

# Công nghệ phần mềm

**Thời gian: 2 ĐVHT = 30 tiết**

**Tài liệu tham khảo:**

Nguyễn Văn Vy, Nguyễn Việt Hà, Giáo  
trình kỹ nghệ phần mềm. NXB ĐHQGHN

R.Pressman, Kỹ nghệ phần mềm, tập 1,2,3.  
NXB Giáo dục, Hà Nội, 1997 (Người dịch:  
Ngô Trung Việt)

# Mục tiêu của môn học

---

- Giới thiệu chung về bản chất phần mềm, lịch sử phát triển của công nghệ phần mềm (CNPM)
  - Cung cấp kiến thức cơ bản về quản lý dự án công nghệ thông tin từ khâu lập kế hoạch đến hoàn tất dự án
  - Cung cấp kiến thức cơ bản về các phương pháp và kỹ thuật xác định, phân tích yêu cầu người dùng;
  - Giới thiệu các phương pháp, kỹ thuật thiết kế và lập trình phần mềm;
  - Cung cấp kiến thức cơ bản về các phương pháp kiểm thử và bảo trì phần mềm.
-

# Nội dung

---

Chương 1: TỔNG QUAN VỀ CÔNG NGHỆ PHẦN MỀM

Chương 2: QUẢN LÝ DỰ ÁN CNTT

Chương 3: YÊU CẦU NGƯỜI DÙNG

Chương 4: THIẾT KẾ VÀ LẬP TRÌNH

Chương 5: KIỂM THỬ VÀ BẢO TRÌ

---

# Chương 1

## Tổng quan về công nghệ phần mềm

# Nội dung

---

1. Một số khái niệm cơ bản
2. Công nghệ phần mềm
3. Quy trình công nghệ phần mềm
4. Khủng hoảng phần mềm
5. Một số quan điểm sai lệch
6. Lịch sử công nghệ phần mềm
7. Hướng tương lai của công nghệ phần mềm

# 1. Một số khái niệm cơ bản

- Phần mềm (software) là gì?
- Đặc trưng của phần mềm
- Thuộc tính của phần mềm
- Các loại phần mềm
- Phần mềm lỗi thời
- Các vấn đề về phần mềm

# Phần mềm máy tính là gì?

---

- Phần mềm máy tính (Computer software) là:
  - Sản phẩm do các nhà phát triển phần mềm thiết kế và xây dựng
- Software bao gồm:
  - Programs
  - Documentation, procedures to setup and operate,..
- **Vấn đề đặt ra:** So sánh chương trình (program) và phần mềm (software)

# Phần mềm máy tính là gì?

- **Chương trình**
  - Một người viết, một người dùng (người viết ≡ người dùng)  
Mục đích thu thập, xử lý số liệu (dùng 1 lần)  
Không tài liệu, không kiểm thử triệt để
- **Sản phẩm phần mềm**
  - Nhiều người viết, nhiều người dùng, độ phức tạp cao, đồng bộ, an toàn, an ninh
  - Kinh nghiệm viết chương trình nhỏ không áp dụng cho sản phẩm lớn

# Phần mềm máy tính là gì?

---

- **Phần mềm là gì?**
- Phần mềm là các chương trình máy tính và những tài liệu liên quan đến nó như: các yêu cầu, mô hình thiết kế, tài liệu hướng dẫn sử dụng...
- Các sản phẩm phần mềm được chia thành 2 loại:
  - Sản phẩm đại trà (Generic Product)
  - Sản phẩm theo đơn đặt hàng (Bespoke Product hoặc Customized Product)

# Phần mềm máy tính là gì?

---

- Sản phẩm đại trà : được phát triển để bán ra ngoài thị trường, đối tượng người sử dụng tương đối đa dạng và phong phú. Những sản phẩm phần mềm thuộc loại này thường là những phần mềm dành cho máy PC.
- Sản phẩm theo đơn đặt hàng được phát triển cho một khách hàng riêng lẻ theo yêu cầu.

*Ví dụ:* Những hệ thống phần mềm chuyên dụng, hỗ trợ nghiệp vụ cho một doanh nghiệp riêng lẻ ...

# Các đặc trưng của phần mềm

## Software Characteristics

---

- **Phần mềm được phát triển chứ không phải là sản xuất**
- **Phần mềm không hao mòn**
- **Phần mềm rất phức tạp, chi phí thay đổi rất lớn**

# Thuộc tính của phần mềm

---

- **Khả năng bảo trì:** Nó có khả năng thực hành những tiến triển để thỏa mãn yêu cầu của khách hàng.
- **Khả năng tin cậy:** Khả năng tin cậy của phần mềm bao gồm một loạt các đặc tính như là độ tin cậy, an toàn, và bảo mật. Phần mềm tin cậy không thể tạo ra các thiệt hại vật chất hay kinh tế trong trường hợp hư hỏng.
- **Độ hữu hiệu:** Phần mềm không thể phí phạm các nguồn tài nguyên như là bộ nhớ và các chu kỳ vi xử lý.
- **Khả năng sử dụng:** Phần mềm nên có một giao diện tương đối dễ cho người dùng và có đầy đủ các hồ sơ về phần mềm

# Các loại phần mềm

---

- Thập niên 1980 hầu hết sản phẩm phần mềm đều làm theo đơn đặt hàng riêng.
- Kể từ khi có PC, các phần mềm được phát triển và bán cho hàng trăm ngàn khách hàng
- Có thể phân loại tổng quát như sau:
  - Sản phẩm **tổng quát**: Được bán cho bất kỳ khách hàng nào có khả năng tiêu thụ.
  - Sản phẩm **chuyên ngành**: Phần mềm được phát triển một cách đặc biệt cho khách hàng qua các hợp đồng.

# Phân loại phần mềm

---

Ba cách phân loại phần mềm

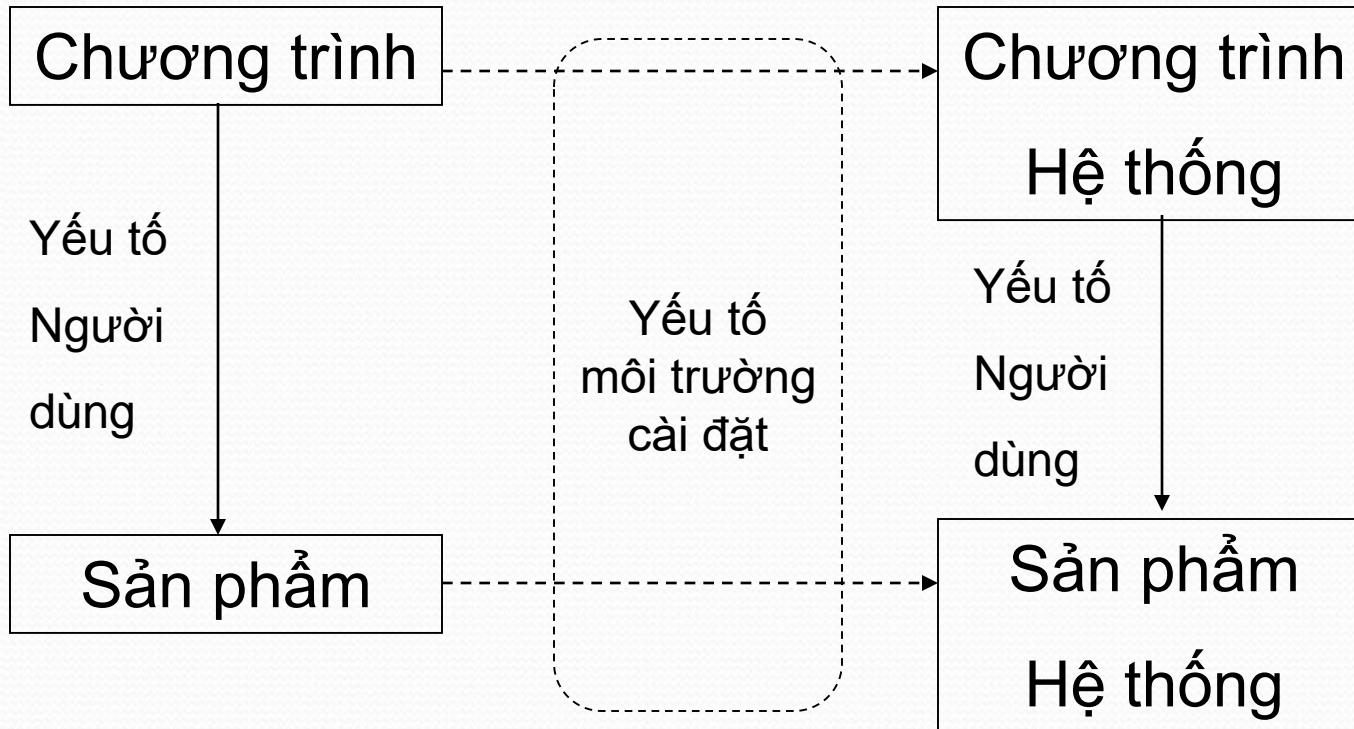
- Theo mức độ hoàn thiện
- Theo chức năng thực hiện
- Theo lĩnh vực ứng dụng

# Phân loại theo mức độ hoàn thiện

---

- Chương trình
  - Một người viết, một người dùng (người viết ≡ người dùng)  
Mục đích thu thập, xử lý số liệu (dùng 1 lần)  
Không tài liệu, không kiểm thử triệt để
- Sản phẩm phần mềm
  - Nhiều người viết, nhiều người dùng, độ phức tạp cao, đồng bộ, an toàn, an ninh
  - ❖ Kinh nghiệm viết chương trình nhỏ không áp dụng cho sản phẩm lớn

# Phân loại theo mức độ hoàn thiện



# Phân loại theo chức năng

---

- **Phần mềm hệ thống**
  - Điều hành hoạt động máy tính, thiết bị và chương trình
  - Trợ giúp các tiện ích (tổ chức tệp, nén, dọn đĩa)
- **Phần mềm nghiệp vụ**
  - Trợ giúp các hoạt động nghiệp vụ khác nhau
  - Có số lượng lớn, đa dạng
  - Phân làm hai loại theo cách làm:
    - Sản phẩm đặt hàng
      - Sản xuất theo đơn đặt hàng
      - Đơn chiếc, yêu cầu đặc thù
    - Sản phẩm chung
      - Bán rộng rãi
      - Thỏa mãn yêu cầu chung số lớn người dùng
      - ❖ Mỗi loại có cách thức tiếp cận riêng, nhất là ở một số các bước -> chi phí, thời gian khác nhau.

# Phân loại theo chức năng

---

- **Phần mềm công cụ**

- Trợ giúp cho quá trình phát triển phần mềm
- Các ngôn ngữ lập trình (soạn thảo, dịch, gõ rối,...)
- Công cụ trợ giúp 1, nhiều giai đoạn phát triển (phân tích, thiết kế, quản lý dự án, kiểm thử, ...)

# Phân loại theo lĩnh vực ứng dụng

---

- **Phần mềm hệ thống**

- Phục vụ cho các chương trình khác nhau
- Tương tác trực tiếp với phần cứng
- Phục vụ nhiều người dùng

- **Phần mềm thời gian thực**

- Thu thập, xử lý các dữ kiện thế giới thực
- Đáp ứng yêu cầu chặt chẽ về thời gian

# Phân loại theo lĩnh vực ứng dụng

---

- **Phần mềm nghiệp vụ**
  - Xử lý thông tin nghiệp vụ, gắn với CSDL
  - Xử lý các giao tác mạng
  - Các lĩnh vực ứng dụng rất lớn (như các hệ điều khiển vũ trụ)
- **Phần mềm khoa học kỹ thuật**
  - Dùng thuật toán phức tạp (vật lí, mô phỏng)
  - Năng lực tính toán cao

# Phân loại theo lĩnh vực ứng dụng

---

- **Phần mềm nhúng**
  - Chỉ đọc ra khi thiết bị khởi động
  - Thực hiện chức năng hạn chế (như điều khiển sản phẩm)
  - Là sự kết hợp giữa hệ thống và thời gian thực
- **Phần mềm máy tính cá nhân**
  - Các bài toán nghiệp vụ nhỏ, học tập, giải trí
  - Giao diện đồ họa
  - Có nhu cầu rất cao
- **Phần mềm trí tuệ nhân tạo**
  - Dùng các thuật toán phi số (logic): suy luận, tìm kiếm.
  - Hệ chuyên gia, nhận dạng, trò chơi,...
- **Phần mềm dựa trên nền web**
  - Cung cấp dịch vụ khai thác thông tin trên web
  - Chương trình khai thác là chung

# Các vấn đề về phần mềm

---

- Hiểu không đúng những gì người dùng cần
  - Không thích ứng với các thay đổi về yêu cầu
  - Các Module không khớp với nhau
  - Khó bảo trì, nâng cấp và mở rộng
  - Phát hiện trễ các lỗ hổng của dự án
  - Chất lượng kém, hiệu năng thấp
  - Các thành viên không biết chính xác ai thay đổi cái gì, khi nào, ở đâu và vì sao phải thay đổi
  - Quá trình Build-and-Release không đáng tin cậy
-

# Các nguyên nhân

---

- Sự quản lý y/c của người dùng không đầy đủ
  - Trao đổi thông tin mơ hồ, không đầy đủ
  - Độ phức tạp vượt quá tầm kiểm soát
  - Kiến trúc không vững chắc
  - Có mâu thuẫn không phát hiện được giữa yêu cầu, thiết kế và cài đặt
  - Kiểm chứng không đầy đủ
  - Lượng giá chưa chính xác về rủi ro
  - Thiếu công cụ tự động hóa
-

## 2. Công nghệ phần mềm là gì? (software engineering)

---

- Phần mềm ứng dụng được sử dụng rộng rãi và trở nên không thể thiếu được trong mọi lĩnh vực
- Công nghệ phần mềm (software industry) ngày nay đã trở thành ưu thế (**dominant factor**) trong thế giới công nghiệp hóa
- Edsger Dijkstra đã nói:

*“Khi máy tính chưa xuất hiện, thì việc lập trình chưa có khó khăn gì cả. Khi mới xuất hiện một vài chiếc máy tính chức năng kém thì việc lập trình bắt đầu gặp một vài khó khăn nhỏ. Giờ đây khi chúng ta có những chiếc máy tính khổng lồ thì những khó khăn ấy trở nên vô cùng lớn. Như vậy ngành công nghiệp điện tử không giải quyết khó khăn nào cả mà họ chỉ tạo thêm ra những khó khăn mới. Khó khăn mà họ tạo nên chính là việc sử dụng sản phẩm của họ”*

---

# Công nghệ phần mềm

---

- **Công nghệ phần mềm** hay **kỹ nghệ phần mềm** (*software engineering*) là sự áp dụng một cách tiếp cận có hệ thống, có kỷ luật, và định lượng được cho việc phát triển, hoạt động và bảo trì phần mềm
  - Ngành học kỹ nghệ phần mềm bao trùm **kiến thức**, các **công cụ**, và các **phương pháp** cho việc định nghĩa yêu cầu phần mềm, và thực hiện các tác vụ **thiết kế** phần mềm, **xây dựng** phần mềm, **kiểm thử** phần mềm (*software testing*) và **bảo trì** phần mềm
  - Kỹ nghệ phần mềm còn sử dụng kiến thức của các lĩnh vực như kỹ thuật máy tính, khoa học máy tính, quản lý, toán học, quản lý dự án, quản lý chất lượng, công thái học phần mềm (*software ergonomics*), và kỹ nghệ hệ thống (*systems engineering*)
-

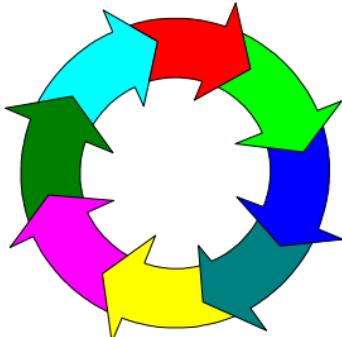
# Công nghệ phần mềm là gì?

(Software Engineering)

- “A discipline whose aim is the production of quality software, software that is delivered on time, within budget, and that satisfies its requirements”

*Là môn học giúp sản xuất phần mềm chất lượng, phần mềm được giao vào đúng thời gian, trong ngân sách, và đáp ứng các yêu cầu đặt ra.*

- Mục đích của công nghệ phần mềm là để cung cấp nền tảng (framework) để xây dựng phần mềm chất lượng cao



# Công nghệ phần mềm

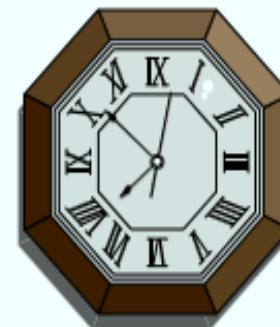
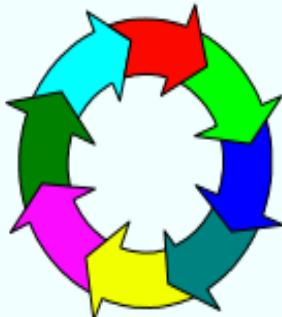
---

## **Định nghĩa cổ điển (của Fritz Bauer)**

Công Nghệ Phần Mềm là sự thiết lập và sử dụng các nguyên tắc khoa học nhằm mục đích tạo ra các phần mềm một cách kinh tế mà các phần mềm đó hoạt động hiệu quả và tin cậy trên các máy tính.

## **Định nghĩa khác:** Công Nghệ Phần Mềm

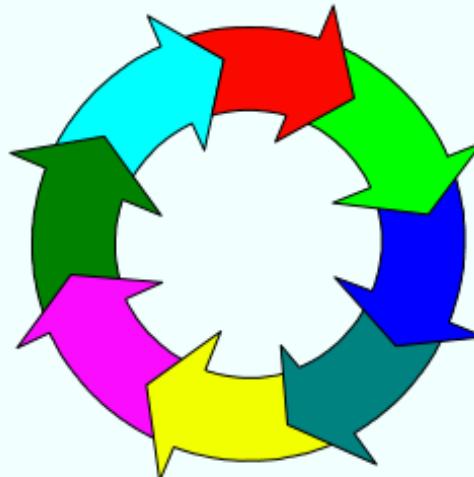
- Là các quy trình đúng kỹ luật và có định lượng được áp dụng cho sự phát triển, thực thi và bảo trì các hệ thống thiên về phần mềm
- Tập trung vào quy trình, sự đo lường, sản phẩm, tính đúng thời gian và chất lượng



# CHU TRÌNH

---

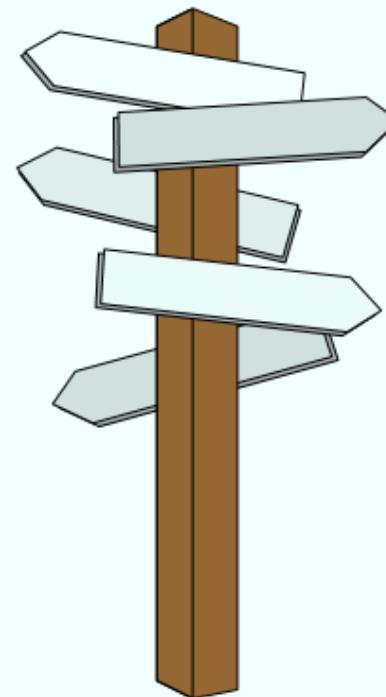
Chu trình (*process*) định nghĩa một bộ khung các tiêu chuẩn phải được thiết lập để triển khai công nghệ phần mềm.



# PHƯƠNG PHÁP

Phương pháp (*method*) chỉ ra cách thực hiện những công việc cụ thể (“*how to*”):

- ◆ phân tích yêu cầu
- ◆ thiết kế
- ◆ xây dựng chương trình
- ◆ kiểm tra
- ◆ sửa lỗi
- ◆ ...



# CÔNG CỤ

---



- Công cụ (*tool*) cung cấp các hỗ trợ tự động hay bán tự động đối với chu trình và phương pháp
- Các công cụ được tích hợp tạo thành CASE (*Computer Aided Software Engineering*)

# MỘT CÁCH NHÌN TỔNG QUAN VỀ CNPM

Gồm 3 giai đoạn lớn

- ◆ Giai đoạn định nghĩa: Phân tích hệ thống (*system engineering*), Hoạch định đê tài (*software project management*), Phân tích yêu cầu (*requirement analysis*).
- ◆ Giai đoạn phát triển: Thiết kế phần mềm (*software design*), sinh mã (*code generation*), kiểm tra phần mềm (*software testing*)
- ◆ Giai đoạn bảo trì: Sửa lỗi (*correction*), thay đổi môi trường thực thi (*adaptation*), tăng cường (*enhancement*)

# 3. Quy trình phần mềm là gì?

(Software Development/Engineering Process - SEP)

---

- Là một tập hợp các hành động mà mục đích của nó là xây dựng và phát triển phần mềm
- Xác định bộ khung và tiêu chuẩn để triển khai CNPM
- Quy trình: Chỉ ra những công việc phải làm
- Phương pháp: Chỉ cách thức thực hiện công việc
- Thông thường một qui trình bao gồm những yếu tố cơ bản sau:
  - Thủ tục (Procedures)
  - Hướng dẫn công việc (Activity Guidelines)
  - Biểu mẫu (Forms/templates)
  - Danh sách kiểm định (Checklists)
  - Công cụ hỗ trợ (Tools)

# 5 bước cơ sở của quy trình phần mềm

---

- **Đặc tả yêu cầu** (Requirements Specification): chỉ ra những “đòi hỏi” cho cả các yêu cầu chức năng và phi chức năng
  - **Phát triển phần mềm** (Development): tạo ra phần mềm thỏa mãn các yêu cầu được chỉ ra trong “Đặc tả yêu cầu”
  - **Kiểm thử phần mềm** (Validation/Testing): để bảo đảm phần mềm sản xuất ra đáp ứng những “đòi hỏi” được chỉ ra trong “Đặc tả yêu cầu”
  - **Triển khai và bảo trì**
  - **Thay đổi phần mềm** (Evolution): đáp ứng nhu cầu thay đổi của khách hàng
-

# Quy trình phần mềm là gì? (Software Process)

- CNPM là công nghệ theo lớp (layered technology)



- Nền tảng của CNPM chính là lớp **Process**, nó là chất kết dính công nghệ và cho phép phát triển các phần mềm hiệu quả và đúng thời hạn

# CASE

## (Computer Aided Software Engineering)

---

- CASE là công cụ thường được sử dụng để hỗ trợ hoạt động trong quy trình xây dựng phần mềm
- Có 2 loại:
  - **Upper - case**: Công cụ hỗ trợ các hoạt động đầu tiên như đặc tả yêu cầu và thiết kế
  - **Lower - case**: Công cụ hỗ trợ các hoạt động sau như lập trình, gỡ rối và kiểm thử
- Nhược điểm:
  - Không thể sáng tạo tự động
  - Không thể hỗ trợ tốt cho hoạt động nhóm

## 4. Khủng hoảng phần mềm (Software crisis)

---

- Khủng hoảng phần mềm xảy ra từ 1970.
- Công nghệ máy tính (phần cứng) đã có những cải thiện cấp số mũ (exponential) về giá cả và mức độ thực thi.
- Nhưng với phần mềm thì sao? Theo báo cáo của IBM (2000)
  - 31% dự án phần mềm bị hủy bỏ trước khi hoàn thành
  - 53% dự án vượt quá chi phí dự kiến
  - 94% dự án phải bắt đầu lại

# **Khủng hoảng phần mềm (Software crisis)**

- Sự cố 2YK (time bomb)
- Patriot missile (1991): Lỗi phần mềm làm phá hủy tên lửa Patriot của Mỹ trong cuộc chiến vùng vịnh
- Chỉ 3 phút sau khi cất cánh từ căn cứ Kourou, trên lãnh thổ Guiana thuộc Pháp (giáp Brazil), tên lửa đẩy Ariane 5-ESCA thế hệ mới của Cơ quan Vũ trụ châu Âu, mang theo hai vệ tinh, đã vỡ tan và rơi xuống biển.



# 5. Một số quan niệm sai lầm

---

Tìm hiểu về các sai lầm sau:

- Sai lầm của người quản lý (Management myth)
- Sai lầm của khách hàng (Customer myth)
- Sai lầm của người lập trình (practitioner myth)

# Lịch sử công nghệ phần mềm

---

- **Thập niên 1940:** Các chương trình cho máy tính được viết bằng tay.
- **Thập niên 1950:** Các công cụ đầu tiên xuất hiện như là phần mềm biên dịch Macro Assembler và phần mềm thông dịch đã được tạo ra và sử dụng rộng rãi để nâng cao năng suất và chất lượng. Các trình dịch được tối ưu hoá lần đầu tiên ra đời.
- **Thập niên 1960:** Các công cụ của thế hệ thứ hai như các trình dịch tối ưu hoá và công việc kiểm tra mẫu đã được dùng để nâng cao sản phẩm và chất lượng. Khái niệm công nghệ phần mềm đã được bàn thảo rộng rãi.

# Lịch sử công nghệ phần mềm

---

- **Thập niên 1970:** Các công cụ phần mềm, chẳng hạn trong UNIX các vùng chứa mã, lệnh make, v.v. được kết hợp với nhau. Số lượng doanh nghiệp nhỏ về phần mềm và số lượng máy tính cỡ nhỏ tăng nhanh.
- **Thập niên 1980:** các PC và máy trạm ra đời. Cùng lúc có sự xuất hiện của mô hình dự toán khả năng. Lượng phần mềm tiêu thụ tăng mạnh.
- **Thập niên 1990:** Phương pháp lập trình hướng đối tượng ra đời. Các quá trình nhanh như là lập trình cực hạn được chấp nhận rộng rãi. Trong thập niên này, WWW và các thiết bị máy tính cầm tay phổ biến rộng rãi.
- **Hiện nay:** Các phần mềm biên dịch và quản lý như là .NET, PHP và Java làm cho việc viết phần mềm trở nên dễ dàng hơn nhiều. : Các công cụ phần mềm, chẳng hạn trong UNIX các vùng chứa mã, lệnh make, v.v. được kết hợp với nhau. Số lượng doanh nghiệp nhỏ về phần mềm và số lượng máy tính cỡ nhỏ tăng nhanh.

# Hướng tương lai của công nghệ phần mềm

- Lập trình định dạng và các phương pháp linh hoạt sẽ giữ vai trò quan trọng trong tương lai của công nghệ phần mềm. ICSE 2005 đã tham gia theo dõi cả hai chủ đề này. (ICSE là dạng viết tắt của *International Conference on Software Engineering* tức là Hội nghị Quốc tế về Kỹ nghệ Phần mềm.)
- Lập trình định dạng (*aspect-oriented programming*) sẽ giúp người lập trình ứng xử với các yêu cầu không liên quan đến các chức năng thực tế của phần mềm bằng cách cung ứng các công cụ để thêm hay bớt các khối mã ít bị thay đổi trong nhiều vùng của mã nguồn. Lập trình định dạng mô tả các đối tượng và hàm nên ứng xử như thế nào trong một tình huống cụ thể.

# Hướng tương lai của công nghệ phần mềm

---

- **Phát triển phần mềm linh hoạt:** nhằm hướng dẫn các đề án phát triển phần mềm mà trong đó bao gồm việc thỏa mãn các nhu cầu thay đổi và sự cạnh tranh của thị trường một cách nhanh chóng. Các quá trình cồng kềnh, nặng về hồ sơ tính như là TickIT, CMM và ISO 9000 đang lu mờ dần tầm quan trọng.

# Công nghiệp phần mềm Việt Nam

---

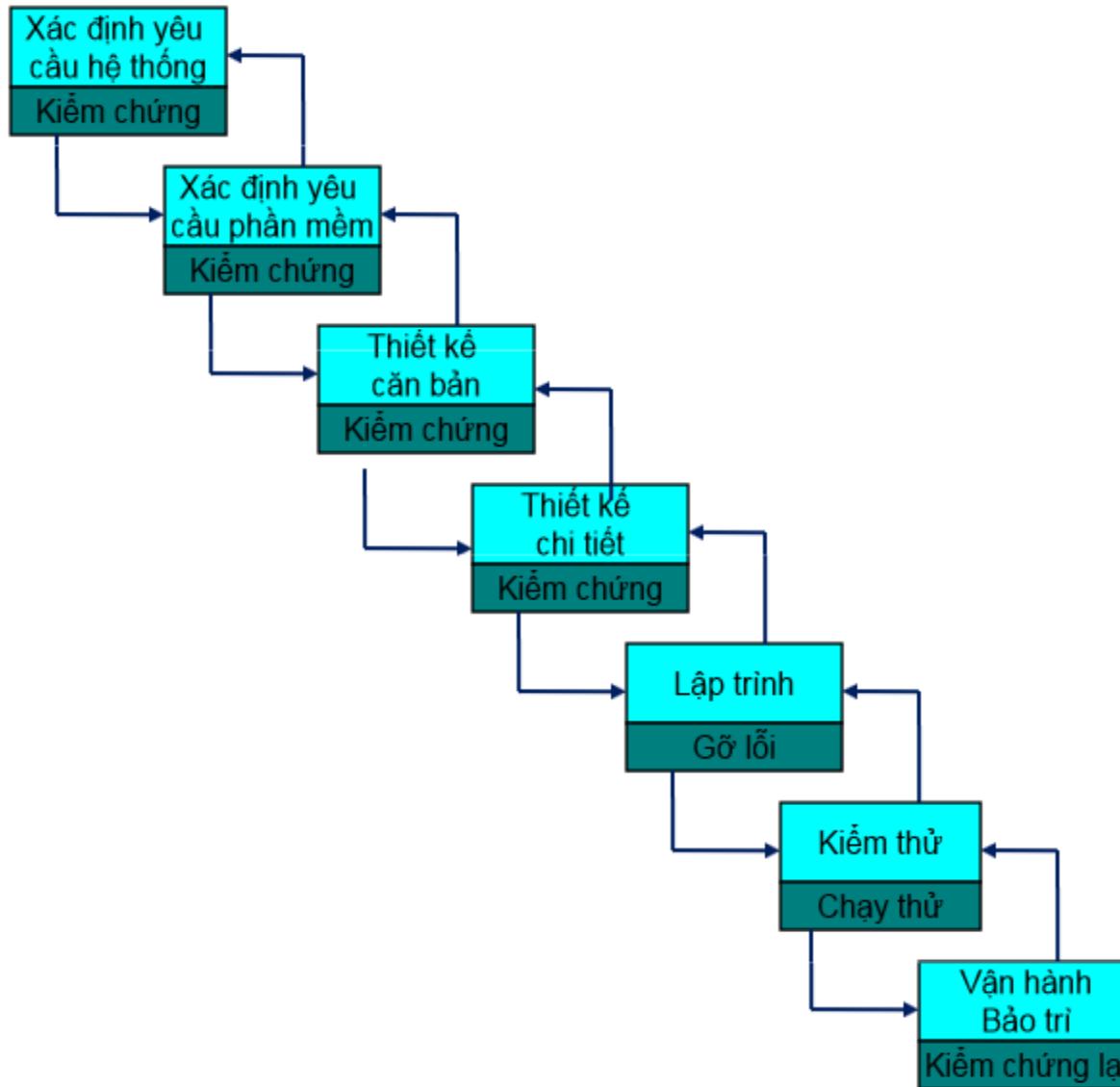
- Báo cáo

# TIẾN TRÌNH PHẦN MỀM

## Vòng đời phần mềm (Software life-cycle)

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

# Mô hình vòng đời phần mềm của Boehm



# Suy nghĩ mới về vòng đời phần mềm

- (1) Pha xác định yêu cầu và thiết kế có vai trò quyết định đến chất lượng phần mềm, chiếm phần lớn công sức so với lập trình, kiểm thử và chuyển giao phần mềm
- (2) Pha cụ thể hóa cấu trúc phần mềm phụ thuộc nhiều vào suy nghĩ trên xuống (top-down) và trừu tượng hóa, cũng như chi tiết hóa
- (3) Pha thiết kế, chế tạo thì theo trên xuống, pha kiểm thử thì dưới lên (bottom-up)

# Suy nghĩ mới về vòng đời phần mềm

- (4) Trước khi chuyển sang pha kế tiếp phải đảm bảo pha hiện nay đã được kiểm thử không còn lỗi
- (5) Cần có cơ chế kiểm tra chất lượng, xét duyệt giữa các pha nhằm đảm bảo không gây lỗi cho pha sau
- (6) Tư liệu của mỗi pha không chỉ dùng cho pha sau, mà chính là đối tượng quan trọng cho kiểm tra và đảm bảo chất lượng của từng quy trình và của chính phần mềm

# Suy nghĩ mới về vòng đời phần mềm

- 7) Cân chuẩn hóa mẫu biểu, cách ghi chép tạo tư liệu cho từng pha, nhằm đảm bảo chất lượng phần mềm
- 8) Thao tác bảo trì phần mềm là việc xử lý quay vòng trở lại các pha trong vòng đời phần mềm nhằm biến đổi, sửa chữa, nâng cấp phần mềm

# Các phương pháp luận và kỹ thuật cho từng pha

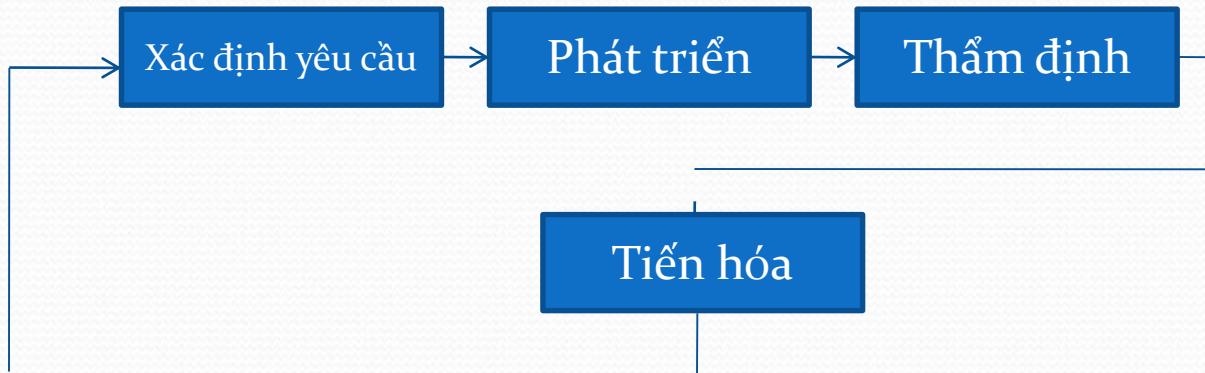
Tên pha	Nội dung nghiệp vụ	Phương pháp, kỹ thuật
Xác định yêu cầu	Đặc tả yêu cầu người dùng Xác định yêu cầu phần mềm	Phân tích cấu trúc hóa
Thiết kế hệ thống	Thiết kế cơ bản phần mềm Thiết kế cấu trúc ngoài của phần mềm	Thiết kế cấu trúc hóa
Thiết kế chương trình	Là thiết kế chi tiết: Thiết kế cấu trúc bên trong của phần mềm (đơn vị chương trình hoặc môđun)	Lập trình cấu trúc Phương pháp Jackson Phương pháp Warnier
Lập trình	Mã hóa bởi ngôn ngữ lập trình	Mã hóa cấu trúc hóa
Đảm bảo chất lượng	Kiểm tra chất lượng phần mềm đã phát triển	Phương pháp kiểm thử chương trình
Vận hành Bảo trì	Sử dụng, vận hành phần mềm đã phát triển. Biến đổi, điều chỉnh phần mềm	Chưa cụ thể

# Chương 2: TIẾN TRÌNH PHẦN MỀM

## 2.1. Tổng quan

### a. Tiến trình

- Khi xây dựng một phần mềm một cách công nghiệp, điều quan trọng là phải vạch ra được một loạt các bước hoạt động dự kiến trước – một lộ trình mà theo đó có thể tạo ra một phần mềm chất lượng, hiệu quả và đúng hạn. Một lộ trình như vậy được gọi là một *tiến trình phần mềm* (*software process*)
- Tiến trình chung cho phát triển phần mềm:



## 2.1. Tổng quan

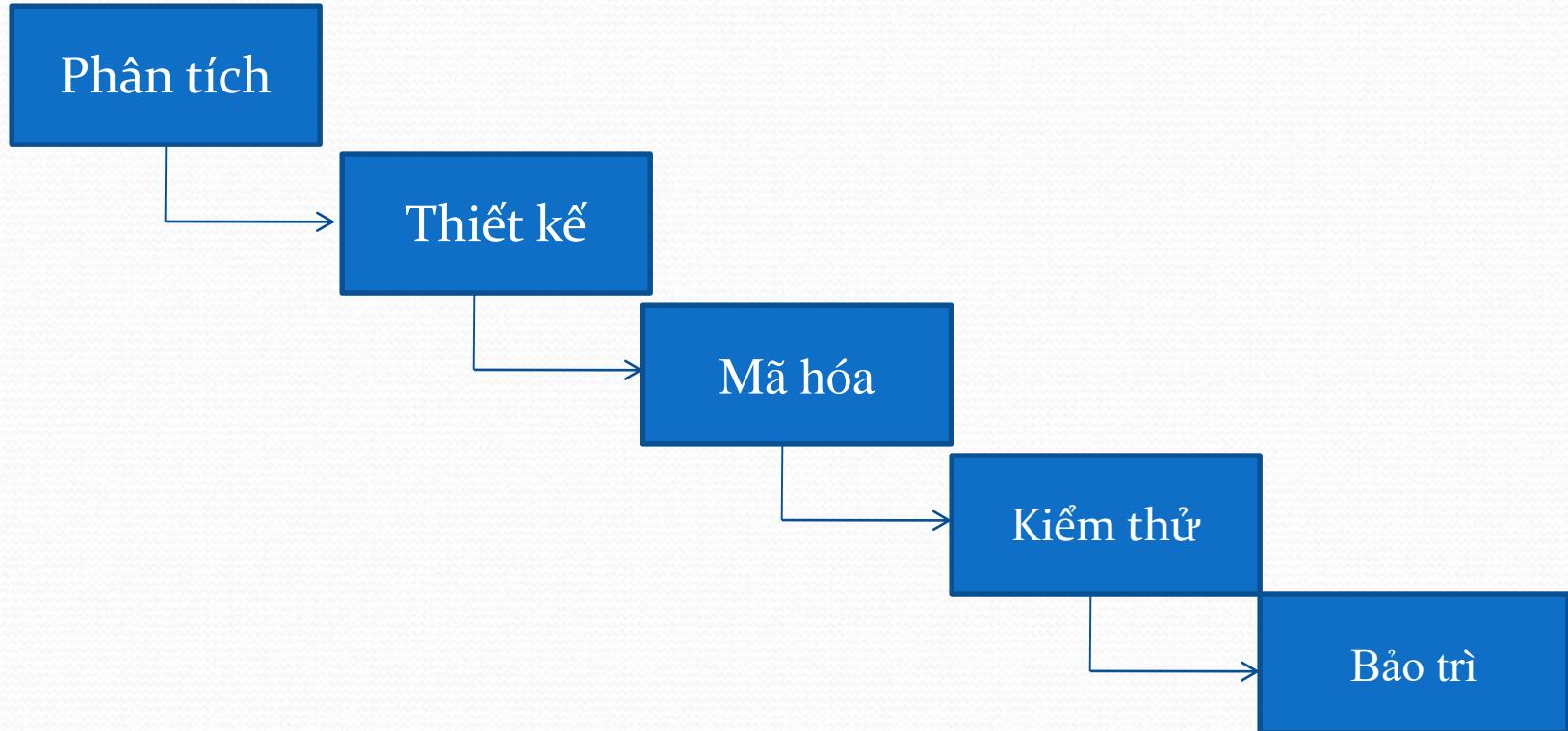
---

- *Xác định yêu cầu phần mềm*: Xác định các chức năng của phần mềm, các ràng buộc mà nó cần tuân thủ khi phát triển và hoạt động, và các đặc tả nó.
- *Phát triển phần mềm*: Tạo ra phần mềm theo đúng đặc tả
- *Thẩm định phần mềm*: Phần mềm cần được kiểm tra xem có đáp ứng yêu cầu người dùng không
- *Tiến hóa phần mềm*: phần mềm cần được tiến hóa để đáp ứng yêu cầu thay đổi của người dùng và môi trường.

# Chương 2: TIẾN TRÌNH PHẦN MỀM

## 2.2. Một số loại mô hình tiến trình phần mềm tiêu biểu

### 2.2.1 Mô hình thác nước (mô hình vòng đời truyền thống)



## 2.2.1 Mô hình thác nước

---

- *Phân tích*: tập trung vào việc thu thập phân tích các thông tin cần cho phần mềm, các chức năng phần mềm cần phải thực hiện, hiệu năng cần có của mỗi chức năng, các giao diện cho người sử dụng, các ràng buộc mà phần mềm cần tuân thủ...
- *Thiết kế*: là quá trình chuyển hóa các yêu cầu phần mềm thành các mô tả thiết kế mà từ đó nhà phát triển và lập trình có thể lắp đặt hệ thống và chuyển thiết kế thành chương trình vận hành được.
- *Mã hóa*: Dịch các đặc tả thiết kế thành các chương trình mã nguồn trong một ngôn ngữ lập trình nào đó mà mã mây có thể thực hiện được.
- *Kiểm thử*: Phát hiện lỗi và sửa lỗi trong chương trình.
- *Bảo trì*: trong môi trường hoạt động thực tế, hệ thống có đáp ứng được các yêu cầu đặt ra ban đầu và đề xuất những thay đổi, bổ sung để hoàn thiện hệ thống.

- **Ưu điểm và nhược điểm của mô hình thác nước?**
  - Thực tế các dự án ít khi tuân theo dòng tuần tự của mô hình, mà thường có lặp lại (như mô hình của Boehm)
  - Khách hàng ít khi tuyên bố rõ ràng khi nào xong hết các yêu cầu
  - Khách hàng phải có lòng kiên nhẫn chờ đợi thời gian nhất định mới có sản phẩm. Nếu phát hiện ra lỗi nặng thì là một thảm họa!

## 2.2.2 Các mô hình phát triển tiến hóa

---

- Phần lớn các hệ phần mềm phức tạp đều tiến hóa theo thời gian: môi trường thay đổi, yêu cầu phát sinh thêm, hoàn thiện thêm chức năng, tính năng.
- Các mô hình tiến hóa (evolutionary models) có tính lặp lại. Kỹ sư phần mềm tạo ra các phiên bản ngày càng hoàn thiện hơn, phức tạp hơn.
- Các mô hình:

## a. Mô hình gia tăng

---

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

# a. Mô hình gia tăng

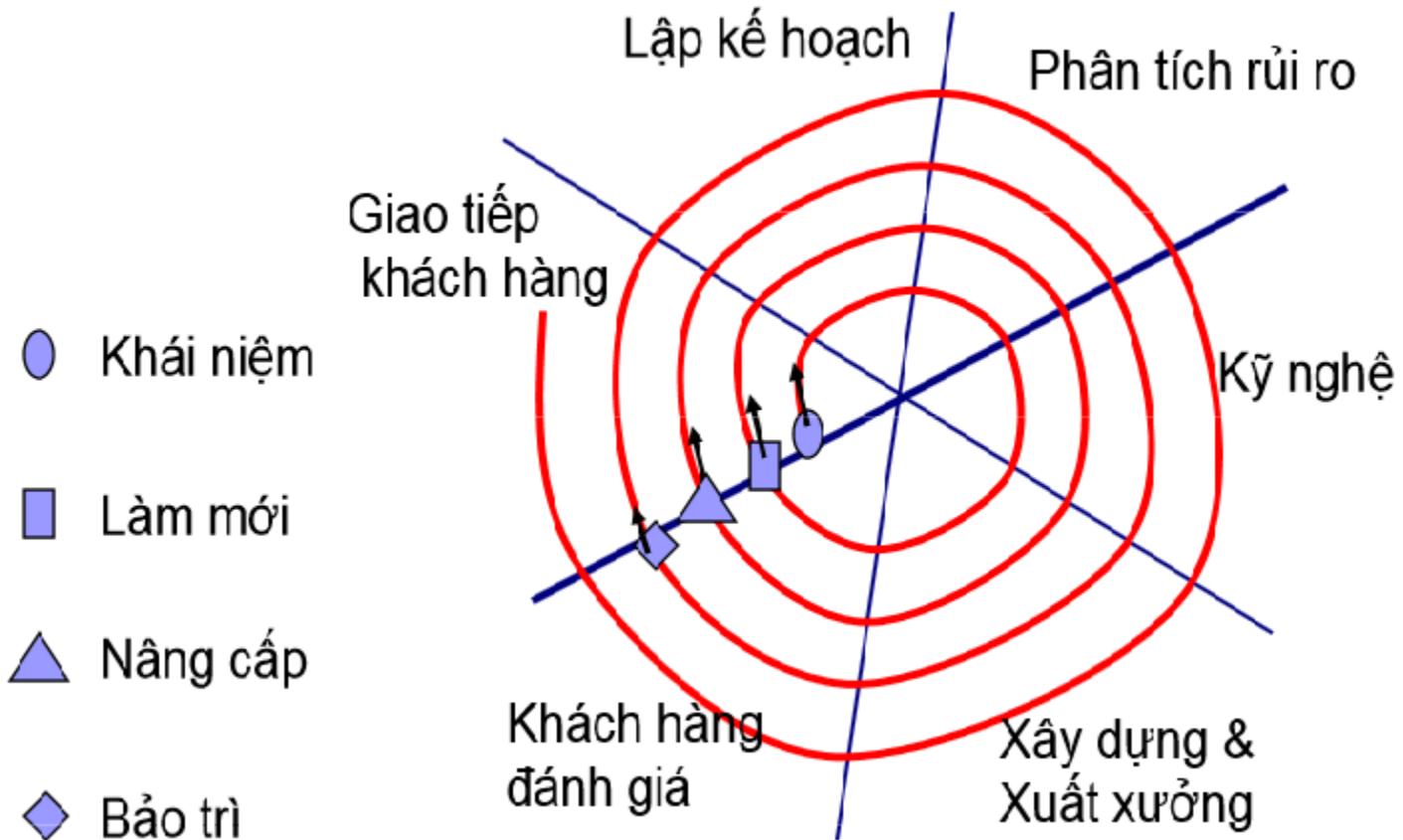


# a. Mô hình gia tăng

---

- **Ưu, nhược điểm của mô hình?**
- Sau mỗi lần tăng vòng thì có thể chuyển giao kết quả thực hiện được cho khách hàng nên các chức năng của hệ thống có thể nhìn thấy sớm hơn.
- Các vòng trước đóng vai trò là mẫu thử để giúp tìm hiểu thêm các yêu cầu ở những vòng tiếp theo.
- Những chức năng của hệ thống có thứ tự ưu tiên càng cao thì sẽ được kiểm thử càng kỹ.

## b. Mô hình xoắn ốc (Spiral)



## b. Mô hình xoắn ốc (Spiral)

---

- *Giao tiếp khách hàng*: giữa người phát triển và khách hàng để tìm hiểu yêu cầu, ý kiến
- *Lập kế hoạch*: Xác lập tài nguyên, thời hạn và những thông tin khác.
- *Phân tích rủi ro*: xem xét mạo hiểm kỹ thuật và mạo hiểm quản lý
- *Kỹ nghệ*: xây dựng một hay một số biểu diễn của ứng dụng.
- *Xây dựng và xuất xưởng*: xây dựng, kiểm thử, cài đặt và cung cấp hỗ trợ người dùng (tư liệu, huấn luyện...)
- *Đánh giá của khách hàng*: nhận các phản hồi của người sử dụng về biểu diễn phần mềm trong giai đoạn kỹ nghệ và cài đặt.

## b. Mô hình xoắn ốc (Spiral)

---

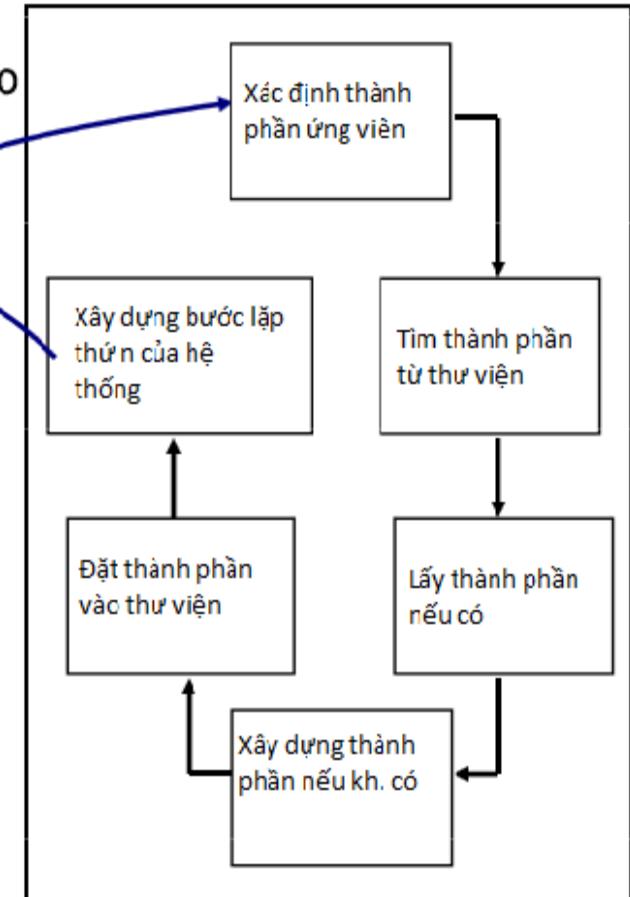
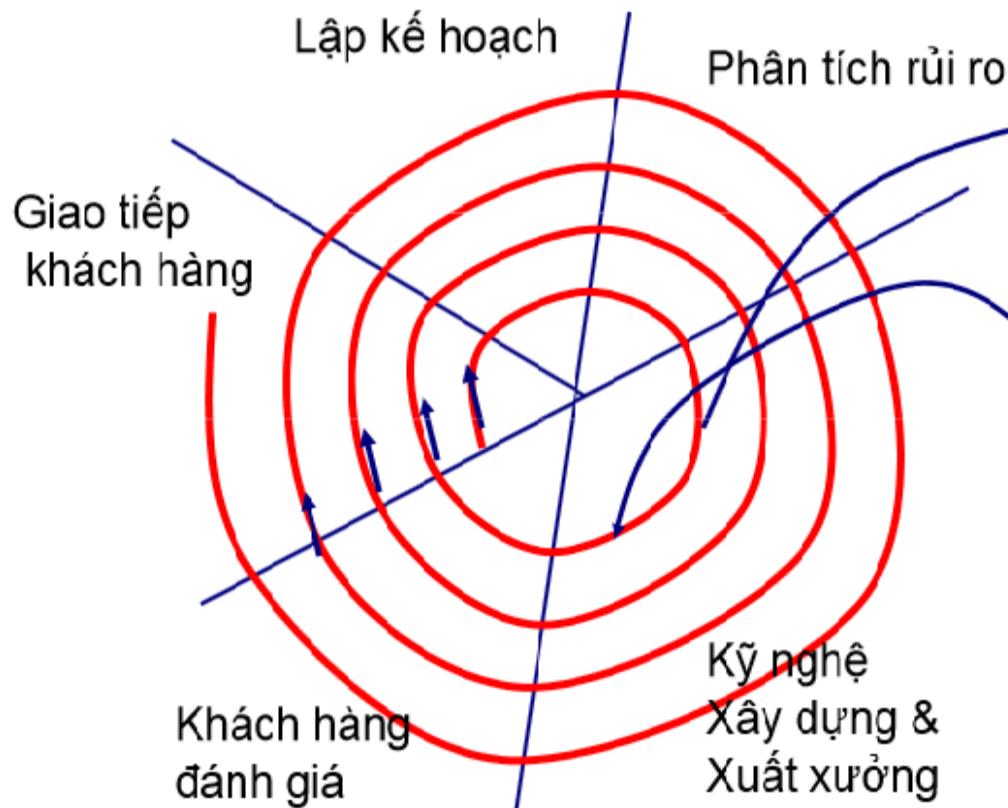
- Ưu và nhược điểm của Mô hình xoắn ốc.
  - Tốt cho các hệ phần mềm quy mô lớn
  - Dễ kiểm soát các mạo hiểm ở từng mức tiến hóa
  - Khó thuyết phục khách hàng là phương pháp tiến hóa xoắn ốc có thể kiểm soát được
  - Chưa được dùng rộng rãi như các mô hình tuyến tính hoặc chế thử

## c. Mô hình phát triển đồng thời

---

- Xác định mạng lưới những hoạt động đồng thời
- Các sự kiện xuất hiện theo điều kiện vận động trạng thái trong từng hoạt động.
- Dùng cho mọi ứng dụng và cho hình ảnh khá chính xác về trạng thái hiện trạng của dự án
- Thường dùng trong phát triển các ứng dụng khách/chủ.

# d. Mô hình theo thành phần



## d. Mô hình theo thành phần

---

- Gắn với những công nghệ hướng đối tượng có chứa cả dữ liệu và giải thuật xử lý dữ liệu.
- Có nhiều tương đồng với mô hình xoắn ốc.
- Với ưu điểm tái sử dụng các thành phần qua thư viện/ kho các lớp tiết kiệm 70% thời gian, 80% giá thành.

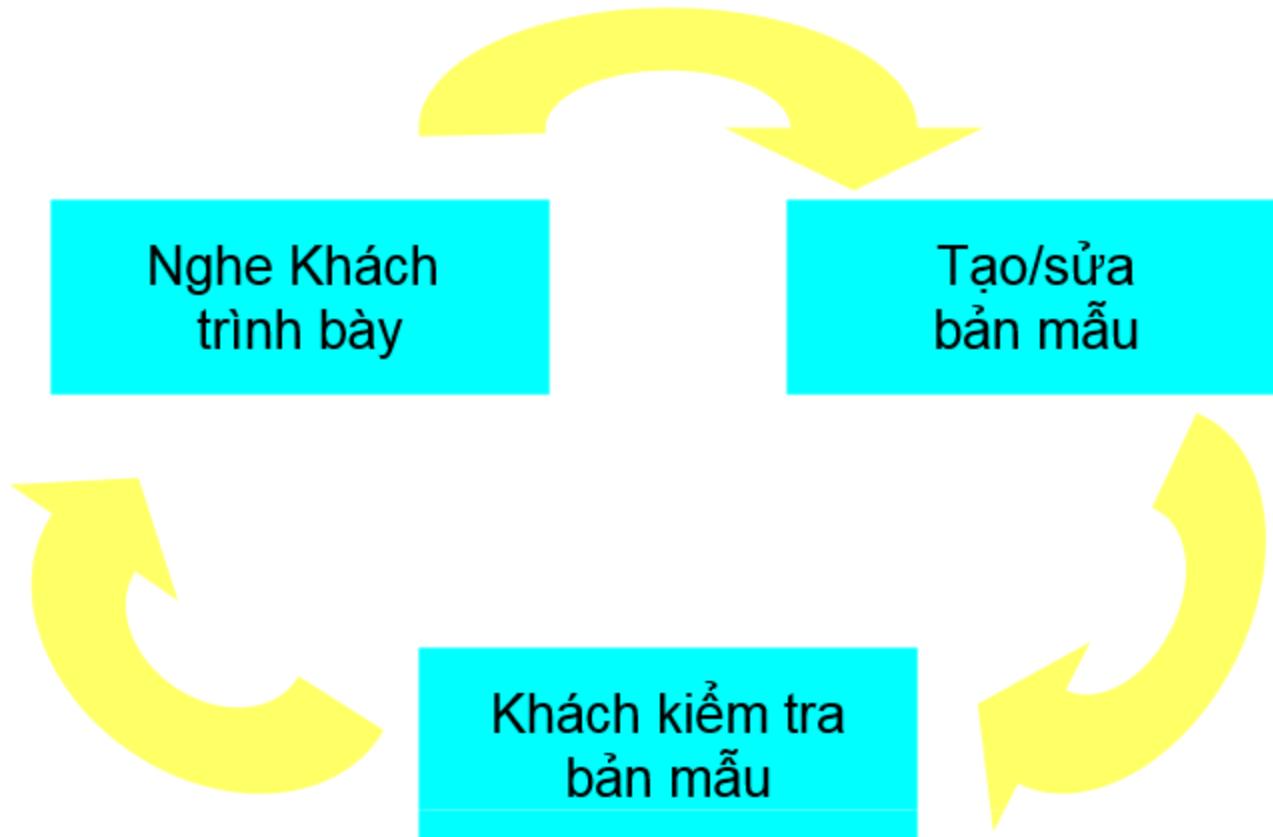
## 2.2.3 Các mô hình phát triển hình thức

---

- Tập hợp các công cụ nhằm đặc tả toán học phần mềm máy tính từ khâu định nghĩa, phát triển đến kiểm chứng.
- Giúp kỹ sư phần mềm phát hiện và sửa các lỗi khó
- Thường dùng trong phát triển phần mềm cần độ an toàn rất cao (y tế, hàng không...)
- *Nhược điểm:*
  - Cần thời gian và công sức để phát triển
  - Phí đào tạo cao vì ít người có nền cản bản cho áp dụng mô hình hình thức
  - Khó sử dụng rộng rãi vì cần kiến thức toán và kỹ năng của khách hàng.

## 2.2.4. Một số mô hình khác

### a. Mô hình chẽ thử



# Mô hình chế thử

---

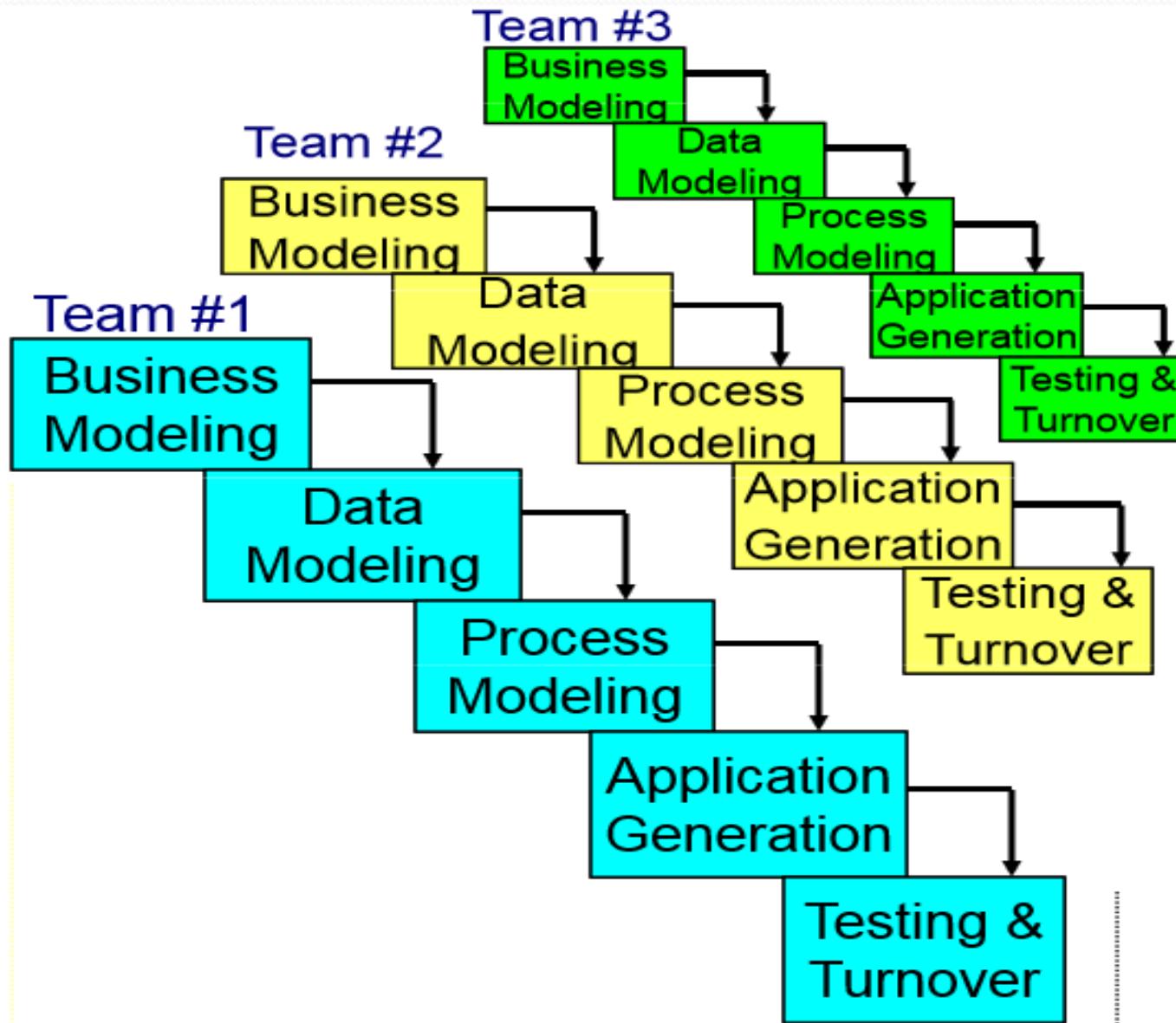
- Khi mới rõ mục đích chung chung của phần mềm, chưa rõ chi tiết đầu vào hay xử lý ra sao hoặc chưa rõ yêu cầu đầu ra.
- Dùng như “hệ sơ khai” để thu thập yêu cầu người dùng qua các thiết kế nhanh.
- Các giải thuật, kỹ thuật dùng làm bản mẫu có thể chưa nhanh, chưa tốt, miễn là có mẫu để thảo luận gợi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

## b. Mô hình phát triển ứng dụng nhanh(Rapid application development:RAD)

---

- Là quy trình phát triển phần mềm gia tăng, tăng dần từng bước với mỗi chu trình phát triển rất ngắn (60-90 ngày)
- Xây dựng dựa trên hướng thành phần với khả năng tái sử dụng.
- Gồm một nhóm, mỗi nhóm làm một RAD theo pha: mô hình nghiệp vụ, mô hình dữ liệu, mô hình xử lý, tạo ứng dụng, kiểm thử và đánh giá(Business, Data, Process, Appl, Reneration, Test)

# Mô hình phát triển ứng dụng nhanh



# RAD: Business Modeling(Mô hình nghiệp vụ)

---

- Luồng thông tin được mô hình hóa để trả lời các câu hỏi:
  - Thông tin nào điều khiển xử lý nghiệp vụ
  - Thông tin gì được sinh ra
  - Ai sinh ra nó
  - Thông tin đi đến đâu
  - Ai xử lý chúng

# RAD: Data (Mô hình dữ liệu) and process modeling( Mô hình xử lý)

---

- Data modeling: các đối tượng dữ liệu cần để hỗ trợ nghiệp vụ. Định nghĩa các thuộc tính của từng đối tượng và xác lập quan hệ giữa các đối tượng.
- Process modeling: các đối tượng dữ liệu được chuyển sang luồng thông tin thực hiện chức năng nghiệp vụ. Tạo mô tả xử lý để cập nhật (thêm, sửa, xóa, khôi phục) từng đối tượng dữ liệu.

# RAD: appl( Mô hình ứng dụng) Generation and testing ( Đánh giá và kiểm thử)

---

- Application Generation: dùng các kỹ thuật thế hệ 4 để tạo phần mềm từ các thành phần có sẵn hoặc tạo ra các thành phần có thể tái sử dụng lại sau này. Dùng các công cụ tự động để xây dựng phần mềm
- Testing and turnover: kiểm thử các thành phần mới và kiểm chứng mọi giao diện(các thành phần cũ đã được kiểm thử và dùng lại).

# RAD: Hạn chế

---

- Cần có nguồn nhân lực dồi dào để tạo các nhóm cho các chức năng chính
- Yêu cầu hai bên phải giao kèo trong thời gian ngắn phải có phần mềm hoàn chỉnh, thiếu trách nhiệm của một bên dễ làm dự án đổ vỡ.
- RAD không phải tốt cho mọi ứng dụng, nhất là với ứng dụng không thể modun hóa hoặc đòi hỏi tính nâng cao.
- Mạo hiểm kỹ thuật cao thì không nên dùng RAD

# Chương 3: Phân tích đặc tả yêu cầu

---

**Câu hỏi:** Chất lượng phần mềm là gì?

- + Chất lượng phần mềm là sự đáp ứng các yêu cầu chức năng, sự hoàn thiện và các chuẩn (đặc tả) được phát triển, các đặc trưng mong chờ từ mọi phần mềm chuyên nghiệp (ngầm định).
- + Chất lượng phần mềm cần xem xét như thế nào?
  - Phần mềm là vô hình
  - Phần mềm không mòn cũ, hỏng hóc nhưng thoái hóa.
  - Phần mềm thay đổi theo thời gian

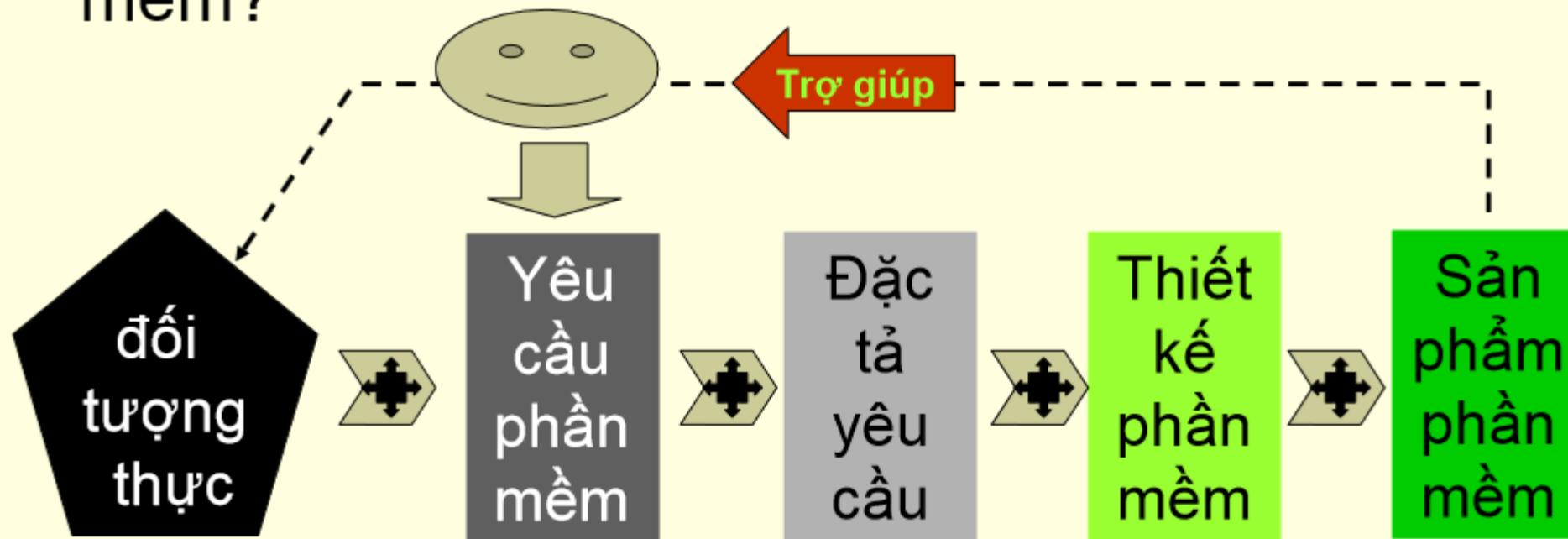
# Câu hỏi: Chất lượng phần mềm là gì?

---

- Với phần mềm có một số vấn đề sau:
  - Yêu cầu có thể bị bỏ sót
  - Các yêu cầu tự nhiên nên không được đặc tả
  - Phần mềm có yêu cầu mà chưa có đặc tả
  - Phần mềm có đặc tả nhưng lại mù mờ

# Câu hỏi: Chất lượng phần mềm là gì?

- Cái gì là **Cơ sở** xem xét chất lượng phần mềm?



**Sự hình thành sản phẩm phần mềm bắt đầu từ yêu cầu**

# Yêu cầu phần mềm là cơ sở xem xét:

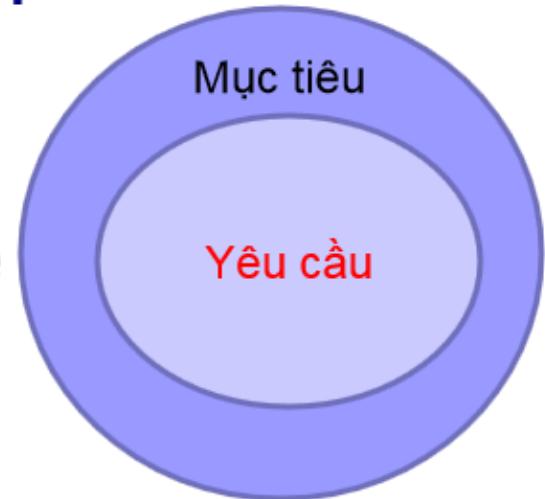
---

- Xem xét chất lượng:
- Yêu cầu thể hiện ra bằng đặc tả- đặc tả có chuẩn mới kiểm tra, đo đạc được:
  - + Các chuẩn đặc tả là một bộ các tiêu chuẩn phát triển và hướng dẫn cách thức làm ra phần mềm
  - + Không tuân thủ các tiêu chuẩn đó thì chắc chắn là chất lượng sẽ kém.

# Mục tiêu và yêu cầu phần mềm

## Mục tiêu và Yêu cầu của PM

- Mục tiêu: cái cần hướng tới
- Yêu cầu: cái cụ thể mà có thể kiểm tra được
- Yêu cầu chức năng
  - mô tả một chức năng (dịch vụ) cụ thể mà phần mềm cần cung cấp
- Yêu cầu phi chức năng
  - Các ràng buộc về chất lượng, về môi trường, chuẩn sử dụng, qui trình phát triển phần mềm



# Mục tiêu và yêu cầu phần mềm

## ■ Yêu cầu về sản phẩm

- Tốc độ, độ tin cậy, bộ nhớ, giao diện, qui trình  
tác nghiệp,...

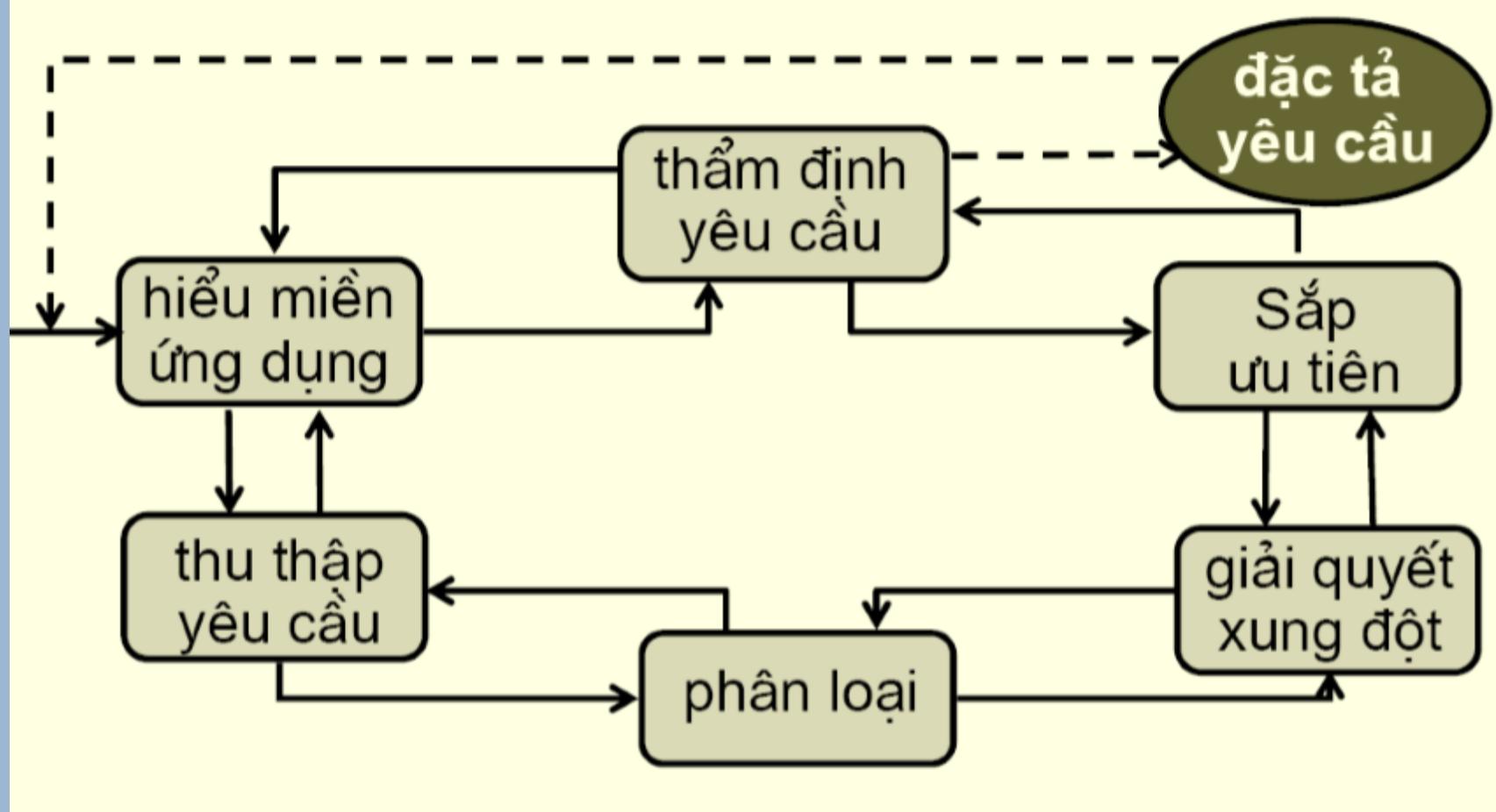
## ■ Yêu cầu về tiến trình phát triển

- các chuẩn, phương pháp thiết kế, ngôn ngữ  
lập trình...

## ■ Yêu cầu ngoại lai

- Về chi phí, về thời gian, về bản quyền, ...

# Tiến trình phân tích yêu cầu



# Những vấn đề liên quan đến đặc tả:

---

- Bỏ sót yêu cầu
- Có các yêu cầu gầm thường ít được nhắc đến (quá thông dụng hiển nhiên, ít được thể hiện ra ngoài)  
*Phần mềm chưa phù hợp với các yêu cầu ngầm thì chất lượng cũng đáng ngờ.*
- Có những sai sót khi đặc tả

# 1. Khái niệm phân tích đặc tả yêu cầu

---

- Phân tích đặc tả yêu cầu là tiến trình xác định:
  - + Các dịch vụ chức năng mà khách hàng yêu cầu từ hệ thống
  - + Các ràng buộc mà hệ thống được phát triển và vận hành

# 1. Khái niệm phân tích đặc tả yêu cầu

---

- Yêu cầu là: Năng lực của phần mềm mà người sử dụng cần để giải quyết vấn đề đặt ra nhằm đạt được mục đích xác định.
  - + Năng lực phần mềm cần có nhằm thỏa mãn một hợp đồng, một chuẩn, một đặc tả.
  - > *Yêu cầu cho một hệ thống phần mềm mô tả những công việc mà hệ thống làm và những ràng buộc mà nó phải tuân thủ khi hoạt động. Yêu cầu có thể là yêu cầu chức năng(các chức năng, dịch vụ) hay yêu cầu phi chức năng (các ràng buộc)*

# Các yêu cầu phân theo mức trừu tượng:

---

- *Yêu cầu người sử dụng:*
  - + Các phát biểu bằng ngôn ngữ tự nhiên, các sơ đồ về dịch vụ và ràng buộc mà hệ thống cung cấp
  - + Dành cho khách hàng
- *Yêu cầu hệ thống:*
  - + Tài liệu có cấu trúc mô tả chi tiết các dịch vụ của hệ thống
  - + Là hợp đồng giữa khách hàng và người phát triển
- *Đặc tả phần mềm:*
  - + Mô tả chi tiết về phần mềm nhằm phục vụ cho thiết kế, mã hóa.
  - + dành cho người phát triển

# Người đọc của những tài liệu đặc tả khác nhau

---

- *Yêu cầu người sử dụng:*

1. Người quản lý của khách hàng
2. Người quản lý thầu
3. Người dùng hệ thống
4. Kỹ sư
5. Người thiết kế hệ thống

# Người đọc của những tài liệu đặc tả khác nhau

---

- *Yêu cầu hệ thống*
  1. Người dùng hệ thống
  2. Kỹ sư
  3. Người thiết kế hệ thống
  4. Người phát triển phần mềm

# Người đọc của những tài liệu đặc tả khác nhau

- *Đặc tả phần mềm*
  1. Kỹ sư
  2. Người thiết kế hệ thống
  3. Người phát triển phần mềm

# Phân loại các yêu cầu phần mềm

---

- *Các yêu cầu chức năng:* là những phát biểu về chức năng hay dịch vụ mà hệ thống cung cấp. Yêu cầu chức năng cũng có thể là chức năng mà hệ thống không nên thực hiện.
- *Các yêu cầu phi chức năng:* những ràng buộc lên các dịch vụ, hoặc chức năng mà hệ thống cung cấp. Nó bao gồm các ràng buộc về thời gian, về ngân sách, các ràng buộc trong quá trình phát triển hay các ràng buộc về chuẩn sử dụng...
- *Các yêu cầu miền lĩnh vực:* là các yêu cầu xuất phát từ miền ứng dụng, phản ánh đặc trưng của lĩnh vực ứng dụng này (các quy tắc nghiệp vụ). Có thể là chức năng hay phi chức năng.

# Yêu cầu chức năng

---

- Mô tả các hoạt động của hệ thống hay các dịch vụ mà hệ thống sẽ cung cấp. Những chức năng này phụ thuộc vào loại phần mềm được xây dựng, vào sự mong đợi của người dùng, vào quy mô hệ thống được phát triển và môi trường mà ở đó nó được cài đặt.

# Yêu cầu phi chức năng

- Là các yêu cầu không liên quan trực tiếp tới các hoạt động chức năng cụ thể của hệ thống.
- Chúng liên quan tới thuộc tính quan trọng của hệ thống, hiệu năng của hệ thống như: độ tin cậy, thời gian đáp ứng của dịch vụ, việc sử dụng bộ nhớ hay đến các ràng buộc lên hệ thống như khả năng vào/ra của thiết bị, sự biểu diễn dữ liệu trên các giao diện của hệ thống...

# Các kiểu khác nhau của yêu cầu phi chức năng

---

- *Yêu cầu về sản phẩm:* yêu cầu về thời gian phải tra lại kết quả, tính tin cậy...
- *Yêu cầu mang tính tổ chức:* Yêu cầu về ngôn ngữ lập trình, phương pháp thiết kế được sử dụng...
- *Yêu cầu mở rộng:* yêu cầu về tính pháp lý, tính văn hóa...

# Yêu cầu miền lĩnh vực

---

- Có thể là những yêu cầu chức năng mới về quyền sở hữu, những hạn chế đối với các yêu cầu chức năng đang tồn tại... Các yêu cầu miền là quan trọng vì nó phản ánh những cơ sở nền tảng của miền ứng dụng, nếu các yêu cầu đó không được thỏa mãn hệ thống có thể sẽ không hoạt động một cách đúng đắn.

# Tiến trình phân tích yêu cầu

---

- 1. Nghiên cứu khả thi
- 2. Phát hiện và phân tích yêu cầu
- 3. Đặc tả yêu cầu
- 4. Thẩm định yêu cầu

# Tiến trình phân tích yêu cầu

---

- **Nghiên cứu khả thi:**
  - Đánh giá khả thi của phương án phải dựa vào phần mềm và công nghệ phần cứng hiện có có thỏa mãn nhu cầu người dùng hay không.
  - Việc nghiên cứu khả thi sẽ quyết định đưa ra một hệ thống đáp ứng được yêu cầu khách hàng, đảm bảo về công nghệ, có giá cả phải chăng có thể thực hiện được trong điều kiện ngân sách và thời hạn định trước.

# Tiến trình phân tích yêu cầu

---

- **Phát hiện và phân tích yêu cầu**
  - Nhà phát triển tiếp tục khảo sát hệ thống đang tồn tại để thu thập thông tin, làm việc với khách hàng và người dùng cuối để nắm và hiểu miền ứng dụng và chuyển thông tin thu thập được từ hoạt động phân tích thành tài liệu xác định các yêu cầu.
  - Các tài liệu này phải phản ánh chính xác mong muốn của khách hàng, được viết sao cho người dùng và khách hàng đặt mua hệ thống có thể hiểu được

# Tiến trình phân tích yêu cầu

---

- **Đặc tả yêu cầu**
- Là mô tả chính xác và chi tiết yêu cầu hệ thống để làm cơ sở cho giao kèo giữa khách hàng và người phát triển hệ thống.
- Sử dụng các công cụ, mô hình, ngôn ngữ đặc tả chuyên dụng
- Thường được tiến hành song song với các thiết kế mức cao
- Các sai sót trong xác định yêu cầu sẽ được phát hiện và sửa chữa.

# Tiến trình phân tích yêu cầu

---

- **Thẩm định yêu cầu**
- Xét xem đặc tả yêu cầu có mô tả chính xác những gì được đặt ra cho hệ thống và có thể thực hiện được không.
- Một số thuộc tính của thẩm định như: tính đúng đắn, tính đầy đủ, tính nhất quán, tính hiện thực.

# Tài liệu yêu cầu

- Yêu cầu của hệ thống được biểu diễn bằng tài liệu yêu cầu phần mềm hay đặc tả yêu cầu phần mềm. Nó là tài liệu chính thức cho người phát triển phần mềm
- Bao gồm tài liệu xác định yêu cầu và tài liệu đặc tả yêu cầu.
- Sáu yêu cầu cho một tài liệu yêu cầu phần mềm:
  - + Mô tả các hoạt động của hệ thống từ bên ngoài
  - + Các ràng buộc của hệ thống trong quá trình vận hành
  - + Dễ thay đổi
  - + Phục vụ như tài liệu tham khảo cho người bảo trì hệ thống
  - + Dự toán trước vòng đời của hệ thống
  - + Mô tả được các đáp ứng đối với những sự cố, thay đổi ngoài dự tính.

# Tài liệu yêu cầu

---

- **Cấu trúc tài liệu yêu cầu phần mềm**

- 1. Giới thiệu***

- Mục tiêu của tài liệu yêu cầu
  - Phạm vi của sản phẩm
  - Các định nghĩa, chữ viết tắt và từ viết rút gọn
  - Tài liệu tham khảo
  - Tổng quan về tài liệu

# Cấu trúc tài liệu yêu cầu phần mềm

---

- 2. *Mô tả chung*
  - Các đặc điểm của sản phẩm
  - Các chức năng của sản phẩm
  - Các đặc trưng của người dùng
  - Các ràng buộc chung
  - Các giả định và các sự phụ thuộc
- 3. *Các yêu cầu cụ thể*
- 4. *Phụ lục*
- 5. *Các chỉ số chỉ dẫn*

*Tài liệu yêu cầu có thể thêm các chương hay phụ lục với các thông tin: Kiến trúc chung của hệ thống phần cứng, cơ sở dữ liệu, chú dẫn*

# Xác định yêu cầu phần mềm

---

- **1. Khảo sát hệ thống và phân tích khả thi**
  - Khả thi về kinh tế
  - Khả thi về kinh tế
  - Khả thi về kỹ thuật
  - Khả thi về pháp lý
  - Khả thi về hoạt động
  - Khả thi về thời gian

# Xác định yêu cầu phần mềm

- **2. Phát hiện và phân tích yêu cầu**
- **Những khó khăn trong việc nhận ra yêu cầu**
- Người liên quan thường không thực sự biết mình cần gì từ hệ thống, họ thường bày tỏ những yêu cầu của mình theo cách nói riêng.
- Họ có thể có cách yêu cầu khác nhau mà các kỹ sư phải nhận ra điểm chung cũng như điểm khác biệt giữa chúng.
- Quá trình phân tích diễn ra trong bối cảnh cụ thể của tổ chức nên các yêu cầu có thể bị ảnh hưởng bởi các yếu tố chính trị.
- Môi trường kinh doanh luôn biến động do vậy yêu cầu mới có thể xuất hiện từ những người liên quan mà lúc đầu không được tham khảo.
- Khó khăn có tính nguyên tắc của việc thiết lập các yêu cầu hệ thống phần mềm, các vấn đề không được định nghĩa, không có công thức cho trước.

# Xác định yêu cầu phần mềm

---

## 2. Phát hiện và phân tích yêu cầu

- **Tiến trình phát hiện và phân tích yêu cầu**
  - Tìm hiểu miền ứng dụng
  - Thu thập các yêu cầu
  - Phân loại các yêu cầu
  - Giải quyết xung đột
  - Sắp ưu tiên
  - Kiểm tra yêu cầu

# Các kỹ thuật phân tích yêu cầu

---

## 1. Tiếp cận yêu cầu hướng cách nhìn

Mỗi cách nhìn có thể được xem xét từ một trong các góc độ sau:

- Từ nguồn hay đích tới của dữ liệu
- Từ khung làm việc trình diễn
- Từ sự tiếp nhận dịch vụ

# Các kỹ thuật phân tích yêu cầu

---

- 2. Kỹ thuật xác định yêu cầu hướng cách nhin-VORD

Các giai đoạn:

- Xác định khung nhìn
- Cấu trúc khung nhìn
- Làm tài liệu khung nhìn
- Ánh xạ hệ thống - khung nhìn

# Các kỹ thuật phân tích yêu cầu

---

- **3. Kỹ thuật phân tích yêu cầu dựa trên mô hình**
  - a. *Mô hình phân tích yêu cầu – mô hình nghiệp vụ*
    - Tiếp cận hướng chức năng
    - Tiếp cận hướng đối tượng

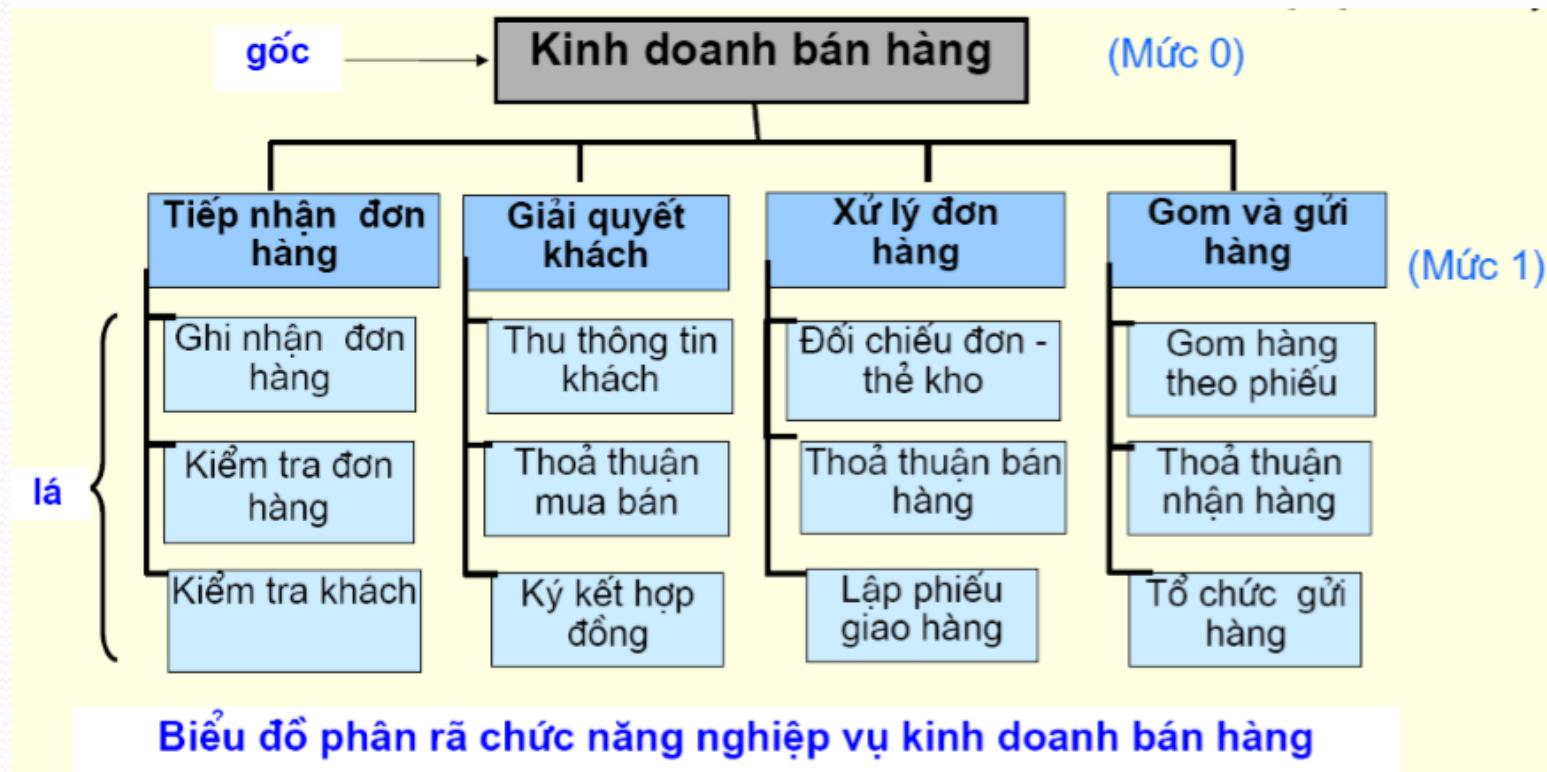
# Đặc tả yêu cầu phần mềm

---

- **Đặc tả chức năng:** thông thường khi đặc tả các chức năng của phần mềm người ta sử dụng các công cụ tiêu biểu sau:
  - Biểu đồ phân rã chức năng (Functional Decomposition Diagram – FDD)
  - Biểu đồ luồng dữ liệu (Data Flow Diagrams- DFD)
  - Máy trạng thái hữu hạn
  - Mạng Petri

# Đặc tả chức năng

- Biểu đồ phân rã chức năng (Functional Decomposition Diagram – FDD)



# Đặc tả chức năng

---

- **Ví dụ: Đặc tả hướng chức năng**

- Hệ quản lý điểm sinh viên trường đại học
- Hệ thống thông tin thư viện

Vẽ sơ đồ chức năng hệ thống Quản lý điểm sinh viên

Vẽ sơ đồ ngũ cảnh Hệ thống Quản lý điểm sinh viên

# Đặc tả chức năng \_ mô hình luồng dữ liệu

---

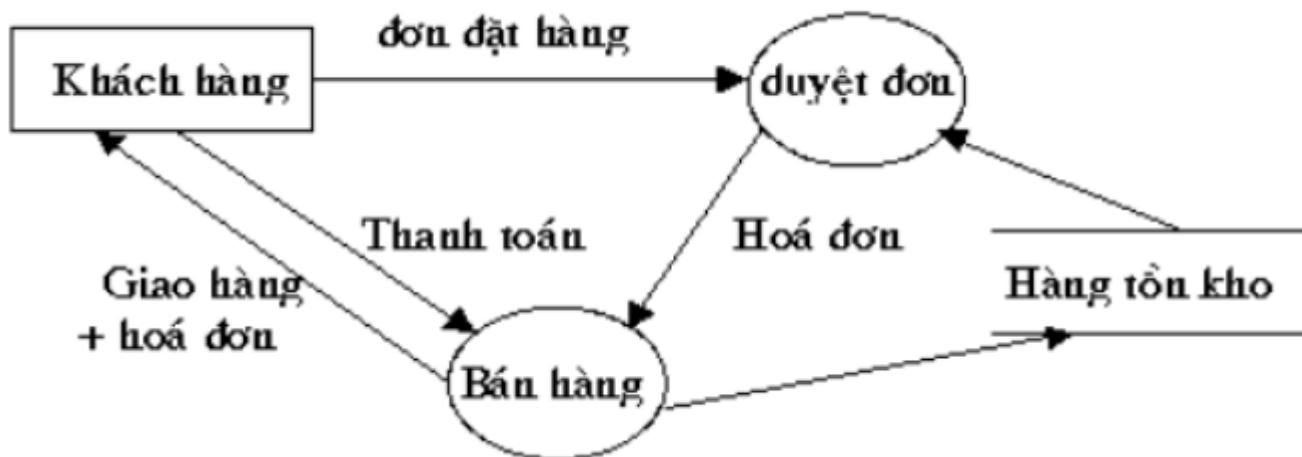
## • Mục đích của mô hình luồng dữ liệu

Mô hình luồng dữ liệu nhằm mục đích:

- Bổ sung khiếm khuyết của mô hình phân rã chức năng bằng việc bổ sung các luồng thông tin nghiệp vụ cần để thực hiện chức năng.
- Cho ta cái nhìn đầy đủ hơn về các mặt hoạt động của hệ thống
- Là một trong số các đầu vào cho quá trình thiết kế hệ thống.

# Mô hình luồng dữ liệu

- Mô hình luồng dữ liệu (DFD - Data Flow Diagram) là một công cụ mô tả mối quan hệ thông tin giữa các công việc
- Ví dụ: Mô hình luồng dữ liệu của hoạt động bán hàng



# Mô hình luồng dữ liệu

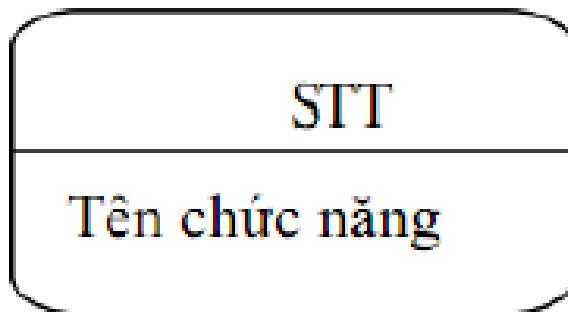
---

- **Các thành phần của mô hình luồng dữ liệu**
- **a. Chức năng (còn gọi là Tiến trình)**
- - Định nghĩa: Là một hoạt động có liên quan đến sự biến đổi hoặc tác động lên thông tin như tổ chức lại thông tin, bổ sung thông tin hoặc tạo ra thông tin mới. Nếu trong một chức năng không có thông tin mới được sinh ra thì đó chưa phải là chức năng trong mô hình luồng dữ liệu.
- - Cách đặt tên: Động từ + bổ ngữ.

# Mô hình luồng dữ liệu

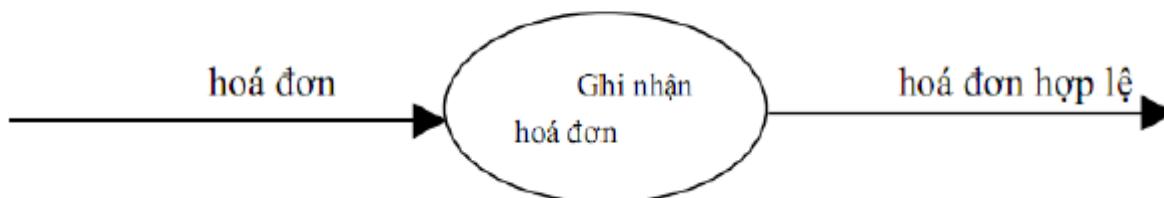
## Chức năng (còn gọi là Tiến trình)

- Biểu diễn: hình chữ nhật góc tròn hoặc hình tròn
- Chú ý : Trong thực tế tên chức năng phải trùng với tên chức năng trong mô hình phân rã chức năng.



# Mô hình luồng dữ liệu

- **b. Luồng dữ liệu:**
- - Định nghĩa: Là luồng thông tin vào hoặc ra khỏi chức năng
- - Cách đặt tên : Danh từ + tính từ
- - Biểu diễn : là mũi tên trên đó ghi thông tin di chuyển



# Mô hình luồng dữ liệu

---

- **c. Kho dữ liệu**
- Kho dữ liệu là nơi biểu diễn thông tin cần lưu giữ, để một hoặc nhiều chức năng sử dụng chúng.
- Cách đặt tên kho dữ liệu như sau : danh từ + tính từ. Tên kho phải chỉ rõ nội dung dữ liệu trong kho.
- Kho dữ liệu được biểu diễn bằng cặp đường thẳng song song chứa tên kho cần cất giữ.

---

Kho dữ liệu

---

# Mô hình luồng dữ liệu

- d. Tác nhân ngoài
  - - Định nghĩa: Là một người hoặc một nhóm người nằm ngoài hệ thống nhưng có trao đổi trực tiếp với hệ thống. Sự có mặt của các nhân tố này trên sơ đồ chỉ ra giới hạn của hệ thống, định rõ mối quan hệ của hệ thống với thế giới bên ngoài
  - - Tên : Danh từ
  - - Biểu diễn : hình chữ nhật

Khách hàng

Nhà cung cấp

# Mô hình luồng dữ liệu

---

- **e. Tác nhân trong**
- - Là một chức năng hoặc một hệ thống con của hệ thống đang xét nhưng được trình bày ở một trang khác của mô hình.
- Mọi sơ đồ luồng dữ liệu đều có thể bao gồm một số trang, thông tin truyền giữa các quá trình trên các trang khác nhau được chỉ ra nhờ kí hiệu này.
  - - Tên: động từ + bổ ngữ
  - - Biểu diễn:

# Mô hình luồng dữ liệu

- **Một số quy tắc vẽ biểu đồ luồng dữ liệu**
- Các đối tượng trong một mô hình luồng dữ liệu phải có tên duy nhất: mỗi tiến trình phải có tên duy nhất. Tuy nhiên, vì lí do trình bày cùng một tác nhân trong, tác nhân ngoài và kho dữ liệu có thể được vẽ lặp lại.
- - Các luồng dữ liệu đi vào một tiến trình phải đủ để tạo thành các luồng dữ liệu đi ra.
- - Nói chung tên luồng thông tin vào hoặc ra kho trùng với tên kho vì vậy không cần viết tên luồng. Nhưng khi ghi hoặc lấy tin chỉ tiến hành một phần kho thì lúc đó phải đặt tên cho luồng
- - Không có một tiến trình nào chỉ có cái ra mà không có cái vào. Đối tượng chỉ có cái ra thì có thể là tác nhân ngoài (nguồn)
- - Không một tiến trình nào mà chỉ có cái vào mà không có cái ra. Một đối tượng chỉ có cái vào thì chỉ có thể là tác nhân ngoài (đích)

# Ví dụ: Mô hình luồng dữ liệu

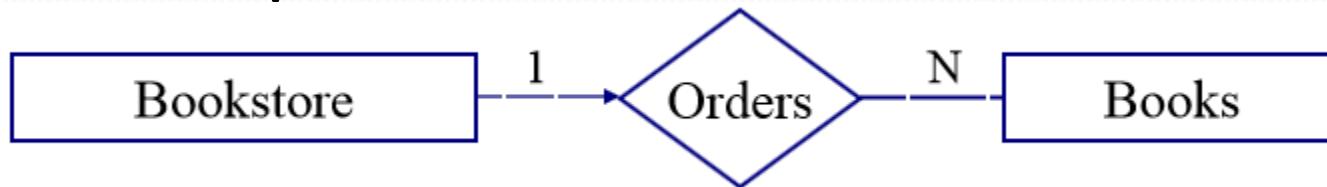
# Đặc tả yêu cầu phần mềm

---

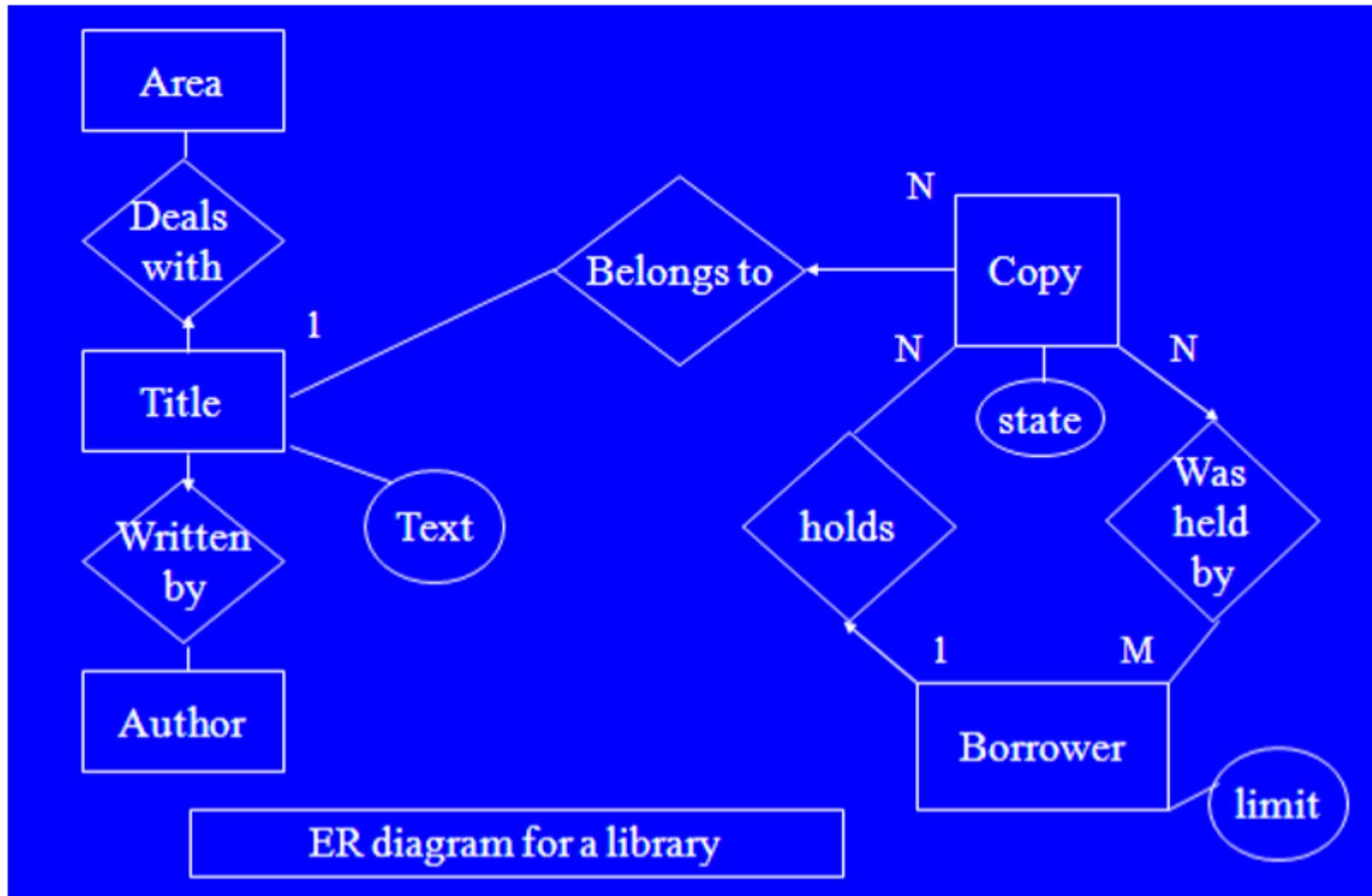
- **Đặc tả mô tả:**
  - Biểu đồ thực thể liên kết (Entity- Relationship Diagrams - ERD)
  - Đặc tả Logic (Logic Specifications)
  - Đặc tả đại số (Algebraic Specifications)

# Biểu đồ 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



# Ví dụ ERD mô tả thư viện



# Đặc tả dữ liệu hướng cấu trúc – Mô hình thực thể mối quan hệ

- **Các thành phần chính của mô hình:**
  - + Thực thể (entity) là khái niệm chỉ một lớp các đối tượng hay khái niệm (độc lập) của thế giới thực có những đặc trưng chung.
  - + Mối quan hệ (relationship) là khái niệm chỉ mối quan hệ vốn có giữa các bản thể của các thực thể.
  - + thuộc tính (attribute) là đặc trưng của thực thể hay mối quan hệ của thực thể.



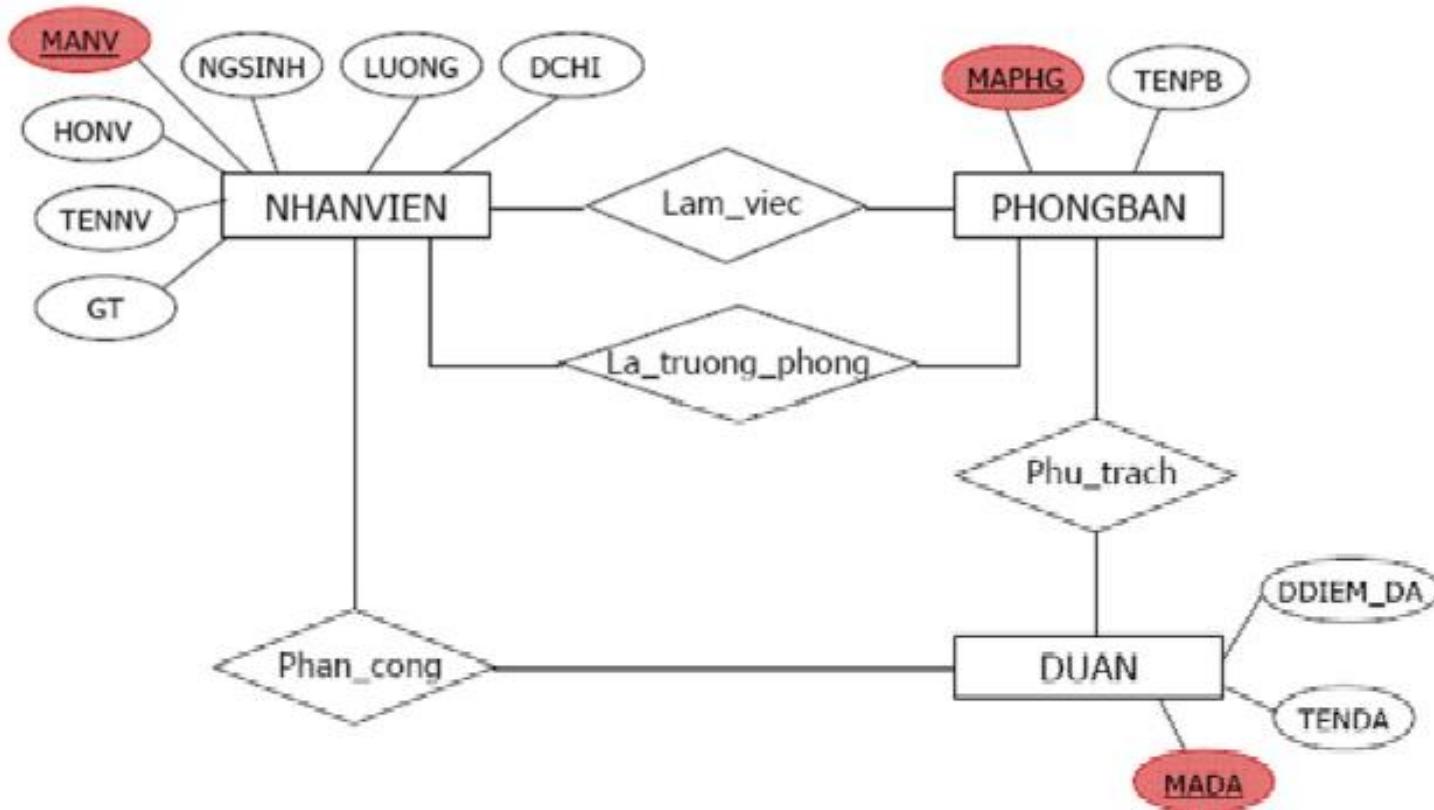
# Đặc tả dữ liệu hướng cấu trúc – Mô hình thực thể mối quan hệ

---

- **Xây dựng mô hình thực thể quan hệ (ERM)**
  - Thực thể là đối tượng chính mà ta có thông tin về chúng. Thực thể có thể là:
    - + Một người như nhân viên, sinh viên
    - + Một nơi chốn như thành phố, đất nước...
    - + Một sự kiện như đấu giá, thi...
    - + Một khái niệm như môn học, tài khoản...
  - Mỗi kiểu thực thể phải có một tên gọi, nên là **danh từ và viết chữ hoa**.

# Mô hình thực thể mối quan hệ

- Mối quan hệ (relationship)



# Mô hình thực thể mối quan hệ

- Xây dựng mô hình thực thể quan hệ (ERM)
  - Thuộc tính:
    - Mỗi kiểu thực thể có 1 số thuộc tính.
    - Thuộc tính là đặc tính của 1 kiểu thực thể hay 1 mối liên kết.
  - + Thuộc tính đơn
  - + Thuộc tính phức hợp
  - + Thuộc tính đơn trị
  - + Thuộc tính đa trị:  
Ví dụ: thuộc tính Giáo viên đa trị vì 1 môn học có thể được dạy bởi nhiều giáo viên.
  - + Thuộc tính khóa

# Mô hình thực thể mối quan hệ

## Xây dựng mô hình thực thể quan hệ (ERM)

- **Thuộc tính:**

*Cách đặt tên và ký hiệu*

- Mỗi thuộc tính nên được biểu diễn là danh từ số ít và viết chữ thường.



# Đặc tả hành vi hướng đối tượng

## – Biểu đồ phân tích tương tác

- SV Xem lại biểu đồ phân tích tương tác

# Các loại đặc tả yêu cầu

---

- **Đặc tả đại số**
- Đặc tả đại số gồm 4 phần:
  - + Phần giới thiệu: gồm các đối tượng là các thực thể được đặc tả
  - + Phần mô tả: gồm các toán tử được mô tả không hình thức, làm đặc tả hình thức dễ hiểu hơn. Nó cung cấp cú pháp và ngữ nghĩa cho toán tử kiểu
  - + Phần chữ ký: xác định cú pháp của giao diện đối với lớp đối tượng hay kiểu dữ liệu trừu tượng.
  - + phần phương trình định nghĩa: xác định ngữ nghĩa của toán tử bằng cách xác định tập hợp các phương trình định nghĩa đặc trưng hành vi của các kiểu dữ liệu trừu tượng.

# Các loại đặc tả yêu cầu

---

- Máy trạng thái
- So sánh các loại hình đặc tả

# Các loại đặc tả yêu cầu

## • So

Phương pháp đặc tả	Loại hình	Mặt mạnh	Mặt yếu
Ngôn ngữ tự nhiên	Không hình thức hóa	Dễ học, dễ sử dụng, dễ hiểu đối với kH	Không chính xác, mù mờ, mâu thuẫn, không đầy đủ
Mô hình thực thể mỗi quan hệ -PSL/DSA -SADT -SCREM -Phân tích hệ thống có cấu trúc	Bán hình thức hóa	-Khách hàng có thể hiểu được -Chính xác hơn phương pháp không hình thức -Có thể tự động hóa phần (Nhờ CASE) -Cho kết quả nhanh	- Mức chính xác chưa cao, không thể điều khiển được thời gian.
-Ann -CSP -Máy trạng thái hữu hạn -Gist -Lưới Petri -VDM	Hình thức hóa	-Tuyệt đối chính xác, có thể làm giảm lỗi đáng kể -Có thể giảm chi phí và công sức phát triển -Có thể trợ giúp chứng minh tính đúng đắn	Khó học, khó sử dụng, khách hàng khó hiểu được.

# Thẩm định yêu cầu

- **Khái niệm thẩm định yêu cầu**
  - Thẩm định yêu cầu là xét xem đặc tả yêu cầu có thực sự xác định được hệ thống mà khách hàng mong muốn hay không. Nó khác với phân tích yêu cầu ở chỗ là đặc biệt chú trọng đến một số tính chất của đặc tả có ảnh hưởng nghiêm trọng đến hoạt động thiết kế và bảo trì hệ thống sau này.
  - Thẩm định yêu cầu là quan trọng, vì nếu việc thẩm định không tốt sẽ sót lỗi. Các lỗi còn sót lại từ yêu cầu sẽ lan truyền sang giao đoạn thiết kế và cài đặt hệ thống.

# Thẩm định yêu cầu

- Khi thẩm định các mặt sau đây của yêu cầu cần phải được kiểm tra:
  - *Tính đúng đắn:* hệ thống cần bao quát được các nhu cầu khác nhau của người dùng và yêu cầu phải đáp ứng được cả cộng đồng người sử dụng.
  - *Tính đầy đủ:* phải xác định được mọi chức năng và ràng buộc mà người sử dụng hệ thống mong muốn.
  - *Tính nhất quán:* mọi yêu cầu đều không được xung đột với bất kỳ một yêu cầu nào khác.
  - *Tính hiện thực:* không có một điểm nào trong đặc tả yêu cầu là không thể thực hiện được.
  - *Tính kiểm tra được:* yêu cầu phải viết sao cho có thể kiểm tra được. Điều đó có nghĩa là có một tập các kiểm tra được thiết kế để có thể chỉ ra rằng sản phẩm gửi khách hàng là đáp ứng các yêu cầu đó.

# Thẩm định yêu cầu

## • Các kỹ thuật thẩm định yêu cầu

- a. Rà soát yêu cầu: Các yêu cầu được rà soát phân tích một cách có hệ thống
- Rà soát yêu cầu là công việc thủ công, đòi hỏi nhiều người đọc: từ khách hàng, người dự thầu, các nhân viên chuyên rà soát.
- Rà soát yêu cầu có thể là chính thức hoặc không chính thức.
- Để rà soát yêu cầu một cách chính thức, người sử dụng phải duyệt các yêu cầu của hệ thống, đội phát triển phải giải thích nội dung các thể hiện của mỗi yêu cầu.
- Đội duyệt lại yêu cầu phải kiểm tra mỗi yêu cầu về tính nhất quán và tính đầy đủ khi xem xét chúng như một thể thống nhất.

# Thẩm định yêu cầu

---

- Đội duyệt yêu cầu phải kiểm tra về:
  - *Tính kiểm tra được*
  - *Tính hiểu được*
  - *Tính lằn vết được*
  - *Tính thích nghi được*

# Thẩm định yêu cầu

---

- **Các kỹ thuật thẩm định yêu cầu**

## b. Làm bản mẫu

- Làm bản mẫu là một kỹ thuật thẩm định rất quan trọng, nhờ nó mà mô hình hệ thống thực hiện được có thể trình diễn cho người sử dụng.
- Bản mẫu giúp đào tạo người sử dụng hệ thống, họ có thể thấy được hệ thống có thật sự trợ giúp công việc của họ hay không.

# Thẩm định yêu cầu

---

## c. Tạo sinh các ca kiểm thử

- Để xem xét xem yêu cầu có vấn đề hay không

## d. Phân tích tính nhất quán được tự động

# Đặc tả hệ thống và làm bản mẫu

- **Tiến trình làm bản mẫu phần mềm**
- Duyệt lại yêu cầu có phê phán chính là nhiệm vụ của quá trình xác định yêu cầu. Tuy nhiên người sử dụng rất khó để hình dung hệ thống hoạt động như thế nào nếu chỉ đọc đặc tả yêu cầu. Do vậy phải có một bản mẫu hệ thống cho phép thử nghiệm trên nó.
- Lợi ích của việc phát triển bản mẫu:
  - + Khắc phục hiểu lầm giữa người phát triển phần mềm và người sử dụng
  - + Phát hiện thiếu hụt các dịch vụ người dùng
  - + Nhận ra sự khó sử dụng các dịch vụ người dùng
  - + Có thể tìm ra được những yêu cầu không nhất quán hoặc không đầy đủ ngay từ khi phát triển bản mẫu
  - + Giúp thuyết minh tính khả thi và hữu ích của ứng dụng với các nhà quản lý
  - + Bản mẫu được dùng làm cơ sở cho việc viết đặc tả

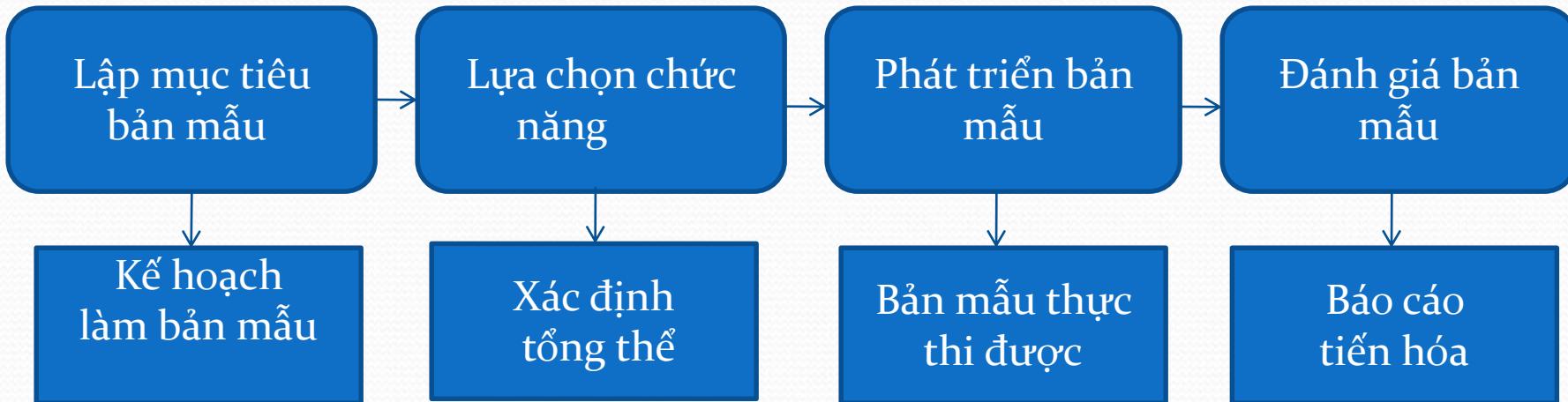
# Đặc tả hệ thống và làm bản mẫu

---

- Ngoài ra bản mẫu phần mềm cũng được dùng cho mục đích khác như:
  - Huấn luyện người sử dụng
  - Thủ nghiệm hệ thống

# Đặc tả hệ thống và làm bản mẫu

- Tiến trình làm bản mẫu phần mềm:



# Tạo bản mẫu trong tiến trình phần mềm

---

- Làm bản mẫu là sự mở rộng quá trình phân tích yêu cầu
- Làm bản mẫu là một phần hệ thống
- Làm bản mẫu là tạo ra một hệ thống

# Tạo bản mẫu trong tiến trình phần mềm

- **Một số cách tiếp cận làm bản mẫu**
- **Cách tiếp cận lập trình thăm dò:** Trình bày cho người dùng hệ thống chưa đầy đủ rồi cải tiến tăng cường nó cho đến khi các yêu cầu người dùng được thỏa mãn. Khi đó bản mẫu bị thải loại và một hệ thống chất lượng được xây dựng.
- **Phát triển gia tăng:** Khi một phần của hệ thống được phân phối người dùng có thể thực nghiệm và phản hồi trở lại. Cách này khắc phục được các vấn đề thay đổi liên tục của lập trình thăm dò. Kiến trúc hệ thống tổng thể được thiết lập sớm như một khung, khung đó được xem như cố định cho đến khi phát hiện sai sót. Tuy nhiên thông tin phản hồi từ phía người dùng về các thành phần đã được phân phối có thể ảnh hưởng đến thiết kế thành phần được lập trong kế hoạch phân phối tiếp theo.
  - **Phát triển tiến hóa:** dễ quản lý hơn lập trình thăm dò vì ở đây các chuẩn tiến trình phần mềm luôn được tuân thủ.

# Tạo bản mẫu trong tiến trình phần mềm

---

- **Các kỹ thuật tạo bản mẫu**
  - Ngôn ngữ đặc tả thi hành được
  - Các ngôn ngữ bậc rất cao
  - Các ngôn ngữ thế hệ thứ tư
  - Lắp ghép từ các thành phần dùng lại được

# Chương 4: Tiến trình phần mềm

---

## 3.1. Tổng quan về thiết kế phần mềm

### 3.1.1. Khái niệm và vai trò của thiết kế

- *Khái niệm thiết kế:*

Thiết kế phần mềm là quá trình chuyển các đặc tả yêu cầu phần mềm thành một biểu diễn thiết kế của hệ thống phần mềm cần xây dựng sao cho người lập trình có thể ánh xạ nó thành chương trình vận hành được.

# Khái niệm và vai trò của thiết kế

---

- Để làm được điều đó kỹ sư phần mềm cần đến một số hoạt động chính sau:
  - Nghiên cứu để hiểu vấn đề
  - Chọn một số giải pháp thiết kế và xác định các đặc điểm thô của nó. Các giải pháp được chọn cần khả thi và cho hiệu quả cao đối với hệ thống.
  - Mô tả trừu tượng cho mỗi giải pháp thiết kế. Trước hết cần xây dựng một mô tả ban đầu sơ khai, rồi chi tiết hóa dần. Các sai sót ở mỗi mức thiết kế trước đó được phát hiện và chỉnh sửa trước khi lập tài liệu thiết kế chính thức. s

# Khái niệm và vai trò của thiết kế

---

- *Vai trò của thiết kế*
  - Thiết kế là cách duy nhất để chuyển hóa một cách chính xác các yêu cầu của khách hàng thành mô hình thiết kế hệ thống phần mềm cuối cùng, làm cơ sở cho việc triển khai chương trình phần mềm.
  - Tài liệu thiết kế phần mềm là công cụ giao tiếp giữa nhóm cùng tham gia vào việc phát triển sản phẩm, để quản lý các rủi ro, đạt đến phần mềm hiệu quả.
  - Thiết kế phần mềm là tài liệu cung cấp đầy đủ các thông tin cần thiết cho những người kỹ sư hệ thống để bảo trì hệ thống sau này.

# Chương 4: Tiến trình phần mềm

---

## 3.1.2. Các khái niệm trong thiết kế

- **Trùu tượng:** Ký pháp trùu tượng mang tính tâm lý, cho phép ta tập trung vào một vấn đề ở một mức nào đó của sự khái quát, bỏ qua các chi tiết ở mức thấp ít liên quan. Việc sử dụng sự trùu tượng cũng cho ta làm việc với các khái niệm và thuật ngữ gần gũi trong môi trường của vấn đề đặt ra, mà không phải chuyển chúng thành một cấu trúc không quen thuộc”

### 3.1.2. Các khái niệm trong thiết kế

---

- **Các mức trừu tượng:**

- Mức cao nhất: mô tả đại thể sử dụng ngôn ngữ của môi trường vấn đề: ngôn ngữ nghiệp vụ, ngôn ngữ người dùng.
- Mức vừa: mô tả hướng thủ tục nhiều hơn. Thuật ngữ hướng vấn đề thường đi đôi với các mô tả bằng các phương pháp được sử dụng.
- Mức thấp: phát biểu theo thuật ngữ chi tiết để có thể chuyển trực tiếp thành chương trình.

### 3.1.2. Các khái niệm trong thiết kế

---

- **Phân rã:** phân rã là việc phân chia một đối tượng thành những đối tượng nhỏ hơn mà chúng vốn đã tồn tại. Nó là cách để có thể dễ dàng nghiên cứu những vấn đề lớn, phức tạp thông qua việc nghiên cứu các đối tượng con đơn giản hơn nó.
- **Làm mịn:** thiết kế được bổ sung các đối tượng thành phần nhỏ hơn nhưng theo chủ đích của người thiết kế.
- **Modun:** Phần mềm được chia thành các thành phần riêng biệt có tên và địa chỉ xác định được gọi là các modun. Tính modun là thuộc tính riêng của phần mềm, nó cho phép tổ chức một chương trình trở nên quản lý được theo một cách thông minh .

### 3.1.2. Các khái niệm trong thiết kế

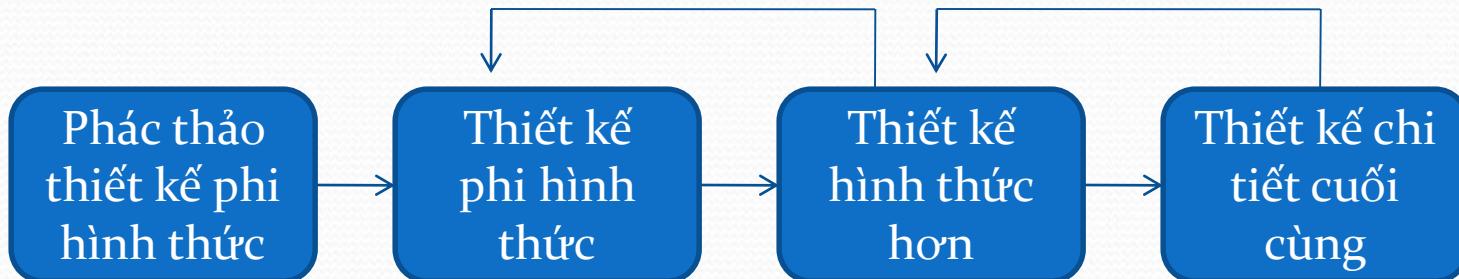
---

- **Thủ tục phần mềm:** Tập trung vào mô tả chi tiết các bước xử lý cho từng modun riêng biệt. Thủ tục cung cấp một đặc tả chính xác về một quá trình xử lý, có đầu vào, đầu ra, trình tự các sự kiện, các điểm quyết định rẽ nhánh điều khiển, các thao tác lặp lại, có thể bao gồm cả cấu trúc/tổ chức dữ liệu được sử dụng.
- **Che dấu thông tin:** Các modun nên được thiết kế sao cho dấu kín dữ liệu đối với mọi modun khác. Che giấu thông tin là một tiêu chuẩn của thiết kế đối với một hệ thống cấu trúc từ các modun. m

### 3.1.3. Triển khai thiết kế

- Tiến trình phần mềm có thể xem xét từ những góc độ khác nhau: nội dung công việc, trình tự thực hiện, phương pháp thiết kế, công cụ sử dụng.

#### 1. Tiến trình thiết kế



### 3.1.3. Triển khai thiết kế

- Kết quả của mỗi hoạt động thiết kế là một đặc tả. Đặc tả này có thể là một đặc tả trừu tượng, bán hình thức, hình thức hay cũng có thể là một đặc tả về một phần nào đó của hệ thống phải được thực hiện như thế nào.
- Tiến trình phát triển phần mềm là một sự hoàn thiện liên tục, nên giữa thiết kế và đặc tả yêu cầu có quan hệ rất chặt chẽ với nhau. Thực tế người thiết kế sẽ phải lặp đi lặp lại giữa các đặc tả yêu cầu và thiết kế.

## 2. Các hoạt động và sản phẩm thiết kế.

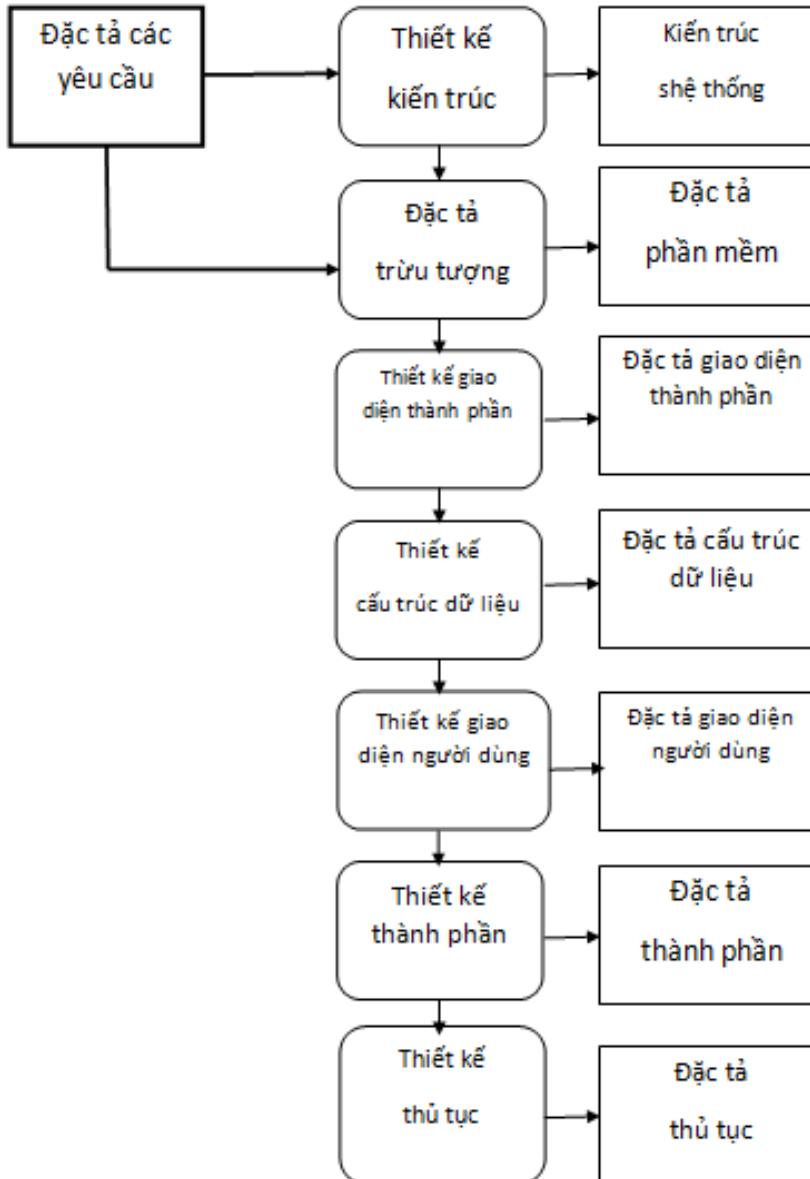
---

- a. Thiết kế kiến trúc: Xác định hệ con tạo nên hệ thống tổng thể và mối quan hệ giữa chúng.
- b. Đặc tả trừu tượng: đối với mỗi hệ con cần mô tả trừu tượng các dịch vụ mà nó cung cấp cùng các ràng buộc mà nó phải tuân thủ khi cung cấp dịch vụ.
- c. Thiết kế các giao diện thành phần: thiết kế các giao diện của hệ con với các hệ con khác, với các hệ thống khác (từ môi trường) sao cho có thể sử dụng hệ con khác thông qua giao diện mà không cần biết nó sẽ thực hiện như thế nào và nhờ vậy có thể phát triển các hệ con một cách độc lập.

## 2. Các hoạt động và sản phẩm thiết kế.

- d. Thiết kế cấu trúc dữ liệu: thiết kế cấu trúc dữ liệu lưu trữ và đặc tả các cấu trúc dữ liệu được dùng trong việc thực hiện hệ thống.
- e. thiết kế hệ thống giao diện người dùng: thiết kế giao diện bên ngoài mà qua đó người dùng có thể sử dụng được hệ thống.
- f. Thiết kế các thành phần: phân chia các dịch vụ mà một hệ thống con cung cấp vào các thành phần hợp thành của nó.
- g. Thiết kế thủ tục: thiết kế đặc tả các thuật toán, quy trình dùng để thực hiện các dịch vụ của mỗi thành phần sao cho có thể ánh xạ trực tiếp nó vào một ngôn ngữ lập trình.

# Các hoạt động thiết kế và sản phẩm

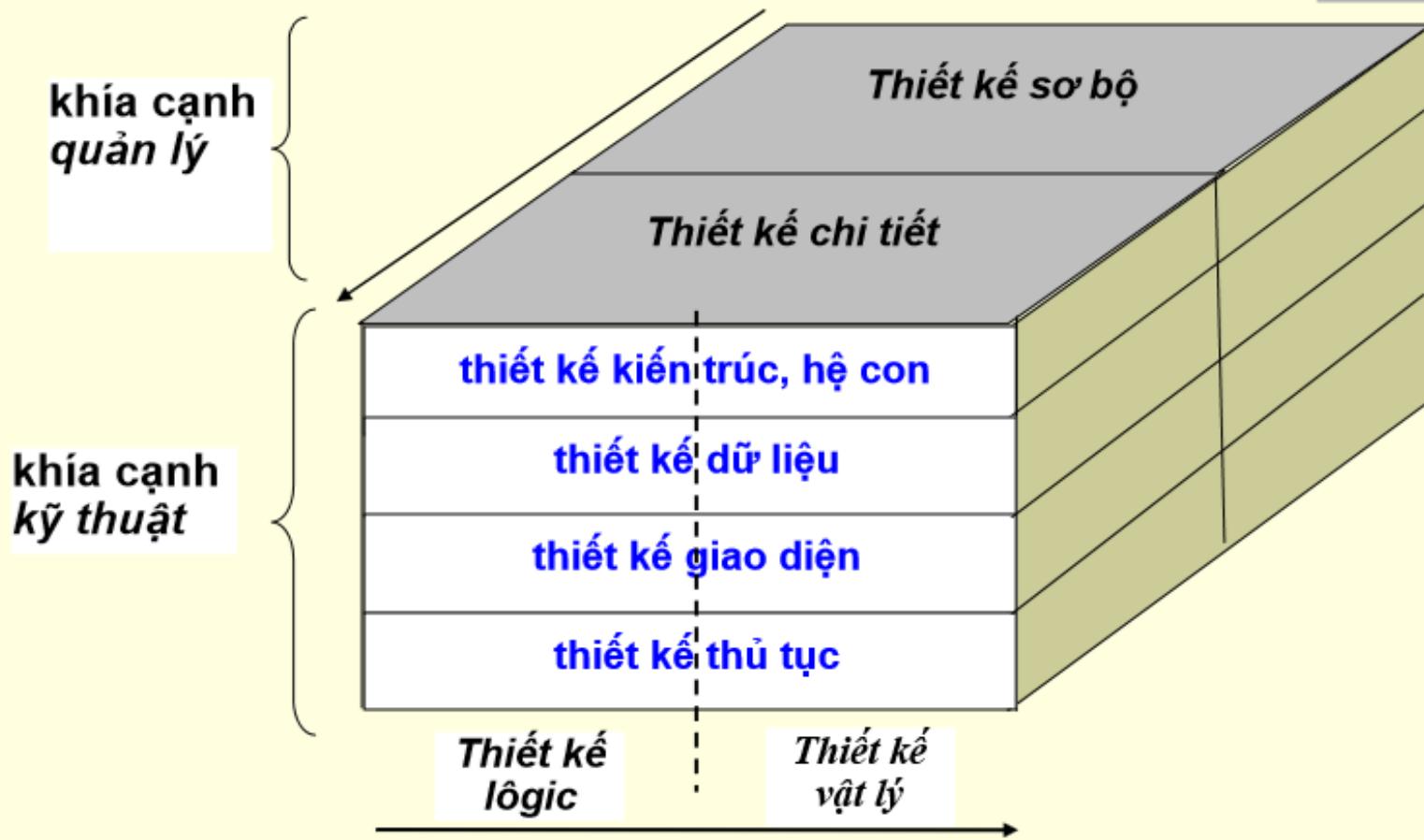


### 3. Biểu diễn thiết kế

---

- Một bản thiết kế phần mềm là một mô hình mô tả một đối tượng của thế giới thực với nhiều thành phần và các mối quan hệ giữa chúng với nhau. Có 3 hình thức thường được sử dụng để mô tả:
  - Các biểu đồ: được dùng để thể hiện các mối quan hệ giữa các thành phần lập nên hệ thống và là mô hình mô tả thế giới thực
  - Ngôn ngữ mô tả chương trình: dùng để kiểm tra và cấu trúc các cơ cấu thiết kế dựa trên các cấu trúc của một ngôn ngữ lập trình
  - Dạng văn bản không hình thức hóa: mô tả các thông tin không thể hình thức hóa được .

# 4. Các giai đoạn thiết kế



# 4. Các giai đoạn thiết kế

---

- Thiết kế logic: xác định cấu trúc thiết kế logic mô tả thành phần của hệ thống và các mối quan hệ giữa chúng mà ko gắn với bất kỳ một phương tiện vật lý nào.
- Thiết kế vật lý: chọn các giải pháp công nghệ hiện hữu để thực hiện các cấu trúc logic đã cho phù hợp với điều kiện của môi trường đích dự kiến của hệ thống phần mềm.

### 3.4. Các chiến lược và phương pháp thiết kế

---

- Chiến lược và phương pháp hướng cấu trúc
- Chiến lược và phương pháp hướng đối tượng
- Chiến lược thiết kế hệ thống tương tranh
- Tiếp cận chung các phương pháp thiết kế

Các phương pháp thiết kế đều có những đặc trưng chung:

- Một cơ chế để chuyển hóa
  - Các ký pháp để biểu diễn
  - Các trực cảm để phân hoạch và làm mịn vấn đề
- > Không có một chiến lược nào là tốt nhất cho mọi dự án lớn. Các cách tiếp cận trên là bổ sung hỗ trợ cho nhau mà không đối kháng nhau. Người kỹ sư phần mềm cần chọn cách tiếp cận thích hợp nhất cho từng giai đoạn thiết kế. ss

### 3.5. Chất lượng thiết kế và các giải pháp đảm bảo chất lượng

- **Sự kết dính:** sự kết dính của một thành phần là độ đo về tính gắn kết chặt chẽ với nhau giữa các bộ phận của nó.
- **Sự ghép nối:** chỉ ra mức độ độc lập giữa các đơn vị thành phần của một chương trình.
- **Tính hiểu được:** liên quan tới một số chức năng: tính kết dính, đặt tên, soạn tư liệu, độ phức tạp.
- **Tính thích nghi được**

# Các giải pháp cho một thiết kế tốt

---

- Có thể nhận được một thiết kế tốt nếu thực hiện đúng tiến trình thiết kế phần mềm thông qua việc áp dụng các nguyên lý thiết kế cơ bản, các phương pháp luận hệ thống, các công cụ trợ giúp và việc xét duyệt nghiêm túc.

# Các giải pháp cho một thiết kế tốt

---

- Những hướng dẫn cần được vận dụng trong thiết kế:
  - Tổ chức phân cấp
  - Tổ chức theo các modun
  - Sử dụng cách biểu diễn phân biệt và tách biệt giữa dữ liệu và thủ tục
  - Tổ chức các modun dùng chung cho các chức năng đặc trưng đặc biệt
  - Hình thành giao diện để rút gọn độ phức tạp của việc ghép nối giữa các modun với môi trường ngoài cũng như giữa chúng với nhau.
  - Sử dụng lại các thành phần phần mềm đã có

# Các giải pháp cho một thiết kế tốt

---

- **Những nguyên lý thiết kế cần vận dụng:**
  - Không ràng buộc tiến trình thiết kế trong một “cách nhìn hạn hẹp”
  - Thiết kế có thể lần vết trở lại mô hình phân tích hay các bước trước đó
  - Thiết kế không nên giải quyết vấn đề đã được giải quyết
  -

# Các giải pháp cho một thiết kế tốt

---

- Thiết kế phải rút ngắn được khoảng cách giữa phần mềm và vấn đề tồn tại trong thế giới thực
- Thiết kế cần thể hiện tính nhất quán và tích hợp
- Thiết kế cần được cấu trúc để dễ thay đổi
- Thiết kế không phải mã hóa, mã hóa không phải thiết kế
- Thiết kế cần được xem xét ngay từ đầu để tối thiểu hóa các lỗi
- Thiết kế cần được đánh giá và rà soát chất lượng trong quá trình phát triển.

## 2. Thiết kế kiến trúc

---

### 3.2.1. Kiến trúc phần mềm và vai trò của nó

- **Định nghĩa kiến trúc:** Kiến trúc phần mềm chỉ một cấu trúc tổng thể của phần mềm và qua đó cung cấp một sự tích hợp về mặt khái niệm của một hệ thống. S
- \_ Kiến trúc biểu diễn các thành phần lớn, cốt lõi của hệ thống và mối quan hệ giữa chúng với nhau được nhìn theo những quan điểm khác nhau.

## 2. Thiết kế kiến trúc

---

- **3.2.1. Kiến trúc phần mềm và vai trò của nó**
  - **Vai trò và tầm quan trọng của kiến trúc:**
    - + Công cụ giao tiếp giữa những người có liên quan
    - + Để phân tích hệ thống
    - + Giúp sử dụng lại ở quy mô lớn
  - Cho phép các kỹ sư phần mềm thực hiện những công việc sau:
    - + Tăng cường hiểu biết về hệ thống cần xây dựng
    - + Phân tích hiệu quả
    - + Xem xét, sửa đổi kiến trúc từ sớm, giảm rủi ro

## 3.2.2. Các mô hình kiến trúc

- Mô hình kiến trúc tĩnh: chỉ ra các hệ con hay các thành phần được phát triển như một đơn vị độc lập
- Mô hình tiến trình động chỉ ra hệ thống được tổ chức thành các tiến trình vận hành được.
- Mô hình giao diện: xác định các giao diện đưa ra các dịch vụ mà mỗi hệ con cung cấp thông qua giao diện của chúng.
- Mô hình liên kết: chỉ ra mối liên kết giữa các hệ con hay giữa các thành phần

## 3.2.2. Các mô hình kiến trúc

---

- **Các kiến trúc theo cách tổ chức dữ liệu:**
  - Mô hình kho dữ liệu
  - Mô hình máy khách/máy dịch vụ
- **Các mô hình điều khiển**
  - Mô hình điều khiển tập trung
  - Mô hình điều khiển dựa trên sự kiện

## 3.2.2. Các mô hình kiến trúc

---

- **Các mô hình kiến trúc miền cụ thể**
  - Mô hình tổng quát
  - Mô hình kiến trúc tham chiếu

### 3.2.3. Tiến trình thiết kế kiến trúc

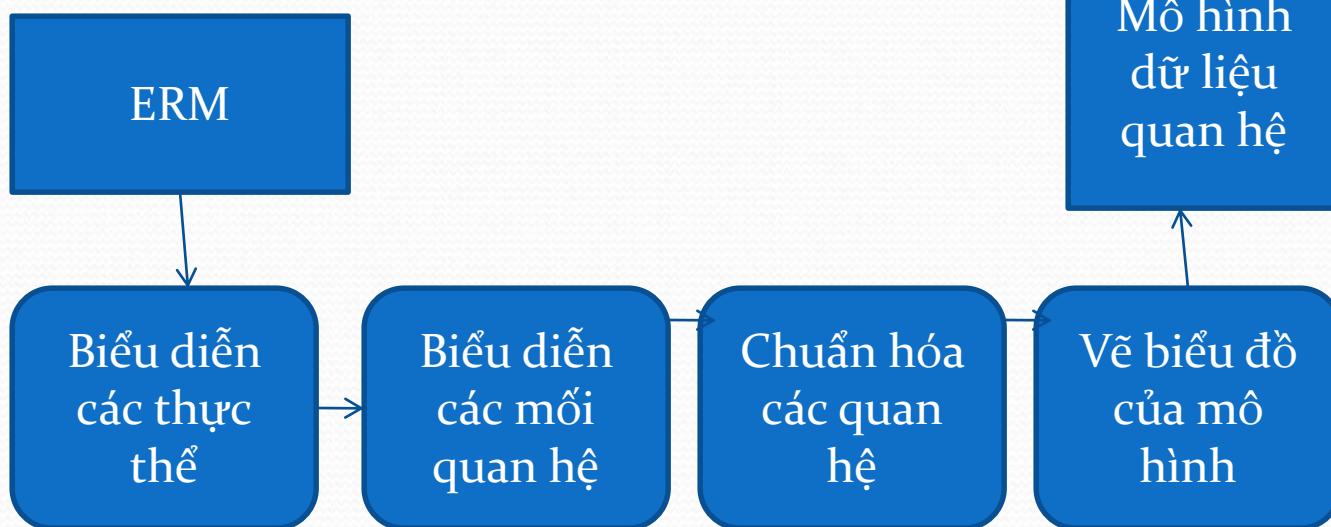
---

- Tiến trình xác định các hệ thống con của một hệ thống và thiết lập một khung làm việc cho việc điều khiển và giao tiếp giữa chúng gọi là thiết kế kiến trúc.
- Hoạt động chung của mọi tiến trình:
  - + Cấu trúc hệ thống: một hệ thống được cấu trúc thành một số hệ thống con
  - + Mô hình hóa điều khiển: thiết lập một mô hình chung mô tả mối quan hệ điều khiển giữa các phần trong hệ thống.
  - + Phân rã các modun: mỗi hệ thống con được phân rã thành các modun. Kiến trúc này chỉ ra các loại modun và liên kết giữa chúng.

### 3.3. Thiết kế hệ thống hướng chức năng

#### 3.3.1. Thiết kế dữ liệu

##### a. Thiết kế mô hình dữ liệu logic



*Tiến trình thiết kế dữ liệu logic*

# Thiết kế mô hình dữ liệu logic

---

- Dựa trên mô hình dữ liệu mức khái niệm
- Gồm 4 bước:
  - Bước 1. Phát triển mô hình dữ liệu mức logic cho mỗi giao diện sử dụng quy tắc chuyển hóa
  - Bước 2. Kết hợp các mô hình dữ liệu mức logic xây dựng cho mỗi giao diện thành một mô hình dữ liệu mức logic hợp nhất
  - Bước 3. Chuyển ERD thành mô hình dữ liệu mức logic sử dụng quy tắc chuyển hóa
  - Bước 4. So sánh mô hình dữ liệu mức logic hợp nhất với mô hình dữ liệu được chuyển từ ERD để tạo nên mô hình dữ liệu mức logic cuối cùng phần mềm
- ERD?(Sơ đồ quan hệ thực thể Entity relationship diagram)

# Thiết kế mô hình dữ liệu logic

---

- Ôn lại quan hệ và chuẩn hóa
  - Mô hình dữ liệu quan hệ
  - Dữ liệu được trình bày như một tập các bảng có liên quan với nhau hay còn gọi là một quan hệ
  - Mỗi quan hệ là một bảng 2 chiều gồm các hàng và cột
  - Chuẩn hóa
  - Chuẩn hóa (normalization) là quy trình biến đổi những cấu trúc dữ liệu phức tạp thành những cấu trúc dữ liệu ổn định và đơn giản
  - Kết quả của quá trình chuẩn hóa là quan hệ có cấu trúc cao
-

# Thiết kế mô hình dữ liệu logic

---

- **Các dạng chuẩn**
- Dạng chuẩn 1: Toàn bộ các thuộc tính của quan hệ đều có giá trị đơn
- Dạng chuẩn 2: Mỗi thuộc tính không phải là khóa chính sẽ được xác định bởi khóa chính (được gọi là phụ thuộc hàm đầy đủ – full functional dependency).
- Dạng chuẩn 3: Các thuộc tính không phải khóa chính không phụ thuộc lẫn nhau (được gọi là không chứa phụ thuộc bắc cầu – no transitive dependencies).
- Kết quả sau dạng chuẩn 3: Tất cả các thuộc tính không khóa đều phụ thuộc hoàn toàn vào khóa chính

# Xây dựng cơ sở dữ liệu mức logic

- Chuyển ERD thành cơ sở dữ liệu mức logic:

Biểu diễn thực thể

Biểu diễn liên kết

Chuẩn hóa quan hệ

Hợp nhất quan hệ

# Chuyển ERD thành cơ sở dữ liệu mức logic:

## B1: Biểu diễn liên kết

- Thực thể - Quan hệ
- Định danh - Khóa chính
- Khóa chính thỏa mãn:
  - Giá trị của khóa xác định duy nhất mọi hàng trong quan hệ

CUSTOMER

<u>Customer_ID</u>	Name	Address	City_State_Zip	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

# Chuyển ERD thành cơ sở dữ liệu mức logic:

---

- **B<sub>2</sub>: Biểu diễn liên kết**
- Liên kết 1-1
- Liên kết 1-n
- Liên kết n-n
- **B<sub>3</sub>: Chuẩn hóa quan hệ**

Dựa trên các quy tắc chuẩn hóa

**B<sub>4</sub>: Hợp nhất quan hệ**

# Thiết kế cơ sở dữ liệu vật lý

---

- Để thiết kế file và CSDL mức vật lý cần có những thông tin sau:
  - Các quan hệ đã được chuẩn hóa, bao gồm cả các ước tính về lượng dữ liệu của chúng
  - Định nghĩa của từng thuộc tính
  - Mô tả khi nào và trong trường hợp nào thì dữ liệu được nhập, truy xuất, xóa và cập nhật (bao gồm cả tần suất thực hiện)
  - Thời gian đáp ứng và mức độ toàn vẹn dữ liệu mong muốn
  - Bản mô tả các công nghệ được sử dụng để triển khai các file và cơ sở dữ liệu
-

# Thiết kế cơ sở dữ liệu vật lý

---

- **Thiết kế trường**

- Mỗi thuộc tính trong quan hệ sẽ được biểu diễn bởi một hoặc nhiều trường
  - Chọn kiểu DL nhằm thỏa mãn
    - + Tối thiểu không gian lưu trữ
    - + Trình bày tất cả các giá trị của trường
    - + Tăng tính toàn vẹn cho dữ liệu
    - + Hỗ trợ tất cả các thao tác của DL

# Thiết kế cơ sở dữ liệu vật lý

---

- **Kiểm soát tính toàn vẹn dữ liệu**

- Giá trị mặc định
- Mặt nạ nhập liệu
- Kiểm soát khoảng giá trị
- Toàn vẹn tham chiếu
- Kiểm soát giá trị null

# Thiết kế cơ sở dữ liệu vật lý

---

## • Thiết kế bảng vật lý

- Bảng vật lý: Là một tập hợp các hàng và cột chỉ ra chính xác các trường trong mỗi hàng của bảng
- Mục tiêu thiết kế:
  - + Sử dụng bộ nhớ thứ cấp hiệu quả
- ✓ Ô đĩa được chia thành các đơn vị mà được đọc bởi chỉ một thao tác
- ✓ Việc đọc hiệu quả nhất khi độ lớn của file vật lý gần bằng đơn vị lưu trữ
  - + Xử lý dữ liệu hiệu quả
- ✓ Thực hiện phi chuẩn hóa

# Thiết kế xử lý

- **Thiết kế xử lý logic**
- Xuất phát từ luồng dữ liệu vật lý, khi bỏ đi các yếu tố vật lý và cấu trúc lại để mô hình vẫn đảm bảo thực hiện các chức năng.
- **Thiết kế kiến trúc modun**
- **Các ưu nhược điểm của thiết kế hướng cấu trúc:**
  - Do đã phát triển hơn ba mươi năm do vậy phương pháp thiết kế và công cụ đã đạt đến mức hoàn thiện
  - Tuy nhiên hạn chế với các bài toán dữ liệu đa dạng và đòi hỏi sự tương tác của nhiều đối tượng khác nhau.

# Thiết kế hệ thống hướng đối tượng

## Một số khái niệm trong UML (ôn lại)

- Lớp
- Các mối quan hệ: kế thừa, phụ thuộc, kết hợp, kết tập, liên kết.

# Tiến trình thiết kế hướng đối tượng

---

- Xác định kiến trúc của hệ thống
- Sắp thứ tự ưu tiên giữa các gói
- Với mỗi gói, thiết kế cho mỗi ca sử dụng thuộc gói bằng cách xác định các lớp thiết kế tham gia triển khai các lớp phân tích
- Xây dựng biểu đồ tương tác giữa các lớp
- Thiết kế chi tiết các lớp
- Phân tích và hoàn thiện biểu đồ lớp dựa trên các mẫu thiết kế
- **(Bài tập)**

## Các ưu, nhược điểm của thiết kế hướng đối tượng

---

- Thiết kế hướng đối tượng dễ bảo trì vì có thể hiểu và cải biên đối tượng như một thực thể độc lập
- Các đối tượng là các thành phần có thể dùng lại được
- Có một vài lớp hệ thống phản ánh một quan hệ rõ ràng giữa các thực thể có thực ( như các thành phần phần cứng với các đối tượng điều khiển nó trong hệ thống).
- Nhược điểm của thiết kế hướng đối tượng là không dễ dàng nhận ra các đối tượng của một hệ thống.

# Thiết kế giao diện

---

- Vai trò:
  - Giao diện của các hệ thống tương tác thường được xem là tiêu chuẩn để đánh giá hệ thống tốt hay không tốt. Một hệ thống có giao diện tốt là một hệ thống thành công.  
Giao diện tương tác người máy đã được tiến hóa qua nhiều thế hệ:
    - + Thế hệ thứ nhất: giao diện chỉ lệnh và hỏi
    - + Thế hệ thứ 2: Giao diện thực đơn
    - + Thế hệ thứ 3: giao diện đồ họa

# Thiết kế giao diện

---

- Ưu điểm giao diện người dùng đồ họa: dễ học, dễ sử dụng, có nhiều cửa sổ cho phép hiện đồng thời nhiều kiểu thông tin khác nhau. Người dùng có thể thực hiện đồng thời nhiều công việc, có thể chuyển cảnh mà không mất mối quan hệ trực quan với những công việc khác.

# Các nguyên tắc thiết kế giao diện

---

- Tính quen thuộc với người sử dụng
- Tính nhất quán
- Làm ngạc nhiên nhỏ nhất: Các hoạt động của hệ thống không được làm người sử dụng phải ngạc nhiên.
- Có khả năng phục hồi: nên có cơ chế cho phép người sử dụng phải ngạc nhiên.
- Hướng dẫn người sử dụng: giao diện phải cung cấp các thông tin phản hồi có ý nghĩa và cung cấp các tiện ích trợ giúp người sử dụng theo từng ngữ cảnh nhạy cảm.
- Đa dạng người dùng

# Tiến trình thiết kế giao diện

---

- Tạo ra các mô hình khác nhau về chức năng của hệ thống
- Phác họa ra các nhiệm vụ hướng con người và máy tính để đạt tới chức năng hệ thống
- Xem xét giải pháp thiết kế được áp dụng cho mọi thiết kế giao diện
- Sử dụng các công cụ làm bản mẫu
- Cài đặt cho mô hình thiết kế và đánh giá kết quả về chất lượng.

# Các mô hình thiết kế giao diện

1. Mô hình thiết kế do kỹ sư phần mềm xây dựng. Nó tổ hợp các biểu diễn dữ liệu, kiến trúc và thủ tục của phần mềm để thực hiện được chức năng.
2. Mô hình người dùng do kỹ sư phần mềm/kỹ sư con người xây dựng. Nó mô tả sơ lược hệ thống cho người dùng cuối.
3. Mô hình người dùng cảm nhận hệ thống do người dùng cuối cùng xây dựng.
4. Hình ảnh hệ thống do người xây dựng hệ thống xây dựng.

# Chương 5: Lập trình hiệu quả

---

- **Ngôn ngữ lập trình**

- Ngôn ngữ lập trình là phương tiện để liên lạc giữa con người và máy tính. Tiến trình lập trình – Sự chuyển các yêu cầu của con người cho máy tính thực hiện thông qua ngôn ngữ lập trình – là một hoạt động trí tuệ rất cao của con người (lập trình viên) và là một bước cốt lõi trong tiến trình kỹ nghệ phần mềm.

# Chương 5: Lập trình hiệu quả

---

- **Đặc trưng của ngôn ngữ lập trình**
  - Dễ dịch thiết kế sang chương trình
  - Từ khóa gần với ngôn ngữ tự nhiên
  - Có trình biên dịch hiệu quả
  - Khả chuyển chương trình gốc
  - Có sẵn công cụ phát triển
  - Dễ bảo trì

# Chương 5: Lập trình hiệu quả

---

- **Lịch sử ngôn ngữ lập trình**
  - Các ngôn ngữ thế hệ thứ nhất:
    - + Ngôn ngữ lập trình mã máy
    - + Ngôn ngữ lập trình assembly
  - Các ngôn ngữ thế hệ thứ hai:
    - + Fortran, cobol, algol, basic
    - + phát triển 1950-1970
  - Các ngôn ngữ thế hệ thứ ba:
    - + Ngôn ngữ lập trình cấu trúc
    - + Ngôn ngữ lập trình hướng đối tượng
    - + **Lập trình hướng suy diễn, logic**

# Lập trình cấu trúc

---

- Câu lệnh không chỉ đơn thuần là gán
- Ba cấu trúc lệnh cơ bản:
  - *Selection*: if B then S1 else S2 if B then S1
  - *Iteration*: While B do S  
repeat S until B
  - *Sequencing*: S1; S2; S3;...

# Lập trình hướng đối tượng

---

- Dựa trên ba đặc trưng cơ bản:
  - Bao gói/ che dấu thông tin
  - Kế thừa
  - Đa hình

# Chương 5: Lập trình hiệu quả

---

- **Lịch sử ngôn ngữ lập trình**
  - Các ngôn ngữ thế hệ thứ tư:
    - + Truy vấn
    - + Các ngôn ngữ hỗ trợ quyết định

# Cấu trúc chương trình

---

- **Cấu trúc dữ liệu dễ hiểu**
  - Nên xác định tất cả các cấu trúc dữ liệu và thao tác cần thực hiện trên từng cấu trúc dữ liệu
  - Việc biểu diễn/ khai báo các cấu trúc dữ liệu chỉ nên thực hiện ở những mô đun sử dụng trực tiếp dữ liệu
  - Nên thiết lập và sử dụng từ điển dữ liệu khi thiết kế dữ liệu

# Cấu trúc chương trình

- **Cấu trúc thuật toán dễ hiểu**

---

- **Algorithm**

- **Structured coding và 9 điểm lưu ý:**

- Tuân theo quy cách lập trình
  - Một đầu vào, một đầu ra
  - Tránh GOTO, trừ khi phải ra khỏi lặp và dừng
  - Dùng comments hợp lý
  - Dùng tên biến có nghĩa, gợi nhớ
  - Cấu trúc lồng rõ ràng
  - Tránh dùng CASE / switch nhiều hoặc lồng nhau
  - Mã nguồn 1 chương trình / module nên viết trên 1 trang
  - Tránh viết nhiều lệnh trên 1 dòng
-

# Cấu trúc chương trình

- IF THEN / IF THEN ELSE

Pascal:

```
If điều kiện then  
Begin  
Công việc 1  
End;  
Else  
Begin  
Công việc 2  
end
```

Ngôn ngữ C

```
If ( điều kiện)  
{ công việc 1}  
Else  
{công việc 2}
```

# Cấu trúc chương trình

## • CASE/SWITCH

Pascal

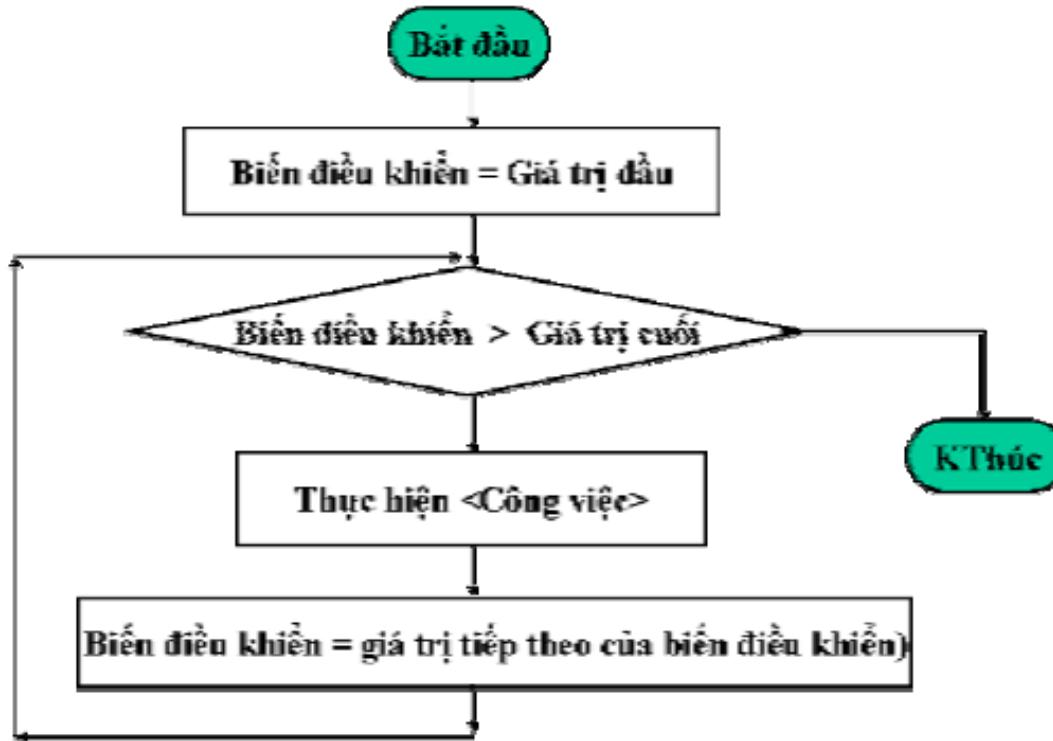
```
CASE <biểu thức> OF  
  Gtri 1: <việc 1>;  
  Gtri 2: <việc 2>;  
  ....  
  Gtri n: <việc n>;  
  Else  
    <việc n+1>;  
  END;
```

Ngôn ngữ C:

```
Switch (<biểu thức>)  
{  
  Case<gtri 1>:<việc 1>; [break;]  
  Case<gtri 2>:<việc 2>; [break;]  
  ....  
  Case<gtri n>:<việc n>; [break;]  
  [default: <việc n+1>; [break;]  
}
```

# Cấu trúc chương trình

- FOR TO,



## PASCAL

```
FOR biếnđkhiển := GTđầu TO GTCuối DO  
begin  
    <việc>
```

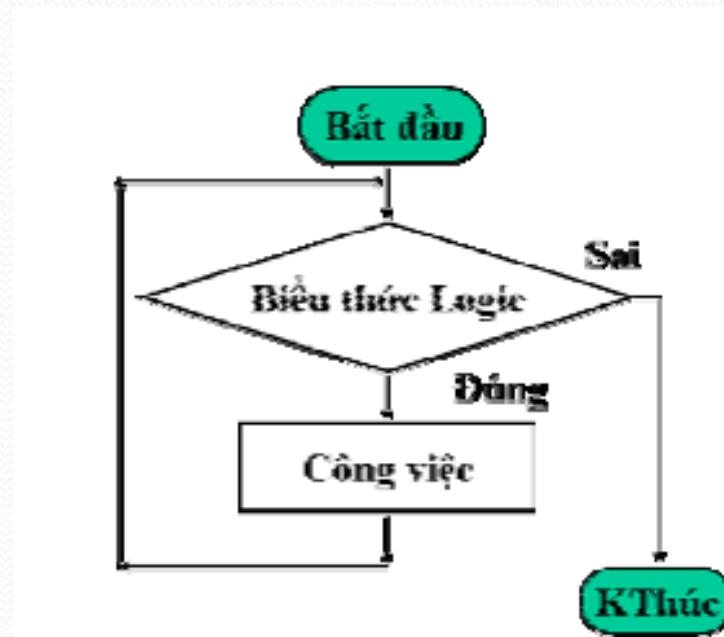
```
end;
```

## Ngôn ngữ C

```
for ( [biểuthíc1] ; [biểuthícĐK]; [biểuthíc2] )  
{ <việc>; }
```

# Cấu trúc chương trình

- DO WHILE



# Cấu trúc chương trình

---

## • DO WHILE

Pascal  
While Biểu thức Do  
Begin  
<công việc>  
End;

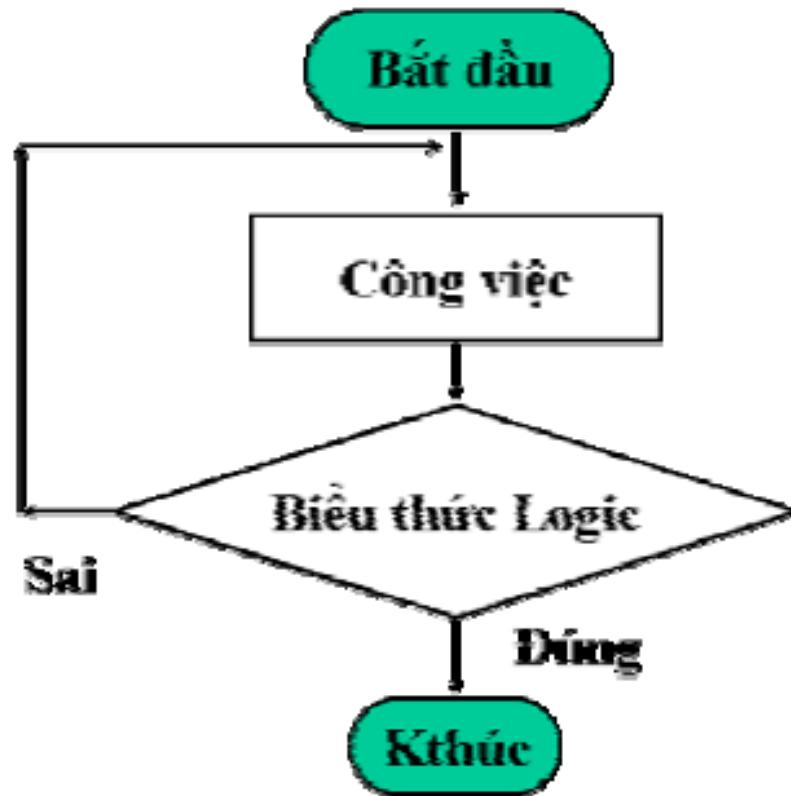
Ngôn ngữ C:  
While(<biểu thức ĐK>)  
{ <Công việc>; }

### Đặc trưng:

- Kiểm tra điều kiện trước khi thực hiện
- Lỗi thường gặp: lặp vô hạn

# Cấu trúc chương trình

## REPEAT UNTIL



# Cấu trúc chương trình

## REPEAT UNTIL

Pascal

```
Repeat  
<công việc>  
Until biểu thức;
```

Ngôn ngữ C

```
do {  
<công việc>;  
} while (<biểu thức ĐK>);
```

# Cấu trúc chương trình

---

- **Chú thích trong chương trình:**
  - **Tại sao cần đặt chú thích trong chương trình**
    - Vị trí đặt các chú thích trong chương trình
      - + thành phần/ Module
      - + Lớp
      - + Hàm/thủ tục
      - + Các vị trí đặc biệt khác
    - Một số quy định đặt chú thích:
      - + Ngắn gọn
      - + Gợi nhớ

# Phân loại ngôn ngữ:

---

- Phân loại theo miền lĩnh vực
- Phân loại theo môi trường mạng- không mạng
- Phân loại theo sự tiện ích: trực quan- không trực quan

# Lựa chọn ngôn ngữ

---

- Các đặc trưng của ngôn ngữ lập trình sẽ quyết định miền ứng dụng của ngôn ngữ. Miền ứng dụng là yếu tố chính để chúng ta lựa chọn ngôn ngữ cho một dự án phần mềm.

# Sự ảnh hưởng của ngôn ngữ tới kĩ nghệ phần mềm

- Ngôn ngữ lập trình ảnh hưởng tới thiết kế do vậy nó sẽ ảnh hưởng tới kĩ nghệ phần mềm
- Các đặc trưng của ngôn ngữ còn ảnh hưởng tới kiểm thử phần mềm.

# Phong cách lập trình

---

- Phong cách lập trình bao hàm một triết lý về lập trình nhấn mạnh tính dễ hiểu của chương trình nguồn. Các yếu tố phong cách bao gồm tài liệu bên trong chương trình, phương pháp khai báo dữ liệu, cách xây dựng câu lệnh và các kỹ thuật vào ra.

# Tài liệu chương trình

---

- Tài liệu bên trong của chương trình gốc bắt đầu với việc chọn lựa các tên gọi định danh (biến và nhãn), tiếp tục với vị trí thành phần của việc chú thích, kết luận với cách tổ chức cấu trúc một cách trực quan của chương trình.

# Khai báo dữ liệu

---

- Một số hướng dẫn trong việc khai báo dữ liệu:
  - Các cấu trúc dữ liệu nên được chú giải đầy đủ về cấu trúc và chức năng và các đặc thù về sử dụng.
  - Thứ tự khai báo dữ liệu nên được chuẩn hóa cho dù ngôn ngữ lập trình không có yêu cầu bắt buộc nào về điều đó.
  - Quy cách đặt tên cho các biến cần nhất quán. Các tên biến ngoài việc có ý nghĩa còn cần mang thông tin về kiểu của chúng.

# Xây dựng câu lệnh

- Việc xây dựng câu lệnh nên tuân theo một quy tắc quan trọng: mỗi câu lệnh nên đơn giản và trực tiếp.
- Các câu lệnh có thể được đơn giản hóa bằng các kỹ thuật:
  - Tránh dùng các phép kiểm tra điều kiện phức tạp
  - Khử định các phép kiểm tra điều kiện phủ định
  - Tránh lồng nhau nhiều giữa các điều kiện hay chu trình
  - Dùng dấu ngoặc để làm sáng tỏ các biểu thức logic hay số học
  - Dùng dấu ngoặc để làm sáng tỏ các biểu thức logic hay số học
  - Dùng dấu cách hoặc ký hiệu dễ đọc để làm sáng tỏ nội dung câu lệnh

# Phong cách vào/ra của các modun

---

- Việc vào ra giữa các modun nên tuân thủ theo một số hướng dẫn sau:
  - Kiểm tra tính hợp lệ của dữ liệu vào
  - Giữ cho định dạng dữ liệu đơn giản và thống nhất
  - Giảm độ ghép nối

# Lập trình tránh lỗi

---

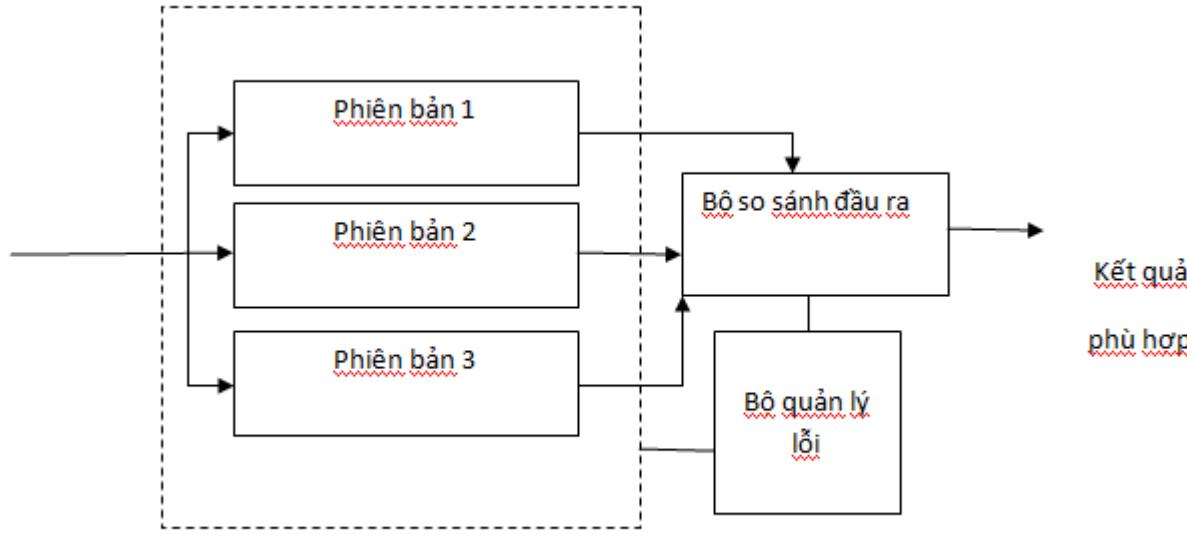
- **Một phần mềm tốt cần phải được phát triển dựa trên các yếu tố sau:**
  - Sản phẩm của một đặc tả hệ thống chính xác
  - Tăng cường duyệt lại trong quá trình phát triển và thẩm định hệ thống phần mềm
  - Tổ chức thử nghiệm một cách có hệ thống
  - Áp dụng tư duy thiết kế hiện đại

# Lập trình dung thứ lỗi

---

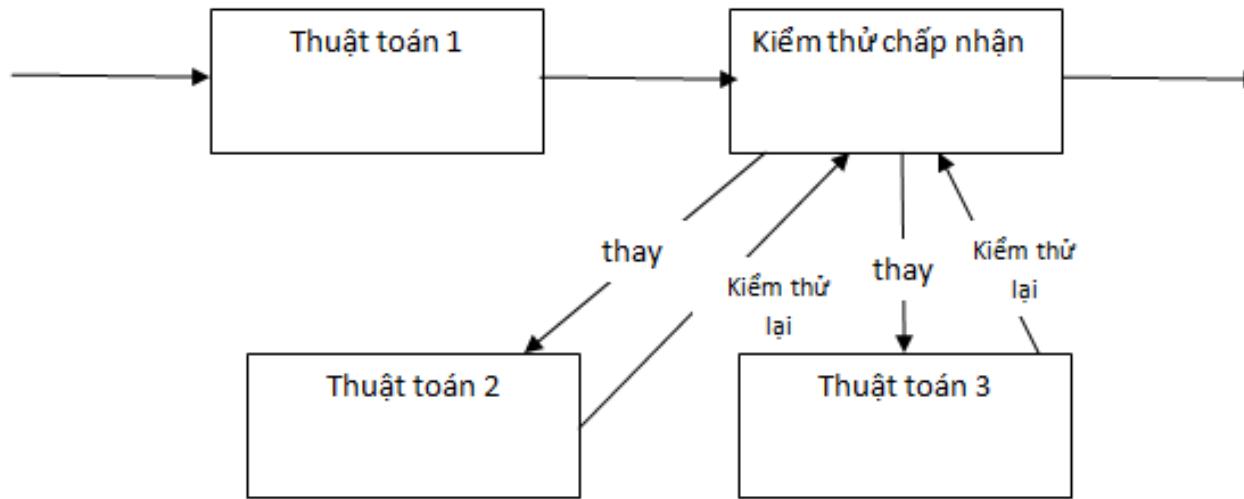
- Có bốn hoạt động cần phải tiến hành để hệ thống có khả năng thứ lỗi là:
  - Phát hiện lỗi
  - Xác định mức độ nghiêm trọng
  - Phục hồi sau khi gặp lỗi
  - Chữa lỗi
- Có hai cách tiếp cận chính để xây dựng một hệ thống dung thứ lỗi là:
  - Lập trình N phiên bản
  - Khối phục hồi

# Lập trình dung thứ lỗi



Lập trình N - phiên bản

# Lập trình dung thứ lỗi



Khôi phục hồi

# Lập trình phòng thủ

---

- Lập trình phòng thủ cũng là một kỹ thuật lập trình dung thứ lỗi nhưng không cần đến các khối chương trình chuyên dụng.
- Việc ngăn ngừa lỗi ở đây vẫn là quá trình phát hiện lỗi, đánh giá lỗi và phục hồi sau lỗi.
- Lập trình phòng thủ chủ yếu tập trung đến các lỗi mang tính khách quan khi thực hiện chương trình như lỗi vào/ra, lỗi môi trường thực hiện (lỗi bộ nhớ), hay một số lỗi như tràn số, lỗi chia số không...

# Lập trình hướng hiệu quả

- Tính hiệu quả của chương trình gốc có liên hệ trực tiếp với tính hiệu quả của thuật toán được xác định trong thiết kế chi tiết.
- Phong cách lập trình có thể tác động nhất định đến tốc độ thực hiện và yêu cầu bộ nhớ.
  - Đơn giản hóa các biểu thức số học và logic
  - Tối ưu khối lệnh trong các chu trình
  - Tránh dùng mảng nhiều chiều
  - Tránh dùng con trỏ và danh sách phức tạp
  - Dùng các phép toán nhanh
  - Tránh trộn lẫn các kiểu dữ liệu, nên sử dụng các kiểu đơn giản.
  - Tránh gọi các chương trình con

# Hiệu quả bộ nhớ

- Tính hiệu quả trong sử dụng bộ nhớ được tính toán dựa vào đặc trưng sử dụng bộ nhớ đệm của hệ điều hành.
- Tính cục bộ của chương trình và dữ liệu (tính modun hóa cao) là một phương pháp để tăng tính hiệu quả của sử dụng bộ nhớ đệm và tăng tốc độ thực hiện của chương trình.

# Hiệu quả vào/ra

- Một trong những khâu lãng phí tốc độ là việc vào/ra dữ liệu.
- Các thiết bị vào/ra thường có tốc độ chậm hơn nhiều so với khả năng tính toán của máy tính và tốc độ truy cập bộ nhớ trong. Do vậy việc tối ưu vào/ra có thể làm tăng đáng kể tốc độ thực hiện.
- Để tăng cường hiệu quả vào/ra:
  - Số các yêu cầu vào/ra nên giữa mức tối thiểu
  - Mọi việc vào/ra nên qua bộ đệm
  - Nên tối ưu thao tác vào/ra với từng chủng loại thiết bị đầu cuối ví dụ như màn hình, máy in, mạng hay các loại bộ nhớ ngoài.

# Một số môi trường phát triển

---

- Môi trường VISUAL STUDIO.NET
- Môi trường lập trình JAVA...

# Tổng kết chương

---

- Bước lập trình là một tiến trình chuyển hóa thiết kế chi tiết thành chương trình, cuối cùng được biến đổi thành mã máy thực hiện được. Lập trình là khâu không thể thiếu trong kỹ nghệ phần mềm và có ảnh hưởng lớn đến chất lượng phần mềm
- Có nhiều ngôn ngữ lập trình với các đặc trưng và miền ứng dụng khác nhau. Đặc trưng của ngôn ngữ lập trình có ảnh hưởng lớn đến quá trình xây dựng, kiểm thử, bảo trì phần mềm.
- Phong cách lập trình quyết định tính dễ hiểu của chương trình gốc
- Chương trình cần độ tin cậy cao nên cần những kỹ thuật cơ bản đảm bảo độ tin cậy cho chương trình.
- Lập trình hướng tới hiệu quả thực hiện tức là tiết kiệm tài nguyên phần cứng
- Môi trường phát triển là một phương tiện trợ giúp đắc lực cho tiến trình phát triển một ứng dụng một cách hiệu quả.

# Chương 6: Kiểm thử

---

- Khái niệm kiểm thử
- Các loại kiểm thử
- Thẩm định và xác minh

# Khái niệm về kiểm thử phần mềm

---

- Kiểm thử phần mềm là yếu tố quyết định của đảm bảo chất lượng phần mềm SQA(Software Quality Assurance), là khâu điển hình của rà soát đặc tả, thiết kế, lập mã
- Kiểm thử theo Glen Myers là quá trình vận hành chương trình để tìm ra lỗi.

# Lý do kiểm thử phần mềm

---

- Muốn nhìn thấy phần mềm như một phần tử của hệ thống hoạt động(xem sản phẩm)
- Hạn chế chi phí cho các thất bại do lỗi gây ra sau này(hiệu quả)
- Có kế hoạch tốt nâng cao chất lượng suốt quá trình phát triển(giải pháp)

# Vai trò kiểm thử phần mềm

---

- Chi phí kiểm thử chiếm:

- + 40% tổng công sức phát triển

- + >30% tổng thời gian phát triển

Với các phần mềm có ảnh hưởng tới sinh mạng, chi phí có thể gấp từ 3 đến 5 lần tổng chi phí khác cộng lại.

Kiểm thử tốt sẽ:

- + Giảm chi phí phát triển

- + Tăng độ tin cậy của sản phẩm phần mềm

# Mục tiêu kiểm thử phần mềm

---

- Mục tiêu trước mắt: tạo ra các ca kiểm thử để chỉ ra lỗi của phần mềm (tức là “đánh đổ” phần mềm)
- Mục đích cuối cùng: có một chương trình tốt, chi phí ít-> xây dựng

# Quan niệm về một kiểm thử tốt

---

- Khi kiểm thử ta tiến hành những ca kiểm thử khác nhau:
  - Ca kiểm thử tốt có xác suất cao tìm ra một lỗi chưa được phát hiện.
  - Ca kiểm thử thắng lợi làm lộ ra ít nhất một lỗi còn chưa được phát hiện.

# Lợi ích khác của kiểm thử

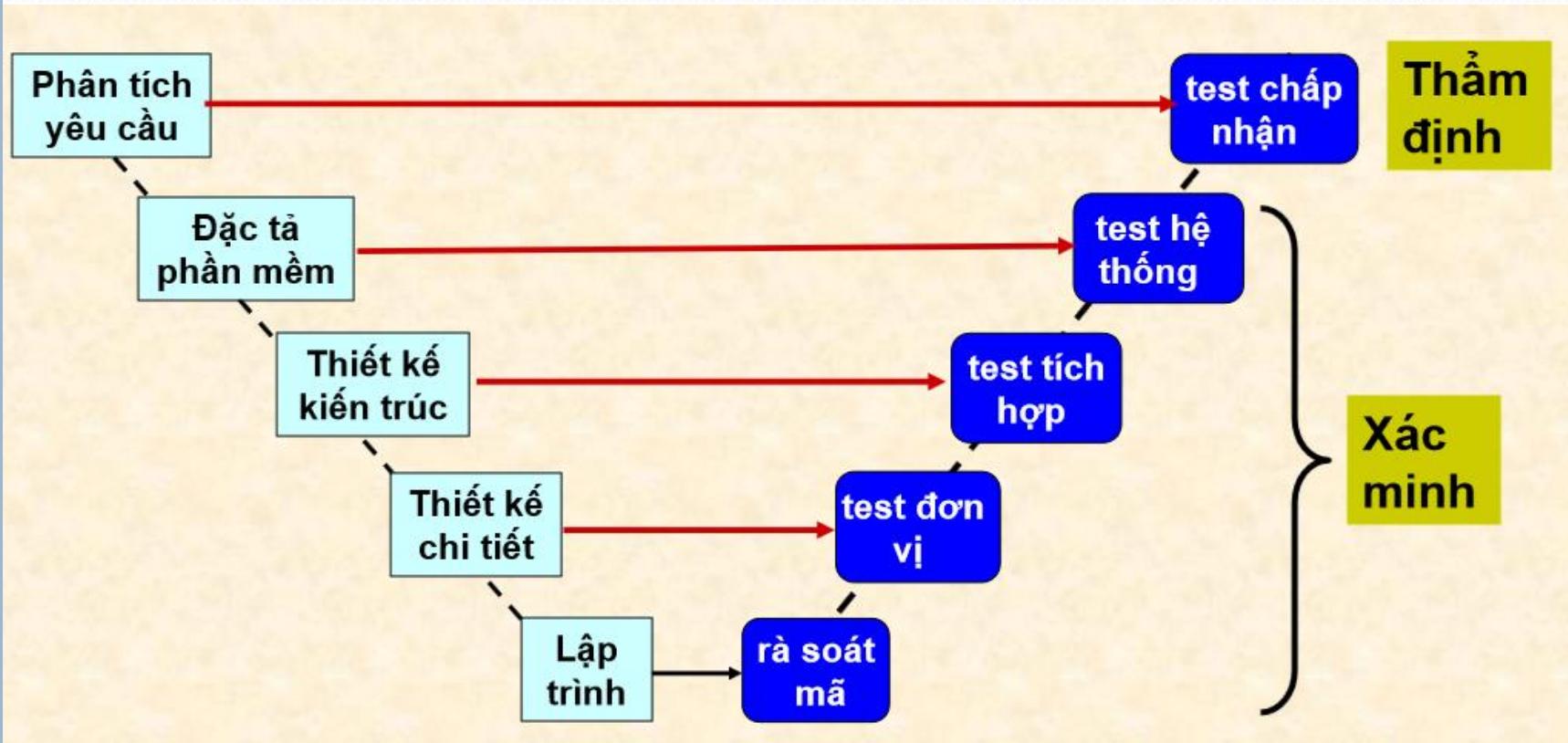
---

Các kiểm thử thắng lợi làm lộ ra khiếm khuyết đồng thời mang lại các lợi ích phụ là thuyết minh:

- Chức năng tương ứng với đặc tả (xác minh)
- Thực thi phù hợp yêu cầu đặc tả (thẩm định, xác minh)
- Cung cấp các chỉ số, độ tin cậy và chất lượng (thẩm định)

\*Tuy nhiên kiểm thử không thể khẳng định rằng phần mềm không có khiếm khuyết

# Mô hình chữ V các mức kiểm thử



# Những khó khăn khi kiểm thử

---

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

# Các điểm lưu ý khi kiểm thử

---

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

# Các mức – loại hình kiểm thử

---

- Kiểm thử đơn vị(unit testing)
- Kiểm thử tích hợp(integration testing)
- Kiểm thử hệ thống(system testing)
  - + Kiểm thử chức năng(functional testing)
  - + Kiểm thử phục hồi(recovery test)
  - + Kiểm thử chịu tải(extra: stress & load test)
  - + Kiểm thử thi hành(performance test)
  - + Kiểm thử an ninh(sercurity test)

# Các loại hình kiểm thử

---

- **Kiểm thử chấp nhận(acceptance testing)**
  - **Kiểm thử alpha(alpha testing)**
    - + Người dùng thực hiện
    - + Trong môi trường được quản lý
  - **Kiểm thử Beta(beta testing)**
    - + Người dùng thực hiện
    - + trong môi trường thực

# Testing levels (detailed)

- **Unit testing**

- Kiểm thử các đơn vị chương trình độc lập như các thủ tục, hàm, phương thức một cách riêng rẽ

- **Integration testing**

- Kiểm thử việc ghép nối các đơn vị chương trình

- **System testing**

- Bao gồm một dải kiểm thử rộng như tính chức năng, khả năng chịu tải

- **Acceptance testing**

- Khách hàng kiểm tra những kỳ vọng của mình về hệ thống

- Hai loại acceptance testing

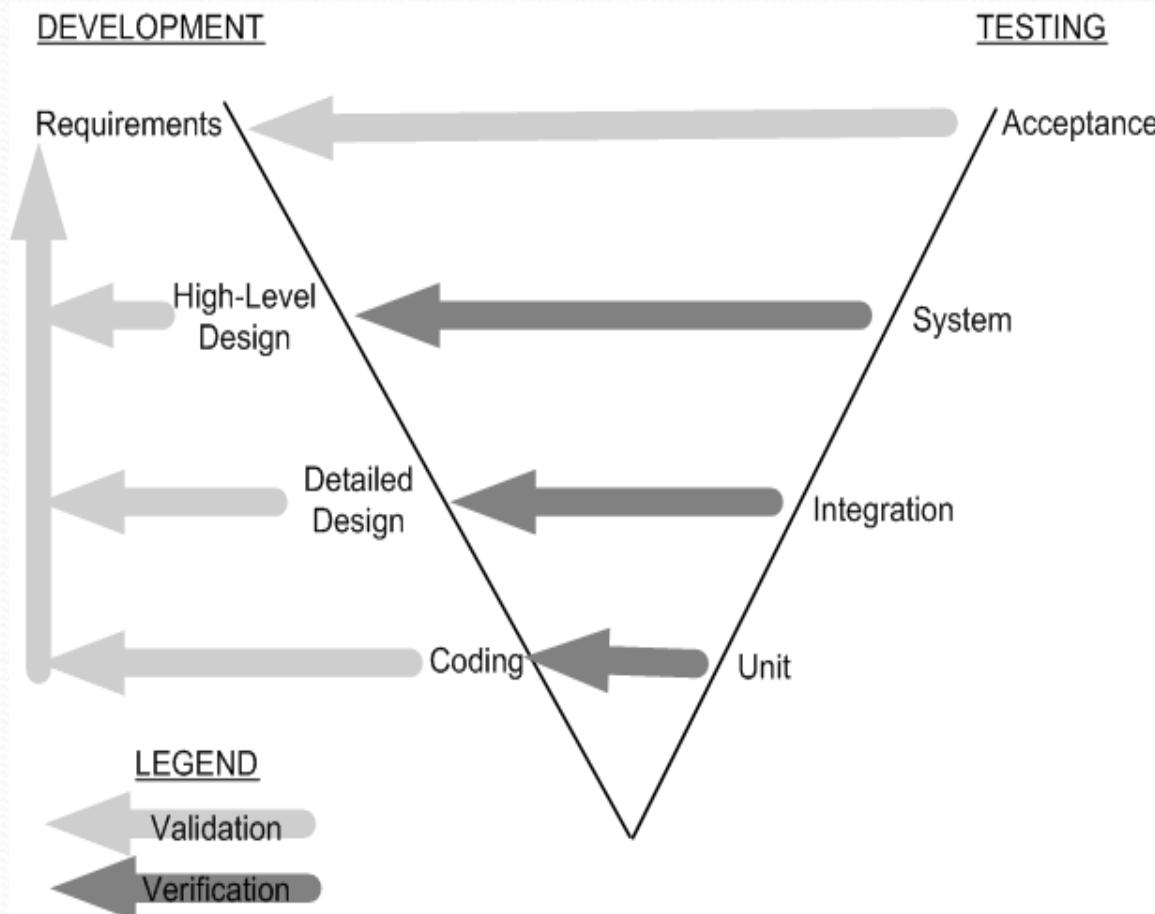
- UAT (User acceptance testing)

- BAT (Business Acceptance Testing)

- UAT: Hệ thống đáp ứng các tiêu chí của hợp đồng

- BAT: Hệ thống chưa, nhưng sẽ đáp ứng được user acceptance test

# Testing levels (tiếp)



# Performance testing

- Có thể liên quan đến việc kiểm thử các tính chất quan trọng của hệ thống như hiệu suất và độ tin cậy.
- Các tests hiệu suất thường liên quan đến việc lập kế hoạch cho một loạt các bài kiểm tra khả năng chịu tải tăng dần cho đến khi hệ thống không thể thực hiện được nữa.

# Stress testing

---

- Là các thử nghiệm hệ thống vượt quá khả năng chịu tải của nó. Việc thử nghiệm quá tải thường giúp phát hiện các khuyết tật của hệ thống.
  - Tập trung vào vấn đề hư hại của hệ thống. Không có hệ thống nào không thể không hỏng. Stress testing kiểm tra sự mất mát dịch vụ và dữ liệu không thể chấp nhận được (giới hạn mất mát dữ liệu và dịch vụ).
  - Stress testing đặc biệt quan trọng đối với các hệ thống phân tán là những hệ thống dễ bị sụp đổ nhanh chóng, ví dụ như một mạng khi bị quá tải.
-

# What is a Test Case?

---

- Test Case là một cặp **<input, expected outcome>**
- Đối với những hệ thống State-less (phi trạng thái): (ví dụ compiler là một hệ thống)
  - Test cases rất đơn giản
    - Outcome chỉ phụ thuộc vào input hiện tại
- Đối với những hệ thống có trạng thái State-oriented: Ví dụ như máy ATM
  - Test cases không đơn giản. Một test case có thể bao gồm một chuỗi **<input, expected outcome>**
    - Outcome phụ thuộc cả vào trạng thái hiện tại của hệ thống và input hiện tại
    - ATM example:
      - <check balance, \$500.00>,
      - <withdraw, “amount?”>,
      - <\$200.00, “\$200.00”>,
      - <check balance, \$300.00>

# Test case design

---

- Liên quan đến việc thiết kế các test cases (inputs và outputs) để kiểm thử hệ thống.
- Mục đích của thiết kế test case là để tạo ra một tập hợp các bài test có hiệu quả trong thẩm định và kiểm tra khuyết tật.
- Một số cách tiếp cận khi thiết kế test case:
  - Dựa vào cấu trúc (Structural testing).
  - Dựa trên yêu cầu (Requirements-based testing);
  - Phân lớp (Partition testing);

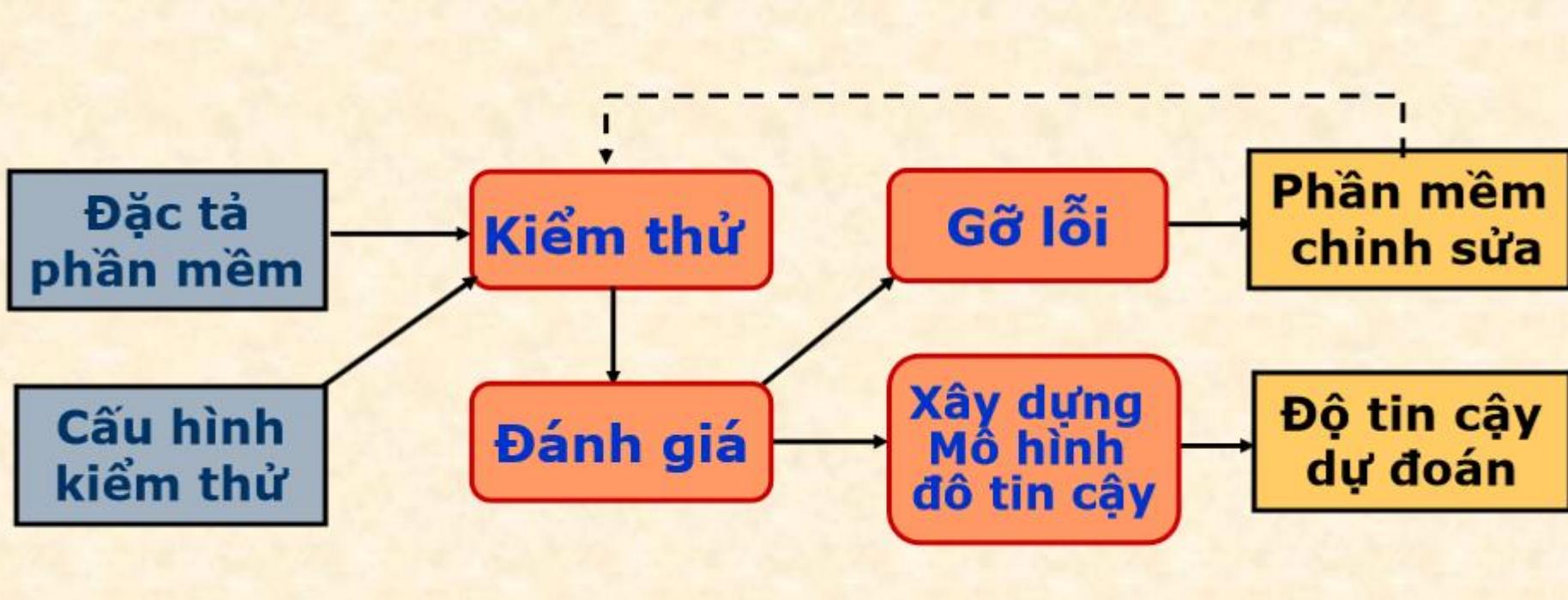
# Phương pháp và chiến lược kiểm thử

---

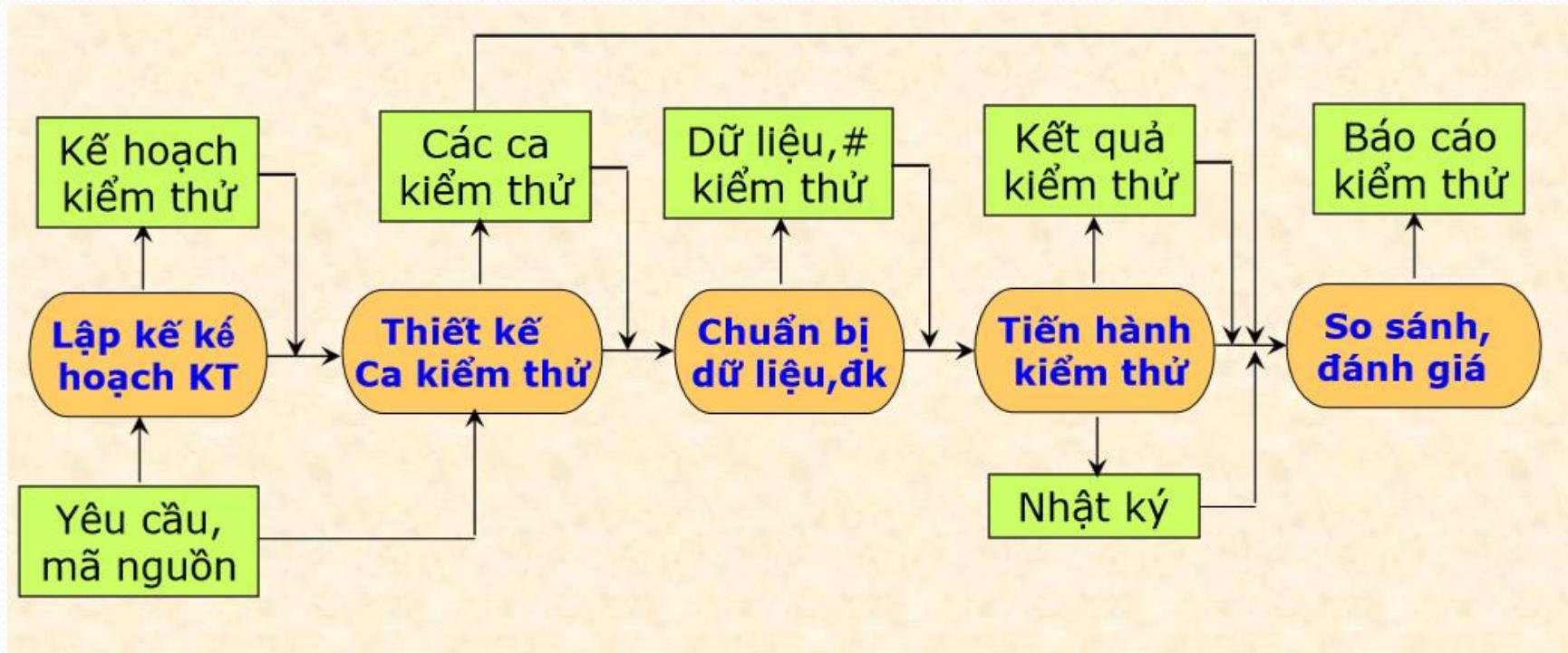
- **Hai phương pháp phổ biến**
  - + Kiểm thử hộp trắng (white box)
  - + Kiểm thử hộp đen(black box)
- **Các chiến lược kiểm thử**
  - **Ứng dụng cho các mức và loại kiểm thử**
  - **Một số chiến lược**
    - + Kiểm thử từ trên xuống/dưới lên/lai (tích hợp)
    - + Kiểm thử vụ nổ lớn (big bang- tích hợp)
    - + Kiểm thử hồi quy(tích hợp)
    - + Kiểm thử luồn sợi( hệ thời gian thực)

# Biểu đồ dòng thông tin kiểm thử

- Sơ đồ dòng thông tin của tiến trình kiểm thử



# Tiến trình thực hiện kiểm thử



# Khái niệm về thiết kế ca kiểm thử

---

- Mục tiêu thiết kế ca kiểm thử nhằm:
  - Tìm ra nhiều sai nhầm
  - VỚI nỗ lực và thời gian nhỏ nhất
- Trong các thập kỷ 80-90 đã nghiên cứu nhiều loại phương pháp thiết kế ca kiểm thử
- Các phương pháp tốt phải có một cơ chế:
  - Đảm bảo tính đầy đủ
  - Cung cấp khả năng thực sự phát hiện được các sai

# Kiểm thử hộp trắng

---

- Khái niệm:
  - Đối tượng: mã nguồn
  - Mức: các modun đơn vị
  - Nội dung là khám xét:
    - + Các chi tiết thủ tục(thuật toán)
    - + Con đường logic(luồng điều khiển)
    - + Các trạng thái của chương trình (dữ liệu)

# Nội dung kiểm thử hộp trắng

---

- Kiểm thử cái gì?
  - Mọi lệnh (đầy đủ)
  - Mọi điều kiện logic có thể (rẽ nhánh)
  - Mọi chu trình trong chương trình (lặp lại)
  - Mọi cấu trúc dữ liệu được dùng (dữ liệu)
  - Mọi tiến trình từ đầu – kết thúc(từng luồng điều khiển)

# Yêu cầu của kiểm thử hộp trắng

---

- Yêu cầu đặt ra:
  - Mọi con đường độc lập trong mọi modun cần được thực hiện ít nhất một lần.
  - Mọi ràng buộc logic được thực hiện cả hai phía đúng và phía sai
  - Tất cả các vòng lặp ở biên của nó và cả các biến vận hành phải được thực hiện
  - Mọi cấu trúc dữ liệu nội tại được dùng để đảm bảo hiệu lực thi hành của nó

# Lý do kiểm thử hộp trắng

---

- Vì sao tốn tiền cho kiểm thử hộp trắng
  - Các sai logic và giả thiết không đúng tỷ lệ nghịch với xác suất để một con đường logic được thi hành
  - Thực tế: mọi con đường logic đều có thể được thi hành trên một cơ sở nhất định
  - Có những sai chính tả có thể là ngẫu nhiên trên đường ta không kiểm tra.

# Các kỹ thuật sử dụng

---

- Đồ thị dòng (Tom MacCabe đưa ra đầu tiên)
- Ma trận kiểm thử (số đường đi, trọng số)
- Điều kiện logic (chiến lược miền và BRO)
- Điều khiển theo dòng dữ liệu
- Các cấu trúc chu trình – giá trị đặc trưng

# Kỹ thuật đồ thị dòng

---

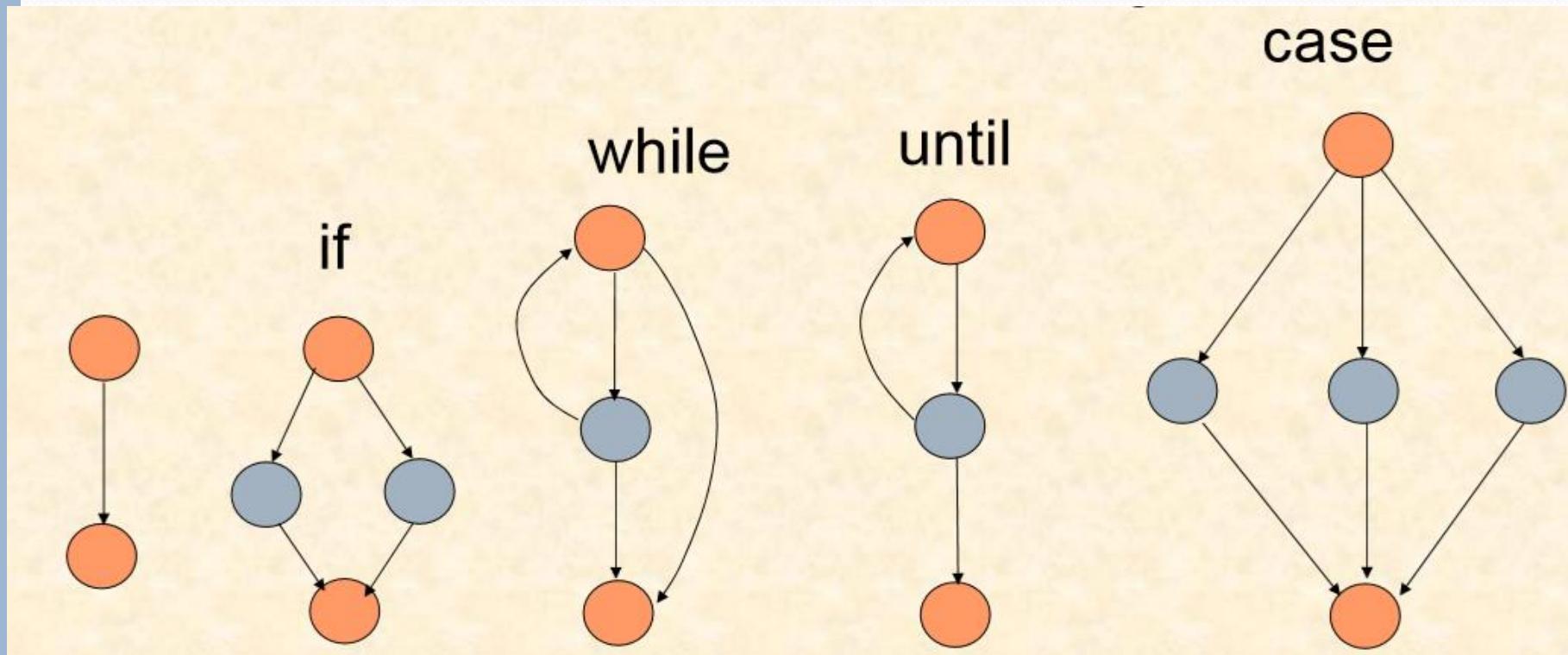
- Khái niệm về đồ thị dòng
- Là một kỹ thuật dựa trên cấu trúc điều khiển của chương trình
- Gần giống đồ thị luồng điều khiển của chương trình
- Nhận được từ đồ thị luồng điều khiển bằng cách:
  - + Gộp các lệnh tuần tự
  - + Thay lệnh rẽ nhánh và điểm kết thúc của các đường điều khiển bằng 1 nút vị tự

# Cấu trúc đồ thị dòng

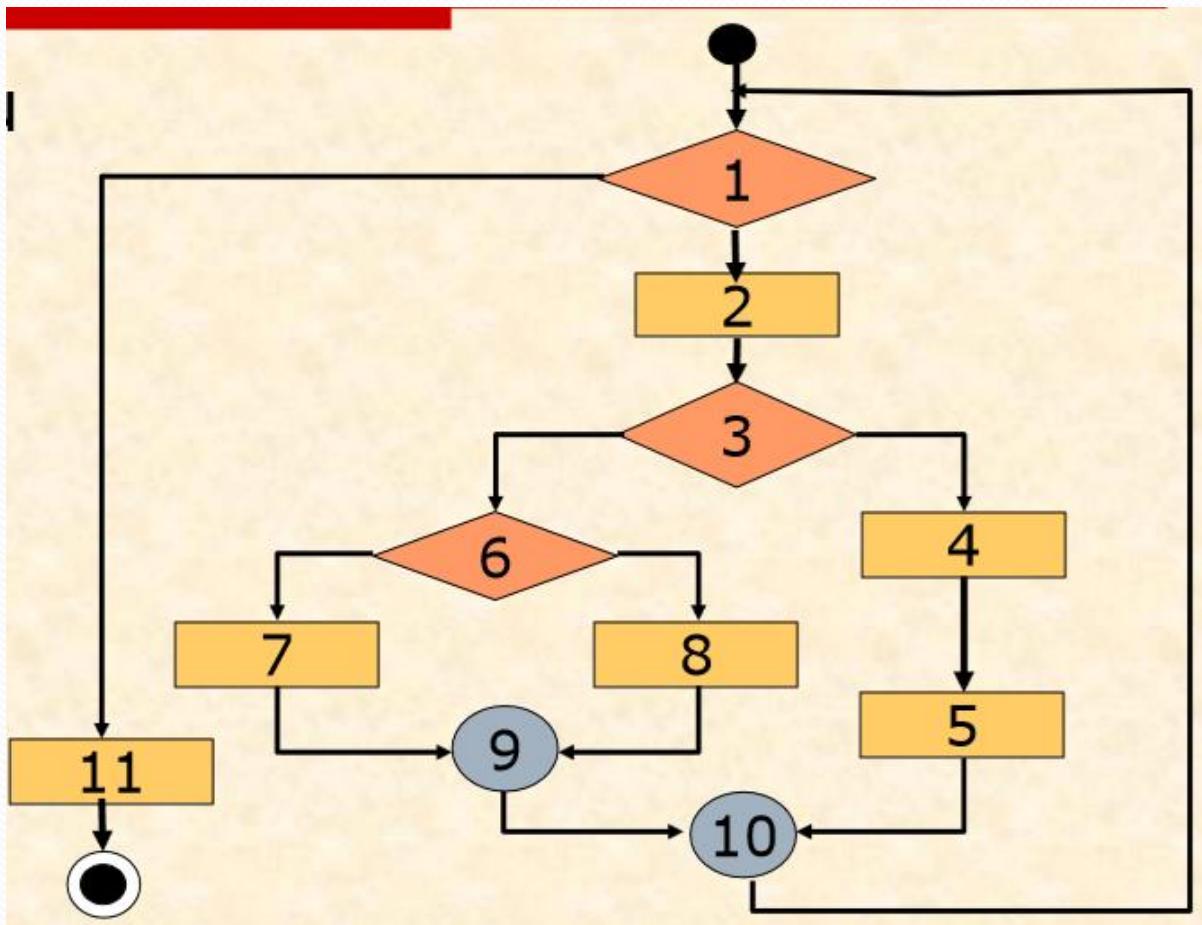
- Câu trúc gồm:
  - một nút hình tròn biểu thị một hay một số lệnh tuần tự hoặc thay cho điểm hội tụ các đường điều khiển
  - Mỗi cạnh nối hai nút biểu diễn dòng điều khiển
- Kết quả đồ thị dòng
  - Chia mặt phẳng thành nhiều miền
  - Có nút vị tự biểu diễn phân nhánh hoặc hội nhập của các cung

# Các kiểu cấu trúc thành phần đồ thị dòng

- Cấu trúc cơ bản của đồ thị dòng:

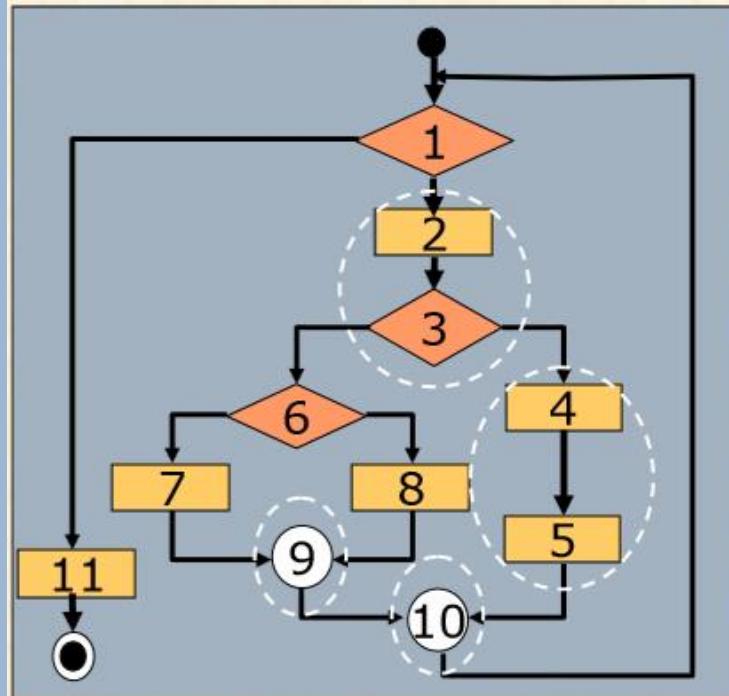


# Ví dụ cấu trúc điều khiển chương trình

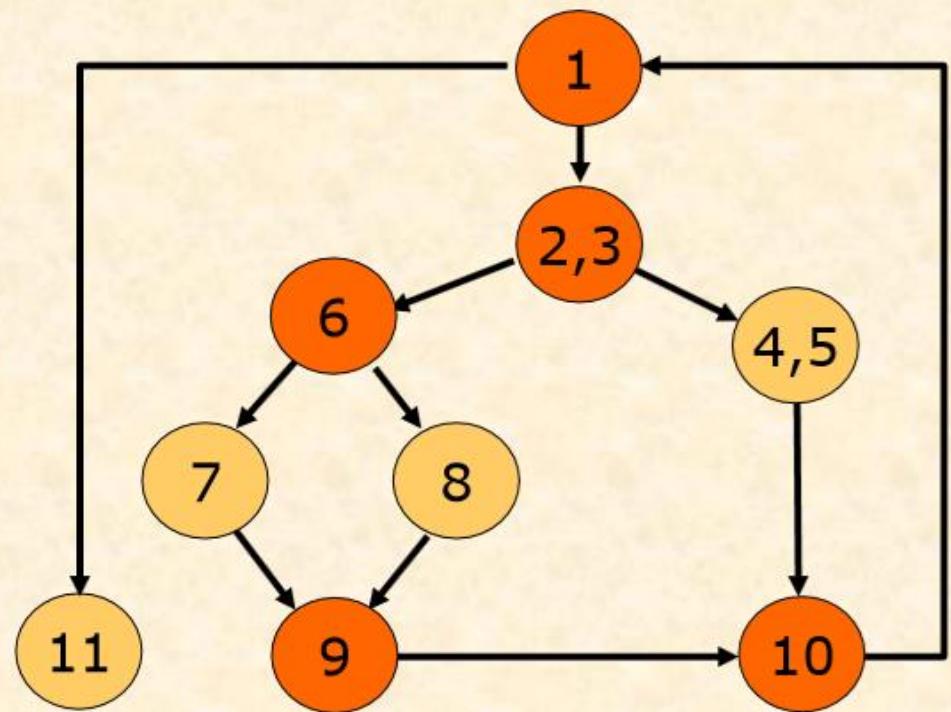


# Ví dụ cấu trúc đồ thị dòng

## ■ luồng điều khiển



## ■ đồ thị dòng



# Ví dụ: xác định các thông số

---

- Đồ thị dòng trên gồm:
  - 9 nút trong đó:
    - 5 nút là vị tự (màu đỏ)
    - 11 cung
    - Chia mặt phẳng thành 4 miền

# Độ phức tạp của chu trình

---

- Để đảm bảo mọi câu lệnh điều khiển được kiểm thử ít nhất 1 lần, cần tìm được tất cả các đường điều khiển độc lập trong chương trình (khác với các đường khác ít nhất 1 lệnh)
- Số các đường độc lập của một chương trình là giới hạn trên số các kiểm thử cần phải tiến hành. Nó được gọi là độ phức tạp chu trình của chương trình.
- Các đường độc lập của một chương trình trùng với các đường độc lập của đồ thị dòng (tìm đơn giản hơn)

# Tính toán độ phức tạp chu trình

---

- Độ phức tạp chu trình  $V(G)$  của đồ thị  $G$  được tính theo các cách sau:

$$V(G) = E - N + 2 \quad (=11 - 9 + 2 = 4)$$

$$V(G) = \text{số miền phẳng} \quad (=4)$$

$$V(G) = P - 1 \quad (=5 - 1 = 4)$$

Trong đó:

$E = \text{số cung}$ ,  $N = \text{số nút}$ ,  $P = \text{số nút vị từ}$

Với ví dụ về độ thì dòng ở trên ta có:  $V(G) = 4$

# Xác định các ca kiểm thử



# Kỹ thuật ma trận kiểm thử

---

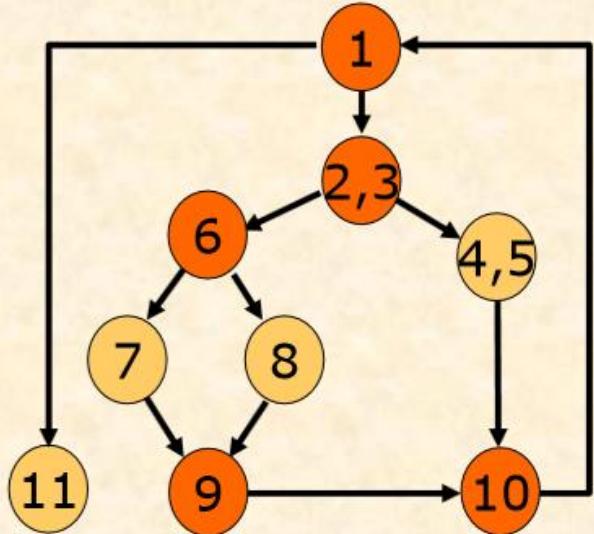
- Ma trận kiểm thử là ma trận vuông có kích thước bằng số các nút trong đồ thị dòng
  - Mỗi dòng/cột ứng với tên một nút
  - Mỗi ô là tên một cung nối nút dòng đến nút cột
- Nhân liên tiếp k ma trận này được ma trận chỉ số con đường k cung từ nút dòng tới nút cột
- Ma trận kiểm thử được sử dụng như một dữ liệu có cấu trúc để kiểm tra các con đường cơ bản: số đường đi qua nút.

# Các ma trận kiểm thử có trọng số

---

- Để ma trận kiểm thử - một công cụ mạnh trong việc đánh giá cấu trúc điều khiển chương trình. Khi kiểm thử ta thêm trọng số cho các cung của ma trận kiểm thử như sau:
  - Xác suất cung đó được thực thi
  - Thời gian xử lý của tiến trình đi qua cung đó
  - Bộ nhớ đòi hỏi của tiến trình đi qua cung đó
  - Nguồn lực đòi hỏi của tiến trình đi qua cung đó

# Ví dụ ma trận kiểm thử



	1	23	45	6	7	8	9	10	11	
1		1								1
23			1	1						
45									1	
6					1	1				
7							1			
8								1		
9									1	
10	1									
11										

= A

# Ví dụ ma trận kiểm thử

Các số trong ma trận cho biết số con đường có hai cạnh đi qua cung đó

$$A^2 =$$

	1	23	45	6	7	8	9	10	11
1			1	1					
23					1	1			
45	1								
6							2		
7								1	
8								1	
9									
10									
11		1							1

# Điều kiện logic và các chiến lược

---

- Trong một chương trình, điều kiện logic có thể là:
  - Điều kiện đơn một biến Bool(có thể có toán tử phủ định): X
  - Điều kiện đơn là biểu thức quan hệ giữa hai biểu thức số học giữa A và B với các phép so sánh:<, =, ≤, ≥, ≠
  - Điều kiện phức hợp: cấu thành từ hơn một điều kiện đơn nhờ các toán tử Bool: hoặc, và, phủ định

# Chiến lược kiểm thử phân nhánh

---

- Kiểm thử từng điều kiện trong chương trình
- Kiểm thử điều kiện không chỉ là phát hiện sai điều kiện đó mà còn phát hiện sai khác của chương trình liên quan
- Đã có một số chiến lược kiểm thử
- Nguyên tắc kiểm thử nhánh: Với mỗi điều kiện phức hợp C, thì với mỗi nhánh true và false của C mỗi điều kiện đơn trong C phải được kiểm thử ít nhất 1 lần.

# Chiến lược kiểm thử phần mềm

---

- Chiến lược kiểm thử miền cần 3 hoặc 4 kiểm thử cho một biểu thức quan hệ gồm các trường hợp:  $<$ ,  $>$ ,  $=$  có thể là  $\neq$  nữa.
- Nếu biểu thức Bool có n biến, mà n nhỏ thì thuận lợi, song n lớn thì khó thực hiện.
- Người ta đưa ra các chiến lược cho các phép thử nhạy cảm bằng cách áp dụng kết hợp chiến lược kiểm thử nhánh và kiểm thử miền (quan hệ).

Làm sao chỉ ra tất cả các trường hợp cần kiểm thử.

# Chiến lược kiểm thử BRO

---

- BRO= Kiểm thử nhánh và toán tử quan hệ (branch and relational operation)
- BRO dùng “ràng buộc điều kiện làm điều kiện cần thử” để phát hiện sai ở nhánh và toán tử khi xảy ra một lần và không có biến chung.

(Đọc thêm về kiểm thử BRO)

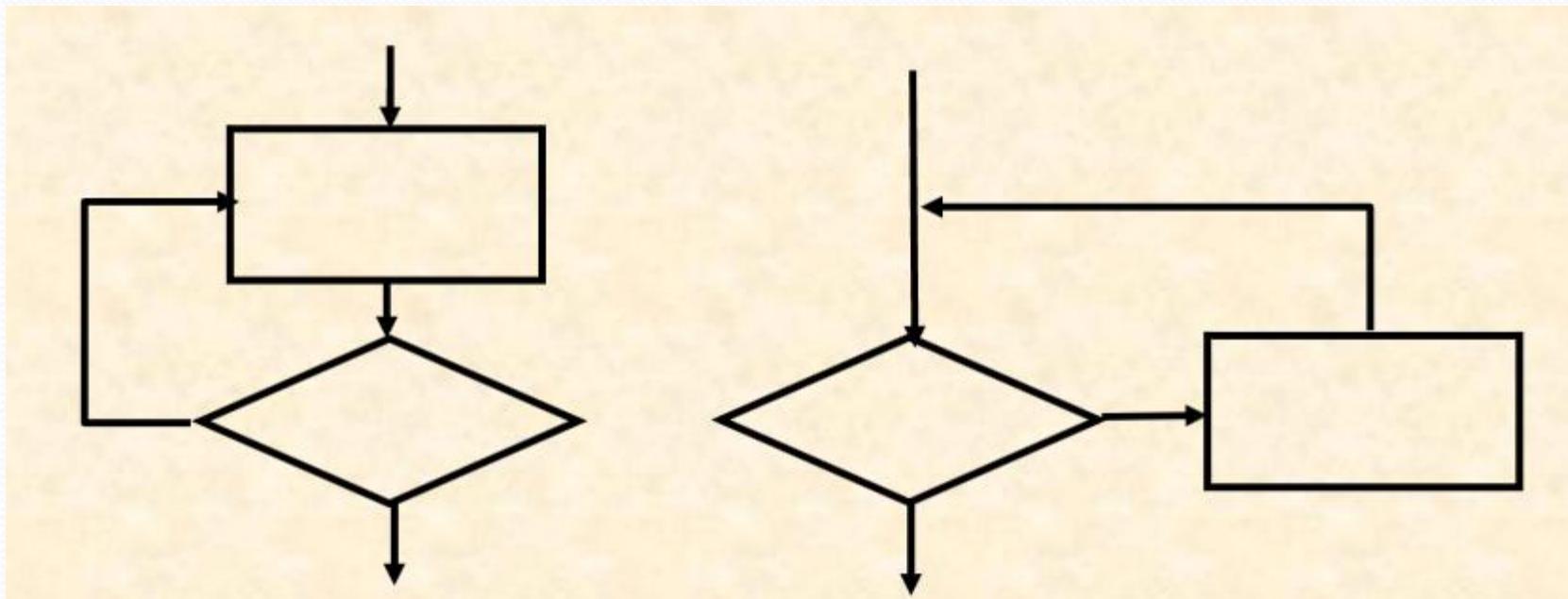
# Kiểm thử điều khiển theo dòng dữ liệu

---

- Phương pháp kiểm thử dòng dữ liệu tuyển chọn các đường của chương trình tương ứng với việc định vị các xác định biến và sử dụng biến trong chương trình. Đã có một số chiến lược kiểm thử dòng dữ liệu và so sánh chúng.
- Giả sử rằng mỗi câu lệnh của chương trình được gán với số câu lệnh duy nhất và mỗi hàm không được cải biên các tham số của nó cho các biến toàn cục.

# Kiểm thử điều khiển theo vòng lặp

- Có 4 loại vòng lặp: mỗi loại dùng một tập các phép thử khác nhau
- Vòng lặp đơn:



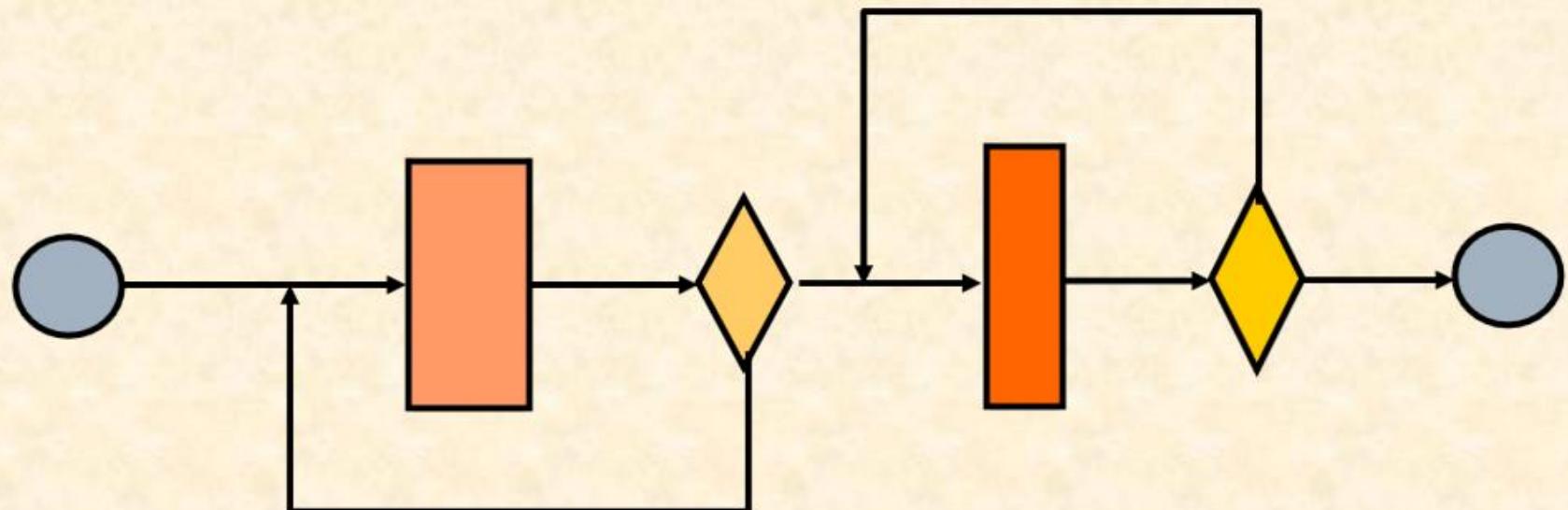
# Kiểm thử điều khiển theo vòng lặp

---

- Kiểu vòng lặp lồng nhau

# Kiểm thử điều khiển theo vòng lặp

- Kiểu vòng lặp nối tiếp

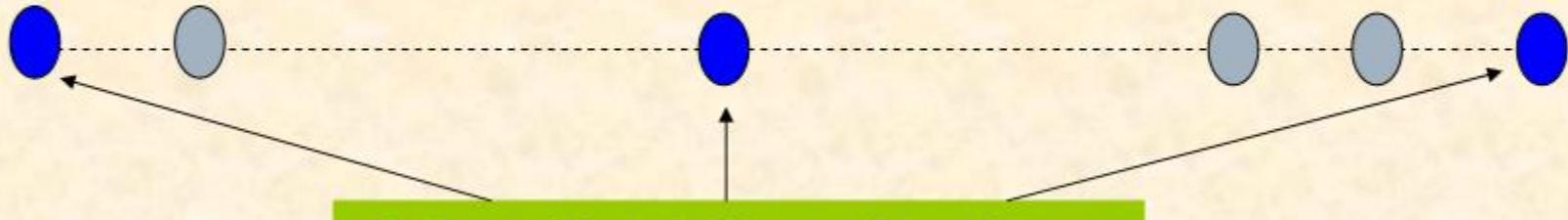


Kết hợp mỗi vòng lặp trước với mọi vòng lặp sau

# Kiểm thử điều khiển theo vòng lặp

- Chọn giá trị cho mỗi loại vòng lặp
- Với mỗi loại vòng lặp thường chỉ sử dụng ba giá trị lặp: hai giá trị biên và một giá trị giữa hai biên.

Các giá trị lặp của một vòng lặp

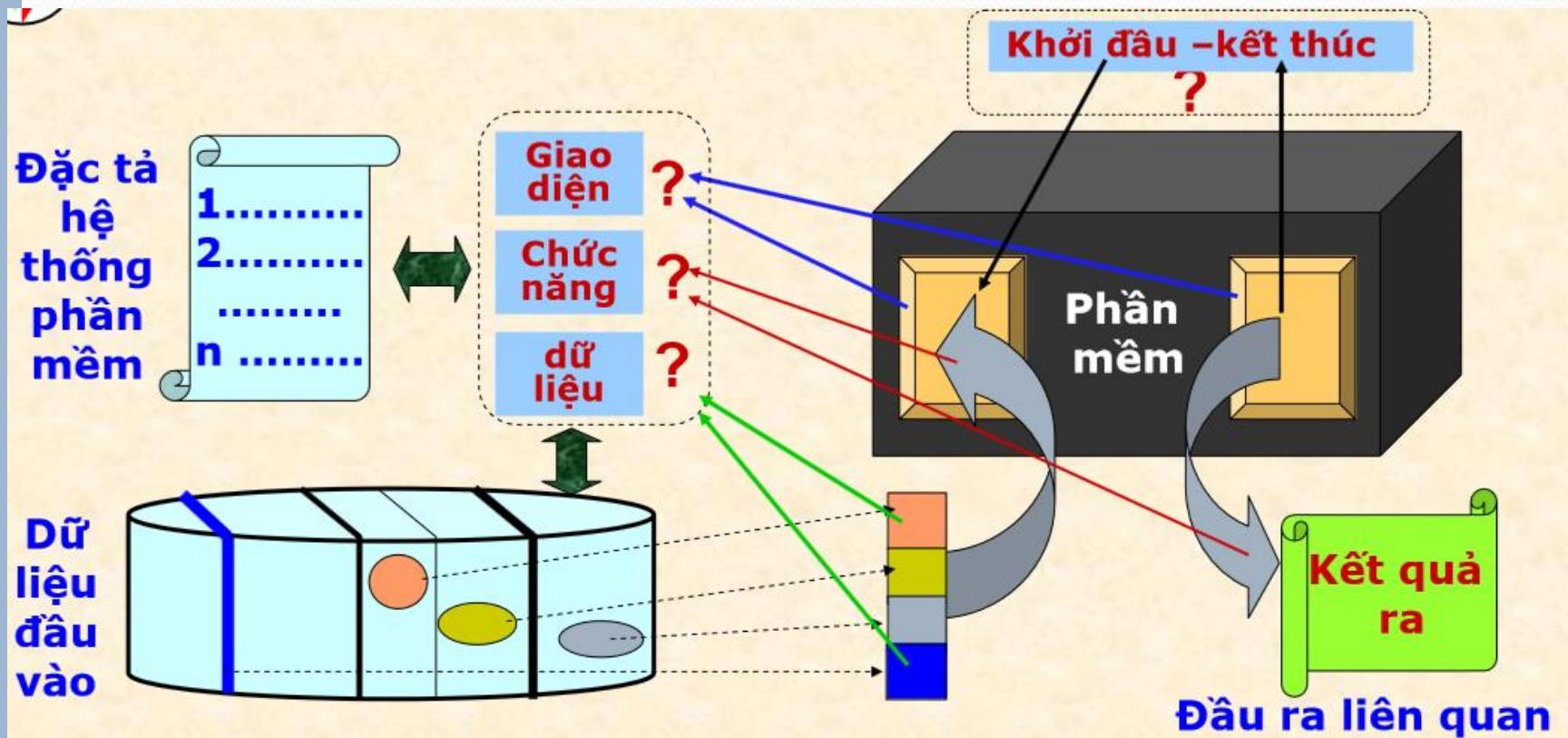


# Kiểm thử hộp đen

---

- Khái niệm
  - Là kiểm thử yêu cầu chức năng
  - Đối tượng: modun, hệ con, toàn hệ thống
  - Đặc trưng:
    - + Thuyết minh: các chức năng đủ & vận hành đúng
    - + thực hiện: qua giao diện
    - + Cơ sở: đặc tả, điều kiện vào/ra và cấu trúc dữ liệu
    - + Ít chú ý tới cấu trúc logic và nội tại của nó

# Mô hình khái niệm kiểm thử hộp đen



# Mục đích kiểm thử hộp đen

---

- Tìm các loại sai liên quan:
  - Chức năng: đủ, đúng đắn
  - Giao diện: vào/ra: đủ, phù hợp, đúng, tiện lợi
  - Cấu trúc truy nhập dữ liệu: thông suốt, đúng đắn
  - Thực thi: trôi chảy, kịp thời, chịu lỗi, phục hồi
  - Khởi đầu- kết thúc: mỗi tiến trình thông suốt

# Câu hỏi cho kiểm thử hộp đen

---

- Các câu hỏi tập trung trả lời:
  - Hiệu lực của chức năng (chức năng, hiệu suất, giao diện) đạt được đến đâu
  - Lớp đầu vào nào cho các ca kiểm thử tốt
  - Sự nhạy cảm của modun với giá trị vào nào
  - Các biên của lớp dữ liệu được cô lập chưa
  - Chịu lỗi với nhịp điệu /khối lượng dữ liệu như thế nào
  - Tổ hợp dữ liệu đặc biệt ảnh hưởng gì tới hoạt động hệ thống
  - Những tiến trình nào (khởi đầu, kết thúc) chưa thông suốt.

# Vấn đề và tiêu chuẩn lựa chọn

---

- Vấn đề:
  - Các tiến trình của mỗi chức năng hệ thống đủ lớn
  - Các dữ liệu dày đặc, đa dạng
  - Không dự kiến tới mọi sự bất thường
- Tiêu chuẩn hướng đến
  - Thu gọn ca kiểm thử đến mức có thể (ít, đơn giản)
  - Phát hiện sai trên lớp dữ liệu, số đặc biệt (không phải một sai cụ thể gắn với một kiểm thử cụ thể)

# Các kỹ thuật kiểm thử hộp đen

---

- 1. Phân hoạch tương đương
- 2. Phân tích giá trị biên
- 3. Đồ thị nhân quả

# Phân hoạch tương đương

---

- Là một kỹ thuật của kiểm thử hộp đen
- Nguyên tắc: chia miền vào của chương trình thành các lớp dữ liệu để lập ra các ca kiểm thử theo mỗi lớp đó
- Cơ sở: dữ liệu trong một lớp tương đương tác động như nhau lên chương trình, tạo ra cùng một trạng thái: đúng hay sai của chương trình
- Mục tiêu: tìm ra một ca kiểm thử để bộc lộ 1 lớp sai -> rút gọn số ca kiểm thử cần phát triển
- Ca kiểm thử được thiết kế cho từng lớp tương đương

# Phân hoạch tương đương

---

- Ví dụ: Ta cần kiểm thử thành phần phần mềm “quản lý nguồn nhân lực” với đặc tả chức năng như sau: Mỗi lần nhận hồ sơ xin việc, thành phần phần mềm sẽ ra quyết định dựa vào tuổi ứng viên theo bảng sau:

Tuổi ứng viên	Kết quả
0-16	Không thuê
16-18	Thuê dạng bán thời gian
18-55	Thuê toàn thời gian
55-99	Không thuê

# Phân hoạch tương đương

---

- Phân tích đặc tả chức năng của phần nền cần kiểm thử, ta thấy có 4 lớp tương đương, mỗi lớp chứa các test case ứng với một chế độ xử lý của phần mềm: không thuê vì quá trẻ, thuê dạng bán thời gian, thuê toàn thời gian, không thuê vì quá già.
- Ứng với mỗi lớp tương đương, ta định nghĩa một testcase đại diện. Ví dụ chọn 4 testcase sau:
  1. Testcase 1 : {Input : 2 tuổi, Output : không thuê}
  2. Testcase 2 : {Input : 17 tuổi, Output : thuê bán thời gian}
  3. Testcase 3 : {Input : 35 tuổi, Output : thuê toàn thời gian}
  4. Testcase 4 : {Input : 90 tuổi, Output : không thuê}

# Phân hoạch tương đương

---

- Trong ví dụ trên thay vì phải kiểm thử 100 test case ta chỉ kiểm thử 4 test case do đó chi phí giảm rất lớn mà chất lượng gần như không bị giảm sút.

# Phân hoạch tương đương

---

- Ví dụ: Chương trình nhập vào 3 số thực, kiểm tra ba số thực đó có là độ dài ba cạnh của một tam giác hay không. Nếu là độ dài ba cạnh của một tam giác thì kiểm tra xem đó là tam giác thường, cân đều cũng như kiểm tra xem tam giác đó là nhọn, vuông hay tù?

# Phân hoạch tương đương

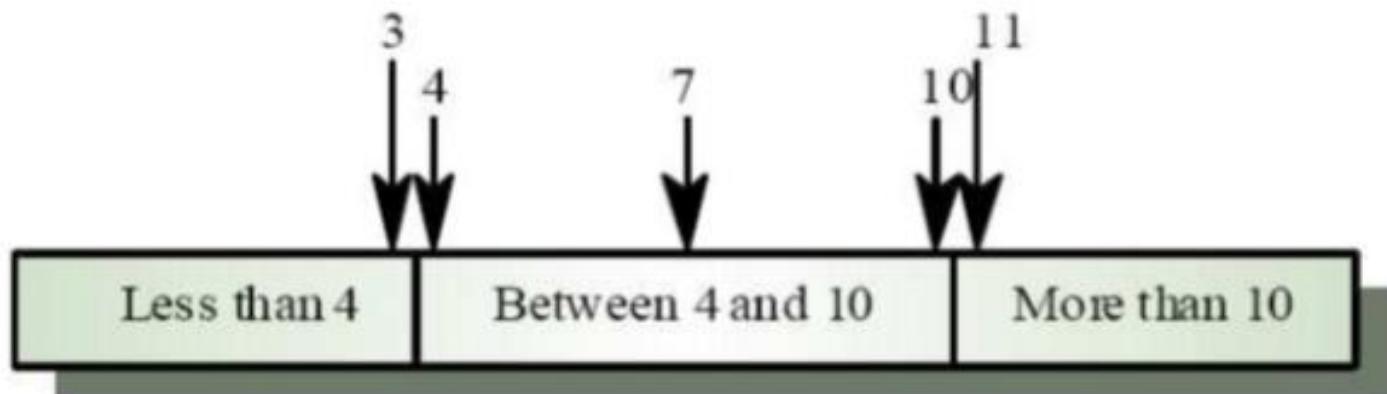
- Xét các lớp tương đương

	Nhọn	Vuông	Tù
Thường	6,5,3	5,6,10	3,4,5
Cân	6,1,6	7,4,4	$\sqrt{2}, 2, \sqrt{2}$
Đều	4,4,4	không thê	không thê
Không là tam giác	-	1,2,8	

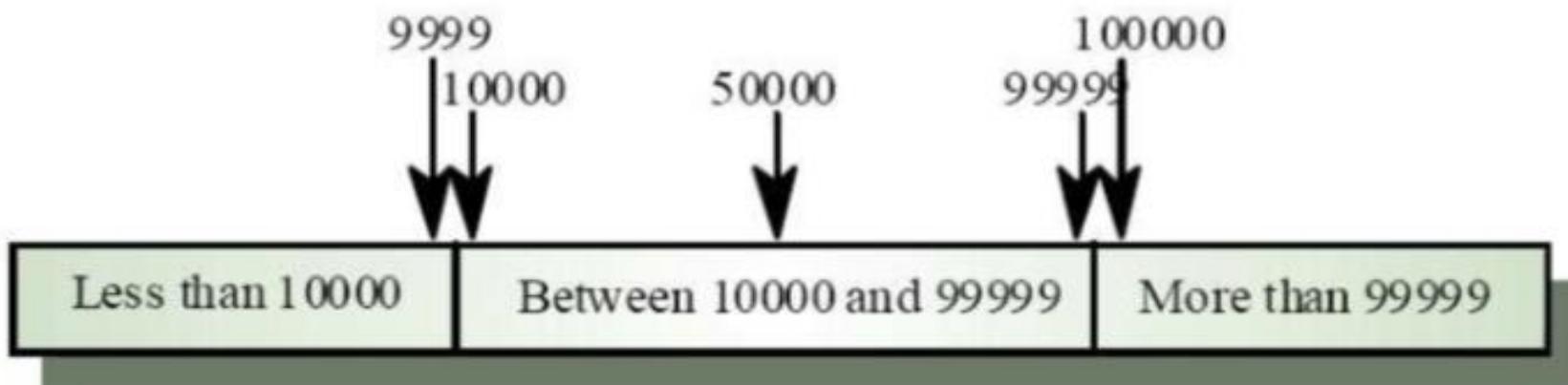
# Kiểm thử giá trị biên

- Cơ sở
  - lỗi thường xuất hiện gần các giá trị biên của miền dữ liệu
- Tập trung phân tích các giá trị biên của miền dữ liệu để xây dựng dữ liệu kiểm thử
- Nguyên tắc: kiểm thử các dữ liệu vào gồm
  - giá trị nhỏ nhất
  - giá trị gần kề lớn hơn giá trị nhỏ nhất
  - giá trị bình thường
  - giá trị gần kề nhỏ hơn giá trị lớn nhất
  - giá trị lớn nhất

# Kiểm thử giá trị biên



Number of input values



Input values

# Kiểm thử giá trị biên

- Nguyên tắc chọn dữ liệu thử
  - Nếu dữ liệu vào thuộc một khoảng, chọn
    - 2 giá trị biên
    - $4 \text{ giá trị} = \text{giá trị biên} \pm \text{sai số nhỏ nhất}$
  - Nếu giá trị vào thuộc danh sách các giá trị, chọn
    - phần tử thứ nhất, phần tử thứ hai, phần tử kế cuối và phần tử cuối
  - Nếu dữ liệu vào là điều kiện ràng buộc số giá trị, chọn
    - số giá trị tối thiểu, số giá trị tối đa và một số các số giá trị không hợp lệ
  - Tự vận dụng khả năng và thực tế để chọn các giá trị biên cần kiểm thử

# Kiểm thử giá trị biên

---

- Ví dụ: Chương trình nhập vào 3 số thực, kiểm tra ba số thực đó có là độ dài ba cạnh của một tam giác hay không. Nếu là độ dài ba cạnh của một tam giác thì kiểm tra xem đó là tam giác thường, cân đều cũng như kiểm tra xem tam giác đó là nhọn, vuông hay tù?

# Kiểm thử giá trị biên

- Ví dụ (2)

- Dữ liệu thử

- 1, 1, 2

- Không là tam giác

- 0, 0, 0

- Chỉ một điểm

- 4, 0, 3

- Một cạnh bằng không

- 1, 2, 3.00001

- Gần là một tam giác

- 0.001, 0.001, 0.001

- Tam giác rất nhỏ

- 99999, 99999, 99999

- Tam giác rất lớn

- 3.00001, 3, 3

- Tam giác gần đều

- 2.99999, 3, 4

- Tam giác gần cân

- 3, 4, 5.00001

- Tam giác giác gần vuông

- 3, 4, 5, 6

- Bốn giá trị

- 3

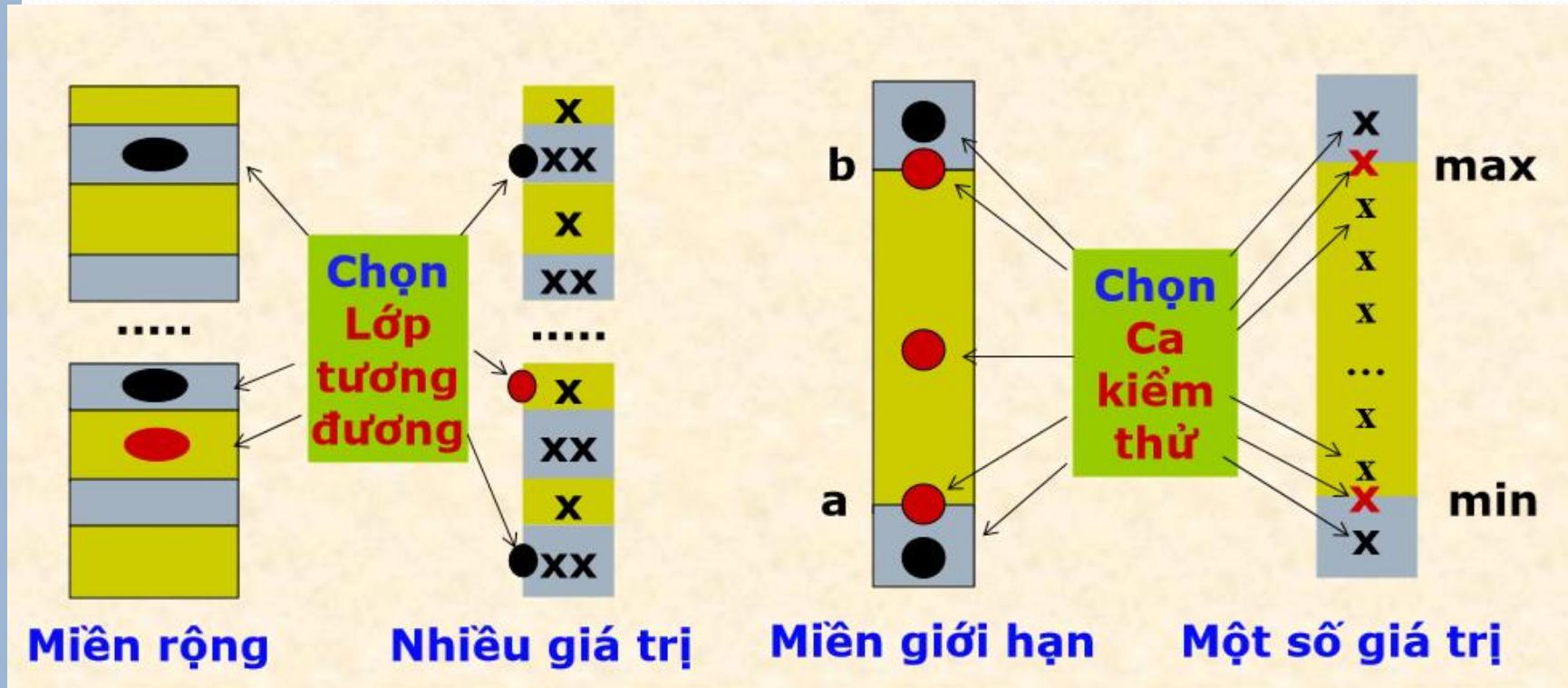
- Chỉ một giá trị

- 3, -3, 5

- Dữ liệu vào rỗng

- Giá trị âm

# Mô hình phân hoạch và phân tích giá trị biên

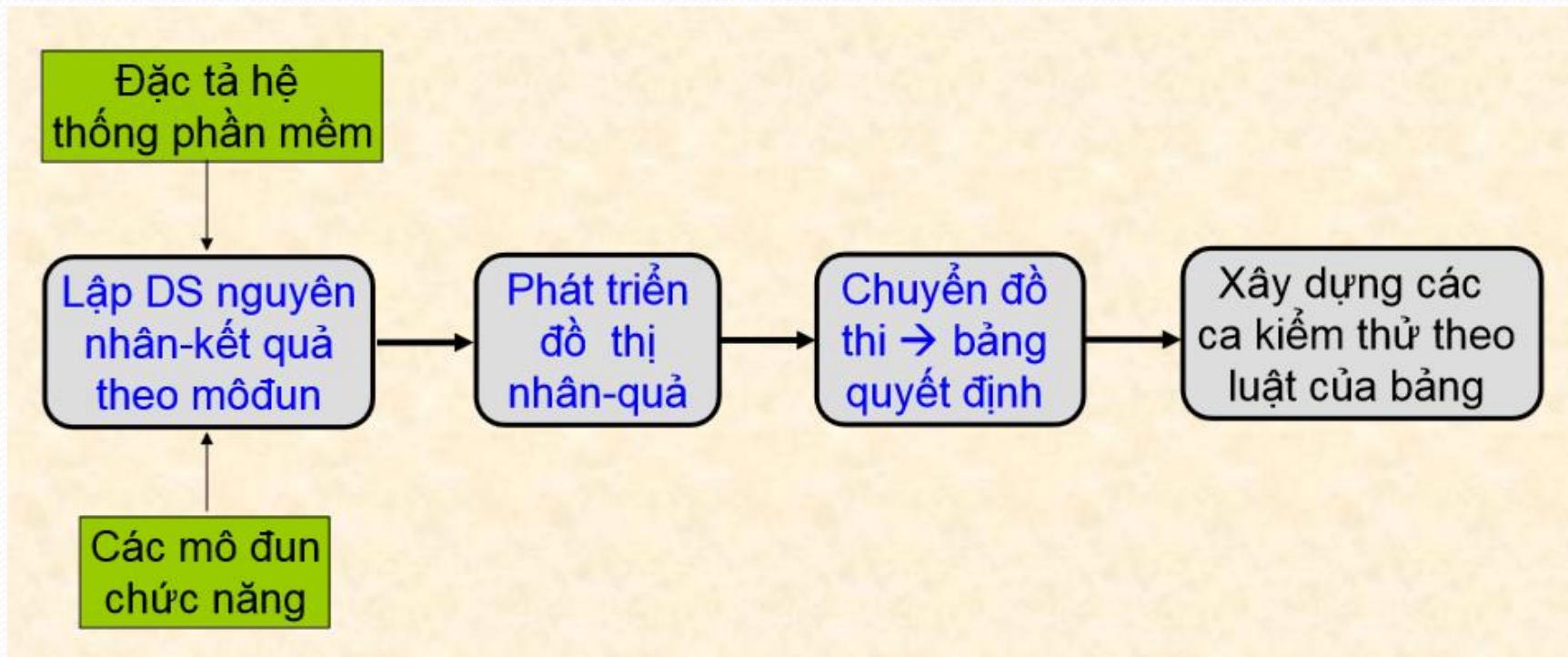


# Kỹ thuật đồ thị nhân quả

---

- Là một kỹ thuật để thiết kế ca kiểm thử
- Cung cấp một biểu diễn chính xác giữa các điều kiện logic (đầu vào) và các hành động tương ứng (đầu ra/kết quả)
- Được xây dựng dựa trên các modun chức năng, logic tiến trình và đặc tả hệ thống.
- Kỹ thuật gồm 4 bước:

# Các bước tiến hành



# Ví dụ kỹ thuật đồ thị nhân quả

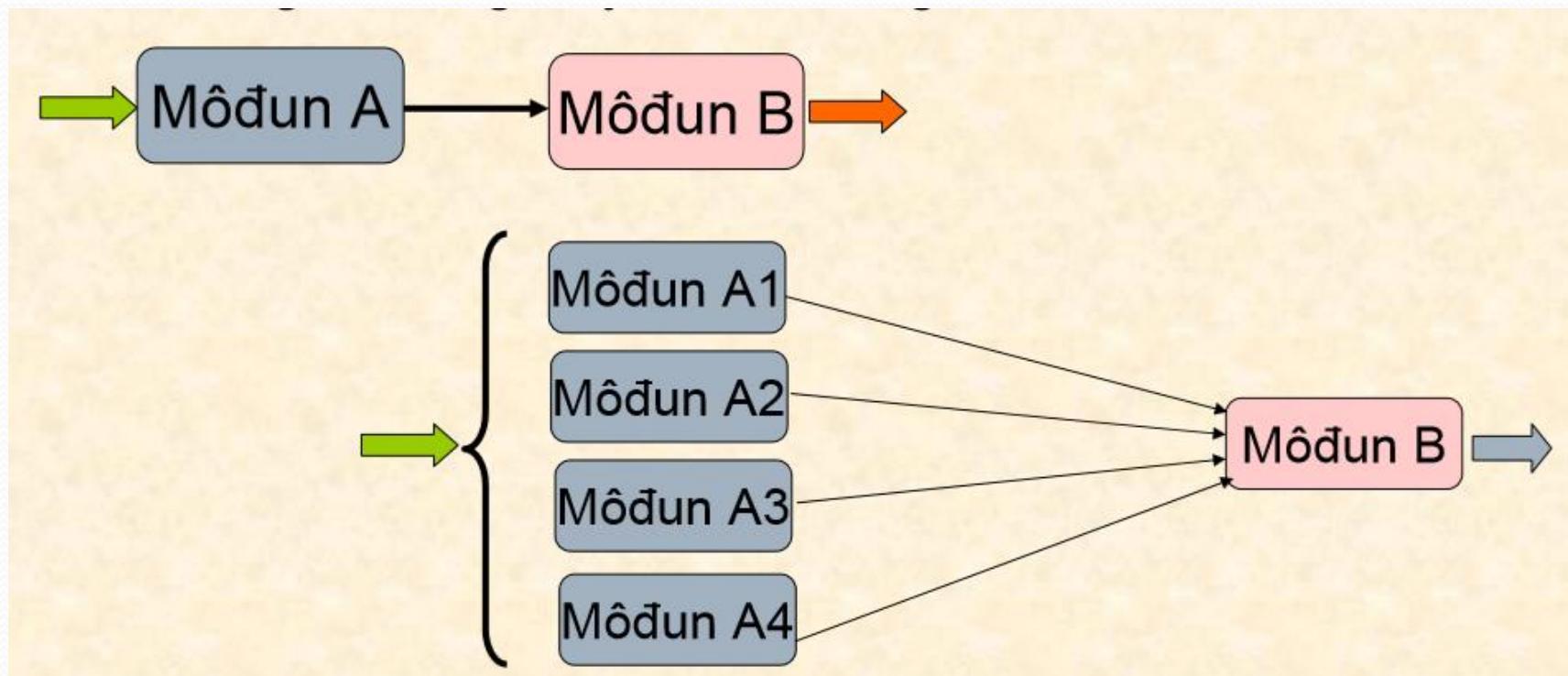
- Danh sách nhân quả theo modul

## Danh sách nhân quả theo modul

Modul	Nguyên nhân	Kết quả	Định danh
A	Số $> a$	đúng	A1
	Số $\geq a$	nghi ngờ	A2
	Số $= a$	nghi ngờ	A3
	Số $< a$	sai	A4
B	Số nguyên	đúng	B1

# Ví dụ kỹ thuật đồ thị nhân quả

- Có nhiều công cụ để xây dựng đồ thị nhân quả. Đồ thị có hướng thường được dùng hơn cả

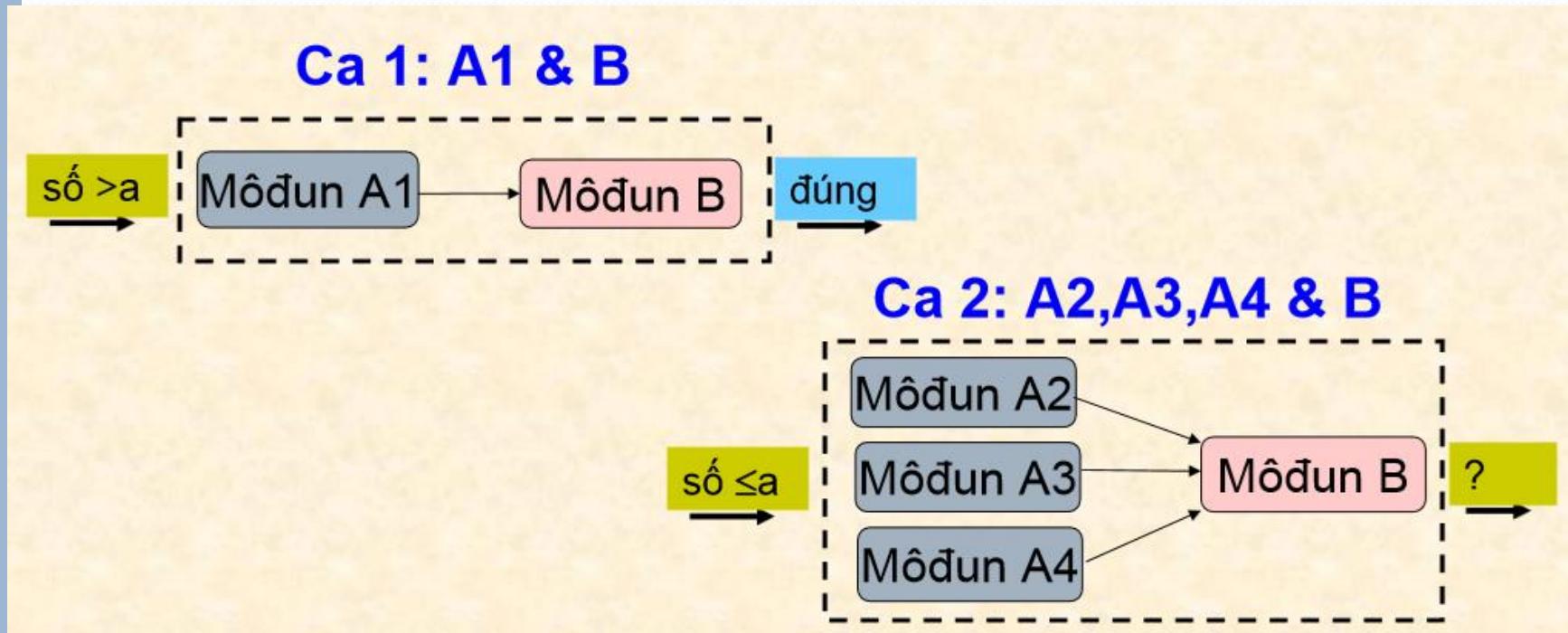


# Bảng quyết định của đồ thị nhân quả

Định danh	Điều kiện	đúng	nghi ngờ	Sai
A1	Số $> a$	X		
A2,A3	Số $\geq a$		X	
A4	Số $< a$			X
B1	Số nguyên	X		
...	...	..	..	..

# Chọn ca kiểm thử

Chọn hai ca kiểm thử



# Những hạn chế của kiểm thử hộp đen

- Khi hệ thống có nhiều loại dữ liệu đầu vào khác nhau, điều này có thể dẫn tới sự bùng nổ các ca kiểm thử.
- Do không thể biết được modun nào của chương trình đã hay chưa được kiểm thử, khi đó phải kiểm thử lại hay bỏ qua những lỗi tiềm ẩn trong gói phần mềm.
- Nếu không có bản đặc tả rõ ràng súc tích rất khó thiết kế các ca kiểm thử đúng đắn

# Các loại hình kiểm thử

---

- Chiến lược kiểm thử tổng thể một hệ thống được sử dụng rộng rãi nhất hiện nay là chiến lược từ mức thấp đến mức cao bao gồm:
  - Kiểm thử đơn vị
  - Kiểm thử tích hợp
  - Kiểm thử hệ thống
  - Kiểm thử chấp nhận

# Kiểm thử đơn vị

---

- Nó nhắm vào kiểm tra đơn vị thiết kế nhỏ nhất – một modun phần mềm.
- Người tiến hành kiểm thử đơn vị thường là người lập trình modun đó hoặc lập trình viên cùng nhóm
- Kỹ thuật kiểm thử đơn vị thường là kiểm thử hộp trắng.
- Nội dung kiểm thử đơn vị bao gồm: kiểm thử giao diện, kiểm thử vào/ra, kiểm thử cấu trúc dữ liệu cục bộ, kiểm thử xử lý, kiểm thử điều kiện logic, kiểm thử sai tiệm ẩn, kiểm thử các giá trị biên.

# Kiểm thử tích hợp

---

- Những lỗi cần phát hiện khi kiểm thử tích hợp là lỗi dữ liệu qua giao diện, hiệu ứng của một modun gây lỗi qua modun khác, lỗi xung đột về bộ nhớ hay các thiết bị ngoại vi... Một số lỗi giao diện modun điển hình:
  - Sử dụng sai giao diện
  - Hiểu nhầm về giao diện
  - Xung đột
  - Tích hợp từng bước: tích hợp từ dưới lên, tích hợp từ trên xuống, kết hợp cả dưới lên và trên xuống
  - Tích hợp đồng thời

# Kiểm thử hệ thống

---

- Sau khi tích hợp tất cả các modun thành một hệ thống hoàn thiện, ta chuyển sang kiểm thử khả năng hoạt động của cả hệ thống.
- Kiểm thử hiệu năng của hệ thống, khả năng phục hồi khi gặp sự cố, độ tin cậy...
- Việc kiểm thử hệ thống do nhóm kiểm thử độc lập chuyên về hệ thống thực hiện.
- Phương pháp chính sử dụng là kiểm thử hộp đen
- Ngoài ra còn một số cách kiểm thử hệ thống: kiểm thử phục hồi, kiểm thử áp lực, kiểm thử hiệu năng, kiểm thử an ninh, kiểm thử so sánh.

# Kiểm thử chấp nhận

---

- Sau khi kiểm thử hệ thống là kiểm thử chấp nhận hay kiểm thử thẩm định được khách hàng thực hiện.
- Mục đích của kiểm thử này là để khách hàng đánh giá xem phần mềm có thỏa mãn tất cả các yêu cầu của họ hay không để chấp nhận sản phẩm

# Đảm bảo chất lượng phần mềm(Software Quality Assurance- SQA)

- 1. Đảm bảo chất lượng phần mềm

- Định nghĩa

- Chất lượng phần mềm là gì?

- **Chất lượng một sản phẩm:**

Chất lượng sản phẩm là mức độ đạt được các đặc trưng hay những thuộc tính nào đó của nó (*Từ điển American Heritage*). Chẳng hạn:

- **Chất lượng thiết kế** (cấu trúc)
- **Sự hoàn thiện** (tính năng, kiểu dáng,...)
- **Sự lâu bền** (thời gian dùng, độ mòn cũ,...)

## ■ Định nghĩa (khác):

Chất lượng của sản phẩm được thể hiện bằng các đặc trưng phù hợp với đặc tả của nó [Crosby, 1979]

## ■ Định nghĩa chất lượng phần mềm

Chất lượng phần mềm là sự đáp ứng các yêu cầu chức năng, sự hoàn thiện và các chuẩn (đặc tả) được phát triển, các đặc trưng mong chờ từ mọi phần mềm chuyên nghiệp (ngầm định).

# Vấn đề chất lượng với phần mềm

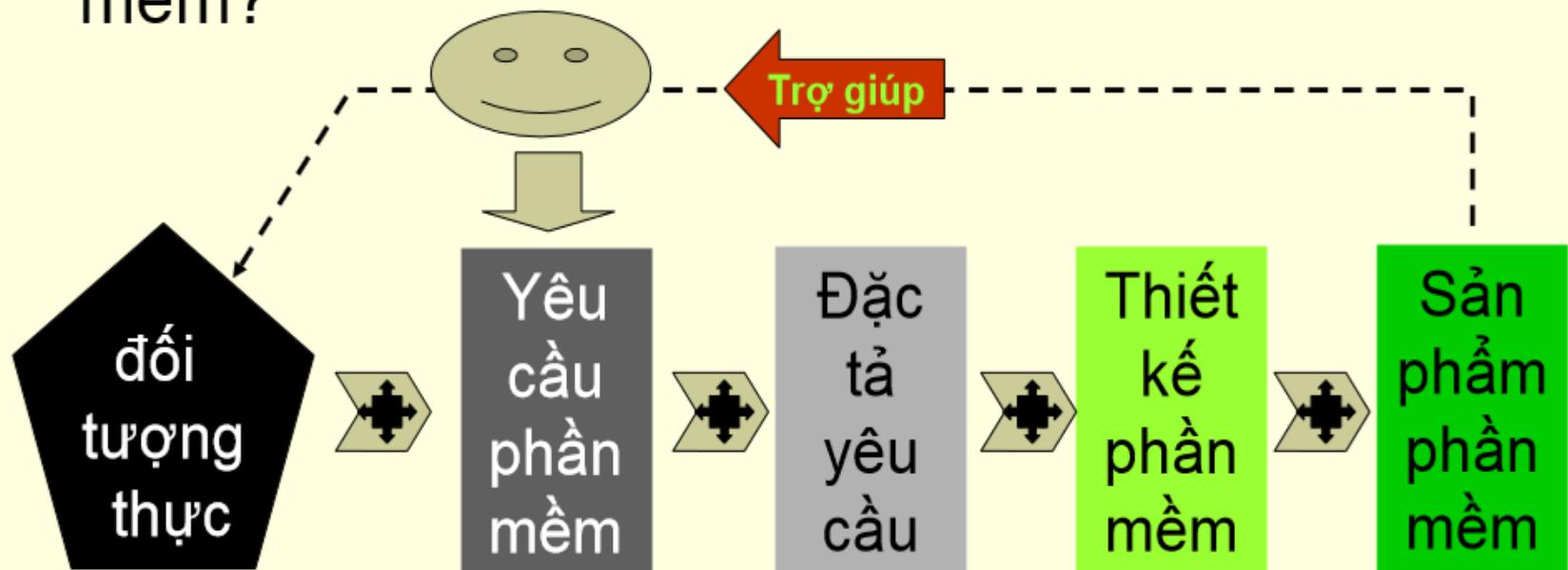
- Chất lượng phần mềm nên xem xét thế nào?
  - Định nghĩa trên là chung cho mọi sản phẩm.
  - Với phần mềm có những vấn đề gì đặt ra?
- Phần mềm có những đặc trưng gì? → chất lượng?

# Vấn đề chất lượng với phần mềm

- Chất lượng phần mềm cần xem xét như thế nào?
  - Phần mềm là vô hình
  - Phần mềm không mòn cũ, hỏng hóc, nhưng thoái hóa
  - Phần mềm thay đổi theo thời gian
- Với phần mềm có một số vấn đề sau:
  - Yêu cầu có thể bị bỏ sót
  - Các yêu cầu tự nhiên nên không được đặc tả
  - Phần mềm có yêu cầu mà chưa có đặc tả
  - Phần mềm có đặc tả nhưng lại mờ

# Vấn đề chất lượng với phần mềm

- Cái gì là **Cơ sở** xem xét chất lượng phần mềm?



**Sự hình thành sản phẩm phần mềm bắt đầu từ yêu cầu**

# Cơ sở xem xét chất lượng phần mềm

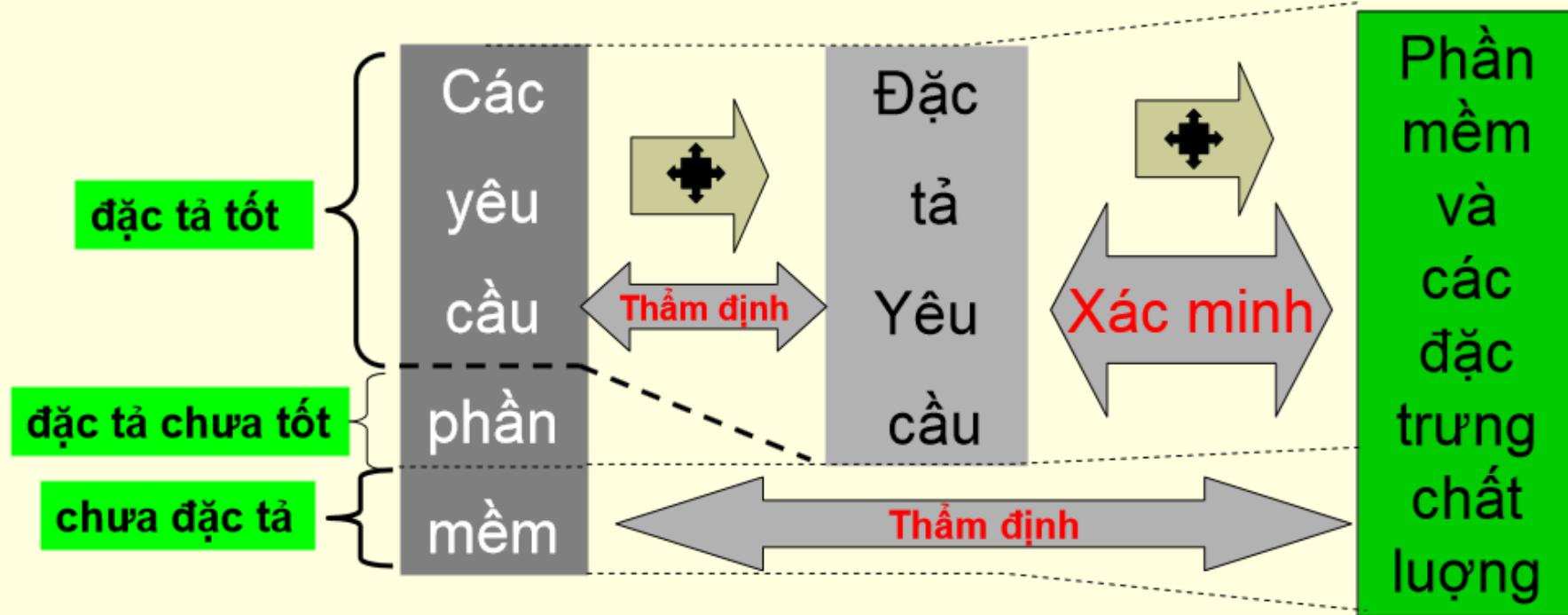
Yêu cầu phần mềm là cơ sở xem xét:

- Xem xét chất lượng:
  - *Sự phù hợp với yêu cầu là có chất lượng*
  - *Phù hợp yêu cầu cả về số lượng & mô tả*
- Yêu cầu thể hiện ra bằng đặc tả - đặc tả có chuẩn mới kiểm tra, đo đạc được
  - *Các chuẩn đặc tả là một bộ các tiêu chuẩn phát triển, và hướng dẫn cách thức làm ra phần mềm.*
  - *Không tuân thủ các tiêu chuẩn đó thì hầu chắc chắn là chất lượng sẽ kém.*

# Những vấn đề liên quan đến đặc tả

- Bỏ sót yêu cầu
- Có các yêu cầu ngầm thường ít được nhắc đến:
  - Quá thông dụng, hiển nhiên (*sử dụng cửa sổ*)
  - Ít được thể hiện ra ngoài (*quy tắc nghiệp vụ*)
  - ❖ Phần mềm chưa phù hợp với các yêu cầu ngầm thì chất lượng cũng đáng ngờ
- Có những sai sót khi đặc tả

# Hoạt động kiểm chứng theo sản phẩm



**Thẩm định và xác minh là 2 hoạt động chính của đảm bảo chất lượng phần mềm diễn ra suốt quá trình phát triển**

# Xác minh và thẩm định phần mềm

- Xác minh** là kiểm tra xem phần mềm có đúng đặc tả hay không
- Thẩm định** là kiểm tra xem phần mềm có đáp ứng đúng yêu cầu người dùng hay không
- Do phần mềm có các đặc trưng khác các sản phẩm thông thường khác, nên việc kiểm chứng nó phải được thực hiện đồng thời bằng cả hai hoạt động trên

# Các loại yêu cầu phần mềm

## ■ Yêu cầu người dùng:

- Yêu cầu chức năng (*số lượng, mô tả*)
- Yêu cầu phi chức năng (*đo được*)
- Yêu cầu miền ứng dụng (*chức năng & phi chức năng*)

## ■ Yêu cầu người phát triển:

- Yêu cầu hệ thống: các đặc trưng hệ thống
- Yêu cầu môi trường, công nghệ và công cụ phát triển

# Thẩm định đặc tả yêu cầu

## ■ Lý do: Đặc tả

- **Là cơ sở để phát triển** (*chiều đi*)
- **Là cơ sở để xác minh** (*chiều về*)

➤ **Cân thẩm định đặc tả**

## ■ Các nội dung thẩm định yêu cầu:

- Tính đầy đủ
- Tính chính xác
- Không mâu thuẫn
- Thực hiện được

# Thẩm định phần mềm

## ■ Cơ sở thẩm định:

- Số lượng chức năng
- Mô tả chức năng
- Các yêu cầu phi chức năng (*đo được*)
- Các yêu cầu khác (*chuẩn, công nghệ, công cụ, mong muốn*)

## ■ Các hoạt động thẩm định:

- Rà soát
- Kiểm toán
- Kiểm thử của người dùng

# Xác minh phần mềm

## ■ Cơ sở để xác minh

- Các đặc tả
- Các thiết kế (*nếu có*)

## ■ Hoạt động xác minh:

- Rà soát
- Kiểm thử

# Các hoạt động đảm bảo khác

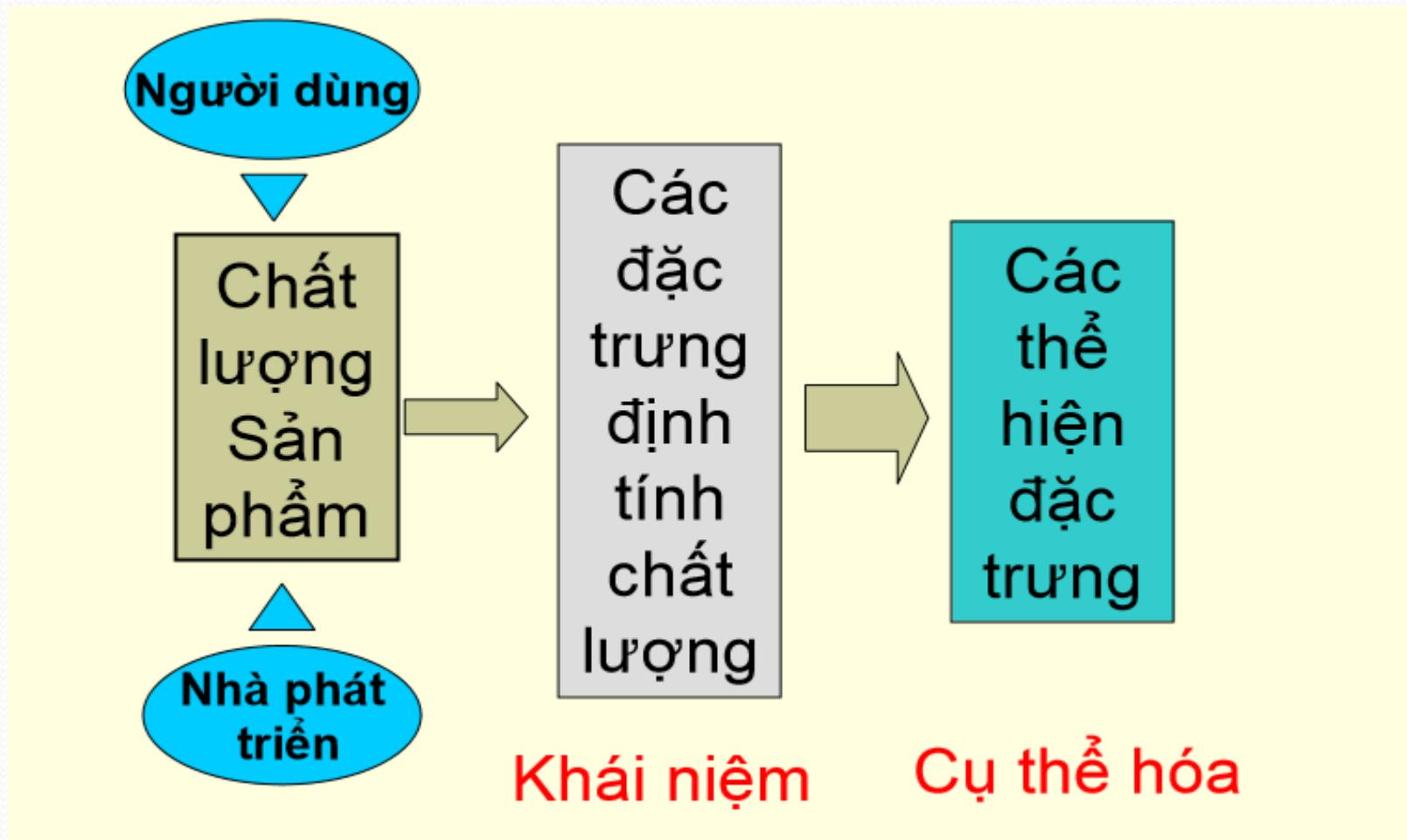
## ■ Lý do

- Có những nhân tố khác ảnh hưởng thực sự lên chất lượng mà không phải là yêu cầu
- Những nhân tố liên quan đến quá trình phát triển: công nghệ, công cụ và quản lý

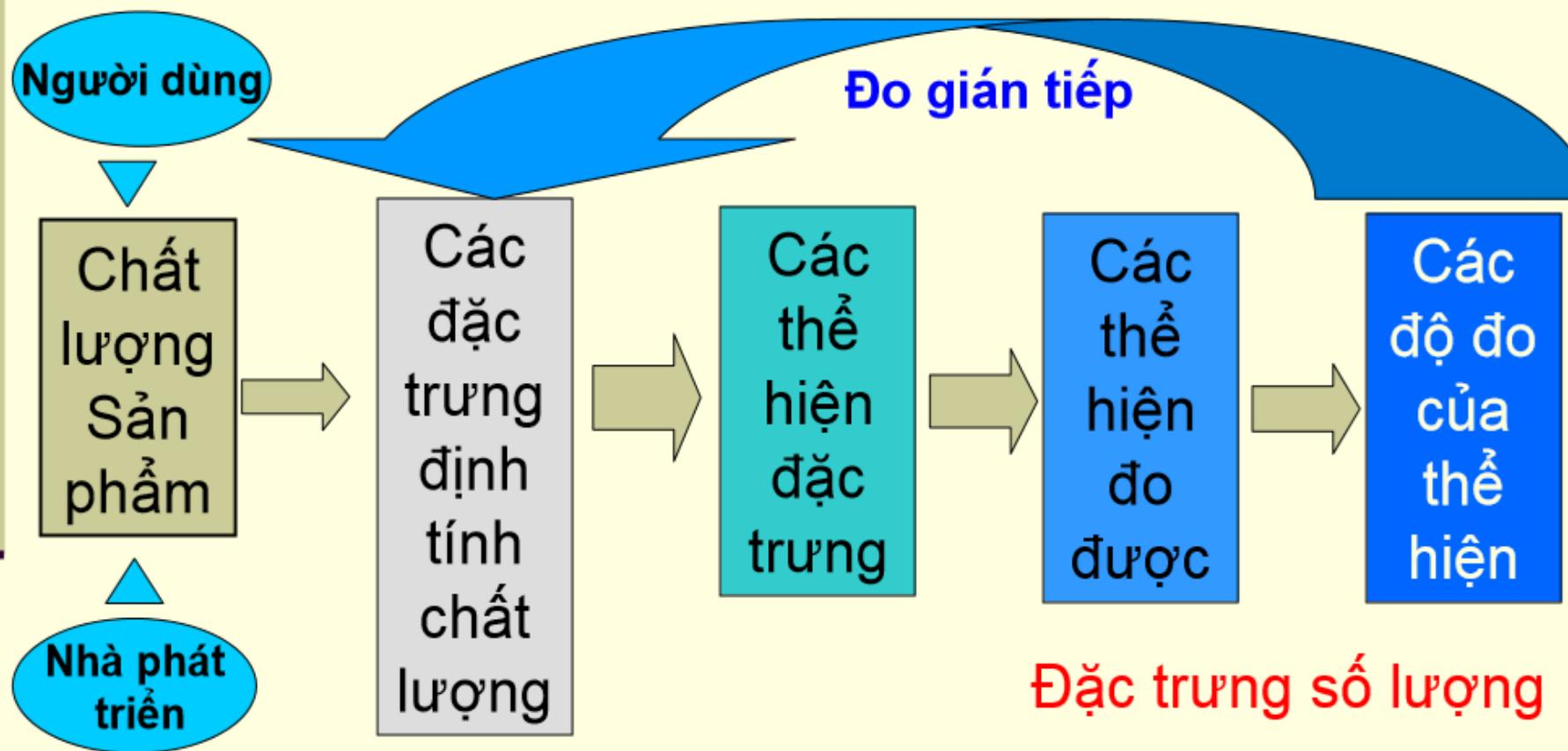
## ■ Một số hoạt động khác

- Sử dụng phương pháp công nghệ, công cụ và tiến trình
- Tuân theo các chuẩn
- Không chế thay đổi (*vốn là bản chất của phần mềm*)
- Tạo, quản lý báo cáo (*trợ giúp các hoạt động – gián tiếp*)

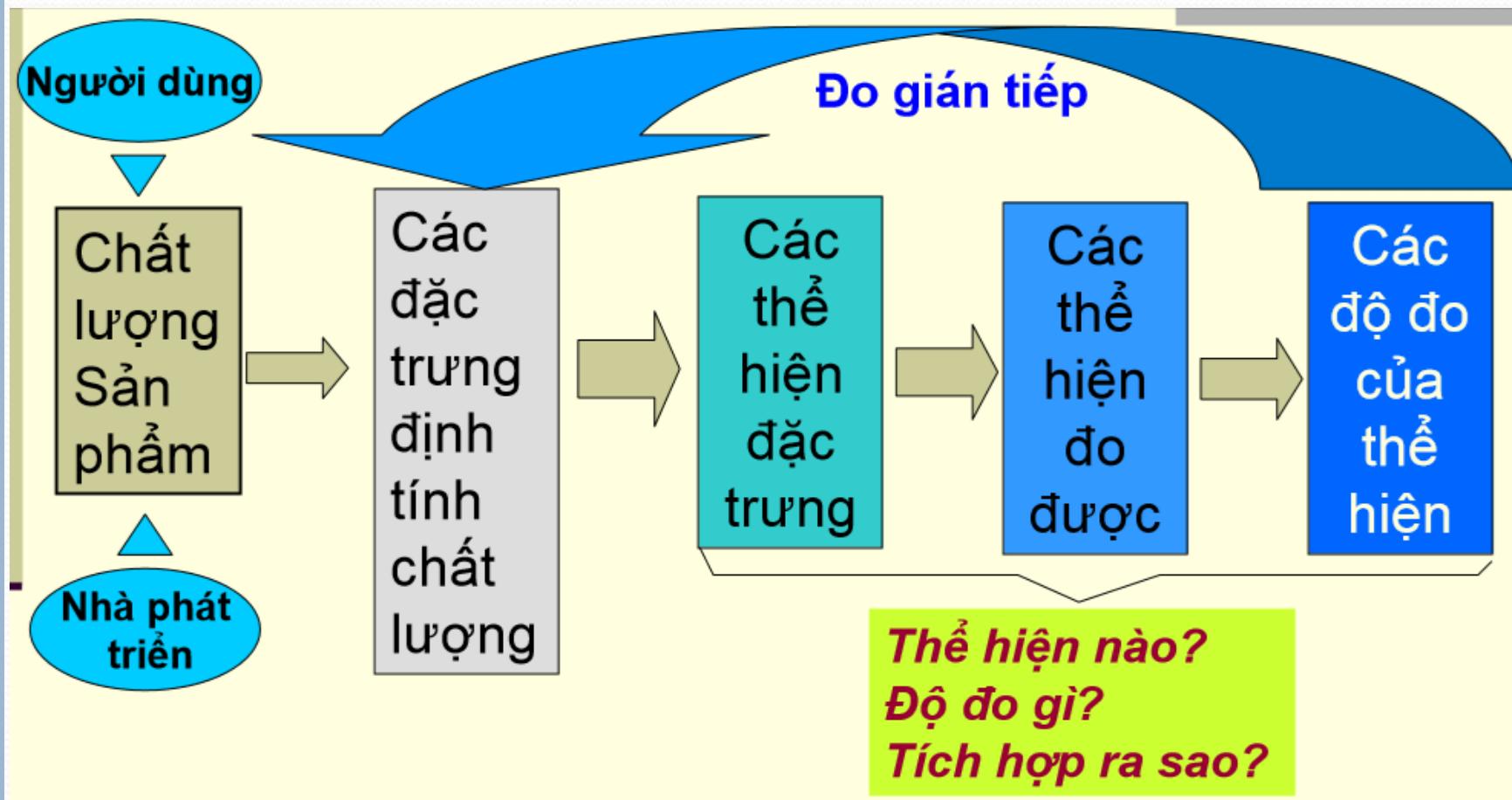
## 2. Đo các đặc trưng chất lượng



# a. Đo chất lượng là gián tiếp



## b. Đo chất lượng qua các thể hiện đo được



## c. Nhân tố ảnh hưởng tới chất lượng

Để tìm ra các thể hiện đo được cần xét:

- Có bao nhiêu loại nhân tố ảnh hưởng?
- Cách thức mà nó ảnh hưởng?

# C1. Loại nhân tố ảnh hưởng

---

- 3 loại (theo McCall ) :

- (1) đặc trưng chức năng
- (2) khả năng đương đầu với những thay đổi,
- (3) khả năng thích nghi với môi trường mới.

## C2. Loại nhân tố và mức ảnh hưởng

Các loại nhân tố và mức độ ảnh hưởng:

- 3 loại (theo McCall ) :
  - (1) đặc trưng chức năng
  - (2) khả năng đương đầu với những thay đổi,
  - (3) khả năng thích nghi với môi trường mới.
- Mức độ ảnh hưởng:
  - **Nhân tố trực tiếp (chức năng)**
  - **Nhân tố gián tiếp (những gì liên quan đến 1 và 2)**

### C3. Các nhân tố ảnh hưởng đến chất lượng

McCall đề xuất 11 nhân tố phân theo loại:

#### ■ Loại (1): Các đặc trưng chức năng

- tính đúng đắn,
- tính tin tưởng được,
- tính hiệu quả,
- tính toàn vẹn,
- tính khả dụng

## C3. Các nhân tố ảnh hưởng đến chất lượng

### ■ Loại (2)- đương đầu với thay đổi

- tính bảo trì được,
- tính mềm dẻo,
- tính thử nghiệm được

## C3. Các nhân tố ảnh hưởng đến chất lượng

### ■ Loại (3)- thích nghi với môi trường mới

- tính mang chuyển được
- tính sử dụng lại được
- tính liên tác được

# d. Các độ đo chất lượng

- Để đo mức độ ảnh hưởng cần có độ đo
- McCall đề xuất **21 độ đo** sau:
  1. Độ kiểm toán được,
  2. Độ chính xác,
  3. Độ tương đồng giao tiếp,
  4. Độ đầy đủ,
  5. Độ phức tạp,
  6. Độ súc tích (conciseness),
  7. Độ nhất quán (consistency),
  8. Độ tương đồng dữ liệu,

# d. Các độ đo chất lượng

---

9. Độ dung thứ lỗi,
10. Độ hiệu quả thực hiện,
11. Độ khuếch trương được,
12. Độ lập phần cứng,
13. Độ trang bị đủ đồ nghề (instrumentation),
14. Độ môđun hoá,
15. Độ dễ thao tác,
16. Độ an ninh,

## d. Các độ đo chất lượng

---

17. Tự tạo tài liệu (self-documentation),
18. Độ đơn giản - dễ hiểu,
19. Độc lập phần mềm,
20. Độ lầm vết được,
21. Khả năng huấn luyện.

# D1. Tính đúng đắn

## 1. Tính đúng đắn:

- làm đúng với cái khách hàng mong muốn
- thoả mãn những điều đã được đặc tả (những yêu cầu của đối tượng khác)

### ■ Các độ đo liên quan:

- Độ đầy đủ,
- Độ nhất quán
- Độ lầm vết được

# D2. Tính tin cậy được

## 2. Tính tin cậy được

Có thể trông đợi vào sự thực hiện các chức năng dự kiến và mức chính xác được đòi hỏi

### ■ Các độ đo liên quan:

- Độ chính xác,
- Độ phức tạp,
- Độ nhất quán,
- Độ dung thứ lỗi,
- Độ môđun hoá,
- Độ đơn giản – dễ hiểu,
- Độ lặp vết được

# D3. Tính hiệu quả

## 3. Tính hiệu quả:

Tổng lượng nguồn lực tính toán và mã yêu cầu để thực hiện các chức năng của chương trình là thích hợp.

### ■ Các độ đo liên quan:

- Độ súc tích,
- Độ hiệu quả thực hiện,
- Độ dễ thao tác,

# D4. Tính toàn vẹn

## 4. Tính toàn vẹn:

Sự không chế được việc truy cập trái phép tới phần mềm và dữ liệu của HT

### ◆ Các độ đo liên quan:

- Độ kiểm toán được,
- Trang bị đồ nghề đủ,
- Độ an ninh,

# D5. Tính khả dụng

---

## 5. Tính khả dụng:

công sức để học hiểu, thao tác dễ, chuẩn bị đầu vào, thể hiện đầu ra của chương trình là chấp nhận được, khả năng nhớ lâu

### ■ Các độ đo liên quan:

- Độ dễ thao tác,
- Khả năng huấn luyện.

# D6. Tính bảo trì được

## 6. Tính bảo trì được:

nỗ lực cần để định vị và xác định được 1 lỗi trong chương trình là chấp nhận được. Dễ thay đổi và mở rộng

### ■ Các độ đo liên quan:

- Độ súc tích,
- Độ nhất quán,
- Trang bị đồ nghề đủ,
- Độ modun hoá,
- Độ tự cấp tài liệu,
- Độ đơn giản
- Độ dễ hiểu,

# D7. Tính mềm dẻo

## 7. Tính mềm dẻo:

Có thể cải biên chương trình và nỗ lực cần để cải biên là chấp nhận được.

### ■ Các độ đo liên quan:

- Độ phức tạp,
- Độ súc tích
- Độ nhất quán,
- Khuếch trương được,
- Độ khái quát,
- Độ modun hoá,
- Tự cấp tài liệu,
- Độ đơn giản - dễ hiểu,

# D8. Tính kiểm thử được

## 8. Tính kiểm thử được:

Công sức cần để kiểm thử chương trình và bảo đảm rằng nó thực hiện đúng chức năng dự định là chấp nhận được

### ■ Các độ đo liên quan:

- Độ kiểm toán được,
- Độ phức tạp,
- Trang bị đồ nghề đủ,
- Độ môđun hoá,
- Tự cấp tài liệu,
- Độ đơn giản - dễ hiểu,

# D9.Tính mang chuyển được

## 9. Tính mang chuyển được:

Công sức đòi hỏi để chuyển nó từ một môi trường (*phần cứng, phần mềm*) này sang một môi trường khác là chấp nhận được

### ■ Các độ đo liên quan:

- Độ khái quát,
- Độ độc lập phần cứng,
- Độ modun hoá,
- Tự cấp tài liệu,
- Độc lập hệ thống phần mềm,

# D10. Tính sử dụng lại được

## 10. Tính sử dụng lại được:

Khả năng hệ thống hoặc một phần của nó có thể  
được dùng lại trong một ứng dụng khác

### ■ Các độ đo liên quan:

- Độ khái quát,
- Độc lập phần cứng,
- Độ môđun hoá,
- Tự tạo tài liệu,
- Độc lập hệ thống phần mềm,

# D11. Tính liên tác được

## 11. Tính liên tác được:

Công sức đòi hỏi để ghép nối hệ thống phần mềm với một hệ thống khác là chấp nhận được

### ■ Các độ đo liên quan:

- Độ tương đồng giao tiếp,
- Độ tương đồng dữ liệu,
- Độ khái quát,
- Độ môđun hoá.

### 3. Tiến hóa của hoạt động SQA

- Khi phần mềm trở thành sản phẩm **có nhu cầu và đòi hỏi** đảm bảo chất lượng:
  - Từ khách hàng (**nhu cầu**)
  - Từ nhà sản xuất (**đòi hỏi**): đảm bảo tính đồng đều của sản phẩm, cải thiện chất lượng thường xuyên
- **Từ thực tiễn:** Kinh nghiệm cho phép hoạt động đảm bảo chất lượng phần mềm ngày càng được hoàn thiện. **Hiểu về vai trò** của nó và **tăng thêm các hoạt động** đảm bảo chất lượng

# A. Sự phát triển của SQA

- Bảo đảm chất lượng là một hoạt động cốt yếu của mọi doanh nghiệp nào làm ra sản phẩm cho người khác dùng
- Đảm bảo chất lượng phần mềm diễn ra song song với bảo đảm chất lượng trong chế tạo phần cứng.
- Các chuẩn bảo đảm chất lượng phần mềm là cơ sở đo chất lượng (Chúng đầu tiên được đưa ra trong quân sự vào những năm 70 và nhanh chóng mở rộng trong thương mại)

## b. Vai trò và trách nhiệm trong SQA

■ Những người có trách nhiệm bảo đảm chất lượng phần mềm (trong tổ chức):

- Kỹ sư phần mềm,
- Nhà quản lý dự án,
- Khách hàng,
- Người bán hàng
- Thành viên trong nhóm SQA.

## b. Vai trò và trách nhiệm trong SQA

- Nhóm SQA phải đóng vai trò như đại diện của khách hàng - để xem xét chất lượng phần mềm với quan điểm của khách hàng:
  - Có đáp ứng được các nhân tố chất lượng không?
  - Có tuân theo các chuẩn dự định trước không?
  - Các **thủ tục, phương pháp, kỹ thuật** có thực sự đóng vai trò của chúng trong hoạt động SQA?

# c. Các hoạt động SQA

## ■ Có 7 hoạt động chính:

- (1) Áp dụng công nghệ, kỹ nghệ hiệu quả (*phương pháp, công cụ và tiến trình*)
- (2) Tiến hành rà soát kỹ thuật chính thức
- (3) Thực hiện kiểm thử nhiều tầng
- (4) Tuân theo các chuẩn phát triển
- (5) Kiểm soát tài liệu FM và thay đổi của chúng
- (6) Thực hiện đo lường
- (7) Báo cáo và quản lý các báo cáo.

# C1. Vai trò của các hoạt động trong SQA

- Tập hợp các phương pháp và công cụ giúp để:
  - giúp phân tích có được đặc tả tốt,
  - giúp thiết kế có được thiết kế chất lượng cao.
  - Trợ giúp kiểm thử hiệu quả
- **Rà soát kỹ thuật chính thức** là hoạt động trung tâm.  
(tác dụng không kém gì thử nghiệm để việc phát hiện khiếm khuyết).
- **Kiểm thử phần mềm** (chỉ kiểm thử không thể tìm ra được hầu hết các sai)

# C1. Vai trò của các hoạt động trong SQA

- Áp dụng các chuẩn và các thủ tục chính thức là một nhu cầu và điều kiện cho SQA. Tuy nhiên, còn tuỳ thuộc mỗi công ty. Có 2 loại chuẩn và thủ tục:
  - do khách hàng, hay chính quyền quy định.
  - tự công ty đặt ra.
- Đánh giá sự phù hợp với các chuẩn là một phần của việc rà soát chính thức.
- Khi cần phải Xác minh sự phù hợp; nhóm SQA có thể tiến hành kiểm toán (audit) riêng.

# C1. Vai trò của các hoạt động trong SQA

## ■ Không chế đổi thay đóng góp trực tiếp vào chất lượng phần mềm nhờ:

- chính thức hóa các yêu cầu đổi thay,
- đánh giá bản chất của sự đổi thay,
- không chế các ảnh hưởng của sự đổi thay.

Lý do:

- ❖ Đe doạ chủ yếu của chất lượng đến từ **sự thay đổi**. Thay đổi là bản chất của phần mềm
- ❖ Thay đổi tạo ra **tiềm năng sinh ra sai** và tạo ra **hiệu ứng phụ** lan truyền.

## ■ Không chế thay đổi áp dụng trong suốt quá trình phát triển và bảo trì

# C1. Vai trò của các hoạt động trong SQA

- Lập và lưu giữ báo cáo về SQA:
  - phô biến các thông tin SQA (*người cần có thể biết*).
  - cung cấp các thủ tục để thu thập thông tin
- Đối tượng báo cáo là kết quả các hoạt động SQA:
  - rà soát,
  - kiểm toán,
  - khống chế đổi thay,
  - Kiểm thử,
  - và các hoạt động SQA khác
- Người phát triển sử dụng quy tắc “**cần-thì-biết**” –  
**trong suốt quá trình dự án**

## C2. Mục tiêu của hoạt động SQA

- Một mục tiêu quan trọng của SQA:
  - Theo dõi chất lượng phần mềm
  - Đánh giá ảnh hưởng của thay đổi về phương pháp luận và thủ tục lên chất lượng phần mềm.
    - Muốn vậy phải thu thập các độ đo phần mềm.
- Các độ đo phần mềm hướng đến các mặt:
  - quản lý (thủ tục)
  - kỹ thuật (phương pháp)
  - Yêu cầu người dùng

### 3. Rà soát phần mềm

- **Rà soát** là việc xem xét, đánh giá sản phẩm được tiến hành **mỗi giai đoạn** để phát hiện ra những khiếm khuyết cần sửa chữa trước khi sang giai đoạn sau
- **Mục tiêu:**
  - Chỉ ra các khiếm khuyết cần phải cải thiện.
  - Khẳng định những phần sản phẩm đạt yêu cầu.
  - Kiểm soát việc đạt chất lượng kỹ thuật tối thiểu của sản phẩm (có diện mạo không đổi, ổn định)
- **Áp dụng** tại các thời điểm khác nhau trong quá trình phát triển phần mềm.

# a. Các hình thức rà soát

---

## ■ Các kiểu rà soát:

- Họp xét duyệt không chính thức,
- Họp chính thức trước với các thành viên: khách hàng, nhà quản lý, nhân viên kỹ thuật. (tập trung vào các rà soát kỹ thuật chính thức - *formal technical review - FTR*)
- FTR chủ yếu do các kỹ sư phần mềm thực hiện (**là một phương tiện hiệu quả để cải thiện chất lượng**)

## b. Lợi ích của việc rà soát

- Sớm phát hiện các “khiếm khuyết” của phần mềm để có thể chỉnh sửa.
- Các nghiên cứu của công nghiệp phần mềm (TRW, Nippon Electric, Mitre Corp., ...) đã chỉ ra: **các hoạt động thiết kế tạo ra đến 50% - 60% tổng số** các khiếm khuyết tạo ra trong phát triển mềm
- Chi phí chỉnh sửa một khiếm khuyết tăng lên nhanh chóng sau mỗi giai đoạn. trong **thiết kế** **tốn phí 1.0**, **trước kiểm thử: 6.5**, **trong kiểm thử: 15** và **sau phân phối** **sẽ từ 60**. đến **100**.

---

# **Thẩm định và Xác minh phần mềm : Verification and Validation**

---

# Khái niệm V&V

---

- Verification – Xác minh:
- Validation – Thẩm định:

# Khái niệm V&V (giải thích)

- **Thẩm định phần mềm:** Là xem phần mềm cho kết quả đúng hay không và có thỏa mãn yêu cầu của người sử dụng hay không.
- **Xác minh phần mềm:** Là xem sản phẩm có đúng là sản phẩm được yêu cầu không và chương trình có đúng với đặc tả không.
- Thẩm định và xác minh phần mềm là 2 quá trình liên tục, xuyên suốt từ lúc phân tích các yêu cầu của khách hàng cho đến khi giao sản phẩm, với mục đích:
  - Xem hệ thống có đáp ứng yêu cầu của khách hàng không, phát hiện lỗi của phần mềm.

# Mục đích của V&V

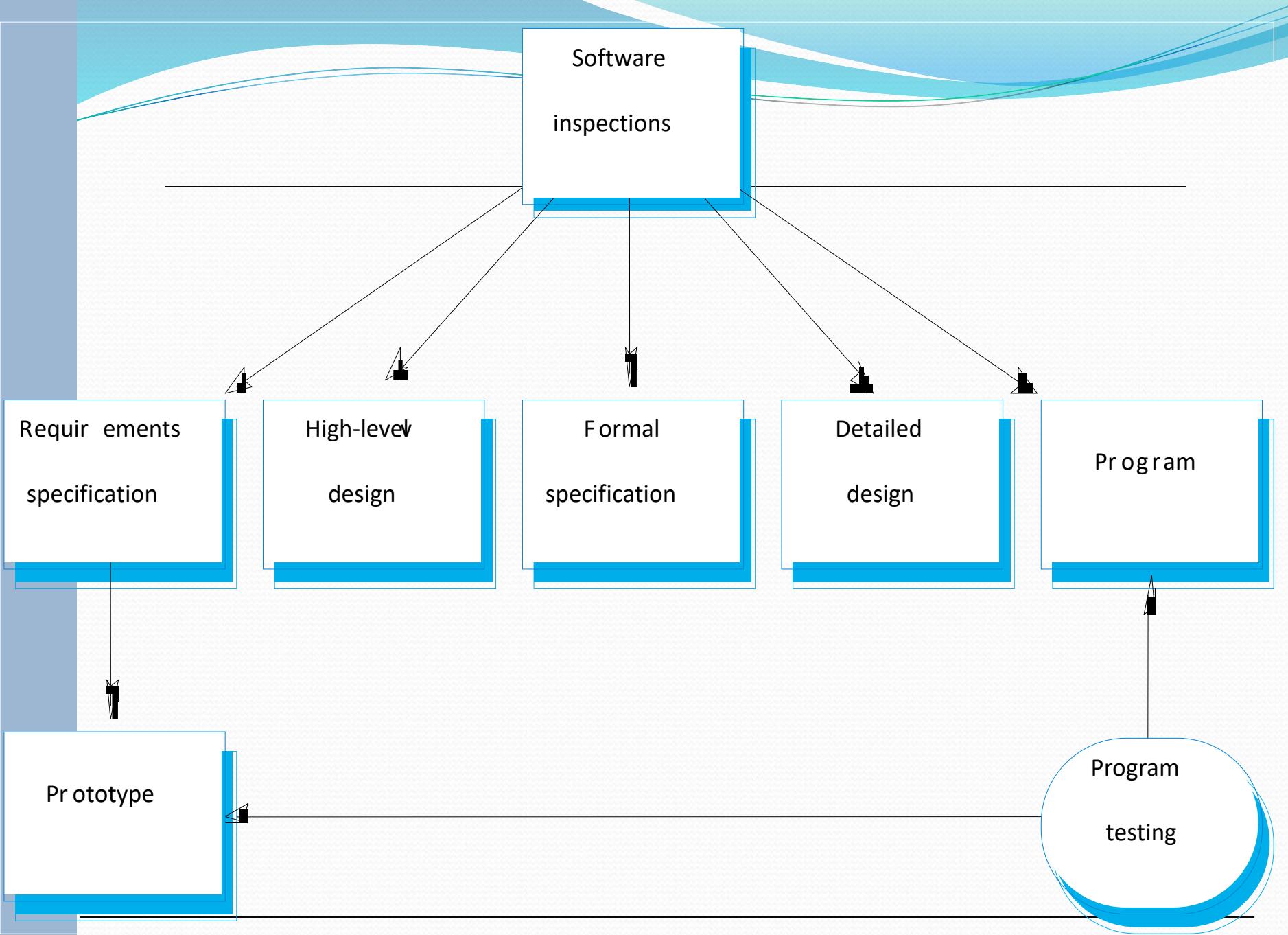
- Tạo sự tự tin về phần mềm sẽ đạt được mục tiêu đề ra.
- Điều này không có nghĩa là sẽ tạo ra phần mềm không có lỗi chút nào.
- Kiểu sử dụng phần mềm sẽ quyết định mức độ tự tin cần thiết:

# Cách thức tiến hành

- Để thẩm định và xác minh phần mềm người ta phải thử nghiệm (kiểm thử) hay thanh tra.
- Hai cách thức này thường có liên hệ với nhau

# Xác minh tĩnh và động

- Thanh tra phần mềm. Liên quan đến việc phân tích hệ thống trong trạng thái tĩnh (không chạy) để phát hiện các vấn đề (Xác minh tĩnh)
  - Có thể sử dụng các công cụ phân tích tài liệu và phân tích mã nguồn để hỗ trợ
- Kiểm thử phần mềm. Liên quan đến việc cho chạy và quan sát hành vi của phần mềm (Xác minh động).
  - Hệ thống được cho chạy cùng với dữ liệu kiểm thử và hành vi của nó sẽ được quan sát



# Các loại thử nghiệm

- **Các loại thử nghiệm:**

- 1. *Thử thống kê*: cho nhiều bộ dữ liệu khác nhau để chạy thử và tính tần suất xuất hiện thất bại -> kiểm tra tính đúng đắn (validation- thẩm định)
- 2. *Thử khuyết tật*: Cho những bộ dữ liệu thật đặc biệt để chạy thử => phải lựa chọn được những bộ dữ liệu thật đặc biệt. Phép thử được coi là thành công nhất nếu phơi được nhiều khuyết tật nhất.
- 3. *Thử giới hạn tải(áp lực)*: Nếu phần mềm có giới hạn tải, ta thử bằng cách tăng dần tải cho đến khi không chịu được.  
= Kiểm tra độ tin cậy

# V&V và Debug

- Kiểm thử khuyết tật và gỡ rối là những tiến trình riêng biệt.
- Thẩm định và xác minh liên quan đến việc xem xét sự tồn tại của các khuyết tật trong chương trình.
- Gỡ rối liên quan đến việc xác định vị trí và sửa chữa những lỗi đã tìm thấy.
- Debugging liên quan đến việc xây dựng một giả thuyết về hành vi của chương trình và kiểm tra giả thuyết đó để tìm ra lỗi.

# Lập kế hoạch cho V&V

- Một kế hoạch cẩn thận là cần thiết để nhận được hiệu quả cao nhất trong kiểm thử và thanh tra
- Việc lập kế hoạch cần được tiến hành sớm trong tiến trình phát triển phần mềm
- Kế hoạch nên xác định rõ sự cân bằng giữa xác minh tĩnh và động
- Kế hoạch kiểm thử nên chỉ ra các chuẩn cần sử dụng cho tiến trình kiểm thử thay vì mô tả các dữ liệu test

# Thanh tra phần mềm

- Liên quan đến việc kiểm tra mã nguồn để tìm ra các vấn đề bất thường và khuyết tật
- Không yêu cầu chạy phần mềm trước và khi thanh tra
- Có thể tiến hành thanh tra mọi đối tượng cấu hình của phần mềm (các bản đặc tả yêu cầu, thiết kế, dữ liệu test,...)
- Là một kỹ thuật hiệu quả để phát hiện ra lỗi
- Nhiều khuyết tật khác nhau có thể được phát hiện chỉ bởi một lần thanh tra.
- Trong một lần kiểm thử, một khuyết tật có thể chưa được phát hiện, vì vậy cần phải tiến hành nhiều lần
- Các lĩnh vực tái sử dụng và tri thức lập trình cho phép phát hiện các loại lỗi thường hay xảy ra

# Thanh tra và kiểm thử phần mềm

- Thanh tra và kiểm thử bổ sung cho nhau, không phải là những kỹ thuật xác minh đối lập nhau
- Cả hai nên được sử dụng trong tiến trình V&V
- Thanh tra có thể kiểm tra được sự phù hợp của phần mềm với đặc tả nhưng không kiểm tra được sự phù hợp của phần mềm với yêu cầu thực tế của khách hàng
- Thanh tra không thể đánh giá được những đặc trưng phi chức năng như hiệu suất, tính khả dụng,...

# Thanh tra chương trình

- Là cách tiếp cận hình thức hóa để rà soát tài liệu
- Có mục đích rõ ràng là phát hiện khuyết tật (nhưng không chỉnh sửa)
- Khuyết tật có thể là những lỗi logic, những dị thường trong mã nguồn như những tình trạng sai sót (ví dụ biến không được khởi tạo) hoặc sự không phù hợp với các chuẩn mã nguồn.

# Điều kiện tiền thanh tra

- Các bản đặc tả đã phải có và sẵn sàng
- Các thành viên của đội thanh tra phải nắm chắc các tiêu chuẩn trong công ty, tổ chức
- Đã có mã nguồn được viết không có lỗi cú pháp
- Danh sách các lỗi cần thanh tra đã được chuẩn bị
- Bộ phận quản lý đã đồng ý với những chi phí phát sinh do thanh tra
- Bộ phận quản lý không được sử dụng kết quả thanh tra để đánh giá nhân viên

# Thủ tục thanh tra

- Đội thanh tra được giới thiệu tổng quan về hệ thống
- Mã và các tài liệu kèm theo được đưa trước cho từng thành viên của đội thanh tra
- Thanh tra ghi chép lại những lỗi đã được phát hiện và vị trí của chúng
- Các sự sửa đổi được thực hiện để sửa chữa những lỗi đã được phát hiện
- Việc thanh tra lại có thể cần hoặc không cần

# checklist

- Danh sách các lỗi chung nên được sử dụng để điều khiển việc thanh tra
- Danh sách lỗi phụ thuộc vào ngôn ngữ lập trình
- Đơn giản hơn là kiểm tra kiểu trong ngôn ngữ lập trình, phức tạp hơn là kiểm tra theo danh sách lỗi. Ví dụ: Khởi tạo, đặt tên hằng, thoát khỏi vòng lặp, giới hạn mảng,...

# Phân tích tĩnh tự động

- Việc phân tích tĩnh được tiến hành bằng các công cụ xử lý mã nguồn phần mềm
- Các công cụ phân tích văn bản chương trình và cố gắng phát hiện các chỗ có khả năng sai lầm và đưa ra cảnh báo cho đội V&V
- Là một sự trợ giúp rất hiệu quả khi thanh tra
  - Chỉ là một sự bổ sung thêm chứ không thay thế thanh tra

# Các giai đoạn phân tích tĩnh

- *Control flow analysis.* Kiểm tra các luồng điều khiển có nhiều lối thoát hoặc nhiều điểm bắt đầu để tìm kiếm những đoạn code không thể truy cập được,...
- *Data use analysis.* Phát hiện các biến không được khởi tạo, các biến được viết hai lần mà không có chỗ gán giá trị xen giữa, các biến được khai báo nhưng không được sử dụng,...
- *Interface analysis.* Kiểm tra tính nhất quán của đoạn chương trình, các khai báo thủ tục và việc sử dụng chúng
- *Information flow analysis.* Xác định sự phụ thuộc của các biến đầu ra. Không cần phát hiện những điều bất thường của chúng nhưng cần làm nổi bật thông tin về sự phụ thuộc đó cho thanh tra rà soát mã nguồn
- *Path analysis.* Xác định các lộ trình của chương trình và lập danh sách các statements trong từng lộ trình. Những danh sách này có thể sẽ cần khi rà soát
- Cả hai giai đoạn trên đều tạo ra một lượng lớn thông tin. Cần phải cẩn thận khi sử dụng.

# Ứng dụng của phân tích tĩnh

- Có giá trị đặc biệt đối với những ngôn ngữ định kiểu yếu như C, những ngôn ngữ này hay đem lại những lỗi không được phát hiện bởi trình biên dịch
- Ít hiệu quả cho những ngôn ngữ định kiểu mạnh như Java do trình biên dịch đã phát hiện được rất nhiều lỗi mã nguồn.