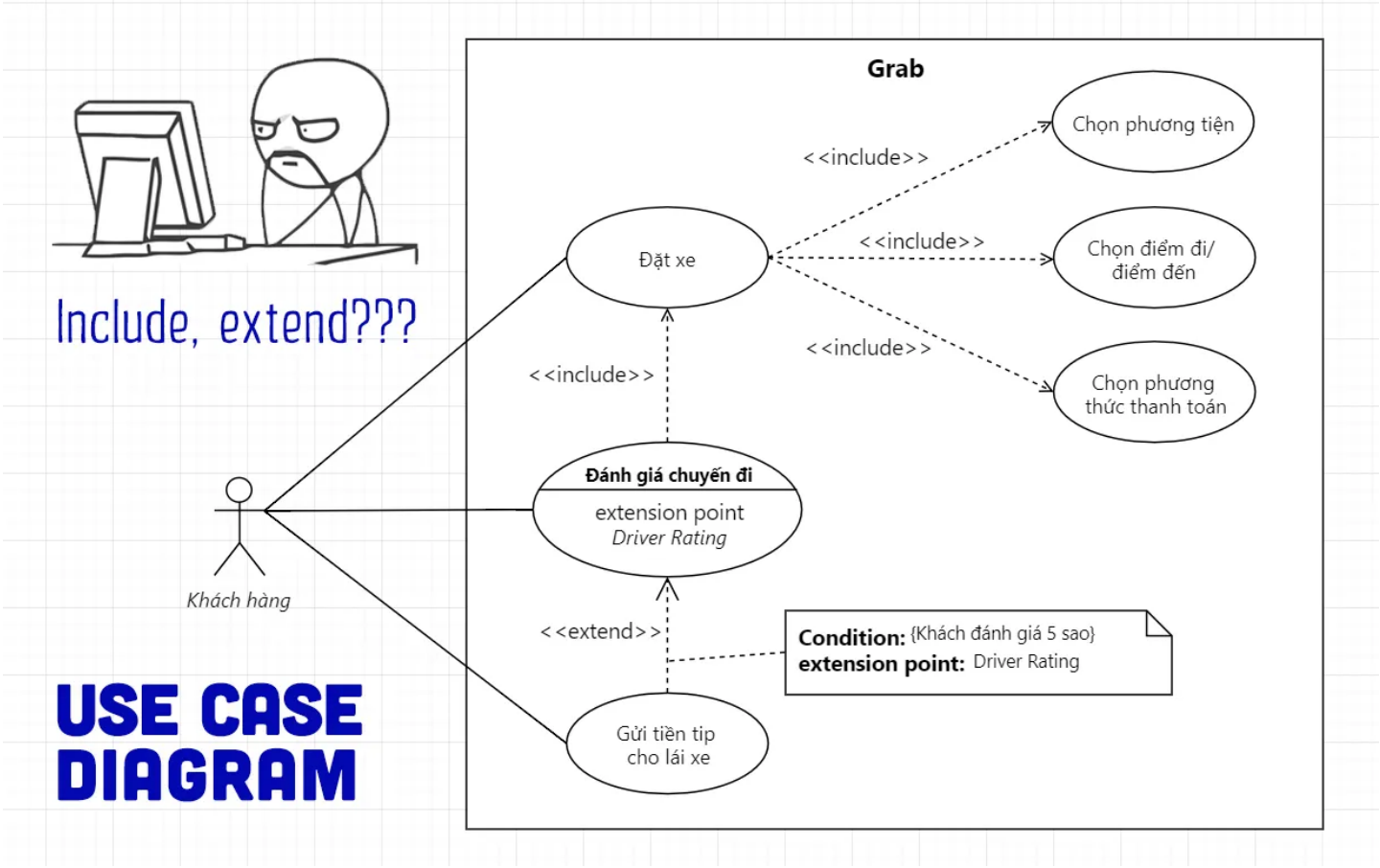
Thinhnotes.com

Use Case Diagram và 5 sai lầm thường gặp

Nguyen Hoang Phu Thinh 8 tháng ago



Hê lô anh em. Ở kỳ trước mình đã nói vê` BPMN – một đô` nghê` khá hữu dụng của BA. Hôm nay mình sẽ tiê´p tục nói vê` 1 trong những đô` nghê` khác cũng cực kỳ quan trọng không kém, đó chính là **Use Case**.

Bản thân mình thời gian đâ`u dùng Use Case cũng gặp râ´t nhiê`u khó khăn. Một mớ bô`ng bông câu hỏi cứ lớn quởn trong đâ`u: *bản châ´t của Use Case là gì, dùng cho mục đích nào, vẽ vậy đúng hay chưa, có chi tiê´t quá không, hoặc thậm chí vẽ Use Case xong cũng chẳng biê´t để làm gì???*

Do đó bài này mình sẽ note vê` những thứ mình học được, làm được và dĩ nhiên quan trọng nhâ´t là những sai lâ`m mà mình từng mă´c phải khi làm Use Case.

Nội dung

1. Use Case là gì?

Đầu tiên Use Case là một *technique* của công việc Business Analyst.

> *Use Case là kỹ thuật dùng để* ***mô tả sự tương tác*** *giữa người dùng và hệ thống với nhau, trong một* ***môi trường cụ thể*** *và vì một* ***mục đích cụ thể.***

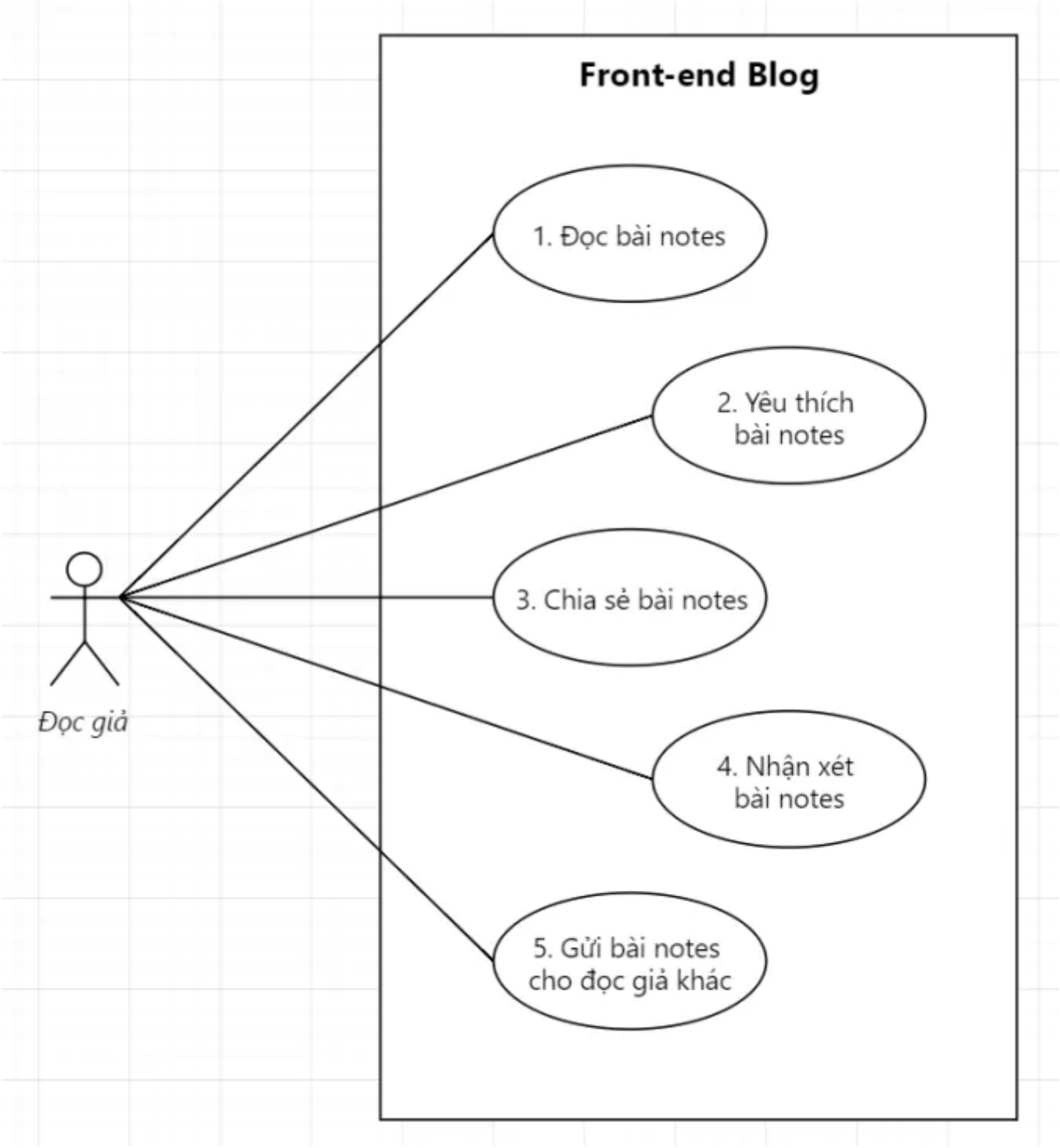
Sự tương tác ở đây có thể là:

- *Người dùng tương tác với hệ thô´ng như thê´ nào?*
- *Hoặc, hệ thô´ng tương tác với các hệ thô´ng khác như thê´ nào?*

Và dĩ nhiên, sự tương tác này phải nă`m trong một môi trường cụ thể, tức là nă`m trong một bô´i cảnh, phạm vi chức năng cụ thể, hoặc rộng hơn là trong một hệ thô´ng/ phâ`n mê`m cụ thể.

Sau cùng, việc mô tả sự tương tác này phải nhă`m diễn đạt một mục đích cụ thể nào đó. Use Case phải diễn rả được Requirement **theo góc nhìn** cụ thể từ phía **người dùng.**

Ví dụ sơ đô` Use Case diễn tả sự tương tác giữa người dùng là độc giả với trang blog Thinhnotes chẳng hạn.



Ví dụ đơn giản về Use Case

- **Tương tác ở đây là gì?**
  1. *Độc giả đọc bài notes*
  2. *Độc giả yêu thích bài notes*
  3. *Độc giả chia sẻ bài notes*
  4. *Độc giả nhận xét bài notes*
  5. *Độc giả gửi bài notes cho độc giả khác qua email*

- **Môi trường cụ thể?**

  Quá đơn giản, đó là trang blog Thinhnotes.com (không phải trang Admin).

- **Mục đích cụ thể?**
  1. *Người dùng có thể đọc được bài notes trên blog (đơn giản bỏ qua)*
  2. *Người dùng có thể bày tỏ được sự yêu thích bài notes*
  3. *Người dùng có thể chia sẻ bài notes này trên các nền tảng khác để nhiều người khác có thể đọc được*
  4. *Người dùng có thể viết nhận xét khen chê gạch đá các kiểu cho tác giả*
  5. *Người dùng có thể gửi bài notes này qua email cho một người bất kỳ.*

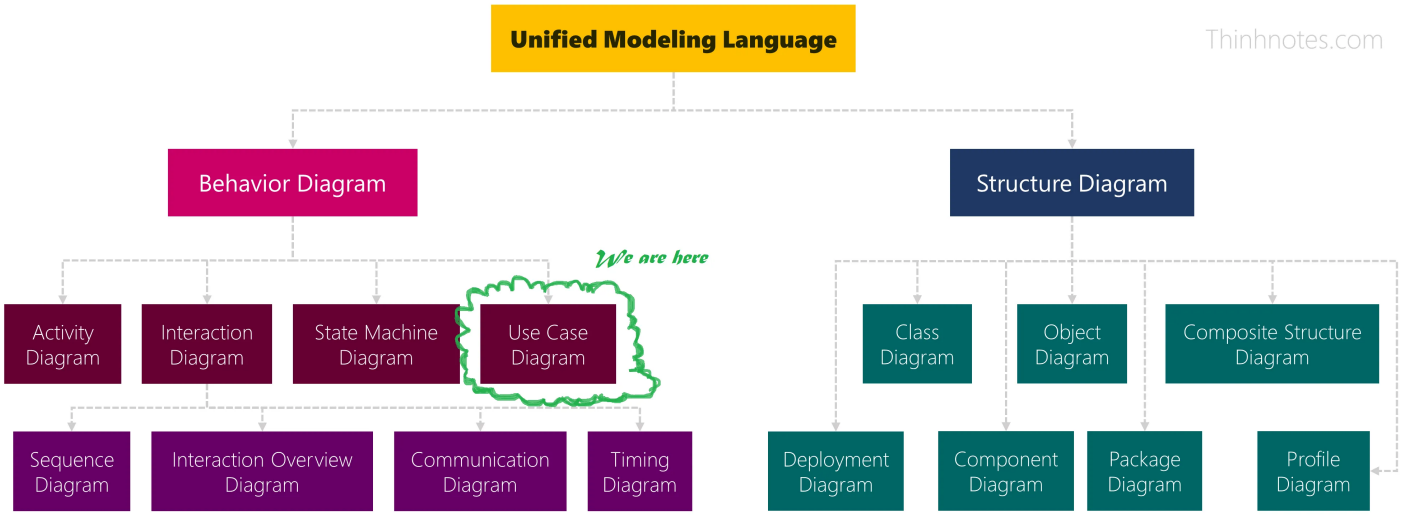
Đó là tất tần tật những nội dung mà một Use Case sẽ thể hiện.

Về hình thức thì Use Case tồn tại ở 2 dạng:

- Hình vẽ Use Case *(Use Case Diagram)*
- Đặc tả Use Case *(Use Case Specification)*.

Ở bài sau mình sẽ nói Use Case Specification sau nhé anh em. Bài này mình sẽ tập trung nói về Use Case Diagram.

Use Case Diagram là một thành viên trong họ UML (Unified Modeling Language).



Mỗi Diagram trong bộ UML này đều có những mục đích khác nhau. Tùy trường hợp, tùy dự án mà anh em sẽ *"rút hàng"* ra chiến như thế nào cho hợp lý.

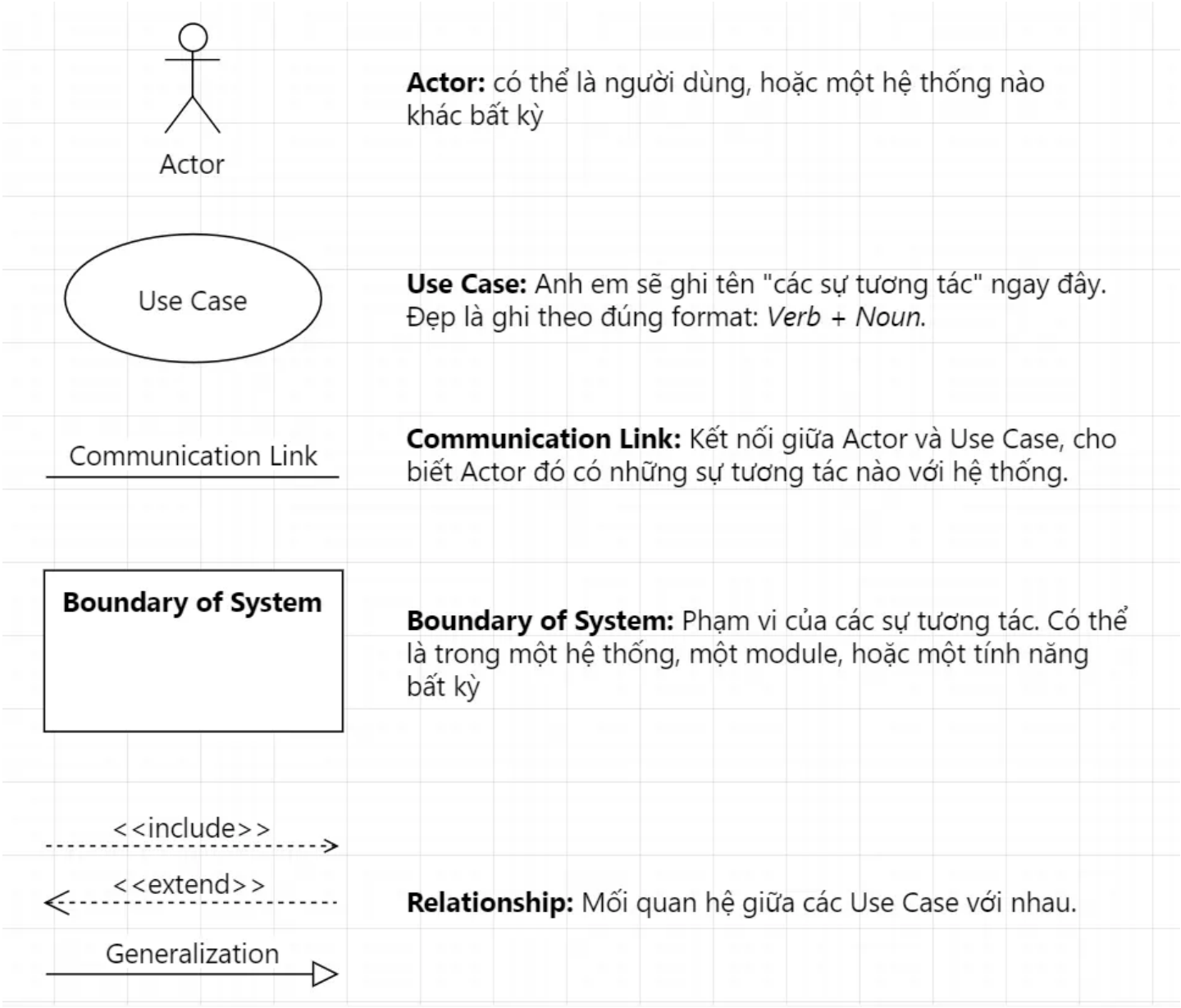
Hiểu sơ bộ *Use Case là gì* và *mục đích* của nó, chúng ta cùng tìm hiểu chi tiết Use Case Diagram và cách vẽ nhé anh em 😎

2. Các thành phâ`n của Use Case Diagram

2.1. Actor, Use Case, Communication Link và Boundary

Cũng không có gì quá phức tạp, Use Case Diagram gô`m 5 thành phâ`n chính:

- Actor
- Use Case
- Communication Link
- Boundary of System
- Và, Relationships.



Actor thì có thể là Người dùng, hoặc một System nào đó. Vì UML quy định Actor là hình thă`ng người nên có thể anh em sẽ nhâ`m lẫn chỗ đó *phải là người dùng* nhưng hổng phải.

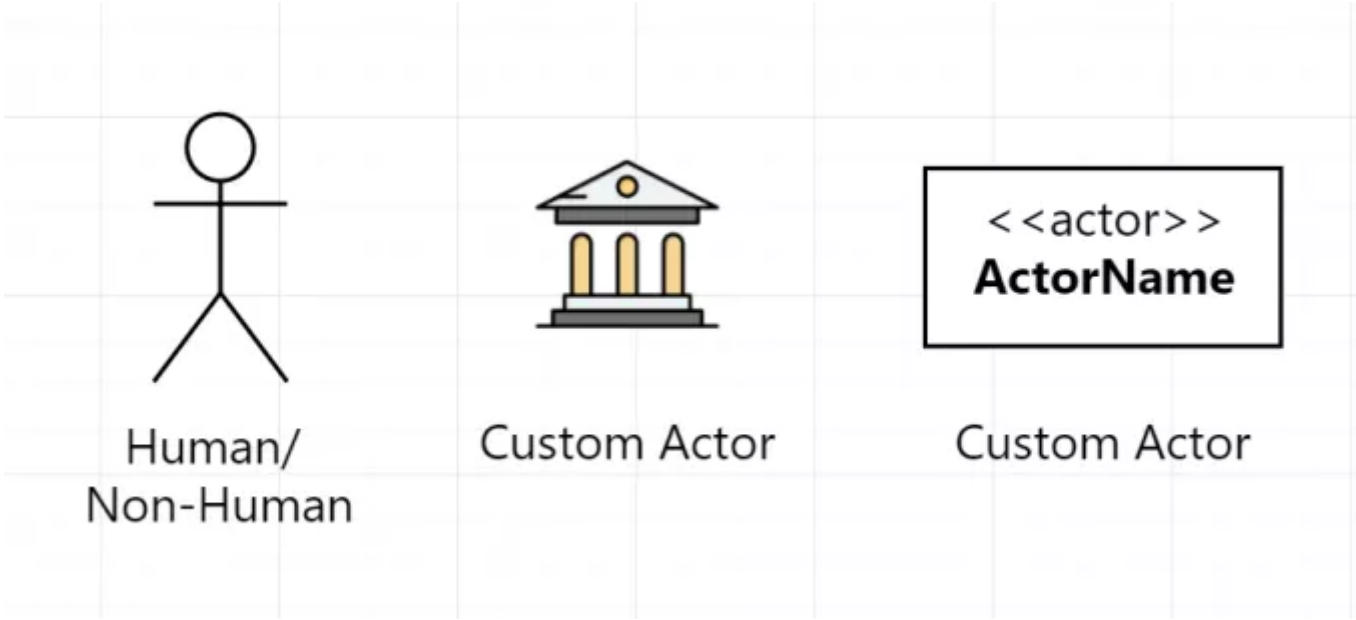
Một sô´ câu hỏi anh em có thể tự lẩm bẩm trong đâ`u để xác định Actor như sau:

- *Ai là người sử dụng hệ thống?*
- *Ai sẽ là người Admin của hệ thống (tức người cài đặt, quản lý, bảo trì... hệ thống)?*
- *Hệ thống này có được sử dụng bởi bất kỳ một hệ thống nào khác không?* ***(*)***
- *Hệ thống lưu trữ dữ liệu, vậy ai là người input dữ liệu vào hệ thống?*
- *Hệ thống lưu trữ dữ liệu, vậy ai là người cần những dữ liệu output?*

Ở mục (*), mình muốn highlight cho anh em chỗ này. Không phải giải pháp/ phần mềm nào làm ra đều được sử dụng bởi con người. Có những phần mềm làm ra, để cho... phần mềm khác sử dụng.

Chẳng hạn như làm các services. Mình có một anh bạn làm BA, giải pháp mà ảnh cùng đồng bọn làm ra là 1 services không được dùng bởi con người, mà được dùng bởi một hệ thống khác để xác thực người dùng.

Ký hiệu của Actor chủ yếu là hình thằng người, nhưng để Diagram thêm phong phú, đa dạng thì anh em có thể sử dụng các hình dưới đây, miễn có ghi chú rõ ràng là được.



Các ký hiệu thể hiện Actor.

Còn **Use Case** là anh em sẽ thể hiện dưới dạng hình Oval, thể hiện sự tương tác giữa các Actor và hệ thống.

Communication Link thể hiện sự tương tác giữa Actor nào với System. Nối giữa Actor với Use Case.

Boundary of System là phạm vi mà Use Case xảy ra. Ví dụ trong hệ thống CRM, phạm vi có thể là từng cụm tính năng lớn như *Quản lý khách hàng*, *Quản lý đơn hàng*, hoặc cả một module lớn như *Quản lý bán hàng*.

...

Ô kê nãy giờ dễ ẹc, mấy cái này nhìn sơ qua là anh em biết ngay cái một.

Cái cuối cùng mới chính là cái mà mình tin là nhiều anh em vẫn còn rất dễ lộn, đó là **Relationship.**

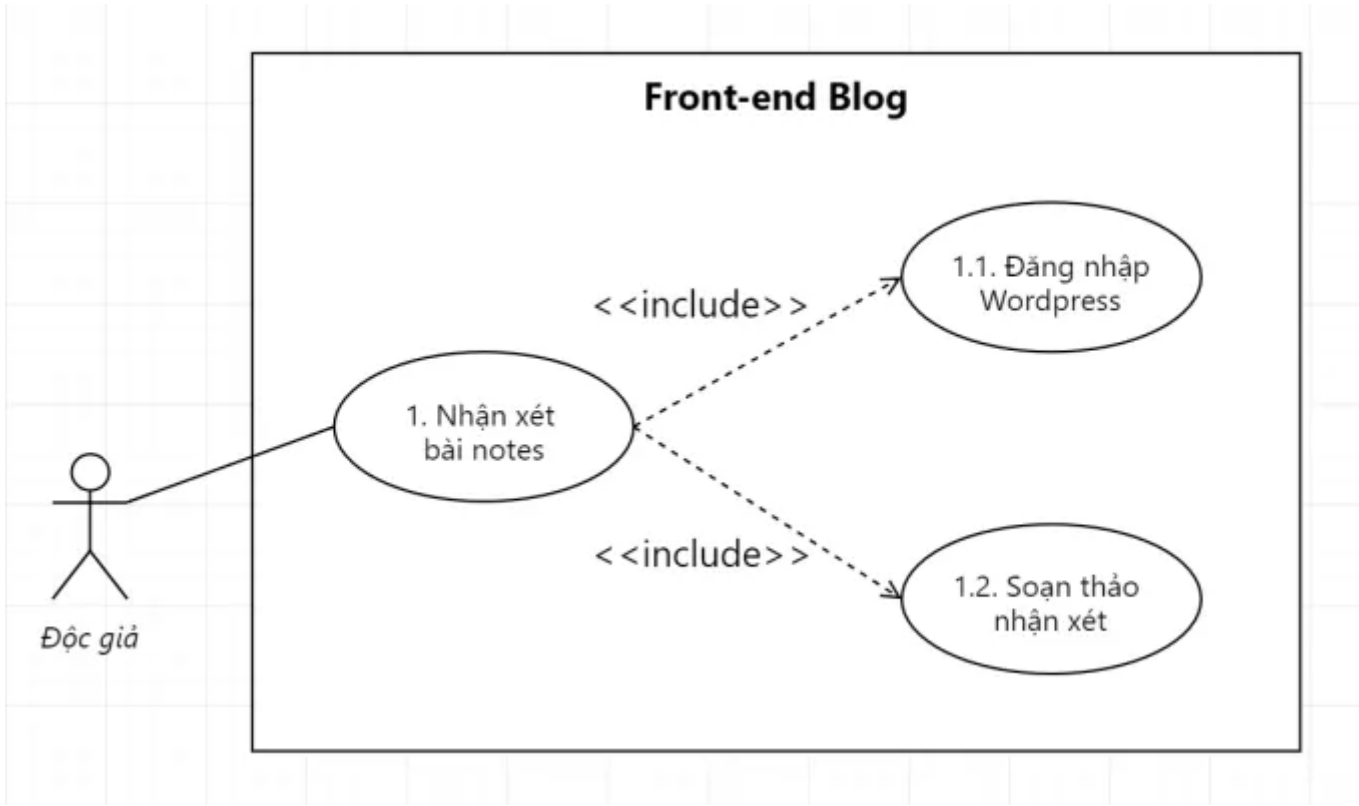
2.2. Relationship

Relationship gồm 3 loại: **Include**, **Extend**, và **Generalization**.

a) Include

Include nghĩa là mối quan hệ **bắt buộc phải có** giữa các Use Case với nhau.

Xét về nghĩa, Include nghĩa là *bao gồm*, tức nếu Use Case A có mối quan hệ include Use Case B, thì nghĩa là: **Use Case A bao gồm Use Case B.** *Để Use Case A xảy ra, thì Use Case B phải đạt được.*



Ví dụ về Include trong Use Case

Xét ví dụ trên, chúng ta có Use Case: *Nhận xét bài notes*. Use Case này include 2 Use Case khác là: *Đăng nhập WordPress* và *Soạn thảo nhận xét*.

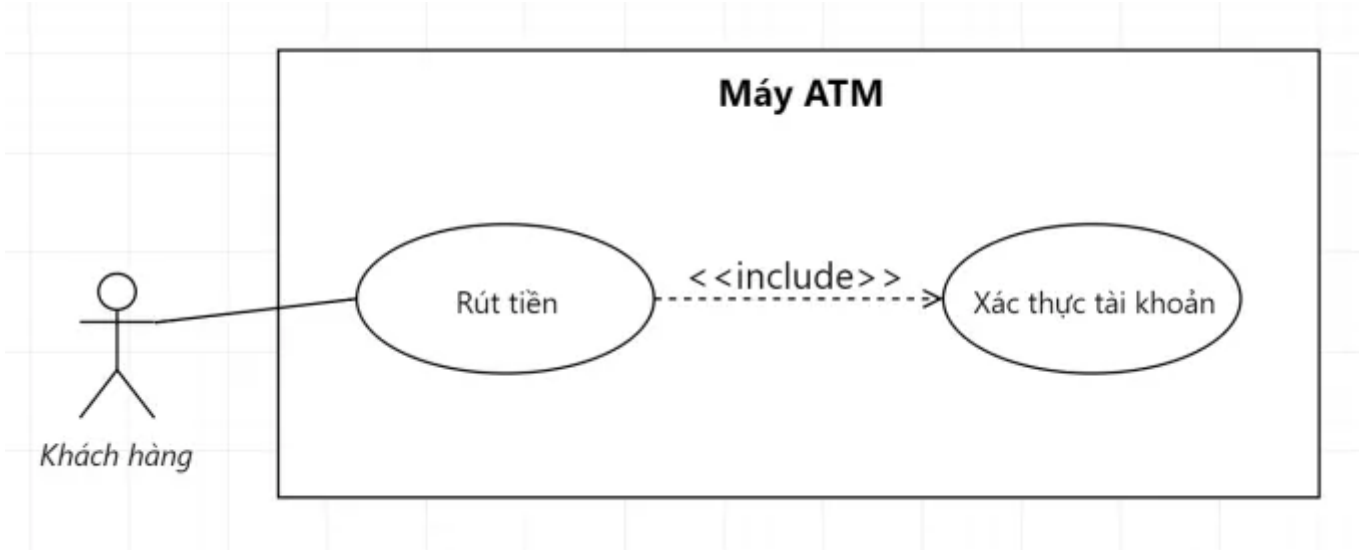
Rõ ràng anh em thấy: để nhận xét được một bài viết, anh em cần phải đăng nhập vào 1 tài khoản nào đó, để blog nhận diện anh em là ai, tên gì, quê quán, giai gái ra sao.

Ví dụ ở blog mình là anh em sẽ cần đăng nhập vào tài khoản WordPress. Sau khi đăng nhập xong, anh em phải soạn thảo nhận xét, tức là gõ nhận xét, chỉnh sửa, xóa tới xóa lui. Sau khi viết xong nhận xét, anh em sẽ bấm nút Submit để hoàn thành chẳng hạn.

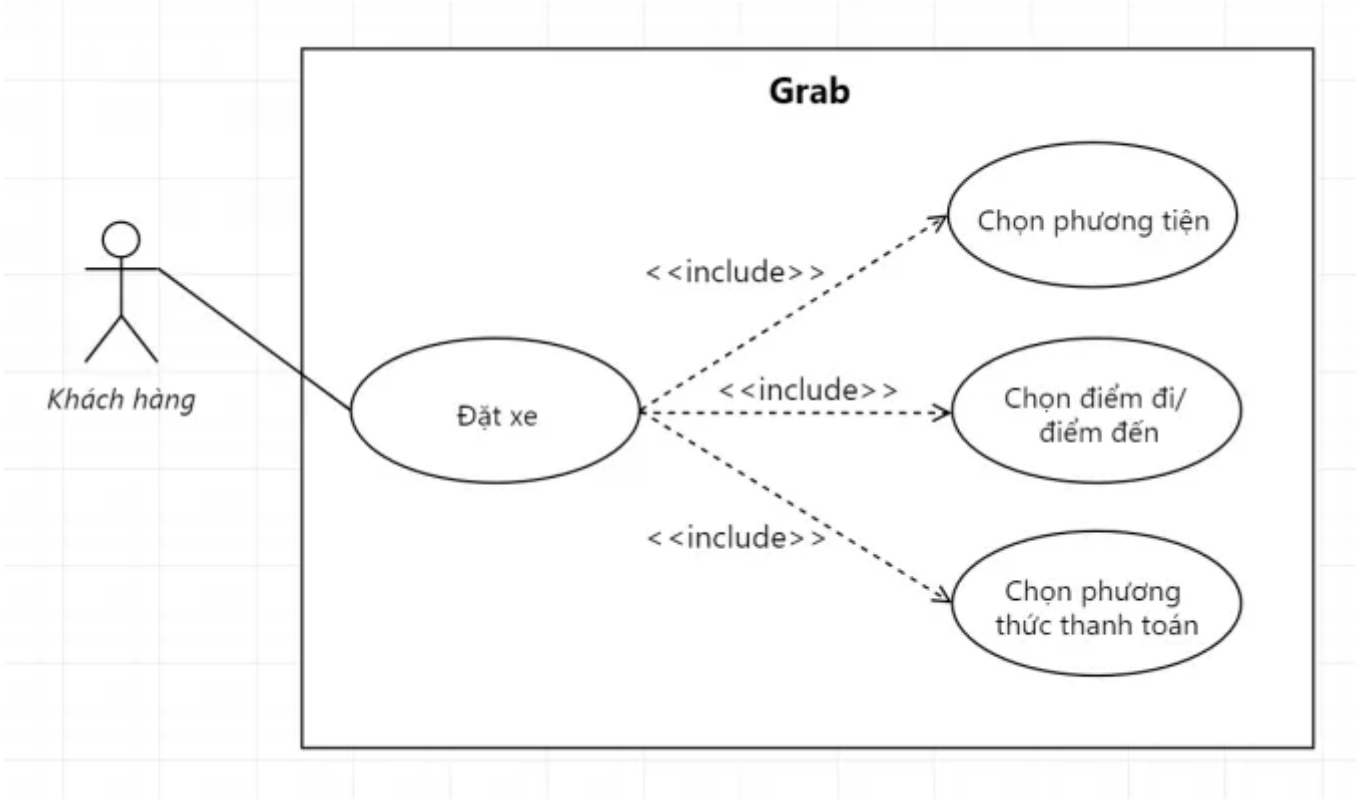
Chỉ khi nào xong 2 bước trên (*đăng nhập* và *soạn thảo nhận xét*), thì anh em mới có thể xong bước *Nhận xét bài notes* được.

Hay nói cách khác **để *Use Case: Nhận xét bài notes xảy ra*, thì *Use Case: Đăng nhập WordPress* và *Use Case: Soạn thảo nhận xét* phải bắt buộc hoàn thành trước tiên.**

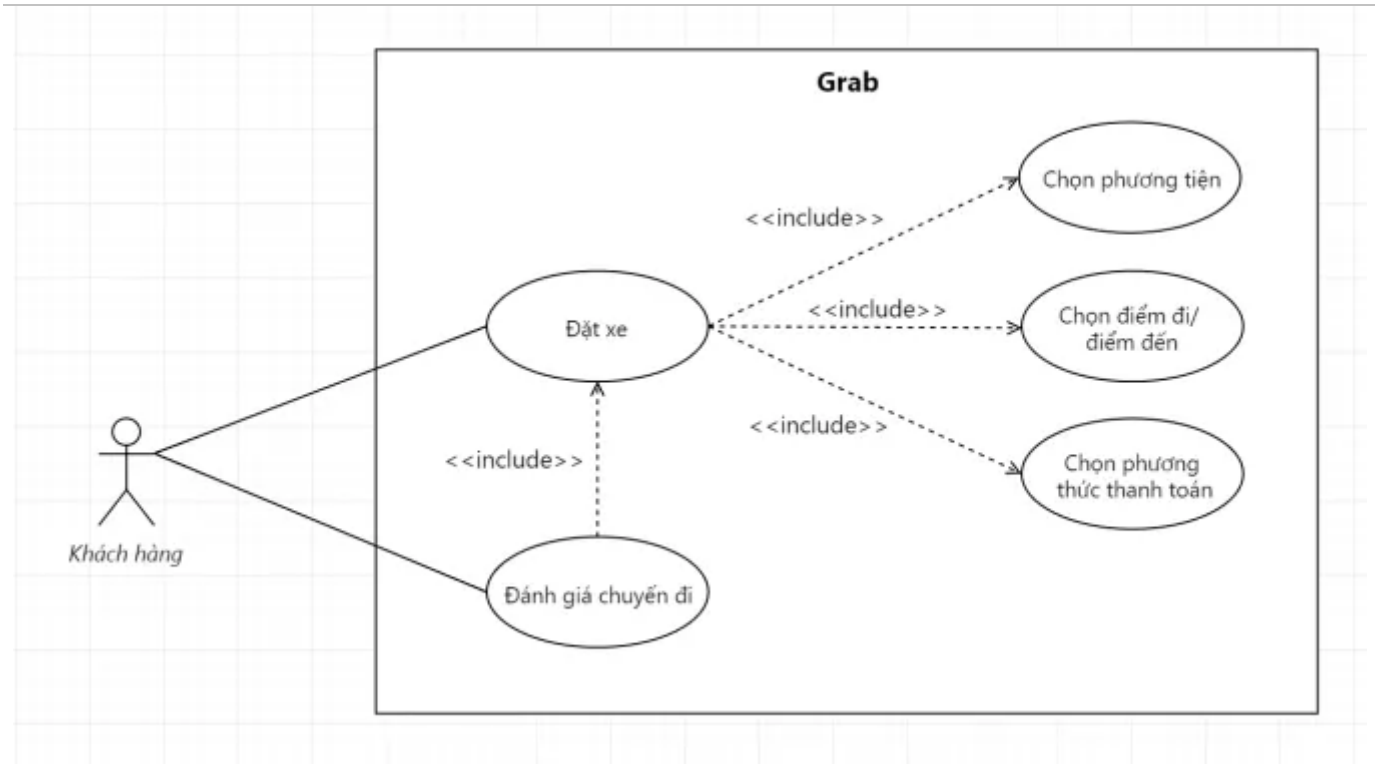
Đó chính là mối quan hệ Include. Anh em xem tiếp 1 số ví dụ dưới cho dễ hình dung nhé.



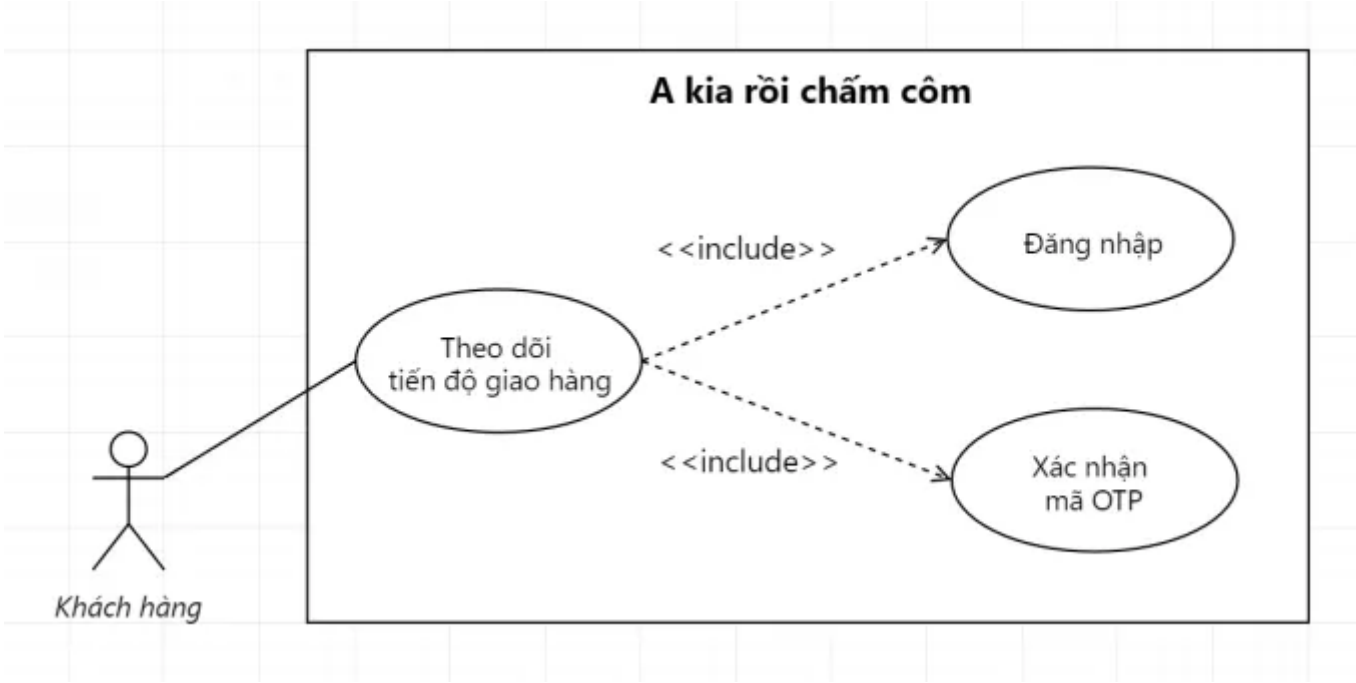
Muốn rút được tiền thì đầu tiên khách hàng phải xác thực tài khoản đi cái đã.



Muốn đặt được xe thì phải hoàn thành được 3 bước này rồi hệ thống mới cho đặt.



Hoặc khách hàng muốn đánh giá được chuyến đi thì trước đó họ phải đặt xe cái đã.



Hoặc tương tự là Use Case thể hiện tính năng Theo dõi tiến độ giao hàng trên một trang e-Commerce bất kỳ.

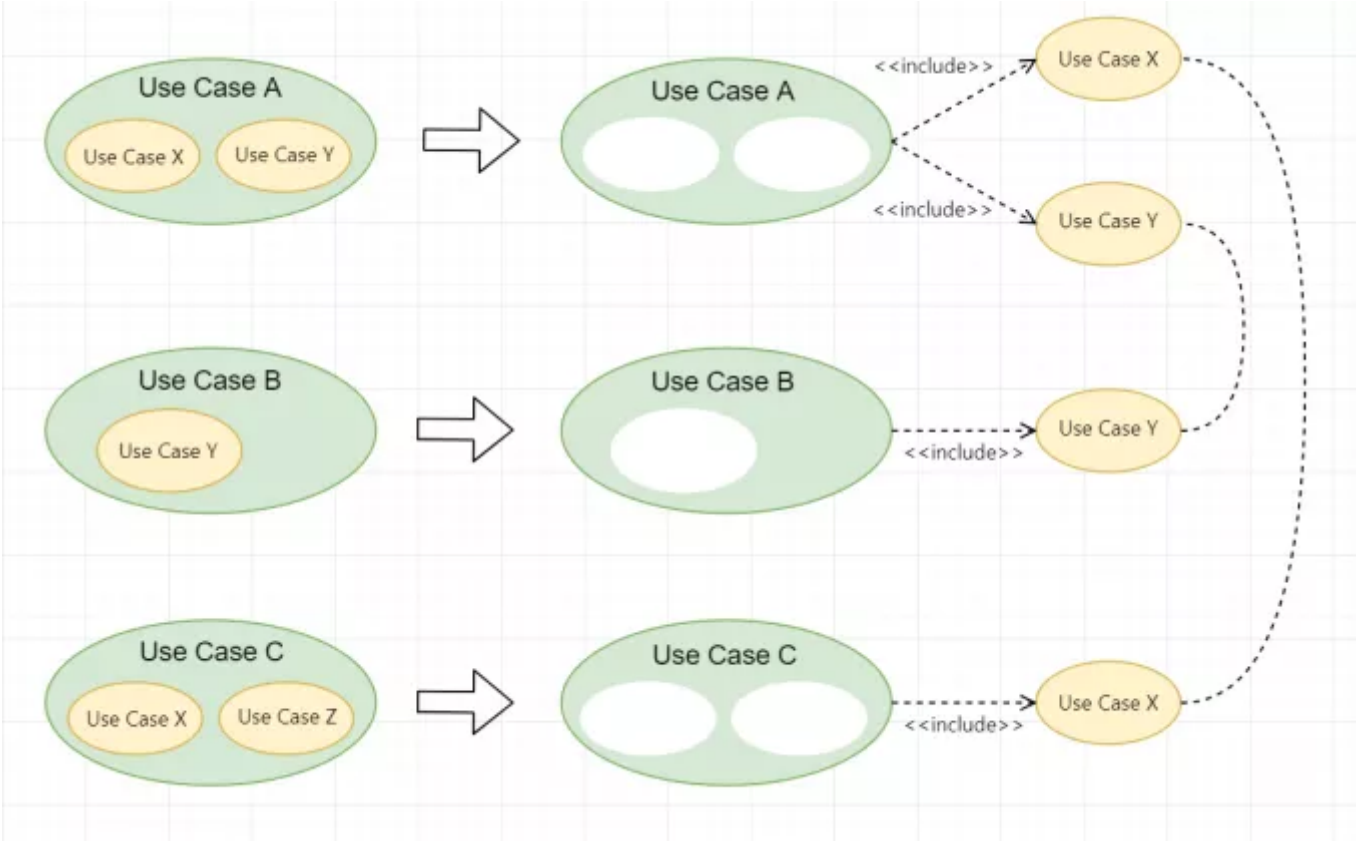
Một số điểm cần chú ý khi vẽ Include cho Use Case

Thực sự không có quy tắc nào rõ ràng cho việc *khi nào cần tách Use Case ra thành các Use Case nhỏ và cho nó một mối quan hệ Include cả.*

Việc tách hay không tách phụ thuộc duy nhất vào người vẽ. Và lý do lớn nhất để mối quan hệ Include ra đời là giúp cho các Use Case của chúng ta **DỄ QUẢN LÝ** hơn; làm cho Use Case Diagram trông có vẻ nguy hiểm hơn mà thôi 😇

Và anh em chỉ nên tách Use Case khi nó có ***độ phức tạp lớn*** và những thứ tách ra được có thể được ***tận dụng ở các Use Case sau này***.

Độ phức tạp lớn thì khi tách ra mình mới có được những Use Case vừa phải, đủ để diễn đạt dễ hiểu cho các stakeholders. Còn tận dụng được ở các Use Case sau là sao?



Sử dụng Include như thế nào cho hợp lý?

Ví dụ Use Case A gồm 2 Use Case nhỏ bên trong là X và Y. Do đó Use Case A được tách thành Use Case X và Use Case Y.

Tương tự, Use Case B gồm Use Case Y bên trong, nên được tách thành Use Case Y.

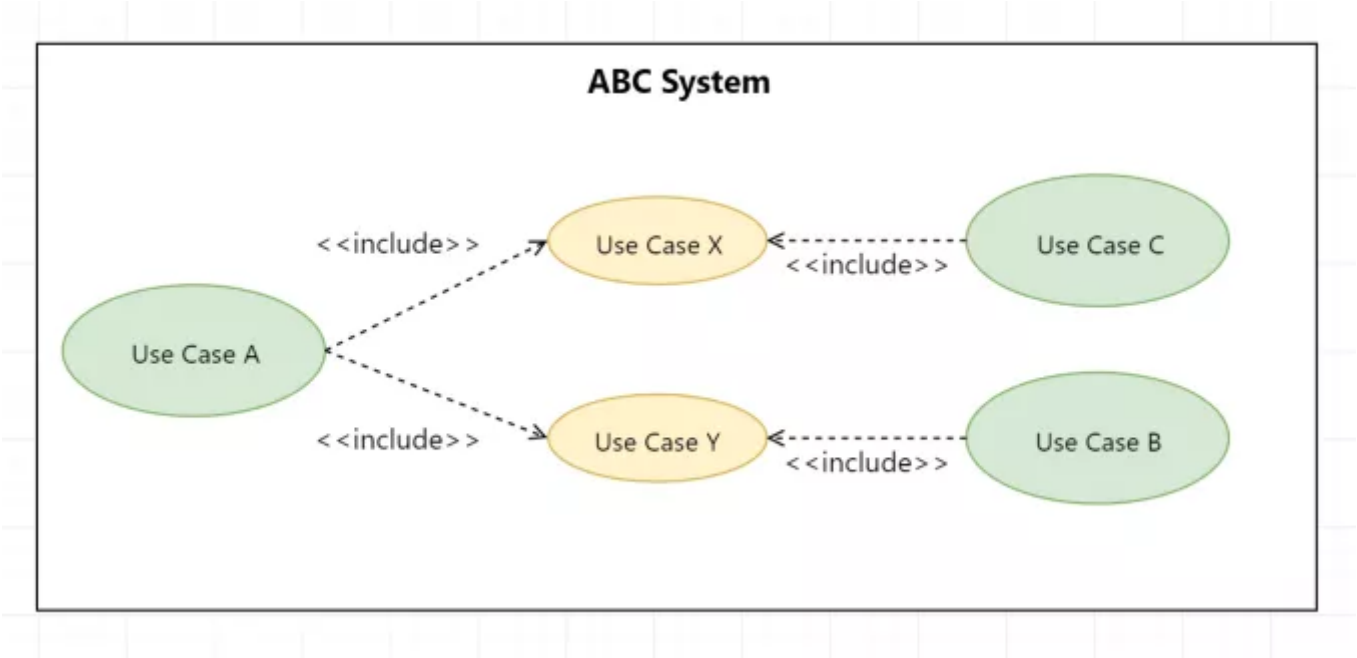
Nhưng, Use Case C gồm Use Case X và Use Case Z bên trong, nhưng chỉ có Use Case X là được tách ra cho mối quan hệ Include. Vì có thể Use Case Z *"không đáng" để tách ra* thành một Use Case nhỏ hơn.

Chúng ta tách Use Case X từ Use Case A để Use Case C có thể tận dụng được mà không cần vẽ lại. Tương tự, tách Use Cas Y từ Use Case B để Use Case A có thể tận dụng mà cũng không cần vẽ lại.

Điều này giúp Use Case Diagram của chúng ta trở nên **chặt chẽ**, **logic** và **gọn nhẹ** hơn rất nhiều.

Còn Use Case Z, vì nó không được *"dùng lại"* ở một Use Case bất kỳ nào sau đó, nên người vẽ có thể cân nhắc có tách nó ra hay không!

Nếu Use Case đó đủ lớn và khá là high-level, thì có lẽ chúng ta nên tách. Còn nếu ngược lại, Use Case đã rõ ràng, là một requirement từ phía User cụ thể thì *không đáng* để anh em tách nó ra thành một Use Case nhỏ, chỉ làm hình thêm thêm rối mà thôi.



Khi tách Use Case ra thành các Use Case nhỏ để tận dụng mối quan hệ Include, anh em hãy nhớ 2 thứ: **độ lớn** của Use Case và **khả năng tái sử dụng** lại của nó.

Còn cách vẽ thì anh em cứ nhớ là include tới thă`ng nào thì dâ´u mũi tên hướng tới thă`ng đó nhé anh em. Nhớ để qua phâ`n Extend cho khỏi lộn.

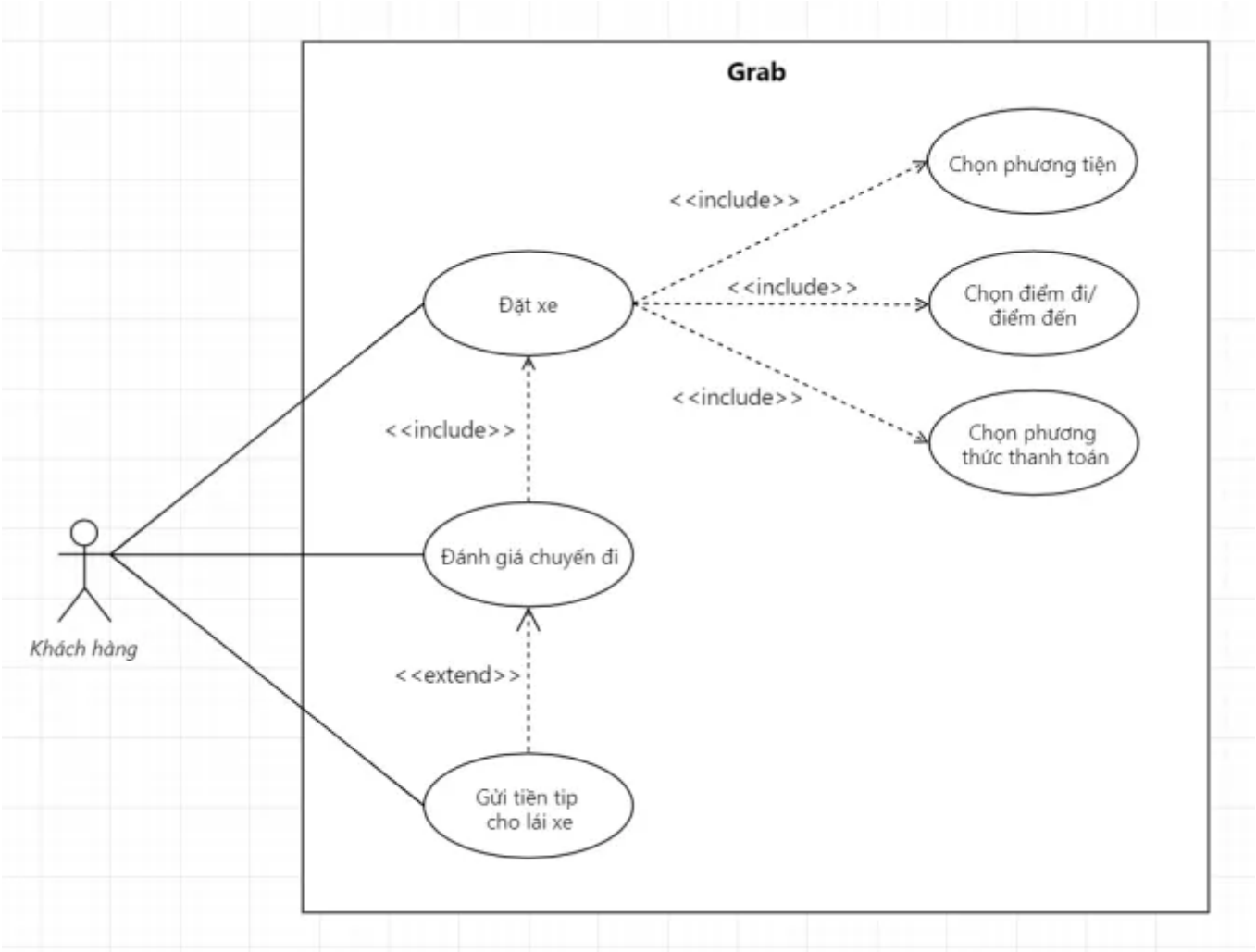
b) Extend

Extend là mô´i quan hệ **mở rộng** giữa các Use Case với nhau.

Nê´u Include là mô´i quan hệ bă´t buộc, thì Extend là một mô´i quan hệ **không bă´t buộc.** Nó thể hiện mô´i quan hệ có thể có hoặc có thể không giữa các Use Case với nhau.

Một Use Case B là extend của Use Case A thì có nghĩa Use Case B chỉ là một thứ optional, và chỉ xảy ra trong một hoàn cảnh cụ thể nào đó.

Lâ´y ví dụ Grab phía trên, anh em sẽ dễ dàng có được một mô´i quan hệ Extend như sau.



Ví dụ về mối quan hệ Extend giữa các Use Case

Trong trường hợp này, ***Use Case: Gửi tiền tip cho lái xe*** là một Use Case có mối quan hệ Extend với ***Use Case: Đánh giá chuyến đi***. Tức, *Use Case: Gửi tiền tip* cho lái xe chỉ là một Use Case có thể xảy ra, hoặc không; và nó **liên quan đến** *Use Case: Đánh giá chuyến đi*, chứ không phải bất kỳ một Use Case nào khác.

À...à...Nhắc tới lúc có lúc không, tức là nhắc tới điều kiện xảy ra.

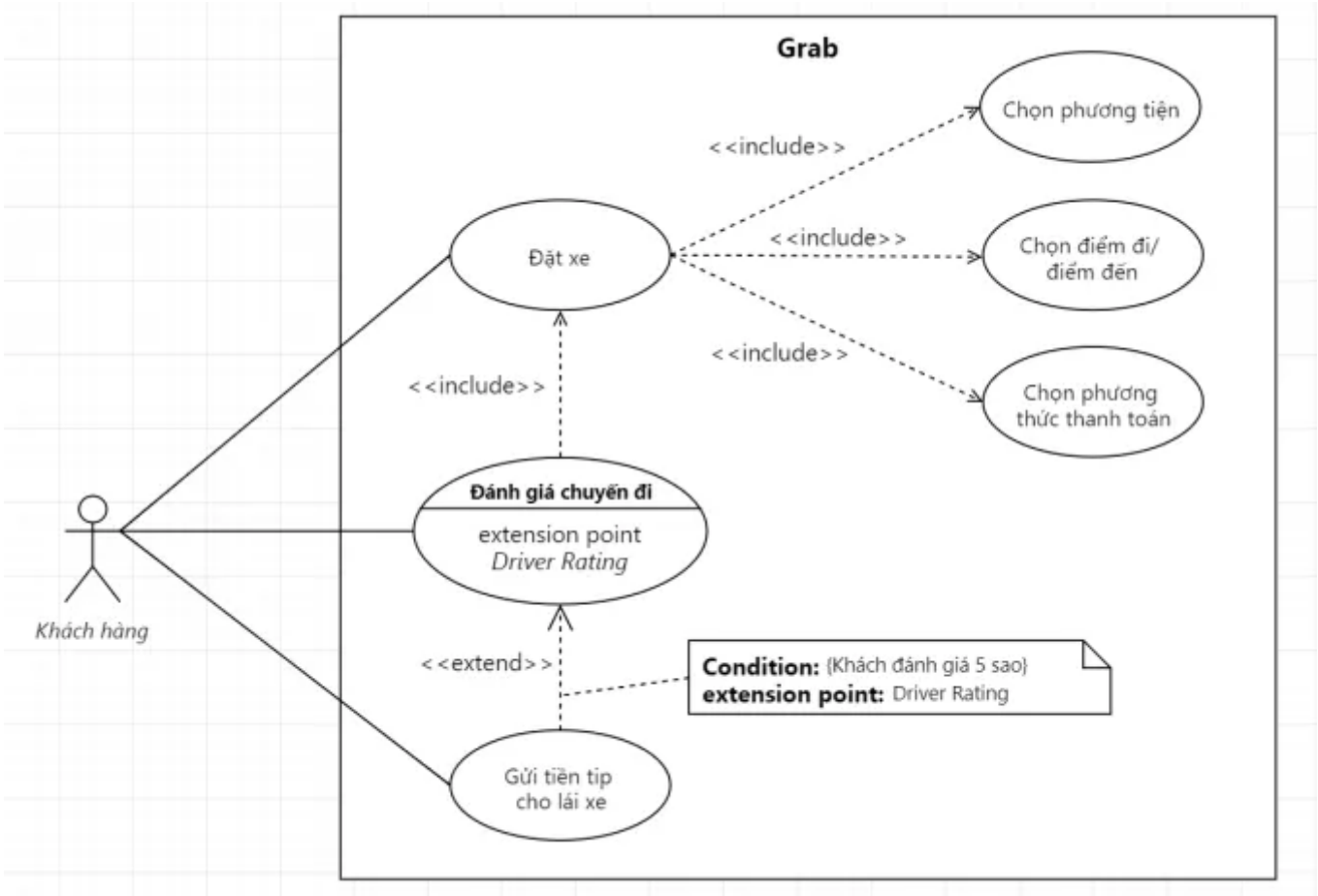
Anh em có thể thể hiện rõ ý chỗ này bằng một thứ luôn đi kèm với Extend, đó là **Extension Point** 🤓

Extension Point nôm na là điều kiện mà Use Case có mối quan hệ Extend sẽ xảy ra. Còn để sát nghĩa thì anh em có thể hiểu chữ Point ở đây nghĩa là *điểm dữ liệu thể hiện sự khác biệt*.

Tức nếu dữ liệu này là A thì Use Case không xảy ra, nhưng nếu dữ liệu này là B thì Use Case sẽ xảy ra.

//Theo mình nhớ là hình như anh em chỉ có thể gửi tiền tip cho tài xế, nếu cuốc xe đó anh em chấm họ maximum là 5 sao.//

Vậy thì anh em sẽ vẽ Use Case Diagram chỗ đó như sau.



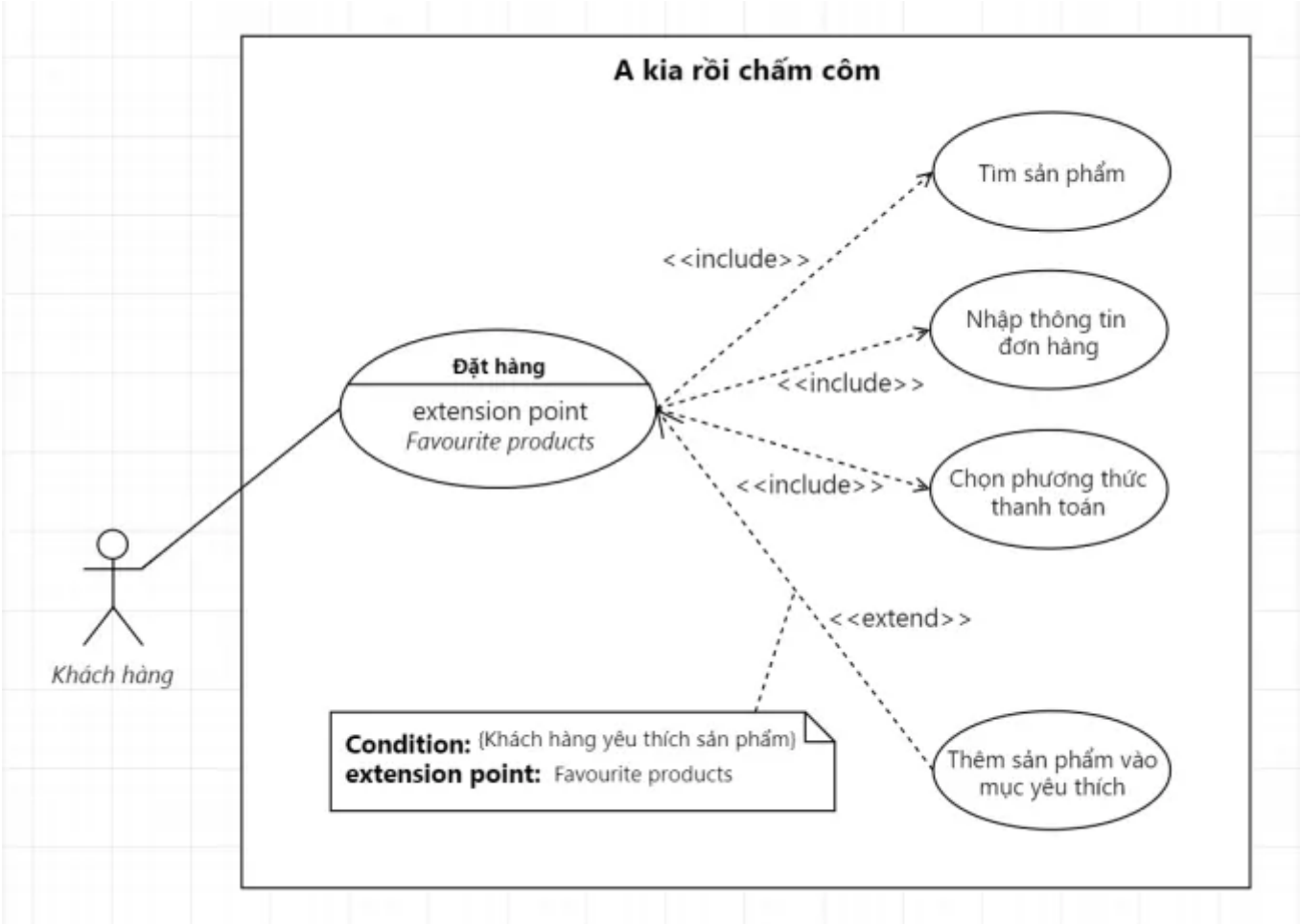
Use Case Diagram có thể hiện rõ khi nào thì mối quan hệ Extend diễn ra.

Extension Point ở đây là dữ liệu Driver Rating. Nê´u Driver Rating đạt giá trị 5 sao, thì *Use Case: Gửi tiê`n tip cho lái xe* sẽ xảy ra, và hoàn toàn optional, tùy thuộc vào khách hàng.

Và nó liên quan mật thiê´t đê´n *Use Case: Đánh giá chuyê´n đi*, là một phâ`n mở rộng của *Use Case: Đánh giá chuyê´n đi*.

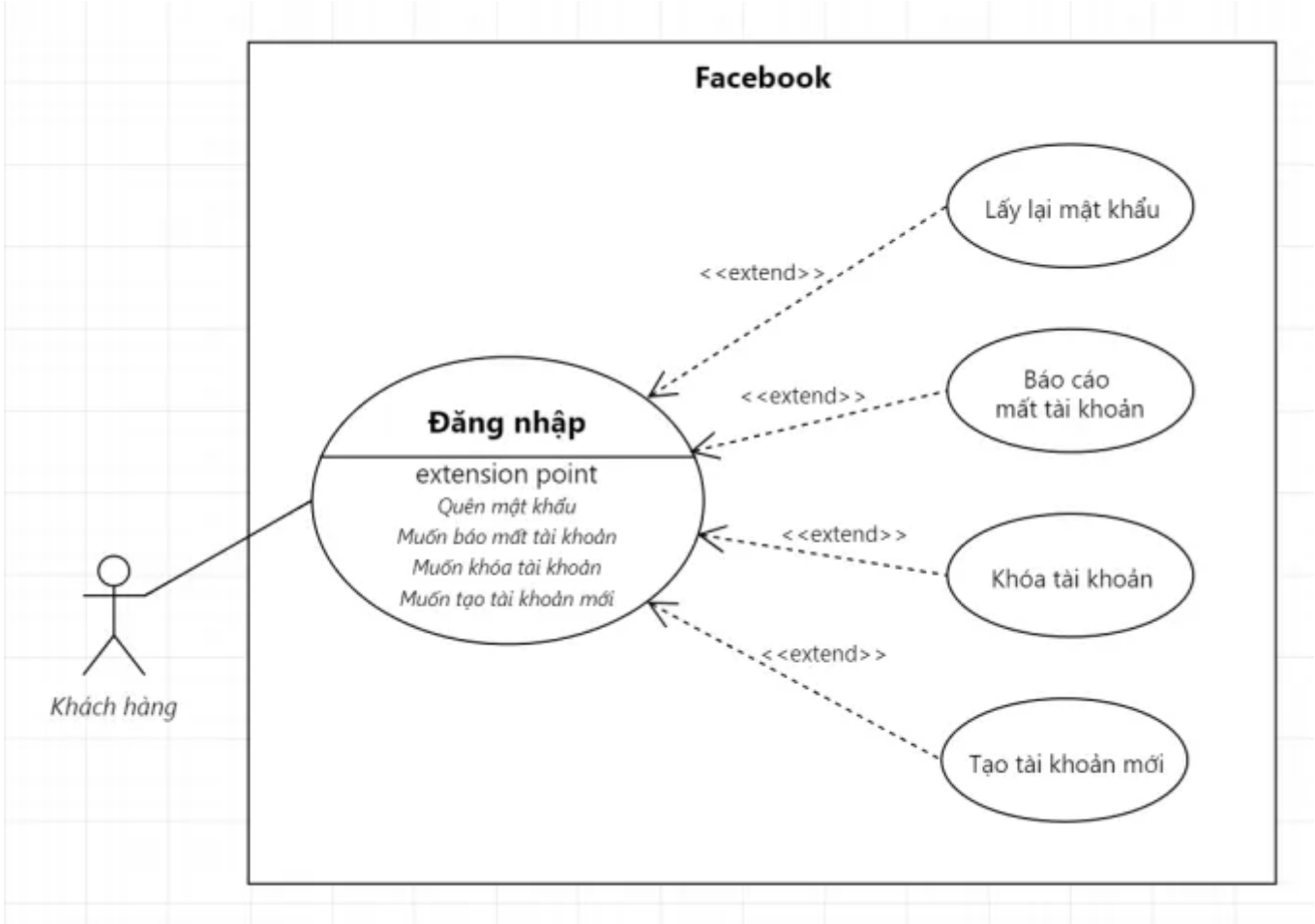
Extension Point không nhâ´t thiê´t phải là một dữ liệu nào đó trên hệ thô´ng, mà có thể là một "điê`u kiện" bâ´t kỳ, miễn là nó thể hiện được trường hợp cụ thể mà Use Case sẽ xảy ra.

Ở một ví dụ khác.



Mối quan hệ Extend trong Use Case Diagram của A kia rồi chấm côm.

Còn nếu Use Case có quá nhiều mối quan hệ Extend, làm cho Diagram nhìn rối bời quá, anh em có thể bỏ luôn phần comment của Extension Point luôn cũng được.



Vẽ vầy cũng hông sao.

---

c) Generalization

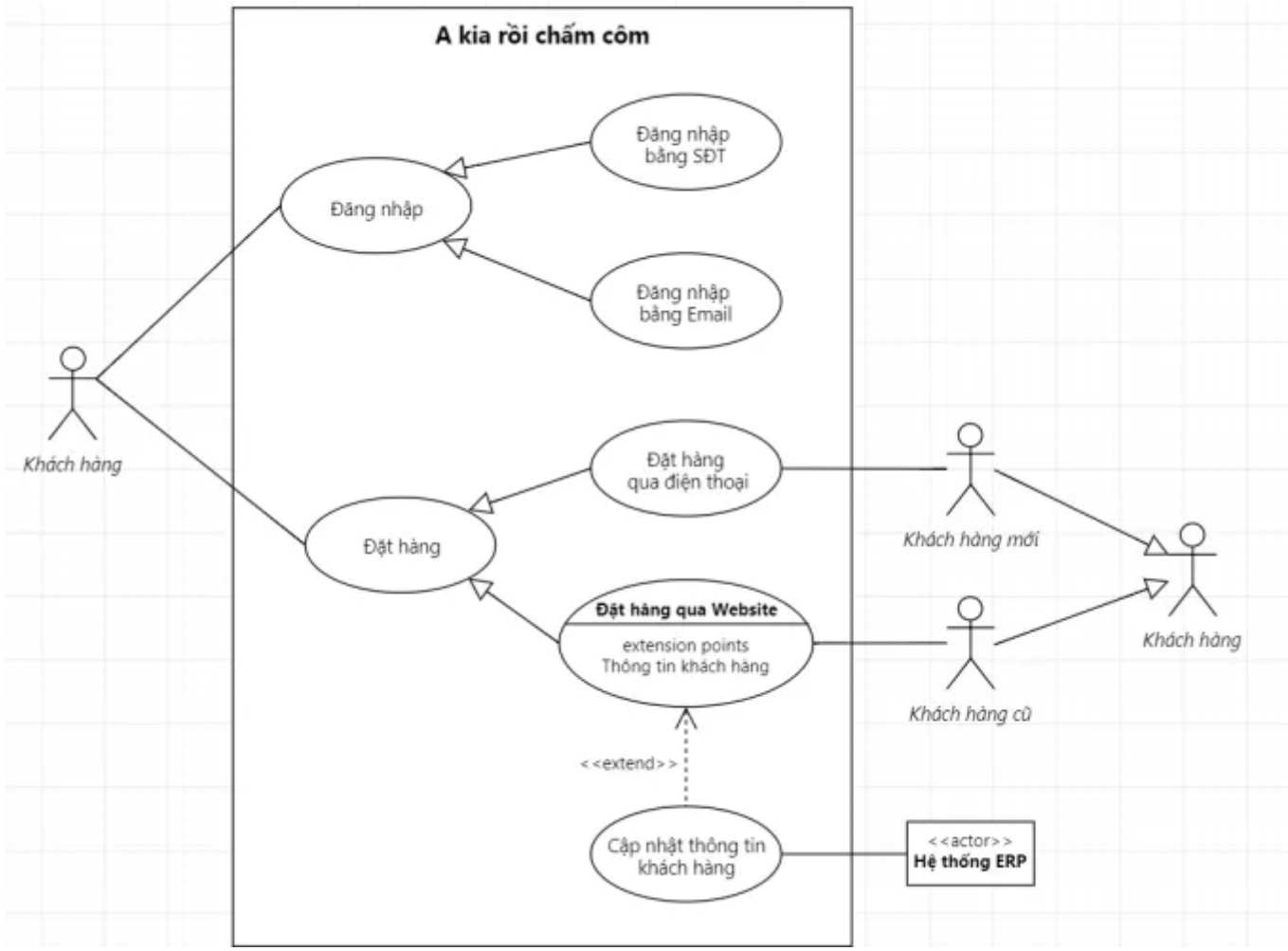
Generalization đơn giản là quan hệ cha con giữa các Use Case với nhau. Nhưng khác biệt với Include và Extend là nó còn được dùng để thể hiện mô´i quan hệ giữa các... Actor với nhau.

Đâ`u tiên là mô´i quan hệ cha-con giữa các Use Case. Ví dụ:

- Đăng nhập thì có thể đăng nhập qua sô´ điện thoại, hoặc đăng nhập qua email.
- Đặt hàng thì có đặt hàng qua điện thoại, hoặc đặt hàng qua website.
- Thanh toán thì thanh toán qua thẻ ATM, qua thẻ thanh toán quô´c tê´, hoặc qua ví điện tử.
- Hoặc tìm kiê´m thì có thể tìm kiê´m bă`ng từ khóa, hoặc tìm kiê´m theo nhóm sản phẩm.

Hoặc mô´i quan hệ cha-con giữa các Actor. Ví dụ:

- Khách hàng gô`m khách hàng cũ và khách hàng mới
- Hoặc Vendor thì có thể gô`m Retailers và Wholesalers.



Ví dụ về quan hệ cha-con (Generalization) trong Use Case.

Nhìn chung, Generalization giúp anh em **thể hiện rõ hơn** các yêu cầu bằng việc *gom nhóm các Use Case* lại theo quan hệ cha-con. Cá nhân mình thì rất ít khi vẽ relationship này, chủ yếu chỉ dùng Include và Extend là chính.

Còn một điểm nữa là Generalization có **tính kế thừa**. Tức thằng cha có gì thì thằng con có cái đó, kể cả Use Case hay Actor.

Ví dụ Use Case A có include đến Use Case B và C. Thì Use Case A' là con của Use Case A cũng sẽ có mối quan hệ Include đến Use Case B và C, mặc dù không được thể hiện trên hình.

...

Ô kê, vậy là xong phần Relationship – một trong những phần chuối nhất, dễ lộn nhất trong Use Case. Hi vọng những ví dụ trên giúp anh em hiểu được cụ thể như thế nào là Include, Extend và Generalization trong một Use Case Diagram 🤓

3. Một số sai lầm phổ biến khi vẽ Use Case

Use Case Diagram là thứ để anh thể hiện được requirement của khách hàng.

Vẽ sao mà khách hàng nhìn vô một phát là thấy khoái liền. Khách hàng mà chân nhịp nhịp, miệng lẩm bẩm: *"Đúng rồi...đúng rồi..., tính năng này có,... tính năng kia có luôn, à... tích hợp lấy dữ liệu này có, ô kê ô kê,... vậy là đủ rồi!"*, thì coi như anh em đã vẽ khá good ☺

Chém nãy giờ mạnh vậy chứ mình vẽ chẳng bao giờ là ổn cả. PM cứ phải duyệt đi duyệt lại cả chục lần. Nhờ đó mới có những sai lầm phổ biến mà mình hay gặp khi vẽ Use Case Diagram cho anh em tham khảo dưới đây, hehe.

3.1. Chuyện đặt tên

Trong mô hình hóa, chuyện đặt tên là rất-rất-rất quan trọng.

Vì đã nói *"mô-hình-hóa"* tức là chúng ta dùng hình ảnh để nói chuyện, thì khi đó hàm lượng chữ chiếm rất ít. Và chính vì nó ít, nên những gì chúng ta ghi trên diagram phải rất súc tích, cô đọng và có giá trị ngay tức thì.

Chỉ cần người đọc họ nhìn vô diagram mà thấy ngay 1 dòng chữ khó hiểu, thì ngay lập tức tụt bà nó hết mood, hết muốn xem tiếp rồi.

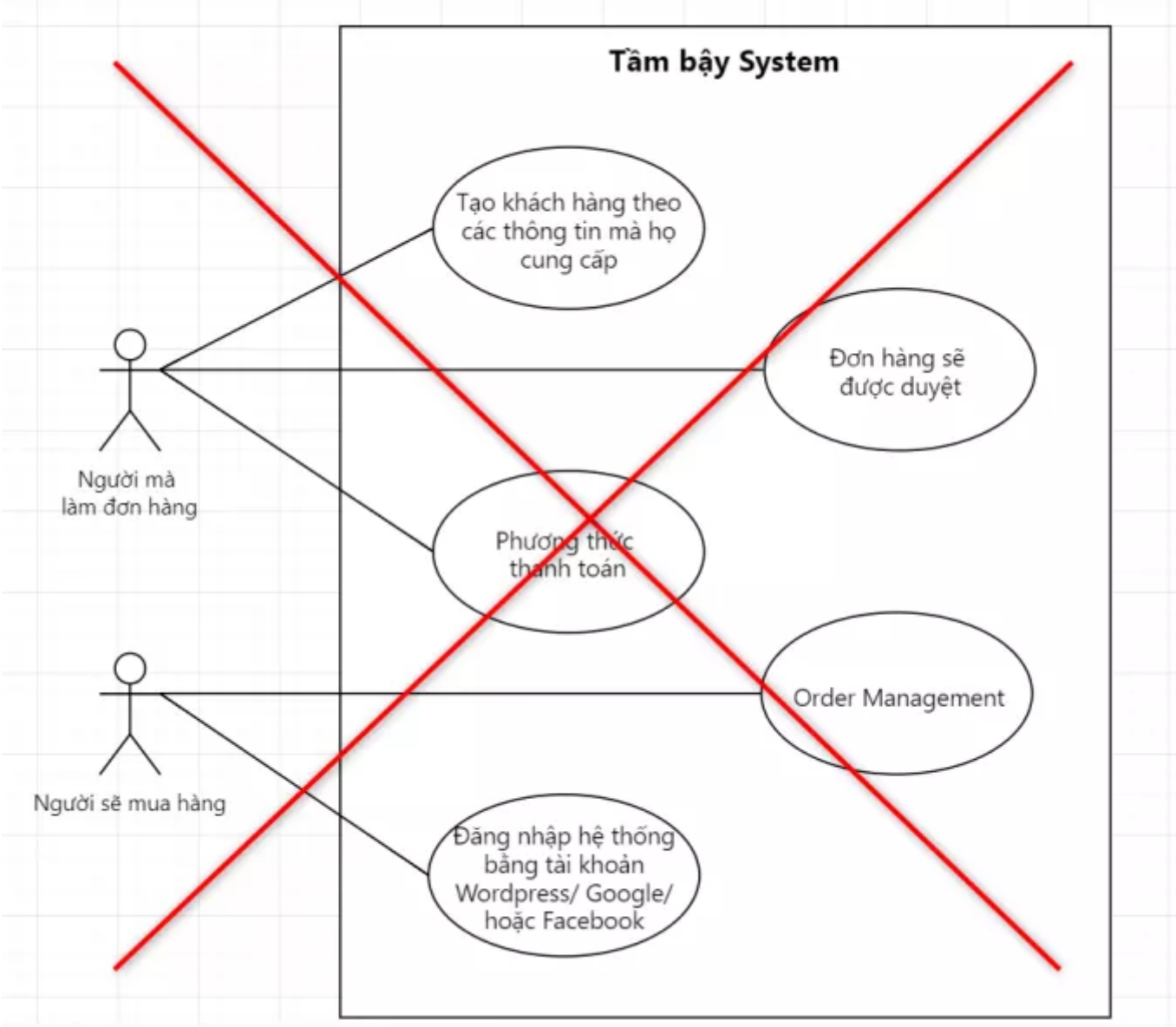
Nói về Use Case thì có 1 vài lưu ý sau cho anh em:

- Actor thì phải đặt tên bằng danh từ, không dùng động từ, và cũng không mệnh đề quan hệ gì hết.
- Tên Use Case thì phải ghi rõ ràng, rành mạch, đẹp nhất là dưới format: ***Verb*** **+** ***Noun*****.**

Ví dụ: *Đổi điểm thành viên, Chuyển tiền nội địa, Chuyển tiền quốc tế, Duyệt nhận xét bài viết.*

BA chúng ta vẽ Use Case nhằm mục đích diễn tả yêu cầu cho stakeholders hiểu, do đó anh em không được dùng những từ kỹ thuật trong đây, không thể hiện sự nguy hiểm ở đây, người ta đọc zô hông hiểu gì hết là trớt quớt.

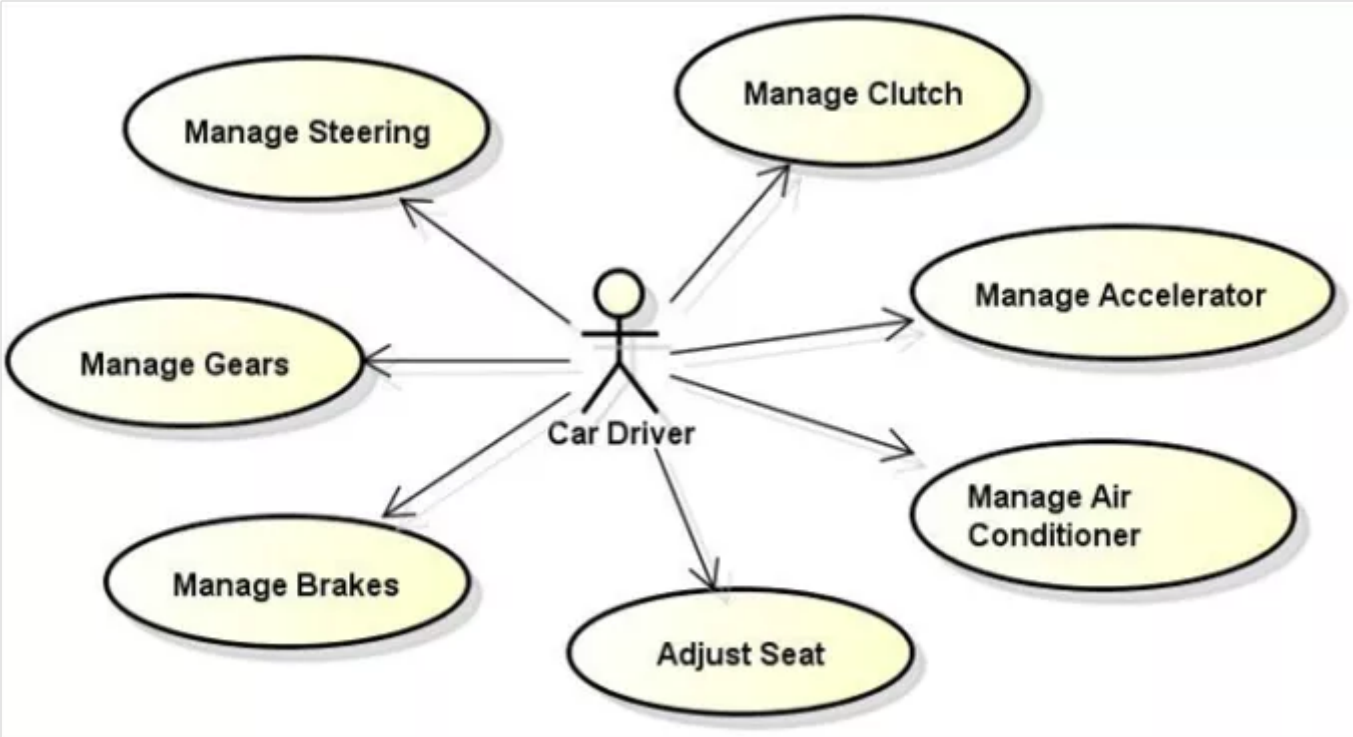
Và đặc biệt là tránh đặt tên quá dài và không nên dùng kiểu bị động.



Sai lầm 1: Chuyện đặt tên.

3.2. Vẽ Use Case mà thành phần rã chức năng

Đây chính xác là lỗi mà mình hay gặp nhất, rất thường xuyên gặp khi vẽ Use Case.



Sai lầm 2: không phân biệt được phân rã chức năng và Use Case (Nguồn ảnh: ModernAnalyst.com)

Dấu hiệu nhận biết rõ ràng nhất là khi Use Case Diagram của anh em đầy rẫy chữ *"manage"*, manage cái này, manage cái kia...

Đầu tiên là chữ Manage rất rộng nghĩa. Yêu cầu quản lý A gồm 5 việc, thì không có nghĩa yêu cầu quản lý B cũng gồm 5 việc. Use Case là diagram thể hiện yêu cầu của End-Users, nhằm **đạt được một mục đích** nào đó.

Ở ví dụ trên, nếu nói Manage Gears, Manage Brakes, hay Manage Air Conditioner thì quá tối nghĩa, chả ai hiểu nhằm mục đích sau cùng là để làm gì.

Thứ hai, hình minh họa trên vẽ Use Case nhưng lại **chưa mang được góc nhìn của End-Users**, tức chưa cho thấy được End-Users muốn đạt được gì sau ngần ấy Use Case được liệt kê ra.

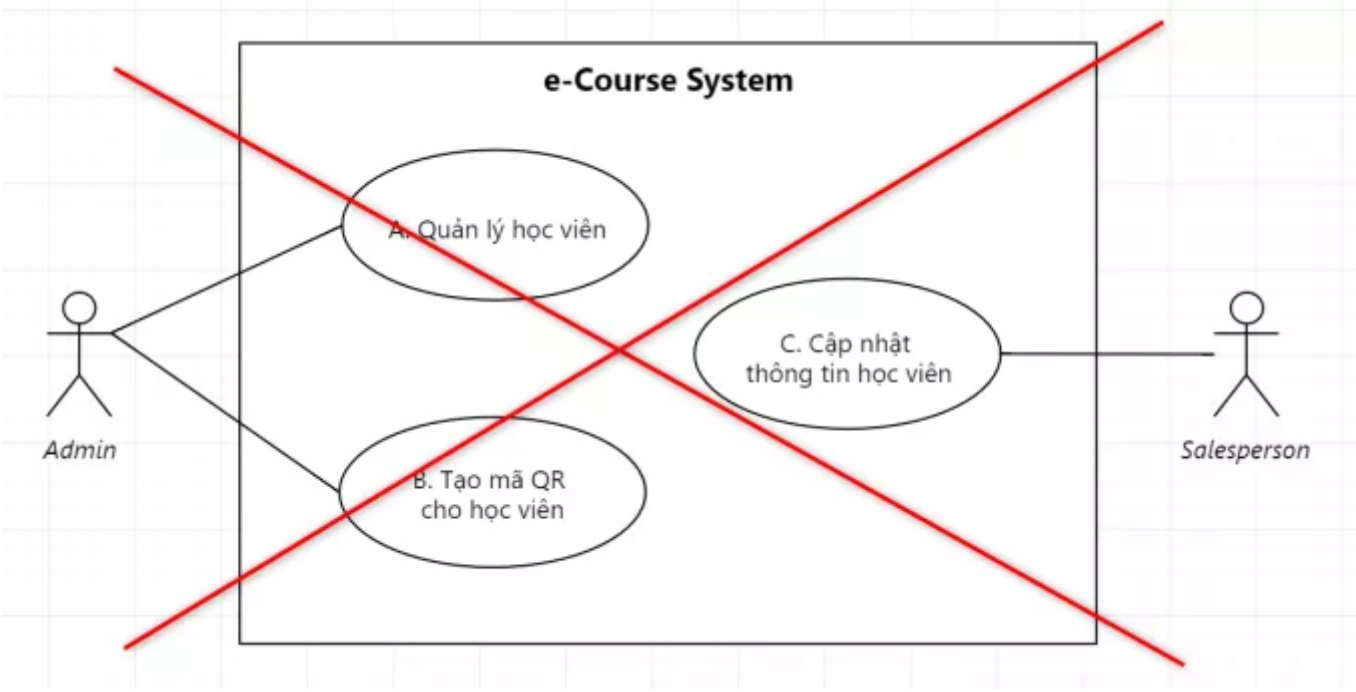
Nguyên nhân có thể do *người vẽ chưa nắm đủ thông tin* về yêu cầu của End-Users, ảnh chưa hiểu rõ rốt cuộc thì người dùng họ muốn làm gì trên hệ thống, hay hệ thống phải tương tác những gì với hệ thống khác.

Từ đó mới có chuyện anh em nhìn vô Use Case Diagram ở trên mà cảm thấy mông lung như một trò đùa. Do đó, chúng ta chỉ vẽ Use Case khi đã có đủ thông tin cần thiết:

- *End-users muốn làm gì? Nhằm mục đích gì?* ==> tương tác giữa end-users và hệ thống
- *Hệ thống phải nhận/ lấy data từ những nguồn nào?* ==> tương tác giữa hệ thống với những hệ thống bên ngoài khác.

Ngoài ra, khi đã có đủ thông tin nhưng Use Case mình vẽ vẫn bị confuse. Lý do có thể do các Use Case mình vẽ bị lệch các cấp độ Requirement với nhau.

Ví dụ Use Case A thì thể hiện Business Requirement, tức là rất high level. Nhưng sang Use Case B và C thì lại nói rất detail tới mức Solution Requirement như.



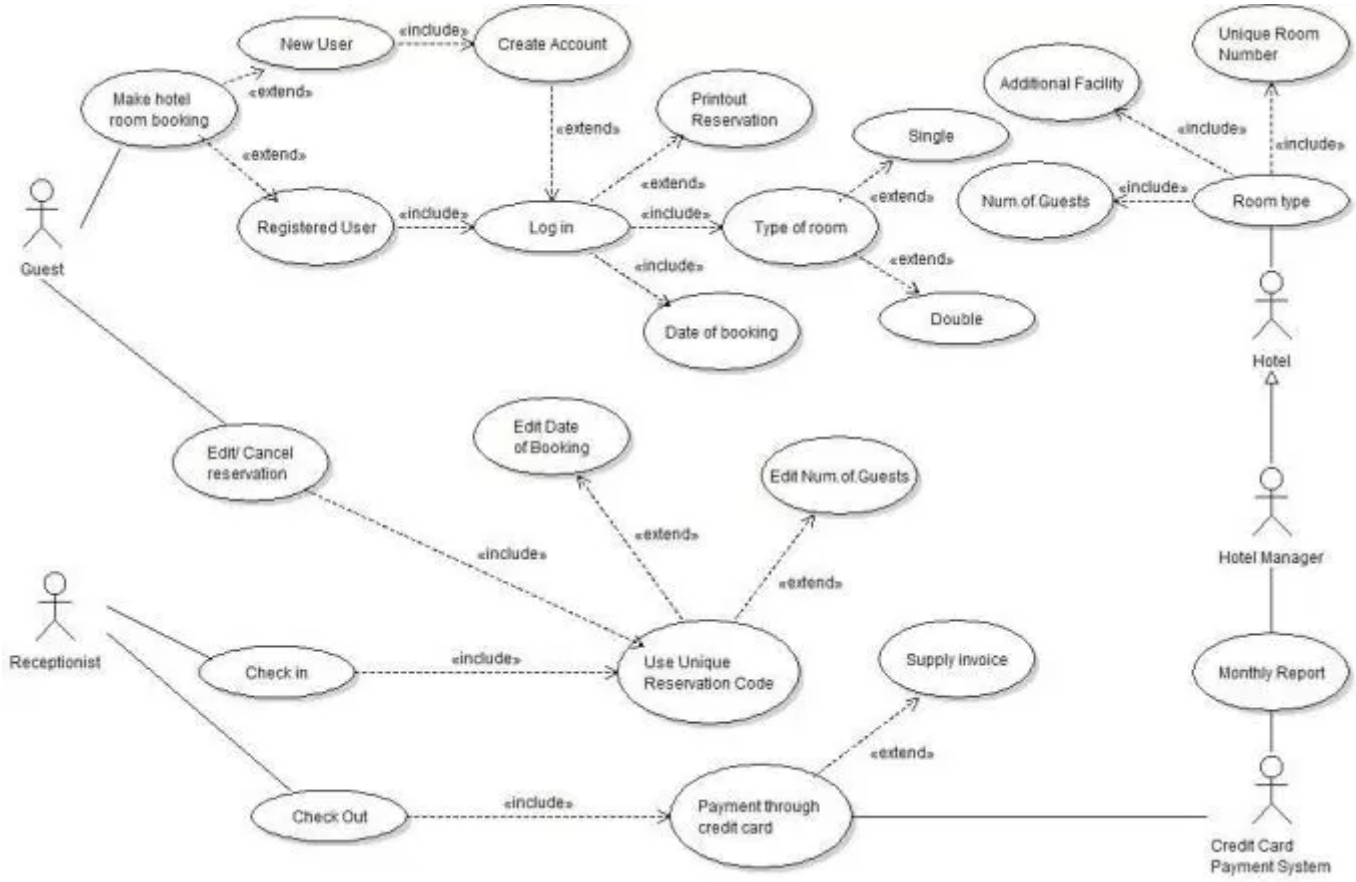
3 Use Case này bị lệch cấp độ với nhau, gây "rối bời" cho người xem.

Để sửa lại Use Case trên, đơn giản mình chỉ cần bỏ *Use Case A: Quản lý học viên* ra, vì nó là thứ rất chung chung, không thể hiện được mục đích cụ thể, so với 2 Use Case còn lại.

Tuy nhiên, ***chữ "Manage" trong Use Case lại rất công dụng,*** *công dụng đến mức không thể không dùng trong các document mình làm, nó sẽ giúp mình giải quyết vấn đề ở mục số 3.4 phía dưới, đọc tiếp nhé anh em.*

3.3. Rối nùi Use Case

Anh em tham khảo một số hình sau sẽ rõ.



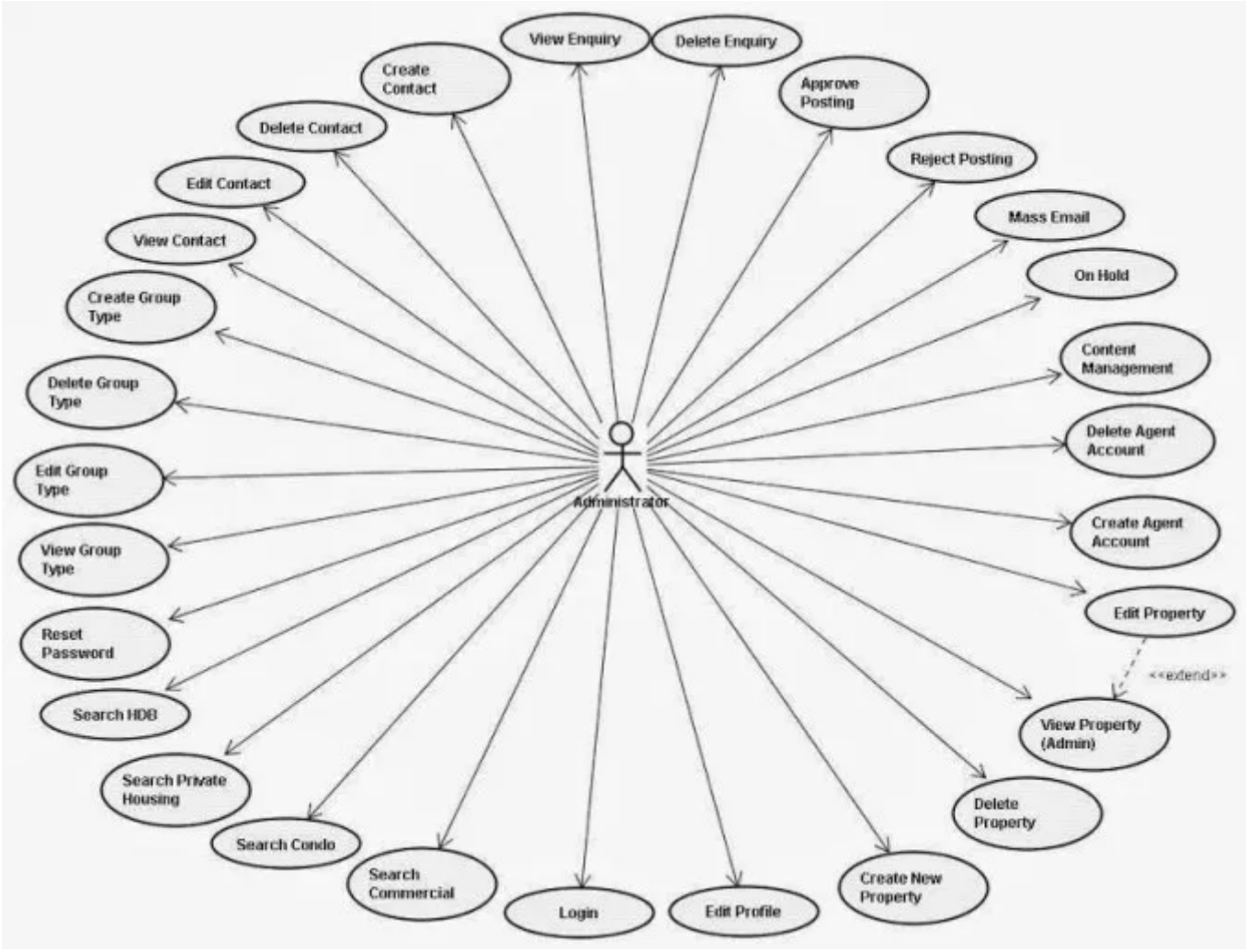
Sai lầm 3: Rối nùi Use Case (Nguồn hình: stackoverflow)

Vấn đề của hình này là ôm đồm quá nhiều. Dẫn đến quá nhiều Use Case xuất hiện trong cùng một Diagram, đã vậy cũng không có Boundary of System rõ ràng.

Như anh em thấy, Use Case này vẽ rất sai ở những điểm như sau:

- *Xác định sai Use Case (nên mới nhiều UC như vậy): những thứ như single, double, num of guest… rõ ràng đâu phải là một Use Case, đâu phải là một sự tương tác.*
- *Đặt tên Use Case sai: quá nhiều cụm danh từ cho Use Case.*
- *Không có Boundary of System.*
- *Những Use Case có extend không ghi chú cụ thể điều kiện khi nào thì UC extend xảy ra.*

Một note nhỏ quan trọng cho anh em, Use Case Diagram sạch đẹp là chỉ nên có trên dưới **10 Use Case** trong đó. Các Use Case còn lại anh em hãy dùng Boundary of System để phân chia theo phân hệ một cách hợp lý nhất có thể.



Nguồn hình: alexander-stoyan.blogspot.com

Hình này rõ ràng là quá thứ dữ. Thật ra trường hợp này cũng khá phổ biế´n, mình trước kia bị hoài. Mấ´u chố´t đế´n từ một số´ điề`u sau:

- Một số´ Use Case đặt tên sai
- **Chưa tận dụng các Relationship** để thể hiện, khiế´n cho các Use Case quá rời rạc nhau, và trông rấ´t không hợp logic.
- Người vẽ không dùng Boundary of System để phân nhóm, giới hạn các Use Case.
- Và đặc biệt, người vẽ quá chú trọng đế´n các chức năng cơ bản nhấ´t, đó là: CRUD – Create/Read/Update/Delete.

3.4. Quá chi tiế´t các chức năng CRUD

Như ví dụ trên, mỗi thực thể là một lầ`n CRUD. Như vậy quá tố´n effort, trong khi 96,69% là ở phân hệ nào, hay dữ liệu nào, anh em cũng đề`u cầ`n phải CRUD dữ liệu hế´t.

Điề`u này tạo ra một sự lặp đi lặp lại ở các Use Case Diagram, nhưng không thể hiện được gì nhiề`u cho người xem. Để giải quyế´t vấ´n đề` này, anh em có thể có làm 1 trong 2 cách sau.

Cách 1

Thêm một dòng note trước đoạn mô tả Use Case trong tài liệu: *"Toàn bộ dữ liệu đều có chức năng Thêm/ Đọc/ Sửa/ Xóa và chịu tác động bởi sự phân quyền từ phía Quản trị hệ thống"* hoặc đại loại vậy. Để cho các stakeholder biết được rằng hệ thống có chức năng CRUD các dữ liệu này.

Nhưng nên nhớ CRUD ở đây là đứng từ góc nhìn End-Users: ***hệ thống có cho phép End-Users CRUD dữ liệu hay không?***

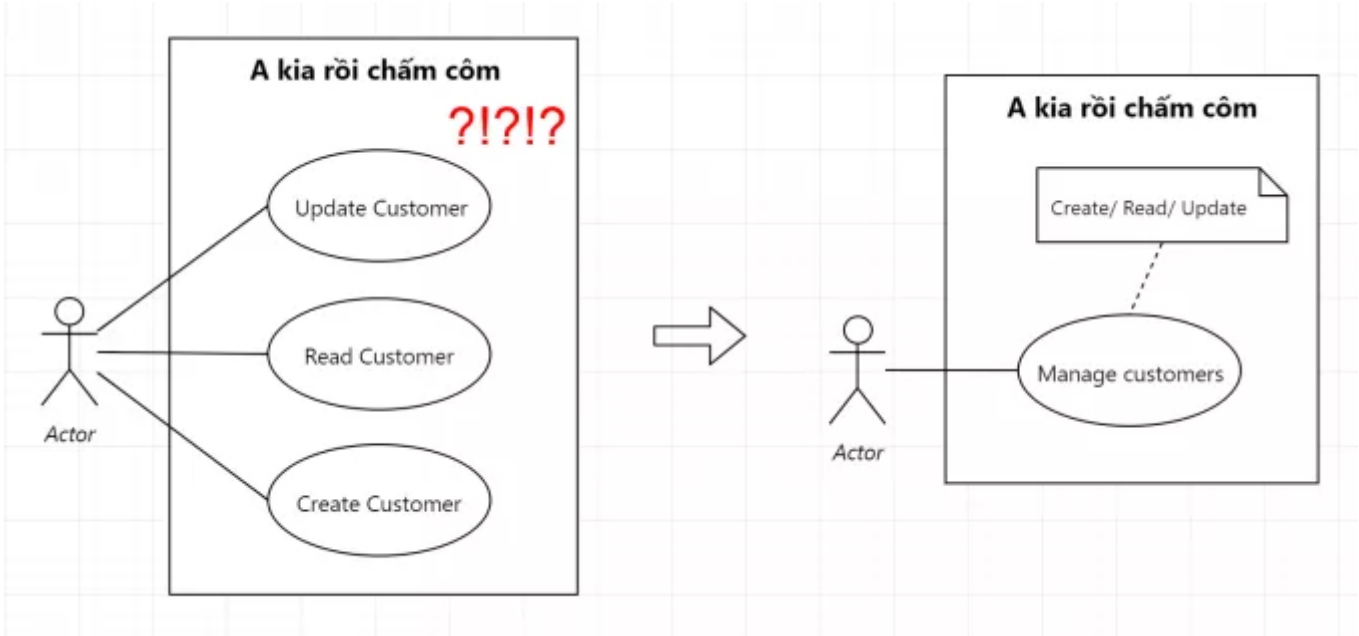
Ví dụ hệ thống CRM lấy dữ liệu khuyến mãi từ hệ thống ERP. Thì về bản chất CRM phải có khả năng Create dữ liệu khuyến mãi, thì mới lấy dữ liệu khuyến mãi từ ERP về được.

Nhưng theo góc nhìn của End-Users, thì không một người dùng nào (kể cả System Admin) có thể tạo thủ công dữ liệu khuyến mãi trên CRM, mà End-Users họ chỉ Đọc/ Sửa/ Xóa dữ liệu được lấy về này thôi.

Do đó ở đây anh em cần mô tả rõ là có phải tất cả dữ liệu đều cho phép End-Users CRUD được hay không (không tính phân quyền).

Cách 2

Tạo hẳn một Use Case với tên là: ***Manage "X"****, với X là một đối tượng bất kỳ.*



Thay vì vẽ như bên trái, thì hãy vẽ như bên phải.

Nê´u không đâ`y đủ 4 tính năng CRUD, thì anh em có thể làm 1 cái note nhỏ bên trên, nói rõ Manage là có những tính năng gì, không có những tính năng gì.

3.5. Thẩm mỹ

Cuô´i cùng vẫn quay vê` vâ´n đê` thẩm mỹ. Nguyên nhân việc Use Case mâ´t thẩm mỹ đê´n từ 2 lý do:

- Mă´t thẩm mỹ kém: chiê´m 0,00000000000069%
- **Ẩu, cẩu thả:** chiê´m 99,000000000000000069%

Làm gì cũng vậy, đặc biệt là mô hình hóa để làm document. Ẩu là thứ mình nên cô´ gă´ng hạn chê´ nó nhâ´t. Vì làm đúng 1 lâ`n, đẹp 1 lâ`n, sau này đỡ mă´c công làm lại chứ hông có gì hê´t.

Một sô´ điểm anh em câ`n chú ý sau:

- **Kích cỡ** các Use Case trong Diagram là phải như nhau, kể cả cha-con, lẫn các mô´i quan hệ Include. Tuy nhiên, Use Case có Extend sẽ được vẽ to hơn một chút.
- Nhớ phải đánh dâ´u **Use Case ID** trong hình vẽ.
- Các **mô´i quan hệ không được chô`ng chéo** lẫn nhau. Anh em có thể vẽ 1 Actor ở 2 vị trí khác nhau để tránh các đường nô´i bă´t chéo lên nhau.
- Khi vẽ Use Case Diagram, tập trung vào **câu hỏi What** để tìm ra Use Case, tránh *câu hỏi How*, vì khi đó anh em râ´t dễ đi vào detail.
- Và nê´u được, hãy **tô màu lên Use Case** để nhìn Diagram được rõ ràng, sáng sủa và mạch lạc ☺

.

.

.

Hi vọng qua bài này anh em đã hiểu rõ bản châ´t của Use Case, và biê´t cách vẽ Use Case Diagram. À mà không những biê´t cách vẽ, mà còn vẽ đúng, vẽ đẹp và tránh được những lỗi sai thường gặp nữa.

Tham khảo thêm:

- medium.com/@warren2lynch/all-you-need-to-know-about-use-case-modeling-828756da3215
- uml-diagrams.org/use-case-diagrams

Bài sau mình sẽ note lại cách viết Use Case Specification sau cho nhanh gọn và đơn giản nhất. Nếu có gì thắc mắc cứ thả còm men bên dưới hoặc email cho mình nhé.

Bái bai và hẹn gặp lại anh em!!!

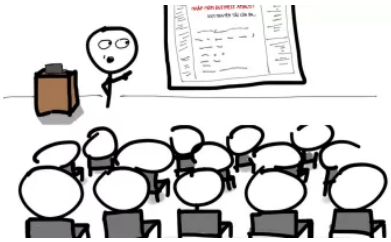
Nguyen Hoang Phu Thinh

Hế lô anh em, mình là Thịnh, và mình đang làm BA tại Bosch. Thinhnotes là nơi mình tập viết và góp nhặt những trải nghiệm trong chặng hành trình BA của mình. Hi vọng những bài note cu te hột me này sẽ giúp ích được anh em ☺

Share để lưu lại

Có liên quan

Kỹ năng cần có của người làm BA (Tập 2)
05/07/2019



Có nên đi học khóa học BA?
18/07/2018

Khi Go-Live, mọi thứ mới chỉ bắt đầu
10/06/2018

6 Comments Sort by Top

Add a comment...

Nguyễn Thị Thảo
Cảm ơn anh Thịnh. Bài viết cực dễ hiểu.
Like · Reply · 3d

Nghĩa Dương Trọng
Anh ơi cho em hỏi là khi mình làm theo cách 2 là tạo Use Case Manage customers và note lại là có 3 chức năng thêm/sửa/xóa. Vậy đến khi mô tả Basic FLow của Use Case Manage customers thì sẽ phải mô tả như nào ạ? Anh có thể ví dụ để rõ hơn không ạ?
Like · Reply · 4w

Nguyen Hoang Phu Thinh
Hi Nghĩa, a thường mô tả basic flow cũng theo trình tự:

tạo >> sửa >> xoá luôn.

Nghĩa là em cứ mô tả các bước tương tác giữa user với system ntn đó, để TẠO record thành công >> rồi tiếp theo mô tả tương tác ntn để SỬA record đó thành công >> rồi tương tự là XOÁ record đó, là kết thúc Flow Manage Customer của mình (gồm tạo/ sửa/ xoá).

E tham khảo thêm các nguồn khác nữa nhé.

Like · Reply · 1 · 4w

Nghĩa Dương Trọng

Nguyen Hoang Phu Thinh Em cảm ơn ạ!

Like · Reply · 4w

Nguyễn Đức Thương

Cảm ơn tác giả nhiều, đọc rất là cuốn và hiểu ra nhiều điều, còn dễ hiểu hơn cả giảng viên mở slide lên dạy nữa :)))

Like · Reply · 5w

Nguyen Hoang Phu Thinh

Cảm ơn Thương nhé

Like · Reply · 5w

Nguyễn Du Long

Hi Thịnh,
Bạn có thể viết một bài về transaction của usecase không. Vì mình thấy hiện nay nhiều người nhầm giữa transaction và usecase. Tuy nhiên các tài liệu định nghĩa về transaction của usecase mình tìm được hầu như rất ít. Rất mong bạn chia sẻ về vấn đề này.

Like · Reply · 3w

Nguyen Hoang Phu Thinh

Chào Long, cảm ơn Long đã đọc blog mà mình chưa hiểu ý transaction ở đây là gì? Các transaction mà UseCase đó tạo ra hay sao Long?

Like · Reply · 2w

Nguyễn Du Long

Nguyen Hoang Phu Thinh Transaction là các giao dịch ấy. Một usecase là tập hợp của nhiều transaction (theo định nghĩa). (Trường hợp sử dụng (use case) là một tập hợp các giao dịch giữa hệ thống phần mềm với các tác nhân bên ngoài hệ thống nhằm đạt được một mục tiêu sử dụng nào đó của tác nhân. Một trường hợp sử dụng mô tả một hoặc nhiều tình huống sử dụng xảy ra khi tác nhân tương tác với hệ thống phần mềm.

- Giao dịch (transaction) là một chuỗi các hành động có tính chất tương tác giữa tác nhân và hệ thống phần mềm. Khởi đầu của chuỗi hành động này là một hành động từ tác nhân tới hệ thống. Kết thúc của chuỗi hành động này là một hành động ngược trở lại của hệ thống lên tác nhân.)

Like · Reply · 6d

Dương

Chuyên mục: Chuyện nghề BA

Thẻ: BA Toolbox, Fundamental, Lesson Learned

Viết bình luận

Thinhnotes.com

Trở lại đầu trang