Chương 6

Nắm bắt yêu cầu phần mềm

6.1 Nhiệm vụ của nắm bắt yêu cầu phần mềm

- Nhiệm vụ của nắm bắt yêu cầu phần mềm là xây dựng và duy trì mô hình use-case để đặc tả góc nhìn từ ngoài vào hệ thống, cho ta thấy tất cả các chức năng mà phần mềm phải đáp ứng cho thế giới bên ngoài, ai là người thực hiện từng chức năng tương ứng.
- Các điểm bắt đầu cho hoạt động nắm bắt yêu cầu có thể là :
  - Từ mô hình nghiệp vụ (business model) cho các ứng dụng nghiệp vụ.
  - Từ mô hình lĩnh vực (domian model) cho các ứng dụng nhúng.
  - Từ đặc tả yêu cầu phần mềm cần xây dựng nhưng được tạo rồi bởi nhóm khác và/hoặc dùng phương pháp đặc tả khác, theo định dạng khác.
  - Từ 1 điểm nào đó nằm giữa các điểm xuất phát trên.
  - Từ không có gì.

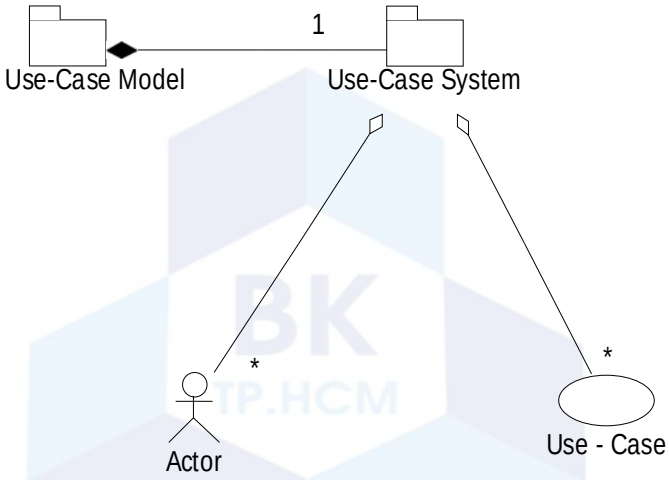
6.2 Các artifacts cần tạo ra

- Mô hình use-case = hệ thống các use-case, nó chứa :
  - các package, nếu có, mỗi package chứa :
    - các lược đồ use-case, mỗi lược đồ chứa :
      - 1 số Actor : người/hệ thống ngoại/thiết bị ngoài tương tác với hệ thống phần mềm.
      - 1 số Use-case : các chức năng có nghĩa mà hệ thống cung cấp cho actor. Thông tin chi tiết kèm theo từng use-case có thể là bảng đặc tả chi tiết, flow of events, các yêu cầu phi chức năng kèm theo.

6.2 Các artifacts cần tạo ra

    - các yêu cầu đặc biệt của từng use-case, hay của toàn bộ các use-case
  - Đặc tả kiến trúc hệ thống phần mềm theo góc nhìn use-case (view of use-case model)
  - Bảng thuật ngữ chung.
  - Các prototype giao diện với user (user-interface prototype)

6.2 Các artifacts cần tạo ra

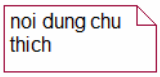


6.2 Các artifacts cần tạo ra

- Ký hiệu miêu tả các phần tử trong lược đồ use-case :
  - Actor :

  - Use-case :

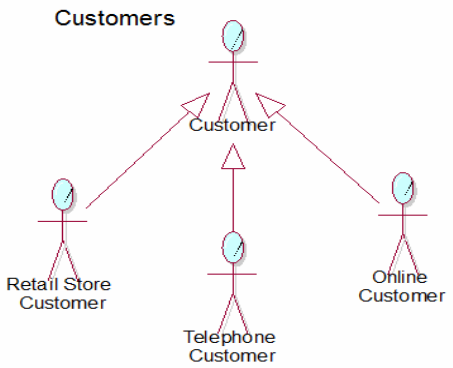
  - Note :



  - Mối quan hệ giữa 2 phần tử :

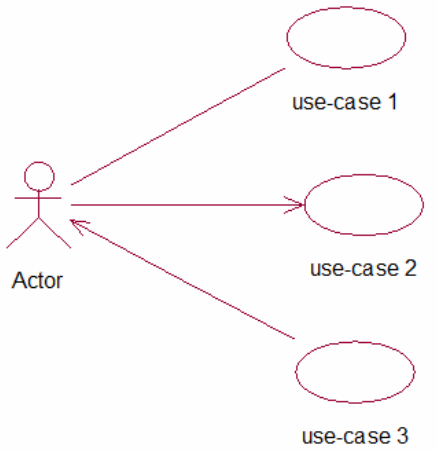
6.2 Các artifacts cần tạo ra

- Mỗi lược đồ use-case sẽ chứa 1 số actor, 1 số use-case, 1 số note và mối quan hệ giữa chúng :
  - Mối quan hệ actor – actor thường chỉ là tổng quát hóa, actor tổng quát sẽ chứa những tính chất chung mà các actor con ít nhất sẽ có.



6.2 Các artifacts cần tạo ra

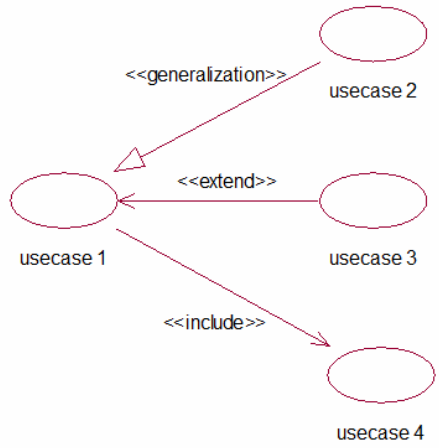
- Mối quan hệ actor – use-case thường chỉ là quan hệ kết hợp (association).



- Actor và use-case 1 có quan hệ nhau, nhưng chưa cụ thể hóa mối quan hệ.
- Actor chủ động quan hệ với use-case 2 và không chờ đợi kết quả hay phản ứng.
- use-case 3 chủ động quan hệ với actor và không chờ đợi kết quả hay phản ứng.

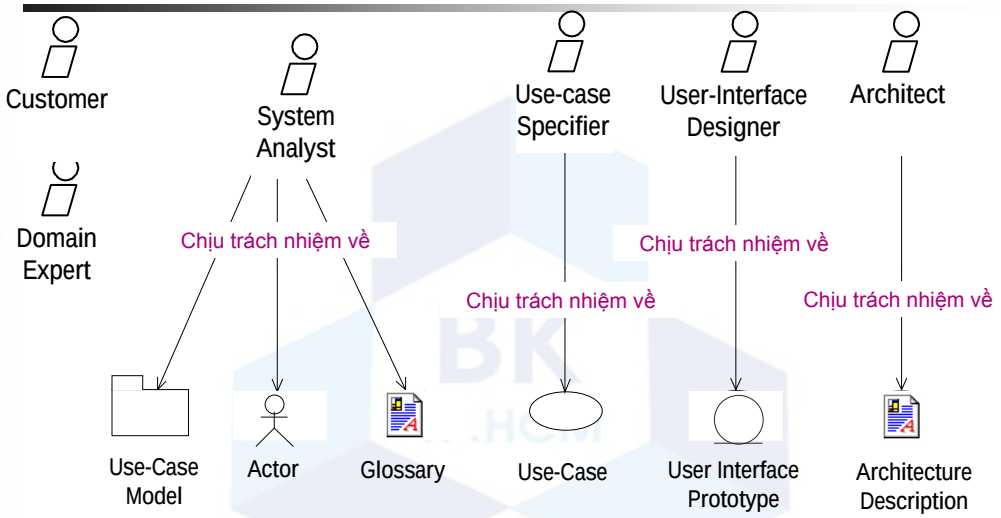
6.2 Các artifacts cần tạo ra

- Mối quan hệ use-case – use-case thường là 1 trong 3 mối quan hệ : tổng quát hóa, include, extend :

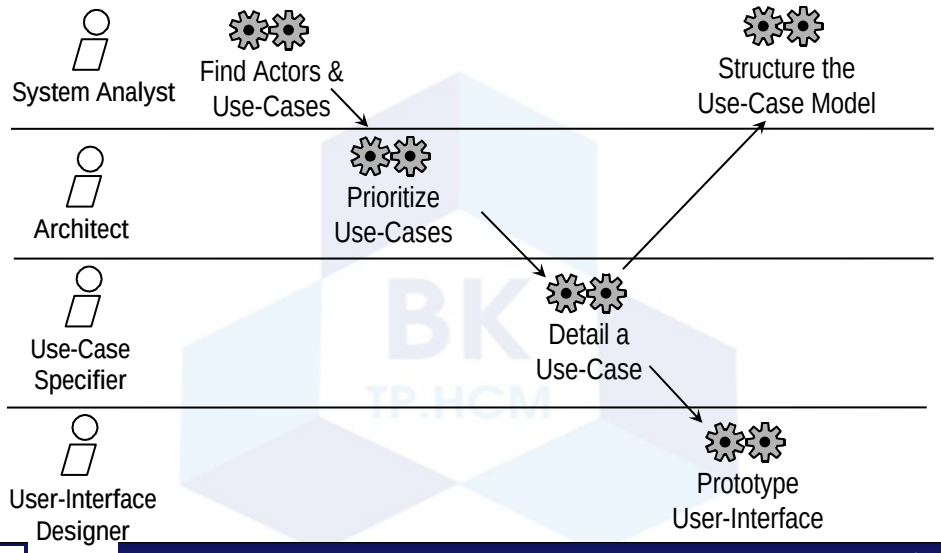


- usecase 1 chứa những điểm chung mà usecase 2 và các usecase con khác sẽ có.
- Trong thời gian thực hiện usecase 1 có thể ta sẽ phải thực hiện usecase 3.
- Trong thời gian thực hiện usecase 1 ta luôn phải thực hiện usecase 4.

6.3 Các worker tham gia nắm bắt yêu cầu



6.4 Qui trình nắm bắt yêu cầu phần mềm



6.5 Tìm các actor và use-case

Tìm các actor của hệ thống phần mềm

- Việc tìm các actor phụ thuộc vào điểm xuất phát :
  - nếu xuất phát từ mô hình nghiệp vụ hay lĩnh vực của hệ thống thì việc tìm actor rất đơn giản : dựa vào mô hình có sẵn, rút trích các actor của mô hình đó.
  - Còn nếu xuất phát từ các ý niệm mơ hồ, thậm chí là null, thì hãy cố gắng trả lời các câu hỏi sau, nội dung trả lời sẽ chứa 1 hay nhiều actor :

6.5 Tìm các actor và use-case

Tìm các actor của hệ thống phần mềm

- Ai là người sử dụng chức năng chính yếu của hệ thống ?
- Ai phải thực hiện công việc bảo dưỡng, quản trị và giữ cho hệ thống hoạt động tốt theo thời gian ?
- Hệ thống sẽ sử dụng, điều khiển thiết bị phần cứng nào ?
- Hệ thống cần tương tác với những hệ thống khác không ? Chúng là ai ?
- Ai hoặc phần tử nào quan tâm đến kết quả được tạo ra bởi hệ thống phần mềm ?
- Ai hoặc phần tử nào chịu ảnh hưởng bởi kết quả được tạo ra bởi hệ thống phần mềm ?

6.5 Tìm các actor và use-case

Tìm các use-case của hệ thống phần mềm

- Việc tìm các use-case cũng phụ thuộc vào điểm xuất phát :
  - nếu xuất phát từ mô hình nghiệp vụ hay lĩnh vực của hệ thống thì việc tìm use-case rất đơn giản : dựa vào mô hình có sẵn, rút trích các use-case của mô hình đó.
  - Còn nếu xuất phát từ các ý niệm mơ hồ, thậm chí là null, thì hãy duyệt tuần tự từng actor tìm được, ứng với mỗi actor hãy cố gắng trả lời các câu hỏi sau, nội dung trả lời sẽ chứa 1 hay nhiều use-case :

6.5 Tìm các actor và use-case

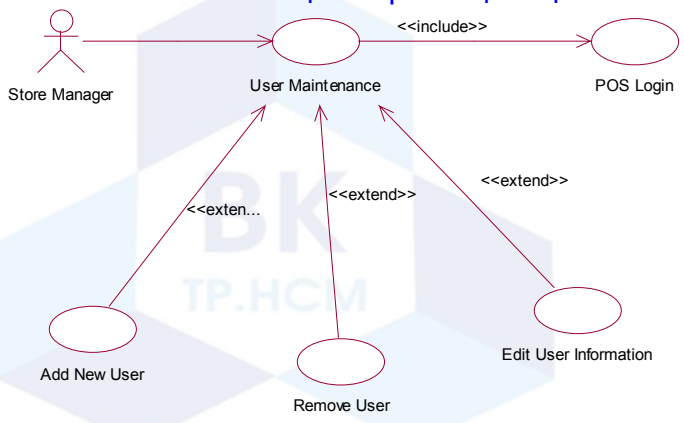
Tìm các use-case của hệ thống phần mềm

- Actor này yêu cầu chức năng gì của hệ thống ?
- Actor này có phải đọc/tạo/xóa/hiệu chỉnh/lưu thông tin nào của hệ thống, chúng là gì ?
- Hệ thống có cần cảnh báo những sự kiện nào cho actor này, chúng là gì ?
- Actor này có cần phải báo hiệu cho hệ thống về vấn đề nào không ? Chúng là gì ?
- Hệ thống có phải hỗ trợ 1 số công việc thương nhật cho actor này, chúng là những công việc gì ?
- …

6.5 Tìm các actor và use-case

Nhận dạng các mối quan hệ giữa các phần tử tìm được

- Cố gắng nhận dạng các mối quan hệ giữa các actor-actor, actor-usecase, usecase-usecase ìm được rồi phát họa lược đồ use-case kết quả.



6.6 Lập thứ tự ưu tiên các use-case

- Hệ thống phần mềm lớn thường cung cấp nhiều use-case, việc hiện thực tất cả use-case 1 lần duy nhất sẽ tốn nhiều thời gian, kinh phí,… dẫn đến độ rủi ro cao. Do đó thường ta sẽ phát triển phần mềm theo cơ chế lập thông qua nhiều version với chức năng tăng dần : verion 1 chỉ chứa 1 số ít các chức năng chính yếu, version 2 tăng cường theo 1 số tính năng cần thiết, … đến khi xây dựng được version n chứa đầy đủ các chức năng mong muốn.
- Để phục vụ việc phát triển phần mềm theo cơ chế tăng tiến như trên, ta cần đánh giá từng use-case, dựa vào 1 số tiêu chí để sắp độ ưu tiên cho các usecase, sắp xếp các use-case có thứ tự theo độ ưu tiên giảm dần. Việc phát triển phần mềm sẽ tuần tự hiện thực từng use-case, từng nhóm use-case theo thứ tự từ trôn xuống.

6.7 Chi tiết hóa từng use-case

- Nhiệm vụ của chi tiết hóa từng use-case là xây dựng các artifacts sau để miêu tả use-case ở mức độ cụ thể, rõ ràng và chi tiết như có thể :
  - Bảng đặc tả use-case chứa những thông tin như : tên, nhiệm vụ, tầm vực, mức ưu tiên, actor chính, điều kiện đi trước,…
  - Thông tin quan trọng nhất trong bảng đặc tả use-case là “flow of events” của use-case đó, nó gồm :
    - 1 kịch bản chính miêu tả thứ tự các hoạt động cần tiến hành để thực hiện use-case. Đây là kịch bản được đánh giá là thường xảy ra nhất.
    - 1 số kịch bản phụ.
  - Các lược đồ động để thể hiện hành vi động trong khi thực hiện use-case tương ứng.

6.7 Chi tiết hóa từng use-case

- Các kịch bản thi hành use-case phụ có thể xảy ra vì các lý do sau :
  - Actor có thể chọn thực hiện 1 nhánh trong nhiều nhánh.
  - Nếu hơn 1 actor dùng use-case, mỗi use-case có thể kích hoạt việc thực hiện 1 kịch bản riêng, và mỗi kịch bản có thể ảnh hưởng lẫn nhau.
  - Hệ thống có thể phát hiện lỗi nhập thông tin từ user hay lỗi trong lúc thực hiện các hoạt động.
  - 1 số tài nguyên không hoạt động tốt làm use-case không hoàn tất công việc đúng.

6.7 Chi tiết hóa từng use-case

Xây dựng các lược đồ thể hiện hành vi động cho use-case

- Khi sự tương tác giữa actor và các use-case liên quan có độ phức tạp cao, ta nên dùng dạng đặc tả trực quan để diễn tả hành vi động, nó sẽ giúp người phân tích hiiễu rõ hơn về use-case :
  - Lược đồ trạng thái để miêu tả sự chuyển trạng thái của thành phần phần mềm trong quá trình thực hiện use-case.
  - Lược đồ cộng tác để miêu tả sự cộng tác giữa các đối tượng use-case và đối tượng actor trong việc thực hiện use-case.
  - Lược đồ hoạt động để miêu tả thuật giải vĩ mô để thực hiện use-case.
- Không nên lạm dụng các lược đồ động vì đây là ngôn ngữ của người phát triển, user và khách hàng khó lòng hiểu nổi.

6.8 Cấu trúc lại mô hình use-case

- Mô hình use-case ban đầu thường còn rất lộn xộn, chưa đầy đủ, khó đọc và duy trì. Do đó, ta phải mô hình lại nó thông qua các hoạt động sau :
  - Nhận dạng actor tổng quát chứa các tính chất chung của nhiều actor khác.
  - Nhận dạng usecase tổng quát chứa các tính chất chung của nhiều usecase khác.
  - Nhận dạng các usecase nới rộng của từng usecase chính.
  - Nhận dạng các usecase phụ thêm (include) của từng usecase chính.
  - Di dời, phân phối lại các actor và use-case vào các lược đồ use-case sao cho chúng thỏa mãn tính kết dính cao nhất và độ phụ thuộc ít nhất.

6.9 Thiết kế prototype giao diện cho từng use-case

- Mỗi use-case là 1 chức năng mà phần mềm phải đáp ứng với bên ngoài. Thường để thực hiện 1 use-case, ta cần 1 hay nhiều đối tượng giao diện, mỗi đối tượng giao diện sẽ giúp actor tương tác được với phần mềm hầu cung cấp thông tin, chọn lựa option, kích hoạt chức năng chạy, xem và kiểm tra kết quả do phần mềm tạo ra.
- Dựa vào đặc tả chi tiết của use-case, ta sẽ phát họa sơ lược (prototype) phần tử giao diện cho use-case.
- Lưu ý, prototype hoàn toàn mới ở mức ý niệm, chưa kết hợp với đối tượng giao diện của công nghệ hiện thực cụ thể nào.

6.10 Kết chương

- Chương này đã giới thiệu các thông tin cơ bản về workflow phân tích yêu cầu chức năng như nhiệm vụ, các artifact cần tạo ra, các worker tham gia, qui trình thực hiện. Chương này còn giới thiệu chi tiết về hoạt động phân tích kiến trúc phần mềm và hoạt động phân tích từng use-case chức năng.