NỘI DUNG VIẾT BÁO CÁO ĐỀ TÀI XÂY DỰNG ỨNG DỤNG

Một ứng dụng (application) là một phần mềm được sử dụng trong một ngữ cảnh đã được xác định từ thực tế, khác với phần mềm minh họa cho một nghiên cứu khoa học (nhận dạng, học máy,…).

Nguyên tắc chung cho việc đọc tài liệu hướng dẫn:

- Nhận thức: hiểu đúng bản chất công việc cần làm.
- Cách làm: từng bước thao tác, phương pháp, công cụ.
- Cách trình bày: viết thành báo cáo, vẽ sơ đồ, mô tả dữ liệu.
- Liên kết các bước: sản phẩm của bước trước là đầu vào cho bước sau.
  Ví dụ: Use Case → Sequence → chức năng phần mềm.
- Minh họa xuyên suốt: dùng usecase "Giao hàng".

I. Giới thiệu đề tài

Đây là phần giới thiệu ngắn gọn về đề tài.

1. Mục đích (nhu cầu sử dụng ứng dụng): Ứng dụng này dùng để làm gì, ích lợi của nó cho đối tượng (tổ chức sử dụng / người dân / chính phủ..) như thế nào ?
2. Mục tiêu (kết quả cần đạt được): Ứng dụng này phải giải quyết được vấn đề gì (chỉ nêu một vài vấn đề chính từ mục đích), bằng các chức năng, dịch vụ gì của ứng dụng (PM của đồ án), ví dụ: PM quản lý kho giúp cho thủ kho kiểm soát các tiến trình nhập/xuất/tồn kho hiệu quả, PM bán hàng giúp khách hàng chọn mua hàng (tìm hàng), giúp công ty kiểm soát được quá trình xử lý đơn (giám sát giao hàng, xử lý ngoại lệ),...
3. Phương pháp tiến hành:
   a) *Tìm hiểu hiện trạng* phát sinh nhu cầu dùng PM của một tổ chức/cá nhân/chính phủ (các vấn đề cần giải quyết, trong số đó PM của đồ án được dùng như một công cụ hỗ trợ giải quyết vấn đề).
   b) *Tìm hiểu các nghiệp vụ/quy định* mà tổ chức và ứng dụng PM phải tuân thủ .
   c) *Tìm hiểu các mô hình/phương pháp/giải thuật* … từ kho kiến thức đã biết (sách, công trình nghiên cứu được đăng trong các tạp chí khoa học kỹ thuật, các sản phẩm công nghệ được công bố rộng rải (các chuẩn của ISO/CMM/IEEE/..., open source được cấp licence trên GIT,...)
   d) *Phân tích, thiết kế, hiện thực, đánh giá*...

II. Cơ sở khoa học của đề tài

Cách thực hiện: Trình bày các nội dung đã tìm hiểu (theo yêu cầu của đề cương, phần lý thuyết), như tìm hiểu quy trình nghiệp vụ, các quy định/quy tắc quản lý/quy trình tại nơi sẽ sử dụng ứng dụng, công nghệ hỗ trợ,...

Chú ý : tìm hiểu công nghệ là tìm hiểu và trình bày ***cách sử dụng công nghệ đó cho việc phát triển ứng dụng của đề tài***, không phải là giới thiệu nó. Câu hỏi phải trả lời cho chính xác là: *phần mềm ứng dụng của đề tài dùng công nghệ đó để làm gì, dùng như thế nào, ưu điểm của nó so với các công nghệ tương tự khác là gì (so sánh, chọn lựa)*.

Do đó, mặc dù được trình bày trước phần phân tích & thiết kế hệ thống, nhưng việc tìm hiểu công nghệ *được thực hiện trong lúc tìm giải pháp thiết kế hệ thống (sau khi đã định nghĩa rõ yêu cầu đối với phần mềm)*.

III. Phân tích hệ thống

1) Bối cảnh/hiện trạng của hệ thống / tổ chức (= môi trường có nhu cầu dùng PM)

Nhận thức: Hệ thống trong mục này là gồm tất cả những đối tượng (con người/thiết bị/phần mềm,..) có cộng tác với nhau (nhờ giúp đỡ lẫn nhau) để thực hiện mục đích/nhiệm vụ của một tổ chức (công ty, trường, bệnh viện,..). Những vấn đề (tình huống) gây khó khăn cho việc thực hiện nhiệm vụ của tổ chức là mục tiêu cần giải quyết của tổ chức đó (và của phần mềm sẽ xây dựng trong đề tài). Một phần mềm ứng dụng sẽ được xây dựng (gọi tắt là PM) sẽ là một thành phần/bộ phận của tổ chức, mà nhiệm vụ của nó là cộng tác với đối tượng khác có liên quan giúp cho tổ chức giải quyết tốt các tình huống của nó. Khi chưa có PM của đề tài, tổ chức vẫn tồn tại và đang hoạt động với một số tình huống có khó khăn (là vấn đề cần giải quyết); việc phân tích hệ thống là định nghĩa ra yêu cầu đối với hệ thống PM sẽ xây dựng, tức là định nghĩa vai trò/nhiệm vụ của PM trong việc giải quyết các tình huống khó khăn của tổ chức (usecases của PM), và làm rõ nhiệm vụ của PM thành các tương tác cần phải có của PM với các đối tượng khác có liên quan (actors) dựa trên mối quan hệ cộng tác được đề xuất từ đề tài (là mô hình vận hành mới của tổ chức, có sử dụng PM của đề tài). Các tương tác trên PM đòi hỏi PM (hoặc các thành phần của nó) phải cung cấp các chức năng xử lý cho từng tương tác (methods): đây chính là yêu cầu chức năng cho PM.

Cách thực hiện: Bắt đầu tìm hiểu những đối tượng (con người/thiết bị/ứng dụng khác/... có vai trò/nhiệm vụ cụ thể trong hệ thống, ví dụ: thủ kho, giao dịch viên, shippers) đang hợp tác cùng nhau để giải quyết những tình huống/vấn đề nào đó của tổ chức (vd: quản lý kho, bán hàng, giao hàng,..), để thấy yêu cầu trong cách giải quyết tình huống.

Vd: với mục đích bán hàng on-line, thì các tình huống cần giải quyết là :

- Nhận đơn đặt hàng (đối tượng: khách hàng, giao dịch viên, điện thoại / website),
- Xuất hàng cho đơn (đối tượng : thủ kho, giao dịch viên, shipper),
- Đi giao hàng (đối tượng : shipper, khách)
- Thu tiền (đối tượng : shipper, khách, thủ quỹ).

Chúng ta chỉ quan tâm đến các tình huống thực tế của tổ chức mà PM có khả năng trợ giúp giải quyết. Đây là yếu tố khẳng định tính thực tiễn của đề tài: Nếu nhận thức hiện trạng không đúng thực tế (vd: quá lạc hậu so với thực tế) thì đề tài sẽ không có giá trị sử dụng (PM làm ra sẽ không dùng được). Cần làm rõ:

a) Mục tiêu mà PM sẽ giải quyết

Trình bày được những tình huống thực tế đưa đến việc cần làm PM, từ đó định nghĩa mục tiêu cụ thể cho đề tài (phân tích, thiết kế PM có thể tham gia xử lý các vấn đề đã nêu).

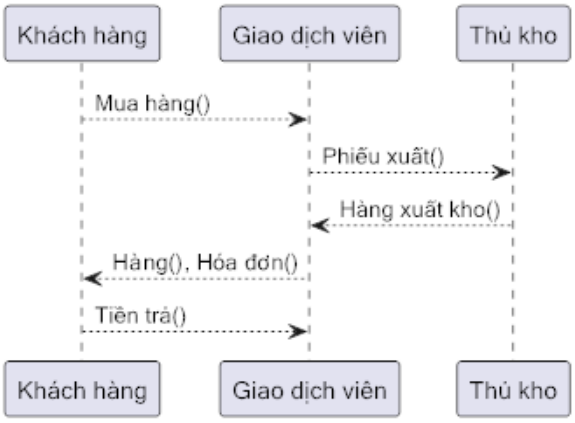
b) Hiện trạng/bối cảnh của tổ chức trước khi có PM (mô hình vận hành hiện tại của tổ chức)

Nhận thức: Theo các mô hình làm PM (SoftWare Development Models), thì đây là kết quả của công đoạn "khảo sát hiện trạng", để làm rõ bối cảnh phát sinh nhu cầu dùng PM (cũng là môi trường hiện thực giải pháp áp dụng PM của đề tài), để từ đó thực hiện tiếp các công đoạn phân tích (định nghĩa cụ thể yêu cầu chức năng, phi chức năng và nhiệm vụ của PM), thiết kế (chỉ ra cách xử lý bên trong PM), hiện thực (viết code, mua thiết bị, tuyển dụng..), triển khai áp dụng,...

Cách trình bày: mô tả hiện trạng xử lý các tình huống của tổ chức (mô hình vận hành của tổ chức), có thể vẽ mô hình này bằng lược đồ cộng tác (collaboration/ communication), hoặc lưu đồ (work-flows, quy trình), và mô tả thêm bằng lời cho rõ nghĩa. Lược đồ này không được dùng cho việc định nghĩa yêu cầu làm phần mềm. Trong mô tả này, các đối tượng trong hệ thống (tổ chức) phải được gọi tên theo vai trò của chúng trong tổ chức, chỉ dùng để kết luận sơ lược về các vấn đề/khó khăn của tổ chức khi nó vận hành theo mô hình này.

Ví dụ:

+ Bán hàng truyền thống: Khách đến cửa hàng → nói yêu cầu với nhân viên → nhân viên bán hàng ghi phiếu → thủ kho xuất hàng → nhân viên bán hàng giao hàng → thu tiền:



Khó khăn: giấy tờ thủ công, thiếu truy vết, sai sót tồn kho,..

c) Đề xuất giải pháp dùng PM của đề tài (mô hình vận hành mới của tổ chức có dùng PM)

Nhận thức: Từ hiện trạng được mô tả ở trên, em nhận định một số tình huống vấn đề mà PM có thể tham gia giải quyết cùng với các loại nguồn lực khác đã có (con người, công cụ), và đề xuất mô hình vận hành mới cho tổ chức bằng lược đồ cộng tác/tuần tự, để chỉ ra cách trợ giúp của PM, phác thảo cách sử dụng PM (như 1 công cụ) trong tổ chức, và nêu rõ lợi ích từ việc sử dụng PM (cho tổ chức/cho những tượng nào). Đây là bước phác thảo nhiệm vụ của PM trong tổ chức. Nội dung chính của lược đồ này là các thông điệp mang nội dung thông tin cụ thể (hoặc món đồ vật lý cụ thể) được gửi/nhận, ie: phải diễn tả cái gì được gửi/nhận từ đâu/đến đâu, để mô tả rõ các tương tác cần thiết giữa các đối tượng. Nội dung thông điệp (và quan hệ) phải phù hợp với khả năng gửi/nhận của các đối tượng.

Cách thực hiện:

Trong ví dụ Use Case "Giao hàng" : Thiếu truy vết trạng thái, không gom lý do thất bại, khó lên lịch giao lại, không đồng bộ tồn kho giữ chỗ,... Với tình huống được phác thảo trong hiện trạng, thì em suy nghĩ xem PM/máy tính có giúp được gì không, coi PM như một công cụ (sẽ làm ra) để nó tham gia (cộng tác) cùng với các đối tượng khác giải quyết các tình huống khó khăn này. Nếu có, các tình huống đó là usecase của PM: một usecase là một tình huống mà PM được vài đối tượng khác sử dụng (là các actors) để nhận được sự trợ giúp thực tế từ PM, ví dụ như:

- Usecase "tìm kiếm hàng": khách hàng là actor, là người cần dùng PM để tìm ra món hàng cần mua,
- Usecase "liệt kê các đơn giao hàng bị trục trặc": actor là người quản lý cần biết các đơn hàng bị ách tắc để đưa ra cách xử lý
- Usecase "Nhập thêm hàng vào kho": actor là thủ kho cần dùng PM để tính số lượng hàng bổ sung thêm cho kho, liên lạc với nhà cung cấp, giám sát hàng nhập kho...

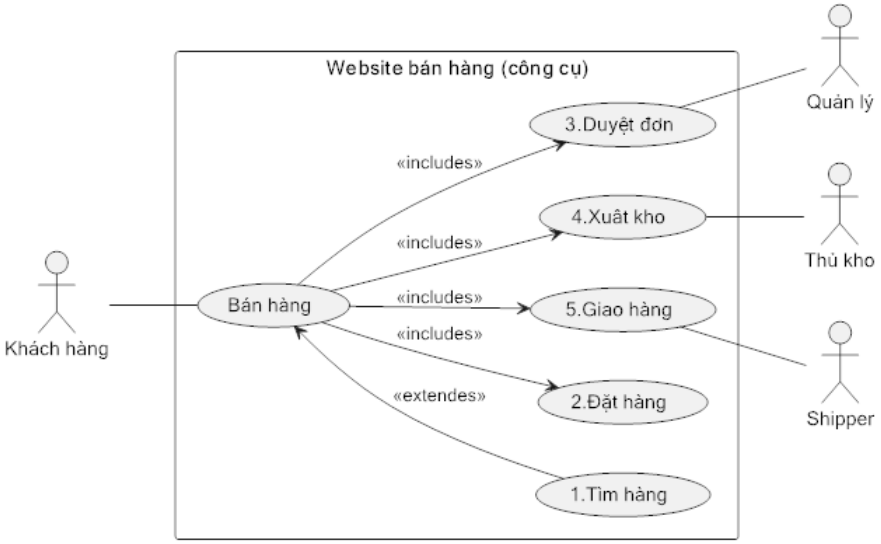
Cách trình bày:

- Những actors của usecase vẫn là những đối tượng có tên và vai trò đã biết trong phần hiện trạng (vd: "khách hàng","người quản lý", "thủ kho",…). Một số đối tượng và quan hệ trong mô hình cũ có thể không xuất hiện vì không còn cần thiết nữa, ví dụ: "người nhân viên thu ngân" trong mô hình thanh toán tiền trực tuyến.
- Tên gọi có ý nghĩa thực tế của usecase sẽ giúp ta hệ thống hóa các usecase và chỉ ra các mối quan hệ giữa chúng: dùng các loại quan hệ: tổng quát hóa ("là"), <<includes>> ("gồm") và <<extendes>> ("đôi lúc cần thêm").
- Vì lược đồ usecase trong phân tích được dùng để định nghĩa các tình huống hỗ trợ của PM (PM là công cụ được sử dụng), nên **lược đồ này sẽ không có**:
- Actor hỗ trợ cho PM, như "admin", "database", "user", "máy POS", "Barcode Reader",... là những đối tượng "phần cứng, công nghệ" được chọn để thiết kế PM. Vai trò của các actor này là giải quyết các vấn đề của PM (phân quyền, lưu trữ,...) chứ không phải của tổ chức, chúng chỉ có ý nghĩa khi thiết kế PM (xem PM là một hệ thống).
- Usecase "login" vì "login" là một thủ tục được đòi hỏi từ PM cho việc bảo mật tài nguyên của PM; nó không phải là nhu cầu mong đợi của người sử dụng.
- Actor "Admin" và "User". Chỉ có thể xuất hiện trong lúc thiết kế PM. Các actor trong lược đồ phân tích là phải là các vai trò đã biết rõ trong tổ chức (giống như lúc PM chưa ra đời), như "thủ kho", "shipper", "khách",..

Ví dụ: + Bán hàng online bằng Website bán hàng

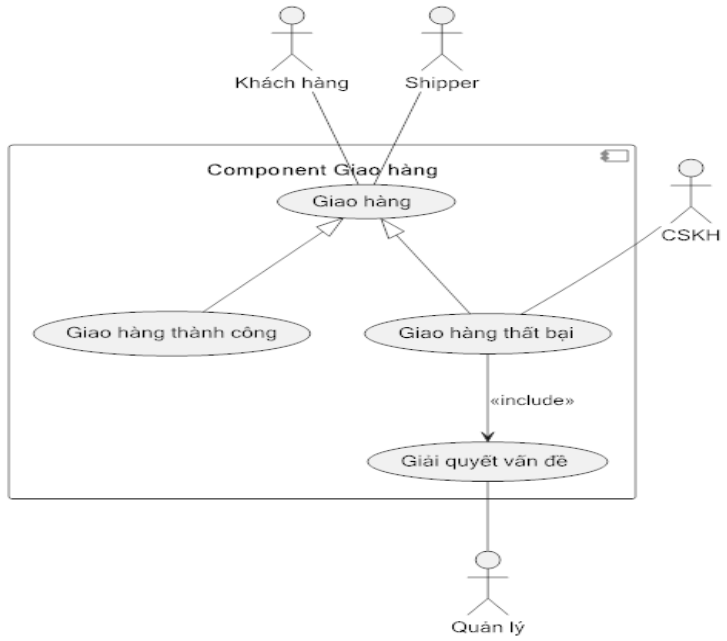
Actors: Khách hàng, Thủ kho, Shipper, Quản lý (không có Giao dịch viên)

Use Cases: Tìm hàng, Đặt hàng, Duyệt đơn, Xuất kho, Giao hàng



Ví dụ 2: Tình huống Giao hàng trong bán hàng online:

Actors: Shipper (giao hàng), Khách hàng (nhận hàng), Nhân viên CSKH (giám sát việc giao hàng), Quản lý (xử lý trục trặc)



Cách dùng lược đồ usecase:

Xem xét hiệu chỉnh các usecase cho phù hợp với thực tế, bỏ bớt những usecase không mong muốn từ actors (ví dụ: login).

2) Định nghĩa các chức năng của PM (xem PM là một công cụ của tổ chức)

Nhận thức:

- Tên của một usecase chỉ là một ý niệm khái quát về tình huống (giống như tên của mỗi thực thể trong lược đồ ERD) mà ta cần làm rõ nội dung của nó trong thực tế, để xác định trong đó PM cần cung cấp các chức năng gì cho các actors. Như vậy mỗi usecase cần có thêm nhiều tài liệu để : mô tả thông tin cộng tác (ược đồ tuần tự/cộng tác), diễn tả trình tự các bước thực hiện (lược đồ hoạt động), diễn tả mối liên hệ giữa các đối tượng được nhắc đến trong usecase (lược đồ class), mô tả trạng thái hợp lệ của các đối tượng này (lược đồ trạng thái). Tùy theo ngữ cảnh, không cần phải vẽ tất cả các lược đồ trên, nhưng tối thiểu phải có lược đồ tuần tự để chỉ ra các tương tác của PM.
- Một usecase không phải là một chức năng của PM, vì để giải quyết một usecase, PM cần (nhiều) tương tác với (nhiều) tác nhân của nó. Mỗi tương tác trên PM thể hiện thành một cặp thông điệp request-response trong lược đồ tuần tự, đó là yêu cầu tương tác trên PM.

- Vì PM không thể xử lý được thông tin (ý niệm) như con người, nó cần có dữ liệu vào/ra cụ thể (có cấu trúc, có quy ước) cho các chức năng xử lý tương tác của nó, do đó, nội dung của thông điệp vào ra trên PM phải có mang dữ liệu (trong thông số của thông điệp).
- Mỗi chức năng của PM được định nghĩa từ cặp thông điệp vào/ra có dạng:
  [điều kiện xuất hiện] **<tên_thông_điệp>** (bộ dữ liệu của thông điệp).

Cách trình bày:

a) Vẽ lược đồ usecase gồm các usecase, actors và quan hệ actor-usecase, actor-actor, usecase-usecase. Lược đồ này giúp ta phân biệt các usecase, tránh trùng lặp hoặc dư thừa usecase.

b) Với mỗi usecase trong lược đồ:

- Nêu rõ ý nghĩa của nó: nêu rõ tình huống mà actor cần PM trợ giúp.
- Mô tả chi tiết kịch bản tương tác giữa actors với PM bằng lược đồ tuần tự, trong đó, thông điệp vào/ra trên PM phải có dữ liệu trong thông số.
- Mô tả thêm cho nó (nếu cần) bằng: lược đồ hoạt động (trình tự thủ tục xử lý), lược đồ lớp (các quan hệ "bất biến" giữa các lớp đối tượng), lược đồ trạng thái (điều kiện xuất hiện của các trạng thái hợp lệ)
- Các lược đồ mô tả cho 1 usecase phải nhất quán nhau, tối thiểu là tên của các lớp.

Ví dụ:

+ Đặc tả usecase "Giao hàng" bằng lời:

Mục đích: Bàn giao đơn cho người nhận, ghi nhận POD hoặc lý do thất bại.

Actor chính: Shipper.

Liên quan: Khách hàng (ký nhận), CSKH (tiếp nhận thất bại), Quản lý (lên lịch).

Tiền điều kiện: Đơn ở trạng thái ALLOCATED và đã bàn giao cho Shipper (handover).

Hậu điều kiện (thành công): Đơn DELIVERED, lưu POD (ảnh/chữ ký, thời điểm).

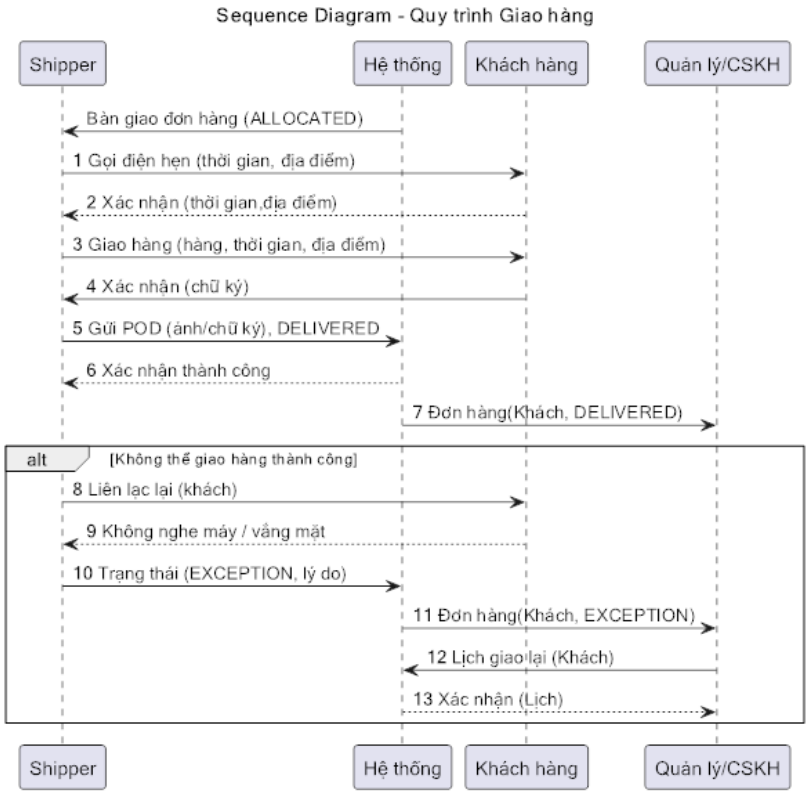
Hậu điều kiện (thất bại): Đơn EXCEPTION, có reason, có nextAttemptAt nếu lên lịch lại.

Luồng chính: Shipper mang hàng → gọi hẹn gặp người nhận → xác minh người nhận → giao hàng → lấy POD → cập nhật hệ thống.

Luồng thay thế: Không gọi được/người nhận vắng mặt/địa chỉ sai → ghi reason → báo thất bại → CSKH/Quản lý lên lịch giao lại.

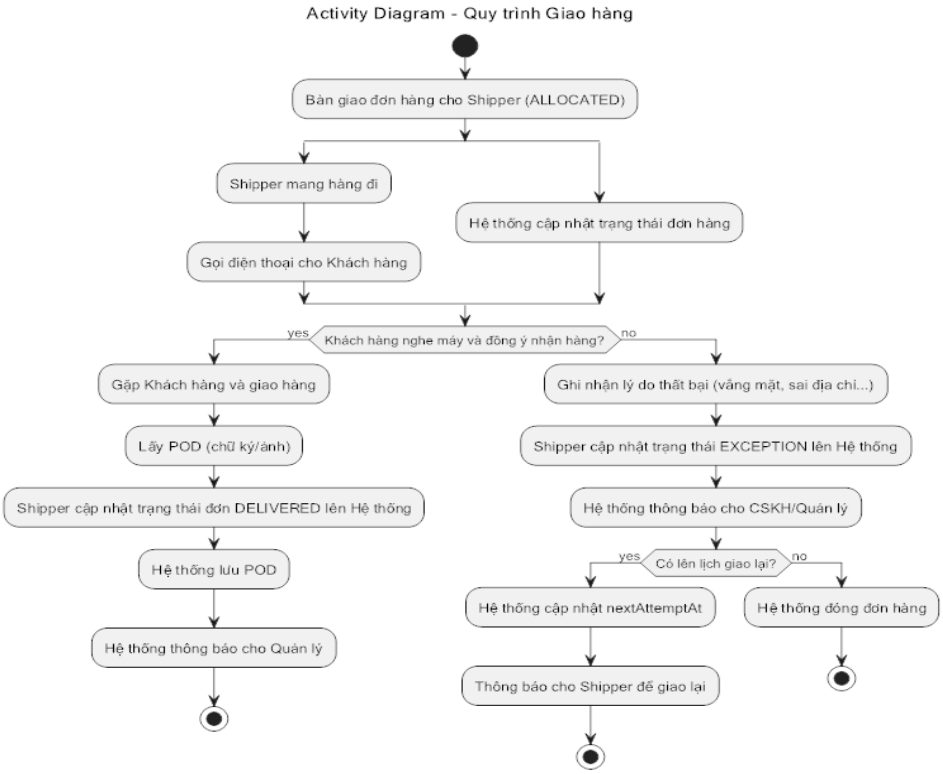
Đặc tả văn bản là cơ sở để kiểm tra Sequence diagram.

Lược đồ tuần tự cho Giao hàng:



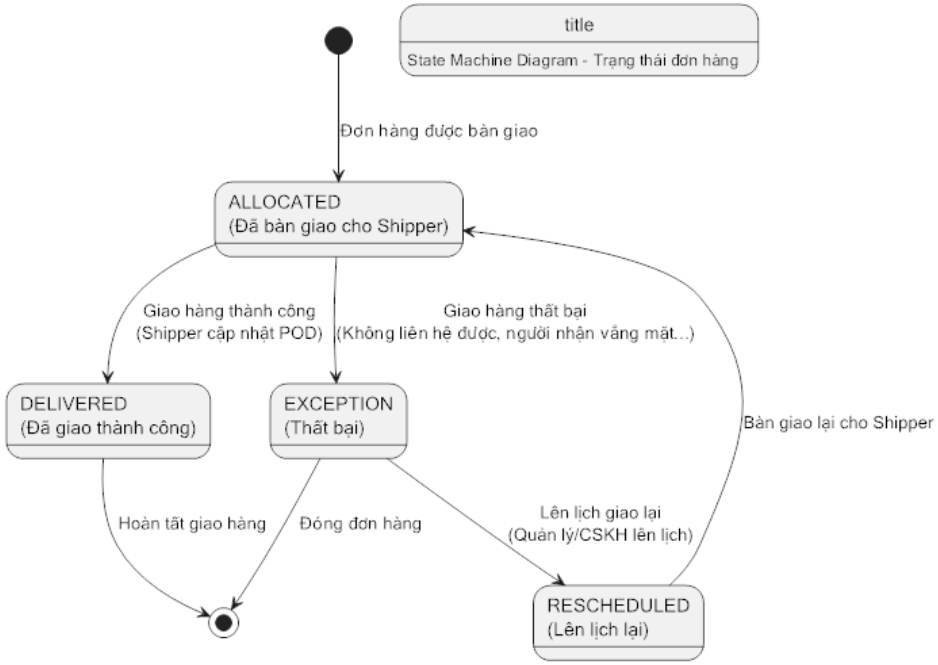
(Mọi thông điệp đều có tham số dữ liệu. Từ lược đồ sequence này, ta sẽ rút ra các chức năng/API cần phải có của PM).

+Activity Diagram cho "Giao hàng":



(Để đối chiếu activity với sequence diagram để kiểm tra độ phủ kịch bản và xác định thêm luồng ngoại lệ nếu thiếu).

+State Diagram cho "Giao hàng":



(Ràng buộc chuyển trạng thái sẽ thành trigger/ constraint trong CSDL và rule trong lớp dịch vụ)

3) Định nghĩa yêu cầu cho từng đối tượng thành phần của PM (xem PM là một hệ thống)

a) Phác thảo các đối tượng thành phần cần thiết của PM cho từng usecase

Cách thực hiện: Xem xét tất cả các lược đồ mô tả cho từng usecase (lược đồ tuần tự /cộng tác, lược đồ hoạt động, lược đồ class, lược đồ trạng thái,...) để tìm ra các thành phần cần thiết của PM có khả năng tham gia vào từng usecase, ví dụ bằng kỹ thuật CRC (class, responsibilyties và collaborators), kỹ thuật phân tích kịch bản của usecase,..

Ví dụ: Chúng ta có thể phân rã hệ thống quản lý giao hàng (WebApp) thành 3 thành phần (component) chính để xử lý các nghiệp vụ khác nhau:

1. API Gateway (hoặc Mobile App API): Điểm tiếp nhận mọi yêu cầu từ bên ngoài (Shipper App, Web Portal) và điều hướng đến các service phù hợp.
2. Order Service: Chịu trách nhiệm quản lý trạng thái, thông tin và vòng đời của đơn hàng.
3. Notification Service: Gửi các thông báo đến các bên liên quan (Quản lý/CSKH).
4. Database: Nơi lưu trữ toàn bộ dữ liệu về đơn hàng và POD.

b) Xác định các trợ giúp cần thiết cho PM để có giải pháp khả thi

Nhận thức: Mỗi đối tượng thành phần của PM có nhiệm vụ tham gia xử lý một (vài) usecase, và đôi khi nó cũng cần trợ giúp từ các actor bên ngoài PM, ví dụ, đối tượng "nhận tiền điện tử" tham gia xử lý usecase "trả online" cũng cần dùng "Payment gateway" như là 1 actor trợ giúp.

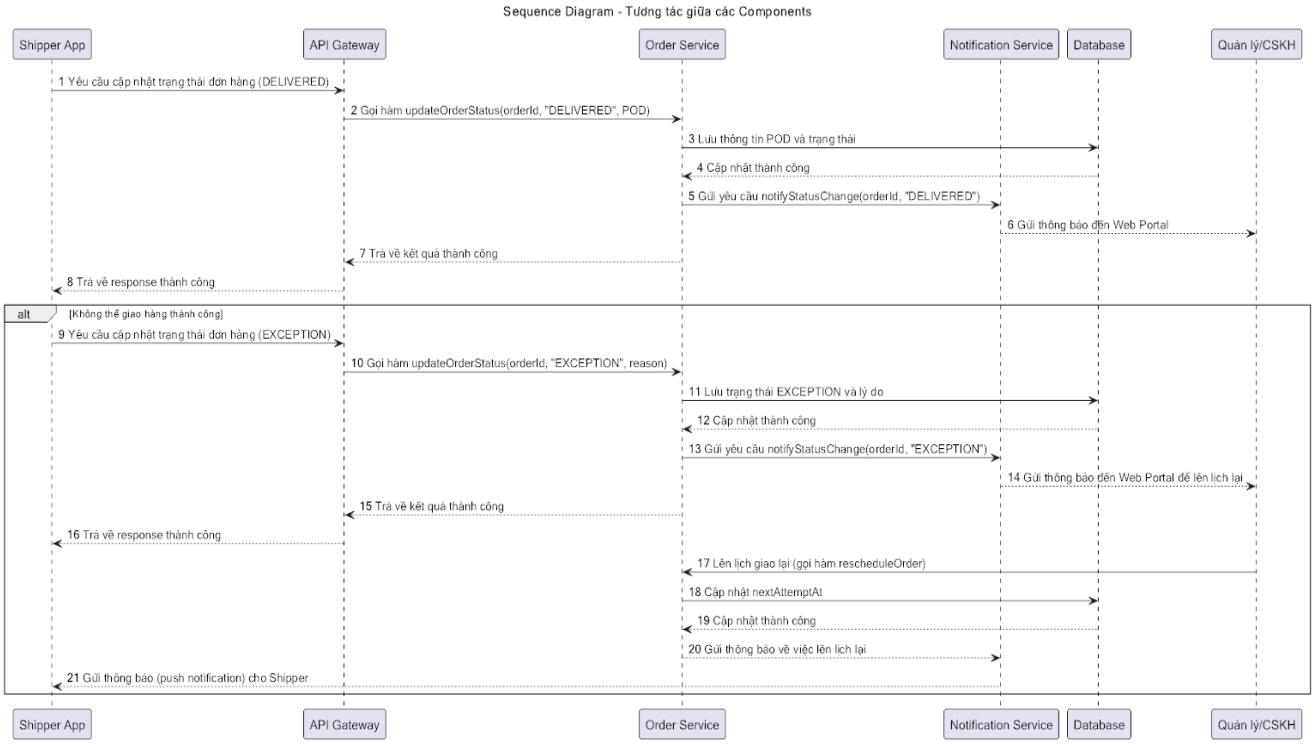
Cách trình bày: Từ lược đồ usecase phân tích, em thêm các usecases và các actors cần thiết để hỗ trợ cho PM (Create Account, Login, barcode reader, Payment Gateway,..) và nối mỗi usecase với thành phần của PM (dùng ký hiệu của class-hình chữ nhật) có tham gia vào usecase này.

c) Xác định các yêu cầu chi tiết (thuộc tính, phương thức) cho từng đối tượng thành phần

Nhận thức: Thuộc tính và phương thức của mỗi thành phần phải **nhất quán về ngữ nghĩa** với tất cả các lược đồ đã vẽ. Ví dụ: từ quan hệ "điều trị bệnh" giữa bác sỹ và bệnh nhân (trong lược đồ lớp), ta biết rằng đối tượng bác sỹ phải có một vài phương thức điều trị cho bệnh nhân, cụ thể là khám bệnh(), kê toa thuốc(), lập phác đồ điều trị(),..

Cách trình bày: mỗi đối tượng thành phần của PM được mô tả như một lớp (UML-class) chỉ gồm có 3 phần: tên, thuộc tính và hành vi, là các yêu cầu chức năng cho từng đối tượng thành phần, để PM có khả năng tham gia xử lý các usecases của nó.

Ví dụ: Lược đồ tuần tự thể hiện trên các thành phần của hệ thống:



Giải thích các tương tác trong lược đồ:

+ Shipper App ↔ API Gateway: Shipper không giao tiếp trực tiếp với các service mà thông qua một API Gateway. Điều này giúp bảo mật và quản lý các yêu cầu tốt hơn.

+ API Gateway → Order Service: API Gateway nhận yêu cầu từ Shipper App và chuyển tiếp (gọi) đến Order Service để xử lý logic nghiệp vụ.

+ Order Service ↔ Database: Order Service là thành phần duy nhất được phép tương tác trực tiếp với Database để đọc và ghi dữ liệu, đảm bảo tính nhất quán của dữ liệu.
+ Order Service → Notification Service: Khi có sự thay đổi quan trọng về trạng thái đơn hàng, Order Service sẽ gọi Notification Service để gửi thông báo đến các bên liên quan (ví dụ: Quản lý/CSKH).

Các thuộc tính và phương thức cần thiết cho mỗi thành phần của SYSTEM:

1. Thành phần API Gateway: Đây là điểm truy cập duy nhất, có trách nhiệm tiếp nhận và định tuyến các yêu cầu từ Shipper App.

Thuộc tính (Attributes):

api_key: Khóa API để xác thực yêu cầu từ Shipper App.

Phương thức (Methods):

+updateOrderStatus(orderId, status, POD_data): Nhận yêu cầu cập nhật trạng thái đơn hàng.
+getOrdersForShipper(shipperId): Lấy danh sách các đơn hàng được giao cho shipper.
+rescheduleOrder(orderId, newDate): Nhận yêu cầu lên lịch lại đơn hàng.

2. Thành phần Order Service: Đây là trái tim của hệ thống, quản lý toàn bộ vòng đời và logic nghiệp vụ của một đơn hàng.

Thuộc tính (Attributes):

order_id: Mã định danh duy nhất của đơn hàng.
status: Trạng thái hiện tại của đơn hàng (ALLOCATED, DELIVERED, EXCEPTION, etc.).
shipper_id: ID của shipper được giao đơn hàng.
pod_data: Dữ liệu bằng chứng giao hàng (chữ ký, ảnh).
reason: Lý do thất bại (nếu có).
next_attempt_at: Thời gian lên lịch lại giao hàng.

Phương thức (Methods):

+createOrder(data): Tạo một đơn hàng mới.
+updateOrderStatus(orderId, newStatus, podData, reason): Cập nhật trạng thái và các thông tin liên quan (lưu POD, lý do).
+rescheduleOrder(orderId, newDate): Cập nhật thời gian giao lại cho một đơn hàng.
+validatePOD(podData): Xác thực tính hợp lệ của bằng chứng giao hàng.
+checkOrderForDelivery(orderId): Kiểm tra các điều kiện để có thể giao hàng (ví dụ: trạng thái đơn).

3. Thành phần Notification Service: Thành phần này chịu trách nhiệm gửi các thông báo đến các bên liên quan.

Thuộc tính (Attributes):

notification_type: Loại thông báo (ví dụ: PUSH, EMAIL).

target_user_id: ID của người nhận thông báo (shipper, quản lý, CSKH).

Phương thức (Methods):

notifyStatusChange(orderId, newStatus): Gửi thông báo khi trạng thái đơn hàng thay đổi.

sendPushNotification(userId, message): Gửi thông báo đẩy đến người dùng.

sendEmail(email, subject, body): Gửi email.

4. Thành phần Database: Đây là nơi lưu trữ dữ liệu bền vững của hệ thống.

Thuộc tính (Attributes):

orders: Bảng lưu trữ thông tin về đơn hàng.

pod_records: Bảng lưu trữ dữ liệu POD.

notification_logs: Bảng ghi lại lịch sử các thông báo đã gửi.

Phương thức (Methods):

saveOrder(data): Lưu thông tin một đơn hàng mới.

updateOrder(orderId, data): Cập nhật thông tin của một đơn hàng.

getOrderById(orderId): Lấy thông tin đơn hàng theo ID.

savePOD(orderId, podData): Lưu dữ liệu bằng chứng giao hàng.

4. Định nghĩa các yêu cầu chất lượng cho phần mềm (Non-functional requirements)

Hầu hết các yêu cầu chất lượng bắt nguồn từ mong muốn của các tác nhân có liên quan tới PM, để đảm bảo rằng PM sẽ trợ giúp tốt cho các users (chạy đủ nhanh, chính xác, an toàn,..). Việc đưa ra yêu cầu chất lượng cho phải có nguồn gốc, có thể là con người có vai trò cụ thể đối với PM, hoặc văn bản đang áp dụng có tính pháp lý như luật của nhà nước, quy trình/quy định của tổ chức,..).

Có 3 loại yêu cầu:

a) Yêu cầu từ môi trường nghiệp vụ (business): là yêu cầu để PM tạo ra giá trị sử dụng tốt nhất cho tổ chức, ví dụ: thời gian xử lý một giao dịch không quá 0.5 giây, dữ liệu có backup

b) Yêu cầu từ môi trường vận hành (operation): là các yêu cầu và ràng buộc để PM hoạt động ổn định trong hệ thống, ví dụ: tuân thủ quy trình của tổ chức, có xác thực, có phân quyền, đảm bảo tương thích, ...

c) Yêu cầu từ môi trường phát triển (development): Là các yêu cầu và ràng buộc cho việc xây dựng phần mềm (tạo mới, nâng cấp), ví dụ: phát triển server trên nền SpringBoot.

IV. Thiết kế phần mềm

Thiết kế cho các đối tượng thành phần của PM để tiến tới cài đặt (implementation)

1.Thiết kế kiến trúc của PM

Nhận thức: Sau công đoạn phân tích, mỗi thành phần của phần mềm được mô tả chi tiết thuộc tính và hành vi chỉ ở mức ý niệm, ta cần phân rã mỗi thành phần này thành các lớp đối tượng thiết kế:

+ Lớp biên (Interface): ví dụ Form cho user và public function cho các đối tượng bên ngoài
+ Lớp xử lý (Process): ví dụ APIs, methods từ package hỗ trợ, services từ hệ điều hành,...
+ Lớp thực thể (Entity): ví dụ database.

Cách thực hiện: Để trực quan và dễ hiểu về cách xử lý của mỗi thành phần, ta bắt đầu từ giao diện của nó (interface: “form”) trong kịch bản của usecase mà nó tham gia; mỗi request trên giao diện (vd: nhấn nút "login"), hãy chỉ ra cách xử lý của nó (method/process: dữ liệu vào, dữ liệu ra, biến đổi input-output như thế nào: gọi API nào (của class nào), gọi stored procedure nào, và nó cần dữ liệu gì (entity: view/table/stored procedure của riêng lớp này). Các mô tả này được nhìn từ góc độ người làm ra PM, chứ không phải là cách sử dụng PM của user.

Cách trình bày:

Với mỗi use-case trong lược đồ usecase thiết kế ở trên và kịch bản tương tác của nó, em hãy mô tả cách xử lý của các đối tượng của PM tham gia vào usecase này bằng các lớp thiết kế của nó:

a) (Lớp biên) Form: <Tên>

Ảnh: có tên của form

Users: là các actors tương tác với form này. Mỗi actor gọi tên bằng vai trò trong thực tế.

Control chính của form (button, input text, ...):

FormControl <tên>

Nhiệm vụ trong form: ...

Inputs: dữ liệu được lấy từ form,

Outputs: kết quả trả về: … (có thể là data, hoặc form mới).

Xử lý: gọi Api <tên>, để: kiểm tra dữ liệu, tính toán,…

b) (Lớp xử lý) Api: <Tên>

Nhiệm vụ: ...

Inputs: bộ dữ liệu / thông số đầu vào

Outputs: dữ liệu trả về (cho control của form)

Xử lý: có thể truy cập dữ liệu của nó (gọi stored procedure), gọi hàm của gói công nghệ hỗ trợ (theo mô tả cách dùng công nghệ trình bày trong mục II), gọi các API của các actors/đối tượng hỗ trợ, … để tính toán và trả kết quả.

c) (Lớp thực thể) dữ liệu của đối tượng

Được trình bày ở dạng lưu trữ được (bảng, view, hoặc stored procedure của CSDL loại quan hệ hoặc hướng đối tượng).

2.Thiết kế cơ sở dữ liệu

Nhận thức: Dựa trên tập dữ liệu có thể đưa vào CSDL từ các API ở trên (từ dữ liệu inputs của các API dùng để cập nhật CSDL). Với tiếp cận OOAD, có thể xem DBMS như một đối tượng riêng cho mỗi thành phần (như trong kiến trúc microservice) hoặc là một đối tượng dùng chung cho cả PM, trong đó đối tượng này chỉ cung cấp các dịch vụ dữ liệu (stored procedures/ functions, không truy vấn trên bảng). Việc che giấu cấu trúc dữ liệu bảng giúp cho DB Admin có toàn quyền thay đổi cấu trúc dữ liệu của DB mà không làm ảnh hưởng đến ứng dụng.

Cách trình bày:

a) ERD hoặc lược đồ Class --> bảng (3NF) cho RDBMS hoặc các lớp cho OODBMS

b) Stored Procedure: <Tên>

Nhiệm vụ trong CSDL:...

Inputs: tham số đầu vào

Outputs: bộ dữ liệu (data set) trả về

Quyền sử dụng: các roles (users) được cấp quyền dùng stored procedure này

c) Trigger: <Tên> để ràng buộc toàn vẹn dữ liệu

Nhiệm vụ trong CSDL:...

Event: sự kiện kích hoạt

Action: xử lý

V. Hiện thực

1) PM đã được xây dựng (version 1) mà user có thể sử dụng

Với mỗi usecase: mô tả cách sử dụng các forms theo quy trình vận hành của tổ chức

2) Ưu khuyết điểm của PM so với yêu cầu

Tài liệu tham khảo

Danh sách các tài liệu là dẫn chứng / tham khảo của đề tài.