

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 rãi (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 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) Hiện trạng phát sinh nhu cầu dùng phần mềm

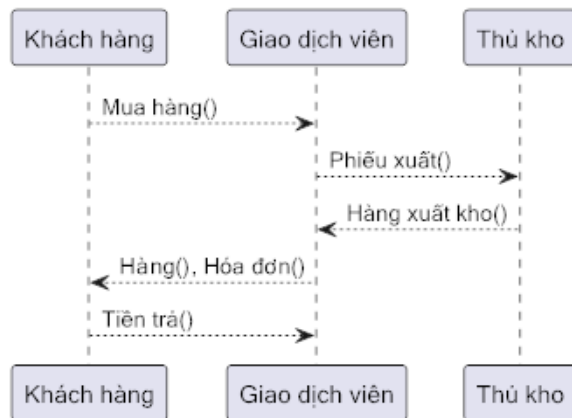
Nhận thức: Theo các mô hình làm phần mềm (SoftWare Development Models), thì đây là kết quả của công đoạn "khảo sát hiện trạng" của nơi (tổ chức) cần dùng phần mềm, để từ đó định nghĩa yêu cầu cho phần mềm, thiết kế, hiện thực,...

Theo lý thuyết hệ thống, một tổ chức (công ty, trường, bệnh viện,...) là một bộ máy gồm những đối tượng (con người/thiết bị/phần mềm,...) cùng hợp tác với nhau thực hiện mục đích của nó (là giải quyết tốt các tình huống trong việc thực hiện mục đích). Những tình huống gây bất lợi cho việc thực hiện mục đích của tổ chức sẽ được tổ chức đó quan tâm tìm cách khắc phục. Vì vậy phân tích hiện trạng là chỉ ra các tình huống bất buộc phải xử lý của tổ chức mà trong đó có các tình huống bất lợi cần khắc phục / cải tiến.

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 tổ chức, *đang hợp tác cùng nhau thực hiện mục đích của tổ chức*, và vẽ ra các lược đồ UML để mô tả. Lược đồ này không được dùng cho việc định nghĩa yêu cầu làm phần mềm, nó chỉ dùng để kết luận sơ lược về các vấn đề/khó khăn của tổ chức khi nó vận hành theo mô hình này.

Cách trình bày: mô tả mô hình vận hành hiện tại của tổ chức bằng các loại lược đồ UML tuần tự, cộng tác, hoặc lưu đồ (work-flows, quy trình), và mô tả thêm bằng lời cho rõ nghĩa. Các đối tượng trong tổ chức được nhắc đến trong các lược đồ này phải được đặt tên theo vai trò của chúng trong tổ chức.

Ví dụ, Quy trình bán hàng truyền thống của một công ty: 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.



Các tình huống trong đó là:

- Nhận đơn đặt hàng (đối tượng: khách hàng, giao dịch viên)
- Xuất hàng cho đơn (đối tượng : thủ kho, giao dịch viên)
- Giao hàng (đối tượng : giao dịch viên, khách)
- Thu tiền (đối tượng : giao dịch viên, khách, thủ quỹ).

Khó khăn: phạm vi bán hàng nhỏ, giấy tờ thủ công, thiếu truy vết, sai sót tồn kho,...

2) Đề xuất giải pháp của đề tài (mô hình vận hành mới cho tổ chức)

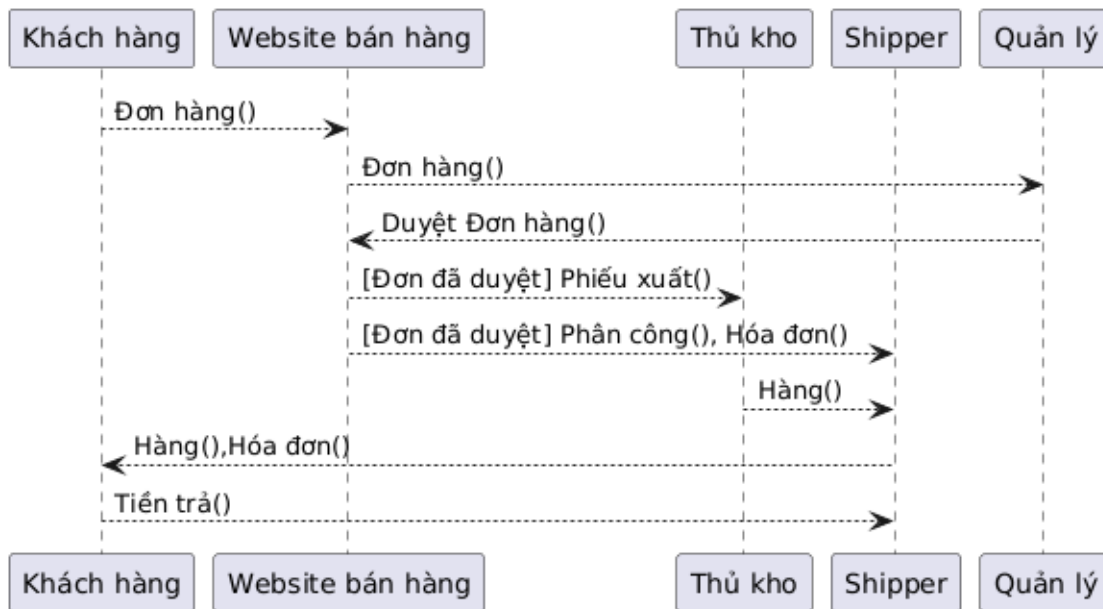
Nhận thức: Giải pháp của đề tài là một quy trình vận hành mới có phần mềm tham gia, thay cho quy trình vận hành cũ. Phần mềm ứng dụng của đề tài (PM) sẽ được dùng như là một thành phần được thêm vào tổ chức để giúp cho tổ chức giải quyết tốt các tình huống của nó. Chúng ta chỉ quan tâm đến các tình huống bất lợi thực tế của tổ chức mà PM có khả năng giải quyết. Đây là yếu tố khẳng định tính thực tiễn của đề tài: Nếu nhận thức hiện trạng không thực tế (vd: quá lạc hậu so với thực tế) thì đề tài sẽ không có giá trị sử dụng.

Cách thực hiện:

Từ hiện trạng được mô tả ở trên, ta nhận định một số tình huống vấn đề mà PM có thể tham gia giải quyết cùng với các đối tượng khác (con người, công cụ, phần mềm đang có), và đề xuất mô hình vận hành mới cho tổ chức bằng lược đồ cộng tác hoặc tuần tự, để chỉ ra cách trợ giúp của PM (như 1 công cụ được sử dụng chung trong tổ chức), và nêu rõ lợi ích từ việc sử dụng PM này.

Ví dụ: Bán hàng online: PM là Website bán hàng.

Sequence Diagram - Bán hàng online



Ích lợi của Website: phạm vi bán hàng rộng, đơn hàng nhận 24/24, có thể truy vết,...

3) Định nghĩa các tình huống mà PM tham gia giải quyết

Nhận thức: Là định nghĩa vai trò/nhiệm vụ của PM (và các thành phần của nó) trong việc tham gia giải quyết các tình huống của tổ chức (=usecases của PM: một usecase là một tình huống mà PM được vài actors sử dụng để nhận được sự trợ giúp thực tế từ nó) theo mô hình vận hành mới được đề xuất từ đề tài. Ví dụ:

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

Ta cũng cần làm rõ các tương tác cần phải có giữa PM với các đối tượng có liên quan (actors). Các tương tác này sẽ được thực hiện bằng các chức năng của PM: đây chính là yêu cầu chức năng cho PM.

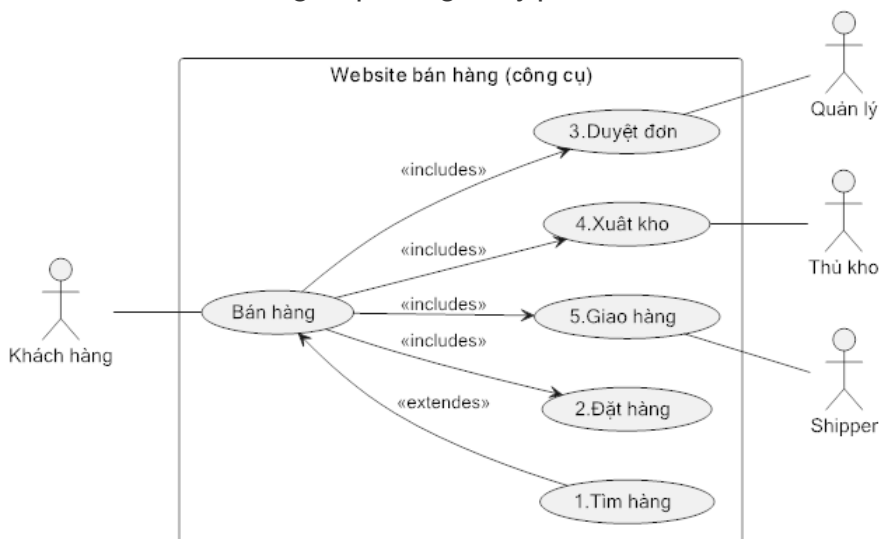
Cách trình bày:

- 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 bán hàng online.
- 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 <<extends>> ("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 "Admin" và "User". Các actor này chỉ xuất hiện trong lúc thiết kế PM. Các actor trong lược đồ usecase này phải là các vai trò đã biết rõ trong tổ chức (giống như lúc PM chưa ra đời), như "thủ kho", "shipper", "khách",...
 - ★ Usecase "đăng nhập", "phân quyền" vì chúng là các thủ tục được đòi hỏi từ PM cho việc bảo mật tài nguyên; chúng không phải là nhu cầu mong đợi của người sử dụng.
 - ★ Các actor hỗ trợ cho PM, như "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 trong khi thiết kế PM (xem PM là một hệ thống).

Ví dụ 1 - Tình huống bán hàng online bằng PM = Website bán hàng:

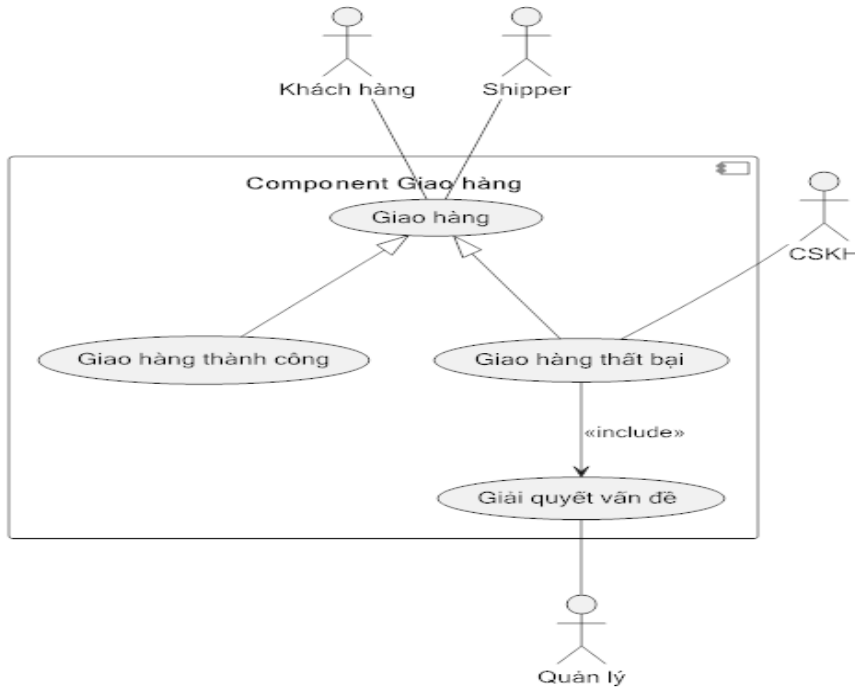
Actors: Khách hàng, Thủ kho, Shipper, Quản lý (không có Giao dịch viên)

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 trực trực)



Mục đích: giao hàng cho người nhận, ghi nhận POD nếu thành công, hoặc xử lý đơn giao không thành công.

Actor chính: Shipper.

Actor liên quan: Khách hàng (nhận hàng), Người quản lý (xử lý bất thường - EXCEPTION).

Tiền điều kiện: Đơn ở trạng thái đã phân công giao hàng (ALLOCATED)

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

Hậu điều kiện (thất bại): Đơn EXCEPTION, có reason, và lập lịch giao lại.

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

Lưuồng ngoại lệ: Không gọi được/người nhận vắng mặt/địa chỉ sai → ghi reason → báo thất bại → Người quản lý lên lịch giao lại.

Đặc tả này là cơ sở để kiểm tra các lược đồ khác cho Giao hàng.

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.

4) Định nghĩa các tương tác của PM

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 tương tác 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 có thể cần nhiều tài liệu để mô tả thông tin cộng tác (lược đồ tuần tự/cộng tác), diễn tả trình tự các bước thực hiện (lược đồ hoạt động), diễn tả mối quan hệ giữa các đối tượng được nhắc đến trong usecase (lược đồ class), diễn tả sự chuyển trạng thái hợp lệ của các đối tượng (lược đồ chuyển trạng thái). Nếu PM có hỗ trợ xử lý thông tin thì phải có lược đồ tuần tự/cộng tác diễn tả các tương tác (có chứa thông tin) với PM.
- Vì PM không thể xử lý được thông tin (ý niệm) như con người, nó cần dữ liệu 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 sẽ được định nghĩa từ cặp thông điệp request-response, mỗi thông điệp có dạng: **[điều kiện xuất hiện] <tên_thông_điệp> (bộ dữ liệu của thông điệp)**.
- 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.

Cách trình bày:

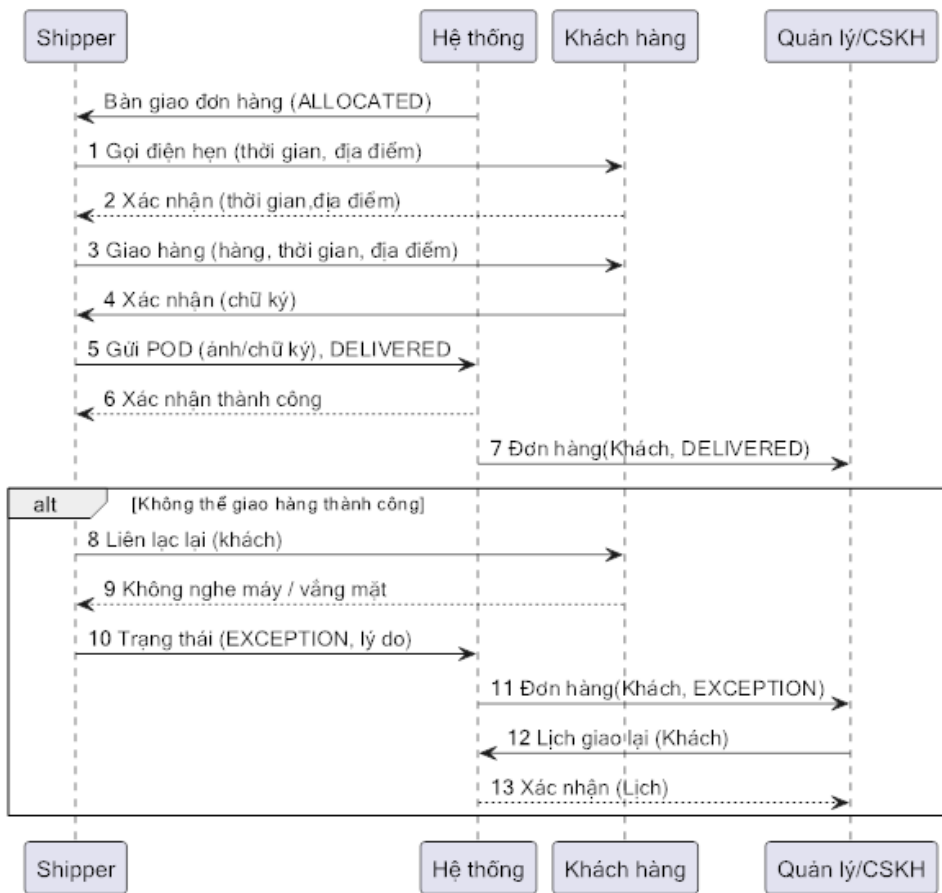
a) Vẽ lược đồ usecase gồm các usecase, actors và quan hệ actor-usecase, actor-actor, usecase-usecase. Lược đồ này giúp ta phân biệt các usecase, tránh trùng lặp hoặc dư thừa usecase.

b) Với mỗi usecase trong lược đồ:

- Nêu mục đích của nó: nêu rõ tình huống mà actor cần PM trợ giúp.
- Mô tả chi tiết kịch bản tương tác giữa actors với PM để xử lý tình huống này bằng **lược đồ tuần tự**
- Mô tả thêm cho nó (nếu cần) bằng các lược đồ hoạt động, lược đồ lớp, lược đồ trạng thái,...
- Các lược đồ mô tả cho 1 usecase phải nhất quán nhau, tối thiểu là tên của các lớp.

Ví dụ:

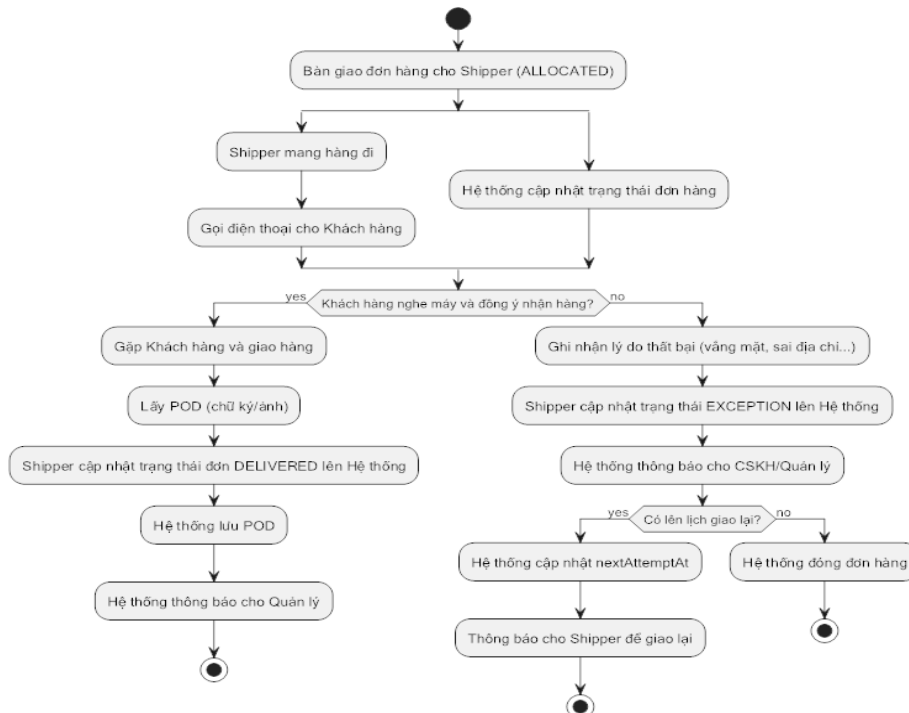
Sequence Diagram - Quy trình Giao hàng



+ Sequence diagram cho "Giao hàng"

Mọi thông điệp đi đến PM (Hệ thống) đều có tham số dữ liệu, để định nghĩa các chức năng cho PM.

Activity Diagram - Quy trình Giao hàng



+Activity Diagram cho "Giao hàng":

Để kiểm tra tính nhất quán với sequence diagram, xác định thêm luồng xử lý khác nếu thiếu.

+State Diagram cho
"Giao hàng":



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

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

a) Định nghĩa các đối tượng thành phần của PM cho từng usecase

Cách thực hiện: Xem xét tất cả các lược đồ đã vẽ 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 của hệ thống PM có khả năng tham gia xử lý từng usecase, bằng kỹ thuật CRC, hay kỹ thuật phân tích kích bản của usecase,...

Ví dụ: Chúng ta có thể phân rã hệ thống quản lý giao hàng (WebApp) thành 4 thành phần chính để xử lý nghiệp vụ giao hàng:

★ **Shipper App:** có nhiệm vụ giao tiếp với Shipper (và OrderService)

+ Thông báo đơn hàng được phân công cho shipper **OrderAllocation**(ShipperId, OrderId, DeliveryStatus)

+ Lấy minh chứng đã giao hàng thành công **capturePOD**(orderId, signature, photoUri). POD = Proof Of Delivery, như chữ ký của khách hàng, ảnh chụp hàng được giao

+ Gửi minh chứng giao hàng thành công đến OrderService **sendPOD**(orderId, POD, DeliveryStatus = DELIVERED)

+ gửi **reportException**(orderId, reason = UNREACHABLE/ABSENT) là thông báo giao hàng thất bại đến OrderService

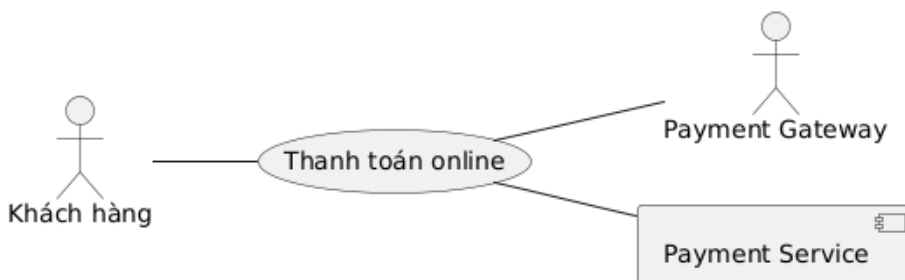
+ Nhận lịch giao hàng lại từ OrderService để thông báo cho shipper (**OrderAllocation**(...))

- ★ **OrderService:** Chịu trách nhiệm cập nhật thông tin đơn hàng và giao hàng của mỗi đơn hàng. Nó là thành phần duy nhất can thiệp vào dữ liệu về đơn hàng.
- + Xử lý yêu cầu **OrderAllocation**(shipperId, orderId, DeliveryStatus) từ Shipper App: lấy đơn hàng cần giao từ DB để trả về cho Shipper App.
- + Nhận yêu cầu **sendPOD**(orderId, POD, DeliveryStatus = DELIVERED) để cập nhật dữ liệu vào DB và thông báo cho NotiService (Giao hàng thành công)
- + Nhận yêu cầu **reportException**(orderId, reason) để cập nhật trạng thái EXCEPTION của đơn hàng orderId vào DB (Giao hàng thất bại) và gửi đến NotiService
- + Nhận lịch giao hàng lại từ NotiService (nextAttemptAt(OrderID, ReSchedule)) để cập nhật DB, và gửi thông báo đến Shipper App
- ★ **NotiService:** chịu trách nhiệm giao tiếp với người quản lý (thông báo tình trạng giao hàng và nhận chỉ thị xử lý)
- + Nhận thông báo giao hàng thành công từ OrderService để gửi đến Manager
- + Nhận thông báo giao hàng thất bại từ OrderService để gửi đến Manager
- + Nhận chỉ thị xử lý (lên lịch gửi lại) gửi cho OrderService
- + Nhận lịch giao hàng lại từ Manager gửi đến OrderService
- ★ **DB :** Lưu thông tin về các đơn hàng, phân công giao hàng, các trạng thái giao hàng : ALLOCATED (Đã bàn giao cho Shipper), DELIVERED (Đã giao thành công), EXCEPTION (Giao hàng thất bại), RESCHEDULED (Lên lịch lại), và POD (minh chứng giao hàng).

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

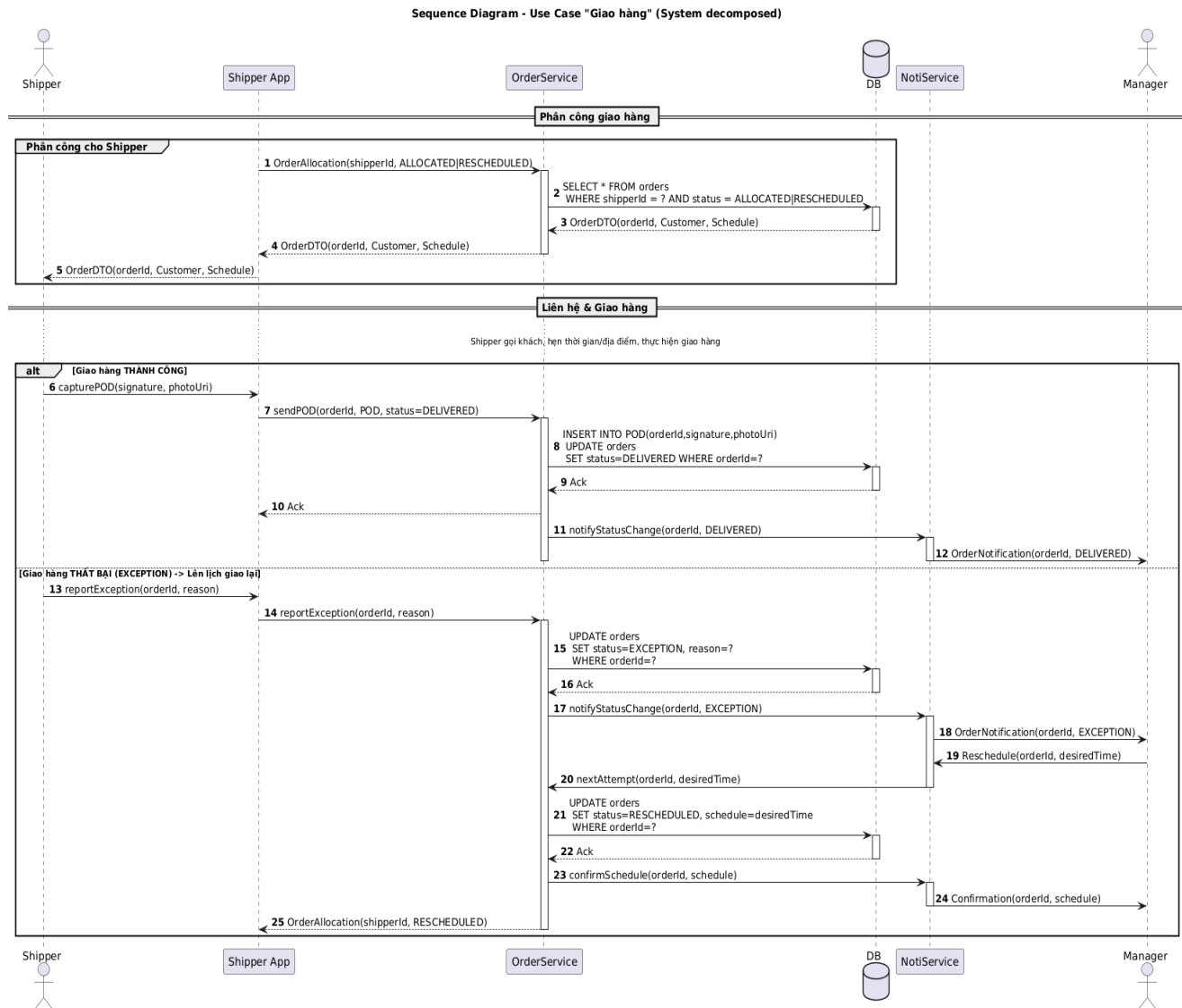
Nhận thức: Mỗi đối tượng thành phần của PM đôi khi cũng cần trợ giúp từ các actor bên ngoài, ví dụ, thành phần "Payment Service" của hệ thống PM tham gia xử lý usecase "Thanh toán online" cũng cần dùng "Payment Gateway" như là 1 actor trợ giúp.

Cách trình bày: thêm các actor hỗ trợ cho PM (Payment Gateway) vào usecase (Thanh toán online) và nối usecase này với thành phần của PM tham gia vào usecase này.



c) Xác định cách phối hợp xử lý tình huống giữa các đối tượng trong và ngoài hệ thống

Cách thực hiện: Từ lược đồ tuần tự diễn tả tương tác giữa hệ thống với các actors, diễn tả lại các tương tác này trên các thành phần của hệ thống. Ví dụ Giao hàng:

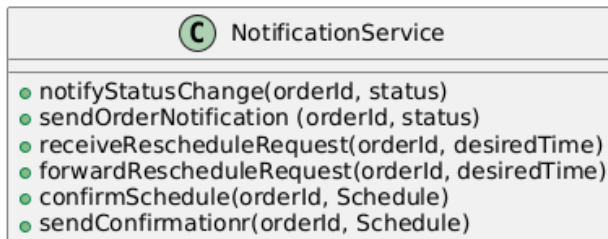
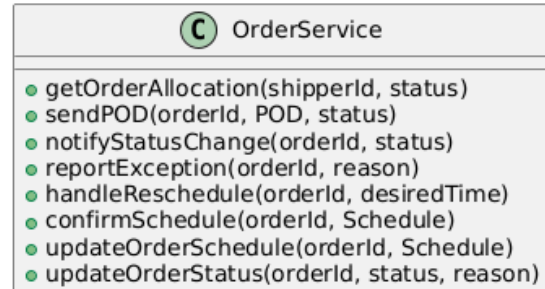
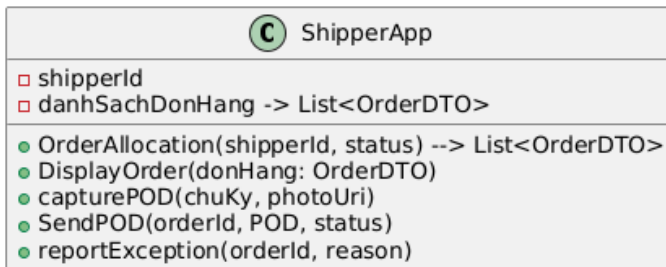


d) Định nghĩa yêu cầu chi tiết (thuộc tính, phương thức) cho từng đối tượng thành phần

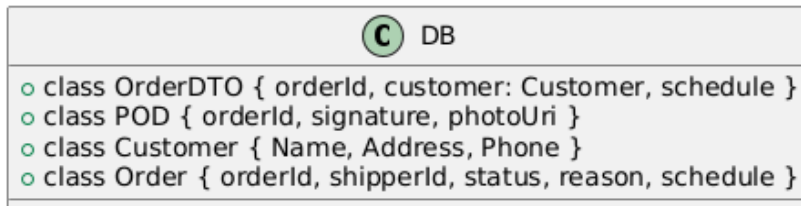
Nhận thức: Tiếp cận hướng đối tượng dựa trên thực tế để định nghĩa các đối tượng trong hệ thống. Vì vậy tên của lớp, tên của quan hệ, tên thuộc tính và phương thức cần lấy từ thực tế, và nhất quán nhau trong các lược đồ. Ví dụ: từ quan hệ "điều trị bệnh" giữa bác sỹ và bệnh nhân vẽ trong lược đồ lớp, ta biết rằng đối tượng bác sỹ phải có một vài phương thức để điều trị cho bệnh nhân, cụ thể là khám bệnh(), kê toa thuốc() hoặc ghi phác đồ điều trị(). Đối với đối tượng thành phần của PM, ta cần xác định thuộc tính và phương thức của nó để nó giúp PM thực hiện được các tương tác trong từng usecase, dựa trên các lược đồ đã vẽ

Cách trình bày: mỗi đối tượng thành phần của PM được mô tả như một lớp (UML-class, chỉ có 3 phần: tên, thuộc tính và hành vi) là các yêu cầu cho từng đối tượng thành phần.

Ví dụ về các lớp đối tượng (ý niệm) của hệ thống PM cho usecase Giao hàng từ lược đồ tuần tự ở trên:




Class Database (DB):



Thay vì định nghĩa DB như một tập các bảng (public) với các câu lệnh truy vấn SqlCommand như trong lược đồ tuần tự trên, theo tiếp cận hướng đối tượng thì ta sẽ định nghĩa DB thành một lớp

Data Access Object (DAO) hoặc Repository để quản lý các tương tác với DB. Việc này giúp tách biệt logic nghiệp vụ của OrderService khỏi các thao tác truy cập dữ liệu chi tiết trên các bảng (sẽ rất khó sửa cấu trúc bảng sau này). Lớp này (OrderRepository) sẽ đóng vai trò như một cầu nối, cung cấp các phương thức public để OrderService gọi đến, thay vì phải viết các câu lệnh SQL trực tiếp.

 OrderRepository
<ul style="list-style-type: none"> ● selectOrders(shipperId, status) --> List<OrderDTO> ● insertPOD(orderId, POD: Object) : void ● updateOrderStatus(orderId, status) : void ● updateOrderStatusAndReason(orderId, status, lyDo) : void ● updateOrderSchedule(orderId, Schedule) : void

Sau công đoạn phân tích, mỗi đối tượng tham gia vào các usecase (các thành phần của PM và các actor) được mô tả thuộc tính và hành vi cần thiết cho usecase theo mô hình vận hành mới; các mô tả này sẽ được các nhà phát triển dùng để tìm các đối tượng trong thực tế để gắn vào hệ thống (ví dụ trang bị máy in, kết nối Payment Gateway, cài đặt hệ quản trị CSDL...) hoặc mô phỏng các thuộc tính và hành vi này vào phần mềm (thiết kế cho phần mềm).

6. 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, để đảm bảo rằng PM sẽ đặc lực cho tổ chức. Các yêu cầu phải có nguồn gốc, có thể là từ người có thẩm quyền trong tổ chức, hoặc văn bản pháp lý như luật của nhà nước, quy định của tổ chức,... Có 3 loại yêu cầu:

a) Yêu cầu từ môi trường nghiệp vụ (business): là yêu cầu và ràng buộc để PM tạo ra giá trị sử dụng cao cho tổ chức, ví dụ: thời gian tìm hàng không quá 0.5 giây/mỗi yêu cầu, dùng Tiếng Việt cho toàn bộ giao diện với người dùng đầu cuối, dễ sử dụng

b) Yêu cầu từ môi trường vận hành (operation): là các yêu cầu và ràng buộc để PM hoạt động ổn định, ví dụ: đảm bảo an ninh (bảo mật, xác thực, phân quyền, chống thâm nhập trái phép), đảm bảo tương thích (tuân thủ chuẩn),

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 PM theo kiến trúc vi dịch vụ (microservices) để chạy phân tán trên đám mây

Từ các yêu cầu này, các nhà phát triển sẽ bổ sung các biện pháp trong PM, ví dụ:

- ★ Yêu cầu đảm bảo an ninh: Bổ sung thêm thủ tục xác thực (mật khẩu, vân tay, gương mặt), user Admin để phân quyền, cơ chế mã hóa dữ liệu, cơ chế phát hiện kết nối bất hợp pháp, backup dữ liệu
- ★ Yêu cầu dễ sử dụng (user friendly): giao diện tương tác theo tình huống, có hướng dẫn, có trích dẫn nguồn gốc, có âm thanh, hình ảnh, Chat-bot,...
- ★ Yêu cầu phát triển: Quy ước viết code (coding convention), thư viện dùng chung,...

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, dựa vào 3 kỹ thuật:

a) Layering (phân tầng), ví dụ

+ **Lớp biên** (Interface): Form cho user và public function cho các đối tượng bên ngoài

+ **Lớp xử lý** (Process): APIs, methods từ package hỗ trợ, services từ hệ điều hành,...

+ **Lớp thực thể** (Entity): database.

b) Segmentation (phân vùng trong mỗi tầng) : định nghĩa các nhóm đối tượng liên quan chặt chẽ nhau trong miền vấn đề của ứng dụng, ví dụ: nhóm các đối tượng liên quan tới đơn hàng, nhóm các đối tượng liên quan tới sản phẩm, nhóm các đối tượng liên quan với khách hàng

c) Factoring (trích lớp đối tượng cơ bản) là trích ra các lớp đối tượng sử dụng được cho nhiều ứng dụng, ví dụ class POD (timetaken, signatureName, signatureImage, photoUri)

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"), 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 cần biết để làm ra PM (lập trình viên), chứ không phải là cách sử dụng PM có sẵn của user.

Cách trình bày:

Với mỗi use-case trong lược đồ usecase thiết kế ở trên và kịch bản tương tác của nó, 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: các actors tương tác với form này. Mỗi actor gọi tên bằng vai trò trong thực tế.

Các 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 bằng 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>**

Nhiệm vụ trong CSDL:...(để ràng buộc toàn vẹn dữ liệu)

Event: sự kiện kích hoạt

Action: xử lý

V. Hiện thực

1) Mô tả phiên bản PM đã được xây dựng (version) 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 version này so với yêu cầu

Tài liệu tham khảo

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