

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



NHẬP MÔN

CÔNG NGHỆ PHẦN MỀM

Giảng viên: TS. Nguyễn Thị Xuân Hương

Email: huongntx@uit.edu.vn

NỘI DUNG MÔN HỌC

- Tổng quan về Công nghệ phần mềm
- **Xác định và mô hình hóa yêu cầu phần mềm**
- Thiết kế phần mềm
- Cài đặt phần mềm
- Kiểm thử và bảo trì
- Đồ án môn học

XÁC ĐỊNH VÀ MÔ HÌNH HÓA YCPM

- Giới thiệu các khái niệm về yêu cầu người dùng và yêu cầu hệ thống
- Miêu tả các yêu cầu chức năng (functional) và yêu cầu phi chức năng (non-functional)
- Cách tổ chức các yêu cầu phần mềm trong mọi tài liệu phần mềm

XÁC ĐỊNH VÀ MÔ HÌNH HÓA YCPM

1. Khái niệm và phân loại yêu cầu

- Các yêu cầu chức năng và phi chức năng
- Yêu cầu người dùng và yêu cầu hệ thống

2. Quy trình kỹ nghệ yêu cầu

3. Đặc tả yêu cầu

- Mô hình hoá hệ thống
- Tài liệu yêu cầu phần mềm

4. Phân tích, thiết kế hướng đối tượng và UML

4. Phân tích, thiết kế hướng đối tượng và UML

1. Phân tích và thiết kế hướng đối tượng
2. UML (Unified Modeling Language)
3. OOAD sử dụng UML

4.1. Phân tích và thiết kế hướng đối tượng (Object Oriented Analysis and Design)

- Quá trình phát triển phần mềm:
 - Thu thập và phân tích yêu cầu
 - Phân tích và thiết kế hệ thống
 - phát triển (coding)
 - kiểm thử
 - triển khai và bảo trì.
- Phân tích, thiết kế hệ thống: giúp hiểu rõ yêu cầu đặt ra, xác định giải pháp, mô tả chi tiết giải pháp. (=> What? và How?).
- Phân tích thiết kế hướng đối tượng: mô tả được tất cả các đối tượng và sự tương tác của chúng sẽ giúp hiểu rõ hệ thống và cài đặt được nó.

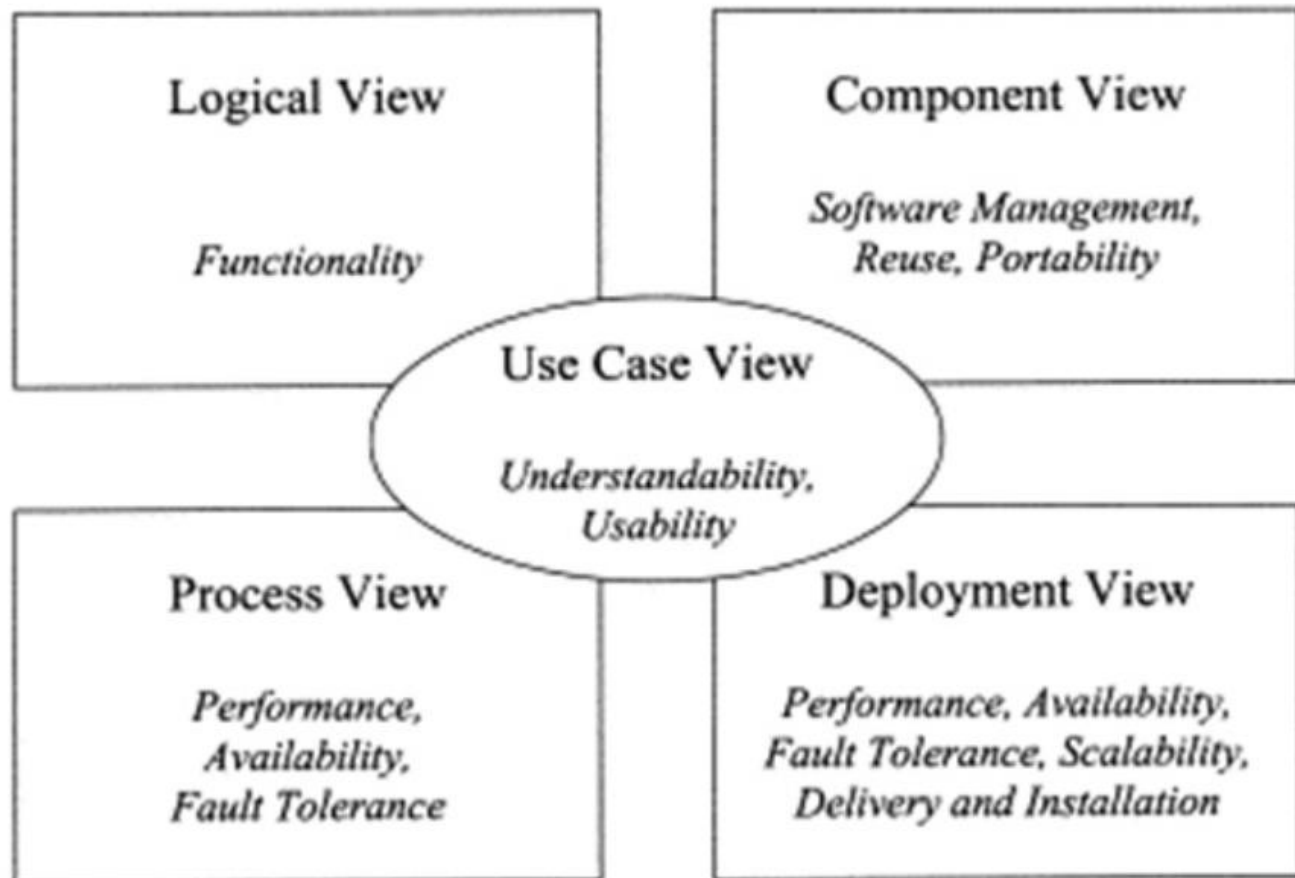
4.2. UML (Unified Modeling Language)

- UML là ngôn ngữ mô hình hóa hợp nhất dùng để biểu diễn hệ thống.
 - Dùng để tạo ra các Biểu đồ nhằm mô tả thiết kế hệ thống.
 - Được sử dụng để các nhóm thiết kế trao đổi với nhau, dùng để thi công hệ thống (phát triển), thuyết phục khách hàng, các nhà đầu tư v.v..
- OOAD sử dụng UML để biểu diễn các thiết kế

4.3. OOAD sử dụng UML

1. View (khung nhìn):

OOAD sử dụng UML có các khung nhìn sau:

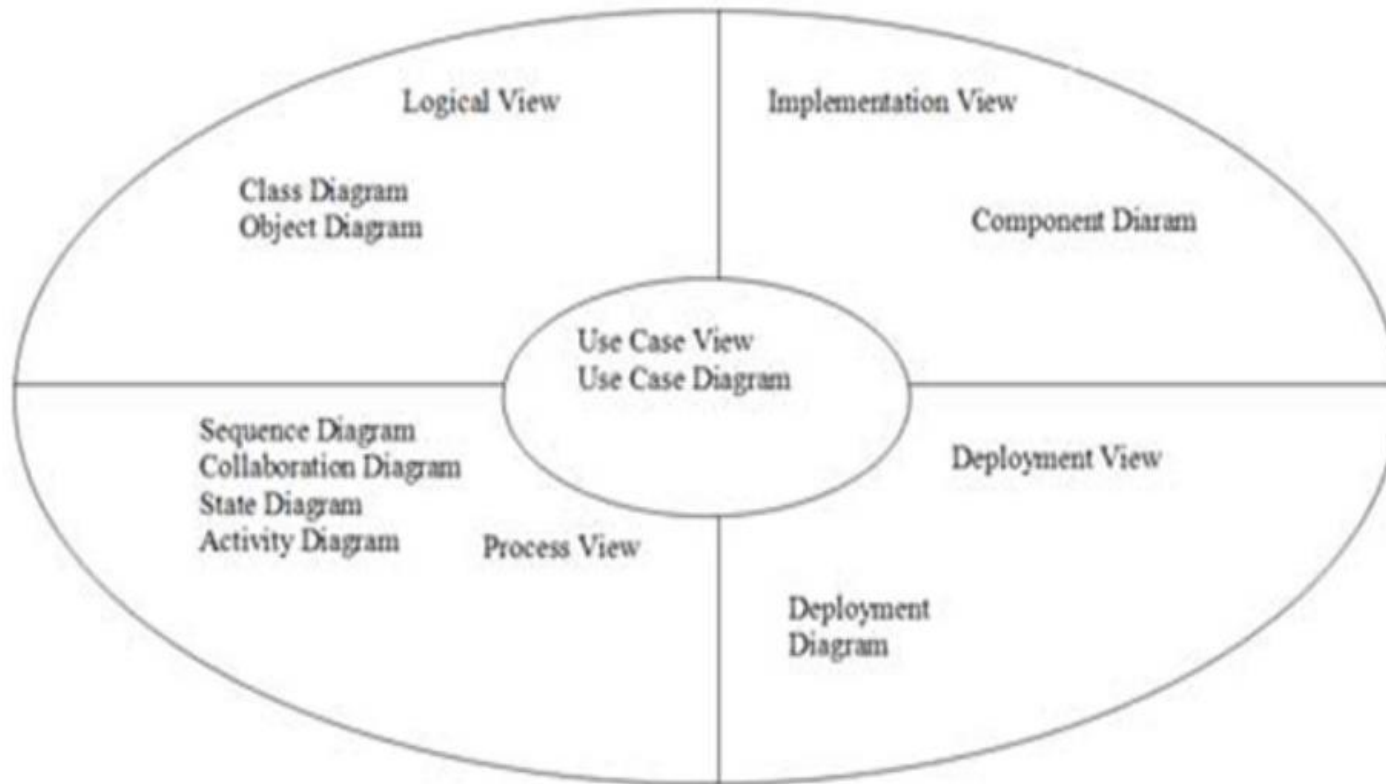


4.3. OOAD sử dụng UML- 4.3.1. View ..

- **Use Case View:** (khung nhìn chức năng) cung cấp khung nhìn về các ca sử dụng giúp hiểu hệ thống có gì? ai dùng và dùng nó như thế nào.
- **Logical View:** (khung nhìn thiết kế) cung cấp khung nhìn về cấu trúc hệ thống, xem nó được tổ chức như thế nào, bên trong nó có gì.
- **Process View:** cung cấp khung nhìn động về hệ thống, xem các thành phần trong hệ thống tương tác với nhau như thế nào.
- **Component View:** Cũng là một khung nhìn về cấu trúc giúp chúng ta hiểu cách phân bổ và sử dụng lại các thành phần trong hệ thống ra sao.
- **Deployment View:** cung cấp khung nhìn về triển khai hệ thống, nó cũng ảnh hưởng lớn đến kiến trúc hệ thống.

4.3. OOAD sử dụng UML- 4.3.2 Diagram (Biểu đồ)

Biểu đồ (Diagram) vẽ được dùng để thể hiện các khung nhìn của hệ thống.



4.3. OOAD sử dụng UML- 4.3.2 Diagram (Biểu đồ)

➤ Use Case Diagram:

- Mô tả về ca sử dụng của hệ thống.
 - Giúp chúng ta biết được ai sử dụng hệ thống, hệ thống có những chức năng gì.
- => Biểu đồ giúp hiểu được yêu cầu của hệ thống cần xây dựng.

➤ Class Diagram:

- mô tả cấu trúc của hệ thống, tức hệ thống được cấu tạo từ những thành phần nào (khía cạnh tĩnh của hệ thống).

➤ Object Diagram:

- Tương tự như Class Diagram nhưng mô tả đến đối tượng thay vì lớp (Class).

➤ Sequence Diagram:

- mô tả sự tương tác của các đối tượng trong hệ thống với nhau được mô tả tuần tự các bước tương tác theo thời gian.

4.3. OOAD sử dụng UML- 4.3.2 Diagram (Biểu đồ)

➤ Collaboration Diagram:

- Tương tự như sequence Diagram nhưng nhấn mạnh về sự tương tác thay vì tuần tự theo thời gian.

➤ State Diagram:

- Mô tả sự thay đổi trạng thái của một đối tượng (dùng để theo dõi các đối tượng có trạng thái thay đổi nhiều trong hệ thống).

➤ Activity Diagram:

- Mô tả các hoạt động của đối tượng, thường được sử dụng để hiểu về nghiệp vụ của hệ thống.

➤ Component Diagram:

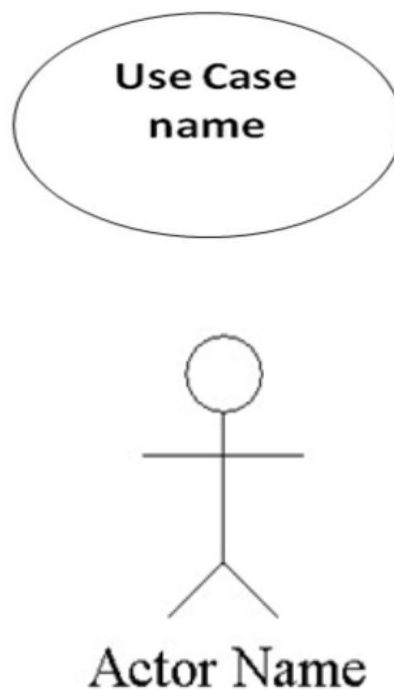
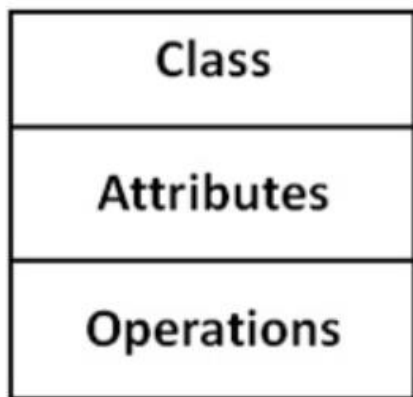
- Mô tả về việc bố trí các thành phần của hệ thống cũng như việc sử dụng các thành phần đó.

➤ Deployment Diagram:

- Mô tả việc triển khai của hệ thống như: kết nối, cài đặt, hiệu năng của hệ thống v.v

4.3. OOAD sử dụng UML- 4.3.3Notations (các ký hiệu)

- Notations là các ký hiệu để vẽ
- Ví dụ về notation.



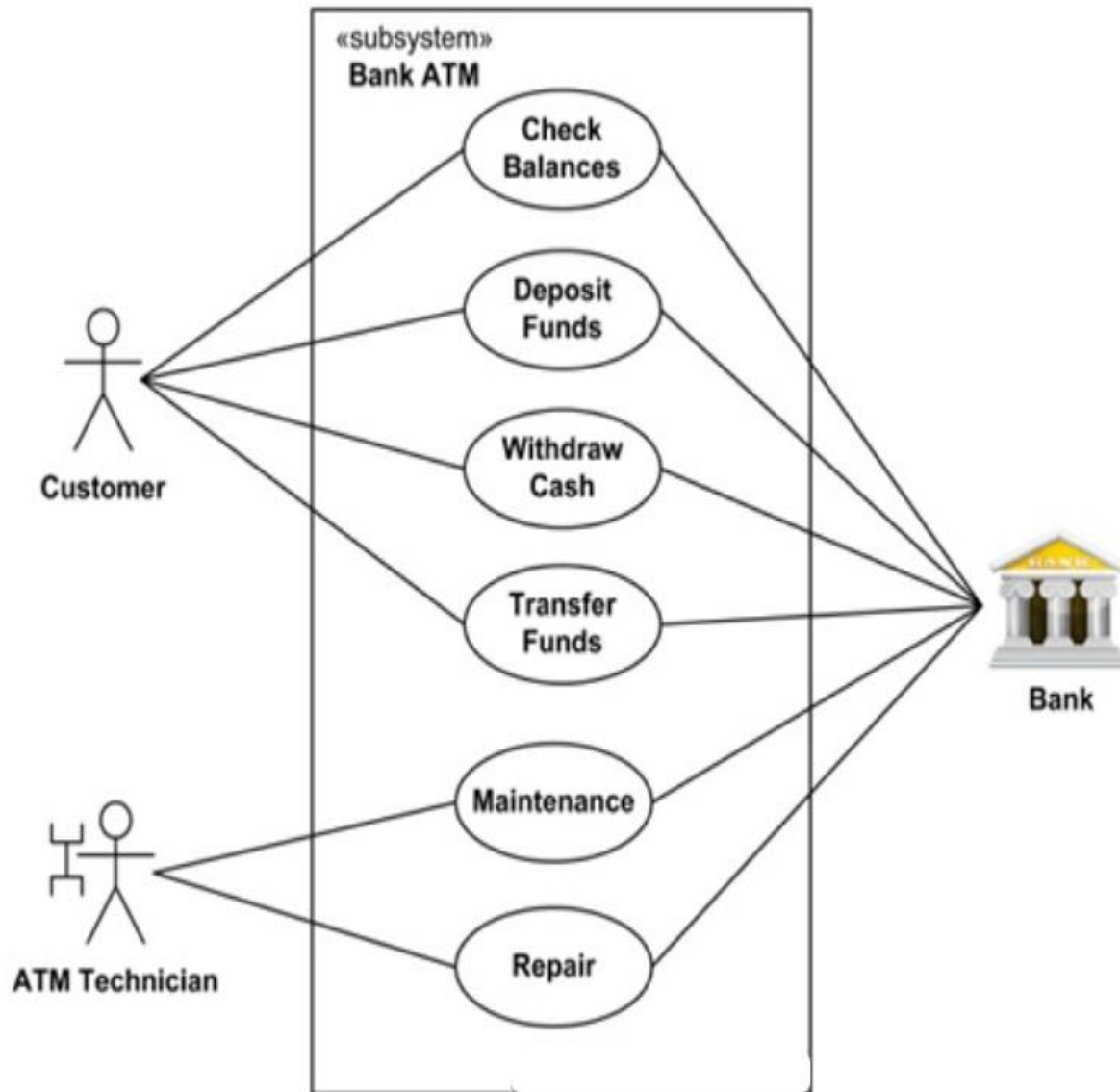
4.3. OOAD sử dụng UML- 4.3.4 Mechanisms (Rules)

- Là các qui tắc để lập nên Biểu đồ, mỗi Biểu đồ có qui tắc riêng và bạn phải nắm được để tạo nên các Biểu đồ thiết kế đúng.

4.4. Biểu đồ Use Case (Use Case Diagram)

4.4.1. Các thành phần trong Biểu đồ Use Case

- Biểu đồ có hai người dùng là **Customer** và **ATM Technician** và một đối tượng sử dụng hệ thống là **Bank**.
- Mô tả các chức năng của hệ thống và người dùng nào dùng chức năng gì.
- Giúp hình dung được xây dựng hệ thống với những chức năng gì? Cho ai dùng.



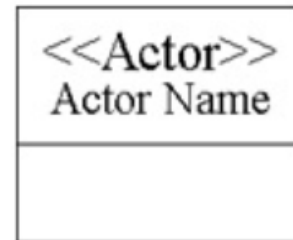
4.4.1. Các thành phần trong Biểu đồ Use Case

1. Actor:

- Chỉ người sử dụng hoặc một đối tượng nào đó bên ngoài tương tác với hệ thống chúng ta đang xem xét.

(Lưu ý, chúng ta hay bỏ quên đối tượng tương tác với hệ thống, ví dụ như Bank ở trên)

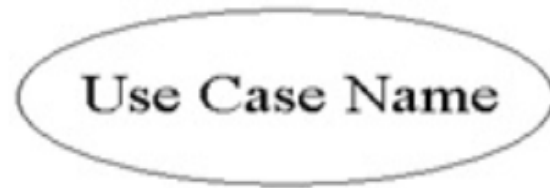
- Biểu diễn Actor:



4.4.1. Các thành phần trong Biểu đồ Use Case

2. Use Case

- là chức năng mà các Actor sẽ sử dụng, được ký hiệu như sau:



- Việc xác định các chức năng mà Actor sử dụng ta sẽ xác định được các Use Case cần có trong hệ thống.

4.4.1. Các thành phần trong Biểu đồ Use Case

3. *Relationship(Quan hệ)*

- Hay còn gọi là connector được sử dụng để kết nối giữa các đối tượng với nhau tạo nên Biểu đồ Use Case.
- Có các kiểu quan hệ cơ bản sau:
 - Association
 - Generalization
 - Include
 - Extend

4.4.1. Các thành phần trong Biểu đồ Use Case

3. *Relationship(Quan hệ)*

- Relationship hay còn gọi là connector được sử dụng để kết nối giữa các đối tượng với nhau tạo nên Biểu đồ Use Case. Có các kiểu quan hệ cơ bản sau:
 - Association
 - Generalization
 - Include
 - Extend

4.4.1. Các thành phần trong Biểu đồ Use Case

3. Relationship (Quan hệ)

- **Quan hệ Association:** thường được dùng để mô tả mối quan hệ giữa Actor và Use Case và giữa các Use Case với nhau.

_____ Association

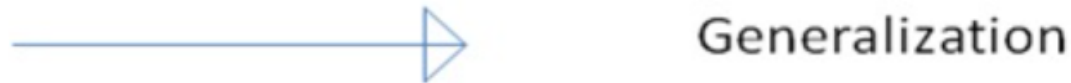
Ví dụ thể hiện Actor User sử dụng Use Case Login



4.4.1. Các thành phần trong Biểu đồ Use Case

3. Relationship (Quan hệ)

- **Quan hệ Generalization:** được sử dụng để thể hiện quan hệ thừa kế giữa các Actor hoặc giữa các Use Case với nhau.



Ví dụ Actor User thừa kế toàn bộ quyền của Actor Guest



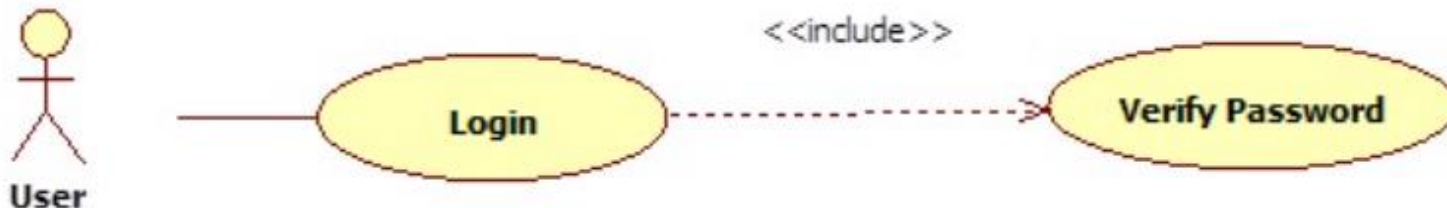
4.4.1. Các thành phần trong Biểu đồ Use Case

3. Relationship (Quan hệ)

- **Quan hệ Include:** quan hệ giữa các Use Case với nhau, mô tả việc một Use Case lớn được chia ra thành các Use Case nhỏ để dễ cài đặt (module hóa) hoặc thể hiện sự dùng lại



Ví dụ về quan hệ Include giữa các Use Case



Use Case “Verify Password” có thể gộp chung vào Use Case Login nhưng ở đây ta tách ra để cho các Use Case khác sử dụng hoặc để module hóa cho dễ hiểu, dễ cài đặt

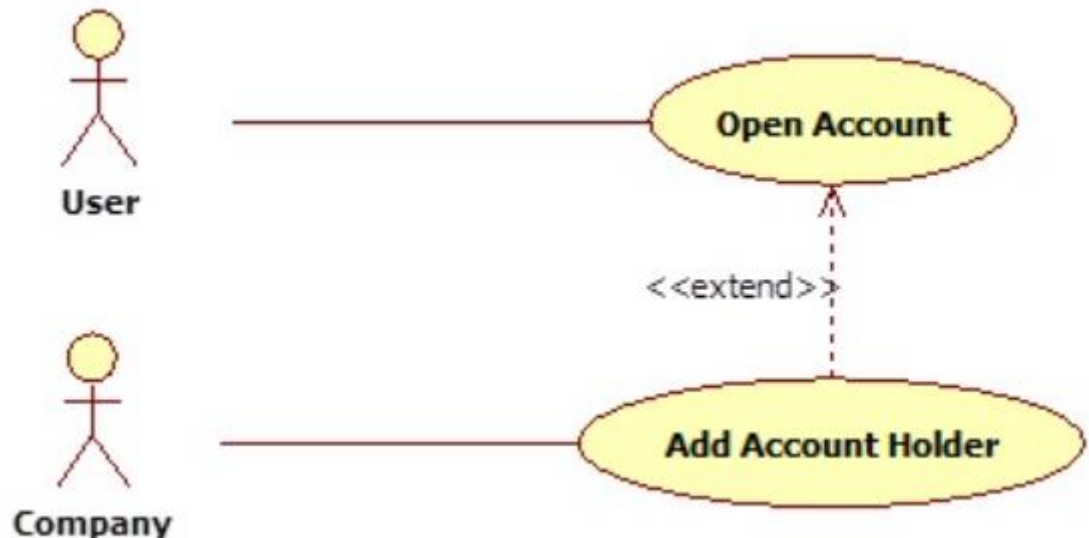
4.4.1. Các thành phần trong Biểu đồ Use Case

3. Relationship (Quan hệ)

- **Quan hệ Extend:** mô tả quan hệ giữa 2 Use Case khi có một Use Case được tạo ra để bổ sung chức năng cho một Use Case có sẵn và được sử dụng trong một điều kiện nhất định nào đó.



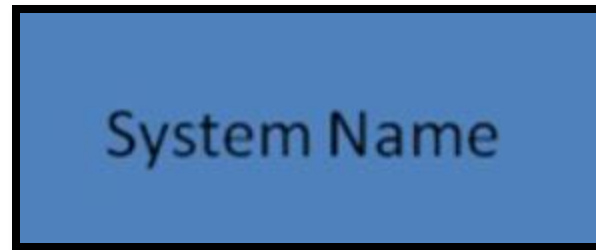
Ví dụ về quan hệ Extend giữa các Use Case



4.4.1. Các thành phần trong Biểu đồ Use Case

4. *System Boundary*

- System Boundary được sử dụng để xác định phạm vi của hệ thống đang thiết kế. Các đối tượng nằm ngoài hệ thống này có tương tác với hệ thống được xem là các Actor.



- System Boundary sẽ giúp dễ hiểu hơn khi chia hệ thống lớn thành các hệ thống con để phân tích, thiết kế.

4.4.2. Các bước xây dựng Use Case Diagram

Bước 1: Tìm các Actor

- Trả lời các câu hỏi sau để xác định Actor cho hệ thống:
 - Ai sử dụng hệ thống này?
 - Hệ thống nào tương tác với hệ thống này?
- Xem xét ví dụ về ATM ở trên:
 - Ai sử dụng hệ thống? -> Customer, ATM Technician
 - Hệ thống nào tương tác với hệ thống này? -> Bank
- Vậy có 03 Actor:
 - Customer
 - ATM Technician
 - Bank

4.4.2. Các bước xây dựng Use Case Diagram

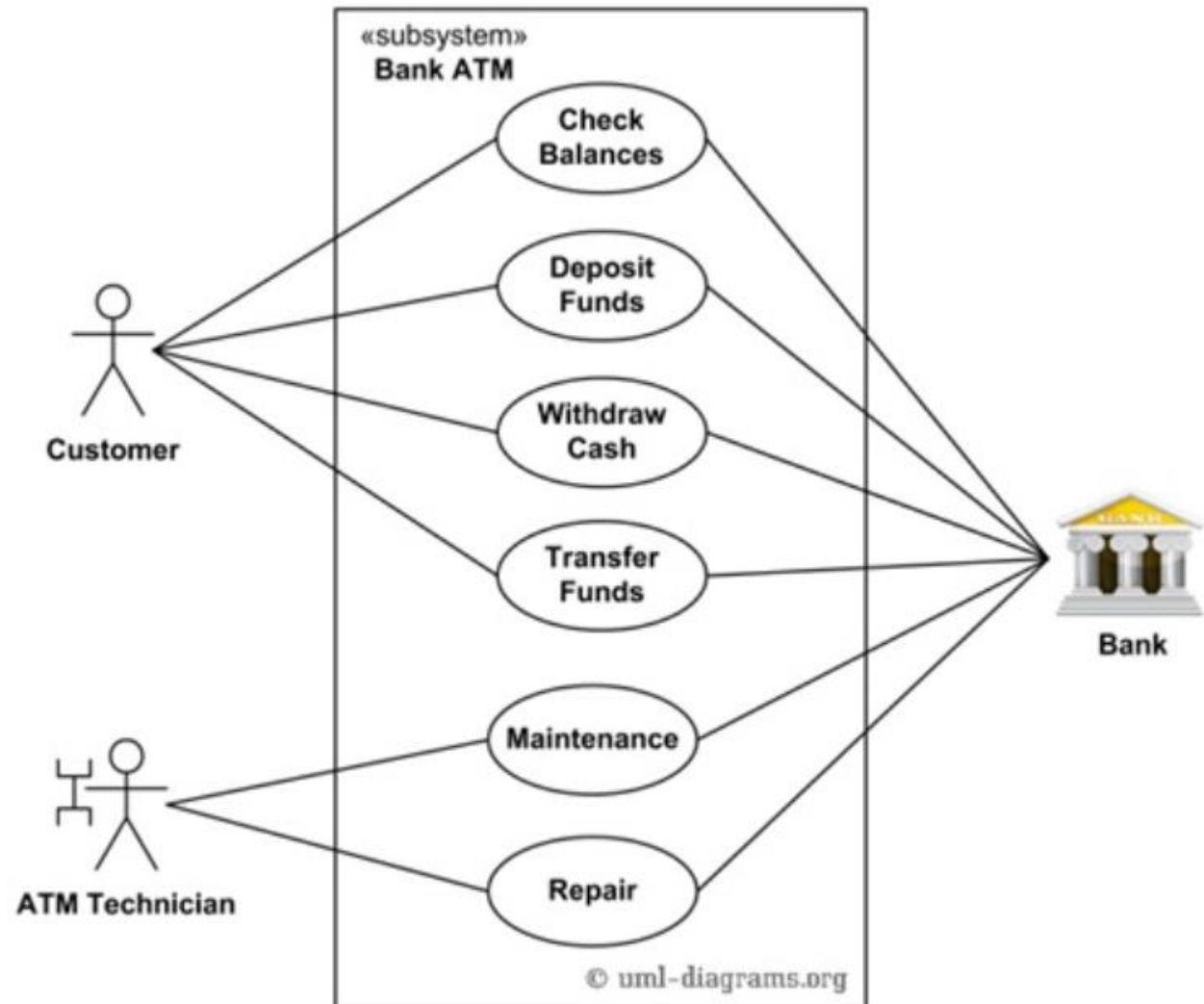
Bước 2: Tìm các Use Case

- Trả lời câu hỏi các Actor sử dụng chức năng gì trong hệ thống
=> xác định được các Use Case cần thiết cho hệ thống.
 - Ví dụ hệ thống ATM:
 - Customer sử dụng các chức năng: Check Balance, Deposit, Withdraw và Transfer
 - ATM technician sử dụng: Maintenance và Repair
 - Bank tương tác với tất cả các chức năng trên.
- => xây dựng hệ thống có các chức năng: **Check Balance, Deposit, Withdraw, Transfer, Maintenance** và **Repair** để đáp ứng được cho người sử dụng và các hệ thống tương tác.

4.4.2. Các bước xây dựng Use Case Diagram

Bước 3: Xác định các quan hệ

- Phân tích và các định các loại quan hệ giữa các Actor và Use Case, giữa các Actor với nhau, giữa các Use Case với nhau sau đó nối chúng lại chúng ta sẽ được Biểu đồ Use Case.



4.4.3. Đặc tả Use Case

- Để hiểu rõ hơn hệ thống, cần phải đặc tả các Use Case: cách vận hành, cách sử dụng

Cách 1: Viết đặc tả cho các Use Case

- Tên Use Case //Account Details
- Mã số Use Case //UCSEC35
- Mô tả tóm tắt//Hiển thị thông tin chi tiết của Account
- Các bước thực hiện//Liệt kê các bước thực hiện
- Điều kiện thoát //Khi người dùng kích nút Close
- Yêu cầu đặc biệt//Ghi rõ nếu có
- Yêu cầu trước khi thực hiện//Phải đăng nhập
- Điều kiện sau khi thực hiện//Ghi rõ những điều kiện nếu có sau khi thực hiện Use Case này

Cách 2: Sử dụng các Biểu đồ để đặc tả: có thể dùng các Biểu đồ như **Activity Diagram, Sequence Diagram** để đặc tả Use Case.

4.4.4. Sử dụng Use Case Diagram

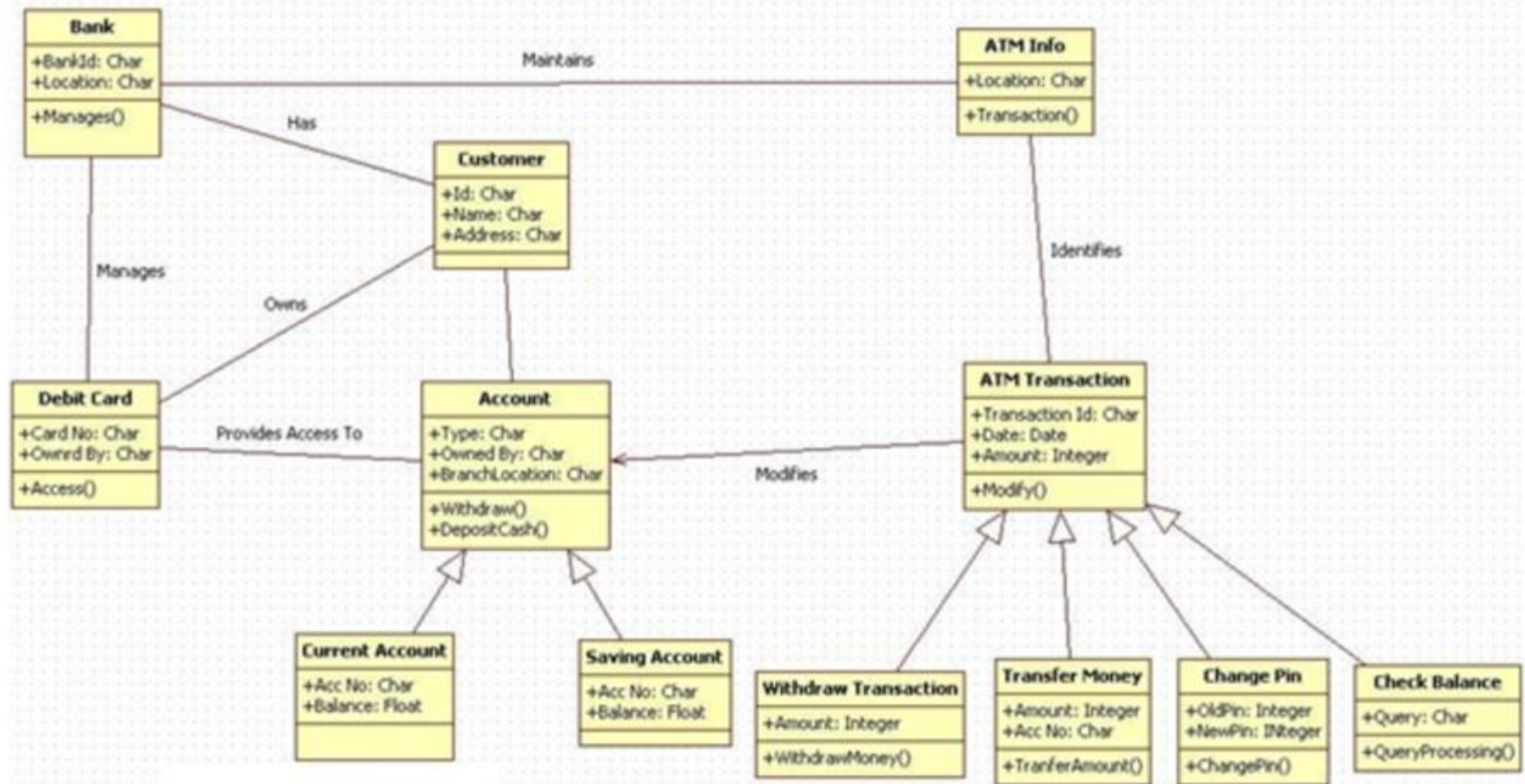
- Use Case Diagram có một vai trò đặc biệt quan trọng trong quá trình phân tích, thiết kế và phát triển hệ thống.
- **Một số ứng dụng tiêu biểu của Use Case Diagram.**
 - Phân tích và hiểu hệ thống
 - Thiết kế hệ thống.
 - Làm cơ sở cho việc phát triển, kiểm tra các Biểu đồ như Class Diagram, Activity Diagram, Sequence Diagram, Component Diagram.
 - Làm cơ sở để giao tiếp với khách hàng, các nhà đầu tư.
 - Giúp cho việc kiểm thử chức năng, kiểm thử chấp nhận.

4.5. Classs Diagarm: Biểu đồ lớp

- Là một trong những Biểu đồ quan trọng nhất của thiết kế phần mềm, cho thấy cấu trúc và quan hệ giữa các thành phần tạo nên phần mềm.
- Trong quá trình xây dựng Class Diagram ta phải quyết định rất nhiều yếu tố về thiết kế nên nó là Biểu đồ khó xây dựng nhất

4.5.1. Các thành phần trong Biểu đồ lớp

Ví dụ về Class Diagram của ATM



4.5.1. Các thành phần trong Biểu đồ lớp

Classes (Các lớp)

- Class là thành phần chính của Biểu đồ Class Diagram.
- Class mô tả về một nhóm đối tượng có cùng tính chất, hành động trong hệ thống.
- Ví dụ mô tả về khách hàng chúng ta dùng lớp “Customer”. Class được mô tả gồm tên Class, thuộc tính và phương thức.

Ký hiệu về Class

Class Name
Attributes
Methods

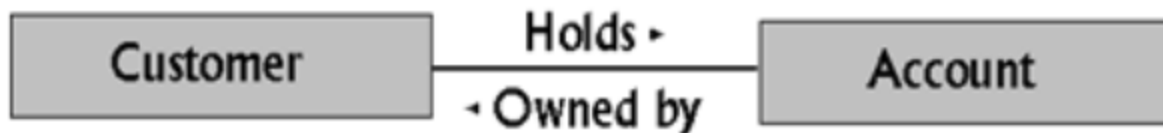
Customer
+CustomerCode -CustomerName -Address -BirthDay
+GetCustomerName(int CCode) +GetCustomerInfo(int CCode)

4.5.2. Relationship (Quan hệ)

- Thể hiện mối quan hệ giữa các Class với nhau.
- Các quan hệ thường sử dụng như sau:
 - Association
 - Aggregation
 - Composition
 - Generalization

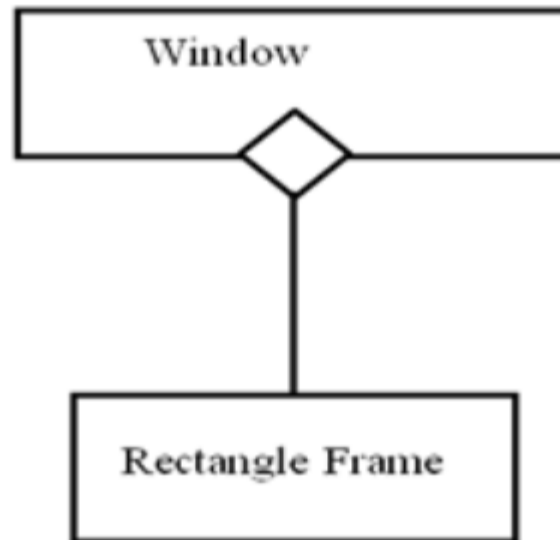
4.5.2. Relationship (Quan hệ)

- **Association:** quan hệ giữa hai lớp với nhau, thể hiện chúng có liên quan với nhau.
 - Association thể hiện qua các quan hệ như “has: có”, “Own: sở hữu” v.v...



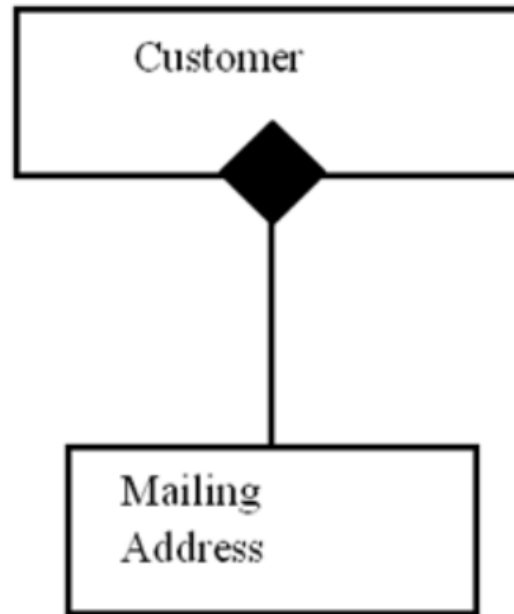
4.5.2. Relationship (Quan hệ)

- **Aggregation:** là một loại của quan hệ Association nhưng mạnh hơn. Nó có thể cùng thời gian sống (cùng sinh ra hoặc cùng chết đi)



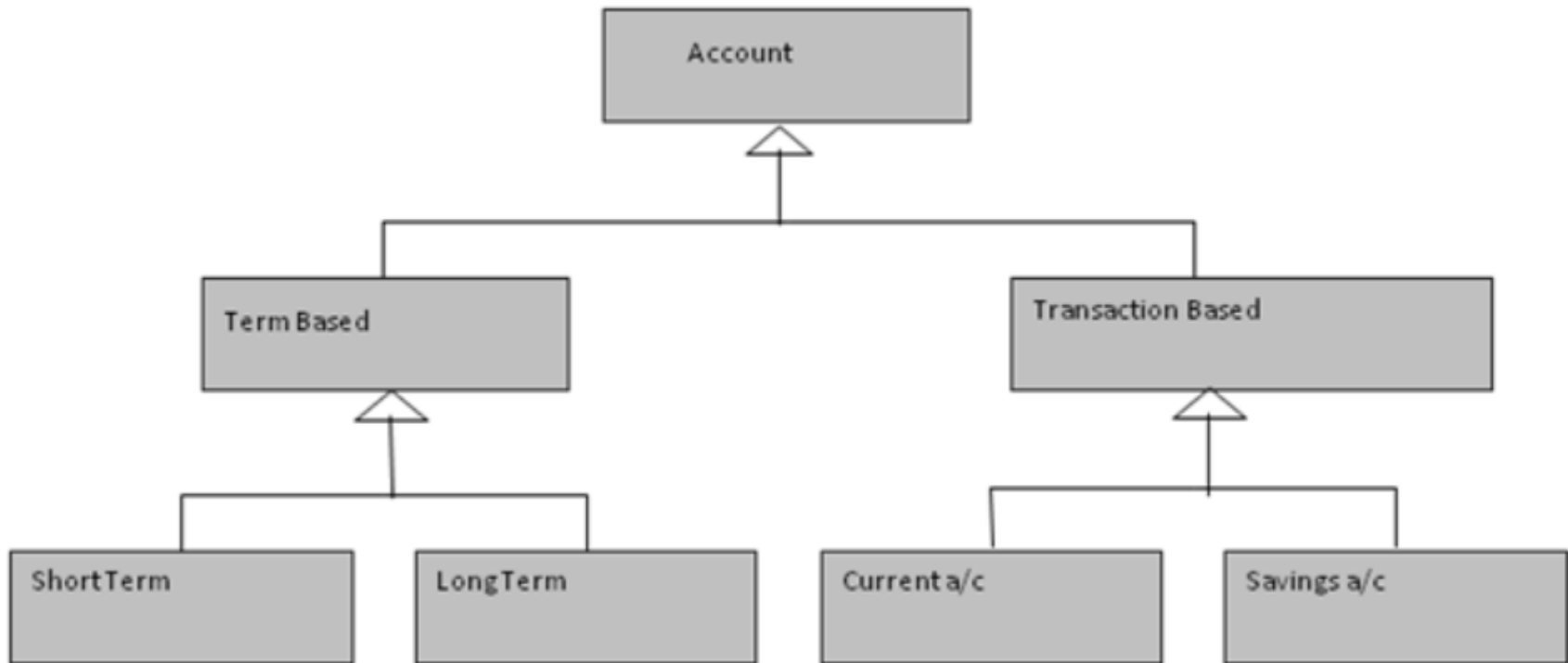
4.5.2. Relationship (Quan hệ)

- **Composition:** là một loại mạnh hơn của Aggregation thể hiện quan hệ class này là một phần của class kia nên dẫn đến cùng tạo ra hoặc cùng chết đi.



4.5.2. Relationship (Quan hệ)

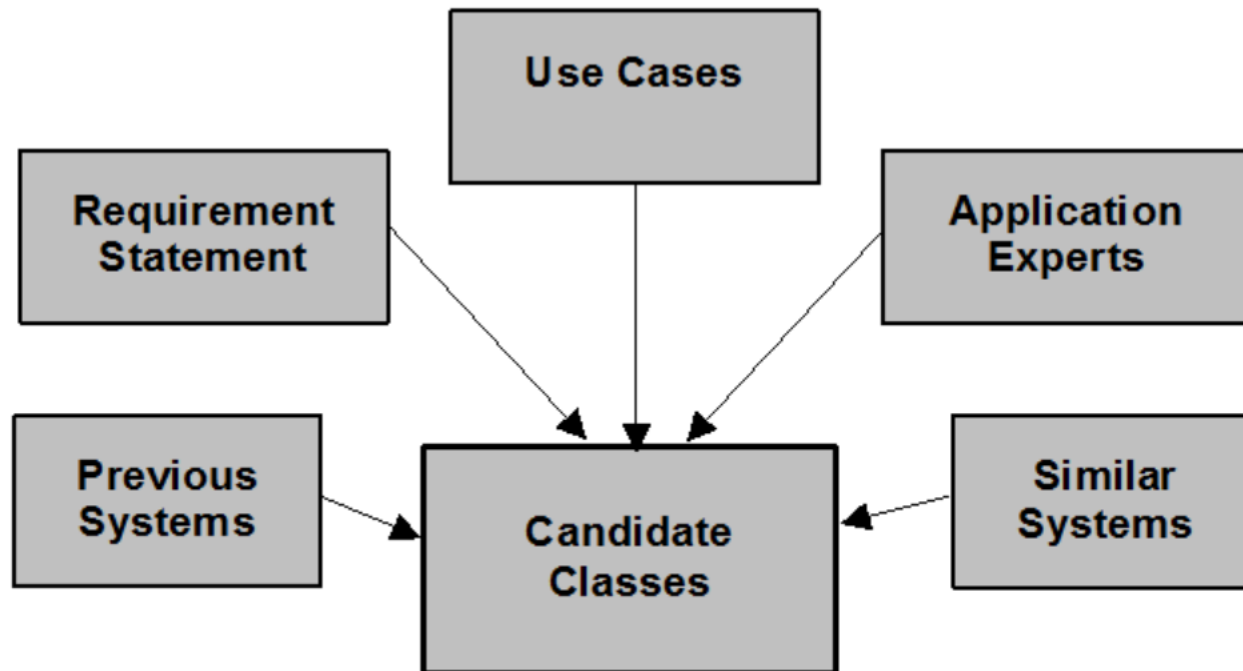
- **Generalization:** là quan hệ thừa kế (được sử dụng rộng rãi trong lập trình hướng đối tượng).



4.5.3. Cách xây dựng Biểu đồ lớp

➤ Bước 1: Tìm các Classes dự kiến

Entity Classes(các lớp thực thể) là các thực thể có thật và hoạt động trong hệ thống, dựa vào các nguồn sau:



4.5.3. Cách xây dựng Biểu đồ lớp

Bước 1: Tìm các Classes dự kiến

- **Requirement statement:** Các yêu cầu, phân tích các danh từ trong các yêu cầu để tìm ra các thực thể.
- **Use Cases:** Phân tích các Use Case sẽ cung cấp thêm các Classes dự kiến.
- **Previous và Similar System:** có thể sẽ cung cấp thêm các lớp dự kiến.
- **Application Experts:** ý kiến của các chuyên gia ứng dụng trong xác định các classes

4.5.3. Cách xây dựng Biểu đồ lớp

Bước 1: Tìm các Classes dự kiến

Ví dụ ATM, các đối tượng là Entity Class như sau:

- **Customers:** khách hàng giao dịch là một thực thể có thật và quản lý trong hệ thống.
- **Accounts:** Tài khoản của khách hàng cũng là một đối tượng thực tế.
- **ATM Cards:** Thẻ dùng để truy cập ATM cũng được quản lý trong hệ thống.
- **ATM Transactions:** Các giao dịch được lưu giữ lại, nó cũng là một đối tượng có thật.
- **Banks:** Thông tin ngân hàng đang giao dịch, nếu có nhiều nhà Bank tham gia vào hệ thống thì phải quản lý nó. Khi đó, Bank trở thành đối tượng phải quản lý.
- **ATM:** Thông tin ATM sử dụng giao dịch, được quản lý tương tự như Banks.

4.5.3. Cách xây dựng Biểu đồ lớp

Bước 2: Tìm các thuộc tính và phương thức cho lớp

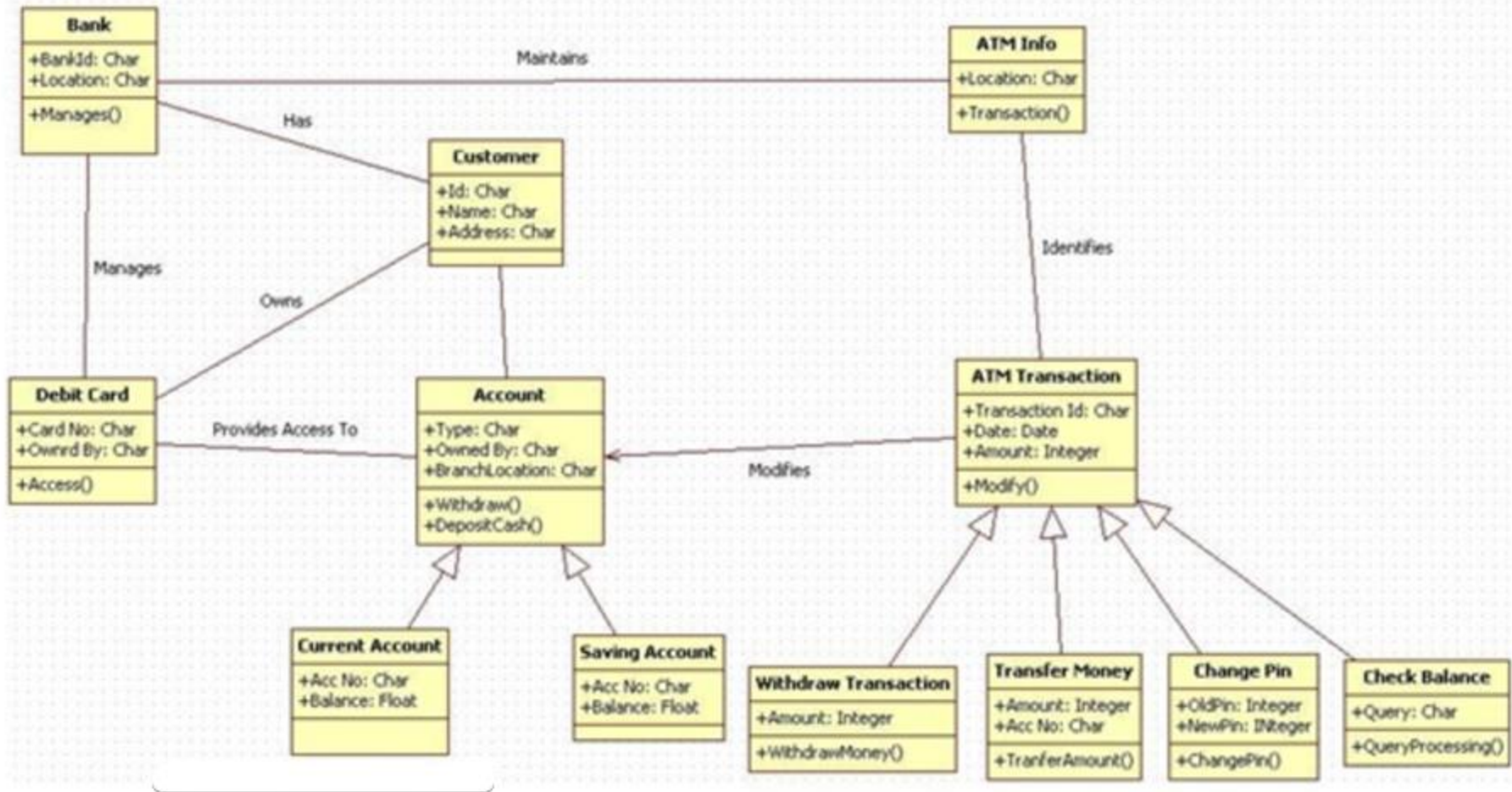
- **Tìm thuộc tính:** phân tích thông tin từ các form mẫu có sẵn để tìm ra thuộc tính cho các đối tượng của lớp.
Ví dụ các thuộc tính của lớp Customer thể hiện trên Form đăng ký thông tin khách hàng.
- **Tìm phương thức:** phương thức là các hoạt động mà các đối tượng của lớp này có thể thực hiện. Có thể bổ sung phương thức đầy đủ cho các lớp khi phân tích Sequence Diagram sau này.

4.5.3. Cách xây dựng Biểu đồ lớp

Bước 3: Xây dựng các quan hệ giữa các lớp và phát hiện các lớp phát sinh

- Phân tích các quan hệ giữa các lớp và định nghĩa các lớp phát sinh do các quan hệ sinh ra.
- Phân tích các thực thể ở trên (hệ thống ATM) và nhận thấy.
 - Lớp *Accounts* có thể chia thành nhiều loại tài khoản như *Current Accounts* và *Saving Accounts* và có quan hệ thừa kế với nhau.
 - Lớp *ATM Transactions* có thể chia thành nhiều loại giao dịch như *Deposit*, *Withdraw*, *Transfer* v.v.. và chúng cũng có quan hệ thừa kế với nhau.
 - Tách chúng ta và vẽ chúng lên Biểu đồ ta sẽ có Class Diagram cho hệ thống ATM như sau:

4.5.3. Cách xây dựng Biểu đồ lớp



4.5.4. Đặc tả Class

Mô tả:

- Các thuộc tính: Tên, kiểu dữ liệu, kích thước
- Các phương thức:
 - + Tên
 - + Mô tả
 - + Tham số đầu vào: Tên, kiểu dữ liệu, kích thước
 - + Kết quả đầu ra: Tên, kiểu dữ liệu, kích thước
 - + Luồng xử lý
 - + Điều kiện bắt đầu
 - + Điều kiện kết thúc

4.5.5. Sử dụng Biểu đồ lớp

- Hiểu cấu trúc của hệ thống
- Thiết kế hệ thống
- Sử dụng để phân tích chi tiết các chức năng (Sequence Diagram, State Diagram v.v...)
- Sử dụng để cài đặt (coding)

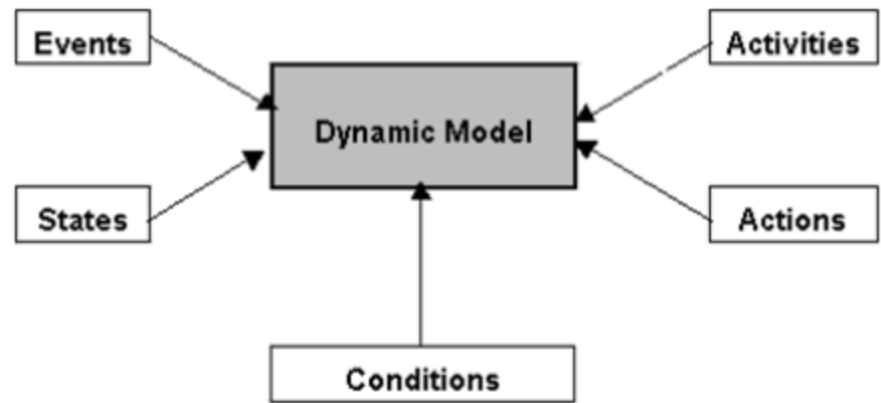
4.6. Activity Diagram: Biểu đồ hoạt động

- Phân tích khía cạnh hoạt động trong hệ thống
- Có thể được mô tả theo 2 mô hình
 - Static Model: mô tả cấu trúc của hệ thống bao gồm các Biểu đồ Class Diagram, Object Diagram, Component Diagram và Deployment Diagram.
 - Dynamic Model: mô tả các hoạt động bên trong hệ thống bao gồm các Biểu đồ Activity Diagram, State Diagram, Sequence Diagram, Collaboration Diagram.
- Sử dụng hai mô hình động phổ biến: Activity Diagram, Sequence Diagram

4.6. Activity Diagram: Biểu đồ hoạt động

➤ Các thành phần cơ bản của Dynamic Model

- Event: là sự kiện, mô tả một hoạt động bên ngoài tác động vào đối tượng và được đối tượng nhận biết và có phản ứng lại.

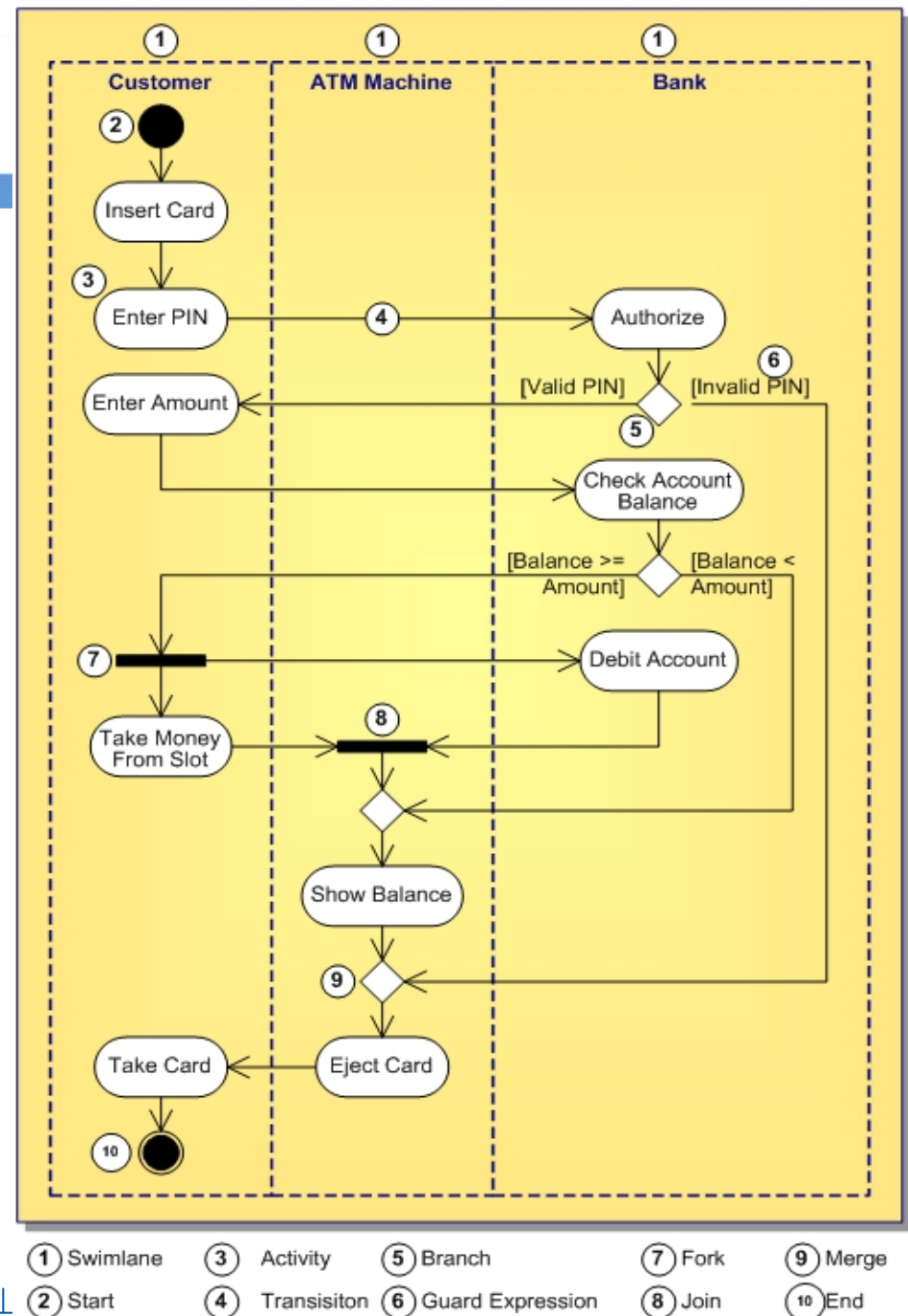


- Activity: mô tả một hoạt động trong hệ thống (có thể do một hoặc nhiều đối tượng thực hiện.)
- State: là trạng thái của một đối tượng trong hệ thống, được mô tả bằng giá trị của một hoặc nhiều thuộc tính.
- Action: chỉ hành động của đối tượng.
- Condition: mô tả một điều kiện.

4.6.2. Activity Diagram

- Là Biểu đồ tập trung vào mô tả các hoạt động, luồng xử lý bên trong hệ thống.
- Có thể được sử dụng để mô tả các quy trình nghiệp vụ trong hệ thống, các luồng của một chức năng hoặc các hoạt động của một đối tượng.

Ví dụ về Activity Diagram của hoạt động rút tiền từ ATM



4.6.2. Activity Diagram

Các ký hiệu:

1. **Swimlane:** dùng để xác định đối tượng nào tham gia hoạt động nào trong một qui trình.

Ví dụ ở trên **Customer** thì ***Insert Card*** còn **ATM Machine** thì ***Show Balance***.



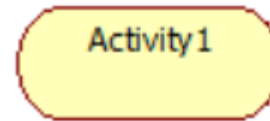
2. Nút Start và End



4.6.2. Activity Diagram

Các ký hiệu:

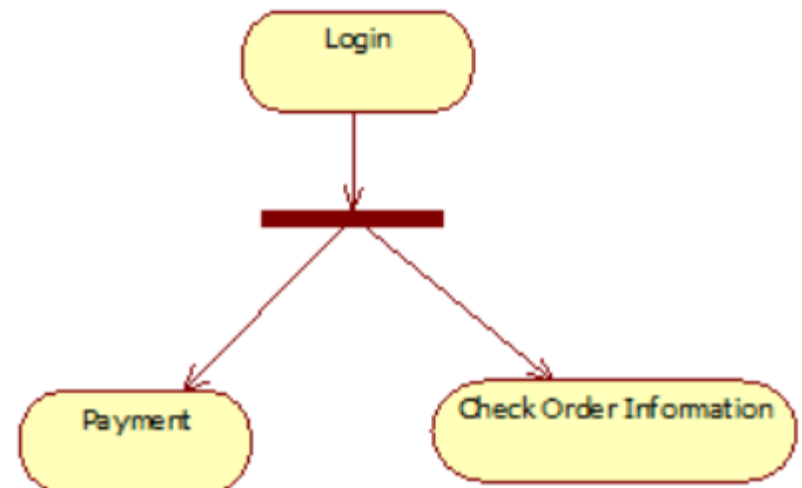
3. Activity: mô tả một hoạt động trong hệ thống. Các hoạt động này do các đối tượng thực hiện.



4. Branch: thể hiện rẽ nhánh trong mệnh đề điều kiện



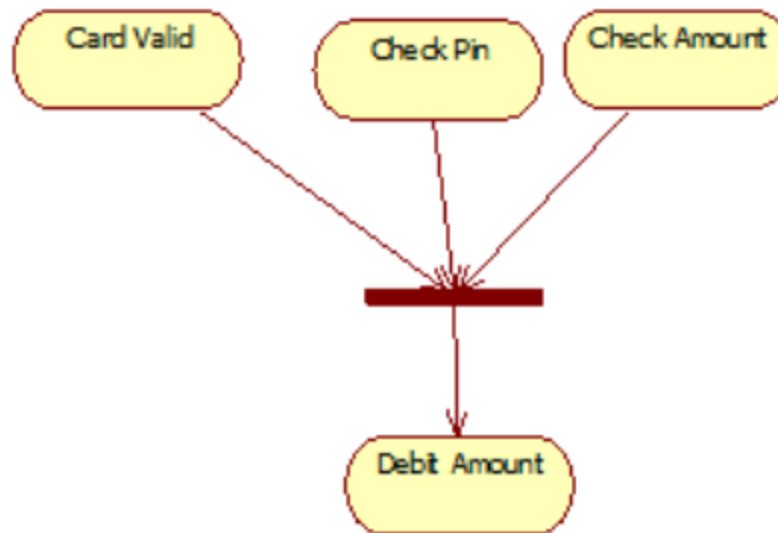
5. Fork: thể hiện cho trường hợp thực hiện xong một hoạt động rồi sẽ rẽ nhánh thực hiện nhiều hoạt động tiếp theo.



4.6.2. Activity Diagram

Các ký hiệu:

6. Join: thể hiện trường hợp phải thực hiện hai hay nhiều hành động trước rồi mới thực hiện hành động tiếp theo.



4.6.3. Xây dựng Activity Diagram

Bước 1: Xác định các nghiệp vụ cần mô tả

Xem xét Biểu đồ Use Case để xác định nghiệp vụ nào cần mô tả.

Bước 2: Xác định trạng thái đầu tiên và trạng thái kết thúc

Bước 3: Xác định các hoạt động tiếp theo

- Xuất phát từ điểm bắt đầu, phân tích để xác định các hoạt động tiếp theo cho đến khi gặp điểm kết thúc để hoàn tất Biểu đồ này.
- Có thể hỏi chuyên gia, học hệ thống tương tự, hỏi khách hàng để nắm rõ về quy trình của hệ thống.

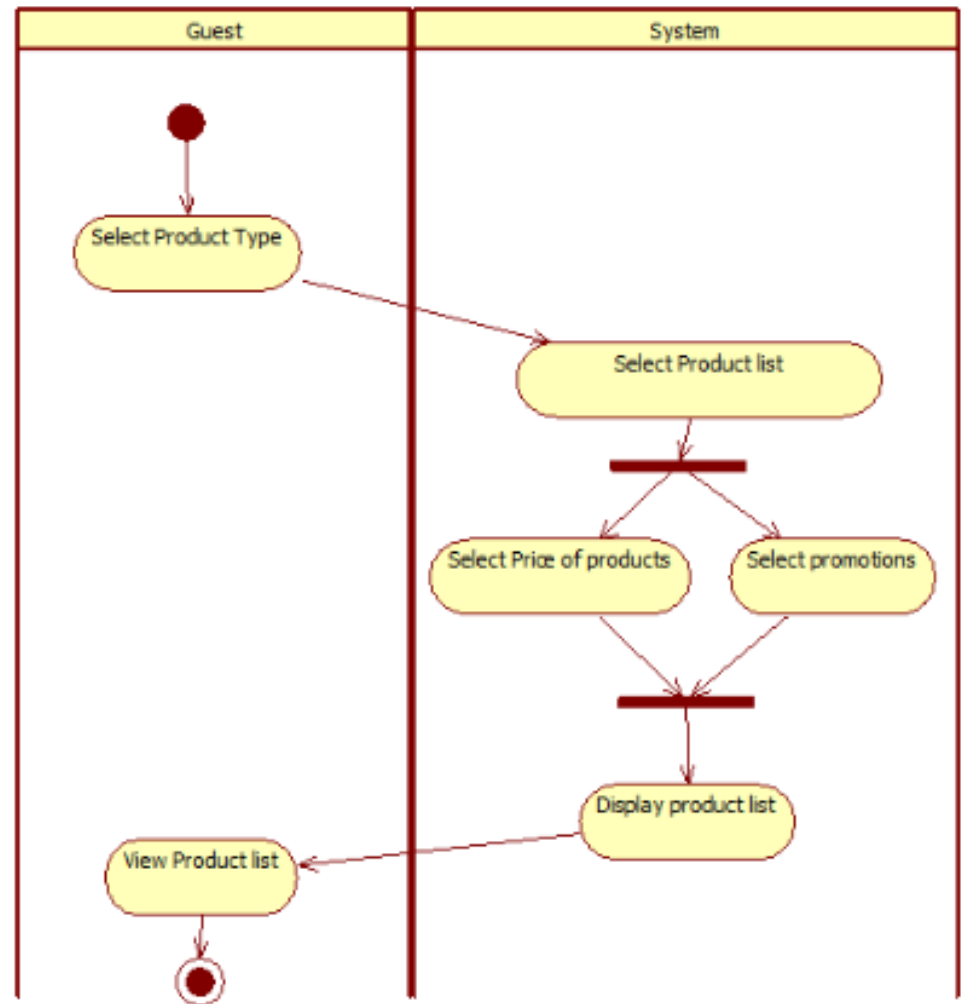
4.6.4 Sử dụng Biểu đồ hoạt động

➤ Sử dụng Biểu đồ hoạt động

- Phân tích nghiệp vụ để hiểu rõ hệ thống
- Phân tích Use Case
- Cung cấp thông tin để thiết kế Biểu đồ Sequence Diagram

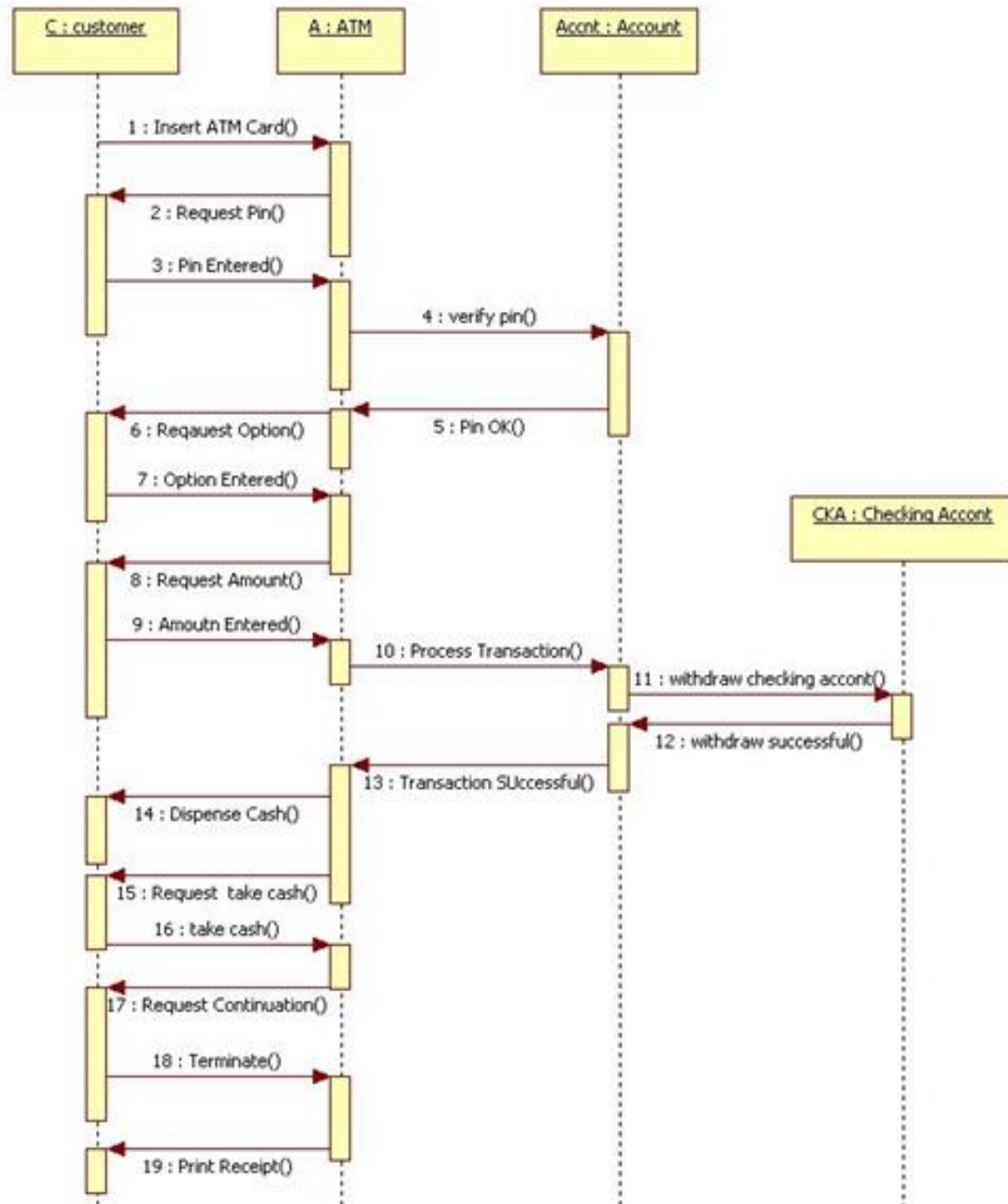
4.6.4 Sử dụng Biểu đồ hoạt động

VD: Biểu đồ hoạt động cho chức năng xem sản phẩm theo chủng loại trong eCommerce



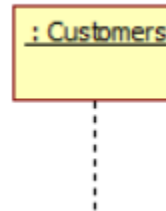
4.7. Sequence Diagram

- Dùng để thiết kế chi tiết chức năng cho hệ thống.
- Mô tả sự tương tác của các đối tượng để tạo nên các chức năng của hệ thống.
- Mô tả sự tương tác theo thời gian nên rất phù hợp với việc sử dụng để thiết kế và cài đặt chức năng cho hệ thống phần mềm.
- Ví dụ Sequence Diagram cho hoạt động rút tiền ở ATM



4.7.1. Các thành phần của Sequence Diagram

- **Objects:** mô tả một đối tượng trong hệ thống. Để phân biệt với Class, Object có dấu “:” phía trước tên của nó.



- Đường gạch chấm bên dưới đối tượng thể hiện thời gian sống của đối tượng.

4.7.1. Các thành phần của Sequence Diagram

- **Stimulus (message):** thể hiện thông điệp từ một đối tượng này tương tác với một đối tượng khác.



- **Axes:** trục tọa độ, trục ngang thể hiện các đối tượng, trục đứng thể hiện thời gian.
- => các đối tượng tương tác với nhau theo tuần tự các bước để hình thành nên chức năng của hệ thống.

4.7.2. Xây dựng Sequence Diagram

Bước 1: Xác định chức năng cần thiết kế (dựa vào Use Case Diagram để xác định xem chức năng nào cần thiết kế).

Bước 2: Dựa vào Activity Diagram để xác định các bước thực hiện theo nghiệp vụ.

Bước 3: Đối chiếu với Class Diagram để xác định lớp trong hệ thống tham gia vào nghiệp vụ.

Bước 4: Vẽ Sequence Diagram

Bước 5: Cập nhật lại Biểu đồ Class Diagram

4.7.3. Ứng dụng Sequence Diagram

- Thiết kế các chức năng
- Kiểm chứng và bổ sung method cho các Class
- Sử dụng trong việc coding các chức năng

4.7.4. Ví dụ: chức năng “**Xem sản phẩm theo chủng loại**” trong hệ thống E-Commerce

- **Bước 1:** Xác định Use Case cần thiết kế: “**Xem sản phẩm theo chủng loại**”
- **Bước 2:** dựa vào *Activity Diagram* cho Use Case này ta xác định các bước sau:
 - Người dùng chọn loại sản phẩm
 - Hệ thống sẽ lọc lấy loại sản phẩm tương ứng, sau đó lấy giá, lấy khuyến mãi và hiển thị lên màn hình.
 - Người dùng xem sản phẩm

4.7.4. Ví dụ: chức năng “**Xem sản phẩm theo chủng loại**” trong hệ thống E-Commerce

➤ **Bước 3:** *Đối chiếu với Class Diagram chúng ta xác định các đối tượng thực hiện như sau:*

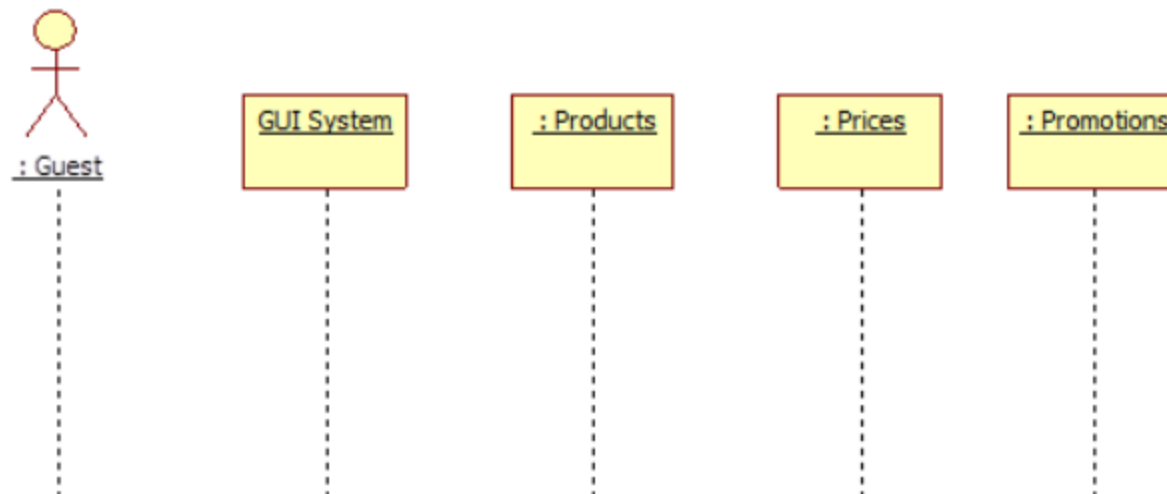
- Người dùng: chọn loại sản phẩm qua giao diện
- Giao diện: sẽ lấy danh sách sản phẩm tương ứng từ Products
- Giao diện: lấy giá của từng sản phẩm từ Class Prices và Promotion Amount từ lớp Promotions
- Giao diện: tổng hợp danh sách và hiển thị
- Người dùng: Xem sản phẩm

4.7.4. Ví dụ: chức năng “**Xem sản phẩm theo chủng loại**” trong hệ thống E-Commerce

➤ **Bước 4: Vẽ sequence Diagram**

Xác định các lớp tham gia vào hệ thống gồm: người dùng (Guest), Giao diện (GUI System), Sản phẩm (Products), Giá (Prices), Khuyến mãi (Promotions). Trong đó GUI System để sử dụng chung cho giao diện, có thể sử dụng cụ thể trang Web nào đã có thiết kế chi tiết của giao diện.

- **Xác định các đối tượng tham gia vào Biểu đồ**

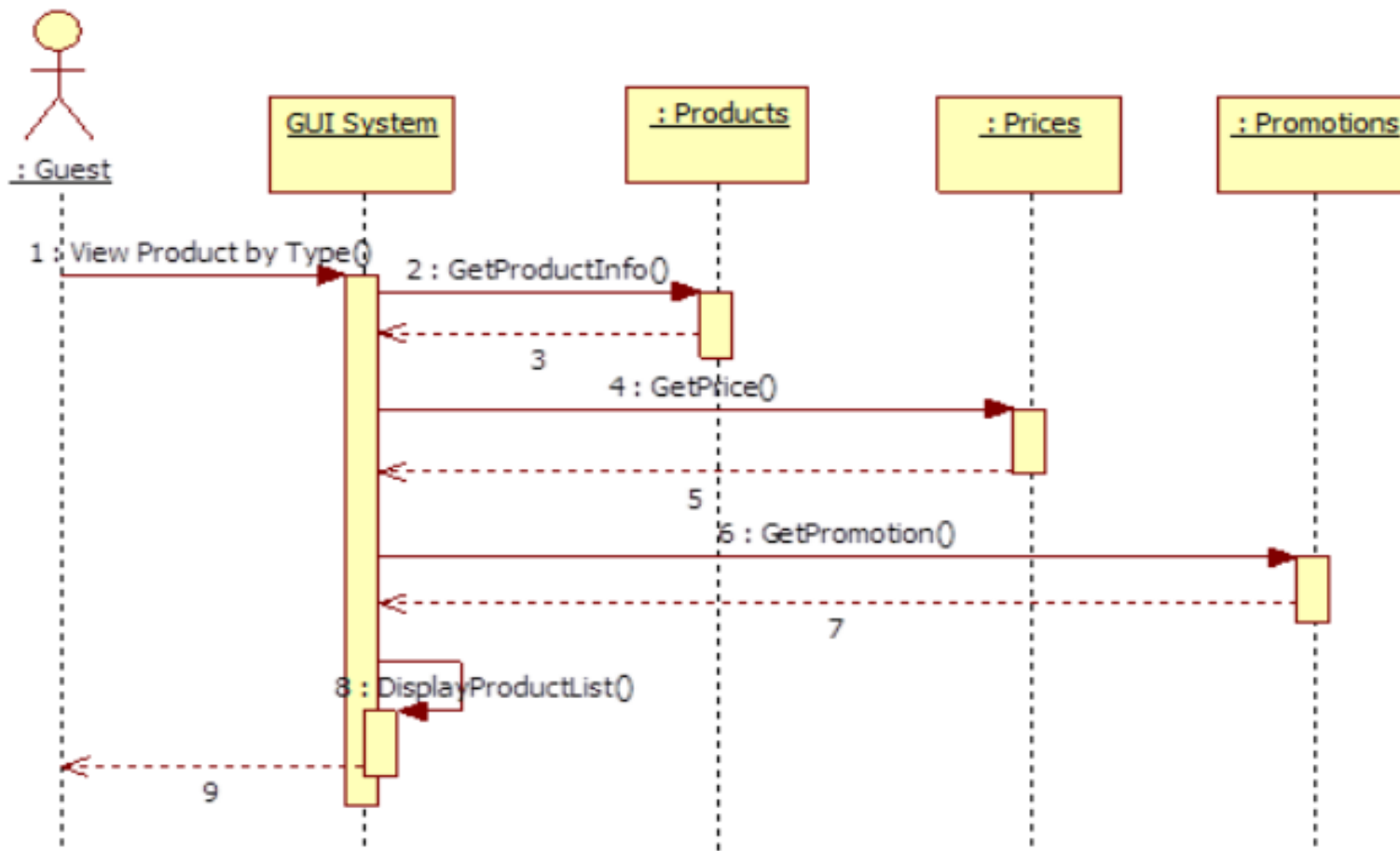


4.7.4. Ví dụ: chức năng “**Xem sản phẩm theo chủng loại**” trong hệ thống E-Commerce

- Các bước thực hiện của Use Case này như sau:
 - Guest gửi yêu cầu xem sản phẩm lên giao diện kèm theo chủng loại
 - GUI system: gửi yêu cầu lấy danh sách các sản phẩm tương ứng với chủng loại cho lớp sản phẩm và nhận lại danh sách.
 - GUI system: gửi yêu cầu lấy Giá cho từng sản phẩm từ Prices
 - GUI system: gửi yêu cầu lấy khuyến mãi cho từng sản phẩm từ Promotions và nhận lại kết quả
 - GUI system: ghép lại danh sách và hiển thị lên browser và trả về cho Guest

4.7.4. Ví dụ: chức năng “Xem sản phẩm theo chủng loại” trong hệ thống E-Commerce

Biểu đồ như sau:



4.7.4. Ví dụ: chức năng “Xem sản phẩm theo chủng loại” trong hệ thống E-Commerce

➤ *Bước 5: Kiểm tra và cập nhật Biểu đồ lớp*

Để thực hiện được Biểu đồ trên chúng ta cần bổ sung các phương thức cho các lớp như sau:

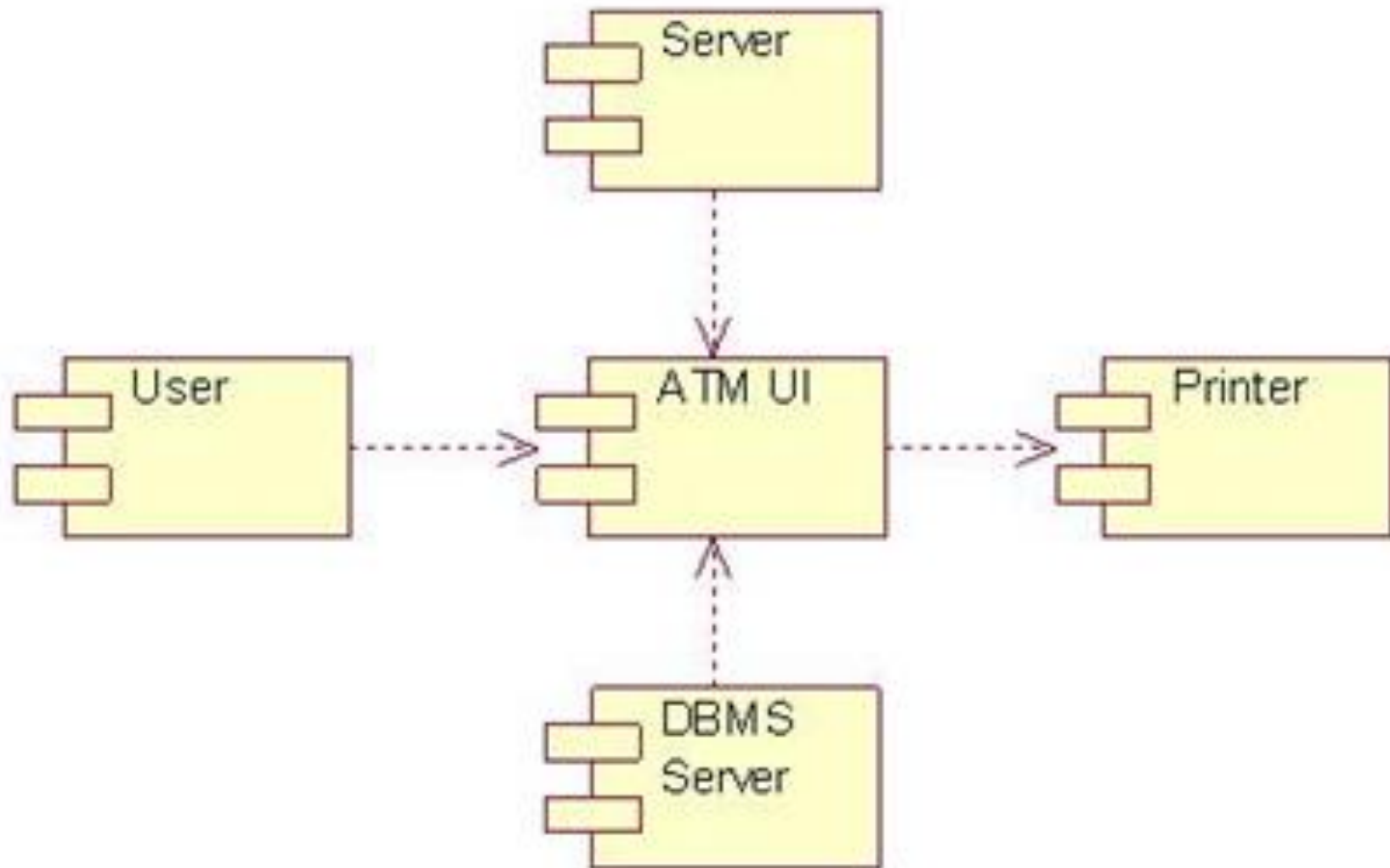
- **Products class:** bổ sung phương thức **GetProductInfo(Product Type)**: trả về thông tin sản phẩm có loại được truyền vào. Việc này các đối tượng của lớp Products thực hiện được vì đã có thuộc tính
- **Prices:** bổ sung phương thức **GetPrice(ProductID)**: UnitPrice. Sau khi lấy được ProductID từ Products, GUI gọi phương thức này để lấy giá của sản phẩm từ lớp giá.
- **Promotions:** tương tự bổ sung phương thức **GetPromotion(ProductID)**.
- **GUI System(View Product Page):** bổ sung phương thức **DisplayProductList(List of product)** để hiển thị danh sách lên sản phẩm. Ngoài ra, cần có thêm phương thức **ViewProductbyType(ProductType)** để mô tả chính hoạt động này khi người dùng kích chọn.

4.8. Component Diagram: Biểu đồ thành phần

- Khi thiết kế các hệ thống phức tạp chúng ta nên chia chúng ra thành nhiều hệ thống con(subsystem) để dễ thiết kế.
- Mỗi hệ thống con sau khi xây dựng có thể đóng gói thành một thành phần phần mềm được triển khai độc lập.
- Biểu đồ Component Diagram sẽ giúp thể hiện cách chia hệ thống ra nhiều thành phần và quan hệ của chúng.
- Component Diagram là Biểu đồ cho biết cấu trúc của hệ thống theo thành phần phần mềm.

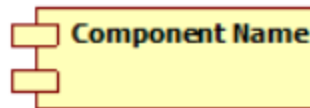
4.8. Component Diagram: Biểu đồ thành phần

- Ví dụ biểu đồ thành phần cho ứng dụng máy ATM



4.8.1. Các thành phần của Component Diagram

- **Component:** là một thành phần phần mềm được đóng gói độc lập, có thể được triển khai độc lập trên hệ thống và có khả năng tương tác với các thành phần khác khi thực hiện các chức năng của hệ thống.



- **Component Dependence:** thể hiện quan hệ giữa các thành phần với nhau. Các thành phần phần mềm luôn cần sử dụng một số chức năng ở các thành phần khác trong hệ thống nên quan hệ Dependence được sử dụng thường xuyên.



4.8.2. Xây dựng Component Diagram

- Để xây dựng Biểu đồ Component chúng ta thực hiện các bước sau:

Bước 1: Chia hệ thống thành những SubSystem

Bước 2: Xác định các thành phần và vẽ

4.8.3. Ví dụ về Component Diagram trong hệ thống eCommerce:

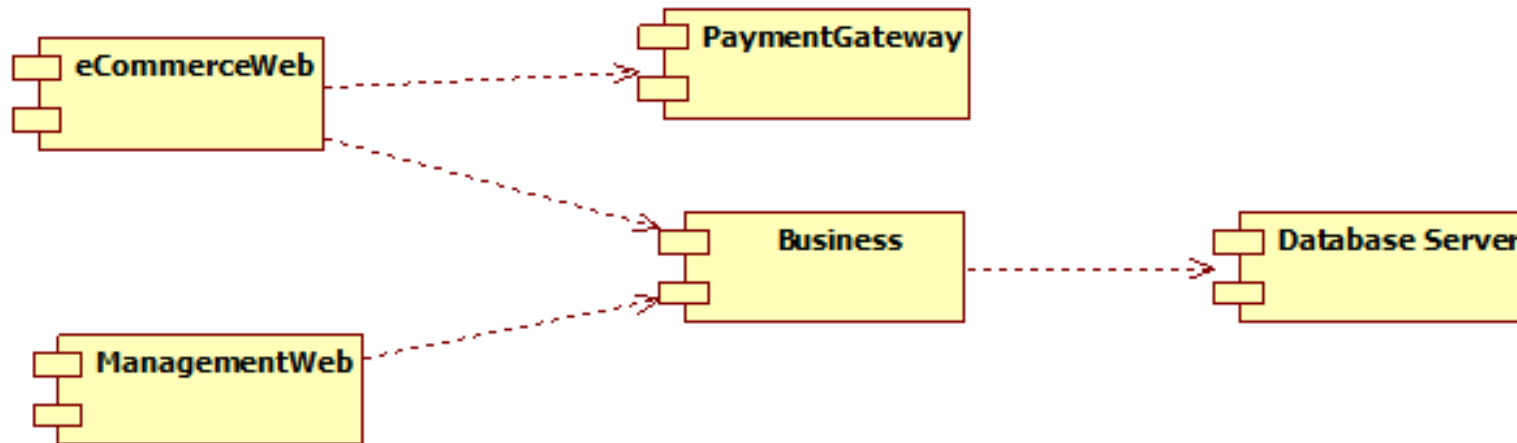
Bước 1: Chia hệ thống thành các SubSystem như sau:

- Chia phần Website phục vụ cho đối tượng bên ngoài công ty là Guest và Customer ra một gói riêng để dễ triển khai và bảo mật. Thành phần này gọi là **EcommerceWeb**.
- Phần Website phục vụ cho đối tượng bên trong công ty được chia thành một gói gọi là **ManagementWeb**.
- Phần **Bussiness** được sử dụng để tương tác CSDL và xử lý các nghiệp vụ.
- Phần **PaymentGateway** để xử lý thanh toán trực tuyến.
- Phần **Database Server** cũng được tách ra một gói riêng.

Lưu ý, việc phân chia này tùy thuộc vào nhu cầu tổ chức phát triển và triển khai của hệ thống (cần có kinh nghiệm về kiến trúc hệ thống khi tham gia thiết kế Biểu đồ này).

4.8.3. Ví dụ về Component Diagram trong hệ thống eCommerce:

- ***Bước 2: Xác định quan hệ và vẽ ta được Biểu đồ Component Diagram***



4.8. 4. Ứng dụng của Component Diagram

- Component được sử dụng vào các công việc sau:
 - Thể hiện cấu trúc của hệ thống
 - Cung cấp đầu vào cho Biểu đồ Deployment
 - Hỗ trợ cho việc thiết kế kiến trúc phần mềm
- Component Diagram là một Biểu đồ đơn giản nhưng có ảnh hưởng lớn đến việc thiết kế chi tiết của hệ thống phần mềm và cách thức tổ chức phát triển sản phẩm

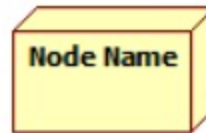
4.9. Deployment Diagram: Biểu đồ triển khai

- Biểu đồ giúp xác định sẽ triển khai hệ thống phần mềm như thế nào. Đồng thời, xác định sẽ đặt các thành phần phần mềm (component lên hệ thống ra sao.
- Deployment Diagram thể hiện rõ kiến trúc triển khai nên nó sẽ ảnh hưởng đến sự thiết kế, phát triển, hiệu năng, khả năng mở rộng của hệ thống v.v...
- Ví dụ về deployment diagram như sau: mô tả hệ thống được triển khai trên 03 Server khác nhau gồm Webserver, Application Server, DB server và 02 thiết bị truy cập đầu cuối.



4.9.1. Các thành phần của Deployment Diagram

- **Node:** là một thành phần vật lý, nó có thể là thiết bị phần cứng hoặc một môi trường nào đó mà các thành phần phần mềm được thực hiện.



- **Relationship:** sử dụng quan hệ Association và Dependence để thể hiện mối quan hệ giữa các node với nhau. Các ký hiệu của chúng như sau:

Ký hiệu về Association



Ký hiệu về Dependence



- Việc xác định quan hệ giữa các thành phần và kết nối chúng lại với nhau chúng ta sẽ có bản vẽ Deployment Diagram.

4.9.2. Xây dựng Deployment Diagram

Bước 1: Xác định các thành phần phần cứng sẽ tham gia vào việc triển khai hệ thống

- Việc này liên quan đến kiến trúc hệ thống, hiệu năng, khả năng mở rộng và cả vấn đề tài chính và hạ tầng của hệ thống nên cần có kinh nghiệm về kiến trúc hệ thống để làm được việc này.

Bước 2: Xác định các thành phần để triển khai lên các Node

- Khi đã có phần cứng, bước tiếp theo chúng ta xác định những component liên quan để triển khai trên mỗi node.

Bước 3: Xác định các quan hệ và hoàn tất bản vẽ

- Xác định các mối quan hệ giữa các thành phần với nhau và nối chúng lại để hoàn tất bản vẽ.

4.9.3. Ví dụ xây dựng Deployment Diagram cho hệ thống eCommerce

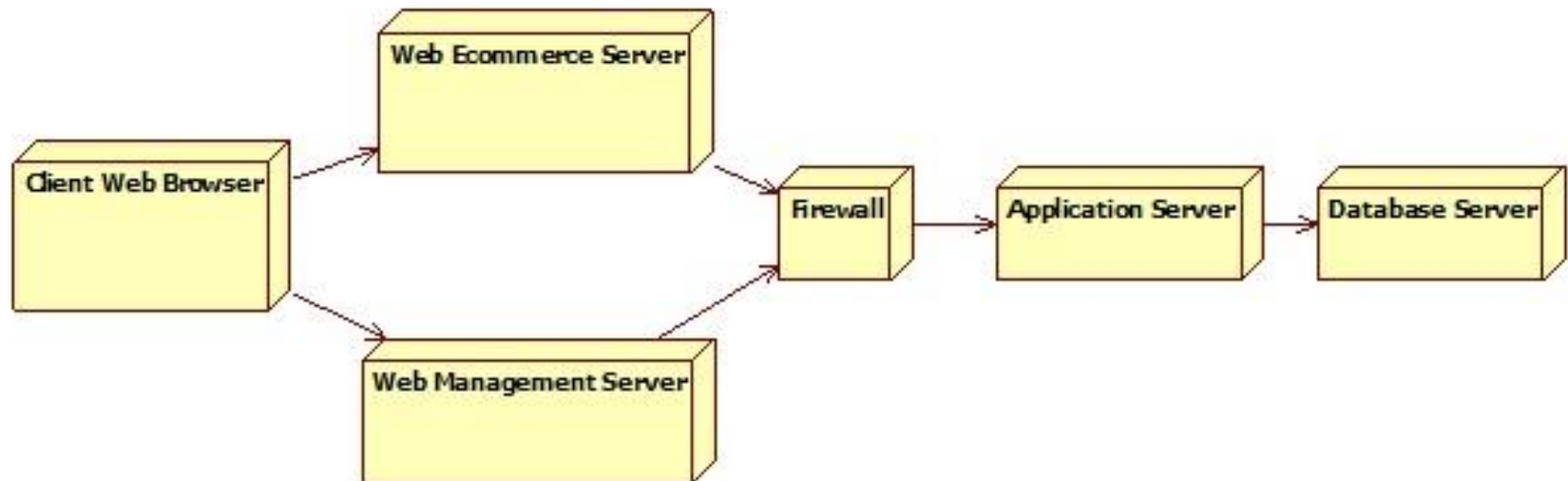
Bước 1: Xác định các Node và bố trí các thành phần lên node

- Để tăng cường an ninh và sức chịu đựng cho hệ thống chúng ta bố trí phần cho người dùng bên ngoài công ty (Guest và Customer) ra một Server riêng gọi là Web E-Commerce Server.
- Website chứa phần tương tác với nhân viên công ty đặt lên một Node riêng gọi là Web Management Server.
- Phần Bussiness được đưa ra một Server ứng dụng gọi là Application Server.
- Database được đặt lên một Server gọi là Database Server.
- Phần PaymentGateWay có thể được đặt trên Web eCommerce Server
- Bổ sung thêm các thiết bị bảo mật và hạ tầng.

4.9.3. Ví dụ xây dựng Deployment Diagram cho hệ thống eCommerce

Bước 2: Xác định quan hệ giữa các thành phần và hoàn tất bản vẽ

- Xem xét các thành phần gọi với nhau để hoàn tất chức năng, chúng ta sẽ xác định các quan hệ của chúng.
- Biểu diễn lên bản vẽ chúng ta sẽ có Deployment Diagram như sau:



4.9. 4. Ứng dụng của Deployment Diagram

4. Ứng dụng của Deployment Diagram


Deployment Diagram có thể ứng dụng vào các trường hợp sau:

- Làm tài liệu để triển khai hệ thống.
- Sử dụng trong thiết kế kiến trúc cho hệ thống.
- Dùng trong giao tiếp với khách hàng, các nhà đầu tư.
- Deployment diagram là một bản vẽ khá đơn giản và dễ xây dựng nhưng có ảnh hưởng rất lớn đến quá trình phát triển, triển khai và kinh phí xây dựng dự án. Do vậy, nên dành thời gian cho bản vẽ này để tránh những khó khăn trong quá trình phát triển các sản phẩm phần mềm.

4.10. Các công cụ sử dụng để xây dựng các Biểu đồ UML

Một số công cụ vẽ UML phổ biến

- Có rất nhiều công cụ được sử dụng để vẽ các bản vẽ UML rất chuyên nghiệp như Rational Rose, Enterprise Architect, Microsoft Visio v.v.. và rất nhiều các công cụ phần mềm nguồn mở miễn phí có thể sử dụng tốt
- StarUML, một phần mềm nguồn mở, miễn phí, có đầy đủ chức năng và có thể sử dụng tốt trên môi trường Windows.
- **Cài đặt:** download bộ cài đặt của phần mềm StarUML tại <http://staruml.sourceforge.net/en/>. Sau khi download và tiến hành các bước cài đặt chúng ta nhanh chóng có được công cụ này trên máy tính.



Q & A