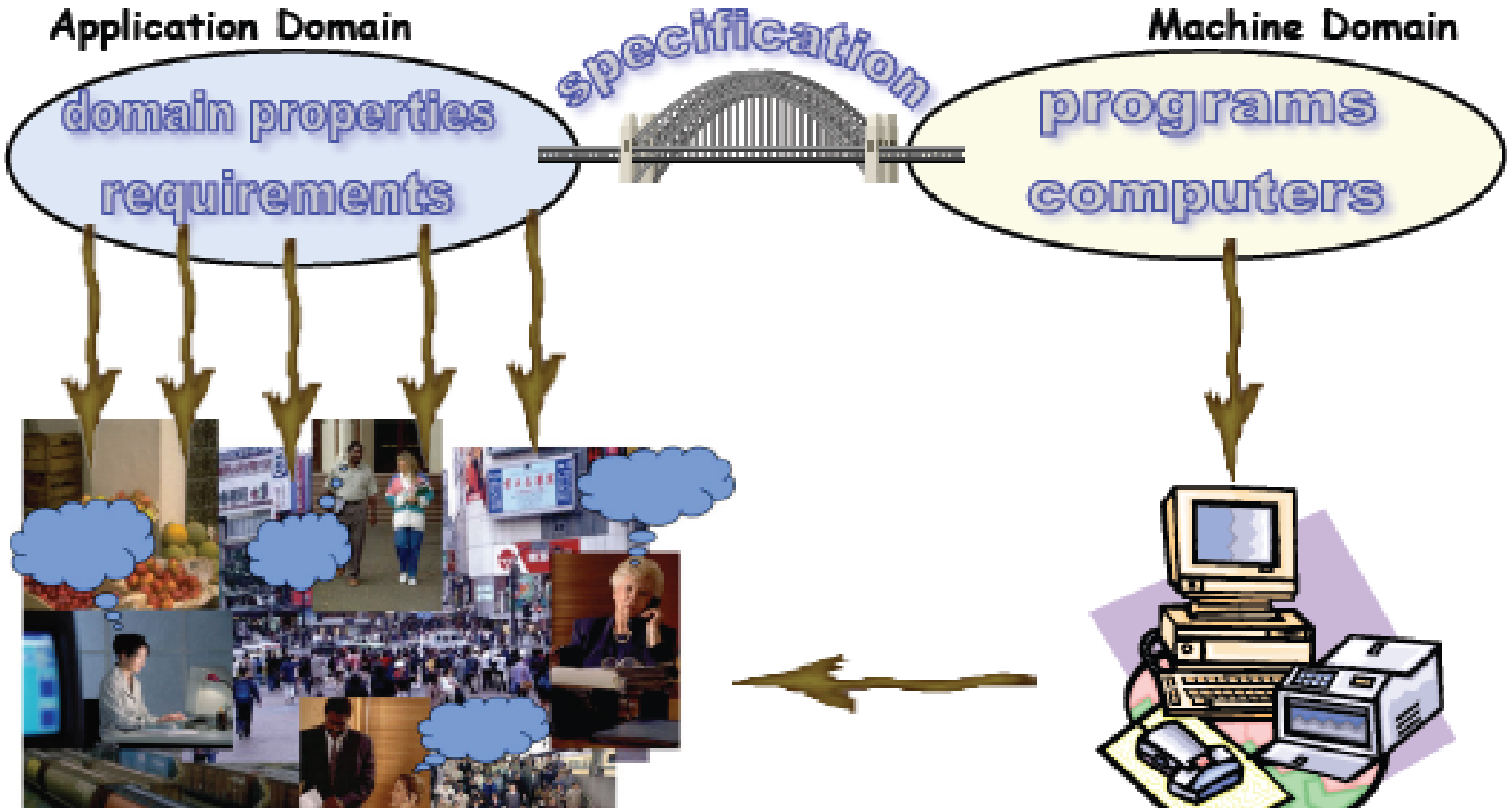
Phân tích yêu cầu phần mềm

Lecture 01 – Công nghệ yêu cầu
Chất lượng = Đáp ứng mục tiêu

- **Công nghệ phần mềm có mặt khắp mọi nơi**
  - **Tác động rất gần đến tất cả các khía cạnh trong cuộc sống**
  - **Nhưng các kinh nghiệm của chúng ta trong kỹ thuật phần mềm thì thường gặp hạn chế**
- **Phần mềm được thiết kế nhằm một mục đích nào đó**
  - **Nếu nó không thực hiện tốt thì hoặc là :**
    - **…người thiết kế không có sự thấu hiểu một cách đầy đủ mục đích**
    - **…hoặc chúng ta đang sử dụng phần mềm cho mục đích khác với dự định ban đầu**
  - **Phân tích yêu cầu nhằm xác định chính xác mục đích này**
  - **Việc hiểu không đầy đủ về mục đích dẫn đến chất lượng phần mềm kém**
- **Mục đích được tìm thấy từ các hoạt động của con người**
  - **E.g. Mục đích của hệ thống ngân hàng đến từ các hoạt động kinh doanh của ngân hàng và nhu cầu từ những khách hàng của họ (e.g. ATM, …)**
  - **Mục đích thường phức tạp**

Thách thức nằm ở đâu ?



Hệ thống nào thì "mềm"?

- **Các thành phần phần mềm cùng loại**
  - **E.g. Các chức năng lõi trong hệ điều hành, dịch vụ mạng, tầng trung gian (middleware), …**
  - **Có quan hệ về mặt chức năng ổn định, xác định bởi các giao diện kỹ thuật**
  - **Nhưng chú ý rằng những hệ thống này vẫn chịu tác động bởi hoạt động của con người ➢ E.g. khái niệm của một 'file', một 'URL', etc.**
- **Các hệ thống quản lý (Control Systems)**
  - **E.g. điều hành quy trình bay, điều hành tiến trình công nghiệp, …**
  - **Hầu hết yêu cầu được xác định bởi các quy trình tự nhiên để điều hành**
  - **Nhưng chú ý rằng các cách thức giao tiếp thì thường mang tính quyết định**
    - **E.g. các tai nạn phát sinh khi hệ thống không ứng xử theo cách thức mong đợi (Tàu vũ trụ Arian 5 - France)**
- **Các hệ thống thông tin (Information Systems)**
  - **E.g. tự động hóa văn phòng, phần mềm nhóm (groupware), web services, phần mềm hỗ trợ kinh doanh,…**
  - **Các hệ thống này không thể tách riêng khỏi các hoạt động mà chúng hỗ trợ**
  - **Thiết kế của phần mềm kế thừa trên thiết kế của hoạt động con người**
    - **Phần mềm và hoạt động con người đồng thiết lập**

Định nghĩa RE (Requirements Engineering)

Requirements Engineering (RE) là một **tập các hoạt động** liên quan tới **việc xác định và truyền đạt mục tiêu** của một hệ thống phần mềm chuyên nghiệp, trong **lĩnh vực** mà chúng được sử dụng. Ở đây, các hoạt động RE như là cầu nối giữa **các nhu cầu trong thực tế** của người dùng, khách hàng, và những **ứng viên** khác có ảnh hưởng đến một hệ thống phần mềm, và **những khả năng và cơ hội** được tạo ra bởi những kỹ thuật phần mềm chuyên nghiệp

Không phải một thời kỳ hay một giai đoạn !

Truyền đạt rất quan trọng khi phân tích

Chất lượng nghĩa là đáp ứng mục tiêu. Không thể nói điều gì về chất lượng trừ khi bạn hiểu rõ mục tiêu

Cần nhận dạng tất cả các đối tác – không chỉ là người dùng và khach hàng !

Người thiết kế cần biết hệ thống sẽ được sử dụng ở đâu và như thế nào?

Yêu cầu là một phần của … nhu cầu là gì ???

Và một phần của … nó thực hiện được gì ???

Hậu quả của sai sót

Giá để sửa chữa lỗi

- Một tiến trình phát triển phần mềm điển hình bao gồm:

  Phân tích yêu cầu ⇨ Thiết kế phần mềm ⇨ Lập trình ⇨ Kiểm thử sự phát triển ⇨ Kiểm thử sự chấp thuận ⇨ Vận hành

- Giá sửa lỗi ngày càng tăng vào thời điểm phát hiện chúng trong tiến trình
  - E.g. Một lỗi về phân tích yêu cầu được tìm thấy phải trả giá 100 lần cao hơn lỗi chương trình.

Nguyên nhân dự án thất bại

- Thống kê các dự án phần mềm US của nhóm Standish:

	1994	1998
Successful	16%	26%
Challenged	53%	46%
Cancelled	31%	28%

Top 3 success factors:
1) User involvement
2) Executive management support
3) Clear statement of requirements

Top 3 factors leading to failure:
1) Lack of user input
2) Incomplete requirements & specs
3) Changing requirements & specs

Hậu quả của sai sót

- **Nguyên nhân dự án thất bại**

  ↳ **Standish Group (US Software) khảo sát 350 công ty với hơn 8000 dự án phần mềm.**

1. **Yêu cầu không hoàn chỉnh (13.1%)**
2. **Thiếu sự hợp tác người dùng (12.4%)**
3. Thiếu tài nguyên (10.6%)
4. Mong muốn phi thực tế (9.9%)
5. Thiếu hỗ trợ pháp lý (9.3%)
6. Thay đổi yêu cầu và đặc tả (8.7%)
7. Thiếu hoạch định (8.1%)
8. Hệ thống không cần đến nữa (7.5%)

Hậu quả của sai sót

Kiến nghị

Lỗi yêu cầu (requirements errors) có thể phải trả giá đắt nếu chúng không được phát hiện và sửa chữa sớm trong tiến trình phát triển.

Báo cáo của Boehm và Papaccio (1988) cho thấy ước lượng giá trị tiêu tốn cho việc phát hiện lỗi ở các giai đoạn của một tiến trình phát triển phần mềm như sau :

Phân tích yêu cầu **(1$)** ⇒ Thiết kế **(5$)** ⇒ Lập trình **(10$)** ⇒ Kiểm thử **(20$)** ⇒ Triển khai hệ thống **(>200$)**

→ Cần dành thời gian để tìm hiểu kỹ vấn đề trong lĩnh vực của chúng và thu thập yêu cầu thật chính xác trong giai đoạn đầu tiên.

Mục tiêu của Phân tích yêu cầu ?

- **Điểm bắt đầu**
  - **Tập trung chú ý rằng có một "vấn đề" cần được giải quyết**
    - **e.g. không bằng lòng với trạng thái hiện tại của công việc**
    - **e.g. một cơ hội kinh doanh mới**
    - **e.g. một cơ hội để tiết kiệm chi phí, thời gian, tài nguyên sử dụng, etc.**
  - **Nhà phân tích yêu cầu là một tác nhân của sự thay đổi**

Phân tích yêu cầu cần đạt được gì?

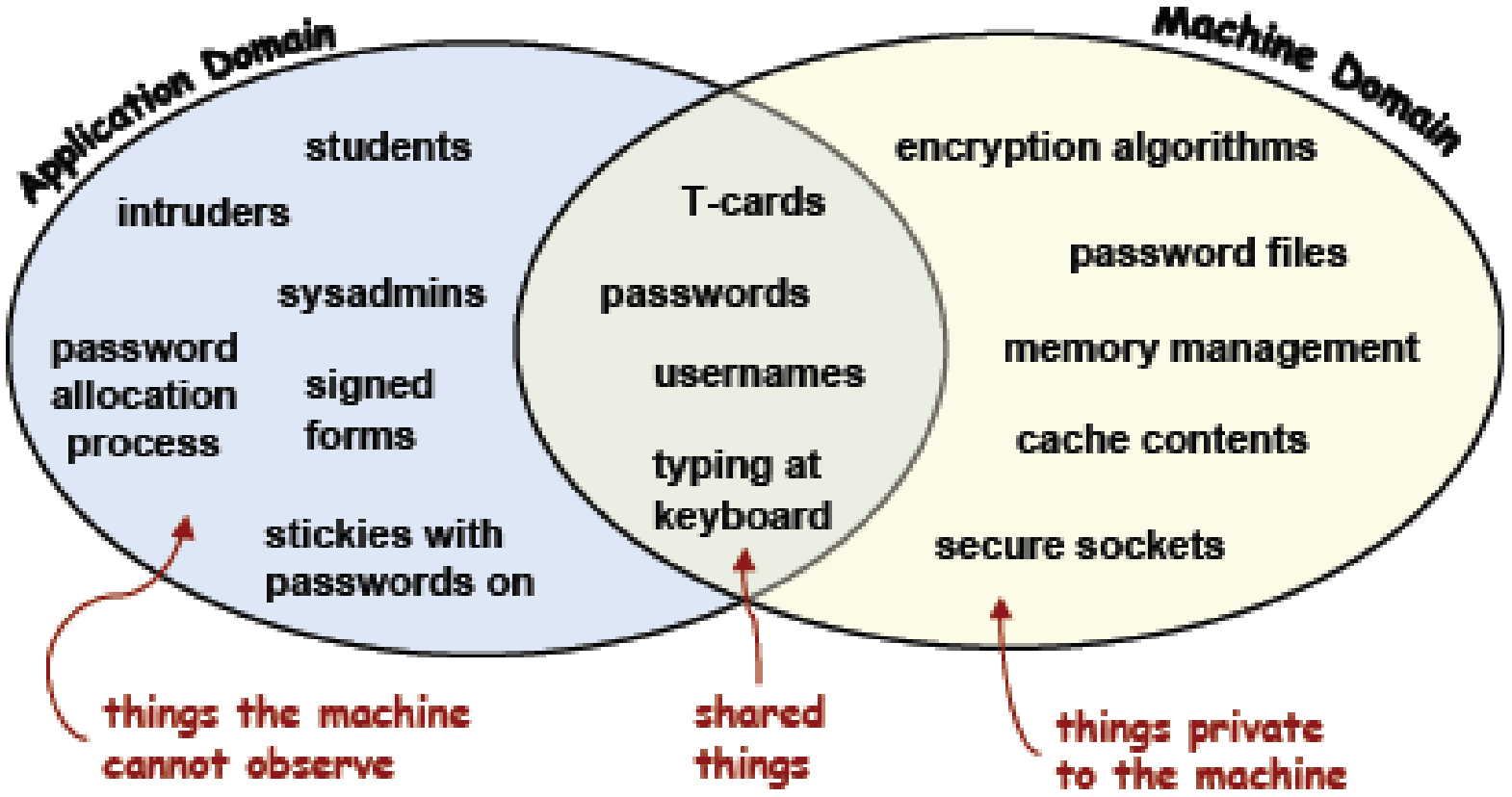
- **Định nghĩa được "vấn đề"** :
  - **(Which) Vấn đề nào cần được giải quyết ? (Xác định ranh giới vấn đề - Boundaries)**
  - **(Where) Vấn đề ở đâu ? (Hiểu ngữ cảnh/ phạm vi vấn đề - Context/Problem Domain)**
  - **(Whose) Vấn đề của ai? (Định nghĩa Đối tác - Stakeholders)**
  - **(Why) Tại sao cần giải quyết? (Định nghĩa Mục tiêu đối tác – 'stakeholders' Goals)**
  - **(How) Hệ thống phần mềm sẽ hỗ trợ như thế nào? (Thu thập Kịch bản - Scenarios)**
  - **(When) Khi nào cần phải giải quyết ? (Định nghĩa các ràng buộc phát triển - Development Constraints)**
  - **(What) Điều gì ngăn chặn việc giải quyết chúng? (Định nghĩa tính khả thi và độ rủi ro - Feasibility and Risk)**
- **Là chuyên gia trong phạm vi của vấn đề.**

Một số khảo sát về RE

- **RE không cần thiết phải theo một tiến trình tuần tự:**
  - **Không cần phải viết mô tả vấn đề trước mô tả giải pháp**
  - **Viết lại một mô tả vấn đề có thể giúp ích ở các giai đoạn phát triển**
  - **Các hoạt động RE tiếp tục xuyên suốt tiến trình phát triển**
- **Khai báo vấn đề sẽ không hoàn hảo**
  - **Các mô hình RE thì chỉ gần đúng với thực tế**
  - **Sẽ chứa sự thiếu chính xác và không nhất quán**
  - **Sẽ bỏ sót một số thông tin.**
  - **Các nhà phân tích luôn làm giảm bớt những rủi ro sẽ có trong vấn đề thực…**
- **Việc hoàn chỉnh một sự đặc tả có thể không mang lại lợi nhuận**
  - **Phân tích yêu cầu có giá của nó**
  - **Đối với những dự án khác nhau, cân bằng lợi nhuận cũng khác nhau**
- **Khai báo vấn đề không khi nào được xem là cố định**
  - **Thay đổi thì chắc chắn sẽ xảy ra, và vì thế phải dự kiến (E.g…) trước**
  - **Đó sẽ là một cách để kết hợp chặt chẽ các thay đổi một cách định kỳ**

Một vấn đề được mô tả

- E.g. "Ngăn chặn việc truy cập trái phép từ các máy tính"



Yêu cầu là gì ?

- **Đặc tính lĩnh vực (Domain Properties D)**

⇘ Những thứ có thật trong lĩnh vực ứng dụng cho dù chúng ta có thiết kế hệ thống dự định hay không

- **Các yêu cầu (Requirement R)**

⇘ Những thứ trong lĩnh vực ứng dụng mà chúng ta mong muốn trở thành hiện thực bằng cách thực hiện hệ thống dự định

➢ Rất nhiều trong chúng bao gồm các hiện tượng mà máy tính không thể truy cập được.

- **Sự đặc tả (Specification S)**

⇘ Là sự mô tả các hành vi mà chương trình phải làm để đáp ứng các yêu cầu

➢ Có thể chỉ được viết trong thuật ngữ của sự chia sẻ các hiện tượng!

Đáp ứng với mục tiêu ?

- **Hai tiêu chuẩn kiểm tra tính chính xác (verification)**

Chương trình (Program) thực hiện trên một máy tính (Computer) cụ thể đáp ứng với đặc tả (Specification)

Đặc tả (Specification) được cho trong thuộc tính của lĩnh vực (Domain properties) thỏa mãn các yêu cầu (Requirements)

- **Hai tiêu chuẩn kiểm chứng sự hoàn thiện (validation)**

Chúng ta đã xem xét (và hiểu) tất cả các yêu cầu (Requirements) quan trọng?

Chúng ta đã xem xét (và hiểu) tất cả các thuộc tính lĩnh vực(Domain properties) liên quan?

Ví dụ

- **Requirement R:**
  - **"Phản lực chỉ có thể xảy ra khi máy bay đang chạy trên đường băng"**
- **Domain Properties D:**
  - **Xung lực bánh xe xảy ra khi và chỉ khi các bánh xe bật ra**
  - **Các bánh xe bật ra khi và chỉ khi nó chạy trên đường băng**
- **Specification S:**
  - **Phản lực có thể xảy ra khi và chỉ khi có xung lực bánh xe**

Kiểm tra

Phần mềm cho máy bay, P, thực thi trên máy tính trong buồng lái của máy bay, C, có hoàn toàn chính xác như đặc tả, S?

S, trong ngữ cảnh của giả thuyết D, có đáp ứng R?

Kiểm chứng

Giả thuyết của chúng ta, D, về lĩnh vực có thật chính xác? Có thiếu gì không?

Yêu cầu, R, có thật sự cần thiết? Có thiếu gì không?

Một ví dụ khác

- **Requirement R:**
  - **"Cơ sở dữ liệu chỉ có thể được truy cập bởi những người có quyền"**
- **Domain Properties D:**
  - **Những người có quyền thì có passwords**
  - **Passwords không bao giờ được chia sẻ với những người không có quyền**
- **Specification S:**
  - **Truy cập vào CSDL chỉ được chấp nhận sau khi người dùng gõ vào một password được cấp**
- **S + D dẫn đến R**
  - **Nhưng có liệu rằng giả thuyết về lĩnh vực là sai?**

Mô hPhần mềm thì khác biệt gì ?

- **Phần mềm thì khác!**
  - **Phần mềm thì vô hình, mơ hồ, trừu tượng**
    - **mục đích của nó là cấu hình một số phần cứng để làm những thứ hữu ích**
  - **Không có quy luật tự nhiên nào bên trong các hành vi phần mềm**
  - **Không có các ràng buộc tự nhiên nào trong các phần mềm phức tạp**
  - **Phần mềm không khi nào mệt mỏi**
    - **…các độ đo truyền thống đáng tin không được áp dụng**
  - **Phần mềm hoàn toàn có thể thực hiện một công việc lặp đi lặp lại**
    - **…không tạo ra sự thay đổi**

Quản lý dự án

- Một nhà quản lý dự án có thể kiểm soát 4 thứ:
  - Tài nguyên (có thể tăng thêm tiền, tiện ích, nhân lực)
  - Thời gian (có thể tăng thời gian, trì hoãn thời hạn, etc.)
  - Sản phẩm (có thể giảm chức năng - e.g. các yêu cầu quá rắc rối)
  - Rủi ro (có thể quyết định các rủi ro nào chấp nhận được)
- Để thực hiện điều này, nhà quản lý cần theo dõi:
  - Công sức – Cần tốn công sức nhiều thế nào? Tiêu hao bao nhiêu?
  - Thời gian – Lịch biểu được mong đợi ra sao? Còn bao lâu nữa ?
  - Kích cỡ – Kế hoạch vấn đề lớn như thế nào? Phải thiết kế ra sao?
  - Hạn chế – Đã tạo ra bao nhiêu lỗi ? Bao nhiêu lần phát hiện lỗi?
    - Và các lỗi này ảnh hưởng như thế nào đến chất lượng?
- Khởi đầu, một nhà quản lý cần có sự đánh giá đúng

  … và điều đó chỉ có thể có từ sự phân tích thấu đáo vấn đề.

Bạn không thể kiểm soát được cái mà bạn không thể đo lường !

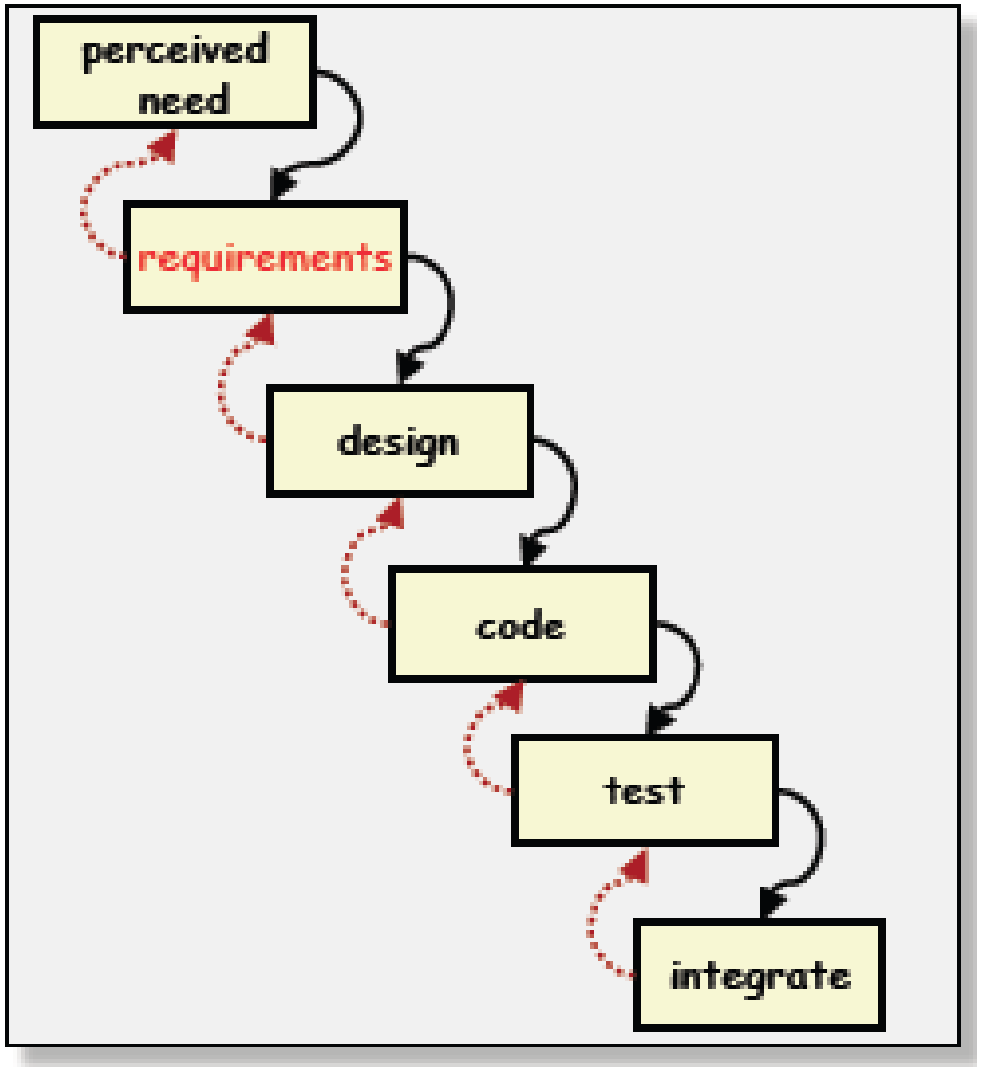
Các kiểu dự án

- **Các lý do khởi đầu cho một dự án phát triển phần mềm**
  - Hướng vấn đề (Problem-driven): sự cạnh tranh, sự khủng hoảng,…
  - Hướng thay đổi (Change-driven): nhu cầu mới, sự lớn mạnh, thay đổi doanh nghiệp hoặc môi trường,…
  - Hướng cơ hội (Opportunity-driven): bùng nổ một kỹ thuật mới,…
  - Hướng kế thừa (Legacy-driven): một phần của kế hoạch trước đó, công việc chưa hoàn thành, …
  - Green field
- **Các kiểu quan hệ với khách hàng:**
  - Customer-specific – một khách hàng với vấn đề cụ thể
    - Có thể là một công ty khác, với hợp đồng thỏa thuận
    - Có thể là một bộ phận trong cùng công ty
  - Market-based – hệ thống bán ra thị trường
    - Trong một số trường hợp, sản phẩm phải sinh ra khách hàng
    - Đội ngũ tiếp thị phải hành động như những người thay thế khách hàng
  - Community-based – dự định sẽ như một tiện ích chung cho cộng đồng
    - E.g. công cụ nguồn mở (open_ source), các công cụ cho nghiên cứu khoa học
    - Khách hàng tài trợ (nếu nhà tài trợ không chiếm giữ kết quả)
  - Hybrid (kết hợp những kiểu trên)

Chu kỳ sống của một dự án phần mềm

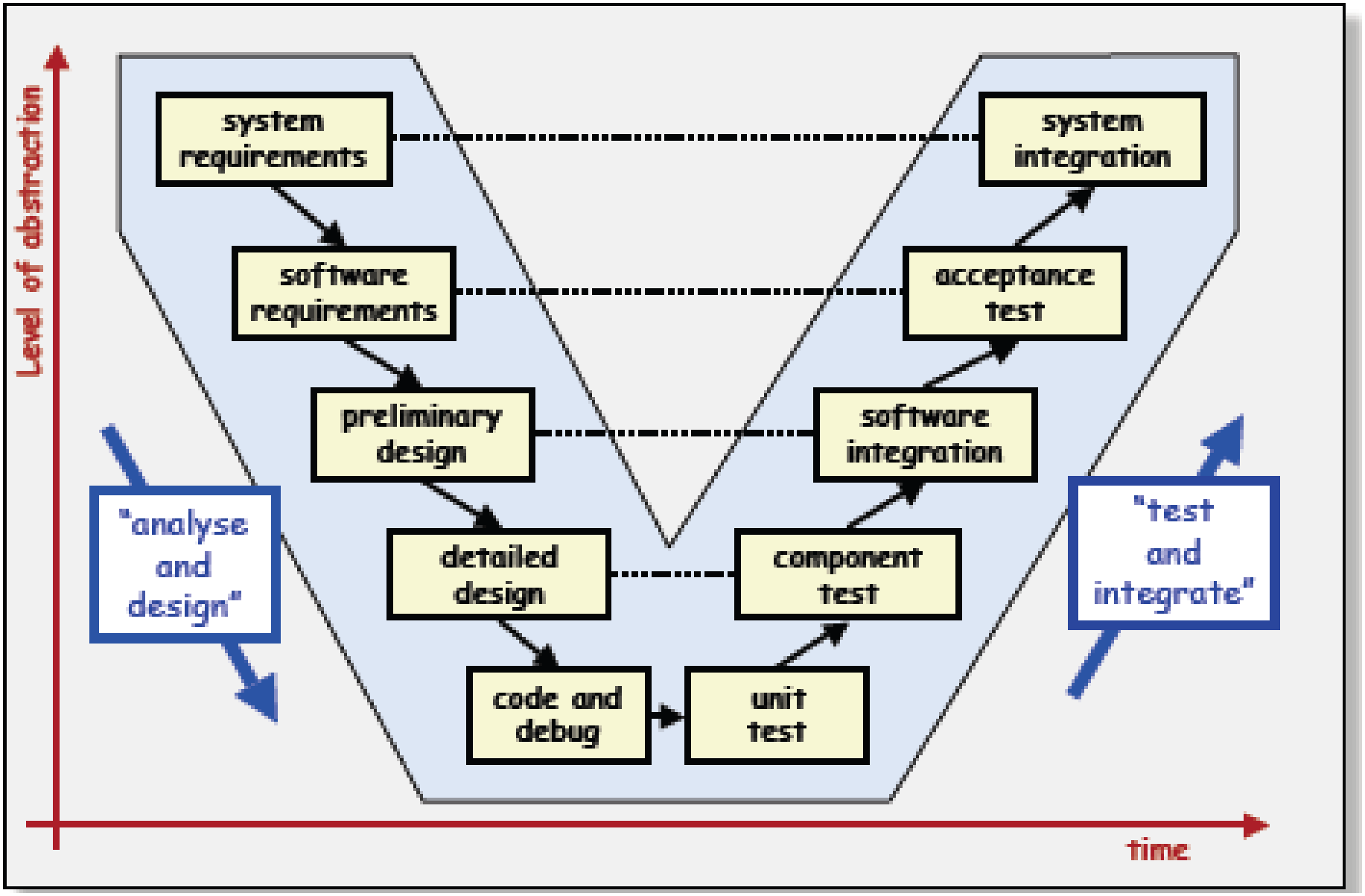
- **Các mô hình chu kỳ sống**
  - **Rất hữu ích để so sánh các dự án trong ngữ cảnh chung**
  - **Không đủ chi tiết cho việc hoạch định dự án**
- **Các ví dụ:**
  - **Các mô hình tuần tự: Waterfall, V model**
  - **Lập bản mẫu nhanh (Rapid Prototyping)**
  - **Các mô hình giai đoạn: Incremental, Evolutionary**
  - **Các mô hình vòng lặp: Spiral**
  - **Các mô hình linh hoạt (Agile Models): eXtreme Programming**
- **Sự so sánh: Process Models**
  - **Dùng cho việc nắm vững và cải tiến tiến trình phát triển phần mềm**

Mô hình thác nước (Waterfall Model)

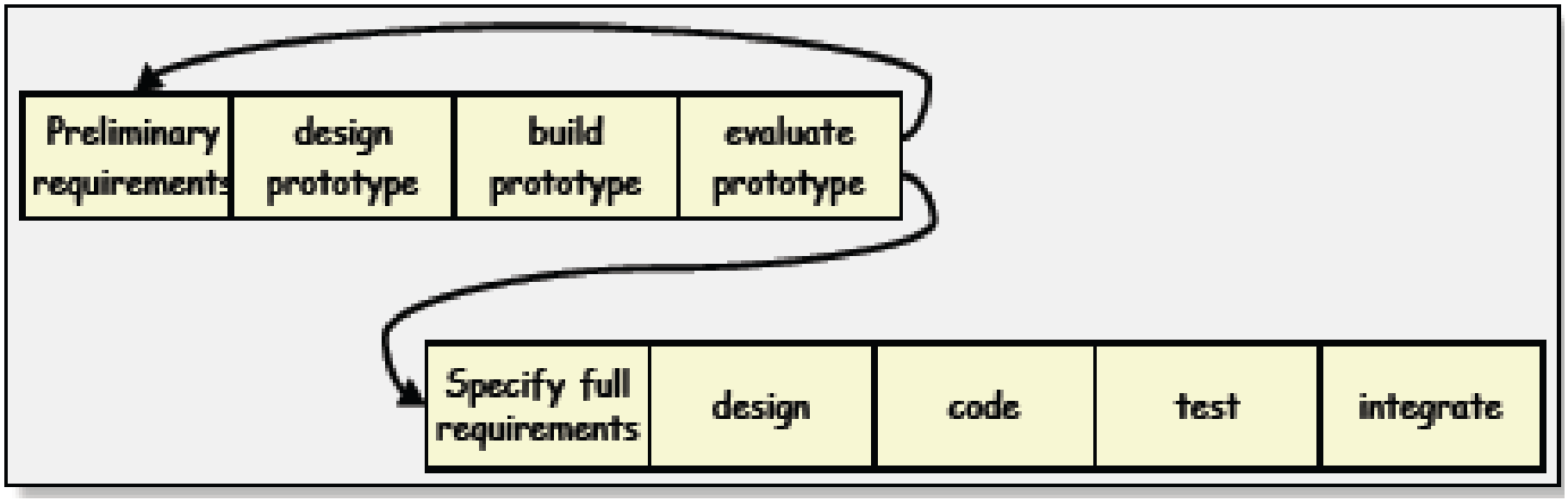


- **Quan điểm phát triển:**
  - **Là một tiến trình của sự tinh chế theo bậc thang**
  - **Quan điểm quản trị cấp cao ở cấp độ lớn**
- **Các vấn đề:**
  - **Cách nhìn tĩnh với các yêu cầu – lờ đi khả năng biến đổi**
  - **Thiếu sự liên quan của người dùng khi đặc tả được viết**
  - **Có tách biệt không thực tế của đặc tả từ thiết kế**
  - **Không hỗ trợ cho việc lập bản mẫu, tái sử dụng, etc**

Mô hình V (V - Model)

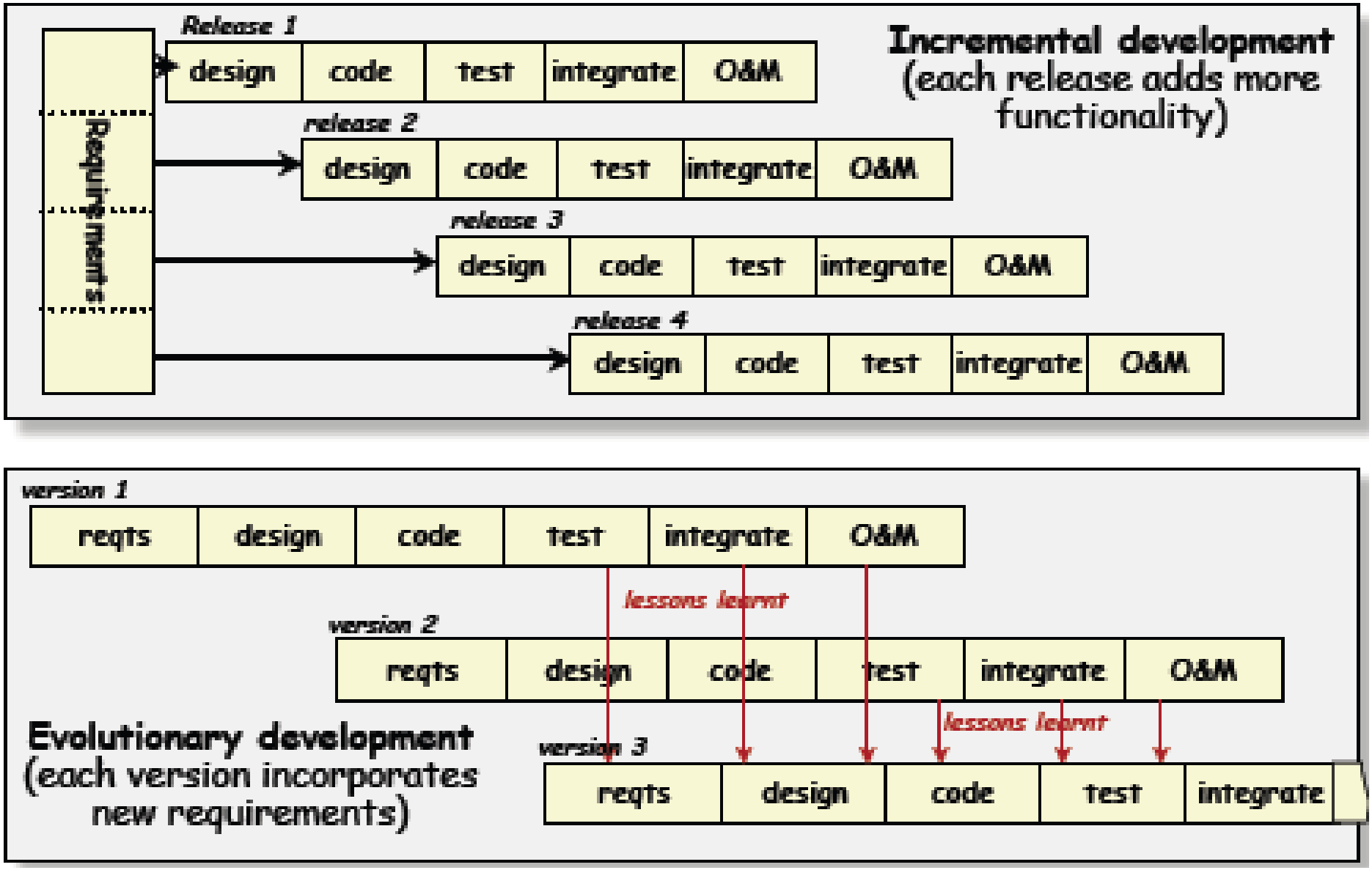


Lập bản mẫu nhanh

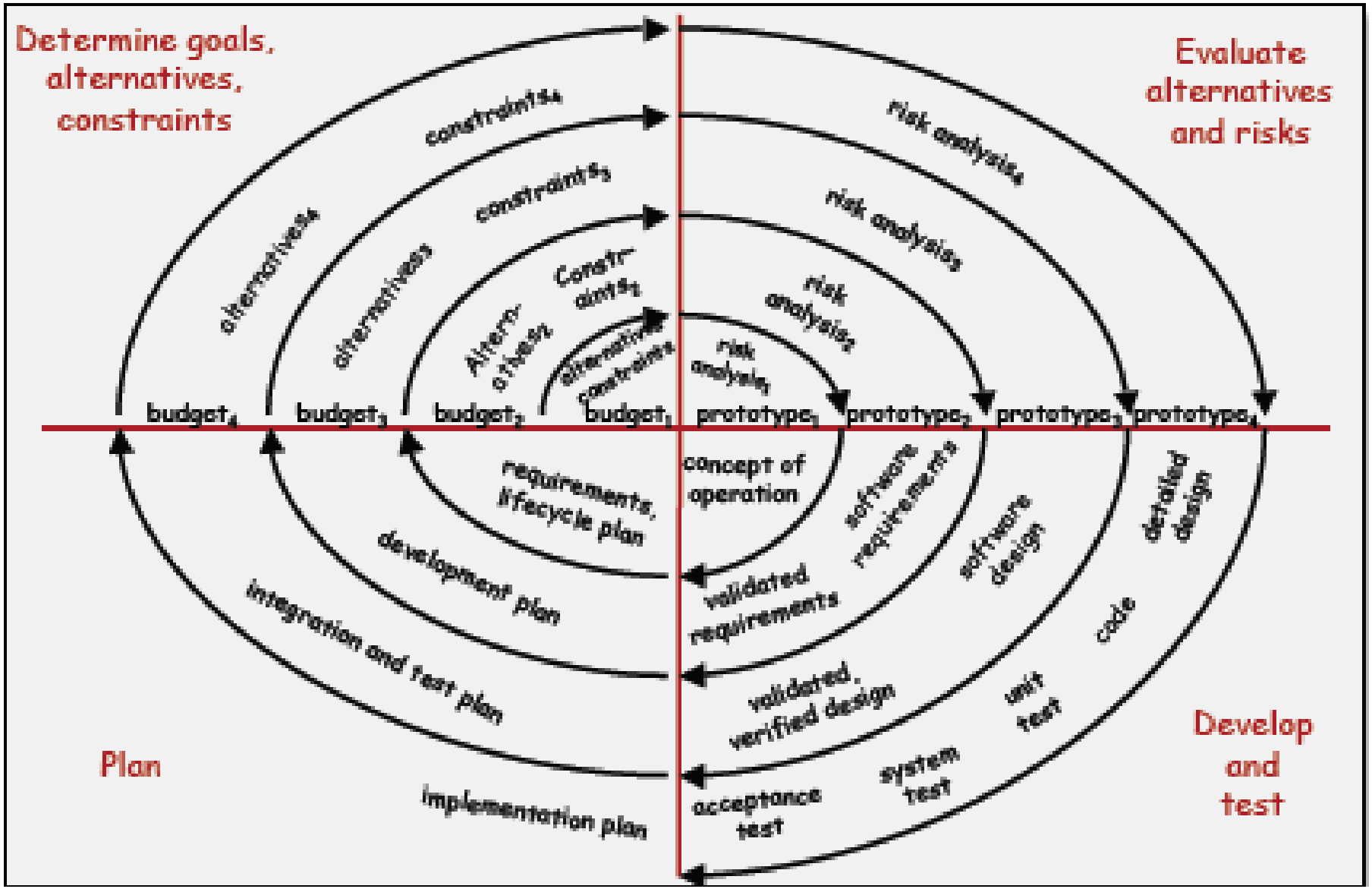


- **Lập bản mẫu thì được dùng cho:**
  - **Hiểu các yêu cầu của giao diện người dùng**
  - **Xem xét các đặc tính của hướng dự định thiết kế**
  - **Khảo sát các quy tắc thực thi của hệ thống**
- **Các vấn đề:**
  - **Những người dùng xem bản mẫu như giải pháp**
  - **Một bản mẫu chỉ là một đặc tả không hoàn chỉnh**

Các mô hình giai đoạn chu kỳ sống



Mô hình xoắn ốc (The Spiral Model)



Các mô hình linh hoạt (Agile Models)

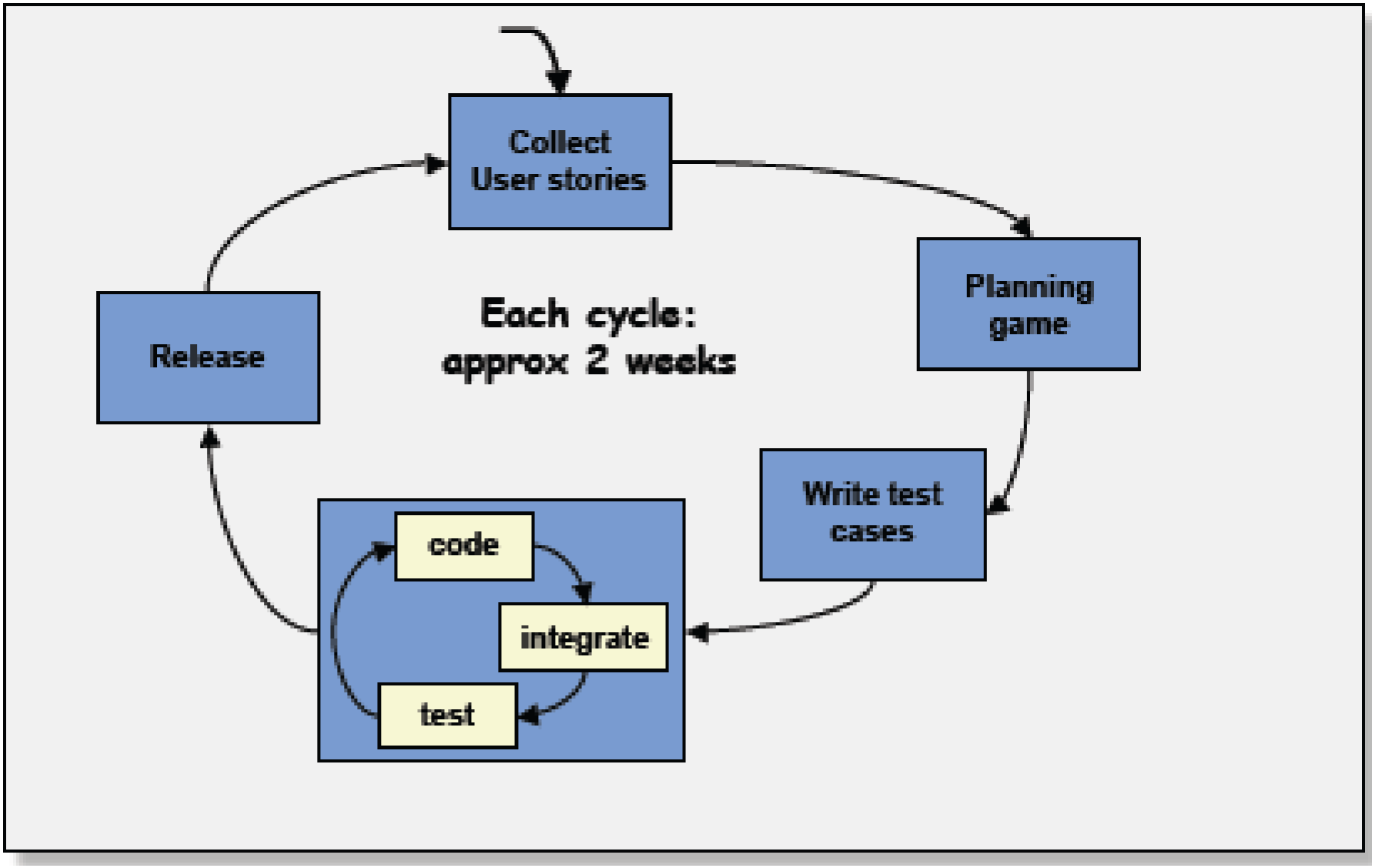
- **Lập luận cơ sở**
  - Giảm rào cản về truyền thông
    - Người lập trình giao tiếp với khách hàng
  - Giảm tiếp cận nặng nề với tài liệu
    - Việc lập tài liệu thì tốn kém và giới hạn sử dụng
  - Có niềm tin giữa con người
    - Không cần thiết phải có các mô hình xử lý thật thu hút để thuyết phục cái sẽ làm!
  - Đáp ứng được cho khách hàng
    - Hơn là tập trung vào việc ký hợp đồng

- **Điểm yếu**
  - Tin tưởng vào trí nhớ của người lập trình
    - Mã lệnh có thể khó bảo trì
  - Tin tưởng vào truyền thông bằng miệng
    - Có thể thiếu rõ ràng
  - Chấp nhận duy nhất khách hàng đại diện
    - Các quan điểm khác nhau thì không thể đưa ra
  - Kế hoạch chỉ lập trong thời gian ngắn
    - Không có tầm nhìn xa

E.g. Lập trình cực độ (XP - Extreme Programming)

- Thay vì viết đặc tả yêu cầu, thì sử dụng:
  - User story cards (Bản chức năng người dùng)
  - Khách hàng trực diện
- Lập trình cặp đôi (Pair Programming)
- Phát hành nhặt
  - E.g. mỗi 3 tuần
- Trò chơi kế hoạch (Planning Game)
  - Chọn lựa và đánh giá các user story cards vào lúc bắt đầu mỗi đợt phát hành
- Viết bản kiểm thử trước viết code
- Mã lệnh chương trình được thiết kế lập tức
- Tương tác liên tục
  - Tích hợp và kiểm thử mã lệnh vài lần trong một ngày

Lập trình cực độ XP (eXtreme Programing)



Kết luận

- Học phần này bao gồm hầu hết các công nghệ về yêu cầu:
  - Phân tích hiện trạng vấn đề
  - Khảo sát hoạt động con người
  - Hình thức hóa các yêu cầu để giải pháp phần mềm có thể được thiết kế

- Học phần này thì khác với hầu hết các học phần CS khác
  - Nó không phải về cách giải quyết vấn đề dùng máy tính như thế nào
  - Nó là việc xác định các vấn đề cần giải quyết như thế nào
  - Nội dung học phần là các hoạt động của con người:
    - Làm sao để thấu hiểu chúng
    - Làm sao để dùng các kỹ thuật phần mềm hỗ trợ chúng

Lecture 2:
Quy trình công nghệ yêu cầu
(RE - The requirements engineering)

- **Khái niệm**
  - **Quy trình dùng để khảo sát, phân tích và kiểm chứng tính hợp lệ của các yêu cầu hệ thống**
  - **Quy trình là một tập các hoạt động nhằm dẫn đến việc phát sinh định nghĩa và đặc tả yêu cầu.**

Các đặc tính chung

- **Quy trình RE có nhiều dạng khác nhau, phụ thuộc vào lĩnh vực ứng dụng, các nhân tố liên quan và tổ chức phát triển yêu cầu.**

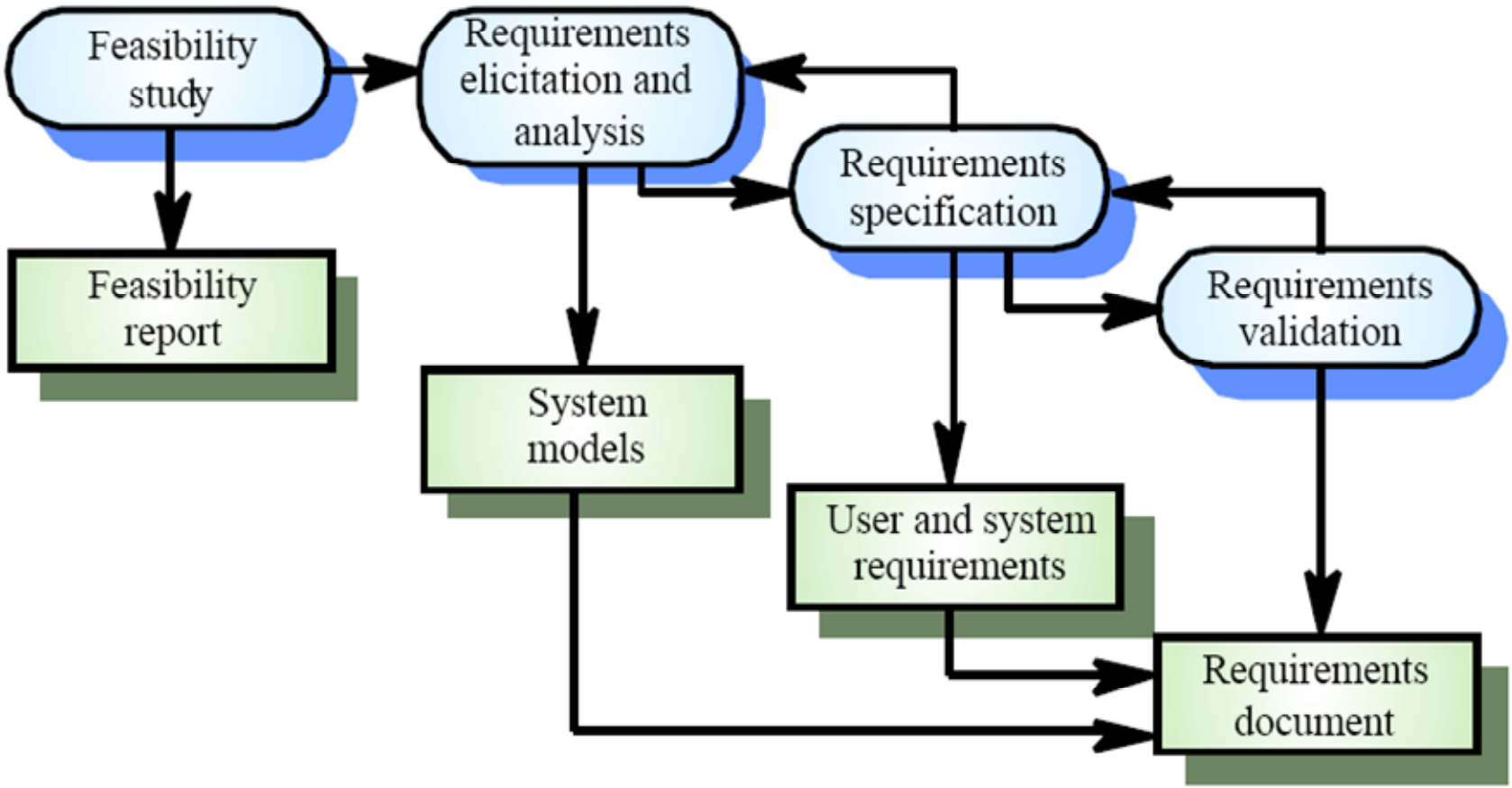
- **Tuy nhiên, có một số đặc tính chung cho các quy trình là :**

  - **Thu thập yêu cầu (Requirements elicitation)**
  - **Phân tích yêu cầu (Requirements analysis)**
  - **Kiểm chứng yêu cầu (Requirements validation)**
  - **Quản trị yêu cầu (Requirements management)**

Các nội dung chính

- **Nghiên cứu khả thi** (Feasibility studies)
- **Thu thập yêu cầu và phân tích** (Requirements elicitation and analysis)
- **Kiểm chứng yêu cầu hợp lệ** (Requirements validation)
- **Quản trị yêu cầu** (Requirements management)

Các bước trong quy trình



Nghiên cứu khả thi

- **Thực hiện ước lượng nhằm đánh giá sự đáp ứng cho yêu cầu:**
  - **Kỹ thuật phần cứng**
  - **Kỹ thuật phần mềm**
- **Nghiên cứu khả thi quyết định hệ thống**
  - **Có giá trị hiệu quả về kinh doanh**
  - **Có thể phát triển với những ràng buộc ngân sách hiện có**
- **Phải rẻ và nhanh chóng**
- **Kết quả : Báo cáo khả thi** (Feasibility Report)
  - **Quyết định điều gì là quan trọng với các lý giải chi tiết**
  - **Bản báo cáo về tính khả thi của hệ thống**
  - **Tài liệu đặc tả yêu cầu người dùng**

Phân tích làm rõ yêu cầu

- **Quá trình đưa ra các yêu cầu hệ thống**
  - **Khảo sát hệ thống hiện tại**
  - **Thảo luận với người dùng và các nhà trung gian tiềm năng**
  - **Phân tích công việc**
- **Có thể phát triển 1 hoặc nhiều mô hình hệ thống khác nhau**
  - **Giúp nhà phân tích hiểu rõ hệ thống để đặc tả**
- **Bản mẫu có thể lập để hiểu rõ các yêu cầu**

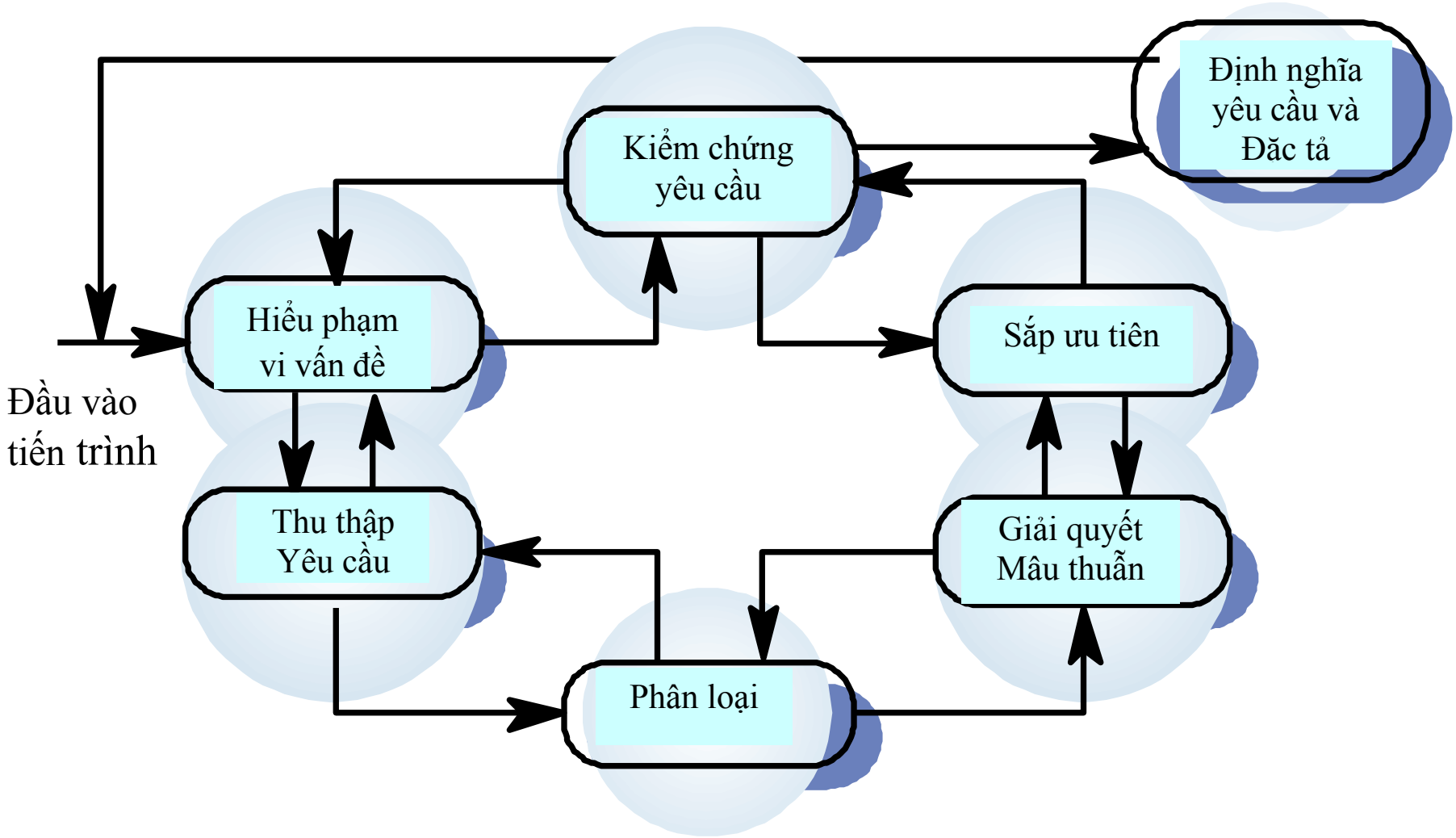
N

g

h

i

Tiến trình phân tích làm rõ yêu cầu



Các hoạt động trong tiến trình

- Hiểu phạm vi vấn đề (Domain understanding)
- Thu thập yêu cầu (Requirements collection)
- Phân loại (Classification)
- Giải quyết mâu thuẫn (Conflict resolution)
- Sắp ưu tiên (Prioritisation)
- Kiểm tra yêu cầu (Requirements checking)

Xác định yêu cầu

- Là hoạt động chuyển thông tin phát sinh trong suốt tiến trình phân tích thành tài liệu định nghĩa tập hợp các yêu cầu
- Phản ánh chính xác điều mà người dùng muốn
- Tài liệu phải được viết để hệ thống sẽ được hiểu bởi
  - Người dùng cuối
  - Những khách hàng của hệ thống.

Đặc tả yêu cầu

- **Bản mô tả các yêu cầu hệ thống được thiết lập như cơ sở của hợp đồng giữa khách hàng và nhà phát triển phần mềm**
  - **Mô tả thật chi tiết về yêu cầu người dùng và yêu cầu hệ thống**
    - **hữu ích cho thiết kế**
  - **Mô tả chính xác để nắm bắt đúng vấn đề**
- **Việc lập tài liệu này được thực hiện song song cùng với một số các thiết kế cấp cao khác.**
- **Lỗi trong định nghĩa yêu cầu cần được xem xét kỹ lưỡng.**
  - **Nó phải được sửa chữa theo đúng vấn đề này.**

Quản lý yêu cầu

- **Quản lý yêu cầu là tiến trình quản lý sự thay đổi của yêu cầu trong suốt quy trình công nghệ yêu cầu và phát triển hệ thống**

- **Yêu cầu thì chắc hẳn là sẽ không hoàn thiện và không nhất quán**

  - **Các yêu cầu mới thì liên tục phát sinh trong suốt tiến trình khi nhu cầu công việc thay đổi và có sự hiểu rõ hơn về hệ thống đang phát triển**
  - **Các quan điểm khác nhau có các yêu cầu khác nhau và điều này thường làm phát sinh mâu thuẫn**

Kết luận

- **Các hoạt động trong quy trình công nghệ yêu cầu thì không đơn giản để thực hiện một cách tuần tự mà chúng phải lặp đi lặp lại.**
  - **Phân tích yêu cầu vẫn tiếp tục trong suốt quá trình định nghĩa và đặc tả**
  - **Các yêu cầu mới vẫn còn tiếp tục phát sinh trong suốt tiến trình**
- **Tài liệu yêu cầu phải thay đổi thường xuyên và được đặt dưới sự kiểm soát của một hệ thống quản lý cấu hình**

Lecture 3:
Nghiên cứu khả thi (Feasibility Study)

- **Nghiên cứu khả thi là gì?**
  - **Nghiên cứu điều gì và kết quả ra sao ?**
- **Các dạng đặc tính khả thi cần khảo sát**
  - **Kỹ thuật**
  - **Kinh tế**
  - **Lịch biểu**
  - **Vận hành**
- **Mức độ lợi nhuận và chi phí**
  - **Phân tích lợi tức**
  - **Phân tích giá trị thực có**
  - **Phân tích lợi nhuận trên vốn đầu tư**
- **So sánh các sự lựa chọn**

Tại sao cần nghiên cứu khả thi

Mục tiêu:

- Chỉ rõ việc phát triển dự án :
- Đề nghị các giải pháp có thể thay đổi.
- Cung cấp cho nhà quản trị đủ thông tin để biết:
  - Dự án có thể thực hiện được
  - Sản phẩm sau cùng mang đến lợi ích cho người dùng
  - Cần thay đổi những gì
  - Có thể thay đổi hoàn thiện không

Hành động của nhà quản trị:

- Sau khi nghiên cứu khả tính, nhà quản trị cần có ngay quyết định "tiếp tục hay không ?"
- Cần xem xét vấn đề trong môi trường chiến lược kinh doanh.

Nội dung nghiên cứu khả thi

- **Những vấn đề cần nghiên cứu trong nghiên cứu khả thi**
  - **Tổ chức của hệ thống hiện hành**
  - **Các vấn đề với hệ thống hiện hành**
  - **Các mục tiêu và những yêu cầu khác đối với hệ thống mới**
  - **Các ràng buộc**
  - **Những lựa chọn có thể**
    - **"Gắn với hệ thống hiện hành" luôn luôn là một chọn lựa**
    - **Các quy trình công việc khác nhau cho việc giải quyết vấn đề**
    - **Các cấp độ/kiểu tin học hóa khác nhau cho giải pháp**
  - **Các thuận lợi và bất lợi của mỗi lựa chọn**
- **Những vấn đề để kết luận:**
  - **Tính khả thi của dự án**
  - **Các lựa chọn tốt hơn**

Thực hiện nghiên cứu khả thi

- Dựa trên các thông tin đã có sẵn (yêu cầu gì), các thông tin thu thập được và viết bản báo cáo.

- Một số câu hỏi liên quan
  - Liệu hệ thống sẽ không cài đặt được gì ?
  - Quy trình hiện tại cho vấn đề?
  - Hệ thống đưa ra những hỗ trợ như thế nào ?
  - Vấn đề gì sẽ được tích hợp?
  - Kỹ thuật mới nào là cần thiết ? Cần những kỹ năng gì?
  - Các hoạt động nào cần được hỗ trợ bởi hệ thống dự định ?

Khảo sát hiện trạng

- **Khung "PIECES" (The PIECES" framework)**
  - **Hữu ích cho việc định nghĩa hoạt động của vấn đề cần giải quyết và sự cấp bách của chúng.**
  - **Performance (Độ thực thi)**
  - **Information (Sự truyền thông)**
  - **Economy (Tính kinh tế)**
  - **Control (Sự kiểm soát)**
  - **Efficiency (Tính hiệu quả)**
  - **Services (Các dịch vụ)**

the PIECES Framework

A checklist for identifying problems with an existing information system.

- Performance
  - Throughput
  - Response Time
- Information (and Data)
  - Outputs
    - Lack of any information
    - Lack of necessary information
    - Lack of relevant information
    - Too much information – information overload
    - Information that is not in a useful format
    - Information that is not accurate
    - Information that is difficult to produce
    - Information that is not timely to its subsequent use
  - Inputs
    - Data is not captured
    - Data is not captured in time to be useful
    - Data is not accurately captured – contains errors
    - Data is difficult to capture
    - Data us captured redundantly – same data is captured more than once
    - Too much data is captured
    - Illegal data is captured
  - Stored Data
    - Data is stored redundantly in multiple files and/or databases
    - Stored data is not accurate
    - Data is not secure from accident or vandalism
    - Data is not well organized
    - Data is not flexible – not easy to meet new information needs from stored data
    - Data is not accessible
- Economics
  - Costs
    - Costs are unknown
    - Costs are untraceable
    - Costs are too high
  - Profits
    - New markets can be explored
    - Current marketing can be improved
- Control (and Security)
  - Too little security or control
    - Input data is not adequately edited
    - Crimes (e.g. fraud, embezzlement) are (or can be) committed against the data
    - Ethics are breached on data or information – refers to data or information getting to unauthorized people
    - Redundantly stored data is inconsistent in different files or databases
    - Data privacy regulations or guidelines are being (or can be) violated
    - Processing errors are occurring (either by people, machines, or software)
    - Decision- making errors are occurring
  - Too much control or security
    - Bureaucratic red tape slows the system
    - Controls inconvenience customers or employees
    - Excessive controls cause processing delays
- Efficiency
  - People, machines, or computers waste time
    - Data is redundantly input or copied
    - Data is redundantly processed
    - Information is redundantly generated
  - People, machines, or computers waste materials and suppliers
    - Effort required for tasks is excessive
    - Materials required for tasks is excessive
- Service
  - The system produces inaccurate results
  - The system produces inconsistent results
  - The system produces unreliable results
  - The system is not easy to learn
  - The system is not easy to use
  - The system is awkward to use
  - The system is inflexible to new or exceptional situations
  - The system is inflexible to change
  - The system is incompatible with other systems
  - The system is not coordinated with other systems

4 dạng khảo sát tính khả thi

- **Khả thi về kỹ thuật**
  - **Dự án có thể thực hiện với các kỹ thuật hiện tại không ?**
  - **Kỹ thuật nào sẽ có rủi ro ?**
  - **Với các kỹ thuật hiện có :**

- **Khả thi về kinh tế**
  - **Dự án có thể thực hiện với các ràng buộc về tài nguyên hiện có ?**
  - **Có những ích lợi gì ?**
  - **Chi phí phát triển và vận hành?**
  - **Có lợi ích đáng kể về chi phí ?**

- **Khả thi về lịch biểu**
  - **Liệu có thể thiết kế được một giải pháp theo đúng kế hoạch thời gian ?**

- **Khả thi về vận hành**
  - **Khi hệ thống đã phát triển, nó sẽ được sử dụng như thế nào ?**
  - **Các nguyên tắc về con người và xã hội**

(1) Khả thi về kỹ thuật

- **Kỹ thuật được đề xuất hoặc giải pháp trên thực tế là gì?**
  - Hiện tại chúng ta có làm chủ được những kỹ thuật cần thiết?
  - Chúng ta có làm chủ được các nhà chuyên môn về kỹ thuật cần thiết?
  - Kỹ thuật liên quan có đủ hoàn chỉnh để dễ dàng áp dụng vào vấn đề của chúng ta không?
- **Loại kỹ thuật mà chúng ta cần là gì?**
  - Một số các tổ chức thích dùng các công nghệ tiên tiến (state-of-the-art)
    - …nhưng tốt nhất là dùng kỹ thuật đã hoàn thiện và đã qua trãi nghiệm
  - Một kỹ thuật hoàn thiện sẽ có một số lượng lớn khách hàng làm cơ sở cho việc thu thập các lời khuyên liên quan đến vấn đề và cải thiện chúng.
- **Kỹ thuật cần đến thì có sẵn (in house) hay không?**
  - Nếu kỹ thuật có sẵn:
    - …nó có khả năng để thao tác được giải pháp?
  - Nếu kỹ thuật không có sẵn:
    - …có thể tìm được nó hay không?

(2) Khả thi về kinh tế

- **Mức độ quan trọng có thể được số lượng hóa hay không?**
  - **Rất sớm khi bắt đầu dự án…**
    - **Cân nhắc liệu vấn đề cần giải quyết có đáng giá không**
  - **Khi đặc tả yêu cầu và giải pháp đã được xác định…**
    - **…Chi phí và lợi nhuận của mỗi sự thay đổi đều được tính toán**
- **Phân tích chi phí-lợi nhuận (costs-benefits)**
  - **Mục tiêu – trả lời các câu hỏi như :**
    - **Dự án có đáng giá để thực hiện (i.e. lợi nhuận rất cao hơn chi phí)?**
    - **Chi phí tối thiểu để chắc chắn có thể thực hiện được hệ thống ?**
    - **Sẽ thu được lợi nhuận trong bao lâu?**
    - **Sự thay đổi nào sẽ cho ra cách đầu tư tốt nhất?**
  - **Ví dụ về những thứ cần xem xét:**
    - **Chọn lựa phần cứng/phần mềm**
    - **Chọn lựa trong số những cách thỏa thuận về tài chính cho sự thay đổi (cho thuê/đi thuê/mua)**
  - **Khó khăn**
    - **Lợi nhuận và chi phí có thể cùng mơ hồ, bị che dấu và/hoặc khó đánh giá**
    - **Có quá nhiều tiêu chuẩn trong phạm vi thay đổi**

Lợi nhuận (Benefits)

- **Lợi nhuận hữu hình**
  - **Số lượng hóa một cách nhanh chóng thành giá trị tiền tệ ($)**
  - **Các ví dụ:**
    - **tăng doanh thu**
    - **giảm chi phí/lỗi**
    - **tăng số liệu nhập/hiệu quả**
    - **tăng lợi nhuận trên doanh thu**
    - **sử dụng thời gian làm việc của nhân viên hiệu quả hơn**
- **Lợi nhuận vô hình**
  - **Rất khó số lượng hóa**
    - **Nhưng có thể quan trọng hơn!**
    - **Nhà phân tích kinh doanh giúp ước lượng giá trị bằng tiền ($)**
  - **Các ví dụ:**
    - **tăng tính linh hoạt của các hoạt động**
    - **chất lượng sản phẩm cao hơn/dịch vụ**
    - **quan hệ khách hàng tốt**
    - **cải thiện tinh thần nhân viên**
- **Lợi nhuận tích lũy như thế nào?**
  - **Khi nào – cần cân đối thời gian?**
  - **Tổ chức ở đâu?**

Chi phí (Costs)

- **Chi phí phát triển (OTO)**
  - **Chi phí phát triển và buôn bán:**
    - **Chi phí cho đội ngũ phát triển**
    - **Phí tư vấn**
    - **Phần mềm đã sử dụng (mua hay thiết kế)?**
    - **Phần cứng (mua những gì, mua/thuê)?**
    - **Các tiện ích (địa điểm, phương tiện truyền thông, nguồn năng lượng,...)**
  - **Chi phí khởi tạo và chuyển đổi:**
    - **Khởi tạo hệ thống,**
    - **Huấn luyện nhân lực,**
    - **Chuyển đổi hồ sơ**
- **Chi phí vận hành (on-going)**
  - **Bảo trì hệ thống:**
    - **Phần cứng (sửa chữa, thuê, được cấp,...),**
    - **Phần mềm (bản quyền và các hợp đồng),**
    - **Các tiện ích**
  - **Nhân sự:**
    - **Cho vận hành (nhập liệu, sao lưu,…)**
    - **Cho hỗ trợ (hỗ trợ người dùng, bảo trì phần cứng và phần mềm, cung cấp,…)**
    - **Chi phí huấn luyện on-going**

Ví dụ : Chi phí cho một dự án Client-Server nhỏ

Personnel :

2	System Analysis (400 hours/ on $35.00/hr)	$28.000
4	Programmer Analysis (250 hours/ on $25.00/hr)	$25.000
1	GUI Designer (200 hours/ on $35.00/hr)	$7.000
1	Telecommunication Specialist (400 hours/ on $35.00/hr)	$2.250
1	System Architect (10 hours/ on $45.00/hr)	$4.500
1	Database Specialist (15 hours/ on $40.00/hr)	$600
1	System Librarian (250 hours/ on $10.00/hr)	$2.500

Expenses :

4	Smalltalk training registration ($3.500.00/ student)	$14.000

New Hardware & Software :

1	Development Server (Pentium Pro class)	$18.700
1	Server Software (operating system, misc, …)	$1.300
1	DBMS Server sofware	$7.300
7	DBMS Client software	$6.650

Total Development Costs : $118.200

PROJECTED ANNUAL OPERATING COSTS:

Personnel :

2	Programmer Analysis (125 hours/ on $25.00/hr)	$6.250
1	System Librarian (20 hours/ on $10.00/hr)	$200

Expenses :

1	Maintenance Agreement for Pentium Pro Server	$995
1	Maintenance Agreement for Server DBMS software	$525
	Preprinted forms (15.000/ year @/22/form)	$3.300

Total Projected Annual Costs : $11.270

Phân tích Chi phí vs. Lợi nhuận

- **Nhận biết chi phí và lợi nhuận**
  - **Hữu hình và vô hình, một lần và định kỳ**
  - **Phân chia giá trị chi phí và lợi nhuận**
- **Xác định luồng tiền mặt (Cash flow)**
  - **Dự kiến chi phí và lợi nhuận lâu dài, e.g. 3-5 năm**
  - **Tính toán Giá trị hiện tại thuần (Net Present Value) *(Hiện giá thuần)* cho toàn bộ chi phí/lợi nhuận trong tương lai**

- **Thực hiện phân tích chi phí/lợi nhuận**
  - **Tính toán Lợi nhuận trên vốn đầu tư (ROI-Return on Investment):**
    - **Cho phép so sánh lợi nhuận suốt chu kỳ sống của một giải pháp lựa chọn.**

$$\text{ROI} = \frac{\text{Tổng lợi nhuận}}{\text{Tổng chi phí}} = \frac{\text{Lợi nhuận chu kỳ sống} - \text{Chi phí chu kỳ sống}}{\text{Chi phí chu kỳ sống}}$$

  - **Tính toán Điểm hòa vốn (Break-Even point):**
    - **Bao lâu (tính bằng số năm) nó sẽ hoàn lại được chi phí tích lũy:**
      @T (Lợi nhuận tích lũy > Chi phí tích lũy)

Tính toán Giá trị hiện tại (Present Value)

- Một đồng hôm nay đáng giá hơn một đồng ngày mai …
  - Sự phân tích của bạn sẽ bình thường hóa giá trị đồng ở "năm hiện tại".
- Tỷ lệ chiết khấu (discount rate)
  - Đo lường chi phí cơ hội (opportunity cost):
    - Tiền đầu tư vào dự án này có nghĩa tiền không sẵn dùng cho những thứ khác
    - Lợi nhuận được mong đợi trong những năm sắp tới thì thiên về rủi ro hơn
  - Con số này là của cả công ty- và là đặc trưng của việc kinh doanh.
    - "Mức trung bình trả về hàng năm cho sự đầu tư vào việc kinh doanh này?"
- Giá trị hiện tại (Present value):
  - Giá trị đồng ở "năm hiện tại" cho chi phí/lợi nhuận năm n trong tương lai
    - … đối với một tỷ lệ chiết khấu i

$$Present\_Value(n) = \frac{1}{(1+i)^n}$$

  - E.g. nếu tỷ lệ chiết khấu là 12%, thì
    - $Present\_Value(1) = 1/(1 + 0.12)^1 = 0.893$
    - $Present\_Value(2) = 1/(1 + 0.12)^2 = 0.797$

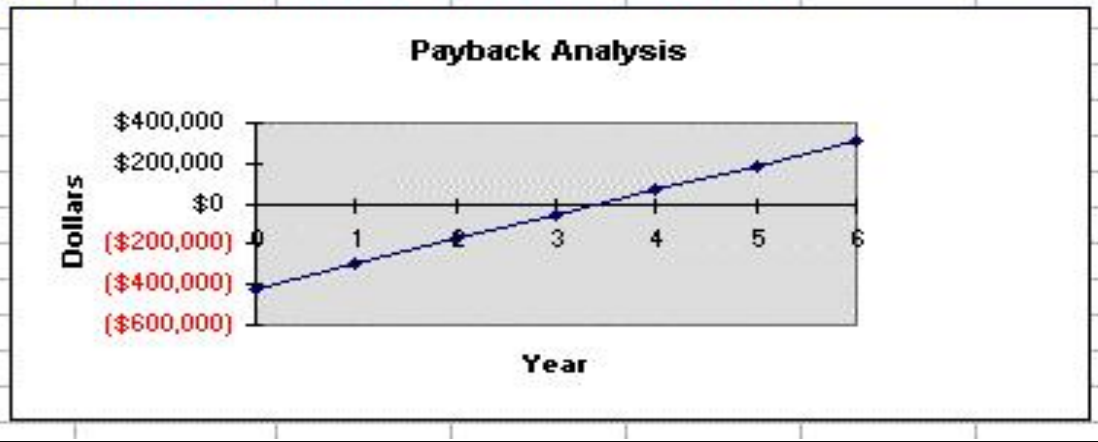
Giá trị hiện tại thuần (Net Present value)

- **Đo lường tổng giá trị đầu tư**
  - **…với tất cả các con số được điều chỉnh bằng giá trị đồng ($) hiện tại**
    - **NPV = PV cộng dồn của tất cả lợi nhuận - PV cộng dồn của tất cả chi phí**

Cash Flow	Year 0	Year 1	Year 2	Year 3	Year 4
Dev. Costs	($100,000)				
Oper.Costs		($4,000)	($4,500)	($5,000)	($5,500)
Present Value	1	0.893	0.797	0.712	0.636
Time-adj Costs	($100,000)	($3,572)	($3,587)	($3,560)	($3,816)
Cumulative Costs	($100,000)	($103,572)	($107,159)	($110,719)	($114,135)
Benefits	0	$25,000	$30,000	$35,000	$50,000
T-adj Benefits	0	$22,325	$23,910	$24,920	$31,800
Cumulative Benefits	0	$22,325	$46,235	$71,155	$102,955
Net Costs+Benefits	($100,000)	($81,243)	($60,924)	($39,564)	($11,580)

- **Giả sử những năm tiếp theo năm thứ 4…**
  - **Giá trị hiện tại thuần của việc đầu tư này trong dự án sẽ là :**
  - **sau 5 năm, $13,652**
  - **sau 6 năm, $36,168**

	A	B	C	D	E	F	G	H	I
1	**Payback Analysis for Client-Server System Alternative**								
2	(Numbers rounded to nearest $1)								
3									
4	**Cash flow description**	**Year 0**	**Year 1**	**Year 2**	**Year 3**	**Year 4**	**Year 5**	**Year 6**	
5	**Development cost:**	($418,040)							
6	**Operation & maintenance cost:**		($15,045)	($16,000)	($17,000)	($18,000)	($19,000)	($20,000)	
7	**Discount factors for 12%:**	1.000	0.893	0.797	0.712	0.636	0.567	0.507	
8	**Time-adjusted costs (adjusted to present**	($418,040)	($13,435)	($12,752)	($12,104)	($11,448)	($10,773)	($10,140)	
9	**Cumulative time-adjusted costs over**	($418,040)	($431,475)	($444,227)	($456,331)	($467,779)	($478,552)	($488,692)	
10									
11	**Benefits derived from operation of new**	$0	$150,000	$170,000	$190,000	$210,000	$230,000	$250,000	
12	**Discount factors for 12%:**	1.000	$0.89	$0.80	$0.71	$0.64	$0.57	$0.51	
13	**Time-adjusted benefits (current of present**	$0	$133,950	$135,490	$135,280	$133,560	$130,410	$126,750	
14	**Cumulative time-adjusted benefits over**	$0	$133,950	$269,440	$404,720	$538,280	$668,690	$795,440	
15		0	1	2	3	4	5	6	
16	**Cumulative lifetime time-adjusted costs +**	($418,040)	($297,525)	($174,787)	($51,611)	$70,501	$190,138	$306,748	
17									
18									
19									
20									
21									
22									
23									
24									
25									
26									
27									
28									



Phân tích thời hạn thu lợi nhuận (payback) cho dự án thay đổi một hệ thống client-server.

Tính toán thời hạn hoàn vốn (Payback)

- **Có thể tính được điểm hòa vốn (break-even point):**
  - Khi nào lợi nhuận của chu kỳ sống vượt trên chi phí chu kỳ sống?
  - Xác định tỷ lệ của một năm sau khi việc thu lợi nhuận thực sự xuất hiện:

$$\frac{\left| \textbf{beginningYear amount} \right|}{\textbf{endYear amount} + \left| \textbf{beginningYear amount} \right|}$$

  - Trong ví dụ trên : 51,611 / (70,501 + 51,611) = 0.42
  - Vì thế, thời hạn hoàn vốn (payback) thì xấp xỉ là 3.4 năm (3 + 0.42 = 3.42 năm)

Phân tích lợi nhuận trên vốn đầu tư (ROI)

- **So sánh trên lợi ích tổng thể**
  - Thay đổi nào là sự đầu tư tốt?
  - Đo lường ROI là tỷ lệ giá trị của một sự đầu tư trên chi phí của nó.
- **ROI thì được tính như sau:**

$$ROI = \frac{\text{Lợi nhuận chu kỳ sống} - \text{Chi phí chu kỳ sống}}{\text{Chi phí chu kỳ sống}}$$

Hoặc: ROI = Giá trị hiện tại thuần / Chi phí chu kỳ sống

  - Trong ví dụ trên
    - ROI = (795,440 - 488,692) / 488,692 ≈ 63%,
    - hoặc ROI = 306,748 / 488,692 ≈ 63%
- **Giải pháp với ROI cao nhất là lựa chọn tốt nhất**
  - Nhưng cũng cần phải biết thời hạn hoàn vốn nữa để có sự hình dung đầy đủ
    - E.g. Một ROI thấp hơn với thời hạn hoàn vốn sớm hơn có thể sẽ lý tưởng hơn trong một số trường hợp

(3) Khả thi về lịch biểu

- **Phải mất bao lâu để tinh thông kỹ thuật ?**
  - Chúng ta có thể có kỹ thuật, nhưng không có nghĩa là chúng ta có các kỹ năng đòi hỏi để áp dụng kỹ thuật đó một cách chính xác.
    - Có thể cần thuê nhân sự mới
    - Hoặc huấn luyện lại đội ngũ nhân viên của hệ thống hiện có.
    - Liệu việc đi thuê hay huấn luyện thì có ảnh hưởng đến lịch biểu ?
- **Đánh giá lịch biểu rủi ro:**
  - Chúng ta đã có sự tinh thông về kỹ thuật, hạn cuối (deadline) của dự án đưa ra có hợp lý không?
  - Nếu có một hạn cuối cụ thể, thì đó là thời hạn bắt buộc hay thời hạn mong muốn?
- **Điều gì là ràng buộc thực sự trên hạn cuối dự án?**
  - Nếu dự án overrun, thì hậu quả là gì?
  - Không kịp lịch biểu thì không hay, nhưng hệ thống không hoàn thiện thì còn tệ hơn nữa!

(4) Khả thi về vận hành

- **Người dùng và các nhà quản lý cảm thấy như thế nào về …**
  - …vấn đề mà bạn đã nhận ra?
  - …các giải pháp thay đổi mà bạn đang khảo sát?
- **Bạn phải đánh giá:**
  - Không chỉ liệu một hệ thống có thể hoạt động…
  - … mà còn liệu một hệ thống sẽ hoạt động hay không.
- **Mọi giải pháp đều gặp sự đối kháng:**
  - Ban quản lý có hỗ trợ dự án hay không?
  - Những người dùng cảm thấy vai trò của họ trong hệ thống mới như thế nào?
  - Những người dùng hay nhà quản lý nào có thể chống đối (hoặc không dùng) hệ thống?
  - Môi trường làm việc của người dùng sẽ thay đổi như thế nào?
  - Người dùng và ban quản lý có thể hoặc sẽ đáp ứng được với thay đổi?

Nội dung báo cáo nghiên cứu khả thi

1. Mục đích và phạm vi nghiên cứu
- Các mục tiêu (của nghiên cứu)
- Ai được ủy quyền và ai thực hiện nó,
- Các nguồn thông tin,
- Các quy trình dùng cho việc nghiên cứu,
- Tốn bao nhiêu thời gian,…

2. Mô tả hiện trạng
- Môi trường tổ chức, các hệ thống hiện tại.
- Những tác nhân và các ràng buộc liên quan.

3. Vấn đề và các yêu cầu
- Cái gì là sai trong hiện trạng ?
- Thay đổi gì thì cần thiết?

4. Các mục tiêu của hệ thống mới

Các mục tiêu cầu đạt và mối liên hệ giữa chúng.

5. Những thay đổi có thể
- …bao gồm cả 'không thực hiện gì'.

6. Tiêu chuẩn so sánh
- Định nghĩa các tiêu chuẩn

7. Phân tích các thay đổi
- Mô tả mỗi thay đổi
- Đánh giá với các khía cạnh tiêu chuẩn
- Phân tích chi phí/lợi nhuận và những gợi ý đặc biệt.

8. Kiến nghị
- Kiến nghị điều gì và các gợi ý
- Tiếp theo cần làm gì;
  - E.g. có thể đề nghị một giải pháp tạm thời và một giải pháp lâu dài

9. Phụ lục
- Bao gồm mọi tài liệu hỗ trợ.

So sánh các sự lựa chọn

- **Chúng ta sẽ so sánh các lựa chọn như thế nào?**
  - Khi nào thì có nhiều tiêu chuẩn lựa chọn ?
  - Khi nào thì không có lựa chọn nào trội hơn trên toàn diện?
- **Dùng một ma trận phân tích khả thi (Feasibility Analysis Matrix)!**
  - Cột tương ứng với các giải pháp ứng viên;
  - Dòng tương ứng với các tiêu chuẩn khả thi;
  - Các ô chứa chú thích về sự đánh giá khả thi cho mỗi ứng viên;
  - Mỗi dòng có thể gán một cấp độ hoặc điểm cho mỗi tiêu chuẩn
    - e.g., cho tính khả thi về vận hành, ứng viên có thể có các cấp độ 1, 2, 3, etc.
  - Một cấp độ hay điểm số cuối cùng được ghi nhận ở dòng cuối cùng.
- **Các tiêu chuẩn đánh giá khác cũng bao gồm trong ma trận**
  - Chất lượng output
  - Dễ sử dụng
  - Hỗ trợ đại lý
  - Chi phí bảo trì
  - Nạp vào hệ thống

Ma trận ví dụ

	Candidate 1 Name	Candidate 2 Name	Candidate 3 Name
Description			
Operational Feasibility			
Technical Feasibility			
Schedule Feasibility			
Economic Feasibility			
Ranking			

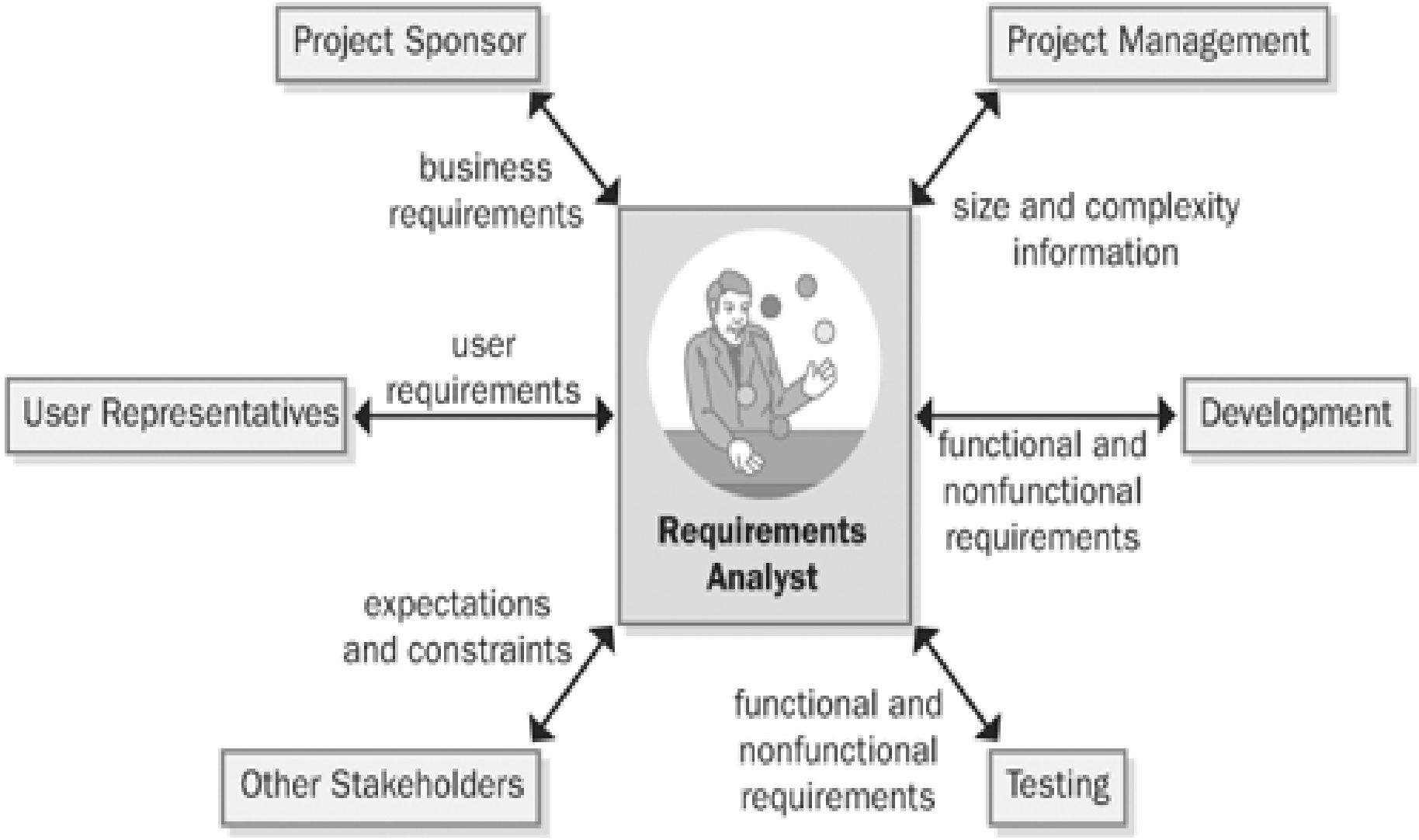
Feasibility Criteria	Wt.	Candidate 1	Candidate 2	Candidate 3	Candidate …
Operational Feasibility **Functionality.** Describes to what degree the alternative would benefit the organization and how well the system would work. **Political.** A description of how well received this solution would be from both user management, user, and organization perspective.	30%	Only supports Member Services requirements and current business processes would have to be modified to take advantage of software functionality Score: 60	Fully supports user required functionality. Score: 100	Same as candidate 2. Score: 100	
Technical Feasibility **Technology.** An assessment of the maturity, availability (or ability to acquire), and desirability of the computer technology needed to support this candidate. **Expertise.** An assessment to the technical expertise needed to develop, operate, and maintain the candidate system.	30%	Current production release of Platinum Plus package is version 1.0 and has only been on the market for 6 weeks. Maturity of product is a risk and company charges an additional monthly fee for technical support. Required to hire or train C++ expertise to perform modifications for integration requirements. Score: 50	Although current technical staff has only Powerbuilder experience, the senior analysts who saw the MS Visual Basic demonstration and presentation, has agreed the transition will be simple and finding experienced VB programmers will be easier than finding Powerbuilder programmers and at a much cheaper cost. MS Visual Basic 5.0 is a mature technology based on version number. Score: 95	Although current technical staff is comfortable with Powerbuilder, management is concerned with recent acquisition of Powerbuilder by Sybase Inc. MS SQL Server is a current company standard and competes with SYBASE in the Client/Server DBMS market. Because of this we have no guarantee future versions of Powerbuilder will "play well" with our current version SQL Server. Score: 60	

Feasibility Criteria	Wt.	Candidate 1	Candidate 2	Candidate 3	Candidate …
Operational Feasibility	30%	Score: 60	Score: 100	Score: 100	
Technical Feasibility	30%	Score: 50	Score: 95	Score: 100	
Economic Feasibility Cost to develop: Payback period (discounted): Net present value: Detailed calculations:	30%	 Approximately $350,000. Approximately 4.5 years. Approximately $210,000. See Attachment A. Score: 60	 Approximately $418,040. Approximately 3.5 years. Approximately $306,748. See Attachment A. Score: 85	 Approximately $400,000. Approximately 3.3 years. Approximately $325,500. See Attachment A. Score: 90	
Schedule Feasibility An assessment of how long the solution will take to design and implement.	10%	Less than 3 months. Score: 95	9-12 months Score: 80	9 months Score: 85	
Ranking	100%	60.5	92	83.5	

Lecture 4:
Thu thập yêu cầu

- **Ranh giới (Boundaries)**
  - **Phạm vi của vấn đề**
- **Các đối tác (Stackholders)**
  - **Xác định những người làm chủ vấn đề**
- **Mục tiêu (Goals)**
  - **Định nghĩa các tiêu chuẩn thành công**
- **Kịch bản (Scenarios)**
  - **Sử dụng các ví dụ cụ thể để hiểu vấn đề**

Nhà phân tích yêu cầu là cầu nối giao tiếp giữa khách hàng và các đối tác của sư phát triển.



Chúng ta bắt đầu từ đâu ?

- **Xác định vấn đề**
  - **Mục tiêu của dự án là gì ?**
  - **Sự nhìn nhận của người nêu ra nó ?**
    - **e.g., "Lập lịch họp hiện giờ thì quá tốn kém"**
- **Phạm vi vấn đề**
  - **Cung cấp phạm vi bàn bạc vấn đề ?**
    - **e.g. "Xây dựng hệ thống lập lịch họp", …hoặc…**
    - **e.g. "Xây dựng hệ thống quản lý lịch làm việc của nhân viên"…hoặc…**
- **Định nghĩa kịch bản cho giải pháp**
  - **Đặt vấn đề - tiến trình tương thích để giải quyết nó ?**
    - **e.g. "Một ai đó muốn lập lịch họp thì phải đến gặp thư ký, viết các chi tiết vào sổ tay thư ký và để lại", …hoặc…**
- **Phạm vi giải pháp**
  - **Nêu quá trình xử lý – phần nào sẽ phải được làm tự động và như thế nào ?**
    - **e.g. "Máy tính cần lập lịch một cách chi tiết, đầu ra là một giải pháp"…hoặc…**
    - **e.g. "Giải pháp đạt đến bằng giao tiếp giữa thư ký và máy tính"…hoặc…**

Làm rõ các yêu cầu

W6H
Kỹ thuật của các nhà báo:

What?
Where?
Who?
Why?
When?
How?
(Which?)

Điểm bắt đầu

- Một số ý kiến cho rằng có một "vấn đề" cần giải quyết
  - e.g. Không hài lòng với tình trạng công việc hiện tại
  - e.g. Một cơ hội kinh doanh mới
  - e.g. Một tiềm năng hứa hẹn sẽ tiết kiệm được về chi phí, thời gian, tài nguyên sử dụng, etc.

Cần thu thập đủ thông tin để:

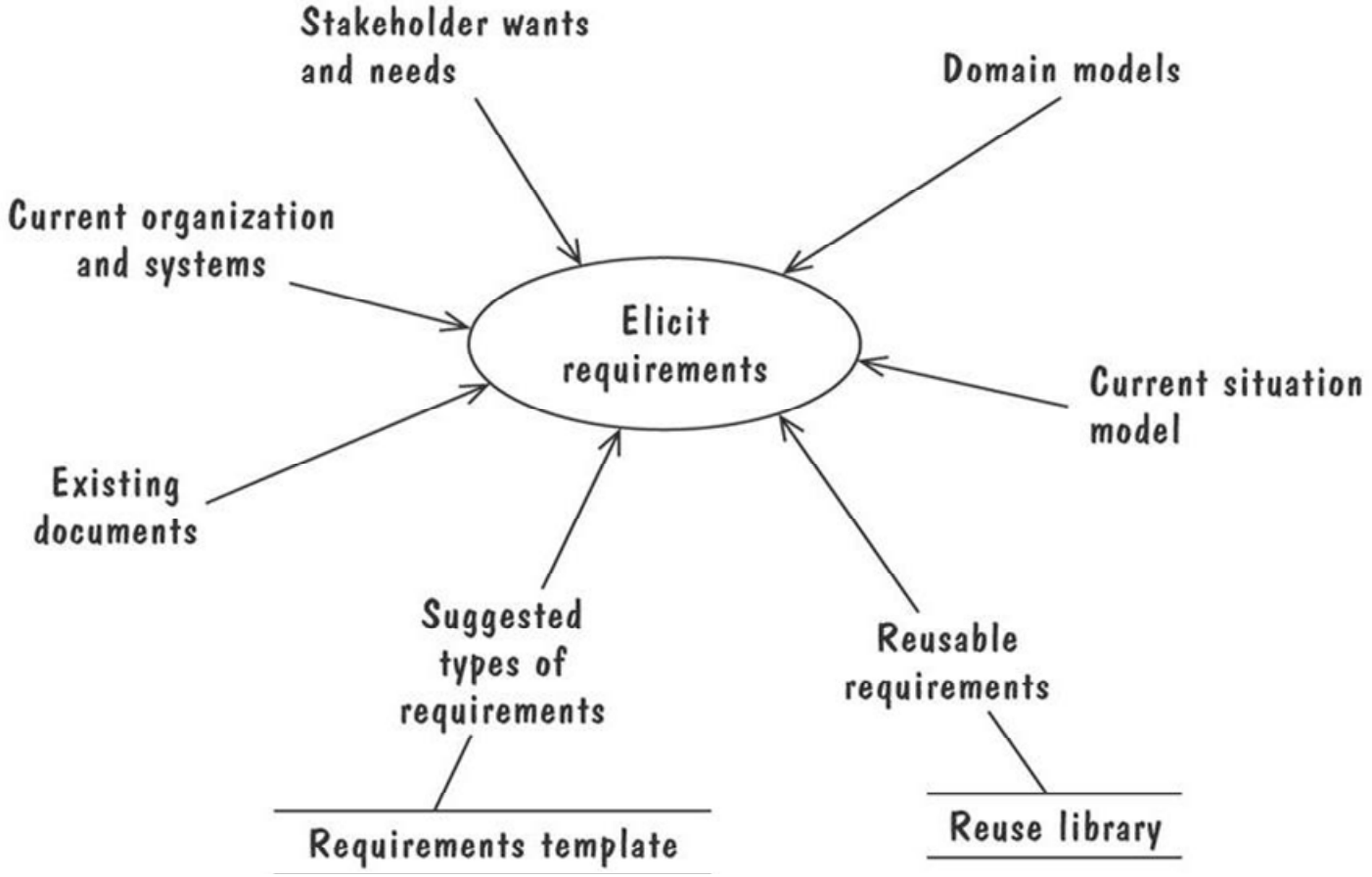
- Định nghĩa được "vấn đề" :
  - (Which) Vấn đề nào cần được giải quyết ? (Ranh giới - Boundaries)
  - (Where) Vấn đề ở đâu ? (hiểu ngữ cảnh/ phạm vi vấn đề – Context/Problem Domain)
  - (Whose) Vấn đề của ai? (Định nghĩa các Đối tác - Stakeholders)
  - (Why) Tại sao cần giải quyết? (Định nghĩa Mục tiêu đối tác – 'stakeholders' Goals)
  - (How) Hệ thống phần mềm sẽ hỗ trợ như thế nào? (Thu thập Kịch bản - Scenarios)
  - (When) Khi nào cần phải giải quyết ? (Định nghĩa các ràng buộc phát triển – Development Constraints)
  - (What) Điều gì ngăn chặn việc giải quyết chúng? (Định nghĩa tính khả thi và độ rủi ro - Feasibility and Risk)
- Là chuyên gia trong phạm vi của vấn đề
  - Nghiên cứu khoanh vùng bao quanh vấn đề mới một cách nhanh chóng
  - Dùng sự ngơ ngác (ban đầu) của bạn như một lý do để đặt những câu hỏi
  - Nhận biết lĩnh vực chuyên môn của người đang nói chuyện với bạn

Nhận dạng vấn đề

- **Vấn đề còn mơ hồ bởi chính khách hàng:**
  - **E.g. Cửa hàng bán sách của Trường**:
    - Người quản lý muốn tin học hóa việc điền vào một form yêu cầu mua sách thay vì nhận yêu cầu bằng lời nói;
  - **E.g. Một công ty bảo hiểm lớn:**
    - Người quản lý bồi thường muốn giảm thời gian trung bình của một hồ sơ bồi thường bảo hiểm từ 2 tháng xuống 2 tuần.
  - **E.g. Một công ty viễn thông**:
    - Một CIO (Chief of Information Officer) muốn tích hợp hệ thống hiện có với hệ thống lưu trữ khách hàng của một số chi nhánh thành một hệ thống duy nhất.
  - **E.g. Trạm không gian vũ trụ của chính phủ (Government Aerospace Agency)**
    - Tổng thống muốn gửi một phái đoàn đến sao Hỏa (Mars) vào năm 2020
- **Thường chỉ thấy 'triệu chứng' hơn là 'nguyên nhân':**
  - **E.g. "Bệnh nhân ở Trung tâm ung bướu muốn chụp X-ray phải chờ hàng tháng"**
  - **Thời gian chờ chỉ là biểu hiện, không phải vấn đề. Vấn đề phải là:**
    - Thiếu máy X-ray;
    - Thiếu đội ngũ chuyên môn;
    - Thiếu bác sĩ xử lý dữ liệu
    - Cách lập lịch hẹn không hiệu quả

Các nguồn bổ sung yêu cầu

 Mô hình tiến trình yêu cầu của Volere gợi ý một số nguồn bổ sung yêu cầu như sau :

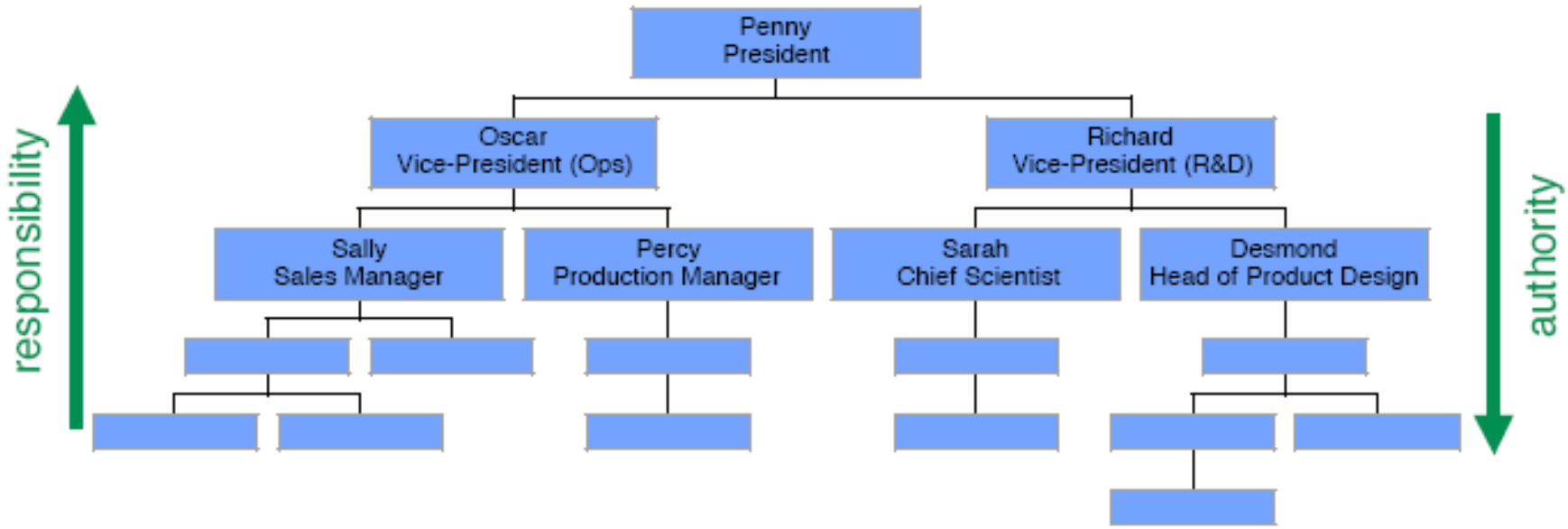


Các đối tác (Stackholders)

- **Xác định đối tác**
  - **Tất cả những người được hỏi ý kiến trong suốt quá trình thu nhận thông tin cho hệ thống.**

- **Ví dụ về đối tác**
  - **Người dùng (Users)**
  - **Nhà thiết kế (Designers)**
  - **Nhà phân tích hệ thống (Systems analysts)**
  - **Đội ngũ huấn luyện và hỗ trợ người dùng (Training and user support staff)**
  - **Nhà phân tích kinh doanh (Business analysts)**
  - **Các tác giả kỹ thuật (Technical authors)**
  - **Người quản lý dự án (The project manager)**
  - **”Khách hàng” (“The customer”)**

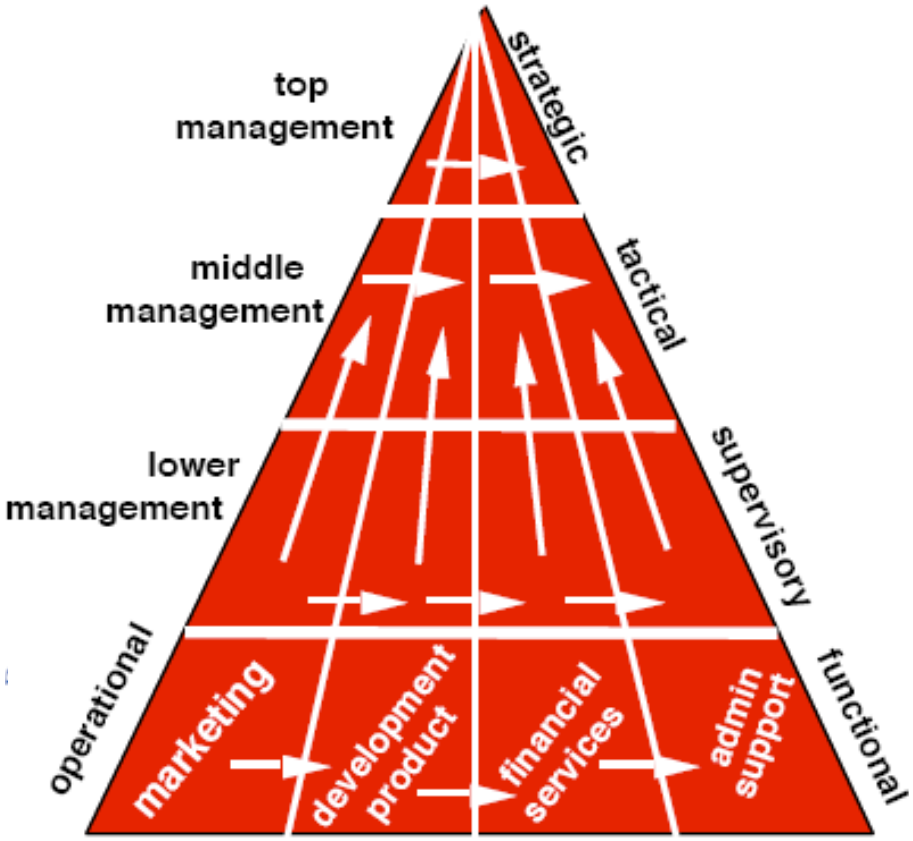
Tìm kiếm đối tác : Biểu đồ Org



- **Sự tổ chức của biểu đồ chỉ ra**
  - **Vùng trách nhiệm (dồn theo hướng đi lên)**
  - **Tuyến phân quyền (giao phó theo hướng đi xuống)**
- **Đây là một công cụ nhằm chỉ rõ các đối tác ở đâu**
  - **…Nhưng cần nhớ rằng hầu hết các hoạt động đều phải bao gồm sự liên kết ngang qua biểu đồ org**

Các cấp độ phân quyền

- **Quản trị cấp cao (top)**
  - Thiết lập các mục tiêu
  - Lập kế hoạch trên phạm vi rộng
  - Xác định thị trường mới và sản phẩm cần phát triển
  - Quyết định đối tượng liên doanh và kết quả đạt được.
- **Quản trị trung gian (middle)**
  - Sắp đặt các mục tiêu
  - Phân phối & kiểm soát tài nguyên
  - Thực hiện kế hoạch
  - Đo lường sự thực thi
- **Quản trị cấp thấp (lower)**
  - Giám sát hoạt động hàng ngày
  - Điều chỉnh các hành động khi cần thiết.
- **Cấp vận hành**
  - Thực hiện các hoạt động hàng ngày



Xác định mục tiêu các đối tác

Source: *Adapted from Anton, 1996*

Cách tiếp cận

- Tập trung vào việc tại sao một hệ thống thì cần đến
- Phát biểu 'tại sao' như là một tập mục tiêu của đối tác
- Dùng cách tinh chế mục tiêu để đạt được sự đặc tả cho các yêu cầu
- Phân tích mục tiêu
- Phát triển mục tiêu
- Phân cấp mục tiêu chỉ ra sự tinh chế (refinements) và sự chuyển đổi (alternatives)

Thuận lợi

- Mang tính trực quan
- Việc khai báo rõ ràng các mục tiêu sẽ cung cấp một nền tảng hợp lý để giải quyết các mâu thuẫn

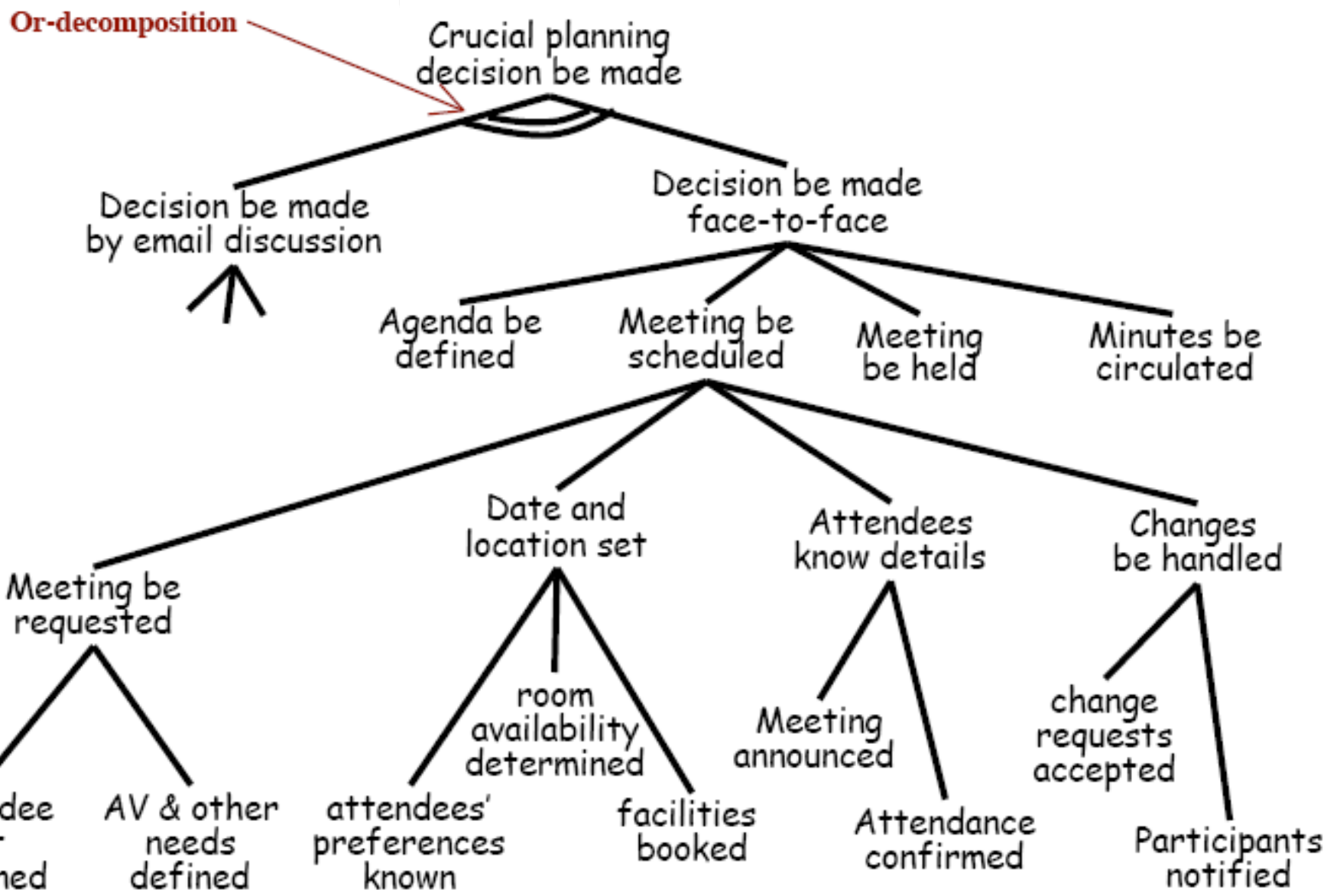
Bất lợi

- Chỉ bắt được một hình ảnh tĩnh – liệu rằng mục tiêu sẽ có thay đổi theo thời gian?
- Có thể có xu hướng lên (hoặc xuống) mãi trên sự phân cấp mục tiêu

Mô hình hóa mục tiêu

- **Mục tiêu cố định (Hardgoals)**
  - Mô tả các chức năng cần phải thực hiện.
    - Sự đáp ứng các mục tiêu
    - Việc thông tin các mục tiêu
- **Mục tiêu linh hoạt (Softgoals):**
  - Không thể thực sự đáp ứng một cách đầy đủ. E.g. Tính chính xác, Độ thực thi, Tính bảo mật, …
- **Cũng có thể phân lớp một cách tạm thời:**
  - **Mục tiêu hoàn tất/ngừng**
    - Chạm tới một số trạng thái mong muốn sau cùng
  - **Mục tiêu duy trì/bỏ đi**
    - Giữ một số đặc tính không thay đổi
  - **Mục tiêu tối ưu**
    - Một tiêu chuẩn để chọn lựa hành vi
- **Các tác nhân (agents):**
  - Là người chủ của các mục tiêu
  - Lựa chọn khi nào thì gán các mục tiêu vào tác nhân:
    - Xác định tác nhân trước, và tiếp đến là mục tiêu của chúng
    - Xác định mục tiêu trước, và sau đó chỉ định chúng cho các tác nhân trong suốt quá trình hoạt động
- **Lời khuyên khi mô hình hóa:**
  - Các nguồn phức tạp thì tốt hơn cho mục tiêu
  - Các đối tác liên đới với mỗi mục tiêu -
  - Dùng kịch bản để khảo sát sự đáp ứng của các mục tiêu
  - Xem xét kỹ lưỡng các trở ngại để giúp suy ra những ngoại lệ

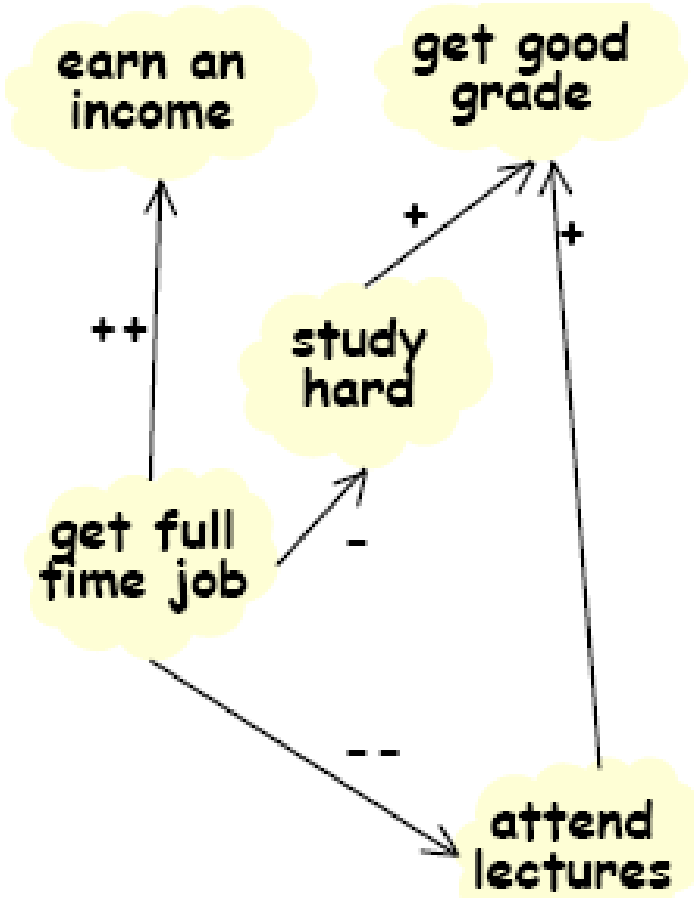
Ví dụ : Cách phát sinh mục tiêu (Cây mục tiêu)



Mô hình mục tiêu

Sự phát sinh mục tiêu

- Câu hỏi "Tại sao? (Why)" khảo sát các mục tiêu cao hơn (ngữ cảnh)
- Câu hỏi "Như thế nào? (How)" khảo sát các mục tiêu thấp hơn (hoạt động)
- Câu hỏi "Cái khác thì thế nào ?(How else)" khảo sát các chọn lựa



Quan hệ giữa các mục tiêu:

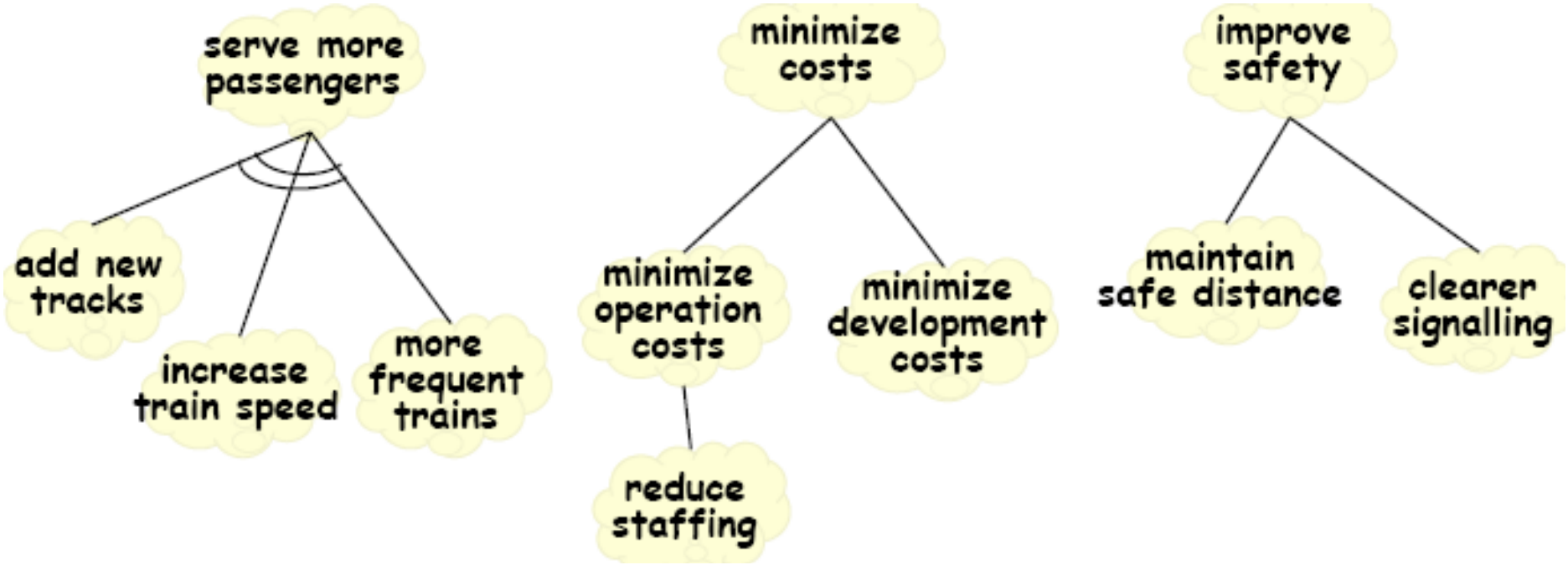
- Một mục tiêu hỗ trợ đạt đến cái khác (+)
- Một mục tiêu làm hại sự đạt đến cái khác (-)
- Một mục tiêu phát sinh cái khác (++)
  - Đạt được mục tiêu A thì chắc chắn đạt được mục tiêu B
- Một mục tiêu ngăn chặn cái khác (--)
  - Đạt được mục tiêu A thì ngăn chặn đạt được mục tiêu B
- Thứ tự ưu tiên – khi các mục tiêu phải đạt đến theo một thứ tự cụ thể

Phân tích trở ngại:

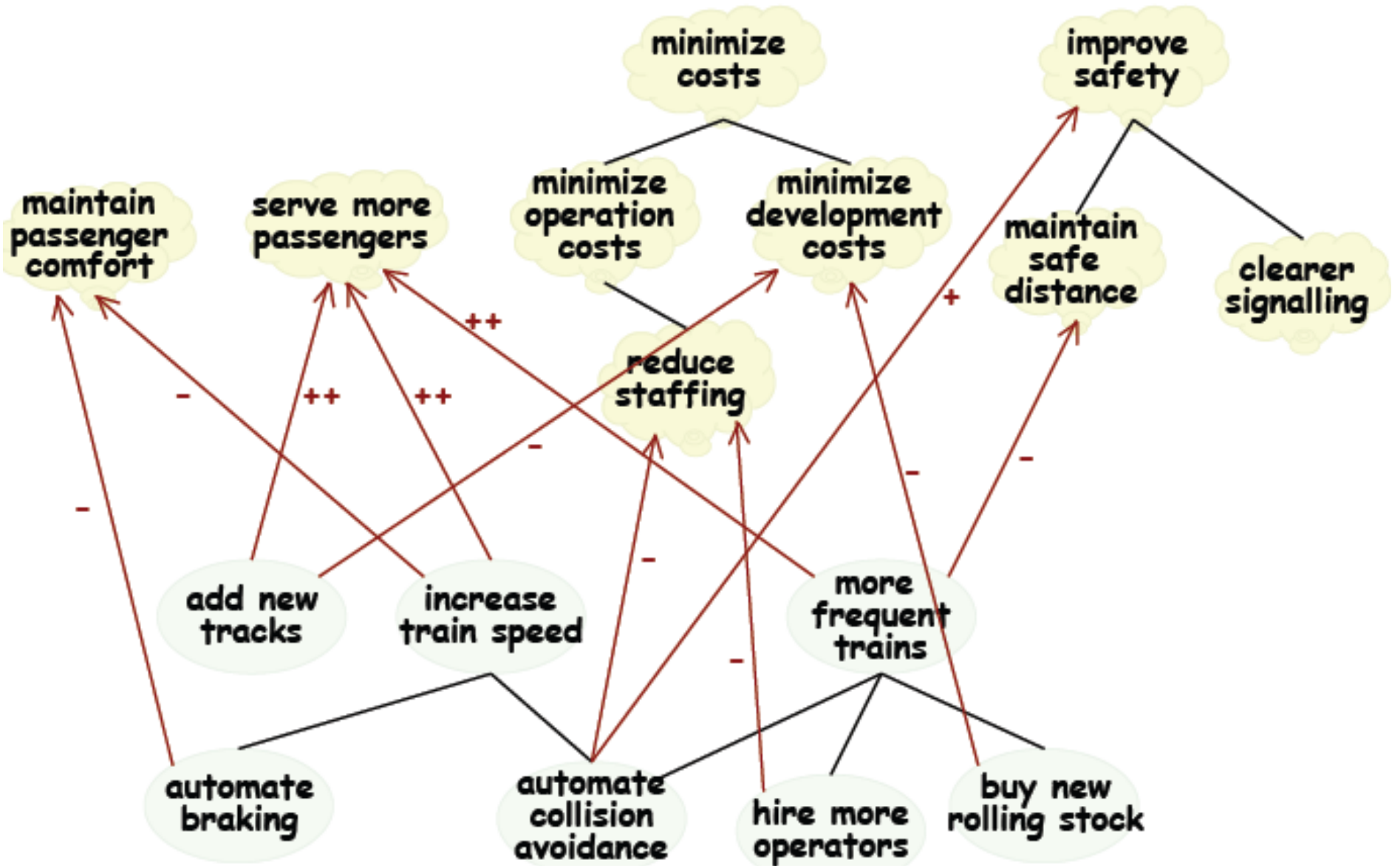
- Mục tiêu này có thể bế tắc hay không, nếu vậy thì thế nào?
- Hậu quả của việc bế tắc này là gì ?

Mục tiêu linh hoạt (SoftGoals)

- **Một số mục tiêu có thể không bao giờ được đáp ứng một cách đầy đủ**
  - Xem những mục tiêu này như mục tiêu linh hoạt
  - E.g. "hệ thống dễ sử dụng"; "truy cập an toàn"
  - Cũng được biết như là 'các yêu cầu phi chức năng'; 'các yêu cầu chất lượng'
  - Sẽ xem xét những thứ góp phần làm đáp ứng các mục tiêu linh hoạt
  - E.g. Đối với một hệ thống xe lửa:



Softgoals như các tiêu chuẩn lựa chọn



Kịch bản (Scenarios)

Source: Adapted from Dardenne, 1993

Kịch bản

- Mô tả hệ thống sẽ được dùng như thế nào trong thực tế, là dòng đặc tả giao tiếp giữa người thực hiện và hệ thống.
- Rất hữu ích cho việc thu thập yêu cầu vì con người có thể quan hệ dễ dàng hơn là các câu lệnh trừu tượng khi họ yêu cầu từ hệ thống
- Có khuynh hướng ngắn gọn (e.g từ 3 đến 9 bước)
- Có thể là kịch bản động (yêu cầu có hành vi) hoặc tĩnh (không cần sự tương tác)
- Có thể trình diễn (mô tả hệ thống hiện tại) hoặc suy diễn (nó sẽ thực hiện thế nào)

Thuận lợi

- Rất tự nhiên: các đối tác có khuynh hướng sử dụng chúng một cách tự động
  - E.g "giả sử tôi phải đi bệnh viện – chuyện gì xảy ra trong suốt thời gian nhập viện của tôi?"
  - Câu trả lời thông thường: "Bạn, hoặc người đi cùng bạn đến bàn tiếp nhận. Bạn phải trình thẻ bảo hiểm y tế của bạn và nói rõ vì sao bạn đến bệnh viện. Sau đó, …"
- Các kịch bản ngắn thì rất tốt cho việc minh họa các giao tiếp cụ thể

Bất lợi

- Thiếu cấu trúc
- Khó để kiểm tra tính hoàn thiện

Ví dụ về kịch bản (1)

Chủ đề: Sắp xếp lịch họp thành công dùng tùy chọn gửi tin nhắn

Thành viên: Nam (người đề nghị, không tham dự); Bảo, Cang, Dung (tham dự)

Hành động	Mục tiêu cần thỏa	Trở ngại / Vấn đề
b1: Nam yêu cầu cuộc họp, nêu thành viên, khung thời gian	Yêu cầu họp; Danh sách thành viên	Liệu khung thời gian đã chọn có thể không thực hiện được ?
b2: Thư ký của Nam gửi tin nhắn đến các thành viên Bảo, Cang, Dung	?	Họ không có mặt ở đó ?
b3: Bảo đọc tin nhắn **Cang đọc tin nhắn** **Dung đọc tin nhắn**	Thông tin đến các thành viên	Không thể phát hiện được khi tin nhắn được đọc, điều gì xảy ra khi Bảo đọc nhưng không phản hồi?
b4: Bảo phản hồi với lịch đề nghị **Cang phản hồi với lịch đề nghị** **Dung phản hồi với lịch đề nghị**	Nhận được lịch đề nghị của các thành viên	Liệu các lịch đề nghị có loại trừ lẫn nhau? Chúng ta sẽ cho phép ai ưu tiên cao hơn?
b5: Thư ký của Nam lập lịch cuộc họp	Xác định phòng họp có thể; Đăng ký phòng	
b6: Thư ký của Nam lưu ý Nam, Bảo, Cang, Dung về thời gian và địa điểm cuộc họp	Thông báo cuộc họp; Xác nhận lại với các thành viên (?)	Làm thế nào để biết chắc họ đã đọc được thông báo? Liệu lịch cuộc họp đã không còn thích hợp nữa cho ai đó trong số họ?

Ví dụ về kịch bản (2a)

Chủ đề: Phân tích giao dịch bán hàng trong siêu thi
Thành viên: Nhân viên bán hàng, Hệ thống

: Scenario thường (trường hợp không có lỗi)

b1: Nhân viên bán hàng nhập vào username và password
b2: Hệ thống kiểm tra username và password
b3: Hệ thống đưa ra thông báo cho biết người dùng đã đăng nhập thành công
b4: Hệ thống đưa ra chức năng tương ứng với quyền của nhân viên này.
b5: Khi khách hàng mang hàng hóa đến, nhân viên tiến hành quét mã vạch của từng món hàng
b6: Tính tiền cho khách hàng sau khi hệ thống quét hết mã vạch
b7: Nhập vào số tiền mà khách hàng đưa
b8: Nhấp vào nút "Tính" trên menu để hệ thống tính số tiền thừa trả lại cho khách hàng
b9: Nhấp nút "In" để in ra hóa đơn.

Ví dụ về kịch bản (2b)

Chủ đề: Phân tích giao dịch bán hàng trong siêu thị
Thành viên: Nhân viên bán hàng, Hệ thống

: Altenate Scenario (trường hợp có lỗi xảy ra)

A1: Username và Password không đúng (chuỗi A1 bắt đầu từ b2)
b3: Hệ thống báo lỗi không đăng nhập được
b4: quay về b1
A2: Mã vạch không hợp lệ (chuỗi A2 bắt đầu từ b6)
b7: Hệ thống đưa ra thông báo lỗi mã vạch cho nhân viên biết
b8: quay lại b6
A3: Nhập vào số tiền không đúng (chuỗi A3 bắt đầu từ b7)
b8: Hệ thống thông báo lỗi vì số tiền nhập vào không đúng
b9: quay lại b7
A4: Số tiền nhập vào nhỏ hơn số tiền cần trả (chuỗi A4 bắt đầu từ b7)
b8: Hệ thống thông báo lỗi vì không thể tính tiền
b9: quay lại b7

Kết luận

- **Phạm vi thì rất quan trọng**
  - Phạm vi vấn đề bạn cần giải quyết là gì?
  - Phạm vi của giải pháp mong muốn là gì?
- **Hỏi các câu hỏi *Ai?* (Who) và *Tại sao*? (Why)**
  - Ai là các đối tác chủ yếu ?
  - Tại sao họ muốn vấn đề này được giải quyết?
  - Phân tích mục tiêu của họ.
- **Hỏi các câu hỏi *Như thế nào*?(How)**
  - Phải đáp ứng mỗi mục tiêu như thế nào?
  - Một hệ thống mới cần được cải tiến những điều gì?
  - Phát triển các kịch bản

Lecture 5: Phân tích làm rõ yêu cầu (Eliciting Requirements)

- **Cơ sở suy luận**
  - Tại sao việc thu thập yêu cầu thì khó khăn
  - Việc đối phó với những *thiên vị* (bias)
- **Một tập hợp lớn các kỹ thuật làm rõ yêu cầu:**
  - Đọc tài liệu cơ bản (Background Reading)
  - Thu thập dữ liệu "cứng" (Hard data collection)
  - Phỏng vấn (Interviews)
  - Bảng câu hỏi (Questionnaires)
  - Kỹ thuật cộng tác (Group Techniques)
  - Tham gia quan sát (Participant Observation)
  - Điều tra xã hội học (Ethnomethodology)
  - Các kỹ thuật làm rõ tri thức

Khó khăn khi thu thập yêu cầu

- **Kiến thức hẹp về lĩnh vực**
  - Rất hiếm khi có biểu mẫu rõ ràng (không viết ra)
  - Phân tán qua nhiều nguồn
  - … Với sự mâu thuẫn giữa kiến thức từ các nguồn khác nhau
- **Kiến thức ẩn ý (Vấn đề "nói và làm")**
  - Người ta rất khó để tìm được cách mô tả cho các công việc mà họ thường làm
- **Thiếu quan sát**
  - Người chủ vấn đề quá bận rộn với công việc từ hệ thống hiện tại
  - Sự hiện diện của người quan sát có thể làm thay đổi vấn đề
- **Thiên vị (Bias)**
  - Người ta không rảnh để nói điều bạn cần biết với bạn
  - Người ta không muốn nói điều bạn cần biết với bạn -
  - Một số dạng thiên vị *(Thiên vị về tính thuyết phục (Motivational bias); Thiên vị về quan sát (Observation bias); Thiên vị về nhận thức (Cognitive bias); Thiên vị về ký pháp (Notation bias); ...)*

Ví dụ

- **Bộ phận phê chuẩn cho vay trong một ngân hàng lớn**
  - Người phân tích đang thử thu thập các quy tắc và luật lệ của việc chấp thuận cho vay

- **Tại sao vấn đề này khó khăn:**
  - **Kiến thức ẩn :** Không có tài liệu về các quy luật chấp thuận cho vay được viết ra.
  - **Thông tin mâu thuẫn :** Nhân viên ở các ngân hàng khác nhau có các ý kiến rất khác nhau về quy luật này.
  - **Vấn đề "nói và làm" :** Quy trình chấp thuận cho vay được mô tả bởi các nhân viên thì khá khác với sự quan sát của bạn về cái thực sự họ làm.
  - **Hiệu ứng thăm dò :** Quy trình chấp thuận cho vay được sử dụng bởi các nhân viên trong khi bạn quan sát thì khác với cái mà họ thường dùng.
  - **Thành kiến :** Nhân viên trong bộ phận này sợ rằng công việc của bạn sẽ tin học hóa công việc hiện có của họ, vì thế họ nhấn mạnh sự cần thiết của họ một cách kỹ lưỡng từng ly từng tí (để thuyết phục bạn rằng công việc chỉ có thể thực hiện được bởi con người!)

Các kỹ thuật làm rõ yêu cầu

- **Kỹ thuật truyền thống**
  - Tự xem xét
  - Đọc tài liệu cơ bản
  - Phân tích "dữ liệu cứng"
  - Phỏng vấn
    - Không cấu trúc (câu hỏi mở)
    - Cấu trúc (câu hỏi đóng)
  - Khảo sát / Lập bảng câu hỏi
  - Hội thảo
- **Kỹ thuật hợp tác**
  - Tập trung nhóm
    - Brainstorming
    - Hội thảo JAD/RAD
  - Lập bản mẫu
  - Cùng thiết kế
- **Hướng ngữ cảnh (xã hội)**
  - Kỹ thuật cộng đồng
    - Tham gia quan sát
    - Nhân chủng học
  - Phân tích ngôn từ
    - Phân tích cuộc đàm thoại
    - Phân tích lời nói
  - Phương pháp xã hội học
    - Phân tích hệ thống mềm
- **Kỹ thuật nhận thức**
  - Phân tích công việc
  - Phân tích giao thức
  - Các kỹ thuật làm rõ tri thức
    - Card Sorting
    - Laddering
    - Repertory Grids
    - Proximity Scaling Techniques

(1) Đọc tài liệu cơ bản

Nguồn thông tin:

- Các báo cáo của công ty, sơ đồ tổ chức, tài liệu hướng dẫn giải pháp, báo cáo quy trình nghiệp vụ, các tài liệu của hệ thống hiện có, etc.

Thuận lợi:

- Giúp bạn hiểu tổ chức trước khi tiếp xúc với những người làm việc ở đó.
- Giúp chuẩn bị về nhiều mặt trước khi tìm hiểu thực tế
  - e.g. nhận thức những mục tiêu kinh doanh của tổ chức.
- Có được các yêu cầu chi tiết về hệ thống hiện hành**.**

Bất lợi:

- Tài liệu đã viết thường không hoàn toàn phù hợp với thực tế.
- Có thể dài dòng với rất nhiều chi tiết không liên quan

Phù hợp :

- Khi bạn không thân thiện với tổ chức mà bạn sắp khảo sát

(2) Dữ liệu "cứng" và Lấy mẫu (Sampling)

- **"Dữ liệu cứng" (Hard data) bao gồm những thông tin chính xác như**
  - Các biểu mẫu, Hóa đơn, Báo cáo tài chính,…
  - Báo cáo được dùng để ra quyết định,…
  - Kết quả thống kê, dữ liệu tiếp thị,…

- **Lấy mẫu (Sampling)**
  - **Lấy mẫu dùng để chọn đại diện từ một tập phổ biến**
    - Mục đích lấy mẫu – chọn các phần mà bạn nghĩ có liên quan mà không phải theo quy luật thống kê
    - Simple Random Sampling – chọn phần tử ngẫu nhiên
    - Stratified Random Sampling – phân tầng và chọn mẫu trên mỗi tầng
    - Clustered Random Sampling – chọn một đại diện trên mỗi tập con phổ biến
  - **Kích thước mẫu thì rất quan trọng**
    - Cân đối giữa giá trị dữ liệu thu thập/ nhà phân tích và các yêu cầu quan trọng
  - **Tiến trình:**
    - Xác định thu thập dữ liệu gì - e.g. các giao dịch ngân hàng
    - Xác định tập phổ biến - e.g. tất cả các giao dịch ở 5 chi nhánh trong 1 tuần
    - Chọn kiểu của mẫu - e.g. simple random sampling
    - Chọn kích thước mẫu - e.g. cứ mỗi 20 giao dịch

Ví dụ về dữ liệu "cứng" (Hard data)

- **Câu hỏi :**

↳ Dữ liệu này cung cấp gì cho bạn ?

↳ Bạn sẽ làm gì với dữ liệu này ?

Agate

Campaign Summary

Date 23rd February 1999

Client Yellow Partridge
Park Road Workshops
Park Road
Jewellery Quarter
Birmingham B2 3DT
U.K.

Campaign Spring Collection 1999

Billing Currency GB £

Item	Curr	Amount	Rate	Billing amount
Advert preparation: photography, artwork, layout etc.	GB £	15,000.00	1	15,000.00
Placement French Vogue	FFr.	47 000,00	11.35	4,140.97
Placement UK Vogue	GB £	5,000.00	1	5,000.00
Placement US Vogue	US $	15,000.00	2.47	6.072.87
Total				30,213.84

This is not a VAT Invoice. A detailed VAT Invoice will be provided separately.

210-212 Carstairs Street, Birmingham B1 5TG
Tel.0121-111-1234 Fax.0121-111-1245
Email.agate@agateltd.co.uk

(3) Kỹ thuật phỏng vấn (Interviews)

Source: *Adapted from Goguen and Linde, 1993, p154*

- **Các dạng**
  - Cấu trúc – chương trình cho các câu hỏi mở rất rõ ràng
  - Không cấu trúc – không có chương trình trước
- **Thuận lợi**
  - Thu thập được nhiều thông tin
  - Tốt cho những quan điểm, cảm giác, mục tiêu không bao phủ cũng như các sự việc khó khăn
  - Có thể thăm dò sâu, thích hợp cho việc đặt những câu hỏi nối tiếp để hiểu rõ họ nói gì với bạn
- **Bất lợi**
  - Một số lớn dữ liệu mang tính định tính có thể khó phân tích
  - Khó để so sánh với những người thực hiện phỏng vấn khác nhau
  - Phỏng vấn là một kỹ năng khó
- **Các lưu ý**
  - Những câu hỏi không thể trả lời -
  - Kiến thức ẩn chứa ngầm
  - Sự sai lệch từ ngữ cảnh
  - Thái độ người phỏng vấn có thể tạo ra một số thiên vị

Một số kinh nghiệm phỏng vấn

- **Bắt đầu …**
  - Hãy bắt đầu cuộc phỏng vấn bằng một chủ đề vô thưởng vô phạt để tạo sự thoải mái
    - e.g. Thời tiết, tỷ số trận đá bóng tối qua, …
    - e.g. Bình luận về một đồ vật trên bàn làm việc của họ : "…Bức hình này đẹp quá ! Bạn chụp nó à ?"
- **Hỏi xem liệu bạn có thể ghi âm cuộc phỏng vấn**
  - Đặt máy ghi âm ở nơi có thể nhìn thấy
  - Cho người được phỏng vấn biết rằng họ có thể tắt máy bất cứ lúc nào.
- **Hỏi các câu hỏi dễ trước**
  - Có thể là các thông tin cá nhân
    - e.g. "Bạn đã làm việc ở vị trí hiện tại bao lâu rồi?"
- **Sau đó dẫn đến điều cần quan tâm**
  - E.g. Liệu bạn đã nghe ai đó nói rằng kế hoạch hoạt động của bạn là sai,
    - e.g.,"Chúng ta có thể nói thêm một chút về điều mà bạn vừa nói không ?"
- **Đặt các câu hỏi còn bỏ ngỏ vào phía cuối cuộc phỏng vấn**
    - e.g. "Còn điều gì khác bạn muốn nói thêm không?"

(4) Bảng câu hỏi (Questionnaires)

Source: *Adapted from Goguen and Linde, 1993, p154.*

Thuận lợi

- Có thể thu thập thông tin từ nhiều người một cách nhanh chóng
- Có thể quản lý được từ xa
- Có thể thu thập được các nội dung như quan điểm, niềm tin, tính cách

Bất lợi

- Việc đơn giản hóa cấu trúc để lập bảng sẽ cung cấp rất ít ngữ cảnh
  - Không có điều kiện cho người dùng chuyển tải những nhu cầu thực sự của họ

Các lưu ý :

- Thành kiến trong việc chọn lựa mẫu người sẽ trả lời bảng câu hỏi
- Thành kiến trong việc trả lời các lựa chọn cá nhân
- Kích thước mẫu nhỏ (thiếu sự thống kê trên diện rộng)
- Các câu hỏi dạng mở (rất khó để phân tích!)
- Các câu hỏi mơ hồ (I.e. không phải mọi người đều trả lời cùng câu hỏi)
- Các câu hỏi chỉ đạo (“Bạn thì phải … ?”)
- Các câu hỏi riêng tư (“Đây là bức hình gì ?”)

Lưu ý : Bảng câu hỏi cần phải được lập mẫu và kiểm tra !

(5) Hội thảo (Meetings)

Dùng cho tổng kết và phản hồi

- E.g. Gặp gỡ các đối tác vào cuối của mỗi giai đoạn:
  - Thảo luận về kết quả các thông tin thu thập được trong giai đoạn đó
  - Kết luận về tập hợp các yêu cầu
  - Thỏa thuận cách thiết kế, etc.
- Dùng hội thảo để xác nhận những điều đã khảo sát, thảo luận về phương hướng sắp tới

Hội thảo là một công cụ quản lý quan trọng

- Nhằm bác bỏ một dự án đã thay đổi
- Mỗi cuộc hội thảo sẽ làm sáng tỏ các mục tiêu:
  - E.g. Cách trình bày, giải quyết vấn đề, giải quyết mâu thuẫn, phân tích tiến trình, thu thập và kết hợp sự kiện, huấn luyện, lập kế hoạch,...
- Cần lập kế hoạch hội thảo một cách kỹ lưỡng
  - Thời gian hội thảo phải được sắp xếp thuận tiện
  - Chuẩn bị lịch biểu và phân phát rộng rãi
  - Giữ đúng thời gian và lịch biểu trong suốt hội thảo
  - Bám theo bản báo cáo tổng kết để thảo luận với các thành viên hội thảo
  - Các quy luật đặc biệt được áp dụng là báo cáo hình thức, đi sâu vào vấn đề (walkthroughs), động não (brainstorming), etc.

(6) Kỹ thuật làm việc nhóm

- **Các dạng:**
  - Nhóm tập trung (Focus Groups)
  - Chiến lược động não (Brainstorming)
- **Thuận lợi**
  - Có sự giao tiếp tự nhiên giữa mọi người hơn là cách phỏng vấn hình thức
  - Có thể đo lường được phản ứng với những chất liệu hỗ trợ (e.g. mô hình, bảng truy vấn, etc)
- **Bất lợi**
  - Có thể tạo ra những nhóm làm việc không thân thiện
  - Các vấn đề phát sinh từ ý kiến chung của nhóm
  - Có thể chỉ cung cấp những giải pháp thiển cận cho vấn đề kỹ thuật
  - Đòi hỏi những kỹ năng huấn luyện cao
- **Các lưu ý**
  - Có định kiến về một mẫu tiêu biểu nào đó
  - Sự áp đảo và phục tùng

(7) Phát triển ứng dụng nhanh Joint/Rapid

Nguyên tắc JAD & RAD:

- Nhóm làm việc linh động – dùng hội thảo thay vì phỏng vấn
- Phương tiện nghe nhìn : nhiều phương tiện trực quan, e.g. biểu đồ, màn ảnh rộng, giao diện đồ họa, …
- Tiến trình tổ chức : Các kỹ thuật được dùng là động não nhóm (bainstorming) và phân tích trên xuống (and top-down analysis)
- Lập tài liệu WYSIWYG : Tài liệu kết quả sau mỗi cuộc họp JAD phải dễ hiểu và phải được lập ra bởi sự thỏa hiệp chung trong suốt cuộc họp

Lưu ý:

- Chọn lựa kỹ lưỡng các thành viên tham dự : Họ sẽ là những người tốt nhất để trình bày lại cho các nhóm đối tác khác nhau
- Hội thảo nên ít nhất từ 3-5 ngày.
  - Phải gom nhóm các thành viên vào thành đội – mất 1-2 ngày.
  - Lãnh đạo cuộc họp cần chắc chắn rằng mỗi bước thực hiện đều đã được chuẩn bị kỹ lưỡng.
  - Lãnh đạo cuộc họp cần có thời gian quyết định khi có nhiều quan điểm khác nhau – “vấn đề mở”
  - Phòng hội thảo phải đủ phương tiện – dụng cụ trình chiếu, máy ghi âm, etc

(8)Tham gia quan sát

Hướng tiếp cận

- Dành thời gian để quan sát vấn đề
  - Tham gia đủ lâu để trở thành thành viên của nhóm làm việc
  - Thích hợp cho việc xem xét theo chiều dọc của vấn đề

Thuận lợi

- Có kiến thức về môi trường công việc (ngữ cảnh);
- Phát hiện được nhiều chi tiết mà các phương pháp khác không có được.

Bất lợi

- Cực kỳ tốn thời gian!
- Thu được quá nhiều kết quả sẽ rất khó để phân tích
- Không thể nói gì nhiều về sự thay đổi của kết quả đầu ra

Cần lưu ý

- Sự hòa nhập cộng đồng !

(9) Điều tra xã hội học (Ethnomethodology)

Source: *Adapted from Goguen and Linde, 1993, p158*

Lập luận cơ sở

- Môi trường xã hội thì có trật tự
  - Trật tự xã hội có thể không rõ ràng, hoặc không thể mô tả được từ nhận thức chung
- Trật tự xã hội không thể giả thiết có một cấu trúc thứ tự
  - Trật tự xã hội được thiết lập trên cơ sở từng thời điểm hiện tại thông qua sự tập hợp các hoạt động của những thành viên (không theo cấu trúc định sẵn trước)
  - i.e. trật tự xã hội chỉ có thể được quan sát khi người quan sát gia nhập chính họ vào trong đó.
- Việc quan sát nên thực hiện trong bối cảnh tự nhiên
- Cần phải xem xét ý nghĩa của *phát triển* và *tiến hóa* trong ngữ cảnh đó là như thế nào

Dùng "phạm trù chủ thể"

- Hầu hết tập quán, tục lệ đều theo hướng thừa nhận các phạm trù đã tồn tại trước đó
  - Điều này có thể sẽ đánh lạc hướng người quan sát (e.g. sự tương đồng)
- Phương pháp điều tra xã hội học cố gắng để dùng các phạm trù chủ thể
  - (Khái niệm) phạm trù nào mà mọi người dùng để lập trật tự cho môi trường xã hội?
- Phương pháp gì người ta dùng để nhận thức về thế giới xung quanh họ?
  - Dùng cùng phương pháp mà các thành viên đã dùng trong suốt khảo sát
  - E.g bằng cách phát triển một quy luật hợp pháp trong cộng đồng dưới sự quan sát.

Lecture 6:
Mô hình hóa yêu cầu

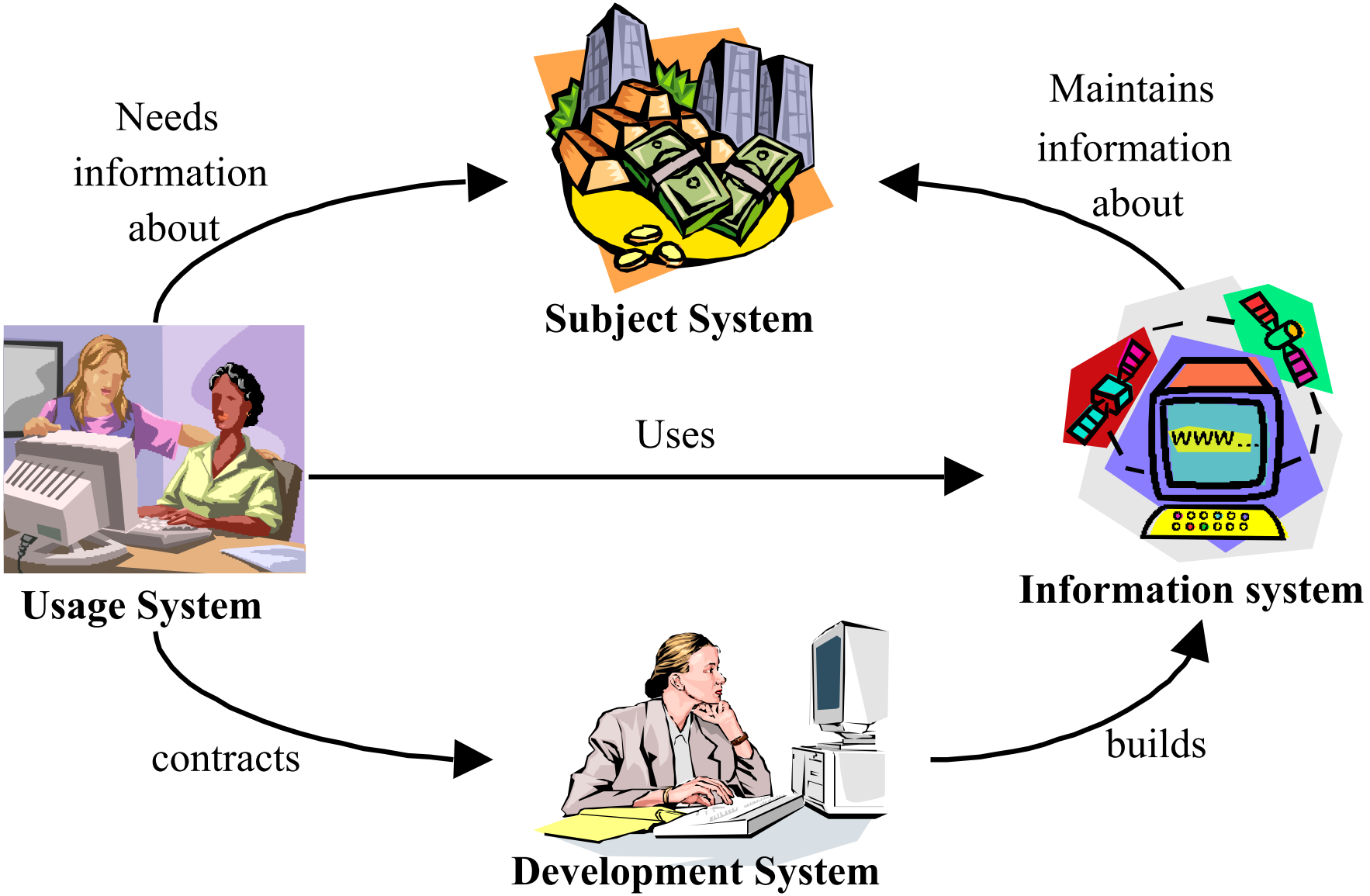
- **Làm rõ các khái niệm**
  - **Mô hình hóa là gì ?**
  - **Các yêu cầu; Hệ thống; Tư duy hệ thống (Systems Thinking)**
- **Vai trò của Mô hình hóa trong RE**
  - **Tầm quan trọng của mô hình hóa**
  - **Hạn chế của mô hình hóa**
- **Tổng quan về các ngôn ngữ mô hình hóa**
- **Nguyên tắc mô hình hóa**
  - **Trừu tượng hóa (Abstraction)**
  - **Phân tách (Decomposition)**
  - **Quy chiếu (Projection)**
  - **Mô-đun hóa (Modularity)**

Khái niệm : Các định nghĩa

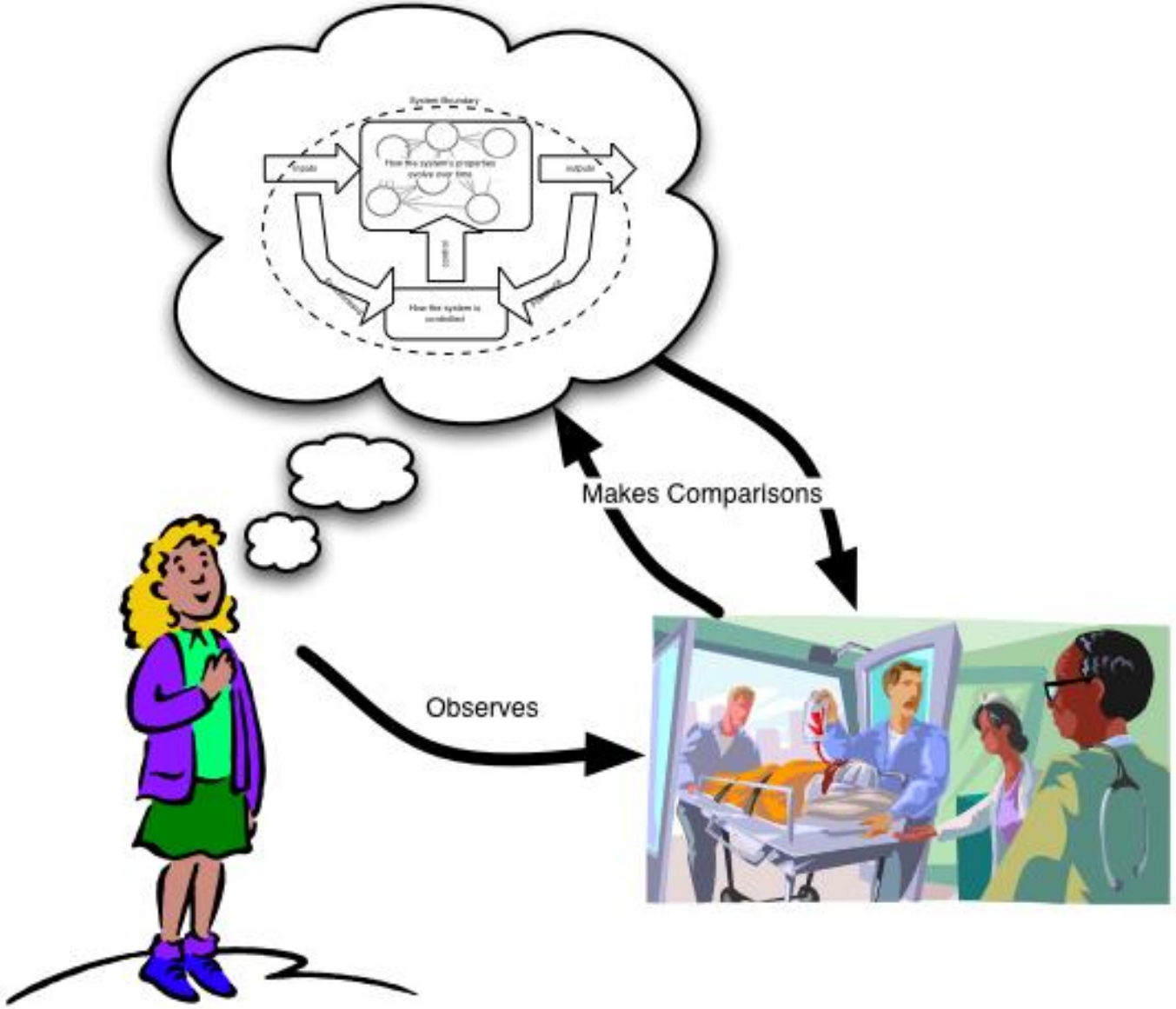
- **Một vài điểm khác biệt**
  - Domain Properties: những điều luôn luôn đúng trong lĩnh vực ứng dụng
  - Requirements: những điều chúng ta mong là đúng trong lĩnh vực ứng dụng
  - Specification: mô tả các hành vi chương trình cần thực hiện để đáp ứng với các yêu cầu
- **Hai tiêu chí cho kiểm tra (verification)**
  - Chương trình (Program) thực hiện trên một máy tính (Computer) cụ thể đáp ứng với đặc tả (Specification)
  - Đặc tả (Specification) được cho trong thuộc tính của lĩnh vực (Domain properties) thỏa mãn các yêu cầu (Requirements)
- **Hai tiêu chí cho kiểm chứng (validation)**
  - Chúng ta đã xem xét (và hiểu) tất cả các yêu cầu (Requirements) quan trọng?
  - Chúng ta đã xem xét (và hiểu) tất cả các thuộc tính lĩnh vực(Domain properties) liên quan?

Khái niệm : Từ hệ thống đến mô hình

Source: *Adapted from Loucopoulos & Karakostas, 1995, p73*



Khái niệm : Tư duy hệ thống



Mô hMô hình hóa

Mô hình hóa có thể hướng dẫn suy luận

- **Nó có thể giúp bạn chỉ ra câu hỏi gì để hỏi**
- **Nó có thể giúp làm nổi rõ các yêu cầu ẩn chứa**
  - i.e. giúp bạn hỏi những câu chính xác?

Mô hình hóa có thể cung cấp sự đo lường cho quy trình:

- **Việc hoàn thiện của mô hình -> hoàn thiện của suy luận (?)**
  - i.e. chúng ta có thể hoàn thiện tất cả các thành phần của mô hinh, được không?

Mô hình hóa có thể giúp phơi bày các vấn đề

- **Sự mâu thuẫn trong các mô hình có thể dẫn đến nhiều thứ đáng quan tâm…**
  - e.g. các yêu cầu xung đột hoặc không thể thực hiện
  - e.g. nhầm lẫn các thuật ngữ, phạm vi, etc
  - e.g. bất đồng giữa các đối tác

Mô hình hóa có thể giúp kiểm tra sự thấu hiểu của bạn

- **Lý giải trên các mô hình để hiểu kết quả của nó**
  - Nó có đạt được những đặc tính mà chúng ta mong muốn?
- **Xây dựng hình ảnh bằng các mô hình giúp quan sát/kiểm chứng các yêu cầu**

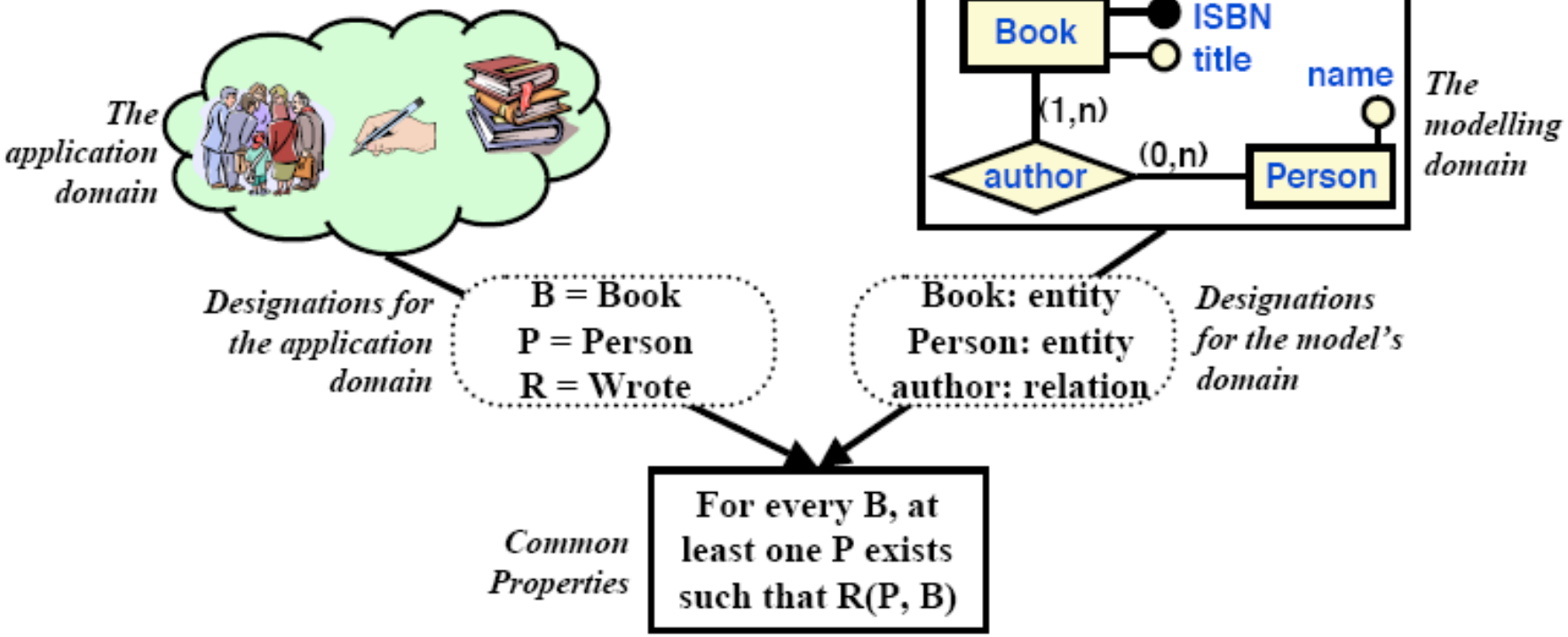
RE gồm nhiều bước mô hình hóa

Source: *Adapted from Jackson, 1995, p120-122*

Mô hình thì tốt hơn chỉ là sự mô tả

- **Nó có các hiện tượng của nó và có quan hệ chủ thể giữa các hiện tượng này.**
  - Mô hình sẽ hữu ích khi các hiện tượng của mô hình phù hợp một cách có hệ thống với các hiện tượng trong lĩnh vực mà nó cần được mô hình hóa

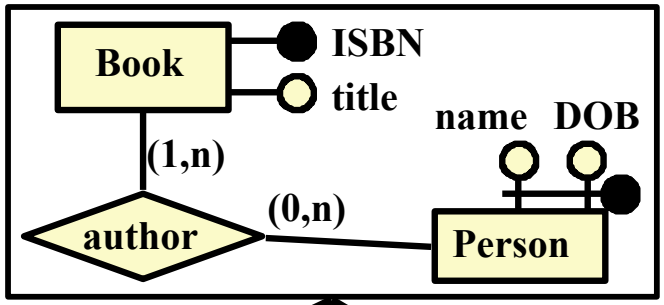
- **Example:**



"Đó chỉ là mô hình"

Source: *Adapted from Jackson, 1995, p124-5*

- **Rất thường thấy rằng:**
  - **Hiện tượng trong mô hình thì không hiện diện trong lĩnh vực ứng dụng**
  - **Hiện tượng trong lĩnh vực ứng dụng thì không có trong mô hình**



Hiện tượng không được nắm bắt trong mô hình

…các tác giả "ma"…
…bút danh …
…nặc danh…

Hiện tượng chung

…mỗi quyển sách có ít nhất một tác giả…
…mỗi quyển sách chỉ có một ISBN duy nhất …

Hiện tượng không thực tế

…không có hai người sinh ra vào cùng ngày và có cùng tên…

- **Một mô hình không khi nào là hoàn hảo**
  - **"Nếu bản đồ và địa hình không giống nhau, hãy tin vào địa hình"**
  - **Tìm kiếm sự hoàn hảo của một mô hình thì không là việc tốt cho thời gian của bạn...**

Chọn ký pháp cho việc mô hình hóa

Source: *Adapted from Loucopoulos & Karakostas, 1995, p72-73*

Ngôn ngữ tự nhiên

- Cực kỳ diễn cảm và linh hoạt
  - hữu ích cho suy diễn, và lập các mô hình ký hiệu dễ đọc
- Khó để nắm bắt được các quan hệ mấu chốt

Ký pháp bán hình thức

- Nắm được cấu trúc và một số ngữ nghĩa
- Có thể thực hiện (một số) hoạt động, kiểm tra tính nhất quán, ảnh động, etc.
  - E.g. lược đồ, bảng, cấu trúc tiếng Anh, etc.
- Gần như là trực quan – cho phép chuyển thông tin một cách nhanh chóng đến các dạng đối tác khác nhau

Ký pháp hình thức

- Ngữ nghĩa chính xác, có thể suy luận rộng
  - các mô hình dựa trên cơ sở toán (e.g. lý thuyết tập hợp, FSMs (finite-state machine), etc)
- Các mô hình rất chi tiết (có thể chi tiết hơn cả cái chúng ta cần)
  - RE hình thức thì không chấp nhận việc mô hình hóa, điều này thì khác với hầu hết các dạng khoa học máy tính khác

Mục tiêu của ký pháp mô hình hóa

Source: *Adapted from Loucopoulos & Karakostas, 1995, p77*

- **Cài đặt độc lập**
  - Không mô hình sự hiển thị dữ kiệu, cách tổ chức bên trong, etc.
- **Tính trừu tượng**
  - Đưa ra các khía cạnh thiết yếu
  - e.g. những thứ không buộc phải thay đổi thường xuyên
- **Tính hình thức**
  - Cú pháp không mơ hồ
  - Ngữ nghĩa biểu cảm
- **Tính kiến trúc**
  - Có thể thiết kế từng phần của mô hình để kiểm soát được độ phức tạp và kích thước của nó
  - Thiết kế cần có sự giao tiếp dễ dàng
- **Dễ phân tích**
  - Cho phép phân tích dữ liệu mơ hồ, chưa đầy đủ và không nhất quán
- **Dễ lần vết**
  - Cho phép các phần tử tham chiếu
  - Cho phép liên kết với thiết kế cài đặt, etc.
- **Tính khả thi**
  - có thể cho mô hình hoạt động để so sánh nó với thực tế
- **Tối thiểu hóa**
  - Không dư thừa các khái niệm trong lược đồ mô hình hóa
    - i.e. không chọn lựa hiển thị các vấn đề nào đó không liên quan

Khảo sát các kỹ thuật mô hình hóa

Mô hình hóa nghiệp vụ

- Mục đích & Mục tiêu
- Kiến trúc tổ chức
- Công việc & các phụ thuộc
- Tác nhân, vai trò, dự định

Mô hình hóa tổ chức:
i*, SSM, ISAC
Mô hình hóa mục tiêu:
KAOS, CREWS

Mô hình hóa thông tin & hành vi

- Cấu trúc thông tin
- Quan điểm hành vi
  - Kịch bản và tình huống
  - Mô hình máy trạng thái
  - Dòng thông tin
- Các yêu cầu về thời gian/trình tự

Mô hình hóa thông tin:
E-R, Class Diagrams
Phân tích cấu trúc:
SADT, SSADM, JSD
Phân tích hướng đối tượng:
OOA, OOSE, OMT, UML
Các phương pháp hình thức:
SCR, RSML, Z, Larch, VDM

Mô hình hóa chất lượng hệ thống

- Những gì 'có thể':
  - Có thể sử dụng, đáng tin cậy, có thể phát triển, an toàn, bảo mật, khả thi, tương tác,…

Thỏa thuận chất lượng:
QFD, win-win, AHP,
Đạc tả NFRs:
Timed Petri nets (mức độ thực thi)
Task models (tính dễ sử dụng)
Probabilistic MTTF (độ tin cậy)

Unified Modelling Language (UML)

Phương pháp hướng đối tượng thế hệ thứ ba

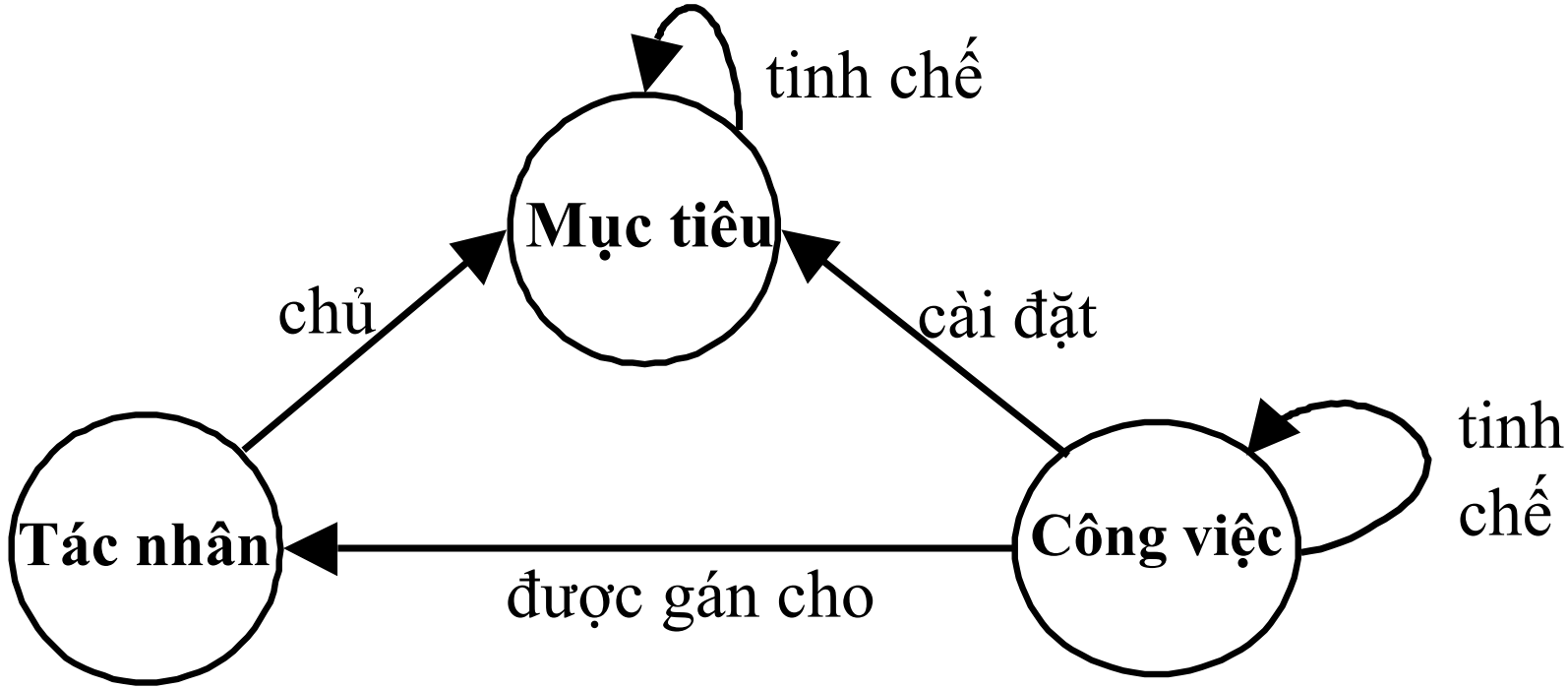
- Booch, Rumbaugh & Jacobson là những tác giả đầu tiên
  - Vẫn còn đang tiến hóa
  - Nổ lực chuẩn hóa sự tiến triển trên các dạng hướng đối tượng khác nhau
- Hoàn toàn là ký pháp
  - Không có phương pháp mô hình nào liên quan tới nó!
  - Được dự định là một thiết kế ký pháp (một số đặc tính không phù hợp với RE)
- Đã trở thành một công nghệ chuẩn
  - Nhưng được làm chủ bởi IBM/Rational (đã bán nhiều công cụ và dịch vụ UML)

Có một khung mô hình (meta-model) chuẩn

- Use case diagrams
- Class diagrams
- Message sequence charts
- Activity diagrams
- State Diagrams
- Module Diagrams
- Platform diagrams

Meta-Modelling

- **Có thể so sánh lược đồ mô hình hóa dùng meta-models:**
  - **Mỗi lược đồ sẽ nắm bắt các hiện tượng gì ?**
  - **Cách thức soạn thảo các mô hình cần dựa theo hướng dẫn nào?**
  - **Cần thực hiện phân tích gì trên các mô hình?**
- **Ví dụ về meta-model:**



Nguyên tắc mô hình hóa

Dễ dàng sửa đổi và tái sử dụng

- **Những nhà phân tích có kinh nghiệm thường sử dụng lại kinh nghiệm trước đây của họ**
  - họ sử dụng lại các thành phần (của mô hình mà họ đã xây dựng trước đó)
  - họ sử dụng lại cấu trúc (của mô hình mà họ đã xây dựng trước đó)
- **Những nhà phân tích thông minh có thể hoạch định cho tương lai**
  - họ tạo ra các thành phần có thể sử dụng lại trong mô hình của họ
  - họ cấu trúc mô hình của họ để chúng dễ dàng sửa đổi

Các ý niệm hữu ích:

- **Trừu tượng hóa (Abstraction) :** Tháo bỏ các chi tiết để tập trung vào những thứ quan trọng
- **Phân tách (Partitioning) :** Phân chia vấn đề thành các phần độc lập, để khảo sát riêng biệt
- **Quy chiếu (Projection) :** Phân chia các khía cạnh (views) khác nhau và mô tả chúng một cách riêng biệt
- **Mô-đun hóa (Modularization) :** Chọn lựa các cấu trúc ổn định theo thời gian để dễ định vị sự thay đổi
- **Mẫu (Patterns) :** Cấu trúc của một mô hình đã có xuất hiện trong nhiều ứng dụng khác nhau

Nguyên tắc 1: Phân tách

Sự phân tách

- **Nắm rõ được sự tập hợp (aggregation)/phần của quan hệ (relationship)**

Ví dụ:

- **Mục tiêu là khai thác một con tàu vũ trụ**
- **Phân tách vấn đề thành các vấn đề con:**
  - Các chỉ dẫn và cách điều khiển;
  - Quản lý dữ liệu;
  - Chỉ huy và kiểm soát;
  - Kiểm soát môi trường;
  - Theo dõi các thiết bị đo đạc;
  - etc
- **Chú ý: đây không phải thiết kế, chỉ là sự phân tích vấn đề**
  - Thiết kế thực sự phải có đủ mọi thành phần, không có liên quan với các vấn đề con này
- **Tuy nhiên, cách chọn lựa của phân tách vấn đề sẽ có thể được phản ánh trong thiết kế**

Nguyên tắc 2: Trừu tượng hóa

Source: *Adapted from Davis, 1990, p48 and Loucopoulos & Karakostas, 1995, p78*

Trừu tượng hóa

- Là cách tìm kiếm sự tương tự giữa các khái niệm bằng việc lờ đi một số các chi tiết
- Tập trung vào mối quan hệ tổng quan/cụ thể giữa các hiện tượng
  - Phân loại vào thành nhóm các thực thể khi chúng có vai trò tương tự như thành phần của một nhóm độc lập
  - Quan hệ kế thừa biểu diễn sự tương tự giữa các lớp khác nhau trong một mối quan hệ kết hợp '(is_a)'

Ví dụ:

- Yêu cầu là : kiểm soát lỗi trên tàu vũ trụ
- Phải nhóm các lỗi khác nhau vào thành lớp LỖI

Dựa vào vị trí:

- lỗi thiết bị đo,
- lỗi truyền thông,
- lỗi xử lý,
- etc

Dựa vào triệu chứng:

- không có đáp ứng từ thiết bị;
- đáp ứng không chính xác;
- tự báo lỗi;
- etc...

Nguyên tắc 3: Quy chiếu

Source: *Adapted from Davis, 1990, p48-51*

Quy chiếu:

- Phân chia các lĩnh vực của mô hình thành nhiều khía cạnh (viewpoints)
  - tương tự các phép chiếu được dùng bởi kiến trúc sư trong xây dựng

Ví dụ:

- Cần lập các mô hình về yêu cầu cho tàu vũ trụ
- Phân chia mô hình :
  - độ an toàn
  - khả năng chỉ huy
  - khả năng chịu lỗi
  - đúng thời gian và trình tự
  - Etc…

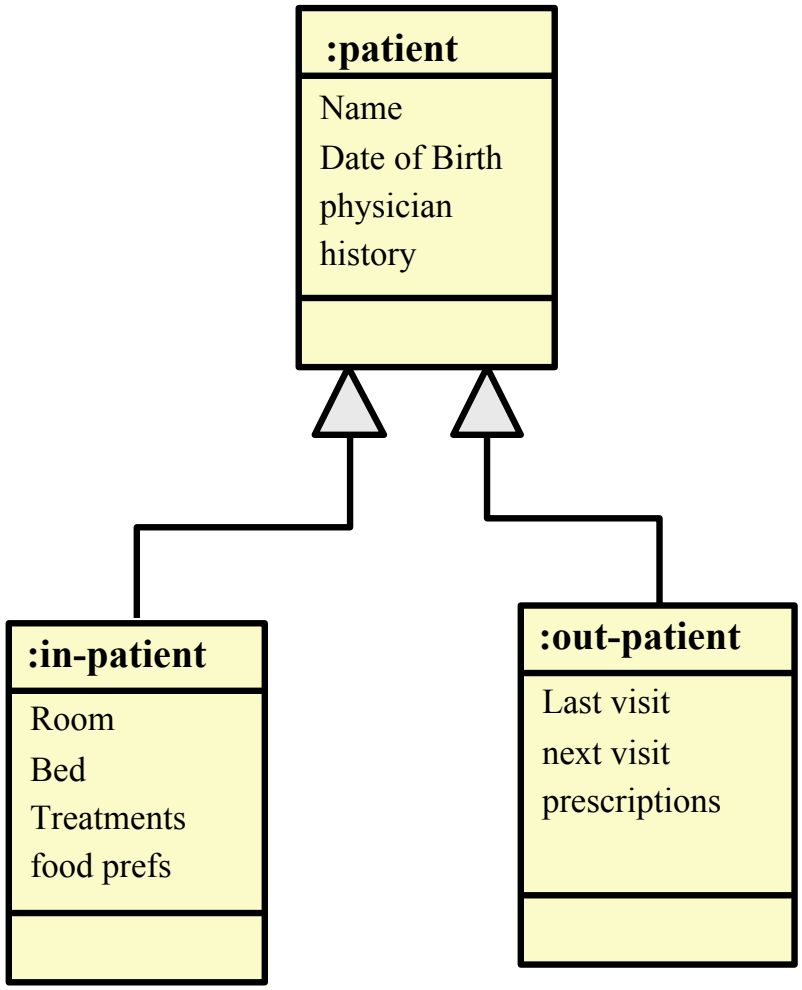
Chú ý:

- Quy chiếu và Phân tách thì tương tự nhau:
  - Phân tách định nghĩa một 'phần' của quan hệ
  - Phép chiếu định nghĩa một 'khía cạnh' của quan hệ
- Phân tách thừa nhận mỗi phần thì tương đối độc lập

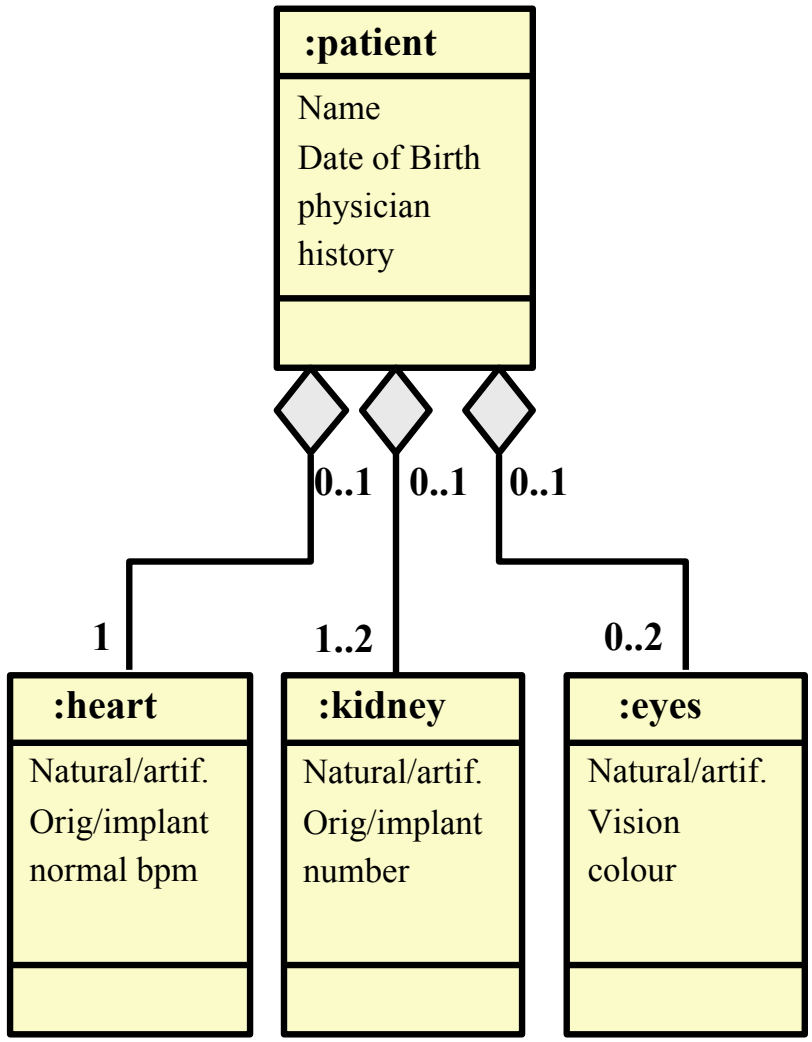
Một ví dụ tổng quan về UML

Source: *Adapted from Davis, 1990, p67-68*

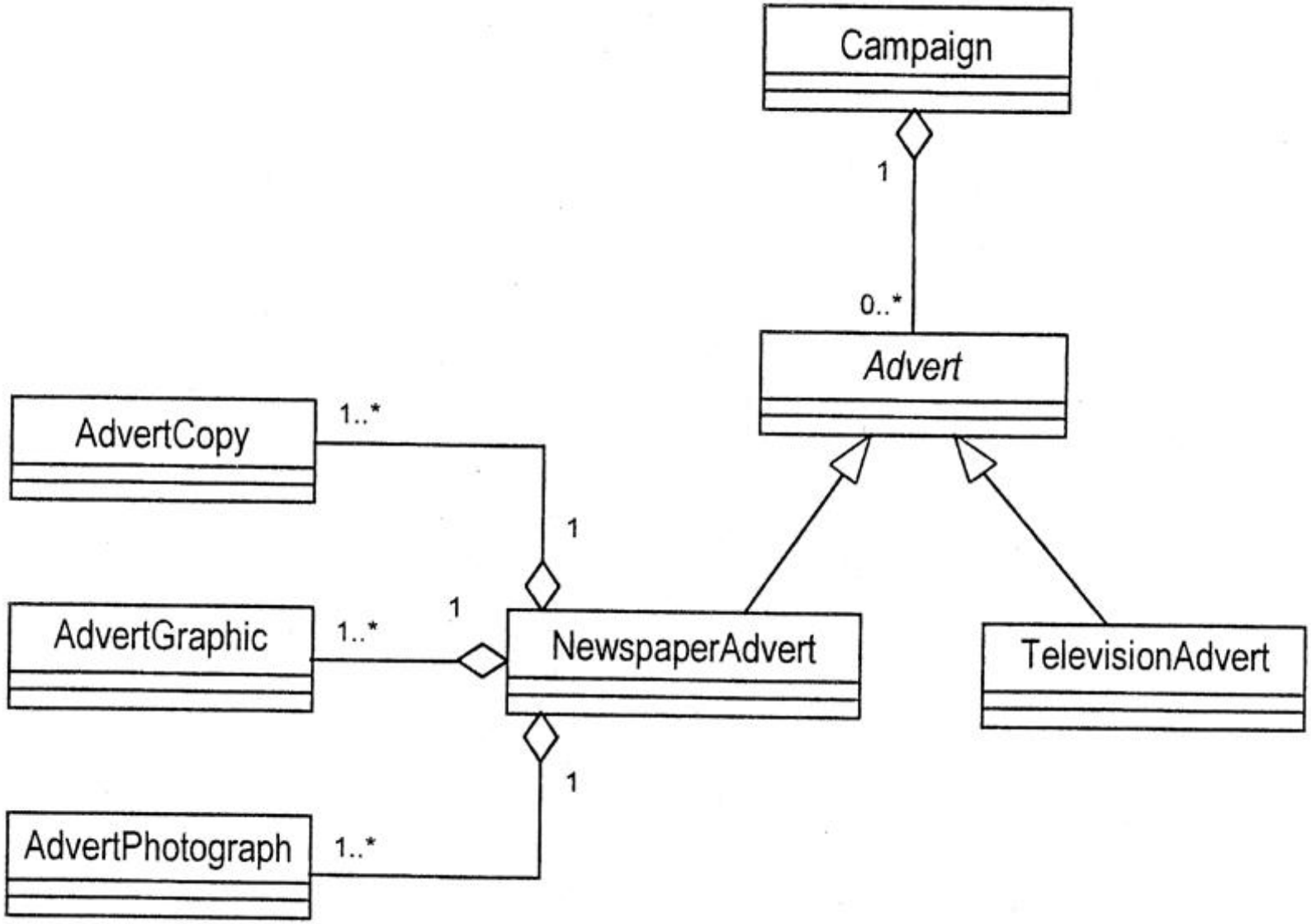
Quan hệ thừa kế
(hệ thống phân cấp trừu tượng)



Quan hệ tập hợp
(hệ thống phân cấp phân chia)



Đây là mô hình cho vấn đề gì ?



Kết luận

- **Mô hình hóa đóng vai trò trọng tâm trong RE**
  - **Cho phép chúng ta khảo sát vấn đề một cách hệ thống**
  - **Cho phép chúng ta kiểm tra sự hiểu biết của mình**
- **Co nhiều lựa chọn ký pháp cho mô hình**
  - **Trong course này, chúng ta sẽ dùng các dạng ký pháp của UML**
- **Tất cả các mô hình thường thiếu chính xác**
  - **Sử dụng các mô hình được hiệu chỉnh liên tục**
  - **…nhưng có thể biết khi nào ngừng việc hoàn chỉnh mô hình**
  - **Mỗi mô hình được tạo ra cho một mục đích riêng**
  - **Mục đích thường không được biểu diễn trong mô hình**
  - **… Vì thế nên mỗi mô hình đều cần có một sự giải thích**

Lecture 07:
Mô hình quan hệ thực thể
(Entity Relationship Modelling)

Mô hình quan hệ - thực thể (Entity-Relationship Model)

- Thực thể (Entities)
- Quan hệ (Relationships)
- Thuộc tính (Attributes)

Các ràng buộc trên thể hiện

- Bản số (Cardinalities)
- Khóa định dạng (Identifiers)
- Tổng quát hóa (Generalization)

Mô hình quan hệ thực thể

Lược đồ quan hệ - thực thể (Entity-Relationship Schema)

- Mô tả các yêu cầu dữ liệu cho một hệ thống thông tin mới
- Dùng ký hiệu đồ họa một cách trực tiếp, dễ hiểu
- Chuyển thành lược đồ quan hệ cho thiết kế cơ sở dữ liệu một cách nhanh chóng
  - Nhưng trừu tượng hơn lược đồ quan hệ
  - E.g. có thể hiển thị một thực thể mà không biết các đặc tính của nó.

Các thực thể (Entities):

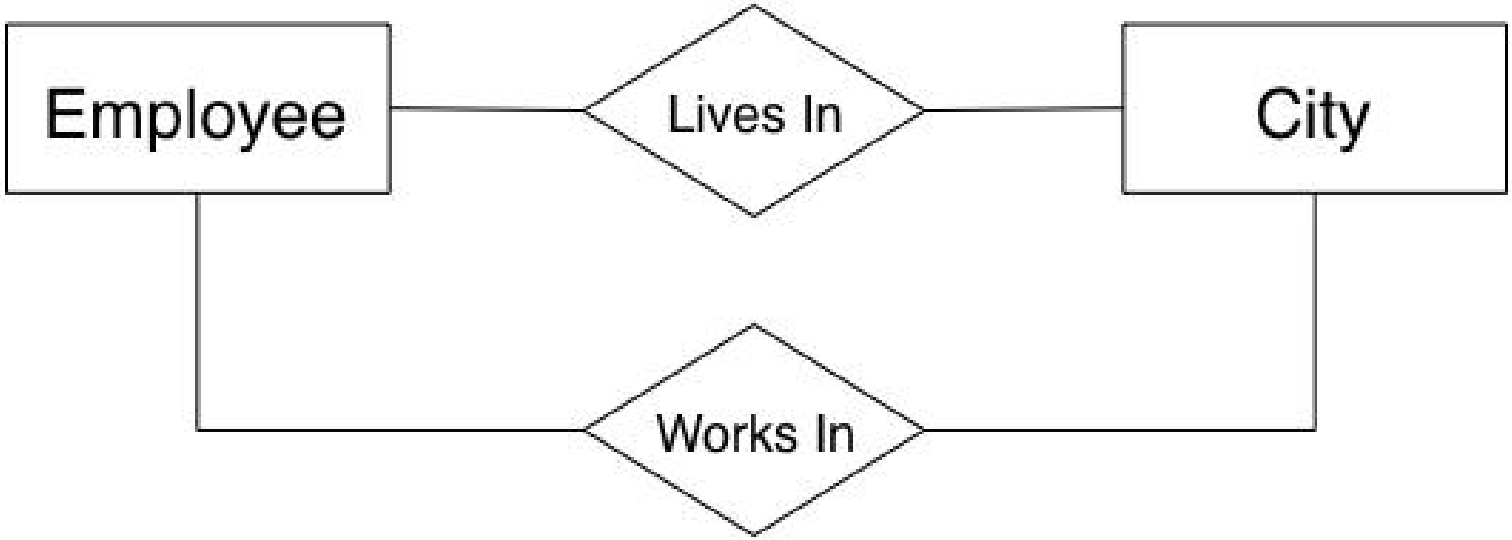
- Lớp các đối tượng với các đặc tính chung và một phạm vi tồn tại
  - E.g. Thành phố, Bộ môn, Nhân viên, Mua và Bán
- Một thể hiện của một thực thể là một đối tượng trong lớp được biểu diễn bởi thực thể
  - E.g. Cần Thơ, Đà Lạt là các ví dụ thể hiện của thực thể Thành phố

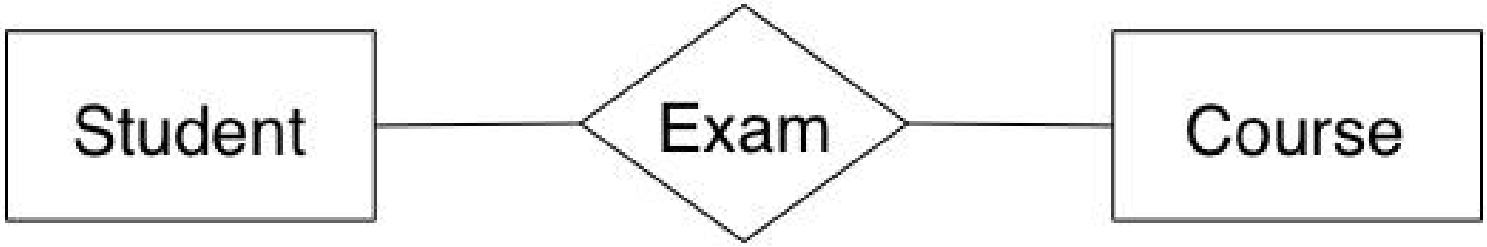
Các quan hệ (Relationships):

- Các nối kết logic giữa hai hoặc nhiều thực thể.
  - E.g. Cư trú là một quan hệ có thể tồn tại giữa Thành phố và Nhân viên
- Một thể hiện của một quan hệ là một thể hiện n-tuple của thực thể
  - E.g. bộ (Nam, Cần Thơ), là một thể hiện trong quan hệ Cư trú.

Các ví dụ

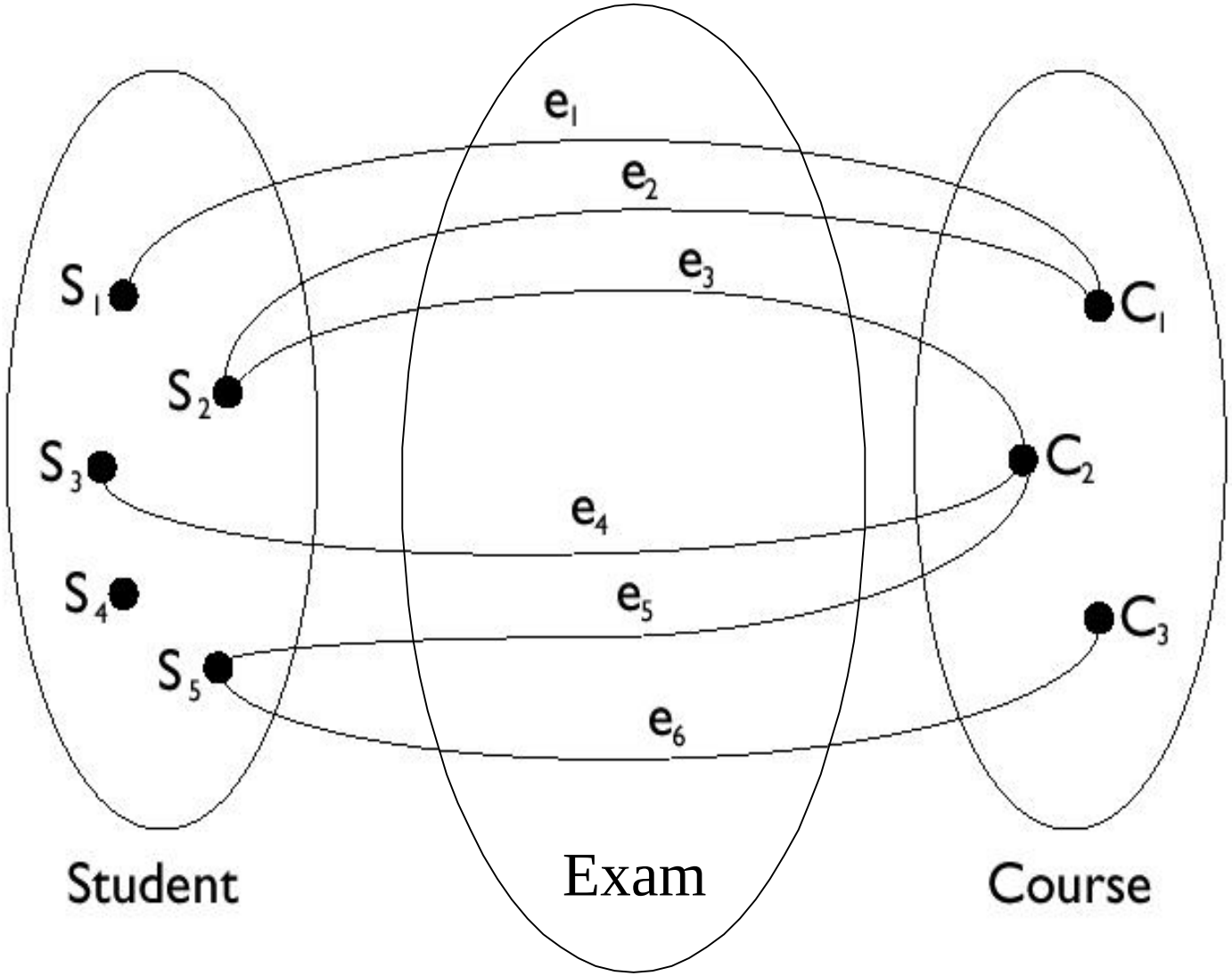
Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999





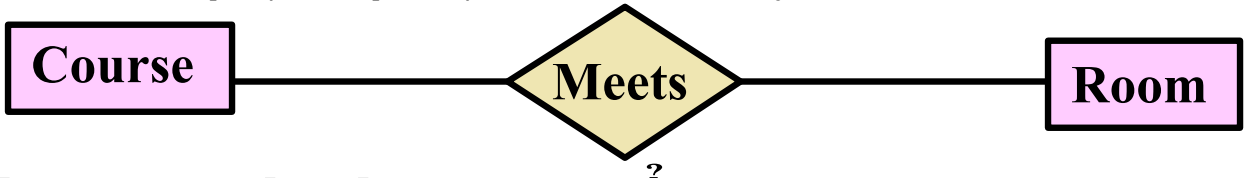
Ví dụ thể hiện cho liên kết Exam

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999



Ý nghĩa thực sự của một sơ đồ ER?

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

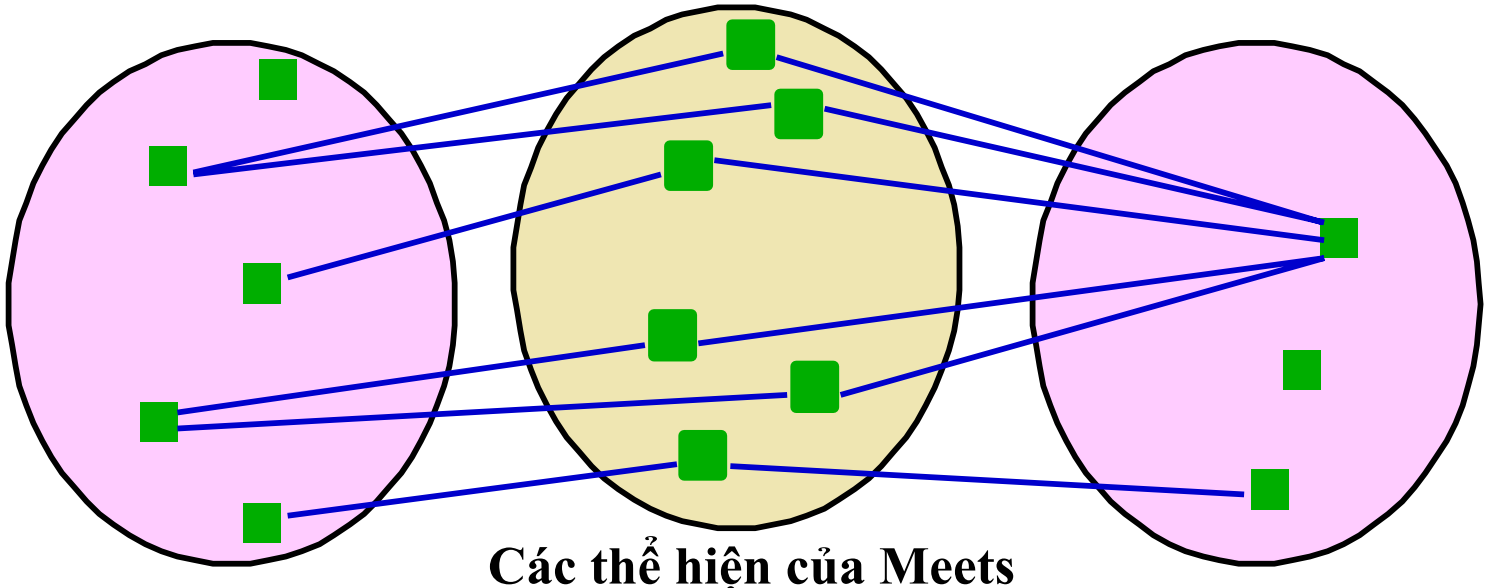


Course và Room là các thực thể.

- Thể hiện của chúng là courses cụ thể (eg CT324) và rooms (eg 202/C1)

Meets là một quan hệ.

- Các thể hiện của nó mô tả các buổi học cụ thể.
- Mỗi buổi học có chính xác một kết hợp giữa course và room.



Các thể hiện của Meets

Các thể hiện của Course

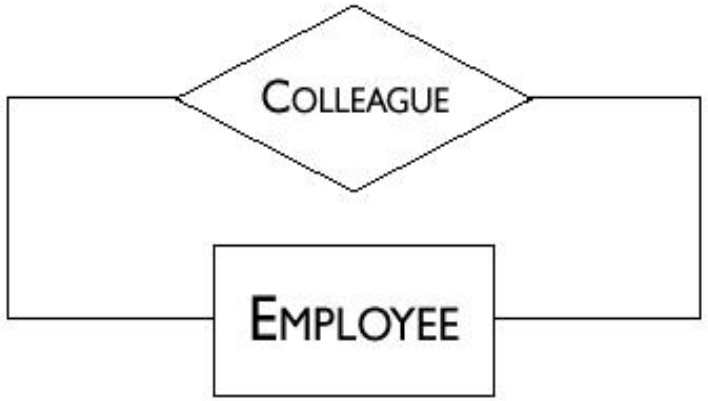
Các thể hiện của Rooms

Quan hệ đệ quy (Recursive)

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

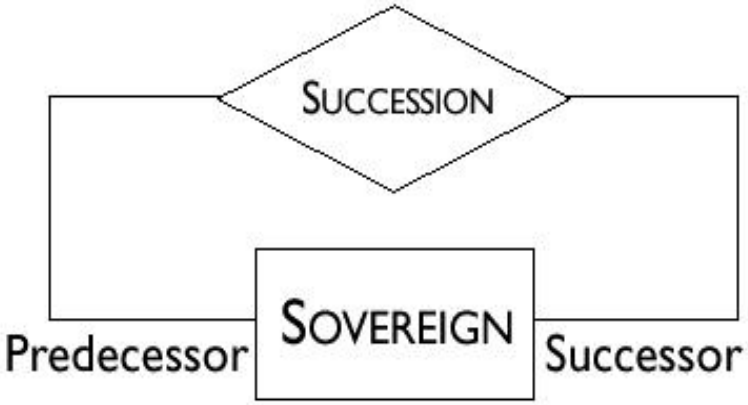
Một thực thể có thể có quan hệ với chính nó…

- **E.g… Thực thể Nhân viên (Empoyee) có quan hệ đồng nghiệp (colleague) với chính nó.**



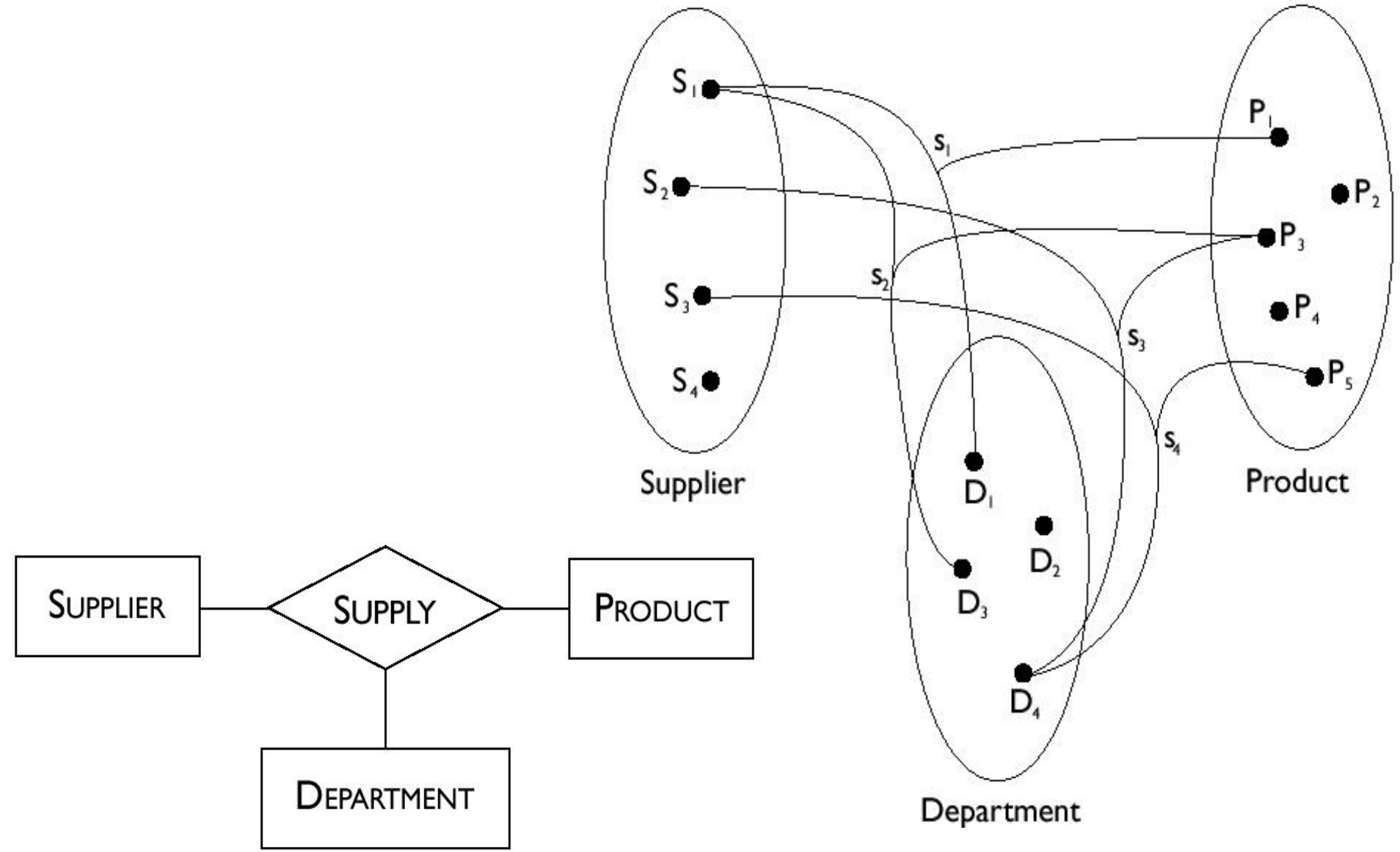
Nếu quan hệ không đối xứng…

- **…Cần định nghĩa hai vai trò mà mỗi thực thể đóng trong quan hệ.**
- **E.g … Thực thể Quốc vương (Sovereign) có quan hệ nối ngôi (Succession) với chính nó, nhưng cần định nghĩa hai vai trò tiền nhiệm (Predecessor) và kế nhiệm (successor) khác nhau cho quan hệ.**

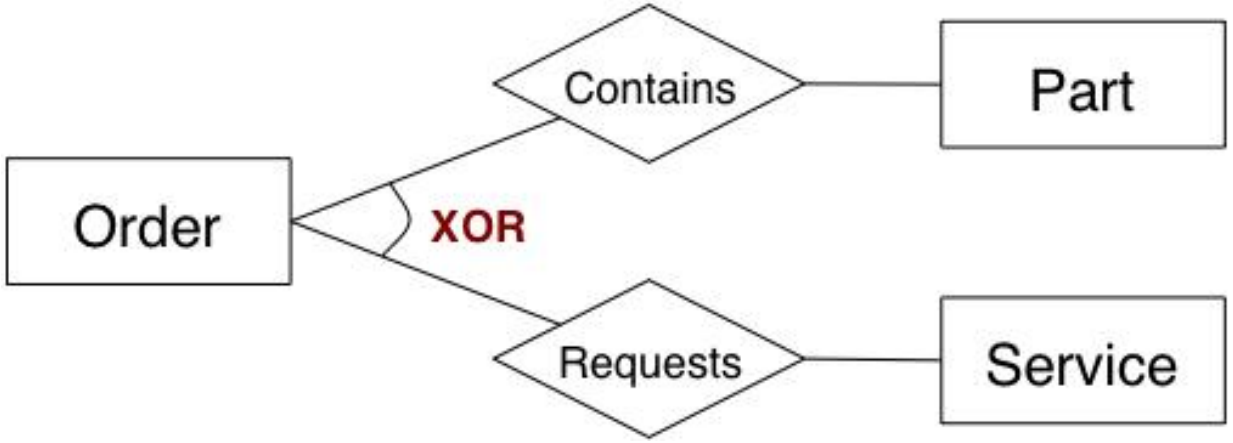


Quan hệ liên kết ba (Ternary)

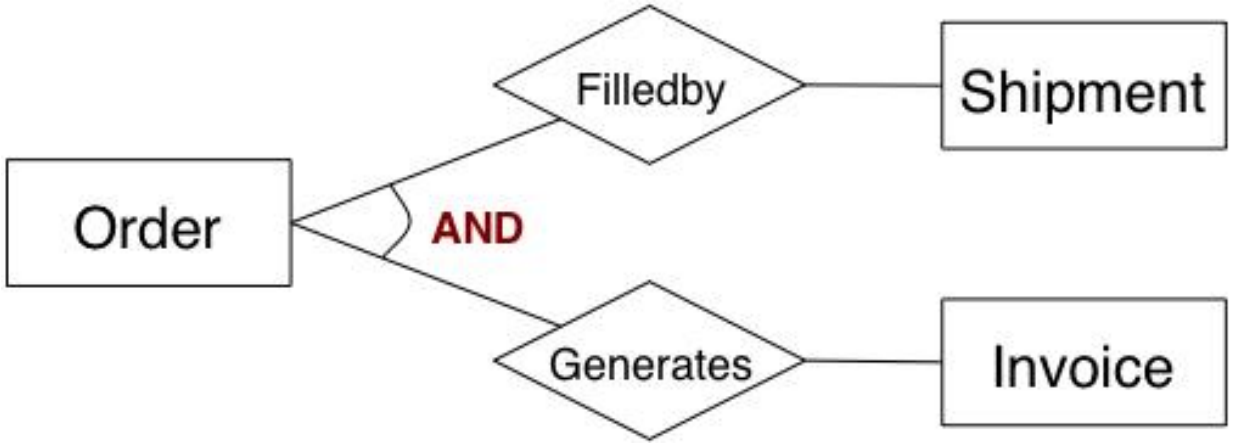
Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999



Quan hệ AND/XOR



"Mỗi *đơn hàng* (Order) hoặc *chứa các món hàng* (contains a part) hoặc *yêu cầu dịch vụ* (requests a service), nhưng không phải cả hai"



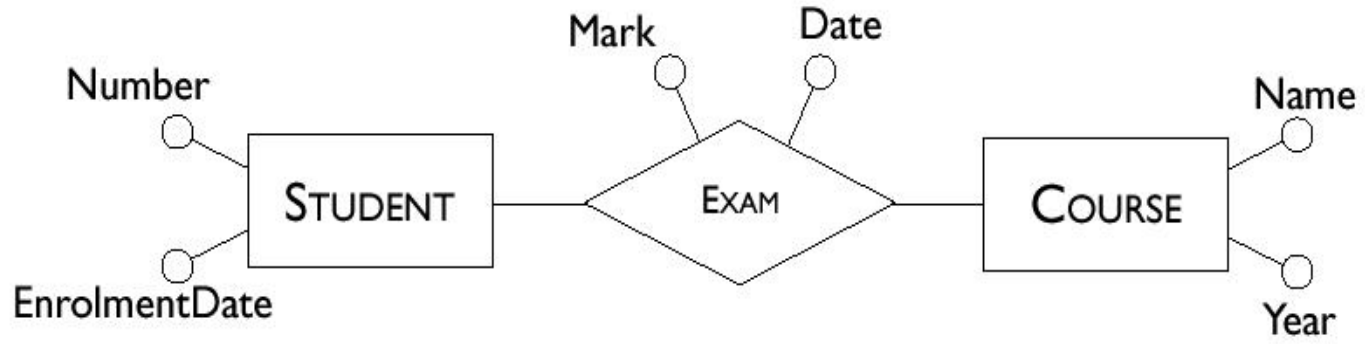
"Đối với một *đơn hàng* (Order), bất cứ khi nào phát sinh một *hóa đơn* (invoice) thì cũng sẽ có một *đợt chuyển hàng* (shipment) được thực hiện và cả hai đều là bắt buộc"

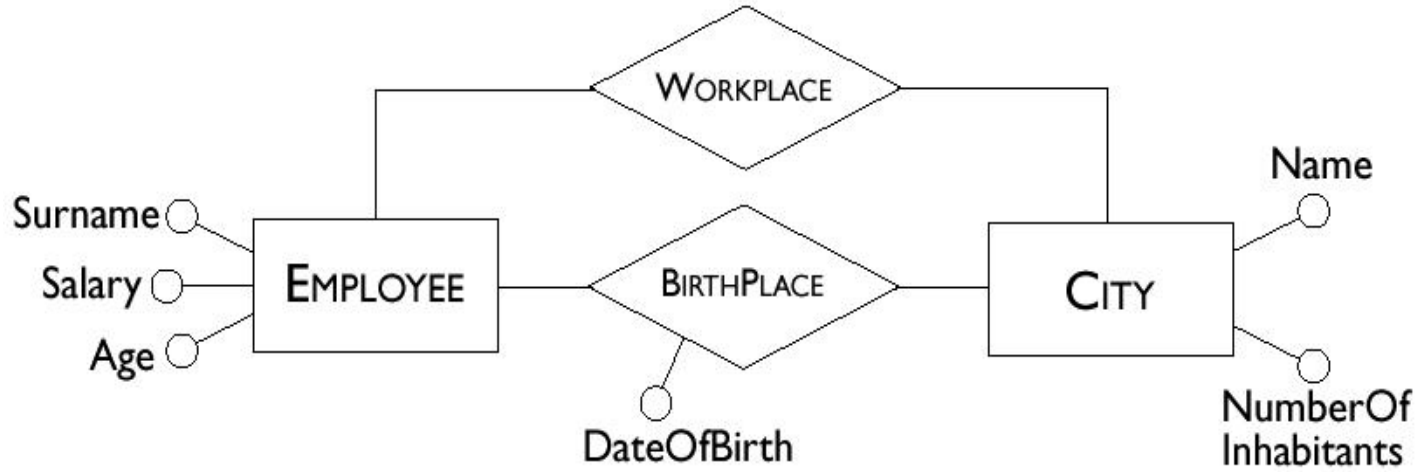
Thuộc tính

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

Liên kết với mỗi thể hiện của một thực thể (hoặc một quan hệ) là một giá trị thuộc về một tập hợp (phạm vi của thuộc tính - attribute).

- **Phạm vi xác định các giá trị có thể nhận được cho thuộc tính.**

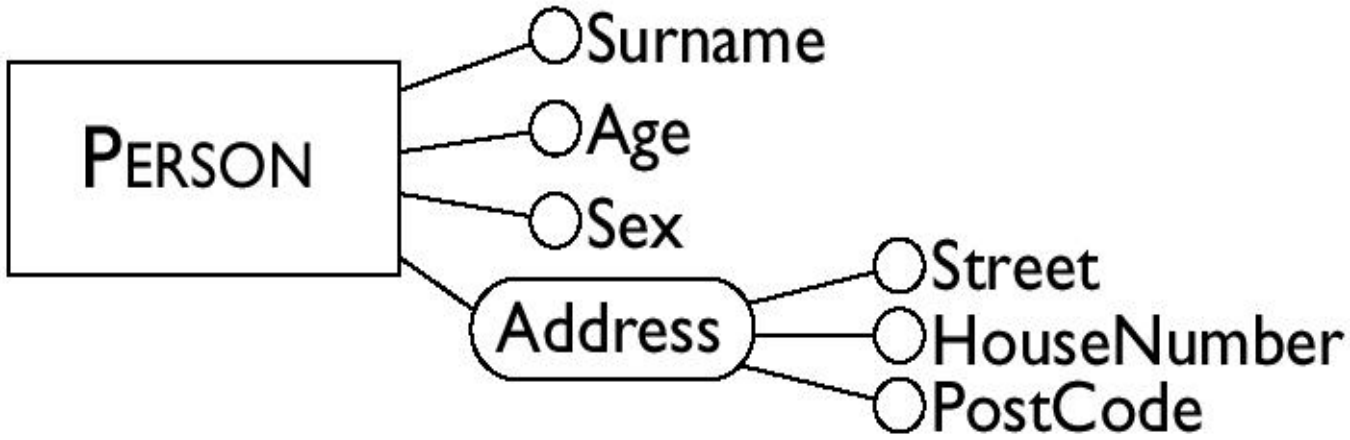




Thuộc tính hợp thành (Composite Attributes)

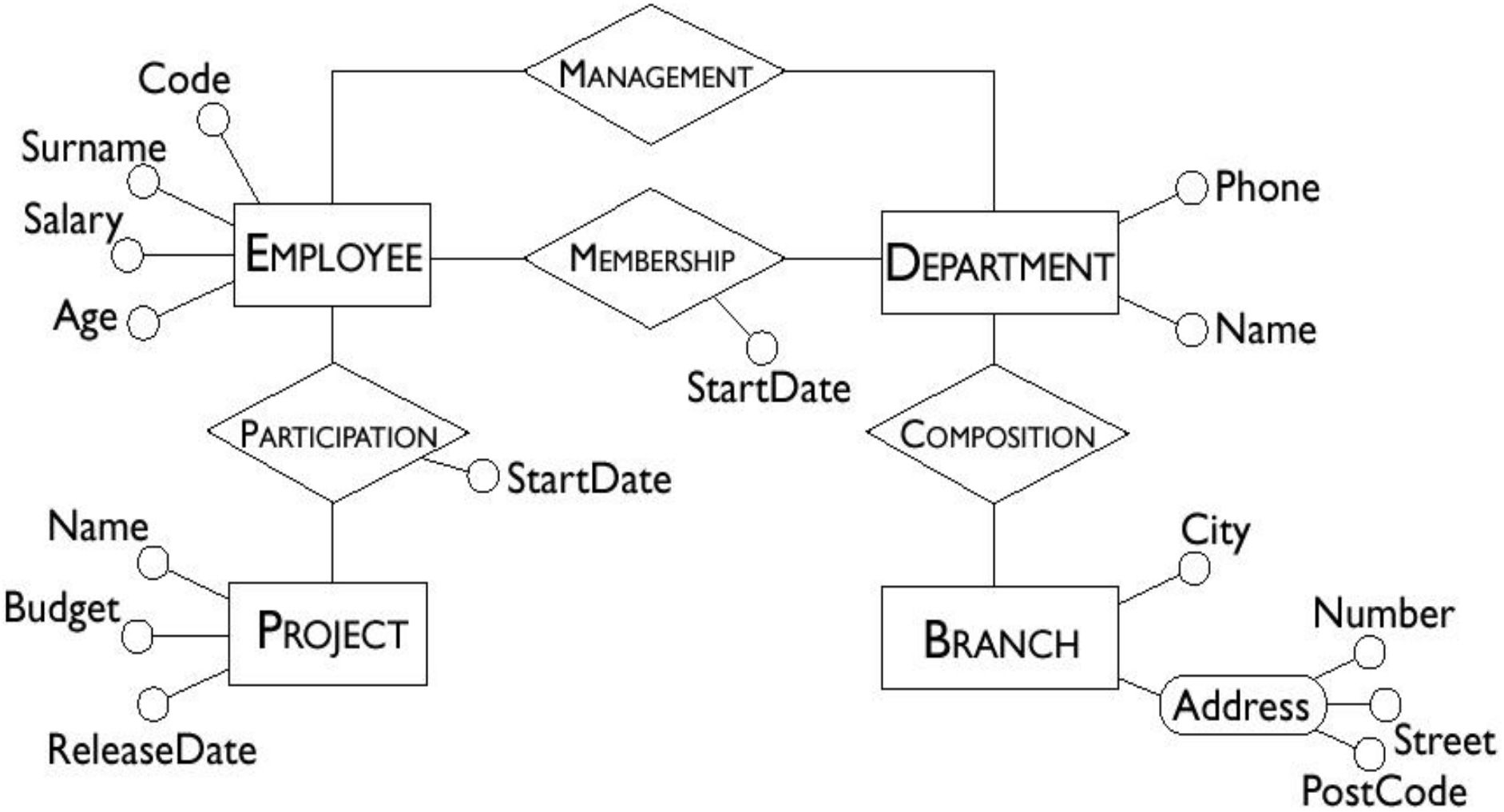
Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

Nhóm thuộc tính của cùng thực thể hoặc quan hệ có ý nghĩa liên kết hoặc cách dùng gần nhau.



Lược đồ với các thuộc tính

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999



Bản số (Cardinalities)

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

Bản số ràng buộc sự tham gia vào quan hệ.

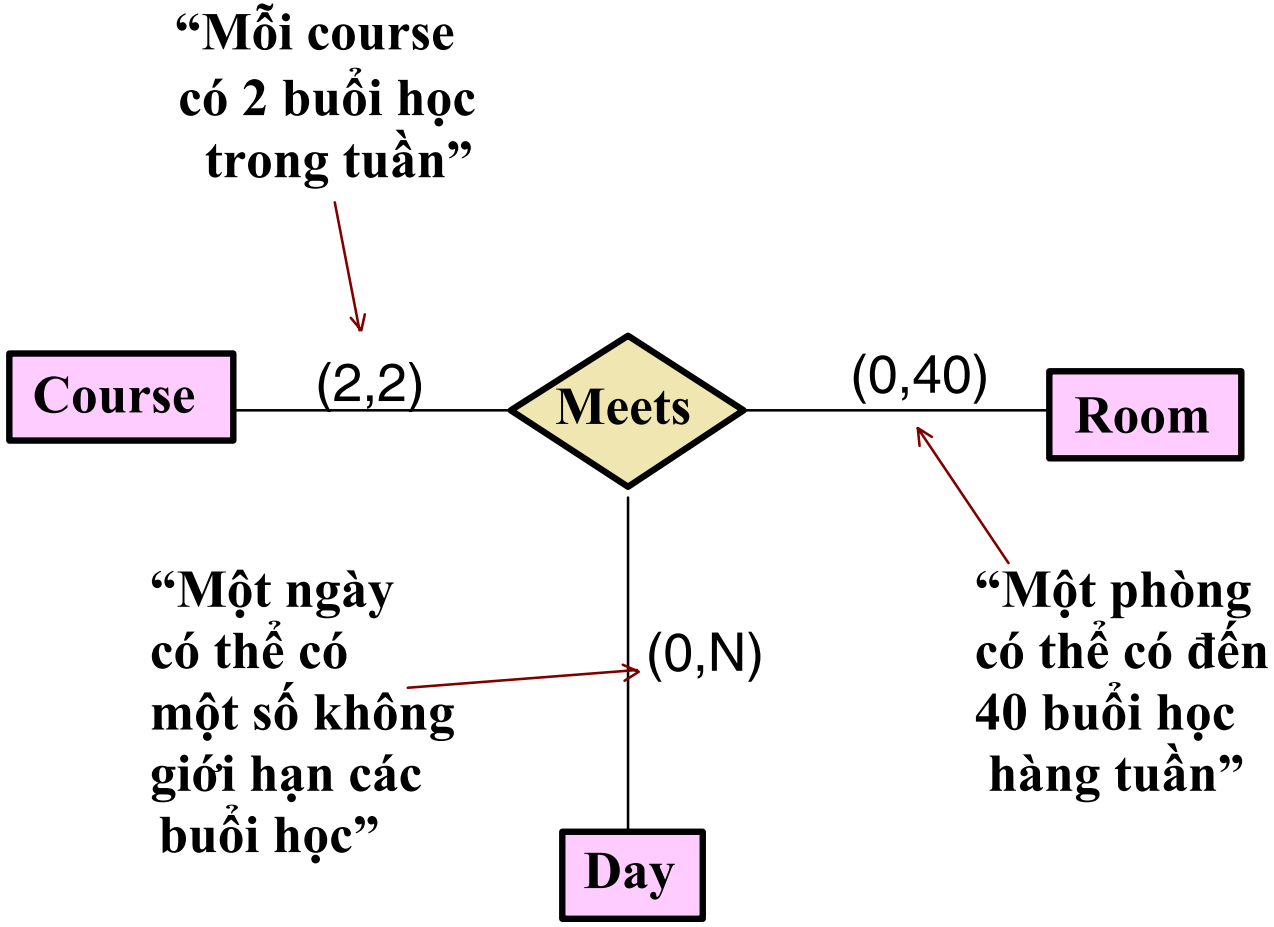
- **Là số tối đa và số tối thiểu của các thể hiện quan hệ mà trong đó một thể hiện của thực thể có thể tham gia vào.**
- **E.g.**

Bản số là mọi cặp số nguyên không âm (a,b)

- **$a \leq b$.**
- **Nếu a=0 thì sự tham gia của thực thể vào quan hệ là tùy ý**
- **Nếu a=1 thì sự tham gia của thực thể vào quan hệ là bắt buộc.**
- **Nếu b=1 thì mỗi thể hiện của thực thể hầu như là liên kết với một thể hiện của quan hệ.**
- **Nếu b="N" thì mỗi thể hiện của thực thể liên kết với một số tùy ý thể hiện của quan hệ.**

Ví dụ về bản số

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

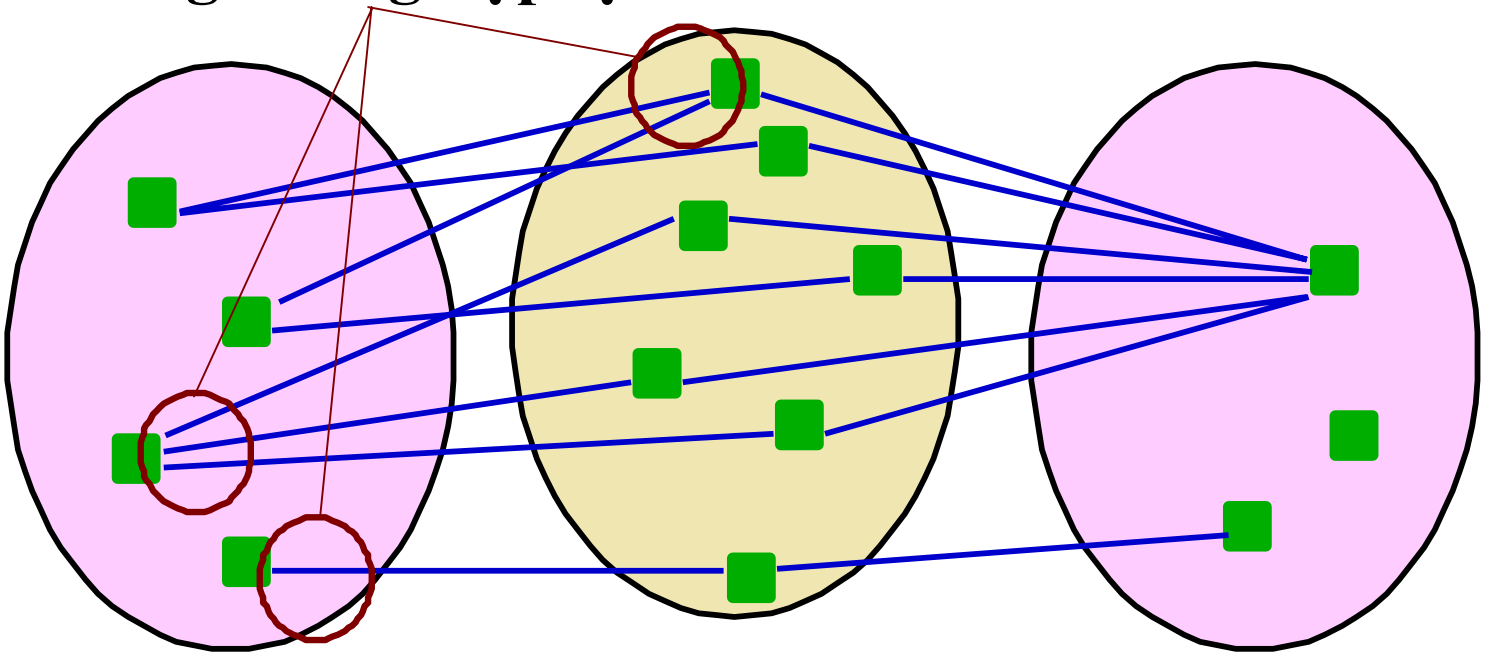


Dẫn chứng về sơ đồ ER

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

Một sơ đồ ER mô tả hiện trạng có thể có trong thế giới thực bằng việc mô hình hóa.

Dẫn chứng không hợp lệ

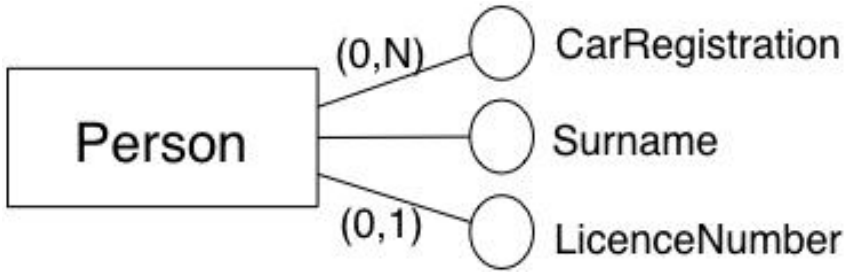


Bản số của thuộc tính

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

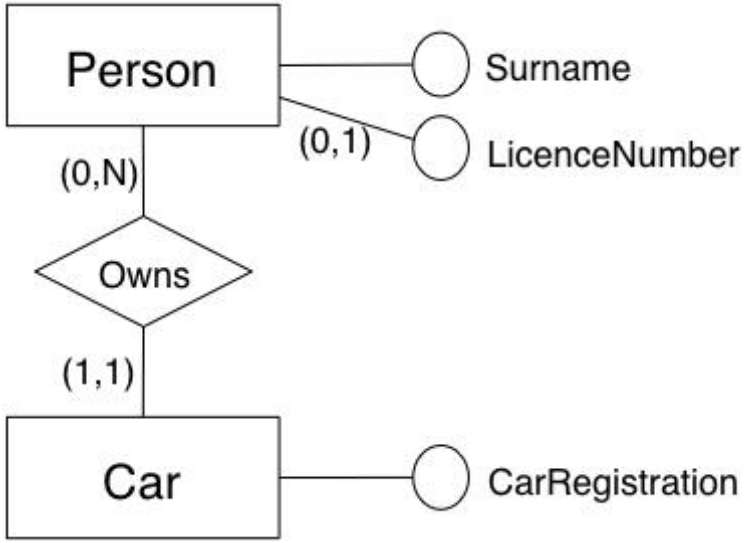
Các thuộc tính cũng có thể có bản số

- **Để mô tả giá trị tối thiểu và tối đa của thuộc tính liên kết với mỗi thể hiện của một thực thể hoặc một liên kết.**
- **Bản số mặc định là (1,1)**
- **Các thuộc tính tùy chọn có bản số là (0,1)**



Các bản số thuộc tính đa giá trị thì rất khó giải quyết

- **Việc mô hình hóa thường sẽ tốt hơn bằng cách thêm vào thực thể các liên kết với quan hệ 1- nhiều (hoặc nhiều-nhiều).**



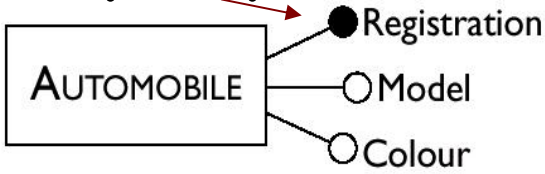
Khóa xác định (Identifiers) ("keys")

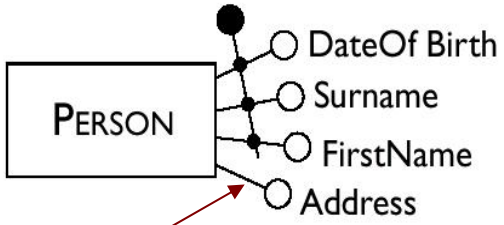
Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

Thế nào là thể hiện dùng xác định duy nhất một thực thể?

- **Một khóa xác định có thể được tạo thành bởi một hoặc nhiều thuộc tính của chính thực thể.**
- **Nếu các thuộc tính của một thực thể không đủ đáp ứng để xác định các thể hiện một cách rõ ràng, các thực thể khác có thể chứa trong sự xác định.**
- **Một quan hệ được xác định bởi khóa xác định của tất cả các thực thể có quan hệ**
  - **E.g. khóa xác định cho quan hệ (Person-) Owns(-Car) là một kết hợp của khóa xác định Person và Car.**

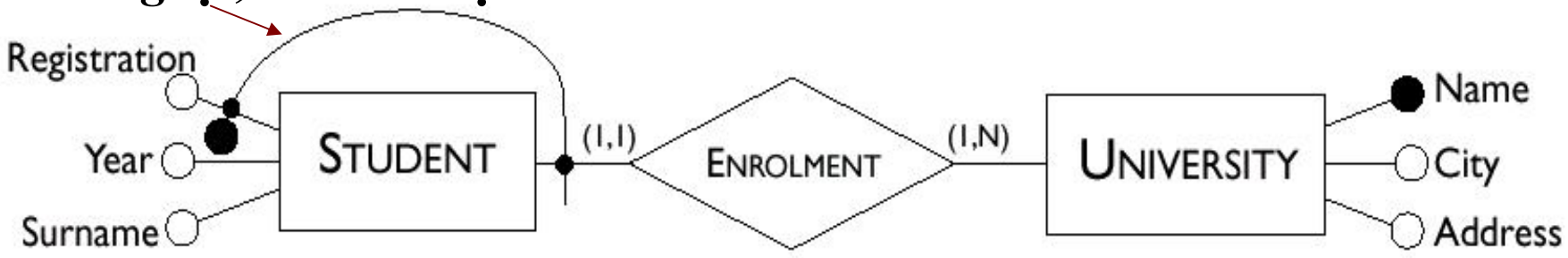
khóa nội, một thuộc tính





khóa nội, nhiều thuộc tính

khóa ngoại, nhiều thuộc tính



Các lưu ý về Khóa xác định

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

Khóa và bản số:

- **Một khóa xác định có thể bao gồm một hoặc nhiều thuộc tính, với điều kiện mỗi trong chúng có bản số (1,1)**
- **Một khóa ngoại (external identifier) có thể chứa một hoặc nhiều thực thể, với điều kiện mỗi trong chúng là một thành viên của quan hệ mà trong đó thực thể tham gia được xác định với bản số (1,1)**

Các chu trình

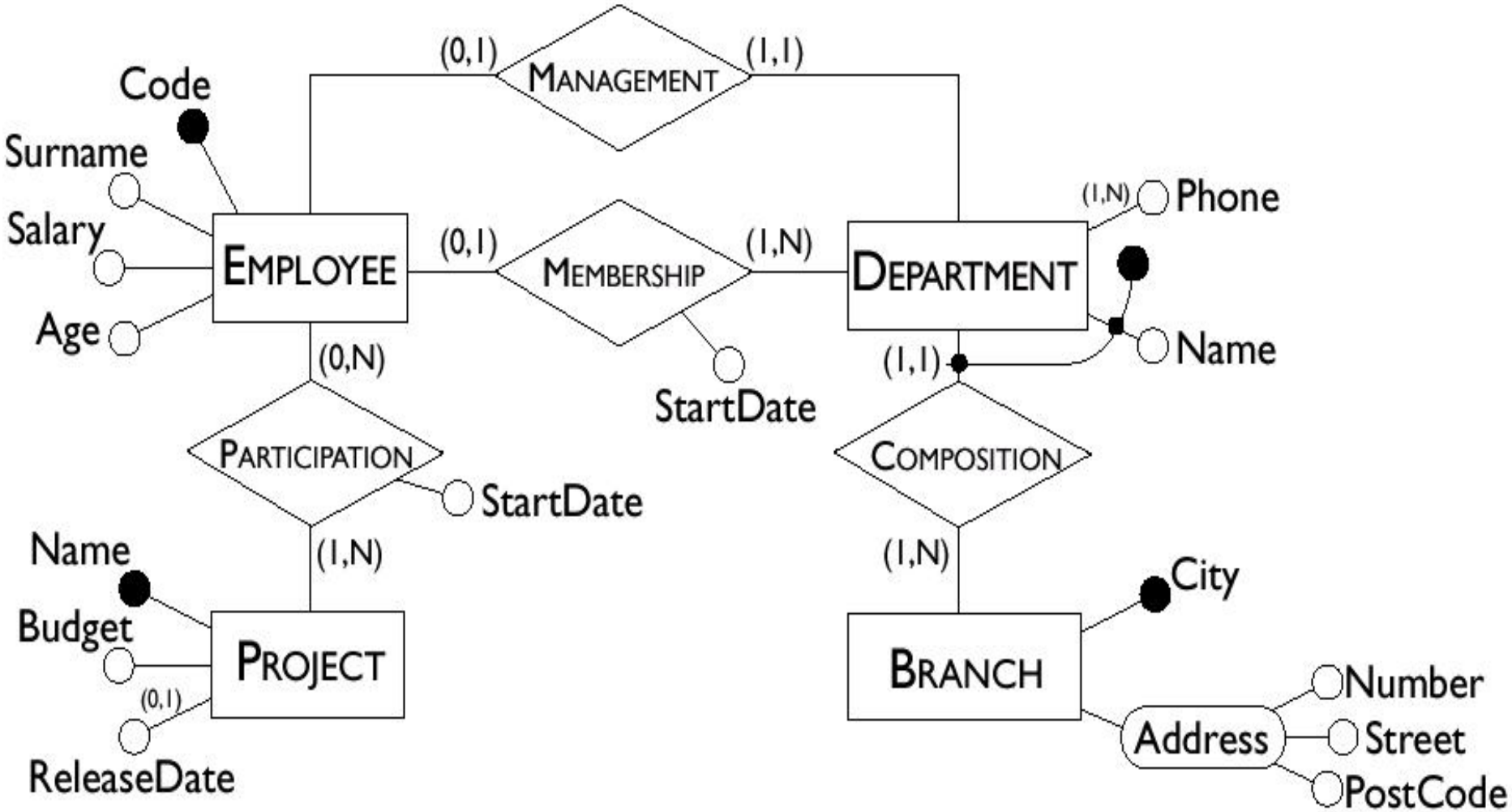
- **Một khóa ngoại có thể bao gồm một thực thể mà nó luân phiên gọi một khóa ngoại khác, chừng nào mà các vòng lặp không sinh nữa;**

Đa khóa

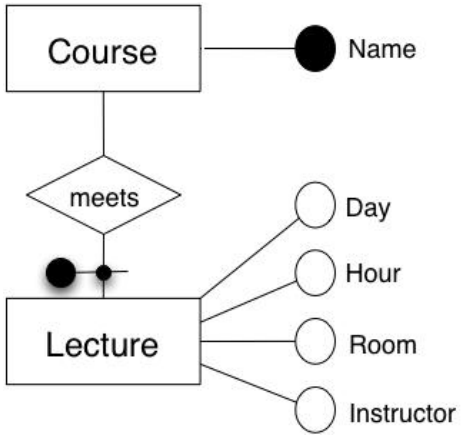
- **Mỗi thực thể phải có ít nhất một khóa xác định (khóa nội hoặc khóa ngoại)**
- **Một thực thể có thể có nhiều hơn một khóa xác định**
  - **Chú ý : nếu có nhiều hơn một khóa xác định, thì các thuộc tính và thực thể chứa trong một sự xác định có thể là tùy chọn (bản số tối thiểu bằng 0).**

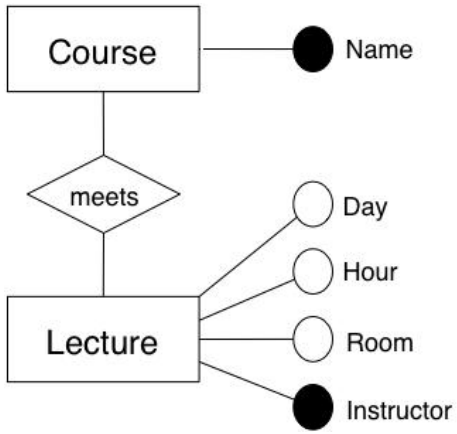
Lược đồ với các khóa

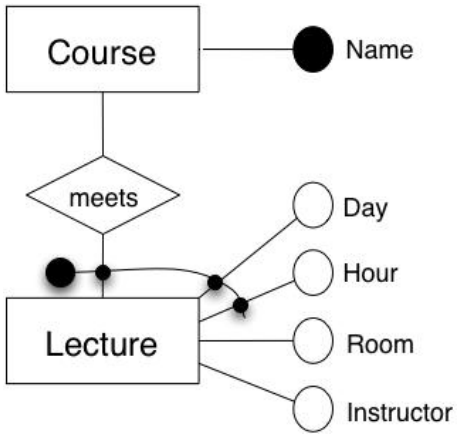
Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

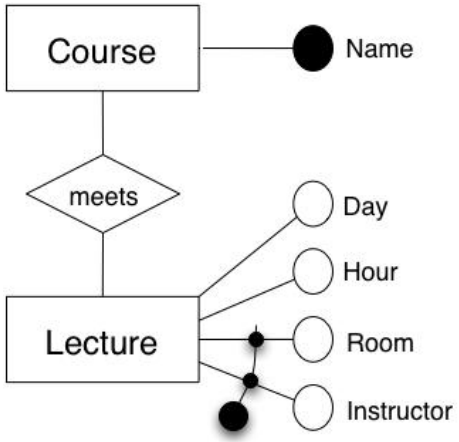


Hiểu rõ việc chọn khóa





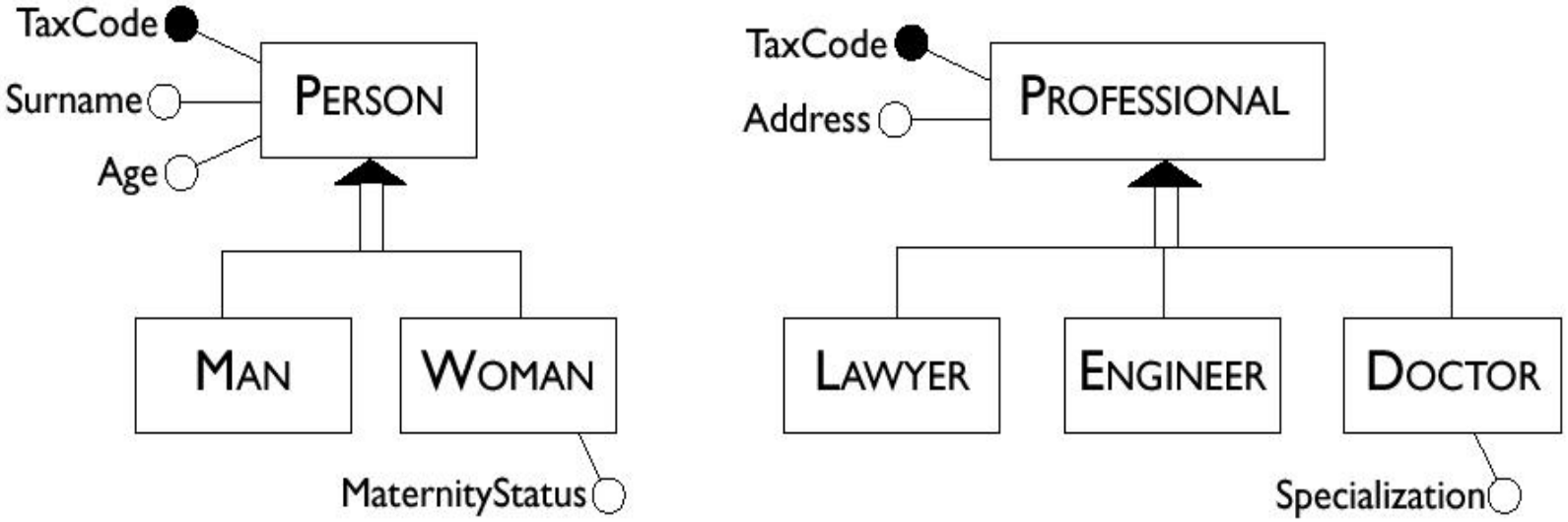




Khái quát hóa (Generalizations)

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

Chỉ ra quan hệ "là-một" ("is-a") giữa các thực thể



Khái quát hóa:

- **Mỗi thể hiện của một thực thể con cũng là một thể hiện của thực thể cha**
- **Mỗi đặc tính của thực thể cha (thuộc tính, khóa xác định, quan hệ hoặc khái quát hóa khác) cũng là một đặc tính của thực thể con.**

Các dạng khái quát hóa

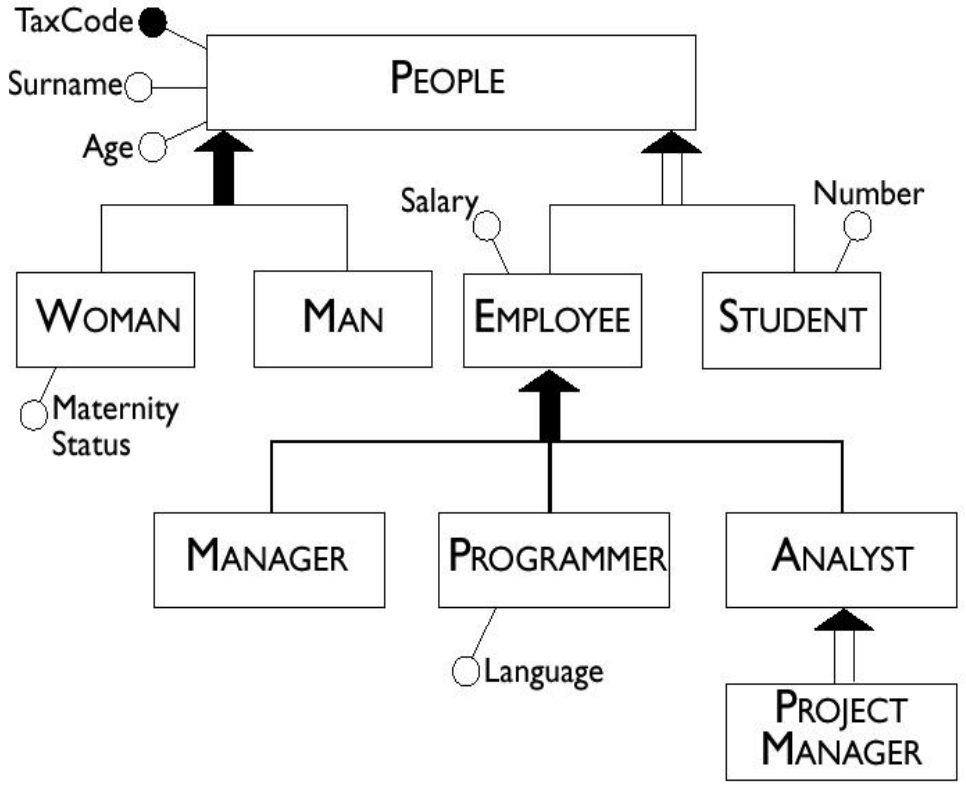
Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999

Khái quát hóa hoàn toàn (Total generalizations):

- … Mỗi thể hiện của thực thể cha là một thể hiện của một trong số các con của nó.
- Chỉ ra bằng dạng mũi tên tô đậm

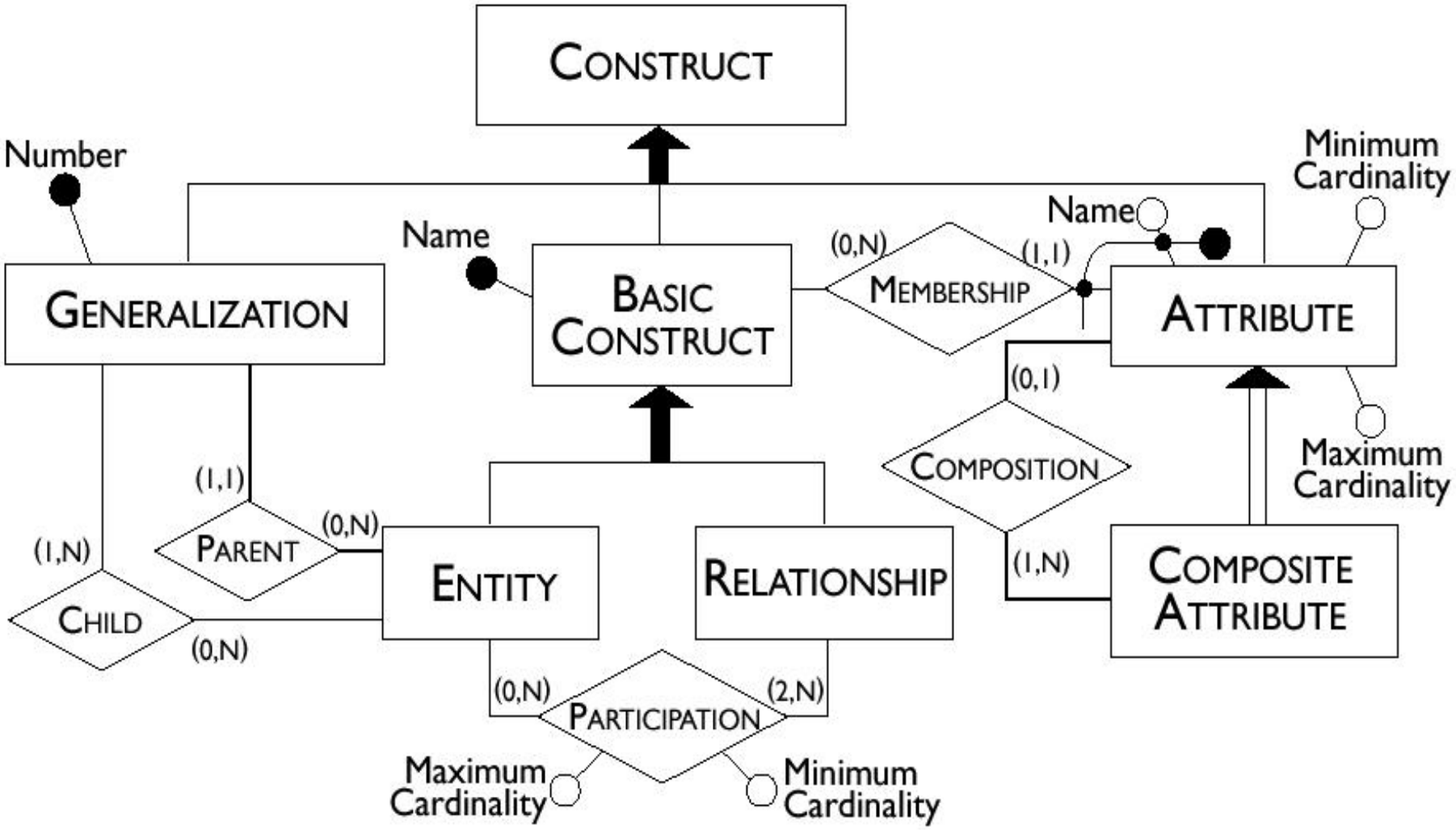
Khái quát hóa loại trừ (Exclusive generalizations):

- … Mỗi thể hiện của thực thể cha có ít nhất một thể hiện của một trong số các con của nó.
- Chỉ ra bằng dạng mũi tên rỗng



Mô hình E-R Meta-Model (như sơ đồ E-R)

Adapted from chapter 5 of Atzeni et al, "Database Systems" McGraw Hill, 1999



Lecture 08:
Mô hình hướng đối tượng

- **Phân tích hướng đối tượng**
  - Phân tích cơ sở
  - Định nghĩa lớp (Classes)
  - Các thuộc tính (Attributes) và phương thức (Operations)
- **Biểu đồ lớp UML (Class Diagrams)**
  - Quan hệ kết hợp (Associations)
  - Tính bội/Bản số (Multiplicity)
  - Quan hệ tập hợp (Aggregation)
  - Quan hệ hợp thành (Composition)
  - Quan hệ thừa kế (Generalization)

Phân tích hướng đối tượng

Background

- Mô hình yêu cầu trong thuật ngữ của các đối tượng và dịch vụ mà chúng cung cấp
- Phát sinh thiết kế hướng đối tượng
  - Được áp dụng để mô hình hóa lĩnh vực ứng dụng hơn là chương trình

Động cơ

- OO (được kiến nghị) là quá 'tự nhiên'
  - Khi triển khai một hệ thống, các chức năng thực thi của nó cần được thay đổi thường hơn là các đối tượng đang hoạt động trên nó…
  - …một mô hình dựa trên các đối tượng (hơn là các chức năng) sẽ rất ổn định…
  - …vì thế, có kiến nghị rằng các thiết kế hướng đối tượng có thể sẽ còn tiếp tục được duy trì
- OO nhấn mạnh sự quan trọng của giao tiếp giữa các đối tượng một cách rõ ràng.
  - đã được so sánh với sự mơ hồ của các quan hệ dòng dữ liệu

NOTE: Áp dụng OO cho kỹ nghệ yêu cầu vì nó là một công cụ mô hình hóa. Song, chúng ta đang mô hình hóa các thực thể trong lĩnh vực chứ không phải thiết kế hệ thống mới.

UML là gì ?

UML (Unified Modeling Language)

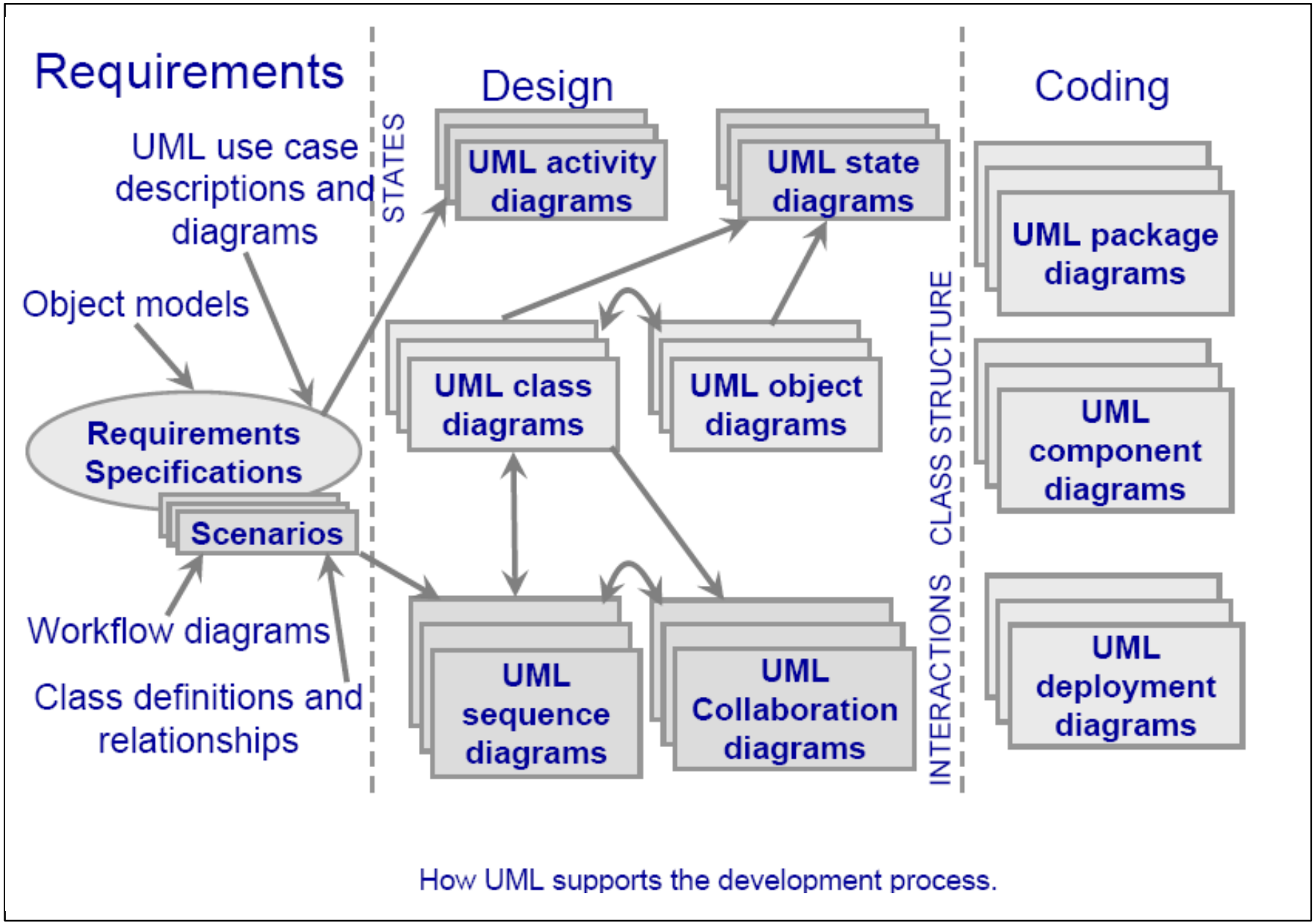
- **Một công nghệ chuẩn cho việc mô hình hóa phần mềm hướng đối tượng**
- **Là kết quả của sự thống nhất hệ thống ký hiệu của 3 phương pháp hướng đối tượng tiêu biểu :**
  - **OMT (James Rumbaugh) -** Object Modeling Technique
  - **OOSE (Ivar Jacobson)** - Object-Oriented Software Enginering
  - **Booch91 (Grady Booch)**

Được hỗ trợ bởi một số CASE tools:

- **Rational ROSE**
- **TogetherJ**

Bạn có thể mô hình 80% của hầu hết vấn đề bằng cách dùng chỉ khoảng 20% UML

Chúng ta học 20% đó



How UML supports the development process.

Gần như mọi thứ đều có thể là object…

Source: *Adapted from Pressman, 1994, p242*

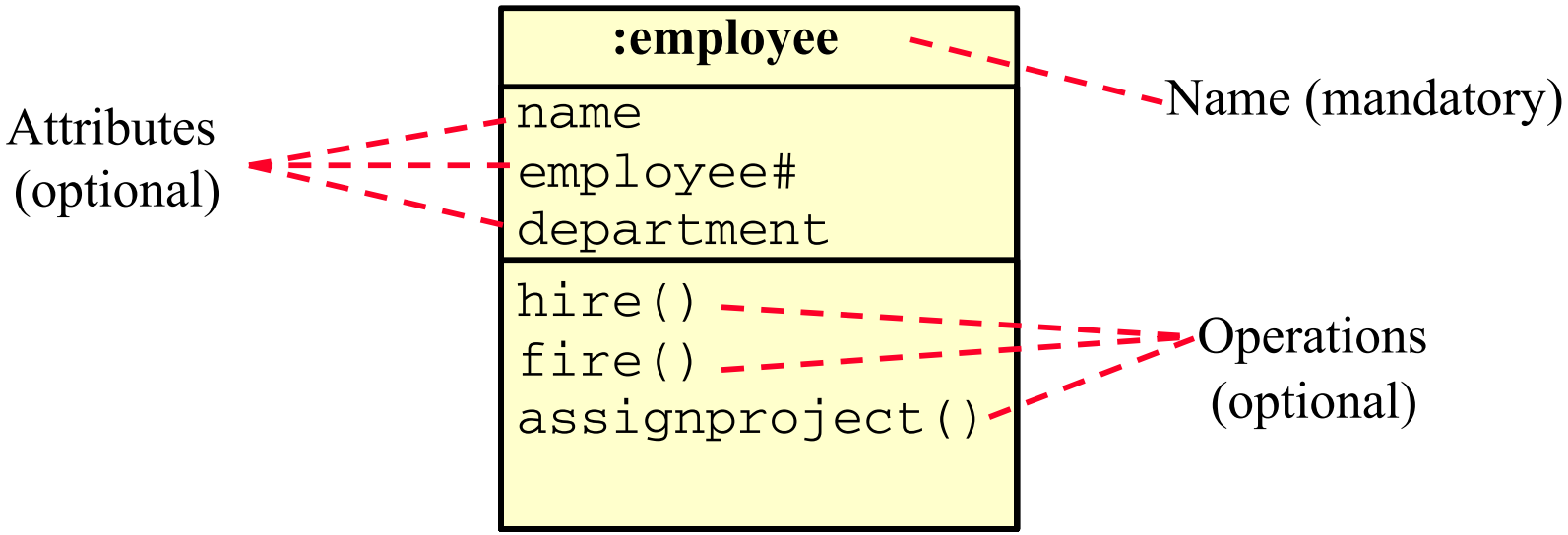
- **Các thực thể bên ngoài**
  - **…tương tác với hệ thống đang được mô hình hóa**
    - **E.g. people, devices, other systems**
- **Các vật thể**
  - **…là những phần của lĩnh vực đang được mô hình hóa**
    - **E.g. báo cáo, màn hình, tín hiệu, etc.**
- **Việc xảy ra hoặc sự kiện**
  - **…xuất hiện trong ngữ cảnh của hệ thống**
    - **E.g. chuyển giao tài nguyên, hành động kiểm soát, etc.**
- **Vai diễn**
  - **được đóng bởi những người đang tương tác với hệ thống**
- **Các thành phần tổ chức**
  - **có liên quan tới ứng dụng**
    - **E.g. phân chia, nhóm, đội, etc.**
- **Nơi chốn**
  - **…thiết lập ngữ cảnh của vấn đề đang được mô hình hóa**
    - **E.g. nhà máy, bến tàu, etc.**
- **Các cấu trúc**
  - **định nghĩa một lớp hay nhóm các objects**
    - **E.g. bộ cảm biến, bộ bánh xe, máy tính, etc.**

Một số thứ không thể là objects:

- **các thủ tục (e.g. in ấn, đảo ngược, etc)**
- **các thuộc tính (e.g. màu xanh, 50Mb, etc)**

Lớp (classes) là gì ?

- **Một *lớp* mô tả một nhóm các đối tượng (objects) với :**
  - Các đặc tính tương tự (thuộc tính - attributes),
  - Cùng hành vi ứng xử (phương thức - operations),
  - Quan hệ như nhau đối với các object khác.
  - Và có chung ngữ nghĩa ("semantics").
- **Ví dụ**
  - **Nhân viên (employee): có 1 tên (name), mã số nhân viên (employee#) và bộ phận trực thuộc (department); một nhân viên thì có thể được thuê (hired), và bị sa thải (fired); mỗi nhân viên làm việc trong một hay nhiều dự án.**



Tìm lớp (Classes)

- **Tìm *lớp* từ dữ liệu nguồn:**
  - **Tìm các danh từ và cụm danh từ trong mô tả vấn đề của các đối tác**
    - **chứa trong mô hình nếu họ giải thích một cách tự nhiên hoặc cấu trúc thông tin trong ứng dụng.**
- **Tìm *lớp* từ các nguồn khác:**
  - **Xem xét các thông tin nền tảng;**
  - **Những người dùng và các đối tác khác;**
  - **Các mẫu phân tích;**
- **Sẽ rất tốt nếu ban đầu có nhiều ứng viên cho *lớp***
  - **Bạn có thể bỏ chúng ngay sau đó nếu chúng hóa ra không hữu ích**
  - **Quyết định dứt khoát loại bỏ các *lớp* thì tốt hơn là chỉ suy nghĩ về điều đó.**

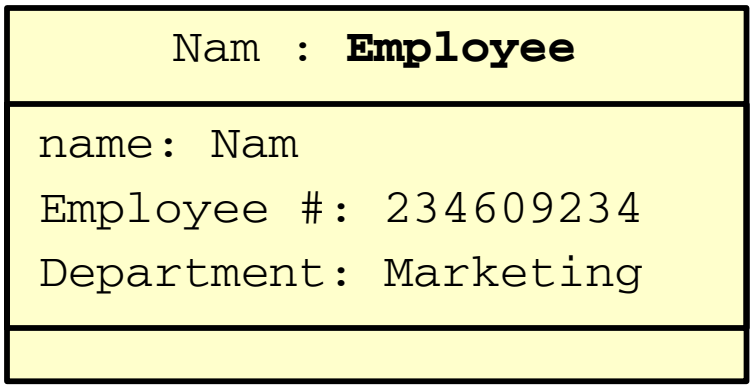
Chọn lựa lớp

Loại bỏ *lớp* là một khái niệm khi :

- Không nằm trong phạm vi phân tích;
- Tham chiếu đến toàn bộ hệ thống;
- Trùng với các lớp khác;
- Có quá nhiều mơ hồ hoặc quá chi tiết
  - e.g. có quá nhiều hoặc quá ít thể hiện (instances)
- Tiêu chuẩn của Coad & Yourdon's:
  - Giữ lại thông tin : Hệ thống sẽ còn nhớ thông tin về các lớp này của objects?
  - Các dịch vụ cần thiết : Objects trong lớp này có nhận biết các phương thức làm thay đổi giá trị các thuộc tính của chúng ?
  - Đa thuộc tính (Multiple Atributes): Nếu lớp chỉ có duy nhất một thuộc tính, nó có thể đại diện tốt hơn một thuộc tính của lớp khác
  - Thuộc tính chung (Common Attributes): Lớp có chứa các thuộc tính mà có thể chia sẻ với tất cả các thể hiện của đối tượng đó không ?
  - Phương thức chung (Common Operations): Lớp có chứa các phương thức mà có thể chia sẻ với tất cả các thể hiện của đối tượng đó không ?
- Các thực thể bên ngoài phát sinh hoặc thu nhận các thông tin chủ yếu cho hệ thống nên được đặt thành *lớp*.

Objects vs. Classes

- **Các thể hiện của một lớp được gọi là đối tượng**
  - **Một đối tượng được trình bày như sau:**



  - **Hai đối tượng khác nhau có thể có các giá trị thuộc tính giống nhau (như hai người với tên và địa chỉ giống nhau)**
- **Đối tượng có quan hệ kết hợp (associations) với đối tượng khác**
  - **E.g. Nam :employee thì có quan hệ kết hợp với đối tượng Mekong1000 :project**
  - **Nhưng chúng ta sẽ tìm hiểu quan hệ này ở cấp độ *lớp !***
  - **Chú ý : Phải bảo đảm rằng thuộc tính thì kết hợp với đúng lớp**

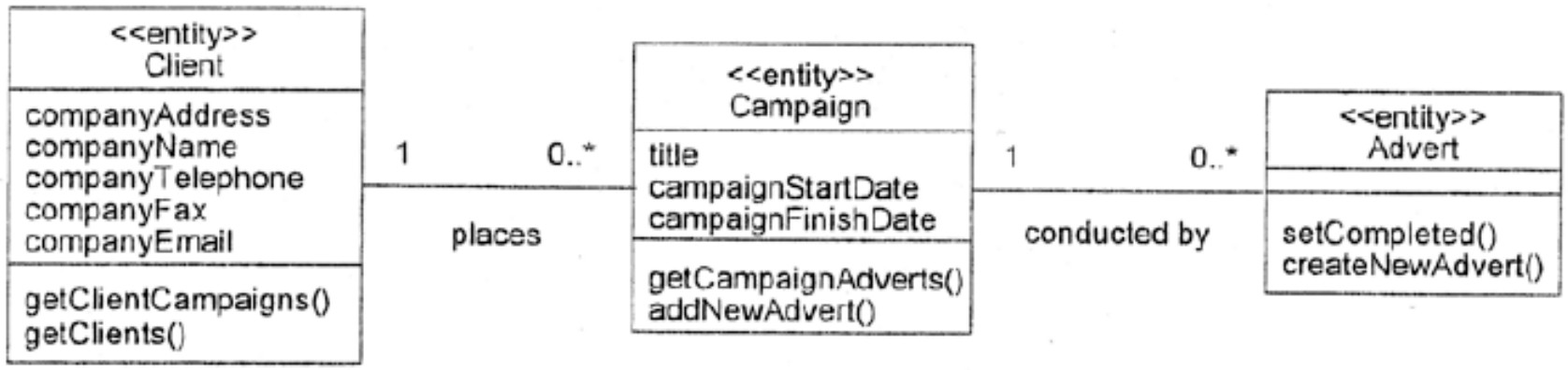
➢**E.g. bạn không muốn cả managerName và manager# đều là thuộc tính của Project !?**

Quan hệ kết hợp (Associations)

Đối tượng không tồn tại độc lập với những cái khác

- Một quan hệ (relationship) diễn tả một sự kết hợp trong số những cái khác.
- Trong UML, có nhiều kiểu quan hệ khác nhau:
  - Quan hệ kết hợp (Association)
  - Quan hệ tập hợp (Aggregation) và Quan hệ hợp thành (Composition)
  - Quan hệ thừa kế (Generalization)
  - Quan hệ phụ thuộc (Dependency)
  - Quan hệ hiện thực hóa (Realization)
- Chú ý : Hai quan hệ cuối không hữu ích trong quá trình phân tích yêu cầu

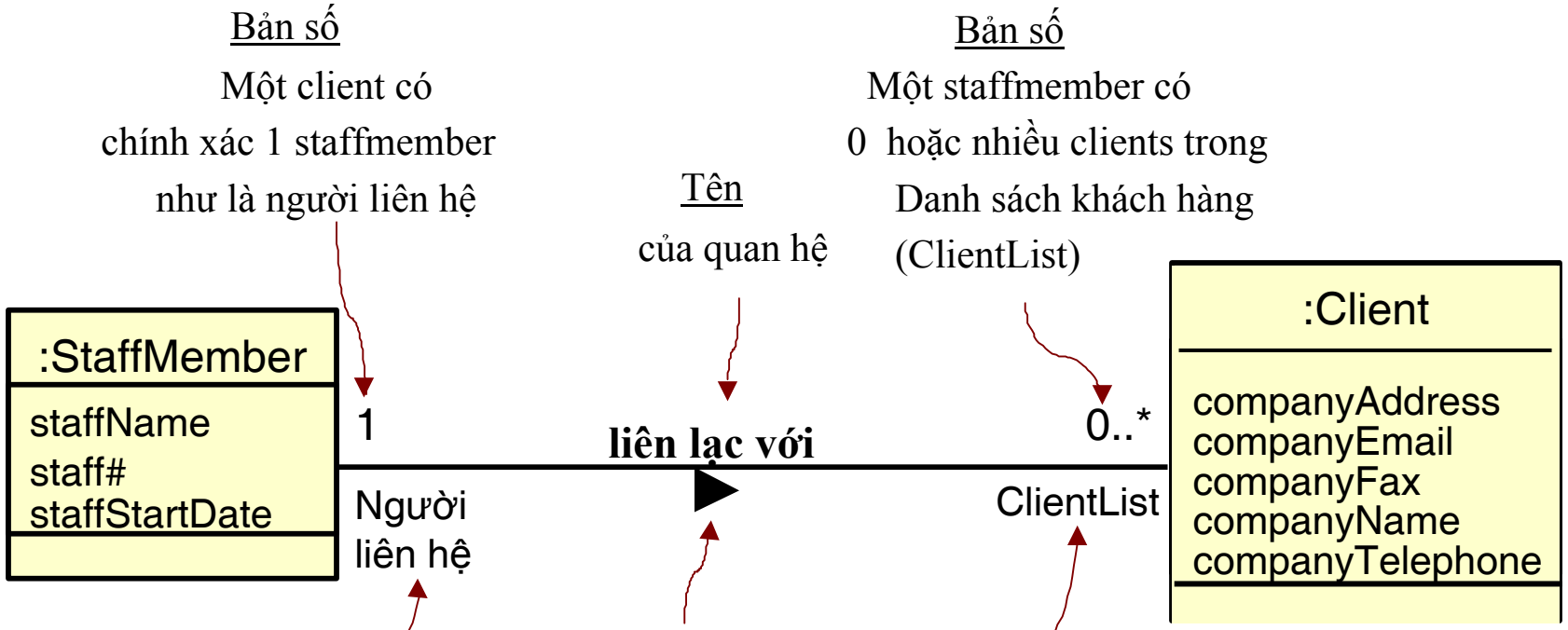
Sơ đồ lớp chỉ rõ các lớp và mối quan hệ giữa chúng



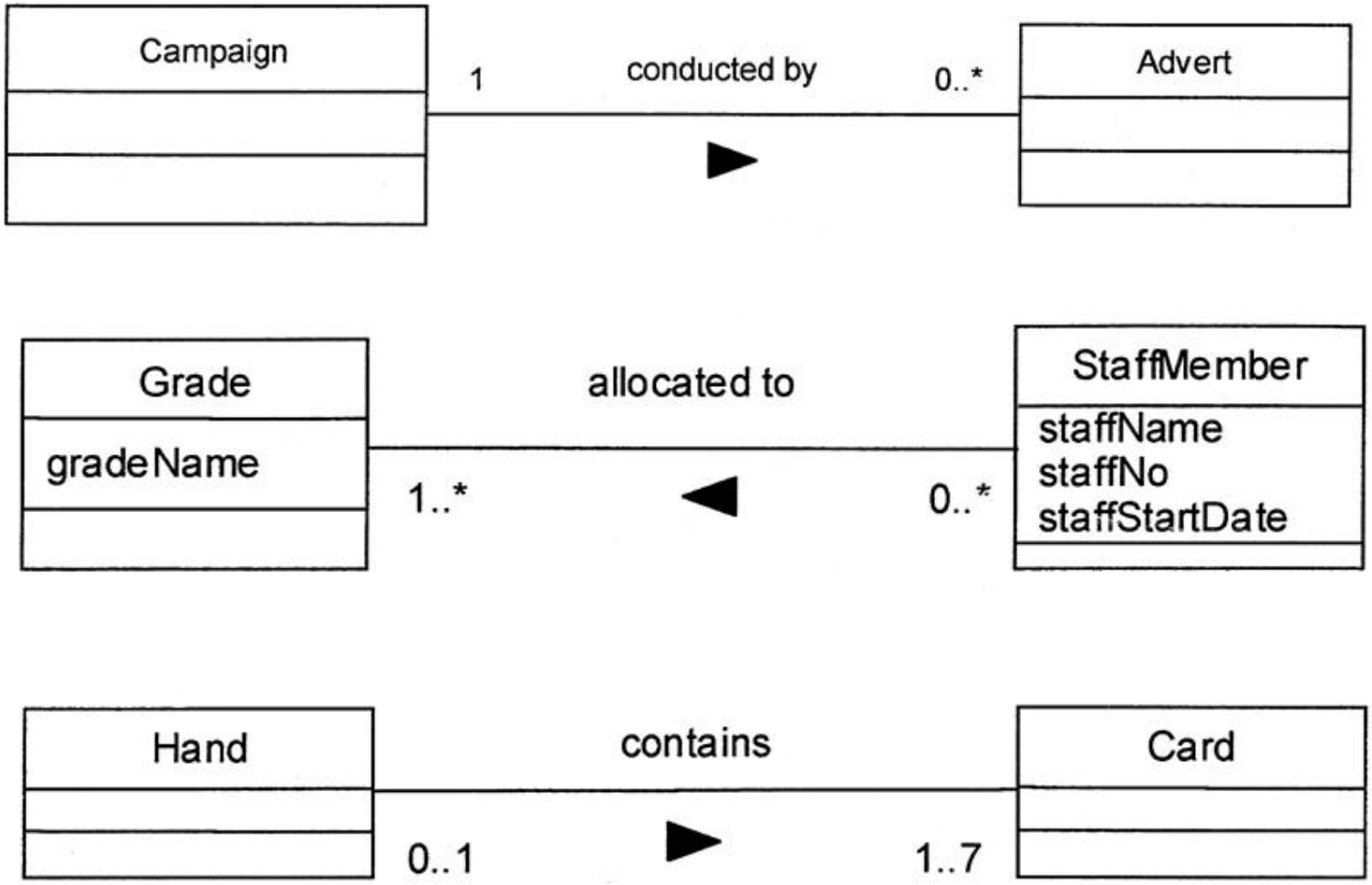
Bản số (Multiplicity) của Quan hệ kết hợp

- **Hỏi các câu hỏi về quan hệ kết hợp:**
  - **Một cuộc vận động (campaign) có thể tồn tại mà không có thành viên trong Ban quản lý hay không ?**
    - **Nếu có, thì quan hệ kết hợp này sẽ là một tùy chọn trong Ban quản lý - 0 hoặc nhiều (0..*)**
    - **Nếu không, thì nó là tùy chọn – 1 hoặc nhiều (1..*)**
    - **Nếu nó cần được quản lý bởi 1 và chỉ 1 thành viên trong Ban – chính xác 1 (1)**
  - **Một câu hỏi khác của quan hệ kết hợp?**
    - **Mỗi thành viên trong Ban quản lý có cần phải quản lý chính xác chỉ một cuộc vận động không?**
    - **Không. Vì thế bản số chính xác là 0 hoặc nhiều (0..*)**
- **Một số ví dụ biểu diễn của bản số:**
  - **Tùy chọn (0 hoặc 1)** **0..1**
  - **Chính xác 1** **1** **= 1..1**
  - **0 hoặc nhiều** **0..*** **= ***
  - **1 hoặc nhiều** **1..***
  - **Một vùng giá trị** **2..6**

Quan hệ kết hợp lớp

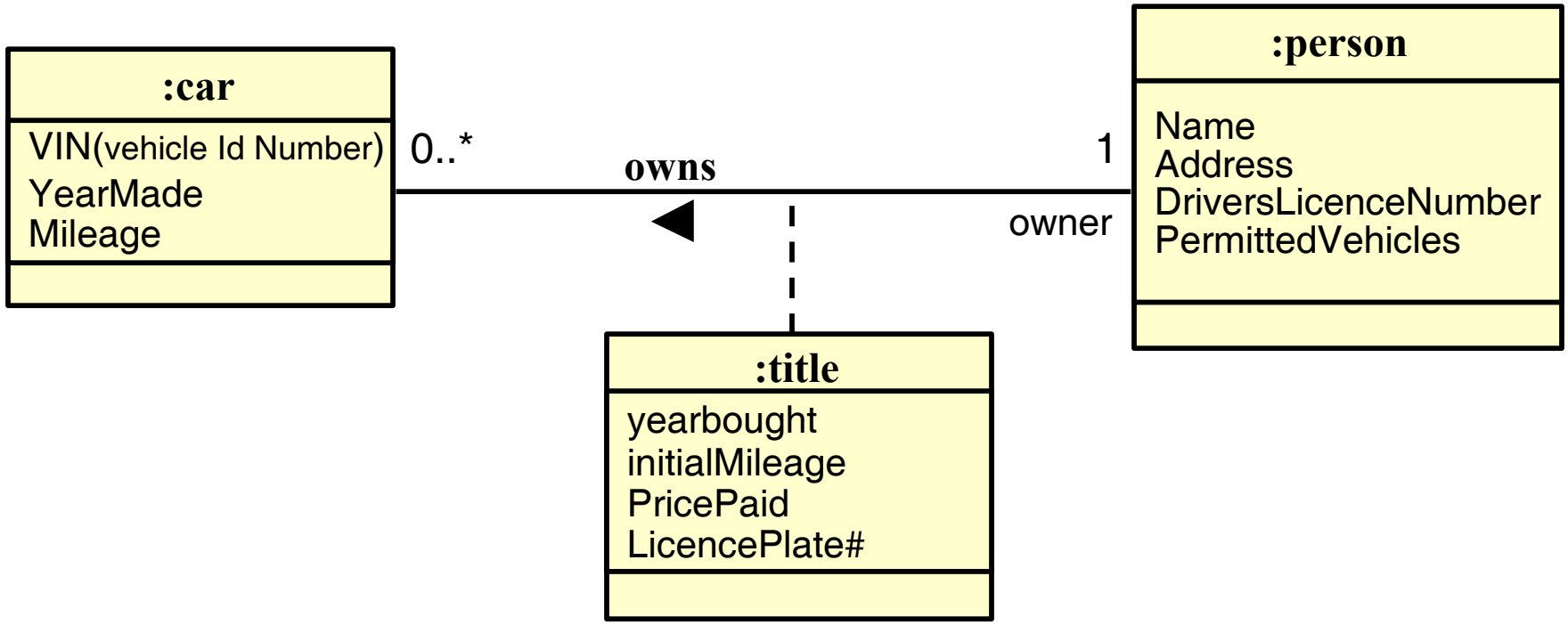


Các ví dụ khác



Các lớp quan hệ kết hợp

- **Đôi khi có quan hệ kết hợp của một lớp với chính quan hệ**
  - **… bởi vì chúng ta cần giữ lại thông tin về quan hệ kết hợp**
  - **… và những thông tin này thì không còn tồn tại trong các lớp vào cuối của quan hệ kết hợp**
    - **E.g. "Chủ quyền" (title) là một đối tượng dùng mô tả thông tin về mối quan hệ giữa người chủ và chiếc xe của cô ấy**



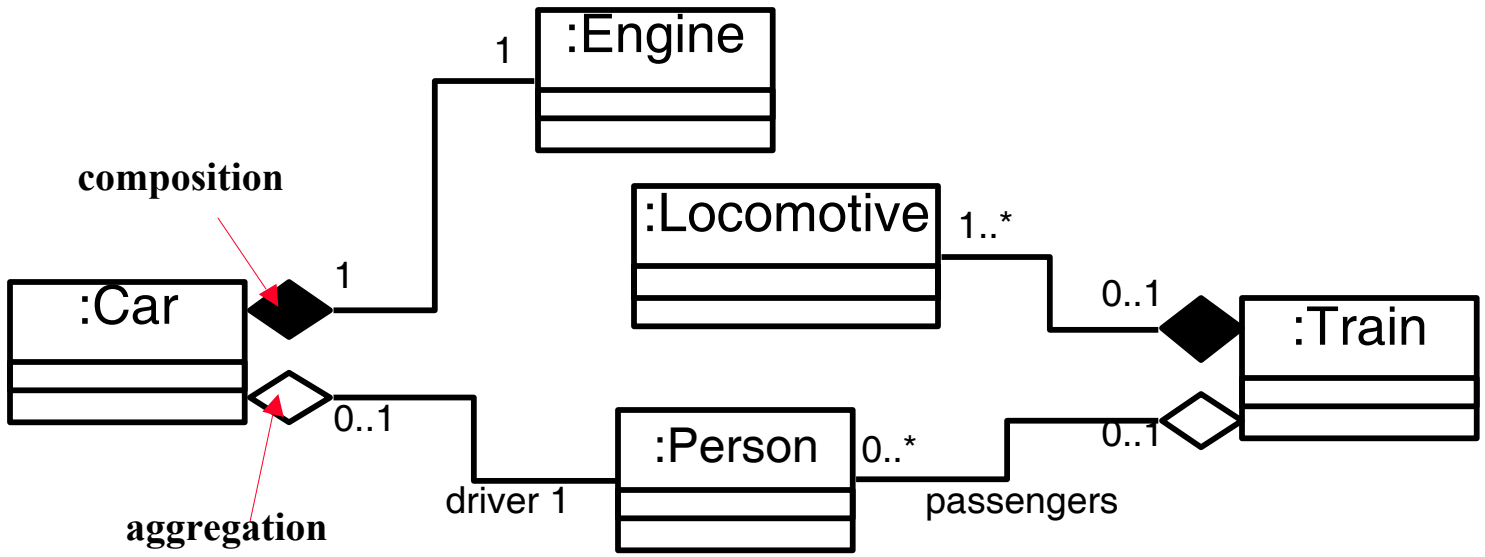
Quan hệ tập hợp và Quan hệ hợp thành

Quan hệ tập hợp (Aggregation)

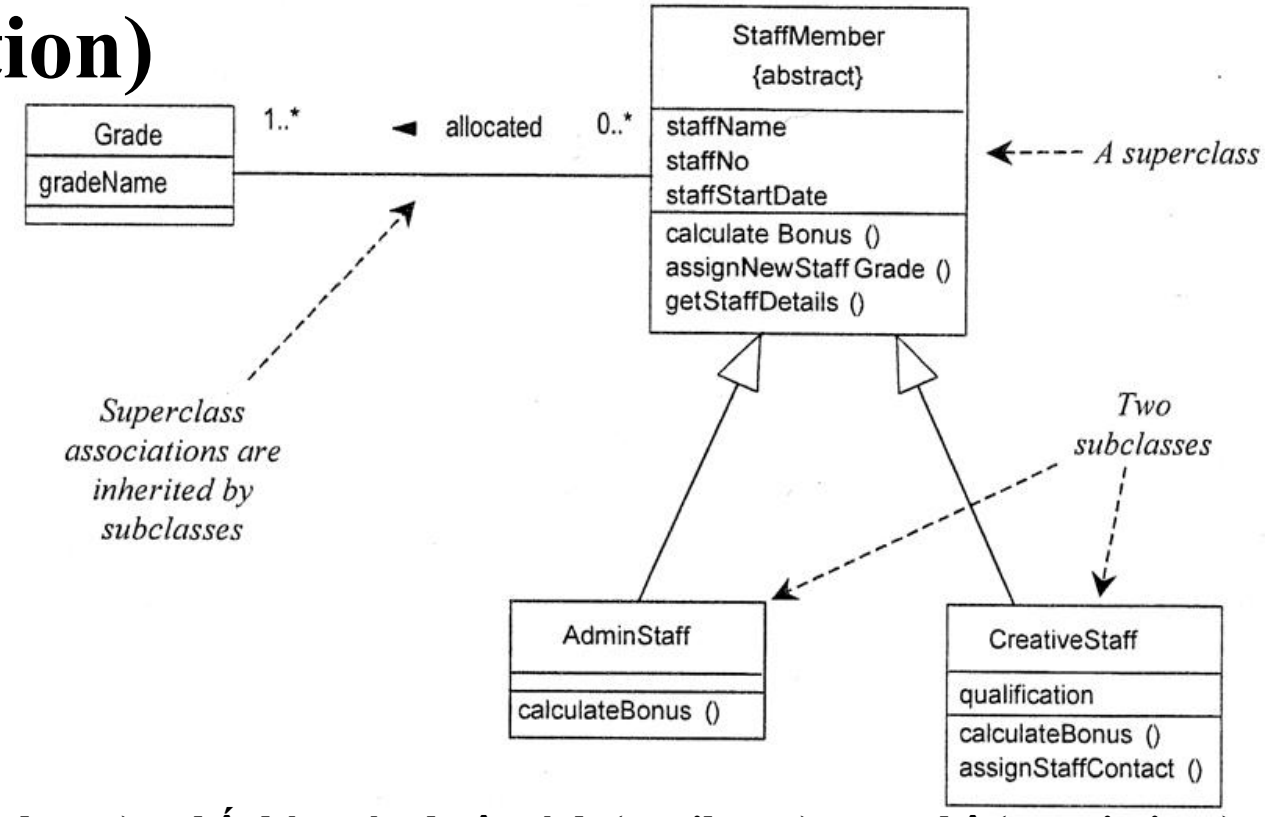
- Đây là một quan hệ "Có một (Has-a)" hay "Toàn thể/Bộ phận (Whole/part)"

Quan hệ hợp thành (Composition)

- Là dạng mạnh hơn của quan hệ tập hợp dùng chứng tỏ quyền sở hữu:
  - nếu đối tượng toàn thể bị hủy, đối tượng bộ phận cũng bị hủy theo.
  - đối tượng toàn thể chịu trách nhiệm về sự sắp xếp các thành phần của nó.



Quan hệ kế thừa (Generalization)



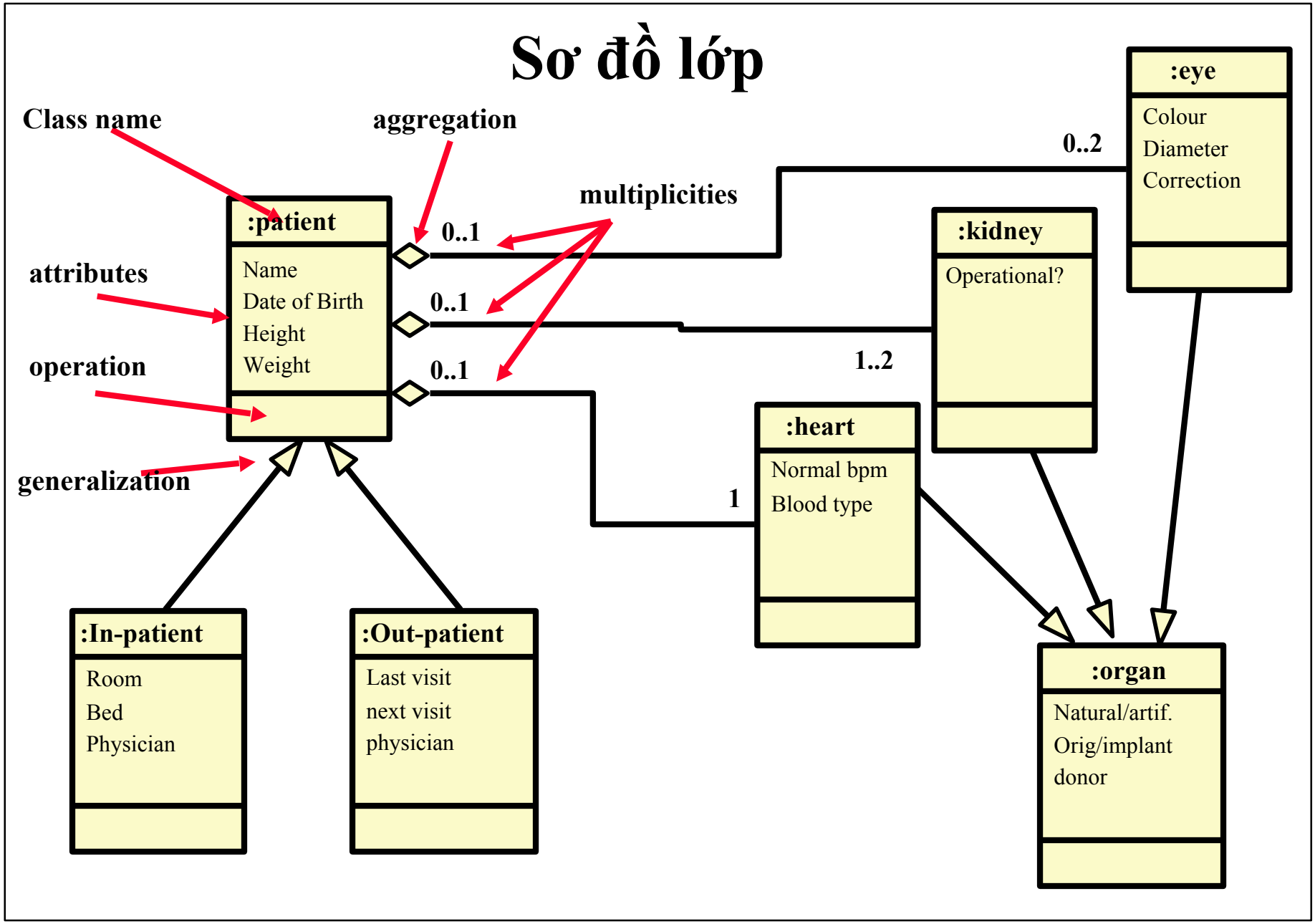
Chú ý :

- Các lớp con (subclasses) sẽ kế thừa các thuộc tính (attributes), quan hệ (associations) và phương thức (operations) từ lớp cha (superclass)
- Một lớp con có thể bỏ qua một khía cạnh kế thừa
  - e.g. AdminStaff & CreativeStaff có các phương pháp khác nhau cho cách tính điểm thưởng
- Các lớp cha có thể khai báo {abstract}, nghĩa là chúng không có thể hiện (instances)
  - Chứng tỏ rằng các lớp con bao phủ tất cả
  - e.g. không có bộ phận nào khác hơn AdminStaff và CreativeStaff

Nói thêm về Quan hệ kế thừa

- **Ích lợi của quan hệ kế thừa**
  - Có thể dễ dàng thêm vào các lớp con mới khi có thay đổi tổ chức
- **Tìm kiếm quan hệ kế thừa theo 2 cách:**
  - Trên xuống (Top Down)
    - Bạn có một lớp, và việc khảo sát nó có thể được chia nhỏ ra
    - Hoặc bạn có một quan hệ kết hợp diễn tả một "loại của (kind of)" quan hệ
    - E.g. "Hầu hết công việc của chúng ta là quảng cáo cho báo chí – các tờ báo và tạp chí, cũng như là trên các pano quảng cáo và videos"
  - Dưới lên (Bottom Up)
    - Bạn cần lưu ý sự tương tự giữa các lớp mà bạn định nghĩa
    - E.g. "Chúng ta có nhiều sách và dĩa CD trong Thư viện, nhưng tất cả chúng đã được ghi số phân loại theo hệ thống Dewey, và tất cả đều có thể được cho mượn và dặn trước"
- **Nhưng đừng kế thừa chỉ những ích lợi của nó**
  - Bảo đảm rằng mọi thứ trong lớp cha được áp dụng vào lớp con
  - Bảo đảm rằng lớp cha thì hữu ích khi một lớp trong nó được sở hữu hoàn toàn
  - Đừng thêm các lớp con hoặc lớp cha không có liên quan vào phân tích của bạn

Sơ đồ lớp


Class name
aggregation
multiplicities
attributes
operation
generalization
:patient
Name
Date of Birth
Height
Weight
0..1
0..1
0..1
0..2
:eye
Colour
Diameter
Correction
:kidney
Operational?
1..2
:heart
Normal bpm
Blood type
1
:In-patient
Room
Bed
Physician
:Out-patient
Last visit
next visit
physician
:organ
Natural/artif.
Orig/implant
donor

Kết luận

- **Hiểu rõ các đối tượng trong lĩnh vực ứng dụng**
  - **Định nghĩa tất cả các đối tượng mà đối tác nêu ra**
  - **Quyết định đối tượng nào quan trọng cho thiết kế của bạn**
  - **Sơ đồ lớp thiết kế tốt khi:**
    - **Có quan hệ trực quan giữa các đối tượng trong lĩnh vực**
    - **Khảo sát các quy tắc nghiệp vụ và giả thiết thông qua bản số**
    - **Đặc tả cấu trúc của thông tin để (cuối cùng) lưu trữ**
- **OO là một cách thức tốt để khảo sát các chi tiết của vấn đề**
  - **Tránh những chấp vá tự nhiên của cấu trúc phân tích**
  - **Cung cấp một phương thức chặt chẽ để hiểu rõ thực tế**
- **Nhưng cẩn thận…**
  - **Nó lôi cuốn để thiết kế hơn là phân tích vấn đề**
    - **Trong RE, sơ đồ lớp KHÔNG phản ánh các lớp chương trình (e.g. Java)**
  - **Đối với các nhà phân tích, dùng các lược đồ UML như là sự phác họa chứ không phải bản thiết kế**
    - **Tuy nhiên vẫn có thể trở thành bản thiết kế khi được dùng trong một sự đặc tả**

Kết luận: UML vs ERD

Sơ đồ ER tương tự như sơ đồ lớp trong UML

- Sơ đồ lớp nhấn mạnh thứ bậc lớp (class hierarchies) và các phương thức (operaions)
- Sơ đồ R nhấn mạnh các mối quan hệ (relationships) và khóa xác định (identity)

Nhưng chỉ cần một cho việc phân tích mọi vấn đề cho trước !

ER cung cấp nhiều ký hiệu hơn cho khái niệm cơ sở dữ liệu:

- Sơ đồ ER cho phép các quan hệ đa chiều (N-ary relationships)
  - (Sơ đồ lớp UML chỉ cho phép quan hệ hai chiều (binary relationships))
- Sơ đồ ER cho phép các thuộc tính đa giá trị.
- Sơ đồ ER cho phép đặc tả các khóa xác định.

Sự lựa chọn tùy thuộc vào mục tiêu cài đặt:

- Sơ đồ lớp UML cho kiến trúc hướng đối tượng (Object Oriented Architecture)
- Sơ đồ ER cho CSDL quan hệ (Relational Databases)
- Nhưng điều này chỉ quan trọng khi bạn dùng chúng cho bản thiết kế
  - Đối với bản phác thảo, sự tương tự với ký hiệu thì quan trọng hơn

Lecture 09:
Mô hình hóa tương tác hệ thống

- **Tương tác với hệ thống mới**
  - Con người sẽ tương tác với hệ thống như thế nào?
  - Khi nào/Vì sao họ tương tác với hệ thống?
- **Use Cases**
  - Giới thiệu mô hình hoạt vụ (use cases)
  - Định nghĩa tác nhân (actors)
  - Định nghĩa tình huống (cases)
  - Các đặc tính mở rộng
- **Biểu đồ tuần tự (Sequence Diagrams)**
  - Trình tự thời gian của các sự kiện bao gồm trong hoạt vụ (use case)

Mô tả về sự chuyển dịch

Hệ thống Use case sẽ cung cấp những chức năng gì ?

- Con người sẽ tương tác với nó như thế nào?
- Mô tả các chức năng từ hướng nhìn của người dùng

UML Use Cases

- Dùng để chỉ:
  - Các chức năng (functions) được cung cấp bởi hệ thống
  - Tác nhân (actors) nào sẽ dùng những chức năng này
- Mỗi Use Case là:
  - Một mẫu ứng xử mà hệ thống mới được yêu cầu biểu diễn
  - Một trình tự của các hành động có liên quan thực thi bởi một tác nhân và hệ thống thông qua việc đối thoại.

Một tác nhân (actor) là :

- Mọi thứ cần tương tác với hệ thống: Một người (a person); Một vai trò (role) mà những người khác nhau có thể đóng; Một hệ thống khác (bên ngoài)

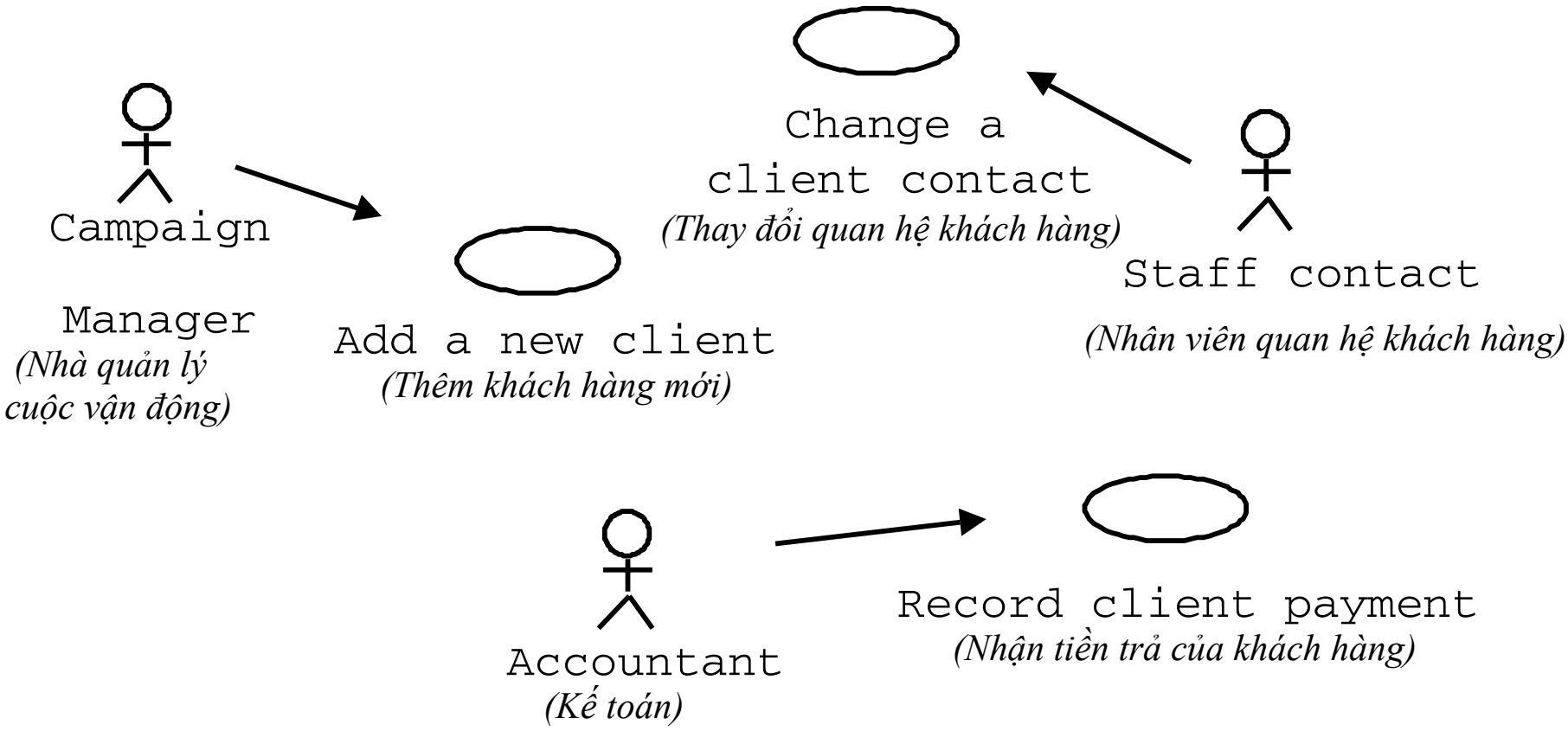
Ranh giới hệ thống (System Boundary) biểu diễn :

- Như một cái hộp chứa tất cả các use case có liên quan (vẽ dưới dạng khung chữ nhật)
- Lưu ý : các tác nhân nằm bên ngoài ranh giới hệ thống.

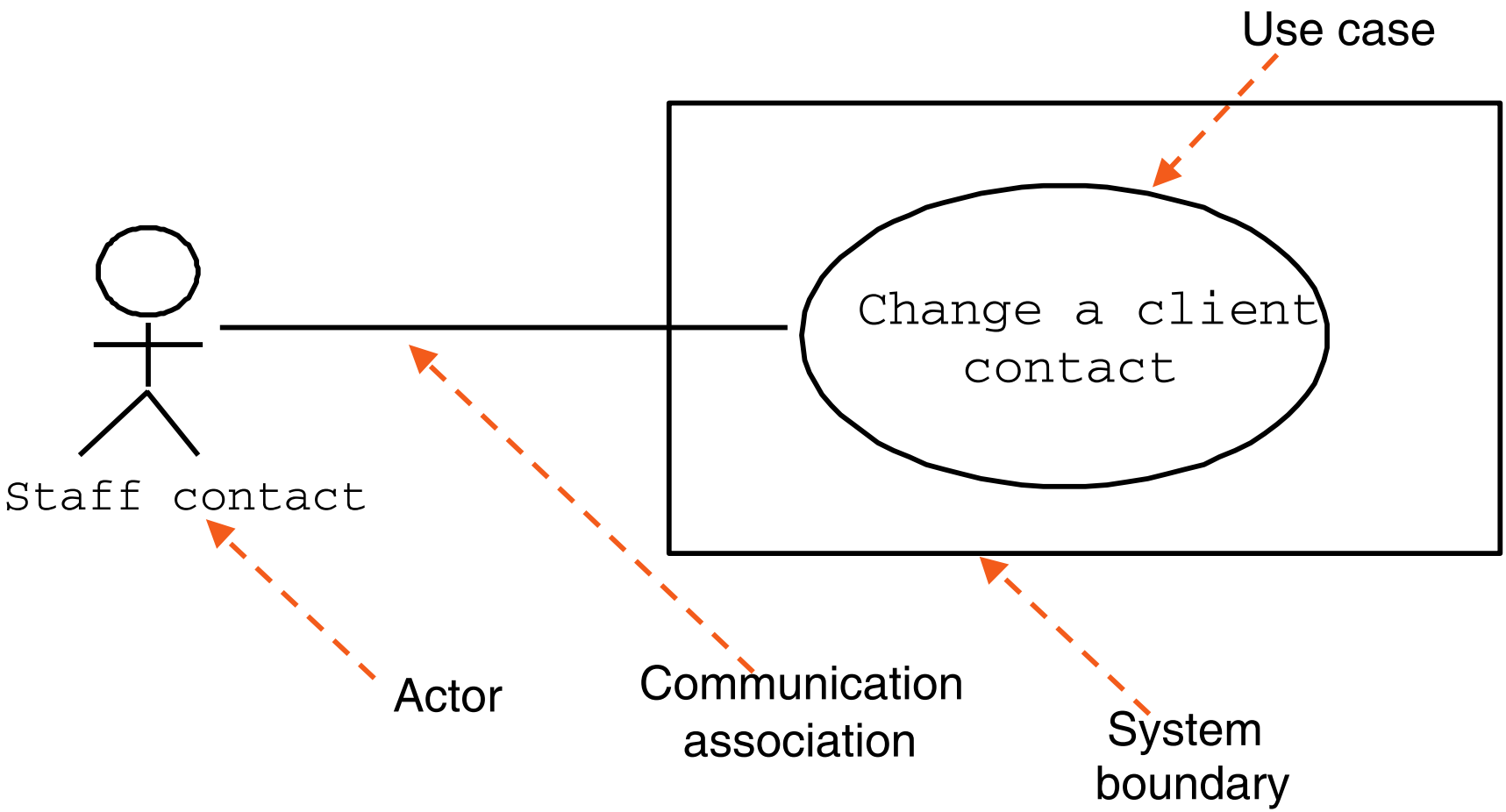
Đường nối kết (Communication association) : nối giữa tác nhân và các usecase.

Sơ đồ hoạt vụ (Use Case Diagrams)

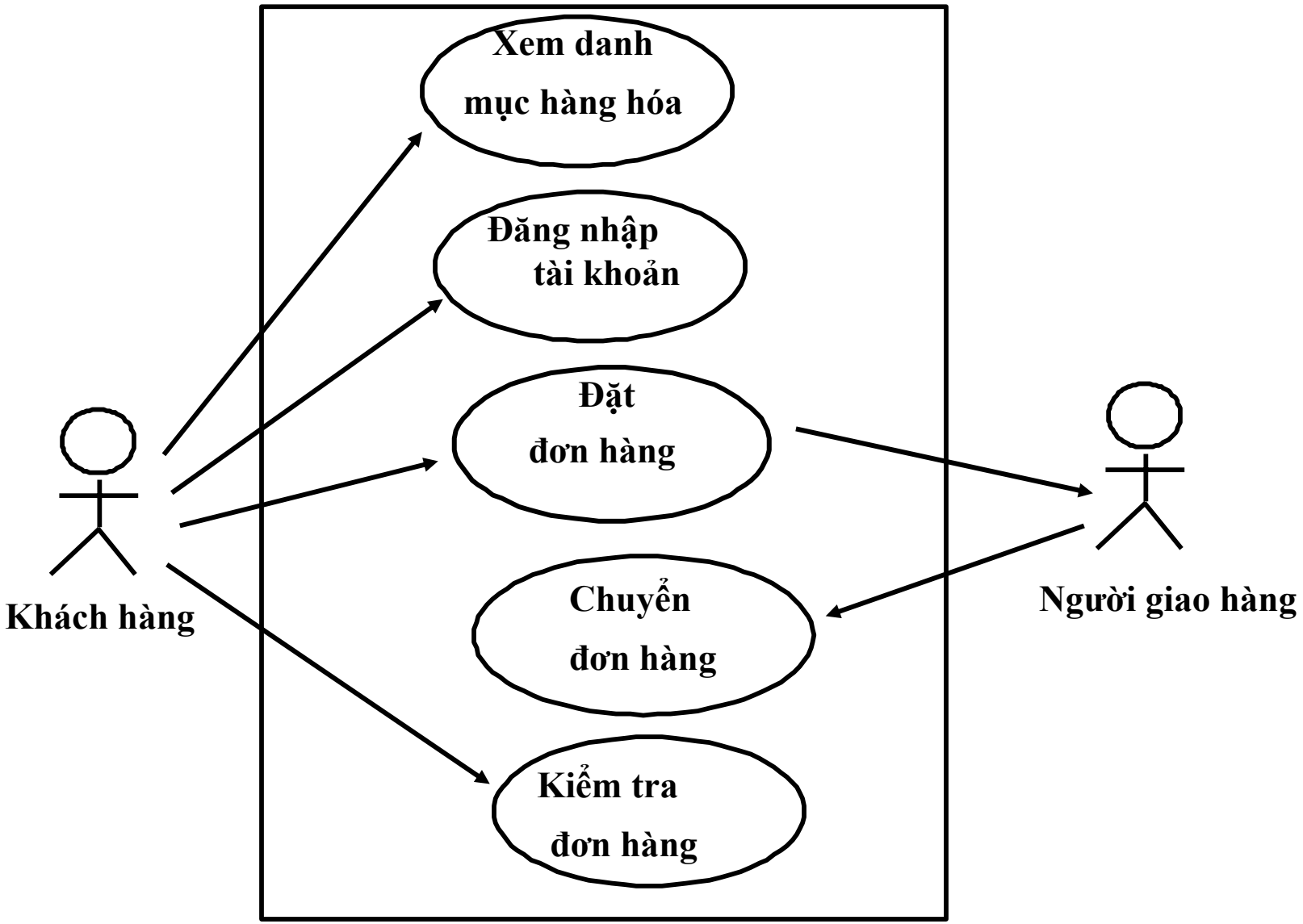
- Nắm bắt mối quan hệ giữa các nhân tố và Use Cases



Các ký hiệu trong Sơ đồ hoạt vụ

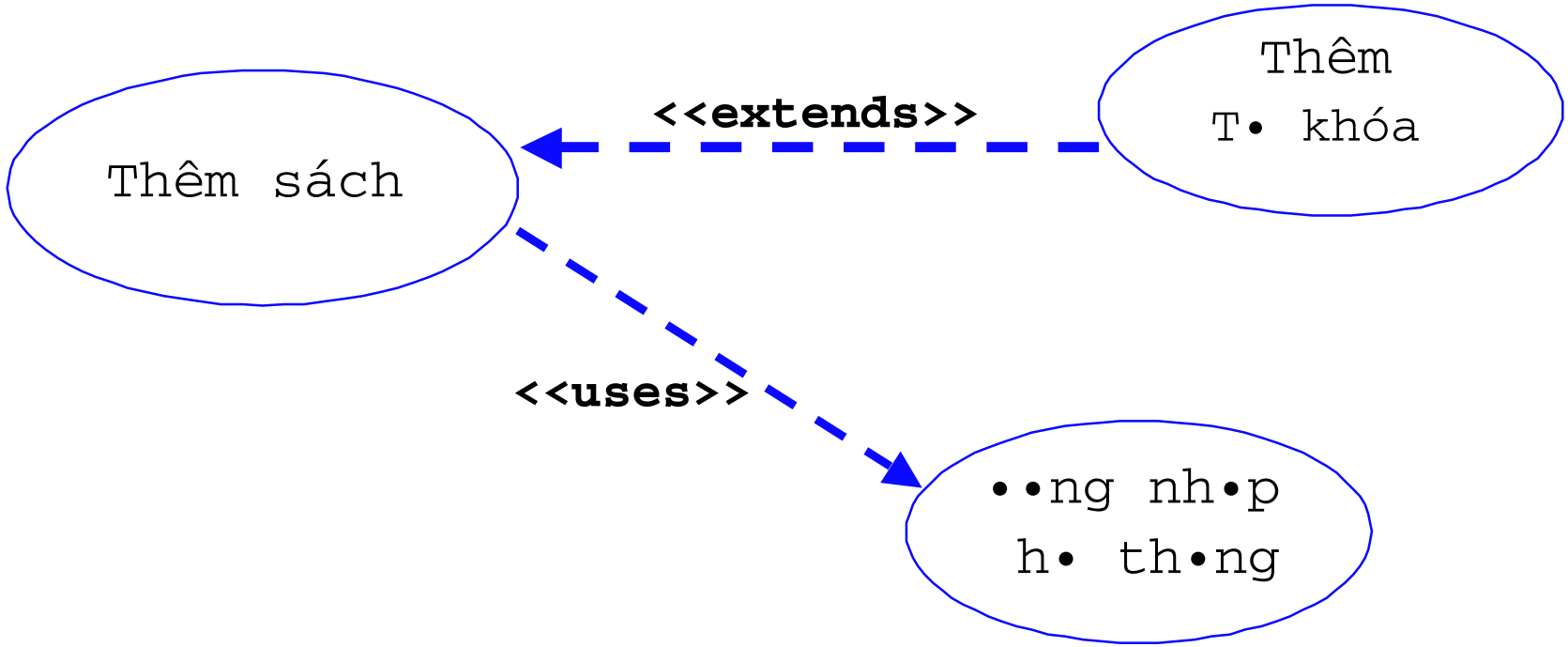


Ví dụ

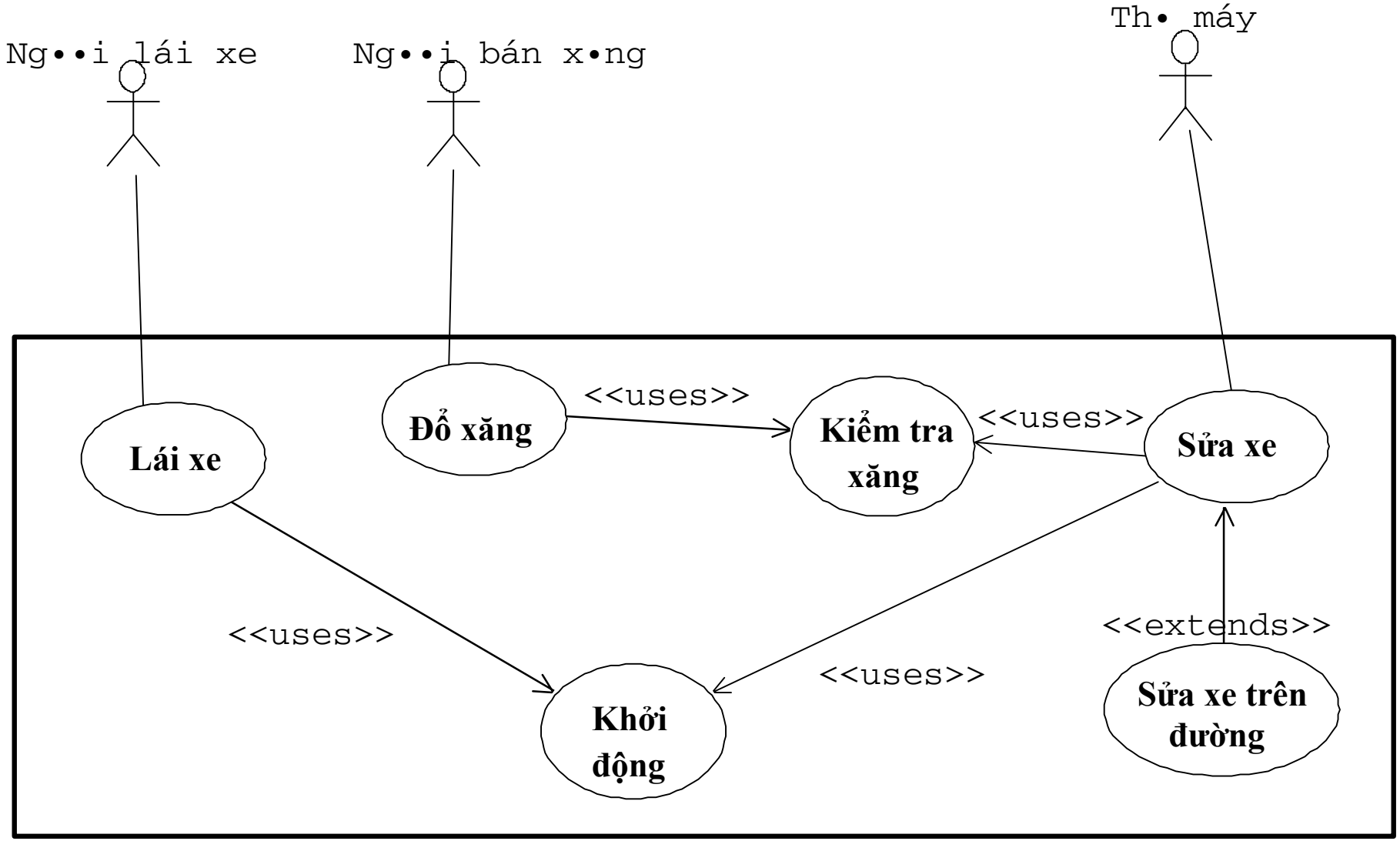

Xem danh
mục hàng hóa
Đăng nhập
tài khoản
Đặt
đơn hàng
Chuyển
đơn hàng
Kiểm tra
đơn hàng
Khách hàng
Người giao hàng

Quan hệ <<extends>> và <<uses>>

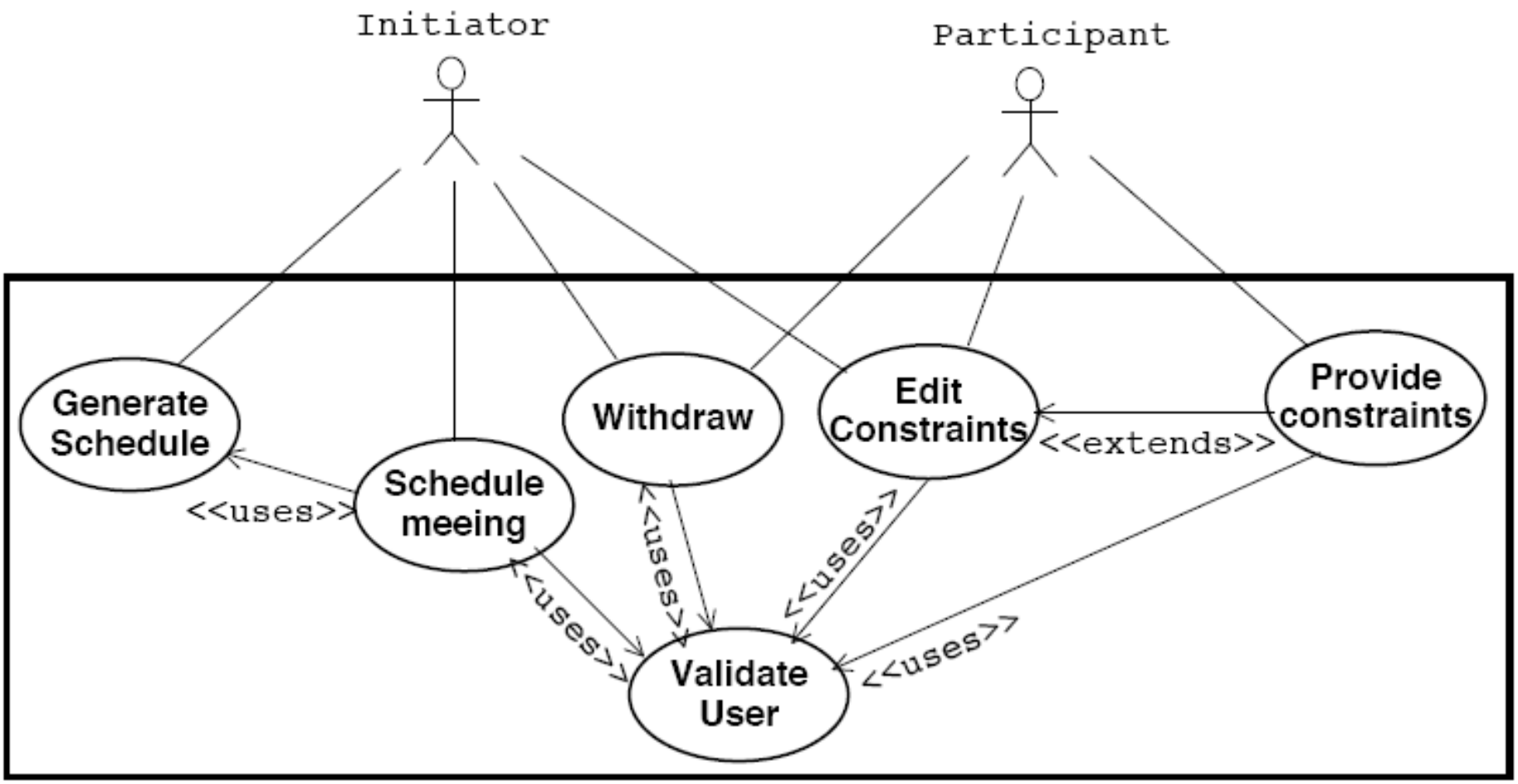
- **<<extends>> : khi một uses case thêm hành vi ứng xử vào base case**
  - **Dùng để mô hình một phần của use case mà người dùng có thể nhìn thấy như hành vi tùy chọn của hệ thống ;**
  - **Cũng mô hình một sub-case riêng lẻ mà nó chỉ thực thi trong một số điều kiện.**
- **<<uses>>: khi một use case gọi đến một cái khác (giống như gọi thủ tục)**
  - **Dùng để tránh việc mô tả cùng một dòng sự kiện một vài lần**
  - **Đặt hành vi chung trong một use case của cái sở hữu nó.**



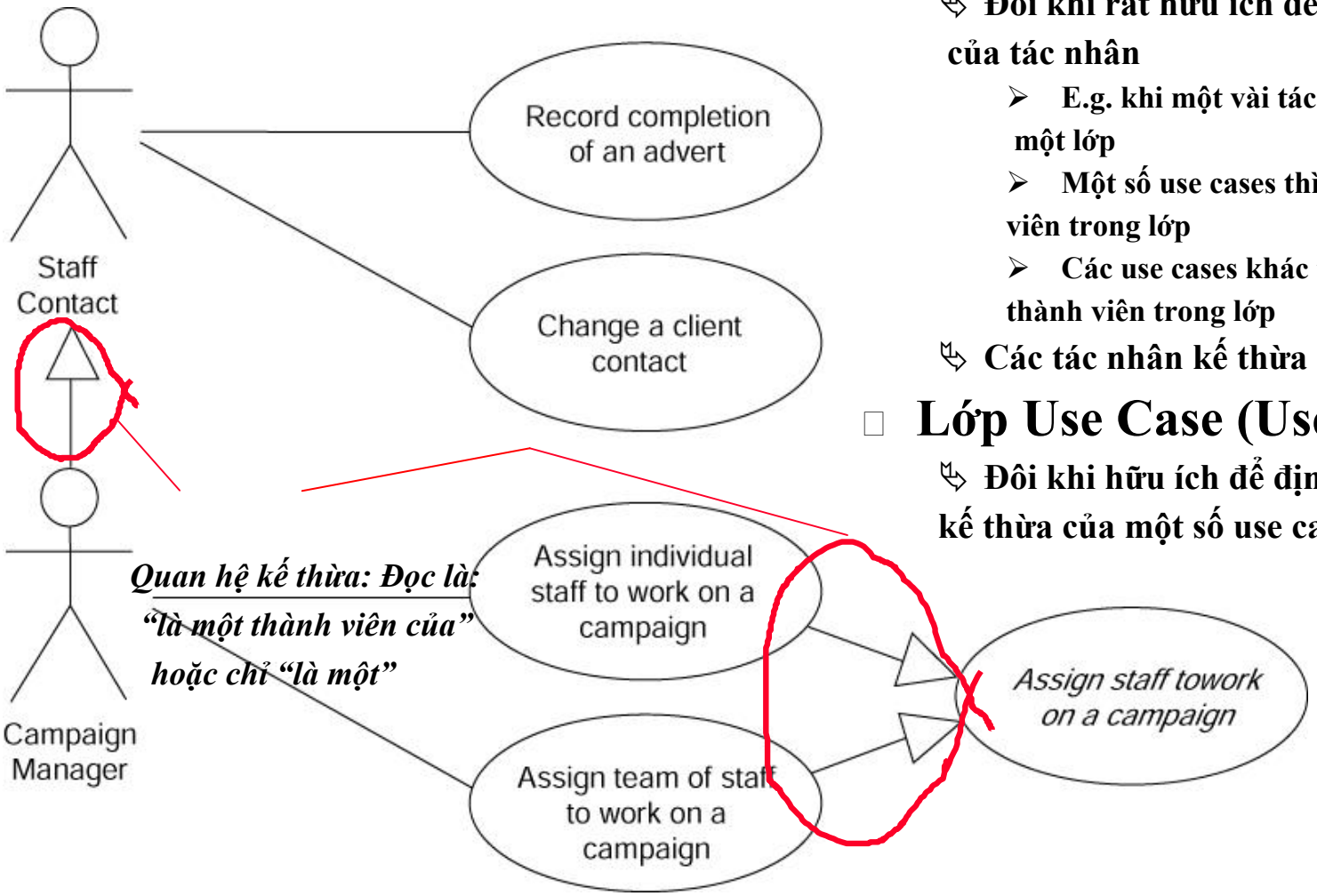
Mẫu use cases cho một chiếc xe



Ví dụ sắp xếp lịch họp



Quan hệ kế thừa



Quan hệ kế thừa: Đọc là: "là một thành viên của" hoặc chỉ "là một"

- **Lớp tác nhân (Actor classes)**
  - **Đôi khi rất hữu ích để định nghĩa các lớp của tác nhân**
    - **E.g. khi một vài tác nhân cùng thuộc về cùng một lớp**
    - **Một số use cases thì cần bởi tất cả các thành viên trong lớp**
    - **Các use cases khác thì chỉ cần bởi một số thành viên trong lớp**
  - **Các tác nhân kế thừa use cases từ lớp**
- **Lớp Use Case (Use Case classes)**
  - **Đôi khi hữu ích để định nghĩa một quan hệ kế thừa của một số use cae**

Nhận biết các tác nhân (Actors))

Hỏi những câu hỏi sau:

- Ai sẽ là người dùng chính của hệ thống? (tác nhân chính)
  - Ai sẽ cần sự hỗ trợ từ hệ thống để làm các công việc hàng ngày của họ?
  - Ai hoặc điều gì sẽ quan tâm đến những kết quả mà hệ thống đưa ra ?
- Ai sẽ bảo trì, quản lý, điều hành hoạt động của hệ thống ? (tác nhân phụ)
- Hệ thống cần các thiết bị phần cứng nào ?
- Hệ thống cần tương tác với những hệ thống nào khác ?

Tìm kiếm:

- Những người trực tiếp sử dụng hệ thống
- Và những người dùng khác – những người cần các dịch vụ từ hệ thống

Có 3 loại Actor chính:

- Người dùng. ➢Ví dụ: sinh viên, nhân viên, khách hàng...
- Hệ thống khác.
- Sự kiện thời gian. ➢Ví dụ: Kết thúc tháng, đến hạn...

Tìm kiếm Use Cases

- **Đối với mỗi tác nhân, hỏi các câu hỏi sau:**
  - Những chức năng nào mà tác nhân đòi hỏi từ hệ thống ?
  - Tác nhân nào cần làm ?
  - Tác nhân có cần đọc (read), tạo ra (create), phá hủy (destroy), sửa đổi (modify) hoặc lưu trữ (store) một số dạng thông tin trong hệ thống ?
  - Tác nhân có phải thông báo về các sự kiện trong hệ thống ?
  - Tác nhân thông báo những gì với hệ thống ?
  - Các sự kiện gì thì cần thiết trong môi trường chức năng của hệ thống?
  - Hoạt động thông thường của tác nhân thì quá đơn giản hoặc quá hiệu quả đối với các chức năng mới được cung cấp bởi hệ thống ?

Lập tài liệu Use Cases

Cho mỗi use case:

- Chuẩn bị tài liệu "luồng sự kiện" ("flow of events"), được viết từ hướng nhìn của một tác nhân.
- Mô tả chi tiết những cái mà hệ thống cần phải làm chứ không chỉ ra hệ thống sẽ làm nó như thế nào?

Các nội dung đặc trưng :

- Use case bắt đầu và kết thúc như thế nào;
- Tiến trình bình thường của các sự kiện;
- Tiến trình thay phiên của các sự kiện;
- Tiến trình ngoại lệ của các sự kiện;

Kiểu tài liệu:

- Chọn lựa cách hiển thị use case:
  - Biểu đồ hoạt động (Activity Diagrams) – tốt cho tiến trình công việc
  - Biểu đồ hợp tác (Collaboration Diagrams) – tốt cho thiết kế cấp cao
  - Biểu đồ trình tự (Sequence Diagrams) – tốt cho thiết kế chi tiết

Mẫu tài liệu Use Cases

Có thể dùng theo mẫu đơn giản như sau:

- **X. Luồng sự kiện cho Use case ABC**
- **X1. Điều kiện bắt đầu: danh sách những điều kiện phải thỏa mãn trước khi Use case được thực hiện.**

  Ví dụ như: một Use case khác phải thực hiện trước khi Use case này được thực hiện hay người dùng phải có đủ quyền để thực hiện Use case này. Không nhất thiết mọi Use case đều phải có điều kiện bắt đầu.
- **X2. Luồng chính: mô tả những bước chính sẽ xảy ra khi thực hiện Use case.**
- **X3. Các luồng phụ (luồng con).**
- **X4. Các luồng rẽ nhánh.**

Trong đó X là số thự tự của Use case trong hệ thống

Ví dụ : Luồng sự kiện mô tả Use Case

Luồng sự kiện mô tả Use case cho hệ thống rút tiền tự động như sau:

1.1 Điều kiện bắt đầu.
1.2 Luồng chính:
1.2.1 Người dùng đưa thẻ vào máy.
1.2.2. Máy hiển thông báo chào mừng và yêu cầu nhập mã số
1.2.3 Người dùng nhập mã số
1.2.4 Máy xác nhận mã số đúng. Nếu nhập sai mã số, luồng rẽ nhánh E-1 được thực hiện.
1.2.5 Máy hiện ra ba lựa chọn:
Rút tiền: luồng con A-1
Chuyển tiền: luồng con A-2
Thêm tiền vào tài khoản: luồng con A-3
1.2.6 Người dùng chọn rút tiền
1.3. Luồng con:
1.3.1 Luồng con A-1:
1.3.1.1 Máy hỏi số lượng tiền cần rút
1.3.1.2 Người dùng nhập số tiền cần rút : Máy kiểm tra trong tài khoản có đủ tiền không. Nếu không đủ luồng rẽ nhánh E-2 được thực hiện
....
1.4. Luồng rẽ nhánh:
1.4.1 E-1: Người dùng nhập sai mã số :Máy thông báo là người dùng đã nhập sai mã số yêu cầu người dùng nhập lại hoặc hủy bỏ giao dịch.
1.4.2 E-2: Không đủ tiền trong tài khoản...

Mô hình hóa trình tự của các sự kiện

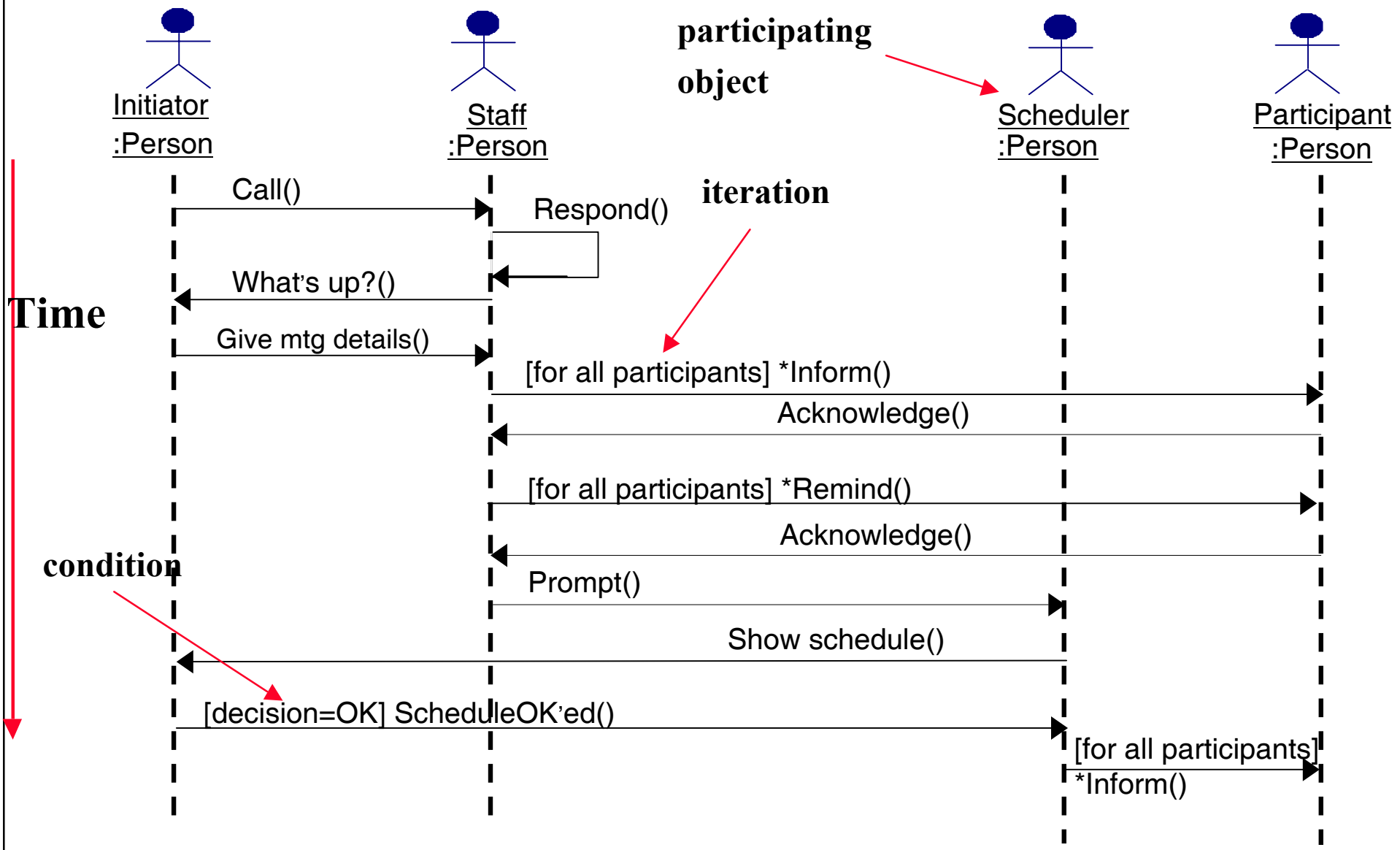
Các đối tượng "làm chủ" thông tin và hành vi

- Chúng có các thuộc tính và phương thức liên quan đến trách nhiệm của chúng.
- Chúng không "biết" thông tin về các đối tượng khác, nhưng có thể gọi đến.
- Để thực hiện tiến trình công việc, các đối tượng phải hợp tác.
  - …bằng cách gửi thông báo đến một cái khác để gọi những phương thức từ chúng
- Các đối tượng chỉ có thể gửi thông báo đến cái khác nếu chúng "biết" cái khác
  - I.e. nếu có một quan hệ kết hợp giữa chúng.

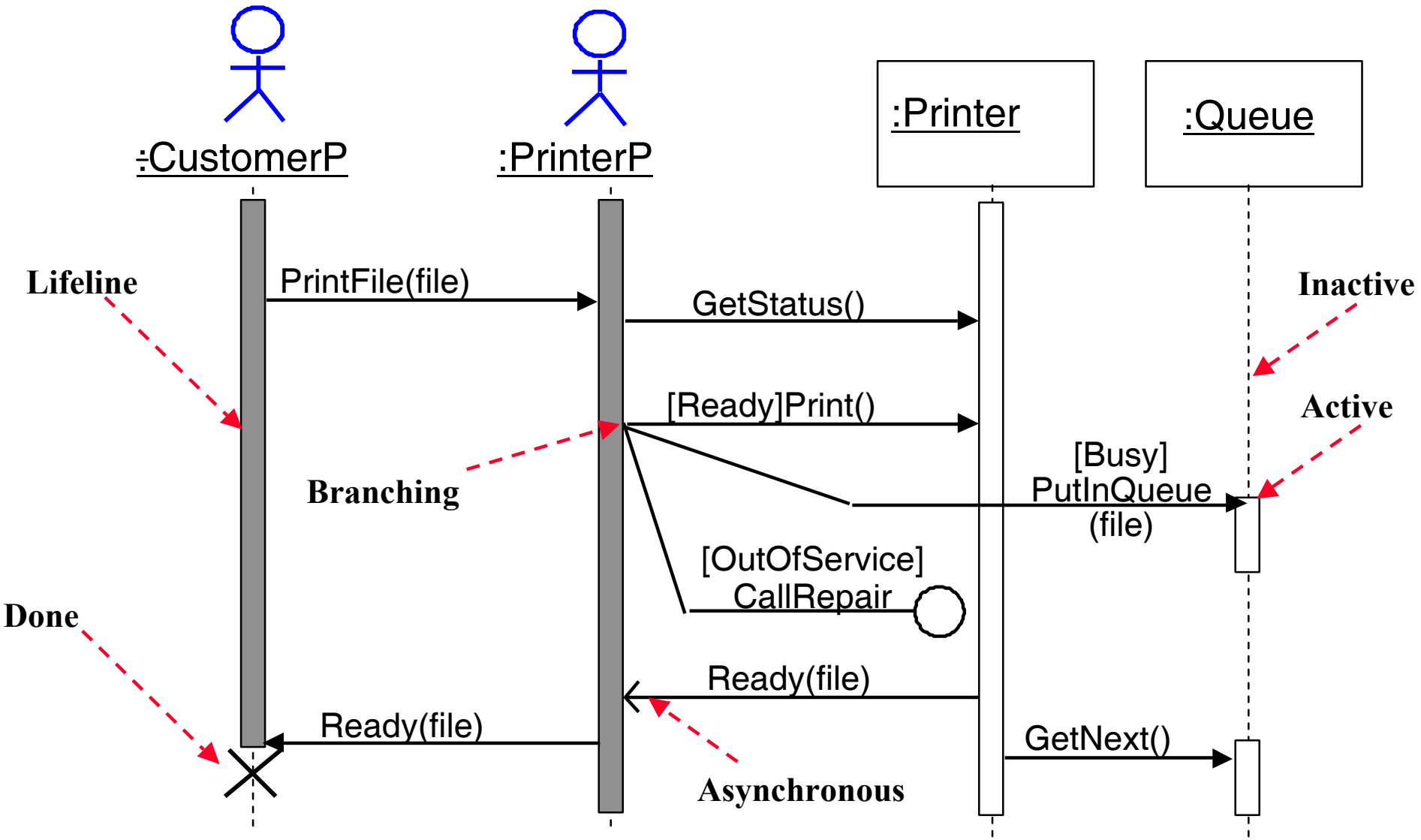
Mô tả một Use Case dùng Biểu đồ trình tự

- Biểu đồ trình tự chỉ ra từng bước những cái bao gồm trong một use case
  - Những đối tượng nào liên quan tới use case
  - Các đối tượng đó tham gia như thế nào trong chức năng
- Có thể cần một vài biểu đồ trình tự để mô tả duy nhất một use case.
  - Mỗi biểu đồ trình tự mô tả một kịch bản có thể cho use case
- Biểu đồ trình tự …
  - … sẽ giữ dễ dàng để đọc và hiểu.
  - … không bao gồm những kiểm soát logic phức tạp

Ví dụ về Biểu đồ trình tự



Branching messages, etc



Đừng quên cái gì đang được lập mô hình !

Trong giai đoạn phân tích

- Chúng ta muốn biết về lĩnh vực ứng dụng và các yêu cầu
- … Vì thế, chúng ta xây dựng mô hình phác họa quá trình diễn tiến (course-grained model) để mô tả các nhiệm vụ sẽ nằm ở đâu, và các đối tượng sẽ tương tác như thế nào
  - Mô hình này sẽ chỉ rõ một thông báo được chuyển đi, nhưng không quan tâm quá nhiều về nội dung mỗi thông báo.
  - Để giữ mọi thứ rõ ràng, dùng các biểu tượng (icons) mô tả các đối tượng và tác nhân bên ngoài, và các khối hộp (boxes) để biểu diễn các đối tượng của hệ thống.

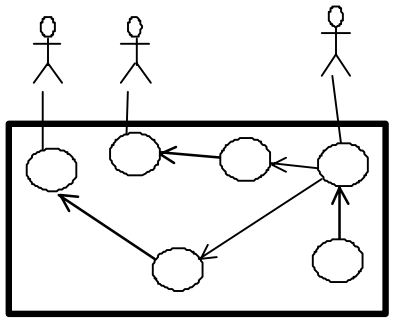
Trong giai đoạn thiết kế

- Chúng ta muốn quyết định về việc phần mềm sẽ hoạt động như thế nào
- … Vì thế, chúng ta phát triển các mô hình phác họa cầu kỳ (fine-grained models) để chỉ ra chính xác cái gì sẽ xảy ra khi hệ thống thực thi
  - E.g. chỉ ra những chi tiết chính xác của mỗi cách truyền thông báo.

Lecture 10:
Yêu cầu phi chức năng
Non-Functional Requirements (NFRs)

- **Tiền khái niệm:**
  - Các dạng ký pháp mô hình hóa mà chúng ta đã biết
- **Yêu cầu phi chức năng (NFRs) là gì ?**
  - Các hệ số chất lượng, tiêu chuẩn thiết kế; các độ đo
  - Ví dụ về NFRs
- **Tiếp cận hướng sản phẩm (Product-oriented) với NFRs**
  - Tạo ra sự đặc tả các hệ số chất lượng
  - Ví dụ: Sự tin cậy
- **Tiếp cận hướng tiến trình (Process-oriented) với NFRs**
  - Phân tích mục tiêu linh động (softgoal) cho các thỏa hiệp trong thiết kế

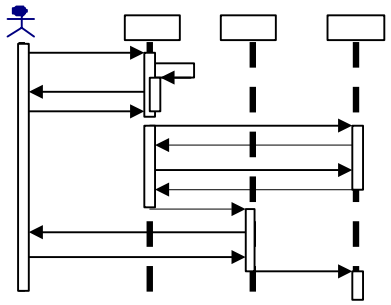
Các dạng biểu đồ trong UML…



Use Cases

Khía cạnh từ người dùng

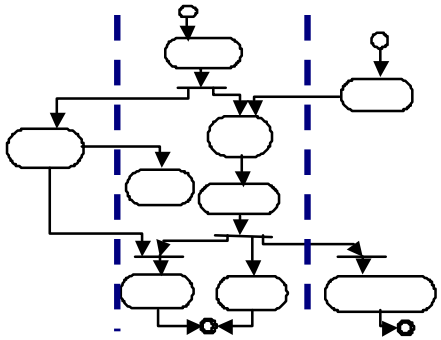
Liệt kê trực quan các chức năng tổng quan của các yêu cầu chính



Biểu đồ trình tự

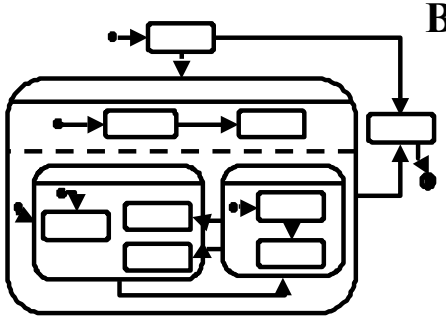
Kịch bản cụ thể giao tiếp giữa những người dùng và hệ thống

Trình tự của việc trao đổi các thông báo



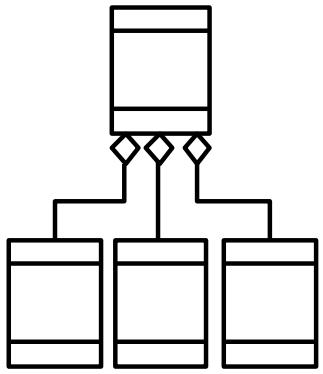
Biểu đồ hoạt động

Tiến trình hoạt động đồng thời và đồng bộ phụ thuộc giữa các công việc



Biểu đồ chuyển trạng thái

Các đáp ứng theo sự kiện của một đối tượng, mô hình hóa hành vi của một lớp giao diện.



Biểu đồ lớp

Cấu trúc thông tin quan hệ giữa các lớp, giao diện, hợp tác. Thể hiện mặt tĩnh của hệ thống.

…và những biểu đồ không thuộc - UML:

- **Mô hình mục tiêu (Goal Models)**
  - **Nắm bắt các mục tiêu chiến lược của các đối tác**
  - **Tốt cho việc khảo sát các câu hỏi 'how' và 'why' với các đối tác**
  - **Tốt để phân tích các thỏa hiệp trade-offs, đặc biệt trên các chọn lựa thiết kế**
- **Mô hình Cây bắt lỗi (Fault Tree Models) - như một ví dụ trong kỹ thuật phân tích rủi ro**
  - **Nắm bắt các lỗi tiềm ẩn của một hệ thống và nguồn gốc nguyên nhân**
  - **Tốt cho phân tích rủi ro, đặc biệt trong những ứng dụng với tiêu chuẩn an toàn**
- **Mô hình chiến lược phụ thuộc (Strategic Dependency Models (i*))**
  - **Nắm bắt quan hệ giữa các tác nhân trong một tổ chức**
  - **Hữu ích cho quan hệ giữa mục tiêu đối tác với tổ chức thiết lập chúng**
  - **Tốt cho việc thấu hiểu tổ chức sẽ thay đổi như thế nào**
- **Mô hình quan hệ - thực thể (Enntity-Relationship Models)**
  - **Nắm bắt quan hệ về cấu trúc thông tin được lưu trữ**
  - **Tốt cho việc hiểu các ràng buộc và những giả thiết về phạm vi chủ thể**
  - **Lập nền tảng tốt cho thiết kế CSDL**
- **Các kiểu bảng lớp (Class Tables), bảng sự kiện (Event Tables) và bảng điều kiện (Condition Tables (SCR))**
  - **Nắm bắt hành vi động của một hệ thống phản ứng trong thực tế**
  - **Tốt cho việc biểu diễn chức năng kết hợp từ inputs đến outputs**
  - **Tốt cho việc tạo các mô hình hành vi chính xác, như suy diễn tự động**

Yêu cầu phi chức năng là gì?

Chức năng vs. Phi chức năng

- Các yêu cầu chức năng mô tả cái hệ thống sẽ làm
  - Các chức năng có thể nắm bắt trong use cases
  - Các hành vi có thể được phân tích bằng việc vẽ biểu đồ trình tự, biểu đồ trạng thái, etc.
  - … và khả năng lần vết để giải quyết những vấn đề rắc rối của một chương trình
- Các yêu cầu phi chức năng là những ràng buộc toàn thể trên hệ thống phần mềm
  - e.g. chi phí phát triển, chi phí vận hành, khả năng thực thi, độ tin cậy, khả năng bảo trì, tính khả chuyển, tính thiết thực, etc.
  - Thường được biết như chất lượng phần mềm, hoặc chỉ là "các khả năng" ("ilities")
  - Thường không được cài đặt trong một mô-đun duy nhất của chương trình

Những trở ngại của NFRs

- Khó để mô hình
- Thường ở trạng thái không hình thức, và vì thế mà:
  - Thường mâu thuẫn,
  - Khó thực hiện trong suốt quá trình phát triển
  - Khó đánh giá khách hàng nào ưu tiên để phân phối
- Khó tạo ra các tiêu chuẩn để có thể đo lường chúng
  - Chúng ta muốn ổn định chúng theo cách có thể đo lường được sẽ đáp ứng chúng như thế nào.

Ví dụ về NFRs

Yêu cầu giao diện

- Giao diện của hệ thống mới sẽ như thế nào trong môi trường của nó?
  - Giao diện người dùng "thân thiện"
  - Giao diện đối với các hệ thống khác

Yêu cầu thực thi

- Giới hạn về thời gian / không gian
  - Thời gian tải nạp, thời gian đáp ứng, kích thước dữ liệu nhập và không gian lưu trữ
  - e.g. "hệ thống phải kiểm soát 1000 giao dịch trên giây"
- Độ tin cậy
  - Tính sẵn dùng của các thành phần
  - Sự nguyên vẹn của thông tin dùng duy trì và cung cấp cho hệ thống
  - e.g. "hệ thống phải có ít hơn 1 giờ đình trệ hoạt động trong 3 thángs"
- Tính bảo mật
  - E.g. Cho phép thông tin lưu hành, hoặc phân quyền người dùng
- Khả năng chịu lỗi
  - E.g. Hệ thống sẽ cần năng lực tồn tại, chịu đựng các sự cố tự nhiên, etc

Yêu cầu vận hành

- Các ràng buộc vật lý (kích thước, trọng lượng),
- Mức kỹ năng & khả năng nhân sự
- Dễ bảo trì
- Các điều kiện về môi trường
- etc

Yêu cầu chu trình sống

- "Future-proofing"
  - Khả năng bảo trì
  - Khả năng mở rộng
  - Tính khả chuyển
  - Thị trường mong đợi hoặc vòng đời sản phẩm
- Những giới hạn phát triển
  - E.g giới hạn thời gian phát triển,
  - Tài nguyên sẵn dùng
  - Các chuẩn về phương pháp
  - etc.

Yêu cầu kinh tế

- e.g. giới hạn nghiêm ngặt đúng thời gian và/hoặc vốn dài hạn.

Tiếp cận NFRs

Sản phẩm vs. Tiến trình?

- Tiếp cận hướng sản phẩm (Product-oriented Approaches)
  - Tập trung vào chất lượng của hệ thống (hoặc phần mềm)
  - Nắm bắt các tiểu chuẩn thiết lập của mỗi yêu cầu
  - … để mà chúng ta có thể đo lường chúng khi sản phẩm được thiết kế
- Tiếp cận hướng tiến trình (process-oriented Approaches)
  - Tập trung vào các yêu cầu phi chức năng (NFRs) nào có thể dùng trong tiến trình thiết kế
  - Phân tích tương tác giữa NFRs và các chọn lựa thiết kế
  - … để mà chúng ta có thể đưa ra các quyết định thiết kế phù hợp

Định lượng (Quantitative) vs. Định tính (Qualitative)?

- Tiếp cận định lượng
  - Tìm thang đo các thuộc tính về chất lượng
  - Tính toán mức độ cho một thiết kế đáp ứng với các mục tiêu chất lượng nào
- Tiếp cận định tính
  - Nghiên cứu các dạng quan hệ giữa các mục tiêu chất lượng
  - Lý do của các sự thỏa hiệp (trade-offs), etc.

Chất lượng phần mềm

- **Nghĩ đến một đồ vật thông thường**
  - **e.g. Một cái ghế – bạn sẽ đo "chất lượng" của nó như thế nào?**
    - **Chất lượng kết cấu?**
    - **Giá trị thẩm mỹ?**
    - **Đáp ứng mục tiêu?**
- **Tất cả các độ đo chất lượng đều có quan hệ**
  - **Không có thang đo nào tuyệt đối**
  - **Đôi khi chúng ta có thể nói A tốt hơn B…**
    - **… nhưng thường rất khó để nói tốt hơn thế nào !**
- **Đối với phần mềm :**
  - **Chất lượng kết cấu?**
    - **Phần mềm thì không được chế tạo (mà là phát triển)**
  - **Giá trị thẩm mỹ?**
    - **nhưng hầu hết phần mềm thì trực quan**
    - **giá trị thẩm mỹ là một sự quan tâm bên lề**
  - **Đáp ứng mục tiêu?**
    - **Cần phải hiểu rõ mục tiêu**

Sự đáp ứng (Fitness)

Source: *Budgen, 1994, pp58-9*

Chất lượng phần mềm là đáp ứng với mục tiêu

- Nó có thực hiện điều cần thực hiện?
- Nó có thực hiện theo cách người dùng cần nó thực hiện?
- Nó có đủ tin cậy? Đủ nhanh? Đủ an toàn? Đủ bảo mật?
- Nó sẽ có khả năng thực hiện? Nó sẽ luôn sẵn sàng khi người dùng cần đến nó?
- Nó có thể thay đổi khi nhu cầu thay đổi?

Chất lượng không phải là một độ đo cho riêng phần mềm

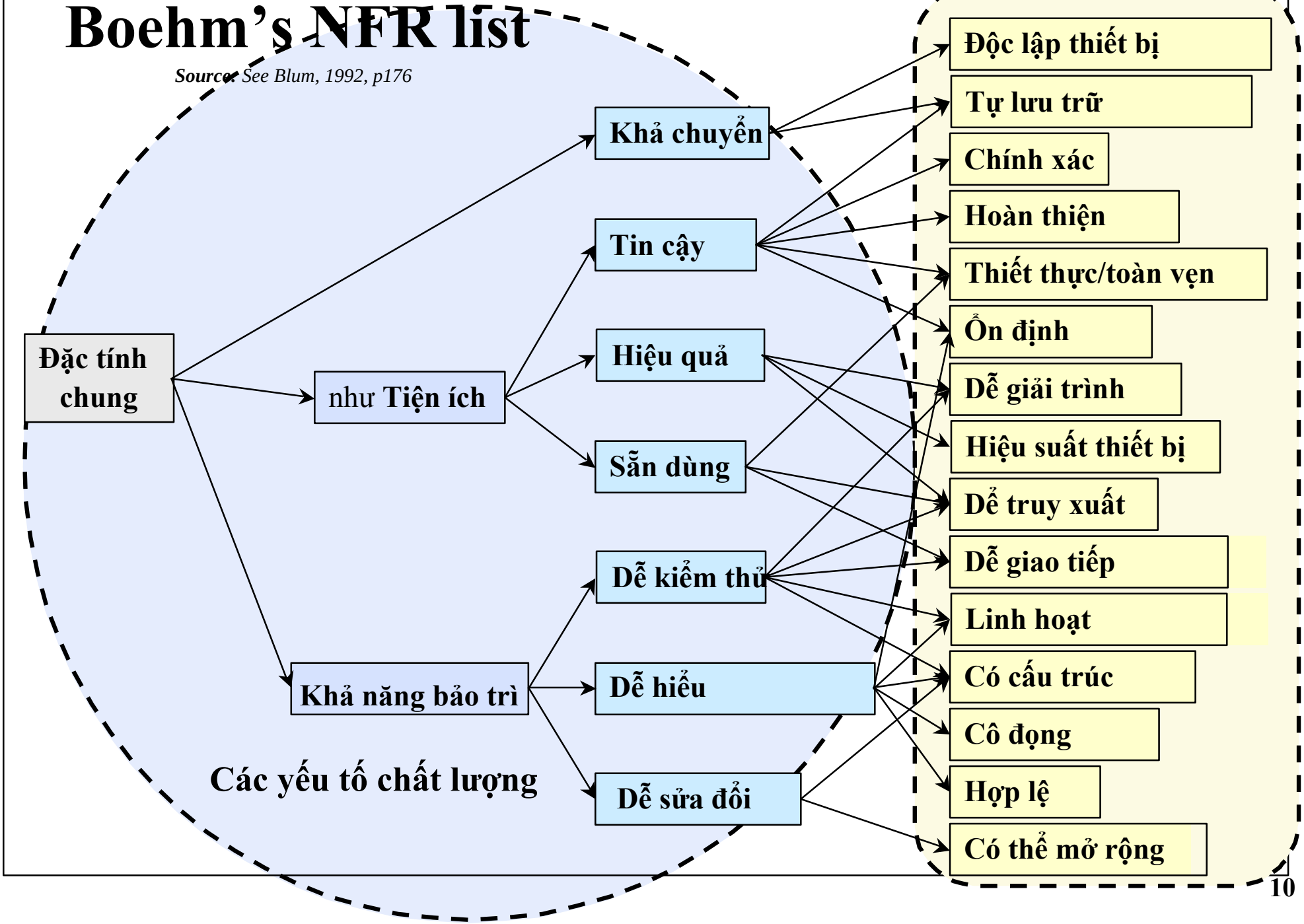
- **Nó đo lường các quan hệ của phần mềm trong lĩnh vực ứng dụng của nó**
  - **không thể đo lường điều này trước khi bạn đặt phần mềm vào môi trường của nó…**
  - **…và chất lượng sẽ khác nhau trong những môi trường khác nhau!**
- **Trong khi thiết kế, chúng ta cần dự đoán phần mềm sẽ đáp ứng với mục tiêu tốt như thế nào**
  - **chúng ta cần những người dự đoán chất lượng tốt (người phân tích thiết kế)**
- **Trong khi phân tích yêu cầu, chúng ta cần hiểu rõ việc đáp ứng với mục tiêu sẽ được đo lường thế nào**
  - **Mục tiêu dự định là gì?**
  - **Các yếu tố chất lượng gì sẽ quan trọng đối với các đối tác?**
  - **Những yếu tố đó sẽ được tổ chức như thế nào?**

Các yêu tố vs. Tiêu chuẩn

- **Các yếu tố chất lượng**
  - **Điều này thì liên quan đến quan hệ khách hàng (customer-related)**
    - **Ví dụ: hiệu năng, tính toàn vẹn, độ tin cậy, tính chính xác, khả năng chịu lỗi, sự tiện dụng,...**
- **Tiêu chuẩn thiết kế**
  - **Điều này liên quan tới kỹ thuật (hướng phát triển) chẳng hạn như quản lý các bất thường, tính hoàn thiện, tính ổn định, khả năng lưu vết, tính trực quan,...**
- **Yếu tố chất lượng và tiêu chuẩn thiết kế thì có liên quan:**
  - **Mỗi yếu tố phụ thuộc vào một số tiêu chuẩn liên quan:**
    - **E.g. Tính chính xác phụ thuộc vào tính hoàn thiện, tính ổn định, khả năng lưu vết,...**
    - **E.g. Tính khả thi phụ thuộc vào sự mô-đun hóa, tính linh động và tính đơn giản**
  - **Có thể có một số chuẩn kết hợp để giúp bạn…**
- **Trong quá trình phân tích:**
  - **Xác định quan hệ quan trọng của mỗi yếu tố chất lượng**
    - **Từ quan điểm của khách hàng!**
  - **Xác định tiêu chuẩn thiết kế mà mỗi yếu tố này phụ thuộc**
  - **Thiết lập độ đo lường các yêu cầu**

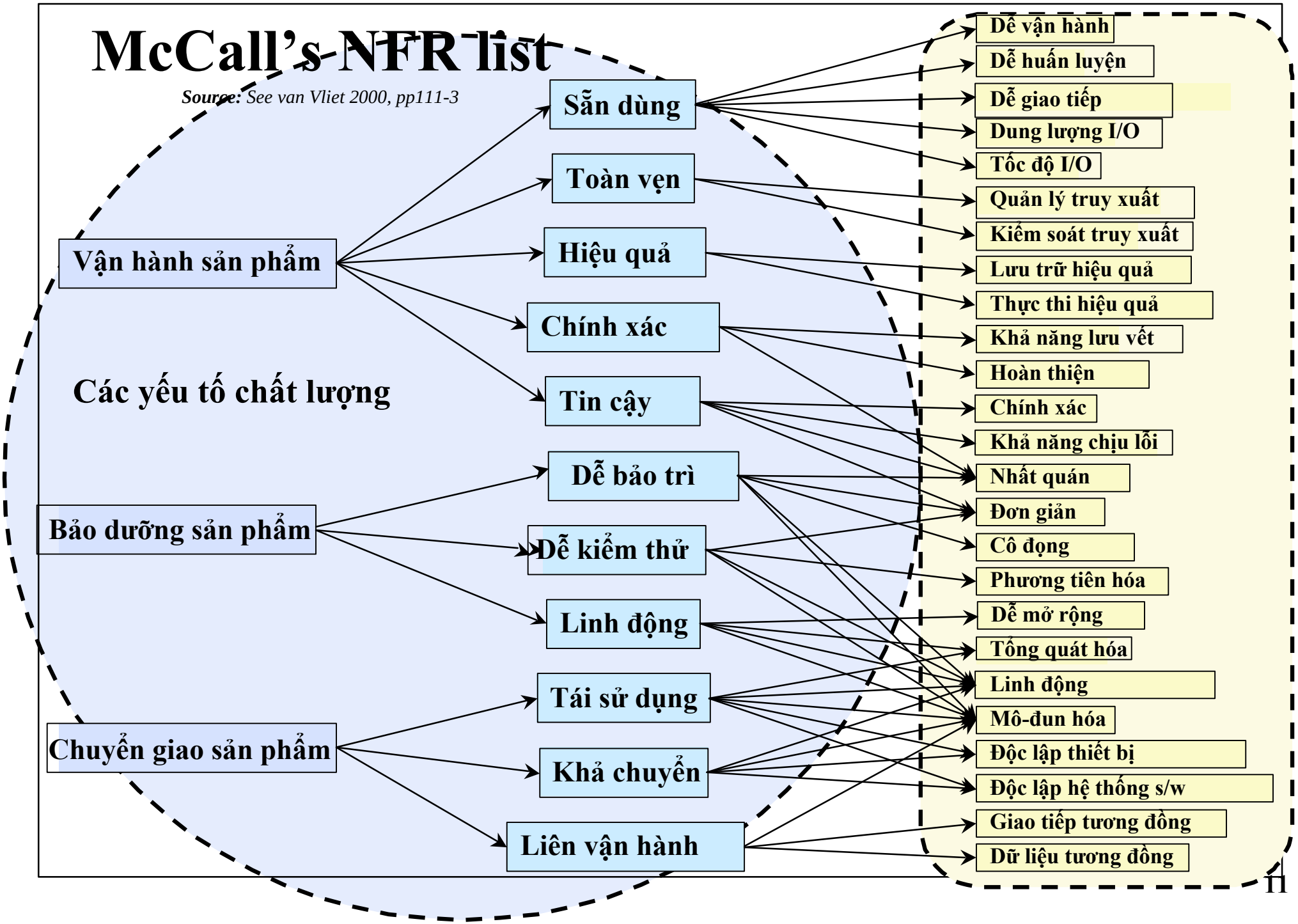
Boehm's NFR list

Source: See Blum, 1992, p176



McCall's NFR list

Source: *See van Vliet 2000, pp111-3*



Thiết lập độ đo các yêu cầu

Source: *Budgen, 1994, pp60-1*

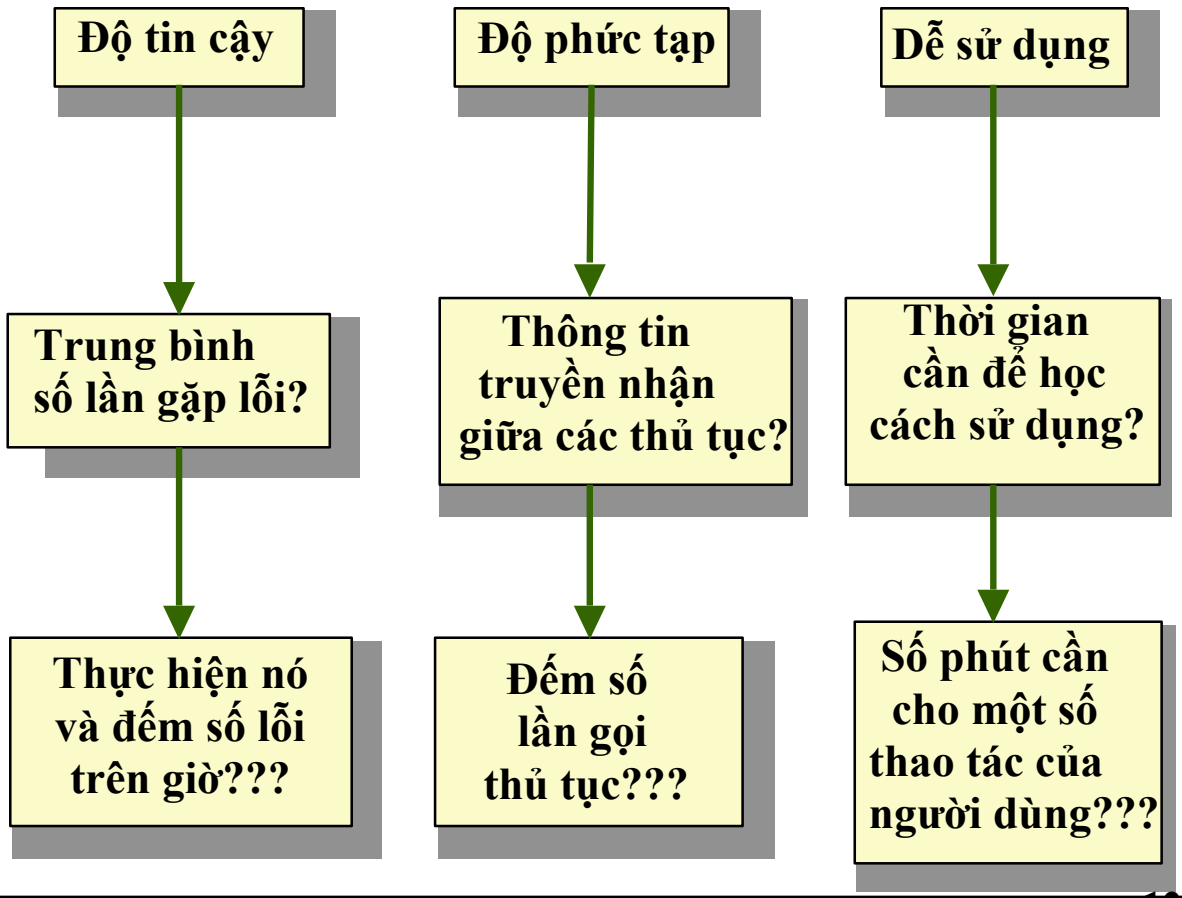
- **Chúng ta phải đổi các khái niệm không rõ ràng về chất lượng thành độ đo lường**

Các yếu tố chất lượng
(các khái niệm trừu tượng
của các đặc tính chất lượng)

Tiêu chuẩn đo lường
(định nghĩa một số độ đo)

Đếm số lần từ kết quả hiển thị của thiết kế
(hiện thực hóa độ đo)

Các ví dụ ...



Ví dụ : Đo độ tin cậy

Định nghĩa ví dụ:

- Khả năng của hệ thống hành xử một cách thống nhất theo phương cách người dùng có thể chấp nhận được khi hệ thống vận hành bên trong môi trường mà nó dự định thực hiện.

Các chú thích:

- Độ tin cậy có thể được xác định dưới dạng một giá trị phần trăm (như, 99.999%)
- Điều này có thể mang ý nghĩa khác nhau đối với các ứng dụng khác nhau:
  - Mạng điện thoại: mạng toàn cục có thể bị lỗi không nhiều hơn, trung bình khoảng 1hr mỗi năm, nhưng lỗi của các chuyển mạng cá nhân có thể xuất hiện thường xuyên hơn nhiều.
  - Hệ thống kiểm tra bệnh nhân: hệ thống có thể bị lỗi đến khoảng 1hr/year, nhưng trong trường hợp này, các bác sĩ/ y tá sẽ báo động lỗi. Lỗi thường xuyên hơn ở các bộ phận cá thể thì không thể chấp nhận được.
- Tốt nhất, chúng ta có thể thực hiện một số việc như sau:
  - "...Không có nhiều hơn X lỗi trên 10KLOC (line of code) có thể được phát hiện trong suốt quá trình tích hợp và kiểm thử; không có nhiều hơn Y lỗi trên 10KLOC có thể tồn tại trong hệ thống sau khi phân phối, theo như tính toán bởi Monte Carlo dùng kỹ thuật tìm kiếm nhân lỗi như trong phụ lục Z; hệ thống phải vận hành 99.9% trên 100% khả năng vận hành theo lịch trong suốt năm vận hành đầu tiên của nó..."

Đo độ tin cậy…

Ví dụ - yêu cầu về độ tin cậy:

- "Phần mềm sẽ có không nhiều hơn X lỗi trên một ngàn dòng mã lệnh"
- ... Nhưng có thể đo được lỗi tại thời điểm phân phối sản phẩm không?

Dùng trình gỡ lỗi (Debuging)

- Đo lường hiệu quả của tiến trình kiểm thử
- Một số nhân lỗi thì được nêu ra cho hệ thống phần mềm
  - sau đó thực hiện kiểm thử và sửa lỗi không toàn bộ

$$\text{Ước lượng số lỗi trong hệ thống} = \frac{\#\text{ nhân lỗi} \times \#\text{ lỗi phát hiện}}{\#\text{ nhân lỗi được phát hiện}}$$

- ...NHƯNG, không phải tất cả lỗi đều quan trọng như nhau!

Thiết lập độ đo yêu cầu

Xác định 'tiêu chuẩn đáp ứng' cho mỗi yêu cầu

- Đặt 'tiêu chuẩn đáp ứng' bên cạnh yêu cầu
- E.g. Đối với phần mềm ATM mới
  - Yêu cầu: "Phần mềm phải trực quan và rõ ràng (không cần giải thích gì thêm)"
  - Tiêu chuẩn đáp ứng: "95% các khách hàng hiện có của ngân hàng sẽ có thể rút tiền và gửi séc trong vòng 2 phút khi sử dụng sản phẩm lần đầu tiên"

Chọn tiêu chuẩn đáp ứng tốt

- Các đối tác thường hiếm khi mô tả điều này
- Các tiêu chuẩn đúng không luôn rõ ràng:
- Làm việc với các đối tác để tìm ra các tiêu chuẩn đáp ứng tốt

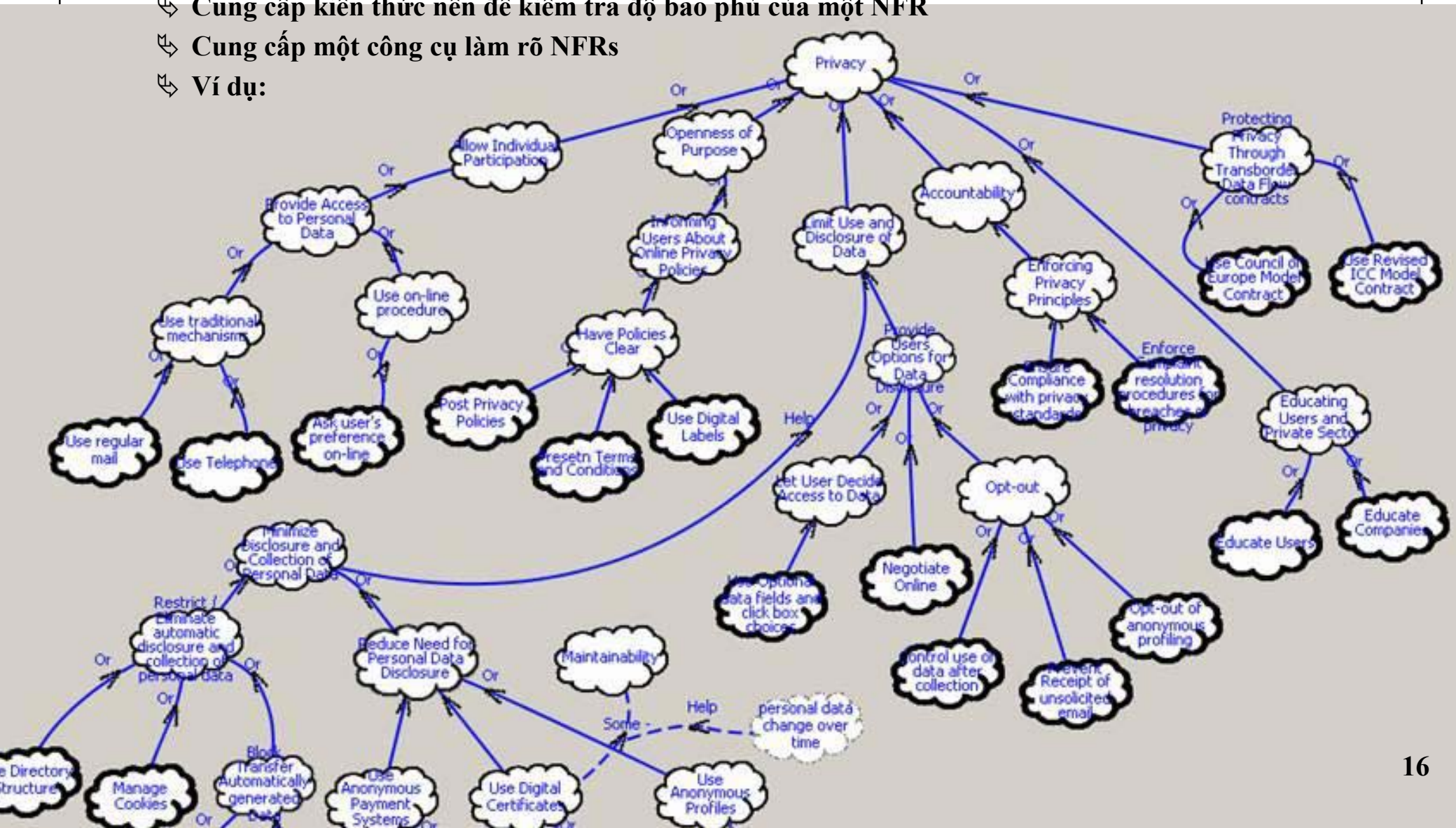
Sự thay thế

- Đôi khi, chất lượng không thể đo lường trực tiếp. Tìm các định danh thay thế:
  - E.g. "Một vài dữ liệu nhập bị lỗi" thế cho Tính dễ sử dụng
  - E.g. "Liên kết không chặt chẽ" thế cho Tính dễ bảo trì

Danh mục NFR

Source: *Cysneiros & Yu, 2004*

- **Danh mục định nghĩa sẵn của sự phân tích NFR**
  - **Cung cấp kiến thức nền để kiểm tra độ bao phủ của một NFR**
  - **Cung cấp một công cụ làm rõ NFRs**
  - **Ví dụ:**



Lecture 11:
Đặc tả yêu cầu
Requirements Specifications

- **Tại sao cần viết đặc tả**
  - **Mục đích và những người tham gia đặc tả**
  - **Chọn kích thước và quy cách thích hợp**
- **Yêu cầu của sự Đặc tả**
  - **Các đặc tính của đặc tả tốt**
  - **Các vấn đề chủ yếu**
  - **Những gì không cần thiết trong đặc tả**
- **Cấu trúc của một tài liệu yêu cầu**
  - **Chuẩn IEEE**

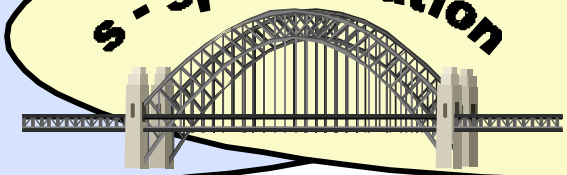
Yêu cầu vs. Đặc tả

Application Domain

Machine Domain

D - domain properties

R - requirements



C - computers

P - programs

R:

Tôi muốn bảng kê các thuốc này lập theo thứ tự thời gian.

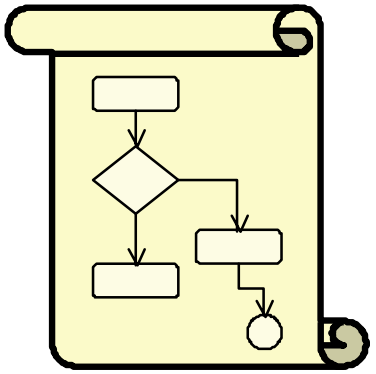
S:

3.11.2.3. Khi nhận được danh mục thuốc phân phối đến, hệ thống sẽ thêm từng loại thuốc theo thứ tự vào các mục trong bảng kê hiện có.
3.11.2.4. Khi bảng kê đã lập xong, hệ thống sẽ …

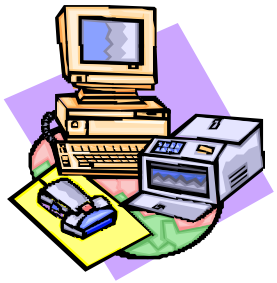
D:

Bảng phân phối và bảng kê chỉ phân loại thuốc theo các nhóm thuốc.

P:



C:



Đặc tả yêu cầu phần mềm

Thực hiện kết nối Yêu cầu với những cái khác như thế nào?

- Cần mô tả chúng trong một tài liệu SRS (Software Requirement Specification)
  - Nhưng một SRS không phải nhất thiết là một tài liệu chỉ trên giấy tờ ...

Mục tiêu SRS

- Chuyển tải thông tin
  - Giải thích lĩnh vực ứng dụng và hệ thống cần phát triển
- Lập hợp đồng
  - Có lẽ là một ràng buộc hợp lệ!
  - Biểu diễn sự thỏa thuận và một lời cam kết
- Cơ sở cho việc đánh giá phần mềm
  - Hỗ trợ kiểm thử, V&V
  - "Đủ thông tin để kiểm tra liệu hệ thống được phân phối có đáp ứng được các yêu cầu"
- Cơ sở cho việc quản lý thay đổi

Người dùng SRS

- Khách hàng & Người dùng
  - Quan tâm đến các yêu cầu hệ thống…
  - …nhưng không biết các chi tiết về yêu cầu phần mềm
- Nhà phân tích (yêu cầu) hệ thống
  - Viết những đặc tả liên quan khác
- Người phát triển, Lập trình viên
  - Phải cài đặc các yêu cầu
- Kiểm thử viên
  - Phải kiểm tra rằng các yêu cầu được đáp ứng
- Quản lý dự án
  - Phải đo lường và kiểm soát dự án

Đặc tả tương thích

- **Xét 2 dự án khác nhau:**

A) Dự án nhỏ, 1 người lập trình, 2 tháng làm việc

Người lập trình thảo luận với khách hàng, sau đó viết khoảng 2-trang ghi chú

B) Dự án lớn, 50 người lập trình, 2 năm làm việc

Đội phân tích lập mô hình các yêu cầu, sau đó viết khoảng 500-trang tài liệu đặc tả yêu cầu phần mềm (SRS – SoftwareRequirements Specifications)

	Project A	Project B
Mục tiêu của đặc tả?	Tạo sự thấu hiểu cho người lập trình; phản hồi cho người dùng	Lập một tài liệu; có chứa đầy đủ các chi tiết cho tất cả các lập trình viên
Nhà quản trị?	Đặc tả thì không nhất thiết; đã có sẵn các nguồn tài nguyên	Dùng đặc tả để ước lượng nguồn tài nguyên cần thiết và hoạch định sự phát triển
Người đọc?	**Chủ yếu** : Tác giả đặc tả **Thứ yếu** : Khách hàng	**Chủ yếu** : Lập trình viên, nhà quản trị, kiểm thử viên **Thứ yếu** : Khách hàng

Một biến dạng: Tài liệu thầu (Procurement)

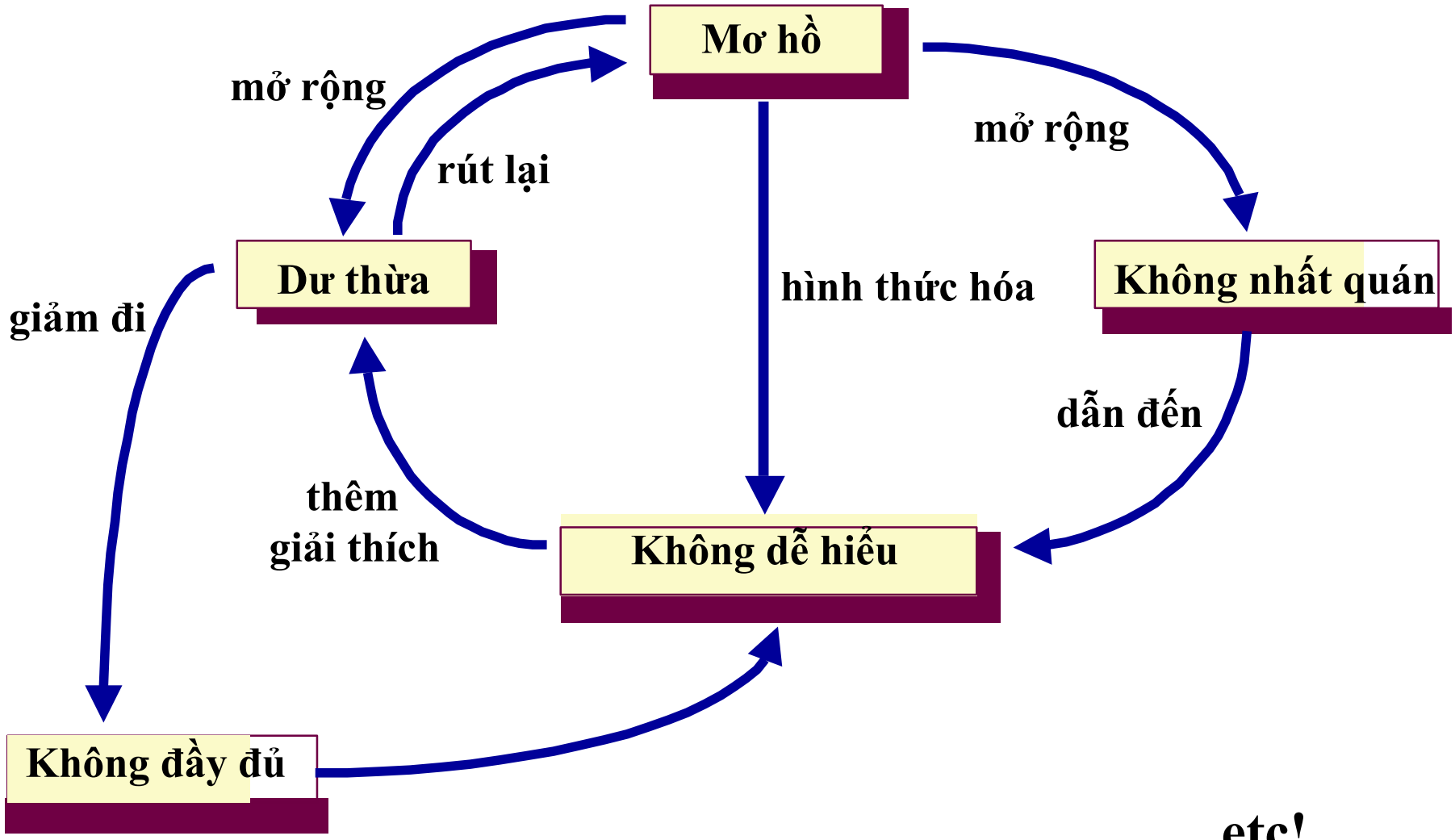
- **Một 'SRS' có thể được viết bởi…**
  - **…Nhà thầu (the procurer):**
    - SRS thì thực sự là một lời mời cho những đề xuất
    - Phải đủ tổng quát để có thể chọn lựa được một người đấu thầu tốt…
    - …và đủ chi tiết để loại bỏ những người đấu thầu không hợp lý
  - **…Người đấu thầu (the bidders):**
    - SRS là một đề xuất để cài đặt một hệ thống đáp ứng khách hàng
    - Phải đủ chi tiết để chứng tỏ tính khả thi và khả năng về kỹ thuật
    - …và đủ tổng quát để tránh vượt quá cam kết
  - **…Nhà phát triển được tuyển chọn:**
    - Phản ánh sự thấu hiểu về các yêu cầu khách hàng của nhà phát triển
    - Một hình thức cơ sở cho sự đánh giá việc thực thi trên hợp đồng
  - **…hoặc bởi một người thầu RE độc lập!**
- **Chọn lựa trên quan điểm nào để hoàn thành hợp đồng**
  - **Sớm (giai đoạn khái niệm)**
    - **chỉ có thể đánh giá các nhà đấu thầu trên năng lực và khả năng biểu lộ**
  - **Trễ (giai đoạn đặc tả chi tiết)**
    - **nhiều công việc hơn cho nhà thầu; các kỹ năng RE phù hợp có thể không có sẵn**
  - **Chuẩn IEEE đề nghị SRS nên được cùng xây dựng bởi nhà thầu và người phát triển**

Các đặc tính của một SRS

Source: *Adapted from IEEE-STD-830-1998*

- **Hợp lệ (hoặc "đúng")**
  - Diễn tả được nhu cầu thực sự của các đối tác (khách hàng, người dùng, …)
  - Không có chứa mọi thứ không là "yêu cầu"
- **Không mơ hồ**
  - Mỗi câu chỉ có thể đọc chính xác theo một cách
- **Hoàn chỉnh**
  - Tất cả mọi thứ hệ thống phải thực hiện…
  - …và tất cả mọi thứ nó không được làm !
  - Hoàn thiện mức khái niệm
    - E.g. đáp ứng tất cả các lớp của input
  - Hoàn thiện mức cấu trúc
    - E.g. không vi phạm các chuẩn!!!
- **Dễ hiểu (Rõ ràng)**
  - E.g. bởi các người không chuyên môn về máy tính
- **Nhất quán**
  - Không chứa các mâu thuẫn nội tại
  - Sử dụng nhất quán tất cả thuật ngữ
- **Có thứ bậc**
  - Chỉ rõ quan hệ quan trọng /ổn định của mỗi yêu cầu
- **Dễ kiểm tra**
  - Một tiến trình tồn tại để kiểm thử sự thỏa mãn mỗi yêu cầu
- **Dễ sửa đổi**
  - Có thể thay đổi không khó khăn
    - **Cấu trúc tốt và tham khảo chéo**
- **Dễ lần vết**
  - Nguồn gốc của mỗi yêu cầu rõ ràng
  - Gán nhãn mỗi yêu cầu cho sự tham khảo về sau này

Không có SRS nào là hoàn chỉnh!



Dùng ký pháp phù hợp

Source: *Adapted from Easterbrook & Callahan, 1997.*

- **Ngôn ngữ tự nhiên?**
  - **"Hệ thống sẽ báo cáo cho người điều khiển tất cả các lỗi phát sinh từ những chức năng then chốt hoặc xuất hiện trong suốt sự thực hiện của một quy trình then chốt và trong đó không có lỗi nào tìm được nguyên nhân."**

    (Điều này phỏng theo một đặc tả thực tế của NASA tại một trạm không gian quốc tế)
- **Hoặc một bảng quyết định (decision table)?**

Phát sinh trong chức năng then chốt?	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**
Xuất hiện trong quy trình then chốt?	**F**	**F**	**T**	**T**	**F**	**F**	**T**	**T**
Không có lỗi nào tìm ra nguyên nhân?	**F**	**F**	**F**	**F**	**T**	**T**	**T**	**T**
Báo cáo cho người điều khiển ?								

Nội dung SRS

Đặc tả yêu cầu phần mềm cần chú trọng:

- **Chức năng hóa.**
  - Nhiệm vụ phần mềm là làm gì?
- **Giao diện bên ngoài.**
  - Phần mềm tương tác thế nào với mọi người, phần cứng của hệ thống, các phần cứng khác, và phần mềm khác?
  - Giả định gì có thể phát sinh từ những thực thể bên ngoài này?
- **Yêu cầu thực thi.**
  - Tốc độ, sự sẵn dùng, thời gian đáp ứng, thời gian phục hồi của những chức năng phần mềm khác nhau và những thứ khác?
- **Các thuộc tính chất lượng.**
  - Tính khả chuyển, tính chính xác, khả năng bảo trì, tính bảo mật và những xem xét khác?
- **Các ràng buộc thiết kế phải tuân theo trong quá trình cài đặt.**
  - Có bất kỳ tác động nào của các chuẩn được yêu cầu, ngôn ngữ cài đặt, các chính sách toàn vẹn CSDL, giới hạn nguồn tài nguyên, môi trường vận hành và những thứ khác?

SRS không cần bao gồm …

Source: *Adapted from Davis, 1990, p183*

Những kế hoạch phát triển dự án

- **E.g. chi phí, đội ngũ nhân viên, lịch biểu, các phương pháp, công cụ, etc**
  - Chu kỳ sống của SRS là cho đến khi phần mềm lỗi thời
  - Chu kỳ sống của kế hoạch phát triển thì ngắn hơn nhiều

Những kế hoạch đảm bảo dự án

- **Quản lý cấu hình, kiểm tra & kiểm chứng, kế hoạch kiểm thử, đảm bảo chất lượng, etc**
  - Nhóm người tham gia khác nhau
  - Chu kỳ sống khác nhau

Các thiết kế

- **Lập yêu cầu và làm thiết kế có những người tham gia khác nhau**
- **Phân tích và thiết kế là những phạm vi chuyên môn khác nhau**
  - I.e. Nhà phân tích yêu cầu sẽ không thực hiện thiết kế!
- **Ngoại trừ những ràng buộc trong phạm vi ứng dụng của thiết kế**
  - e.g. Sự giao tiếp giới hạn giữa những hệ thống con khác nhau vì lý do bảo mật.

Các dạng lỗi điển hình

Source: *Adapted from Kovitz, 1999*

- **Nhiễu (Noise)**
  - Văn bản chứa những thông tin không liên quan đến bất kỳ đặc tính nào của vấn đề.
- **Im lặng (Silence)**
  - Đặc tính chính không được đề cập.
- **Đặc tả thừa (Over-specification)**
  - Văn bản mô tả các quyết định thiết kế một cách rất chi tiết hơn là mô tả vấn đề.
- **Mâu thuẫn**
  - Văn bản định nghĩa một đặctính duy nhất theo một số cách trái ngược nhau.
- **Mơ hồ**
  - Văn bản có thể thông dịch theo ít nhất là 2 cách khác nhau.
- **Tham khảo lùi (Forward Ref.)**
  - Sự tham khảo đến một thuật ngữ hoặc một đặc tính mà chưa hề được định nghĩa.
- **Mơ mộng (Wishful thinking)**
  - Văn bản mô tả siêu thực – một đặc tinh mà không thể kiểm tra được
- **Đặt yêu cầu với các người dùng**
  - Không thể yêu cầu người dùng thực hiện những việc nào đó, mà chỉ có thể giả sử rằng họ sẽ làm
- **Chơi trò xếp hình (Jigsaw puzzles)**
  - Thông tin then chốt được phân bố chéo trong tài liệu và có sự tham khảo chéo
- **Yêu cầu bề ngoài (Duckspeak)**

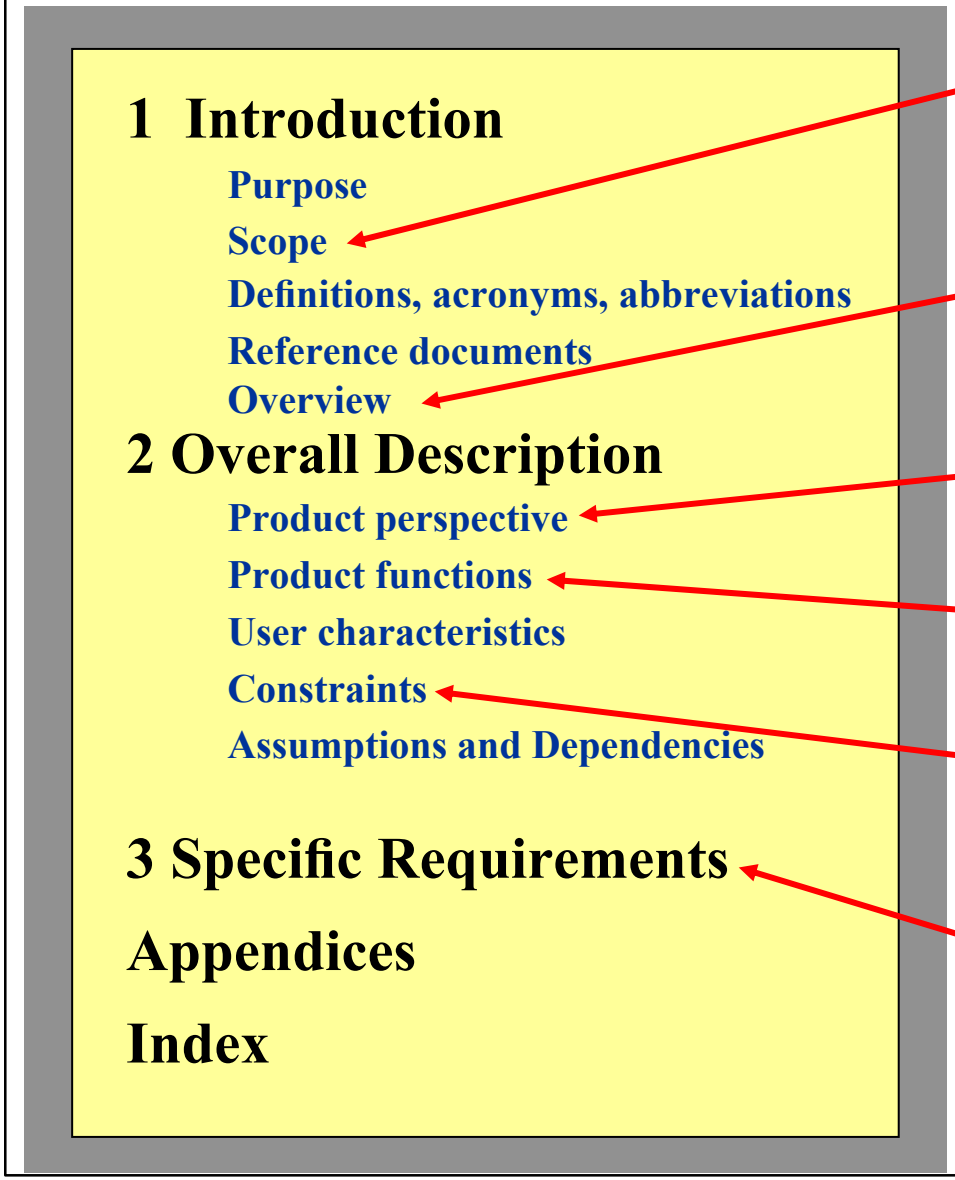
  Yêu cầu chỉ dùng để xác nhận theo chuẩn
- **Phát minh không cần thiết**
  - E.g. 'chức năng trình diễn input người dùng'
- **Thuật ngữ không nhất quán**
  - Phát minh và sau đó thay đổi thuật ngữ
- **Đặt trách nhiệm vào người phát triển**
  - i.e. làm cho người đọc phải rất vất vả để có thể đoán ra mục đích
- **Viết cho người đọc thù địch**
  - Có ít những người đọc dạng này hơn những người đọc thân thiện

Tổ chức các Yêu cầu

- **Cần một sự tổ chức logic cho tài liệu**
  - **Chuẩn IEEE cung cấp các kiểu mẫu khác nhau cho việc này**
- **Ví dụ các cấu trúc – tổ chức bởi …**
  - **…Tác nhân bên ngoài hoặc tình trạng bên ngoài**
    - **e.g., cho hệ thống điều khiển máy bay hạ cánh, mỗi kiểu khác nhau của tình trạng hạ cánh: gió mạnh, không có nhiên liệu, đường băng ngắn, etc**
  - **…Đặc tính hệ thống**
    - **e.g., cho một hệ thống điện thoại: chuyển hướng cuộc gọi, ngăn chặn cuộc gọi, nhóm cuộc gọi, etc**
  - **…Đáp ứng hệ thống**
    - **e.g., cho một hệ thống tính lương: phát sinh số tiền, báo cáo chi phí, in thông tin thuế;**
  - **…Đối tượng bên ngoài**
    - **e.g. cho một hệ thống thông tin thư viện, được tổ chức bằng cách phân loại sách**
  - **…Kiểu người dùng**
    - **e.g. cho một hệ thống hỗ trợ dự án: nhà quản trị, đội kỹ thuật, người quản lý, etc.**
  - **…Cách thức (mode)**
    - **e.g. cho xử lý từ (word processor): cách dàn trang (page layout mode), cách định dạng (outline mode), cách soạn thảo văn bản (text editing mode), etc**
  - **…Hệ thống con**
    - **e.g. cho tàu vũ trụ: điều khiển & kiểm soát, quản lý dữ liệu, truyền thông tin, thiết bị đo, etc.**

Chuẩn IEEE cho SRS

Source: *Adapted from IEEE-STD-830-1993* ***See also****, Blum 1992, p160*



Định nghĩa sản phẩm và lĩnh vực ứng dụng

Mô tả nội dung và cấu trúc của tài liệu yêu cầu

Mô tả tất cả giao diện bên ngoài: hệ thống, người dùng, phần cứng, phần mềm, hệ điều hành, các ràng buộc về phần cứng

Tổng quát các chức năng chủ yếu, như các trường hợp sử dụng

Mọi thứ hạn chế lựa chọn của nhà phát triển (như luật lệ, độ tin cậy, chỉ trích, giới hạn phần cứng, sự tương quan, …)

Tất cả yêu cầu viết ở đây (đây là phần thân của tài liệu). Chuẩn IEEE-STD cung cấp 8 mẫu khác nhau cho mục này.

Chuẩn IEEE STD Mục 3 (Ví dụ)

Source: *Adapted from IEEE-STD-830-1993.* ***See also****, Blum 1992, p160*

3.1 Yêu cầu giao diện bên ngoài

- **3.1.1 Giao diện người dùng**
- **3.1.2 Giao diện phần cứng**
- **3.1.3 Giao diện phần mềm**
- **3.1.4 Giao diện truyền thông tin**

3.2 Các yêu cầu chức năng

Mục này được tổ chức bởi chế độ vận hành (mode), lớp người dùng, đặc tính, etc. Chẳng hạn:

- **3.2.1 Mode 1**
  - **3.2.1.1 Yêu cầu chức năng 1.1**
  - **…**
- **3.2.2 Mode 2**
  - **3.2.1.1 Yêu cầu chức năng 1.1**
  - **…**

...

- **3.2.2 Mode n**
  - **...**

3.3 Các yêu cầu thực thi

Lưu ý các sự mô tả ở đây là trong ngữ cảnh của độ đo!

3.4 Các ràng buộc thiết kế

- **3.4.1 Các chuẩn thỏa thuận**
- **3.4.2 Các giới hạn phần cứng**

etc.

3.5 Các đặc tính của hệ thống phần mềm

- **3.5.1 Độ tin cậy**
- **3.5.2 Tính sẵn dùng**
- **3.5.3 Tính bảo mật**
- **3.5.4 Khả năng bảo trì**
- **3.5.5 Tính khả chuyển**

3.6 Các yêu cầu khác

Kết luận

- **Đặc tả yêu cầu nhằm một số mục đích:**
  - **Chuyển tải thông tin**
  - **Lập hợp đồng**
  - **Làm cơ sở cho việc kiểm tra**
  - **Làm cơ sở cho việc quản lý các thay đổi**
- **Đặc tả yêu cầu có một số dạng người dùng:**
  - **Có chuyên môn và không chuyên môn**
- **Đặc tả tốt thì rất khó viết**

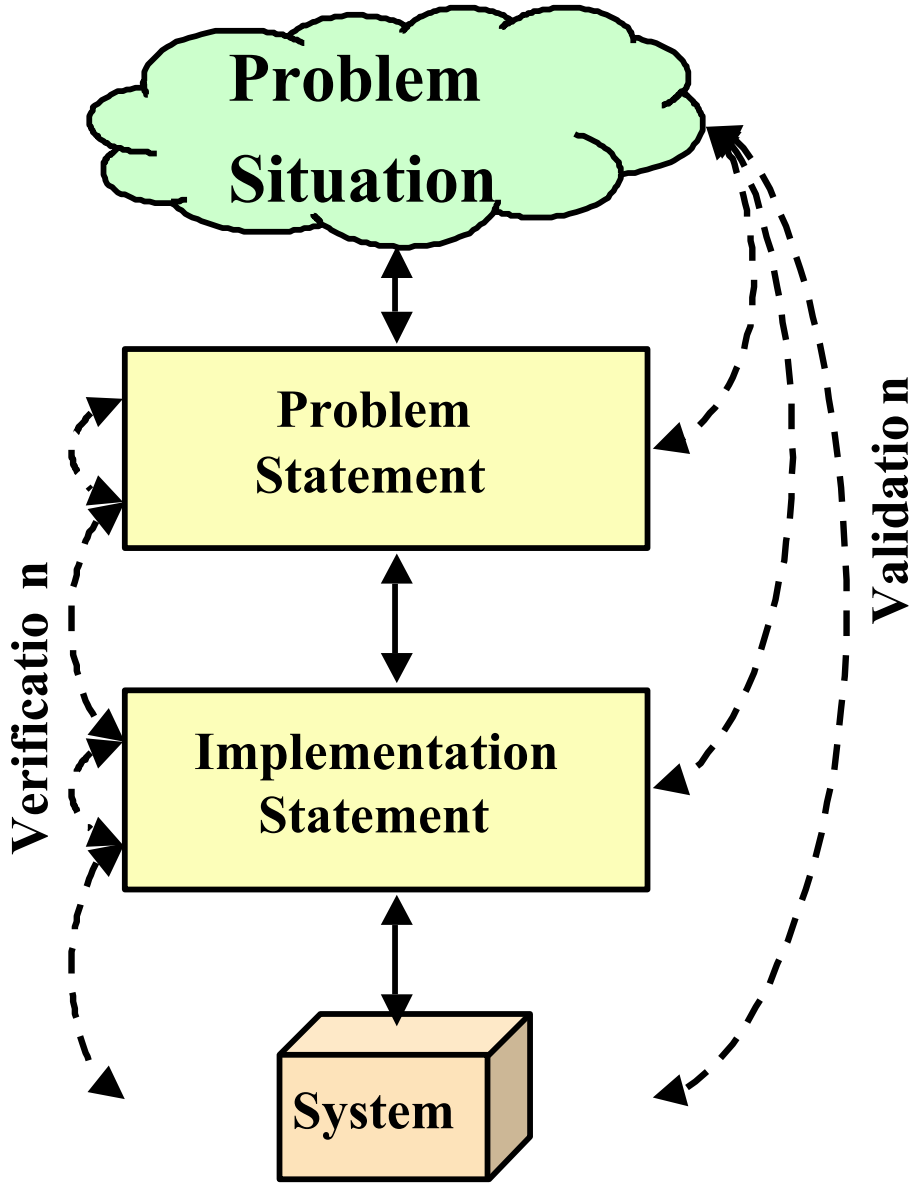
  Hoàn chỉnh, nhất quán, hợp lệ, không mơ hồ, dễ kiểm tra, dễ sửa đổi, dễ lần vết…
- **Dự án cần phải thay đổi**
  - **Tổng công sức đặt vào một đặc tả đúng sẽ phụ thuộc vào hậu quả có thể phát sinh của các lỗi trong yêu cầu**

Lecture 12:
Kiểm tra và Kiểm chứng
(Verification and Validation)

- **Khái niệm: Định nghĩa V&V**
- **Các kỹ thuật kiểm chứng**
  - Lập bản mẫu (Prototyping)
  - Phân tích mô hình (Model Analysis) (e.g. Model Checking)
  - Kiểm duyệt (Inspection)
- **Các kỹ thuật kiểm tra (Verification Techniques)**
  - Thực hiện lưu vết đặc tả (Specifications Traceable) (Bài 19)
  - Kiểm thử (Testing)
  - Kiểm duyệt mã lệnh (Code Inspection)
  - Phân tích mã lệnh (Code analysis)
- **V&V độc lập**

Verification and Validation

- **Kiểm chứng (Validation)**
  - "Chúng ta đã xây dựng đúng hệ thống ?"
  - Khai báo vấn đề đã thực sự nắm bắt được vấn đề thực tế?
  - Hệ thống đã đáp ứng được nhu cầu của tất cả đối tác?
- **Kiểm tra (Verification)**
  - "Chúng ta đã xây dựng hệ thống đúng?"
  - Thiết kế đáp ứng đặc tả?
  - Cài đặc đáp ứng đặc tả?
  - Hệ thống được phân phối sẽ thực hiện điều mà nó phải làm?
  - Các mô hình yêu cầu thống nhất với những mô hình khác?



Khái niệm: Tiêu chuẩnV&V

Source: *Adapted from Jackson, 1995, p170-171*

- **Sự khác biệt:**

↳ Domain Properties: những điều luôn luôn đúng trong lĩnh vực ứng dụng

↳ Requirements: những điều chúng ta mong là đúng trong lĩnh vực ứng dụng

↳ Specification: sự mô tả các hành vi mà chương trình cần thực hiện để đáp ứng với các yêu cầu

- **Hai tiêu chuẩn kiểm tra (verification)**

↳Chương trình (Program) thực hiện trên một máy tính (Computer) cụ thể đáp ứng với đặc tả (Specification)

↳ Đặc tả (Specification) được cho trong thuộc tính của lĩnh vực (Domain properties) thỏa mãn các yêu cầu (Requirements)

- **Hai tiêu chuẩn kiểm chứng (validation)**

↳ Chúng ta đã xem xét (và hiểu) tất cả các yêu cầu (Requirements) quan trọng?

↳ Chúng ta đã xem xét (và hiểu) tất cả các thuộc tính lĩnh vực(Domain properties) liên quan.

Ví dụ V&V

Ví dụ

Requirement R:

- "Phản lực chỉ có thể xảy ra khi máy bay đang chạy trên đường băng"

Domain Properties D:

- Xung lực bánh xe xảy ra khi và chỉ khi các bánh xe bật ra
- Các bánh xe bật ra khi và chỉ khi nó chạy trên đường băng

Specification S:

- Phản lực có thể xảy ra khi và chỉ khi có xung lực bánh xe

Kiểm tra

Phần mềm cho máy bay, P, thực thi trên máy tính trong buồng lái của máy bay, C, có hoàn toàn chính xác như đặc tả, S?

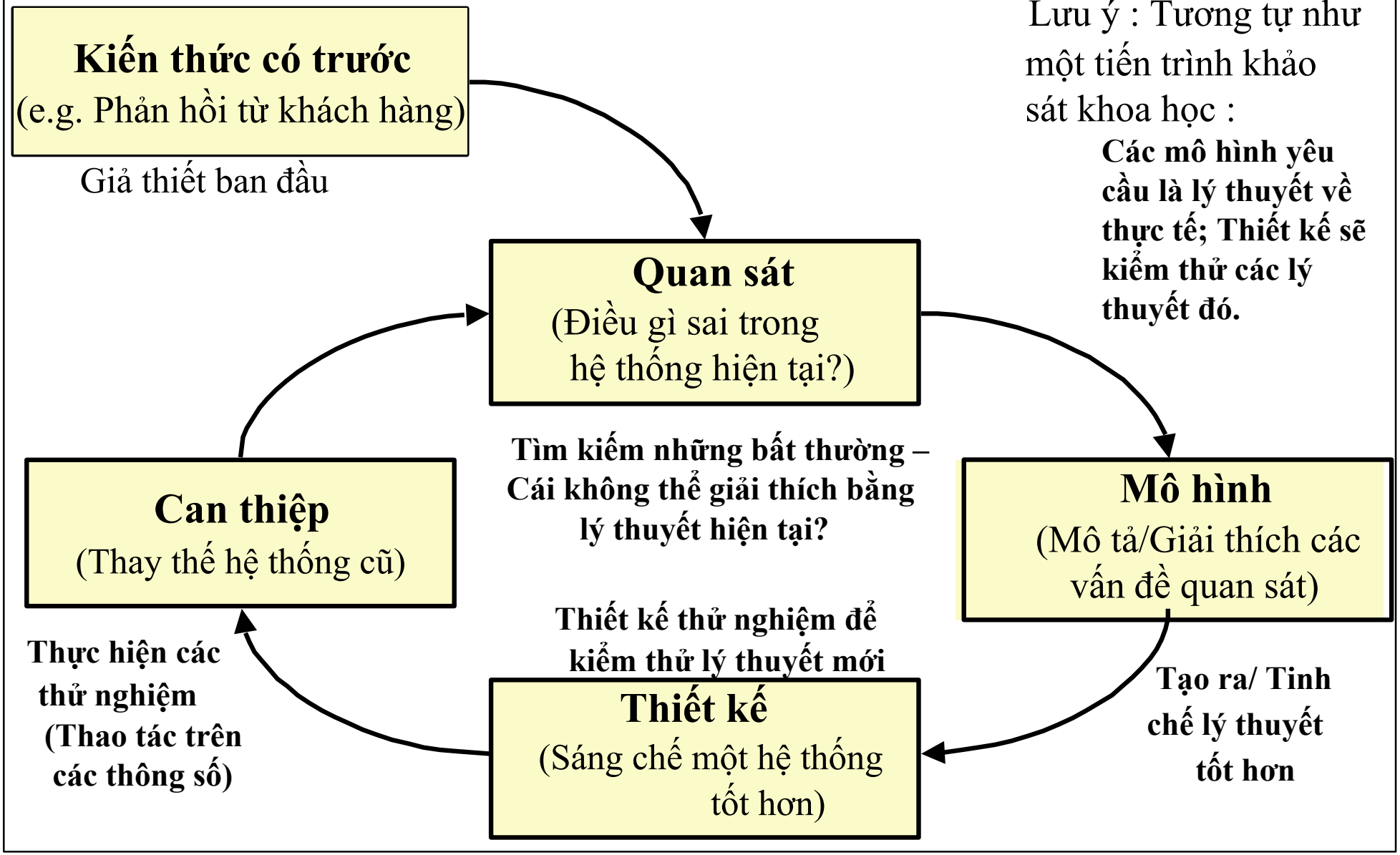
S, trong ngữ cảnh của giả thuyết D, có đáp ứng R?

Kiểm chứng

Giả thuyết của chúng ta, D, về lĩnh vực có thật chính xác? Có thiếu sót gì không?

Yêu cầu, R, có thật sự cần thiết? Có thiếu sót gì không?

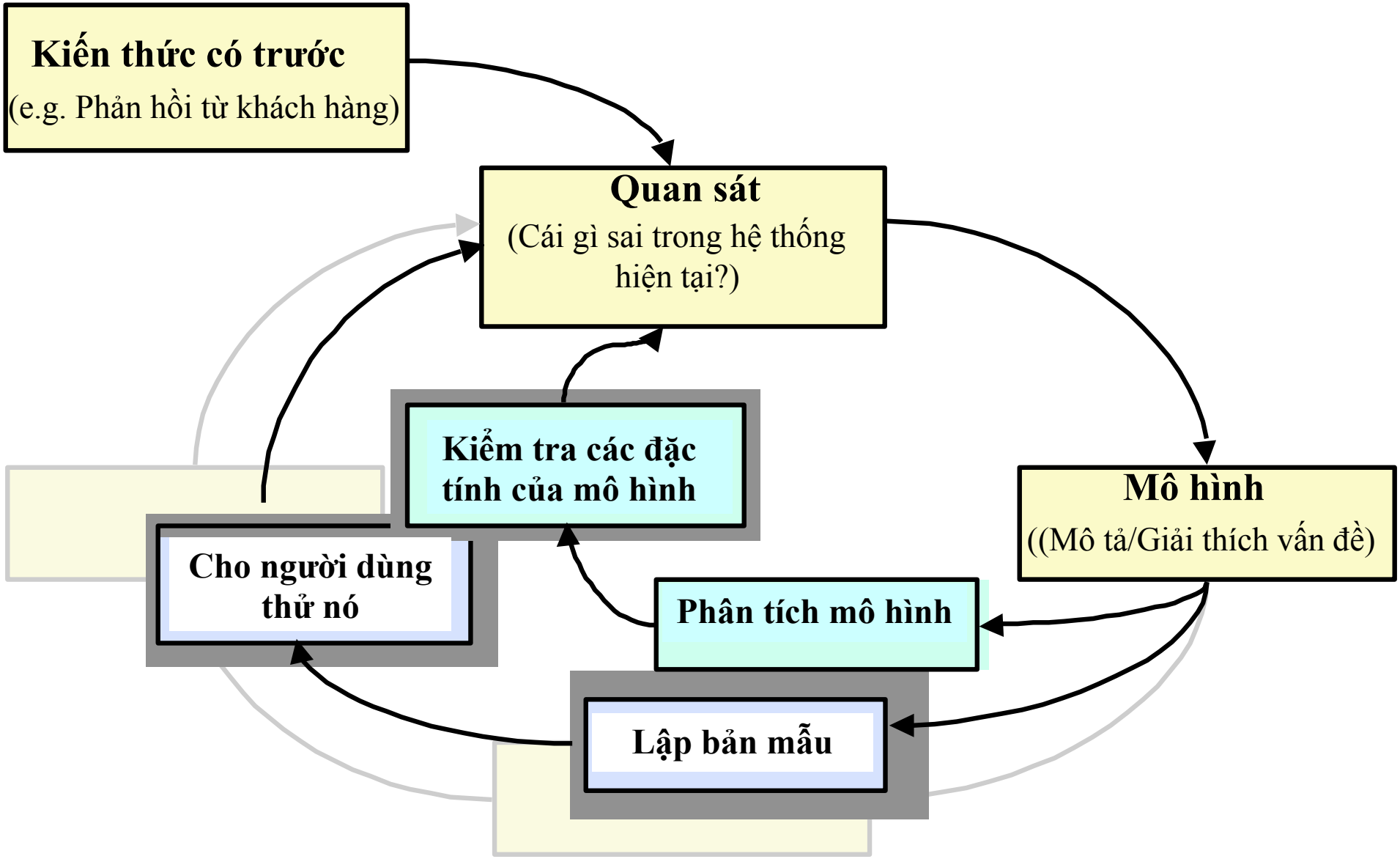
Chu trình điều tra



Lưu ý : Tương tự như một tiến trình khảo sát khoa học :

Các mô hình yêu cầu là lý thuyết về thực tế; Thiết kế sẽ kiểm thử các lý thuyết đó.

Chu trình điều tra nhanh



Lập bản mẫu

"Một bản mẫu phần mềm là một kiến trúc được cài đặt cụ thể trước để các khách hàng, người dùng hoặc nhà phát triển có thể hiểu rõ thêm về một vấn đề hay giải pháp của nó." **[Davis 1990]**

"*Lập bản mẫu là tiến trình xây dựng mô hình làm việc của hệ thống*"[Agresti 1986]

Hướng tiếp cận lập bản mẫu

Bản mẫu trình diễn

- Dùng để chứng minh khái niệm; giải thích các đặc tính thiết kế; etc.
- Giải thích, minh họa và thông báo – sau đó bỏ đi

Bản mẫu thăm dò

- Dùng để xác định vấn đề, thu thập nhu cầu, làm rõ mục tiêu, so sánh các lựa chọn thiết kế
- Không hình thức, không cấu trúc và cũng được bỏ đi.

Bản mẫu thử nghiệm

- Khai thác các đặc tính kỹ thuật; kiểm tra sự thích hợp của một kỹ thuật
- Thông thường không bao gồm người dùng/khách hàng

Bản mẫu tiến triển (e.g. "bản mẫu vận hành", "hệ thống lái máy bay"):

- Được phát triển khi thấy tiến trình tiếp diễn sẽ tương thích với hệ thống
- "Bản mẫu (Prototype)" như là một phân phối sớm, để tiếp tục cải tiến.

Dùng 1 lần hay tiếp tục phát triển ?

Bản mẫu dùng thử

- **Mục đích:**
  - để nghiên cứu nhiều hơn về vấn đề hoặc các giải pháp của nó…
  - bỏ đi khi đã thu được kiến thức mong muốn.
- **Cách dùng:** sớm hoặc trễ
- **Hướng tiếp cận:**
  - theo chiều ngang (horizontal) – thiết kế chỉ một tầng (e.g. UI)
  - "quick and dirty"
- **Thuận lợi:**
  - Phương tiên nghiên cứu cho sự hội tụ tốt hơn
  - Phân phối sớm → kiểm thử sớm → chi phí thấp
  - Thành công ngay cả khi nó bị thất bại!
- **Bất lợi:**
  - Nổ lực sẽ bị lãng phí nếu các yêu cầu thay đổi một cách nhanh chóng
  - Thường dùng thay thế một cách hợp lý các tài liệu yêu cầu
  - Có thể mong muốn của khách hàng là quá cao
  - Có thể phát triển thành sản phẩm cuối cùng

Bản mẫu tiến hóa

- **Mục đích**
  - để nghiên cứu nhiều hơn về vấn đề hoặc các giải pháp của nó…
  - …và giảm rủi ro bằng cách xây dựng sớm các bộ phận
- **Cách dùng:** phát triển dần; làm tiến hóa
- **Hướng tiếp cận:**
  - theo chiều dọc (vertical) – cài đặt từng phần của tất cả các tầng;
  - được thiết kế để mở rộng/thích ứng
- **Thuận lợi:**
  - Các yêu cầu không nhất thiết cố định
  - Trả về phiên bản sau cùng nếu phát hiện lỗi
  - Linh động(?)
- **Bất lợi:**
  - Có thể kết thúc với một hệ thống phức tạp, không cấu trúc – rất khó để bảo trì
  - Kiến trúc lựa chọn sớm có thể kém
  - Các giải pháp tối ưu thì không được bảo đảm
  - Thiếu kiểm soát và phương hướng

Brooks: "Kế hoạch để bỏ đi một bản mẫu – sẽ luôn luôn làm"

Phân tích mô hình (Model Analysis)

- **Kiểm tra:**
  - **"Mô hình có định dạng chuẩn?"**
  - **Các thành phần trong mô hình thống nhất với những cái khác?**
- **Kiểm chứng:**
  - **Dựng hoạt cảnh của mô hình trên các ví dụ nhỏ**
  - **Các thay đổi hình thức**
  - **Các câu hỏi 'Cái gì sẽ xảy ra nếu …?' ('What if'):**
    - nguyên nhân về hậu quả của những yêu cầu ngoại lệ;
    - nguyên nhân về sự tác động của những thay đổi có thể
    - "hệ thống có khi nào thực hiện như sau ..."
  - **Khảo sát trạng thái**
    - E.g. sử dụng kiểm tra mô hình để phát hiện ra sự đáp ứng của một số đặc tính

Cơ sở kiểm tra chéo với UML

Biểu đồ Use Case

- Mỗi use case có một người dùng?
  - Mỗi người dùng có ít nhất một use case?
- Có tài liệu cho mỗi use case không?
  - Sử dụng sơ đồ trình tự hoặc tương tự

Biểu đồ lớp (Class Diagrams)

- Sơ đồ lớp có nắm bắt được tất cả các lớp được đề cập đến trong những sơ đồ khác không?
- Mỗi lớp có các phương pháp để lấy/đặt các thuộc tính của nó không?

Biểu đồ trình tự (Sequence Diagrams)

- Mỗi lớp có trong sơ đồ lớp không?
- Mỗi thông điệp có thể được gửi đi không?
  - Có một quan hệ kết hợp lớp người gửi và lớp người nhận trong biểu đồ lớp không?
  - Có một phương pháp gọi trong lớp gửi cho mỗi thông điệp gửi không?
  - Có một phương pháp gọi trong lớp nhận cho mỗi thông điệp nhận không?

Biểu đồ chuyển trạng (StateChart)

- Mỗi biểu đồ chuyển trạng có nắm bắt được (các trạng thái của) một lớp không?
  - Lớp đó có trong biểu đồ lớp không?
- Mỗi bước chuyển (transition) có một sự kiện kích hoạt (trigger event) không?
  - Có rõ ràng đối tượng nào khởi tạo mỗi sự kiện?
  - Mỗi sự kiện có được liệt kê như một phương thức thuộc lớp của đối tượng đó trong sơ đồ lớp?
- Mỗi trạng thái có biểu diễn một sự kết hợp phân biệt của các giá trị thuộc tính?
  - Có rõ ràng sự kết hợp nào với những giá trị thuộc tính?
  - Tất cả những thuộc tính đó được chỉ ra trong biểu đồ lớp?
- Có phương pháp gọi trong biểu đồ lớp cho mỗi bước chuyển không?
  - …một phương pháp gọi sẽ cập nhật các giá trị thuộc tính cho một trạng thái mới?
  - …phương pháp gọi sẽ kiểm tra mọi điều kiện trên bước chuyển?
  - …phương pháp gọi sẽ thực hiện mọi hoạt động của bước chuyển?

Reviews, Walkthroughs, Inspections…

- **"Management reviews"**
  - **E.g. preliminary design review (PDR), critical design review (CDR), …**
  - **Dùng để nêu ra sự chắc chắn mà thiết kế báo hiệu**
  - **Tham gia bởi nhà quản trị và các nhà bảo trợ (khách hàng)**
  - **Thường chỉ là một "dog-and-pony show"**
- **"Walkthroughs"**
  - **Kỹ thuật của người phát triển (thường là không hình thức)**
  - **Được sử dụng bởi đội ngũ phát triển để cải tiến chất lượng sản phẩm**
  - **Tập trung vào việc tìm kiếm khuyết điểm**
- **"(Fagan) Inspections"**
  - **Một công cụ quản lý tiến trình (luôn hình thức)**
  - **Dùng để cải tiến chất lượng của tiến trình phát triển**
  - **Thu thập dữ liệu lỗi để phân tích chất lượng của tiến trình**
  - **Viết output thì quan trọng**
  - **Đóng vai trò chính trong việc huấn luyện đội ngũ kế thừa và truyền tải kinh nghiệm**

- **Các định nghĩa này thì không được thống nhất chung!**
  - **Các thuật ngữ khác đã dùng:**
    - Formal Technical Reviews (FTRs)
    - Formal Inspections
- **"Cách thức" có thể đa dạng:**
  - **Không hình thức:**
    - Gặp gỡ (over coffee),
    - Họp mặt nhóm thông thường,
    - etc.
  - **Hình thức:**
    - Lập lịch hội thảo,
    - Chuẩn bị thành viên tham dự,
    - Xác định thời gian,
    - Mô tả cách thức,
    - Lập tài liệu kết quả

Lợi ích của kiểm duyệt hình thức (formal inspection)

Source: *Adapted from Blum, 1992, Freedman and Weinberg, 1990, & notes from Philip Johnson.*

Formal inspection tác động tốt đối với việc lập trình:

- **Đối với lập trình ứng dụng:**
  - Hiệu quả hơn kiểm thử
  - Hầu hết chương trình reviewed thực hiện chính xác lần đầu tiên
  - So sánh: 10-50 lần thử kiểm tra/gặp lỗi
- **Dữ liệu từ các dự án lớn**
  - Giảm lỗi bởi một nguyên nhân : 5; (10 trong một số báo cáo)
  - Cải thiện hiệu quả : 14% đến 25%
  - Phần trăm lỗi đươc phát hiện bởi kiểm duyệt: 58% đến 82%
  - Giảm chi phí 50%-80% cho V&V (bao gồm cả chi phí kiểm duyệt)
- **Hiệu quả trên thu nhập của nhân viên:**
  - Tăng tinh thần, giảm thay đổi nhân sự
  - Lập lịch biểu và đánh giá tốt hơn (nhiều kiến thức về sơ lược các khuyết điểm)
  - Quản lý tốt hơn việc nhận định năng lực của nhân viên

Các lợi ích này cũng ứng dụng vào kiểm duyệt yêu cầu

- **Nhiều khảo sát theo lối kinh nghiệm phát hiện rằng các tiến trình kiểm duyệt rất đa dạng**
- **Các kết quả hòa lẫn trong các lợi ích liên quan của các tiến trình khác nhau**

Các vai trò (Roles)

Source: *Adapted from Blum, 1992, pp369-373*

Formal Walkthrough

- **Lãnh đạo Review**
  - Chủ trì cuộc họp
  - Bảo đảm các sự chuẩn bị
  - Giữ sự tập trung review
  - Báo cáo kết quả
- **Người ghi chép**
  - Giữ vết của các kết quả công bố
- **Reader**
  - Tổng kết các mẫu sản phẩm bởi các mẫu trong suốt quá trình review
- **Tác giả**
  - Người tham dự một cách tích cực (e.g. như reader)
- **Các Reviewers khác**
  - Công việc là tìm và viết ra báo cáo

Fagan Inspection

- **Người điều tiết (Moderator)**
  - Phải là một lập trình viên thành thạo
  - Cần được huấn luyện đặc biệt
  - Có thể đến từ dự án khác
- **Người thiết kế**
  - Các lập trình viên – người phát sinh thiết kế thông qua kiểm duyệt
- **Người viết code/cài đặt**
  - Các lập trình viên có trách nhiệm dịch từ thiết kế sang mã lệnh
- **Người kiểm thử**
  - Người có trách nhiệm viết/thực thi các tình huống kiểm thử

Lập cấu trúc sự kiểm duyệt

- **Checklist**
  - Dùng 1 danh sách các câu hỏi kiểm tra/vấn đề
  - review được cấu trúc bởi vấn đề trong danh sách
- **Walkthough**
  - Một người phải có mặt trong từng bước kiểm tra sản phẩm
  - review được cấu trúc bởi sản phẩm
- **Round Robin**
  - Mỗi reviewer lần lượt nếu ra một vấn đề
  - review thì được cấu trúc bởi đội review
- **Tốc độ Review**
  - Mỗi reviewer có 3 phút để xem xét lại 1 vấn đề, sau đó chuyển sang cho người kế tiếp
  - Tốt cho việc đánh giá khả năng hiểu thấu vấn đề!

Tại sao dùng sự kiểm duyệt?

- **Sự kiểm duyệt thì rất hiệu quả**
  - Kiểm duyệt mã lệnh thì tốt hơn việc kiểm thử để tìm khuyết điểm
  - Đối với đặc tả (Specifications), kiểm duyệt thì đã có sẵn (bạn không thể "kiểm thử" một đặc tả!)
- **Các khóa:**
  - Chuẩn bị: những người kiểm duyệt thì thực hiện riêng lẻ trước
  - Tập hợp họp mặt: người kiểm duyệt họp lại để kết hợp các danh sách khuyết điểm lại với nhau
  - Phân tích mỗi khuyết điểm, nhưng không tốn thời gian cố gắng sửa nó
  - Buổi họp đóng một vai trò quan trọng:
    - Người kiểm duyệt học hỏi từ người khác khi họ so sánh danh sách của họ
    - Bổ sung thêm những khuyết điểm chưa thấy được
  - Các hồ sơ có khuyết điểm trong khâu kiểm duyệt thì rất quan trọng cho tiến trình cải tiến
- **Mở rộng việc lựa chọn các kỹ thuật kiểm duyệt:**
  - Đóng vai trò gì trong buổi họp?
  - Cấu trúc buổi họp như thế nào?
  - Loại danh sách kiểm tra nào được dùng?

V&V độc lập

V&V được thực hiện bởi một nhà thầu riêng

- **Để đáp ứng V&V độc lập thì cần một quan điểm kỹ thuật độc lập.**
- **Chi phí khoảng giữa 5% đến 15% của chi phí phát triển**
- **Các khảo sát chỉ ra tăng gấp 5 lần lợi nhuận trên vốn đầu tư:**
  - Lỗi phát hiện sớm, rẻ hơn để sửa, rẻ hơn để thử lại
  - Đặc tả rõ hơn
  - Các nhà phát triển thích dùng các thực nghiệm tốt hơn.

3 kiểu của sự độc lập:

- **Độc lập quản lý:**
  - Phân chia trách nhiện từ việc phát triển các phần mềm
  - Có thể quyết định khi nào và ở đâu cần tập trung nổ lực V&V
- **Độc lập tài chánh:**
  - Phân chia chi phí và nguồn tài chính
  - Không mạo hiểm phân phối nguồn tài nguyên khi sắp sửa gặp khó khăn
- **Độc lập kỹ thuật:**
  - Tổ chưc nhân sự riêng biệt, để tránh thành kiến của các nhà phân tích
  - Dùng những công cụ và kỹ thuật khác nhau

Kết luận

- **Kiểm chứng (Validation) bạn đã giải quyết đúng vấn đề.**
  - **Prototyping – cho phản hồi của khách hàng sớm**
  - **Inspection – các chuyên gia lĩnh vực đọc đặc tả một cách kỹ lưỡng**
  - **Formal Analysis – phân tích toán học các mô hình của bạn**
  - **…cộng thêm các buổi họp & giao tiếp thường xuyên với các đối tác**
- **Kiểm tra (Verification) các bước trong công nghệ là đúng**
  - **Kiểm tra tính nhất quán – các mô hình có thống nhất với cái khác?**
  - **Khả năng lưu vết – làm cho việc thiết kế/viết mã lệnh/kiểm thử phản ánh đúng các yêu cầu?**
- **Sử dụng V&V thích hợp :**
  - **Lấy thông tin phản hồi từ khách hàng sớm nếu mô hình mới chỉ là phác thảo.**
  - **Phân tích và kiểm tra tính nhất quán nếu mô hình đã là bản đặc tả**
  - **V&V độc lập nếu hệ thống cần an toàn nghiêm ngặt.**

Lecture 13:
Sắp xếp yêu cầu ưu tiên
Requirements Prioritization

- **Tại sao cần sắp xếp thứ tự ưu tiên yêu cầu**
  - **Cần cân bằng các yêu cầu khác nhau để đạt được sự kết hợp tốt nhất (Trade-offs)**
- **Định hướng theo Chi phí –Giá trị (Cost-Value Approach)**
  - **Sắp xếp các yêu cầu theo tiêu chí Chi phí/Giá trị**
  - **Đánh giá quan hệ Chi phí/Giá trị (dùng AHP)**
- **Liệu các đối tác có đồng ý?**
  - **Hình dung sự chênh lệch khi sắp xếp ưu tiên**
  - **Giải quyết bất đồng**

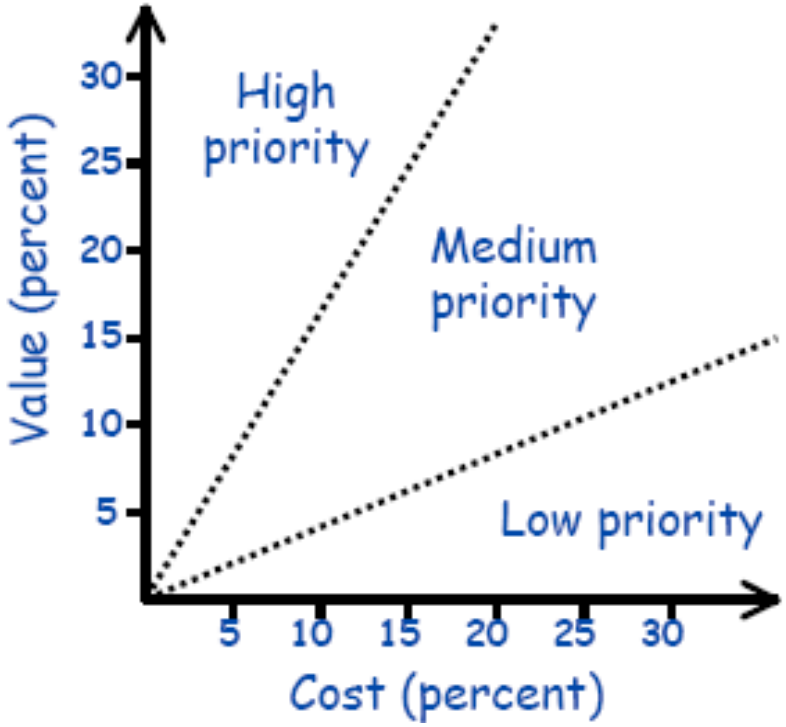
Cơ sở của sự ưu tiên

- **Cái gì cần được chọn để cài đặt**
  - Khách hàng (thường) hỏi quá nhiều về cách thức
  - Cân đối giữa thời gian tiếp thị với tổng số các chức năng
  - Quyết định đặc tính nào sẽ được phát hành kế tiếp
- **Đối với mỗi yêu cầu / đặc tính, cần hỏi:**
  - Nó quan trọng thế nào với khách hàng?
  - Chi phí để cài đặt nó là bao nhiêu ?
  - Sẽ có rủi ro nào khi cố gắng thực hiện nó?
- **Thực thi khẩn cấp:**
  - Một số yêu cầu bắt buộc "phải"
  - Một số yêu cầu nên dứt khoát loại bỏ
  - Điều này sẽ dẫn đến một phạm vi các "yêu cầu hợp lý" mà chúng ta có thể chọn lựa dễ dàng.

Một tiếp cận theo Chi phí/Giá trị (Cost-Value)

Source: *Adapted from Karlsson & Ryan 1997*

- **Tính toán lợi nhuận trên vốn đầu tư**
  - **Đánh giá tổng thể tầm quan trọng của mỗi yêu cầu đối với dự án**
  - **Ước lượng quan hệ về chi phí của mỗi yêu cầu**
  - **Tính toán sự thỏa hiệp giữa chi phí và giá trị**



Ước lượng Chi phí & Giá trị

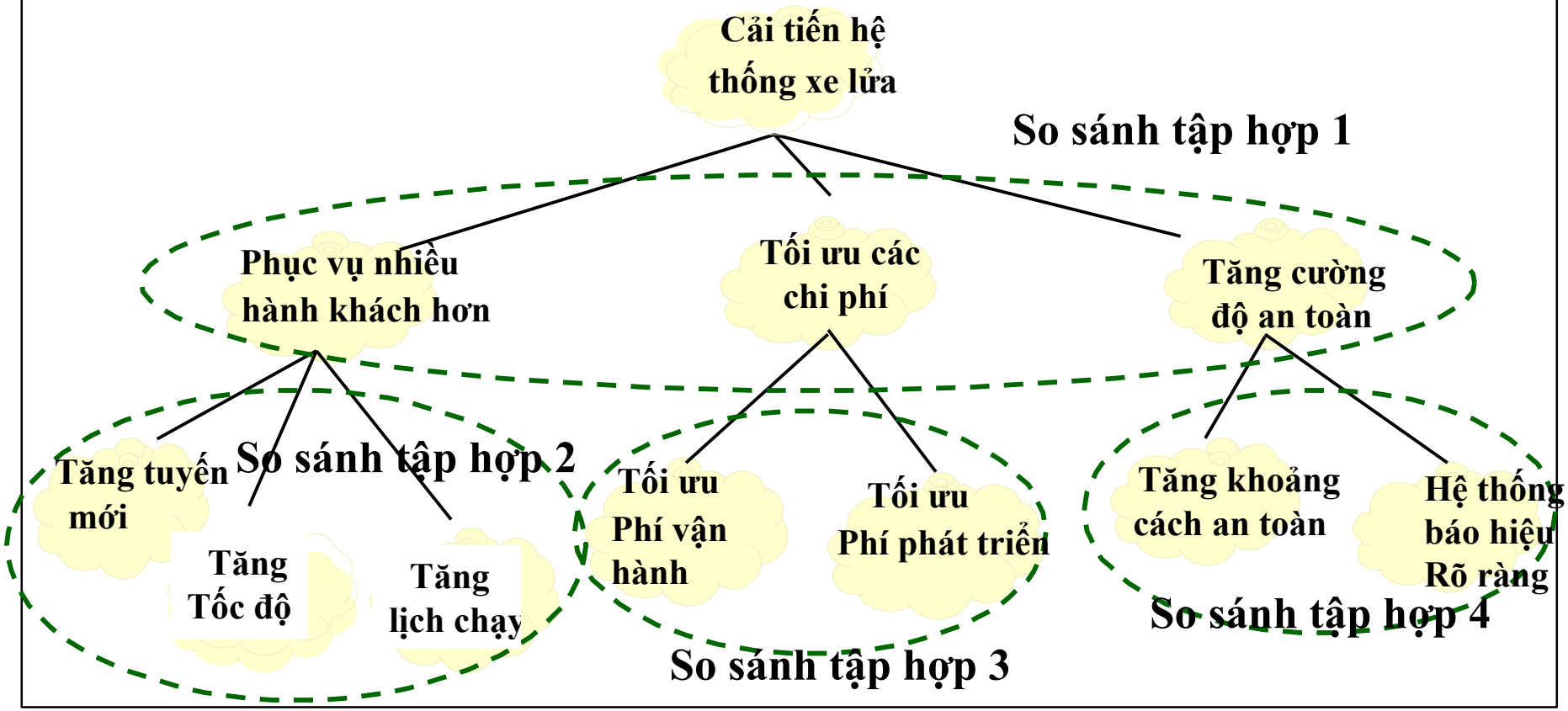
- **2 cách tiếp cận:**
  - **Định mức tuyệt đối (e.g. giá trị đồng ($))**
    - Đòi hỏi phải có kinh nghiệm chuyên môn
  - **Các giá trị liên quan (e.g. *ít/nhiều; một ít, một chút, rất*)**
    - Dễ dàng để làm rõ hơn
    - Sắp thứ tự ưu tiên dựa trên sự sắp xếp các vấn đề
- **Quá trình so sánh – chọn lựa**
  - **Cơ sở để sắp xếp – với mỗi cặp yêu cầu (i,j), xét i>j?**
    - E.g. bubblesort – bắt đầu với thứ tự ngẫu nhiên và hoán đổi mỗi cặp nếu sai thứ tự
    - Cần n*(n-1)/2 bước so sánh
  - **Dựng Cây thứ tự nhị phân (Binary Sort Tree)**
    - Cần O(n log n) bước so sánh
  - **Dựng cây phủ tối tiểu (Minimal Spanning Tree)**
    - Với mỗi cặp (Ri, Ri+1) : tính khoảng cách giữa chúng
    - Cần n-1 bước so sánh

Một vài rắc rối

- Khó để xác định mức độ chênh lệch
  - Dễ dàng để nói "x thì quan trọng hơn y"…
  - …hơn là ước lượng sự quan trọng nhiều như thế nào.
- Không phải mọi yêu cầu đều có thể so sánh được
  - E.g. mức độ trừu tượng khác nhau
  - E.g. chức năng chủ yếu vs. những mở rộng của khách hàng
- Các yêu cầu có thể không độc lập
  - Không có điểm chọn lựa giữa X và Y nếu chúng phụ thuộc lẫn nhau
- Các đối tác có thể không kiên định
  - E.g. Nếu X > Y, và Y > Z, thì có lẽ X > Z?
- Các đối tác có thể không thống nhất
  - Có sự đánh giá về chi phí/giá trị khác nhau với những dạng đối tác khác nhau.

Sự phân cấp thứ tự ưu tiên

- Nhóm các yêu cầu theo một cấu trúc phân cấp
  - E.g. Cây mục tiêu (A goal tree)
  - E.g. Cây NFR (NFR-Non function requirements tree)
- Chỉ thực hiện sự so sánh giữa các nhánh của cùng một nút:



Analytic Hierarchy Process (AHP)

Source: *Adapted from Karlsson & Ryan 1997*

- **Lập ma trận n x n (cho n yêu cầu)**
  - **Với mỗi phần tử (x,y) trong ma trận, nhập:**
    - 1 – nếu x bằng y
    - 3 – nếu x lớn hơn y một chút
    - 5 – nếu x lớn hơn y nhiều
    - 7 – nếu x rất lớn hơn y
    - 9 – nếu x cực kỳ lớn hơn y
    - (dùng các giá trị trung gian 2,4,6,8 nếu cần)
  - …và với phần tử (y,x) thì nhập giá trị nghịch đảo.
- **Đánh giá giá trị eigenvalues:**
  - **E.g. "trung bình trên các cột đã bình quân hóa"**
    - Tính tổng mỗi cột
    - Chia mỗi phần tử trong ma trận với số tổng cột của nó
    - Tính tổng mỗi hàng
    - Chia mỗi tổng hàng với số hàng
- **Cuối cùng sẽ có giá trị cho mỗi yêu cầu:**
  - **…cung cấp ước lượng phần trăm trên tổng giá trị dự án**

Ví dụ AHP - Đánh giá chi phí

Source: *Adapted from Karlsson & Ryan 1997*

	Req1	Req2	Req3	Req4
Req1	1	1/3	2	4
Req2	3	1	5	3
Req3	1/2	1/5	1	1/3
Req4	1/4	1/3	3	1

Bình quân hóa các cột

	Req1	Req2	Req3	Req4
Req1	0.21	0.18	0.18	0.48
Req2	0.63	0.54	0.45	0.36
Req3	0.11	0.11	0.09	0.04
Req4	0.05	0.18	0.27	0.12

Tổng các hàng

sum	sum/4
1.05	0.26
1.98	0.50
0.34	0.09
0.62	0.16

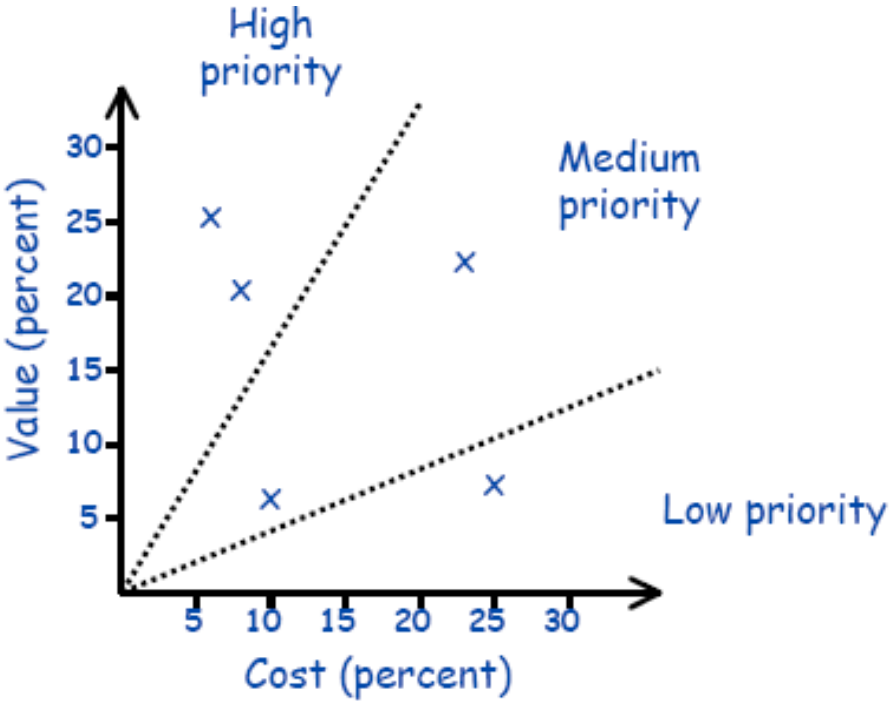
Kết quả

Req1 - 26% of the cost
Req2 - 50% of the cost
Req3 - 9% of the cost
Req4 - 16% of the cost

Vẽ đồ thị lợi nhuận trên vốn đầu tư (ROI graph)

Source: *Adapted from Karlsson & Ryan 1997*

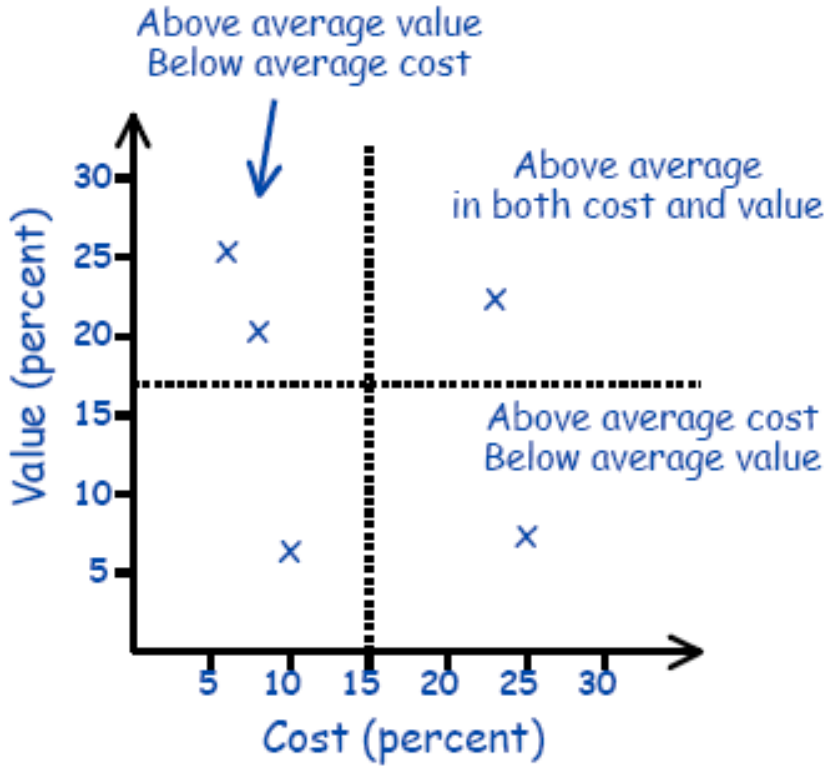
- **Thực hiện quá trình AHP hai lần:**
  - **Một lần để đánh giá quan hệ Giá trị**
  - **Một lần để đánh giá quan hệ Chi phí**
- **Dùng kết quả để tính toán tỷ số ROI :**

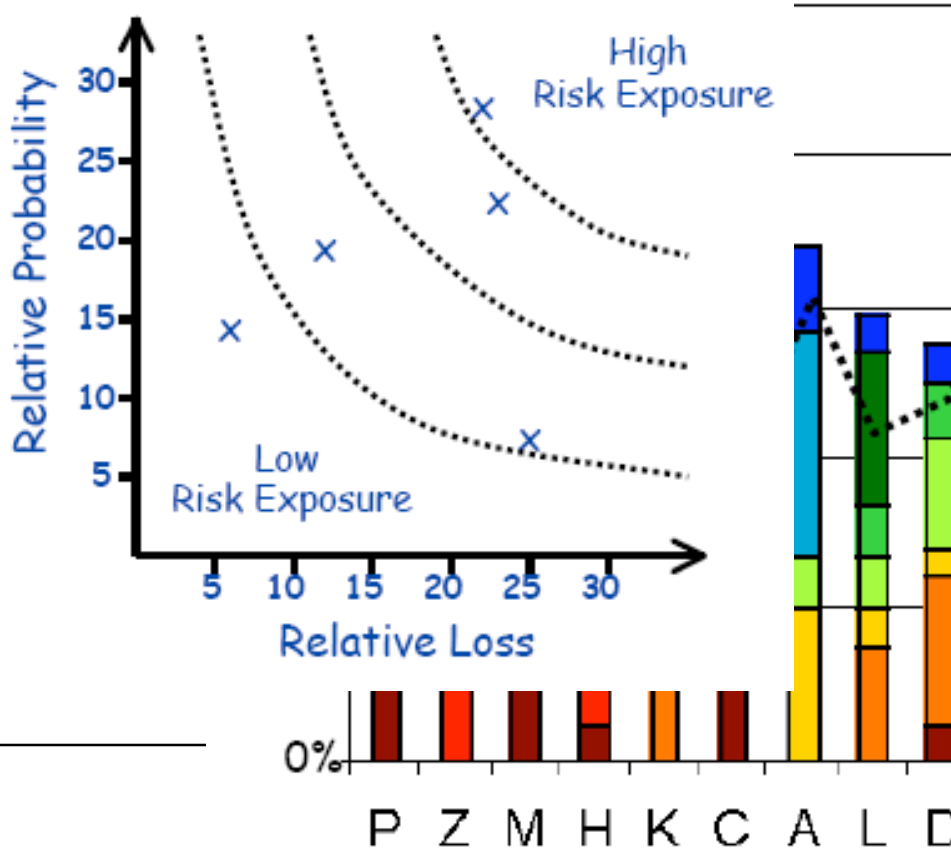


Tiêu chuẩn chọn lựa khác

Source: *Adapted from Park et al, 1999*

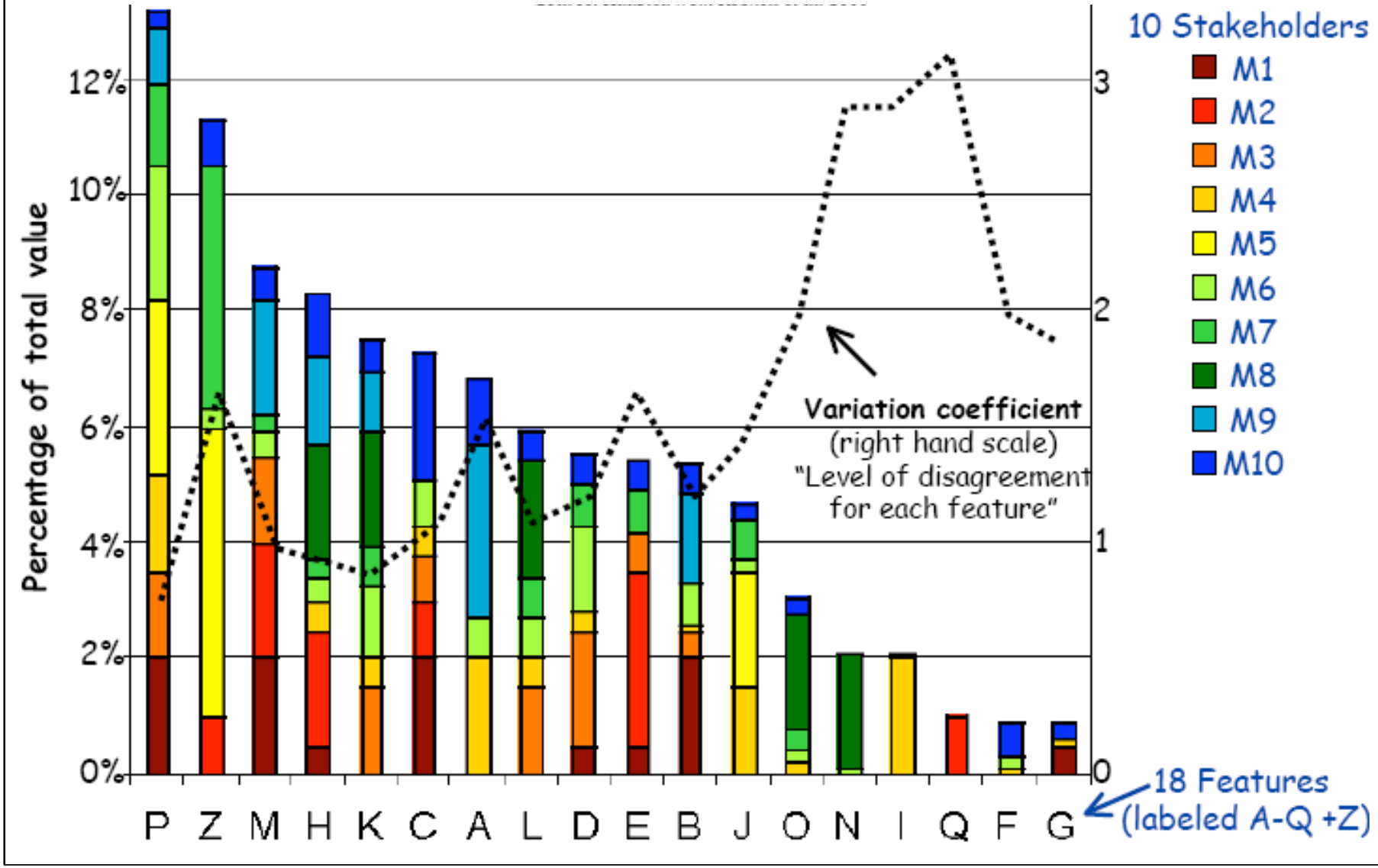
- Tỷ lệ ROI không phải là cách duy nhất để nhóm các yêu cầu





Minh họa "Giá trị" từ các đối tác

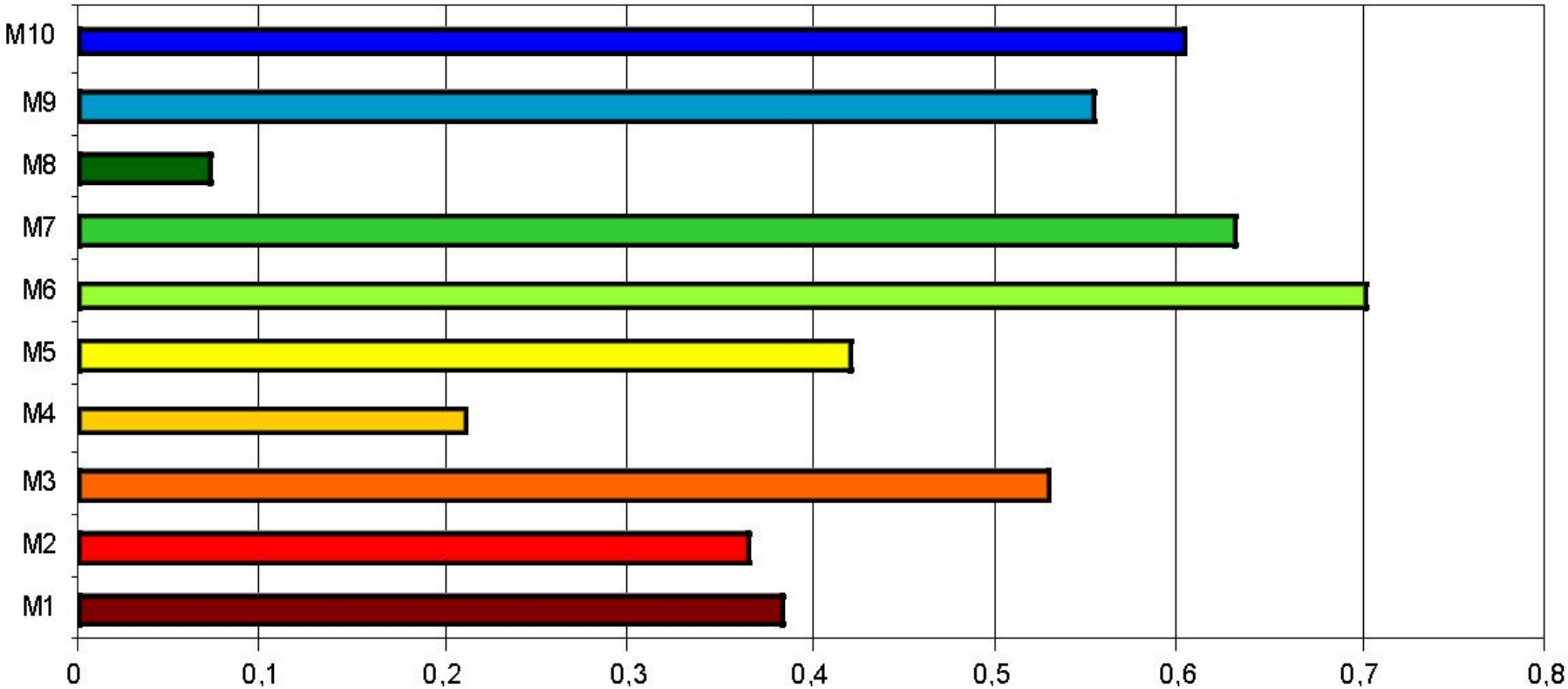
Source: *Adapted from Regnell et al, 2000*



Minh họa sự đáp ứng các đối tác

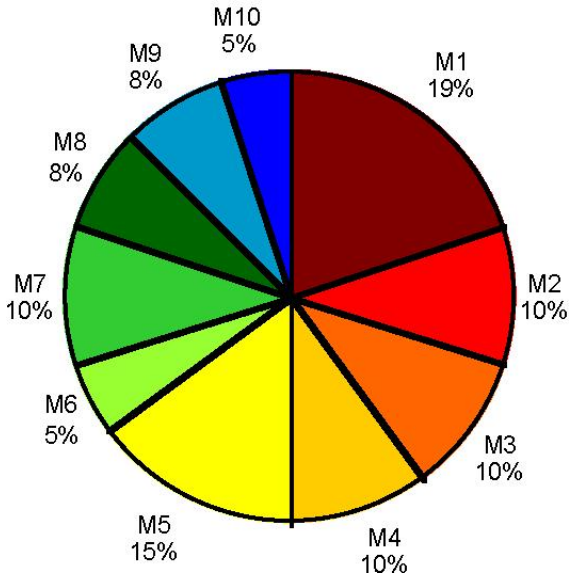
Source: *Adapted from Regnell et al, 2000*

- **Đồ thị cho thấy sự tương quan giữa mức ưu tiên của các đối tác và mức ưu tiên của nhóm.**
  - **Cũng có thể xem đây là "ảnh hưởng của mỗi đối tác trên nhóm"**



Cũng có thể tính trọng số mỗi đối tác

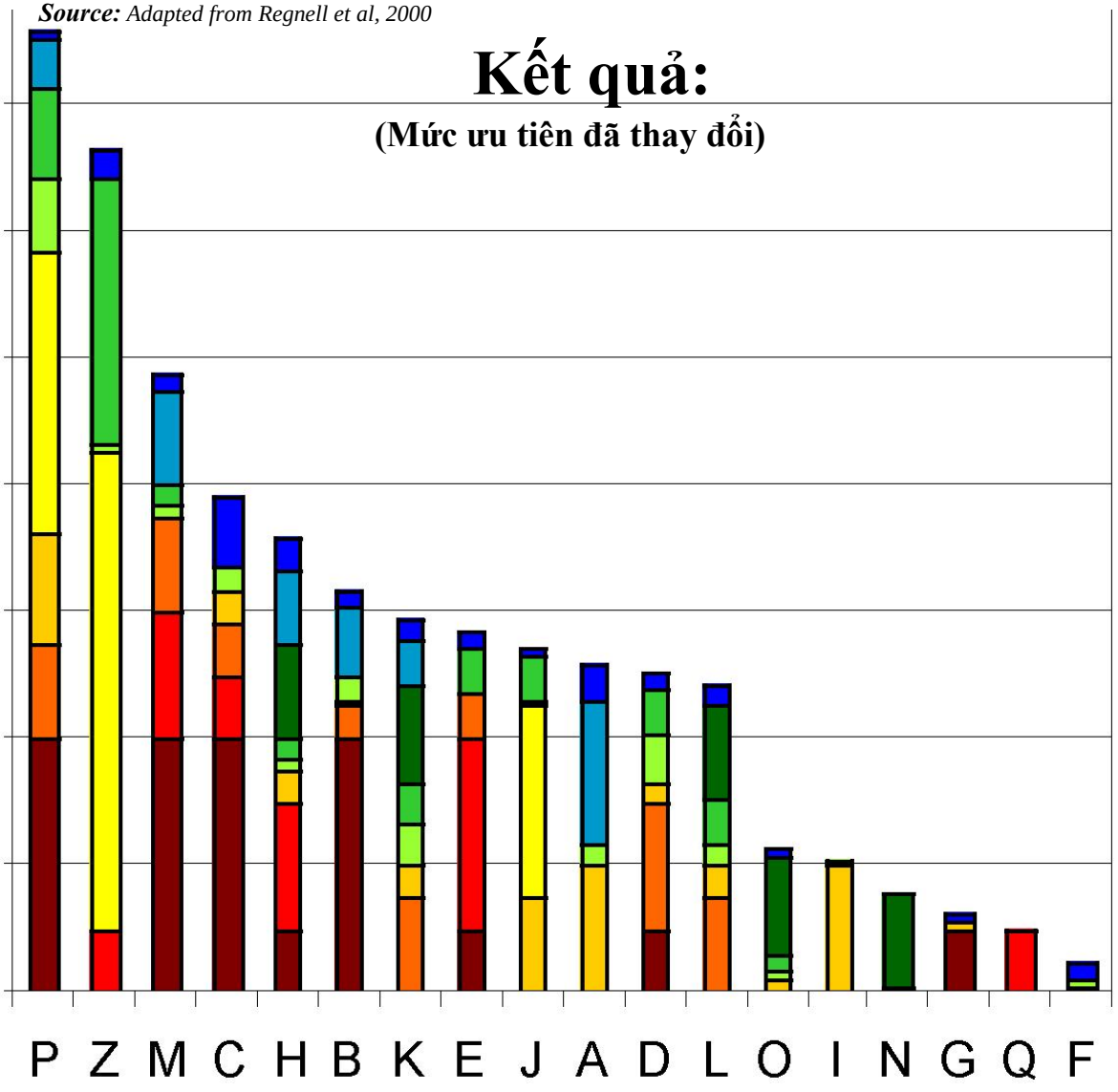
- **Trọng số của mỗi đối tác**
  - **E.g. để phản ánh sự tin cậy?**
  - **E.g. để phản ánh kích thước của kết quả bình chọn ?**
- **Ví dụ:**



Source: *Adapted from Regnell et al, 2000*

Kết quả:

(Mức ưu tiên đã thay đổi)



Giải quyết mâu thuẫn đối tác

Nguyên nhân gây mâu thuẫn

- **Deutsch (1973):**
  - Quyền kiểm soát tài nguyên
  - Sở thích và phiền toái (khiếu thẩm mỹ hoặc hành động của một thành viên đụnh chạm những người khác)
  - Các giá trị (có ý kiến cho rằng một giá trị hay tập giá trị nào đó thì nổi trội)
  - Niềm tin (nghi ngờ các sự kiện, thông tin, tính xác thực, etc.)
  - Bản chất quan hệ giữa các đối tác.
- **Robbins (1989):**
  - Sự truyền thông (trao đổi thông tin không đầy đủ, tin đồn, nhận thức hạn chế)
  - Cấu trúc (sự tương thích với mục tiêu, sự phân xử rõ ràng, kiểu lãnh đạo)
  - Các yếu tố cá nhân (giá trị cá nhân, tính cách cá nhân, ...)

Những kết quả đáng quan tâm

- Cư xử sai lệch & mâu thuẫn thường có trong một nhóm lập quyết định nhỏ
- Nhiều gây hấn và ít hợp tác trong giao tiếp thì có hạn chế :
  - tăng giao tiếp có khuynh hướng càng tăng thêm mâu thuẫn
- Những nhóm làm việc nhiều thành phần thường gặp nhiều mâu thuẫn;
- Những nhóm làm việc đồng nhất thường thích tạo ra những quyết định mang tính rủi ro cao
- Tác động của cá nhân thường bị phủ lấp bởi hoàn cảnh và giác quan

Cơ sở giải quyết mâu thuẫn

Đàm phán

- **… Là thăm dò sự cộng tác:**
  - Một thành viên cố gắng tìm một sự thỏa thuận có thể đáp ứng cho tất cả các thành viên khác.
- **Cũng được biết như:**
  - Hành vi hợp nhất
  - Thiết lập thương lượng
- **Khác với:**
  - phân phối/cạnh tranh sự thỏa hiệp

Cạnh tranh

- **Là phóng to chính bạn :**
  - Không quan tâm đến mức độ hài lòng của các thành viên khác.
  - Nhưng không nhất thiết phải gây thù hằn!
- **Cực điểm đến:**
  - Khi tất cả thành công cho một người được trả giá bởi những người khác
  - I.e trò chơi zero-sum.

Giải pháp "người thứ ba" (Third party)

- **Các thành viên yêu cầu hỗ trợ từ một nguồn bên ngoài**
  - Theo quy luật sách vở, chỉ rõ tác giả hoặc ném đồng xu.
  - Có thể dùng khi các phương pháp đàm phán và cạnh tranh đều thất bại.
- **Khiếu kiện:** người đại diện của mỗi thành viên phải đến tham dự.
- **Không khiếu kiện:** một quyết định được đưa ra bởi một nhân tố khác hơn những người đại diện.
- **May rủi:** e.g. ném đồng xu