**3.5. Các yêu cầu**

Xác định các use case của hệ thống là một hoạt động thuộc về Yêu cầu kỷ luật của Quy trình Thống nhất. Yêu cầu và phân tích yêu cầu là một phần quan trọng của giai đoạn khởi động. Nhóm có thể và phải sử dụng mọi nguồn có sẵn để xác định các yêu cầu (chuyên gia, người sử dụng, tài liệu, giao diện, tài liệu, vv), và, đối với từng nguồn, một bộ các chức năng mà hệ thống phải thực hiện có thể được xác định.

**3.5.1. Các yêu cầu bắt buộc**

Các yêu cầu bắt buộc tương ứng với việc tìm kiếm thông tin về các chức năng mà hệ thống phải thực hiện và đối với các ràng buộc mà hệ thống phải hoạt động theo. Trong cách tiếp cận các yêu cầu được trình bày trong cuốn sách này, chúng sẽ được ghi lại trong use case của hệ thống.

Một cách khác để đăng ký yêu cầu mà không cần use case là sử dụng một tài liệu bao gồm một danh sách các yêu cầu chức năng, có thể kèm theo một danh sách các rằng buộc (Sommerville, 2006). Lợi thế của việc sử dụng các use case thay vì một danh sách các chức năng là một use case tốt có các chi tiết được xác định rõ, cho phép tạo ra một danh sách các yêu cầu cấp cao có thể hiểu được nhiều hơn một danh sách các chức năng riêng lẻ. Thông thường, số use case mức cao thấp hơn nhiều so với số lượng các chức năng riêng lẻ. Các chức năng riêng lẻ sẽ xuất hiện trong việc mở rộng use case khi các use case mức cao có cấu trúc chi tiết.

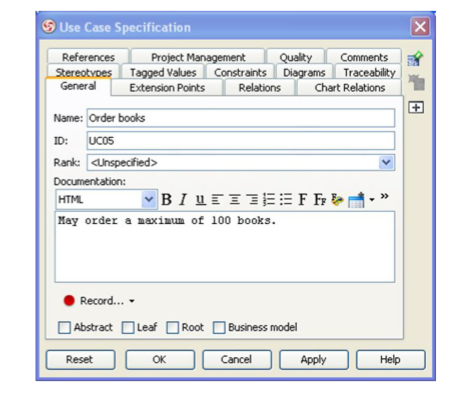
Trong trường hợp của Livir, yêu cầu tạo ra sẽ cho phép nhóm phát hiện ra rằng hệ thống phải kiểm soát việc mua và bán sách, nhận thanh toán, cho phép đăng ký sách bị hư hỏng, tạo báo cáo bán hàng, kiểm tra xem sách có trong kho hay không. Các hoạt động này và nhiều hoạt động khác sẽ tạo thành các chức năng của hệ thống, và đó là lý do tại sao các yêu cầu này được gọi là các yêu cầu chức năng. Các chức năng này sẽ được đưa vào một hoặc nhiều use cases.

Mặt khác, trong quá trình yêu cầu các nhà phân tích có thể phải đối mặt với các quy tắc kinh doanh hoặc những hạn chế về cách các chức năng phải được thực hiện bởi hệ thống. Ví dụ: quy tắc kinh doanh có thể tuyên bố rằng hiệu sách chỉ gửi sách sau khi thanh toán được xác nhận. Loại quy tắc đó là yêu cầu không có chức năng có thể được ghi lại dưới dạng chú thích hoặc nhận xét, để được thu hồi lại và xác minh khi use case phải được chi tiết. Ngoài ra, nó có thể được ghi lại riêng rẽ dưới dạng một danh sách hoặc bảng tính có đánh số, có tham chiếu đến trường hợp sử dụng thông qua một số duy nhất



Hình 3.7: Các yêu cầu phi chức năng được chú thích trong use case

Như có thể thấy trong hình 3.7, sơ đồ use case với các ghi chú sẽ nhanh chóng trở nên quá phức tạp để hữu ích. Đây là lý do tại sao khuyên bạn nên ghi chú như vậy trong use case hoặc trong một tài liệu riêng biệt. Hầu hết các công cụ CASE cho phép một phần tử như một use case có một cửa sổ đặc tả như thể hiện trong hình 3.8. Một ngoại lệ đối với quy tắc đó sẽ là trường hợp khi chú thích hoàn toàn cần thiết để giúp bạn hiểu sơ đồ. Tuy nhiên, điều này hiếm khi xảy ra



Hình 3.8: Ví dụ về cửa sổ đặc tả cho một use case.

**3.5.2. Các yêu cầu bắt buộc không phải là thiết kế**

Hệ thống sẽ được phân tích giống như một khu rừng. Để khám phá một khu rừng, người ta không thể bắt đầu bằng cách kiểm tra từng loài thực vật và động vật. Có một câu nói rằng một số người không thể nhìn thấy khu rừng đến từng cái cây. Rừng là hệ thống và cây cối là các yêu cầu. Chỉ khi kết thúc quá trình, một nhóm có thể nói rằng họ đã có được kiến ​​thức về những phần nhỏ nhất. Thứ nhất, phải xem xét toàn cảnh và chỉ sau đó có thể nghiên cứu chi tiết

Do đó, Giai đoạn Khởi động phải cung cấp cái nhìn toàn cảnh – nghĩa là phải nhìn ra những cái quan trọng hơn- và sau đó hệ thống có thể được chia thành các phần để các chi tiết có thể được phân tích và cuối cùng là một giải pháp được thiết kế. Tổ chức lặp lại trong các giai đoạn xây dựng tương ứng sẽ chia rừng thành các khu vực, để xem mỗi khu vực một lúc, và theo cách đó, có thể giải quyết được sự phức tạp vốn có. Do đó, một trong những mục tiêu cuối cùng của Giai đoạn Khởi động là tổ chức bộ phận làm việc trong các use case, sẽ được khám phá trong các lần lặp lại sắp tới.

Trong suốt quá trình Khởi động, yêu cầu bắt buộc phải nhanh chóng và chung chung. Cách đúng để thực hiện điều này là xem xét việc mở rộng các yêu cầu chứ không phải chi tiết của họ. Nhà phân tích phải hiểu được sự mở rộng của những gì hệ thống phải làm, mà không nêu chi tiết cách nó sẽ làm điều đó. Chỉ trong thời gian thực hiện Lập trình, các phân tích yêu cầu sẽ được làm chi tiết hơn

Thời gian sử dụng cho các yêu cầu bắt buộc phải là về khám phá chứ không phải là phát minh. Trong thời gian này, nhóm chuyên gia phân tích, với khách hàng, người dùng và các bên liên quan khác sẽ cố gắng liệt kê hầu hết các năng lực và hạn chế mà không cần quan tâm đến chi tiết. Thông tin chi tiết về yêu cầu sẽ được cung cấp trong các lần lặp tiếp theo.

Cũng phải rõ ràng rằng các yêu cầu là điều mà khách hàng yêu cầu, chứ không phải nhóm thiết kế. Một số nhà phân tích trộn lẫn các yêu cầu thu thập, đó là, các yêu cầu của khách hàng, với sự khởi đầu của thiết kế hệ thống. Một ví dụ của loại nhầm lẫn này là một bộ các yêu cầu liên quan đến thiết kế cơ sở dữ liệu quan hệ. Trừ khi có các hệ thống kế thừa phải tương thích với hệ thống mới, làm thế nào nó có thể được chứng minh rằng một bộ các bảng quan hệ là một nhu cầu của khách hàng? Điều này cuối cùng có thể đạt được với một số khách hàng tinh vi có trình độ về Khoa học Máy tính, nhưng đó không phải là quy luật chung. Các bảng cơ sở dữ liệu là một phần của miền giải pháp chứ không phải miền vấn đề. Nhà phân tích phải tìm kiếm các yêu cầu tương ứng với nhu cầu và mục tiêu của khách hàng về thông tin. Sau đó, cô ấy có thể quyết định xem thông tin đó có được lưu giữ trong một cơ sở dữ liệu quan hệ hay trong một cấu trúc khác.

**3.5.3. Các yêu cầu thách thức**

Các tài liệu yêu cầu, có thể được hình thành bởi sơ đồ use case với các chú thích và theo các thông số bổ sung, phải xem xét khả năng của hệ thống và các điều kiện mà nó phải hoạt động. Những thách thức liên quan đến yêu cầu, ít nhất, là như sau (Pressman, 2010):

• Làm thế nào để khám phá các yêu cầu.

• Làm thế nào để truyền đạt yêu cầu cho các giai đoạn khác và các đội của dự án.

• Làm thế nào để thu hồi các yêu cầu trong quá trình phát triển để xác minh xem chúng đã được thực hiện chưa.

• Làm thế nào để quản lý yêu cầu thay đổi.

Sẽ không có ích nếu phát triển một sơ đồ use case tốt và sau đó không thể biết được các yêu cầu đã được kết hợp trong thiết kế. Sự tồn tại của các cơ chế tự động để thực hiện việc xác minh này là rất quan trọng. Do đó, điều quan trọng là phải duy trì quan hệ truy xuất nguồn gốc giữa các use case và các phần khác của thiết kế. Các chương sau đây cho thấy làm thế nào để những mối quan hệ giữa các thiết kế được thu thập.

Cần lưu ý rằng các yêu cầu nhất thiết phải thay đổi trong quá trình phát triển dự án. Do đó, sự thay đổi phải được quản lý, không tránh khỏi và sảy ra sau quá trình khởi động.

Đôi khi, các yêu cầu thay đổi sau khi hệ thống được triển khai. Điều kiện bối cảnh, quy định, chính sách của công ty, hoặc phương pháp làm việc có thể thay đổi bất cứ lúc nào. Mặc dù nhà phân tích không thể đoán trước được những thay đổi này, nhưng các cơ chế để điều chỉnh chúng có thể được tạo ra để giảm bớt quá trình thay đổi khi cần thiết. Có những mẫu thiết kế cụ thể để giải quyết những bất ổn yêu cầu này (ví dụ như mô hình Chiến lược, được trình bày trong Phần 7.6). Những thay đổi là hoàn toàn không thể đoán trước. Nếu hệ thống không được cấu trúc để đáp ứng các thay đổi trong yêu cầu nó có thể sẽ rất khó để thực hiện chúng.

Loại tình huống này khiến các quy trình phân tích và thiết kế theo yêu cầu (Alford, 1991) không phù hợp với hầu hết các hệ thống. Sử dụng các yêu cầu làm cơ sở để hỗ trợ kiến ​​trúc hệ thống giống như xây dựng một ngôi nhà trên cát di chuyển; Khi yêu cầu thay đổi, cấu trúc ảnh hưởng. Tuy nhiên, Quá trình Hợp nhất (UP) sử dụng nền tảng ổn định hơn cho kiến ​​trúc, dựa trên các lớp và các thành phần đóng gói thông tin và hành vi.

Các lớp này thực hiện các chức năng kết hợp cho phép thực hiện các yêu cầu. Nếu yêu cầu thay đổi, các kết hợp thay đổi, nhưng không phải là cấu trúc cơ bản. Loại kiến ​​trúc này tuân theo nguyên tắc đóng mở (Meyer, 1988), theo nghĩa nó luôn luôn đóng cho sửa đổi (nó hoạt động), nhưng mở để mở rộng (nó có thể đáp ứng chức năng mới).

Điều quan trọng là phải tìm ra nguồn gốc của mỗi yêu cầu (ví dụ như một doanh nhân, một nhân viên kinh doanh, chính khách hàng, hoặc thậm chí một chuyên gia) bởi vì nó là cần thiết để xác nhận các yêu cầu với những nguồn này, xác minh xem chúng có được viết hay không, đầy đủ và hợp lý.

Đôi khi nó cũng có thể xảy ra rằng những người khác nhau hoặc các bộ phận có các đặc điểm kỹ thuật khác nhau cho cùng một yêu cầu. Trong trường hợp đó, cần tạo ra một sự thỏa thuận giữa các bên hoặc xác định ai có thẩm quyền cao nhất để xác định hình thức chấp nhận được đối với yêu cầu.

**3.5.4. Các yêu cầu chức năng rõ ràng và tiềm ẩn**

Các yêu cầu chức năng có thể được tùy ý xác định là rõ ràng hoặc ẩn (Gause và Weinberg, 1989):

• Các yêu cầu chức năng rõ ràng là các chức năng được thực hiện với sự hiểu biết của người dùng. Các yêu cầu này thường tương ứng với trao đổi thông tin giữa người dùng và hệ thống, chẳng hạn như các truy vấn và nhập dữ liệu, qua giao diện hệ thống.

• Yêu cầu chức năng ẩn là các chức năng được thực hiện bởi hệ thống mà không có kiến ​​thức rõ ràng của người sử dụng. Thông thường các chức năng này là các hoạt động toán học và cập nhật dữ liệu được thực hiện bởi hệ thống mà không có kiến ​​thức rõ ràng về người sử dụng, nhưng nó như là kết quả của các chức năng khác do người dùng thực hiện.

Yêu cầu ẩn được thực hiện nội bộ bởi hệ thống. Do đó, mặc dù chúng không xuất hiện rõ ràng như các use case, nhưng chúng phải được liên kết với nhau một cách hợp lý để được gọi lại vào thời điểm thiết kế và thực hiện. Do đó, chúng cũng có thể được thêm vào làm chú thích cho use case.

Một ví dụ về một yêu cầu rõ ràng là việc xuất một báo cáo về sách bán chạy nhất, đó là điều mà người quản lý có thể yêu cầu. Ví dụ về yêu cầu ẩn là áp dụng chính sách chiết khấu cho bán hàng. Trong trường hợp đó, người sử dụng không yêu cầu rõ ràng hệ thống thực hiện các hoạt động. Vì nó là một hoạt động mà hệ thống thực hiện tự động, nó là một yêu cầu ẩn.

**3.5.5. Các yêu cầu phi chức năng**

Các yêu cầu phi chức năng là những ràng buộc hoặc chất lượng có thể được liên kết với các chức năng cụ thể của một hệ thống (ví dụ: "một đơn đặt hàng không thể chứa nhiều hơn một trăm cuốn sách", "giao dịch lệnh phải được giữ nguyên nếu truyền thông bị hỏng", …) , do đó chúng có thể được coi là chú thích về các trường hợp sử dụng kết hợp các chức năng tương ứng. Tuy nhiên, đôi khi các yêu cầu phi chức năng có thể là chung chung, nghĩa là không nhất thiết phải gắn với một chức năng (ví dụ: "hệ thống phải được thực hiện trong Java") và trong trường hợp đó, chúng sẽ xuất hiện trong tài liệu kỹ thuật bổ sung. Những trở ngại và phẩm chất đặc biệt gắn liền với một chức năng được gọi là các yêu cầu phi chức năng và các ràng buộc, phẩm chất chung được gọi là các yêu cầu bổ sung. Có hai loại yêu cầu phi chức năng:

• Các vấn đề logic: Các quy tắc kinh doanh gắn liền với một chức năng. Ví dụ: trong quá trình đăng ký bán hàng, có thể xem xét một loạt các ràng buộc, chẳng hạn như chưa đóng giao dịch cho đến khi nhà điều hành thẻ tín dụng xác nhận thanh toán hoặc chưa đóng việc bán hàng nếu việc phân phối cuối cùng đến một địa chỉ được trả lại do địa chỉ không hợp lệ .

• Các vấn đề về công nghệ: Những trở ngại và phẩm chất liên quan đến công nghệ được sử dụng để thực hiện chức năng, ví dụ như giao diện người dùng, loại giao thức truyền thông, các ràng buộc về bảo mật, khả năng chịu lỗi, v.v ...

Ví dụ, thiết lập giao diện người dùng để thực hiện một lệnh phải tuân theo mẫu thiết kế dựa trên luồng tuần tự của màn hình là một hạn chế về công nghệ (ràng buộc giao diện) về cách mà hàm phải được thực hiện. Một ví dụ khác về yêu cầu không có chức năng là "xác nhận thanh toán không được quá 5 giây". Đây là một hạn chế về công nghệ liên quan đến hiệu năng của hệ thống và nó sẽ ảnh hưởng đến cách nhà thiết kế suy nghĩ về cơ chế truyền thông với nhà khai thác thẻ tín dụng. Trong trường hợp này, thiết kế hệ thống sẽ phải xem xét nghiêm túc các kết nối băng thông rộng cho các nhà điều hành

Một yêu cầu không xác định làm thế nào để ràng buộc được thực hiện, nó chỉ yêu cầu. Việc thiết kế và thực hiện hệ thống phải đáp ứng yêu cầu này bằng cách này hay cách khác, hoặc nếu không nhà phân tích nên thương lượng với khách hàng về một số yêu cầu linh hoạt.

**3.5.6. Tính vĩnh viễn và tính tạm thời của các yêu cầu phi chức năng**

Một trong những tính năng cơ bản nhất của một yêu cầu là liệu một yêu cầu phi chức năng hoặc bổ sung được cung cấp là vĩnh viễn hoặc tạm thời.

Các yêu cầu phi chức năng và bổ sung có thể được coi là vĩnh viễn (không thay đổi) hoặc tạm thời (chúng được mong đợi thay đổi) tùy thuộc vào quyết định của khách hàng. Sự bền bỉ hoặc chuyển tiếp không phải là tính năng nội tại của một yêu cầu: nó được quyết định tùy thuộc vào sự thuận tiện. Các yêu cầu tương tự có thể được coi là vĩnh viễn hoặc tạm thời tùy thuộc vào những gì mong muốn về thời gian và chi phí phát triển phần mềm và bảo trì.

Nếu đầu tư vào thiết kế linh hoạt, hầu hết các khó khăn là tạm thời, các nỗ lực sẽ được bỏ ra ít hơn trong thời gian bảo trì để thích ứng với thay đổi. Tuy nhiên, chi phí thiết kế linh hoạt trong quá trình phát triển có thể trở nên nghiêm trọng đối với một số dự án. Nó sẽ luôn luôn là một ý tưởng tốt để suy nghĩ về những khó khăn thực sự cần được coi là yêu cầu tạm thời.

Ví dụ, một yêu cầu bổ sung có thể thiết lập rằng hệ thống Livir phải đối phó với một đơn vị tiền tệ: đồng đô la. Nếu yêu cầu đó được coi là vĩnh viễn, thì hệ thống sẽ được thiết kế cho một đơn vị tiền tệ ("đô la" thậm chí có thể là một kiểu dữ liệu được sử dụng để xác định các biến và các thuộc tính). Tuy nhiên, nếu yêu cầu được coi là thoáng qua, thì ngay cả khi không có đồng tiền nào được sử dụng ngày nay, toàn bộ hệ thống phải chuẩn bị sẵn sàng để hỗ trợ tiền tệ trong tương lai, hoặc thậm chí nhiều hơn một loại tiền tệ tại một thời điểm.

Hệ quả của việc quyết định rằng một yêu cầu là vĩnh viễn là như sau:

• Rẻ hơn và nhanh hơn để phát triển hệ thống.

• Nó đắt hơn và khó thay đổi hệ thống nếu, bất cứ khi nào, các yêu cầu thay đổi trong tương lai.

Mặt khác, việc quyết định rằng yêu cầu là tạm thời có những hệ quả sau:

• Nó đắt hơn và phức tạp hơn để phát triển hệ thống (ví dụ như các chức năng để thay đổi tiền tệ).

• Dễ dàng hơn và nhanh hơn để duy trì hệ thống (nếu tiền tệ thay đổi, hệ thống đã sẵn sàng để đáp ứng với một cấu hình lại đơn giản).

Do đó, nó không phải là bản chất của các yêu cầu phi chức năng mà sẽ quyết định xem nó là vĩnh viễn hoặc tạm thời. Đó là khách hàng, với sự giúp đỡ của đội, mà phải đưa ra quyết định. Tình huống lý tưởng là liệt kê các yêu cầu có tầm quan trọng lớn hơn (những người thực sự mong muốn thay đổi trong tương lai gần với một tác động lớn hơn lên hệ thống) và coi chúng là tạm thời, trong khi để những yêu cầu khác là vĩnh viễn

**3.5.7. Các yêu cầu bắt buộc và mong muốn**

Yêu cầu cũng có thể được coi là bắt buộc hoặc mong muốn, nghĩa là những gì cần phải có được bằng bất kỳ phương tiện và những gì có thể đạt được nếu không có vấn đề lớn hơn gây cản trở quá trình phát triển.

Trong trường hợp các yêu cầu về chức năng, phân loại đó cho biết các ưu tiên phát triển. Nếu có sự linh hoạt trong hợp đồng để chỉ thực hiện các trường hợp sử dụng quan trọng nhất nếu không có thời gian để thực hiện tất cả, thì nhóm phải biết những trường hợp nào bắt buộc.

Tuy nhiên, nếu nhóm nghiên cứu có thể ước tính được những nỗ lực cần thiết để phát triển hệ thống, và nếu họ có lịch sử tốt về ước tính chính xác, sẽ có ít động lực hơn cho sự phân biệt như vậy đối với các yêu cầu về chức năng, bởi vì tất cả các yêu cầu Được thực hiện trong thời gian.

Mặt khác, các yêu cầu phi chức năng và bổ sung không thể đoán trước được nhiều hơn các yêu cầu chức năng liên quan đến việc ước tính nỗ lực. Do đó, trong một số trường hợp, cần cân nhắc đến những yêu cầu đó một cách linh hoạt.

Trong trường hợp này, một số hạn chế được định nghĩa như vậy phải được thu bằng bất kỳ phương tiện nào và những yêu cầu khác có thể chỉ đơn giản là mong muốn và một số thời gian được phân bổ theo đuổi chúng.

Ví dụ, trong trường hợp của hệ thống Livir, yêu cầu sử dụng một giao diện web có thể được coi là một yêu cầu bổ sung bắt buộc. Trong trường hợp đó, các giải pháp khác không được chấp nhận. Tuy nhiên, truy cập bổ sung thông qua một điện thoại di động có thể được coi là yêu cầu mong muốn, bởi vì truy cập như vậy là không hoàn toàn cần thiết cho sự thành công có hiệu quả của hệ thống.

Ngày nay, với việc chính thức hóa các hợp đồng phát triển phần mềm, có ít tính linh hoạt hơn về các yêu cầu mong muốn. Trong hầu hết các trường hợp, nhà phát triển phải nêu rõ những yêu cầu nào sẽ được thực hiện, thời gian cần thiết và chi phí sẽ là bao nhiêu. Nếu làm sai với các thỏa thuận đó, có thể yêu cầu thanh toán tiền phạt hoặc thậm chí hủy bỏ dự án.

**3.5.8. Các yêu cầu bổ sung**

Yêu cầu bổ sung là tất cả các loại ràng buộc và chất lượng liên quan đến toàn bộ hệ thống và không chỉ các chức năng riêng biệt. Ví dụ, một yêu cầu bổ sung có thể thiết lập rằng hệ thống phải tương thích với cơ sở dữ liệu kế thừa nhất định, hoặc được thực hiện với một ngôn ngữ lập trình nhất định, hoặc thậm chí làm theo một cái nhìn nhất định.

Phải cẩn thận khi cần xác định các yêu cầu bổ sung. Một yêu cầu như "hệ thống phải dễ sử dụng" không đủ rõ ràng. Nó sẽ là tốt hơn để nói điều gì đó như "người dùng mới làm quen phải có khả năng hoàn thành nhiệm vụ mà không có lỗi về nỗ lực đầu tiên của họ." Điều đó đưa ra một ý tưởng chính xác hơn những gì cần phải được thiết kế để thực hiện yêu cầu.

Các yêu cầu bổ sung và phi chức năng cũng có thể được xác định với các nhóm khác nhau như giao diện, triển khai, thực hiện, hiệu suất, khả năng chịu lỗi, ... Mục đích của việc tạo ra sự phân biệt như vậy là để cho phép tổ chức tốt hơn.

Mặc dù hầu hết sinh viên UP sẽ chọn hệ thống phân loại FURPS1 (Grady, 1992) để tổ chức các yêu cầu bổ sung, nguồn mới nhất để quyết định phân loại yêu cầu bổ sung là ISO / IEC 25010,4 như trong Bảng 3.1.

Bảng 3.1: Phân loại yêu cầu bổ sung Dựa trên tiêu chuẩn ISO / IEC 25010: 2011 và các câu hỏi yêu cầu về thế hệ

|  |  |  |
| --- | --- | --- |
| 25010  Tính năng | 25010  Tính năng phụ | Yêu cầu tạo câu hỏi |
| Sự phù hợp về chức năng | Tính đầy đủ chức năng  Tính phù hợp chức năng  Tính chính xác về chức năng (chính xác) | Đây có phải là tất cả các chức năng cần thiết phải được thực hiện? Một số trong số chúng có thể để lại bên ngoài phạm vi hệ thống được không? Một số chức năng có thể được thực hiện bởi một dự án tiếp theo được không?  Mức độ nào mà các hoạt động của người dùng sẽ được hệ thống thực hiện dễ dàng hơn?  Liệu có các chi tiết kỹ thuật chính xác, nghĩa là độ chính xác mong muốn cho dữ liệu? Có hay không những giới hạn cho phép không chính xác? |
| Độ tin cậy | Trưởng thành  Khả dụng  Khả năng chịu lỗi  Khả năng phục hồi | Cần phải đặc biệt chú ý để ngăn không cho hệ thống đưa ra các khiếm khuyết trong quá trình sử dụng? Các phần quan trọng được xác định bởi các thông số kỹ thuật chính thức? Mức độ và loại kiểm tra nào được thực hiện để đảm bảo hệ thống không có lỗi?  Mức độ sẵn có của hệ thống là gì? Có bao nhiêu truy cập đồng thời phải được hỗ trợ? Bao nhiêu giờ mỗi ngày? Bao nhiêu ngày một năm?  Làm thế nào để hệ thống phản ứng trong trường hợp dị thường bên ngoài gây ra, chẳng hạn như gián đoạn truyền thông?  Hệ thống phải tự động phục hồi trong trường hợp thiên tai? Trong trường hợp nào phải mất dữ liệu và quá trình bị hủy bỏ sẽ được phục hồi? |
| Khả năng sử dụng | Khả năng nhận dạng phù hợp  Tính dễ học  Khả năng hoạt động  Bảo vệ lỗi của người dùng  Giao diện người dùng thẩm mỹ  Khả năng tiếp cận | Bằng cách nào phần mềm sẽ tự xuất hiện cho người dùng tiềm năng để có thể nhận ra tính khả dụng của nó? Phần mềm nên được đóng gói như thế nào?  Các khái niệm vốn có đối với phần mềm được trình bày cho người dùng như thế nào để người dùng có thể trở nên có thẩm quyền trong việc sử dụng nó?  Sản phẩm phải dễ sử dụng và kiểm soát ở mức độ nào? Sự trợ giúp nào mà hệ thống phải cung cấp? Những mẫu tài liệu và hướng dẫn cần có sẵn? Chúng sẽ được sản xuất như thế nào? Những thông tin nào họ nên trình bày?  Các loại bảo vệ chống lại lỗi người dùng nào là cần thiết?  Những mẫu thiết kế nào sẽ được sử dụng cho giao diện để cung cấp niềm vui thị giác và sự tương tác thỏa đáng?  Sản phẩm phải được thiết kế để đáp ứng những người có nhu cầu đặc biệt đến mức độ nào? |
| Hiệu suất | Hành vi thời gian  Tận dụng nguồn tài nguyên  Sức chứa | Những giới hạn thời gian tồn tại liên quan đến các quy trình phần mềm và các chức năng?  Có hạn chế không gian lưu trữ dữ liệu không? Năng lượng hạn chế? Giới hạn mạng lưới truyền thông?  Về khả năng cho phép truy cập đồng thời, các giá trị mong đợi là gì và những giá trị quan trọng là gì? Ví dụ, phần mềm có thể được thiết kế để hỗ trợ tối đa 2.000 lượt truy cập đồng thời, nhưng đảm bảo để tiếp tục làm việc khi có tới 10.000 lần truy cập đồng thời. |
| Bảo vệ | Bảo mật  Chính trực  Không bác bỏ  Trách nhiệm giải trình  Tính xác thực | Các thông tin và chức năng của hệ thống chỉ có sẵn cho những người có thẩm quyền để truy cập?  Dữ liệu và chức năng được bảo vệ ở mức nào để chống lại những thay đổi trái phép của con người và hệ thống?  Hệ thống đảm bảo rằng các bản ghi được lưu giữ bởi nó có hiệu quả như thế nào để các tác giả của chúng không thể phủ nhận chúng?  Hành động của người dùng được ghi lại bởi hệ thống nằm ở mức nào? Những thông tin gì được lưu giữ?  Đảm bảo rằng người đã đã đăng nhập thực sự là chủ tài khoản? |
| Khả năng tương thích | Sự cùng tồn tại  Khả năng tương tác | Với sản phẩm nào phải có phần mềm cùng tồn tại? Những công cụ và ngôn ngữ lập trình nào phải được sử dụng? Có cần thiết để chạy hệ thống với các hệ thống kế thừa?  Các sản phẩm khác phải liên lạc với hệ thống? Hệ thống nào phải gửi dữ liệu? Từ đó hệ thống phải nhận dữ liệu? |
| Khả năng bảo trì | Mô đun  Khả năng tái sử dụng  Khả năng phân tích  Khả năng biến đổi  Khả năng kiểm tra | Loại kiến trúc nào sẽ được sử dụng? Lớp? Phân vùng? Các thành phần? Dịch vụ Web?  Hệ thống sẽ được sản xuất từ các bộ phận của các hệ thống khác không? Phải hệ thống được thiết kế để các bộ phận của nó có thể được tái sử dụng trong các dự án trong tương lai?  Kỹ thuật hoặc công cụ đặc biệt sẽ được sử dụng để dễ dàng gỡ lỗi mã?  Kỹ thuật hoặc công cụ đặc biệt sẽ được sử dụng để đảm bảo rằng những thay đổi được giới thiệu trong hệ thống sẽ không tạo ra các khuyết tật mới?  Kỹ thuật và công cụ đặc biệt sẽ được sử dụng để làm giảm các bài kiểm tra hồi quy? Liệu các công cụ kiểm tra tự động có được sử dụng? Liệu TDD (phát triển theo thử nghiệm) sẽ được sử dụng? Làm thế nào để các tài sản được sản xuất trong quá trình test được giữ lại đến quá trình phát triển? |
| Tính di động | Tính thích nghi  Khả năng cài đặt  Thay thế | Phần mềm tự thích ứng với các ngữ cảnh khác ngoài những gì nó được thiết kế ban đầu ở mức độ nào? Nó phải được biên dịch lại hay chỉ cần cấu hình lại? Những gì có thể được cấu hình trong hệ thống? Ví dụ về các mục có thể cấu hình được là máy in, tiền tệ, chính sách của công ty, phông chữ giao diện và màu sắc, ngôn ngữ, vv Yêu cầu chuyển tiếp có lẽ sẽ liên quan đến các mục có thể định cấu hình được.  Tài nguyên cài đặt sẽ được cung cấp như thế nào? Nên cài đặt tự động? Tự động di chuyển dữ liệu?  Các nguồn lực nào mà hệ thống phải cung cấp khi nó thay thế các hệ thống khác có cùng mục tiêu? Các nguồn lực nào mà hệ thống phải cung cấp khi nó được thay thế bởi các hệ thống khác có cùng mục đích? Nếu nó tạo ra dữ liệu và cấu hình trong các định dạng được hiểu rộng rãi như XML? |
| Tính hiệu quả | Tính hiệu quả | Những mục tiêu kinh doanh nào đối với môi trường sử dụng thực tế mà hệ thống đạt được? Hệ thống phải chịu trách nhiệm gì đối với những mục đích đó một cách đầy đủ và chính xác? |
| Hiệu quả | Hiệu quả | Loại trả về đầu tư (ROI) nào mà hệ thống phải trả cho khách hàng? |
| Sự hài lòng | Tiện ích  Vui lòng  Tiện nghi  Lòng tin | Phần mềm phải được thiết kế theo cách nào để giúp người dùng nhận thức được hiệu quả của việc sử dụng nó?  Hệ thống sẽ cung cấp niềm vui cho người dùng của nó đến mức nào?  Mức độ nào mà hệ thống phải bảo toàn hoặc cải thiện sự thoải mái của người sử dụng thể chất và tinh thần?  Hệ thống phải được thiết kế bằng cách nào đó sao cho các bên liên quan tin rằng nó sẽ thực hiện công việc? |
| Tránh rủi ro | Giảm nhẹ rủi ro kinh tế  Giảm nhẹ rủi ro về sức khoẻ và an toàn  Giảm nhẹ rủi ro môi trường | Hệ thống giảm thiểu rủi ro tài chính (bao gồm cả tài sản và thiệt hại về đạo đức) như thế nào?  Loại rủi ro vật lý nào đối với con người mà hệ thống phải giảm?  Loại rủi ro môi trường và bất động sản nào mà hệ thống phải giảm? |
| Bối cảnh | Bối cảnh hoàn thiện  Mềm dẻo | Hệ thống phải được sử dụng hiệu quả, không có rủi ro và sự hài lòng của người sử dụng trong bối cảnh sử dụng của nó đến mức độ nào? Những bối cảnh sử dụng là gì? Có yêu cầu pháp lý liên quan đến việc sử dụng phần mềm?  Hệ thống phải được sử dụng hiệu quả, không có rủi ro và mức độ hài lòng của người dùng trong các ngữ cảnh sử dụng khác với những gì nó được thiết kế ban đầu cho mức độ nào? Những bối cảnh sử dụng là gì? |

Mặc dù danh sách này rất phong phú, nhóm nghiên cứu phải ghi nhớ rằng đây chỉ là một phân loại để nâng cao khả năng xác định các yêu cầu nào là quan trọng. Không cần phải tìm kiếm các yêu cầu không tồn tại, ví dụ như thiết lập các yêu cầu đóng gói phức tạp cho một khách hàng không quan tâm đến cách thức phần mềm sẽ được đóng gói.

Cũng không nên để mất thời gian thảo luận nếu một yêu cầu nhất định thuộc về loại này hay kiểu đó. Quan trọng hơn việc quyết định loại hình của nó là phải biết rằng nó tồn tại: các cuộc thảo luận dài về phân loại yêu cầu không làm tăng thêm kiến ​​thức cho dự án.

Bảng 3.2 trình bày một ví dụ về các yêu cầu bổ sung có thể được gán cho ví dụ Livir.

Không phải mọi lĩnh vực đều được làm đầy đủ bởi vì, như đã nói ở trên, các yêu cầu không được phát minh, họ phải được khách hàng yêu cầu. Do đó, thông thường không có yêu cầu trong tất cả các loại. Các câu hỏi tạo ra các yêu cầu được đề cập trong Bảng 3.1 là một cơ sở tốt cho việc tìm kiếm các nhu cầu cuối cùng. Nhưng nó không phải là bắt buộc để có một câu trả lời cho tất cả những câu hỏi.

Bảng 3.2: Các yêu cầu bổ sung cho hệ thống Livir

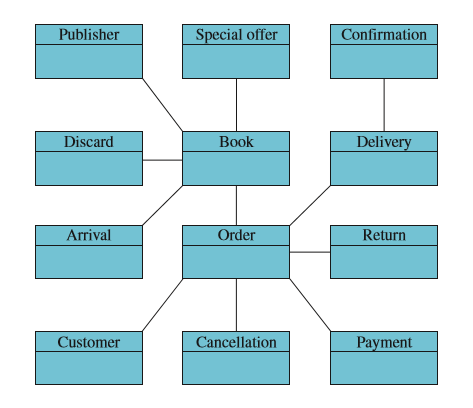
|  |  |  |
| --- | --- | --- |
| 25010  Tính năng | 25010  Tính năng phụ | Các yêu cầu bổ sung |
| Tính phù hợp về chức năng | Tính đầy đủ chức năng  Tính phù hợp chức năng  Tính chính xác về chức năng (chính xác) |  |
| Độ tin cậy | Trưởng thành  Khả dụng  Khả năng chịu lỗi  Khả năng phục hồi | Hệ thống phải có sẵn liên tục (24 giờ một ngày, 7 ngày một tuần).  Các giao dịch do người sử dụng ký kết trong trường hợp lỗi tạm thời phải được bảo quản để phục hồi ít nhất 24 giờ. Sau thời hạn đó, họ có thể bị rollback.  Trong trường hợp năng lượng, truyền thông, hoặc thất bại dữ liệu, hệ thống phải trở lại hoạt động trở lại mà không cần sự can thiệp của người vận hành. |
| Khả năng sử dụng | Khả năng nhận dạng phù hợp  Tính dễ học  Khả năng hoạt động  Bảo vệ lỗi của người dùng  Giao diện người dùng thẩm mỹ  Khả năng tiếp cận | Tất cả các chức năng truy cập tạo ra giá trị cho mỗi loại người dùng phải dễ dàng nhận biết được trong trang ban đầu của hệ thống. Các chức năng ban đầu phải được làm nổi bật.  Tên của các chức năng (ví dụ: nút và trình đơn) phải phù hợp với tên Windows thông thường để nâng cao khả năng của người dùng để tìm hiểu về việc sử dụng chúng.  Việc sử dụng hệ thống phải rõ ràng đối với người sử dụng mới làm quen: không có nhu cầu trợ giúp hay giải thích thêm.  Ứng dụng phải ngăn người dùng nhập thông tin không nhất quán.  Hệ thống phải được cấu hình cho nhiều ngôn ngữ của con người. |
| Hiệu quả hoạt động | Hành vi thời gian  Sức chứa | Trong những giờ cao điểm, điều quan trọng là hiệu năng của hệ thống không làm suy giảm. Tận dụng nguồn tài nguyên  Dự kiến sẽ có tới 10.000 kết nối đồng thời. |
| Bảo vệ | Bảo mật  Chính trực  Không bác bỏ  Trách nhiệm giải trình  Tính xác thực | Dữ liệu về doanh thu và khách hàng chỉ có thể truy cập được bởi chính bản thân khách hàng, người quản lý bán hàng và nhân viên được quản lý bán hàng cho phép.  Khách hàng chỉ có thể thay đổi dữ liệu của riêng mình và mở đơn đặt hàng. Tất cả các dữ liệu khác chỉ có thể được thay đổi bởi nhân viên có thẩm quyền. Các cấp độ thẩm quyền khác nhau có thể được xác định.  Hệ thống phải lưu trữ dữ liệu sau đây về từng đơn đặt hàng: nội dung, ngày và thời gian được phát hành và nhận và bản sắc của khách hàng.  Mật khẩu của người dùng không được hiển thị bởi người dùng hoặc người quản lý hệ thống khác. |
| Khả năng tương thích | Sự cùng tồn tại  Khả năng tương tác | Hệ thống phải có khả năng giao tiếp tự động với các nhà khai thác thẻ tín dụng để cho phép xác nhận thanh toán. |
| Khả năng bảo trì | Mô đun  Khả năng tái sử dụng  Khả năng phân tích  Khả năng biến đổi  Khả năng kiểm tra |  |
| Tính di động | Tính thích nghi  Khả năng cài đặt  Thay thế | Máy chủ phải được cài đặt trên một nền tảng duy nhất. Ứng dụng khách hàng phải được truy cập ít nhất bằng Internet Explorer, Google Chrome và Mozilla Firefox. Ứng dụng khách hàng phải hợp lý với cấu hình của máy tính người dùng để xác định ngôn ngữ, tiền tệ và các định nghĩa khác.  Khách hàng có thể truy cập vào hệ thống thông qua Web mà không cần phải cài đặt thêm bất kỳ thành phần bổ sung ngoại trừ những dịch vụ thường có sẵn cho các trình duyệt. |
| Tính hiệu quả | Tính hiệu quả |  |
| Hiệu quả | Hiệu quả |  |
| Sự thỏa mãn | Tiện ích  Vui lòng  Tiện nghi  Lòng tin |  |
| Tự do khỏi rủi ro | Giảm nhẹ rủi ro kinh tế  Giảm nhẹ rủi ro về sức khoẻ và an toàn  Giảm nhẹ rủi ro môi trường |  |
| Bối cảnh bảo hiểm | Bối cảnh hoàn thiện  Mềm dẻo |  |

**3.6. Mô hình khái niệm sơ bộ**

Mặc dù Chương 6 và 7 trình bày chi tiết các kỹ thuật lập mô hình khái niệm, cần phải đề cập đến ở đây có mối quan hệ phụ thuộc lẫn nhau giữa hệ thống các use case và mô hình khái niệm sơ bộ (Larman, 2004). Mô hình khái niệm sơ bộ được xây dựng trong Inception, và bao gồm một sơ đồ lớp đại diện cho các đơn vị thông tin chính của hệ thống. Nó không cần thiết để đại diện cho thuộc tính. Mặc dù các hiệp hội phải xuất hiện trong mô hình đó, nhưng không cần phải chi tiết các tính năng của họ.

Bằng cách phân tích sơ đồ use case của hệ thống, nhiều khái niệm quan trọng có thể được phát hiện. Những khái niệm này được biểu diễn dưới dạng các lớp trong mô hình khái niệm sơ bộ; Chúng đại diện cho cấu trúc của thông tin sẽ được quản lý bởi hệ thống. Đồng thời, một nhà phân tích, bằng cách quan sát mô hình khái niệm, có thể nhận thấy nếu use case diagram hoàn thành đầy đủ đầy đủ. Xác minh này thường xảy ra khi các thực thể mới được xác định trong quá trình kinh doanh và nó là cần thiết mà chúng được ghi lại bởi hệ thống.

Hình 3.9 trình bày mô hình khái niệm sơ bộ có thể cho sơ đồ use case trình bày trong Hình 3.6. Quá trình phát hiện các lớp bao gồm việc suy nghĩ về các use case và tưởng tượng được thông tin nào (cấp cao) được trao đổi giữa các tác nhân và hệ thống để cho phép quy trình hoạt động. Các mối liên kết giữa các lớp biểu diễn các quan hệ phụ thuộc hoặc các mối quan hệ giữa các mẩu tin do họ đại diện.



Hình 3.9: Mô hình khái niệm ban đầu cho các use case trong Hình 3.6.

Nhìn vào hình 3.6 với mục đích tìm các khái niệm thể hiện trong hình 3.9, có thể thấy rằng:

• **Order book** có lẽ là use case quan trọng nhất trong hệ thống. Nó cho thấy hai khái niệm, **Book** và **Orde**r, có liên quan.

• **Pay for book** thêm một khái niệm mới, **Payment**, có liên quan đến **Order**. Vào thời điểm đó, nhóm có thể quyết định thay đổi tên của use case thành **Pay order**.

• Use case **Receive books** phát sinh khái niệm **Arrival**, có liên quan đến **Book**. Có lẽ lúc đó **EntryOrder** hoặc **BuyOrder** có thể được giới thiệu như là một khái niệm mới.

• **Deliver book** tạo ra một khái niệm mới, **Deliver**, được liên kết với **Order**. Bây giờ **Deliver book** có thể / nên được đổi tên thành **Deliver order**.

• **Register delivery confirmation** tạo ra khái niệm **Confirmation**, được liên kết với **Delivery**.

• **Register book return** tạo ra **Return**, được liên kết với **Order**. Có thể tại thời điểm đó tên của use case nên được thay đổi để **Register order return**.

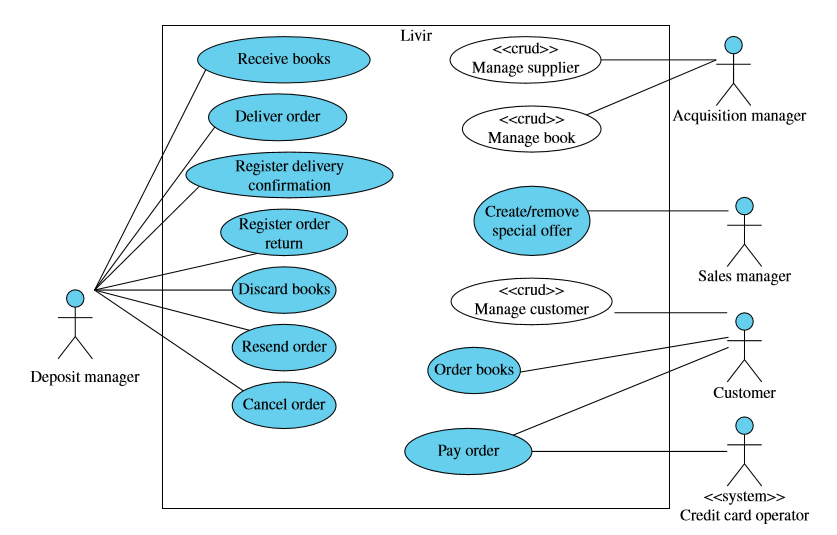
• **Discard book** tạo ra **Discard**, liên quan đến **Book**.

• **Resend book** không tạo ra một khái niệm mới ngay từ cái nhìn đầu tiên vì nó có thể được coi là sự lặp lại của use case **Deliver books**. Tuy nhiên, vẫn phải phân tích thêm.

• **Cancel sale** tạo ra khái niệm **Cancellation**, có liên quan đến **Order**. Có lẽ tên use case cần được thay đổi thành **Cancel order** vào thời điểm này.

• **Create / Remove special offer** tạo ra khái niệm **SpecialOffer**, được liên kết với **Book**.

Việc xác định mô hình khái niệm sơ bộ đó đặc biệt hữu ích để làm giảm sự trực quan hóa cấu trúc của thông tin sẽ được hệ thống quản lý; Điều này giúp thống nhất từ vựng giữa các thành viên trong nhóm và các bên liên quan khác. Các quyết định về thay đổi tên use case để làm sạch từ vựng đã được thực hiện trong Hình 3.10.



Hình 3.10: Use case model với tên được chỉnh sửa và CRUDs đã được thêm

Nhưng cũng có những tiện ích thực tiễn quan trọng sau đây cho mô hình khái niệm sơ bộ. Trong số các khái niệm được trình bày, hầu hết là các yếu tố thông tin được tạo ra hoặc thay đổi trong bối cảnh các use case đã được xác định. Tuy nhiên, một số trong số đó không được tạo ra hoặc không thay đổi bởi các use case, và điều đó có nghĩa là một số use case vẫn có thể bị thiếu. Đây là trường hợp đặc biệt đối với các lớp **Book**, **Publisher** và **Customer**. Những khái niệm này có thể được coi là CRUD, bởi vì chúng cho phép bốn hoạt động cổ điển: **Create, Retrieve, Update và Delete**. Nếu chúng được thêm vào sơ đồ, thay vì đại diện cho chúng một cách riêng lẻ, tốt hơn nên đại diện cho bốn hoạt động bằng cách sử dụng một use case CRUD duy nhất, mà là khuôn mẫu với *<<scrud>>*, như trong hình 3.10.

Một actor mới đã được định nghĩa để quản lý các nhà xuất bản và sách: **Acquisition manager**. Mặt khác, nó đã được quyết định rằng khách hàng sẽ chịu trách nhiệm về những thông tin sổ sách của hộ.

Tại sao không có CRUD cho đơn hàng, hủy, trở lại, thanh toán, …? Bởi vì những điều này đã được quản lý bởi các use case trên biểu đồ, và nó không phải là cần thiết để tạo các use case khác đặc biệt cho họ. Ví dụ, một **order** được tạo ra bởi use case **Order books**; nó được thay đổi bởi các use case như **Cancel order** và **Pay order**; Nó được hình dung (lấy ra) trong nhiều use case; Và, cuối cùng, giả định rằng nó không thể được gỡ bỏ khỏi hệ thống.

Một câu hỏi khác có thể được nêu ra với các bên liên quan là: Các báo cáo mà hệ thống phải đưa ra là gì? Mặc dù đây là use case đơn giản nhất, được giải thích sau, các báo cáo là một nguồn thông tin quan trọng để xác định thông tin nào là cần thiết để đáp ứng nhu cầu của các bên liên quan.

Có một sự khác biệt giữa các báo cáo và truy vấn của CRUD: truy vấn chỉ đơn giản là khôi phục dữ liệu được lưu trữ về một đối tượng; Tuy nhiên, báo cáo thường liên quan đến một số đối tượng, đôi khi từ các lớp khác nhau và nó nhất thiết phải bao gồm một số loại lọc hoặc kết hợp dữ liệu (tổng hợp, sản phẩm, trung bình, giá trị lớn hơn, giá trị thấp hơn, vv). Ví dụ: truy vấn thông tin về một cuốn sách, được cấp ISBN, không phải là trường hợp sử dụng báo cáo vì truy vấn đó đã được bao gồm trong trường hợp sử dụng CRUD.

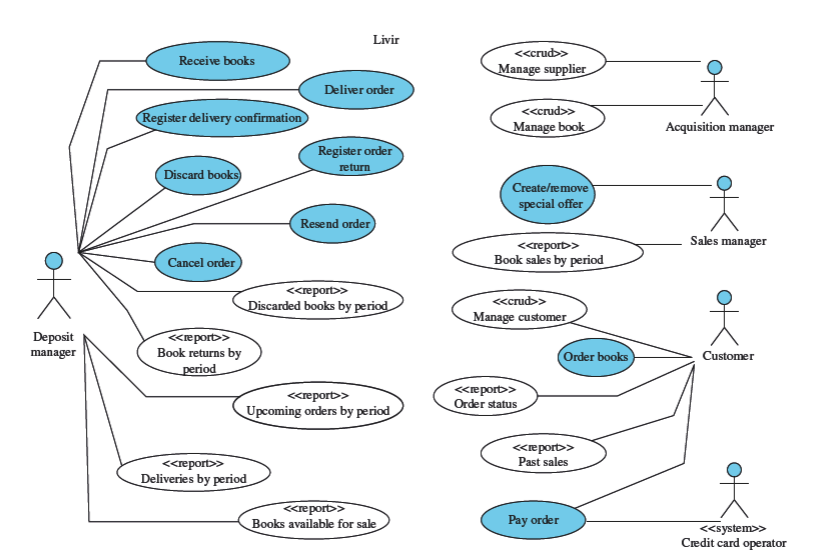
Đối với ví dụ đang chạy, một số báo cáo thú vị đối với một số tác nhân nhất định có thể được xác định, chẳng hạn như:

• **Deposit manager**: Báo cáo các đơn đặt hàng sắp tới theo thời gian, báo cáo về việc phân phối theo thời gian, báo cáo về tổng số sách có sẵn để bán, báo cáo về thu nhập thu được theo thời gian và báo cáo về sách bị loại bỏ theo thời gian .

• **Customer**: Báo cáo tình trạng đơn đặt hàng, và báo cáo về doanh thu trong quá khứ.

• **Sales manager**: Báo cáo về việc bán sách theo thời gian.

Khách hàng có thể yêu cầu các báo cáo khác, và ở trên chỉ là ví dụ. Những nội dung được sử dụng độc quyền bên trong một trong các use case hiện tại phải được loại trừ khỏi danh sách báo cáo. Ví dụ: người quản lý tiền gửi có thể quan tâm đến danh sách các đơn đặt hàng đang đến. Nếu danh sách đó chỉ được lấy ý kiến ​​tại thời điểm lệnh thực sự đến và phải được đăng ký, thì truy vấn phải được coi là một phần của use case **Receive books** và không được đưa vào biểu đồ. Chỉ những báo cáo không nhất thiết là một phần của các use case khác phải được đưa vào sơ đồ; Nếu không danh sách này có thể nhanh chóng phát triển vượt xa mức có thể quản lý được, và có thể là dư thừa và không đầy đủ (vì đây là use case mở rộng với các sơ đồ tuần tự thực sự cho biết các bước nào, kể cả truy vấn, là cần thiết cho từng use case). Hình 3.11 chỉ ra sơ đồ ca sử dụng được cập nhật với các báo cáo (rập khuôn với *<<report>>*) như đã nêu ở trên.



Hình 3.11: Use case diagram với reports

Số lượng các use case << report >> sẽ phụ thuộc vào cấu trúc của thông tin mà chúng hiện diện. Khi sự tham số hóa khả thi, nó phải được sử dụng. Ví dụ: không cần phải có báo cáo **Sales report by week** và **Sales report by month**. Trừ khi các cấu trúc dữ liệu hiện tại khác nhau, chúng là một báo cáo. Tham chiếu đến một tuần, tháng hoặc bất kỳ khoảng thời gian nào khác chỉ là một tham số.

Tuy nhiên, không nên lập các báo cáo nhóm với các tính chất đa dạng, ví dụ như **Sales report by book** và **Deliveries report by period of time**: các đầu vào và đầu ra khác nhau. Do đó, hai báo cáo này phải được coi là hai use case khác nhau.

Như trong hình 3.11, số use case có thể trở nên cao và sơ đồ có thể nhanh chóng trở nên khó khăn để tổ chức. Đây là một trong những lý do để tránh, bao gồm các mảnh trong các biểu đồ này. Mỗi use case sẽ được chi tiết bằng các phương tiện khác. Ngoài ra, các use case CRUD và báo cáo có thể không được bao gồm trong sơ đồ use case chính nếu hệ thống có kích thước vừa phải. Chúng phải được liệt kê ở một nơi khác (ví dụ như bảng tính hoặc biểu đồ use case tách biệt), để không làm phức tạp sơ đồ use case chính, Một giải pháp khác có thể được sử dụng đôi khi là đại diện cho chúng với một màu khác (như đã làm trong Hình 3.10 và 3.11), để chúng không cản trở việc hiển thị các trường hợp sử dụng chính trong biểu đồ

**3.7. Quá trình cho đến nay**

|  |  |  |
| --- | --- | --- |
|  | **Inception** | **Elaboration** |
| **Mô Hình Kinh Doanh** | Xây dựng một cái nhìn chung của hệ thống:  • Xây dựng business use case diagram và xác định phạm vi tự động hóa cho dự án.  • Xây dựng sơ đồ hoạt động sơ bộ cho các business use cases  • Xây dựng sơ đồ sơ đồ nhà nước sơ bộ cho các đối tượng kinh doanh chính. |  |
| **Các Yêu cầu** | **Chuẩn bị sơ đồ use case (yêu cầu chức năng):**  **• Xác định các actor của hệ thống từ business use case diagram.**  **• Xác định use case của hệ thống từ business use case diagram, activity & state machine diagrams từ business modeling.**  **Xác định các yêu cầu phi chức năng như chú thích use case:**  **• Xác định các quy tắc kinh doanh chính liên quan đến các use case.**  **• Xác định các vấn đề chính về chất lượng liên quan đến các use case. Xác định các yêu cầu bổ sung** |  |
| **Phân tích và Thiết kế** | **Chuẩn bị mô hình khái niệm sơ bộ bằng cách quan sát các use case và các khái niệm cần thiết của chúng** |  |
| **Thực hiện** |  |  |
| **Thử nghiệm** |  |  |
| **Quản lý dự án** |  |  |

**3.8. Câu hỏi**

1. Giải thích sự khác biệt giữa business use case và system use case.

2. Tiện ích của một system use case thông qua quá trình phát triển phần mềm là gì?

3. Các business actor và business worker nào được chuyển đổi thành các system actor?

4. Có những yêu cầu chức năng, phi chức năng và bổ sung nào? Các tính năng nào chúng có thể có?

5. Tại sao một mô hình khái niệm sơ bộ phải được thực hiện trong giai đoạn Khởi động?

Alternatively it could be recorded separately as a numbered list or spreadsheet, with a reference to the use cases through a unique number

As can be seen in Figure 3.7, the use case diagram with notes will quickly become too complex to be useful. This is why it is recommend that such annotations should be recorded inside the use case spectification or in a separate document. Most CASE tools allow that a diagram element such as a use case have a specification window as shown in Figure 3.8. one exception to that rule would be the case when the annotation is absolutely required to help understanding the diagram. however, this is seldom the case

The system that is going to be analyzed is like a forest. To explore an unknown forest one cannot start by examining each plant and animal. There is a saying that states that some people cannot see the forest for the trees. The forest is the system and the trees arre the requirements. Only at the end of the process could a team say that they acquired knowledge about the smallest parts. First, a view of the whole must be taken, and only after that the details may be studied

Thus, the Inception phase must provide a view of the whole - so that what is more importants can be seen first - and then the whole can be divided into parts so that the details can be analyzed and finally a solution designed. The organization of iterations in the Elaboration and Construction