Kỹ nghệ phần mềm

Software Engeneering

Đại học Kinh doanh và Công nghệ Hà Nội
Khoa CNTT
GV: Đào Thị Phượng
Email: phuongdt102@gmail.com
Page fb: facebook.com/it.hubt
Phone: 0946.866.817

Bài 3: Tiến trình phần mềm

Nội dung

- Tiến trình và mô hình tiến trình
- Các giai đoạn của tiến trình
- Tiến trình và vấn đề liên quan

TÀI LIỆU THAM KHẢO

1. Nguyễn Văn Vỵ, Nguyễn Việt Hà. *Giáo trình kỹ nghệ phần mềm*. Nhà xuất bản Đại học Quốc gia Hà nội, 2008
2. Grady Booch, James Rumbaugh, Ivar Jacobson. *The Unified Modeling language User Guid.* Addison-Wesley, 1998.
3. M. Ould. *Managing Software Quality and Business Risk*, John Wiley and Sons, 1999.
4. Roger S.Pressman, *Software Engineering, a Practitioner's Approach.* Fifth Edition, McGraw Hill, 2001.
5. Ian Sommerville, *Software Engineering*. Sixth Edition, Addison-Wasley, 2001.
6. Nguyễn Văn Vỵ. *Phân tích thiết kế hệ thống thông tin hiện đại. Hướng cấu trúc và hướng đối tượng,* NXB Thống kê, 2002, Hà Nội.

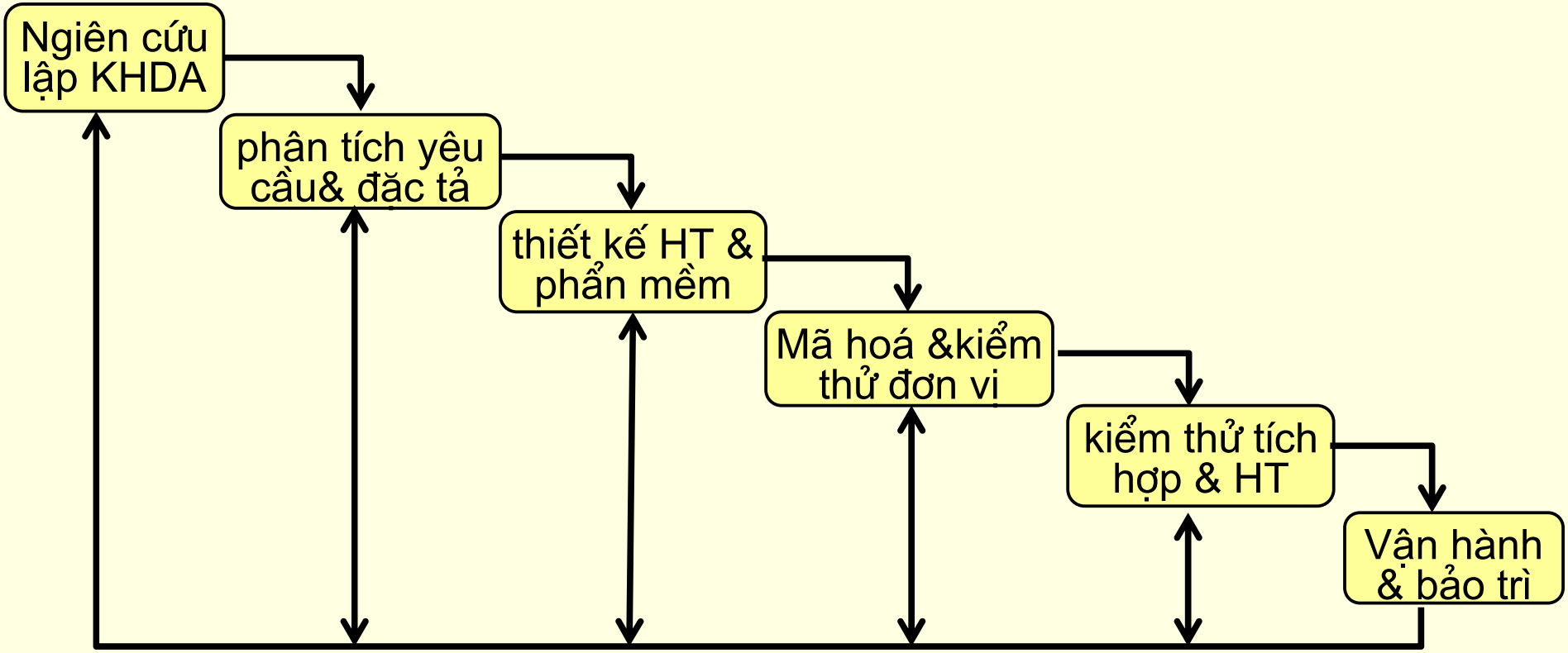
Các loại mô hình tiến trình

5 loại mô hình tiến trình phần mềm tiêu biểu:

- Mô hình thác nước
- Các mô hình phát triển tiến hóa
- Các mô hình phát triển hình thức
- Phát triển dựa trên sử dụng lại
- Khác

Mỗi loại bao gồm một số các mô hình tiến trình.

Mô hình vòng đời truyền thống



Mô hình thác nước – waterfall model

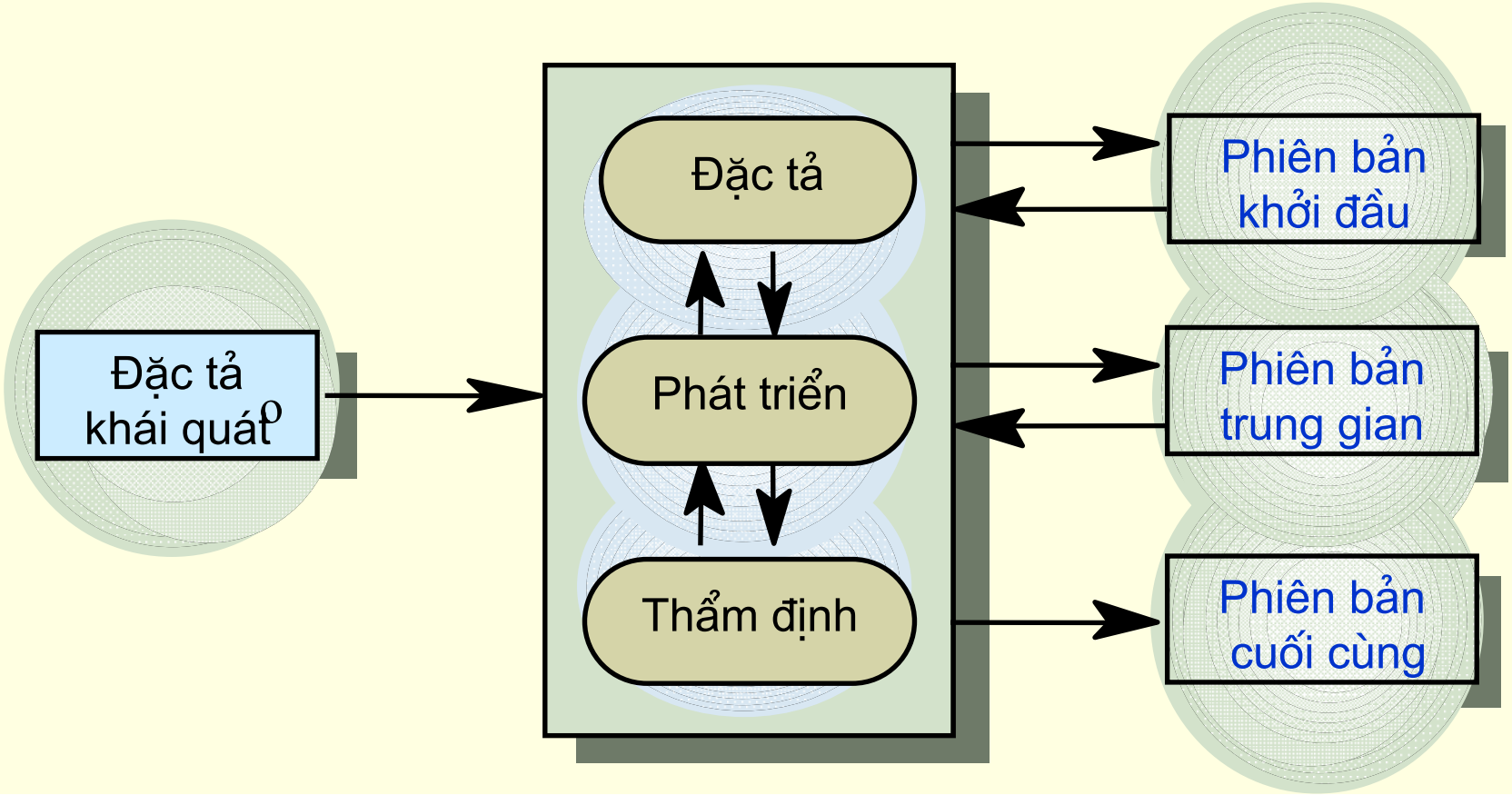
Mô hình thác nước: đặc điểm

- Tách biệt giữa các pha, tiến hành tuần tự
  - Khó tuân thủ tuần tư: dự án lớn thường phải quay lại
  - Khó đáp ứng yêu cầu thường thay đổi của khách
- Chậm có phiên bản thực hiện được
  - đòi hỏi khách hàng phải kiên nhẫn
  - sai sót phát hiện muộn có thể là thảm họa
- Đặc tả kỹ, phân công chuyên trách, hướng tài liệu
  - Tài liệu quá nhiều, tốn sức người, thời gian dài

❖ Có sớm và được sử dụng rộng rãi (tốt > tự nhiên)
❖ Thích hợp khi yêu cầu hiểu tốt, hệ lớn & phức tạp
❖ Bảo trì thuận lợi

Mô hình phát triển tiến hóa

b1. Lược đồ chung nhất



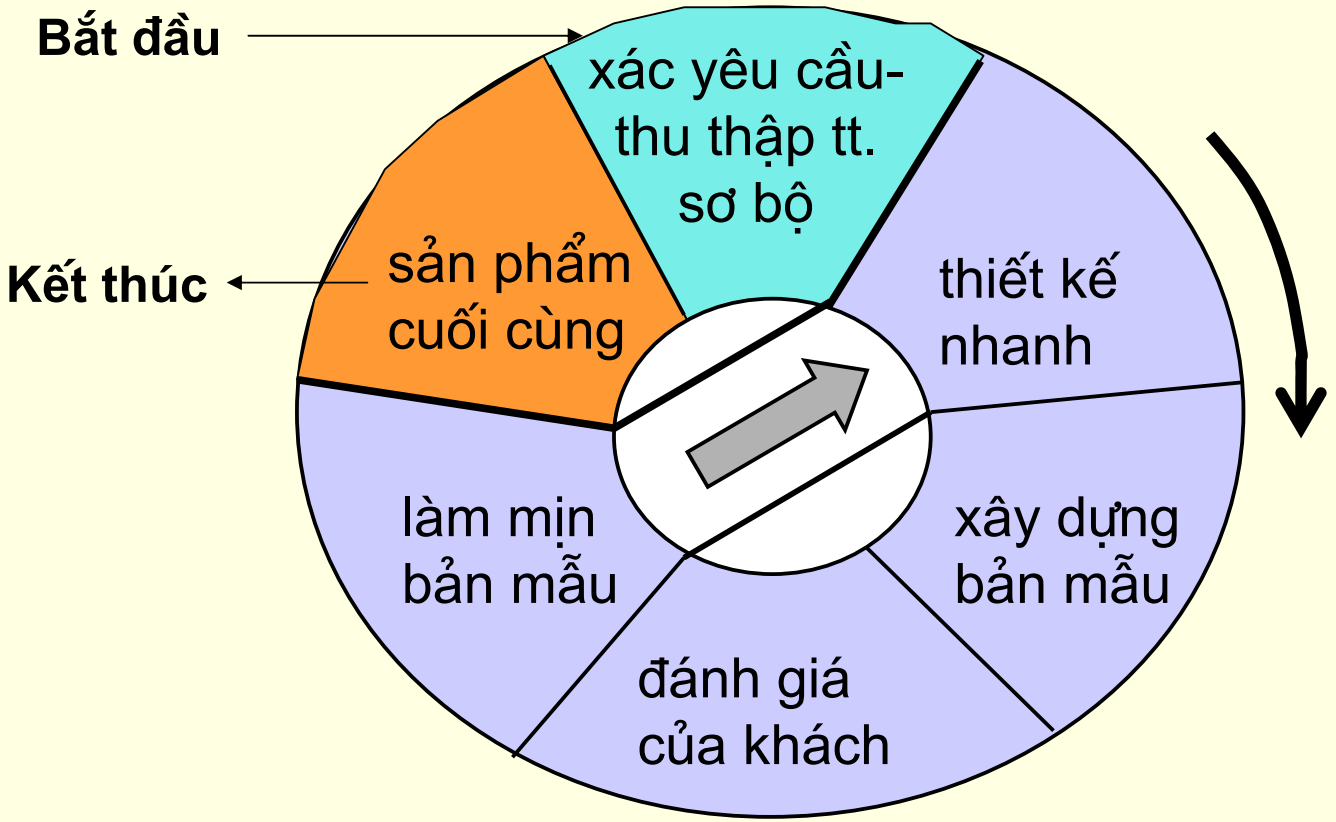
Lược đồ chung nhất

- Phát triển ban đầu
  - Làm việc với khách, đặc tả khái quát hệ thống (bắt đầu với hiểu biết có thể chưa đầy đủ)
- Thực hiện phát triển bằng cách làm mẫu
  - Mục tiêu là để hiểu hệ thống. Bản mẫu ban đầu có thể còn sơ sài.
- Thẩm định phiên bản có được, lặp lại các bước cho đến khi có phiên bản cuối cùng

Lược đồ chung

- Hạn chế
  - Không trực quan
  - Hệ thống thường có cấu trúc nghèo nàn
  - Đòi hỏi có kỹ năng đặc tả (ngôn ngữ làm mẫu)
- Khả năng ứng dụng
  - Cho các hệ tương tác vừa, nhỏ
  - Cho những phần của hệ lớn
  - Hệ có vòng đời ngắn

Mô hình làm bản mẫu

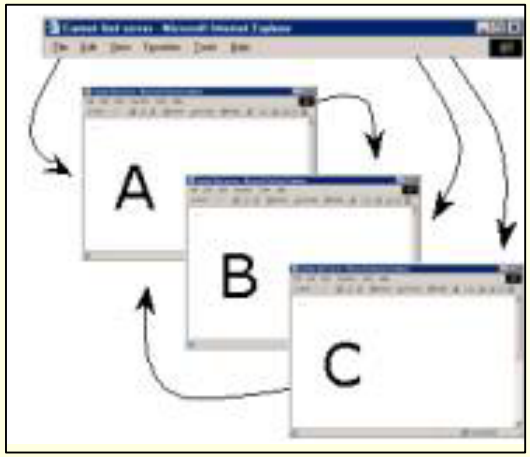


Mô hình làm bản mẫu - *Prototyping model*

Mô hình làm bản mẫu

Loại mẫu:

- mẫu trên giấy
- mẫu mô tả một phần chức năng
- mẫu giao diện
- mẫu hướng tới sản phẩm



Mức độ mẫu:

- mẫu dùng xong bỏ đi (*throw-away approach*)
- mẫu dùng tiếp cho bước sau (*CASE chuyên dụng*)
- mẫu là phần hệ thống vận hành được (*dựa trên thành phần dùng lại)*

Mô hình làm bản mẫu

Nhược điểm:

- tính cấu trúc không cao
- khách hàng ít tin tưởng

Ưu thế:

- nhanh chóng xác định được yêu cầu, tốt
- tạo cơ sở ký kết hợp đồng
- giúp đào tạo huấn luyện người sử dụng

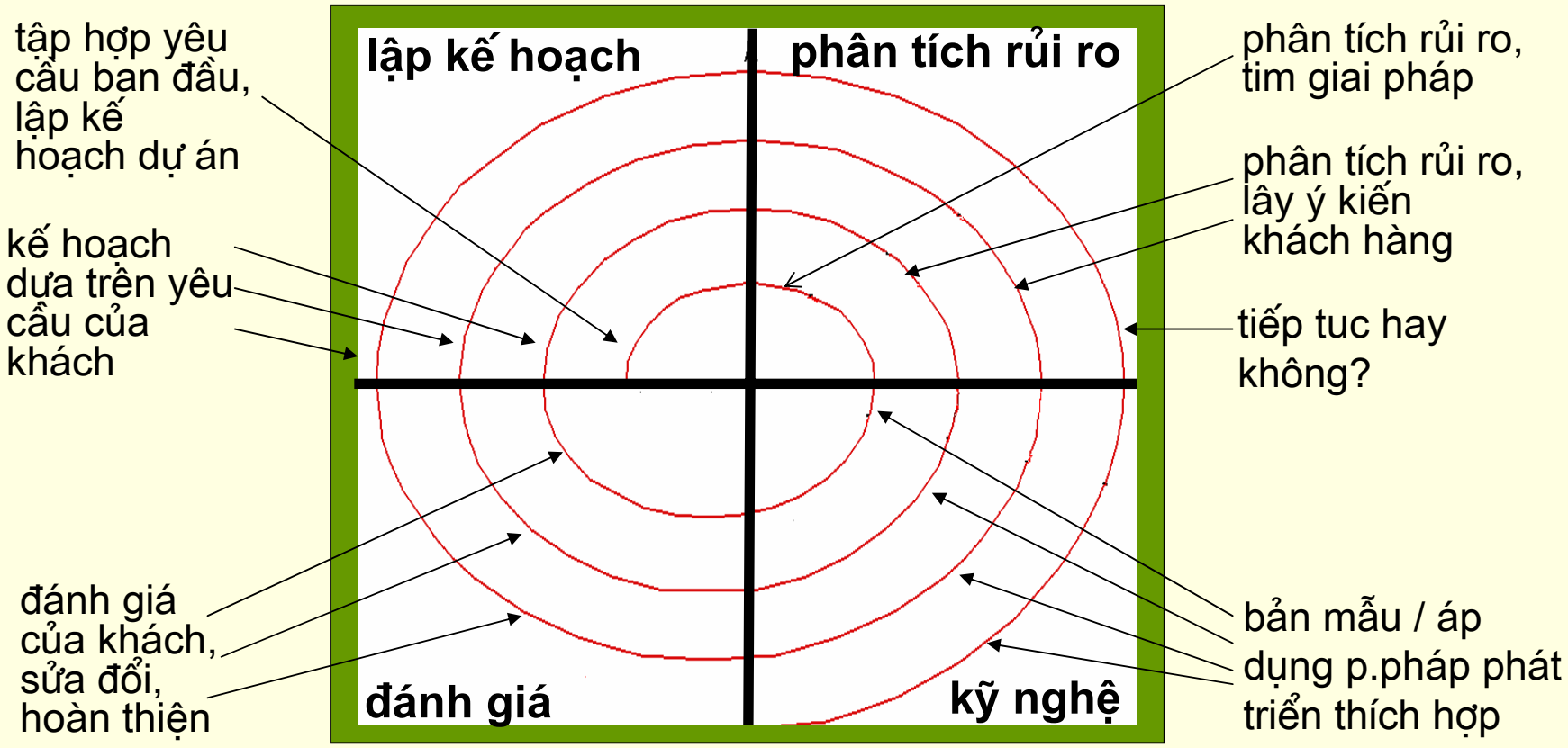
Thích hợp:

- các yêu cầu chưa rõ ràng
- input/output chưa rõ ràng
- khó đánh giá tính hiệu quả thuật toán

Mô hình xoắn ốc (*spiral model*)

- Cải tiến của mô hình tuần tự và làm mẫu
- Thêm phân tích rủi ro
- Là quá trình lặp hướng mở rộng, hoàn thiện dần
  - **Lập kế hoạch**: xác lập vấn đề, tài nguyên, thời hạn.
  - **Phân tích rủi ro**: xem xét mạo hiểm, tìm giải pháp
  - **Kỹ nghệ**: phát triển một phiên bản của phần mềm (*chọn mô hình thích hợp: làm mẫu, thác nước,..)*
  - **Đánh giá của khách**: khách đánh giá phiên bản phát triển; → làm mịn, sửa đổi

Mô hình xoắn ốc



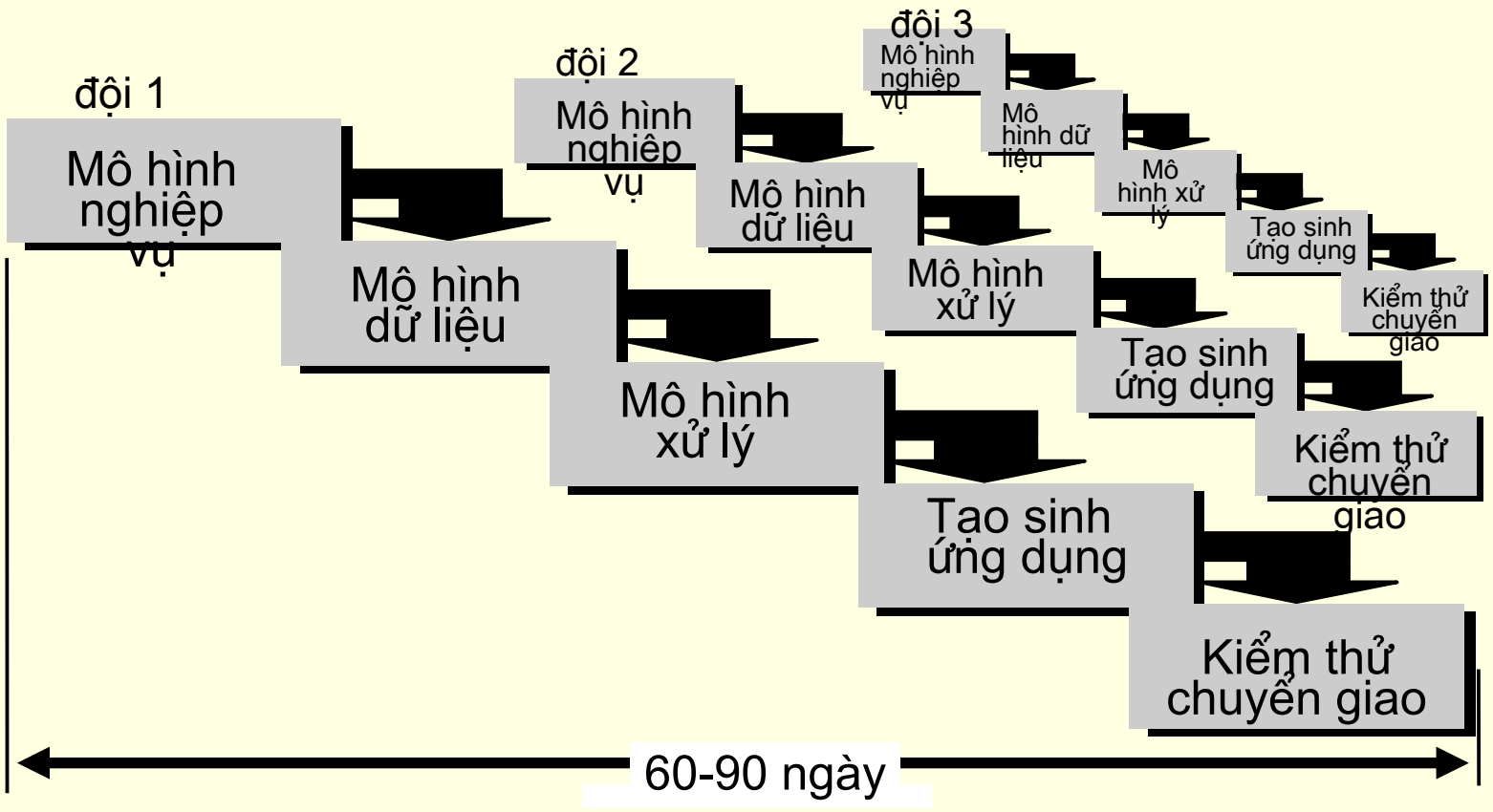
spiral model

Mô hình xoắn ốc: đặc điểm

- Hợp với hệ lớn có thể phân chia phần cốt lõi → thứ yếu
- Có thể kiểm soát rủi ro ở từng mức tiến hóa
- Khó thuyết phục khách là kiểm soát được sự tiến hóa linh hoạt *(đòi hỏi năng lực quản lý, năng lực phân tích rủi ro -> chi phi chuyên gia lớn*)
- Chưa được dùng rộng rãi như mô hình thác nước hoặc làm mẫu

Mô hình phát triển ứng dụng nhanh

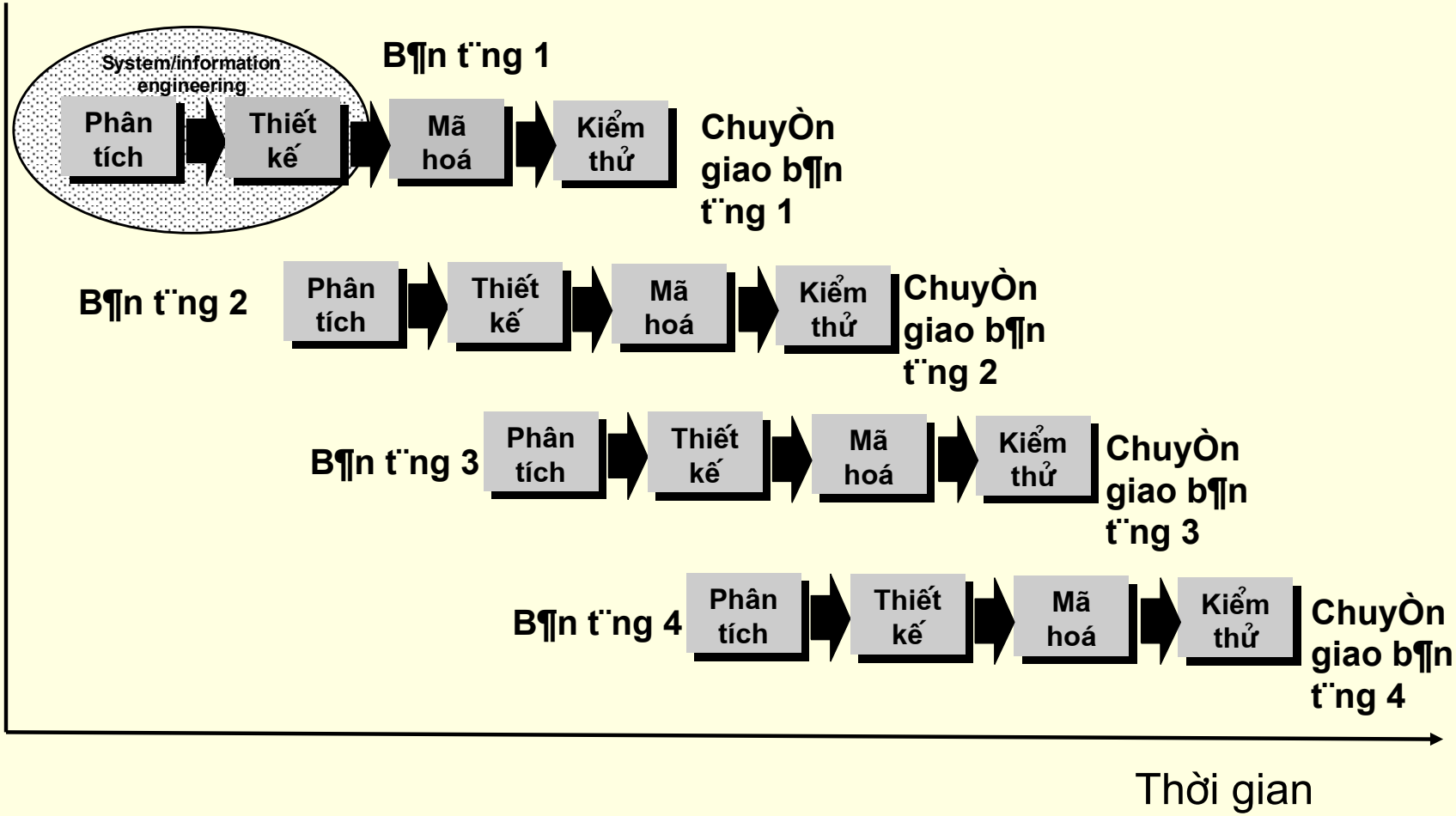
Rapid Application Development- RAD



RAD - đặc điểm

- Hợp với các hệ thống có khả năng môđun hóa cao
  - hướng thành phần, tái sử dụng
  - sử dụng công cụ tự động
- Thời gian phát triển sản phẩm ngắn (60~90 ngày)
- Không phù hợp với sản phẩm:
  - khó phân chia thành các thành phần
  - đòi hỏi hiệu năng cao

Mô hình tăng trưởng (*incremental model*)



Mô hình tăng trưởng

- Chuyển giao dần từng phần của hệ thống
  - Sản phẩm chia thành từng phần tăng theo yêu cầu chức năng
  - Yêu cầu người dùng ưu tiên theo thứ tự phần tăng
- Cho sản phẩm dùng trong thời gian ngắn
  - đáp ứng nhanh yêu cầu của khách
  - chiếm lĩnh thị trường
  - khác với bản mẫu
- Công ty phát triển phải có tiềm lực cao *(công nghệ, tài sản phần mềm)*

Lập trình cực đoan (*Extreme Programming-XP*)

- Cách tiếp cận dựa trên việc phát triển, chuyển giao dần từng phần nhỏ chức năng
- Tạo các ca thử nghiệm trước khi lập trình
  - đòi hỏi phải nắm vững yêu cầu; giao diện trước khi bắt tay vào mã hóa
- Lập trình đội
  - tránh lỗi, nâng cao chất lượng
  - đảm bảo sự tuân theo các chuẩn XP đã đề ra
- Viết lại khi có thể
  - chủ động tấn công lỗi

Kỹ thuật thế hệ thứ 4

Fourth generation technology - 4GT

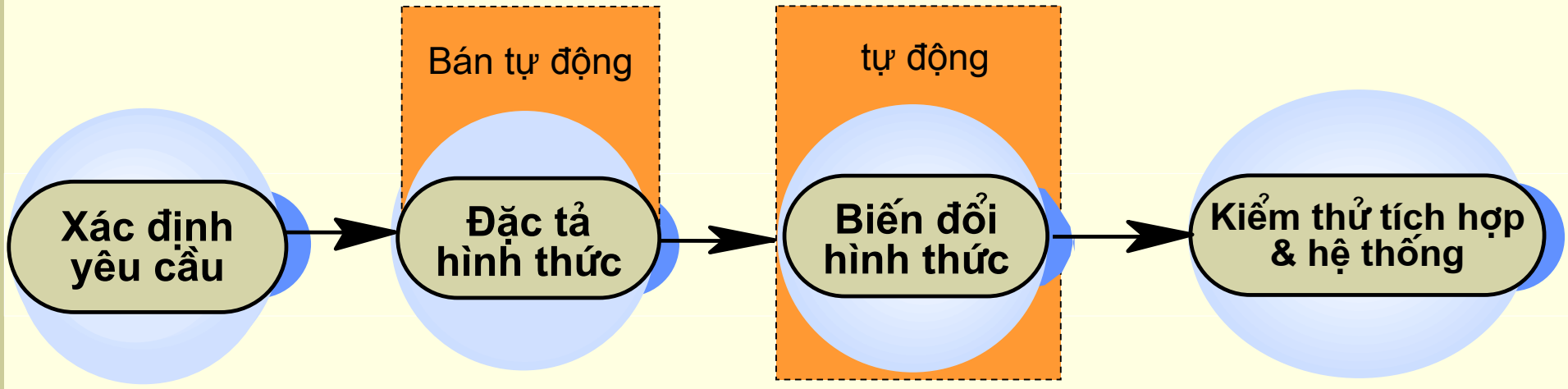
- Các kỹ thuật xác định đặc trưng phần mềm ở mức cao:
  - hướng mục đích (chức năng)
  - tự động sinh mã chương trình theo đặc tả
- Các công cụ/ứng dụng điển hình
  - truy vấn CSDL (SQL)
  - tạo báo cáo, bảng tính
  - sinh giao diện (giao diện Web)

4GT: đặc điểm

- Phân tích/thiết kế vẫn là bước quan trọng
- 4GT chỉ trợ giúp (tự động hóa) việc sinh mã chương trình đối với từng chức năng cụ thể
- Ứng dụng còn hạn chế; không phải mọi 4GTcũng dễ dùng
- Tiết kiệm công sức cho phát triển phần mềm nhỏ
- Không hiệu quả với phần mềm lớn: mã hóa chỉ chiếm một tỷ lệ nhỏ so với phân tích thiết kế

Phát triển hệ thống hình thức hóa

formal systems development

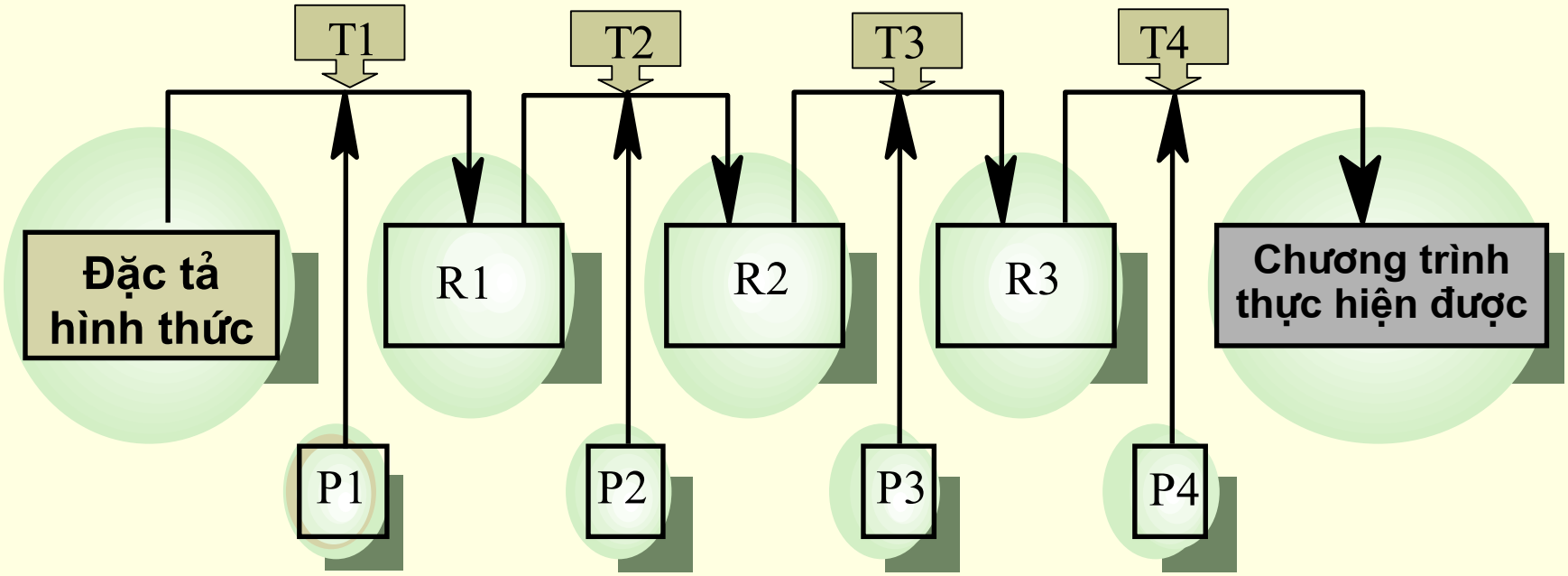


Các bước của tiến trình phát triển

(đặc tả yêu cầu 1 cách hình thức bằng các công cụ toán học trừu tượng)

Biến đổi hình thức

Các phép biến đổi hình thức



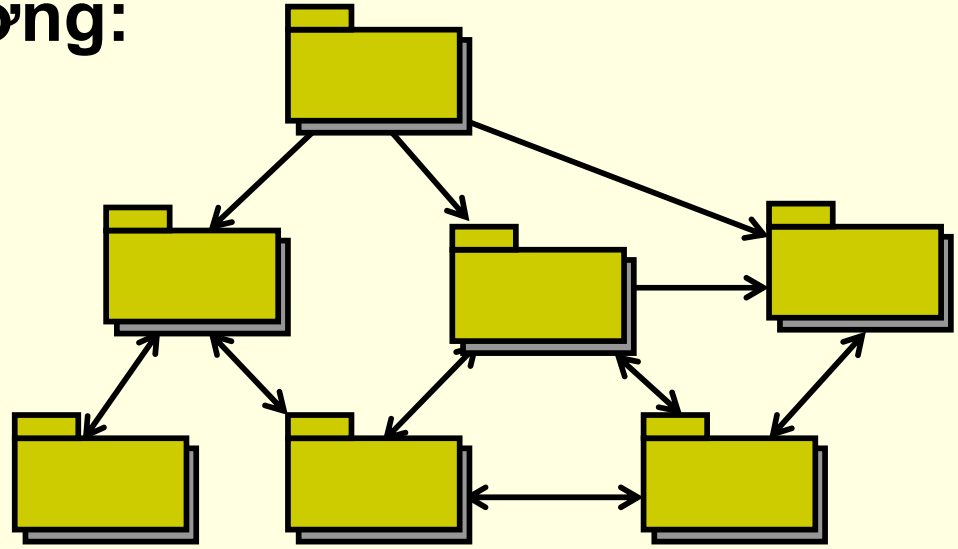
Các chứng minh tính đúng đắn của phép biến đổi

Hạn chế phát triển hình thức hóa

- Hạn chế
  - Cần có kỹ năng đặc tả và sử dụng kỹ thuật tiên tiến
  - Khó đặc tả được mọi khía cạnh của hệ thống, chẳng hạn như giao diện
- Khả năng ứng dụng
  - Những hệ thống quan trọng cần phải đảm bảo độ an toàn, bảo mật trước khi đưa vào sử dụng, xử lý nhiều, tương tác hạn chế.
  - Ít nhà phát triển có thể sử dụng.

Phát triển hướng sử dụng lại

- Hướng sử dụng lại dựa trên nền tảng của phát triển hệ thống hướng đối tượng
- **Ý tưởng hướng đối tượng:**
  - Hệ thống cấu thành từ các đối tượng
  - Đối tượng bao gói cả dữ liệu và xử lý
  - Liên kết với nhau bằng truyền thông



Mô hình cấu trúc của hệ thống

Phát triển hướng đối tượng

- bao gói, che dấu thông tin:

  Do bao gói cả dữ liệu và xử lý nên độc lập tương đối, cho một chức năng xác định, che thông tin với bên ngoài

  ⇨ tác động cục bộ, dễ bảo trì, dễ dùng lại

- tính kế thừa:
  - xây dựng được các lớp cơ sở (chung)
  - khi thêm các chi tiết dùng cho trường hợp cụ thể

  ⇨ nâng cao khả năng dùng lại

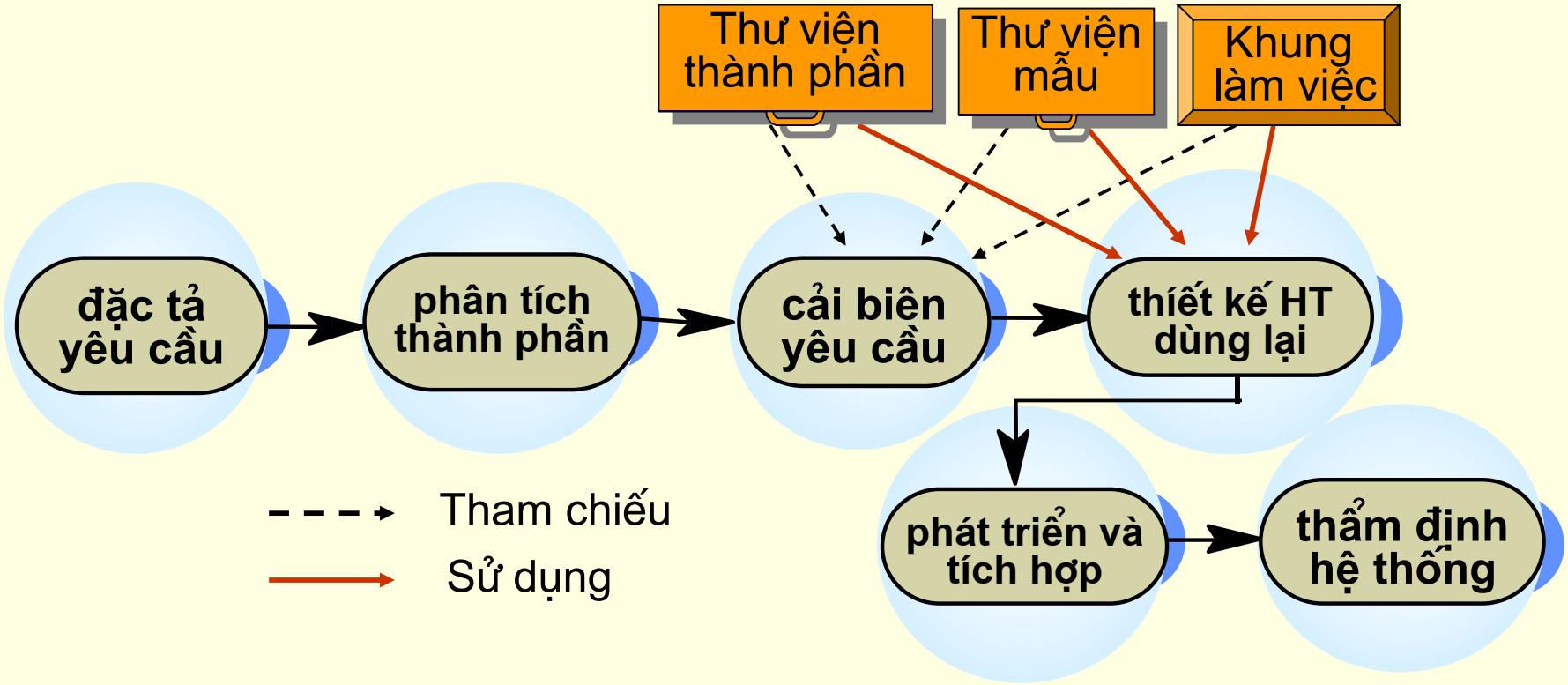
- liên kết tự do, yếu

  ⇨ mở rộng đơn giản, không hạn chế

Các hướng sử dụng lại

- Các hướng sử dụng lại:
  - Định hướng thành phần (*component - mã nguồn*)
  - Định hướng mẫu (*pattern - phân tích, thiết kế*)
  - Phát triển khung làm việc (*framworks: lớp ứng dụng*)
- Các giai đoạn của tiến trình
  - Phân tích hệ thống thành các phần yêu cầu nhỏ
  - Cải biên các yêu cầu hướng *(thành phần, mẫu, khung)*
  - Thiết kế hệ thống hướng tới tài sản sử dụng lại
  - Phát triển và tích hợp
- Quan trọng, cần kinh nghiệm, công cụ còn hạn chế

Tiến trình hướng sử dụng lại



Tiến trình phát triển

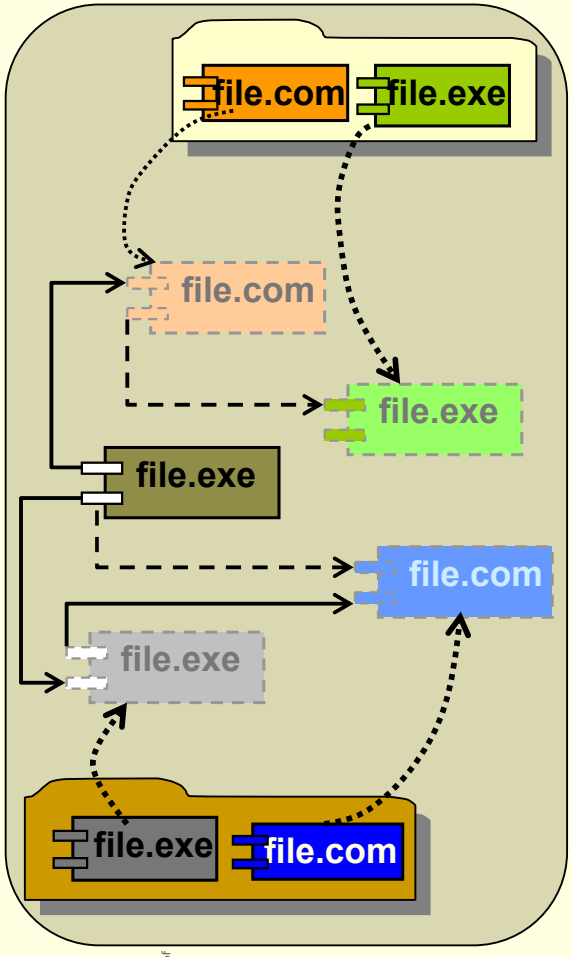
Kỹ nghệ hướng thành phần
Component-based software engineering

Nội dung

- Thành phần là mô đun chức năng mã đóng
- Lắp ráp các thành phần đúng với yêu cầu
- Dùng lại thành phần độc lập với ngôn ngữ
- Thay thế thành phần là động (không cần dịch) không cần đường dẫn, chỉ cần định danh.

Ưu, nhược

- Nhanh, ổn định, hiệu quả
- Cần có các thành phần được mô tả, hiểu về nó, có cách tìm kiếm tốt

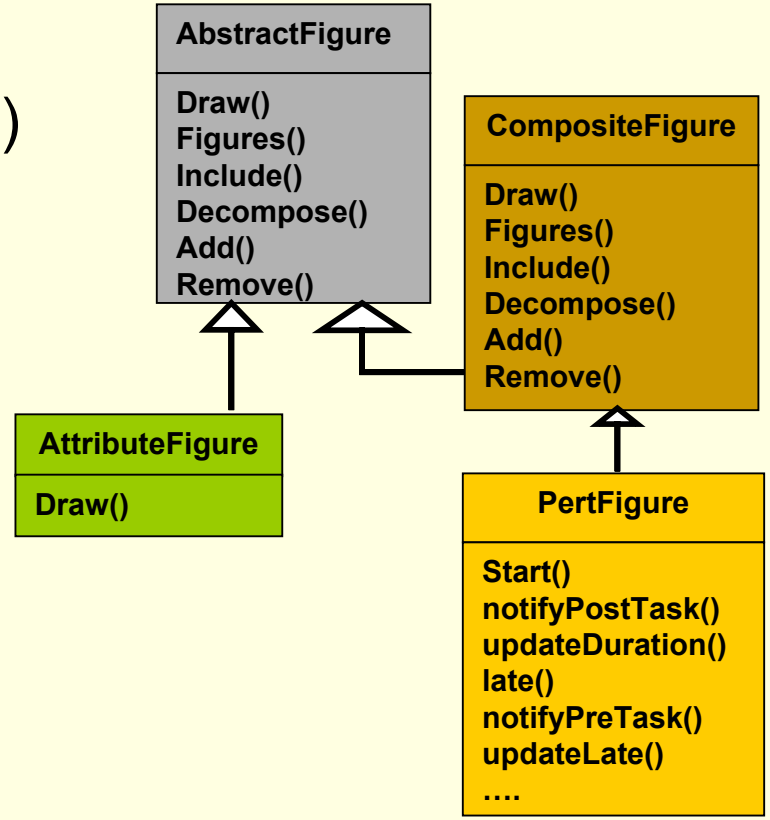


Phân tích thiết kế hướng mẫu

pattern oriented analysis & design - POAD

Nội dung

- Phân tích yêu cầu hướng theo mẫu
- Xem xét các mẫu đã có (từ thư viện)
- Tìm, lựa chon mẫu thích hợp cho phần được phân tích
- Chuyển thiết kế thành chương trình (tự động, bán tự động)



Ưu, nhược

- Bắt đầu ứng dụng rộng rãi
- Cần có khả năng phân tích tốt
- Có hiểu biết thành thạo về mẫu

Phát triển khung làm việc
Framework development

Xây dưng khung làm việc

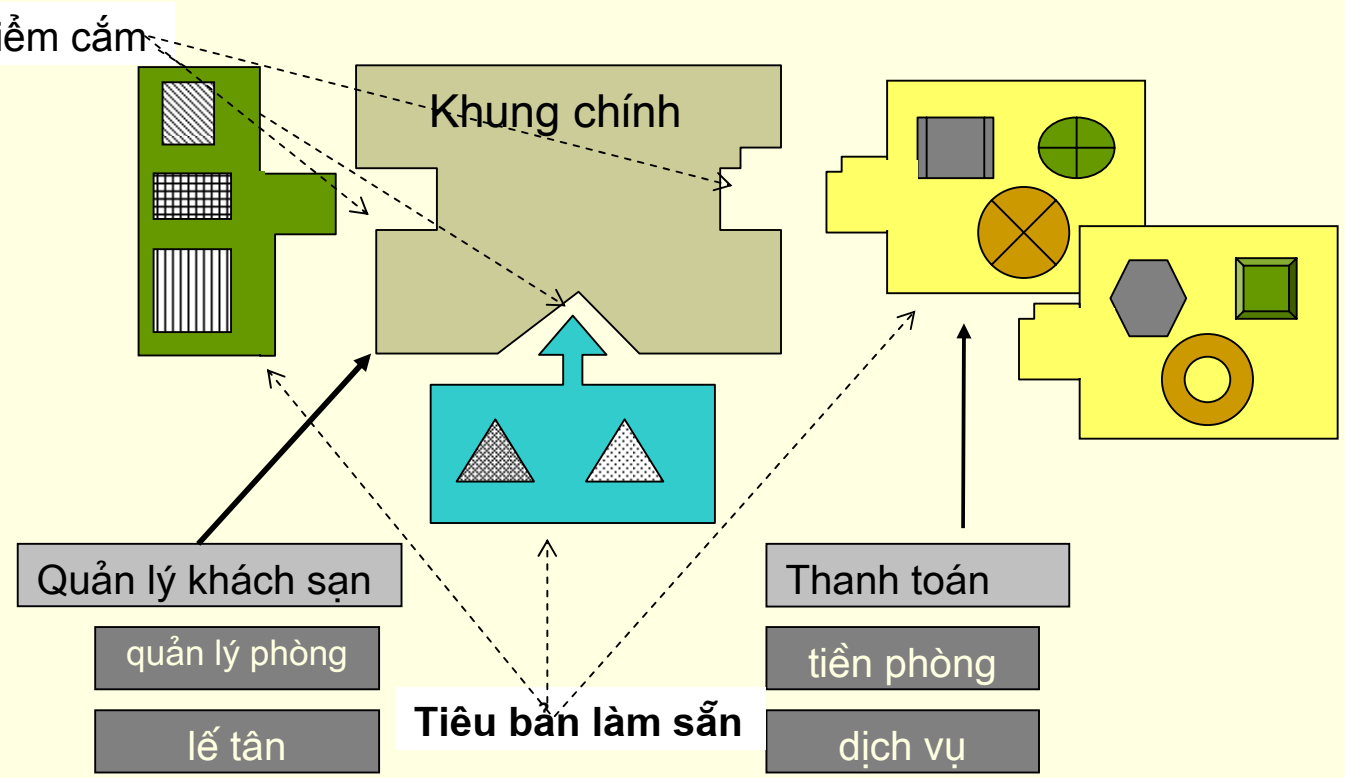
- Xác định lớp bài toán cần giải quyết
- Xây dựng khung bài toán (dựa trên patterns)
- Làm sẵn các tiêu bản mẫu (dùng ngay được)

Triển khai

- Phân tích bài toán theo khung
- Xác định tiêu bản thích hợp
- Lắp ghép hay tìm phần thay thế

Phát triển khung làm việc

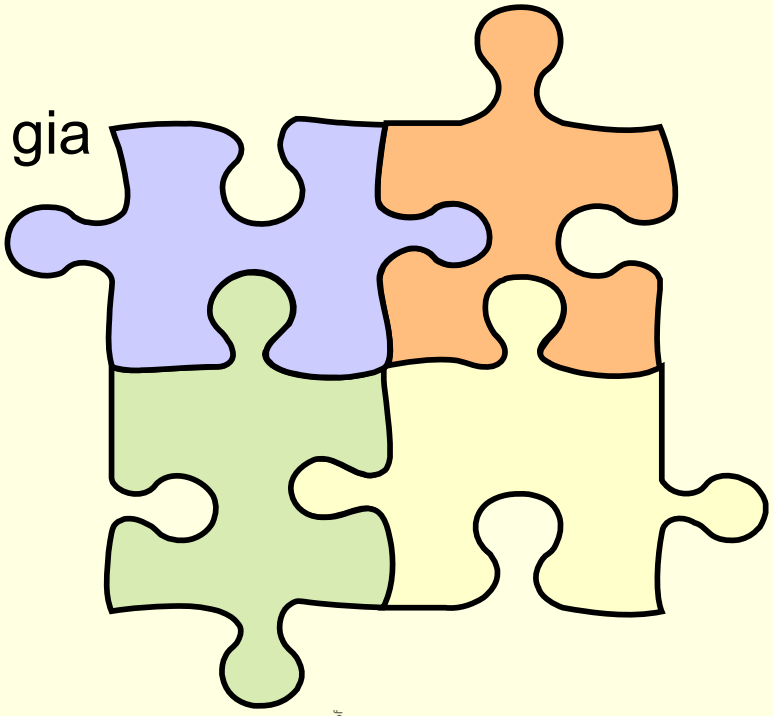
Framework development



Mô hình frameworrk

Phát triển phần mềm mã nguồn mở

- Công khai thiết kế, công khai mã nguồn, dùng chung
  - chất lượng tăng, chuẩn hóa cao?
- Phát triển phân tán, nhiều người tham gia
- Xuất phát từ các mối quan tâm chung
- Nhiều vấn đề được giải quyết
  - Lý do, lợi ích, động lực chưa rõ
- Ví dụ: GNU, Linux



Lựa chọn mô hình

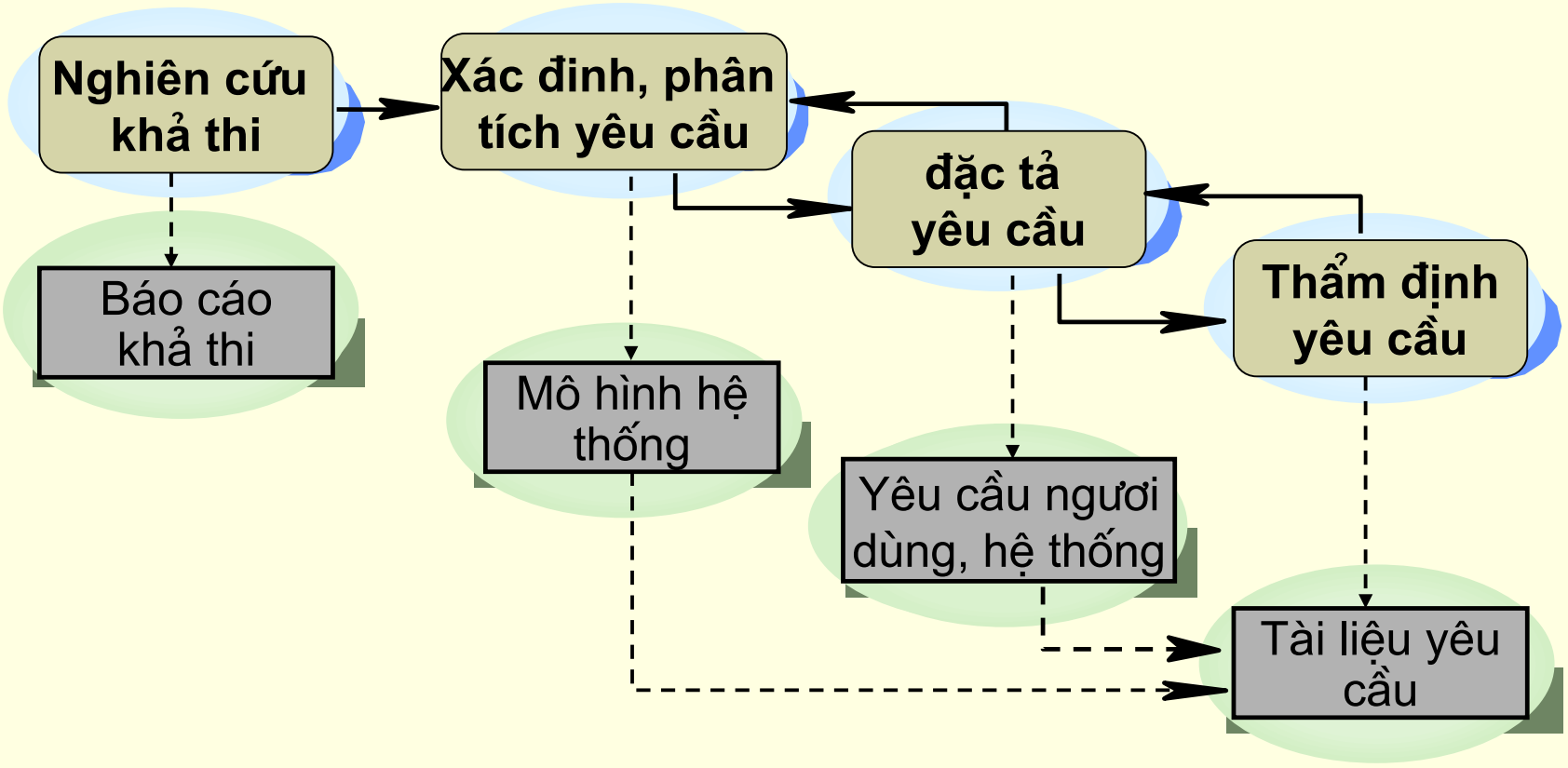
- Phụ thuộc vào bài toán, vào môi trường cụ thể
- Yêu cầu rõ ràng: Mô hình thác nước thích hợp
- Hệ phức tạp, điều khiển: Hướng sử dụng lại
- Khác:
  - Yêu cầu chưa rõ ràng, độ phức tạp cao
  - Yêu cầu có khả năng thay đổi
  - Không chắc chắn về tính hiệu quả, tính khả thi

Sử dụng: Làm bản mẫu, mô hình xoắn ốc

Các giai đoạn của tiến trình
Đặc tả yêu cầu phần mềm

- Là quá trình thiết lập các chức năng, dịch vụ mà hệ thống cần có và các ràng buộc lên sự phát triển và vận hành hệ thống
- Tiến trình kỹ nghệ yêu cầu bao gồm:
  - Nghiên cứu khả thi
  - Phân tích và xác định yêu cầu
  - Đặc tả yêu cầu
  - Thẩm định yêu cầu

Tiến trình kỹ nghệ yêu cầu



Thiết kế phần mềm

- Chuyển yêu cầu thành đặc tả thành hệ thống như nó tồn tại trong thế giới thực với các giải pháp công nghệ thích hợp để người lập trình có thể chuyển thành chương trình vận hành trên máy
- Chiến lược thiết kế phù hợp với phân tích
- Là quá trình lặp: có thể quay lại hoàn thiện phân tích→ rồi lại thiết kế
- Hai giai đoạn thiết kế: logic→thiết kế vật lý

Tiến trình thiết kế phần mềm

Các phương pháp thiết kế

- Cách tiếp cận mang tính hệ thống, có phương pháp
- Tài liệu thiết kế ở dạng một tập các mô hình đồ họa và chú giải đi kèm
- Các mô hình thường gặp:
  - Mô hình ***kiến trúc theo nhiều khung nhìn***
  - Mô hình ***luồng dữ liệu*** (xử lý)
  - Mô hình ***Thực thể – mối quan hệ/ MH Quan hệ*** (dữ liệu)
  - Mô hình ***cấu trúc mô đun*** (cấu trúc)
  - Các mô hình ***lớp đối tượng*** (kiến trúc, thực thi)

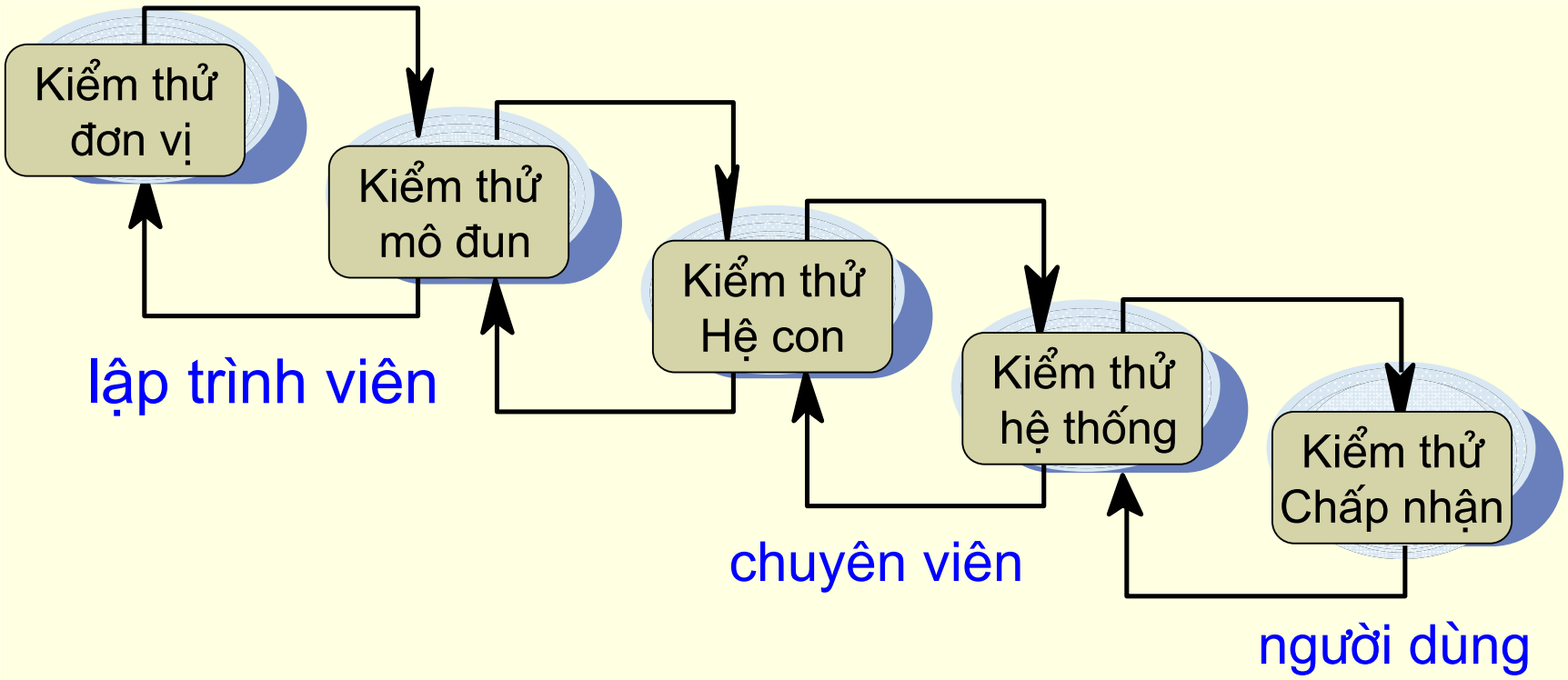
Lập trình và gỡ rối

- Là chuyển thiết kế thành chương trình, bắt lỗi và sửa lỗi
- Là một hoạt động cá nhân không có một tiến trình chung cho mọi người
- Người lập trình phải tiến hành kiểm thử để gỡ lỗi chương trình. Gỡ lỗi là hoạt động của lập trình
- Lập trình đòi hỏi chọn ngôn ngữ thích hợp và có kinh nghiệm □ phong cách lập trình tốt
- Để đảm bảo độ tin cây, cần biết lập trình tránh lỗi, thứ lỗi và ném ngoại lệ

Thẩm định phần mềm

- Thẩm định và xác minh nhằm đảm bảo hệ thống phù hợp với đặc tả và đáp ứng yêu cầu người dùng
- Nội dung bao gồm việc thanh tra, xét duyệt và kiểm thử hệ thống. Kiểm thử là quan trọng nhất
- Có nhiều mức:
  - Xác minh: *kiểm thử đơn vị, tích hợp, hệ thống*
  - Thẩm định: *làm mẫu yêu cầu, kiểm thử chấp nhận*
- Có nhiều phương pháp kiểm thử. Mỗi phương pháp áp dụng ở mức xác định, và sử dụng kỹ thuật nhất định
- Mục tiêu kiểm thử là tìm ra lỗi với chi phí ít nhất có thể

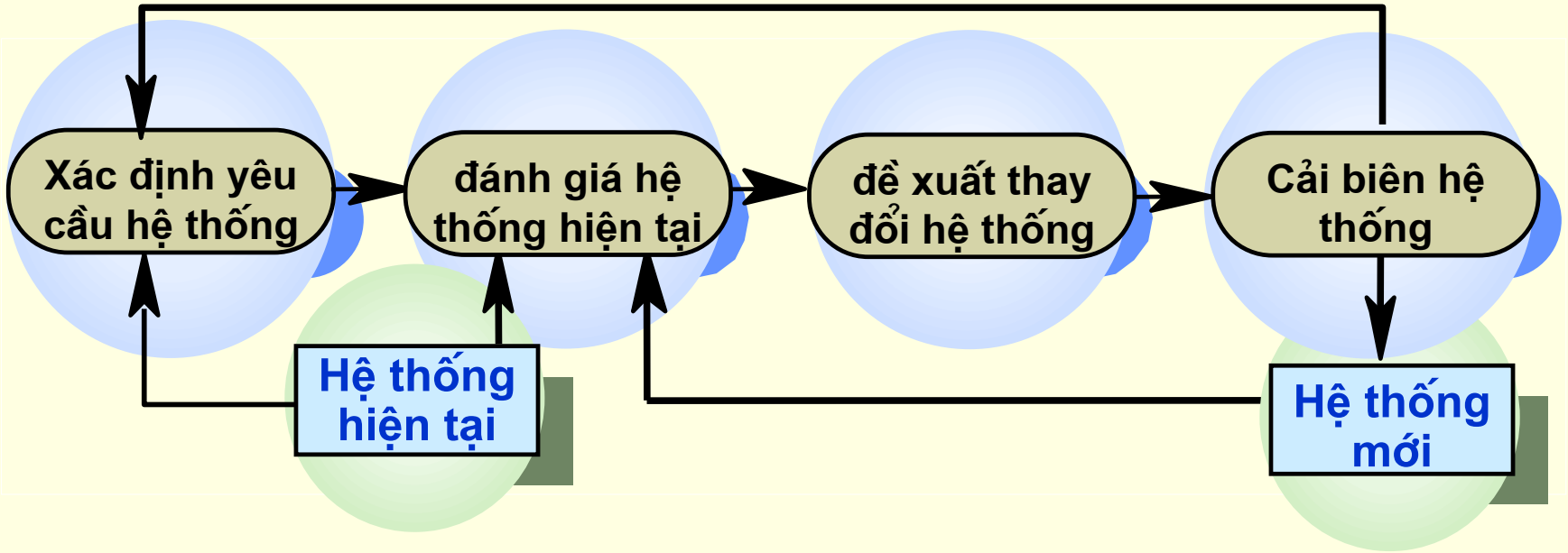
Tiến trình kiểm thử



Tiến hoá phần mềm

- Phần mềm phải mềm dẻo vì nó cần phải thay đổi.
- Môi trường nghiệp vụ và kỹ thuật luôn thay đổi: Phần mềm cần thay đổi để phù hợp với chúng → tiến hóa là tất yếu
- Phân định phát triển và tiến hoá là tương đối: giữa chúng có quan hệ chặt chẽ với nhau. Phát triển là làm mới, tiến hóa trên cơ sở hệ đã có.

Tiến trình tiến hoá



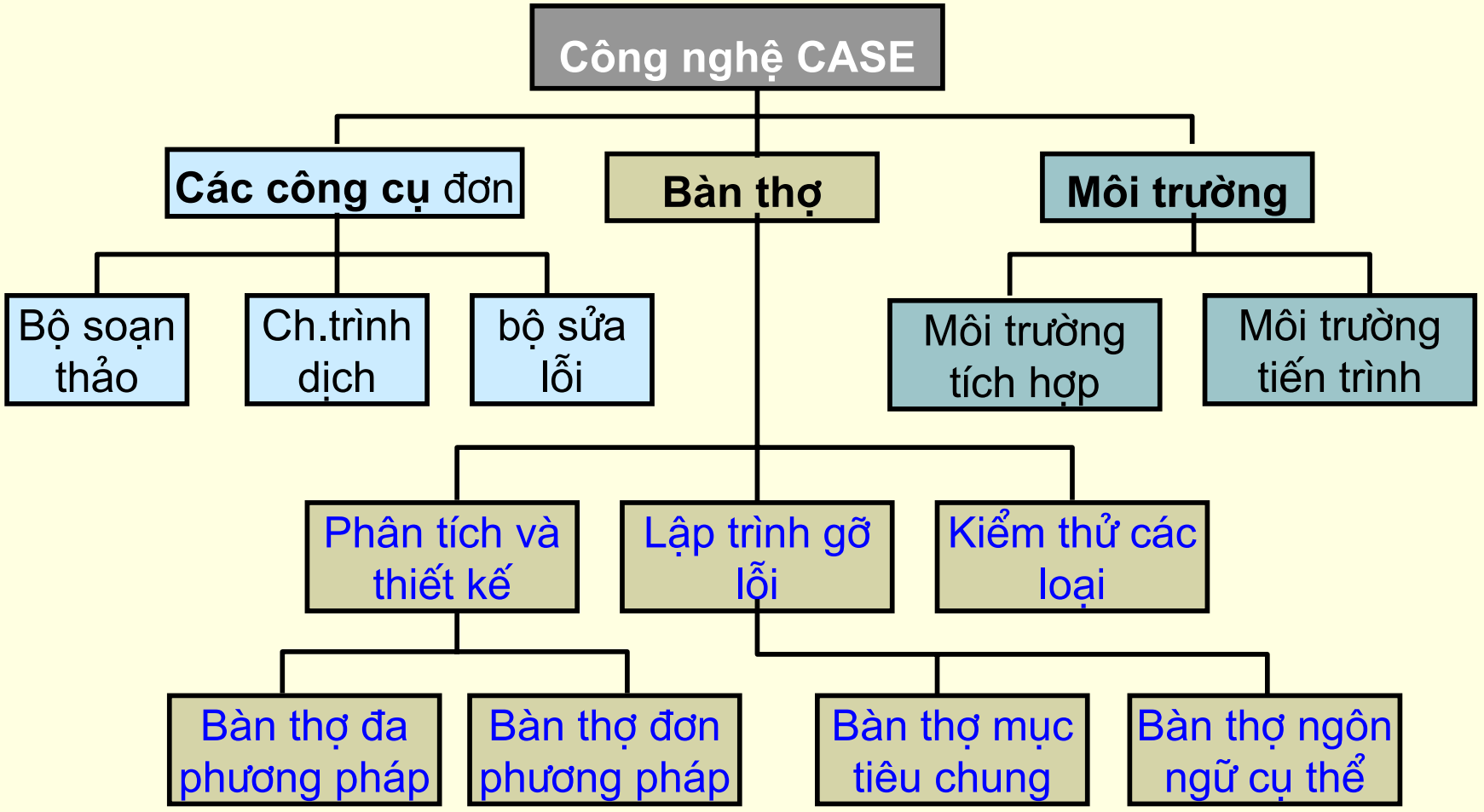
Trợ giúp tự động hoá phát triển

- *Computer-aided software engineering:*CASE là các phần mềm trợ giúp phát triển và tiến hoá hệ thống
- Các công cụ thường gặp:
  - **Bộ soạn thảo đồ thị:** để phát triển mô hình hệ thống
  - **Từ điển dữ liệu:** để quản lý các thực thể thiết kế
  - **Bộ xây dựng giao diện:** để thiết kế giao diện
  - **Bộ gỡ rối:** trợ giúp tìm lỗi trong chương trình
  - **Bộ chuyển đổi tự động:** tạo sinh các phiên bản mới từ thiết kế / hay chương trình

Công nghệ CASE (*CASE technology)*

- CASE góp phần đáng kể hoàn thiện tiến trình phần mềm cả về trình tự, tiến độ và chất lượng: *tự động hóa một phần hoạt động mô hình hóa và quản lý*
- Những hoạt động không thể tự động hóa*:*
  - Sự suy nghĩ sáng tạo trong SE
  - Lựa chọn giải pháp công nghệ
  - Giao tiếp khi làm việc nhóm
  - Thực hiện việc quản lý

Công nghệ CASSE



Phân loại CASE

- Phân loại CASE giúp hiểu và sử dụng chúng trong phát triển
- Có thể phân loại CASE theo:
  - Hướng chức năng: Công cụ cho các chức năng cụ thể: soạn thảo, lập kế hoạch, làm mẫu,..
  - Hướng tiến trình: Công cụ cho hoạt động của tiến trình được trợ giúp: mô hình nghiệp vụ, E-R
  - Hướng tích hợp: Công cụ trợ giúp tổ chức việc tích hợp các đơn vị các mức trong hệ thống
  - Dựa trên loại hoạt động: Đặc tả, thiết kế, triển khai, thẩm định và xác minh

CASE tích hợp

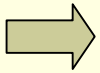
- **Các công cụ đơn (*Tools*)** : trợ giúp những nhiệm vụ riêng rẽ của tiến trình: kiểm tra sự nhất quán, soạn thảo, tạo mô hình
- **Bàn thợ (*Workbenches*):** trợ giúp 1 pha của tiến trình phát triển, như đặc tả, thiết kê,..
- **Môi trường phát triển (*Environments*):** trợ giúp toàn bộ hay một phần của toàn tiến trình phần mềm (có thể bao gồm một số bàn thợ)

Các vấn đề liên quan

- Xác định yêu cầu và thiết kế có vai trò quyết định đến chất lượng phần mềm, chiếm phần lớn công sức so với phát triển
- Khi chuyển tiếp giữa các pha phát triển phải thẩm định tốt để đảm bảo lỗi không ảnh hưởng đến pha sau
- Tài liệu tạo ra ở mỗi pha không chỉ dùng cho pha kế tiếp mà còn dùng để đảm bảo chất lượng của phần mềm và bảo trì
- Cần chuẩn hóa mẫu biểu, cách thức ghi chép, tạo tài liệu nhằm đảm bảo chất lượng phần mềm

Tính khả thi của tiến trình phát triển

Phần tử logic, khó kiểm soát



- tạo ra các tài liệu
- xét duyệt tài liệu mỗi bước

Ví dụ tài liệu:

nghiên cứu khả thi; đặc tả yêu cầu;
đặc tả thiết kế

So sánh:

- vòng đời cổ điển khả thi cao
- làm bản mẫu kém
- 4GT ??

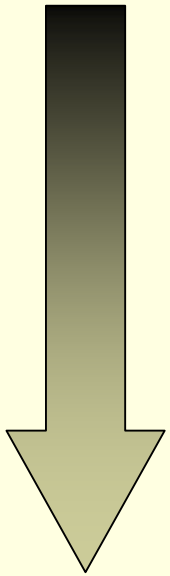
Làm tài liệu phần mềm

Vấn đề:

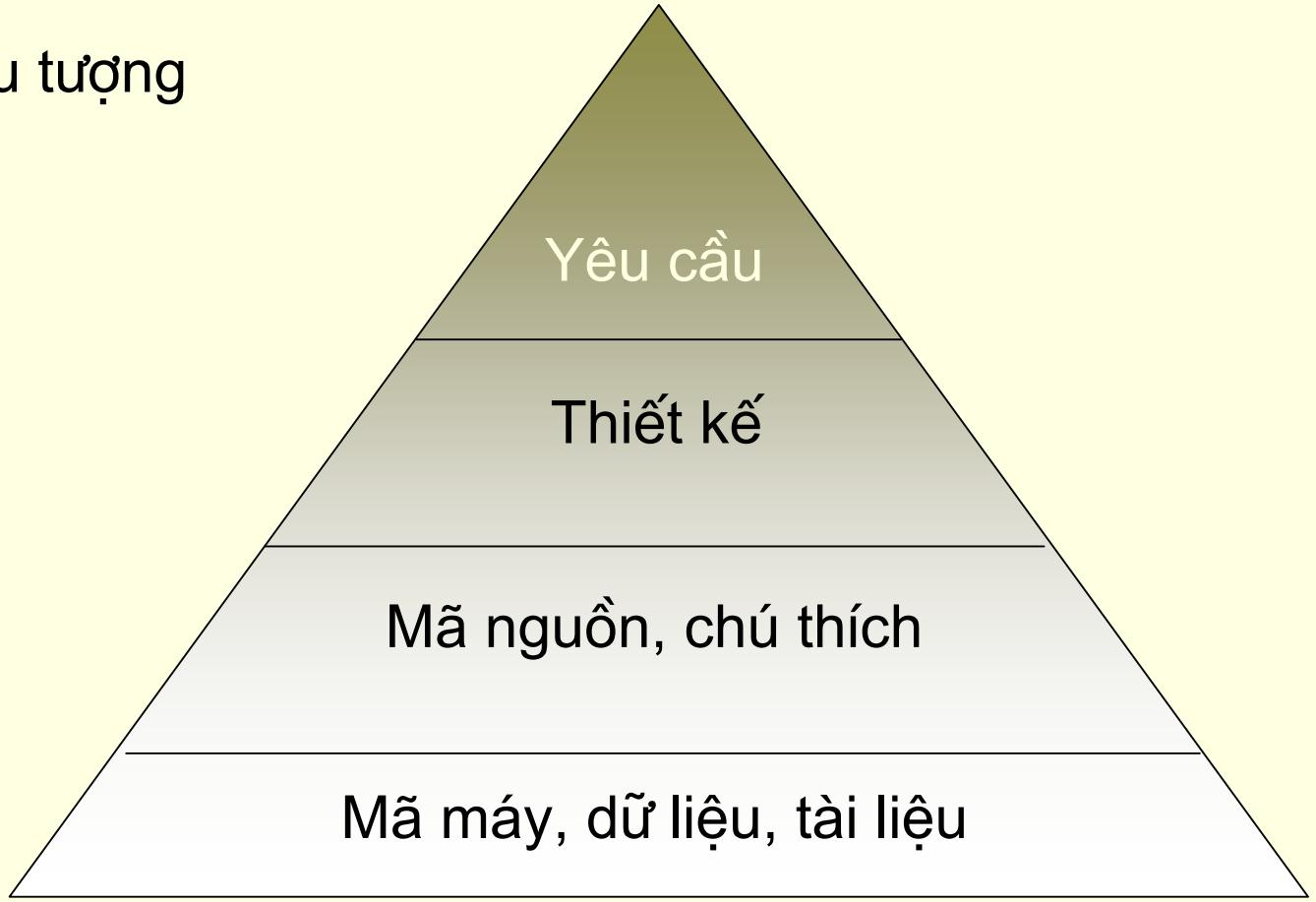
- Tạo ra chi phí phụ của phát triển
  (*người lập trình không thích viết tài liệu)*
- Sử dụng các giải pháp cục bộ để tránh sửa đổi tài liệu
- Sử dụng các mẫu có sẵn
- Sử dụng CASE trợ giúp làm tài liệu theo chuẩn

Sản phẩm (tài liệu) của dự án

Phi hình thức, trừu tượng



Hình thức hoá, cụ thể



Giảm kích cỡ, chi phí phần mềm

- Phần mềm ngày càng lớn, phức tạp
- Tổ chức làm việc theo nhóm, tiến hành song song
- Cần phân rã chức năng; giảm kích cỡ (mã nguồn), tăng năng suất:
  - tái sử dụng: thư viện thương mại,...
  - tự sinh mã: công cụ tạo giao diện,...
  - hướng đối tượng: kế thừa, bảo trì
  - ngôn ngữ bậc cao: năng lực biểu diễn cao

Quan hệ tiến trình và sản phẩm

Tiến trình và sản phẩm là hai mặt của phát triển:

- Tiến trình tốt đảm bảo ràng buộc về sản phẩm
- Sản phẩm tốt là sự tổng hoà của nhiều yếu tố:
  - Tiến trình thích hợp
  - Đội ngũ chuyên môn tốt
  - Công cụ trợ giúp mạnh
  - Năng lực quản lý tiến trinh của tổ chức cao (CMM)

5 mức CMM (Capability Maturity Model):

1. Tùy biến
2. Lặp lại được
3. Được xác định
4. Được quản lý
5. Tối ưu hóa

Câu hỏi ôn tập

1. Có mấy loại mô hình tiến trình? Là loại nào?
2. Trình bày nội dung của các mô hình: thác nước, làm mẫu, xoáy ốc, tiến hoá, tăng trưởng, ứng dụng nhanh, hình thức hoá, đối tượng, mô hình sử dụng lại, mô hình mã nguồn mở, mô hình thế hệ thứ 4 theo các nội dung sau:
   - Nội dung? đặc trưng?
   - ưu, nhược điểm?
   - cần yêu cầu gì?
   - thích hợp khi nào?
3. Mô tả tiến trình kỹ nghệ yêu cầu?
4. Mô tả tiên trình thiết kế phần mềm? Nêu các mô hình thiết kế thường sử dụng?

Câu hỏi ôn tập (t)

5. Định nghĩa thẩm định phần mềm? Thẩm định và xác minh gồm những hoạt động gì?
6. Có những loại kiểm thử nào? Mô tả tiến trình kiểm thử?
7. Tiến hoá phần mềm là gì? Lý do?
8. Mô tả tiến trình tiến hoá hệ thống?
9. CASE là gì? Các cách phân loại CASE?
10. CASE tích hợp gồm những loại nào? vẽ sơ đồ cấu trúc các loại CASE?
11. Làm thế nào để đảm bảo khả thi của mô hình phát triển? So sánh sự khả thi giữa các mô hình, giải thích vì sao?
12. Làm thế nào để có thể giảm kích cỡ, chi phí của mô hình? Những mô hình nào, ngôn ngữ nào có ưu thế về mặt này?

Câu hỏi và thảo luận