# PHÂN TÍCH THIẾT KẾ HƯỚNG ĐỐI TƯỢNG VỚI UML

**Giảng viên: ThS. Nguyễn Đình Loan Phương**

**Email: phuongndl@uit.edu.vn**

# Giới thiệu môn học

- Lý thuyết : 45 tiết
- Thực hành, đồ án: 30 tiết
- Thang điểm:
  - Lý thuyết: 4/10
  - Đồ án : 4/10
  - Giữa kỳ : 2/10

# Tài liệu tham khảo

- Giáo trình “Phân tích & thiết kế hướng đối tượng bằng UML” và “Qui trình phát triển phần mềm RUP” – ĐHKHTN, Dương Anh Đức
- Giáo trình “Phân tích & thiết kế hướng đối tượng bằng UML” – ĐHKHTN, Phạm Nguyễn Cương
- UML Fundamental, Dr. Ernest Cachia, 2001- 2004
- ….

- Các trang WEB
  - www.omg.org
  - www.rational.com
  - Các trang WEB về CASE Tools, OOAD & UML

# Nội dung môn học

- Tổng quan về UML
- Xác định yêu cầu
- Tổng quan về phân tích thiết kế
- Mô hình hóa nghiệp vụ bằng UML

# Giới thiệu

- Mục đích
  - Giới thiệu một số nét chính về lịch sử của UML, phạm vi và mục đích của UML và nội dung chính của môn học
- Nội dung chính
  - Động cơ đối với OOA/D
  - UML là gì, những gì không thuộc phạm vi của UML
  - Lịch sử của UML
  - Mục đích của UML
  - Các khung nhìn và lược đồ UML
  - Nội dung của môn học

# Phân tích thiết kế hướng đối tượng

- “Tất cả các lược đồ chỉ là những bức tranh đẹp”
- “Người sử dụng sẽ không cảm ơn những bứctranh đẹp, những gì người sử dụng muốn là một phần mềm chạy tốt”
- Chúng ta không thể hiểu được các hệ thống phức tạp trong trạng thái nguyên vẹn của nó (phải chia nhỏ, mổ xẻ mô hình )
- Những biểu tượng được chọn lựa kĩ càng có thể:
  - Làm cho thông tin dễ tiếp cận ,dễ hiểu
  - Đưa ra cái nhìn thấu đáo vào hệ thống

# 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 luận cấu trúc
  - Phương pháp luận hướng đối tượng

# Phương pháp luận cấu trúc

- Phương pháp này còn gọi là phương pháp cổđiển
- Được nhìn nhận dưới sự phức tạp của chức năng hệ thống máy tính
- Chức năng được phân rã theo một hệ thống cấutrúc nhất định do người phân tích hệ thống đưa ra(cấu trúc phân nhánh, lặp...)
- Bao gồm mô hình quá trình chức năng cũng nhưcác mô hình dữ liệu. Sự liên kết giữa hai mô hìnhdữ liệu này còn đơn giản qua các mối liên kết và luồng thông tin từ quá trình chức năng này sang chức năng khác

# Ưu/khuyết điểm của phương pháp

- 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.
- 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, phí phạm mã, khó mở rộng, khó thích nghi của phầm mềm xây dựng dựa vào phương pháp này.

# Phương pháp luận hướng đối tượng

- Phương pháp này xác định rằng, cấu trúc thông tin trong hệ thống thông tin là ít thay đổi.
- Thế giới xung quanh dưới dạng đối tượng rời rạc. Phương pháp đưa ra khái niệm đối tượng để mô tả thôngtin.
- Giới thiệu thêm mối quan hệ kế thừa cha con. Các chức năng được xây dựng trên hệ cấu trúc đối tượng nhờ sự kếthợp thông tin và chức năng trên cấu trúc đối tượng.

# Ưu/khuyết điểm của phương pháp

- Tăng cường tính sử dụng: qua mối liên kết kế thừa, không chỉ những hành vi, đoạn mã được tái sử dụng mà cả những thông tin tĩnh của lớp cha cũng được lớp con tái sử dụng.
- Tăng cường tính mở rộng: việc mở rộng chức năng có thể được thực hiện qua việc tạo lớp con. Vì vậy không ảnh hưởng đến cấu trúc thông tin đã có. Hơn thế nữa phần mềm trở nên linh động hơn hẳn.
- 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 củathông tin.

# Các khái niệm cơ bản - Trừu tượng hoá

- Xem xét các vấn đề cụ thể rồi thông qua quá trình tư duy trừu tượng để rút ra các khái niệm, nguyên tắc quyluật.
- Là công cụ chủ yếu để con người nhận thức và mô hình hoá thế giới.
- Trừu tượng hoá bao gồm hai quá trình chính : khái quát hoá và cụ thể hoá.

# Khái quát hoá (Generalization)

- Khái quát hoá là quá trình tập trung vào những điểm chung cơ bản của các đối tượng, sự kiện công việc cụ thể, loại bỏ những điểm riêng có tính vụn vặt không quan trọng để tạo một khái niệm mới mang đặc tính chung liên quan đến tất cả những cái cụ thể ấy.
- Nói cách khác, khái quát hoá là phép đồng hoá những điểm chung của các sự việc cụ thể mà con người quan sát được.
- Khái quát hoá có thể được phân thành nhiều cấp độ khác nhau tuỳ vào mức độ khi thực hiện phép khái quát cũng như những điểm chung cơ bản được rút ra từ các đối tượng cụ thể.

# Cụ thể hoá (refinement)

- Ngược với khái quát hoá là tinh chế hoá hay cụ thể hoá.
- Quá trình tinh chế là quá trình đi từ những khái niệm sự việc trừu tượng khái quát để mô tả chi tiết, cụ thể các đối tượng sự việc cụ thể hay các khái niệm sự việc trừu tượng ở mức thấp hơn. Nói cách khác, tinh chế là những cái khái quát trừu tượng cho các trường hợp cụ thể.
- Do tinh chế là quá trình ngược của khái quát hoá, nó bao gồm nhiều mức độ khác nhau tuỳ theo mức độ mô tả cụ thể vấn đề khái quát cũng như hướng mô tả.

# Độ phức tạp

- Khi con người đối đầu với mọi bài toán họ luôn luôn phải đối đầu với độ phức tạp.
- Phương hướng chủ yếu là chia nhỏ đến mức có khả năng giải quyết được (cụ thể hoá) “chia để trị”.
- Ngược lại, vấn đề khó khăn của phân rã độ phức tạp lại là khả năng tích hợp và quản lý các vấn đề nhỏ để giải quyết vấn đề lớn (khái quát hoá). Những bước phân rã ban đầu (tầm vĩ mô, khái quát nhất) thường tạo ra cái nhìn tổng quát nhất cho toàn bộ bài toán và được xem như là sườn cho toàn bộ bài toán.
- Những bước phân rã đó được gọi là phân rã kiến trúc. Mỗi phần của phần mềm được phân rã và tích hợp khác nhau tuỳ theo phương pháp luận khác nhau.

# Che dấu thông tin

- Từ hai phương pháp cơ bản là trừu tượng hóa và phân rã độ phức tạp dẫn đến câu hỏi : “Làm thế nào để xây dựng được phần mềm với các mức độ phức tạp và trừu tượng khác nhau?”
- Nguyên tắc che dấu thông tin - thông tin của hai phần được che dấu nếu thật sự không cần thiết - được đưa ra nhằm đảm bảo các phần khác nhau của bài toán có thể tồn tại độc lập và bỏ qua những chi tiết không cần thiết trong mối quan hệ giữa chúng với nhau.
- Che dấu thông tin vì vậy đảm bảo được sự độc lập của từng phần của bài toán, chia bài toán thành từng phần trừu tượng khác nhau và giải quyết bài toán trên từng mức trừu tượng đó.

# Chia sẻ và tái sử dụng

- Tính độc lập của từng bài toán có thể giúp cho bài toán được sử dụng lại trong nhiều hệ thống khác nhau mà không cần thiết phải giải lại.
- Kế thừa (inheritance)
  - Kế thừa cho phép chúng ta định nghĩa một lớp mới tương tự những lớp trước (đã có), ngoài ra bổ sung thêm những thuộc tính và các phương thức mô tả chi tiết hơn về một nhóm các đối tượng cụ thể.
  - Những lớp kế thừa gọi là lớp con (subclass) hoặc là lớp dẫn xuất(derived).
  - Những lớp được kế thừa còn gọi là lớp cha (supperclass).

# Yêu cầu của mô hình hoá

- 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 (soundness) trong kiến trúc

# Lịch sử phát triển

- Vào những năm 1980s-các bước đầu tiên của lập trình hướng đối tượng
  - 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 niện nhưng lại dùng kí hiệu (notation) khác nhau
  - Mỗi phương thức đều có các mặt mạnh và yếu
- Yêu cầu chuẩn hóa
  - Nhóm OMG (Object Management Group) đã thử và thất bại

# Phương thức được hợp nhất

- Sự kiện lớn vào năm 1994 - James Rumbaugh liên kết với Grady Booch thành lập RationalSoftware Corporation
- Vào thời gian đầu, là sự hợp nhất của 2 phương pháp
  - Booch và OMT (Object Modelling Technique)
- Phương pháp này được gọi là Unified Method
- 1995 - Rational thông báo rằng đã mua Ivar, Jacobson's method Objectory. Jacobson muốn liên kết với Rational Software Corporation

# 3 nhân vật quan trọng

- Grady Booch
  - Làm việc cho ADA, sau đó là US Air Force Academy
  - Giám đốc khoa học của RSC
- Jame Rumbaugh
  - Nhân vật đứng đầu của General Electric
  - Tác giả của cuốn Object Modelling Technique
- Ivar Jacobson
  - Làm cho Ericson, sau đó sở hữu công ty Objective System ở Thụy Điển
  - Là cha đẻ của Use Case

# Hợp nhất

- Ba nhân vật nói trên chuyển tên của “Unified method” thành “Unified Modelling Language”
- Mục đích của UML
  - Mô hình các hệ thống (không chỉ là phần mềm) bằng cách sử dụng các khái niệm hướng đối tượng
  - Thiết lập các hiện thực với khái niệm
  - Hướng tới các kế thừa phức tạp trong các hệ thống
  - Tạo ra một ngôn ngữ mô hình khả dụng cho cả người và máy

# Công bố UML

- 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 | I-Logix | MCI |
| Oracle | IntelliCorp | Systemhouse |
| IBM | Softteam | ObjectTime |
| DECHP | Sterling | PlatiumTechnology |
| TI | Software | Ptech |
| Unisys | ICON Computing | ReichTechnologies |
| | | ... |

# Các hoạt động tiến tới chuẩn hóa

- Mục đích căn nguyên của UML là để trở thành một chuẩn hóa trên thực tế
- "Chấp thuận" bằng cách được sử dụng rộng rãi
- Nhưng OMG muốn có một chuẩn hóa thực sự
  - Yêu cầu một chuẩn hoá chính thức hơn là chỉ trên thực tế
  - Các định nghĩa rõ ràng của cú pháp và ngữ nghĩa
- OMG Task force được thiết lập cho vấn đề chuẩn hóa

# UML và các khái niệm

- 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.
- Có 4+1 khung nhìn: Logical, Component, Process, Deployment và Use case

# UML là gì?

- UML là một cách phân tích và thiết kế mô hình theo hướng đối tượng
  - Hiểu theo cách thông thường, UML bao gồm các mô hình đặc trưng cho việc phân tích và thiết kế
- 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

# UML-NN mô hình hướng đối tượng

- UML được tạo ra phục vụ cho việc mô hình hóa hướng đối tượng
- Hướng đối tượng sản sinh ra các mô hình thể hiện một lĩnh vực
  - Một lĩnh vực kinh doanh, ví dụ: banking
  - Các thuật ngữ và đối tượng của lĩnh vực (ví dụ: tiền, séc)
- UML có thể được sử dụng để mô hình nhiều kiểu hệ thống khác nhau.

# UML-Ngôn ngữ mô hình hoá 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

# 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

# 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

# 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)…

# UML và sự phát triển phần mềm

- Giai đoạn nghiên cứu sơ bộ
- Giai đoạn phân tích
- Giai đoạn thiết kế
- Giai đoạn xây dựngThử 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).
  - Mỗi Use case được mô tả trong tài liệu sẽ đặc tả các yêu cầu của khách hàng: khách hàng chờ đợi điều gì từ phía hệ thống mà không hề để ý đến việc chức năng này sẽ được thực thi ra sao.

# 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 – lập trình

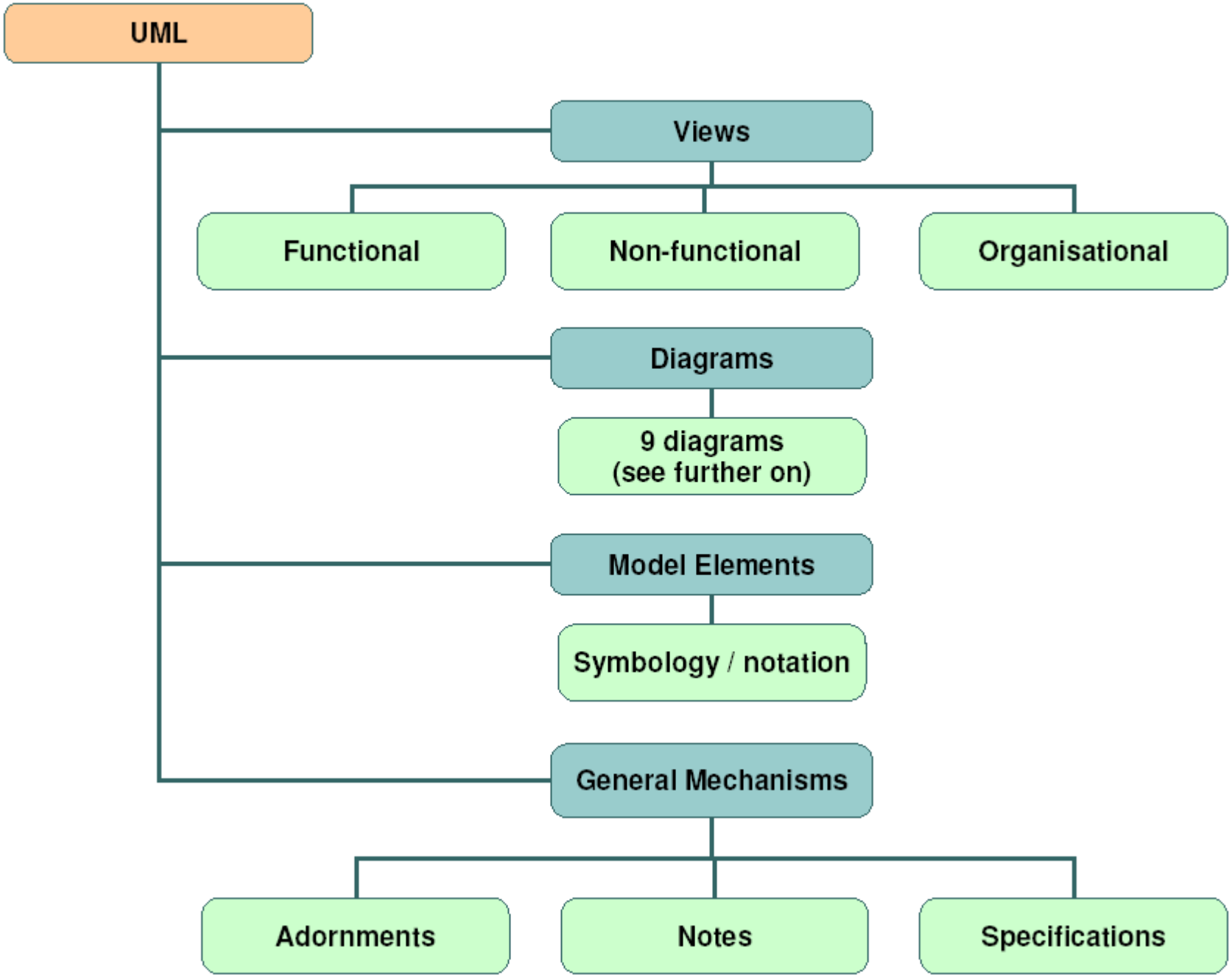
- Các lớp của giai đoạn thiết kế sẽ được biến thành những dòng code cụ thể trong một ngôn ngữ lập trình hướng đối tượng cụ thể
  - Phụ thuộc vào khả năng của ngôn ngữ được sử dụng, đây có thể là một công việc khó khăn hay dễ dàng.
  - Khi tạo ra các mô hình phân tích và thiết kế trong UML, tốt nhất nên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

# 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 (collaborationdiagram)
  - 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

# Các thành phần của UML

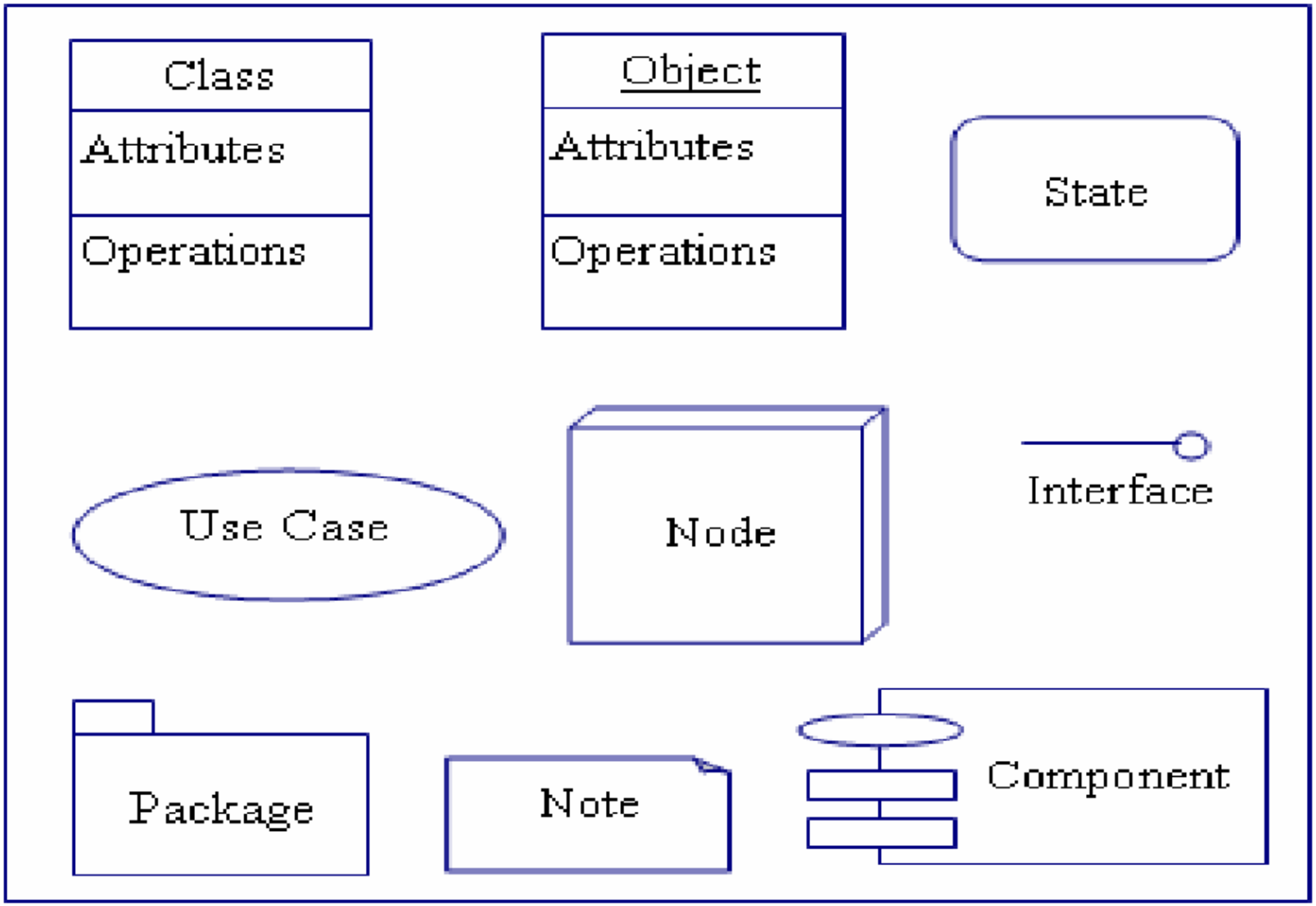
- Khung nhìn (view): chỉ ra những khía cạnh khác nhau của hệ thống cần mô hình hóa.
- Biểu đồ (diagram): các hình vẽ miêu tả nội dung trong một khung nhìn.
  - UML có tất cả 9 loại biểu đồ
- Phần tử mô hình hóa (model element): Các khái niệm sử dụng trong các biểu đồ được gọi là các phần tử mô hình
- 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; chúng còn 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 trong UML

- Các khối để hình thành mô hình UML gồm ba loại
  - Phần tử
  - Quan hệ
  - Lược đồ.
- Phần tử mô hình (model element) : các khái niệm được sử dụng trong các biểu đồ
  - Có bốn loại phần tử mô hình: cấu trúc, hành vi, nhóm và chú thích.
- Quan hệ: gắn các phần tử này lại với nhau
- Lược đồ: tập hợp các phần tử.

# Phần tử mô hình

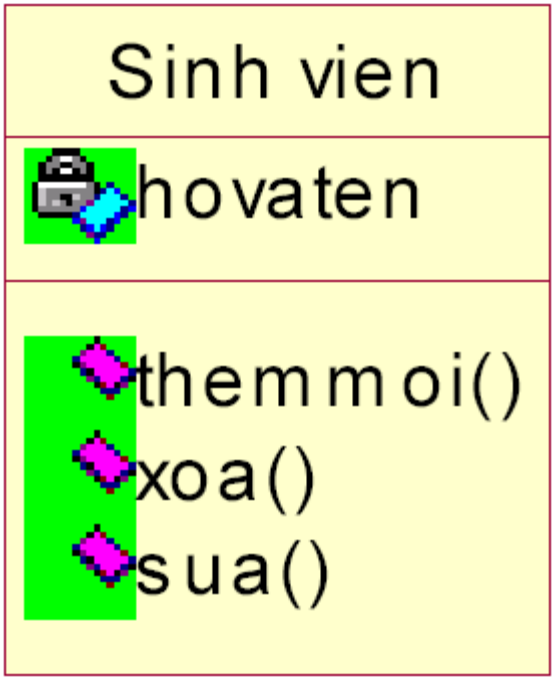


# Phần tử cấu trúc

- 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): thể hiện thành phần vật lý, tồn tại khi chương trình chạy, biểu diễn các tài nguyên tính toán. Có thể đặt tập các thành phần trên nút và chuyển từ nút này sang nút khác, có thể là máy tính, thiết bị phần cứng.

# Lớp

- Là mô tả tập các đối tượng cùng chung thuộc tính, thao tác, quan hệ và ngữ nghĩa.
- Lớp được minh hoạ bằng hình chữ nhật, thông thường chúng có tên, thuộc tính và thao tác

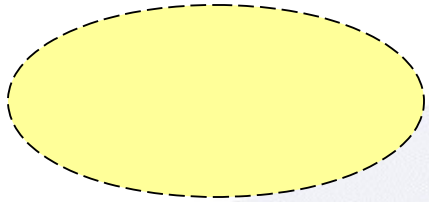


# Giao diện

- Tập hợp các thao tác làm dịch vụ của 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 tập đặc tả thao tác, không định nghĩa cài đặt của chúng.

# Phần tử cộng tác

- 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.



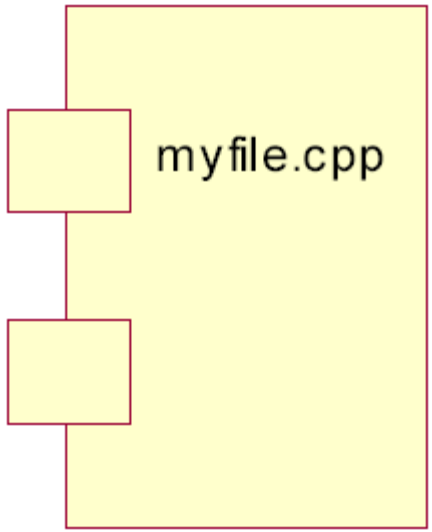
# Trường hợp sử dụng (Use case)

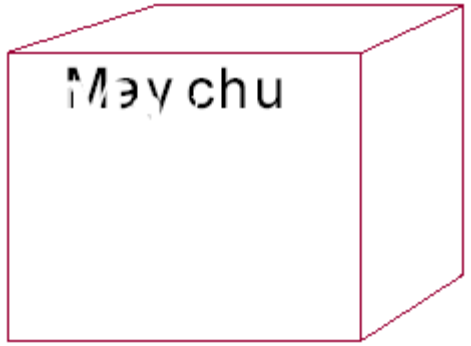
- Mô tả trình tự các hành động hệ thống sẽ thực hiện để đạt được một kết quả cho tác nhân nào đó.
- Tác nhân: những đối tượng bên ngoài tương tác với hệ thống.
- Tập hợp các Use case của hệ thống sẽ hình thành các trường hợp mà hệ thống được sử dụng.
- Sử dụng Use case để cấu trúc các phần tử có tính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

Them sinh vien

# Thành phần và nút

- Biểu diễn vật lý 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 đặt tập các thành phần trên nút và chuyển từ nút này sang nút khác, có thể là máy tính, thiết bị phần cứng.



# Phần tử hành vi

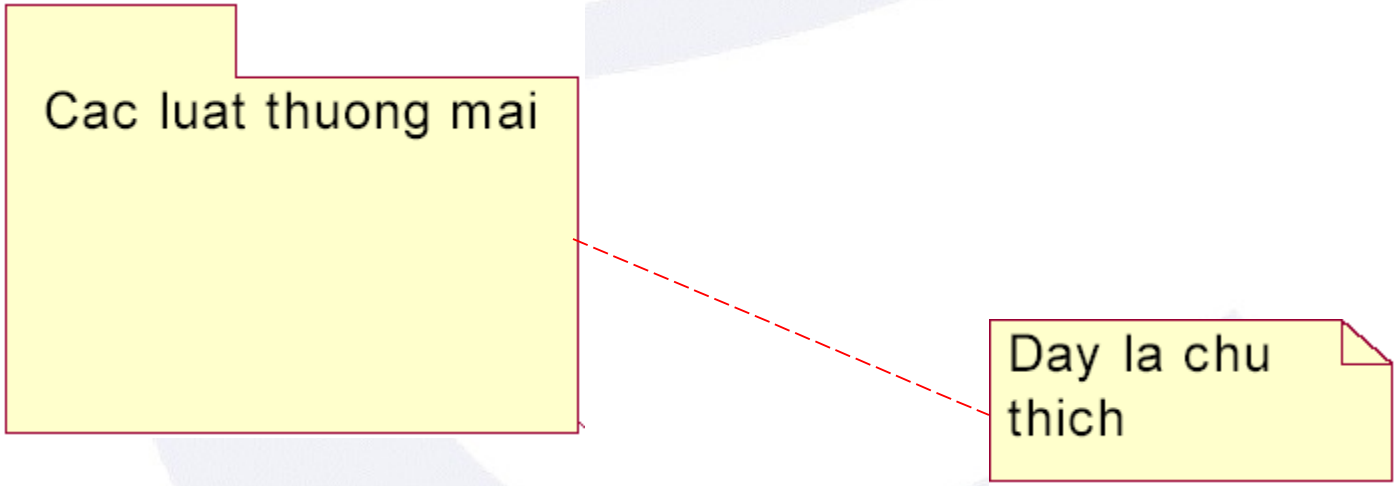
- Là bộ phận động hay các động từ của mô hình, biểu diễn hành vi theo thời gian và không gian dưới hai hình thức
  - Tương tác: là hành vi bao gồm tập các thông điệp trao đổi giữa các đối tượng trong ngữ cảnh cụ thể để thực hiện mục đích cụ thể. Hành vi của nhóm đối tượng hay của thao tác có thể được chỉ ra bằng tương tác.
  - Máy trạng thái: là hành vi chỉ ra trật tự các trạng thái mà đối tượng hay tương tác sẽ đi qua để đáp ứng sự kiện. Hành vi của lớp hay cộng tác của lớp có thể được xác định bằng máy trạng thái. Máy trạng thái kích hoạt nhiều phần tử, bao gồm trạng thái, chuyển tiếp từ trạng thái này sang trạng thái khác, sự kiện và các hoạt động đáp ứng sự kiện

# Phần tử nhóm

- Là bộ phận tổ chức của mô hình.
- Chỉ có một phần tử thuộc nhóm này có tên là gói (package).
- Gói là cơ chế đa năng để tổ chức các phần tử vào 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

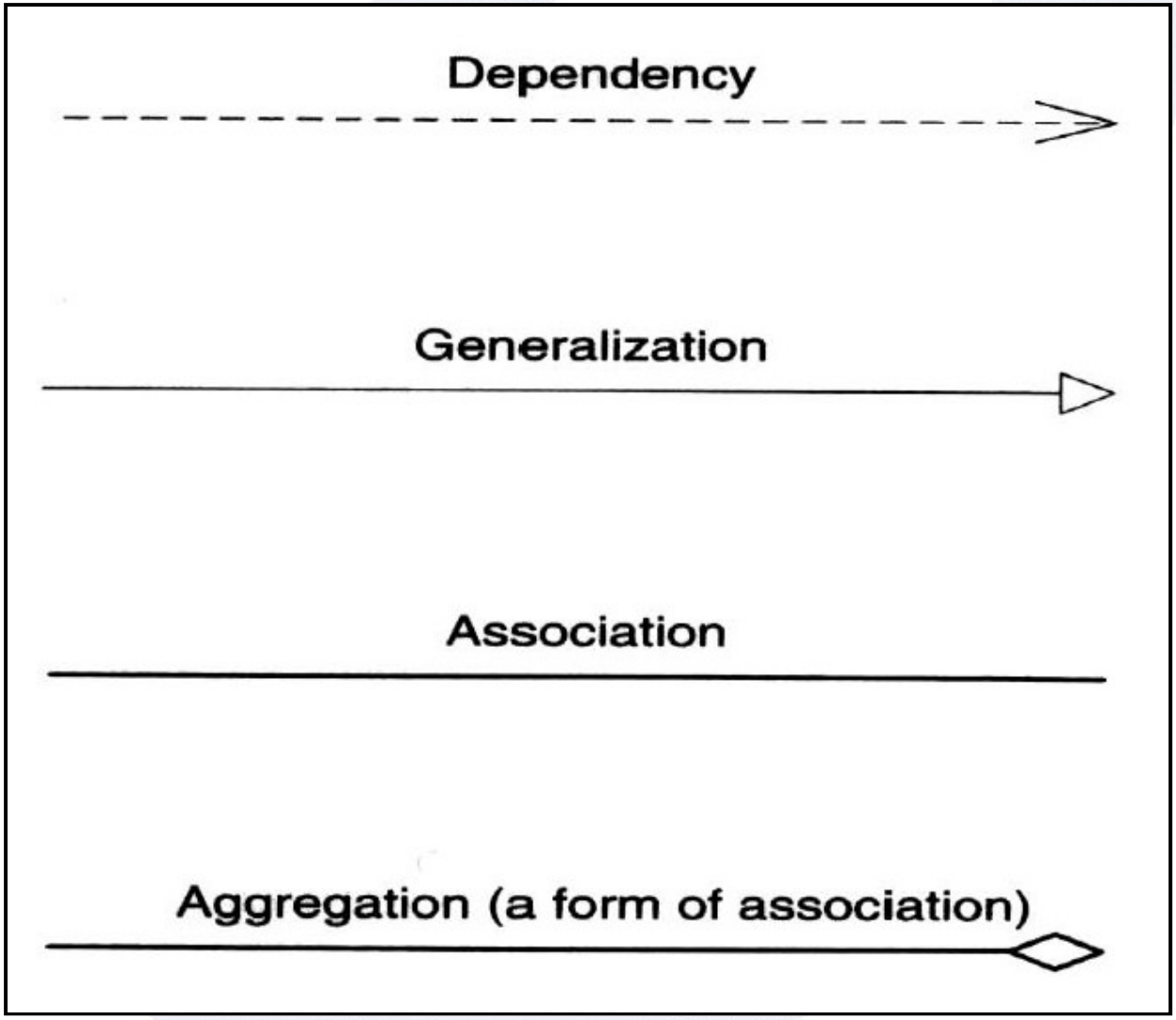
- Là bộ phận chú giải của mô hình UML.
- Phần tử chú thích được gọi là lời ghi chú (note) và dùng để mô tả các phần tử khác trong mô hình.



# Các quan hệ trong UML

- Có bốn loại quan hệ trong UML
  - Kết hợp (Association) : nối các phần tử và các thực thể nối (link).
  - 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.
  - Tụ hợp/bao gộp (Aggregation): Một dạng của nối kết, trong đó một phần tử này chứa các phần tử khác.
- Là cơ sở để xây dựng mọi quan hệ trong UML.

# Quan hệ trong UML

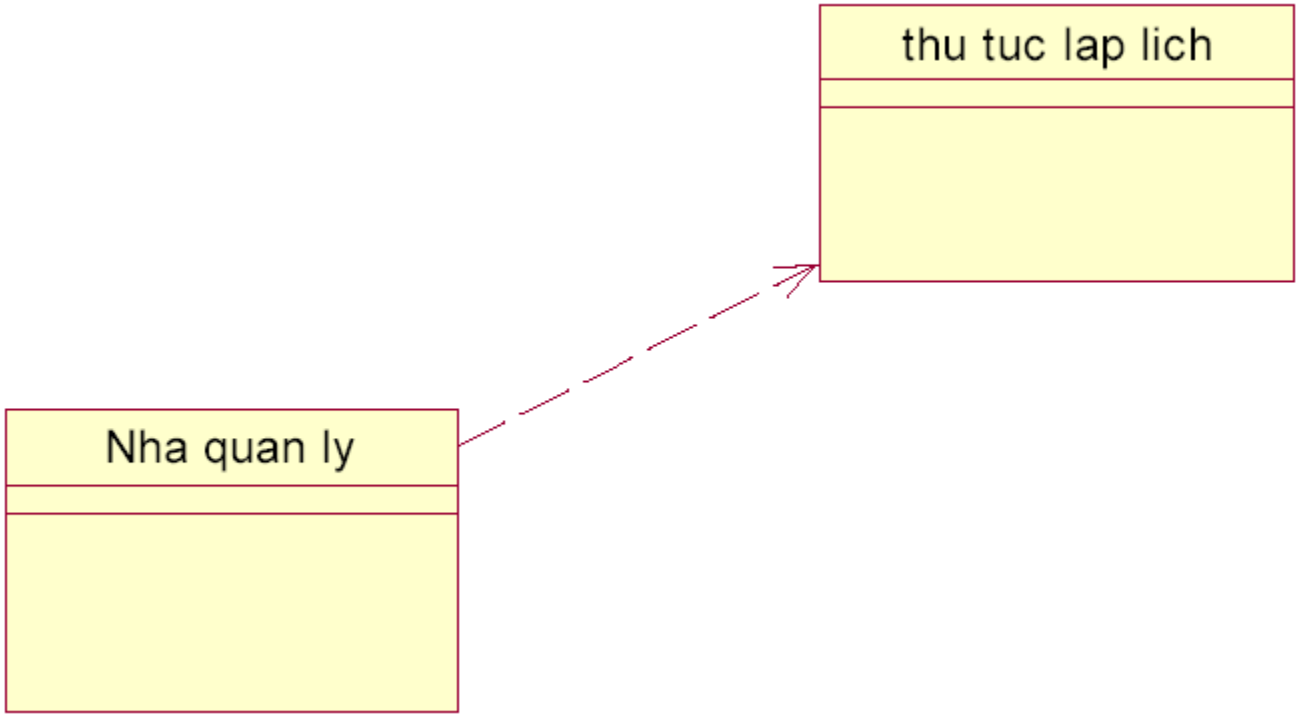


# Quan hệ kết hợp (Association)

- 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.
- Chúng có thể chứa tên nhiệm vụ và tính nhiều (mutiplicity).

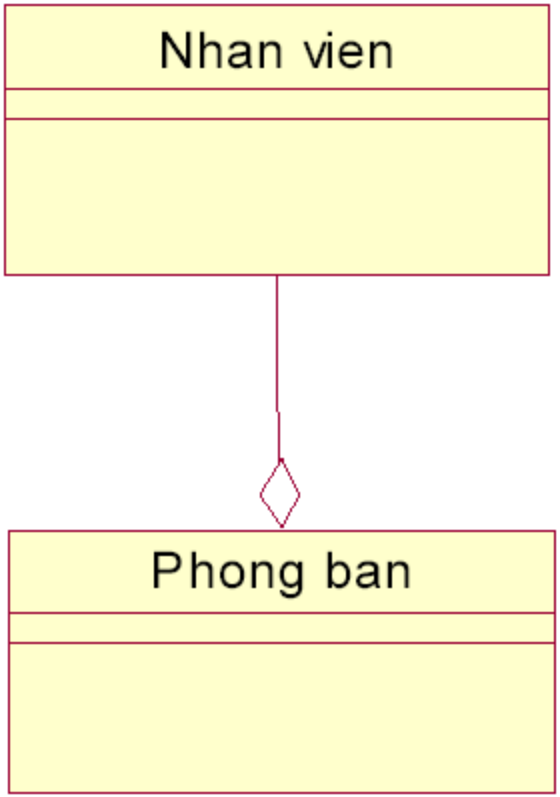
# Quan hệ phụ thuộc (dependency)

- 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.



# Tụ hợp/Bao gộp (aggregration)

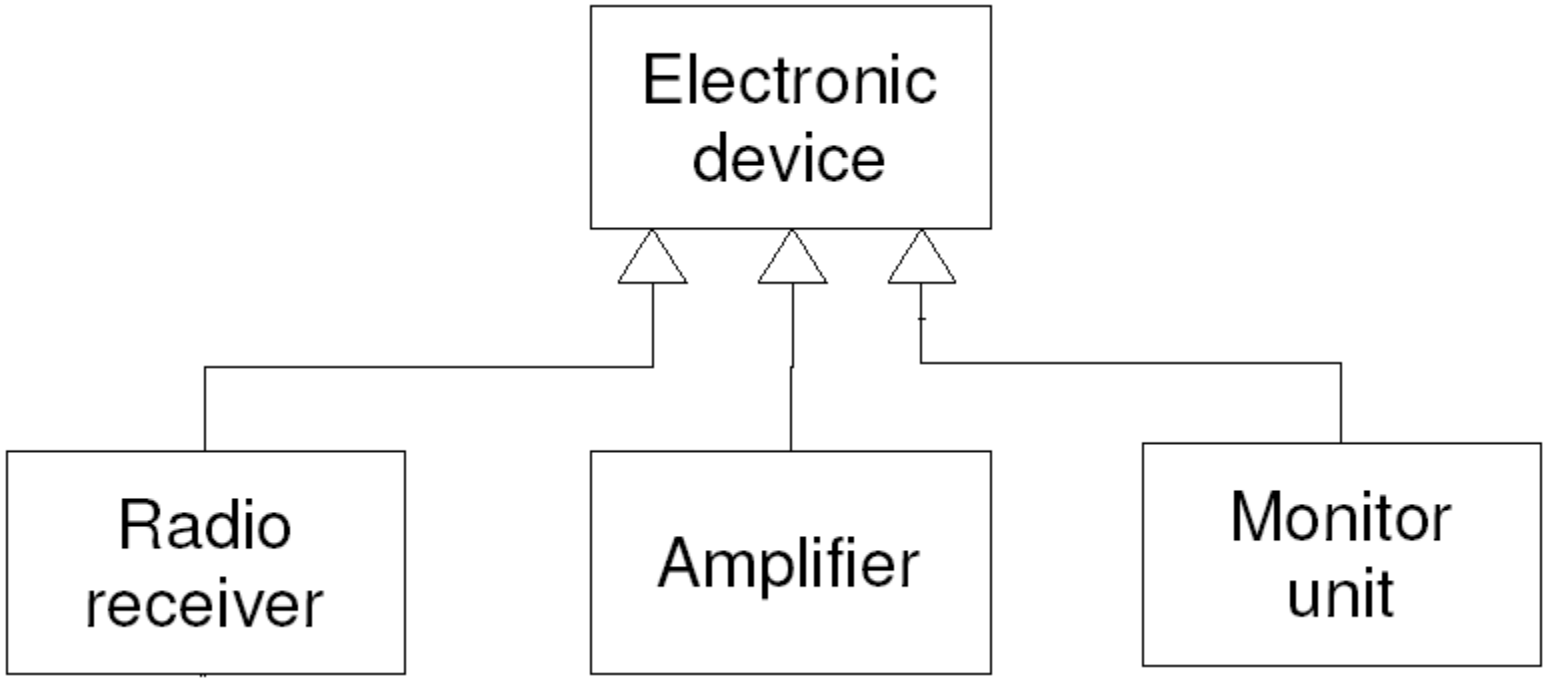
- Là dạng đặc biệt của kết hợp, biểu diễn quan hệ cấu trúc giữa toàn thể và bộ phận.
- Một dạng đặc biệt của tụ hợp là quan hệ hợp thành (composition), trong đó nếu như đối tượng toàn thể bị huỷ bỏ thì các đối tượng bộ phận của nó cũng bị huỷ bỏ theo.



# 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 hợp tác hiện thực Use case.

# Khái quát hóa và hiện thực hóa



# Kiến trúc hệ thống

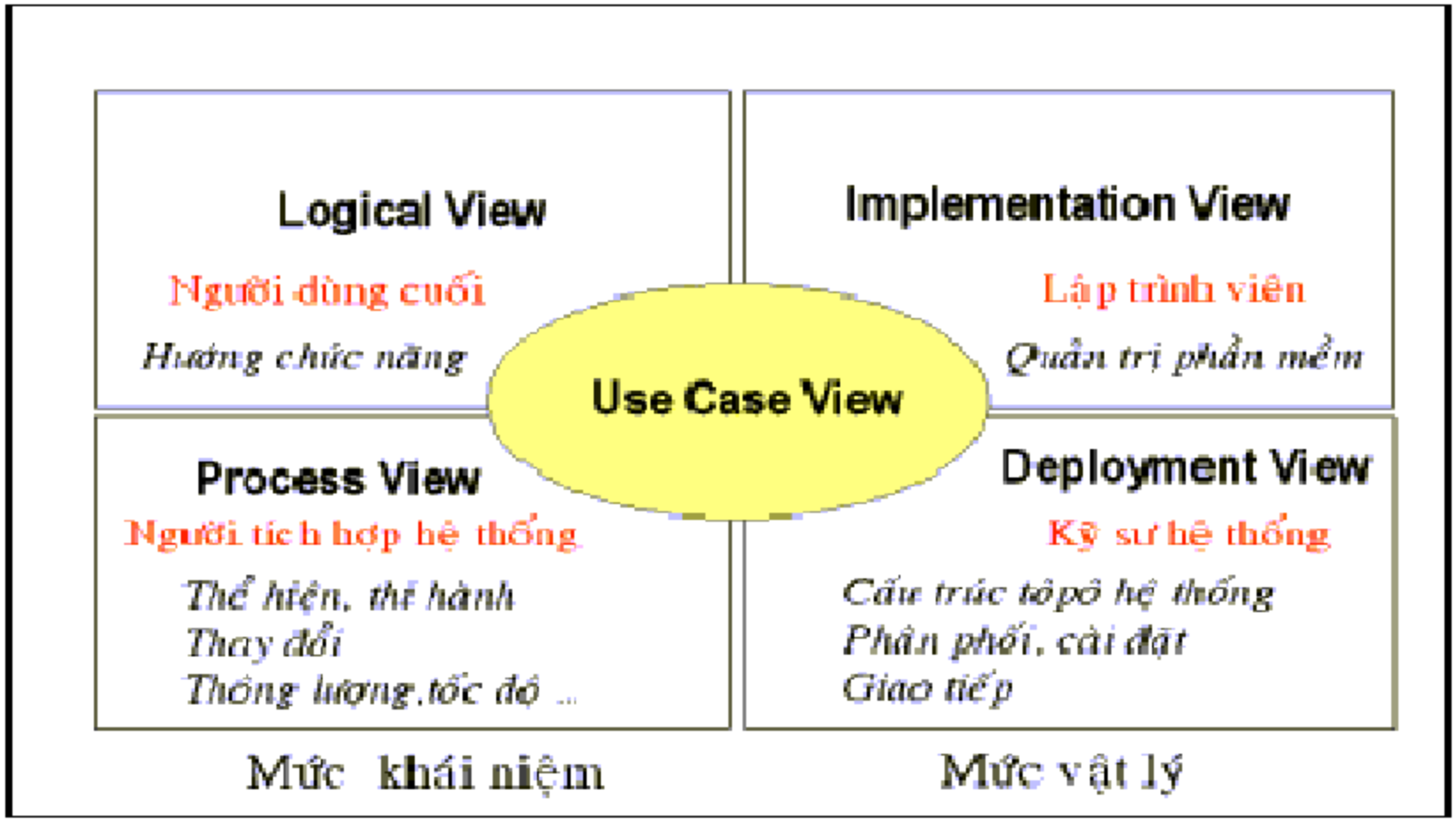
- Không thể mô hình hoá một hệ thống phức tạp chỉ bằng một mô hình hay lược đồ, hệ thống phải được phân tích dưới những góc độ khác nhau.
- Việc hiển thị, đặc tả, xây dựng và làm tài liệu hệ thống đòi hỏi hệ thống phải được xem xét từ nhiều khía cạnh khác nhau
- Kiến trúc hệ thống được sử dụng để quản 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

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

# Khung nhìn Use Case

- 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 do được tác nhân từ bên ngoài mongđợi.
- 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

- Khung nhìn Use case là khung nhìn dành cho khách hàng, nhà thiết kế, nhà phát triển và người thử nghiệm; nó 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 (Hỏi: "Đây có phải là thứ bạn muốn") cũng như có đúng với hệ thống vừa được hoàn thành (Hỏi: "Hệ thống có hoạt động như đã đặc tả?").

# Khung nhìn Logic (Logic View)

- Miêu tả phương thức mà các chức năng của hệ thống sẽ được cung cấp.
  - Chủ yếu nó được sử dụng cho các nhà thiết kế và nhà phát triển.
- Đi vào phía bên trong của hệ thống 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).
- Định nghĩa các thuộc tính như trường tồn (persistency), song song (concurrency), giao diện cũng như cấu trúc nội tại của các lớp

# Khung nhìn song song (Cuncurrency View)

- Khung nhìn song song 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 trìnhđó.
- 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).

# Khung nhìn thành phần (Component View)

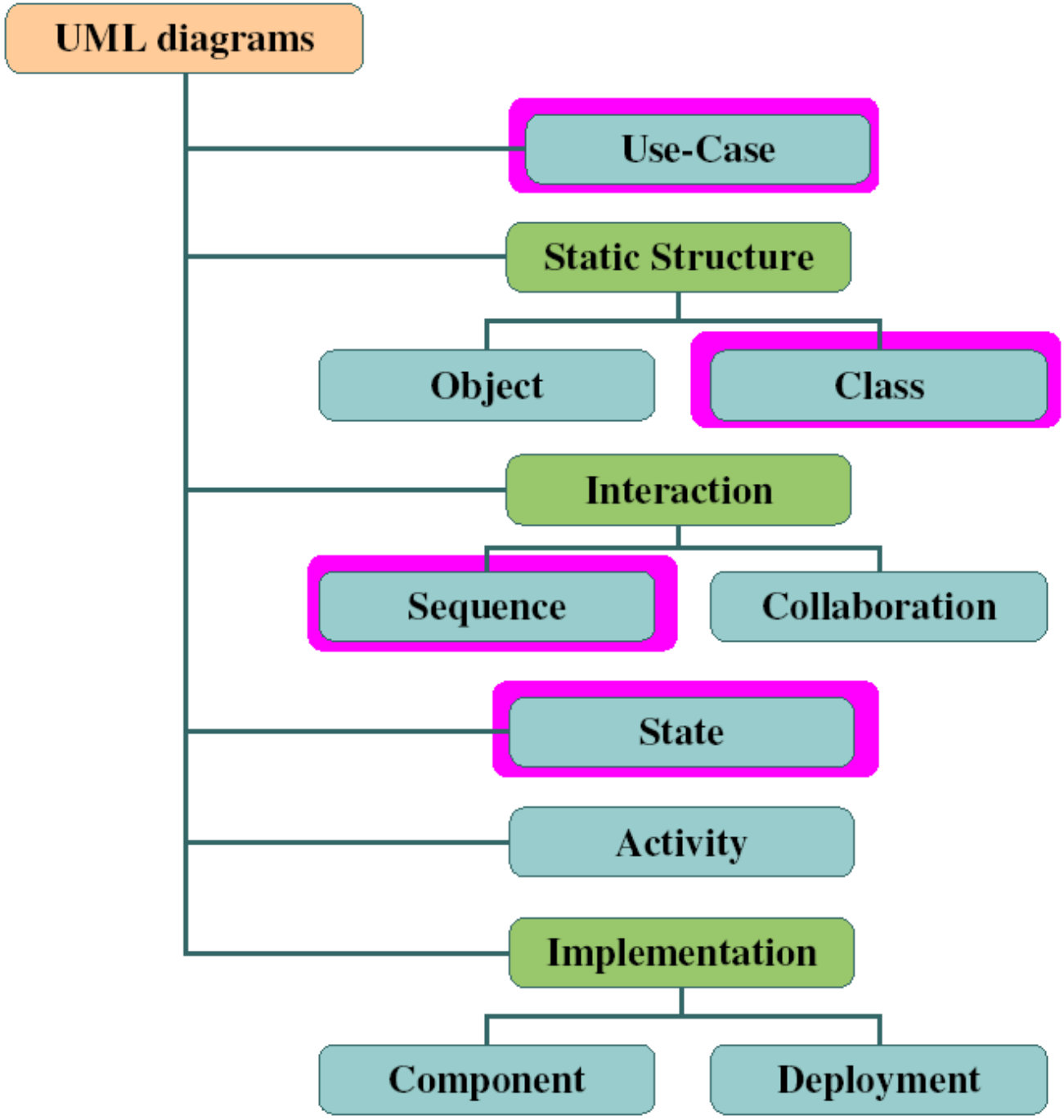
- Đặc tả của các mô đun cài đặt
  - Thường được sử dụng cho nhà phát triển, bao gồm nhiều biểu đồ thành phần.
- Thành phần ở đây là các modul thuộc nhiều loại khác nhau, được chỉ ra trong biểu đồ cùng với cấu trúc cũng như sự phụ thuộc của chúng.

# Khung nhìn triển khai (Deployment View)

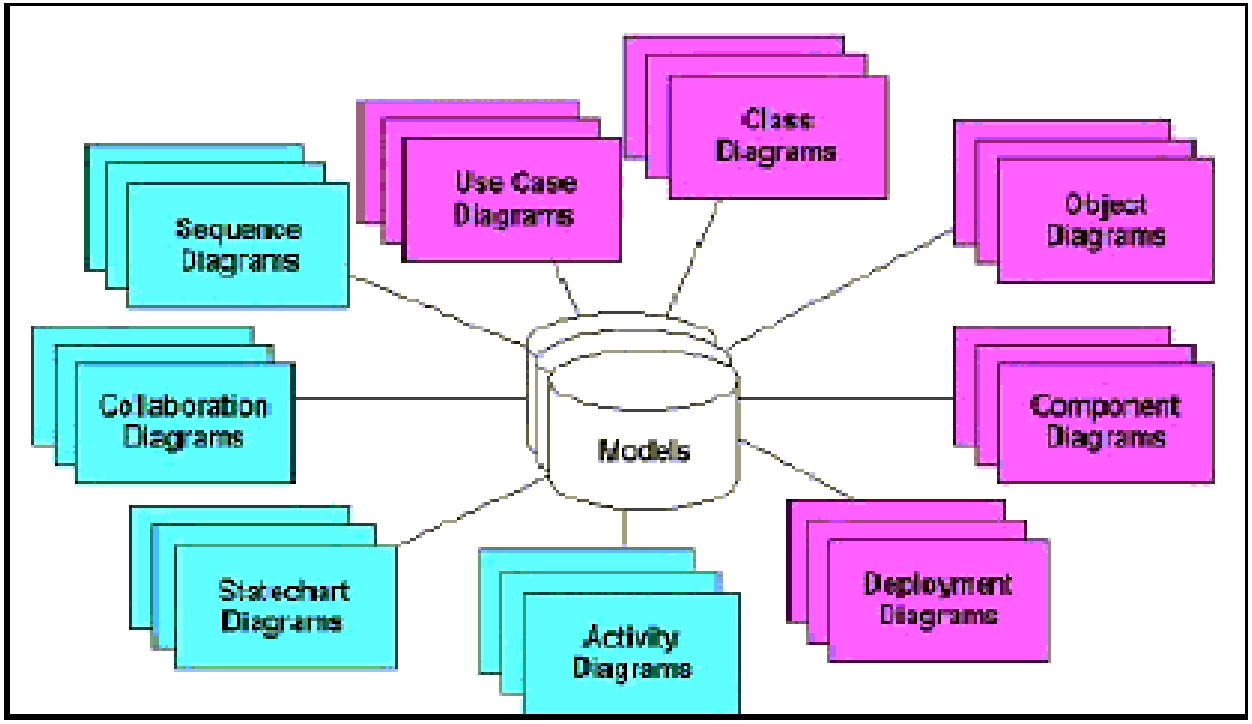
- Hướng nhìn triển khai chỉ cho chúng ta sơ đồ triển khai về mặt vật lý của hệ thống
  - Ví dụ: máy tính, máy móc và sự liên kết giữa chúng.
- Được sử dụng bởi người phát triển, tích hợp và test hệ thống
- Được thể hiện bằng các biểu đồ triển khai - Deployment diagram
- Bao gồm sự ánh xạ các thành phần 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)

- Là các hình vẽ bao gồm các ký hiệu phần tử mô hình hóa được sắp xếp để minh họa một thành phần cụ thể hay một khía cạnh cụ thể của hệ thống.
- Một mô hình hệ thống thường có nhiều loại biểu đồ, mỗi loại có nhiều biểu đồ khác nhau. Một biểu đồ là một thành phần của một khung nhìn cụ thể; và khi được vẽ ra, nó thường được xếp vào một khung nhìn. Mặt khác, một số loại biểu đồ có thể là thành phần của nhiều khung nhìn khác nhau, tùy thuộc vào nội dung của biểu đồ.


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

# Biểu đồ


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

# Biểu đồ

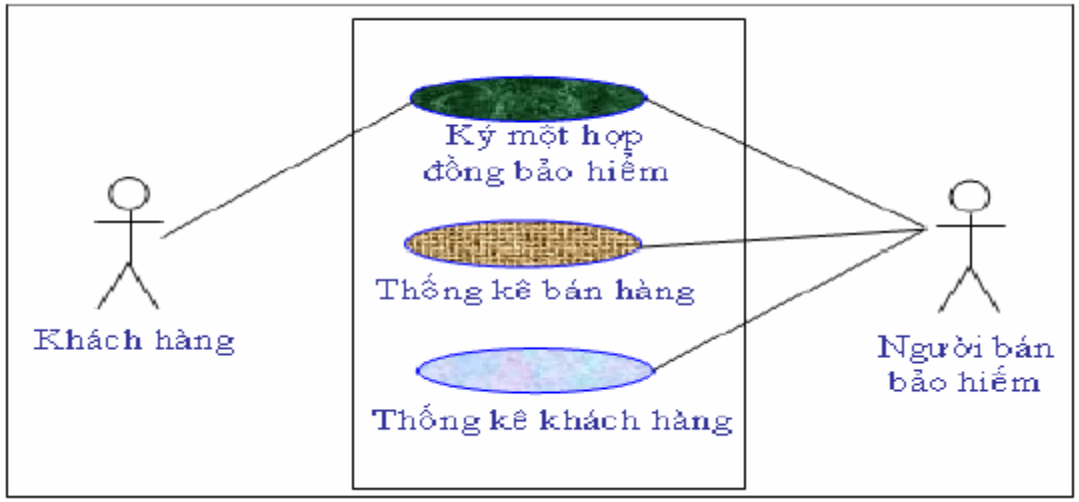
- Use case: nắm bắt các yêu cầu
- Sequence: Thể hiện thời gian tương tác giữa các đối tượng
- Collaboration: Làm nổi bật các quan hệ giữa các đối tượng và liên kết giữa chúng
- Class: thể hiện cấu trúc tĩnh của hệ thống
- State: các trạng thái của đối tượng
- Activity: các hoạt động song song và các tiến trình kinh doanh
- 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.
- Hình thức mô tả: văn bản, tài liệu, biểu đồ hoạt động
- Use case diagram thể hiện các Actor liên hệ như thế nào với các use case
  - Actors: là một người hay cái gì đó nằm ngoài hệ thống và tương tác với hệ thống
  - Use case có thể được tinh lọc từ các use case khác

# Use Case Diagram

- 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 l

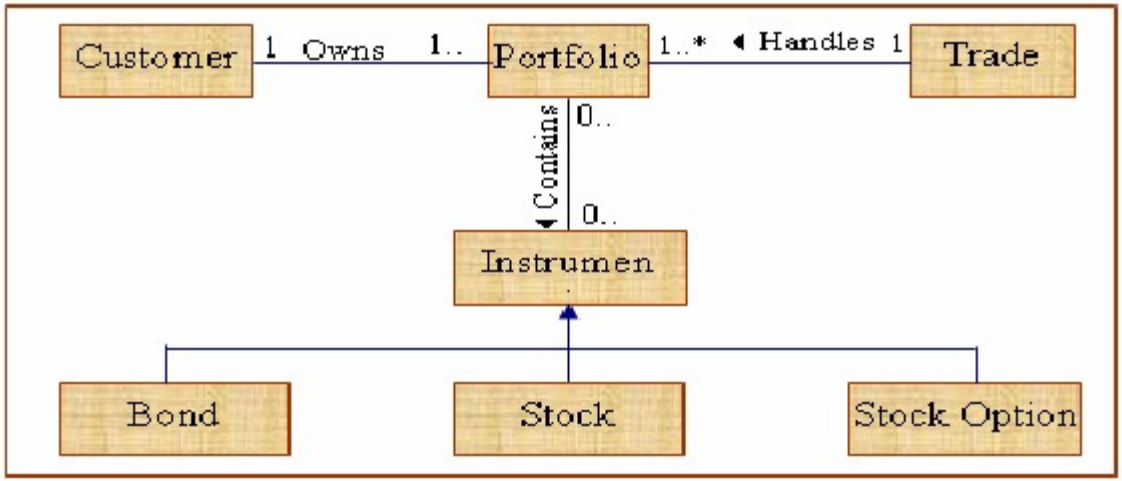


# Biểu đồ lớp (Class Diagram)

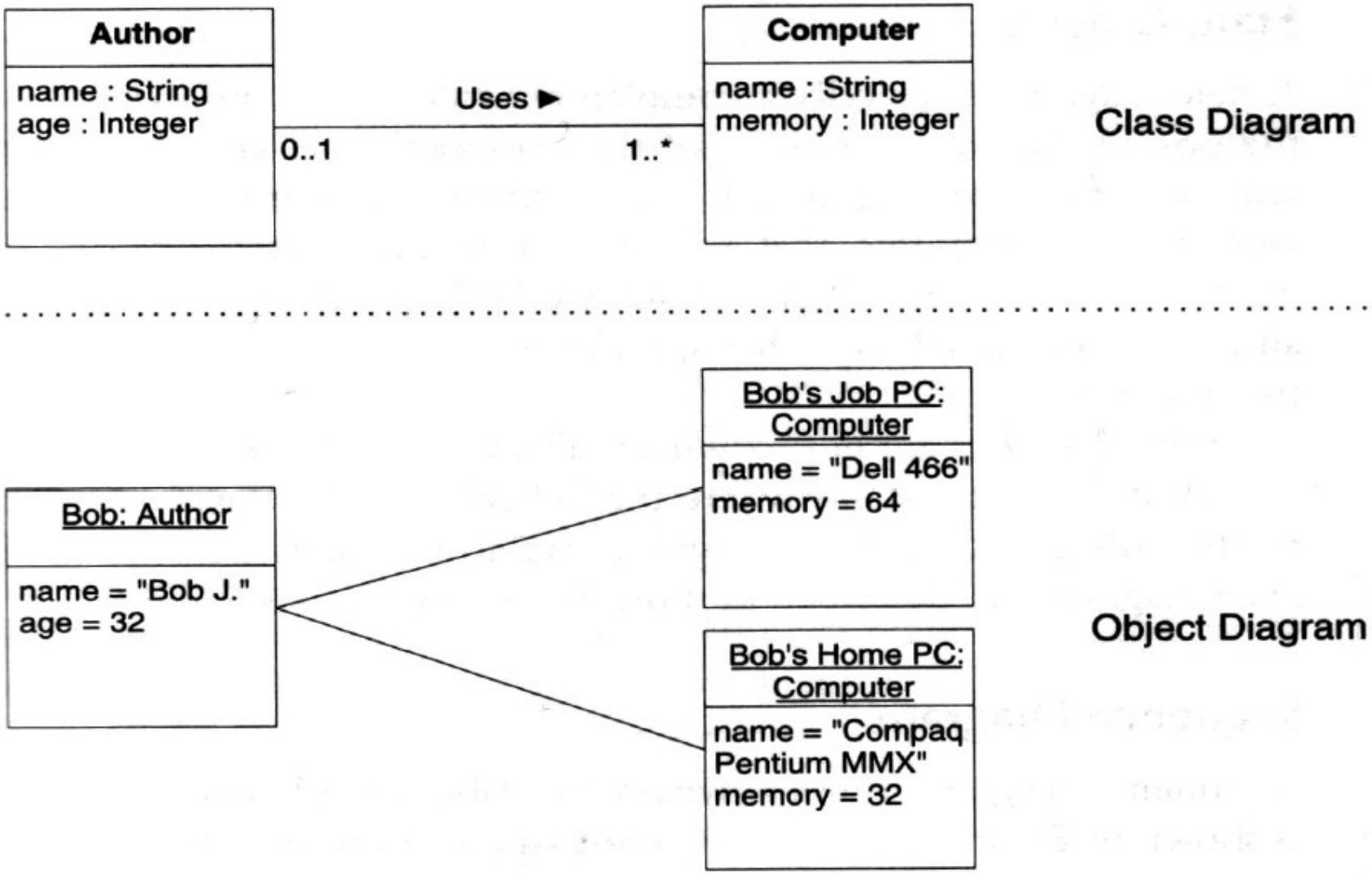
- Chuyển từ khung nhìn use case sang khung nhìn logic của hệ thống
- Class là thành phần chính dùng để xây dựng bất kì một hệ thống hướng đối tượng nào
- Class diagram quản lý các class và quan hệ giữa chúng
- Class là một mô tả của kiểu object, bao gồm cấu trúc và các tác động
- Các thể hiện của một class là các objects

# Biểu đồ lớp (Class Diagram)

- Biểu đồ lớp chỉ ra cấu trúc tĩnh của các lớp trong hệ thống
- Một hệ thống thường sẽ có một loạt các biểu đồ lớp

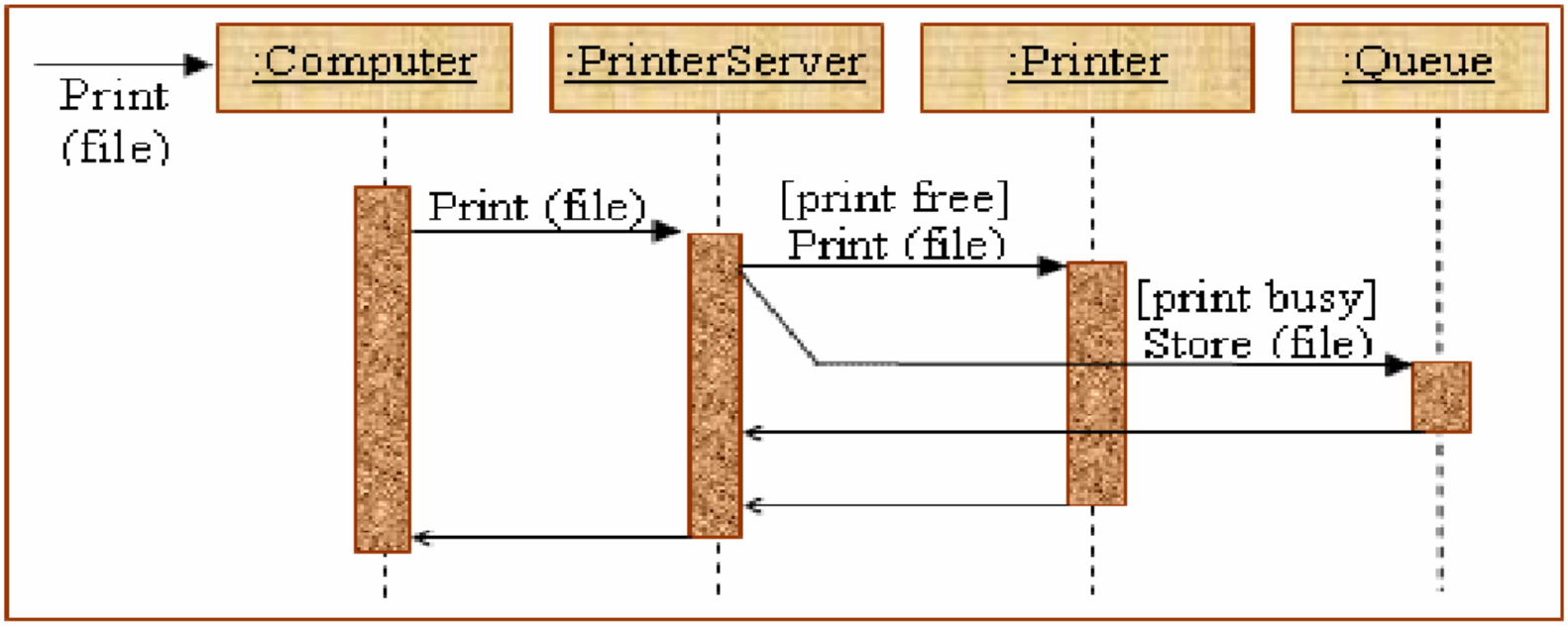


# Biểu đồ đối tượng (Object Diagram)



# 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.

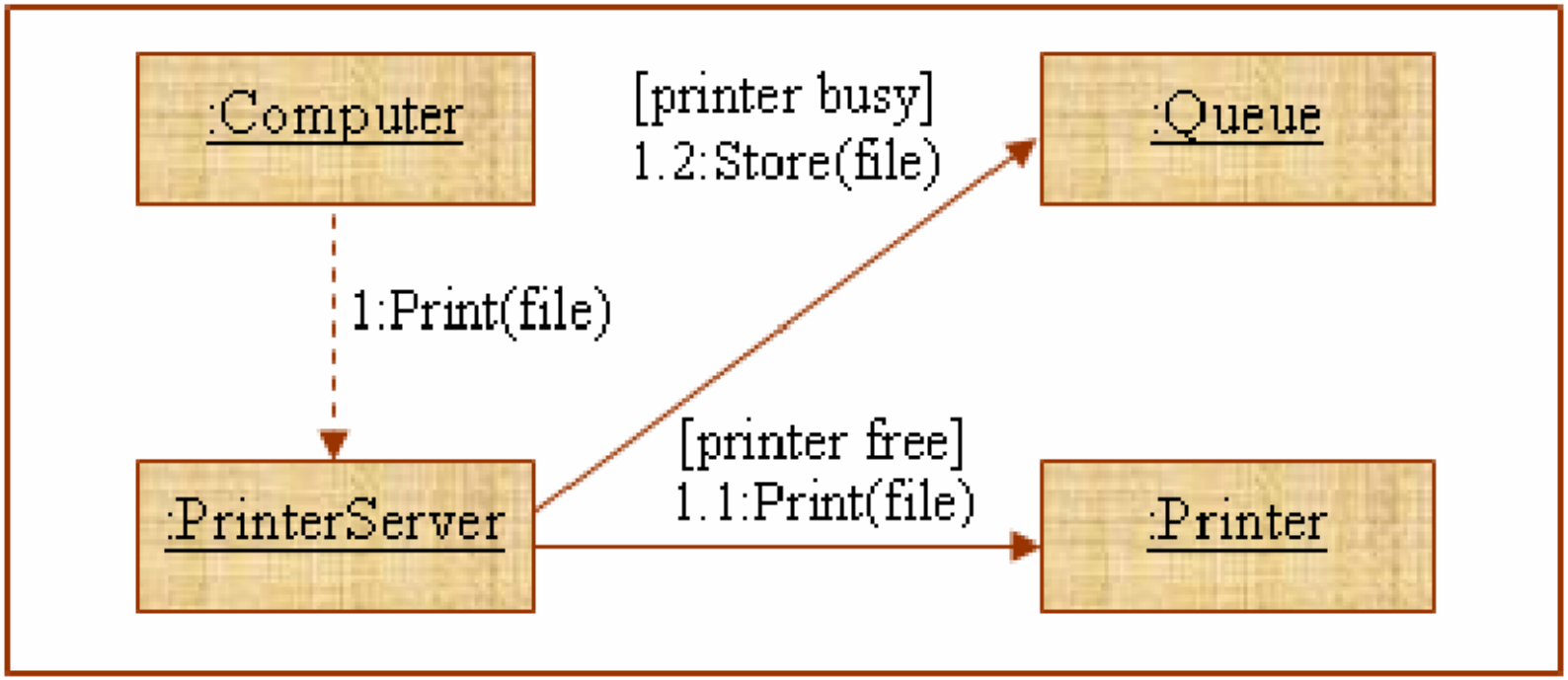


# Biểu đồ cộng tác (CollaborationDiagram)

- 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 (CollaborationDiagram)

- Bên cạnh việc thể hiện sự trao đổi thông điệp (gọi là tương tác), biểu đồ cộng tác chỉ ra các đối tượng và quan hệ của chúng (đôi khi được gọi là ngữ cảnh).

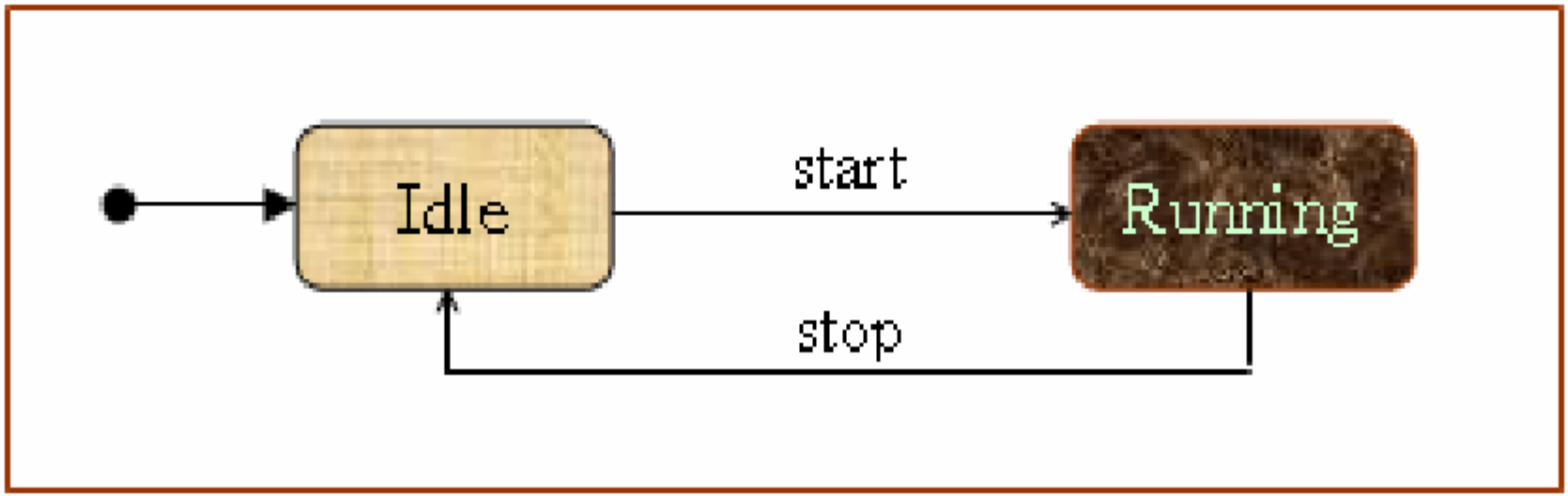


# Biểu đồ trạng thái (State Diagram)

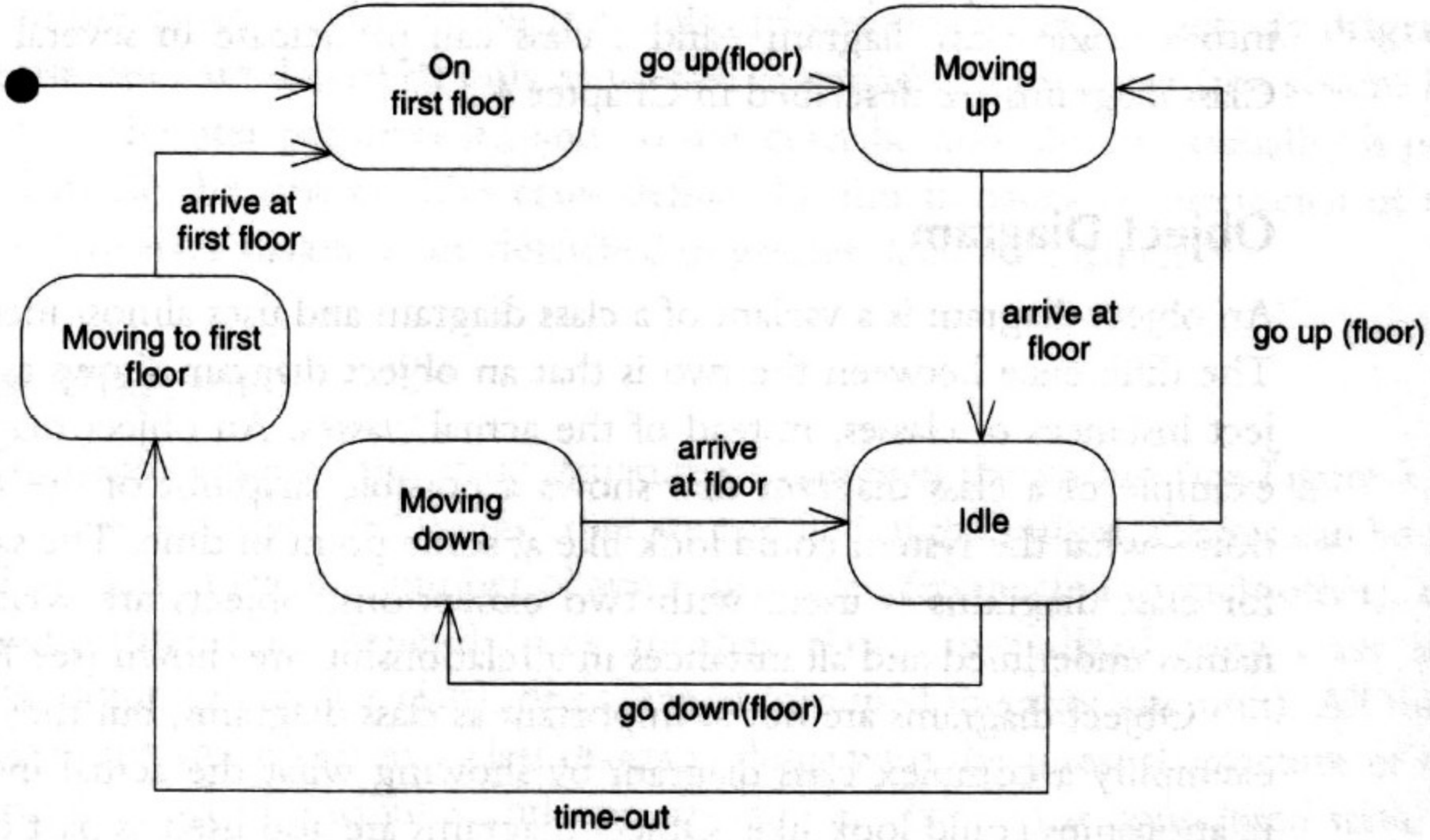
- Mô hình hóa các tác động bên trong đối tượng
- UML state diagram được xây dựng dựa trên StateCharts của David Harel
  - Nâng cấp các khả năng của state diagram để mô hình các hệ thống lớn
  - Một cách chính thức, state diagram được gọi là “statechart diagrams”
- State diagram nên được sử dụng cho các object với các hành vi động

# Biểu đồ trạng thái (State Diagram)

- 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 (State Diagram)



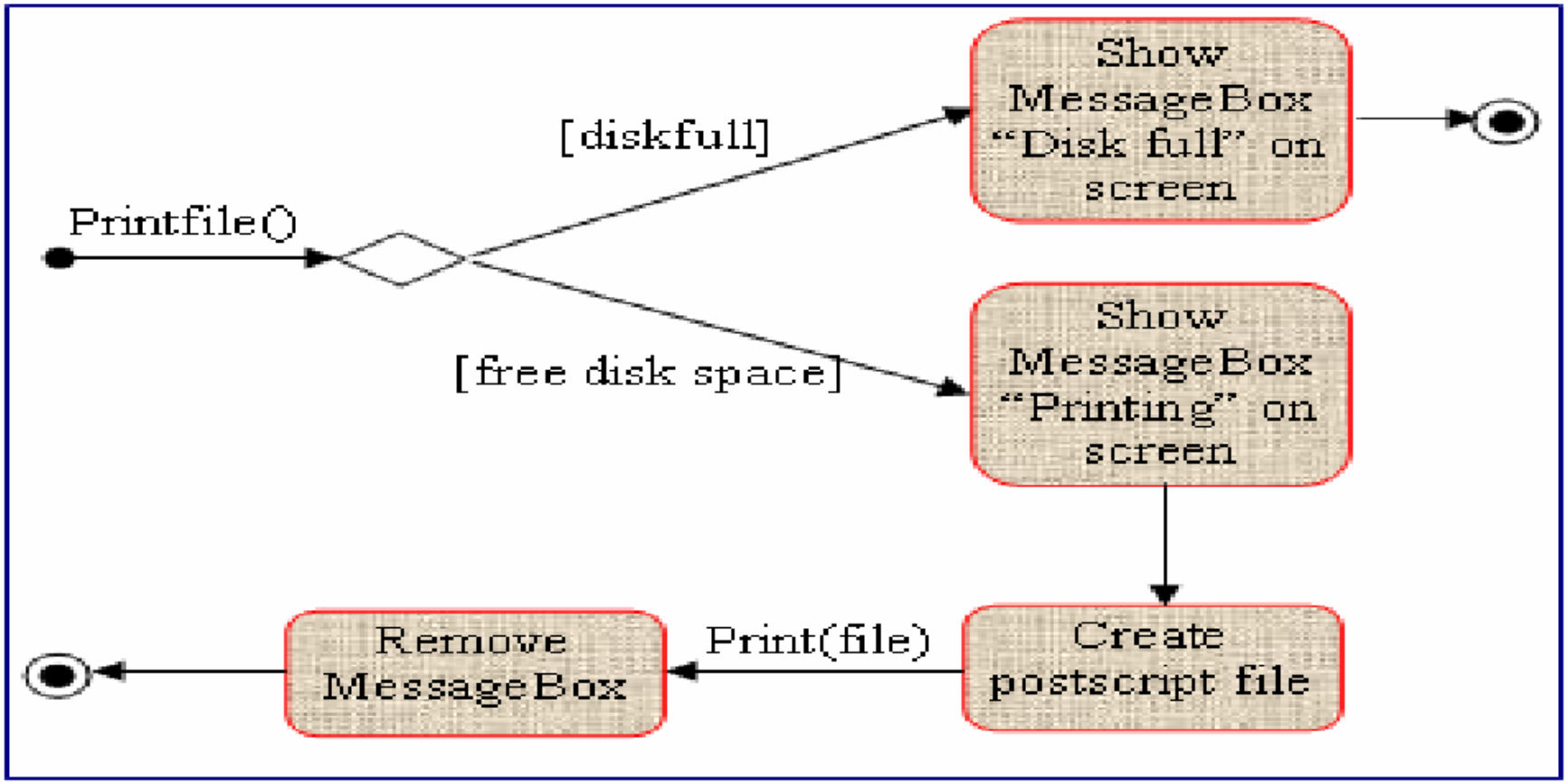
# Biểu đồ hoạt động (Activity Diagram)

- Mô hình hóa các tương tác động của hệ thống
- Miêu tả các tiến trình song song của hệ thống
- Được hình hành bởi sự kết hợp các ý tưởng và khái niệm của nhiều nguồn:
  - BPR của Rummler-Brache
  - Event diagram của J Odell
  - SDL state
- Dùng để mô tả workflow

# 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.
- Biểu đồ hoạt động bao gồm các trạng thái hành động, chứa đặc tả của một hoạt động (một hành động - action).
- 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 (Activity Diagram)

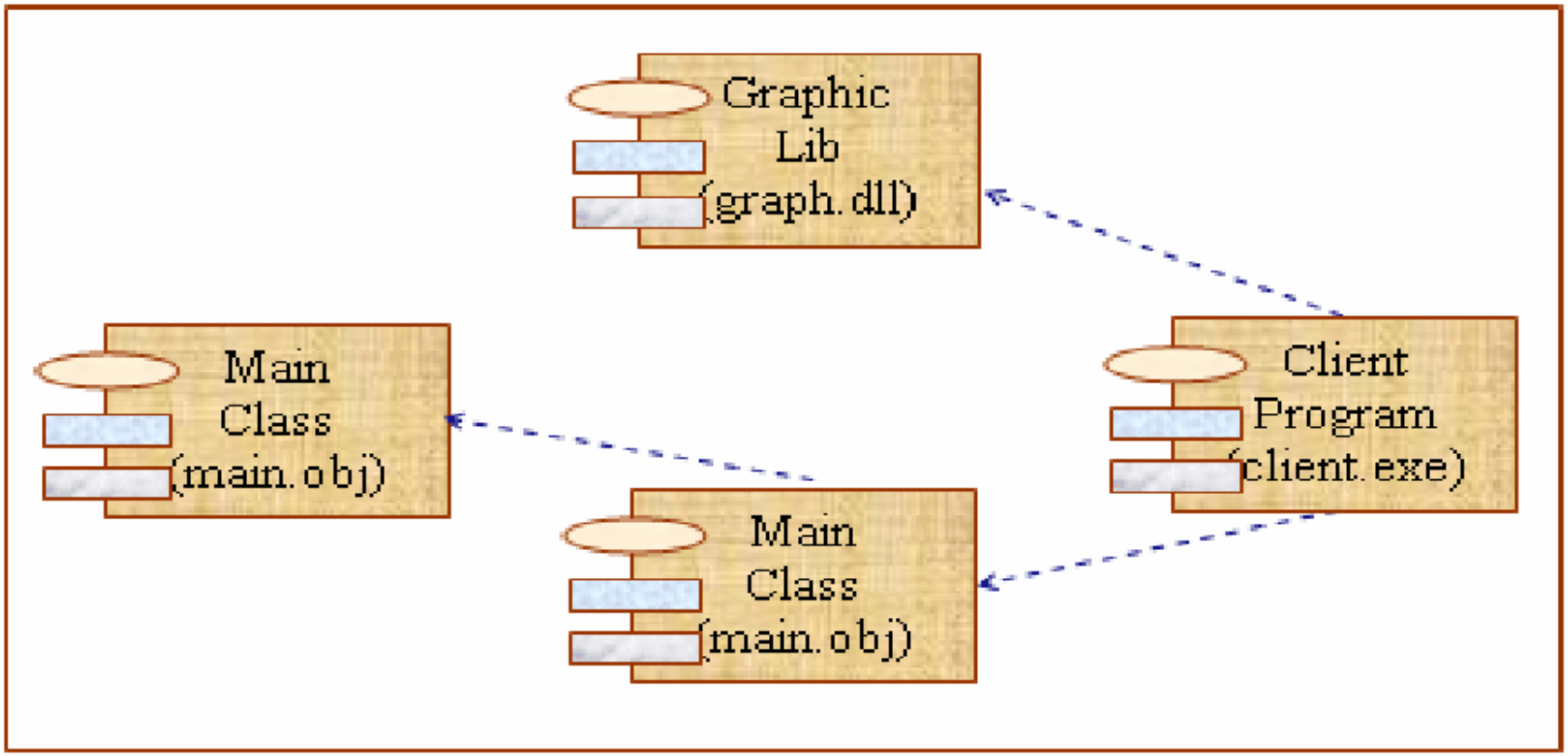


# Biểu đồ thành phần (Component)

- Miêu tả cấu trúc vật lí của phần mềm
  - Chỉ ra cấu trúc vật lý của các dòng lệnh (code) theo khái niệm thành phần code
  - Một thành phần chứa các thông tin về các lớp logic hoặc các lớp mà nó thi hành, như thế có nghĩa là nó tạo ra một ánh xạ từ hướng nhìn logic vào hướng nhìn thành phần.
- Liên kết phụ thuộc nghĩa là một component cần một component khác để có một định nghĩa hoàn chỉnh
- Một component có thể định nghĩa các interface (giao diện) tới các component khác và thể hiện rõ ràng trong diagram

# Biểu đồ thành phần (Component)

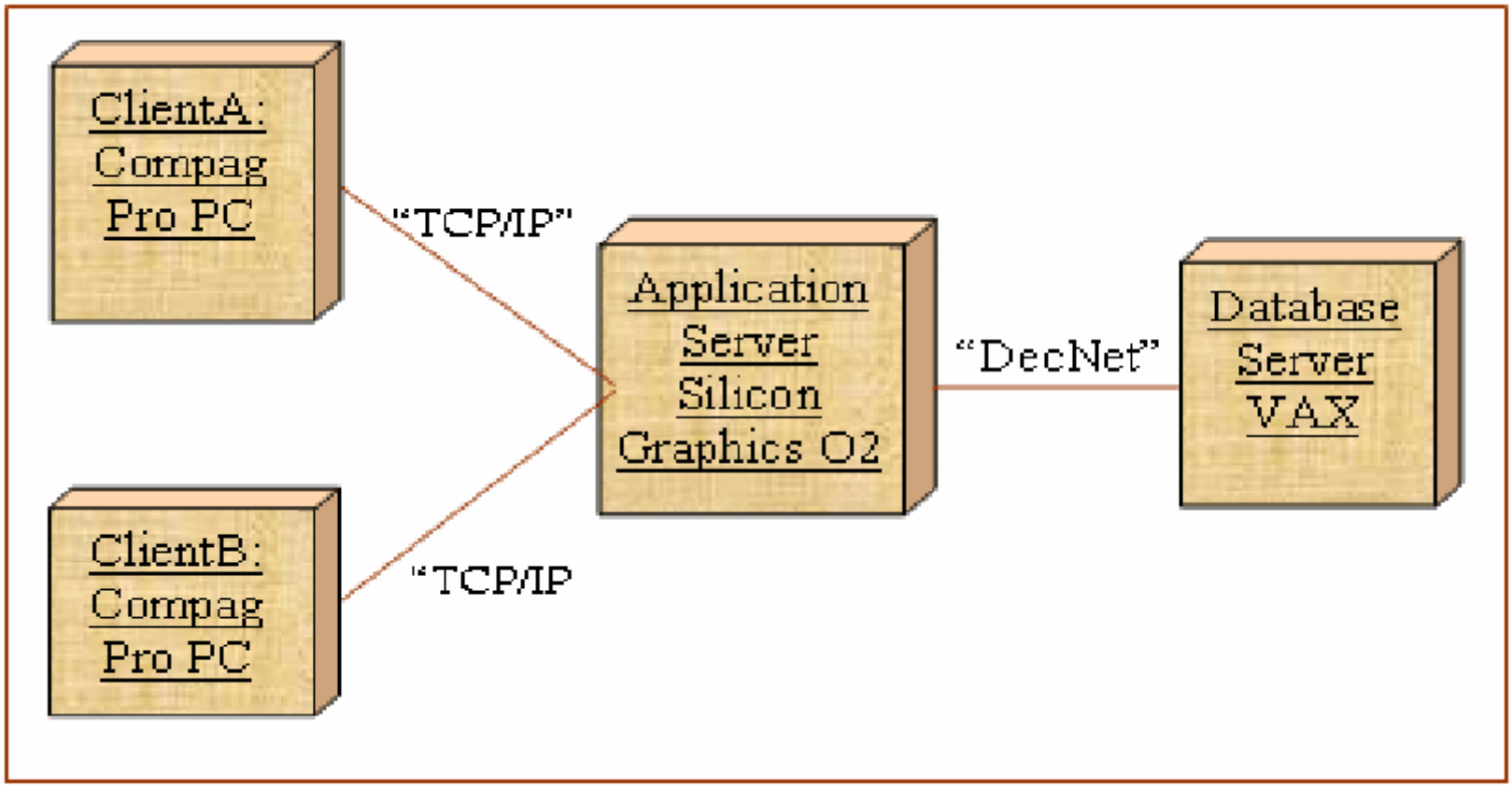
- Biểu đồ thành phần chỉ ra sự phụ thuộc giữa các thành phần mã



# Biểu đồ triển khai (Deployment)

- 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
  - Đây là một hướng nhìn rất xa lối miêu tả duy chức năng của hướng nhìn Use case. Mặc dù vậy, trong một mô hình tốt, có thể chỉ tất cả những con đường dẫn từ một nút mạng trong một kiến trúc vật lý cho tới những thành phần của nó, cho tới lớp mà nó thực thi, cho tới những tương tác mà các đối tượng của lớp này tham gia để rồi cuối cùng, tiến tới một Use case.

# Biểu đồ triển khai (Deployment)



# Q/A