ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THỐNG THÔNG TIN

Chương 1
TỔNG QUAN VỀ UML

Nội dung

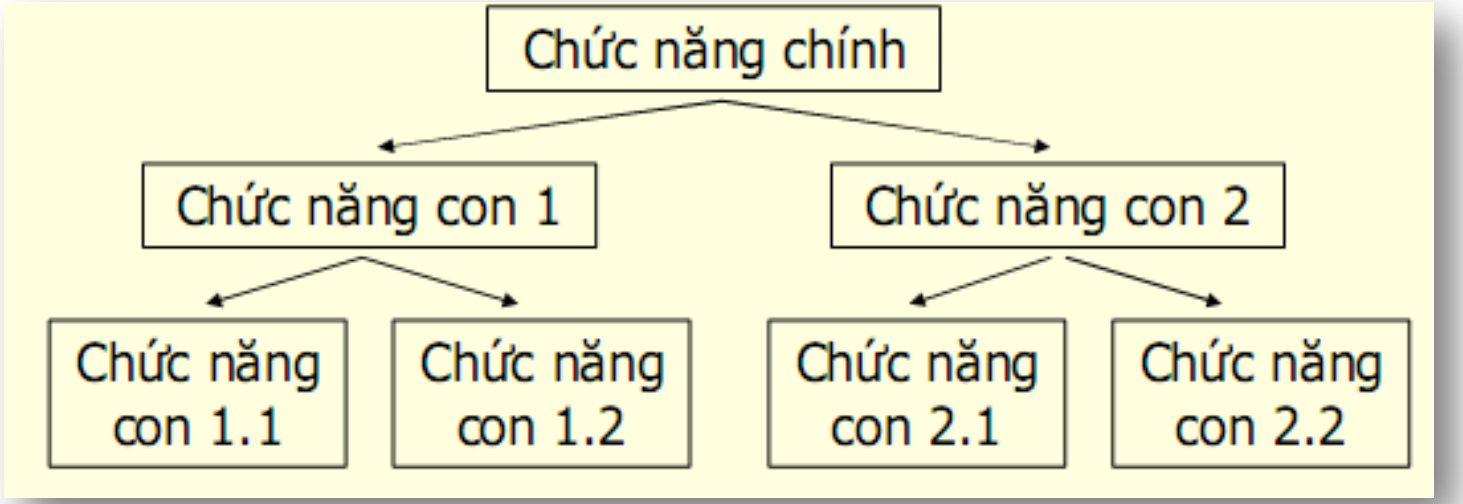
1. Tổng quan về phân tích thiết kế hướng đối tượng và các khái niệm cơ bản
2. Lịch sử UML
3. UML và các khái niệm liên quan
4. Mục tiêu của UML
5. UML và các giai đoạn phát triển phần mềm
6. Các thành phần của UML
7. Các quan hệ trong UML
8. Các khung nhìn và lược đồ UML

Phương pháp luận

- Phương pháp là tập hợp các bước cần thực hiện để đạt được một mục đích nào đó.
- Phương pháp luận là môn khoa học chuyên nghiên cứu về các phương pháp.
- Hầu hết các tài liệu mô tả quá trình xây dựng phần mềm là phương pháp.
  - Phương pháp hướng cấu trúc
  - Phương pháp hướng đối tượng

Phương pháp hướng cấu trúc

- Phương pháp này còn gọi là phương pháp cổ điển
- Chức năng được phân rã theo một hệ thống cấu trúc nhất định do người phân tích hệ thống đưa ra (cấu trúc phân nhánh, lặp...)



Phương pháp hướng cấu trúc

- Ưu điểm:
  - Phân rã được chức năng, quá trình hoạt động phần mềm được thực hiện từng bước như thế nào
  - Khá đơn giản và dễ hiểu.

- Khuyết điểm:
  - Việc dựa vào cấu trúc của quá trình chức năng dẫn đến khi chức năng hệ thống thay đổi, cấu trúc ấy có thể bị thay đổi rất nhiều, thậm chí phải thay đổi toàn bộ.
  - Sự tách biệt giữa mô hình chức năng và mô hình dữ liệu dẫn đến những chức năng hoàn toàn giống nhau nhưng xử lý những kiểu dữ liệu khác nhau phải được viết lại liên tục.
  - Thiếu linh động, khó mở rộng, không phù hợp cho phát triển các phần mềm lớn.

Phương pháp hướng đối tượng

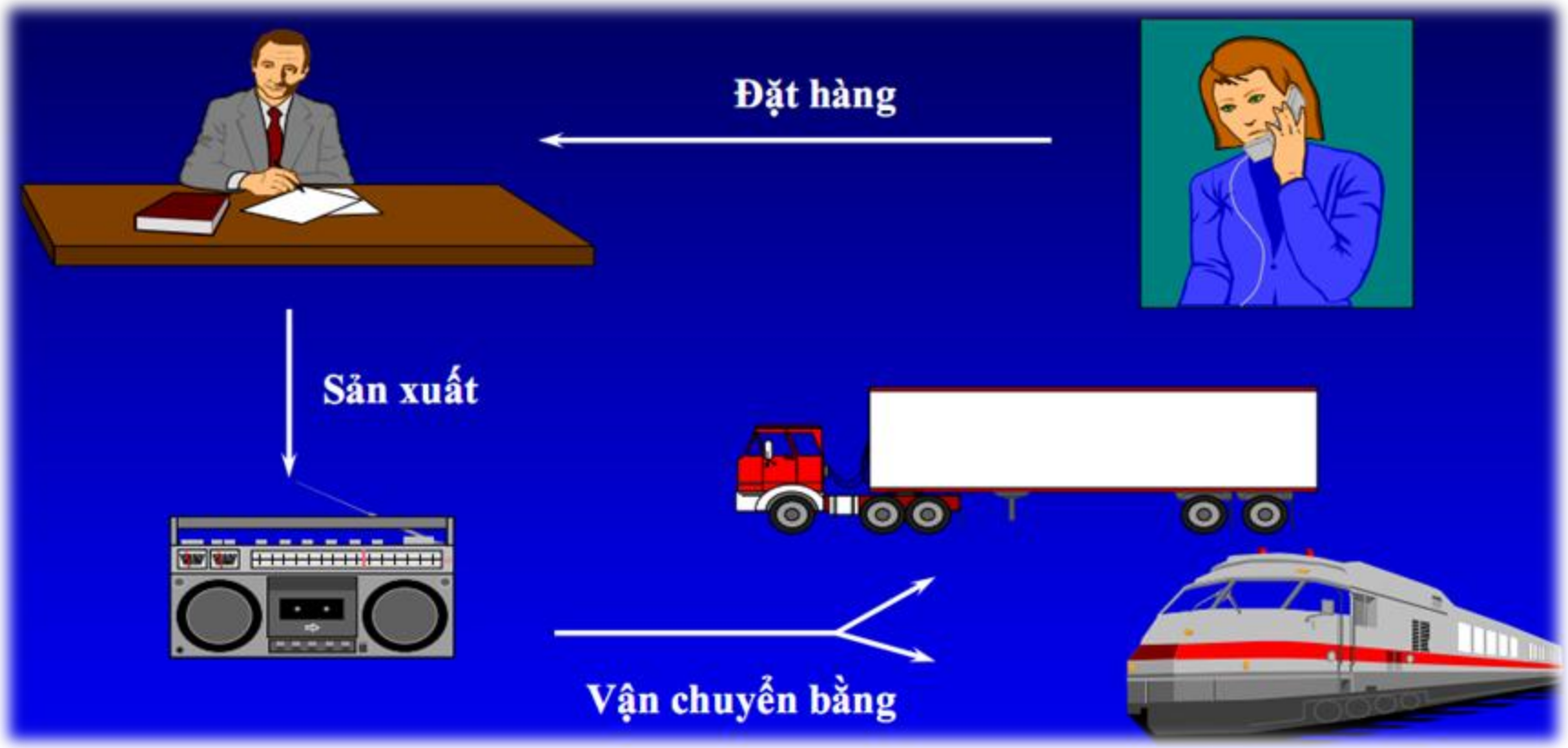
- Phương pháp này xác định rằng, cấu trúc thông tin trong hệ thống thông tin là ít thay đổi.
- Xem các thành phần trong hệ thống như các đối tượng ngoài đời thực.
- Các chức năng được xây dựng trên hệ cấu trúc đối tượng.

Phương pháp hướng đối tượng

- Ưu điểm:
  - Tăng cường tính tái sử dụng
  - Tăng cường tính mở rộng, không ảnh hưởng đến cấu trúc thông tin đã có, phần mềm trở nên linh động hơn.
  - Phù hợp với các hệ thống lớn: các đối tượng hoạt động độc lập
- Khuyết điểm:
  - Do dựa vào cấu trúc thông tin thay vì chức năng: Nếu cấu trúc này thay đổi (lĩnh vực ứng dụng thay đổi) thì việc xây dựng lại một hệ thống khác là không tránh khỏi. Do đó phương pháp này thiếu sự linh động với sự thay đổi của thông tin.

Các nguyên tắc cơ bản của hướng đối tượng

Ví dụ



Thế nào là trừu tượng hóa?

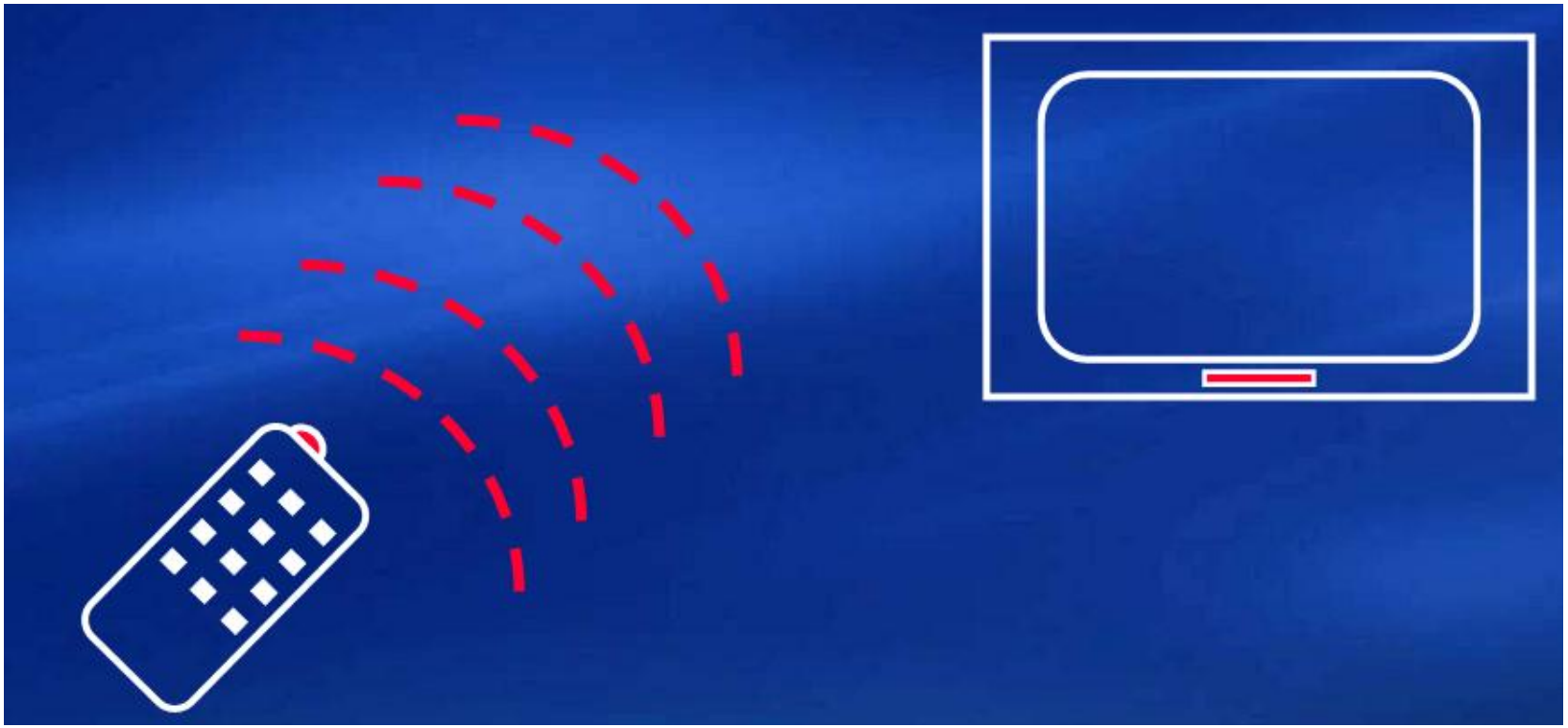
Người bán hàng

Khách hàng

Sản phẩm

Đóng gói là gì?

- Che dấu cài đặt bên trong với clients
  - Clients phụ thuộc vào interface

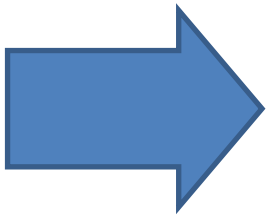


Tính đơn thể là gì?

- Phân chia nhỏ một vấn đề phức tạp thành nhiều phần nhỏ, đơn giản hơn quản lý được.

Nhận
đơn đặt hàng

Hệ thống xử lý
đơn đặt hàng



Thực hiện
đơn đặt hàng

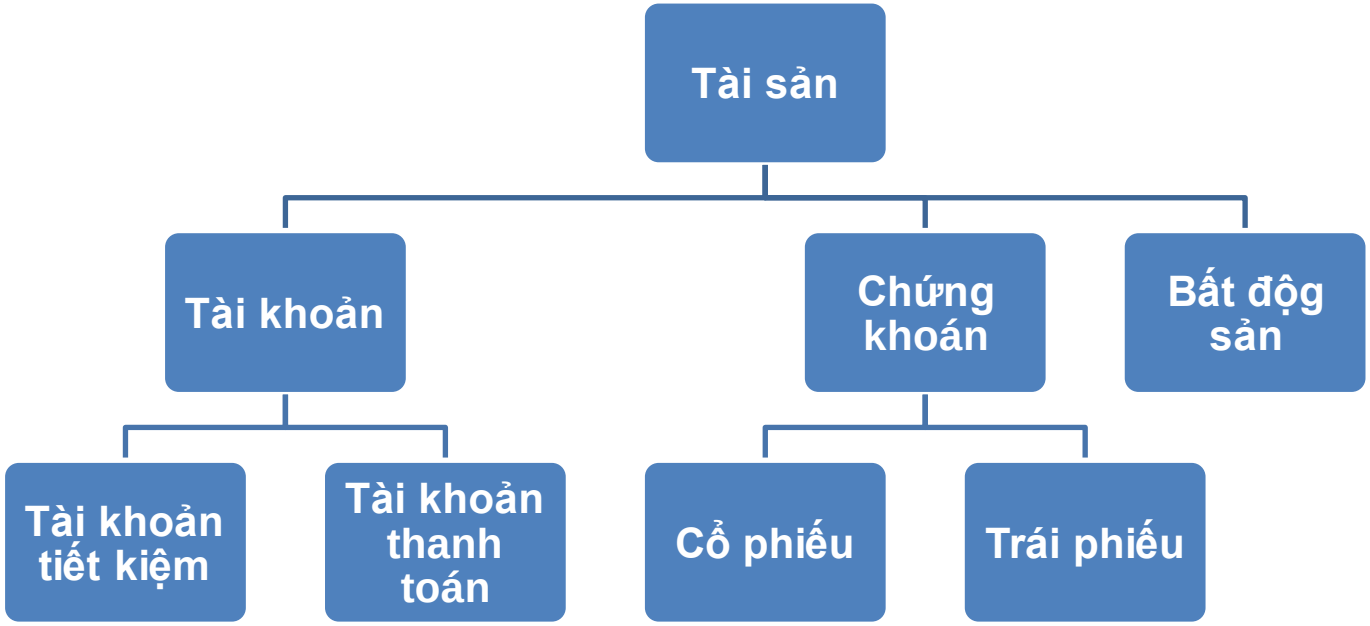
Tính tiền

Sự phân cấp là gì?

- Mức độ trừu tượng hóa

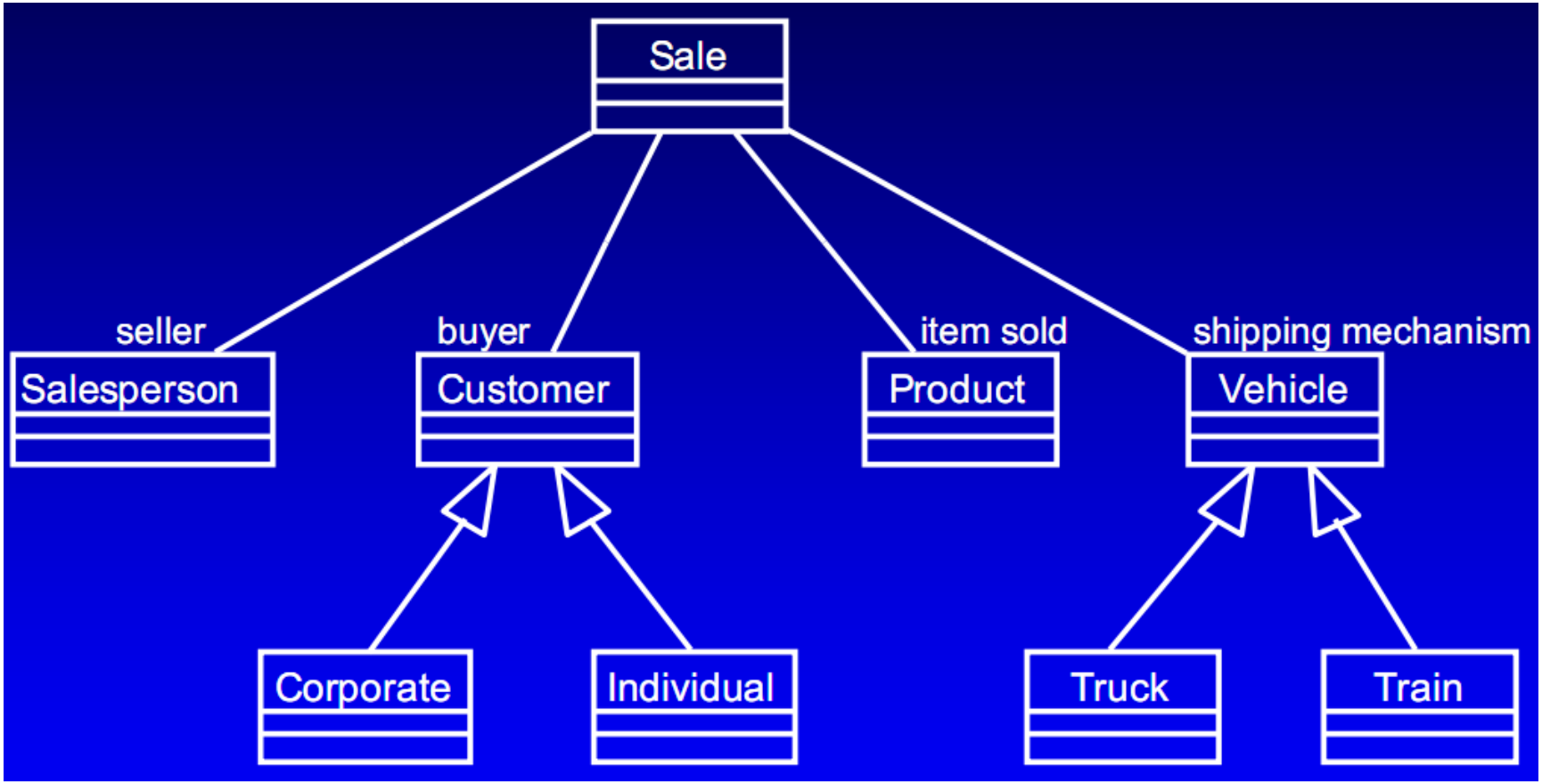
Tăng mức độ trừu tượng

Giảm mức độ trừu tượng

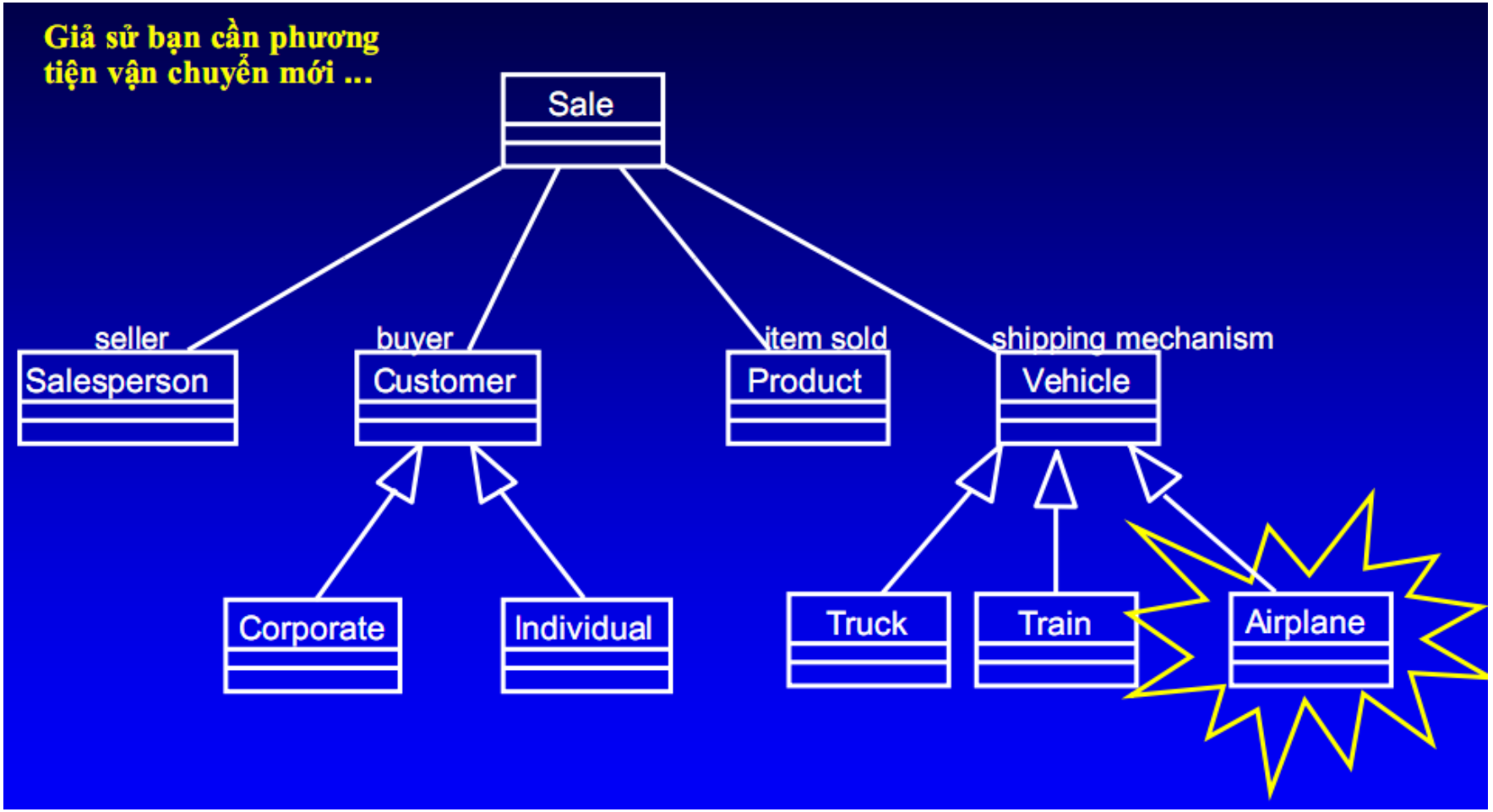


Các phần tử trên cùng một mức độ phải có cùng độ trừu tượng

Ví dụ (tt): Sơ đồ lớp



Ví dụ (tt)


Giả sử bạn cần phương tiện vận chuyển mới ...
Sale
seller
Salesperson
buyer
Customer
item sold
Product
shipping mechanism
Vehicle
Corporate
Individual
Truck
Train
Airplane

Yêu cầu của mô hình hóa

- Không một mô hình đơn nào là đầy đủ, cần thiết phải có các khung nhìn khác nhau
  - Cách tốt nhất để tiếp cận mỗi hệ thống phức tạp là đi từ tập các mô hình nhỏ độc lập gần nhất
- Mỗi mô hình có thể được diễn đạt tại các mức độ phức tạp (độ thô) khác nhau
- Những mô hình tốt là cần thiết
  - Cho sự giao tiếp giữa những người thực hiện dự án với người sử dụng nó
  - Đảm bảo sự hợp lý, đúng đắn trong kiến trúc

Nội dung

1. Tổng quan về phân tích thiết kế hướng đối tượng và các khái niệm cơ bản
2. **Lịch sử UML**
3. UML và các khái niệm liên quan
4. Mục tiêu của UML
5. UML và các giai đoạn phát triển phần mềm
6. Các thành phần của UML
7. Các quan hệ trong UML
8. Các khung nhìn và lược đồ UML

2. Lịch sử phát triển

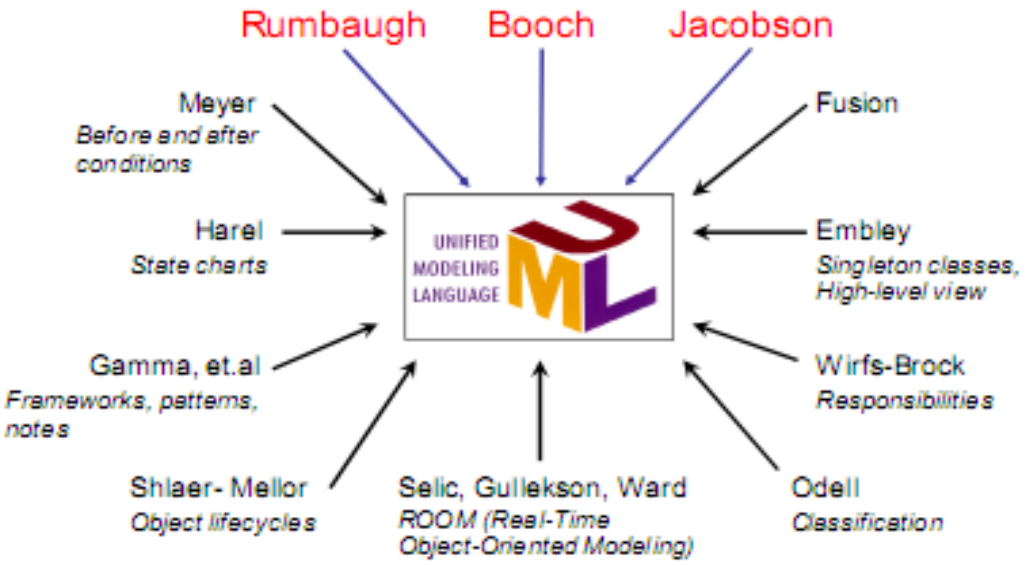
- Vào những năm 1980s, các ngôn ngữ lập trình hướng đối tượng xuất hiện
  - Smalltalk được chính thức chuyển từ phòng thí nghiệm ra phổ dụng
  - C++ được sinh ra
- Chuyển từ phương thức phân tích và thiết kế theo kiểu chức năng sang phương thức hướng đối tượng
- Các phương thức hướng đối tượng được phát triển vào những năm 1980s và giữa 1990s

Chuẩn hóa phương thức

- 1994- các phương thức đã gần như hoàn chỉnh và tương tự nhau
  - Cùng khái niệm (concepts): objects, classes, relationships, attributes, etc.
  - Cùng khái niệm nhưng lại dùng kí hiệu (notation) khác nhau
  - Mỗi phương thức đều có các mặt mạnh và yếu
- Yêu cầu chuẩn hóa

Phương thức được hợp nhất

- 3 phương pháp phổ biến nhất
  - OMT (Object Modeling Technique) [Jame Rumbaugh]
  - Booch [Grady Booch]
  - OOSE (Object-Oriented Software Enginering) [Ivar Jacobson



Hợp nhất và công bố UML

- Ba nhân vật nói trên chuyển tên của “Unified method” thành “Unified Modelling Language”
- Một cách duy nhất để giành được sự chấp thuận của các phương pháp là đem UML ra cộng đồng
- Năm 1996: thiết lập cộng đồng UML
  - Dưới sự lãnh đạo của Rational SC
  - Một số công ty lớn khác như: I-Logic, Intellicorp, IBM,….

Động cơ thúc đẩy sử dụng UML

- Sự chấp nhận UML - Những tập đoàn thành viên (cộng tác) của UML

Microsoft
Oracle
IBM
DECHP
TI
Unisys

I-Logix
IntelliCorp
Softteam
Sterling
Software
ICON Computing

MCI
Systemhouse
ObjectTime
PlatiumTechnology
Ptech
ReichTechnologies
...

Nội dung

1. Tổng quan về phân tích thiết kế hướng đối tượng và các khái niệm cơ bản
2. Lịch sử UML
3. **UML và các khái niệm liên quan**
4. Mục tiêu của UML
5. UML và các giai đoạn phát triển phần mềm
6. Các thành phần của UML
7. Các quan hệ trong UML
8. Các khung nhìn và lược đồ UML

3. UML là gì?

- UML là một ngôn ngữ dùng để
  - Trực quan hóa (visualizing)
  - Đặc tả (specifying)
  - Xây dựng (constructing)
  - Lập và cung cấp tài liệu (documenting)

  các kết quả (artifact) của hệ thống phần mềm.

- UML là một ngôn ngữ mô hình sử dụng các kí hiệu cho việc viết tài liệu, phân tích, thiết kế và thực hiện tiến trình phát triển hệ thống hướng đối tượng.

3. UML là gì? (tt)

- UML không phải là một phương pháp, đơn thuần nó chỉ là một ngôn ngữ kí hiệu
  - Là một tập các kí hiệu
  - Là một tập các luật (cú pháp, ngữ nghĩa, kiểm tra) cho việc sử dụng các kí hiệu
  - Dùng để hiển thị, đặc tả, xây dựng, làm tài liệu

3. UML là gì? (tt)

- **UML – Ngôn ngữ trực quan**
  - Ngôn ngữ mô hình hóa trực quan là một phát minh lớn của những năm 1990s trong việc thiết kế phần mềm.
  - Thể hiện trực quan là cách tốt nhất để giao tiếp và quản lí độ phức tạp
  - Làm cho mô hình hóa ngày càng gần hơn với việc cài đặt
- **UML – Ngôn ngữ đặc tả**
  - Xây dựng các mô hình một các tỉ mỉ, rõ ràng, đầy đủ ở các mức độ chi tiết khác nhau

3. UML là gì? (tt)

- **UML - Ngôn ngữ dùng để cấu trúc**
  - Các mô hình xây dựng bởi UML có thể ánh xạ tới một ngôn ngữ lập trình cụ thể như: Java, C++…
  - UML cho phép cập nhật một mô hình từ các mã thực thi.

- **UML - Ngôn ngữ dùng để lập và cung cấp tài liệu**
  - Tạo các tài liệu: yêu cầu của hệ thống, kiến trúc hệ thống, thiết kế, mã nguồn, kế hoạch dự án, kiểm thử, và quản lý phiên bản phát hành

Ưu điểm của UML

- UML hợp nhất các mô hình của Booch, OMT, và Jacobson
  - Thống nhất hầu hết các khái niệm của 3 phương pháp trên
  - Thêm các ký hiệu và khái niệm chưa có ở trong 3 phương pháp này

Những điểm ngoài phạm vi UML

- UML không là một phương pháp
- UML không xác định/hướng vào (address) toàn bộ quá trình
- UML không quy định cách tiếp cận vào việc xác định các lớp, các phương thức và phân tích các mô hình...
- UML không bao gồm bất kỳ quy tắc thiết kế hay cách thức giải quyết vấn đề nào

Nội dung

1. Tổng quan về phân tích thiết kế hướng đối tượng và các khái niệm cơ bản
2. Lịch sử UML
3. UML và các khái niệm liên quan
4. **Mục tiêu của UML**
5. UML và các giai đoạn phát triển phần mềm
6. Các thành phần của UML
7. Các quan hệ trong UML
8. Các khung nhìn và lược đồ UML

4. Mục tiêu của UML

- Cung cấp một ngôn ngữ mô hình hoá trực quan có sẵn và gợi tả (ready to use, expressive), từ đó có thể phát triển và thay đổi các mô hình một cách hiệu quả.
- Cung cấp các kỹ thuật chuyên môn để mở rộng các khái niệm cốt lõi (core concepts)
- Độc lập với các ngôn ngữ lập trình riêng biệt (particular) và các tiến trình phát triển

4. Mục tiêu của UML

- Cung cấp kiến thức cơ bản để hiểu ngôn ngữ mô hình hoá.
- Ủng hộ/khuyến khích sự phát triển của môi trường sử dụng công cụ hướng đối tượng (the OO tools market)
- Hỗ trợ các khái niệm cho sự phát triển ở mức độ cao hơn như là sự cộng tác (collaborations), khung (frameworks), mẫu (patterns), thành phần (components)…

Nội dung

1. Tổng quan về phân tích thiết kế hướng đối tượng và các khái niệm cơ bản
2. Lịch sử UML
3. UML và các khái niệm liên quan
4. Mục tiêu của UML
5. **UML và các giai đoạn phát triển phần mềm**
6. Các thành phần của UML
7. Các quan hệ trong UML
8. Các khung nhìn và lược đồ UML

5. UML và sự phát triển phần mềm

- Giai đoạn nghiên cứu sơ bộ
- Giai đoạn phân tích
- Giai đoạn thiết kế
- Giai đoạn xây dựng
- Giai đoạn thử nghiệm

Giai đoạn nghiên cứu sơ bộ

- UML đưa ra khái niệm Use Case để xác định các yêu cầu của khách hàng (người sử dụng).
  - Dùng biểu đồ Use case (Use Case Diagram) để nêu bật mối quan hệ cũng như sự giao tiếp với hệ thống.

- Qua phương pháp mô hình hóa Use case, các tác nhân (Actor) bên ngoài quan tâm đến hệ thống sẽ được mô hình hóa song song với chức năng mà họ đòi hỏi từ phía hệ thống (tức là Use case).

Giai đoạn phân tích

- Quan tâm đến quá trình trừu tượng hóa đầu tiên (các lớp và các đối tượng), cơ chế hiện hữu trong phạm vi vấn đề.
- Sau khi nhận biết được các lớp thành phần và mối quan hệ giữa chúng, các lớp cùng các mối quan hệ sẽ được miêu tả bằng công cụ biểu đồ lớp (class diagram).
- Sự cộng tác giữa các lớp nhằm thực hiện các Use case cũng sẽ được miêu tả nhờ vào các mô hình động (dynamic models).

Giai đoạn thiết kế

- Kết quả của giai đoạn phân tích được mở rộng thành một giải pháp kỹ thuật.
  - Bổ sung các lớp mới => tạo thành hạ tầng cơ sở kỹ thuật: Giao diện người dùng, các chức năng để lưu trữ các đối tượng trong ngân hàng dữ liệu, giao tiếp với các hệ thống khác, giao diện với các thiết bị ngoại vi và các máy móc khác trong hệ thống, ...
  - Các lớp thuộc phạm vi vấn đề có từ giai đoạn phân tích sẽ được "nhúng" vào hạ tầng cơ sở kỹ thuật này, tạo ra khả năng thay đổi trong cả hai phương diện: Phạm vi vấn đề và hạ tầng cơ sở.

  => Kết quả là bản đặc tả chi tiết cho giai đoạn xây dựng hệ thống.

Giai đoạn xây dựng

- Các lớp của giai đoạn thiết kế sẽ được biến thành những dòng code cụ thể trong một ngôn ngữ lập trình hướng đối tượng cụ thể
  - Phụ thuộc vào khả năng của ngôn ngữ được sử dụng.
  - Khi tạo ra các mô hình phân tích và thiết kế trong UML, tốt nhất nên cố gắng né tránh việc ngay lập tức biến đổi các mô hình này thành các dòng code.

Giai đoạn thử nghiệm

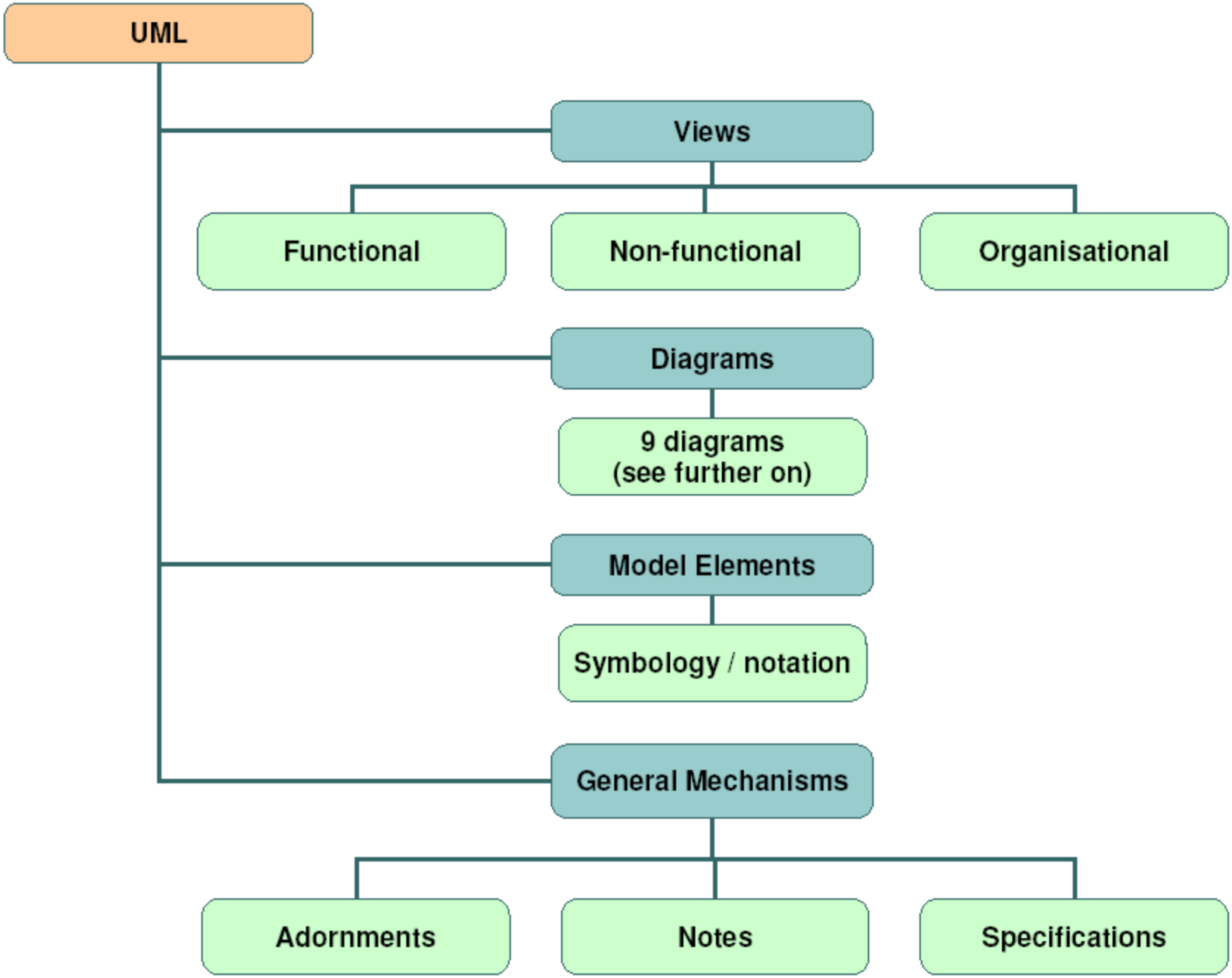
- Một hệ thống phần mềm thường được thử nghiệm qua nhiều giai đoạn và với nhiều nhóm thử nghiệm khác nhau.
- Các nhóm sử dụng nhiều loại biểu đồ UML khác nhau làm nền tảng cho công việc của mình
  - Thử nghiệm đơn vị sử dụng biểu đồ lớp (class diagram) và đặc tả lớp
  - Thử nghiệm tích hợp thường sử dụng biểu đồ thành phần (component diagram) và biểu đồ cộng tác (collaboration diagram)
  - Thử nghiệm hệ thống sử dụng biểu đồ Use case (use case diagram) để đảm bảo hệ thống có phương thức hoạt động đúng như đã được định nghĩa từ ban đầu

Nội dung

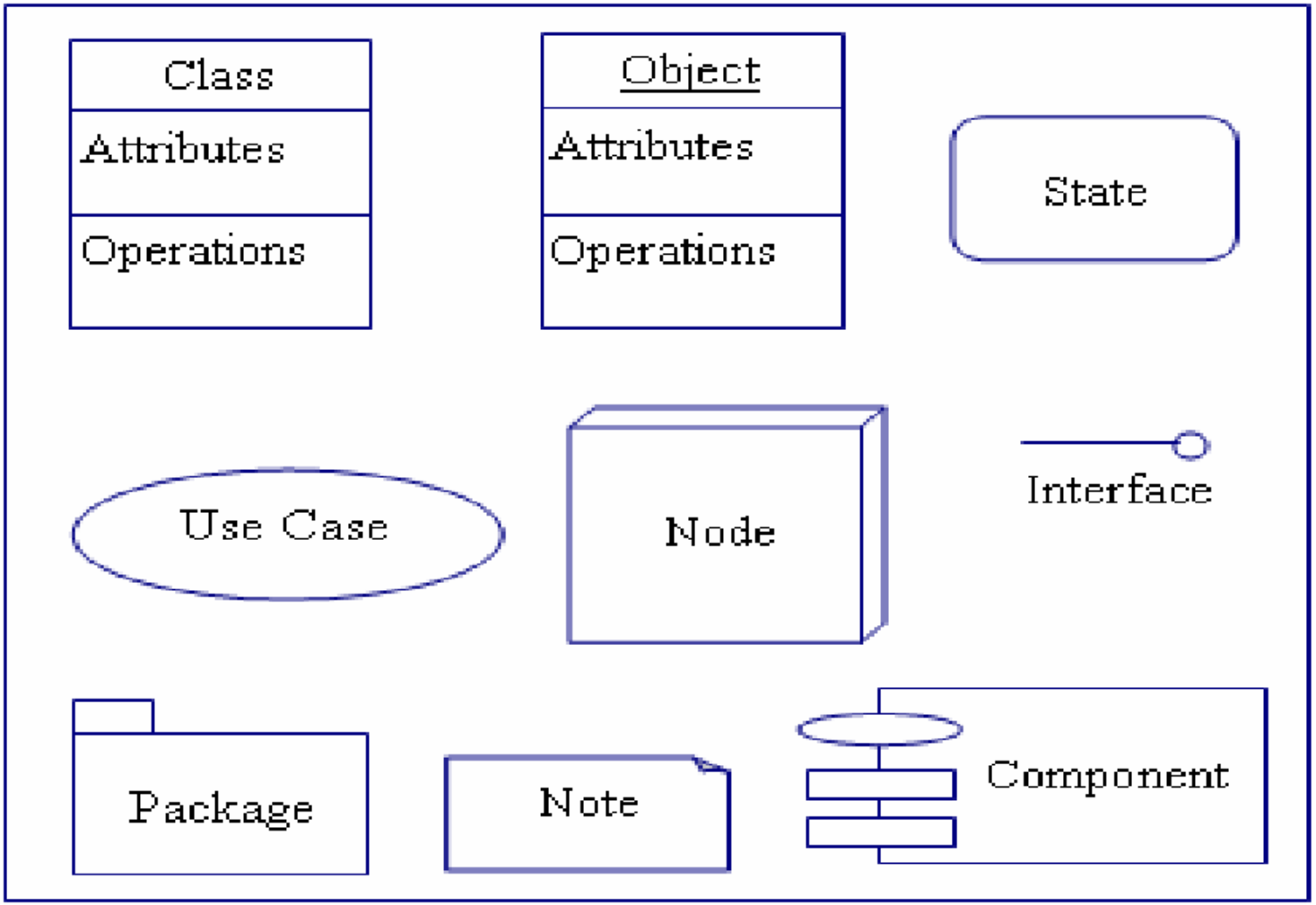
1. Tổng quan về phân tích thiết kế hướng đối tượng và các khái niệm cơ bản
2. Lịch sử UML
3. UML và các khái niệm liên quan
4. Mục tiêu của UML
5. UML và các giai đoạn phát triển phần mềm
6. **Các thành phần của UML**
7. Các quan hệ trong UML
8. Các khung nhìn và lược đồ UML

6. Các thành phần của UML

- **Khung nhìn (view)**: chỉ ra những khía cạnh khác nhau của hệ thống cần mô hình hóa.
- **Biểu đồ (diagram)**: các hình vẽ miêu tả nội dung trong một khung nhìn.
  - UML có tất cả 9 loại biểu đồ
- Phần tử mô hình hóa (model element): Các khái niệm sử dụng trong các biểu đồ được gọi là các phần tử mô hình
- Cơ chế chung
  - Cung cấp thêm những lời nhận xét bổ sung
  - Các thông tin cũng như các quy tắc ngữ pháp chung về một phần tử mô hình
  - Cung cấp thêm các cơ chế để có thể mở rộng ngôn ngữ UML cho phù hợp với một phương pháp xác định (một quy trình, một tổ chức hoặc một người dùng).


UML
Views
Functional
Non-functional
Organisational
Diagrams
9 diagrams
(see further on)
Model Elements
Symbology / notation
General Mechanisms
Adornments
Notes
Specifications

Phần tử mô hình



Phần tử mô hình

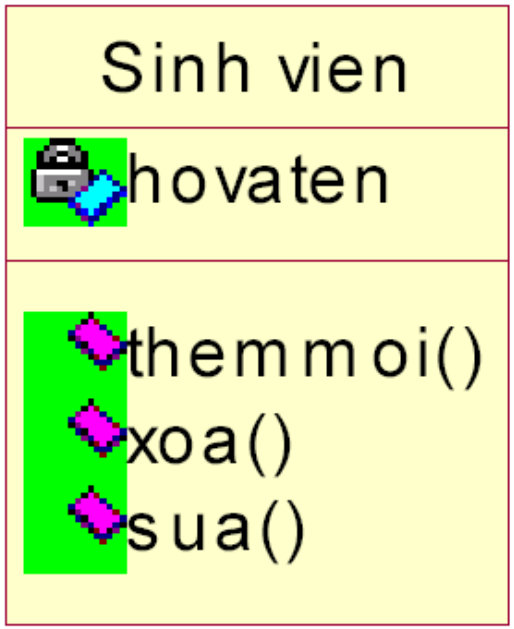
- Các phần tử mang tính cấu trúc
- Các phần tử thể hiện hành vi
- Các phần tử mang tính nhóm

6.1 Các phần tử cấu trúc

- Là bộ phận tĩnh của mô hình
- Biểu diễn các thành phần khái niệm hay vật lý
- Bao gồm:
  - Lớp
  - Giao diện
  - Phần tử cộng tác
  - Trường hợp sử dụng (Use case)
  - Thành phần (component), Nút (node)

Lớp (Class)

- Là mô tả tập các đối tượng cùng chung thuộc tính, thao tác, quan hệ và ngữ nghĩa.



Giao diện (interface)

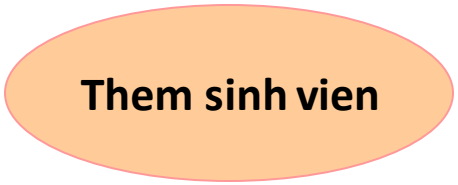
- Tập hợp các thao tác tạo nên dịch vụ của một lớp hay thành phần.
- Mô tả hành vi thấy được từ bên ngoài của thành phần.
- Biểu diễn toàn bộ hay một phần hành vi của lớp.
- Định nghĩa một tập các thao tác ở mức khai báo, không định nghĩa cài đặt của chúng.

Phần tử cộng tác (Collaboration)

- Mô tả ngữ cảnh của tương tác.
- Ký hiệu đồ hoạ: hình elip với đường vẽ nét đứt, kèm theo tên.
- Thể hiện giải pháp thi hành bên trong hệ thống, bao gồm các lớp, quan hệ và tương tác giữa chúng để đạt được một chức năng mong đợi của Use case.

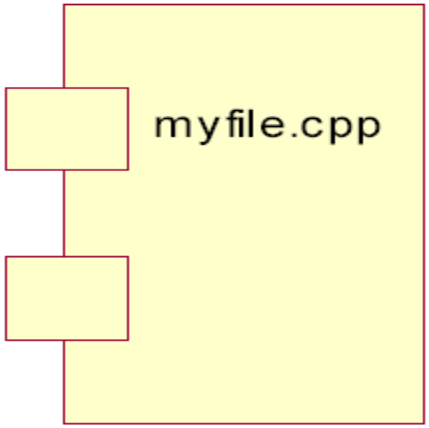
Use case

- Mô tả trình tự các hành động hệ thống sẽ thực hiện để đạt được một kết quả cho tác nhân nào đó.
- Tác nhân: những đối tượng bên ngoài tương tác với hệ thống.
- Tập hợp các Use case của hệ thống sẽ hình thành các trường hợp mà hệ thống được sử dụng.
- Sử dụng Use case để cấu trúc các phần tử có tính hành vi trong mô hình

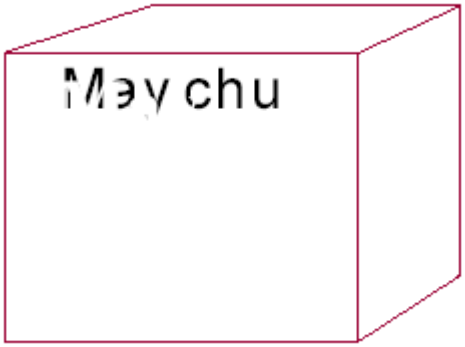


Thành phần và nút

- **Thành phần (Component)**: Biểu diễn vật lý của mã nguồn, các file nhị phân trong quá trình phát triển hệ thống.



- **Nút (node)**: thể hiện thành phần vật lý, tồn tại khi chương trình chạy và biểu diễn các tài nguyên tính toán.
  - Có thể là máy tính, thiết bị phần cứng.



6.2 Các phần tử hành vi

- Là bộ phận động của mô hình, biểu diễn hành vi theo thời gian và không gian dưới hai hình thức
  - Tương tác (Interaction): bao gồm tập các thông điệp trao đổi giữa các đối tượng trong ngữ cảnh cụ thể để thực hiện mục đích cụ thể.

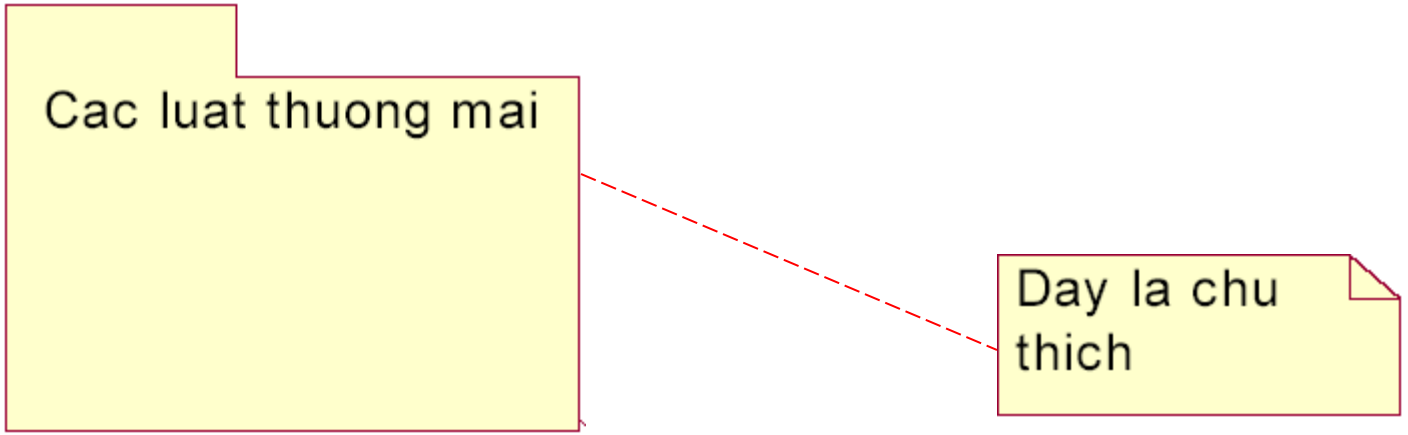
  - Máy trạng thái (States machine): thể hiện các trạng thái của một đối tượng trong thời gian sống của nó nhằm đáp ứng các sự kiện, tác động từ bên ngoài.

6.3 Các phần tử nhóm

- Là bộ phận tổ chức của mô hình.
- Gói (Package): dung để nhóm các phần tử có một ý nghĩa chung nào đó vào một nhóm.
- Các phần tử cấu trúc, hành vi và ngay cả phần tử nhóm có thể cho vào gói.
- Không giống thành phần (component), phần tử nhóm hoàn toàn là khái niệm, có nghĩa rằng chúng chỉ tồn tại vào thời điểm phát triển hệ thống chứ không tồn tại vào thời gian chạy chương trình.

Chú thích

- Là bộ phận chú giải của mô hình UML.
- Phần tử chú thích được gọi là lời ghi chú (note) và dùng để mô tả các phần tử khác trong mô hình.



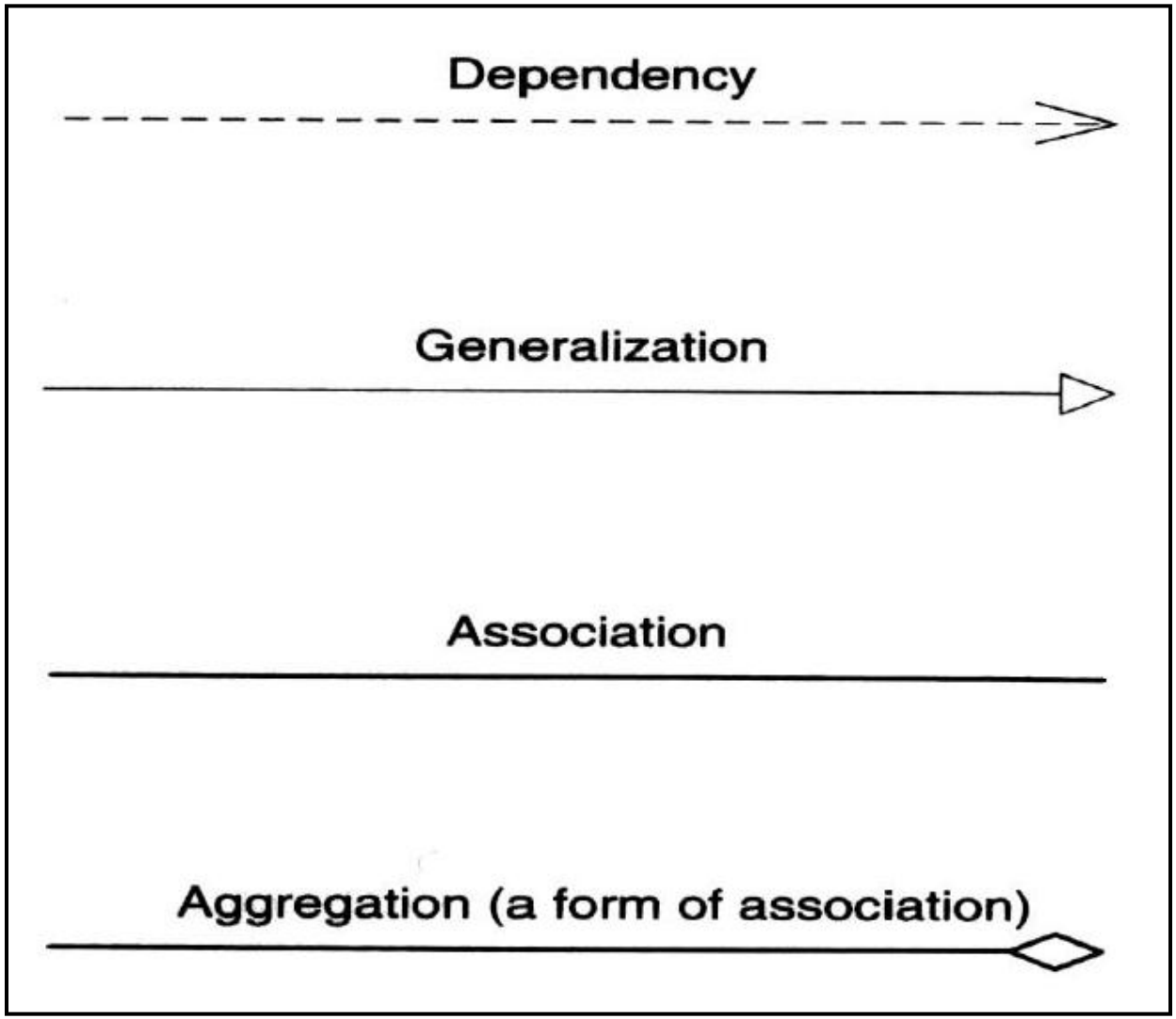
Nội dung

1. Tổng quan về phân tích thiết kế hướng đối tượng và các khái niệm cơ bản
2. Lịch sử UML
3. UML và các khái niệm liên quan
4. Mục tiêu của UML
5. UML và các giai đoạn phát triển phần mềm
6. Các thành phần của UML
7. **Các quan hệ trong UML**
8. Các khung nhìn và lược đồ UML

7. Các quan hệ trong UML

- Có bốn loại quan hệ trong UML
  - **Kết hợp** (Association): mối quan hệ liên kết giữa 2 lớp.
  - **Khái quát hóa** (Generalization): còn được gọi là tính thừa kế. Ý nghĩa: một phần tử này có thể là một sự chuyên biệt hóa của một phần tử khác.
  - **Phụ thuộc** (Dependency): chỉ ra rằng một phần tử này phụ thuộc trong một phương thức nào đó vào một phần tử khác.
  - **Tập hợp/bao gộp** (Aggregation): Một dạng của quan hệ kết hợp, trong đó một phần tử này chứa các phần tử khác.

7. Các quan hệ trong UML

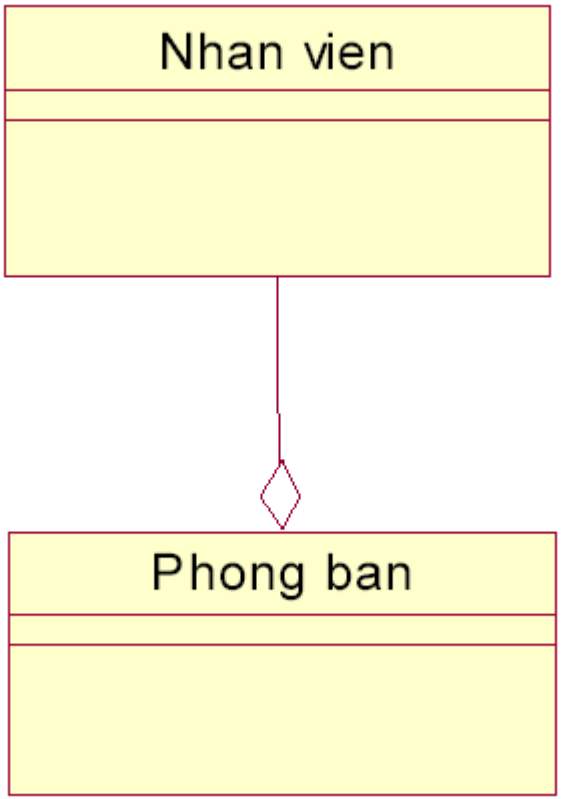


Quan hệ kết hợp (Association)

- Là quan hệ cấu trúc
- Mô tả tập liên kết (kết nối giữa các đối tượng).
- Khi đối tượng của lớp này gửi/nhận thông điệp đến/từ đối tượng của lớp kia thì ta gọi chúng là có quan hệ kết hợp.

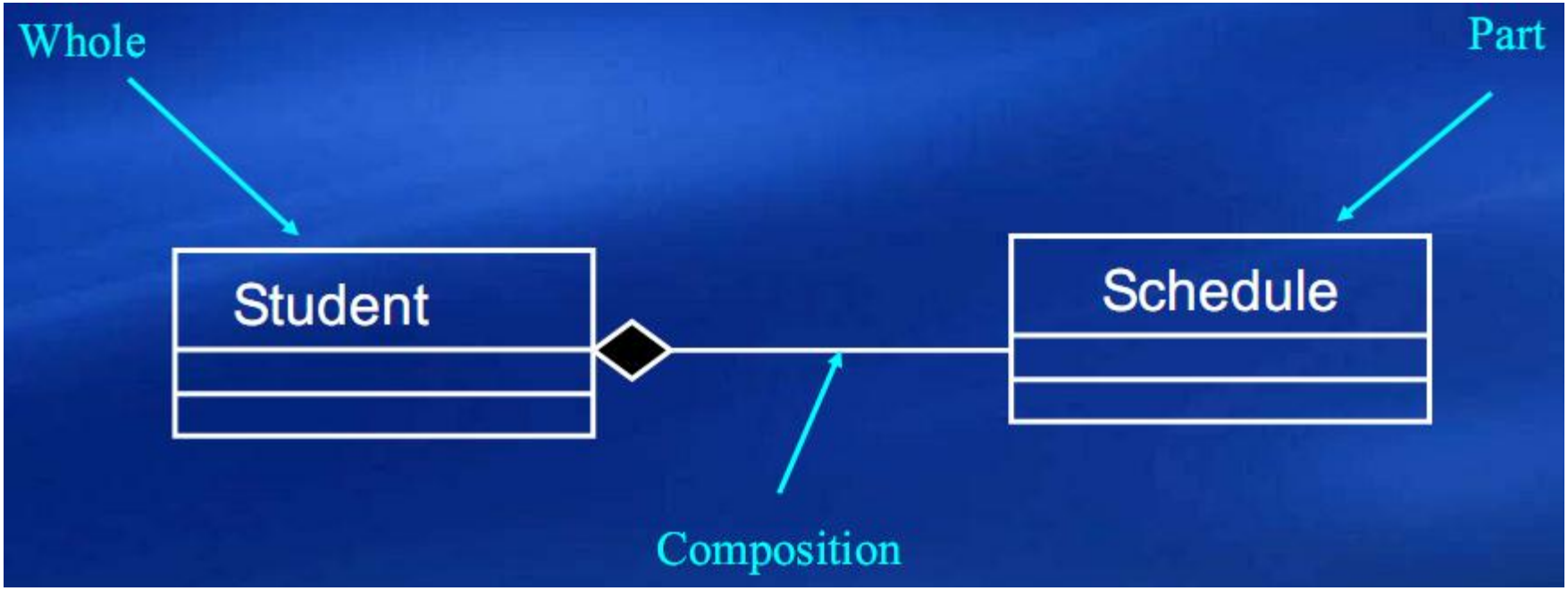
Aggregation

- Tập hợp (Aggregation) là dạng đặc biệt của quan hệ kết hợp, biểu diễn mối quan hệ toàn thể - bộ phận.



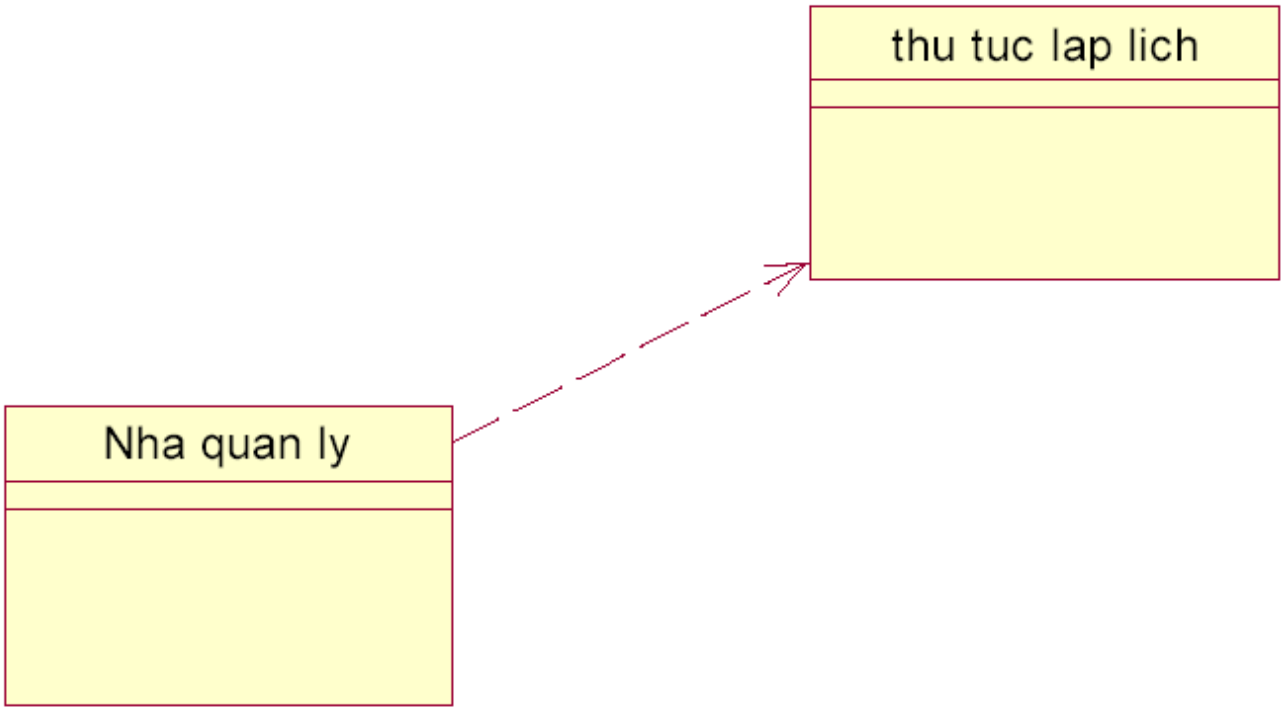
Composition

- Bao gộp (Composition): một dạng đặc biệt của tập hợp, trong đó nếu như đối tượng toàn thể bị huỷ bỏ thì các đối tượng bộ phận của nó cũng bị huỷ bỏ theo.



Quan hệ phụ thuộc (dependency)

- Là quan hệ ngữ nghĩa giữa hai phần tử trong đó thay đổi phần tử độc lập sẽ tác động đến ngữ nghĩa của phần tử phụ thuộc.

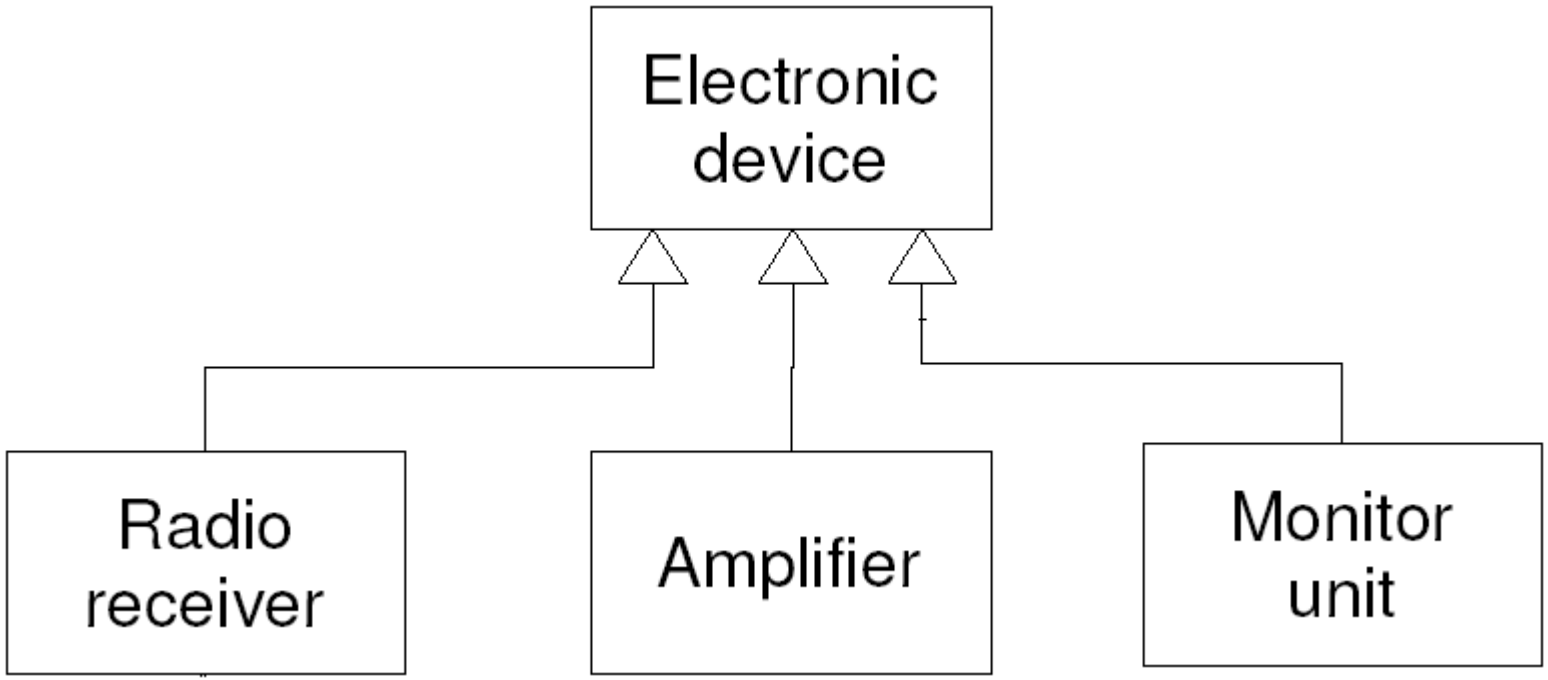


Khái quát hóa và hiện thực hóa

- Khái quát hoá (generalization): là quan hệ đặc biệt hoá / khái quát hoá, đối tượng cụ thể kế thừa các thuộc tính và phương thức của đối tượng tổng quát.

- Hiện thực hoá (realization): là quan hệ ngữ nghĩa giữa giao diện và lớp hay thành phần hiện thực lớp; giữa use case và hiện thực Use case.

Khái quát hóa



Nội dung

1. Tổng quan về phân tích thiết kế hướng đối tượng và các khái niệm cơ bản
2. Lịch sử UML
3. UML và các khái niệm liên quan
4. Mục tiêu của UML
5. UML và các giai đoạn phát triển phần mềm
6. Các thành phần của UML
7. Các quan hệ trong UML
8. **Các khung nhìn và lược đồ UML**

Kiến trúc hệ thống

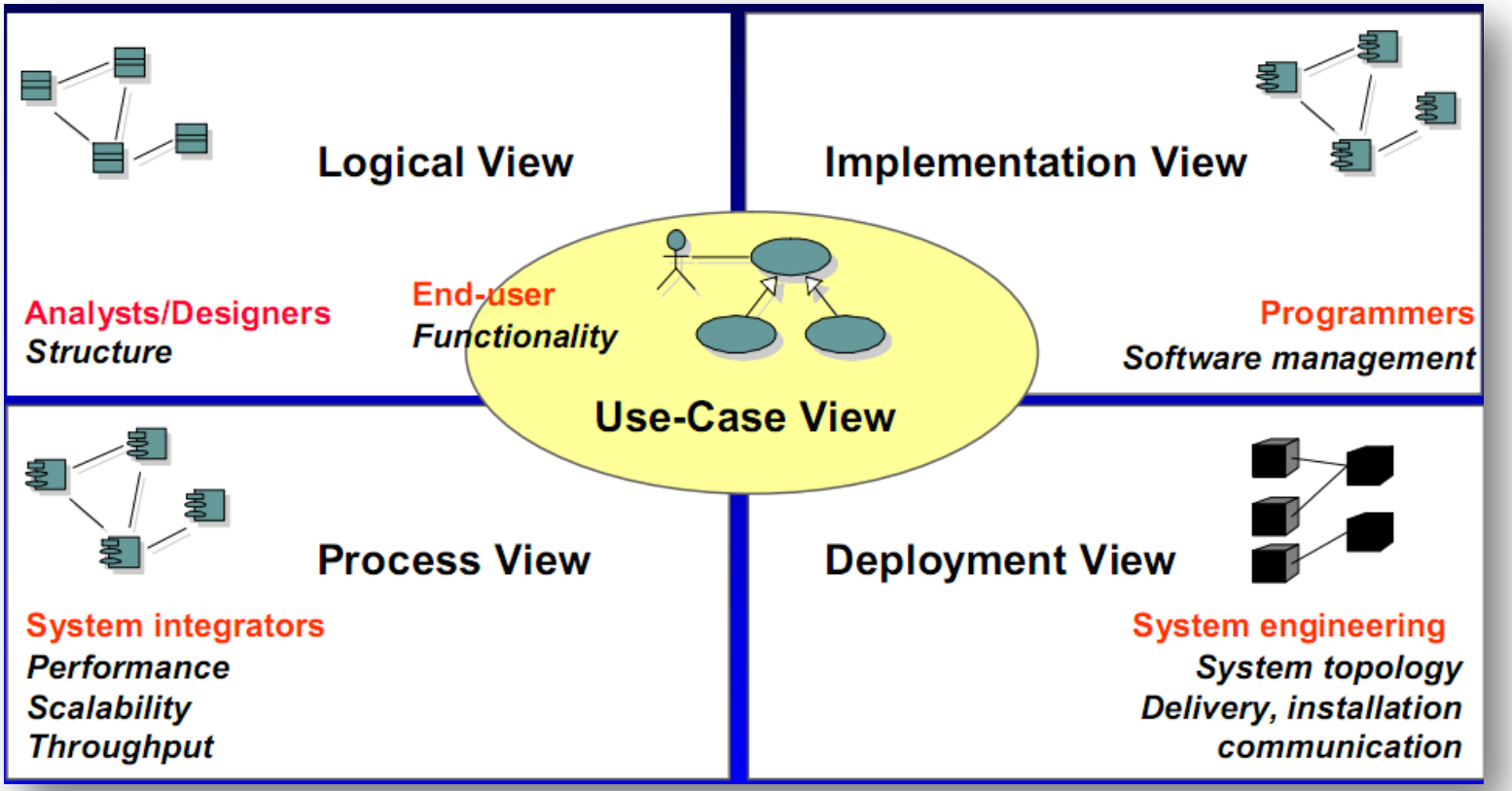
- Không thể mô hình hoá một hệ thống phức tạp chỉ bằng một mô hình hay lược đồ, hệ thống phải được phân tích dưới những góc độ khác nhau.
- Việc hiển thị, đặc tả, xây dựng và làm tài liệu hệ thống đòi hỏi hệ thống phải được xem xét từ nhiều khía cạnh khác nhau
- Kiến trúc hệ thống được sử dụng để quản lí các điểm nhìn khác nhau để điều khiển sự phát triển hệ thống tăng dần và lặp trong suốt chu kì sống.

Cấu trúc view

- Một hệ thống cần phải được miêu tả với một loạt các khía cạnh khác nhau
- Một hệ thống thường được miêu tả trong một loạt các hướng nhìn khác nhau, mỗi hướng/khung nhìn sẽ thể hiện một bức ảnh ánh xạ của toàn bộ hệ thống và chỉ ra một khía cạnh riêng của hệ thống.
- Mỗi View là một thể hiện của hệ thống được mô hình hoá, mỗi View có thể bao gồm nhiều loại biểu đồ khác nhau

4+1 khung nhìn của UML

- Một hệ thống được mô tả trong 1 số khía cạnh khác nhau



4+1 khung nhìn của UML

- ***Use Case View*** (User model View hoặc Scenario View)- thể hiện các vấn đề và các giải pháp liên quan đến chức năng tổng quát của hệ thống.
- ***Logical View*** (Structural model View hoặc Static View)- thể hiện các vấn đề liên quan đến cấu trúc thiết kế của hệ thống.
- ***Implementation View*** (Component View) thể hiện các vấn đề liên quan đến việc tổ chức các thành phần trong hệ thống.
- ***Process View*** (Dynamic, Concurrent, hoặc Collaboration View) thể hiện các vấn đề liên quan đến việc xử lý giao tiếp và đồng bộ trong hệ thống.
- ***Deployment View*** (Environment model View) thể hiện các vấn đề liên quan đến việc triển khai hệ thống.

Use case view

- Tác giả của Use case là Ivar Jacobson
- Use case trở thành một phương pháp được sử dụng cho việc nắm bắt các yêu cầu vào những năm 1990s
- Khung nhìn Use case miêu tả chức năng của hệ thống sẽ phải cung cấp được tác nhân từ bên ngoài mong đợi.
- Use case không chỉ được sử dụng cho việc mô hình hóa các hệ thống kĩ thuật mà còn dùng để mô hình các hệ thống kinh doanh
- Use case thể hiện tương tác nổi bật giữa user và hệ thống.

Use case view (tt)

- Khung nhìn Use case là khung nhìn dành cho khách hàng, nhà thiết kế, nhà phát triển và người thử nghiệm; được miêu tả qua các biểu đồ Use case (***use case diagram***) và thỉnh thoảng cũng bao gồm cả các biểu đồ hoạt động (***activity diagram***).
- Khung nhìn Use case mang tính trung tâm, bởi nó đặt ra nội dung thúc đẩy sự phát triển các khung nhìn khác.
- Khung nhìn này cũng được sử dụng để thẩm tra (verify) hệ thống qua việc thử nghiệm xem khung nhìn Use case có đúng với mong đợi của khách hàng, có đúng với hệ thống vừa được hoàn thành.

Logical View

- Được dùng để mô tả các phần tử bên trong hệ thống và mối quan hệ, sự tương tác giữa chúng để thực hiện các chức năng mong đợi của hệ thống.
  - Chủ yếu nó được sử dụng cho các nhà thiết kế và nhà phát triển.

- Trong đó:
  - Cấu trúc tĩnh được miêu tả bằng các biểu đồ lớp (***class diagram***) và biểu đồ đối tượng (***object diagram***).
  - Quá trình mô hình hóa động được miêu tả trong các biểu đồ trạng thái (***state diagram***), biểu đồ trình tự (***sequence diagram***), biểu đồ tương tác (***collaboration diagram***) và biểu đồ hoạt động (***activity diagram***).

Process View (Concurrency View)

- Khung nhìn xử lý nhắm tới việc chia hệ thống thành các qui trình (process) và các bộ xử lý (processor).
- Sử dụng hiệu quả các tài nguyên (efficient usage of resources)
  - Thực hiện song song
  - Xử lý các sự kiện không đồng bộ từ môi trường.
- Phải quan tâm đến vấn đề giao tiếp và đồng bộ hóa các tiểu trình đó.
- Dùng cho các người phát triển và tích hợp hệ thống
- Bao gồm các diagrams:
  - Biểu đồ động (trạng thái, trình tự, tương tác và hoạt động)
  - Biểu đồ thực thi (biểu đồ thành phần và biểu đồ triển khai).

Implementation View

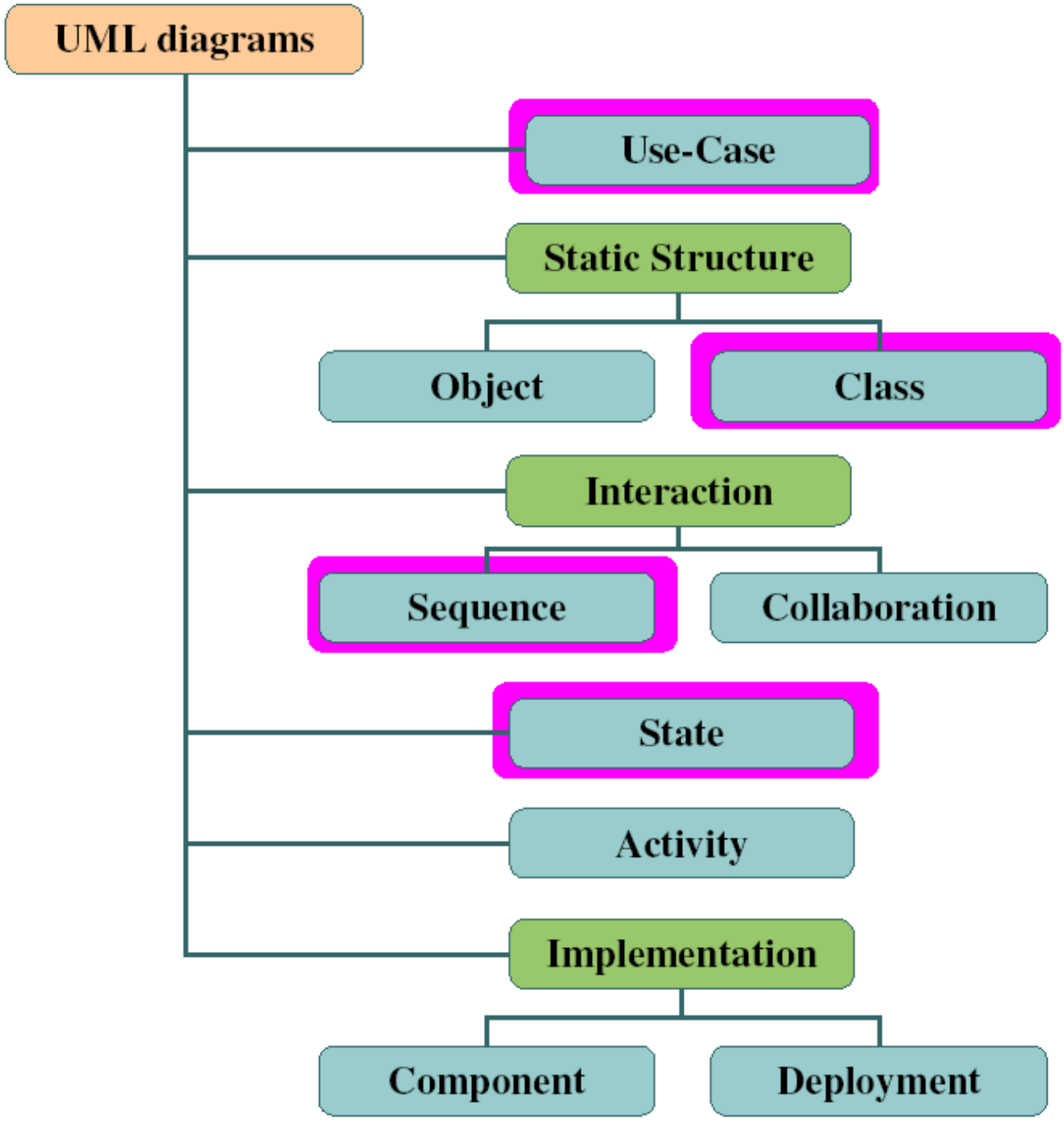
- Đặc tả của các module cài đặt
  - Thường được sử dụng cho nhà phát triển, bao gồm nhiều biểu đồ thành phần.
- Bao gồm các component và file tạo nên hệ thống vật lý.
- Nó chỉ ra sự phụ thuộc giữa các thành phần này, cách kết hợp chúng lại với nhau để tạo ra một hệ thống thực thi.

Deployment View

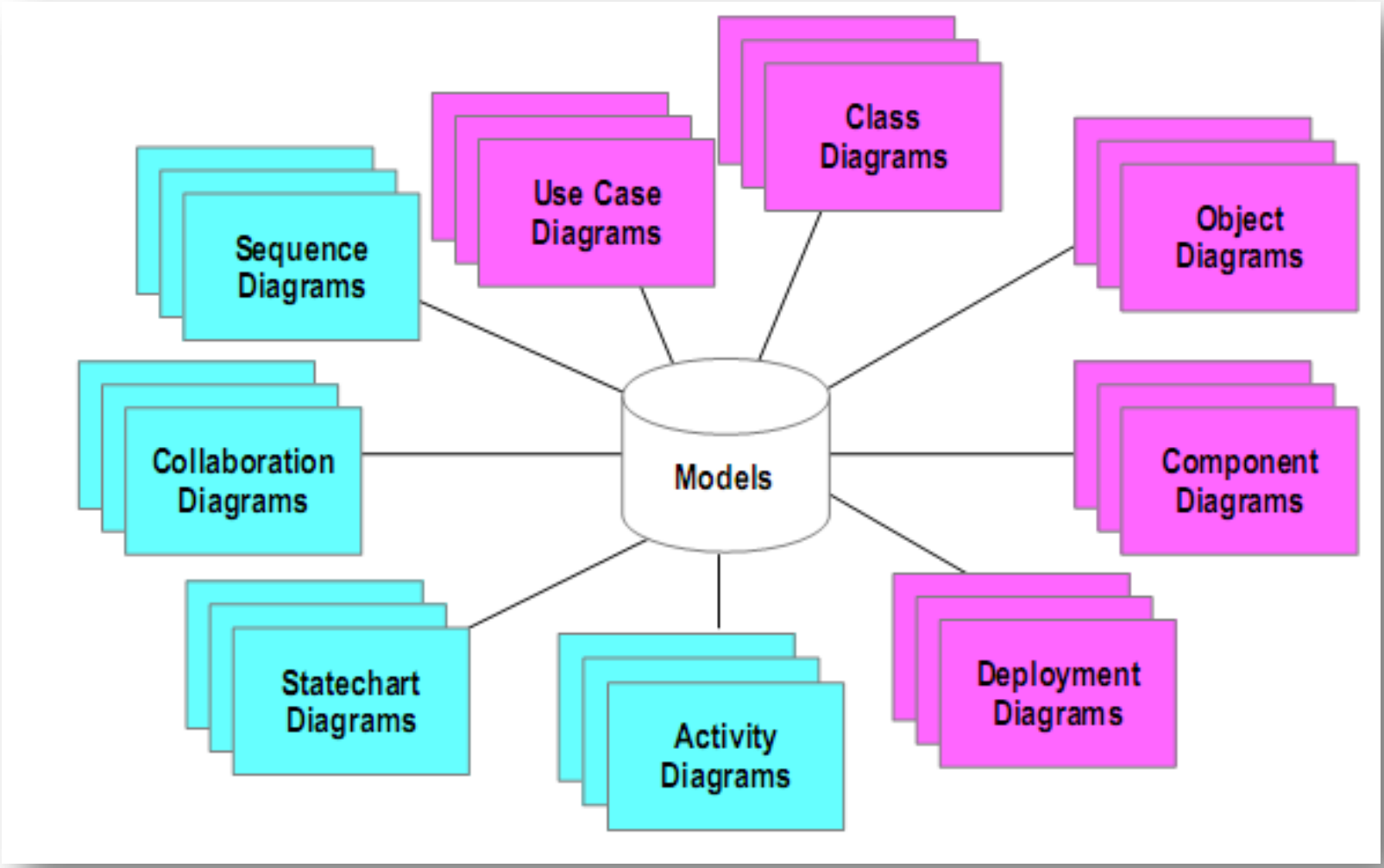
- Hướng nhìn triển khai chỉ cho chúng ta sơ đồ triển khai về mặt vật lý của hệ thống
  - Ví dụ: máy tính, máy móc và sự liên kết giữa chúng.
- Được sử dụng bởi người phát triển, tích hợp và test hệ thống
- Được thể hiện bằng các biểu đồ triển khai - ***Deployment diagram***
- Bao gồm sự ánh xạ các thành phần của hệ thống vào cấu trúc vật lý
  - Ví dụ: chương trình nào hay đối tượng nào sẽ được thực thi trên máy tính nào.

Biểu đồ (diagram)

- Trong UML, có 9 loại biểu đồ chuẩn
  - Các sơ đồ mô tả khía cạnh tĩnh
    - Sơ đồ đối tượng (object diagram)
    - Sơ đồ lớp (class diagram)
    - Sơ đồ use case (use case diagram)
    - Sơ đồ thành phần (component diagram)
    - Sơ đồ triển khai (deployment diragram)
  - Các sơ đồ mô tả khía cạnh động
    - Sơ đồ tuần tự (sequence diagram)
    - Sơ đồ cộng tác (collaboration diagram)
    - Sơ đồ hoạt động (activity diagram)
    - Sơ đồ trạng thái (state transition diagram)


UML diagrams
Use-Case
Static Structure
Object
Class
Interaction
Sequence
Collaboration
State
Activity
Implementation
Component
Deployment

Biểu đồ (tt)


Class Diagrams
Use Case Diagrams
Sequence Diagrams
Object Diagrams
Collaboration Diagrams
Models
Component Diagrams
Statechart Diagrams
Activity Diagrams
Deployment Diagrams

Biểu đồ (tt)

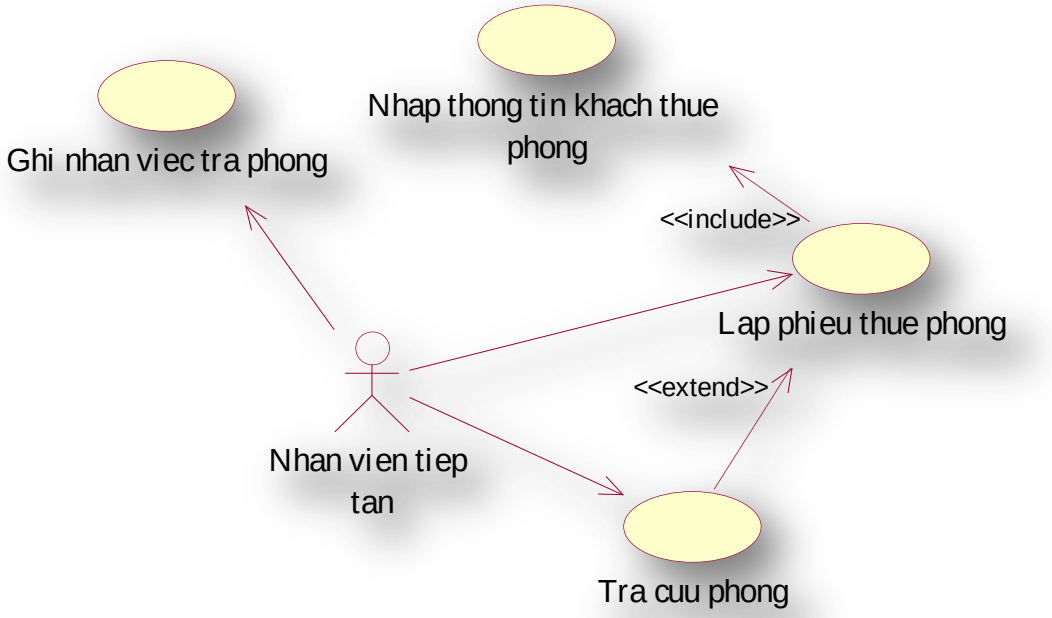
- Use case: nắm bắt các yêu cầu
- Sequence: Thể hiện thời gian tương tác giữa các đối tượng
- Collaboration: Làm nổi bật các quan hệ giữa các đối tượng và liên kết giữa chúng
- Class, Object: thể hiện cấu trúc tĩnh của hệ thống
- State: các trạng thái của đối tượng
- Activity: các hoạt động song song và các tiến trình
- Component: cấu trúc vật lí của các thành phần chính
- Deployment: phân bố vật lí của hệ thống

Use case diagram

- Mô tả chức năng mà hệ thống cung cấp.
- Được xây dựng trong những giai đoạn đầu của quy trình
- Mục tiêu:
  - Đặc tả ngữ cảnh của một hệ thống
  - Nắm bắt các yêu cầu của hệ thống
  - Xác định tính hợp lệ của kiến trúc hệ thống
  - Định hướng quá trình cài đặt và phát sinh các trường hợp test
- Được phát triển bởi nhà phân tích và chuyên gia trong lĩnh vực ứng dụng

Use case diagram (tt)

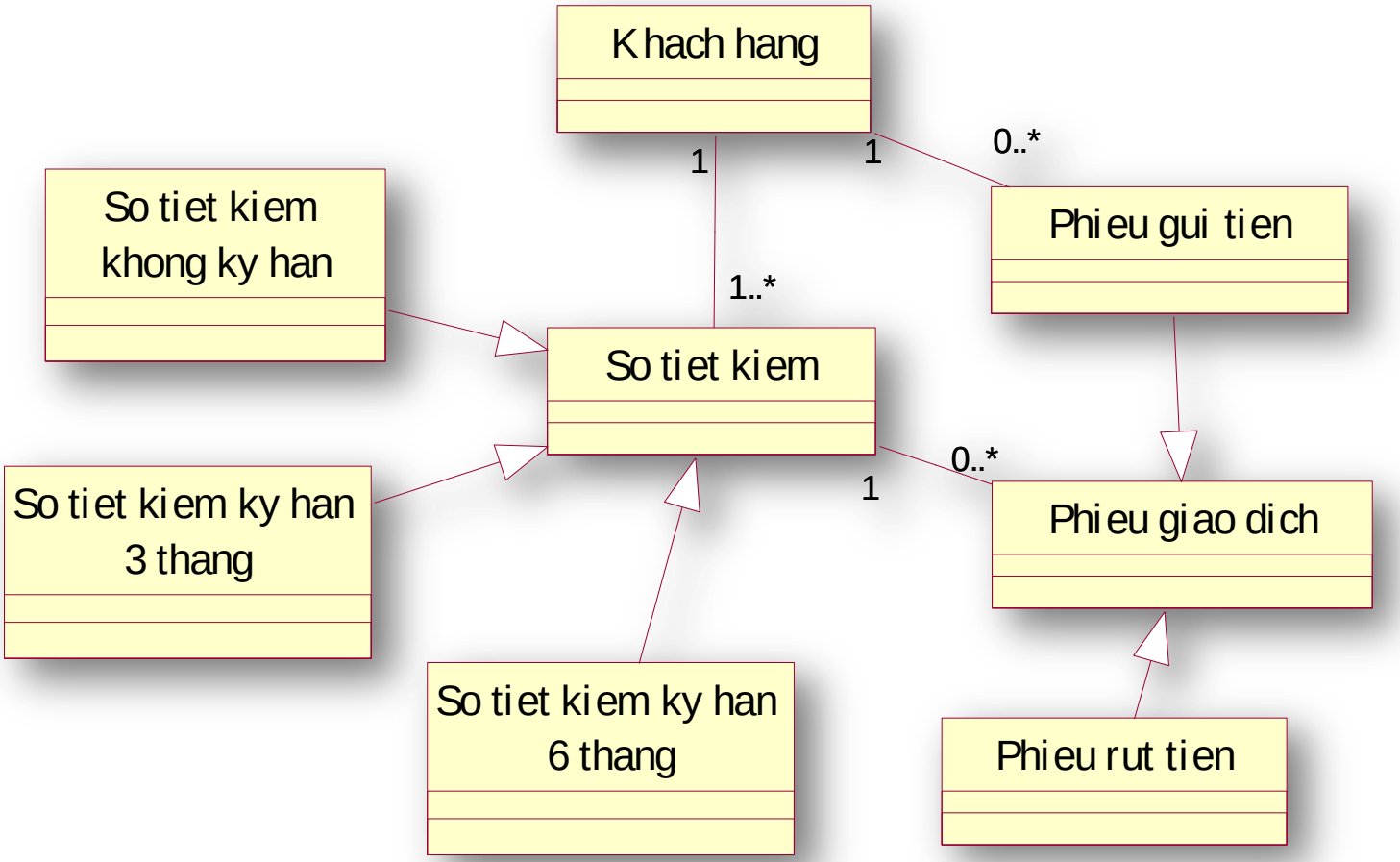
- Use case được miêu tả duy nhất theo hướng nhìn từ ngoài vào của các tác nhân (hành vi của hệ thống theo như sự mong đợi của người sử dụng), không miêu tả chức năng được cung cấp sẽ hoạt động nội bộ bên trong hệ thống ra sao.



Biểu đồ lớp (Class Diagram)

- Chuyển từ khung nhìn use case sang khung nhìn logic của hệ thống
- Bao gồm một tập hợp các lớp, các giao diện, các collaboration và mối quan hệ giữa chúng. Nó thể hiện mặt tĩnh của hệ thống.
- Được phát triển bởi phân tích viên, thiết kế viên và lập trình viên

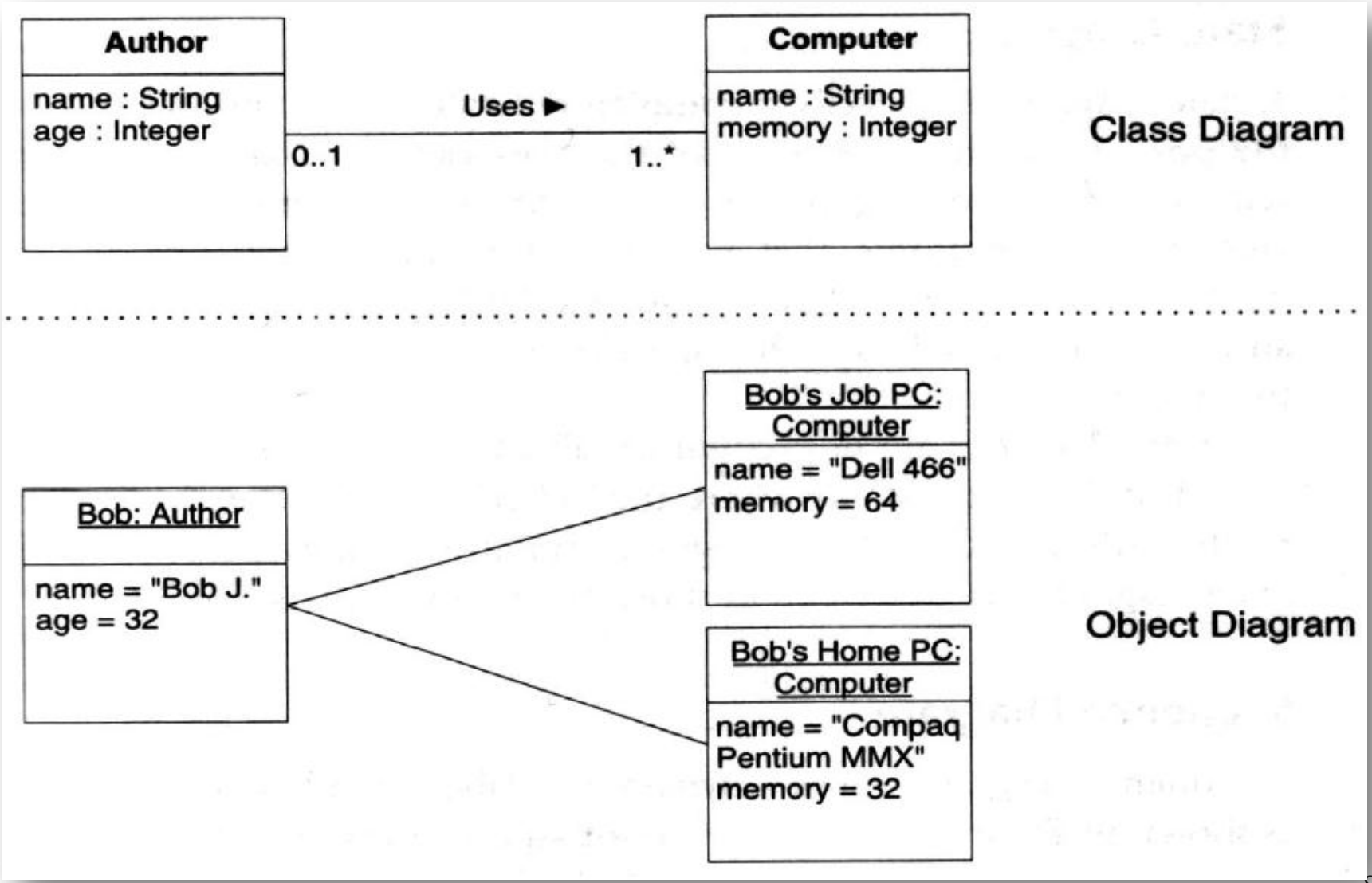
Biểu đồ lớp (tt)



Biểu đồ đối tượng (Object Diagram)

- Bao gồm một tập hợp các đối tượng và mối quan hệ giữa chúng.
- Đối tượng là một thể hiện của lớp, biểu đồ đối tượng là một thể hiện của biểu đồ lớp.
- Được phát triển bởi phân tích viên, thiết kế viên và lập trình viên

Biểu đồ đối tượng (tt)

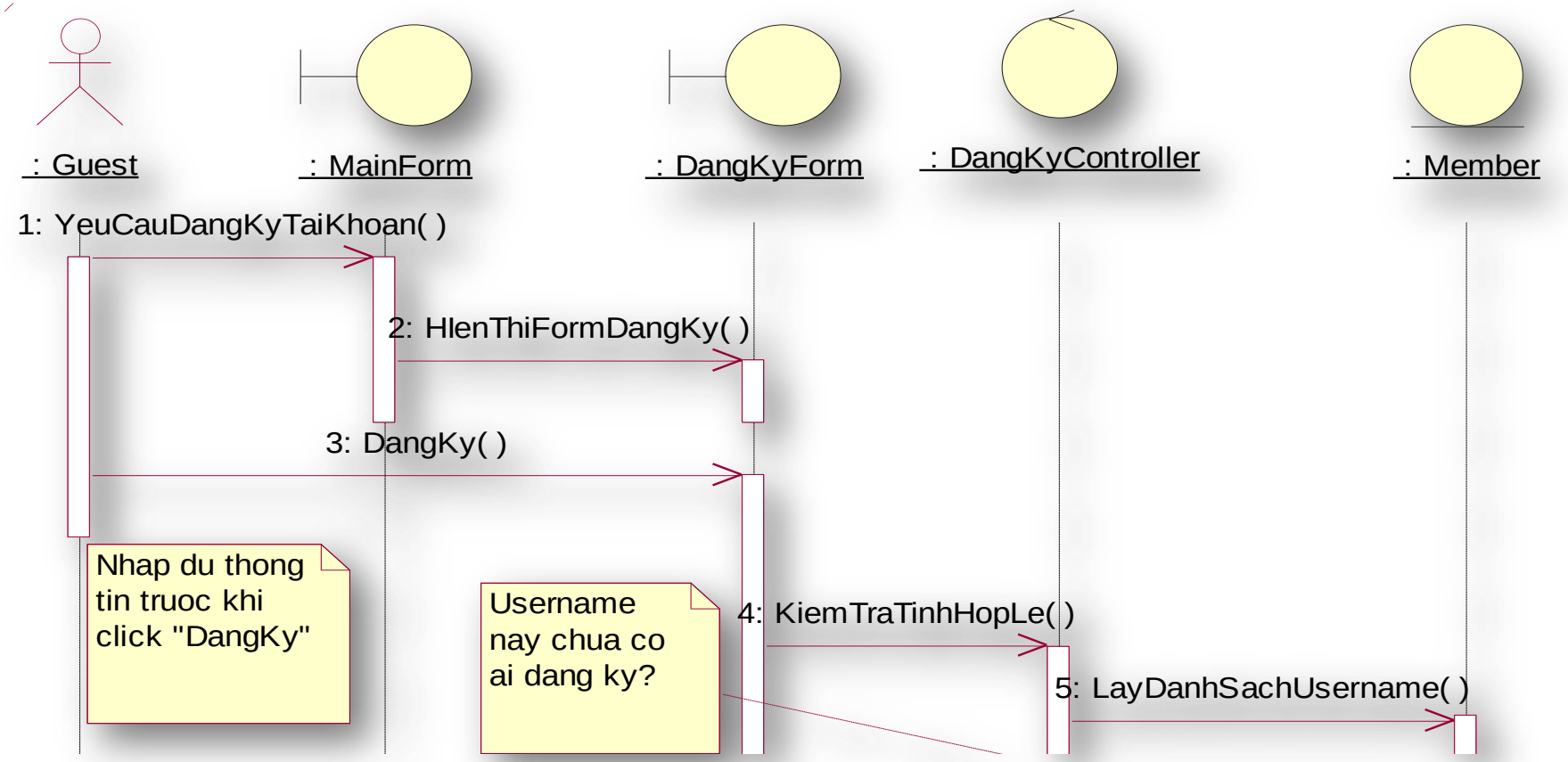


Biểu đồ trình tự (Sequence Diagram)

- Một biểu đồ trình tự chỉ ra sự tương tác động giữa các đối tượng

- Khía cạnh quan trọng của biểu đồ này là chỉ ra trình tự các thông điệp (message) được gửi giữa các đối tượng.

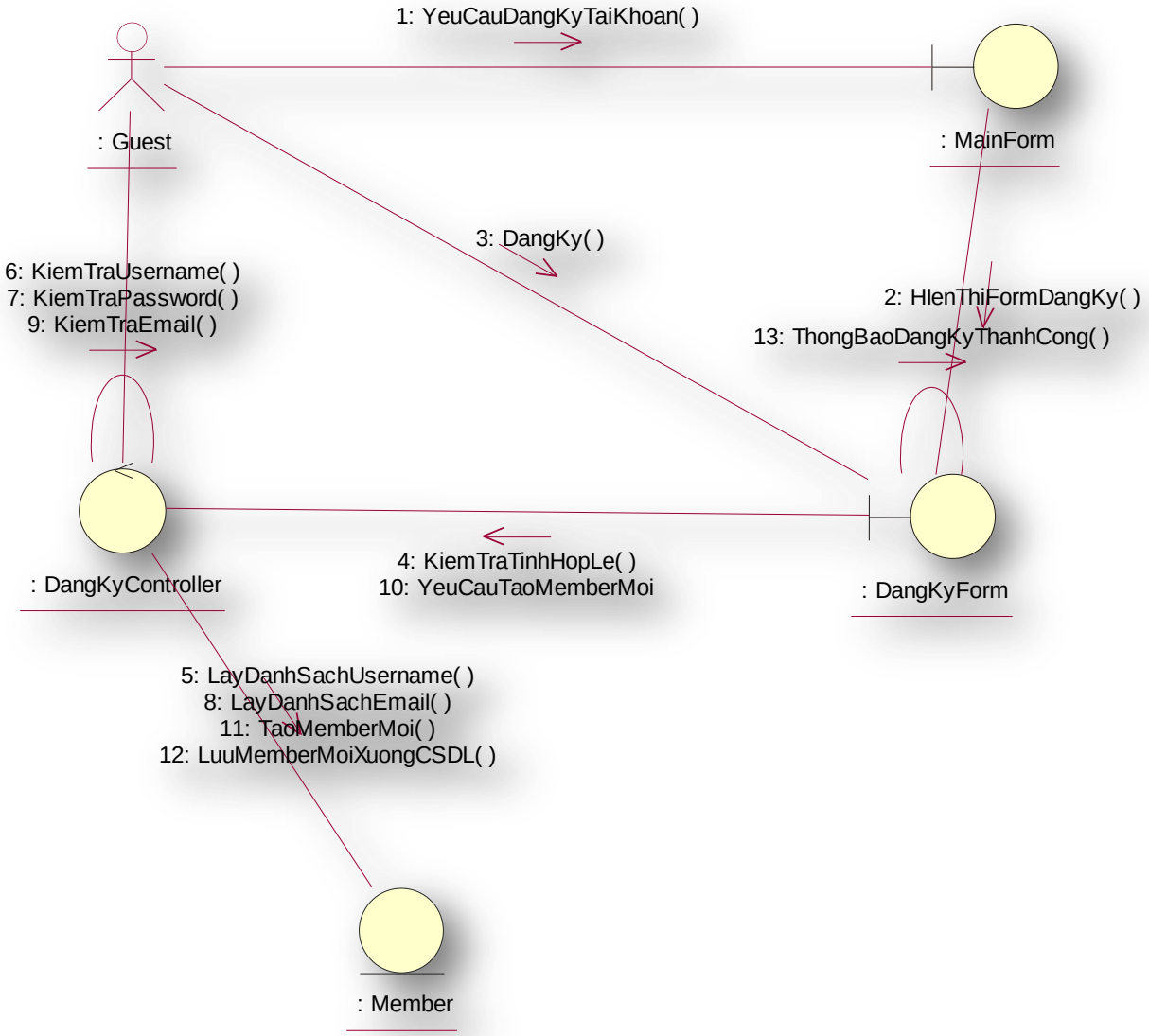
Biểu đồ trình tự (tt)



Biểu đồ cộng tác (Collaboration Diagram)

- Chỉ ra sự tương tác động, giống như biểu đồ trình tự.
- Lựa chọn:
  - Nếu thời gian hay trình tự là yếu tố quan trọng nhất cần nhấn mạnh → chọn biểu đồ trình tự
  - Nếu ngữ cảnh là yếu tố quan trọng hơn → chọn biểu đồ cộng tác.
  - Trình tự tương tác giữa các đối tượng được thể hiện trong cả hai loại biểu đồ này.

Biểu đồ cộng tác (tt)

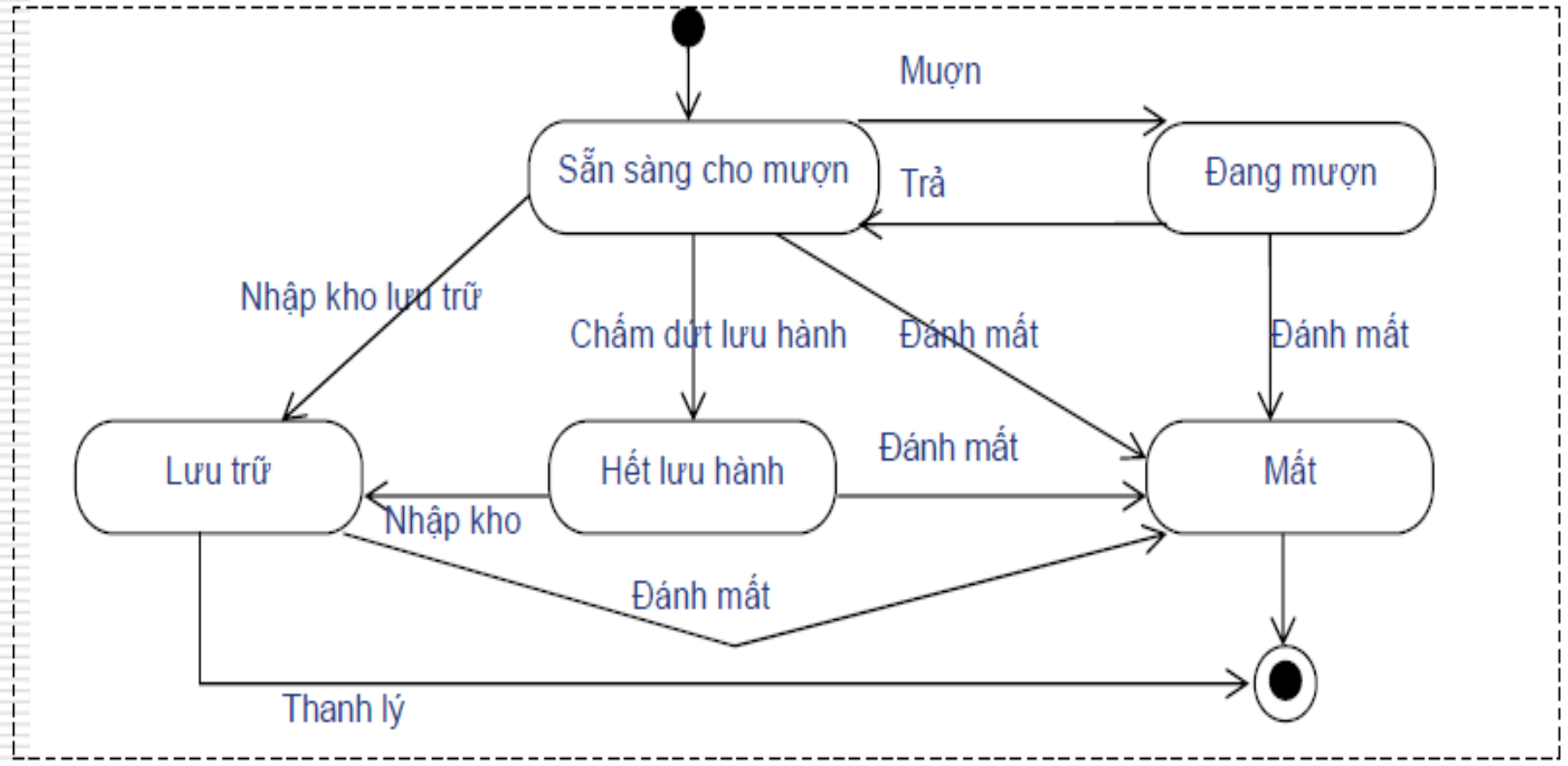


Biểu đồ trạng thái (State Diagram)

- Chỉ ra tất cả các trạng thái mà đối tượng của lớp có thể có, và những sự kiện (event) nào sẽ gây ra sự thay đổi trạng thái

- Chỉ sử dụng cho những lớp có số lượng các trạng thái được định nghĩa rõ ràng, hành vi bị ảnh hưởng và thay đổi qua các trạng thái khác nhau.

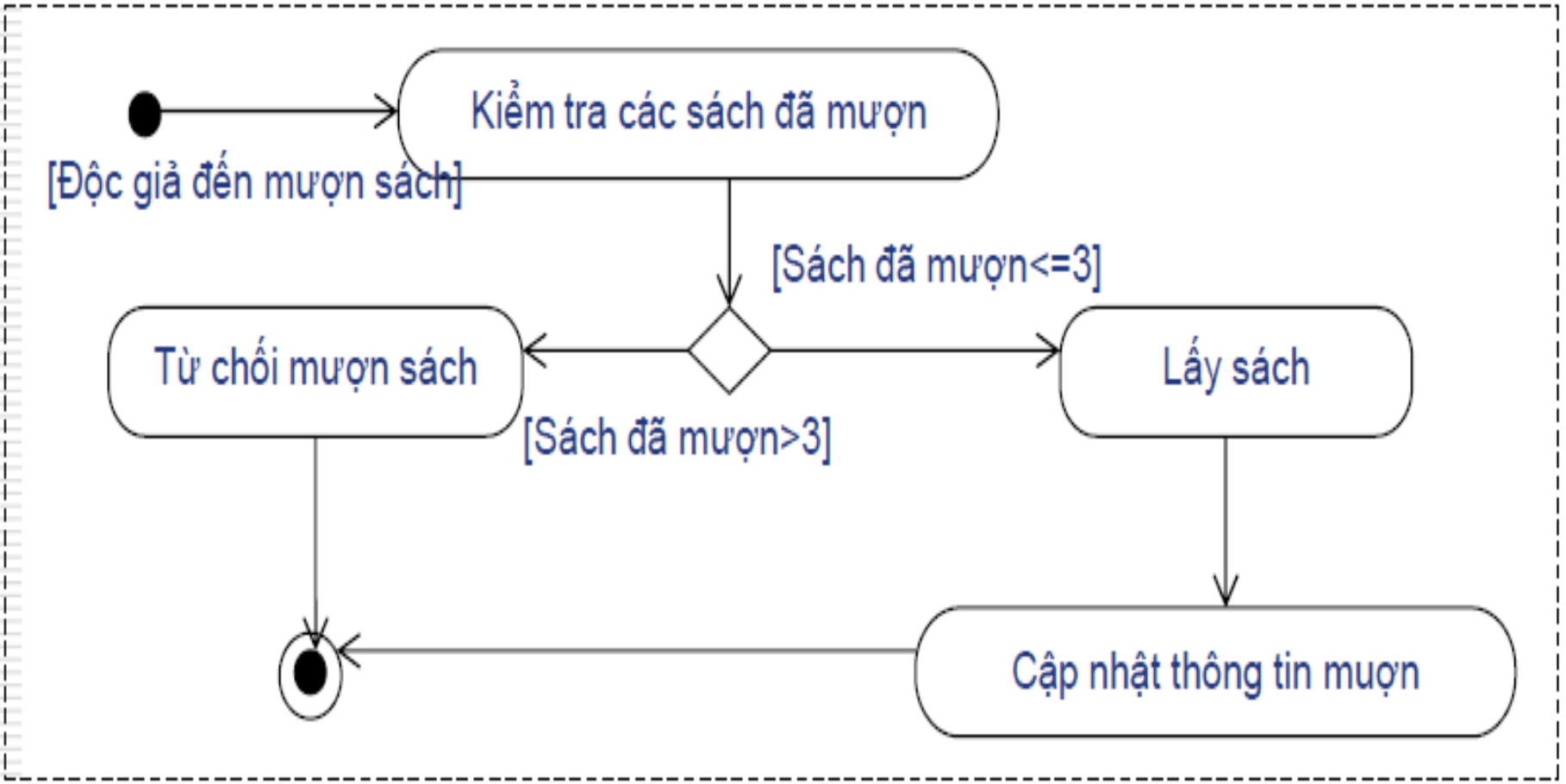
Biểu đồ trạng thái (tt)



Biểu đồ hoạt động (Activity Diagram)

- Chỉ ra trình tự của các hoạt động (activity)
- Miêu tả các hoạt động được thực hiện trong một thủ tục
- Ngoài ra, dùng để miêu tả các dòng chảy hoạt động khác, ví dụ: luồng sự kiện trong một Use case hay trong một trình tự tương tác.
- Một trạng thái hành động sẽ qua đi khi hành động được thực hiện xong (khác với biểu đồ trạng thái: một trạng thái chỉ chuyển sang trạng thái khác sau khi đã xảy ra một sự kiện rõ ràng!)

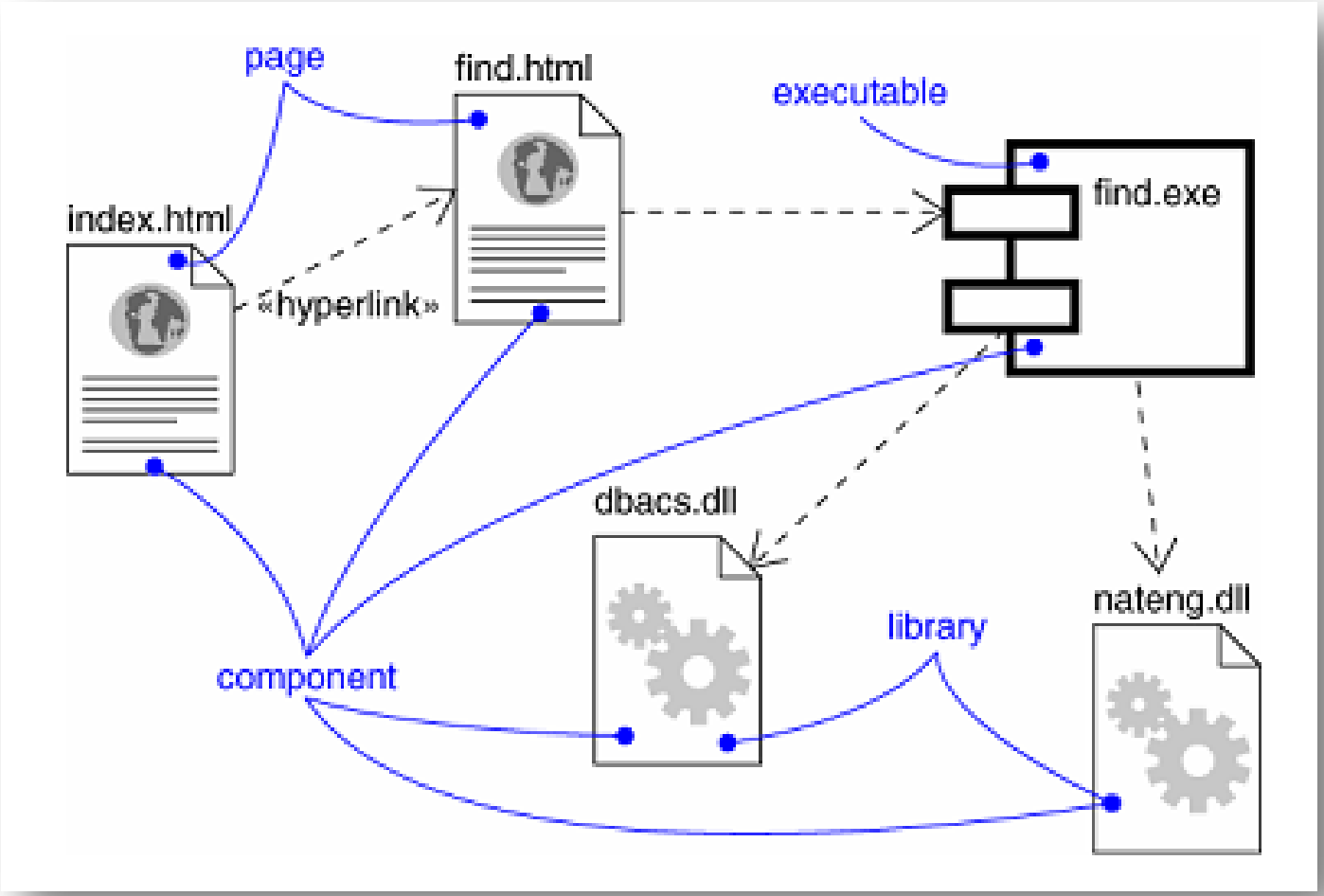
Biểu đồ hoạt động (tt)



Biểu đồ thành phần (Component Diagram)

- Miêu tả cấu trúc vật lí của phần mềm
- Chỉ ra cách tổ chức và sự phụ thuộc của các thành phần(component).
- Liên quan tới biểu đồ lớp, trong đó một thành phần thường ánh xạ tới một hay nhiều lớp, giao diện, collaboration.

Biểu đồ thành phần (Component Diagram)



Biểu đồ triển khai (Deployment Diagram)

- Chỉ ra kiến trúc vật lý của phần cứng, phần mềm trong hệ thống
  - Nodes là các object vật lí, ví dụ như máy tính, máy in , vv
  - Connections là các đường dẫn giữa các node
- Chỉ ra hướng nhìn triển khai, miêu tả kiến trúc vật lý thật sự của hệ thống

Biểu đồ triển khai (Deployment Diagram)