MÔ HÌNH HÓA YÊU CẦU

Các phương pháp dựa trên hành vi và bản mẫu

Mô hình hóa

- **Structure:**
  - CRC card
  - Class diagram
  - Object diagram
- **Functions:**
  - Usecase diagram
  - Activity diagram
- **Behaviors**:

Enough ???

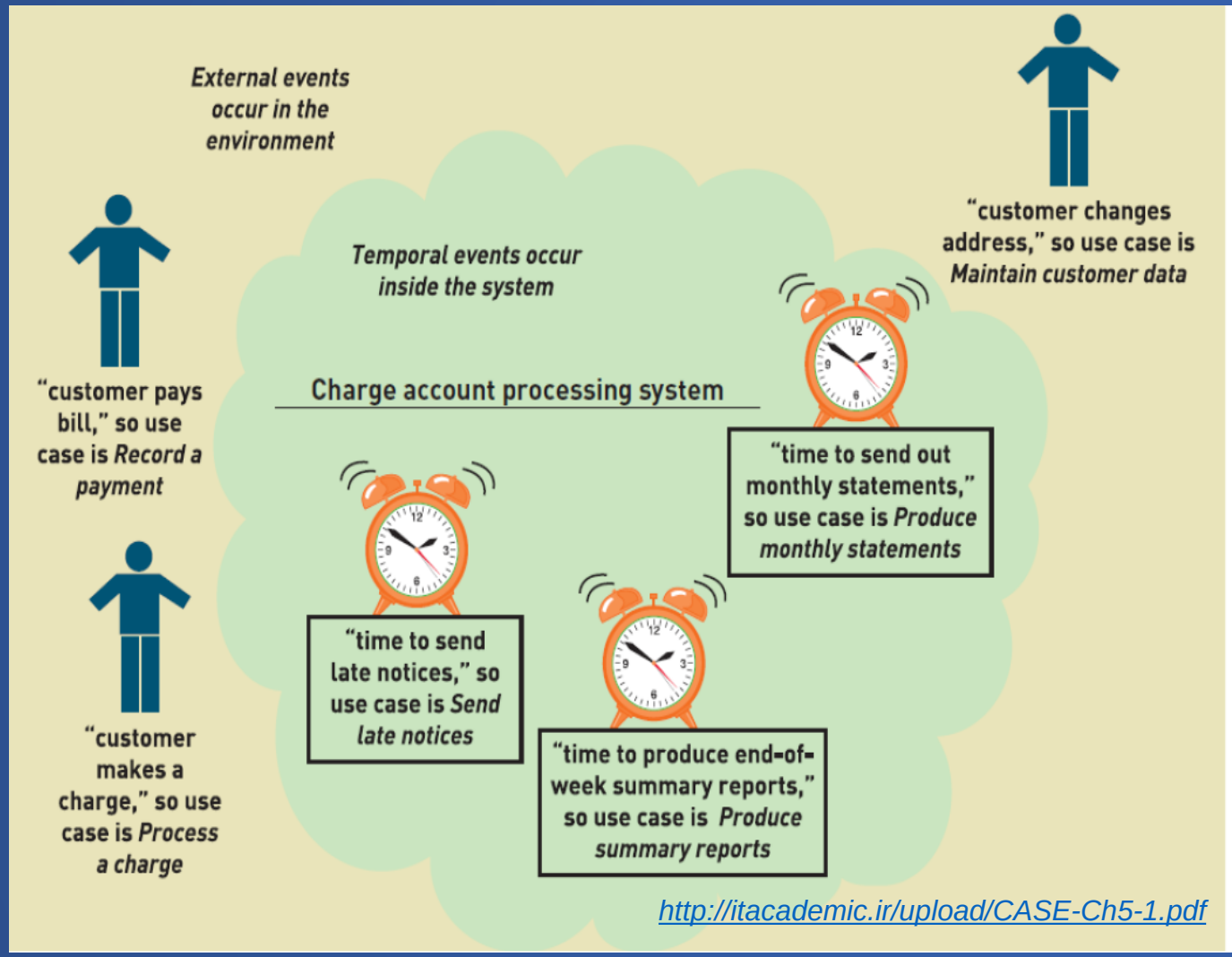
NỘI DUNG

- Mô hình hóa dựa trên hành vi
  - Sequence diagram
  - State diagram
- Mô hình hóa dựa trên bản mẫu
- Mô hình hóa ứng dụng web

Mô hình hóa hành vi

- Các mô hình trước đại diện cho CÁC YẾU TỐ TĨNH của hệ thống.
- Các mô hình dựa trên hành vi (gọi tắt là mô hình hành vi) sẽ biểu diễn HÀNH VI ĐỘNG của hệ thống.
- Hành vi của hệ thống được biểu diễn dưới dạng hàm của SỰ KIỆN và THỜI GIAN cụ thể.
- Mô hình hành vi chỉ ra cách phần mềm phản ứng với các sự kiện hoặc tương tác bên ngoài.

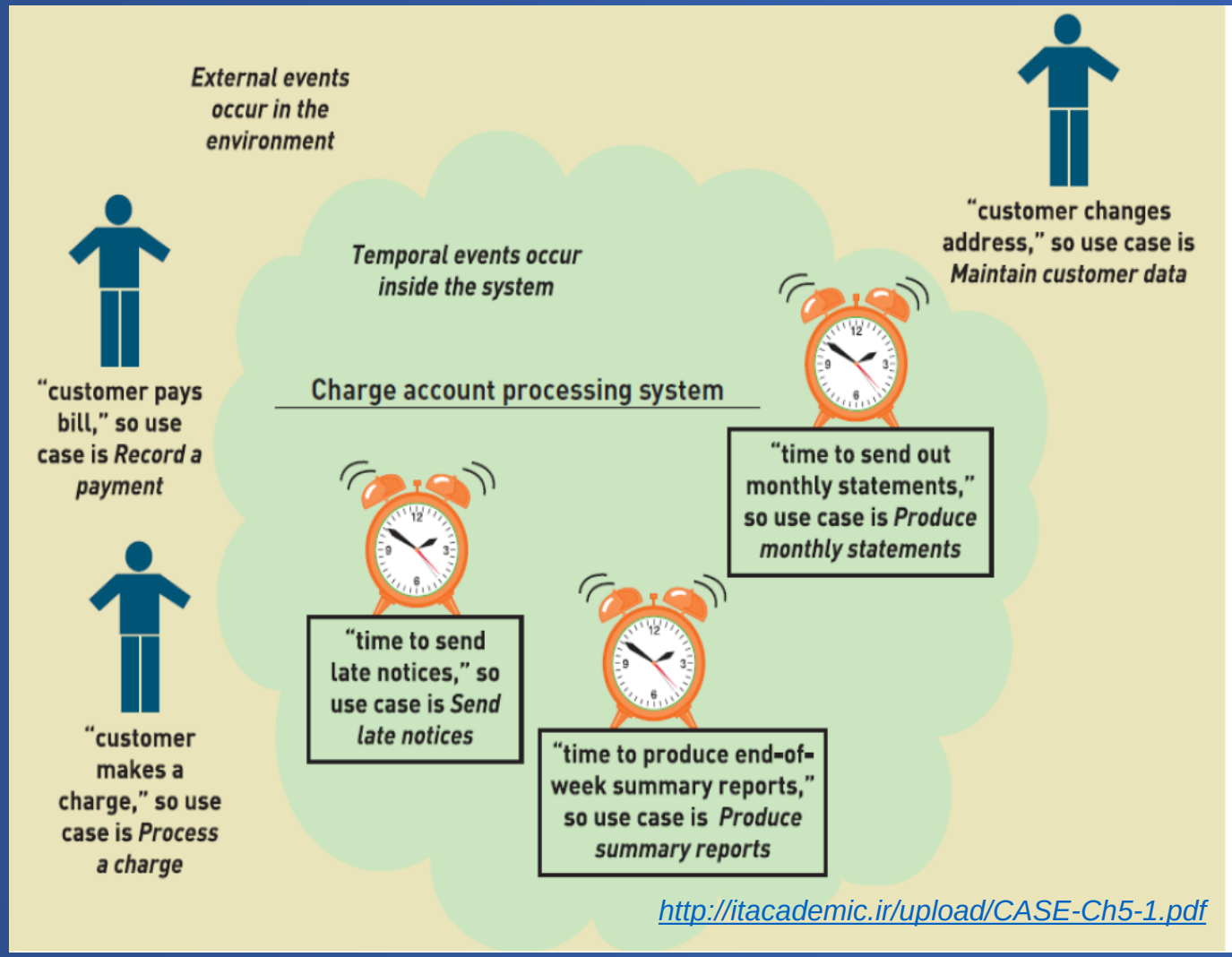
Sự kiện vs Usecase



http://itacademic.ir/upload/CASE-Ch5-1.pdf

- Sự kiện xảy ra bất cứ khi nào hệ thống và một tác nhân trao đổi thông tin.
- Một sự kiện không phải là thông tin đã được trao đổi, mà thực tế là thông tin đã được trao đổi.

Sự kiện vs Usecase



http://itacademic.ir/upload/CASE-Ch5-1.pdf

Các loại sự kiện:

- **Bên ngoài**
  - Được phát sinh bởi các tác nhân bên ngoài tác động vào hệ thống (actor hoặc tác nhân khác)
- **Thời gian**
  - Được phát sinh khi đến một thời điểm nào đó
- **Trạng thái**
  - Được phát sinh khi trạng thái của các đối tượng trong hệ thống thay đổi

Sequence diagram

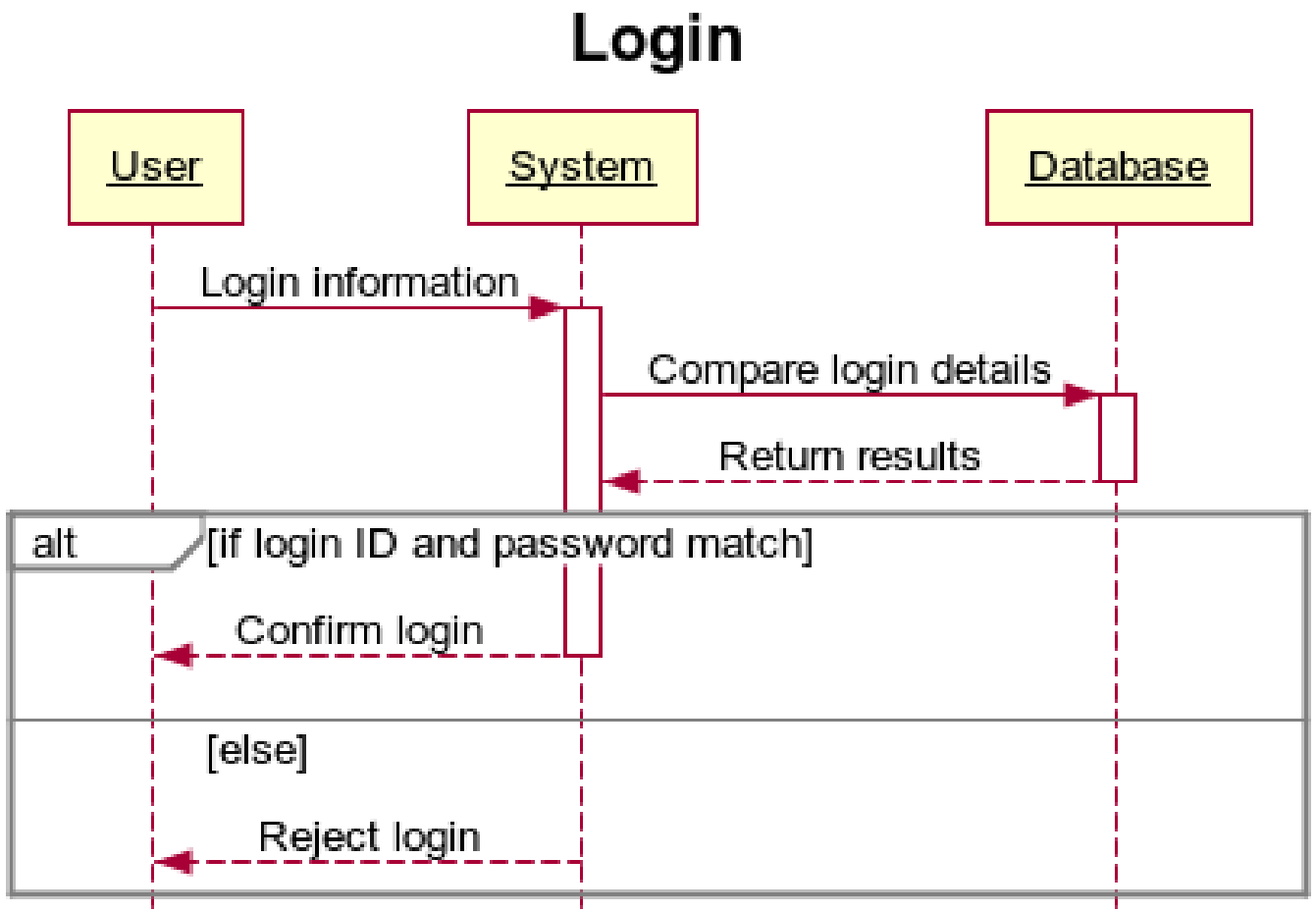
Sequence diagram

- Biểu diễn khía cạnh "động" của hệ thống theo thời gian
- Mô tả sự tương tác giữa các đối tượng
  - How? (tương tác như thế nào?)
  - Order? (thứ tự các tương tác?)
- Là trình tự các xử lý diễn ra bên trong 1 usecase hoặc một sự kiện nào đó trong hệ thống.
  - mô tả chi tiết trình tự thời gian gởi và nhận các thông điệp giữa các đối tượng.
- Là sơ đồ hành vi, hiển thị các kịch bản cụ thể, đơn giản, nhấn mạnh vào các đối tượng cộng tác và thông điệp mà chúng trao đổi trong quá trình thực hiện một kịch bản.

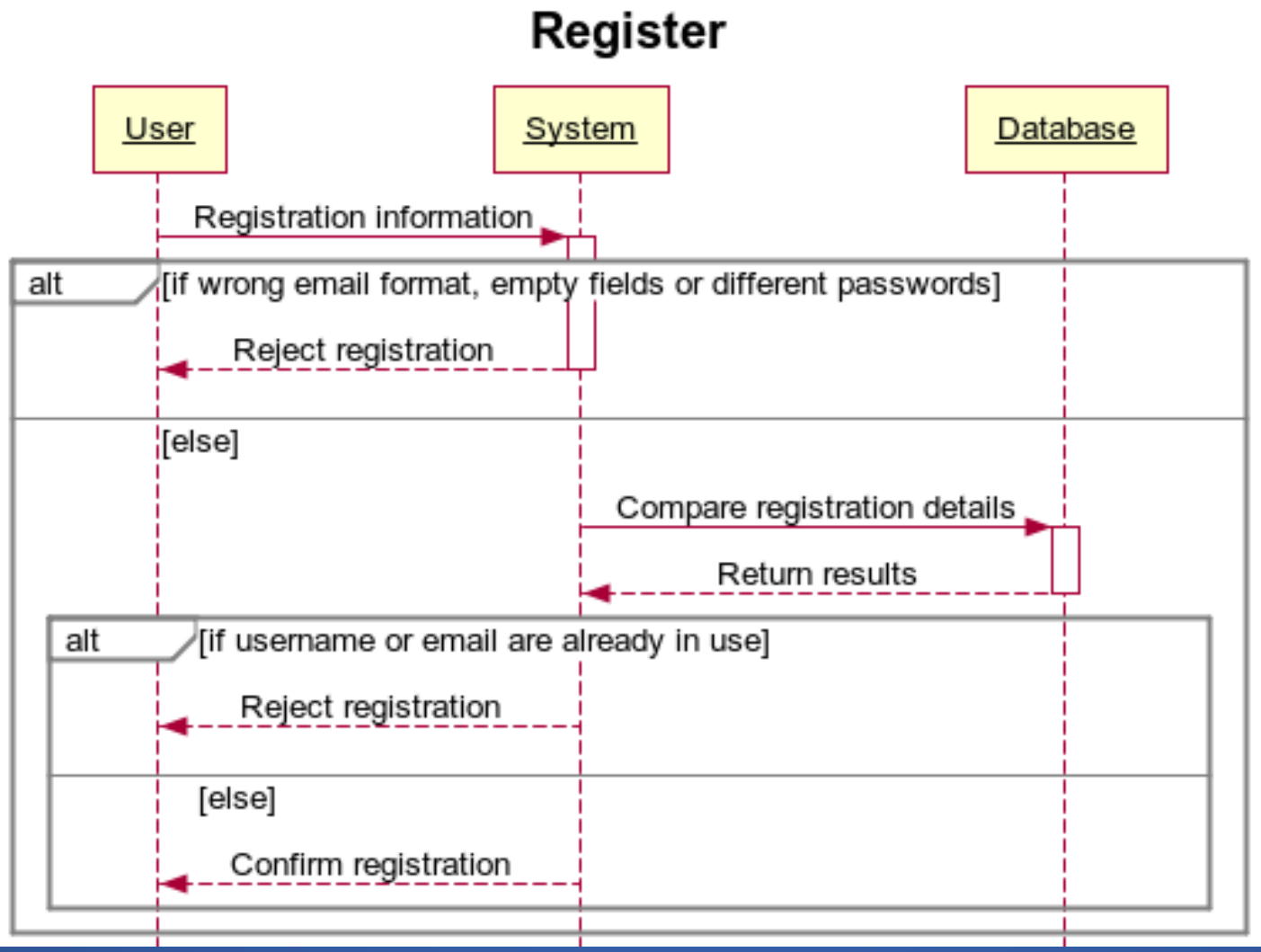
Sequence diagram

- Khi nào cần Sequence diagram?
  - Mô tả chi tiết các yêu cầu
  - Mô tả quy trình cụ thể cho một chức năng/xử lý
- Còn được gọi là “event diagrams/event scenarios”

Sequence diagram

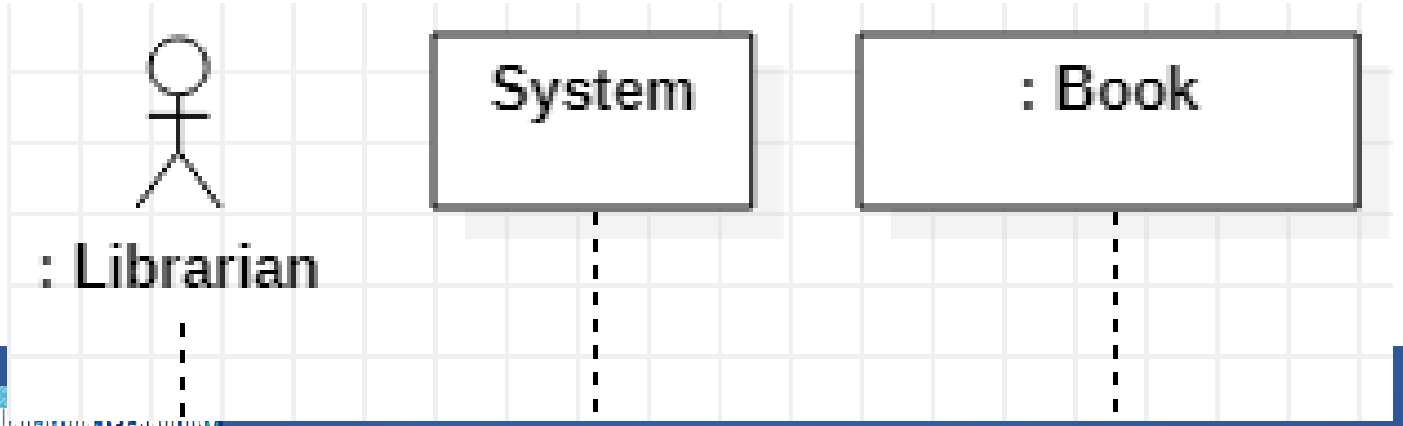


Sequence diagram



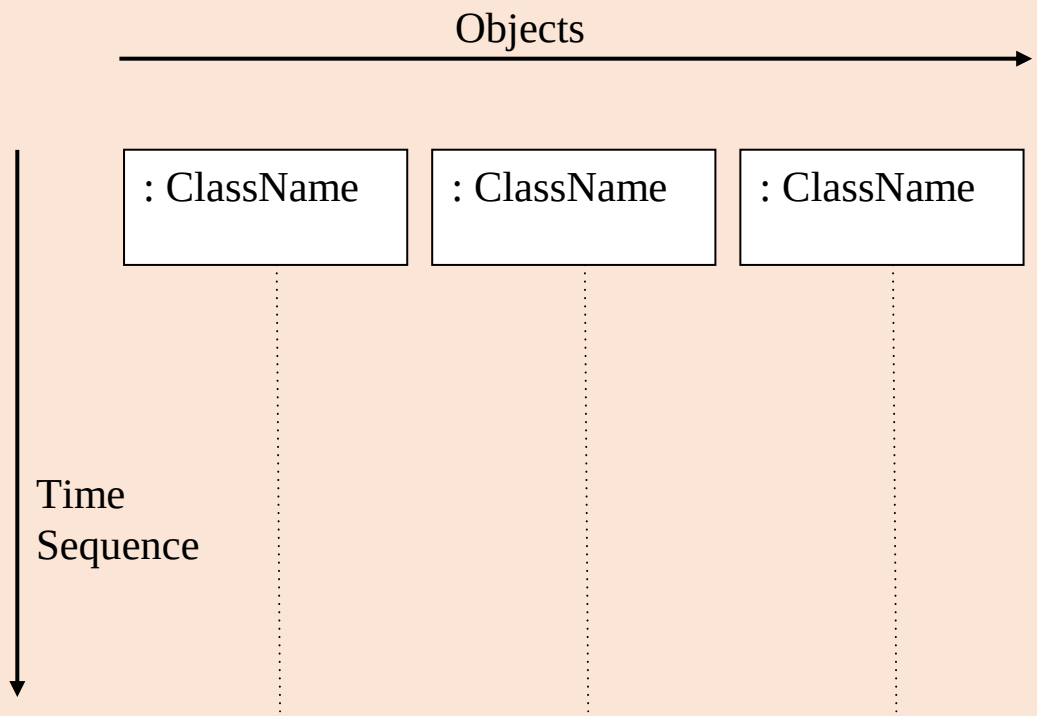
Lifeline (đường đời)

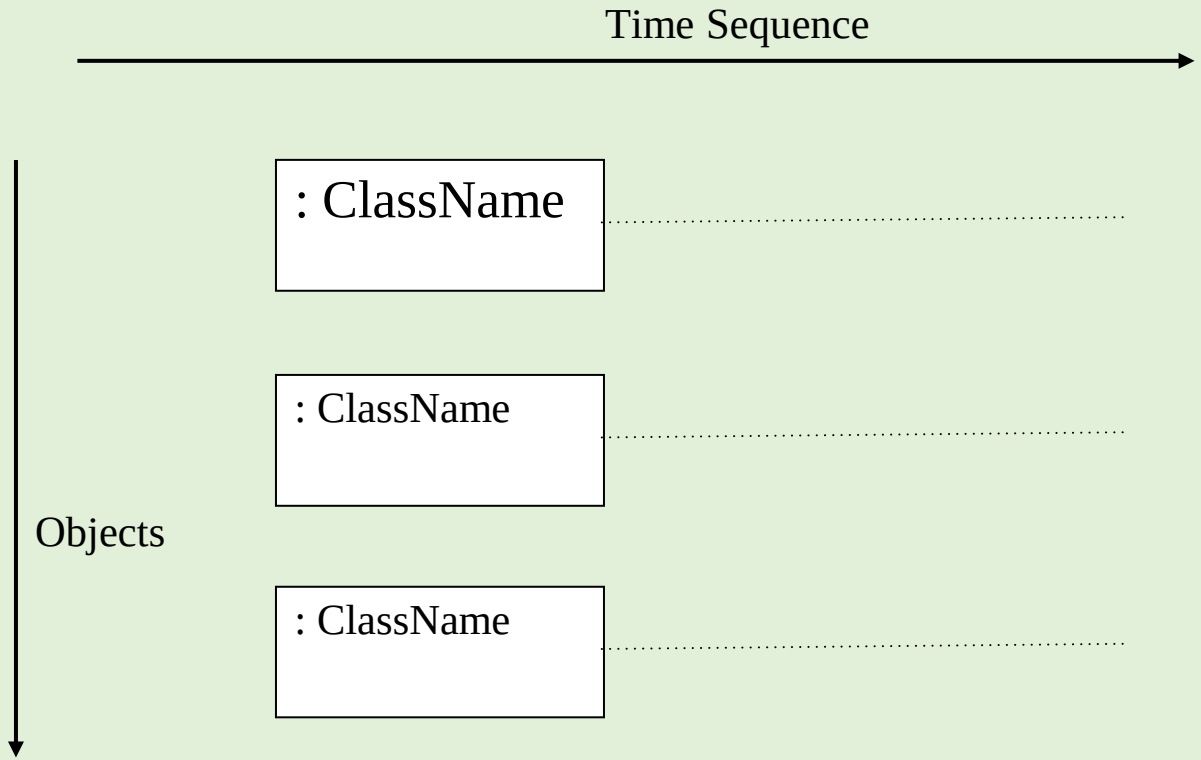
- Là thời gian tồn tại của actor/đối tượng trong một chức năng/sự kiện
  - Actor: những tác nhân bên ngoài nhưng có sự tương tác lên hệ thống
  - Object: đối tượng bên trong hệ thống
- Được biểu diễn là đường thẳng nét đứt bắt nguồn từ actor hoặc object
- Vị trí sinh ra và mất đi của Lifeline được xác định trong sơ đồ



Trục tọa độ

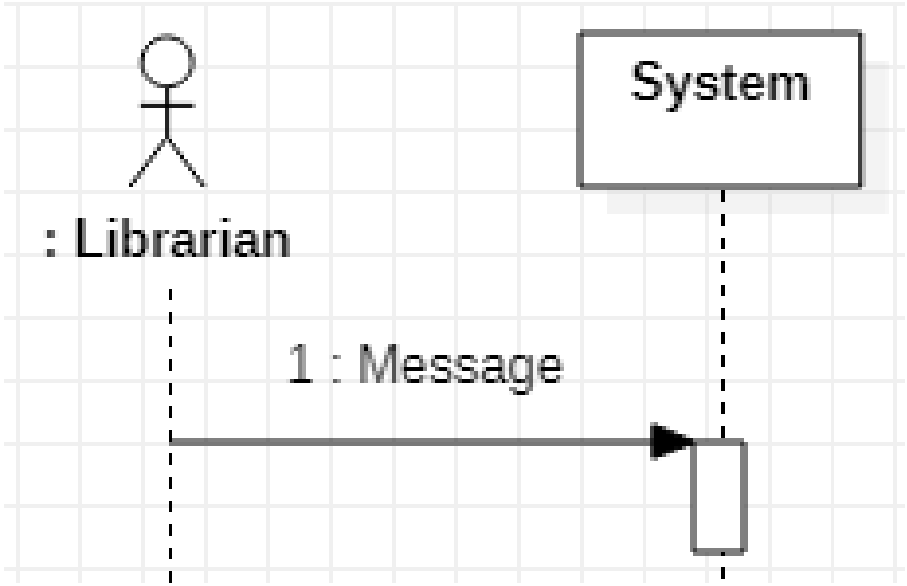
- Trục actor/đối tượng & Trục thời gian





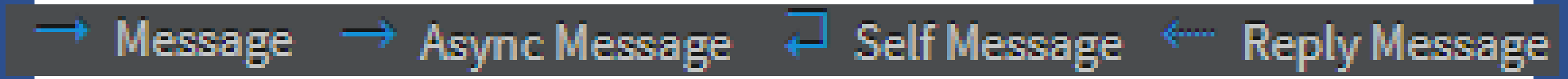
Message/stimulus

- Message: là giao tiếp giữa các lifeline, ký hiệu là đường mũi tên giữa các lifeline.



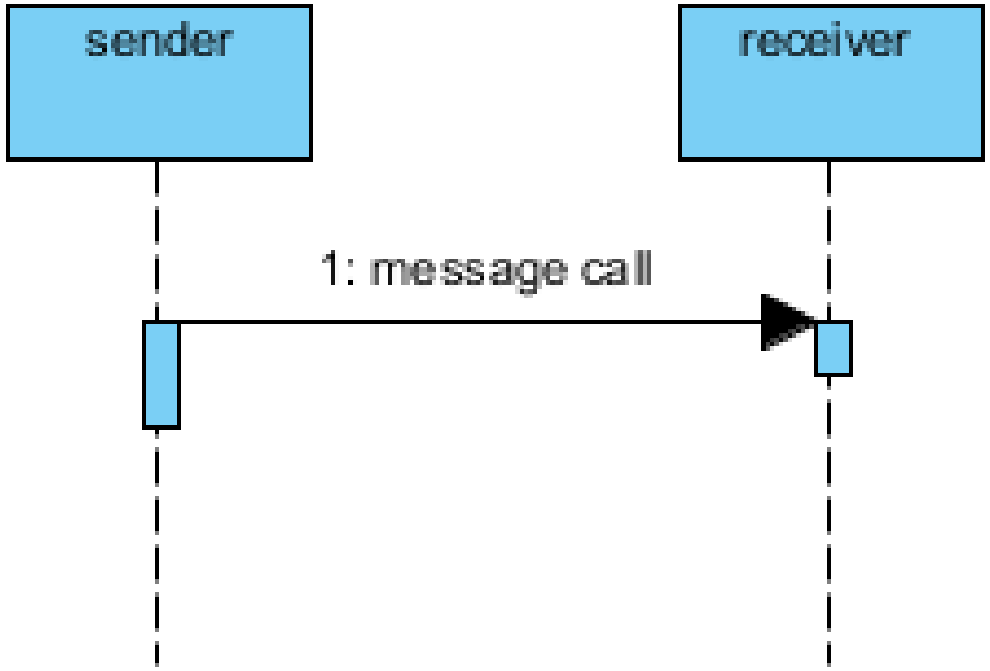
Các loại message

- Đồng bộ (Synchronous)
- Bất đồng bộ (Asynchronous)
- Tự thân (Self message)
  - Message được gởi cho bản thân đối tượng
- Phản hồi (Return)
  - Kết quả trả về của 1 message



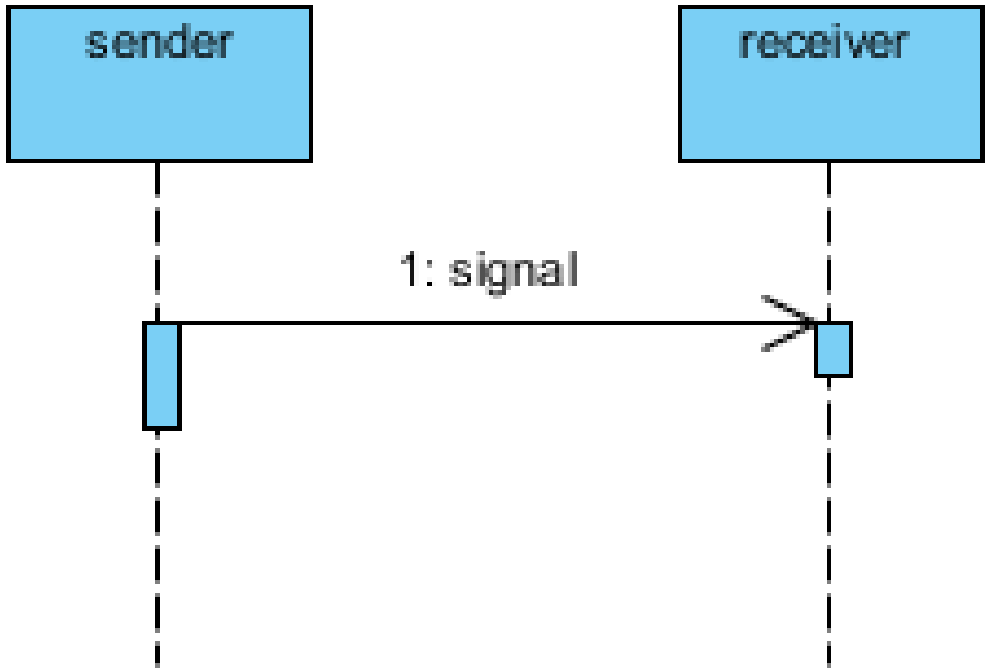
Các loại message

- **Synchronous message:** mũi tên có fill màu ở đầu.



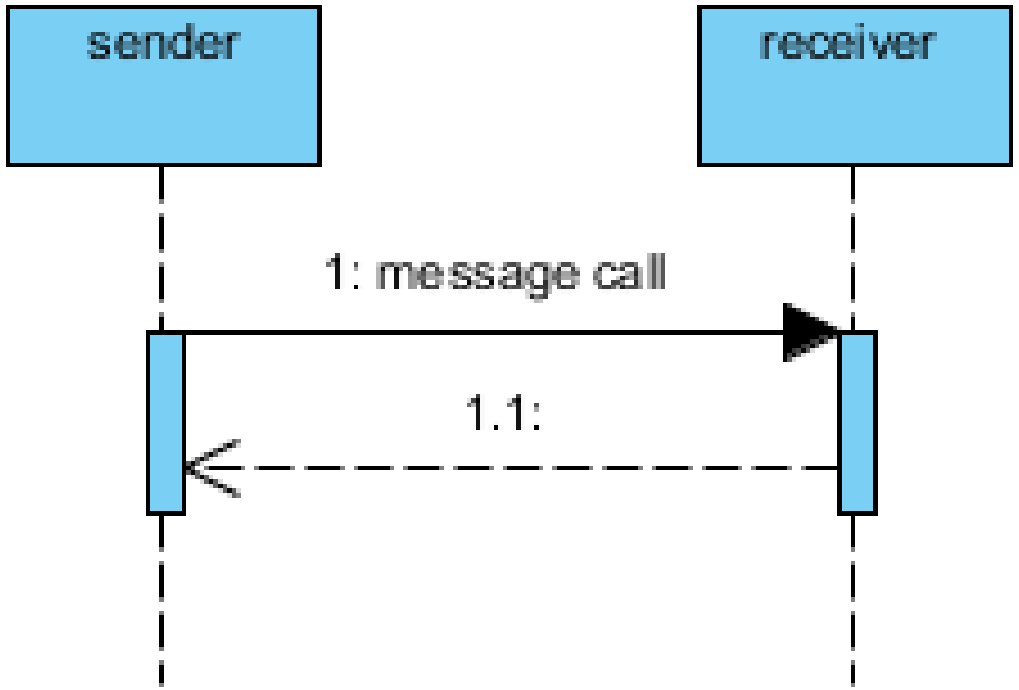
Các loại message

- **Asynchronous message:** mũi tên không fill màu ở đầu.



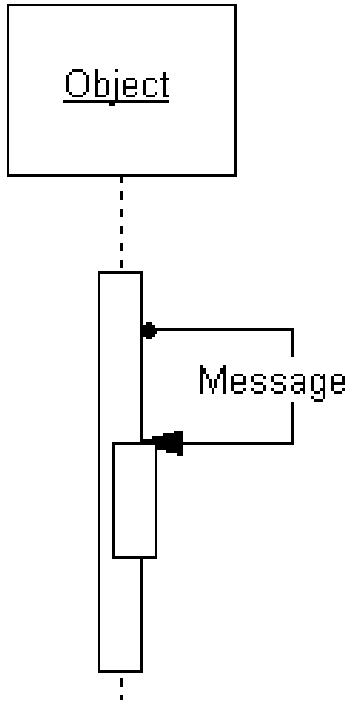
Các loại message

- **Return message:** mũi tên nét đứt.



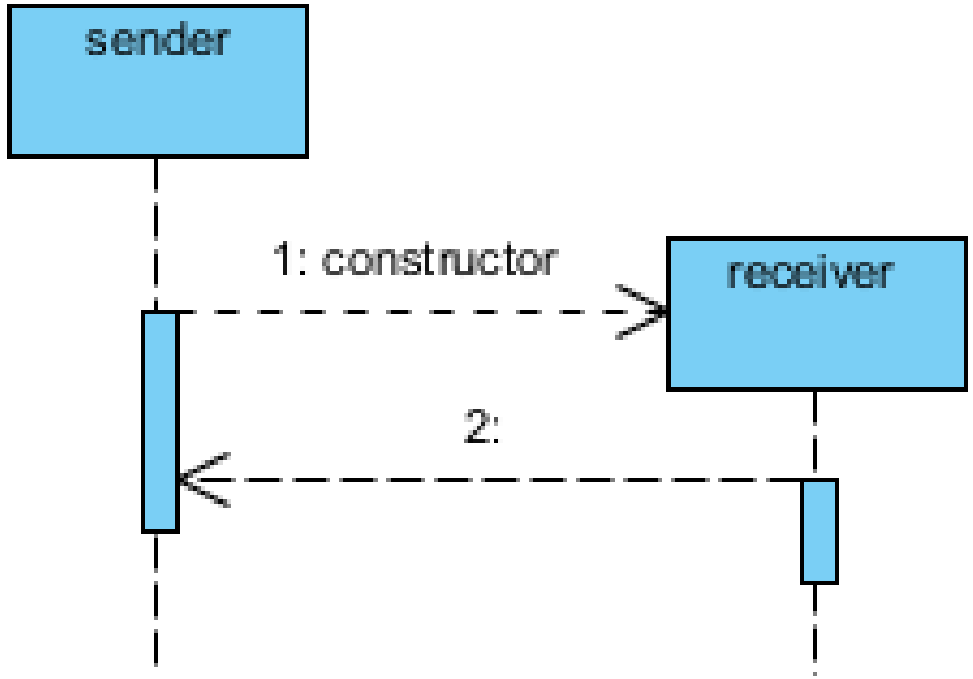
Các loại message

- **Self message:** mũi tên xuất phát và đến cùng một sender.



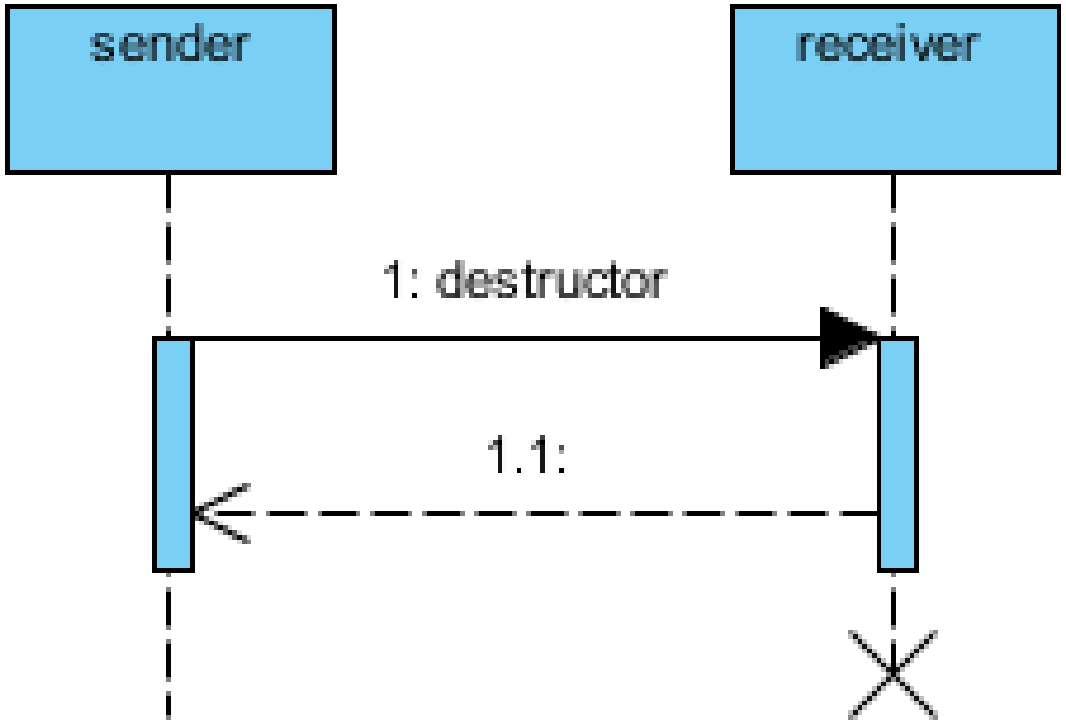
Các loại message

- **Constructor message:** message tạo một object



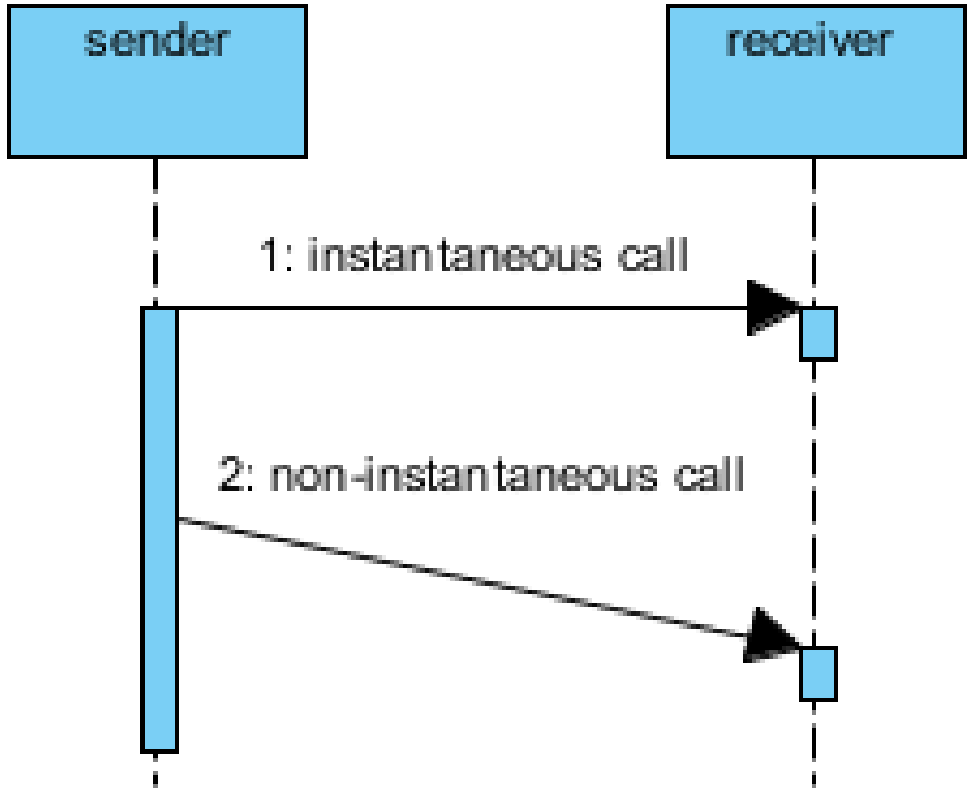
Các loại message

- **Destructor message:** message hủy một object



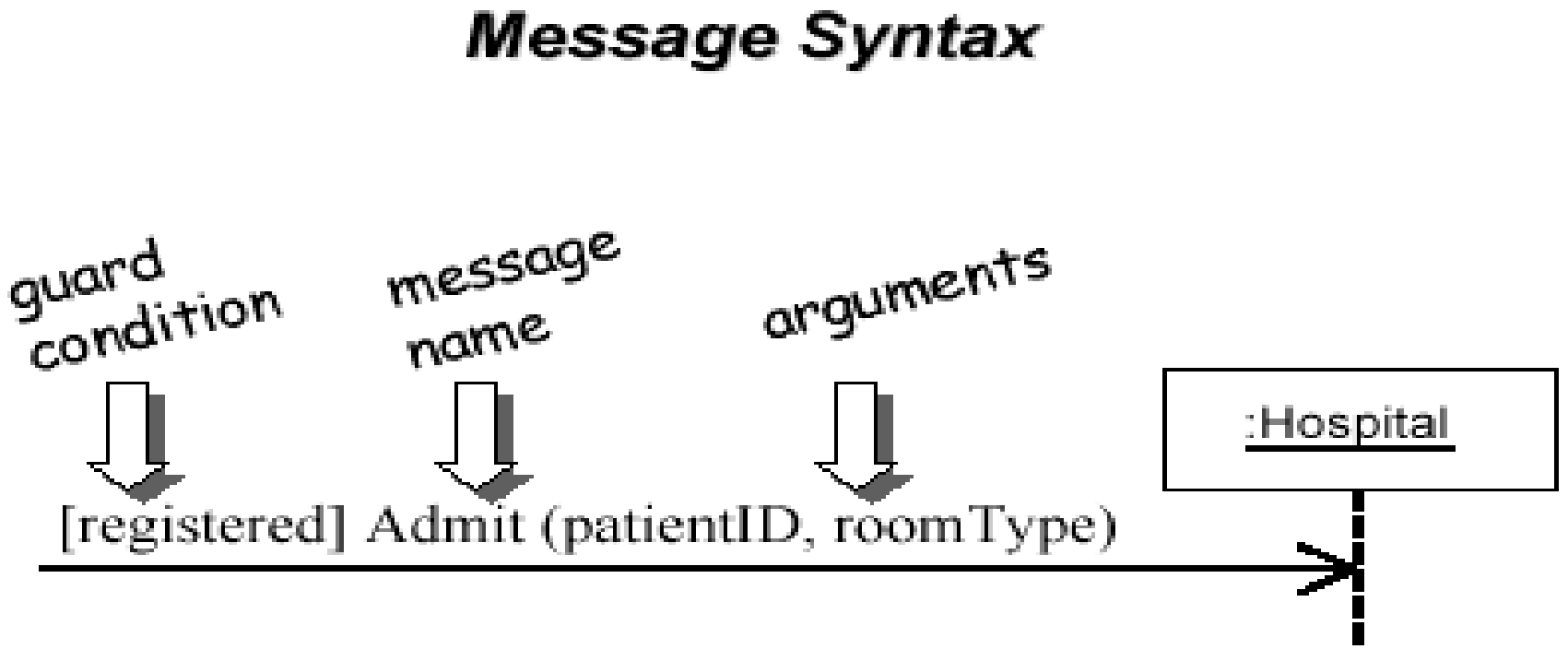
Các loại message

- **Non-instantaneous message:** message có thời gian delay trước khi được gởi



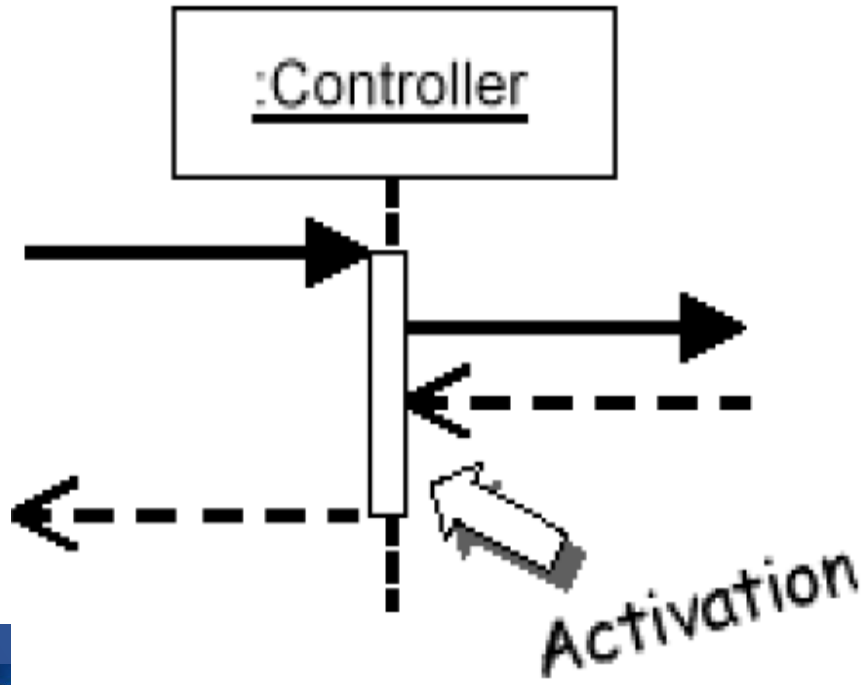
Message/stimulus

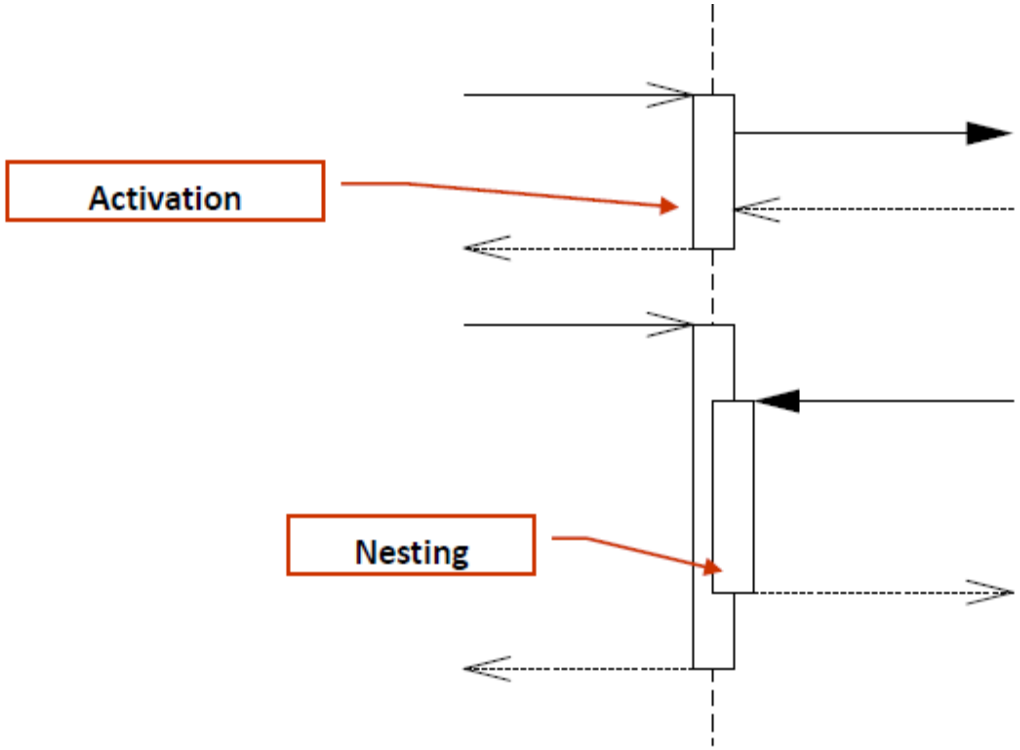
- Trong sơ đồ thiết kế hệ thống, message thường là lời gọi hàm/phương thức



Activation

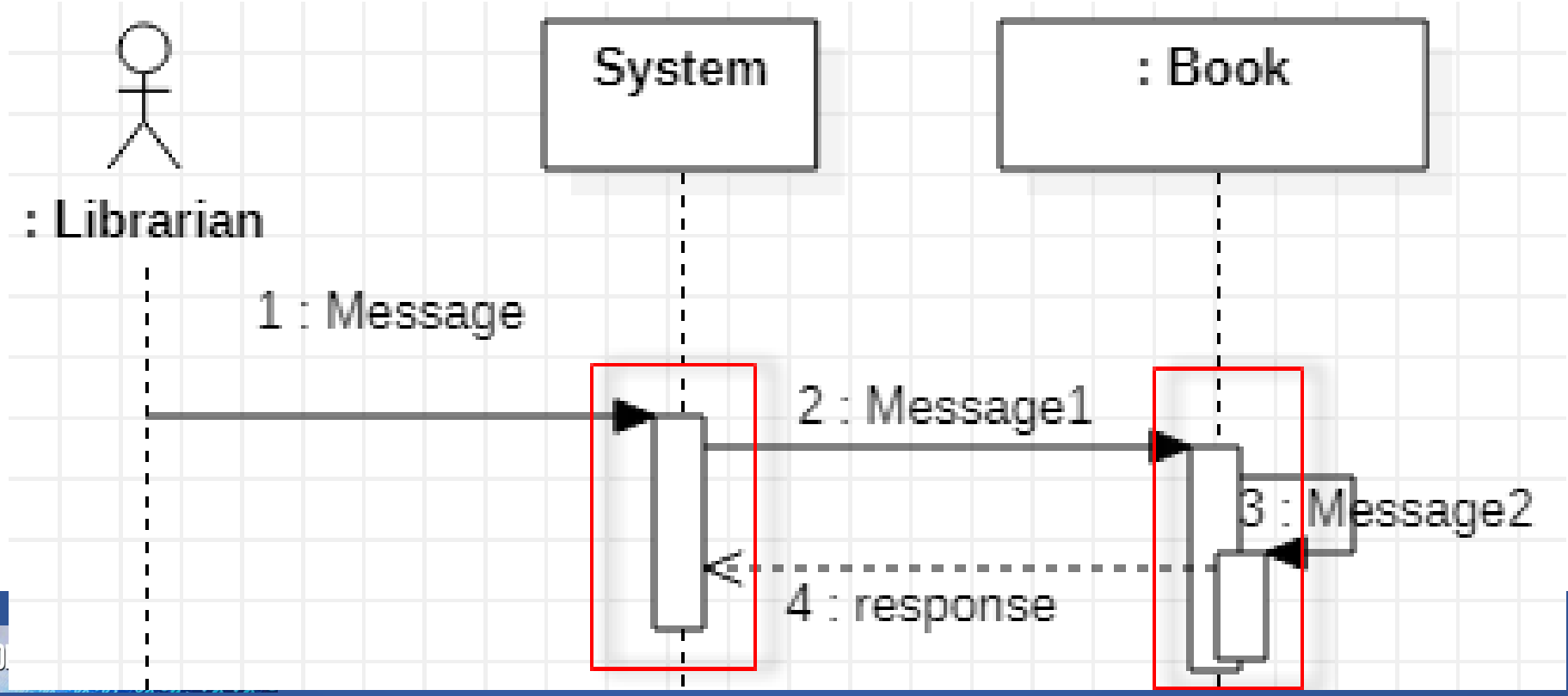
- Khoảng thời gian gởi mesage hoặc chờ đợi phản hồi của message trên Lifeline
- Có thể lồng vào nhau trong trường hợp đệ qui





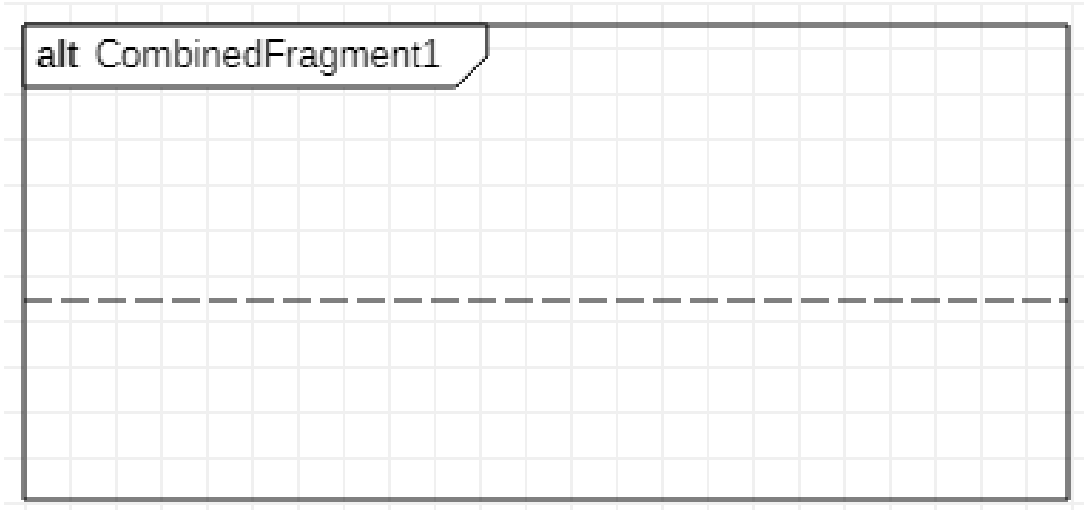
Activation

- Khoảng thời gian gởi mesage hoặc chờ đợi phản hồi của message trên Lifeline
- Có thể lồng vào nhau trong trường hợp đệ qui



Fragment

- Mô tả một nhóm các message (tương tác)
  - điều kiện, vòng lặp và một số cấu trúc lập trình khác
- Ký hiệu là một "box" (hình chữ nhật)



Fragment

Operator	Meaning
alt	**Alternative multiple fragments**: rẽ nhánh if…else
opt	**Optional**: điều kiện thực thi: if
par	**Parallel**: thực thi song song
loop	**Loop**: vòng lặp
sd	**Sequence diagram**: tạo khung cho sequence diagram

State diagram

State diagram

- Biểu diễn quá trình biến đổi trạng thái của các phần tử trong hệ thống/hệ thống từ lúc sinh ra đến lúc mất đi.
  - Phần tử: đối tượng của các class hoặc hệ thống phần mềm.
- Ví dụ: Một đơn hàng sẽ có các trạng thái: mới tạo, được xác nhận, đã thanh toán, đã giao hàng, đã hủy, ….

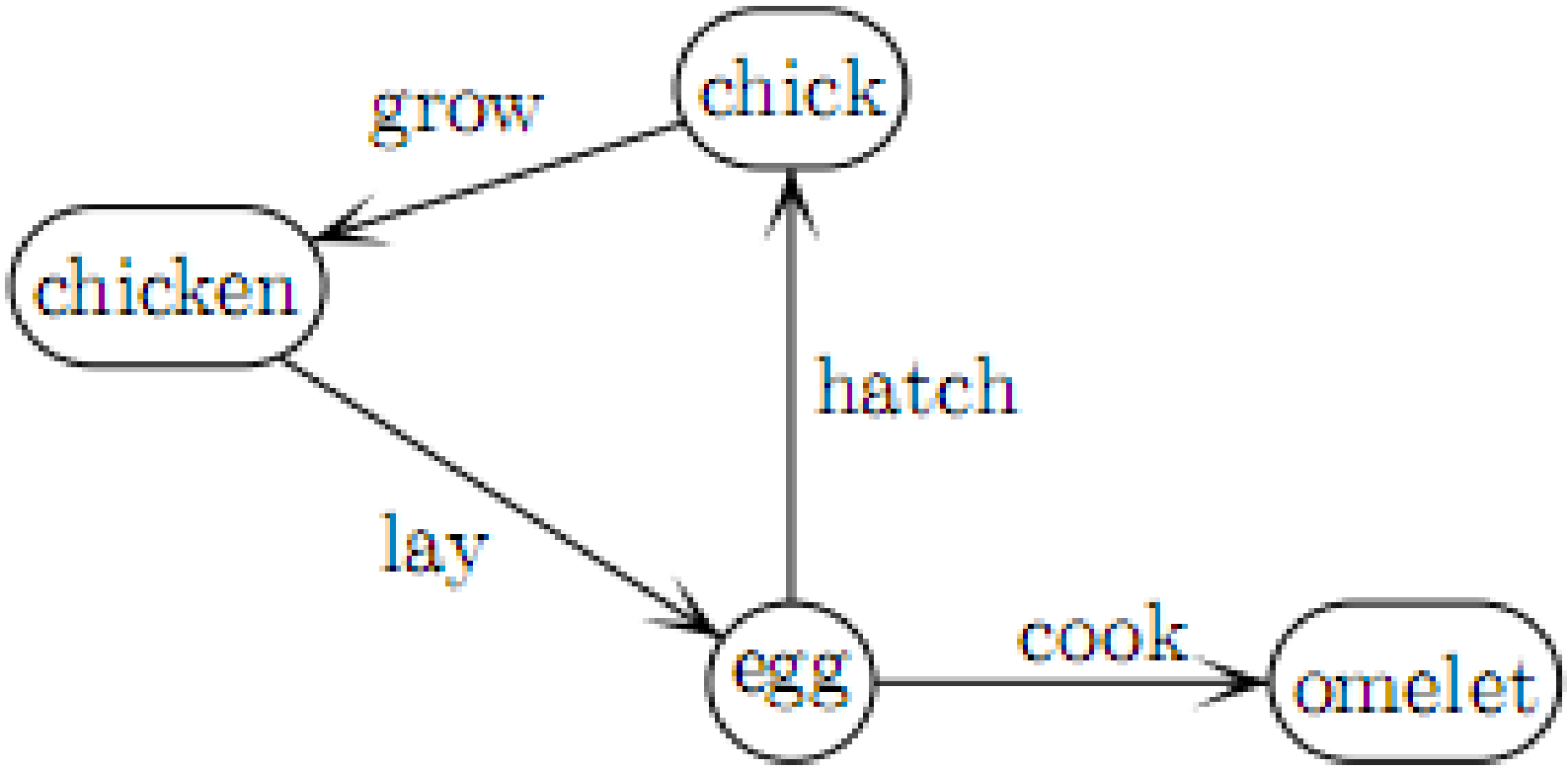
State diagram

Mục đích: mô hình hóa

- Khía cạnh "động" của hệ thống
- Sự tương tác bên trong hệ thống
- Phản ứng của các phần tử khi có các sự kiện xảy ra
- Vòng đời của các phần tử
- Sự thay đổi trạng thái của các phần tử

State diagram

Quá trình thay đổi trạng thái của đối tượng “con gà”:



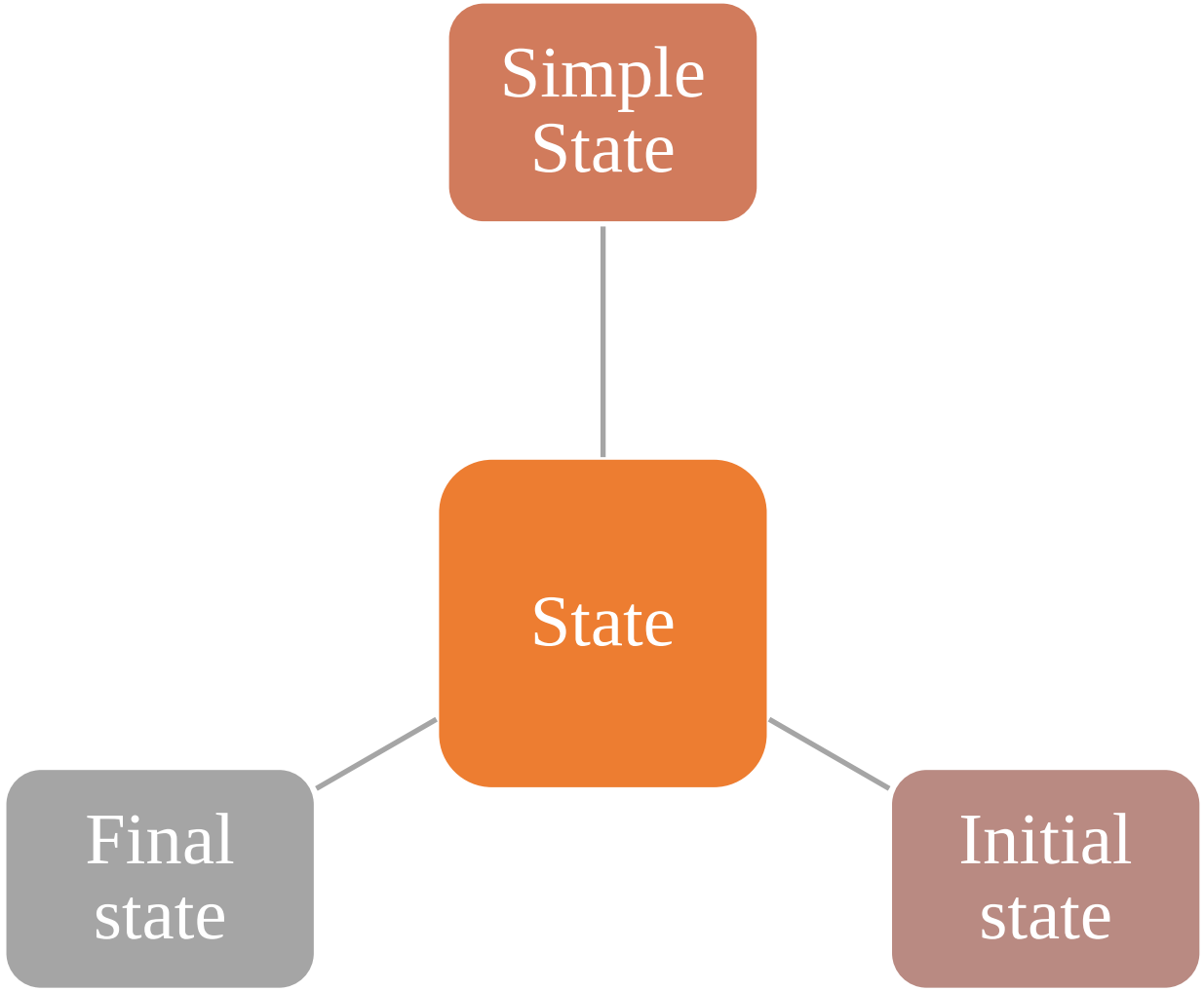
Vòng đời của 1 phần tử (đối tượng/hệ thống)

- Là các giai đoạn mà phần tử thay đổi trạng thái, tính từ lúc nó được tạo ra cho đến khi nó bị hủy.
- Bao gồm:
  - Được tạo ra
  - Nhận thức
  - Hành động
  - Giao tiếp với các đối tượng khác
  - Bị hủy

State

- Là trạng thái/tình trạng của một phần tử.
- State sẽ được thay đổi khi có một sự kiện/thao tác nào đó xảy ra
- Ví dụ:
  - Một quyển sách đang ở trạng thái "sẵn dùng"
  - Khi đọc giả thực hiện chức năng "Mượn sách" (mượn quyển sách đó), nó chuyển sang trạng thái "bận".
  - Khi đọc giả thực hiện chức năng "Trả sách" (trả quyển sách đó), nó chuyển về trạng thái "sẵn dùng"

Các loại State



Initial State

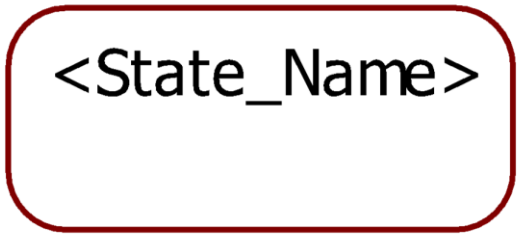
- Là trạng thái được sinh ra của phần tử
- Được biểu diễn là hình tròn nhỏ tô màu
- Là bắt đầu của State diagram

Final State

- Là trạng thái bị hủy của phần tử
- Được biểu diễn là hình tròn nhỏ tô màu có đường viền nét đôi
- Là kết thúc của State diagram

Simple State

- Là trạng thái/tình trạng (ngoài sinh ra và hủy) của phần tử
- Biểu diễn là hình chữ nhật với 4 góc tròn.
  - Tên/mô tả ngắn gọn của trạng thái được đặt bên trong hình chữ nhật



Simple State

- Là trạng thái/tình trạng (ngoài sinh ra và hủy) của phần tử
- Biểu diễn là hình chữ nhật với 4 góc tròn.
  - Tên/mô tả ngắn gọn của trạng thái được đặt bên trong hình chữ nhật

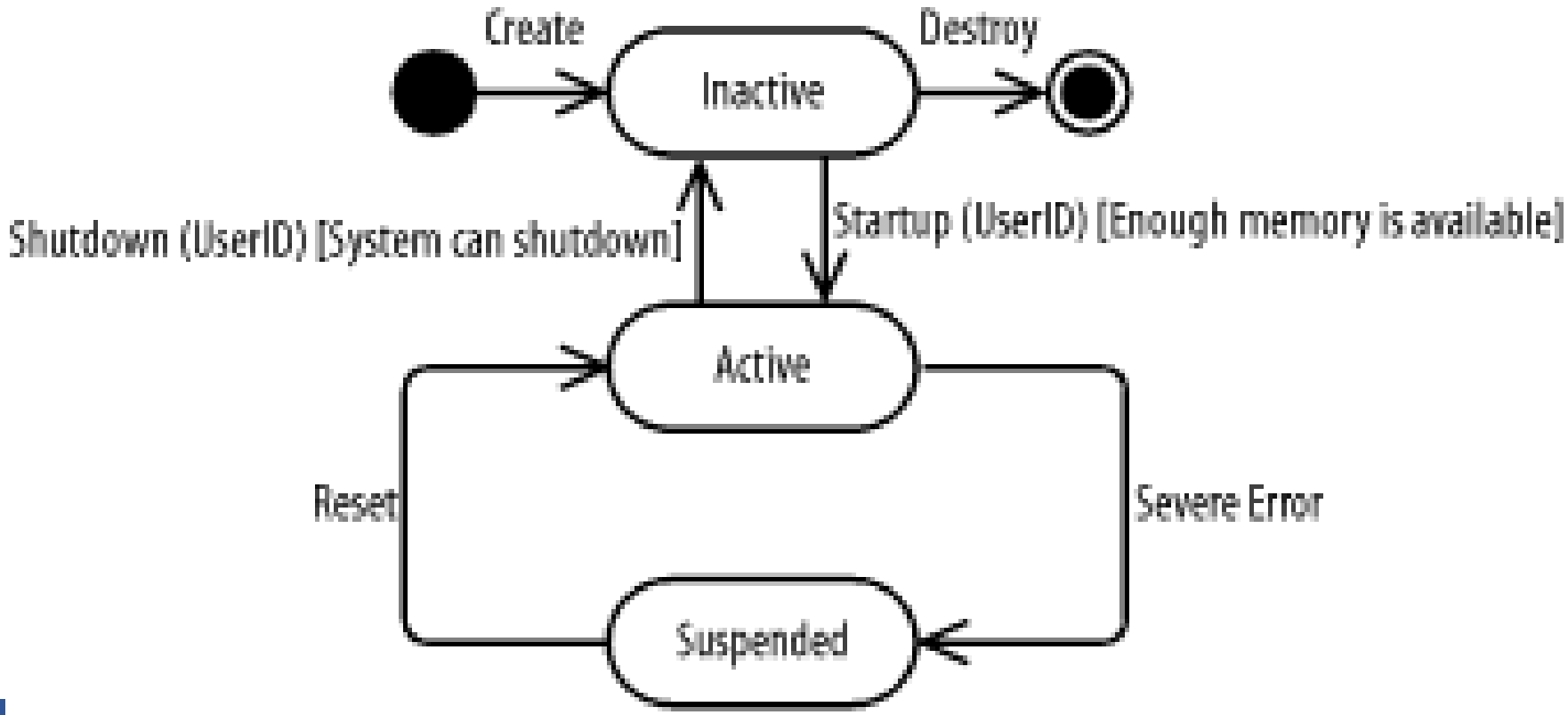
Transition

- Là sự kiện làm thay đổi trạng thái
- Xác định trình tự thay đổi giữa các trạng thái
- Mô tả sự liên quan giữa các trạng thái
- Được biểu diễn như sau:

  - Mũi tên: hướng thay đổi trạng thái
  - Label: tên sự kiện/thao tác/xử lý làm thay đổi trạng thái
    - Label có thể là một lời gọi hàm

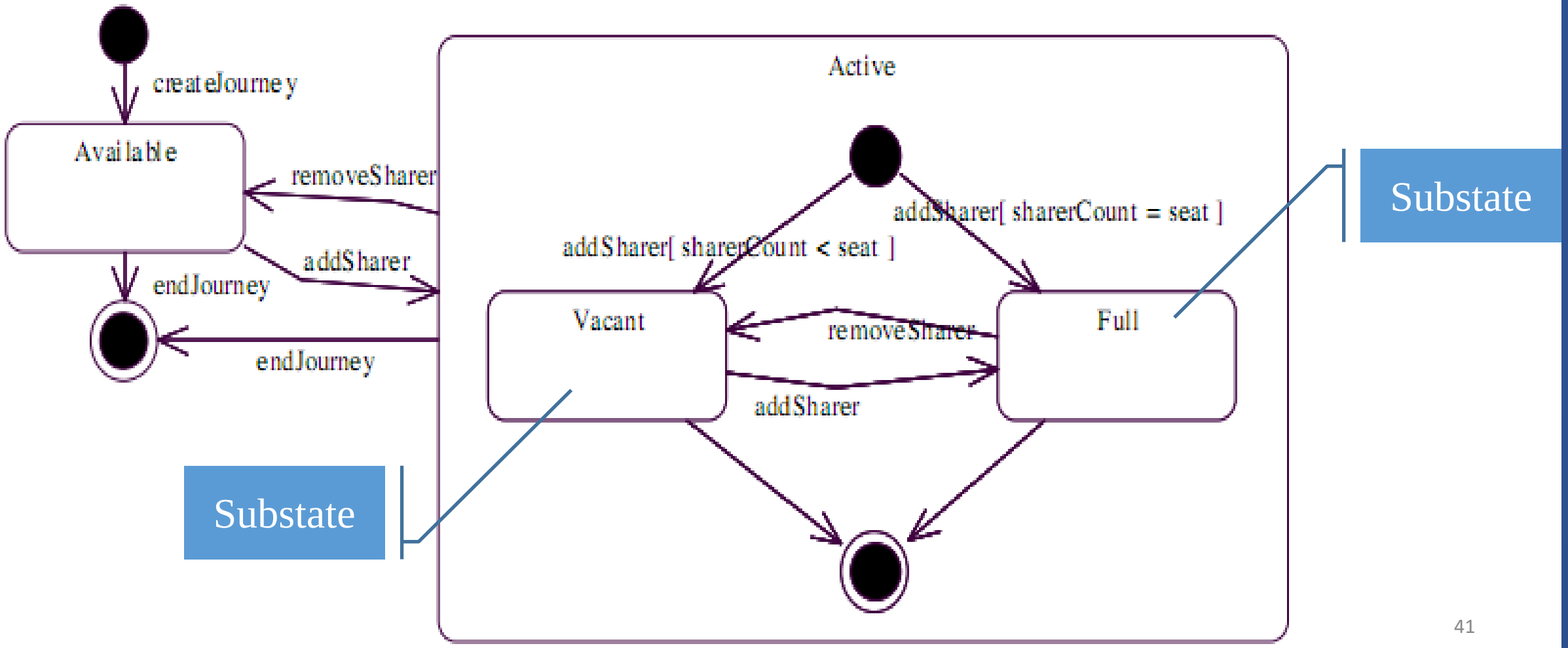
Transition

- Ví dụ: Sơ đồ trạng thái của Hệ thống phần mềm Quản lý dự án



Nested State

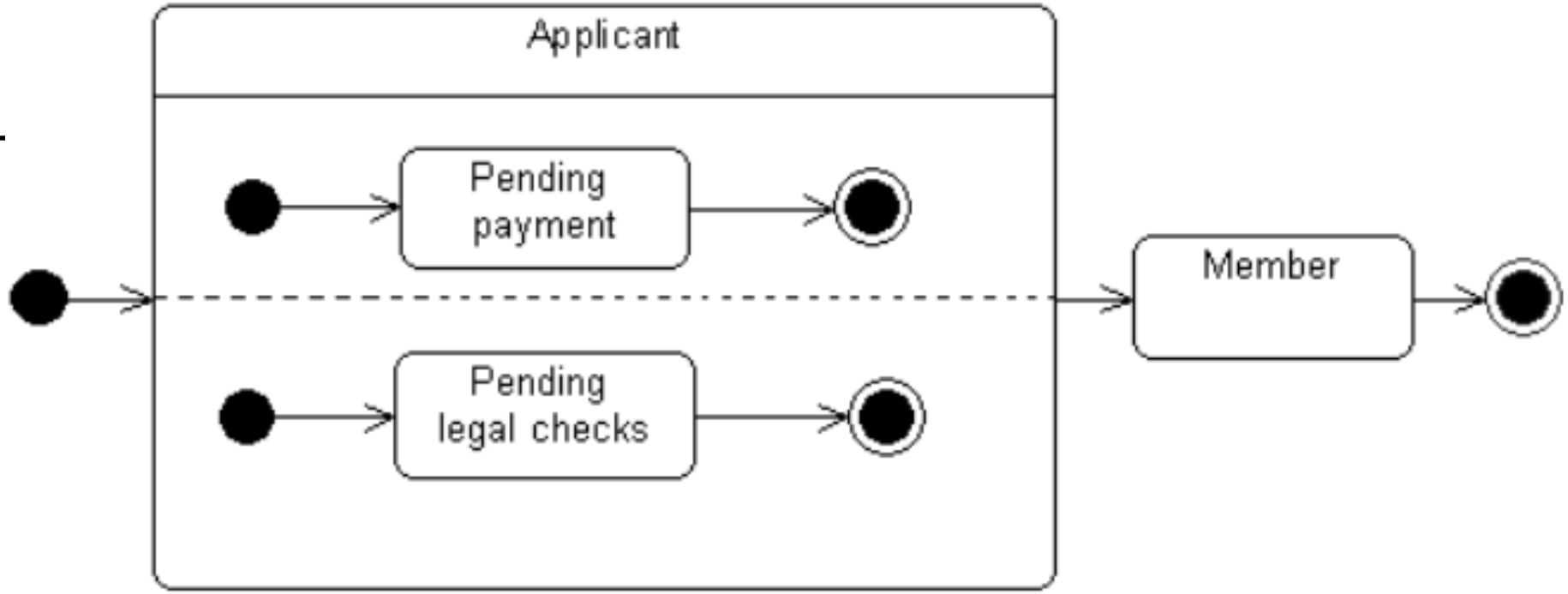
- Là một state có thể chứa bên trong 1 state diagram khác



Concurrent substate

- Substate: các trạng thái con của một state
- Concurrent substate: các trạng thái con có thể tồn tại song song với nhau
  - Các concurrent substate có thể kết thúc ở các thời điểm khác nhau

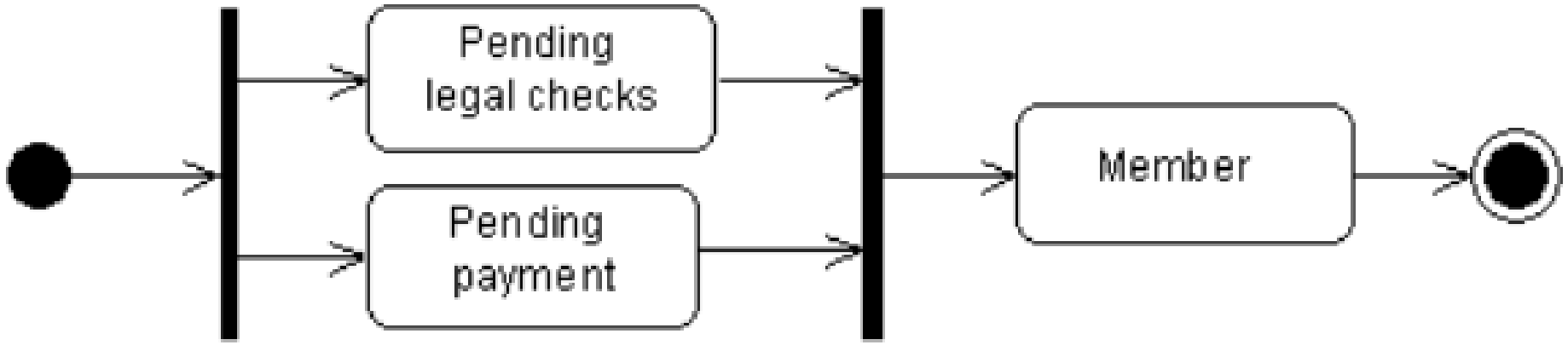
Biểu diễn cách 1:



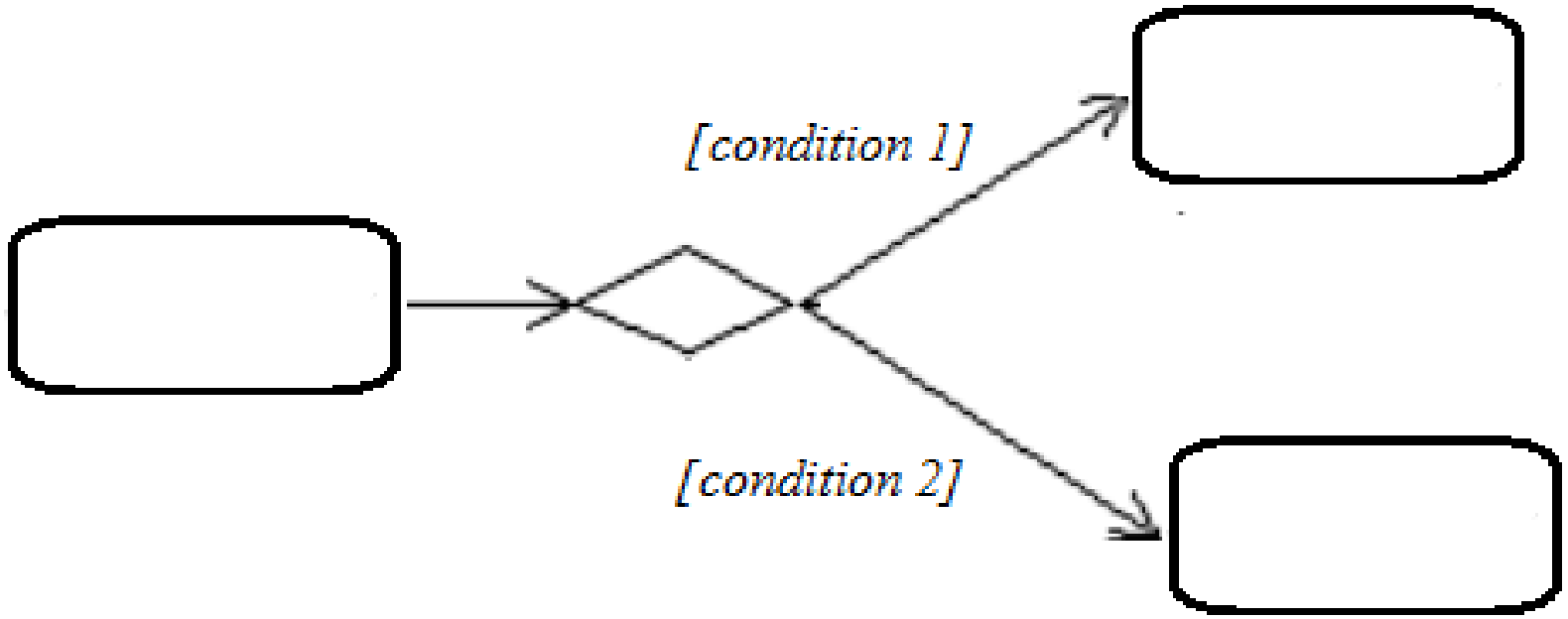
Concurrent substate

- Substate: các trạng thái con của một state
- Concurrent substate: các trạng thái con có thể tồn tại song song với nhau
  - Các concurrent substate có thể kết thúc ở các thời điểm khác nhau

Biểu diễn cách 2:

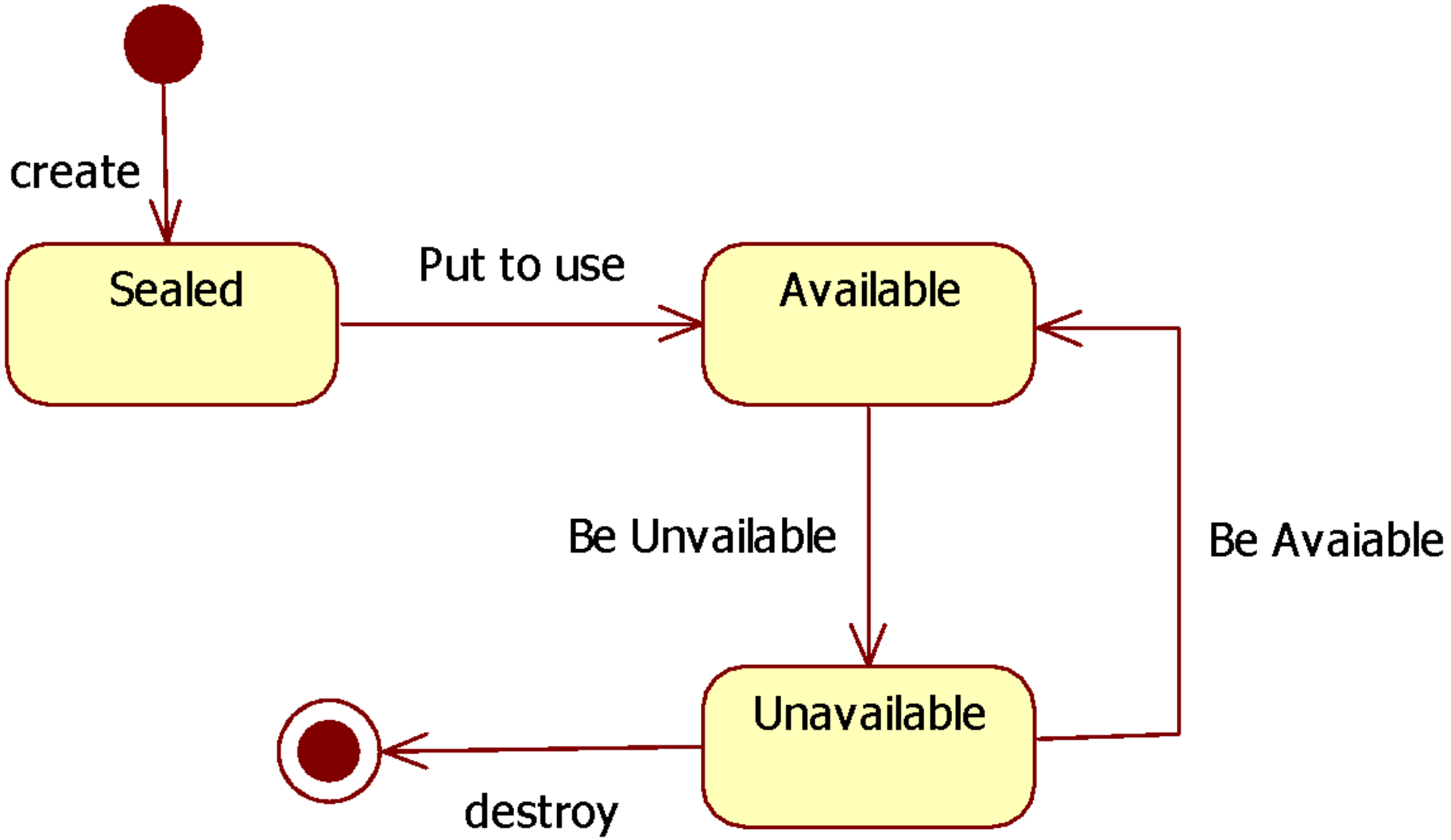


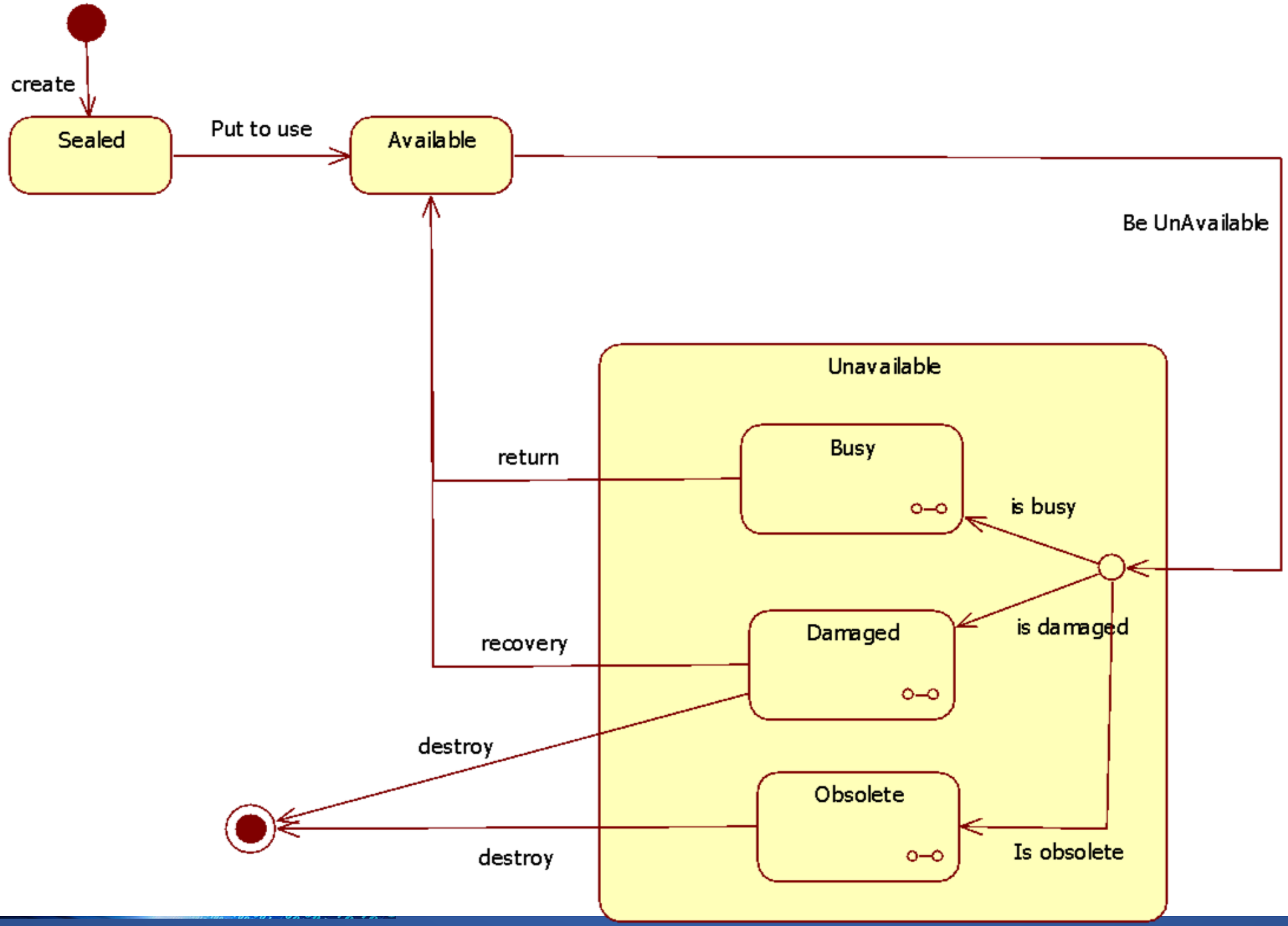
Conditions



Vẽ state diagram cho đối tượng “quyển sách” trong hệ thống Quản lý thư viện

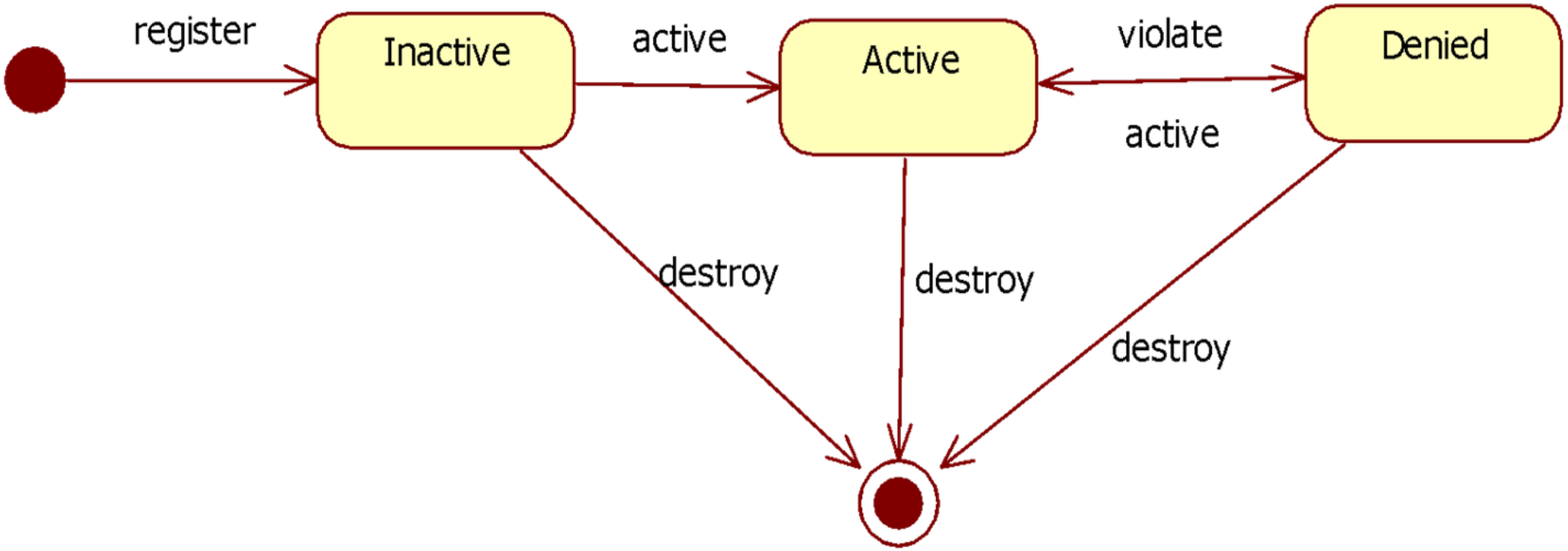
- Khi một quyển sách được nhập về, thủ thư sẽ nhập thông tin quyển sách vào cơ sở dữ liệu. Quyển sách này chưa được sử dụng cho đến khi thủ thư kích hoạt nó.
- Sau khi được thủ thư kích hoạt, quyển sách đó sẽ được đưa vào sử dụng (có thể cho mượn).
- Khi quyển sách được một đọc giả mượn, quyển sách đó không được các đọc giả khác mượn nữa cho đến khi quyển sách đó được trả.
- Khi một quyển sách bị hư hỏng, nó cũng sẽ không được cho mượn cho đến khi nó được khắc phục hư hỏng.
- Khi một quyển sách bị hư hỏng nặng hoặc lỗi thời, nó sẽ bị xóa khỏi hệ thống.

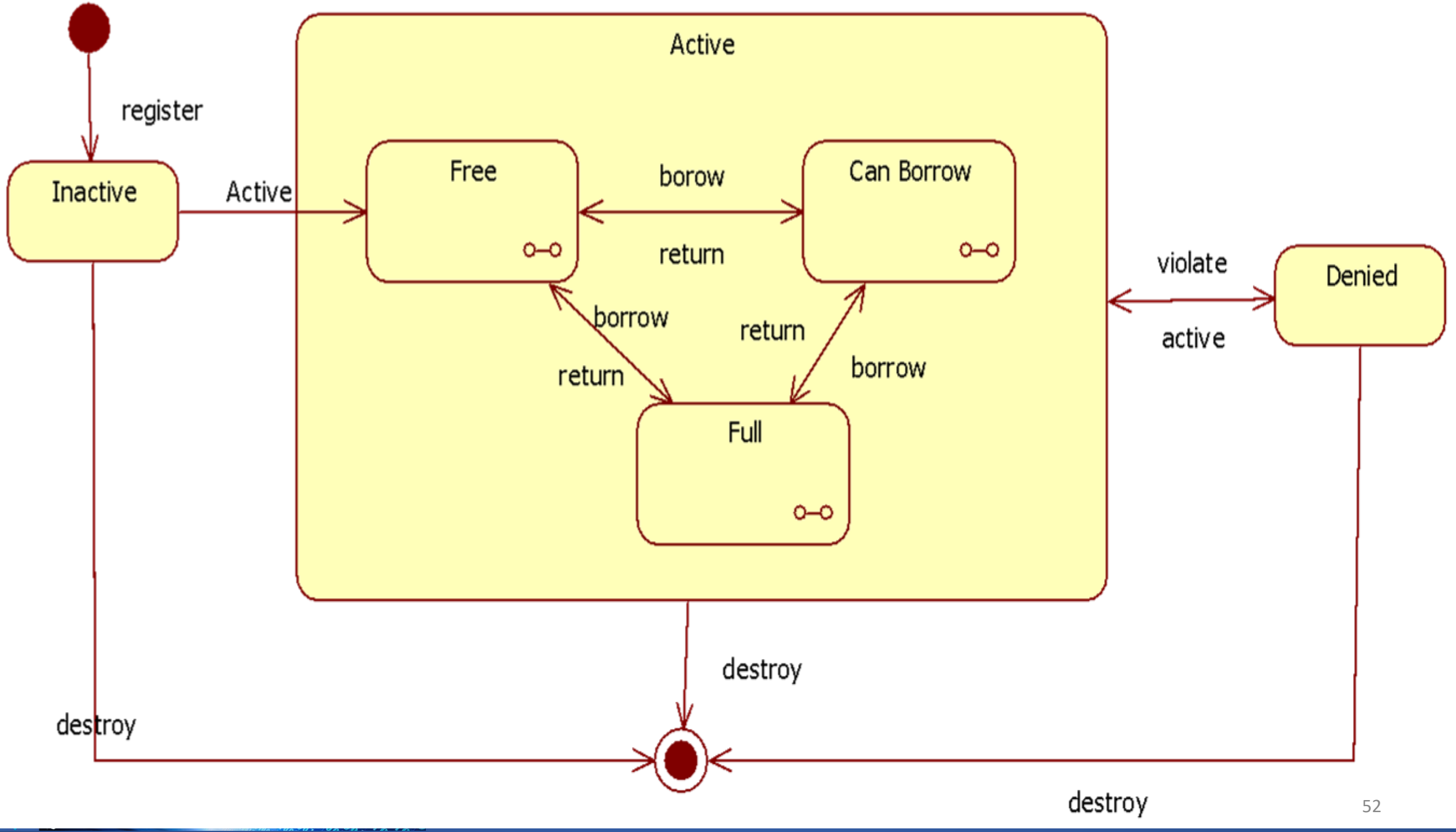

create
Sealed
Put to use
Available
Be Unvailable
Be Avaiable
Unavailable
destroy


create
Sealed
Put to use
Available
Be UnAvailable
Unavailable
Busy
return
is busy
Damaged
recovery
is damaged
destroy
Obsolete
destroy
Is obsolete

Vẽ state diagram cho đối tượng “đọc giả” trong hệ thống Quản lý thư viện

- Tài khoản đọc giả khi mới đăng ký sẽ được lưu trong cơ sở dữ liệu nhưng chưa được kích hoạt.
- Khi thủ thư kích hoạt tài khoản, đọc giả sẽ sử dụng được các dịch vụ của thư viện
- Đọc giả có thể mượn tối đa 5 quyển sách
- Đọc giả có thể bị khóa tài khoản nếu như vi phạm nội quy thư viện
- Thủ thư có thể mở khóa tài khoản đọc giả đang bị khóa
- Tài khoản đọc giả có thể bị hủy bởi chính đọc giả hoặc thủ thư.


register
Inactive
active
Active
violate
Denied
active
destroy
destroy
destroy


Active
register
Inactive
Active
Free
borow
Can Borrow
return
violate
Denied
borrow
return
active
return
borrow
Full
destroy
destroy
destroy

Phân tích dựa trên mẫu (pattern)

- Trong phân tích yêu cầu phần mềm, một số vấn đề có thể gặp lại nhiều lần ở các dự án phần mềm khác nhau.
  - *Ví dụ?*
  - → Tái sử dụng kết quả phân tích yêu cầu ở các trường hợp tương tự.

Phân tích dựa trên mẫu (pattern)

- Là cơ chế phân tích kiến thức về nghiệp vụ/yêu cầu phần mềm dựa trên các mẫu phân tích tương tự đã có trước đó.
- Kết quả phân tích yêu cầu/nghiệp vụ được thực hiện và lưu lại thành một bản mẫu.
  - Bản mẫu mô tả "rõ ràng" vấn đề chung mà mẫu có thể áp dụng, giải pháp được quy định, các giả định và hạn chế của việc sử dụng mẫu trong thực tế và một số thông tin khác như ưu điểm và nhược điểm của mẫu và tham chiếu đến một số ví dụ đã biết về việc sử dụng mẫu đó trong các ứng dụng thực tế.
- Bản mẫu đó được áp dụng cho những trường hợp tương tự về sau.

Phân tích dựa trên mẫu (pattern)

- Ưu điểm:
  - Tăng tốc độ phân tích
  - Tiết kiệm chi phí phân tích
  - Phân tích yêu cầu sâu sắc hơn
  - Tạo tạo thuận lợi cho việc chuyển đổi mô hình phân tích thành mô hình thiết kế bằng cách đề xuất các mẫu thiết kế và giải pháp đáng tin cậy cho các vấn đề phổ biến.

Mô hình hóa yêu cầu cho ứng dụng Web và Mobile

- Mức độ nhấn mạnh việc lập mô hình yêu cầu cho ứng dụng Web và thiết bị di động tùy thuộc vào các yếu tố liên quan đến kích thước sau:
  - Kích thước và độ phức tạp của ứng dụng
  - Số lượng các bên liên quan (phân tích có thể giúp xác định các yêu cầu xung đột nhau của các bên)
  - Quy mô của nhóm phát triển ứng dụng
  - Mức độ mà các thành viên trong nhóm đã làm việc cùng nhau trước đây (phân tích có thể giúp phát triển sự hiểu biết chung về dự án)
  - Mức độ thành công của tổ chức phụ thuộc trực tiếp vào thành công của ứng dụng.

Mô hình hóa yêu cầu cho ứng dụng Web và Mobile

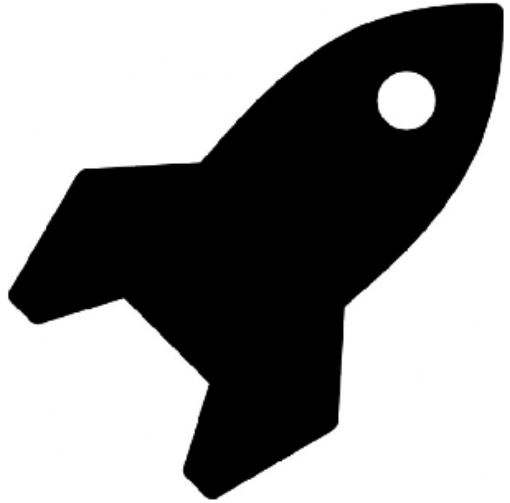
5 lớp mô hình:

- Mô hình nội dung: nội dung/dữ liệu của ứng dụng.
  - Bao gồm văn bản, đồ họa và hình ảnh, video và dữ liệu âm thanh.
- Mô hình tương tác: cách người dùng tương tác với ứng dụng.
- Mô hình chức năng: các hoạt động có hoặc không có xử lý dữ liệu
- Mô hình điều hướng: điều hướng tổng thể cho ứng dụng.
- Mô hình cấu hình: môi trường và cơ sở hạ tầng của ứng dụng

Ứng dụng web: các tiếp cận

Web app

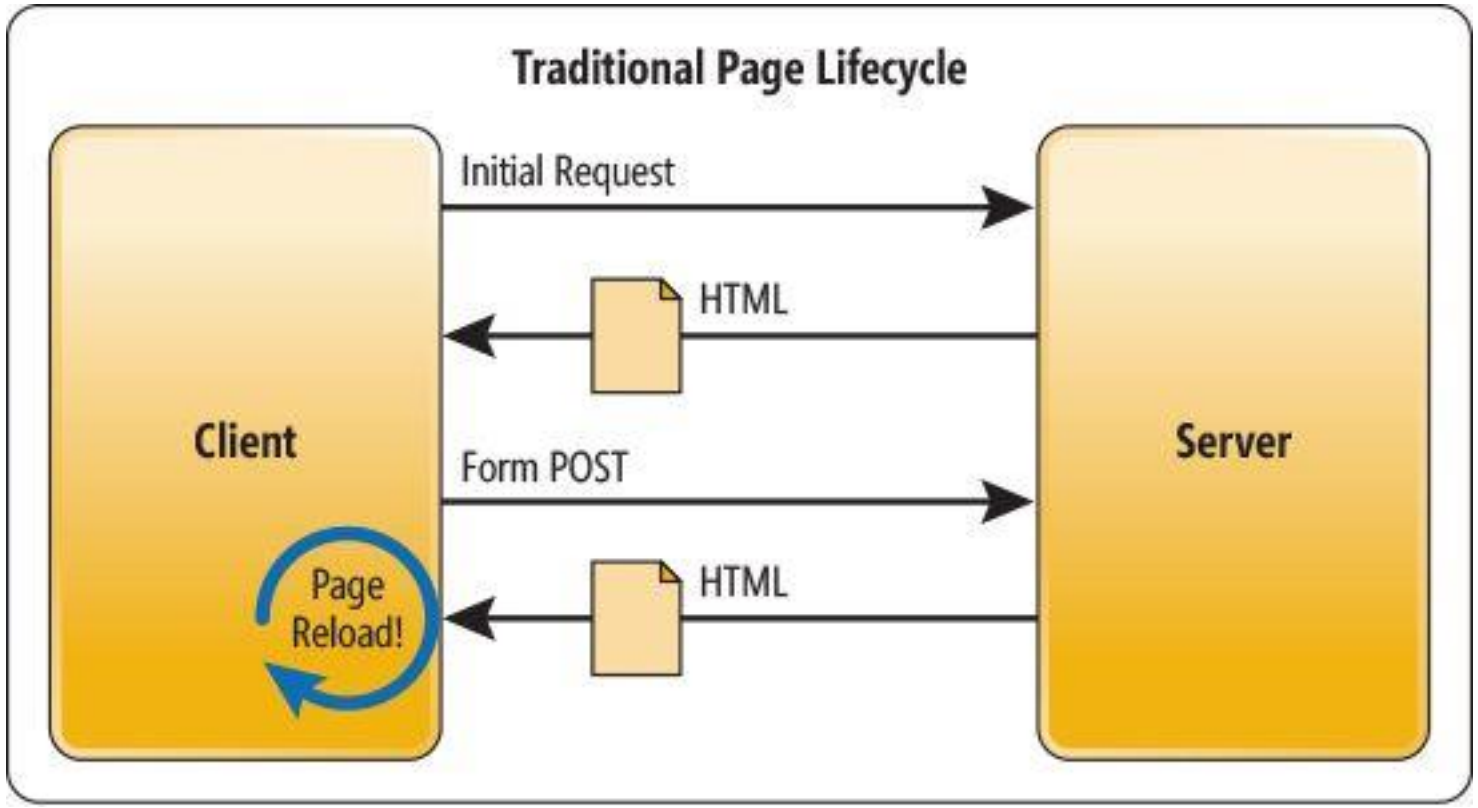
Single page app



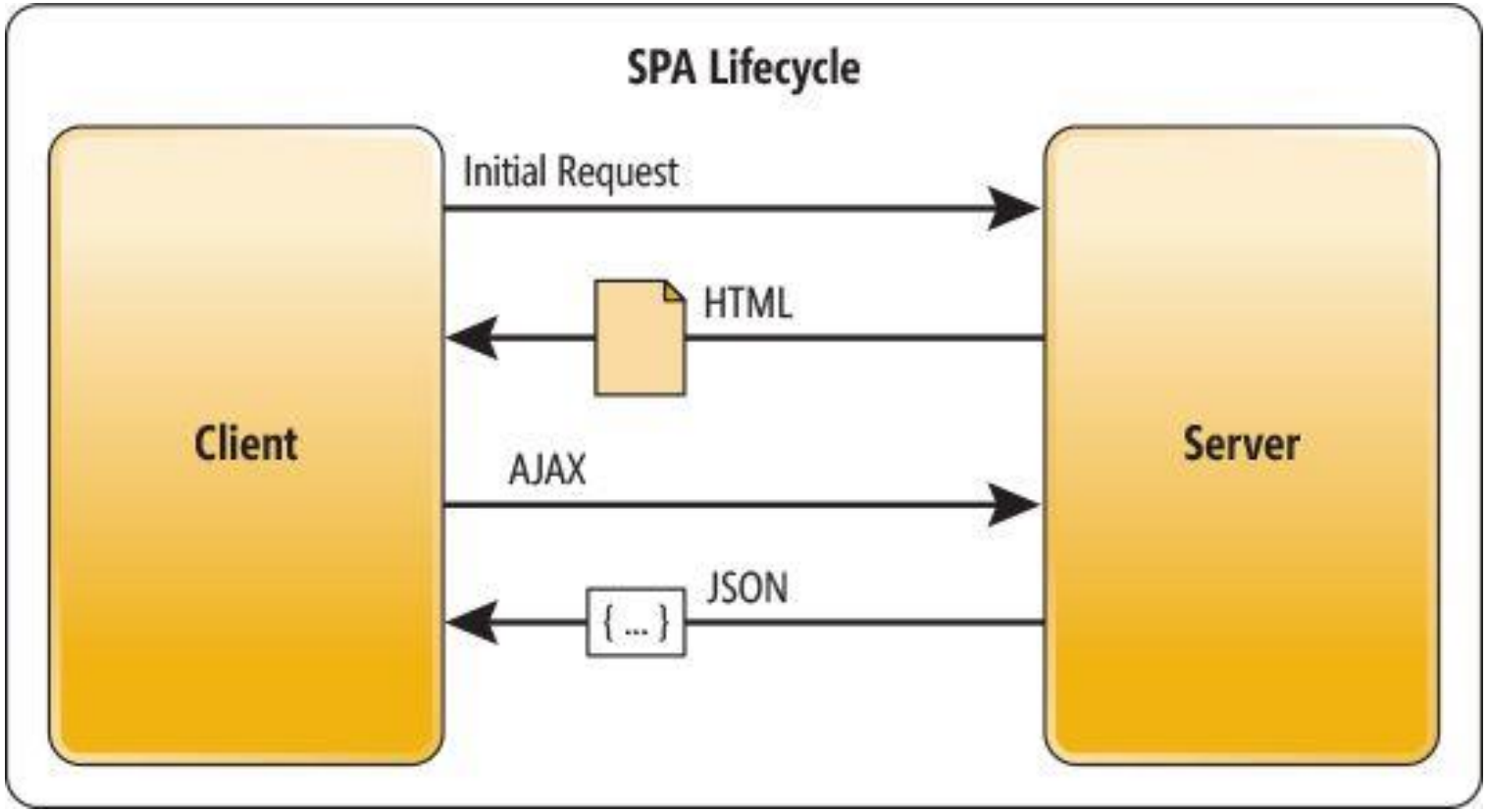
Progressive web app

https://lcdung.top/sanh-single-page-app-spa-vs-progressive-web-app-pwa/

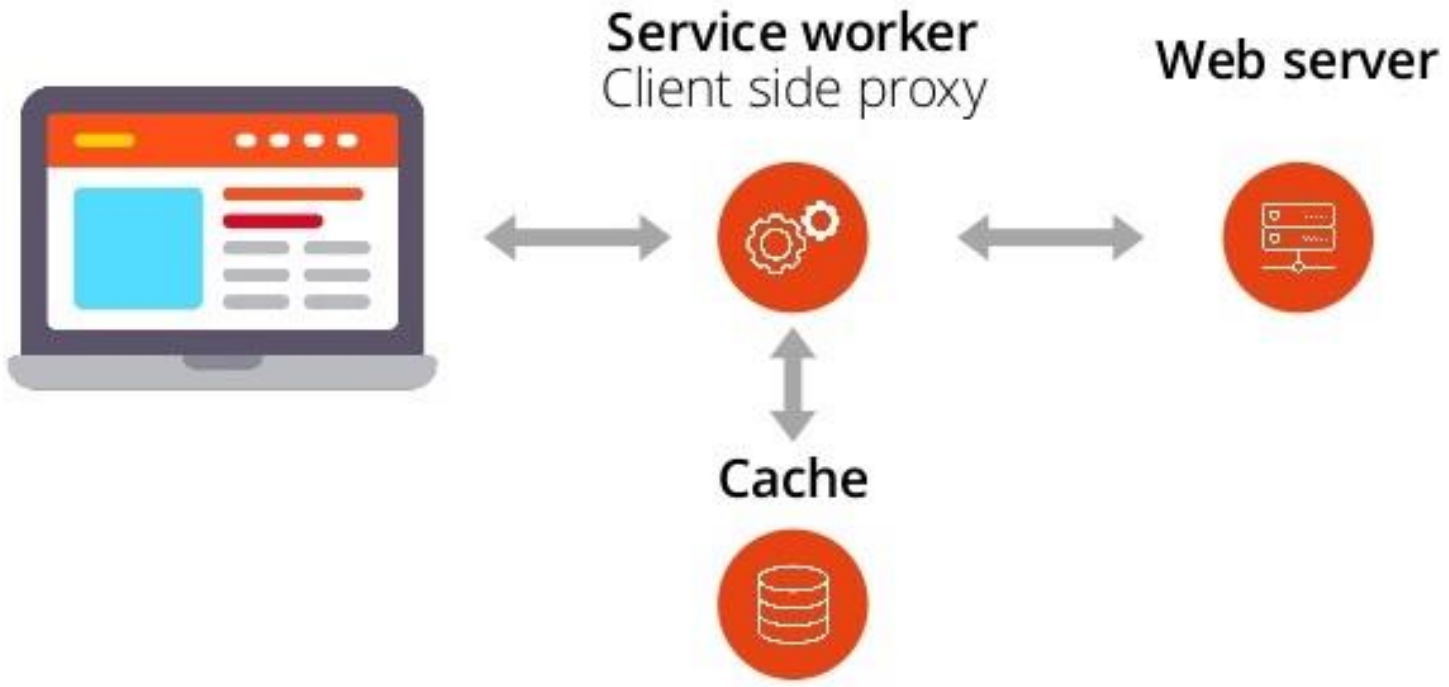
Ứng dụng Web truyền thống



Single Page App (SPA)



Progressive Web App (PWA)



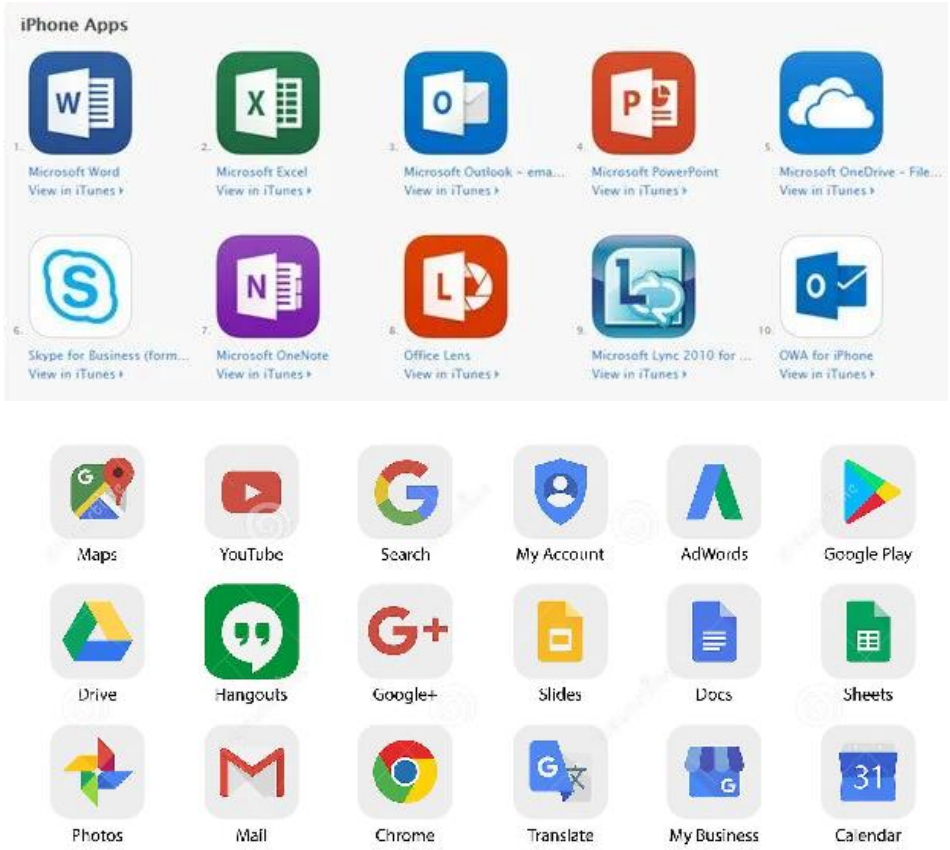
Resposive web design

- Sử dụng HTML và CSS để tự động thay đổi kích thước, ẩn, thu nhỏ hoặc phóng to trang web để làm cho trang web trông đẹp trên tất cả các thiết bị (máy tính, máy tính bảng và điện thoại)

```
<meta name="viewport"
content="width=device-width, initial-scale=1.0">
```

https://www.w3schools.com/html/html_responsive.asp

Ứng dụng mobile



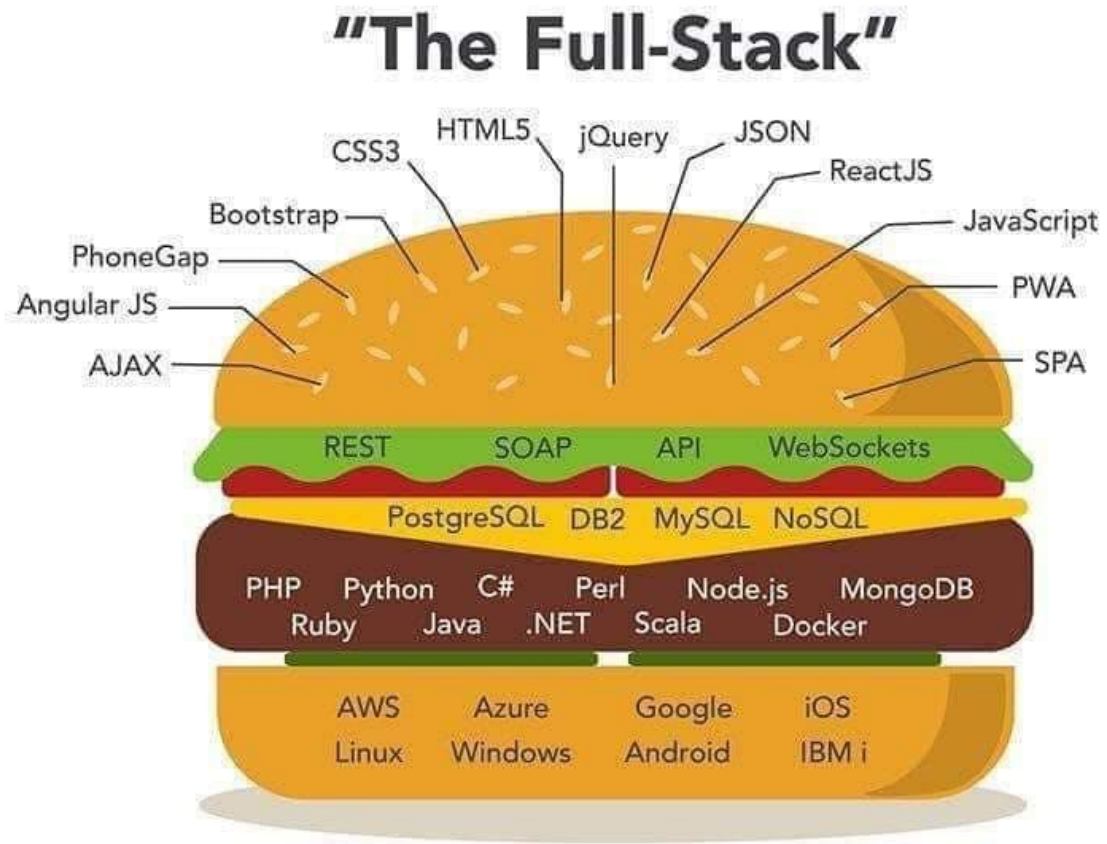
Mobile Apps vs. Mobile Website

Top 10 Cross Platform App Development Frameworks

- 1. Flutter
- 2. React Native
- 3. Ionic
- 4. PhoneGap
- 5. Xamarin
- 6. Native Script
- 7. Appcelerator Titanium
- 8. Qt
- 9. Felgo
- 10. Rho Mobile

https://www.macandro.com/blog/cross-platform-app-development-frameworks-2021

FULL-STACK PROGRAMMING



SMAC/SMACS/SMACT

- Social: tiếp cận nhiều đối tượng
- Mobile: thiết bị được sử dụng nhiều nhất trên thế giới
- Analytics: đã và đang thay đổi thế giới kinh doanh

SMACS: Security (Defined by IBM)
SMACT: IoT (Things)

- Cloud: lưu trữ, quản lý và xử lý dữ liệu trên đám mây internet
- Security: Bảo mật
- Things: Tất cả mọi thứ

https://www.techfunnel.com/information-technology/how-smac-technology-can-revolutionize-your-business/

Tóm tắt

- Mô hình hóa hành vi trong quá trình phân tích yêu cầu mô tả hành vi động của phần mềm.
- Các mẫu phân tích cho phép kỹ sư phần mềm sử dụng kiến thức miền hiện có để tạo thuận lợi cho việc tạo mô hình yêu cầu.
- Yêu cầu lập mô hình cho các ứng dụng di động và WebApps: nội dung, tương tác, chức năng, điều hướng và cấu hình.