kế, cần phải sử dụng ngôn ngữ lập trình bậc cao.

Translating from High-level Language to Binary



2. Lịch sử ngôn ngữ lập trình

- Các ngôn ngữ thế hệ thứ nhất:
  - Ngôn ngữ lập trình mã máy (machine code)
  - Ngôn ngữ lập trình assembly
- Các ngôn ngữ thế thế thứ hai:
  - FOTRAN, COBOL, ALGOL, BASIC
  - Phát triển 1950-1970
- Các ngôn ngữ thế hệ thứ ba
  - Ngôn ngữ lập trình cấp cao vạn năng (cấu trúc)
  - Lập trình hướng đối tượng
  - Lập trình hướng suy diễn – logic
- Các ngôn ngữ thế hệ thứ tư

Các loại ngôn ngữ lập trình

Procedural: Các chương trình nguyên khối chạy từ đầu đến cuối và không có sự can thiệp của người dùng ngoài đầu vào

- Basic, QBasic, QuickBasic
- COBOL
- FORTRAN
- C

Object Oriented/Event Driven (OOED): Các chương trình sử dụng các objects để đáp ứng các sự kiện (events); sử dụng các đoạn mã cho mỗi object

- JAVA
- Visual Basic
  - **Visual Basic for Applications (VBA)**
- Visual C++

Các đặc điểm của ngôn ngữ lập trình

3. Các công cụ lập trình

- Môi trường: DOS, WINDOWS, UNIX/LINUX
- Editors, Compilers, Linkers, Debuggers
- TURBO C, PASCAL
- MS C, Visual Basic, Visual C++, ASP
- UNIX/LINUX: C/C++, gcc (Gnu C Compiler)
- JAVA, CGI, perl
- C#, .NET

Các công cụ lập trình

- Công cụ lập trình C/C++:
  - Turbo C
  - Visual Studio
  - Eclipse
- Công cụ lập trình Java
  - Eclipse
  - Netbean
- Công cụ lập trình C#, .NET
  - Visual Studio.NET
  - MSDN Library
- Công cụ lập trình web
  - Visual Studio và SQL Server

4. Quy trình lập trình

- **Xác định bài toán**
  - **Đầu vào**
  - **Đầu ra**
- **Các bước xử lý để tạo kết quả**

Input	**Processing**	**Output**
Num-1	Read 3 numbers	Total
Num-2	Add numbers together	
Num-3	Print Total number	

Các bước trong lập trình

- Lập trình
- Kiểm tra thuật toán đã được cài đặt
- Thực hiện chương trình trên máy tính
  - Compile
  - Correct syntax errors
  - Run program with test data
  - Correct logic errors
- Viết tài liệu

Phương pháp lập trình có cấu trúc

- Outline Solution
  - Nhiệm vụ chính
  - Các bước xử lý chính
  - Các cấu trúc điều khiển chính (vòng lặp, rẽ nhánh, v.v.)
  - Các biến chính

Phương pháp lập trình có cấu trúc

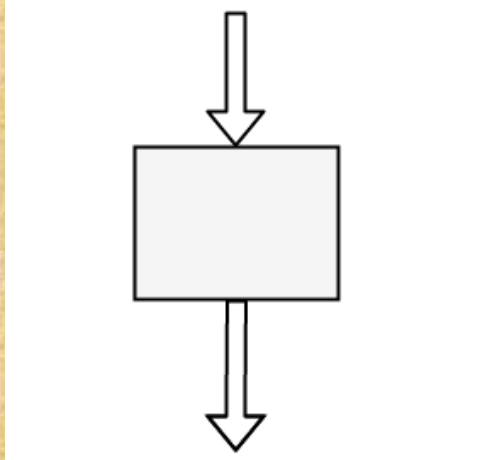
- Chia toàn bộ chương trình thành các mô-đun nhỏ để chương trình trở nên dễ hiểu. Mục đích của lập trình cấu trúc là tuyến tính hóa luồng điều khiển thông qua một chương trình máy tính sao cho trình tự thực thi tuân theo trình tự mà mã được viết.
- Cấu trúc động của chương trình giống với cấu trúc tĩnh của chương trình. Điều này nâng cao khả năng đọc, khả năng kiểm tra và khả năng sửa đổi của chương trình. Luồng kiểm soát tuyến tính này có thể được quản lý bằng cách hạn chế tập hợp các cấu trúc ứng dụng được phép thành một mục nhập duy nhất, các định dạng lối ra duy nhất.
- Chúng tôi sử dụng lập trình có cấu trúc vì nó cho phép người lập trình hiểu chương trình một cách dễ dàng. Nếu một chương trình bao gồm hàng nghìn lệnh và một lỗi xảy ra thì rất phức tạp để tìm ra lỗi đó trong toàn bộ chương trình, nhưng trong lập trình có cấu trúc, chúng ta có thể dễ dàng phát hiện lỗi và sau đó đến vị trí đó và sửa nó. Điều này giúp tiết kiệm rất nhiều thời gian.

Lập trình có cấu trúc

- Cho phép người lập trình hiểu chương trình một cách dễ dàng.
- Trong lập trình có cấu trúc, chúng ta có thể dễ dàng phát hiện lỗi và tìm đến vị trí để sửa nó.
- Điều này giúp tiết kiệm rất nhiều thời gian.

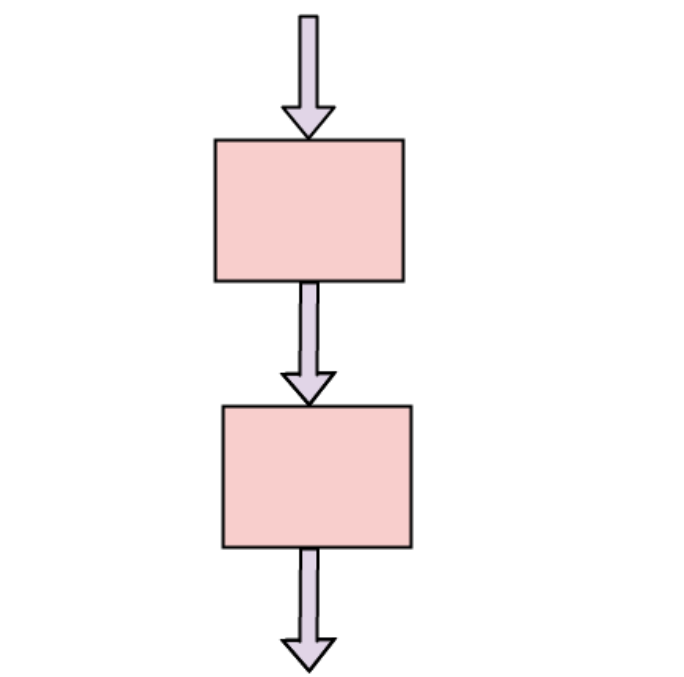
Rule 1: Code Block

- Nếu điều kiện đầu vào đúng, nhưng điều kiện thoát sai thì lỗi đó phải nằm trong khối. Điều này không đúng nếu việc thực thi được phép nhảy vào một khối (do đó lỗi có thể ở bất kỳ đâu trong chương trình, gỡ lỗi trong những trường hợp này khó hơn nhiều).
- **Rule 1 of Structured Programming:** Một khối mã được cấu trúc, như thể hiện trong hình. Trong điều kiện biểu đồ luồng, một hộp có một điểm vào và một điểm thoát được cấu trúc. Lập trình có cấu trúc là một phương pháp làm rõ ràng rằng chương trình là đúng.


Rule1: Code block is structured

Rule 2: Sequence

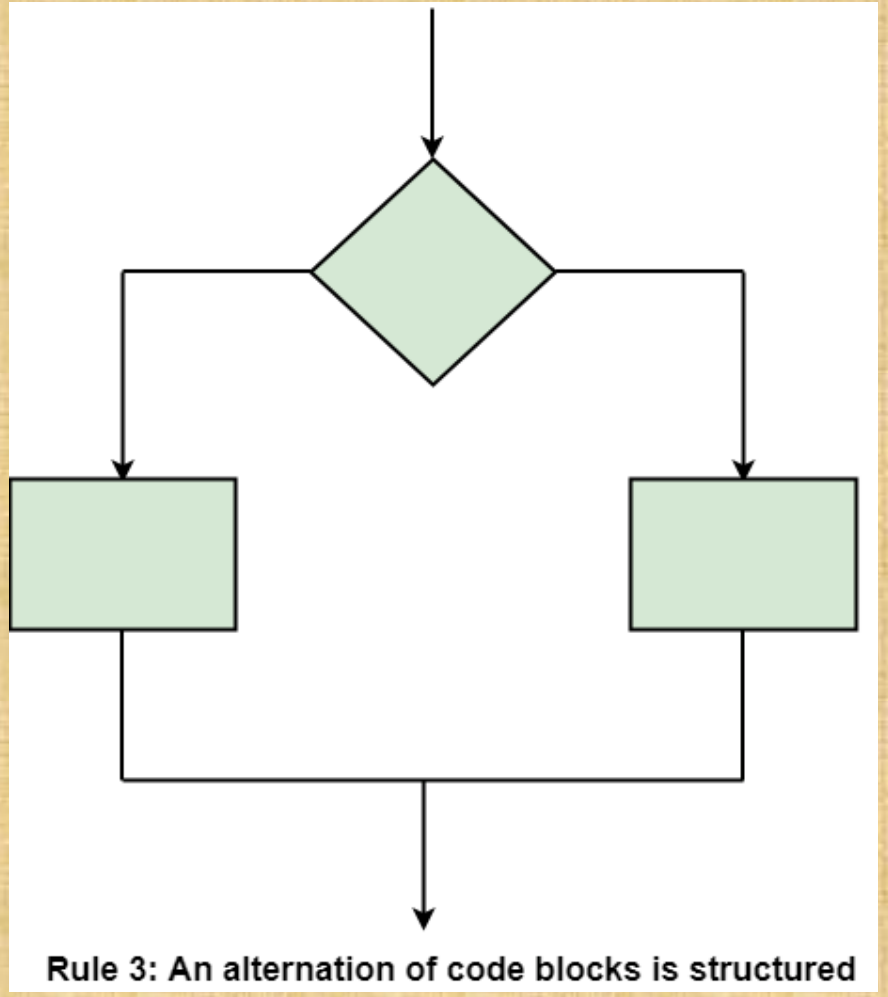
- Một chuỗi khối là đúng nếu điều kiện thoát của mỗi khối khớp với điều kiện vào của khối sau.
- Toàn bộ chuỗi có thể được coi là một khối duy nhất, có điểm vào và điểm ra.
- **Rule 2 of Structured Programming:** Hai hoặc nhiều khối được kết nối liền nhau



Rule2: A sequence of code blocks is structured

Rule Three: Alternation

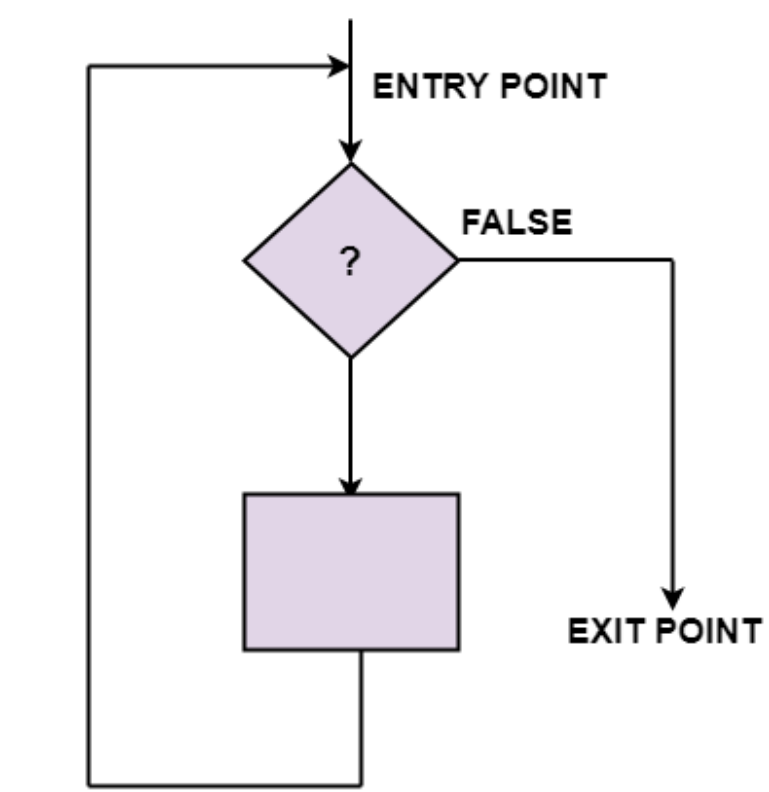
- **Rule 3 of Structured Programming**: Tùy chọn If-then-else, mỗi lựa chọn là một khối mã. Cấu trúc rẽ nhánh được sử dụng để đáp ứng điều kiện thoát.



Rule 3: An alternation of code blocks is structured

Rule 4: Iteration

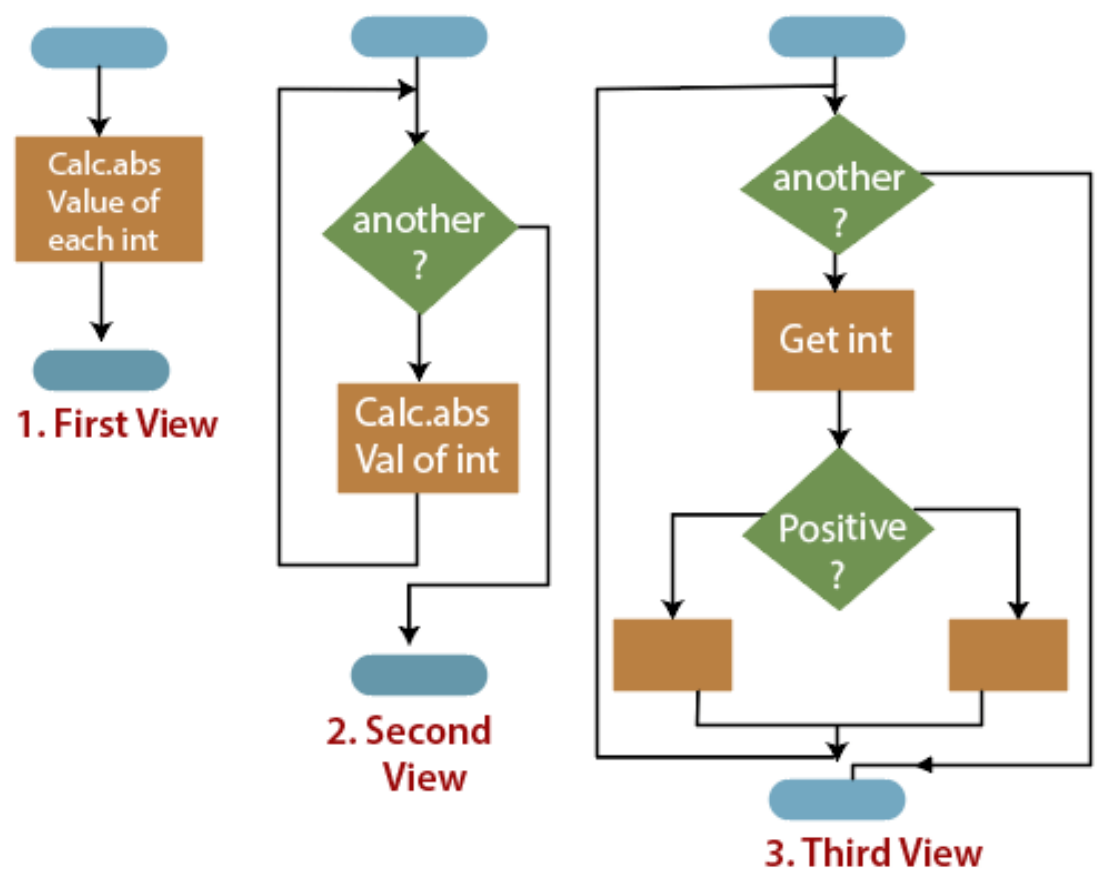
- **Rule 4 of Structured Programming**: Vòng lặp (trong khi) gồm một điểm vào và một điểm ra. Điểm vào có điều kiện phải được thỏa mãn và điểm ra có các yêu cầu sẽ được đáp ứng. Không có bước nhảy vào biểu mẫu từ các điểm bên ngoài của mã.



Rule 4: Iteration of code blocks is structured

Rule 5: Nested Structures

- **Rule 5 of Structured Programming**: Một cấu trúc có một điểm vào và ra duy nhất



Object-Oriented Event-driven Programming (OOED)

OOED sử dụng các đối tượng **objects**, hoặc các modules độc lập kết hợp dữ liệu và mã chương trình để chuyển các message cho nhau.

OOED dễ làm việc hơn vì nó trực quan hơn các phương pháp lập trình truyền thống.

Ví dụ: Visual Basic

Người dùng có thể kết hợp các đối tượng một cách tương đối dễ dàng để tạo ra các hệ thống mới hoặc mở rộng các hệ thống hiện có.

Thuộc tính của đối tượng là thuộc tính liên kết với một đối tượng.

Phương thức của đối tượng là những hoạt động mà đối tượng có thể thực hiện.

Đối tượng phản hồi các sự kiện.

OOED Programming Process

Quy trình sáu bước để viết một chương trình máy tính OOED:

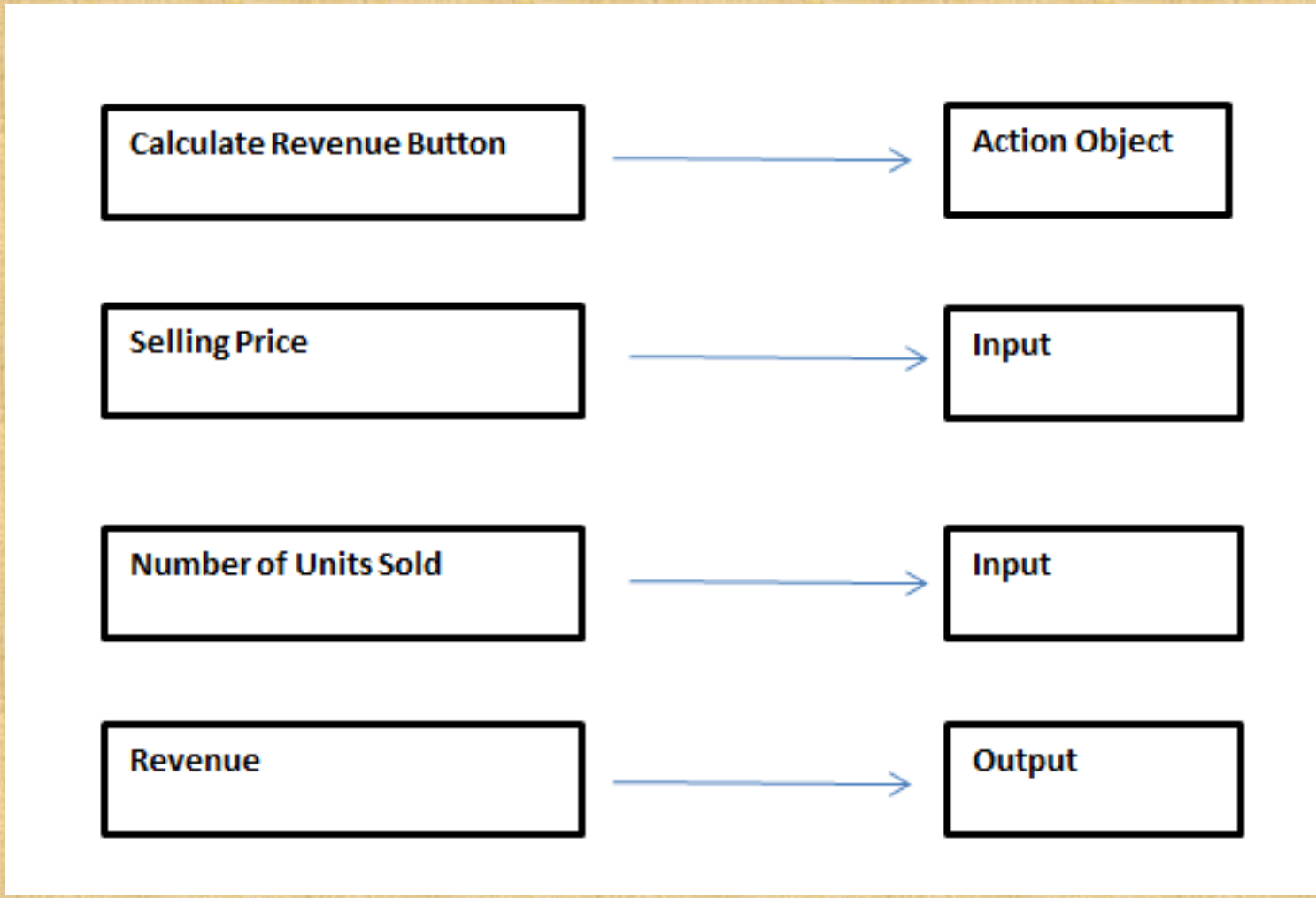
1. Xác định vấn đề.
2. Tạo giao diện
3. Phát triển logic cho các đối tượng hành động
4. Viết và kiểm tra mã cho các đối tượng hành động
5. Kiểm tra dự án tổng thể
6. Làm tài liệu

Bất kể một chương trình được viết tốt như thế nào, mục tiêu sẽ không đạt được nếu chương trình không giải quyết được vấn đề (sai).

Step One: Define Problem

- Trước khi tạo ứng dụng máy tính để giải quyết một vấn đề, trước tiên cần phải xác định rõ nó.
- Cần xác định đầu vào và đầu ra.
- Phải xác định dữ liệu được đưa vào chương trình và kết quả được xuất ra từ nó.
- Phác thảo giao diện (giao tiếp) ứng dụng.
- Xác định các hành động của các đối tượng cần mã hóa.

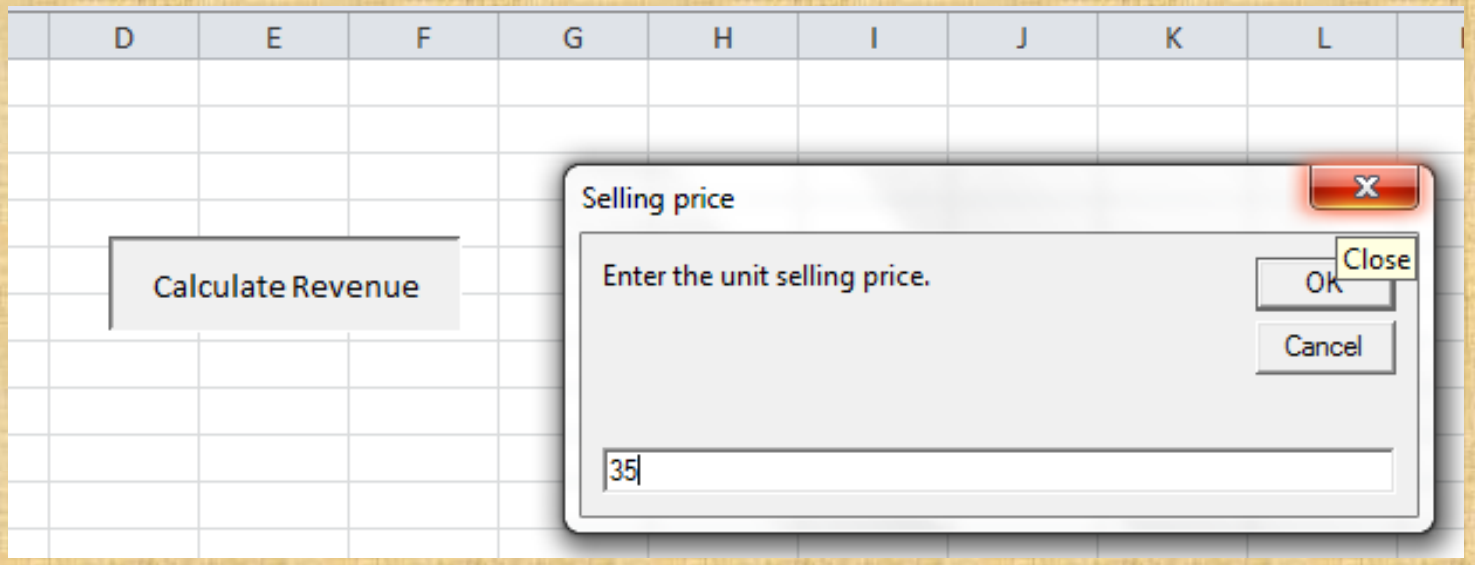
Ví dụ. Giao diện tính toán doanh thu

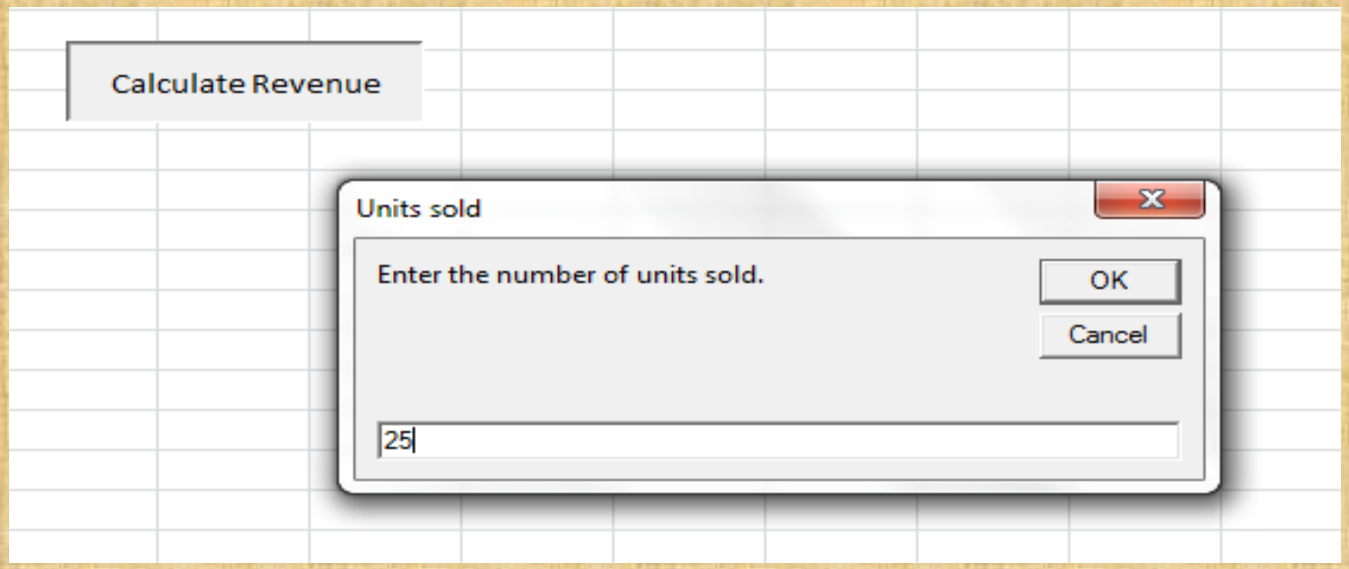


Step Two: Create Interface

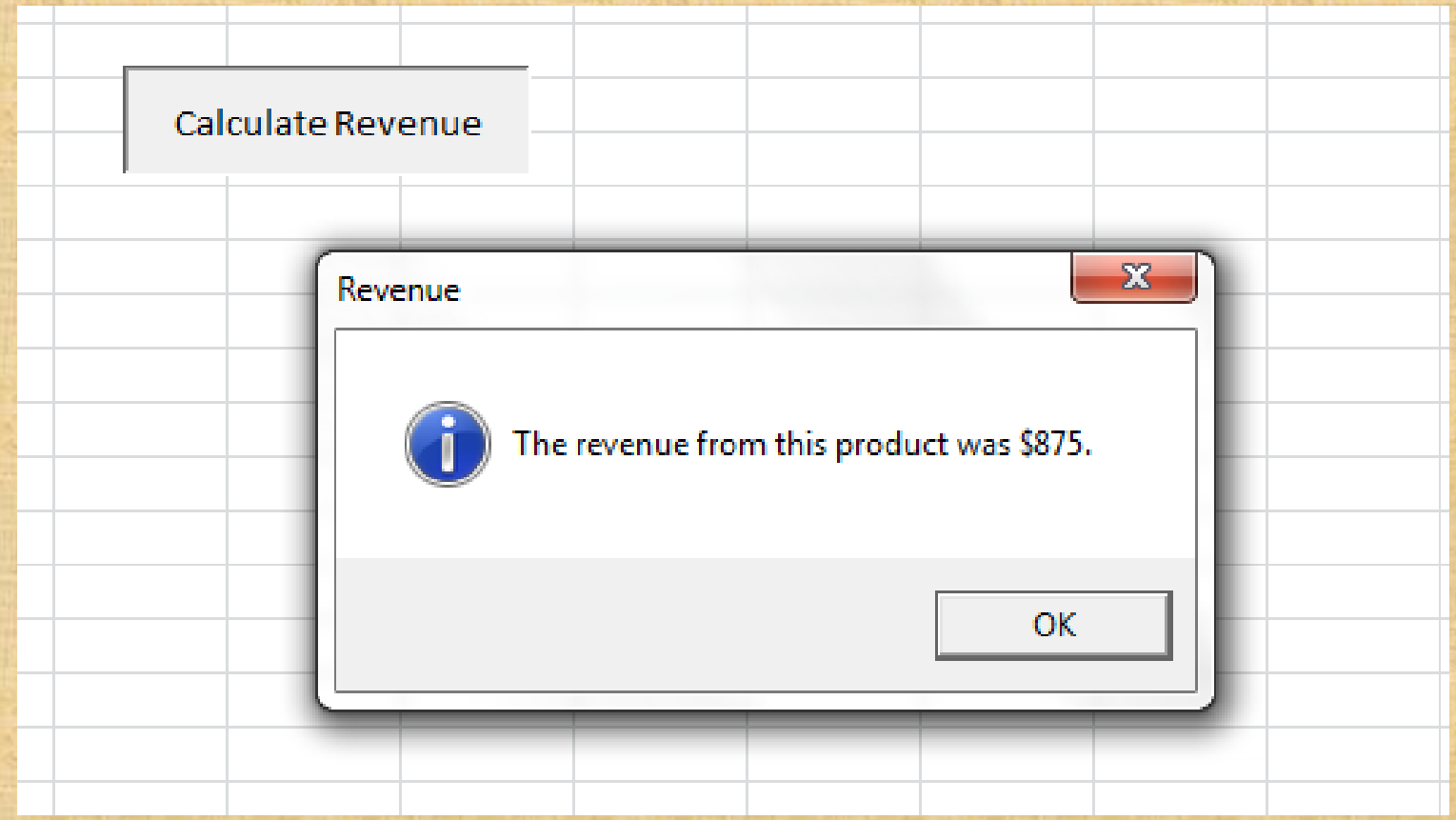
- Ví dụ: tạo giao diện tính toán doanh thu.
  - button for action
  - inputBox for input (Selling Price)
  - inputBox for input (Units Sold)
  - MessageBox for output

Calculate Revenue Interface - Input





Calculate Revenue Interface - Output



Step Three: Develop Logic for Action Objects

- Xác định mỗi đối tượng hành động làm gì để phản ứng với các sự kiện. Đây là logic cho từng đối tượng.
- Sử dụng Bảng Nhập / Xử lý / Đầu ra (IPO) và Mã giả để phát triển logic
- Bảng IPO hiển thị các đầu vào, đầu ra và quá trình xử lý để chuyển đầu vào thành đầu ra

IPO Table for Calculate Revnue Button

Input	Processing	Output
Unit Price Quantity Sold	revenue = unitPrice X quantitySold	Revenue

Pseudocode for Count High Values Button

```
Begin procedure
    Input Selling Price
   Input Quantity Sold
    Revenue = Selling Price x Quantity Sold
    Output Revenue

End procedure
```

Step Four: Write and Test Code of Action Objects

- Chuyển đổi logic được thể hiện trong mã giả sang ngôn ngữ lập trình (sử dụng bộ từ vựng và cú pháp của một ngôn ngữ).
- Kiểm tra và sửa mã đối tượng được viết, giai đoạn này được gọi là gỡ lỗi.

VBA Code for Calculate Revenue Button

```
Sub CalculateRevenue()
    Dim unitPrice As Currency
    Dim quantitySold As Integer
    Dim revenue As Currency

    ' Get the user's inputs, then calculate revenue.
    unitPrice = InputBox("Enter the unit selling price.", "Selling price")
    quantitySold = InputBox("Enter the number of units sold.", "Units sold")
    revenue = unitPrice * quantitySold

    ' Report the results.
    MsgBox "The revenue from this product was " & Format(revenue, "$#,##0") _
        & ".", vbInformation, "Revenue"
End Sub
```

Step Five: Test Overall Project

- Kiểm tra toàn bộ project cần đảm bảo mỗi câu lệnh phải hoàn toàn chính xác nếu không thì chương trình có thể sẽ có lỗi.
- Cần đảm bảo chương trình được kiểm tra trên môi trường và dữ liệu thực thi thực tế mà chương trình sẽ được sử dụng.

Step Six: Document Project in Writing

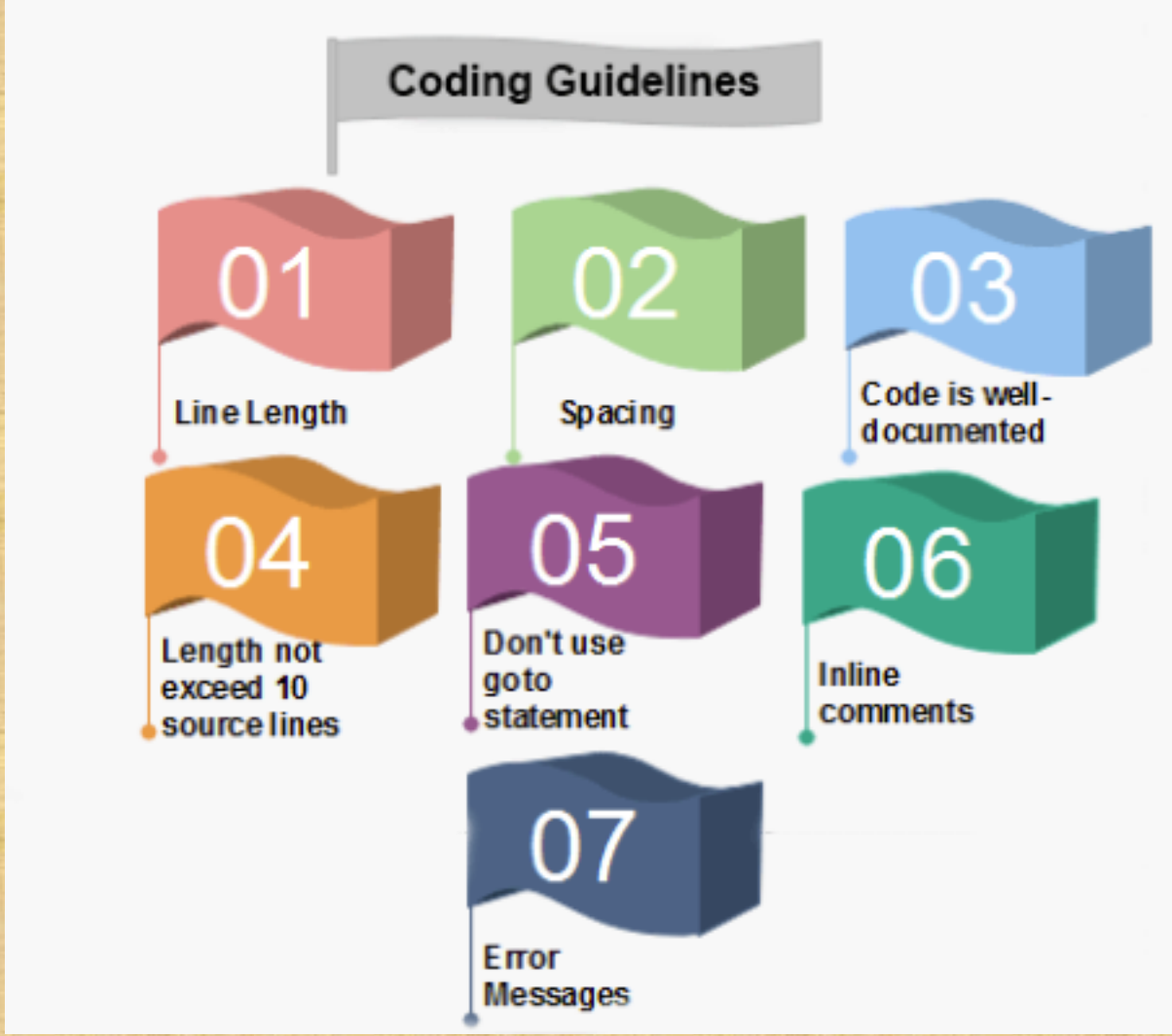
- Tài liệu bao gồm các mô tả bằng hình ảnh và văn bản của phần mềm trong đó chứa các mô tả phần lập trình và các mô tả hoặc hướng dẫn bên ngoài.
- Tài liệu là cần thiết vì dù sớm hay muộn, chương trình sẽ cần được bảo trì (sửa lỗi, thêm tính năng mới, xử lý các vấn đề chưa từng nghĩ tới, v.v.) Điều này KHÔNG thể thực hiện được nếu không có tài liệu.
- Tài liệu có thể bao gồm sách, hướng dẫn sử dụng và các mục tiêu của phần mềm.

5. Quy ước viết mã

Hướng dẫn viết mã

- Nguyên tắc viết mã cung cấp cho lập trình viên một tập hợp các phương pháp
- làm cho các chương trình dễ đọc và bảo trì.

Hướng dẫn viết mã



Hướng dẫn viết mã

1. Line Length: It is considered a good practice to keep the length of source code lines at or below 80 characters. Lines longer than this may not be visible properly on some terminals and tools. Some printers will truncate lines longer than 80 columns.

2. Spacing: The appropriate use of spaces within a line of code can improve readability. Example:

```
Bad: cost=price+(price*sales_tax)
fprintf(stdout ,"The total cost is %5.2f\n",cost);
Better: cost = price + ( price * sales_tax )
fprintf (stdout,"The total cost is %5.2f\n",cost);
```

3. The code should be well-documented: As a rule of thumb, there must be at least one comment line on the average for every three-source line.

Hướng dẫn viết mã

4. The length of any function should not exceed 10 source lines: A very lengthy function is generally very difficult to understand as it possibly carries out many various functions. For the same reason, lengthy functions are possible to have a disproportionately larger number of bugs.

5. Do not use goto statements: Use of goto statements makes a program unstructured and very tough to understand.

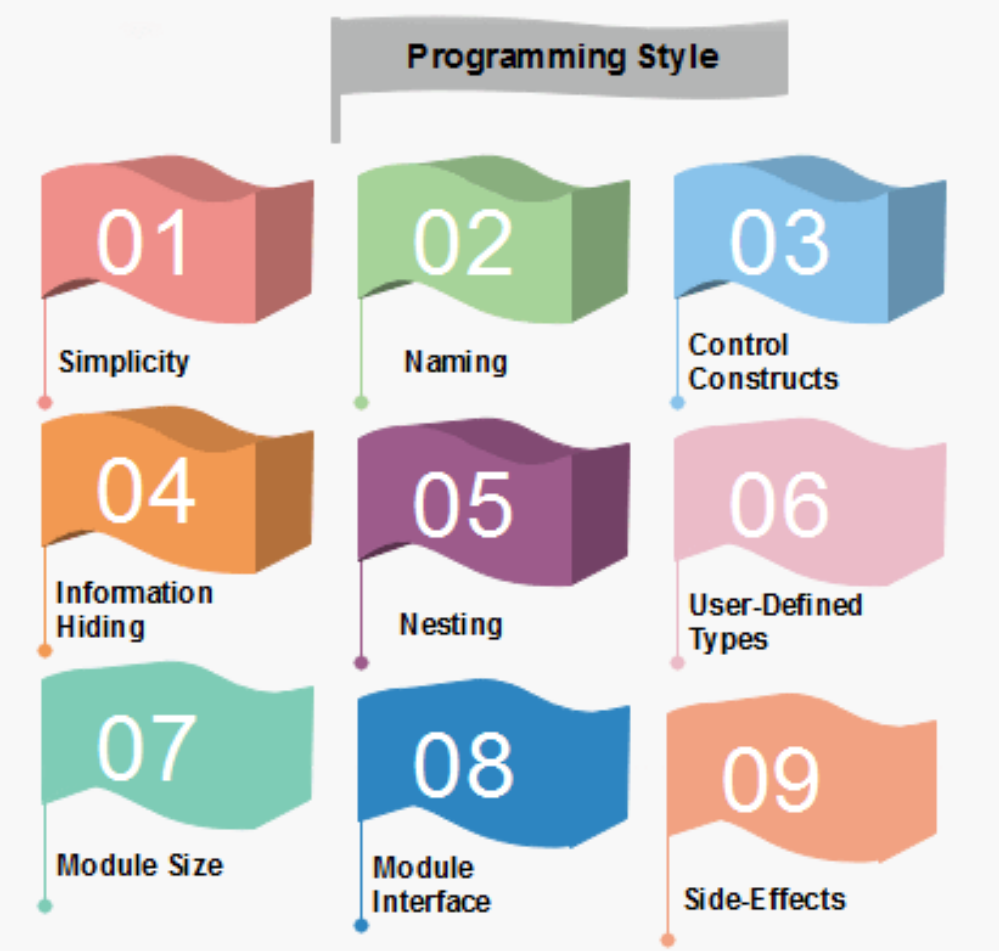
6. Inline Comments: Inline comments promote readability.

7. Error Messages: Error handling is an essential aspect of computer programming. This does not only include adding the necessary logic to test for and handle errors but also involves making error messages meaningful.

Phong cách lập trình

- Các kỹ thuật được sử dụng để viết mã nguồn cho một chương trình. Hầu hết các phong cách lập trình được thiết kế để giúp lập trình viên nhanh chóng đọc và hiểu chương trình cũng như tránh mắc lỗi.
- Một phong cách viết mã tốt có thể khắc phục nhiều khiếm khuyết của ngôn ngữ lập trình.
- Mục tiêu của phong cách lập trình tốt là cung cấp mã đơn giản và dễ hiểu.
- Phong cách lập trình được sử dụng trong một chương trình khác nhau có thể bắt nguồn từ các tiêu chuẩn mã hóa hoặc quy ước mã của một công ty hoặc tổ chức máy tính khác, cũng như sở thích của lập trình viên thực tế.

Phong cách lập trình



Phong cách lập trình

- **1. Clarity and simplicity of Expression:** The programs should be designed in such a manner so that the objectives of the program is clear.
- **2. Naming:** In a program, you are required to name the module, processes, and variable, and so on. Care should be taken that the naming style should not be cryptic and non-representative.
- **For Example:** a = 3.14 * r * r
  area of circle = 3.14 * radius * radius;
- **3. Control Constructs:** It is desirable that as much as a possible single entry and single exit constructs used.
- **4. Information hiding:** The information secure in the data structures should be hidden from the rest of the system where possible. Information hiding can decrease the coupling between modules and make the system more maintainable.

Phong cách lập trình

- **5. Nesting:** Deep nesting of loops and conditions greatly harm the static and dynamic behavior of a program. It also becomes difficult to understand the program logic, so it is desirable to avoid deep nesting.
- **6. User-defined types:** Make heavy use of user-defined data types like enum, class, structure, and union. These data types make your program code easy to write and easy to understand.
- **7. Module size:** The module size should be uniform. The size of the module should not be too big or too small. If the module size is too large, it is not generally functionally cohesive. If the module size is too small, it leads to unnecessary overheads.
- **8. Module Interface:** A module with a complex interface should be carefully examined.
- **9. Side-effects:** When a module is invoked, it sometimes has a side effect of modifying the program state. Such side-effect should be avoided where as possible.

NHẬP MÔN
CÔNG NGHỆ PHẦN MỀM
(INTRODUCTION TO SOFTWARE ENGINEERING)

MỘT SỐ CHỦ ĐỀ KHÁC

- Ước lượng chi phí phần mềm (SE Cost Estimation)
  1. Năng suất (Productivity)
  2. Các kỹ thuật ước lượng (Estimation Techniques)
  3. Mô hình chi phí thuật toán (Algorithmic Cost Model)
  4. Nhân lực và thời gian dự án (Project duration and staffing)
- Quản lý chất lượng (Quality Management)
- Cải tiến quy trình (Process Improvement)
- Khác

1. Năng suất (Productivity)

- Năng suất là số đơn vị đầu ra trên số giờ làm việc
- Trong SE, năng suất có thể ước lượng bởi một số thuộc tính chia cho tổng số nỗ lực để phát triển:
  - Số đo kích thước (thí dụ số dòng lệnh)
  - Số đo chức năng (số chức năng tạo ra trên 1 khoảng thời gian)

2. Các kỹ thuật ước lượng (Estimation Techniques)

- Mô hình chi phí thuật toán: sử dụng các thông tin có tính lịch sử (thường là kích thước)
  - Ý kiến chuyên gia
  - Đánh giá tương tự: chỉ áp dụng khi có nhiều dự án trong cùng một lĩnh vực
  - Luật Parkinson: chi phí phụ thuộc thời gian và số nhân công
  - Giá để thắng thầu: phụ thuộc khả năng KH

3. Mô hình chi phí thuật toán (Algorithmic Cost Model)

- Nguyên tắc: Dùng một phương trình toán học để dự đoán (Kitchenham 1990a) dạng:

  Cố gắng = C x PMs x M với:

  - C là độ phức tạp
  - PM là số đo năng suất
  - M là hệ số phụ thuộc và quá trình, năng suất
  - s được chọn gần với 1, phản ánh độ gia tăng của yêu cầu với các dự án lớn
- Chú ý:
  - Rất khó dự đoán PM vào giai đoạn đầu
  - Việc dự đoán C và M là khách quan và có thể thay đổi từ người này sang người khác.

a. Mô hình COCOMO (Boehm 1981)

- Mô hình COCOMO tuân theo PT trên, với các lựa chọn sau:
  - Đơn giản: PM = 2,4 (KDSI)1,05 x M
  - Khiêm tốn: PM = 3,0 (KDSI)1,12 x M
  - Lồng nhau: PM = 3,6 (KDSI)1,20 x M
    - với KDSI (kilo delivered source instructions) là số lệnh nguồn theo đơn vị nghìn

b. Mô hình định cỡ (calibrate model)

- Sử dụng một mô hình ước đoán có hiệu quả, do vậy cần có 1 CSDL về phân lịch và các cố gắng của một dự án trọn vẹn.
- Có thể dùng kết hợp với mô hình COCOMO

c. Mô hình chi phí thuật toán trong lập kế hoạch dự án

- Dùng để đánh giá chi phí đầu tư nhằm giảm chi phí
- Có 3 thành phần phải xem xét trong khi tính chi phí DA.
  - Chi phí phần cứng của HT
  - Chi phí phương tiện, thiết bị (máy tính, phần mềm) trong phát triển HT
  - Chi phí của các nỗ lực yêu cầu
- Chi phí phần mềm (Software Cost) được tính:
  - SC = Basic Cost x RELY x TIME x STOR x TOOL x EXP x lương TB 1 người/tháng

    với: STOR là không gian lưu trữ, TIME là thời gian cần thiết, TOOL là công cụ, EXP là kinh nghiệm,

    RELY là độ tin cậy (có thể chọn là 1,2)

4. Nhân lực và thời gian dự án (Project duration and staffing)

- Mô hình COCOMO cũng dự đoán lịch cho một DA trọn vẹn:
  - Dự án đơn giản: TDEV = 2.5 (PM)0.38
  - Dự án trung bình: TDEV = 2.5 (PM)0.35
  - Dự án lồng: TDEV = 2.5 (PM)0.32

  với TDEV là tổng thời gian cần thiết cho một DA

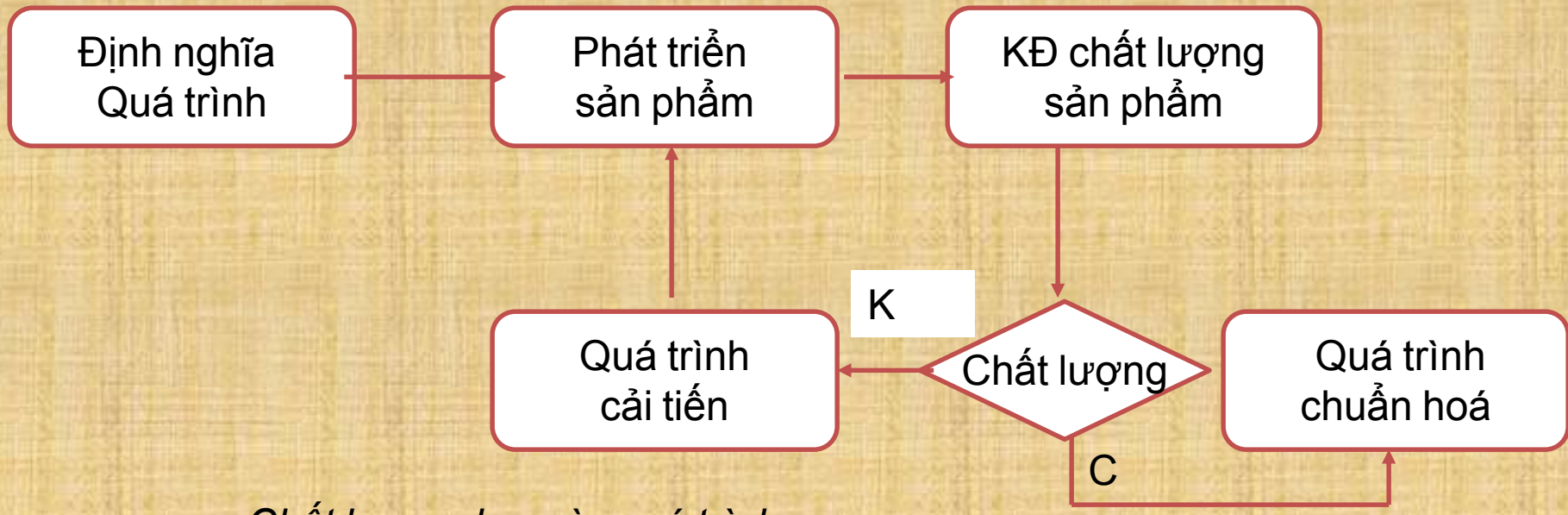
MỘT SỐ CHỦ ĐỀ KHÁC

- Ước lượng chi phí phần mềm (SE Cost Estimation)
- Quản lý chất lượng (Quality Management)
  1. Đảm bảo chất lượng quá trình
  2. Xem xét lại chất lượng
  3. Các chuẩn phần mềm
  4. Các chuẩn tài liệu
  5. Độ đo phần mềm
  6. Độ đo chất lượng sản phẩm
- Cải tiến quy trình (Process Improvement)
- Khác

1. Đảm bảo chất lượng quy trình

- Đảm bảo chất lượng quy trình là một khái niệm đa chiều. chưa có định nghĩa rõ ràng. Nhìn chung khái niệm này có thể xem như là phát triển SP phải đáp ứng được đặc tả của nó (Crossby, 1979)
  - Đặc tả phải hướng về đặc trưng SP mà KH muốn
  - Chúng ta không biết đặc tả thế nào về chất lượng
  - Đặc tả phần mềm luôn luôn không đầy đủ
- Quản lý chất lượng là đáp ứng 3 loại hoạt động sau:
  - Đảm bảo chất lượng
  - Kế hoạch chất lượng: chọn thủ tục tương ứng, chuẩn và kích thước
  - Điều khiển chất lượng: các thủ tục và chuẩn phải được tôn trọng

Đảm bảo chất lượng quy trình(tiếp)



Chất lượng dựa vào quá trình

2. Xem xét lại chất lượng

- Là phương pháp chính để khẳng định chất lượng của quá trình sản xuất
- 3 kiểu xem xét:
  - Thanh tra thiết kế hay chương trình
  - Xem xét tiến triển
  - Xem xét chất lượng

3. Các chuẩn phần mềm

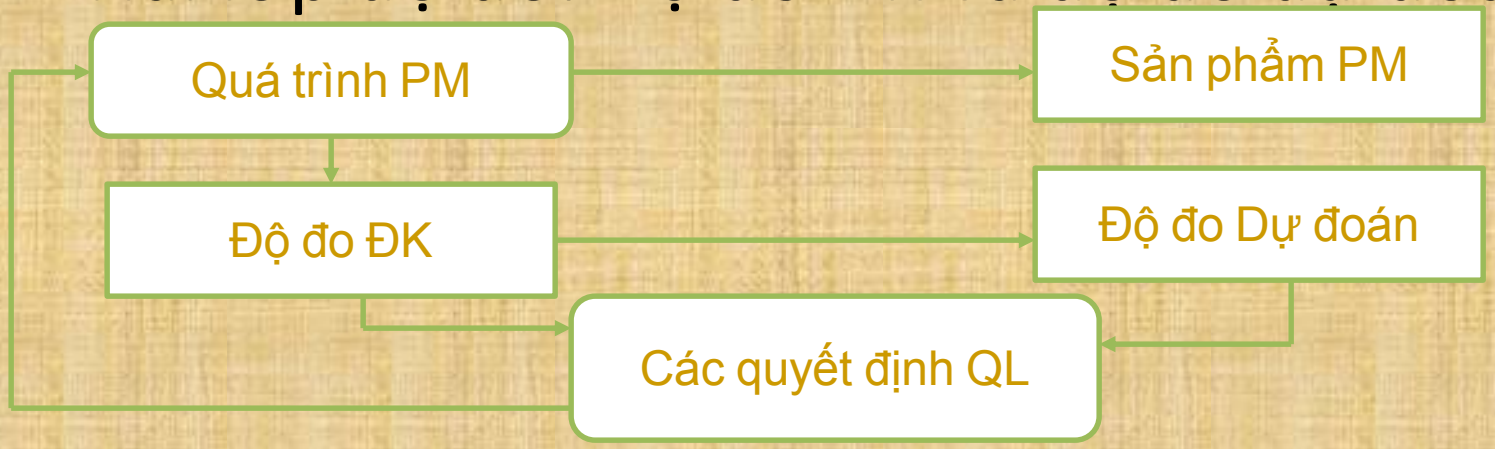
- Vai trò quan trọng của ĐBCLPM là chuẩn hoá các sản phẩm và quá trình
- Tầm quan trọng:
  - Cung cấp SP tương ứng và thực tế
  - Cung cấp các framework để cài đặt cá quá trình ĐBCL
  - Đảm bảo tính liên tục: công việc thực hiện bởi 1 người có thể thực hiện tiếp bởi người khác

4. Các chuẩn tài liệu

- Tài liệu là 1 phần quan trọng trong SE để theo dõi, để hiểu và để làm
- 3 kiểu chuẩn tài liệu:
  - Các chuẩn của quá trình lập tài liệu: Qui định chuẩn khi tạo tài liệu
  - Chuẩn TL: Chuẩn để quản trị chính TL đó
  - Chuẩn trao đổi TL: Dùng trong trao đổi qua E-mail, copy hay lưu trữ trong CSDL

5. Độ đo phần mềm (Software Metric)

- Độ đo phần mềm là một kiểu độ đo liên quan đến HT phần mềm, quá trình hay TL, Thí dụ như số dòng lệnh, số thông báo lỗi khi cung cấp SP
- Hai lớp độ đo: Độ đo ĐK và độ đo dự đoán



6. Độ đo chất lượng SP

- Việc biểu diễn, đánh giá độ đo bằng các số liệu hơn là kinh nghiệm
- Độ đo chất lượng thiết kế
  - tính liên kết
  - độ liên kết
  - dễ hiểu
  - thích hợp
- Độ đo chất lượng chương trình
  - chiều dài mã
  - Độ phức tạp
  - Mức lồng điều kiện

MỘT SỐ CHỦ ĐỀ KHÁC

- Ước lượng chi phí phần mềm (SE Cost Estimation)
- Quản lý chất lượng (Quality Management)
- Cải tiến quy trình (Process Improvement)
  1. Chất lượng quy trình và sản phẩm
  2. Mô hình hoá và phân tích quy trình
  3. Độ đo
  4. Mô hình thuần thục khả năng SEI
  5. Phân loại quy trình
- Khác

Mở đầu

- Cải tiến quy trình có nghĩa hiểu quy trình tồn tại và thay đổi quy trình này để nâng cao chất lượng SP hay giảm chi phí & thời gian phát triển
- Không đơn giản là chấp nhận 1 phương pháp hay công cụ đặc biệt nào hay sử dụng 1 mô hình quy trình đã sử đâu đó
- Cải tiến quy trình phải được xem xét như 1 hoạt động đặc biệt trong 1 tổ chức hoặc 1 phần của tổ chức lớn

Sơ đồ khái quát của Quá trình cải tiến quy trình

Phân tích quy trình: xem xét quy trình đã tồn tại, tạo ra mô hình quy trình để lập TL và hiểu quy trình đó
Xác định cải tiến: sử dụng kết quả phân tích để xác định chất lượng, lập lịch hay chi phí những pha gay cấn

Xác định thay đổi: Thiết lập các thủ tục, phương pháp, công cụ mới và tích hợp với các cái đã tồn tại
Đào tạo: không đào tạo quy trình sẽ thất bại
Hiệu chỉnh thay đổi: các thay đổi có tác dụng ngay với HT

1. Chất lượng quy trình và sản phẩm

- Xem chương trước

2. Mô hình hoá và phân tích quy trình

- Vai trò: nghiên cứu các quy trình đang tồn tại và phát triển mô hình trừu tượng cho các quy trình này (thâu tóm các đặc trưng)
- Phân tích là nghiên cứu để hiểu mối liên quan giữa các phần của quy trình. Điểm xuất phát là mô hình hình thức đã sử dụng
- Kỹ thuật:
  - Hỏi và phỏng vấn
  - Kỹ thuật Ethnographic: dùng để hiểu bản chất của phát triển phần mềm như các hoạt động của con người

Mô hình hoá (tiếp)

- Các ký pháp dùng trong mô hình:
  - Activity (hoạt động): biểu diễn bởi hình chữ nhật tròn
  - Process (quá trình): tập các hoạt động, biểu diễn bởi hình chữ nhật tròn có bóng mờ
  - Deliverable (phân phối): biểu diễn bởi 1 hình chữ nhật có bóng mờ. Nó là đầu ra của 1 hoạt động
  - Condition (điều kiện): biểu diễn bởi 1 hình chữ nhật. Nó là tiền hay hậu điều kiện
  - Role (vai trò): biểu diễn bởi hình tròn
  - Exception (Ngoại lệ): Hộp bao kép. Việc thay đổi do một sự kiện nào đó
  - Communication (Giao tiếp): Biểu diễn trao đổi thông tin giữa con người với nhau hay với HT

3. Độ đo quy trình

- Độ đo của 1 quy trình là các dữ liệu định lượng về quy trình phần mềm (Tập các độ đo là chủ yếu cho quá trình cải tiến quy trình –Humphey, 1989).
- Phân loại:
  - Thời gian để thực hiện 1 quy trình đặc biệt
  - Tài nguyên yêu cầu cho 1 quy trình đặc biệt
  - Số các biến cố
- Khó khăn: Cái nào là cần định lượng đo đếm. Tuy nhiên có thể xem: mục đích (Goals, Câu hỏi, Độ đo)

4. Mô hình thuần thục khả năng (của SEI)

- Viện CNPM (SEI) Carnegie-Melon-University đề xuất. Mô hình SEI phân quá trình phần mềm thành 5 mức khác nhau:
  - Mức khởi đầu: 1 tổ chức không quản lý thực sự các thủ tục hay DA. Phần mềm có thể phát triển song không thể dự đoán trước (ngân sách, thời gian, . . .)
  - Mức lặp: 1 tổ chức có thể có quản lý hình thức về đảm bảo chất lượng, các thủ tục điều khiển cấu hình. Tổ chức có thể lặp lại các DA cùng kiểu
  - Mức có định nghĩa: ở mức này, một tổ chức có định nghĩa các qua trình của mình mà như vậy có 1 cơ sở cho quá trình cải tiến chất lượng. Các thủ tục hình thức đảm bảo rằng các quá trình đã định là sẽ được tuân thủ

Mô hình thuần thục khả năng SEI (tiếp)

- Mức được quản trị: 1 tổ chức đã định nghĩa các quá trình và 1 CT để thu thập dữ liệu về chất lượng. Số đo quá trình và thủ rục được sưu tập cho quá các hoạt động của quá trình cải tiến
- Mức tối ưu: Đã thoả thuận tiếp tục quá trình cải tiến. Quá trình này có ngân sách và kế hoạch để thực hiện và là phần tích hợp của quá trình tổ chức

5. Phân loại quy trình

- Việc phân loại độ chín của các quy trình như trên thường áp dụng cho các DA lớn
- Phân loại:
  - Quy trình không hình thức : các quá trình mà mô hình không định nghĩa 1 cách chặt chẽ
  - Quy trình được quản lý: mô hình quá trình được định nghĩa (định hướng)
  - Quy trình có phương pháp: một số phương pháp phát triển đã được định nghĩa
  - Quy trình cải tiến:

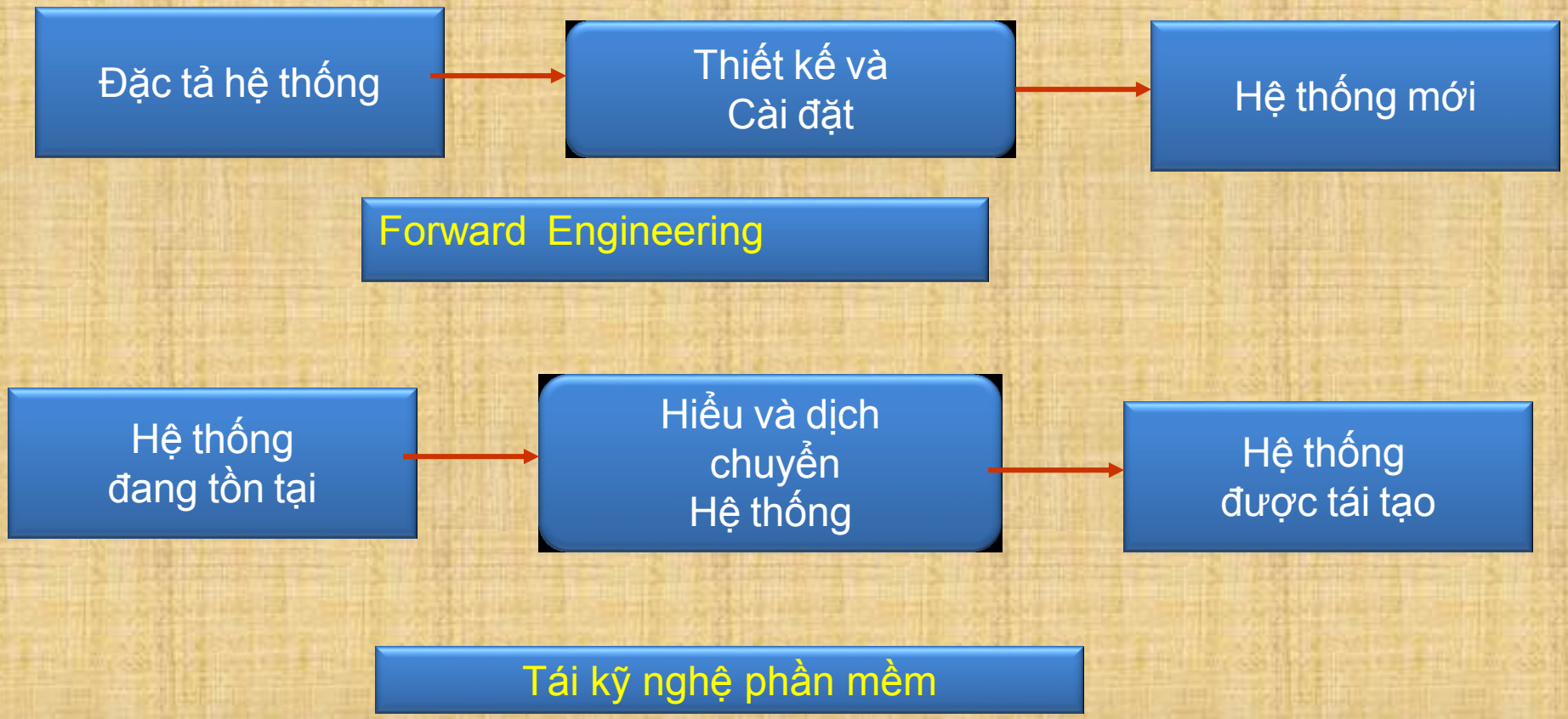
MỘT SỐ CHỦ ĐỀ KHÁC

- Ước lượng chi phí phần mềm (SE Cost Estimation)
- Quản lý chất lượng (Quality Management)
- Cải tiến quy trình (Process Improvement)
- Khác
  1. Phương pháp hình thức (Formal methods)
  2. Công nghệ học phần mềm phòng sạch (Cleanroom SE)
  3. CNHPM hướng thành phần (CBSE)
  4. CNHPM khách/chủ (Client/Server SE)
  5. Kỹ nghệ Web (Web Engineering)
  6. Tái kỹ nghệ (Re-engineering)
  7. CNHPM dựa trên máy tính (CASE)
  8. (Chi tiết xem trong các tài liệu)

Tái kỹ nghệ phần mềm (Software Re-engineering)

- Dịch chuyển mã nguồn
- Cấu trúc lại CT
- Tái lập DL
- Kỹ nghệ ngược

Phát triển và tái kỹ nghệ



- Tập đề thi tự luận
- Tập hợp về bộ câu hỏi trắc nghiệm
- Tập hợp các projects.
- Tìm hiểu, biên tập lại (thầy TrungTT).

Một số chủ đề nâng cao

- Ước lượng chi phí phần mềm (SE Cost Estimation)
- Quản lý chất lượng (Quality Management)
- Cải tiến quy trình (Process Improvement)
- Khác

Ước lượng chi phí phần mềm (SE Cost Estimation)

1. Năng suất (Productivity)
2. Các kỹ thuật ước lượng (Estimation Techniques)
3. Mô hình chi phí thuật toán (Algorithmic Cost Model)
4. Nhân lực và thời gian dự án (Project duration and staffing)

Quản lý chất lượng (Quality Management)

1. Đảm bảo chất lượng quá trình
2. Xem xét lại chất lượng
3. Các chuẩn phần mềm
4. Các chuẩn tài liệu
5. Độ đo phần mềm
6. Độ đo chất lượng sản phẩm

Cải tiến quy trình (Process Improvement)

1. Chất lượng quy trình và sản phẩm
2. Mô hình hoá và phân tích quy trình
3. Độ đo
4. Mô hình thuần thục khả năng SEI
5. Phân loại quy trình

Các chủ đề khác

1. Phương pháp hình thức (Formal methods)
2. Công nghệ học phần mềm phòng sạch (Cleanroom SE)
3. CNHPM hướng thành phần (CBSE)
4. CNHPM khách/chủ (Client/Server SE)
5. Kỹ nghệ Web (Web Engineering)
6. Tái kỹ nghệ (Re-engineering)
7. CNHPM dựa trên máy tính (CASE)
8. (Chi tiết xem trong các tài liệu)

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Nhập môn Công nghệ phần mềm
Introduction to Software Engineering
(IT3180)

Tổng kết và ôn tập

Nội dung chính của môn học

- Mô hình vòng đời phần mềm
- Quy trình phát triển phần mềm
- Giới thiệu tổng quan về các phương pháp luận và các kĩ thuật trong xây dựng phần mềm
- Giới thiệu chung quá trình quản lý dự án phần mềm, đảm bảo chất lượng phần mềm
- Các xu hướng trong nghiên cứu về công nghệ phần mềm

Các chủ đề kiến thức môn học

STT	Nội dung
1	Giới thiệu môn học Chương 1: Tổng quan về Công nghệ phần mềm 1.1 Phần mềm là gì? 1.2 Phân loại phần mềm 1.3 Công nghệ phần mềm là gì? 1.4 Các vấn đề trong Công nghệ phần mềm
2	Chương 2: Vòng đời phần mềm 2.1 Hệ thống vs Phần mềm 2.2 Vòng đời hệ thống/phần mềm 2.3 Quy trình phát triển phần mềm 2.4 Các mô hình quy trình phần mềm: Thác nước, mẫu thử, tăng dần, nhanh, xoắn ốc Ví dụ và bài tập
3	2.5. So sánh các mô hình quy trình phần mềm 2.6. Thảo luận nhóm và lựa chọn mô hình quy trình phù hợp

Các chủ đề kiến thức môn học

4	Chương 3: Phương pháp Agile 3.1 Khái niệm 3.2 Các nguyên lý cơ bản 3.2 Ưu, nhược điểm của phương pháp Agile 3.3 Extreme Programming 3.4 Scrum 3.5 Các phương pháp Agile khác
5	Chương 4: Quản lý cấu hình phần mềm 4.1 Khái niệm quản lý cấu hình phần mềm 4.2 Quy trình cấu hình phần mềm 4.3 Quản lý phiên bản 4.4 Quản lý thay đổi
6	Chương 5: Kỹ nghệ yêu cầu phần mềm (Requirement Engineering) 5.1 Khái niệm 5.2 Tầm quan trọng của yêu cầu phần mềm 5.3 Yêu cầu chức năng và yêu cầu phi chức năng 5.4 Các hoạt động chính trong kỹ nghệ yêu cầu phần mềm: Thu thập, Phát hiện, Phân tích, Đặc tả, Thẩm định, Quản lý
7	5.5. Quy trình kỹ nghệ yêu cầu phần mềm

Các chủ đề kiến thức môn học

8	Chương 6: Thiết kế phần mềm 6.1 Tổng quan về thiết kế phần mềm 6.2 Các khái niệm trong thiết kế phần mềm 6.3 Tính móc nối (Coupling) và tính kết dính (Cohesion) 6.4 Thiết kế kiến trúc 6.5 Thiết kế chi tiết
9	6.6 Thiết kế giao diện người dùng –Các vấn đề thiết kế –Quy trình thiết kế UI –Phân tích người dùng –Tạo mẫu thử giao diện, mẫu thử tương tác –Đánh giá UI –Các công cụ thiết kế UI
10	Chương 7: Xây dựng phần mềm 7.1 Khái niệm 7.2 Quy trình xây dựng phần mềm 7.3 Quy ước viết mã nguồn 7.4 Tái cấu trúc mã nguồn 7.5 Rà soát mã nguồn

Các chủ đề kiến thức môn học

11	Chương 8: Quản lý chất lượng phần mềm 8.1 Mô hình V&V 8.2 Các thuật ngữ về kiểm thử 8.3 Phương pháp kiểm thử hộp trắng - Khái niệm, Vai trò - Kỹ thuật bao phủ luồng điều khiển 8.4 Phương pháp kiểm thử hộp đen - Khái niệm, Vai trò - Kỹ thuật phân vùng tương đương 8.5 Quản lý chất lượng phần mềm
12	Chương 9: Quản lý dự án phần mềm 9.1 Khái niệm về dự án phần mềm. –Yếu tố con người: Stokeholder/ TeamLeader/ Software Team/ Communication issue –Yếu tố Sản phẩm: Software scope/ Processs/ Project 9.2 Quy trình quản lý dự án phần mềm. –Ước lượng dự án –Lập kế hoạch dự án –Quản lý rủi ro dự án

Đánh giá môn học

- Hình thức: thi trên giấy (tự luận và trắc nghiệm)
  - Không sử dụng tài liệu, thời gian khoảng ~90’
- Trắc nghiệm về các chủ đề:
  - Chu trình phần mềm và các mô hình phát triển phần mềm
  - Phương pháp Agile
  - Phân tích yêu cầu phần mềm và các kĩ thuật lấy yêu cầu phần mềm
  - Thiết kế phần mềm và các kĩ thuật thiết kế
  - Quản lý cấu hình phần mềm
  - Quản lý dự án phần mềm
  - Kiểm thử phần mềm, Bảo trì phần mềm,…
- Tự luận (bài tập hoặc câu hỏi viết):
  - Bài tập xác định yêu cầu phần mềm (xác định tác nhân, usecase, xây dựng biểu đồ usecase, đặc tả usecase, ràng buộc đầu vào,…)
  - Bài tập về phân tích và thiết kế phần mềm (các sơ đồ phân tích (UML: biểu đồ trình tự, giao tiếp, hoạt động), xây dựng và đặc tả sơ đồ lớp thiết kế, thiết kế dữ liệu, thiết kế giao diện màn hình)
  - Bài tập về kiểm thử (hộp đen và hộp trắng)
  - Câu hỏi về Quản lý dự án phần mềm,…

Tài liệu học tập

- Tài liệu chính: slides và bài tập hàng tuần / bài tập lớn môn học mà nhóm đã thực hiện
- Tài liệu tham khảo:
  - [1] I. Sommerville, Software Engineering 10th Edition, Addison Wesley 2017
  - [2] R. Pressman, Software Engineering: A practitioner's approach, 8th Edition, McGraw Hill 2016