



Phân tích và đặc tả yêu cầu

Lê Thị Mỹ Hạnh
Khoa Công nghệ Thông tin
Trường Đại học Bách khoa, Đà Nẵng



NỘI DUNG

- Đại cương về phân tích và đặc tả
- Nền tảng của phân tích
- Phân tích và nắm bắt yêu cầu
- Đặc tả yêu cầu
- Thẩm định yêu cầu
- Một số phương pháp và công cụ hỗ trợ



Phân tích và đặc tả

- Xác định và phân tích yêu cầu là khâu kỹ thuật đầu tiên của quá trình phát triển phần mềm.
 - Xác định các chức năng/dịch vụ mà khách hàng yêu cầu từ hệ thống
 - Các ràng buộc mà hệ thống được phát triển và vận hành.
- Là sự phối hợp của nhà phát triển và khách hàng.
- Quyết định chất lượng phần mềm đạt được và chi phí bỏ ra.

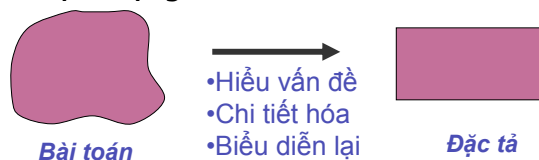


Phân tích và đặc tả

- Mục đích: Đặc tả yêu cầu phần mềm
 - Cơ sở cho việc mời thầu (cần có giải thích)
 - cơ sở để ký kết hợp đồng (cần đầy đủ chi tiết)
 - Làm tư liệu phục vụ thiết kế và triển khai (cần đầy đủ, chính xác, không mâu thuẫn)

Yêu cầu là gì?

- Các yêu cầu là các **mô tả từ mức trừu tượng rất cao đến một đặc tả rất chi tiết** về dịch vụ mà hệ thống cần cung cấp cũng như các ràng buộc lên sự phát triển và hoạt động của nó.

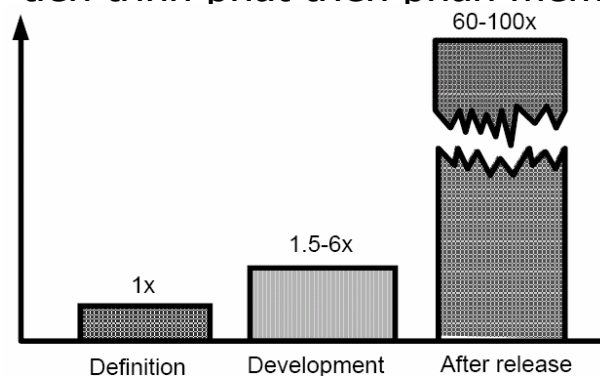


□ Yêu cầu là:

- 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 của phần mềm cần có nhằm thỏa mãn một hợp đồng, một chuẩn, một đặc tả.

Vai trò

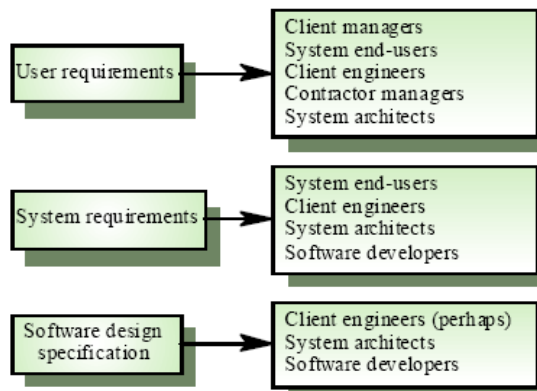
- Có vai trò đặc biệt quan trọng trong tiến trình phát triển phần mềm.



Phân loại yêu cầu

- Yêu cầu người dùng
 - Diễn đạt bằng ngôn ngữ tự nhiên và sơ đồ
 - Nêu rõ dịch vụ hệ thống cung cấp và các ràng buộc trong hoạt động của nó.
- Yêu cầu hệ thống
 - Mô tả đủ chi tiết về các dịch vụ hệ thống
 - Như một hợp đồng giữa khách hàng và nhà thầu
- Đặc tả phần mềm
 - Mô tả chi tiết về phần mềm làm cơ sở cho thiết kế và cài đặt
 - Dành cho người phát triển

Đối tượng đọc yêu cầu



⇒ *Yêu cầu viết ra cần đáp ứng được tất cả các đối tượng này.*



Các loại yêu cầu

- Các yêu cầu chức năng
 - Mô tả các chức năng hay các dịch vụ mà hệ thống phần mềm cần cung cấp.
- Các yêu cầu phi chức năng
 - Mô tả các ràng buộc đặt lên dịch vụ và quá trình phát triển hệ thống (về chất lượng, về môi trường, chuẩn sử dụng, qui trình phát triển..)
- Các yêu cầu miền/lĩnh vực
 - Những yêu cầu đặt ra từ miền ứng dụng, phản ánh đặc trưng của miền đó.



Yêu cầu chức năng

- Mô tả chức năng hay các dịch vụ của hệ thống
- Chúng phụ thuộc vào:
 - Loại phần mềm sẽ được xây dựng
 - Sự mong muốn của khách hàng
 - Loại hệ thống mà phần mềm trợ giúp
- Mức độ các yêu cầu
 - Trừu tượng: hệ thống làm gì
 - Chi tiết: dịch vụ cụ thể hệ thống thực hiện



Yêu cầu chức năng

■ Ví dụ

- Người sử dụng có thể tìm kiếm tài liệu dựa trên từ khóa có trong tài liệu hoặc tên tài liệu
- Hệ thống cần cung cấp cho người sử dụng phương tiện hiển thị để dàng các tài liệu từ CSDL
- Hệ thống phải đọc được các định dạng khác nhau của tài liệu: text, doc, pdf, ppt, bảng tính excel,..



Yêu cầu chức năng

■ Sự không chính xác của yêu cầu

- Yêu cầu không được phát biểu chính xác
- Yêu cầu nhập nhằng có thể được hiểu khác nhau giữa người sử dụng và người phát triển
- Ví dụ: yêu cầu "hiển thị dễ dàng"
 - Người sử dụng: có thể hiển thị các loại tài liệu khác nhau
 - Người phát triển: cung cấp giao diện hiển thị tài liệu ở chế độ văn bản

■ Trên nguyên tắc yêu cầu phải thỏa mãn

- Đầy đủ: yêu cầu phải mô tả đầy đủ các chức năng cần thiết
- Gắn bó: các yêu cầu không mâu thuẫn nhau

■ Thực tế không đơn giản để có các yêu cầu đầy đủ và gắn bó



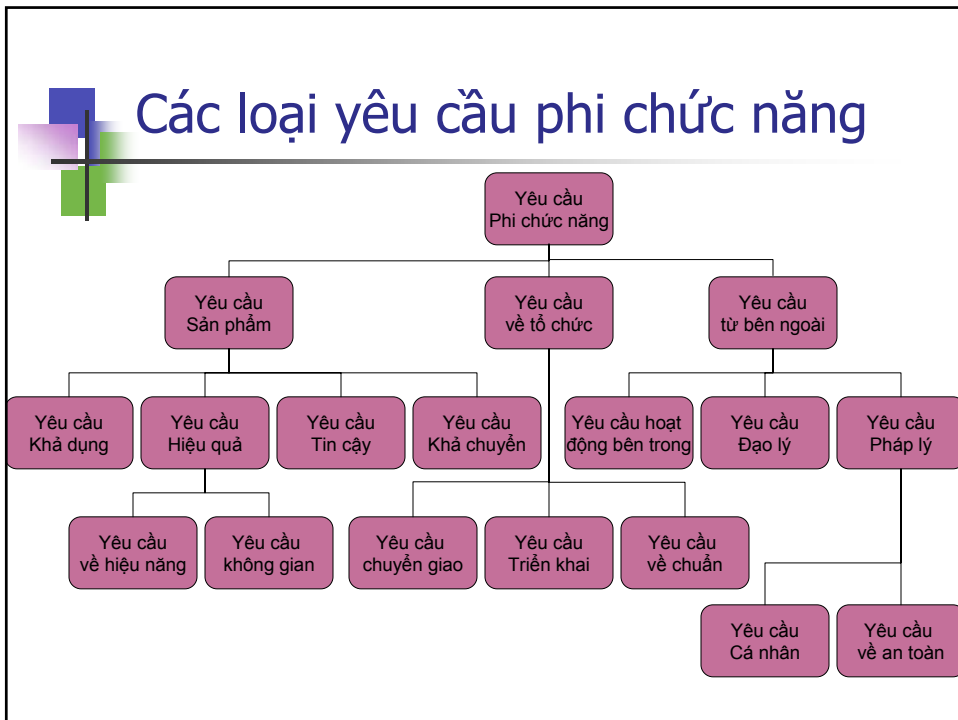
Yêu cầu phi chức năng

- Định nghĩa các tính chất và ràng buộc của hệ thống
 - Yêu cầu tiến trình
 - Phương pháp thiết kế
 - Ngôn ngữ lập trình
 - Công cụ sử dụng
 - Thời gian trả lời
 - Độ tin cậy
 - Yêu cầu về lưu trữ dữ liệu
- Yêu cầu phi chức năng có thể quan trọng hơn yêu cầu phi chức năng
 - Nếu yêu cầu phi chức năng không được đáp ứng hệ thống trở nên vô dụng



Yêu cầu phi chức năng

- Yêu cầu về sản phẩm
 - Tốc độ, độ tin cậy, bộ nhớ cần, giao diện...
- Yêu cầu về tổ chức, tiến trình phát triển
 - Các chuẩn áp dụng, phương pháp thiết kế, ngôn ngữ lập trình, mô hình tiến trình,..
- Yêu cầu từ bên ngoài
 - Về chi phí, về bản quyền, liên kết,..



Yêu cầu phi chức năng

Ví dụ

- **Yêu cầu về sản phẩm**
 - Phần mềm chỉ nên yêu cầu tối đa 256 MB bộ nhớ
 - Yêu cầu về tổ chức
 - Yêu cầu phải đáp ứng chuẩn DO178
 - Yêu cầu bên ngoài
 - Hệ thống không được để lộ thông tin cá nhân của khách hàng ra bên ngoài



Yêu cầu người sử dụng (user requirements)

- Nên mô tả
 - Yêu cầu chức năng
 - Yêu cầu phi chức năng
- Dễ hiểu đối với người sử dụng
 - Không có kiến thức chi tiết về kỹ thuật/tin học
- Nên mô tả
 - Bằng ngôn ngữ tự nhiên
 - Biểu đồ, bảng biểu



Ngôn ngữ tự nhiên (NNTN)

- Ưu điểm
 - Dễ hiểu
 - Dễ sử dụng
- Hạn chế
 - Không rõ ràng, thiếu chính xác
 - Nhập nhãng
 - Lẫn lộn giữa yêu cầu chức năng và yêu cầu phi chức năng
 - Quá mềm dẻo
 - Có thể trình bày nhiều cách



Giải pháp thay thế cho NNTN

- Ngôn ngữ có cấu trúc
 - Sử dụng ngôn ngữ gần gũi với ngôn ngữ lập trình
- Các mô hình
 - Các ký hiệu đồ họa
- Ký hiệu toán học
 - Ngôn ngữ hình thức



Yêu cầu hệ thống (System Requirements)

- Là đặc tả chi tiết hơn yêu cầu người sử dụng
- Phục vụ cơ bản cho bước thiết kế
- Có thể sử dụng làm một phần của hợp đồng
- Có thể sử dụng các mô hình để mô tả



Tài liệu yêu cầu

- Chỉ mô tả về **chức năng** (các hành vi bên ngoài) của hệ thống và các **ràng buộc (WHAT)**
- Không mô tả về phương thức cài đặt (**HOW**)
 - không phải là tài liệu thiết kế
- Phải dễ thay đổi
 - Khó xác định được đầy đủ, chính xác ngay
 - Phải qua nhiều bước xét duyệt lại
- Sử dụng như là tài liệu tham khảo khi bảo trì
- Đặc tả trả lời các sự kiện không mong đợi



Tài liệu yêu cầu

■ Khó khăn

- Khách hàng chỉ có khái niệm mơ hồ
 - người phát triển phải chi tiết hóa thành các yêu cầu phần mềm có thể xây dựng được
- Khách hàng rất hay thay đổi yêu cầu
 - người phát triển cần tỉnh táo để xác định đúng các yêu cầu
- Các yêu cầu thường mang tính đặc thù
 - Khó hiểu, khó định nghĩa
 - Không có chuẩn biểu diễn



Tài liệu yêu cầu

■ Khó khăn

- Hệ thống đa người sử dụng
 - Các yêu cầu đa dạng, mức ưu tiên khác nhau
 - Yêu cầu mâu thuẫn nhau
- Người đặt hành khác người sử dụng thật sự
 - Không nắm vững yêu cầu



Cấu trúc của tài liệu yêu cầu

- Giới thiệu
- Thuật ngữ
- Định nghĩa yêu cầu người sử dụng
- Kiến trúc hệ thống
- Đặc tả yêu cầu hệ thống
- Mô hình hệ thống
- Phát triển/thay đổi của hệ thống
- Phụ lục
- Chỉ mục



Định dạng của tài liệu yêu cầu

■ Theo chuẩn IEEE 830-1984

1. Giới thiệu
2. Mô tả chung
3. Yêu cầu chi tiết



Định dạng của tài liệu yêu cầu

■ 1. Giới thiệu

- 1.1. Mục đích
- 1.2. Phạm vi
- 1.3. Định nghĩa (định nghĩa, từ viết tắt)
- 1.4. Tài liệu tham khảo
- 1.5. Mô tả cấu trúc tài liệu



Định dạng của tài liệu yêu cầu

2. Mô tả chung

- 2.1. Tổng quan về sản phẩm
- 2.2. Chức năng sản phẩm
- 2.3. Đối tượng người dùng
- 2.4. Ràng buộc tổng thể
- 2.5. Giả thiết và sự lệ thuộc



Định dạng của tài liệu yêu cầu

3. Yêu cầu chi tiết

3.1. Yêu cầu về chức năng

3.1.1. Chức năng 1

3.1.1.1. Giới thiệu

3.1.1.2. Dữ liệu vào

3.1.1.3. Xử lý

3.1.1.4. kết quả

3.1.2. Chức năng 2

...

3.1.n. Chức năng n

3.2. Yêu cầu về giao diện ngoài

3.2.1. Giao diện người dùng

3.2.2. Giao diện phân cứng

3.2.3. Giao diện phần mềm

3.2.4. Giao diện truyền thông

3.3. Yêu cầu hiệu suất

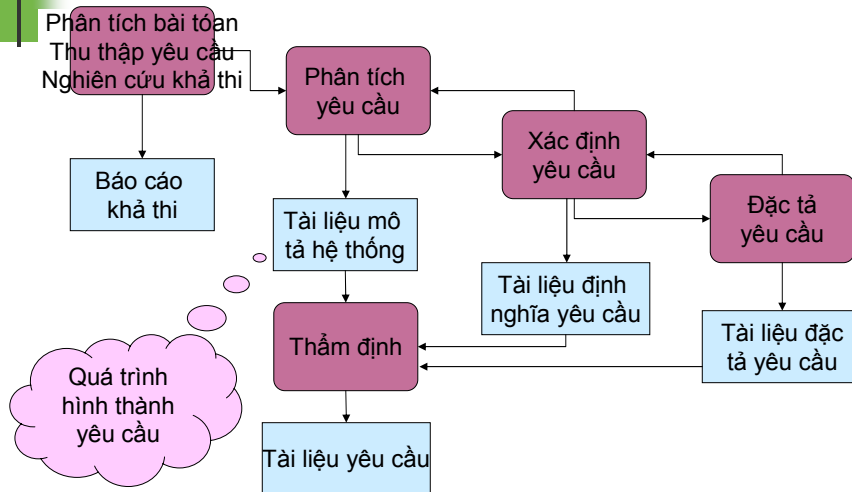
3.4 Ràng buộc thiết kế

3.5. Thuộc tính: - bảo mật - bảo trì

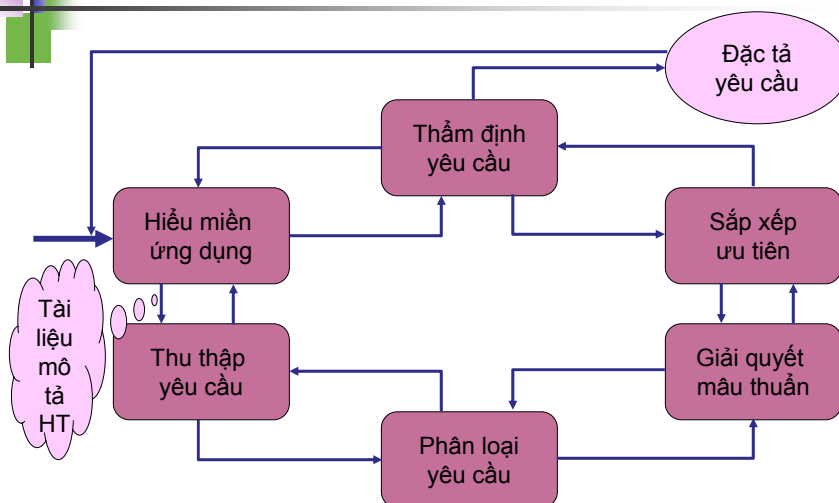
3.6. Yêu cầu khác

Phụ lục

Các công đoạn phân tích và đặc tả



Tiến trình phân tích yêu cầu





Phân tích bài toán

- Mô tả nghiệp vụ
 - Mô tả các luồng nghiệp vụ, các xử lý và vai trò của con người trong hệ thống hiện tại
 - Hiểu được nghiệp vụ
 - Chủ yếu tập trung vào các miền cần tự động hoá
 - Hỗ trợ cho việc xác định các thay đổi và cải tiến yêu cầu trong hệ thống mới



Phân tích bài toán

- Mô tả hệ thống
 - Mô tả hệ thống đề xuất
 - Mô tả luồng thông tin giữa hệ thống đề xuất và môi trường của nó
 - Đáp ứng được các mô tả nghiệp vụ
 - Cải tiến nghiệp vụ hiện tại
 - Dựa trên mô tả nghiệp vụ hiện tại



Thu thập yêu cầu

- Xác định tính khả thi của hệ thống
 - Kinh tế
 - Kỹ thuật
 - Vận hành
- Xác định những người liên quan đến hệ thống và những người sử dụng cuối
- Xác định các ràng buộc khi sử dụng hệ thống đề xuất



Thu thập yêu cầu

- Xác định phương pháp thu thập yêu cầu
 - Phỏng vấn
- Xác định các yêu cầu nhập nhằng
 - Có thể sử dụng kỹ thuật nguyên mẫu
- Xác định các yêu cầu khác mà khách hàng không yêu cầu rõ
 - Ví dụ: giao diện dễ sử dụng



Thu thập yêu cầu

- Kết quả của bước thu thập yêu cầu
 - Phát biểu về sự cần thiết và tính khả thi
 - Giới hạn lĩnh vực/chức năng của phần mềm
 - Danh sách những người liên quan/sử dụng phần mềm
 - Mô tả môi trường sẽ vận hành phần mềm
 - Danh sách các yêu cầu phần mềm đề xuất
 - Các ràng buộc phần mềm đề xuất



Thu thập yêu cầu

- Các kỹ thuật thu thập yêu cầu
 - Phỏng vấn
 - Thực hiện các cuộc họp/hội thảo
 - Chuẩn bị bảng câu hỏi điều tra
 - Chức năng mong đợi
 - Thời gian yêu cầu hoàn thành dự án
 - Kết quả một tiến trình nghiệp vụ
 - Quan sát hoạt động nghiệp vụ hiện tại
 - Đến nơi làm việc của khách hàng để khảo sát, quay phim,...
 - Tham khảo các chuyên gia trong lĩnh vực
 - Hiểu rõ các nghiệp vụ chuyên môn phức tạp



Thu thập yêu cầu

- Phỏng vấn

- Hiểu rõ nghiệp vụ hiện tại
- Hiểu rõ chi tiết yêu cầu
- Hiểu rõ mong muốn thực sự của khách hàng
- Chuẩn bị
 - Lập danh sách đối tượng, hẹn gặp, lập kế hoạch, chuẩn bị câu hỏi, phương tiện,...
- Câu hỏi
 - Câu hỏi đóng – mở
 - Câu hỏi ngắn gọn, tập trung vào chủ đề
- Cách thức tiến hành
 - Theo nhóm, ít nhất hai người
 - Phân công hỏi và ghi chép



Nghiên cứu khả thi

- Mục tiêu của nghiên cứu khả thi là đi đến kết luận:

- *Có nên phát triển hệ thống hay không?*
- Nội dung nghiên cứu khả thi tập trung vào những câu hỏi sau:
 - *Hệ thống xây dựng sẽ giúp gì cho tổ chức?*
 - *Hệ thống xây dựng sử dụng công nghệ nào và kinh phí bao nhiêu?*
 - *Hệ thống cần tích hợp với hệ thống nào đang sử dụng?*



Nghiên cứu khả thi

- Báo cáo khả thi được viết dựa trên thông tin, báo cáo thu thập được, những đánh giá ban đầu về hệ thống hiện tại và phác họa các phương án dự kiến.
- Câu hỏi đặt ra cho người của tổ chức:
 - Cái gì xảy ra nếu hệ thống không được triển khai?
 - Những vấn đề gì đang đặt ra cần giải quyết?
 - Hệ thống được đề xuất giúp họ như thế nào?
 - Những tích hợp gì cần phải có?
 - Công nghệ mới gì, kỹ năng gì cần có?
 - Những tiện ích gì cần sự trợ giúp từ hệ thống?



Phân tích yêu cầu

Những khó khăn của phân tích

- Khách hàng thường mơ hồ về các yêu cầu, không biết mình muốn cụ thể gì, dễ lẫn lộn giữa yêu cầu và mong muốn.
- Họ thể hiện yêu cầu theo thuật ngữ riêng
- Khách hàng đa dạng có thể đưa ra các yêu cầu mâu thuẫn.
- Những yếu tố tổ chức và chính sách có thể ảnh hưởng đến yêu cầu
- Yêu cầu thường mang tính đặc thù, khó hiểu, khó có chuẩn chung.
- Các yêu cầu thay đổi trong quá trình phân tích: môi trường nghiệp vụ thay đổi,..



Phân tích yêu cầu

- Đôi khi còn gọi là phát hiện yêu cầu.
- Nhà kỹ thuật cùng với khách hàng (người dùng, kỹ sư, nhà quản lý, chuyên gia,...-người liên quan) làm rõ lĩnh vực ứng dụng, các dịch vụ mà hệ thống cần cung cấp và các ràng buộc trong hoạt động của nó.
- Xây dựng các mô hình phân tích để làm rõ các yêu cầu miền.



Phân tích yêu cầu

- Phân loại yêu cầu
 - Chức năng
 - Phi chức năng
- Yêu cầu chức năng xuất phát từ các yêu cầu của khách hàng và nghiệp vụ trong hệ thống hiện tại
- Yêu cầu phi chức năng thường không lộ rõ
 - Thường do người phát triển đề xuất



Phân tích yêu cầu

Mục tiêu, mong muốn và yêu cầu

- **Mục tiêu, mong muốn:** *là cái hướng tới.*
 - Ví dụ: “**Xây dựng** giao diện thân thiện”
- **Yêu cầu:** là cái cụ thể kiểm tra được
 - Ví dụ: “Giao diện đồ họa có các lệnh được chọn bằng menu”
- Nhiệm vụ của người phân tích là các xác định đúng, đầy đủ và chính xác các yêu cầu.



Nguyên lý phân tích yêu cầu

Nguyên lý 1: *Mô hình hóa miền thông tin*

- Phải hiểu và biểu diễn được miền thông tin (problem domain)
 - định danh dữ liệu (đối tượng, thực thể)
 - định nghĩa các thuộc tính
 - thiết lập các mối quan hệ giữa các dữ liệu
- ⇒ *Từ điển dữ liệu, mô hình thực thể mối quan hệ, mô hình khái niệm*



Nguyên lý phân tích yêu cầu

Nguyên lý 2: *Mô hình hóa chức năng*

- Bản chất của phần mềm là biến đổi thông tin
 - định danh các chức năng (*biến đổi thông tin*)
 - xác định cách thức dữ liệu (*thông tin*) di chuyển trong hệ thống (*luồng dữ liệu*)
 - xác định các tác nhân tạo dữ liệu (*nguồn*) và tác nhân tiêu thụ dữ liệu (*đích*)

⇒ *Mô hình phân rã chức năng, mô hình luồng dữ liệu*



Nguyên lý phân tích yêu cầu

Nguyên lý 3: *Mô hình hóa hành vi*

- Phần mềm (hệ thống) có trạng thái (hành vi)
 - xác định các trạng thái của hệ thống
 - ví dụ: *giao diện đồ họa, section trong ứng dụng web*
 - xác định các dữ kiện làm thay đổi hành vi hệ thống
 - ví dụ: *bàn phím, chuột, các cổng thông tin...*

⇒ *Biểu đồ trạng thái*



Nguyên lý phân tích yêu cầu

Nguyên lý 4: *Phân hoạch các mô hình*

- Làm mịn, phân hoạch và biểu diễn các mô hình ở các mức khác nhau
 - làm mịn các mô hình dữ liệu
 - tạo cây (mô hình) phân rã chức năng
 - biểu diễn hành vi ở các mức chi tiết khác nhau

⇒ *Mô hình luồng dữ liệu các mức 1,2,3,..*



Nguyên lý phân tích yêu cầu

Nguyên lý 5: *Tìm hiểu vấn đề bản chất*

- Nhìn nhận bản chất của yêu cầu (*làm gì, điều kiện gì,..*)
- Không quan tâm đến cách thức cài đặt (*làm như thế nào*)



Phân hoạch yêu cầu

Có thể phân hoạch yêu cầu theo hai cách

- Phân loại theo đặc trưng:
 - Yêu cầu tương hỗ: chịu ảnh hưởng của môi trường
 - Yêu cầu nảy sinh: nhận ra trong quá trình phát triển
 - Yêu cầu hệ quả: là kết quả của việc áp dụng hệ thống dựa trên máy tính
 - Yêu cầu tương thích: phụ thuộc vào hệ khác hay tiến trình tổ chức
- Phân loại theo mức độ quan trọng: chính, phụ



Thẩm định yêu cầu

- Thẩm định yêu cầu liên quan tới việc kiểm tra tính đúng đắn, tính đầy đủ, tính hiện thực và kiểm tra được của yêu cầu.
 1. Thỏa mãn được nhu cầu của người dùng?
 2. Yêu cầu không mâu thuẫn nhau?
 3. Yêu cầu phải đầy đủ (chức năng, ràng buộc)?
 4. Yêu cầu là hiện thực?
 5. Yêu cầu có thể kiểm tra được?



Thẩm định yêu cầu

- Thường xuyên thẩm định yêu cầu
- Cả khách hàng và người phát triển đều phải thẩm định yêu cầu
- Việc thẩm định có thể tổ chức hình thức hoặc không hình thức
- Trao đổi giữa khách hàng, nhà phát triển và người sử dụng cuối có thể giải quyết sớm các khó khăn



Thẩm định yêu cầu

Các kỹ thuật thẩm định

1. Xem xét lại yêu cầu: bằng cách phân tích một cách có hệ thống các yêu cầu (có sự kết hợp cả nhà phát triển và khách hàng, tiến hành thường xuyên)
2. Làm bản mẫu: sử dụng các mô hình có khả năng thực thi để kiểm tra lại yêu cầu
3. Tạo các trường hợp kiểm thử: kiểm tra tính có thể kiểm tra được của yêu cầu
4. Phân tích tính nhất quán bằng cách tự động: sử dụng CASE (Computer Aided Software Engineering)



Đặc tả yêu cầu phần mềm

- Đặc tả các yêu cầu phần mềm là công việc xây dựng các tài liệu đặc tả
 - Có thể sử dụng tới các công cụ như: mô hình hóa, mô hình toán học hình thức (a formal mathematical model), tập hợp các kịch bản sử dụng, các nguyên mẫu hoặc bất kỳ một tổ hợp các công cụ nói trên
- Đặc tả chi tiết các yêu cầu đã phân tích
- Chất lượng của hồ sơ đặc tả đánh giá qua các tiêu thức
 - Tính rõ ràng, chính xác
 - Tính phù hợp
 - Tính đầy đủ, hoàn thiện



Đặc tả yêu cầu (Requirements Specification)

- Có thể sử dụng các cấu trúc tài liệu đặc tả yêu cầu khác nhau (IEEE)
- Đầy đủ
 - Mọi yêu cầu của người dùng phải được mô tả
- Không mâu thuẫn nhau
- Chính xác : yêu cầu không được mơ hồ
 - Chi phí phát sinh cho sửa đổi rất cao



Các mức trừu tượng của yêu cầu

- Yêu cầu nên được biểu diễn ở nhiều mức trừu tượng khác nhau phục vụ cho nhiều đối tượng khác nhau
- Xác định yêu cầu
 - Mô tả các dịch vụ mà phần mềm cần cung cấp
 - Viết bằng ngôn ngữ tự nhiên
 - Hướng người dùng
- Đặc tả yêu cầu
 - Tài liệu có cấu trúc
 - Mô tả chi tiết, chính xác về yêu cầu
 - Dùng làm bản hợp đồng



Ví dụ: Chức năng kiểm tra lỗi chính tả

- ⇒ Định nghĩa:
 - Thông báo các lỗi chính tả của văn bản
- ⇒ Đặc tả:
 - Các lỗi chính tả được gạch đỏ bên dưới
 - Các lỗi soạn thảo được gạch xanh bên dưới
 - Lỗi chính tả:
 - *Từ đơn không có trong từ điển*
 - Lỗi soạn thảo
 - *Thừa dấu cách*
 - *Không viết hoa đầu câu*



Các thành phần của hồ sơ đặc tả

- Đặc tả phi hình thức (Informal specifications) được viết bằng ngôn ngữ tự nhiên
- Đặc tả hình thức (Formal specifications) được viết bằng tập các ký pháp có các quy định về *cú pháp (syntax)* và *ý nghĩa (sematic)* rất chặt chẽ
- Đặc tả vận hành chức năng (Operational specifications) mô tả các hoạt động của hệ thống phần mềm sẽ xây dựng
- Đặc tả mô tả (Descriptive specifications) – đặc tả các đặc tính đặc trưng của phần mềm



Đặc tả yêu cầu

- Đặc tả chức năng (Operational Specifications)
 - sử dụng các công cụ tiêu biểu sau
 - Biểu đồ luồng dữ liệu (Data Flow Diagrams)
 - Máy trạng thái hữu hạn (Finite State Machines)
 - Mạng Petri (Petri nets)



Đặc tả yêu cầu

- Đặc tả mô tả (Descriptive Specifications)

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



Các kỹ thuật đặc tả



Khái niệm đặc tả

- Định nghĩa một hệ thống, mô-đun hay một sản phẩm cần phải **làm cái gì**
- Không mô tả nó phải làm như thế nào
- Mô tả những **tính chất của vấn đề** đặt ra
- Không mô tả những tính chất của giải pháp cho vấn đề đó



Khái niệm đặc tả

- Đặc tả là hoạt động được tiến hành trong các pha khác nhau của tiến trình phần mềm:
 - Đặc tả yêu cầu (requirement specification)
 - Sự thống nhất giữa người sử dụng tương lai và người thiết kế
 - Đặc tả kiến trúc hệ thống (system architect specification)
 - Sự thống nhất giữa những người thiết kế và những người cài đặt
 - Đặc tả mô-đun (module specification)
 - Sự thống nhất giữa những người lập trình cài đặt mô-đun và những người lập trình sử dụng mô-đun



Tại sao phải đặc tả

- Hợp đồng
 - Sự thống nhất giữa người dùng và người phát triển sản phẩm
- Hợp thức hóa
 - Sản phẩm cuối ra phải thực hiện chính xác những gì mong muốn
- Trao đổi
 - giữa người sử dụng và người phát triển giữa những người phát triển
- Tái sử dụng



Phân loại các kỹ thuật đặc tả

- Đặc tả phi hình thức (informal)
 - Ngôn ngữ tự nhiên tự do
 - Ngôn ngữ tự nhiên có cấu trúc
 - Các ký hiệu đồ họa
- Đặc tả nửa hình thức (semi-informal)
 - trộn lẫn cả ngôn ngữ tự nhiên, các kí hiệu toán học và các kí hiệu đồ họa
- Đặc tả hình thức (formal)
 - Đặc tả bằng mô hình
 - Đặc tả bằng ngôn ngữ hình thức hoá
 - Đặc tả bằng công thức toán học



Đặc tả hình thức hay không hình thức?

- Đặc tả hình thức
 - Chính xác (toán học)
 - Hợp thức hóa hình thức (công cụ hóa)
 - công cụ trao đổi: khó đọc, khó hiểu
 - khó sử dụng
- Đặc tả không hình thức
 - Dễ hiểu, dễ sử dụng
 - Mềm dẻo
 - Thiếu sự chính xác
 - Không chắc chắn, Nhập nhằng



Đặc tả phi hình thức

- Ví dụ: chức năng kiểm tra lỗi chính tả
 - Yêu cầu: thông báo các lỗi chính tả khi duyệt
 - Đặc tả:
 - Các lỗi chính tả được gạch đỏ bên dưới
 - Các lỗi soạn thảo được gạch xanh bên dưới
 - Lỗi chính tả:
 - Từ đơn không có trong từ điển
 - Lỗi soạn thảo
 - Thừa dấu cách
 - Không viết hoa đầu câu



Ứng dụng đặc tả hình thức

- ứng dụng trong các giai đoạn sớm của tiến trình phát triển
- Hạn chế lỗi trong phát triển phần mềm
- Ứng dụng chủ yếu trong phát triển các hệ thống “quan trọng” (critical system)
 - Hệ thống điều khiển
 - Hệ thống nhúng
 - Hệ thống thời gian thực



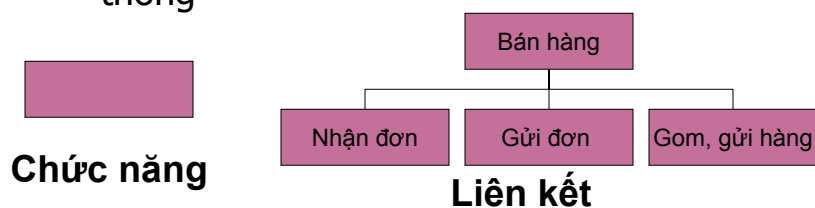
Các kỹ thuật đặc tả hình thức

- Biểu đồ luồng dữ liệu
- Biểu đồ thực thể kết hợp
- Làm bản mẫu
- Máy trạng thái hữu hạn
- Mạng Petri
- Điều kiện trước sau
- Kiểu trừu tượng
- Ngôn ngữ đặc tả Z

Biểu đồ phân rã chức năng

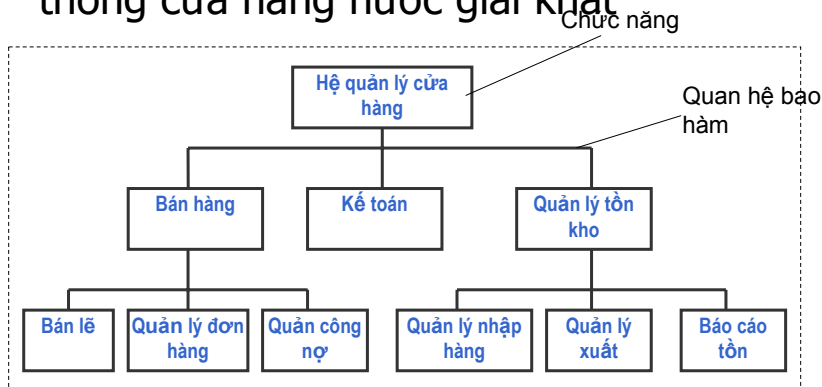
■ Function Decomposition Diagram – FDD

- Xác định phạm vi của hệ thống
- Phân hoạch chức năng
- Tạo nền tảng cho thiết kế kiến trúc hệ thống



Biểu đồ phân rã chức năng

- Ví dụ: biểu diễn các chức năng của hệ thống cửa hàng nước giải khát



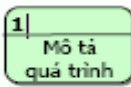

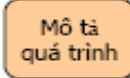
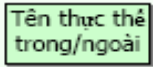
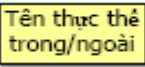
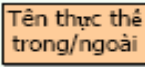
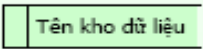
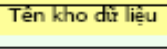
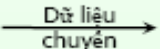
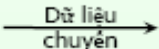
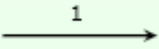
Biểu đồ dòng dữ liệu

Data Flow Diagram – DFD

- Mô tả cách thức dữ liệu di chuyển và được xử lý, lưu trữ trong hệ thống
- DFD dễ viết, dễ đọc, dễ hiểu được sử dụng phổ biến
- DFD được xây dựng từ các hình vẽ và các kí hiệu qui ước.
- Có nhiều mức chi tiết khác nhau (phân tích có cấu trúc)
- Có nhiều cách xây dựng DFD, thông dụng là phương pháp DeMarco-Yourdon, Gane Sarson và Merise (pháp)

Biểu đồ dòng dữ liệu

Một số phương pháp vẽ DFD

Phương pháp	Gane-Sarson	DeMarco-Yourdon	Merise
Quá trình			
Thực thể			
Kho dữ liệu			
Dòng dữ liệu			

Biểu đồ dòng dữ liệu

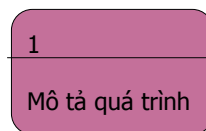
■ DFD – Các thành tố

■ Quá trình (Process)

- Mô tả hoạt động (activities) hay phép biến đổi (Transform) một hoặc nhiều dòng dữ liệu vào (input) thành một hoặc nhiều dòng dữ liệu ra (output)
- Quá trình không chỉ ra chi tiết logic hay thủ tục xử lý
- Trong sơ đồ DFD, quá trình có thể là một người sử dụng hoặc máy tính xử lý

■ Ví dụ:

- Nhận yêu cầu khách hàng
- Thủ tục giao hàng
- Thông báo
- Xử lý điểm...



Biểu đồ dòng dữ liệu

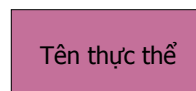
■ DFD – Các thành tố

■ Thực thể (Entity)

- Xác định ranh giới, phạm trù hay phạm vi, ngữ cảnh của hệ thống đang xét
- Cung cấp cái vào cho hệ thống và lấy cái ra từ hệ thống
- Các thực thể có thể nằm bên trong hay bên ngoài, tạo thành các nguồn hay các đích cho hệ thống
- Mỗi thực thể có thể là một người, tổ chức hoặc một hệ thống khác tương tác với hệ thống đang xét

■ Ví dụ:

- Khách hàng
- Học viên
- Giáo viên





Biểu đồ dòng dữ liệu

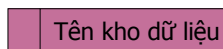
■ DFD – Các thành tố

■ Kho dữ liệu (data stores):

- Chỗ chứa các thông tin được lưu theo thời gian, được xử lý thủ công hay tự động
- Đó là các tập tin, cơ sở dữ liệu hay bất cứ các hình thức lưu trữ dữ liệu nào (bảng biểu báo cáo, từ điển, hộp thư, danh bạ,...)

■ Ví dụ:

- Hồ sơ học viên
- Sổ nợ
- Kết quả tuyển sinh...



Biểu đồ dòng dữ liệu

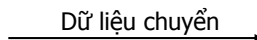
■ DFD – Các thành tố

■ Dòng dữ liệu (Data flows)

- Phương tiện luân chuyển thông tin thể hiện cái vào và cái ra, thể hiện sự tương tác trong hệ thống
- Có thể là báo cáo, biểu mẫu, văn bản, thư tín, thông điệp hay dữ liệu nói chung
- Được tạo thành từ các vật mang thông tin (giấy, màn hình,...) có cùng bản chất
- Đi từ nơi phát (nguồn) đến nơi nhận (đích)

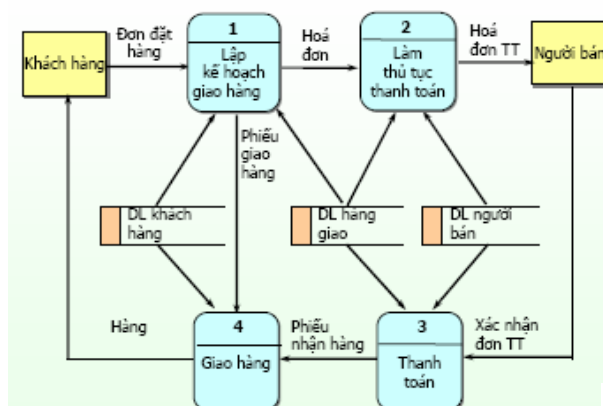
■ Ví dụ:

- Yêu cầu cho vay
- Tra lời vay
- Báo kết quả thi



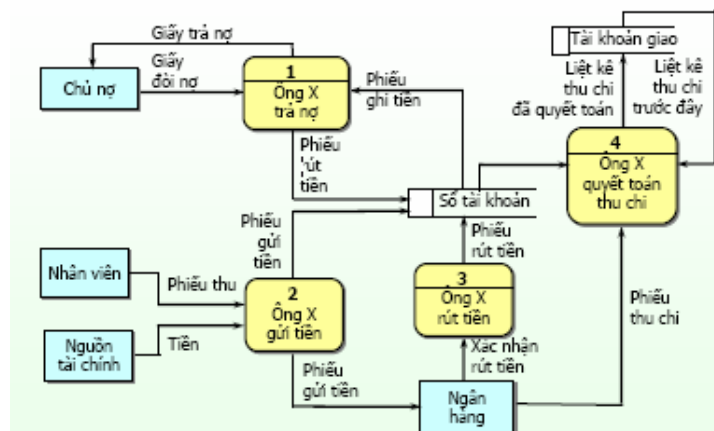
Biểu đồ dòng dữ liệu

- Ví dụ: DFD hệ thống mua bán hàng



Biểu đồ dòng dữ liệu

- Ví dụ: DFD hệ thống tài chính cá nhân





Biểu đồ dòng dữ liệu

■ DFD – Các bước thực hiện

- Phân rã chức năng hệ thống
- Liệt kê các tác nhân, các khoản mục dữ liệu
- Vẽ DFD cho các mức
- Làm mịn DFD cho các mức từ DFD ngữ cảnh



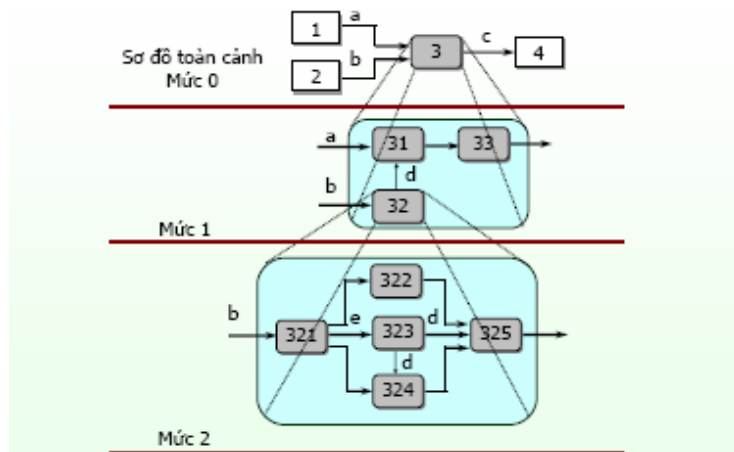
Biểu đồ dòng dữ liệu

■ DFD – Một số nguyên tắc

- Các tiến trình phải có luồng vào và luồng ra
- Không có luồng dữ liệu giữa các tác nhân – tác nhân, tác nhân – kho dữ liệu, kho dữ liệu – kho dữ liệu.
- Luồng dữ liệu không quay lại nơi xuất phát
- Làm mịn cần đảm bảo đầy đủ các yếu tố của bước trước (tác nhân, luồng dữ liệu, kho dữ liệu).

■ Biểu đồ dòng dữ liệu

■ Các mức DFD



■ Biểu đồ thực thể quan hệ

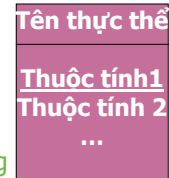
■ ERD – Entity Relation Diagram

- Phân tích dữ liệu độc lập với xử lý
- Nghiên cứu miền thông tin
- Tạo ra mô hình trừu tượng hướng khách hàng
- Xác định mối liên hệ giữa các dữ liệu

Biểu đồ thực thể quan hệ

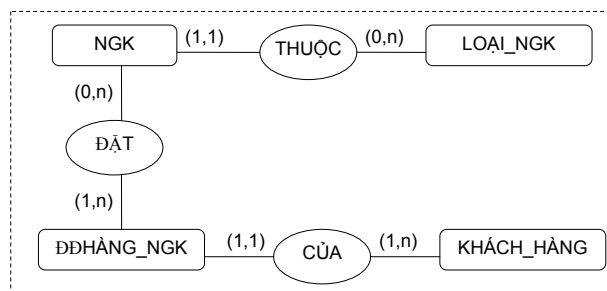
ER – khái niệm

- Thực thể:
 - Là các đối tượng thế giới thực mà chúng ta muốn xử lý
 - Có thể là đối tượng thực hoặc trừu tượng
- Thuộc tính:
 - là các đặc điểm, tính chất của thực thể
- Quan hệ:
 - Mỗi liên hệ giữa các thực thể
 - Là thông tin cần lưu trữ, xử lý
- Kế thừa: quan hệ thừa kế giữa các thực thể



Biểu đồ thực thể quan hệ

ER – ví dụ



NGK(MA_NGK, TEN_NGK, HIEU, LOAI, DVTINH, DON_GIA)
 ĐDHÀNG_NGK(SO_DDH, NGÀY_DAT, KHÁCH_HÀNG, NGÀYGIAO, TRẠNG THAI)
 CHITIET_DDH(MA_NGK, SO_DDH, SL_DAT, DONGIA_DAT)



Biểu đồ thực thể quan hệ

- ER – Xác định thực thể quan hệ
 - Thực thể:
 - Là các danh từ (trong câu mô tả các yêu cầu)
 - Có các thuộc tính khóa (xác định duy nhất)
 - Quan hệ
 - Hoạt động xảy ra giữa các thực thể
 - Có nghĩa với hoạt động của hệ thống
 - Là động từ



Biểu đồ thực thể quan hệ

- ER – Các bước
 - Mô tả thực thể và sự liên kết giữa các thực thể
 - Mô tả thực thể và các mối quan hệ
 - Mô tả thực thể, các mối quan hệ và các thuộc tính



Từ điển dữ liệu

- Phương pháp có tính hình thức để mô tả dữ liệu mà hệ thống xử lý
- Ký pháp để mô tả các dữ liệu điều khiển và miền giá trị của chúng
- Chứa đựng các thông tin về nơi (modul) và cách thức xử lý dữ liệu
- Thường được tạo bằng các công cụ trợ giúp

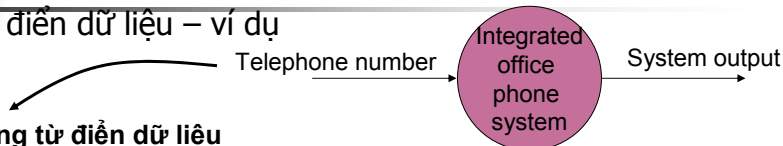


Từ điển dữ liệu

- Các khoản mục từ điển dữ liệu
 - Tên (name): tên dữ liệu
 - Bí danh (alias): tên gọi khác của dữ liệu
 - Vị trí (where): tên các modul xử lý
 - Cách thức (how): vai trò của dữ liệu và cách thức xử lý
 - Ký pháp (Description): ký pháp mô tả dữ liệu
 - Format: Kiểu dữ liệu, giá trị mặc định,...

Từ điển dữ liệu

Từ điển dữ liệu – ví dụ



Bảng từ điển dữ liệu

Name:	Telephone number
Aliases:	Phone number, number
Where/ How used:	Read_phone_number(input) Display_phone_number(output) Analyze_long_distance_calls (input)
Description:	telephone no. = [local extension outside no. 0] outside no. = 9 + [service code domestic no.] service code = [211 411 611 911] domestic no. = ((0) + area code) + local number area code = *three numeral designator*
Format:	alphanumeric data

Làm bản mẫu trong phân tích

- Trong nhiều trường hợp, mô hình chạy thử là phương pháp duy nhất để xác định yêu cầu
- Phần mềm lớn, phức tạp \Rightarrow **bản mẫu** (trực quan)
 - Bản mẫu phần mềm: phát triển yêu cầu
 - Bản mẫu phần cứng: kiểm tra thiết kế



Làm bản mẫu trong phân tích

Các bước thực hiện

- **Bước 1:** Đánh giá yêu cầu và xác định có nên làm bản mẫu không; độ phức tạp, chủng loại, khách hàng
- **Bước 2:** Biểu diễn văn tắt yêu cầu (yêu cầu chức năng, yêu cầu phi chức năng)
- **Bước 3:** Thiết kế nhanh kiến trúc, cấu trúc dữ liệu
- **Bước 4:** Phát triển, kiểm thử
 - Sử dụng các thành phần sẵn có
 - Dùng các ngôn ngữ bậc cao
 - Các thuật toán để cài đặt
- **Bước 5:** Người dùng đánh giá (bước quan trọng???)
- **Bước 6:** Lặp lại bước 2-5 cho đến khi đủ yêu cầu



Làm bản mẫu trong phân tích

Ưu điểm

- Loại bớt hiểu nhầm
- Phát hiện thiếu hụt chức năng
 - Ví dụ: soạn thảo kéo theo kiểm tra chính tả
- Phát hiện các điểm yếu
 - Khó sử dụng
 - Thao tác không an toàn
- Kiểm tra tính khả thi, hữu ích
 - Có thể xây dựng được không
 - Có thực sự cần không
- Làm cơ sở cho đặc tả: *làm giống như thế này nhưng tốt hơn*
- Huấn luyện người sử dụng
- Hỗ trợ kiểm thử (so sánh kết quả)

⇒ *Làm bản mẫu là kỹ thuật tránh rủi ro*



Làm bản mẫu trong phân tích

■ Nhược điểm

- Các yêu cầu phi chức năng thường không được thể hiện đầy đủ
- Làm tăng chi phí: cần phải ước lượng chi phí chính xác của bản mẫu
- Các yêu cầu thay đổi quá nhanh làm bản mẫu trở nên vô nghĩa
- Người dùng không sử dụng bản mẫu theo cách thông thường (kết quả đánh giá không có giá trị)



Làm bản mẫu trong phân tích

■ Ngôn ngữ làm bản mẫu

- Java
- Visual Basic
- Prolog, Lisp, Python
- Các ngôn ngữ script khác
 - Perl, PHP, shell script



Làm bản mẫu trong phân tích

- Bản mẫu giao diện/trực quan
 - Sử dụng các ngôn ngữ, công cụ trực quan
 - VB
 - Delphi, J Buidar
 - FrontPage
 - Sử dụng lại một lượng lớn các thư viện sẵn có
 - Tính cấu trúc không cao, khó tích hợp kết quả của nhiều nhóm, khó bảo trì

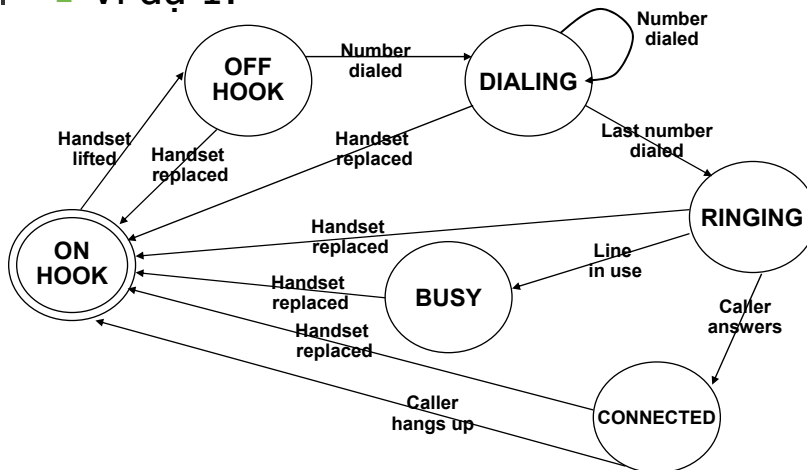


Máy trạng thái hữu hạn

- Finite State Machine
 - Mô tả các luồng điều khiển
 - Biểu diễn dạng đồ thị
 - Bao gồm:
 - Tập hợp các trạng thái S (các nút của đồ thị)
 - Tập hợp các dữ liệu vào I (các nhãn của các cung)
 - Tập hợp các chuyển tiếp T: $S \times I \rightarrow S$ (các cung có hướng của đồ thị)
 - Khi có một dữ liệu vào một trạng thái sẽ chuyển sang trạng thái khác

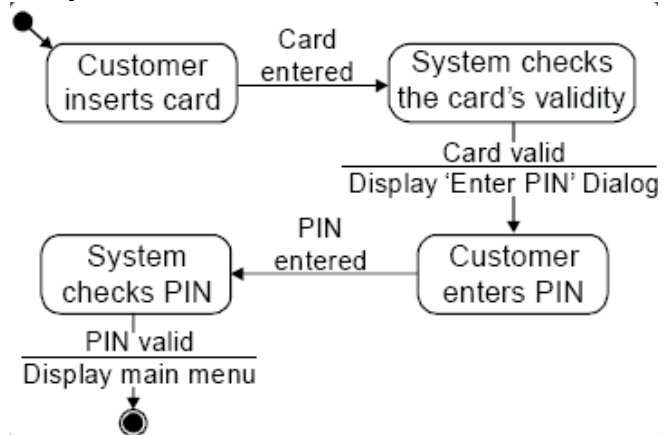
Máy trạng thái hữu hạn

Ví dụ 1:

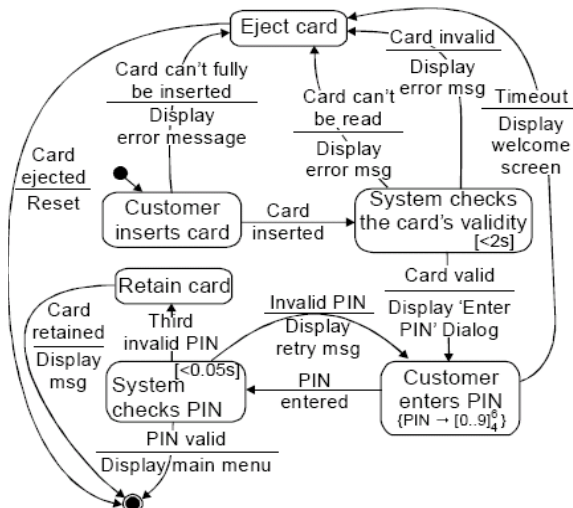


Máy trạng thái hữu hạn

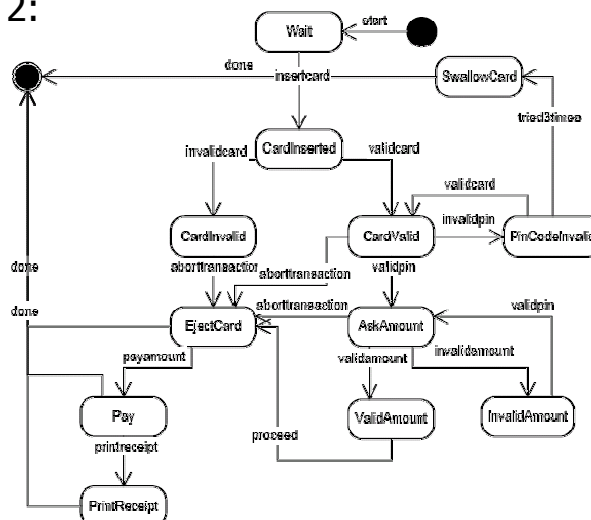
Ví dụ 2:



- Ví dụ 2:



- Ví dụ 2:





Máy trạng thái hữu hạn

- Biểu diễn hệ thống phức tạp
- Hạn chế khi đặc tả những hệ thống không đồng bộ
 - Các thành phần của hệ thống hoạt động song song hoặc cạnh tranh



Điều kiện trước/sau (Pre/Post Condition)

- Được dùng để đặc tả các hàm hoặc mô đun
- Đặc tả các tính chất của dữ liệu trước và sau khi thực hiện hàm
 - Pre-condition: đặc tả các ràng buộc trên tham số **trước** khi hàm được thực hiện
 - Post-condition: đặc tả các ràng buộc trên tham số **sau** khi hàm được thực hiện
- Có thể sử dụng ngôn ngữ phi hình thức, hình thức hoặc ngôn ngữ lập trình để đặc tả các điều kiện.



Điều kiện trước/sau

- Ví dụ: đặc tả hàm tìm kiếm

Function search(a: danh sách phần tử kiểu K,
size: số phần tử của danh sách,
e: phần tử kiểu K,
result: kiểu Boolean)

pre	$\forall i, 1 \leq i \leq \text{size}, a[i] \leq a[i+1]$
post	$\text{result} = (\exists i, 1 \leq i \leq \text{size}, a[i] = e)$



Điều kiện trước/sau

- Bài tập: đặc tả các hàm
 - Sắp xếp một danh sách số nguyên
 - Đảo ngược các phần tử của danh sách
 - Đếm số phần tử có giá trị e trong một danh sách số nguyên

Kiểu trừu tượng (Abstract type)

- Mô tả dữ liệu và các thao tác trên dữ liệu đó ở một mức trừu tượng độc lập với cách cài đặt dữ liệu bởi ngôn ngữ lập trình
- Đặc tả một kiểu trừu tượng gồm:
 - Tên của kiểu trừu tượng
 - Dùng từ khóa **sort**
 - Khai báo các kiểu trừu tượng đã tồn tại được sử dụng
 - Dùng từ khóa **import**
 - Các thao tác trên kiểu trừu tượng
 - Dùng từ khóa **operations**

Kiểu trừu tượng

- Ví dụ 1: đặc tả kiểu trừu tượng Boolean

```
sort Boolean
operations
  true      : → Boolean
  false     : → Boolean
  ¬ _       : Boolean → Boolean
  _ ^ _     : Boolean x Boolean → Boolean
  _ v _     : Boolean x Boolean → Boolean
```

- Một thao tác không có tham số là một hằng số
- một giá trị của kiểu trừu tượng được định nghĩa biểu diễn bởi kí tự “_”



Kiểu trừu tượng

- Ví dụ 2: đặc tả kiểu trừu tượng Vector

sort Vector

imports Integer, Element, Boolean

operations

vect : Integer x Integer \rightarrow Vector

init : Vector x Integer \rightarrow Boolean

ith : Vector x Integer \rightarrow Element

change-ith : Vector x Integer x Element \rightarrow Vector

supborder : Vector \rightarrow Integer

infborder : Vector \rightarrow Integer



Kiểu trừu tượng

- Các thao tác trên kiểu trừu tượng chỉ được định nghĩa mà không chỉ ra ngữ nghĩa của nó
 - Tức là ý nghĩa của thao tác
- Sử dụng các tiên đề để chỉ ra ngữ nghĩa của các thao tác
 - Sử dụng từ khóa **axioms**
- Định nghĩa các ràng buộc mà một thao tác được định nghĩa
 - Sử dụng từ khóa **precondition**

Kiểu trừu tượng

- Ví dụ 2: đặc tả kiểu trừu tượng Vector

precondition

ith(v, i) is-defined-ifonlyif

$\text{infborder}(v) \leq i \leq \text{supborder}(v) \ \& \ \text{init}(v, i) = \text{true}$

axioms

$\text{infborder}(v) \leq i \leq \text{supborder}(v) \Rightarrow \text{ith}(\text{change-ith}(v, i, e), i) = e$

$\text{infborder}(v) \leq i \leq \text{supborder}(v) \ \& \ \text{infborder}(v) \leq j \leq \text{supborder}(v) \ \& \ i \neq j \Rightarrow$

$\text{ith}(\text{change-ith}(v, i, e), j) = \text{ith}(v, j)$

$\text{init}(\text{vect}(i, j), k) = \text{false}$

$\text{infborder}(v) \leq i \leq \text{supborder}(v) \Rightarrow \text{init}(\text{change-ith}(v, i, e), i) = \text{true}$

$\text{infborder}(v) \leq i \leq \text{supborder}(v) \ \& \ i \neq j \Rightarrow \text{init}(\text{change-ith}(v, i, e), j) = \text{init}(v, j)$

$\text{infborder}(\text{vect}(i, j)) = i$

$\text{infborder}(\text{change-ith}(v, i, e)) = \text{infborder}(v)$

$\text{supborder}(\text{vect}(i, j)) = j$

$\text{supborder}(\text{change-ith}(v, i, e)) = \text{supborder}(v)$

with

v: Vector; i, j, k: Integer; e: Element

Kiểu trừu tượng

- Bài tập:

- Đặc tả kiểu trừu tượng Cây nhị phân
- Đặc tả kiểu trừu tượng Tập hợp



Kiểu trừu tượng

- **sort** Circle

- Import Integer, float
- Operations
 - Init : Integer x Integer x float -> Circle
 - Unit_Circle: ->Circle
 - Bankinh: Circle -> float
 - Get_X: Circle -> Integer
 - Get_Y: Circle -> Integer
 - Zoom: Circle x float -> Circle
 - Tinh_tien:Circle x Integer x Integer -> Circle



Kiểu trừu tượng

- axioms

- Bankinh(Unit_Circle) = 1
- Get_x(Unit_Circle) = Get_Y(Unit_Circle) = 0
- Bankinh(Init(x, y, r)) = r
- Get_X(Init(x, y, r)) = x
- Get_Y(Init(x, y, r)) = y
- $C = \text{Init}(x, y, r) \Rightarrow \text{Bankinh}(\text{Tinhhtien}(C, dx, dy)) = r$
- $C = \text{Init}(x, y, r) \Rightarrow \text{Get_X}(\text{Tinhhtien}(C, dx, dy)) = x + dx$
- $C = \text{Init}(x, y, r) \Rightarrow \text{Get_Y}(\text{Tinhhtien}(C, dx, dy)) = y + dy$
- $C = \text{Init}(x, y, r) \Rightarrow \text{Bankinh}(\text{Zoom}(C, dr)) = r + dr$
- $C = \text{Init}(x, y, r) \Rightarrow \text{Get_X}(\text{Zoom}(C, dr)) = x$
- $C = \text{Init}(x, y, r) \Rightarrow \text{Get_Y}(\text{Zoom}(C, dr)) = y$

- With

- C: Circle ; x, y, dx, dy: integer; r, dr: float



Mạng Petri (Petri Nets)

- Thích hợp để mô tả các hệ thống không đồng bộ với những hoạt động đồng thời
- Mô tả luồng điều khiển của hệ thống
- Đề xuất năm 1962 bởi Carl Adam
- Có loại
 - Mạng Petri (cổ điển)
 - Mạng Petri mở rộng

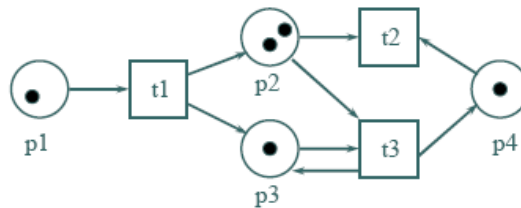


Mạng Petri

- Gồm các phần tử
 - Một tập hữu hạn các *nút* (O)
 - Một tập hữu hạn các *chuyển tiếp* (\square)
 - Một tập hữu hạn các *cung* (\rightarrow)
 - Các cung nối các nút với các chuyển tiếp hoặc ngược lại
 - Mỗi nút có thể chứa một hoặc nhiều *thẻ* (\bullet)

Mạng Petri

■ Ví dụ:



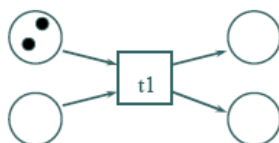
Mạng Petri

- Mạng Petri được định nghĩa bởi sự đánh dấu các nút của nó
- Việc đánh dấu các nút được tiến hành theo nguyên tắc sau:
 - Mỗi chuyển tiếp có các nút vào và các nút ra
 - Nếu tất cả các nút vào của một chuyển tiếp có ít nhất một thẻ, thì chuyển tiếp này có thể *vượt qua* được.
 - Nếu chuyển tiếp này được thực hiện thì tất cả các nút vào của chuyển tiếp sẽ bị lấy đi một thẻ, và một thẻ sẽ được thêm vào tất cả các nút ra của chuyển tiếp
 - Nếu chuyển tiếp là có thể vượt qua thì chọn chuyển tiếp nào cũng được

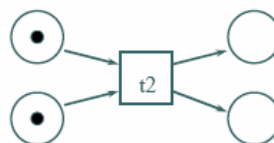


Mạng Petri

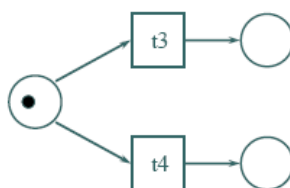
■ Ví dụ



t1 không thể vượt qua được



t2 có thể vượt qua được

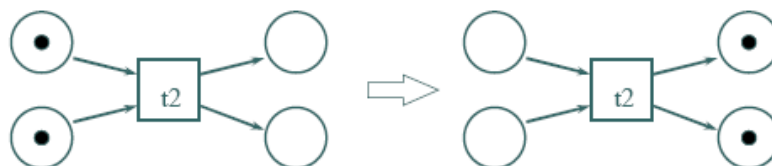


hoặc t3 được vượt qua
hoặc t4 được vượt qua



Mạng Petri

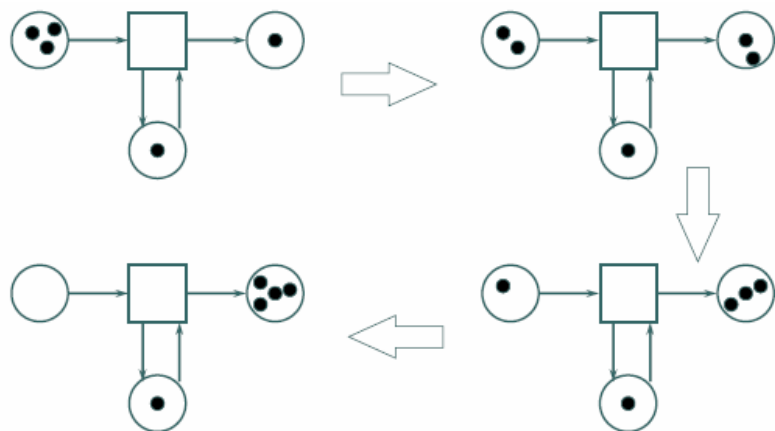
■ Ví dụ



khi t2 được vượt qua

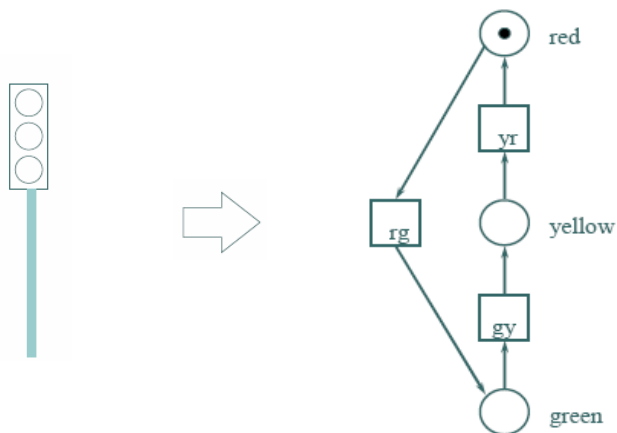
Mạng Petri

Ví dụ



Mạng Petri

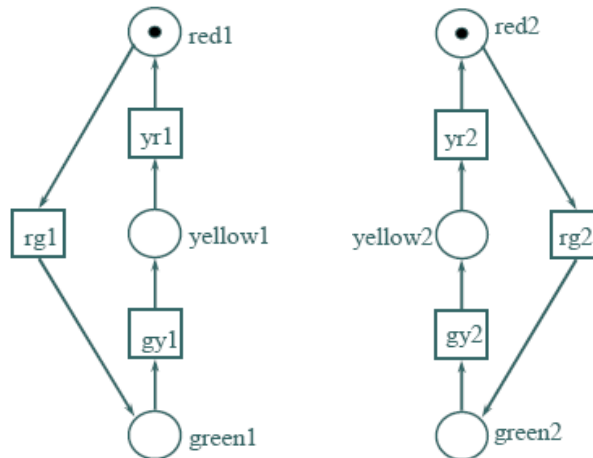
Ví dụ 1: Mô tả hoạt động của đèn giao thông





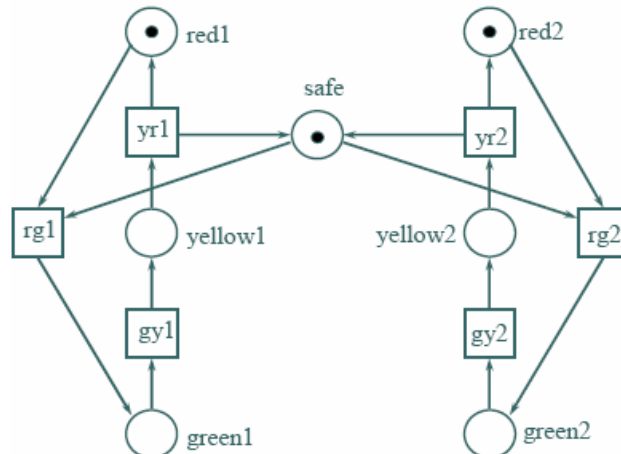
Mạng Petri

- Ví dụ 1: Mô tả hoạt động của 2 đèn giao thông



Mạng Petri

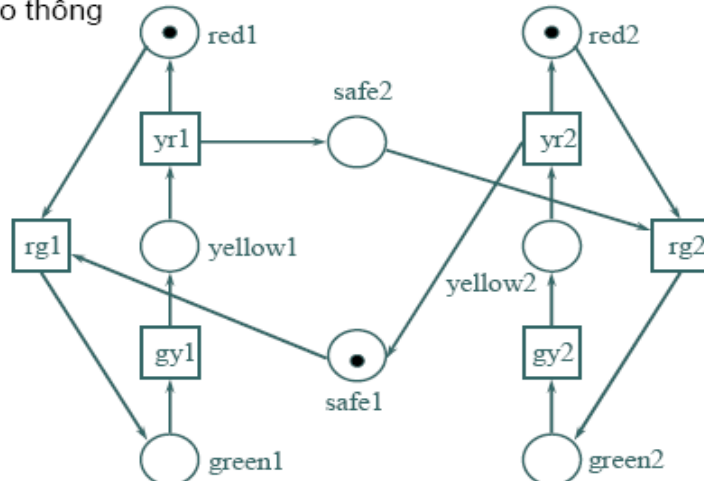
- Ví dụ 1: Mô tả hoạt động an toàn của 2 đèn giao thông





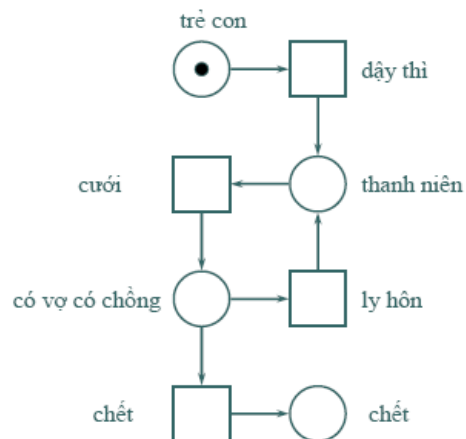
Mạng Petri

- Ví dụ 1: mô tả hoạt động an toàn và hợp lý của 2 đèn giao thông



Mạng Petri

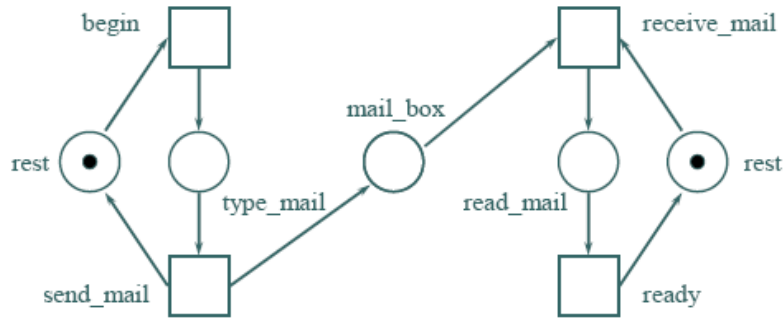
- Ví dụ 2: Chu kì sống của con người





Mạng Petri

■ Ví dụ 3: Viết thư và đọc thư



Mạng Petri

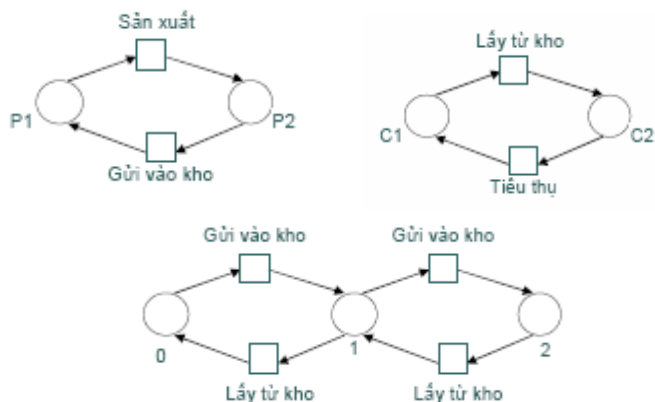
■ Ví dụ 4:

- Hệ thống cần mô tả bao gồm một nhà sản xuất, một nhà tiêu thụ và một kho hàng chỉ chứa được nhiều nhất hai sản phẩm
- Nhà sản xuất có 2 trạng thái:
 - P1: đang sản xuất
 - P2: không sản xuất
- Nhà tiêu thụ có 2 trạng thái:
 - C1: có sản phẩm để tiêu thụ
 - C2: không có sản phẩm để tiêu thụ
- Kho hàng có 3 trạng thái
 - Chứa 0 sản phẩm
 - Chứa 1 sản phẩm
 - Chứa 2 sản phẩm



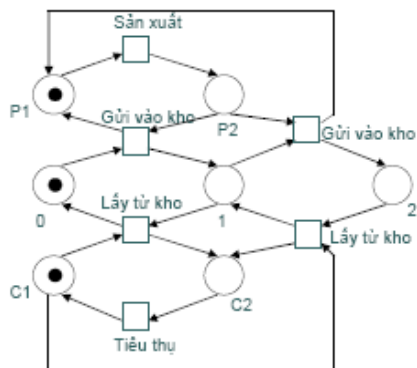
Mạng Petri

- Ví dụ 4: mô tả tách rời mỗi thành phần



Mạng Petri

- Ví dụ 4: Mô tả kết hợp các thành phần





So sánh các kỹ thuật đặc tả

<i>Phương pháp đặc tả</i>	<i>Thể loại</i>	<i>Điểm mạnh</i>	<i>Điểm yếu</i>
- Ngôn ngữ tự nhiên	Không hình thức	<ul style="list-style-type: none">- Dễ học- Dễ sử dụng- Dễ hiểu đối với khách hàng.	<ul style="list-style-type: none">- Không chính xác- Không rõ ràng, mâu thuẫn và không đầy đủ.
- Mô hình thực thể quan hệ - Phân tích hệ thống	Bán hình thức	<ul style="list-style-type: none">- Khách hàng có thể hiểu.- Chính xác hơn các phương pháp không hình thức	<ul style="list-style-type: none">- Không chính xác như các phương pháp hình thức.- Khó định lượng thời gian.
- Máy hữu hạn trạng thái - Mạng Petri	Hình thức	<ul style="list-style-type: none">- Cực kỳ chính xác- Có thể giảm các lỗi đặc tả- Có thể giảm chi phí và nhân lực- Có thể hỗ trợ việc chứng minh tính chính xác	<ul style="list-style-type: none">- Khó học- Khó sử dụng- Khách hàng hầu như không hiểu được