

HỌC VIỆN CÔNG NGHỆ Bưu Chính Viễn Thông



**BÁO CÁO HẾT MÔN
PHÂN TÍCH THIẾT KẾ HỆ THỐNG THÔNG TIN**

| | |
|----------------|---------------|
| Họ và tên: | Bùi Thị Thắm |
| Mã Sinh Viên: | B20DCCN656 |
| Nhóm học phần: | Nhóm 06 |
| Hệ: | Chính quy |
| Giảng viên | Trần Đình Quê |

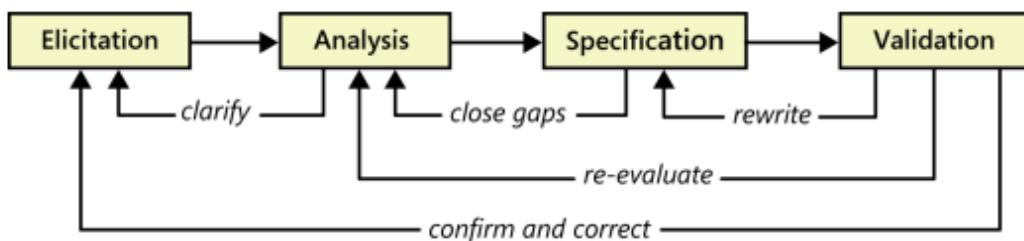
Hà Nội, 2023

Mục lục

| | |
|--|-----------|
| 1. 4 pha trong framework phát triển yêu cầu phần mềm..... | 5 |
| 1.1. Requirements elicitation | 5 |
| 1.2. Requirements analysis | 7 |
| 1.3. Requirements specification | 9 |
| 1.4. Requirements validation | 10 |
| 2. Giải thích Figure 4.1..... | 11 |
| 3. Công việc của người làm BA: Knowledge & Skills. Để trở thành BA | 13 |
| 3.1. Knowledge | 13 |
| 3.2. Skills | 15 |
| 3.3. Kiến thức phân tích cần thiết: Knowledge | 18 |
| 3.4. Để trở thành BA..... | 19 |
| 4. Tiến trình phát triển phần mềm Agile & Waterfall..... | 22 |
| 4.1. 6 giai đoạn của mô hình thác nước | 22 |
| 4.2. 6 bước triển khai mô hình Agile | 24 |
| 5. BA trong các dự án Agile | 33 |
| 5.1. Vai trò của người phân tích trên các dự án Agile | 33 |
| 5.2. Tạo ra một nhóm cộng tác | 34 |
| 6. Mô hình yêu cầu phần mềm: Liệt kê các biểu đồ có thể sử dụng cho mô hình yêu cầu phần mềm..... | 35 |
| 6.1. Các biểu đồ có thể sử dụng cho mô hình yêu cầu phần mềm: | 36 |
| 6.2. Liên hệ giữa yêu cầu của khách hàng với các thành phần của mô hình phân tích | 37 |
| 6.3. Biểu đồ luồng dữ liệu:..... | 37 |
| 6.4. Biểu đồ Swimlane:..... | 38 |
| 6.5. Biểu đồ trạng thái:..... | 38 |
| 6.6. Biểu đồ giao diện: | 39 |
| 6.7. Bảng quyết định và cây quyết định:..... | 39 |
| 6.8. Bảng phản ứng sự kiện: | 40 |
| 6.9. Biểu đồ lớp (Class diagrams): | 40 |
| 6.10. Biểu đồ Usecase diagrams: | 40 |
| 6.11. Biểu đồ hoạt động (Activity diagrams): | 41 |
| 6.12. Biểu đồ trạng thái (State diagrams): | 41 |
| 7. Các giai đoạn thiết kế phần mềm: Thiết kế kiến trúc, Thiết kế chi tiết..... | 41 |
| 7.1 Khái niệm về thiết kế phần mềm | 41 |
| 7.5 Mô tả kiến trúc | 43 |
| 7.6 Thiết kế kiến trúc | 46 |
| 7.7 Thiết kế chi tiết | 47 |
| 8. Phương pháp Agile trong phát triển phần mềm..... | 52 |
| 8.1. Ưu điểm của mô hình Agile | 53 |
| 8.2. Nhược điểm của mô hình Agile | 54 |

| | |
|--|------------|
| 8.3. Các nguyên tắc cần tuân thủ trong phương pháp Agile | 54 |
| 8.4. Cách thức thực hiện phương pháp Agile | 55 |
| 8.5. Ứng dụng phương pháp Agile vào quản trị công việc..... | 56 |
| 8.6. Đặc trưng cơ bản của Agile | 57 |
| 9. Chọn kỹ thuật mô hình hay biểu diễn thích hợp | 58 |
| 9.1. Kỹ thuật mô hình | 58 |
| 9.2. Biểu diễn thích hợp | 59 |
| 9.3. So sánh kỹ thuật mô hình và biểu diễn thích hợp..... | 60 |
| 9.4. Chọn kỹ thuật mô hình hay biểu diễn thích hợp | 61 |
| 10. Trình bày ngắn gọn 14 biểu đồ UML:..... | 62 |
| 10.1. Structural diagram..... | 62 |
| 10.2. Behavior Diagrams | 65 |
| 11. Trình bày các đặc trưng phi chức năng hay thuộc tính chất lượng phần mềm..... | 68 |
| 12. Lớp và quan hệ lớp từ phân tích đến thiết kế. Code tương ứng. Cho ví dụ..... | 73 |
| 13. Khảo sát trên mạng các Hệ thống sau đây: | 81 |
| A. Hệ quản lý thư viện (Library Management System LibMaS)..... | 81 |
| 13a.1. Trình bày bằng ngôn ngữ tự nhiên và biểu đồ context cho các hệ thống ... | 81 |
| 13a.2. Mô tả Actor hệ thống | 82 |
| 13a.3 Mô tả yêu cầu chức năng và phi chức năng của hệ thống | 83 |
| 13a.4. Trình bày Biểu đồ Usecase THEO 2 LEVEL | 85 |
| 13a.5. Biểu đồ activity hệ thống..... | 87 |
| 13a.6 Trình bày Biểu đồ LỚP THIẾT KẾ với DAO và layer MVC cho các Hệ thống | 91 |
| B. HỆ THỐNG THƯƠNG MẠI ĐIỆN TỬ E-COMMERCE | 93 |
| 13b.1. Trình bày bằng ngôn ngữ tự nhiên và biểu đồ context cho các hệ thống ... | 93 |
| 13b.2 Mô tả các actor..... | 96 |
| 13b.3 Mô tả yêu cầu các chức năng và phi chức năng của hệ thống bằng ngôn ngữ tự nhiên và dạng Bảng | 99 |
| Yêu cầu phi chức năng quan trọng cho hệ thống thương mại điện tử: | 102 |
| 13b.4. Trình bày Biểu đồ use case THEO 2 LEVEL cho các Hệ thống..... | 102 |
| 13b.5 Biểu đồ activity | 109 |
| 13b.6 Trình bày Biểu đồ LỚP THIẾT KẾ với DAO và layer MVC cho Hệ thống | 122 |
| 13b.7 Thiết kế Bảng và sinh ra cơ sở dữ liệu với mySQL | 124 |
| C.HỆ ĐĂNG KÍ HỌC THEO TÍN CHỈ Ở TRƯỜNG ĐẠI HỌC | 129 |
| 13c.1 Trình bày bằng ngôn ngữ tự nhiên và biểu đồ context cho các hệ thống.. | 129 |
| 13c.2 Mô tả các actor | 131 |
| 13c.3 Mô tả yêu cầu các chức năng và phi chức năng của hệ thống bằng ngôn ngữ tự nhiên và dạng bảng | 134 |
| 13c.4 Trình bày Biểu đồ use case THEO 2 LEVEL cho các Hệ thống | 135 |
| 13c.5. Biểu đồ activity | 137 |
| 13c.6. Trình bày Biểu đồ LỚP THIẾT KẾ với DAO và layer MVC cho các Hệ thống | 138 |

1. 4 pha trong framework phát triển yêu cầu phần mềm



1.1. Requirements elicitation

Xác định tầm nhìn sản phẩm và phạm vi dự án:

- Tầm nhìn và phạm vi bao gồm các thông tin yêu cầu về sản phẩm.
- Tầm nhìn cung cấp cho tất cả các bên liên quan một sự hiểu biết chung về kết quả của sản phẩm.
- Phạm vi xác định ranh giới giữa nội dung bên trong và nội dung bên ngoài cho một bản phát hành hoặc lần lặp lại cụ thể.
- Cùng với nhau, tầm nhìn và phạm vi cung cấp một tài liệu tham khảo để đánh giá các yêu cầu được đề xuất.
- Tầm nhìn sẽ vẫn tương đối ổn định trong suốt dự án, nhưng mỗi lần phát hành hoặc lặp lại theo kế hoạch đều cần có tuyên bố phạm vi riêng.

Xác định các nhóm người dùng và đặc điểm của họ: Để tránh bỏ sót các nhu cầu của bất kỳ cộng đồng người dùng nào, phải xác định các nhóm người dùng khác nhau cho sản phẩm. Các nhóm này có thể khác nhau về tần suất sử dụng, tính năng được sử dụng, mức đặc quyền, hoặc kinh nghiệm. Mô tả các khía cạnh của nhiệm vụ công việc, thái độ, vị trí hoặc đặc điểm cá nhân của họ có thể ảnh hưởng đến thiết kế sản phẩm. Tạo ra những nhân vật người dùng, mô tả về những người tưởng tượng sẽ đại diện cho các nhóm người dùng cụ thể.

Chọn người đại diện cho từng nhóm người dùng: Chọn một cá nhân có thể hiệu quả đại diện cho khách hàng thực sự của từng nhóm người dùng. Người đại diện sản phẩm sẽ trình bày những nhu cầu của nhóm người dùng và đưa ra quyết định thay mặt họ. Điều này dễ dàng nhất trong việc phát triển hệ thống thông tin nội bộ, nơi người dùng là đồng nghiệp. Đối với việc phát triển sản phẩm thương mại, tận dụng mối quan hệ hiện tại với các khách hàng chính hoặc các trang thử nghiệm Beta để tìm người đại diện sản phẩm phù hợp.

Tổ chức nhóm tập trung với người dùng điển hình: Tập hợp các nhóm người dùng đại diện cho sản phẩm trước đó hoặc sản phẩm tương tự. Thu thập ý kiến từ họ về cá tính năng và các đặc điểm về chất lượng cho sản phẩm đang phát triển. Nhóm tập trung đặc biệt hữu ích trong việc phát triển sản phẩm thương mại, nơi có thể có một cơ sở khách hàng lớn và đa dạng. Khác với người đại diện sản phẩm, nhóm tập trung thường không có quyền quyết định.

Hợp tác với đại diện người dùng để xác định yêu cầu người dùng: Tương tác với các đại diện người dùng để khám phá các nhiệm vụ mà họ cần thực hiện với phần mềm và giá trị mà họ đang cố gắng đạt được. Yêu cầu người dùng có thể được biểu thị dưới dạng các tình huống sử dụng, câu chuyện người dùng hoặc kịch bản. Thảo luận về sự tương tác giữa người dùng và hệ thống mà sẽ cho phép họ hoàn thành mỗi nhiệm vụ.

Xác định sự kiện hệ thống và phản hồi: Liệt kê các sự kiện bên ngoài mà hệ thống có thể gặp phải và phản hồi dự kiến của nó đối với mỗi sự kiện. Có ba loại sự kiện bên ngoài. *Sự kiện tín hiệu* là các tín hiệu điều khiển hoặc dữ liệu nhận từ các thiết bị phần cứng bên ngoài. *Sự kiện thời gian*, hoặc dựa trên thời gian, kích hoạt một phản hồi, chẳng hạn như luồng dữ liệu bên ngoài mà hệ thống của bạn tạo ra vào cùng một thời điểm hàng đêm. *Sự kiện kinh doanh* kích hoạt các trường hợp sử dụng trong các ứng dụng kinh doanh.

Tiến hành cuộc phỏng vấn thu thập yêu cầu: Phỏng vấn có thể được thực hiện một cách cá nhân hoặc với một nhóm nhỏ các bên liên quan. Đây là một cách hiệu quả để thu thập yêu cầu mà không tốn quá nhiều thời gian của các bên liên quan vì bạn chỉ gặp gỡ họ để thảo luận về những yêu cầu cụ thể quan trọng với họ. Cuộc phỏng vấn hữu ích để riêng lẻ thu thập yêu cầu từ những người chuẩn bị cho các buổi làm việc nhằm giải quyết các xung đột.

Tổ chức các buổi làm việc thu thập yêu cầu dưới sự hỗ trợ: Các buổi làm việc thu thập yêu cầu được hỗ trợ, cho phép sự hợp tác giữa các nhà phân tích và khách hàng, là một cách mạnh mẽ để khám phá nhu cầu của người dùng và soạn thảo tài liệu yêu cầu.

Quan sát người dùng trong quá trình thực hiện công việc của họ: Theo dõi người dùng thực hiện các nhiệm vụ kinh doanh giúp xác định ngữ cảnh cho việc sử dụng tiềm năng của một ứng dụng mới. Sơ đồ luồng quy trình đơn giản có thể mô tả các bước và quyết định liên quan và cho thấy cách các nhóm người dùng khác nhau tương tác.

Việc tài liệu hóa luồng quy trình kinh doanh sẽ giúp xác định yêu cầu cho một giải pháp dự định hỗ trợ quy trình đó.

Phân phát các bảng câu hỏi: Các bảng câu hỏi là cách để khảo sát các nhóm người dùng lớn để xác định những gì họ cần. Bảng câu hỏi hữu ích với bất kỳ nhóm người dùng lớn nào nhưng đặc biệt hữu ích với các nhóm phân tán. Nếu câu hỏi được viết kỹ càng, bảng câu hỏi có thể giúp nhanh chóng xác định thông tin phân tích về các nhu cầu. Các nỗ lực thu thập yêu cầu bổ sung sau đó có thể tập trung theo kết quả bảng câu hỏi.

Phân tích tài liệu hiện có: Tài liệu hiện có có thể giúp tiết lộ cách các hệ thống hiện tại hoạt động như thế nào hoặc điều gì mà chúng nên thực hiện. Tài liệu bao gồm bất kỳ thông tin nào viết về hệ thống hiện tại, quy trình kinh doanh, đặc tả yêu cầu, nghiên cứu đối thủ và hướng dẫn sử dụng gói COTS (commercial off-the-shelf). Xem xét và phân tích các tài liệu có thể giúp xác định chức năng cần phải duy trì, chức năng không được sử dụng, cách mà mọi người thực hiện công việc hiện tại, những gì đối thủ cung cấp và những gì nhà cung cấp nói về phần mềm của họ nên thực hiện.

Nghiên cứu báo cáo vấn đề của hệ thống hiện tại để có ý tưởng về yêu cầu: Các báo cáo về vấn đề và yêu cầu nâng cao từ người dùng cung cấp một nguồn phong phú ý tưởng cho khả năng bao gồm trong phiên bản sau hoặc trong sản phẩm mới.

Tái sử dụng các yêu cầu hiện có: Nếu khách hàng yêu cầu chức năng tương tự đã có trong sản phẩm hiện tại, xem liệu yêu cầu có đủ linh hoạt để cho phép tái sử dụng hoặc điều chỉnh các thành phần phần mềm hiện có. Dự án thường có thể tái sử dụng các yêu cầu tuân theo các quy tắc kinh doanh của tổ chức, chẳng hạn như yêu cầu về bảo mật, và các yêu cầu tuân theo các quy định của chính phủ, chẳng hạn như yêu cầu về khả năng tiếp cận.

1.2. Requirements analysis

Phân tích yêu cầu bao gồm việc *làm rõ yêu cầu* để đảm bảo tất cả các bên liên quan hiểu rõ chúng và kiểm tra chúng để tìm lỗi, thiếu sót và những khuyết điểm khác. Phân tích bao gồm việc phân rã các yêu cầu cấp cao thành các mức chi tiết thích hợp, xây dựng nguyên mẫu, đánh giá khả thi và thương lượng ưu tiên. Mục tiêu là phát triển yêu cầu đủ chất lượng và chính xác để các quản lý có thể xây dựng các ước tính dự án thực tế và nhân viên kỹ thuật có thể tiến hành thiết kế, xây dựng và kiểm thử.

Mô hình hóa môi trường ứng dụng: Sơ đồ ngữ cảnh là một mô hình phân tích đơn giản cho thấy cách hệ thống mới khớp vào môi trường của nó. Nó xác định ranh giới và giao diện giữa hệ thống đang phát triển và các thực thể bên ngoài.

Tạo nguyên mẫu giao diện người dùng và kỹ thuật: Khi các nhà phát triển hoặc người dùng không chắc chắn về yêu cầu, xây dựng một nguyên mẫu - một phiên bản một phần, có thể có hoặc dự thảo - để làm cho các khái niệm và khả năng trở nên cụ thể hơn. Nguyên mẫu cho phép các nhà phát triển và người dùng đạt được hiểu biết chung về vấn đề đang được giải quyết, cũng như giúp xác nhận yêu cầu.

Phân tích khả năng thực hiện yêu cầu: BA nên làm việc với các nhà phát triển để đánh giá khả năng thực hiện từng yêu cầu với chi phí và hiệu suất chấp nhận được trong môi trường hoạt động dự kiến. Điều này cho phép các bên liên quan hiểu rõ rู้ rõ liên quan đến việc thực hiện mỗi yêu cầu, bao gồm xung đột và phụ thuộc với các yêu cầu khác, phụ thuộc vào các yếu tố bên ngoài và các rào cản kỹ thuật. Những yêu cầu kỹ thuật không thể thực hiện hoặc quá đắt để triển khai có thể được đơn giản hóa và vẫn đóng góp vào việc đạt được mục tiêu kinh doanh của dự án.

Ưu tiên các yêu cầu: Việc ưu tiên các yêu cầu quan trọng để đảm bảo đội triển khai chức năng có giá trị cao nhất hoặc có tính cấp thiết nhất trước tiên. Áp dụng một phương pháp phân tích để xác định mức độ ưu tiên thực hiện tương đối của các tính năng sản phẩm, trường hợp sử dụng, câu chuyện người dùng hoặc yêu cầu chức năng. Dựa trên mức độ ưu tiên, xác định phiên bản hoặc bước tăng cường nào sẽ chứa mỗi tính năng hoặc tập hợp yêu cầu.

Tạo từ điển dữ liệu: Các định nghĩa về các mục dữ liệu và cấu trúc liên quan đến hệ thống được lưu trữ trong từ điển dữ liệu. Điều này cho phép tất cả mọi người làm việc trên dự án sử dụng các định nghĩa dữ liệu nhất quán.

Mô hình hóa yêu cầu: Mô hình phân tích là một biểu đồ mô tả yêu cầu một cách hình ảnh, khác biệt so với biểu diễn văn bản của một danh sách yêu cầu chức năng. Các mô hình có thể tiết lộ yêu cầu sai, không nhất quán, thiếu sót và thừa thãi. Các mô hình như sơ đồ luồng dữ liệu, sơ đồ thực thể-quan hệ, sơ đồ chuyển trạng thái, bảng trạng thái, bản đồ hội thoại, cây quyết định và các mô hình khác.

Phân tích giao diện giữa hệ thống và thế giới bên ngoài: Tất cả các hệ thống phần mềm đều có kết nối với các phần khác của thế giới thông qua các giao diện bên ngoài. Hệ thống thông tin có giao diện người dùng và thường trao đổi dữ liệu với các hệ

thống phần mềm khác. Các hệ thống nhúng liên quan đến mối quan hệ giữa phần mềm và các thành phần phần cứng. Ứng dụng kết nối mạng có giao diện truyền thông. Phân tích những cái này giúp đảm bảo ứng dụng sẽ khớp một cách mượt mà vào môi trường của nó.

Phân bổ yêu cầu cho các hệ thống phụ: Yêu cầu cho một sản phẩm phức tạp chứa nhiều hệ thống phụ cần được chia phần cho các phần mềm, phần cứng và hệ thống và thành phần con người khác nhau.

1.3. Requirements specification

Đặc tả yêu cầu chính là việc tài liệu hóa các yêu cầu khác nhau một cách nhất quán, dễ tiếp cận và có thể được xem xét, để hiểu rõ bởi các khán giả dự định. Yêu cầu kinh doanh có thể được ghi lại trong một tài liệu tầm nhìn và phạm vi. Yêu cầu người dùng thường được biểu diễn dưới dạng trường hợp sử dụng hoặc câu chuyện người dùng. Các yêu cầu chức năng và không chức năng chi tiết của phần mềm được ghi lại trong tài liệu yêu cầu phần mềm (SRS) hoặc công cụ quản lý yêu cầu khác.

Áp dụng mẫu tài liệu yêu cầu: Áp dụng các mẫu tiêu chuẩn để tài liệu hóa yêu cầu trong tổ chức.

Xác định nguồn gốc yêu cầu: Để đảm bảo rằng tất cả các bên liên quan biết tại sao mỗi yêu cầu cần thiết, hãy theo dấu từng yêu cầu về nguồn gốc của nó. Điều này có thể là một trường hợp sử dụng hoặc một loại thông tin khách hàng khác, một yêu cầu hệ thống cấp cao hoặc một quy tắc kinh doanh. Ghi lại các bên liên quan bị ảnh hưởng bởi mỗi yêu cầu cho biết phải liên hệ với ai khi có yêu cầu thay đổi. Nguyên nhân của yêu cầu có thể được xác định thông qua các liên kết có thể theo dấu hoặc thông qua việc xác định một thuộc tính yêu cầu cho mục đích này.

Đặt nhãn duy nhất cho mỗi yêu cầu: Xác định một quy ước để đánh dấu định danh duy nhất cho mỗi yêu cầu. Quy ước này phải đủ mạnh để chịu được các thay đổi thêm, xóa và thay đổi được thực hiện trong các yêu cầu trong thời gian. Đặt nhãn cho yêu cầu cho phép theo dấu yêu cầu và ghi lại các thay đổi được thực hiện.

Ghi lại quy tắc kinh doanh: Quy tắc kinh doanh bao gồm chính sách doanh nghiệp, quy định của chính phủ, tiêu chuẩn và thuật toán tính toán. Ghi lại các quy tắc

kinh doanh riêng biệt khỏi yêu cầu dự án vì chúng thường tồn tại vượt ra ngoài phạm vi của một dự án cụ thể.

Xác định yêu cầu không chức năng: Có khả năng triển khai một giải pháp hoạt động chính xác như mong muốn nhưng không đáp ứng kỳ vọng chất lượng của người dùng. Để tránh vấn đề đó, cần vượt xa khỏi cuộc thảo luận về chức năng để hiểu các đặc điểm về chất lượng quan trọng cho sự thành công. Những đặc điểm này bao gồm hiệu suất, độ tin cậy, tính sử dụng, khả năng thay đổi và nhiều đặc điểm khác. Ý kiến đóng góp của khách hàng về mức quan trọng tương đối của các thuộc tính chất lượng này giúp nhà phát triển đưa ra quyết định thiết kế thích hợp. Hơn nữa, xác định yêu cầu giao diện bên ngoài, ràng buộc thiết kế và thực hiện, vấn đề toàn cầu hóa và các yêu cầu không chức năng khác.

1.4. Requirements validation

Xác thực đảm bảo rằng các yêu cầu là chính xác, thể hiện các đặc điểm chất lượng mong muốn và sẽ đáp ứng nhu cầu của khách hàng.

Xem xét các yêu cầu: Xem xét về yêu cầu, đặc biệt là loại xem xét nghiêm ngặt gọi là kiểm tra, là một trong những thực hành chất lượng phần mềm có giá trị cao nhất có sẵn. Tập hợp một nhóm nhỏ các nhà xem xét đại diện cho các góc nhìn khác nhau, và cẩn thận xem xét các yêu cầu viết, mô hình phân tích và thông tin liên quan để tìm các lỗi. Những xem xét sơ bộ không chính thức trong quá trình phát triển yêu cầu cũng rất có giá trị. Việc đào tạo các thành viên trong nhóm về cách thực hiện xem xét yêu cầu hiệu quả và áp dụng quy trình xem xét cho tổ chức của bạn là rất quan trọng.

Kiểm tra các yêu cầu: Kiểm tra đại diện cho một cách nhìn khác về yêu cầu. Viết các kiểm tra đòi hỏi bạn phải suy nghĩ về cách xác định xem chức năng dự kiến đã được triển khai đúng cách hay chưa. Tạo ra các kiểm tra từ các yêu cầu của người dùng để mô tả hành vi kỳ vọng của sản phẩm dưới điều kiện cụ thể.

Xác định tiêu chí chấp nhận: Yêu cầu người dùng mô tả cách họ sẽ xác định xem giải pháp có đáp ứng nhu cầu của họ và có phù hợp để sử dụng hay không. Tiêu chí chấp nhận bao gồm sự kết hợp của việc phần mềm vượt qua một tập hợp xác định các kiểm tra chấp nhận dựa trên yêu cầu của người dùng, thể hiện sự thỏa mãn của các yêu cầu phi chức năng cụ thể, theo dõi các lỗi và vấn đề chưa giải quyết, có cơ sở hạ tầng và đào tạo để triển khai thành công, và nhiều yếu tố khác.

Mô phỏng các yêu cầu: Có sẵn các công cụ thương mại cho phép một nhóm dự án mô phỏng một hệ thống để xuất hoặc bổ sung thông qua các đặc tả yêu cầu viết. Mô phỏng đưa prototyping lên một cấp độ tiếp theo, cho phép các nhà phân tích kinh doanh cùng người dùng nhanh chóng xây dựng các mẫu thực thi của một hệ thống. Người dùng có thể tương tác với hệ thống mô phỏng để xác thực yêu cầu và đưa ra các lựa chọn thiết kế, giúp yêu cầu trở nên sống động trước khi chúng được biến thành mã nguồn. Mô phỏng không thể thay thế việc thu thập và phân tích yêu cầu cẩn thận, nhưng nó cung cấp một bổ sung mạnh mẽ.

2. Giải thích Figure 4.1

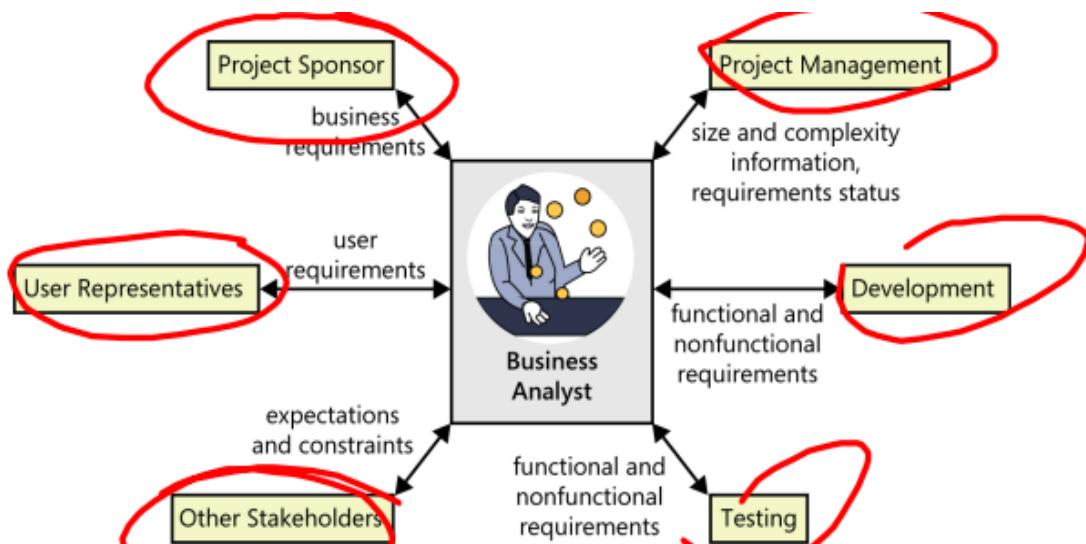


FIGURE 4-1 The business analyst bridges communication between customer and development stakeholders.

Người phân tích kinh doanh là cá nhân có trách nhiệm chính để thu thập, phân tích, tài liệu và xác nhận nhu cầu của các bên liên quan trong dự án.

Business analyst là một vai trò trong dự án, không nhất thiết phải là một chức danh công việc cụ thể. Đồng nghĩa với Business analyst bao gồm những từ như nhà phân tích yêu cầu, nhà phân tích hệ thống, kỹ sư yêu cầu, quản lý yêu cầu, nhà phân tích ứng dụng, nhà phân tích hệ thống doanh nghiệp, nhà phân tích kinh doanh IT và chỉ đơn giản là nhà phân tích. Các chức danh công việc này được sử dụng một cách không đồng nhất từ tổ chức này sang tổ chức khác. Một hoặc nhiều chuyên gia chuyên nghiệp có thể thực hiện vai trò này trong dự án cụ thể hoặc nó có thể được giao cho các thành viên trong nhóm thực hiện cũng làm các chức năng khác trong dự án. Những thành viên trong nhóm này bao gồm người quản lý dự án, quản lý sản phẩm, chủ sở hữu sản phẩm, chuyên gia về chủ đề (SME), nhà phát triển và đôi khi cả người dùng.

Một số hoạt động điển hình có thể thực hiện khi đóng vai trò nhà phân tích:

- **Xác định yêu cầu kinh doanh:** Công việc như một BA bắt đầu khi giúp doanh nghiệp hoặc nhà tài trợ dự án, người quản lý sản phẩm hoặc quản lý tiếp thị xác định yêu cầu kinh doanh của dự án. Có thể đề xuất một mẫu cho tài liệu tầm nhìn và phạm vi và làm việc với những người giữ tầm nhìn để giúp họ diễn đạt nó một cách rõ ràng.

- **Lập kế hoạch cho phương pháp yêu cầu:** Nhà phân tích nên phát triển kế hoạch để thu thập, phân tích, tài liệu, xác nhận và quản lý yêu cầu trong suốt dự án. Làm việc chặt chẽ với người quản lý dự án để đảm bảo rằng các kế hoạch này phù hợp với kế hoạch tổng thể của dự án và sẽ giúp đạt được mục tiêu của dự án.

- **Xác định các bên liên quan trong dự án và các lớp người dùng:** Làm việc với các nhà tài trợ kinh doanh để chọn ra các đại diện phù hợp cho mỗi lớp người dùng, kêu gọi họ tham gia và thương lượng về trách nhiệm của họ. Giải thích những gì BA muốn từ đồng cộng tác và đồng tình về một mức độ tham gia thích hợp từ mỗi người.

- **Thu thập yêu cầu:** Một nhà phân tích giúp người dùng diễn đạt những khả năng hệ thống họ cần để đáp ứng các mục tiêu kinh doanh bằng cách sử dụng nhiều kỹ thuật thu thập thông tin khác nhau.

- **Phân tích yêu cầu:** Tìm kiếm các yêu cầu dẫn xuất là hậu quả logic của những gì khách hàng yêu cầu và các yêu cầu ngầm định mà khách hàng có vẻ mong đợi mà không nói ra. Sử dụng các mô hình yêu cầu để nhận biết các mẫu, xác định các khoảng trống trong yêu cầu, tiết lộ các yêu cầu xung đột và xác nhận rằng tất cả yêu cầu được chỉ định nằm trong phạm vi. Làm việc với các bên liên quan để xác định mức độ chi tiết cần thiết để chỉ định yêu cầu người dùng và yêu cầu chức năng.

- **Ghi chép yêu cầu:** Nhà phân tích có trách nhiệm ghi chép yêu cầu một cách có tổ chức và rõ ràng mà mô tả một cách rõ ràng giải pháp sẽ giải quyết vấn đề của khách hàng. Sử dụng các mẫu tiêu chuẩn giúp tăng tốc quá trình phát triển yêu cầu bằng cách nhắc nhở nhà phân tích về các chủ đề để thảo luận với đại diện của người dùng.

- **Truyền đạt yêu cầu:** Truyền đạt yêu cầu một cách hiệu quả và hiệu suất cho tất cả các bên liên quan. Nhà phân tích nên xác định khi nào nên đại diện cho yêu cầu bằng cách sử dụng các phương pháp khác ngoài văn bản, bao gồm các loại mô hình phân tích hình ảnh khác nhau, bảng, phương trình toán học và mẫu nguyên mẫu. Truyền đạt không chỉ là việc đưa yêu cầu lên giấy. Nó liên quan đến sự hợp tác liên tục với nhóm để đảm bảo họ hiểu thông tin đang được truyền đạt.

- Kiểm tra và xác nhận yêu cầu: Nhà phân tích phải đảm bảo rằng các tuyên bố yêu cầu có những đặc điểm mong muốn và một giải pháp dựa trên các yêu cầu sẽ đáp ứng nhu cầu của các bên liên quan. Các nhà phân tích là những người tham gia chính trong việc xem xét yêu cầu. Xem xét các thiết kế và thử nghiệm được dẫn xuất từ yêu cầu để đảm bảo rằng yêu cầu đã được hiểu đúng cách. Nếu tạo ra các bài kiểm tra chấp nhận thay vì yêu cầu chi tiết trong dự án agile, thì những bài kiểm tra đó cũng nên được xem xét.

- Hỗ trợ quyết định ưu tiên yêu cầu: Nhà phân tích phải tạo sự hợp tác và thương lượng giữa các bên liên quan khác nhau và các nhà phát triển để đảm bảo họ đưa ra quyết định ưu tiên hợp lý phù hợp với việc đạt được mục tiêu kinh doanh.

- Quản lý yêu cầu: Một nhà phân tích kinh doanh tham gia trong toàn bộ vòng đời phát triển phần mềm, vì vậy họ nên giúp tạo ra, xem xét và thực hiện kế hoạch quản lý yêu cầu của dự án. Sau khi thiết lập một cơ sở yêu cầu cho một phiên bản sản phẩm cụ thể hoặc vòng lặp phát triển, trọng tâm của nhà phân tích dịch chuyển sang việc theo dõi tình trạng của các yêu cầu đó, xác nhận sự hài lòng của chúng trong sản phẩm và quản lý các thay đổi đối với cơ sở yêu cầu. Với sự đóng góp từ các đồng nghiệp khác nhau, nhà phân tích thu thập thông tin về tính liên kết nối các yêu cầu cá nhân với các yếu tố hệ thống khác.

3. Công việc của người làm BA: Knowledge & Skills. Để trở thành BA

3.1. Knowledge

Nhà phân tích trước tiên phải hiểu mục tiêu kinh doanh của dự án và sau đó xác định người dùng, các yêu cầu về chức năng và chất lượng cho phép các nhóm ước tính và lập kế hoạch cho dự án cũng như thiết kế, xây dựng và kiểm tra sản phẩm. BA còn là người lãnh đạo và là người giao tiếp, biến những điều khách hàng còn mơ hồ thành các thông số kỹ thuật rõ ràng nhằm hướng dẫn công việc của nhóm phần mềm.

Xác định các yêu cầu kinh doanh: Người BA xác định các yêu cầu kinh doanh của dự án và có thể đề xuất mẫu cho tài liệu tổng quan và làm việc với những người có tầm nhìn để giúp họ thể hiện nó một cách rõ ràng.

Lập kế hoạch tiếp cận: yêu cầu Nhà phân tích nên phát triển các kế hoạch để suy ra, phân tích, ghi lại, xác nhận và quản lý các yêu cầu trong suốt dự án. Làm việc chặt chẽ

với người quản lý dự án để đảm bảo các kế hoạch này phù hợp với kế hoạch tổng thể của dự án và sẽ giúp đạt được các mục tiêu của dự án.

Xác định các bên liên quan của dự án và các lớp người dùng, làm việc với các nhà tài trợ kinh doanh để lựa chọn đại diện phù hợp cho từng loại người dùng , tranh thủ sự tham gia của họ và thương lượng trách nhiệm của họ. Giải thích những gì mong muốn từ cộng tác viên của khách hàng và thống nhất về mức độ tham gia phù hợp của mỗi người.

Gợi ý các yêu cầu: Một nhà phân tích chủ động giúp người dùng trình bày rõ ràng các khả năng của hệ thống mà họ cần để đáp ứng các mục tiêu kinh doanh của họ bằng cách sử dụng nhiều kỹ thuật thu thập thông tin

Phân tích yêu cầu : Tìm kiếm các yêu cầu phái sinh là hệ quả logic của những gì khách hàng yêu cầu và những yêu cầu tiềm ẩn mà khách hàng dường như mong đợi mà chưa được nói ra. Sử dụng các mô hình yêu cầu để nhận ra các mẫu, xác định các lỗ hổng trong các yêu cầu, phát hiện các yêu cầu xung đột và xác nhận rằng tất cả các yêu cầu được chỉ định đều nằm trong phạm vi. Làm việc với các bên liên quan để xác định mức độ chi tiết cần thiết để xác định người dùng và chức năng yêu cầu.

Yêu cầu về tài liệu: Nhà phân tích có trách nhiệm ghi lại các yêu cầu một cách logic chặt chẽ, mô tả rõ ràng giải pháp sẽ giải quyết các vấn đề của khách hàng.

Truyền đạt các yêu cầu: BA phải truyền đạt các yêu cầu một cách hiệu quả và hiệu quả cho tất cả các bên. BA nên xác định khi nào việc thể hiện các yêu cầu bằng cách sử dụng các phương pháp là hữu ích ngoài văn bản, bao gồm nhiều loại mô hình phân tích trực quan khác nhau. Giao tiếp không chỉ đơn giản là việc đưa ra những yêu cầu, nó liên quan đến sự hợp tác liên tục với nhóm để đảm bảo rằng họ hiểu được thông tin được truyền đạt.

Xác thực các yêu cầu: chính BA phải đảm bảo rằng các tuyên bố yêu cầu có những đặc điểm mong muốn và rằng một giải pháp dựa trên các yêu cầu sẽ đáp ứng được nhu cầu của các bên liên quan. Các nhà phân tích là người tham gia chính để đánh giá các yêu cầu. BA cũng nên xem lại các thiết kế và thử nghiệm được rút ra từ các yêu cầu để đảm bảo rằng các yêu cầu được giải thích một cách chính xác.

Tạo điều kiện thuận lợi cho các yêu cầu ưu tiên: Sự hợp tác và đàm phán giữa các nhà môi giới phân tích các bên liên quan khác nhau và các nhà phát triển để đảm bảo

rằng họ đưa ra các quyết định ưu tiên hợp lý và phù hợp với việc đạt được mục tiêu kinh doanh.

Quản lý yêu cầu: Một nhà phân tích kinh doanh tham gia vào toàn bộ quá trình phát triển phần mềm, vì vậy BA nên giúp tạo, xem xét và thực hiện quản lý yêu cầu của kế hoạch dự án. Sau khi thiết lập cơ sở yêu cầu cho một lần phát hành hoặc lặp lại quá trình phát triển sản phẩm nhất định, trọng tâm của BA chuyển sang theo dõi trạng thái của các yêu cầu đó, xác minh sự hài lòng của khách hàng trong sản phẩm và quản lý các thay đổi đối với yêu cầu.

3.2. Skills

Công việc bao gồm nhiều “**kỹ năng mềm**” hướng tới con người hơn là kỹ thuật. Nhà phân tích cần biết cách sử dụng nhiều kỹ thuật gợi ý khác nhau và cách trình bày thông tin dưới các hình thức khác nhau. Một BA làm việc hiệu quả khi kết hợp khả năng giao tiếp, những điều kiện thuận lợi và kiến thức về lĩnh vực kỹ thuật và kinh doanh cũng như tính cách phù hợp với công việc. Sự kiên nhẫn và mong muốn thực sự được làm việc với mọi người là những yếu tố thành công then chốt.

Kỹ năng nghe : Để thành thạo giao tiếp hai chiều, BA phải học cách lắng nghe hiệu quả. Lắng nghe tích cực bao gồm việc loại bỏ sự xao lãng, duy trì tư thế chăm chú và giao tiếp bằng mắt, và trình bày lại những điểm chính để xác nhận sự hiểu biết về vấn đề. BA cần nắm bắt những gì mọi người đang nói và cũng để có thể phát hiện những gì họ còn do dự khi nói. Tìm hiểu tính cách của khách hàng và tránh áp đặt suy nghĩ cá nhân lên những gì nghe được.

Kỹ năng phỏng vấn và đặt câu hỏi: Hầu hết các yêu cầu đầu vào đều được đưa ra thông qua thảo luận, vì vậy BA phải có khả năng tương tác với các cá nhân và nhóm khác nhau về nhu cầu của họ. Nó có thể không thoải mái khi làm việc với các nhà quản lý cấp cao và với những cá nhân có thái độ cố chấp hoặc hung hăng. BA cần đặt những câu hỏi phù hợp để tìm ra thông tin yêu cầu thiết yếu.

Với kinh nghiệm, bạn sẽ trở nên thành thạo trong nghệ thuật đặt câu hỏi tiết lộ và làm rõ những điều không chắc chắn, những bất đồng, giả định và những kỳ vọng không được nói ra (Gause và Weinberg 1989).

Suy nghĩ đa phương diện: Các nhà phân tích kinh doanh luôn cần phải nhận thức được các thông tin hiện có và để xử lý thông tin mới chống lại nó. Họ cần phát hiện ra những mâu thuẫn, sự không chắc chắn, mơ hồ và giả định để họ có thể thảo luận chúng vào

thời điểm thích hợp. BA có thể viết kịch bản bộ câu hỏi phỏng vấn; tuy nhiên, sẽ luôn cần hỏi những điều mà nhà phân tích kinh doanh không thể đoán trước được. BA cần soạn thảo những câu hỏi hay, lắng nghe rõ ràng câu trả lời và nhanh chóng đưa ra câu tiếp theo.

Kỹ năng phân tích: Một nhà phân tích kinh doanh hiệu quả có thể suy nghĩ ở cả mức độ trừu tượng cao và thấp và biết khi nào nên di chuyển từ nơi này sang nơi khác. Đôi khi, phải đi sâu vào từ cấp độ cao và chi tiết. Trong các tình huống khác, BA sẽ cần khai quát hóa từ một nhu cầu cụ thể từ một người dùng được mô tả theo một tập hợp các yêu cầu để đáp ứng được nhiều bên liên quan. BA cần phải hiểu thông tin phức tạp đến từ nhiều nguồn và giải quyết các vấn đề khó khăn liên quan đến từ thông tin đó. Họ cần đánh giá nghiêm túc thông tin để dung hòa xung đột, tách biệt người dùng “mong muốn” với nhu cầu thực sự cơ bản và phân biệt ý tưởng giải pháp với yêu cầu.

Kỹ năng tư duy hệ thống: Mặc dù một nhà phân tích kinh doanh phải có định hướng chi tiết, nhưng cũng phải nhìn thấy tổng quan bức tranh lớn. BA phải kiểm tra các yêu cầu dựa trên những hiểu biết về toàn bộ doanh nghiệp, môi trường kinh doanh và ứng dụng để tìm kiếm sự không nhất quán và tác động. BA cần phải hiểu sự tương tác và mối quan hệ giữa con người, quy trình và công nghệ liên quan. Nếu khách hàng yêu cầu một yêu cầu về khu vực chức năng của mình, BA cần đánh giá liệu yêu cầu có ảnh hưởng đến các phần khác của hệ thống theo những cách không rõ ràng hay không.

Kỹ năng học tập: Các nhà phân tích phải học tập và cập nhật kiến thức mới một cách nhanh chóng, cho dù đó là về các yêu cầu mới, phương pháp tiếp cận hoặc miền ứng dụng. Họ cần có khả năng áp dụng kiến thức đó vào thực tế một cách hiệu quả. Các nhà phân tích phải là những độc giả hiệu quả và có óc phê phán vì họ phải trải qua rất nhiều tình huống và phải nắm bắt được bản chất một cách nhanh chóng. Nếu không phải là chuyên gia trong lĩnh vực đó, đừng ngần ngại đặt câu hỏi làm rõ. *Hãy trung thực về những gì không biết. Không sao cả khi không biết, nhưng không thể che giấu sự thiếu hiểu biết của mình.*

Kỹ năng điều phối: Khả năng tạo điều kiện cho các cuộc thảo luận về yêu cầu và hội thảo khơi gợi là một khả năng phân tích quan trọng. Tạo điều kiện là hành động dẫn dắt một nhóm hướng tới thành công. Tạo thuận lợi là cần thiết khi hợp tác xác định các yêu cầu, ưu tiên các nhu cầu và giải quyết xung đột. Một người điều phối trung lập có kỹ năng đặt câu hỏi, quan sát và điều phối giỏi có thể giúp nhóm xây dựng niềm tin và cải thiện mối quan hệ đôi khi căng thẳng giữa doanh nghiệp và nhân viên CNTT.

Kỹ năng lãnh đạo: Một nhà phân tích giỏi có thể tác động đến một nhóm các bên liên quan để họ đi theo một hướng nhất định, thực hiện mục tiêu chung. Lãnh đạo đòi hỏi phải hiểu biết nhiều kỹ thuật khác nhau để đàm phán, thỏa thuận giữa các bên liên quan của dự án, giải quyết xung đột và đưa ra quyết định. Nhà phân tích nên tạo ra một môi trường hợp tác, nuôi dưỡng niềm tin giữa các nhóm bên liên quan khác nhau, gồm những người có thể không hiểu đồng cơ, nhu cầu và ràng buộc của nhau.

Kỹ năng quan sát: Một nhà phân tích có óc quan sát sẽ phát hiện những nhận xét được đưa ra thoáng qua có thể khiến vấn đề trở nên đáng kể. Bằng cách quan sát BA thực hiện công việc của mình, một người quan sát giỏi có thể phát hiện ra những điều tinh tế mà người khác có thể không nghĩ tới. Kỹ năng quan sát mạnh mẽ đòi hỏi đưa ra các vấn đề mới để thảo luận khơi gợi, từ đó bộc lộ các yêu cầu bổ sung.

Kỹ năng giao tiếp: Sản phẩm chính có thể chuyển giao từ quá trình phát triển yêu cầu là một tập hợp các yêu cầu bằng văn bản để truyền đạt thông tin một cách hiệu quả giữa các khách hàng, tiếp thị, quản lý, nhân viên kỹ thuật. Nhà phân tích cần có trình độ ngôn ngữ vững chắc và khả năng để diễn đạt những ý tưởng phức tạp một cách rõ ràng, cả bằng văn bản và bằng lời nói. Bạn phải có khả năng viết cho nhiều đối tượng, bao gồm cả những khách hàng phải xác thực các yêu cầu và những nhà phát triển cần có yêu cầu rõ ràng, chính xác để thực hiện. BA cần ăn nói rõ ràng, thích nghi với các thuật ngữ và sự khác biệt trong phương ngữ.

Kỹ năng tổ chức thiết lập kiến trúc thông tin : Ngoài ra, BA phải có khả năng tóm tắt và trình bày thông tin ở mức độ chi tiết mà đối tượng mục tiêu cần. Các BA phải đổi mới với một lượng lớn thông tin lộn xộn được thu thập trong quá trình suy diễn và phân tích. Đổi phó với thông tin thay đổi nhanh chóng và cấu trúc tất cả thành một tổng thể mạch lạc đòi hỏi những kỹ năng tổ chức đặc biệt cũng như sự kiên nhẫn và bền bỉ. Là một nhà phân tích, BA cần có kỹ năng tổ chức thiết lập kiến trúc thông tin để hỗ trợ thông tin dự án khi nó phát triển trong suốt dự án (Beatty và Chen 2012).

Kỹ năng lập mô hình: Các mô hình bao gồm từ biểu đồ luồng công việc truyền thông cho đến các mô hình phân tích có cấu trúc (đồ thị luồng dữ liệu, đồ thị quan hệ thực thể và các biểu đồ tương tự). sang Ngôn ngữ mô hình hóa thống nhất (UML) phải là một phần trong vốn liếng của mỗi nhà phân tích (Beatty và Chen 2012). BA sẽ cần biết khi nào nên chọn các mô hình cụ thể dựa trên cách chúng tăng thêm giá trị. Ngoài ra, BA cần hướng dẫn các bên liên quan khác về giá trị của việc sử dụng các mô hình này và cách đọc chúng.

Kỹ năng giao tiếp và tương tác cá nhân: Người phân tích phải có khả năng đưa những người có lợi ích cạnh tranh làm việc cùng nhau như một đội. Một người phân tích nên cảm thấy thoải mái khi nói chuyện với những người thuộc các chức năng công việc khác nhau và ở mọi cấp bậc trong tổ chức. Một người phân tích nên sử dụng ngôn ngữ phù hợp với đối tượng người nghe, không sử dụng thuật ngữ kỹ thuật với các bên liên quan kinh doanh. BA có thể cần làm việc với các nhóm ảo có thành viên phân tán về địa lý, múi giờ, văn hóa hoặc ngôn ngữ bản địa. Một người phân tích nên dễ giao tiếp và rõ ràng và nhất quán trong việc giao tiếp với các thành viên trong đội.

Khả năng sáng tạo: BA không chỉ đơn thuần là người ghi chép những gì khách hàng nói. Các nhà phân tích giỏi nhất phát hiện ra các yêu cầu tiềm năng của khách hàng. Họ nghĩ ra sự đổi mới của sản phẩm, tưởng tượng ra các thị trường và cơ hội kinh doanh mới và nghĩ ra cách để gây bất ngờ và làm hài lòng khách hàng của họ. Một BA thực sự có giá trị sẽ tìm ra những cách sáng tạo để đáp ứng những nhu cầu mà người khác không làm được. Tuy nhiên, các nhà phân tích phải cẩn thận để tránh giải pháp mạ vàng; đừng chỉ thêm các yêu cầu mới vào đặc tả mà không cần sự chấp thuận của khách hàng.

3.3. Kiến thức phân tích cần thiết: Knowledge

Ngoài những khả năng cụ thể và đặc điểm cá nhân, người phân tích kinh doanh cần có kiến thức đa dạng, một phần lớn được đạt được thông qua kinh nghiệm. Họ cần hiểu về các phương pháp kỹ thuật yêu cầu hiện đại và cách áp dụng chúng trong ngữ cảnh của các vòng đời phát triển phần mềm khác nhau. Họ có thể cần giáo dục và thuyết phục những người không quen thuộc với các phương pháp yêu cầu đã được thiết lập. Người phân tích hiệu quả có một bộ công cụ phong phú các kỹ thuật và biết khi nào - và khi không - sử dụng mỗi kỹ thuật.

Người phân tích kinh doanh cần liên kết các hoạt động phát triển và quản lý yêu cầu trong suốt quá trình dự án. Một người phân tích hiểu biết về quản lý dự án, vòng đời phát triển, quản lý rủi ro và kỹ thuật chất lượng có thể giúp ngăn chặn các vấn đề liên quan đến yêu cầu làm đổ dự án. Trong một môi trường phát triển thương mại, người phân tích sẽ có lợi từ kiến thức về các khái niệm quản lý sản phẩm. Người phân tích có lợi từ một mức độ kiến thức cơ bản về kiến trúc và môi trường hoạt động, để có thể tham gia vào các cuộc trò chuyện kỹ thuật về ưu tiên và yêu cầu phi chức năng.

Kiến thức về doanh nghiệp, ngành công nghiệp và tổ chức là tài sản mạnh mẽ cho một người phân tích kinh doanh hiệu quả. Người phân tích có hiểu biết về kinh doanh có thể giảm thiểu các sự không hiểu với người dùng. Những người phân tích hiểu về tổ

chức và lĩnh vực kinh doanh thường phát hiện ra các giả định chưa được nêu rõ và yêu cầu ngầm. Họ có thể đề xuất các cách người dùng có thể cải thiện quy trình kinh doanh hoặc đề xuất các chức năng có giá trị mà không có bất kỳ bên liên quan nào nghĩ tới. Hiểu biết về lĩnh vực ngành công nghiệp có thể hữu ích đặc biệt trong một môi trường thương mại để người phân tích có thể cung cấp phân tích thị trường và sản phẩm cạnh tranh.

3.4. Để trở thành BA

Những người phân tích kinh doanh xuất sắc được phát triển từ các nền tảng giáo dục và kinh nghiệm làm việc đa dạng, do đó họ có thể có những khoảng trống trong kiến thức và kỹ năng của mình. Viện Quốc tế về Phân tích Kinh doanh (IIBA) mô tả các năng lực mà những người phân tích kinh doanh cấp nhập môn, cấp trung, và cấp cao nên thể hiện trong các hoạt động phổ biến của ngành BA (IIBA 2011). Tất cả những người mới làm ngành BA sẽ được hưởng lợi từ việc được hướng dẫn và đào tạo từ những người có kinh nghiệm hơn, có thể thông qua hình thức thực tập. Khám phá cách mà những người có nền tảng khác nhau có thể chuyển sang vai trò người phân tích và xem một số thách thức và rủi ro mà họ sẽ đối mặt.

Người dùng trước đây

Các phòng ban công nghệ thông tin của các công ty thường có những người phân tích kinh doanh đã chuyển sang vai trò đó sau khi làm việc phía kinh doanh như là người sử dụng hệ thống thông tin. Những cá nhân này hiểu về doanh nghiệp và môi trường làm việc, do đó họ dễ dàng có được sự tin tưởng của đồng nghiệp cũ. Họ nói chuyện theo ngôn ngữ của người dùng và hiểu về hệ thống hiện tại và quy trình kinh doanh.

Tuy nhiên, điều tiêu cực là những người dùng trước đây giờ đây làm ngành BA có thể không biết nhiều về kỹ thuật phần mềm hoặc cách giao tiếp với những người kỹ thuật. Nếu họ không quen thuộc với các kỹ thuật mô hình hóa, họ sẽ diễn đạt tất cả thông tin dưới dạng văn bản. Những người dùng trở thành BA cần phải tìm hiểu thêm về phía kỹ thuật trong phát triển phần mềm để có thể đại diện cho thông tin theo hình thức phù hợp nhất với nhiều đối tượng người nghe.

Một số người dùng trước đây tin rằng họ hiểu về những gì cần thiết tốt hơn so với người dùng hiện tại, do đó họ không tìm kiếm hoặc tôn trọng ý kiến đóng góp từ những người sẽ thực sự sử dụng hệ thống mới. Người dùng gần đây có thể bị mắc kẹt trong cách làm việc hiện tại, đến mức họ không nhìn thấy cơ hội để cải thiện quy trình kinh doanh với

sự trợ giúp của hệ thống thông tin mới. Đôi với một người dùng trước đây, dễ dàng chỉ tập trung vào giao diện người dùng. Tập trung vào ý tưởng giải pháp có thể áp đặt các ràng buộc thiết kế không cần thiết và thường không giải quyết được vấn đề thực sự.

Nhà phát triển hoặc người kiểm thử trước đây

Các quản lý dự án thiếu một người phân tích kinh doanh cố định thường kỳ vọng một nhà phát triển thực hiện công việc đó. Những kỹ năng và tính cách cần thiết cho việc phát triển yêu cầu không giống như những kỹ năng cần thiết cho phát triển phần mềm. Một số nhà phát triển có ít kiên nhẫn với người dùng, thích làm việc với mã nguồn và khăng định sự hấp dẫn của công nghệ. Tuy nhiên, nhiều nhà phát triển khác nhận ra tính quan trọng của quá trình xác định yêu cầu và có thể làm việc như những người phân tích khi cần thiết. Những người thích cộng tác với khách hàng để hiểu nhu cầu thúc đẩy phát triển phần mềm là những ứng viên tốt để chuyên môn hóa trong phân tích kinh doanh.

Nhà phát triển chuyển sang vai trò người phân tích có thể cần tìm hiểu thêm về lĩnh vực kinh doanh. Nhà phát triển dễ dàng rơi vào suy nghĩ kỹ thuật và ngôn ngữ chuyên môn, tập trung vào việc xây dựng phần mềm thay vì nhu cầu của khách hàng. Họ sẽ cần nắm bắt những phương pháp tốt nhất hiện tại trong kỹ thuật xác định yêu cầu. Nhà phát triển sẽ được hưởng lợi từ việc đào tạo và hướng dẫn trong các kỹ năng mềm đa dạng mà những người phân tích giỏi nhất đã nắm vững.

Người kiểm thử không thường được yêu cầu đảm nhận vai trò người phân tích. Tuy nhiên, người kiểm thử thường có tư duy phân tích có thể giúp trở thành một người phân tích kinh doanh giỏi. Người kiểm thử đã quen với việc suy nghĩ về các trường hợp ngoại lệ và cách phá vỡ các yêu cầu, đây là một kỹ năng hữu ích để phát hiện những khoảng trống trong yêu cầu. Tương tự như một nhà phát triển trước đây, một người kiểm thử sẽ phải tìm hiểu về các phương pháp kỹ thuật tốt trong kỹ thuật xác định yêu cầu. BA cũng có thể cần nắm vững hơn về lĩnh vực kinh doanh.

Người quản lý dự án trước đây (hoặc đồng thời)

Đôi khi, người quản lý dự án được yêu cầu đảm nhận vai trò của người phân tích kinh doanh, có lẽ vì họ có một số kỹ năng và kiến thức lĩnh vực tương tự cần thiết. Đây có thể là một sự thay đổi vai trò hiệu quả. Người quản lý dự án đã quen với việc làm việc với các nhóm thích hợp, hiểu về tổ chức và lĩnh vực kinh doanh, và thể hiện kỹ năng giao tiếp mạnh mẽ. Họ có thể giỏi trong việc lắng nghe, đàm phán và tạo điều kiện thuận lợi. Họ cũng nên có kỹ năng tổ chức và viết tốt.

Tuy nhiên, người quản lý dự án trước đây sẽ phải tìm hiểu thêm về các phương pháp kỹ thuật tốt trong kỹ thuật xác định yêu cầu. Điều đó khác biệt giữa việc lập kế hoạch, phân bổ tài nguyên và phối hợp các hoạt động của các nhà phân tích và các thành viên khác trong nhóm. Đó là một vấn đề rất khác khi thực hiện vai trò người phân tích kinh doanh một cách độc lập. Người quản lý dự án trước đây phải học cách tập trung vào việc hiểu nhu cầu kinh doanh và ưu tiên những nhu cầu đó trong khuôn khổ dự án hiện tại, thay vì tập trung vào tiến độ, tài nguyên và ràng buộc ngân sách. Họ sẽ cần phát triển kỹ năng phân tích, mô hình hóa và phỏng vấn, những kỹ năng ít quan trọng đối với người quản lý dự án nhưng lại là quan trọng đối với sự thành công của người phân tích kinh doanh.

Chuyên gia về lĩnh vực

Young (2001) đề nghị rằng người phân tích kinh doanh nên là một chuyên gia lĩnh vực ứng dụng hoặc một chuyên gia về lĩnh vực (SME), thay vì chỉ là một người dùng thông thường: "SME có thể xác định, dựa trên kinh nghiệm của họ, liệu các yêu cầu có hợp lý, cách chúng mở rộng hệ thống hiện có, cách kiến trúc đề xuất nên được thiết kế và tác động lên người dùng, trong số những vấn đề khác." Một số tổ chức phát triển sản phẩm thuê người dùng chuyên gia của họ, có kinh nghiệm lĩnh vực rộng, để làm việc trong công ty của họ, có thể làm người phân tích hoặc người đại diện cho người dùng.

Tuy nhiên, cũng có những rủi ro ở đây. Người phân tích kinh doanh là chuyên gia lĩnh vực có thể đặc tả các yêu cầu hệ thống sao cho phù hợp với sở thích của mình, thay vì đáp ứng nhu cầu hợp lệ của các lớp người dùng khác nhau. Anh ta có thể có những hạn chế khi suy nghĩ về yêu cầu và ít sáng tạo trong việc đề xuất ý tưởng mới. Chuyên gia lĩnh vực thường am hiểu về hệ thống "như hiện tại"; họ đôi khi gặp khó khăn trong việc tưởng tượng về hệ thống "như mong muốn". Thường thì việc có một người phân tích kinh doanh từ nhóm phát triển làm việc cùng với chuyên gia lĩnh vực, người sau đó sẽ đóng vai trò là người đại diện người dùng quan trọng hoặc nhà bảo trợ sản phẩm.

Người mới vào nghề

Trở thành một người phân tích kinh doanh là một điểm khởi đầu tốt vào lĩnh vực công nghệ thông tin cho những người mới ra trường hoặc đến từ một công việc hoàn toàn không liên quan. Những người mới tốt nghiệp sẽ có ít kinh nghiệm hoặc kiến thức liên quan, nếu có thì rất ít. Họ có thể được thuê vào vai trò người phân tích kinh doanh vì họ thể hiện nhiều kỹ năng cần thiết để trở thành một người phân tích giỏi. Một lợi thế khi thuê một người mới vào nghề là họ không có nhiều quan điểm định sẵn về cách thức

công việc phân tích yêu cầu nên diễn ra. Do thiếu kinh nghiệm và kiến thức liên quan, người mới tốt nghiệp sẽ phải học rất nhiều về cách thực hiện các nhiệm vụ người phân tích kinh doanh và những chi tiết phức tạp của các phương pháp. Người mới tốt nghiệp cũng cần tìm hiểu đủ về quy trình phát triển phần mềm để hiểu rõ những thách thức mà các nhà phát triển, những người kiểm thử và các thành viên khác trong nhóm đang đối mặt, từ đó có thể hợp tác hiệu quả với họ. Sự hướng dẫn có thể giảm thiểu sự khó khăn của quá trình học đối với một người phân tích kinh doanh mới và tạo ra những thói quen tốt từ đầu.

Bất kể quá trình học tập và công việc trước đó, một người phân tích kinh doanh sáng tạo có thể áp dụng chúng để nâng cao hiệu suất của mình. Người phân tích cần tích lũy kiến thức và kỹ năng mà anh ta thiếu, xây dựng trên bất kỳ kinh nghiệm nào trong quá khứ và thực hành thực hiện các nhiệm vụ người phân tích kinh doanh để trở nên thành thạo hơn. Tất cả những điều này giúp tạo ra một người phân tích kinh doanh đa năng (Hình 4-2).

4. Tiết trình phát triển phần mềm Agile & Waterfall.

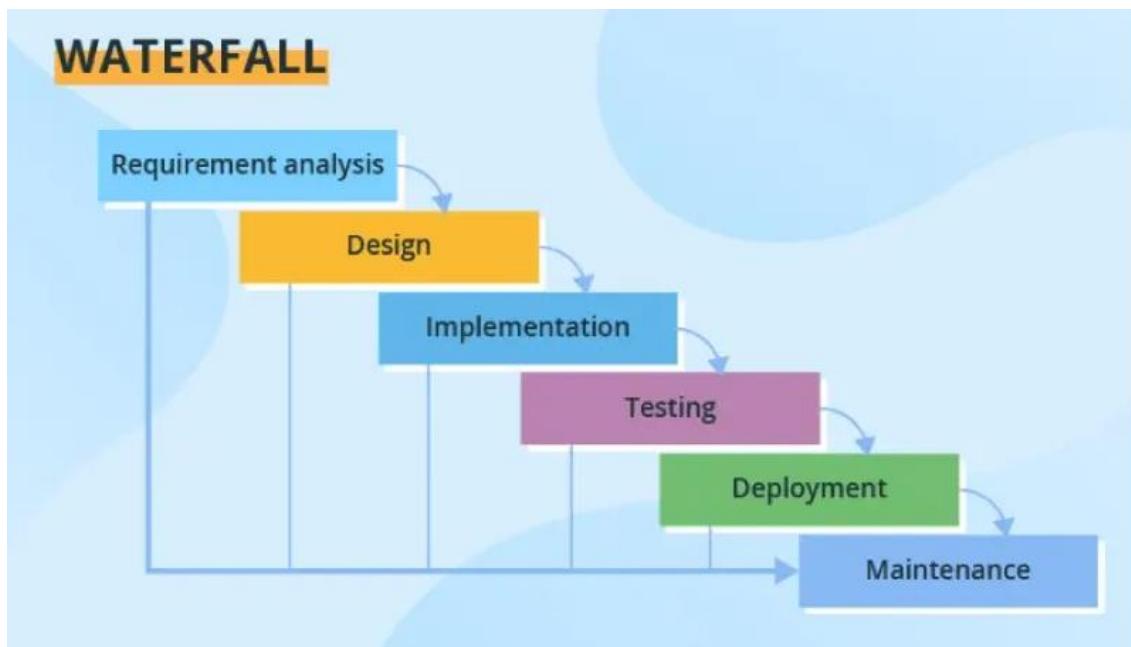
Mô hình thác nước được hiểu đơn giản là một mô hình phát triển theo tính trình tự. Các pha (phase) ở phía trước sẽ được hoàn thiện trước khi bắt đầu ở pha tiếp theo. Mỗi pha lại có một nhiệm vụ riêng biệt. Việc phát triển theo một chu trình tuần tự như vậy giống với dòng chảy của một thác nước và là một trong những chìa khóa để tạo nên sự thành công trong việc phát triển phần mềm. Tuy vậy, waterfall cũng bộc lộ khá nhiều khuyết điểm khi công nghệ ngày càng tiên tiến. Có nhiều phương pháp phát triển khác linh hoạt hơn như Agile Software Development đã được ra đời với mục đích thay thế.

4.1. 6 giai đoạn của mô hình thác nước

Mô hình thác nước được phát triển theo 6 giai đoạn cơ bản. Các bước được thực hiện tuần tự như sau:

- Phân tích yêu cầu
- Thiết kế hệ thống
- Thực hiện
- Kiểm thử hệ thống
- Triển khai hệ thống
- Bảo trì hệ thống

Cụ thể mỗi giai đoạn sẽ thực hiện một số thao tác và nhiệm vụ nhất định. Đặc biệt bước trước sẽ là tiền đề cho bước sau phát triển. Cụ thể:



Phân tích yêu cầu

Pha đầu tiên trong mô hình thác nước là phân tích yêu cầu. Đúng như tên gọi, pha này có nhiệm vụ chính là tìm hiểu và xác định nhu cầu khách hàng đối với sản phẩm đang phát triển. Người nghiên cứu sẽ phải trả lời được các câu hỏi: Đây có phải điều người dùng đang cần? Đâu là các ràng buộc? Rủi ro có thể xảy ra là gì? Họ mong muốn nhận được gì?... Từ đó các kỹ sư IT sẽ xác định được yêu cầu có thực sự phù hợp và có thể kiểm chứng được hay không.

Thiết kế hệ thống

Dựa vào các yêu cầu đã được tìm hiểu, các kỹ sư IT sẽ thiết kế hệ thống phần cứng/phần mềm; ngôn ngữ lập trình và cả lưu trữ dữ liệu. Sau đó ghi chú lại trên từng phần để đảm bảo quy trình thiết kế, kiểm thử được diễn ra thuận lợi. Quá trình thiết kế sẽ không tốn quá nhiều thời gian nhưng cần sự tập trung, tỉ mỉ.

Thực hiện

Đây là một pha quan trọng ảnh hưởng đến chất lượng sản phẩm. Các kỹ sư IT sẽ dựa vào bản thiết kế ở giai đoạn 2 để viết code. Sau đó, tạo ra các chương trình, phần mềm và chuyển qua pha tiếp theo.

Kiểm thử hệ thống

Giai đoạn kiểm thử được thực hiện trong quá trình phát triển phần mềm. Trong đó, các thành viên QA và tester sẽ có nhiệm vụ kiểm tra hoạt động của hệ thống; tìm kiếm lỗi sai và góp ý sửa chữa hoàn thiện chương trình. Quá trình này rất quan trọng và cần được thực hiện tỉ mỉ. Nhất là trước khi đưa sản phẩm đến tay khách hàng.

Các công việc cần thực hiện gồm có:

- Sử dụng unit tested code để đảm bảo các chức năng hoạt động bình thường.
- Thực hiện các thử nghiệm (Functional and non functional) dựa trên kịch bản test để chắc chắn rằng hệ thống đáp ứng được yêu cầu đặt ra.
- Theo dõi và viết báo cáo test.

Triển khai hệ thống

Sau khi đã chắc chắn sản phẩm đáp ứng được yêu cầu kiểm thử thì chúng ta sẽ đến với giai đoạn sử dụng và trải nghiệm. Đây chính là lúc chương trình, phần mềm sẽ thực sự đi vào hoạt động. Các lập trình viên cần đảm bảo môi trường hoạt động bình thường, không có lỗi mờ server, đáp ứng được các tiêu chí test. Sau đó thực hiện kiểm tra về môi trường hoạt động để đảm bảo mọi thứ không xảy ra sai sót.

Bảo trì hệ thống

Đây là giai đoạn cuối nhưng cũng là một phần không thể thiếu trong toàn bộ quy trình dự án. Mặc dù sản phẩm đã được bàn giao cho khách hàng nhưng vẫn có thể xảy ra các lỗi phát sinh không mong muốn. Lúc này, các kỹ sư phần mềm sẽ phải tìm hiểu nguyên nhân và khắc phục để đảm bảo rằng ứng dụng luôn hoạt động một cách tốt nhất. Bên cạnh đó, các bản update cũng được tiến hành thường xuyên để nâng cấp và sửa lỗi. Mục tiêu cuối cùng vẫn là đáp ứng tối đa nhu cầu cũng như mong muốn của người dùng.

4.2. 6 bước triển khai mô hình Agile

Xây dựng kế hoạch dự án

Tương tự như những dự án khác, đầu tiên, nhóm phát triển phải xác định mục tiêu cuối cùng, giá trị mà dự án đem lại cho khách hàng là gì. Sau đó, đội ngũ phải vạch ra những công việc cần thực hiện để đạt được mục tiêu đã đề ra. Cần lưu ý, trong quá trình áp dụng **mô hình Agile**, các công việc thực hiện này hoàn toàn có thể được điều chỉnh để phù hợp hơn với sự thay đổi trong nhu cầu.

Thiết lập lộ trình sản phẩm

Lộ trình sản phẩm được hiểu là những giai đoạn, cột mốc quan trọng trên hành trình tạo ra sản phẩm cuối cùng. Ở giai đoạn này, đội ngũ thực hiện cần xây dựng lộ trình cụ thể, đầy đủ nhất để có thể cho ra sản phẩm cuối cùng hoàn thiện.

Xây dựng kế hoạch phát hành

Một trong những khác biệt của *mô hình Agile* và *Waterfall* truyền thống là mô hình Agile cho phép người thực hiện hoàn thành một tính năng/ công việc cụ thể sau mỗi giai đoạn ngắn. Thay vì phải chờ đợi hàng tháng hay cả năm trời để nhìn được sản phẩm cuối cùng, đội nhóm thực hiện và các bên liên quan có thể mường tượng được thông qua các tính năng được hoàn tất trong từng chu kỳ ngắn.

Vì thế, trước khi bắt đầu triển khai dự án, hãy xây dựng một kế hoạch cho các bản phát hành tính năng. Từ đó, các bên liên quan có thể dễ dàng truy cập và đánh giá lại kế hoạch phát hành cho mỗi tính năng ấy.

Xây dựng kế hoạch từng sprint

Trước khi mỗi Sprint bắt đầu, những bộ phận có liên quan phải lên kế hoạch Sprint và xác định những công việc gì cần phải hoàn thành. Lưu ý, cần phân chia nhiệm vụ đồng đều giữa những cá nhân trong nhóm để đảm bảo nhiệm vụ được hoàn thành đúng hạn trước thời gian chạy nước rút.

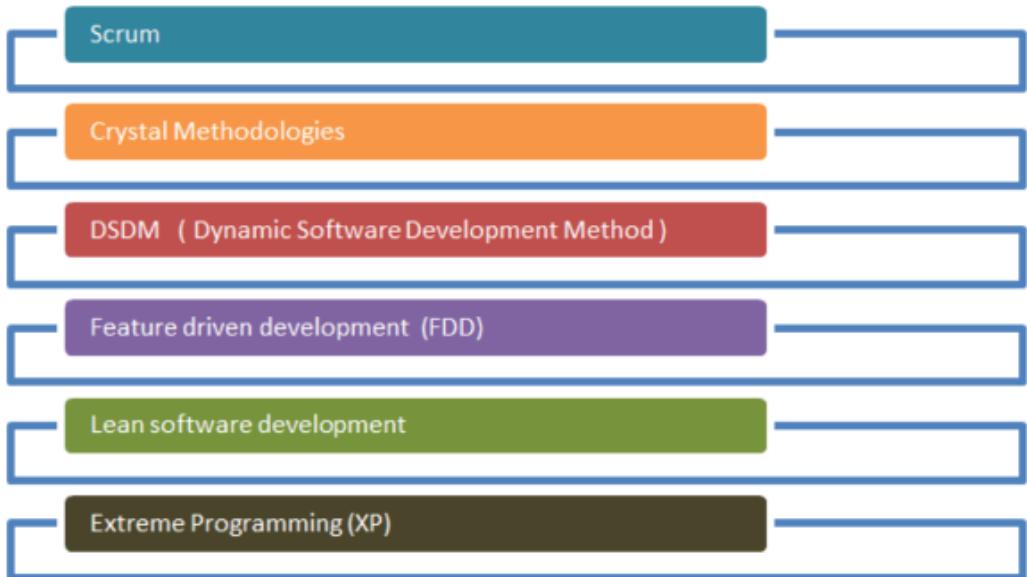
Đánh giá hiệu quả dự án hàng ngày

Vận hành dự án theo mô hình Agile đòi hỏi nhóm thực hiện phải tổ chức những cuộc họp, trao đổi ngắn hàng ngày để đánh giá và cân nhắc công việc. Trong cuộc trao đổi đó, mỗi người sẽ thông tin ngắn gọn về những công việc đang đảm nhận, có gặp khó khăn hay vấn đề gì không.

Đánh giá Sprint

Sau khi kết thúc mỗi Sprint, nhóm thực hiện sẽ tổ chức hai cuộc họp. Đó là cuộc họp với những bộ phận liên quan để nghiệm thu sản phẩm đã hoàn thành, và một cuộc họp trực tiếp để các bên thảo luận về những phát sinh về sản phẩm (nếu có).

Có nhiều phương pháp Agile khác nhau trong kiểm thử nhanh



A. Scrum

SCRUM là một phương pháp phát triển nhanh tập trung đặc biệt vào cách quản lý các nhiệm vụ trong môi trường phát triển dựa trên nhóm. Về cơ bản, Scrum có nguồn gốc từ hoạt động xảy ra trong một trận đấu bóng bầu dục. Scrum tin tưởng vào việc trao quyền cho nhóm phát triển và ủng hộ việc làm việc trong các nhóm nhỏ (ví dụ – 7 đến 9 thành viên). Agile và Scrum bao gồm ba vai trò và trách nhiệm của chúng được giải thích như sau:

Scrum Master

- Scrum Master chịu trách nhiệm thiết lập nhóm, cuộc họp chạy nước rút và loại bỏ các trở ngại đối với tiến độ

Product owner

- The Product Owner tạo product backlog, đánh giá độ ưu tiên của backlog và chịu trách nhiệm cung cấp chức năng ở mỗi lần lặp lại.

Scrum Team

- Nhóm tự quản lý công việc của mình và tổ chức công việc để hoàn thành sprint hoặc cycle

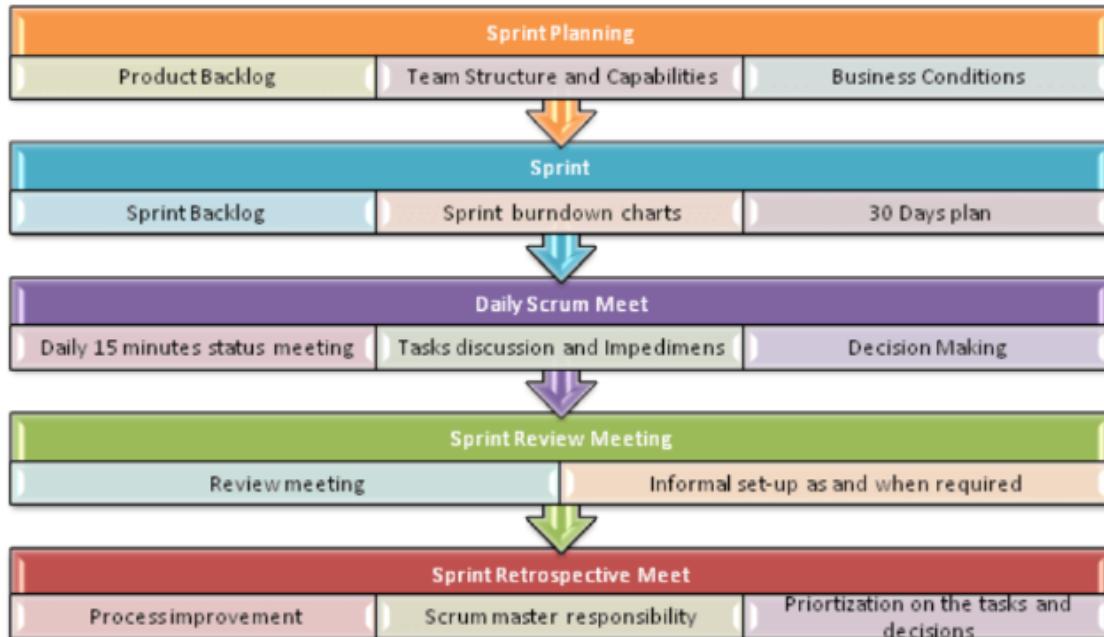
Product Backlog

Đây là một kho lưu trữ mà các yêu cầu được theo dõi với các chi tiết về không có yêu cầu (câu chuyện người dùng) được hoàn thành cho mỗi bản phát hành. Nó nên được

Product Owner duy trì và ưu tiên, và nó nên được phân phối cho nhóm scrum. Nhóm cũng có thể yêu cầu bổ sung hoặc sửa đổi hoặc xóa yêu cầu mới

Scrum Practices

Các thực hành được mô tả chi tiết:



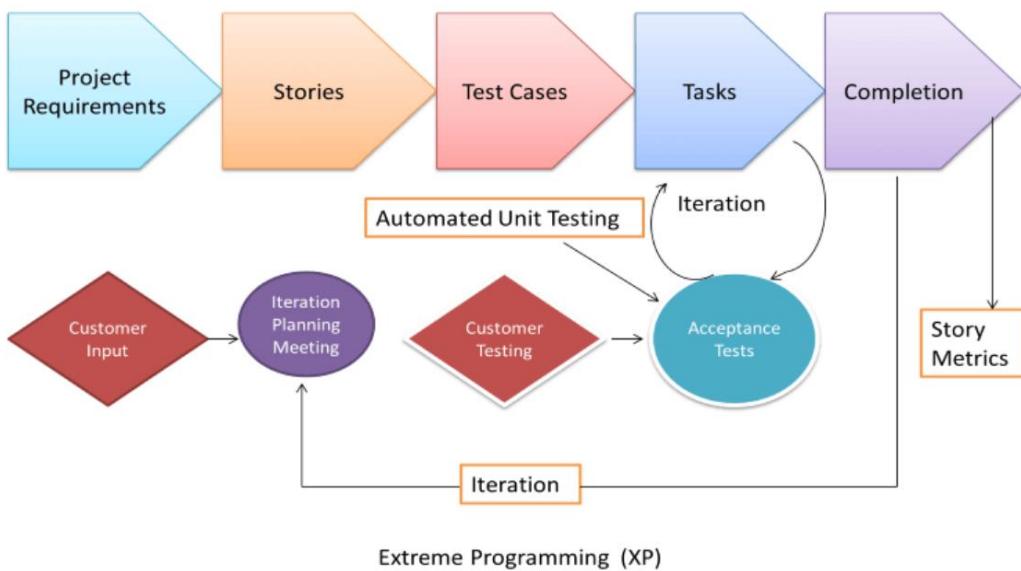
Quy trình kiểm tra scrum như sau:

- Mỗi lần lặp lại của một scrum được gọi là Sprint
- Product backlog là một danh sách nơi tất cả các chi tiết được nhập để có được sản phẩm cuối cùng
- Trong mỗi Sprint, các câu chuyện người dùng hàng đầu về Product backlog được chọn và chuyển thành Sprint backlog
- Nhóm làm việc trên sprint backlog đã xác định
- Nhóm kiểm tra công việc hàng ngày
- Vào cuối sprint, nhóm cung cấp chức năng sản phẩm

B. Extreme Programming (XP)

Kỹ thuật **Extreme Programming** rất hữu ích khi có nhu cầu hoặc yêu cầu thay đổi liên tục từ khách hàng hoặc khi họ không chắc chắn về chức năng của hệ thống. Nó ủng hộ việc “phát hành” sản phẩm thường xuyên trong các chu kỳ phát triển ngắn, điều này vốn giúp cải thiện năng suất của hệ thống và cũng giới thiệu một điểm kiểm tra nơi có thể

dễ dàng thực hiện bất kỳ yêu cầu nào của khách hàng. XP phát triển phần mềm giữ khách hàng trong tầm ngắm.



Các yêu cầu nghiệp vụ được tập hợp dưới dạng các câu chuyện. Tất cả những câu chuyện đó được lưu trữ ở một nơi gọi là parking lot.

Trong loại phương pháp luận này, các bản phát hành dựa trên các chu kỳ ngắn hơn được gọi là Lặp lại với khoảng thời gian 14 ngày. Mỗi lần lặp lại bao gồm các giai đoạn như mã hóa, thử nghiệm đơn vị và thử nghiệm hệ thống, trong đó ở mỗi giai đoạn, một số chức năng nhỏ hoặc chính sẽ được xây dựng trong ứng dụng.

Các giai đoạn của lập trình eXtreme:

Có 6 giai đoạn có sẵn trong phương pháp Agile XP và chúng được giải thích như sau:

Planning

- Identification of stakeholders and sponsors
- Infrastructure Requirements
- Security related information and gathering
- Service Level Agreements and its conditions

Analysis

- Capturing of Stories in Parking lot
- Prioritize stories in Parking lot
- Scrubbing of stories for estimation

- Define Iteration SPAN(Time)
- Resource planning for both Development and QA teams

Design

- Break down of tasks
- Test Scenario preparation for each task
- Regression Automation Framework

Execution

- Coding
- Unit Testing
- Execution of Manual test scenarios
- Defect Report generation
- Conversion of Manual to Automation regression test cases
- Mid Iteration review
- End of Iteration review

Wrapping

- Small Releases
- Regression Testing
- Demos and reviews
- Develop new stories based on the need
- Process Improvements based on end of iteration review comments

Closure

- Pilot Launch
- Training
- Production Launch
- SLA Guarantee assurance
- Review SOA strategy
- Production Support

Có hai phân cảnh có sẵn để theo dõi công việc hàng ngày và chúng được liệt kê bên dưới để tham khảo.

Story Cardboard

- Đây là một cách truyền thống để thu thập tất cả các câu chuyện trong một bảng dưới dạng ghi chú que để theo dõi các hoạt động XP hàng ngày. Vì hoạt động thủ công này đòi hỏi nhiều nỗ lực và thời gian hơn, nên tốt hơn là chuyển sang hình thức trực tuyến.

Online Storyboard

- Online tool Storyboard có thể được sử dụng để lưu trữ stories. Several teams có thể sử dụng nó cho các mục đích khác nhau.

C. Crystal Methodologies

Phương pháp luận tinh thể dựa trên ba khái niệm

1. **Chartering:** Các hoạt động khác nhau liên quan đến giai đoạn này là tạo nhóm phát triển, thực hiện phân tích tính khả thi sơ bộ, phát triển kế hoạch ban đầu và tinh chỉnh phương pháp phát triển
2. **Cyclic delivery:** Giai đoạn phát triển chính bao gồm hai hoặc nhiều chu kỳ phân phối, trong đó:
 1. Nhóm cập nhật và tinh chỉnh kế hoạch phát hành
 2. Triển khai một tập hợp con các yêu cầu thông qua một hoặc nhiều lần lặp lại tích hợp kiểm tra chương trình
 3. Sản phẩm tích hợp được chuyển đến tay người dùng thực
 4. Rà soát kế hoạch dự án và phương pháp phát triển được thông qua
3. **Wrap Up:** Các hoạt động được thực hiện trong giai đoạn này là triển khai vào môi trường người dùng, đánh giá sau triển khai và phản ánh được thực hiện.

D. Dynamic Software Development Method (DSDM)

DSDM là cách tiếp cận Rapid Application Development (RAD) để phát triển phần mềm và cung cấp một khung phân phối dự án nhanh. Khía cạnh quan trọng của DSDM là người dùng bắt buộc phải tham gia tích cực và các nhóm được trao quyền đưa ra quyết định. Việc phân phối sản phẩm thường xuyên trở thành tâm điểm tích cực với DSDM.

Các kỹ thuật được sử dụng trong DSDM là

1. Time Boxing
2. MoSCoW Rules
3. Prototyping

Dự án DSDM bao gồm 7 giai đoạn:

1. Pre-project
2. Feasibility Study
3. Business Study
4. Functional Model Iteration
5. Design and build Iteration
6. Implementation
7. Post-project

E. Feature Driven Development (FDD)

Phương pháp này tập trung vào các tính năng “thiết kế và xây dựng”. Không giống như các phương pháp Agile khác trong kỹ thuật phần mềm, FDD mô tả các giai đoạn rất cụ thể và ngắn của công việc phải được thực hiện riêng biệt cho từng tính năng. Nó bao gồm domain walkthrough, design inspection, promote to build, code inspection và design. FDD phát triển sản phẩm tuân theo những điều trong mục tiêu

1. Domain object Modeling
2. Development by feature
3. Component/ Class Ownership
4. Feature Teams
5. Inspections
6. Configuration Management
7. Regular Builds
8. Visibility of progress and results

F. Lean Software Development

Lean software development method dựa trên nguyên tắc “Sản xuất đúng lúc”. Nó nhằm mục đích tăng tốc độ phát triển phần mềm và giảm chi phí. Lean development có thể được tóm tắt trong bảy bước.

1. Eliminating Waste
2. Amplifying learning
3. Defer commitment (deciding as late as possible)
4. Early delivery
5. Empowering the team
6. Building Integrity
7. Optimize the whole

G. Kanban

Kanban ban đầu xuất phát từ từ tiếng Nhật có nghĩa là, một thẻ chứa tất cả thông tin cần thiết để thực hiện trên sản phẩm ở mỗi giai đoạn dọc theo con đường hoàn thành của nó. Khung hoặc phương pháp này được áp dụng khá phổ biến trong phương pháp kiểm thử phần mềm, đặc biệt là trong các khái niệm Agile.

Dưới đây là một mô tả về các bước trong tiến trình phát triển phần mềm Kanban:

1. Xác định bảng Kanban: Ban đầu, bạn cần xác định bảng Kanban của mình, gồm các cột đại diện cho các giai đoạn khác nhau trong quy trình phát triển phần mềm của bạn. Các cột thường gồm "To Do" (Cần làm), "Doing" (Đang làm), và "Done" (Đã hoàn thành), nhưng bạn có thể tùy chỉnh chúng để phù hợp với quy trình làm việc của bạn.
2. Tạo ra các thẻ công việc: Mỗi công việc được tạo thành một thẻ và đặt trong cột "To Do" để biểu thị rằng nó cần được thực hiện. Thẻ công việc nên được mô tả rõ ràng về nhiệm vụ, thời gian ước tính và các yêu cầu khác.
3. Di chuyển thẻ công việc qua các cột: Khi một thành viên trong nhóm hoặc bạn chịu trách nhiệm cho một công việc, thẻ công việc sẽ được di chuyển từ cột "To Do" sang cột "Doing". Khi công việc hoàn thành, nó sẽ được di chuyển sang cột "Done". Việc di chuyển thẻ công việc này sẽ giúp mọi người trong nhóm có cái nhìn tổng quát về tiến độ và trạng thái của các công việc.
4. Quản lý công việc đang thực hiện: Trong quá trình thực hiện công việc, các thành viên trong nhóm có thể cập nhật tiến độ, thời gian dự kiến hoàn thành và thêm bất kỳ thông

tin cần thiết nào liên quan đến công việc trên thẻ công việc. Điều này giúp cải thiện khả năng giao tiếp và theo dõi tiến trình công việc.

5. Quản lý luồng công việc: Một nguyên tắc quan trọng của Kanban là giới hạn số lượng công việc đang được thực hiện trong cột "Doing". Điều này nhằm mục đích tránh quá tải và tăng cường hiệu suất làm việc của nhóm. Khi số lượng công việc trong cột "Doing" đạt đến giới hạn, không thêm công việc mới được bắt đầu cho đến khi một công việc hoàn thành và được di chuyển sang cột "Done".
6. Đánh giá và cải tiến: Quá trình Kanban liên tục được cải tiến dựa trên phản hồi và đánh giá. Nhóm cần thường xuyên kiểm tra tiến độ công việc, hiệu suất làm việc và tìm cách để cải thiện quy trình phát triển phần mềm. Các điểm yếu và vấn đề phát sinh được xác định và giải quyết trong quá trình này.

5. BA trong các dự án Agile

5.1. Vai trò của người phân tích trên các dự án Agile

Trên các dự án sử dụng phương pháp phát triển Agile, các chức năng của người phân tích kinh doanh vẫn cần được thực hiện, nhưng người thực hiện chúng có thể không được gọi là BA. Một số phương pháp Agile có một thành viên quan trọng được gọi là chủ sở hữu sản phẩm. Người đảm nhiệm vai trò đó có thể thực hiện một số hoạt động phân tích kinh doanh truyền thống, cung cấp tầm nhìn về sản phẩm, truyền đạt các ràng buộc, xác định ưu tiên cho danh sách công việc còn lại của sản phẩm và đưa ra quyết định cuối cùng về sản phẩm (Cohn 2010).

Các dự án khác duy trì một vai trò người phân tích kinh doanh riêng biệt so với chủ sở hữu sản phẩm. Ngoài ra, các thành viên khác trong nhóm, như các nhà phát triển, thực hiện một phần công việc của người phân tích kinh doanh. Điểm quan trọng là, bất kể phương pháp phát triển dự án, các nhiệm vụ liên quan đến vai trò người phân tích kinh doanh vẫn phải được hoàn thành. Nhóm sẽ có lợi từ việc có các thành viên sở hữu những kỹ năng liên quan đến người phân tích kinh doanh.

Thường, trong một tổ chức chuyển đổi sang phương pháp phát triển Agile, người phân tích kinh doanh thường cảm thấy không chắc chắn về cách thức cống hiến hiệu quả nhất cho dự án. Theo tinh thần của phát triển Agile, người phân tích phải sẵn lòng thoát ra khỏi vai trò định trước của "người phân tích kinh doanh" và đóng vai trò cần thiết để giúp giao hàng một sản phẩm thành công. Ellen Gottesdiener (2009) cung cấp một danh sách chi tiết về cách các hoạt động phân tích kinh doanh truyền thống có thể được điều

chỉnh trong môi trường Agile. Dưới đây là một số gợi ý để người phân tích kinh doanh áp dụng kỹ năng của mình trong dự án Agile:

- Xác định một quy trình yêu cầu linh hoạt, nhẹ nhàng và điều chỉnh nó theo yêu cầu của dự án.
- Đảm bảo tài liệu yêu cầu ở mức phù hợp: không quá ít và không quá nhiều. (Nhiều người phân tích kinh doanh thường có xu hướng ghi chép mọi thứ trong các tài liệu yêu cầu. Một số người tán thành cho rằng các dự án Agile nên có ít hoặc không có tài liệu yêu cầu. Cả hai cực đoan đều không lý tưởng.)
- Giúp xác định phương pháp tốt nhất để tài liệu hóa danh sách công việc còn lại, bao gồm xem liệu việc sử dụng thẻ story hay các công cụ hình thức hơn là phù hợp hơn.
- Áp dụng kỹ năng tạo điều kiện và lãnh đạo để đảm bảo các bên liên quan thường xuyên trao đổi với nhau về các yêu cầu, câu hỏi và quan ngại liên quan.
- Giúp xác nhận rằng nhu cầu của khách hàng được đại diện một cách chính xác trong danh sách công việc còn lại của sản phẩm và tạo điều kiện cho việc ưu tiên danh sách công việc.
- Làm việc với khách hàng khi họ thay đổi ý kiến về yêu cầu và ưu tiên, và giúp ghi lại những thay đổi đó. Làm việc với phần còn lại của nhóm để xác định tác động của các thay đổi lên nội dung vòng lặp và kế hoạch phát hành.

Việc có một vai trò như chủ sở hữu sản phẩm để đại diện cho người dùng trong suốt quá trình phát triển mang lại rất nhiều giá trị. Tuy nhiên, người đảm nhiệm vai trò chủ sở hữu sản phẩm có thể không có đầy đủ kỹ năng phân tích kinh doanh hoặc thời gian để thực hiện tất cả các hoạt động liên quan. Một người phân tích kinh doanh có thể mang lại những khả năng quan trọng đó cho nhóm.

5.2. Tạo ra một nhóm cộng tác

Các dự án phần mềm đôi khi gặp khó khăn trong việc xây dựng mối quan hệ cảng thẳng giữa các nhà phân tích, nhà phát triển, người dùng, quản lý và tiếp thị. Các bên không luôn tin tưởng vào động cơ của nhau hoặc đánh giá đúng nhu cầu và ràng buộc của nhau. Tuy nhiên, thực tế là những người sản xuất và người tiêu dùng của một sản phẩm phần mềm có các mục tiêu chung. Đối với việc phát triển hệ thống thông tin doanh nghiệp, tất cả các bên đều làm việc cho cùng một công ty, vì vậy tất cả đều hướng lợi từ việc cải thiện kết quả kinh doanh của công ty. Đối với các sản phẩm thương mại, khách hàng hài lòng tạo ra doanh thu cho nhà sản xuất và sự hài lòng cho những nhà phát triển.

Người phân tích kinh doanh có trách nhiệm quan trọng trong việc xây dựng một môi quan hệ cộng tác giữa các đại diện người dùng và các bên liên quan khác trong dự án. Một người phân tích hiệu quả đánh giá đúng những thách thức mà các bên liên quan kinh doanh và kỹ thuật đối mặt và luôn thể hiện sự tôn trọng đối với đồng nghiệp của mình. Người phân tích điều hướng các thành viên dự án đến một thỏa thuận yêu cầu dẫn đến kết quả win-win-win như sau:

- Khách hàng hài lòng với sản phẩm.
- Tổ chức phát triển hài lòng với kết quả kinh doanh.
- Tất cả thành viên trong nhóm tự hào về công việc tốt mà họ đã làm trong một dự án đầy thách thức và đáng giá.

6. Mô hình yêu cầu phần mềm: Liệt kê các biểu đồ có thể sử dụng cho mô hình yêu cầu phần mềm

Các nhà phân tích kinh doanh có thể hy vọng tìm ra một kỹ thuật có thể kết hợp mọi thứ lại với nhau thành một tổng thể mô tả các yêu cầu của hệ thống. Nhưng không có sơ đồ nào bao gồm tất cả như vậy. Thực tế, nếu có thể mô hình hóa toàn bộ hệ thống trong một sơ đồ duy nhất thì sơ đồ đó sẽ không thể sử dụng được.

Như một danh sách dài các yêu cầu riêng, mục tiêu ban đầu của phân tích hệ thống có cấu trúc là thay thế đặc tả chức năng cổ điển là sử dụng ngôn ngữ, tường thuật chữ bằng các sơ đồ và ký hiệu mang tính hình thức. Tuy nhiên, kinh nghiệm đã chỉ ra rằng các mô hình phân tích nên tăng cường - thay vì thay thế - một đặc tả yêu cầu được viết bằng ngôn ngữ tự nhiên.

Mô hình yêu cầu phần mềm (Software Requirement Modeling) đề cập đến quá trình tạo ra biểu đồ để tăng cường sự hiểu biết, giao tiếp và tài liệu hóa về những gì hệ thống phần mềm cần thực hiện. Mô hình hóa yêu cầu hiệu quả giúp đảm bảo rằng các dự án phát triển phần mềm được định rõ, được tài liệu hóa đúng cách và đáp ứng nhu cầu của các bên liên quan. Nó cũng hỗ trợ quản lý thay đổi và giao tiếp liên tục trong toàn bộ vòng đời phát triển phần mềm.

Các mô hình yêu cầu phần mềm có thể giúp xác định những yêu cầu còn thiếu, không liên quan và không nhất quán yêu cầu. Với những hạn chế về trí nhớ của con người, việc phân tích danh sách những yêu cầu không nhất quán, trùng lặp, các yêu cầu

không liên quan là gần như không thể thực hiện được. Khó có thể tìm thấy tất cả các lỗi chỉ bằng cách xem lại các yêu cầu về văn bản.

Các ký hiệu được trình bày ở đây cung cấp một ngôn ngữ chung, tiêu chuẩn ngành cho những người tham gia dự án sử dụng. Những mô hình này rất hữu ích cho việc xây dựng và khám phá các yêu cầu cũng như thiết kế các phần mềm. Các sơ đồ này cho phép tạo các biểu diễn khái niệm của hệ thống mới. Chúng mô tả các khía cạnh logic của các thành phần dữ liệu, các đối tượng trong thế giới thực và những thay đổi về trạng thái hệ thống. Bạn có thể căn cứ vào các mô hình theo yêu cầu văn bản để thể hiện chúng từ các góc độ khác nhau hoặc bạn có thể rút ra các yêu cầu chức năng từ các mô hình cấp cao dựa trên đầu vào của người dùng.

Gần như không bao giờ có được một mô hình đúng lần đầu tiên thực hiện, vì vậy việc lặp lại là chìa khóa để lập mô hình thành công. Các công cụ cũng có thể thực thi các quy tắc cho từng phương pháp lập mô hình mà chúng hỗ trợ, xác định các lỗi cú pháp và sự không nhất quán mà mọi người xem lại sơ đồ có thể không nhìn thấy. Các công cụ quản lý yêu cầu hỗ trợ mô hình hóa cho phép bạn theo dõi các yêu cầu đối với mô hình.

6.1. Các biểu đồ có thể sử dụng cho mô hình yêu cầu phần mềm:

- + Biểu đồ đồ luồng dữ liệu (Data Flow Diagram)
- + Biểu đồ quy trình/ biểu đồ đường bơi
- + Biểu đồ trạng thái (State Diagram)
- + Bản đồ hộp thoại
- + Bảng quyết định và cây quyết định
- + Bảng phản hồi sự kiện
- + Biểu đồ đồ Use case
- + Biểu đồ hoạt động
- + Biểu đồ mối quan hệ thực thể (ERD)

6.2. Liên hệ giữa yêu cầu của khách hàng với các thành phần của mô hình phân tích

| Type of word | Examples | Analysis model components |
|--------------|---|---|
| Noun | People, organizations, software systems, data elements, or objects that exist | <ul style="list-style-type: none"> ■ External entities, data stores, or data flow (DFD) ■ Actors (use case diagram) ■ Entities or their attributes (ERD) ■ Lanes (swimlane diagram) ■ Objects with states (STD) |
| Verb | Actions, things a user or system can do, or events that can take place | <ul style="list-style-type: none"> ■ Processes (DFD) ■ Process steps (swimlane diagram) ■ Use cases (use case diagram) ■ Relationships (ERD) ■ Transitions (STD) ■ Activities (activity diagram) ■ Events (event-response table) |
| Conditional | Conditional logic statements, such as if/then | <ul style="list-style-type: none"> ■ Decisions (decision tree, decision table, or activity diagram) ■ Branching (swimlane diagram or activity diagram) |

Với các từ khóa là danh từ, nó có thể đóng vai trò như các thực thể bên ngoài, kho dữ liệu hay luồng dữ liệu trong biểu đồ luồng dữ liệu; các tác nhân (Actors) trong biểu đồ Use case; các thực thể và thuộc tính của chúng trong biểu đồ quan hệ thực thể; các lanes trong biểu đồ đường bơi và các đối tượng với trạng thái của nó trong biểu đồ trạng thái.

Với các từ khóa là động từ, nó có thể đóng vai trò như là các quy trình (DFD); các bước xử lý (biểu đồ đường bơi), ca sử dụng (biểu đồ Use case); các mối quan hệ (ERD), chuyển tiếp (STD); các hoạt động (biểu đồ hoạt động); sự kiện (bảng phản hồi sự kiện).

Với các câu điều kiện ví dụ như nếu thì, nó có thể đóng vai trò quyết định trong cây quyết định, biểu đồ hoạt động, ...

Hiếm khi cần tạo một bộ mô hình phân tích hoàn chỉnh cho toàn bộ hệ thống. Tập trung mô hình hóa trên các phần phức tạp nhất và rủi ro nhất của hệ thống cũng như trên các phần đó hầu hết đều có sự mơ hồ hoặc không chắc chắn. Các phần tử hệ thống là những ứng cử viên tốt cho việc lập mô hình vì tác động của các khuyết điểm trong đó là rất lớn và rất nghiêm trọng. Đồng thời chọn các mô hình để sử dụng cùng nhau nhằm giúp đảm bảo tất cả các mô hình đều hoàn chỉnh. Ví dụ, việc kiểm tra các đối tượng dữ liệu trong DFD có thể phát hiện ra các thực thể bị thiếu trong ERD, ...

6.3. Biểu đồ luồng dữ liệu:

Biểu đồ luồng dữ liệu là công cụ cơ bản của phân tích có cấu trúc (DeMarco 1979; Robertson và Robertson 1994), xác định các quy trình chuyển đổi của một hệ

thống, các bộ sưu tập (lưu trữ) dữ liệu mà hệ thống thao tác và các luồng dữ liệu hoặc vật liệu giữa các quy trình và thế giới bên ngoài. Mô hình hóa luồng dữ liệu áp dụng cách tiếp cận phân rã chức năng để phân tích hệ thống, chia các vấn đề phức tạp thành các mức độ chi tiết tăng dần. Điều này hoạt động tốt cho các hệ thống xử lý giao dịch và các ứng dụng chuyên sâu về chức năng khác. Hiện nay, biểu đồ luồng dữ liệu đã được mở rộng để cho phép mô hình hóa các hệ thống thời gian thực (Hatley, Hruschka, và Pirhai 2000).

Biểu đồ luồng dữ liệu có thể biểu diễn các hệ thống ở nhiều mức độ trừu tượng khác nhau. DFD cấp cao cung cấp cái nhìn toàn diện, toàn cảnh về dữ liệu và các thành phần xử lý trong một hoạt động nhiều bước, bổ sung cái nhìn chính xác, chi tiết thể hiện trong các yêu cầu chức năng.

6.4. Biểu đồ Swimlane:

Biểu đồ Swimlane cung cấp một cách để biểu diễn các bước được thực hiện trong một quy trình kinh doanh hoặc các hoạt động của một hệ thống phần mềm được đề xuất. Biểu đồ Swimlane là một biến thể của biểu đồ luồng, được chia thành các thành phần con gọi là "lanes" (dây bơi). Các lanes có thể đại diện cho các hệ thống hoặc người tham gia khác nhau thực hiện các bước trong quy trình. Biểu đồ Swimlane thường được sử dụng để hiển thị các quy trình kinh doanh, luồng làm việc hoặc tương tác giữa hệ thống và người dùng. Chúng tương tự với biểu đồ hoạt động UML và thường được gọi là biểu đồ chức năng chéo (cross-functional diagrams).

6.5. Biểu đồ trạng thái:

Các hệ thống thời gian thực và các ứng dụng điều khiển quá trình có thể tồn tại ở một trong số ít trạng thái tại bất kỳ thời điểm nào. Sự thay đổi trạng thái chỉ có thể diễn ra khi các tiêu chí được xác định rõ ràng được thỏa mãn, chẳng hạn như nhận được một kích thích đầu vào cụ thể trong những điều kiện nhất định.

Các hệ thống phần mềm liên quan đến sự kết hợp giữa hành vi chức năng, xử lý dữ liệu và thay đổi trạng thái. Các hệ thống thời gian thực và các ứng dụng điều khiển quy trình có thể tồn tại trong một số trạng thái vào bất kỳ thời điểm nào. Sự thay đổi trạng thái chỉ có thể xảy ra khi các tiêu chí được xác định rõ ràng được đáp ứng, chẳng hạn như khi nhận một đầu vào cụ thể trong điều kiện nhất định. Việc mô tả một tập hợp các thay đổi trạng thái phức tạp bằng ngôn ngữ tự nhiên có khả năng cao bỏ sót một sự thay đổi trạng thái hoặc bao gồm một thay đổi bị cấm. Tùy thuộc vào cách mà bản mô

tả yêu cầu phần mềm được tổ chức, các yêu cầu liên quan đến hành vi dựa trên trạng thái có thể được phân tán trong toàn bộ tài liệu mô tả yêu cầu phần mềm.

6.6. Biểu đồ giao diện:

Biểu đồ giao diện biểu diễn một thiết kế giao diện người dùng ở mức độ trừu tượng cao. Nó thể hiện các yếu tố giao diện trong hệ thống và các liên kết điều hướng giữa chúng, nhưng không hiển thị các thiết kế màn hình chi tiết. Một giao diện người dùng có thể được coi là một loạt các thay đổi trạng thái. Chỉ có một yếu tố giao diện (như một menu, không gian làm việc, hộp thoại, lời nhắc dòng, hoặc màn hình cảm ứng) có sẵn tại bất kỳ thời điểm nào để người dùng nhập liệu. Người dùng có thể điều hướng đến một số yếu tố giao diện khác dựa trên hành động mà anh ta thực hiện tại vị trí nhập liệu đang hoạt động. Số lượng các bộ trình điều hướng có thể lớn trong một hệ thống phức tạp, nhưng số lượng đó là hữu hạn và thông thường các tùy chọn đã được biết đến. Một bản đồ giao diện thực sự chỉ là một giao diện người dùng được mô hình hóa dưới dạng biểu đồ chuyển đổi trạng thái.

Một bản đồ giao diện cho phép khám phá các khái niệm giao diện người dùng giả định dựa trên hiểu biết về các yêu cầu. Người dùng và nhà phát triển có thể nghiên cứu bản đồ giao diện để đạt được một tầm nhìn chung về cách người dùng có thể tương tác với hệ thống để thực hiện một nhiệm vụ. Các liên kết điều hướng xây dựng vào trang web sẽ xuất hiện như các chuyển đổi trên bản đồ giao diện.

Biểu đồ giao diện trông hơi giống các biểu đồ nhưng chúng phục vụ mục đích khác nhau. Biểu đồ hiển thị các bước xử lý và điểm quyết định, nhưng không hiển thị giao diện người dùng. Ngược lại, biểu đồ giao diện không hiển thị quá trình xử lý diễn ra dọc theo các đường chuyển tiếp kết nối phần tử hộp thoại này sang phần tử hộp thoại khác.

6.7. Bảng quyết định và cây quyết định:

Một hệ thống phần mềm thường được điều khiển bởi một logic phức tạp, với sự kết hợp của các điều kiện dẫn đến các hành vi khác nhau của hệ thống. Nhà phát triển cần yêu cầu chức năng mô tả hệ thống nên làm gì trong tất cả các kết hợp điều kiện có thể có. Tuy nhiên, rất dễ bỏ sót một điều kiện, dẫn đến việc thiếu yêu cầu. Các khoảng trống này khó nhận biết bằng cách xem xét một mô tả bằng ngôn ngữ tự nhiên như văn bản.

Bảng quyết định và cây quyết định là hai kỹ thuật thay thế để biểu diễn hệ thống nên làm gì khi logic phức tạp và quyết định đặt ra (Beatty và Chen 2012). Một bảng quyết định liệt kê các giá trị khác nhau cho tất cả các yếu tố ảnh hưởng đến hành vi và chỉ ra hành động của hệ thống dự kiến đáp ứng từng kết hợp của các yếu tố. Các yếu tố có thể được hiển thị dưới dạng các câu lệnh với các điều kiện có thể là đúng hoặc sai, là các câu hỏi với các câu trả lời có thể là có hoặc không, hoặc là các câu hỏi với nhiều giá trị có thể.

6.8. Bảng phản ứng sự kiện:

Các use cases và user stories không phải lúc nào cũng hữu ích hoặc đủ để khám phá chức năng mà các nhà phát triển phải triển khai. Một cách tiếp cận khác để định yêu cầu của người dùng là xác định các sự kiện bên ngoài mà hệ thống phải phản ứng. Sự kiện là sự thay đổi hoặc hoạt động nào đó diễn ra trong môi trường của người dùng, kích thích phản ứng từ hệ thống phần mềm (Wiley 2000). Một bảng sự kiện-phản ứng (còn được gọi là bảng sự kiện hoặc danh sách sự kiện) liệt kê tất cả các sự kiện như vậy và hành vi mà hệ thống được kỳ vọng thể hiện trong phản ứng đối với mỗi sự kiện. Có ba loại sự kiện hệ thống:

- **Business event:** một hành động của người dùng con người gây ra một cuộc trò chuyện với phần mềm, như khi người dùng khởi tạo một use case. Các chuỗi sự kiện-phản ứng tương ứng với các bước trong một use case hoặc biểu đồ Swimlane.
- **Signal event:** được đăng ký khi hệ thống nhận được một tín hiệu điều khiển, đọc dữ liệu hoặc ngắt từ một thiết bị phần cứng bên ngoài hoặc một hệ thống phần mềm khác.
- **Temporal event:** sự kiện được kích hoạt bởi thời gian.

6.9. Biểu đồ lớp (Class diagrams):

Được sử dụng để biểu thị các lớp đối tượng liên quan đến lĩnh vực ứng dụng, bao gồm các thuộc tính, hành vi và tính chất của chúng, cũng như các mối quan hệ giữa các lớp. Biểu đồ lớp cũng có thể được sử dụng cho mô hình dữ liệu, nhưng ứng dụng giới hạn này không khai thác hết khả năng ngữ nghĩa của một biểu đồ lớp.

6.10. Biểu đồ Usecase diagrams:

Sử dụng để hiển thị các mối quan hệ giữa các tác nhân bên ngoài hệ thống và các trường hợp sử dụng mà họ tương tác với. Sử dụng để mô hình hóa các tình huống sử dụng cụ thể của hệ thống.

6.11. Biểu đồ hoạt động (Activity diagrams):

Biểu thị cách các luồng khác nhau trong một trường hợp sử dụng xen kẽ, hoặc những vai trò nào thực hiện các hành động cụ thể (như trong biểu đồ Swimlane), hoặc để mô hình hóa luồng của các quy trình kinh doanh.

6.12. Biểu đồ trạng thái (State diagrams):

Sử dụng để biểu diễn các trạng thái khác nhau mà hệ thống hoặc đối tượng dữ liệu có thể thay đổi và các chuyển đổi cho phép giữa các trạng thái này.

7. Các giai đoạn thiết kế phần mềm: Thiết kế kiến trúc, Thiết kế chi tiết

7.1 Khái niệm về thiết kế phần mềm

Thiết kế tạo ra các biểu diễn và dữ kiện của hệ thống phần mềm cần xây dựng từ kết quả phân tích yêu cầu để có thể tiếp tục thực hiện giai đoạn tiếp theo trong tiến trình phát triển phần mềm.

Kết quả của pha thiết kế là mô hình thiết kế của phần mềm:

- Mô hình thiết kế đủ chi tiết để giai đoạn lập trình có thể thực hiện
- Là phương tiện để trao đổi thông tin và đảm bảo chất lượng
- Mô hình thiết kế dễ sửa đổi hơn mã chương trình, cung cấp cái nhìn tổng thể đồng thời có nhiều mức chi tiết.

Các giai đoạn thiết kế: hoạt động thiết kế xuất hiện trong các mô hình phát triển phần mềm khác nhau, gồm hai giai đoạn thiết kế chính:

- Thiết kế kiến trúc / Thiết kế mức cao (high level design) :
 - Mô hình tổng thể của hệ thống
 - Cách thức hệ thống được phân rã thành các mô đun
 - Mối quan hệ giữa các môđun
 - Cách thức trao đổi thông tin giữa các môđun
 - Thực hiện bởi nhiều mức trừu tượng
- Thiết kế chi tiết / Thiết kế mức thấp (low level design): Thiết kế chi tiết lớp, module, thiết kế thuật toán, thiết kế dữ liệu, thiết kế giao diện,...

7.2 Quan hệ giữa thiết kế và chất lượng

- Thiết kế phải thực hiện tất cả các yêu cầu rõ ràng chứa trong mô hình phân tích, và nó phải đáp ứng tất cả các yêu cầu tiềm ẩn khách hàng mong muốn .
- Thiết kế phải là một hướng dẫn dễ hiểu dễ đọc cho những người tạo ra code và cho những người kiểm thử và sau đó hỗ trợ cho phần mềm.
- Thiết kế nên cung cấp một bức tranh hoàn chỉnh của phần mềm, giải quyết vấn đề dữ liệu, chức năng, và hành vi từ một quan điểm thực thi.
- Một thiết kế nên thể hiện một kiến trúc mà (1) đã được tạo ra bằng cách sử dụng phong cách kiến trúc hoặc các pattern được công nhận , (2) bao gồm các thành phần mang những đặc tính thiết kế tốt và (3) có thể được thực hiện một cách tiến hóa
- Đối với các hệ thống nhỏ hơn, thiết kế đôi khi có thể được phát triển tuyếntính.
- Một thiết kế nên môđun hoá: đó là, phần mềm nên được phân chia thành các thành phần hợp lý hoặc hệ thống con
- Một thiết kế cần có biểu diễn riêng biệt của dữ liệu, kiến trúc, giao diện, và các thành phần.

7.3 Nguyên tắc thiết kế

- Quá trình thiết kế không nên mắc phải ‘tunnel vision.’
- Việc thiết kế nên có thể truy ngược về mô hình phân tích.
- Việc thiết kế không nên ‘phát minh lại bánh xe’.
- Việc thiết kế nên “giảm thiểu khoảng cách trí tuệ” [DAV95] giữa phần mềm và bài toán như nó tồn tại trong thế giới thực.
- Việc thiết kế nên biểu lộ tính đồng nhất và tích hợp
- Việc thiết kế nên được cấu trúc để thích ứng với thay đổi.
- Việc thiết kế nên được cấu trúc để làm suy thoái (degrade) nhẹ nhàng, ngay cả khi đang gặp phải dữ liệu bất thường, các sự kiện, hoặc điều kiện hoạt động .
- Thiết kế không phải là coding, coding không phải là thiết kế.
- Việc thiết kế nên được đánh giá về chất lượng khi nó được tạo ra, chứ không phải sau thực tế.

- Việc thiết kế cần được xem xét để giảm thiểu lỗi khái niệm (ngữ nghĩa).

7.4 Tại sao cần thiết kế kiến trúc?

Các kiến trúc không phải là phần mềm hoạt động. Thay vào đó, nó là một đại diện cho phép một kỹ sư phần mềm:

1. Phân tích hiệu quả của thiết kế trong việc đáp ứng các yêu cầu đề ra,
2. Xem xét lựa chọn thay thế kiến trúc khi thay đổi thiết kế vẫn tương đối dễ dàng, và
3. Giảm thiểu rủi ro gắn liền với việc xây dựng các phần mềm.

Đại diện của kiến trúc phần mềm là một tạo khả năng cho truyền thông giữa tất cả các bên (các bên liên quan) quan tâm đến sự phát triển của một hệ thống dựa trên máy tính.

Những kiến trúc làm nổi bật thiết kế quyết định ban đầu mà sẽ có một tác động sâu sắc trên tất cả các công việc kỹ thuật phần mềm sau và, quan trọng hơn, vào sự thành công cuối cùng của hệ thống như là một thực thể hoạt động.

Kiến trúc "tạo thành một mode minh bạch tương đối nhỏ về cách hệ thống được cấu trúc và cách các thành phần của nó làm việc cùng nhau"

7.5 Mô tả kiến trúc

The IEEE Standard định nghĩa một mô tả kiến trúc (AD) là một "một tập hợp các sản phẩm để tài liệu hóa một kiến trúc". Các mô tả chính nó được đại diện bằng cách sử dụng nhiều quan điểm, nơi từng xem là "một đại diện của cả một hệ thống từ quan điểm của một tập hợp có liên quan của [các bên liên quan] các mối quan tâm"

Thể loại kiến trúc:

- Thể loại ngữ ý một phân loại cụ thể trong lĩnh vực phần mềm tổng thể.
- Trong mỗi thể loại, bạn gặp phải một số tiêu phân loại: Ví dụ, trong các thể loại của các tòa nhà, bạn sẽ gặp phải những phong cách chung sau đây: nhà ở, căn hộ, chung cư, cao ốc văn phòng, tòa nhà công nghiệp, nhà kho, vv.
- Trong mỗi phong cách chung, phong cách cụ thể hơn có thể được áp dụng. Mỗi phong cách sẽ có một cấu trúc có thể được mô tả bằng một tập các mô hình dự đoán được.
 - Mỗi phong cách mô tả một loại hệ thống bao gồm:
 - (1) một tập hợp các thành phần (ví dụ, một cơ sở dữ liệu, mô đun tính toán) thực hiện một chức năng cần thiết của một hệ thống,

- (2) một tập hợp các kết nối cho phép "truyền thông, phối hợp và hợp tác" giữa các thành phần,

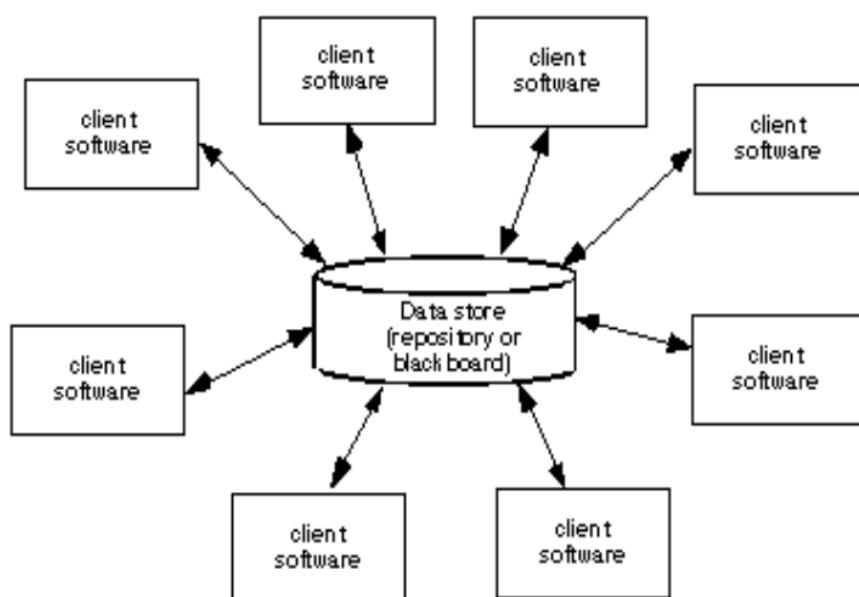
- (3) khó khăn để xác định cách các thành phần có thể được tích hợp để tạo thành hệ thống, và

- (4) các mô hình ngữ nghĩa cho phép một nhà thiết kế phải hiểu được tính chất tổng thể của hệ thống bằng cách phân tích các đặc tính được biết đến của các bộ phận cấu thành của nó.

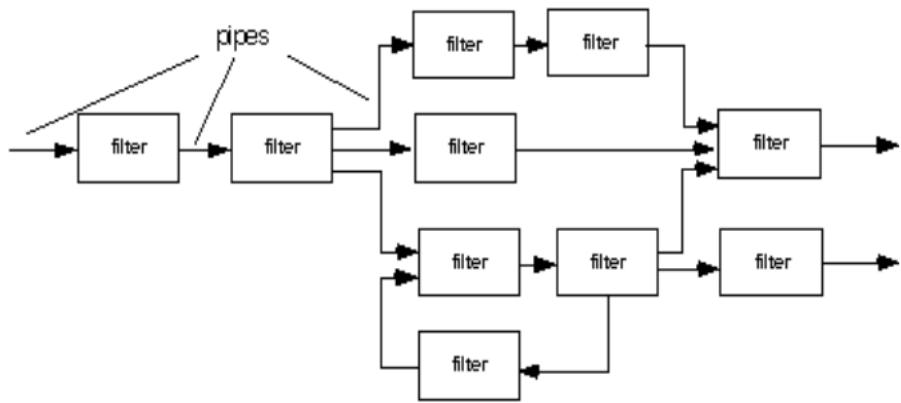
- Một số kiểu kiến trúc hệ thống:

- Kiến trúc lấy dữ liệu làm trung tâm
- Kiến trúc luồng dữ liệu
- Kiến trúc gọi và trả về
- Kiến trúc phân lớp
- Kiến trúc hướng đối tượng
- ...

Kiến trúc lấy dữ liệu làm trung tâm



Kiến trúc luồng dữ liệu

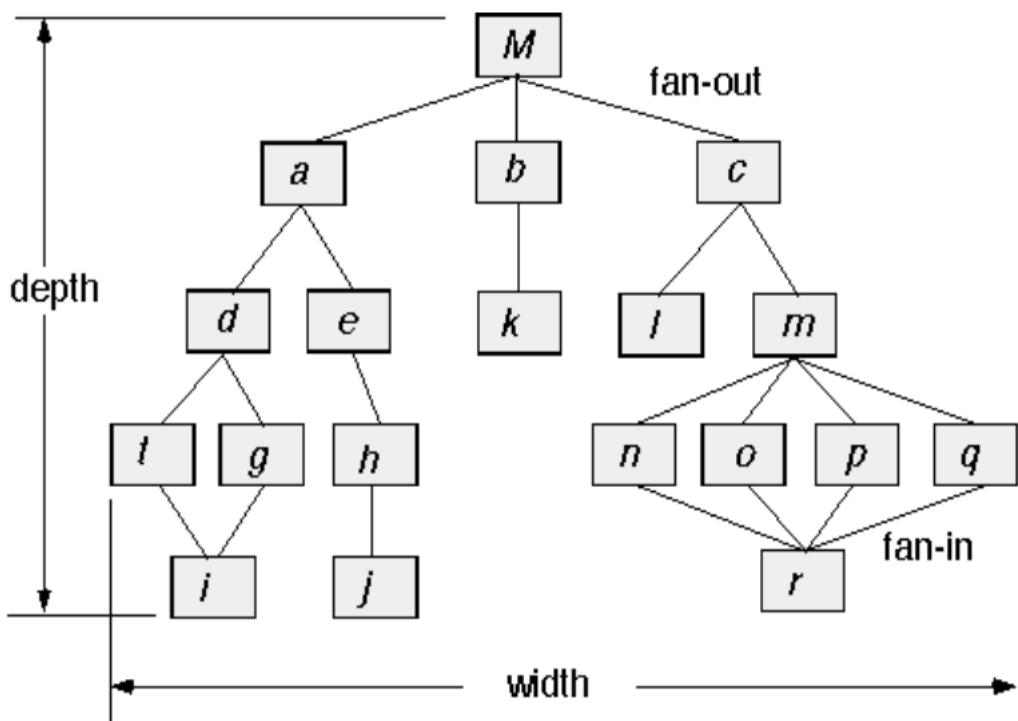


(a) pipes and filters

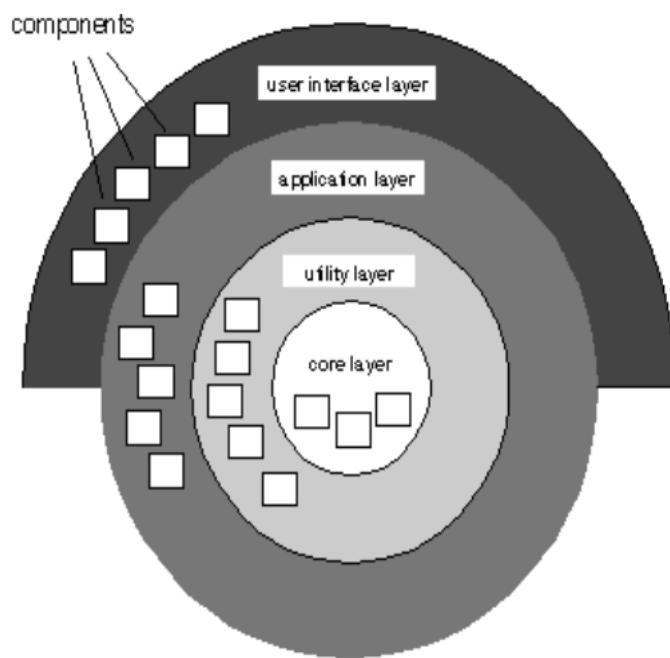


(b) batch sequential

Kiến trúc gọi và trả về



Kiến trúc phân lớp



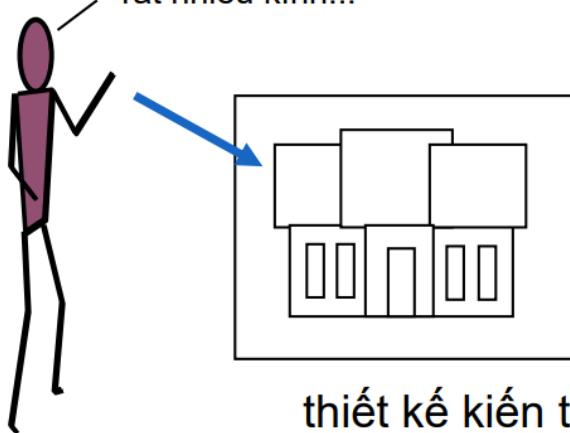
7.6 Thiết kế kiến trúc

- Phần mềm phải được đặt trong bối cảnh: Thiết kế cần xác định các thực thể bên ngoài (các hệ thống khác, thiết bị, con người) mà phần mềm tương tác với và bản chất của sự tương tác
- Một tập hợp các nguyên mẫu kiến trúc cần được xác định: Một nguyên mẫu là một trừu tượng (tương tự như một lớp) đại diện cho một phần tử của hệ thống hành vi
- Các nhà thiết kế xác định cấu trúc của hệ thống bằng cách xác định và tinh chỉnh các thành phần phần mềm thực hiện từng nguyên mẫu.

Ví dụ

yêu cầu khách hàng

"bốn phòng ngủ, ba phòng tắm,
rất nhiều kính..."



thiết kế kiến trúc

Các bước thiết kế kiến trúc

1. Thu thập các kịch bản.
2. Gợi ý các yêu cầu, ràng buộc, và đặc tả môi trường.
3. Mô tả các phong cách / mô hình kiến trúc đã được lựa chọn để giải quyết các tình huống và yêu cầu:
 - quan điểm mô-đun
 - góc nhìn quá trình
 - quan điểm dòng dữ liệu
4. Đánh giá chất lượng thuộc tính bằng cách coi mỗi thuộc tính trong sự cô lập.
5. Xác định mức độ nhạy cảm của các thuộc tính chất lượng với các thuộc tính kiến trúc khác nhau cho một phong cách kiến trúc cụ thể.
6. Kiến trúc ứng cử viên phê bình (được phát triển trong bước 3) bằng cách sử dụng phân tích độ nhạy được thực hiện ở bước 5.

7.7 Thiết kế chi tiết

Thiết kế chi tiết: chỉ ra chi tiết và đầy đủ về thành phần, tạo điều kiện xây dựng phần mềm trong pha sau đó → thiết kế mức thành phần

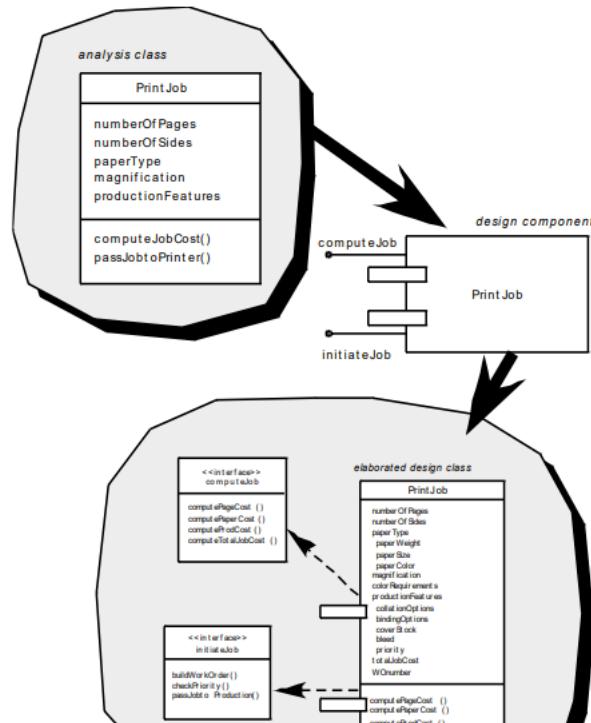
Các hoạt động thiết kế chi tiết:

- Thiết kế chi tiết lớp, module,
- Thiết kế thuật toán,
- Thiết kế dữ liệu,
- Thiết kế giao diện,
- ...

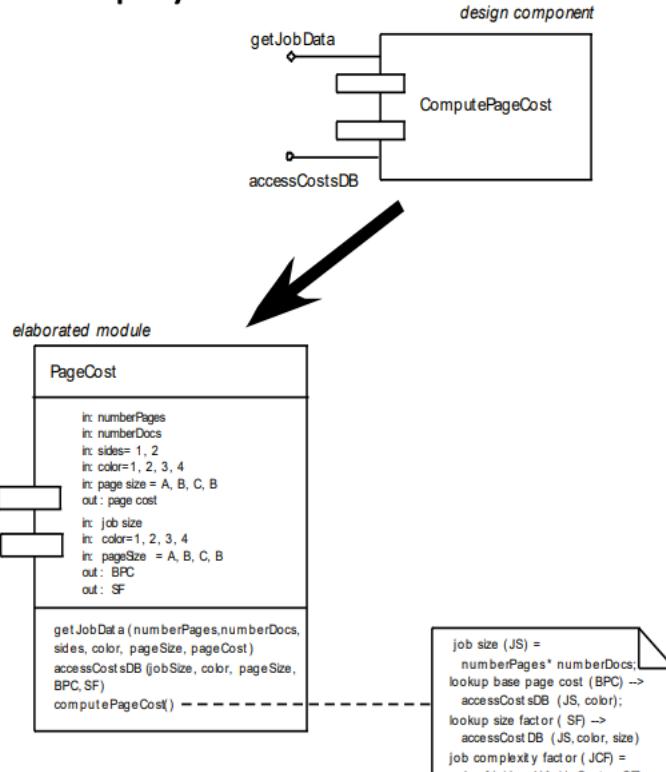
Thế nào là một thành phần? OMG Unified Modeling Language Specification [OMG01] định nghĩa một thành phần (component) là: Một phần có tính mô-đun, có thể triển khai và thay thế của một hệ thống mà đóng gói việc thực thi và đưa ra một tập các giao diện.

- Góc nhìn hướng đối tượng: Một thành phần bao gồm một tập các lớp cộng tác.
- Góc nhìn quy ước: Một thành phần bao gồm logic xử lý, cấu trúc dữ liệu bên trong cần thiết để triển khai logic đó và một giao diện cho phép thành phần được gọi và dữ liệu được truyền đến nó.

Thành phần hướng đối tượng



Thành phần quy ước



Nguyên lý thiết kế cơ bản :

- Nguyên lý mở-đóng (OCP): Một mô-đun (thành phần) nên được mở cho việc mở rộng nhưng nên đóng cho việc hiệu chỉnh.
- Nguyên lý thay thế Liskov (LSP): Các lớp con nên thay thế được các lớp cha.
- Nguyên lý tráo đổi phụ thuộc (DIP): Phụ thuộc vào trừu tượng hóa, không phụ thuộc vào kết cấu.
- Nguyên lý tách biệt giao diện (ISP): Nhiều giao diện client cụ thể thì tốt hơn một giao diện mục đích chung.
- Nguyên lý phát hành và tái sử dụng tương đương (REP): "Nguyên tắc tái sử dụng là nguyên tắc phát hành". Nói cách khác, nếu một thành phần được coi là có thể tái sử dụng thì nó phải là một đơn vị có thể thay thế được.
- Nguyên lý đóng chung (CCP): Các lớp thay đổi cùng nhau thì thuộc về nhau
- Nguyên lý tái sử dụng chung (CRP). Các lớp không được tái sử dụng cùng nhau thì không nên nhóm lại với nhau"

Hướng dẫn thiết kế

- Thành phần: Quy ước đặt tên nên được thiết lập cho các thành phần được xác định như là một phần của mô hình kiến trúc rồi sau đó tinh chỉnh và xây dựng như là một phần của mô hình mức thành phần
- Giao diện: Giao diện cung cấp thông tin quan trọng về sự giao tiếp và cộng tác (đồng thời cũng giúp chúng ta đạt được OPC)
- Phụ thuộc và kế thừa: Việc mô hình hóa sự phụ thuộc từ trái sang phải và sự kế thừa từ dưới lên trên.

Thiết kế mức thành phần - I :

- Bước 1. Xác định tất cả các lớp thiết kế tương ứng với miền vấn đề.
- Bước 2. Xác định tất cả các lớp thiết kế tương ứng với kết cấu hạ tầng.
- Bước 3. Xây dựng tất cả các lớp không có được như là thành phần tái sử dụng.
- Bước 3a. Xác định chi tiết thông điệp khi các lớp hoặc các thành phần cộng tác.
- Bước 3b. Xác định các giao diện thích hợp cho mỗi thành phần.

Thiết kế mức thành phần - II

- Step 3c. Xây dựng các thuộc tính và xác định kiểu dữ liệu và cấu trúc dữ liệu cần thiết để thực hiện chúng.
- Step 3d. Mô tả luồng xử lý trong từng hoạt động cụ thể.
- Step 4. Mô tả các nguồn dữ liệu bền vững (cơ sở dữ liệu và các tập tin) và xác định các lớp cần thiết để quản lý chúng.
- Step 5. Phát triển và xây dựng biểu diễn hành vi cho một lớp hoặc một thành phần.
- Step 6. Xây dựng sơ đồ triển khai để cung cấp thêm thông tin về chi tiết thực hiện.
- Step 7. Nhân tố hóa mỗi thể hiện thiết kế mức thành phần và luôn luôn xem xét phương án thay thế.

Thiết kế thành phần quy ước :

- Việc thiết kế các xử lý logic được quy định bởi các nguyên tắc cơ bản của thiết kế thuật toán và lập trình có cấu trúc.
- Việc thiết kế các cấu trúc dữ liệu được định nghĩa bởi các mô hình dữ liệu được phát triển cho hệ thống.

- Việc thiết kế các giao diện được quy định bởi sự hợp tác mà một thành phần phải có ảnh hưởng.

Thiết kế thuật toán :

- Là hoạt động thiết kế sát nhất với việc coding.
- Cách tiếp cận:
 - Xem xét mô tả thiết kế cho các thành phần
 - Sử dụng tinh chỉnh từng bước để phát triển thuật toán
 - Sử dụng lập trình có cấu trúc để thực hiện logic có tính thủ tục
 - Sử dụng ‘phương pháp chính thống’ để chứng minh logic.

Mô hình biểu diễn thiết kế thuật toán:

- Trình bày các thuật toán ở mức độ chi tiết mà có thể được xem xét lại chất lượng.
- Tùy chọn:
 - Đồ họa (e.g. flowchart, box diagram)
 - Mã giả (e.g., PDL)
 - Ngôn ngữ lập trình
 - Bảng quyết định

Thiết kế dữ liệu:

- Tìm kiếm biểu diễn logic cho các phần tử dữ liệu đã được nhận diện trong giai đoạn phân tích yêu cầu.
- Thiết kế các cấu trúc dữ liệu của chương trình và cơ sở dữ liệu • Các tiêu chí thiết kế dữ liệu
 - Không dư thừa dữ liệu
 - Tối ưu hóa không gian lưu trữ
 - Được biểu diễn ở các mức trừu tượng khác nhau
- Mô hình dữ liệu quan niệm, mô hình dữ liệu logic, mô hình dữ liệu vật lý

Một số nguyên tắc thiết kế dữ liệu :

- Nhận diện cả cấu trúc dữ liệu và tác vụ truy xuất
- Chú ý sử dụng từ điển dữ liệu
- Trì hoãn thiết kế dữ liệu mức thấp cho đến cuối giai đoạn này
- Che giấu biểu diễn bên trong của cấu trúc dữ liệu
- Nên áp dụng kiểu ADT trong thiết kế cũng như trong lập trình

8. Phương pháp Agile trong phát triển phần mềm

Về cơ bản, Agile là mô hình theo vòng lặp: doanh nghiệp sẽ chia một dự án lớn thành các nhiệm vụ nhỏ được thực hiện trong khoảng thời gian cố định (sprint). Nhờ đó, quá trình quản lý dự án trở nên đơn giản và dễ dàng hơn, doanh nghiệp có thể chuẩn bị tốt hơn cho dự án.

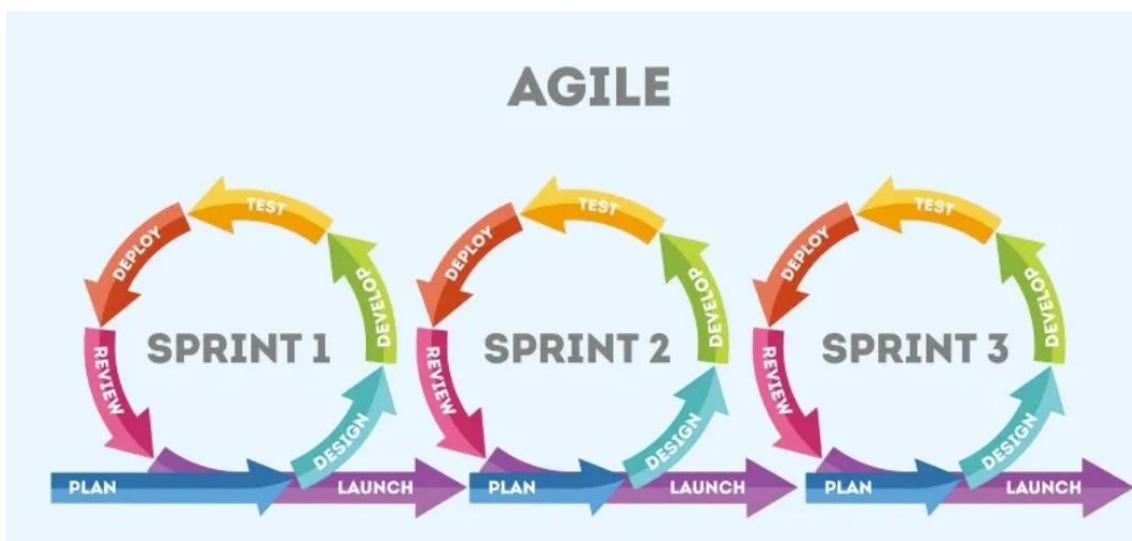
Đặc biệt trong trường hợp khách hàng thay đổi mục tiêu, **mô hình Agile** sẽ cho phép doanh nghiệp đánh giá nhiệm vụ đang làm và từng bước điều chỉnh dễ dàng hơn.

Những ưu điểm và tiềm năng của **quản lý dự án Agile** là vô cùng lớn. Mô hình giúp làm giảm được thời gian giữa các công việc, đẩy nhanh tốc độ dự án. Trong thời gian gần đây, hơn một nửa trong 19 quốc gia được khảo sát đã áp dụng **mô hình Agile**. Bên cạnh đó, nhiều doanh nghiệp khác cũng đang lựa chọn phương thức Agile để có thể theo kịp tiến độ công việc trong tương lai. Những giá trị mà mô hình Agile đem lại cho việc vận hành và quản lý dự án là vô cùng lớn. Dưới đây là 7 đặc điểm tiêu biểu của mô hình có thể tham khảo:

- **Sự phát triển:** Với mô hình Agile, các nhiệm vụ riêng biệt có thể được hoàn thành trong mỗi lần chạy nước rút được sắp xếp trong sprint backlog bởi các nhóm riêng biệt và độc lập. Theo thời gian, phần mềm sẽ phát triển dần dần cho đến khi nó đáp ứng được yêu cầu của khách hàng.
- **Tính lặp:** Mô hình Agile sẽ chia nhỏ công việc lớn thành những phần công việc nhỏ hơn và thực hiện lặp lại trong một khoảng thời gian cụ thể. Trong mỗi phần nhỏ đó, mỗi cá nhân hay nhóm sẽ được giao nhiệm vụ lên kế hoạch, nghiên cứu yêu cầu khách hàng, xây dựng và triển khai dự án, cuối cùng là đánh giá kết quả.
- **Nhóm tự tổ chức:** Tính độc lập, tự lập, không bị phụ thuộc là đặc trưng tiêu biểu của một nhóm nhân sự trong mô hình Agile. Đứng trước một khối lượng công việc đề ra,

mỗi thành viên trong tổ chức đều chủ động lên kế hoạch thực hiện công việc của riêng mình và đảm bảo kết quả chung luôn tốt nhất.

- **Tính thích ứng:** Mỗi phần công việc nhỏ sẽ được thực hiện trong khoảng thời gian cố định khá ngắn, vì vậy các thành viên đều phải có khả năng thích nghi cao.
- **Tính minh bạch:** Trong mô hình Agile thì giao tiếp đóng vai trò cốt lõi rất quan trọng. Việc giao tiếp minh bạch giúp kết nối các thành viên trong dự án, kể cả khách hàng. Thông qua đó, quy trình thực hiện sẽ trở nên rõ ràng, trơn tru hơn, hạn chế các “điểm nghẽn” do không phát hiện vấn đề kịp thời. Kết quả công việc cũng được tối ưu hơn, giảm được tình trạng phải sửa chữa nhiều.
- **Phát triển dựa trên những giá trị:** Phát triển dựa trên các giá trị được xây dựng dựa theo sự tương tác giữa nhóm phát triển và người ra yêu cầu cho sản phẩm/ công việc. Với đặc điểm này, nhóm thực thi sẽ ưu tiên những công việc quan trọng và loại bỏ những việc không cần thiết.



8.1. Ưu điểm của mô hình Agile

- + Phần mềm làm việc được bàn giao thường xuyên (vài tuần chứ không phải vài tháng).
- + Cuộc đối thoại trực tiếp (face-to-face) là hình thức giao tiếp tốt nhất.
- + Gần gũi với nhau hơn, hợp tác hàng ngày giữa các khách hàng và các lập trình viên.
- + Chú ý liên tục đến sự xuất sắc về kỹ thuật và bản thiết kế tốt.
- + Thường xuyên thích nghi với hoàn cảnh thay đổi.

8.2. Nhược điểm của mô hình Agile

- + Các sản phẩm lớn khó để đánh giá những nỗ lực bắt buộc khi bắt đầu chu trình phát triển phần mềm.
- + Thiếu sự nhấn mạnh vào thiết kế và tài liệu cần thiết.
- + Dự án có thể dễ dàng off-track nếu đại diện khách hàng không rõ kết quả cuối cùng mà họ muốn.
- + Chỉ những lập trình viên cao cấp mới có thể đưa ra các quyết định cần thiết trong quá trình phát triển.

8.3. Các nguyên tắc cần tuân thủ trong phương pháp Agile

Bất cứ tổ chức nào, quy trình hay phương pháp đều cần có nguyên tắc. Agile cũng vậy, 12 nguyên tắc dưới đây sẽ giúp cho công việc được diễn ra suôn sẻ và thành công:

Nguyên tắc #1: Ưu tiên sự hài lòng của khách hàng thông qua việc giao phần mềm sớm và liên tục.

Nguyên tắc #2: Đáp ứng yêu cầu thay đổi trong suốt quá trình phát triển

Nguyên tắc #3: Ra mắt thường xuyên phần mềm làm việc.

Nguyên tắc #4: Hợp tác giữa các bên liên quan và các nhà phát triển kinh doanh trong suốt dự án

Nguyên tắc #5: Hỗ trợ, tin tưởng và thúc đẩy những người liên quan

Nguyên tắc #6: Cho phép tương tác trực tiếp

Nguyên tắc #7: Phần mềm làm việc là thước đo chính của sự tiến bộ

Nguyên tắc #8: Các quy trình cần nhanh chóng để hỗ trợ tốc độ phát triển nhất quán của nhóm

Nguyên tắc #9: Chú ý đến chi tiết kỹ thuật và thiết kế giúp tăng cường sự nhanh nhẹn, linh hoạt

Nguyên tắc #10: Sự đơn giản

Nguyên tắc #11: Các kiến trúc tốt nhất, yêu cầu tốt nhất, và thiết kế tốt nhất sẽ được làm ra bởi các nhóm tự tổ chức.

Nguyên tắc #12: Đội sản xuất sẽ thường xuyên suy nghĩ về việc làm sao để trở nên hiệu quả hơn, sau đó họ sẽ điều chỉnh và thay đổi các hành vi của mình cho phù hợp.

8.4. Cách thức thực hiện phương pháp Agile

Vậy quy trình triển khai **mô hình Agile** bao gồm những bước nào? Quý độc giả có thể tham khảo 6 bước dưới đây trong việc ứng dụng trong phương thức Agile.

- Xây dựng kế hoạch dự án**

Tương tự như những dự án khác, đầu tiên, nhóm phát triển phải xác định mục tiêu cuối cùng, giá trị mà dự án đem lại cho khách hàng là gì. Sau đó, đội ngũ phải vạch ra những công việc cần thực hiện để đạt được mục tiêu đã đề ra. Cần lưu ý, trong quá trình áp dụng **mô hình Agile**, các công việc thực hiện này hoàn toàn có thể được điều chỉnh để phù hợp hơn với sự thay đổi trong nhu cầu.

- Thiết lập lộ trình sản phẩm**

Product roadmaps được hiểu là các giai đoạn quan trọng tương tự **Project milestones** trên hành trình tạo ra sản phẩm cuối cùng. Ở giai đoạn này, đội ngũ thực hiện cần xây dựng lộ trình cụ thể và đầy đủ nhất để có thể cho ra sản phẩm cuối cùng hoàn hảo nhất.

- Xây dựng kế hoạch phát hành**

Một trong những khác biệt của *mô hình Agile* và *Waterfall* truyền thống là mô hình Agile cho phép người thực hiện hoàn thành một tính năng/ công việc cụ thể sau mỗi giai đoạn ngắn. Thay vì phải chờ đợi hàng tháng hay cả năm trời để nhìn được sản phẩm cuối cùng, đội nhóm thực hiện và các bên liên quan có thể mường tượng được thông qua các tính năng được hoàn tất trong từng chu kỳ ngắn.

Vì thế, trước khi bắt đầu triển khai dự án, hãy xây dựng một kế hoạch cho các bản phát hành tính năng. Từ đó, các bên liên quan có thể dễ dàng truy cập và đánh giá lại kế hoạch phát hành cho mỗi tính năng ấy.

- Xây dựng kế hoạch từng sprint**

Trước khi mỗi Sprint bắt đầu, những bộ phận có liên quan phải lên kế hoạch Sprint và xác định những công việc gì cần phải hoàn thành. Lưu ý, cần phân chia nhiệm vụ đồng đều giữa những cá nhân trong nhóm để đảm bảo nhiệm vụ được hoàn thành đúng hạn trước thời gian chạy nước rút.

- Đánh giá hiệu quả dự án hàng ngày**

Vận hành dự án theo mô hình Agile đòi hỏi nhóm thực hiện phải tổ chức những cuộc họp, trao đổi ngắn hàng ngày để đánh giá và cân nhắc công việc. Trong cuộc trao đổi

đó, mỗi người sẽ thông tin ngắn gọn về những công việc đang đảm nhận, có gắp khó khăn hay vấn đề gì không.

- **Đánh giá Sprint**

Sau khi kết thúc mỗi Sprint, nhóm thực hiện sẽ tổ chức hai cuộc họp. Đó là cuộc họp với những bộ phận liên quan để nghiệm thu sản phẩm đã hoàn thành, và một cuộc họp trực tiếp để các bên thảo luận về những phát sinh về sản phẩm (nếu có).

8.5. Ứng dụng phương pháp Agile vào quản trị công việc

Sau đây là danh sách các phương pháp Agile phổ biến được sử dụng trong các dự án quản trị công việc (ngoài ra còn có một số phương pháp khác ít phổ biến hơn):

- + Phương pháp ASD – Adaptive Software Development
- + Phương pháp Agile Modeling
- + Phương pháp AUP – Agile Unified Process
- + Phương pháp Crystal Clear
- + Phương pháp DSDM – Dynamic System Development Method
- + Phương pháp XP – Extreme Programming
- + Phương pháp FDD – Feature Driven Development
- + Phương pháp Lean Software Development
- + Phương pháp Kanban
- + Phương pháp Scrum
- + Phương pháp Scrumban
- + ...

Mỗi phương pháp đều có điểm nổi trội và chưa hoàn thiện, tùy vào đặc thù của doanh nghiệp và dự án mà chúng ta có thể lựa chọn cách tiếp cận phù hợp. Một trong những yếu tố gây trở ngại nhất hiện nay cho doanh nghiệp trong việc triển khai theo phương pháp Agile là cơ cấu tổ chức nhân sự chưa phù hợp.

Có 3 yếu tố chính trong phương pháp Agile:

- + Hợp tác với khách hàng trong việc xác định yêu cầu của dự án.

+ Có quy trình và công cụ trong việc tương tác giữa các thành viên.

+ Sẵn sàng thay đổi kế hoạch khi thực hiện.

Như vậy, Agile ra đời góp phần tạo ra một phần mềm thật đơn giản đáp ứng đúng yêu cầu của khách hàng hôm nay và sẵn sàng cho những thay đổi vào ngày mai. Hiểu một cách đơn giản Agile là cách thức làm phần mềm linh hoạt để làm sao đưa sản phẩm đến tay người dùng càng nhanh càng tốt càng sớm càng tốt.

8.6. Đặc trưng cơ bản của Agile

Tính lặp (Iterative): Dự án sẽ được thực hiện trong các phân đoạn lặp đi lặp lại (Iteration hoặc Sprint), thường có khung thời gian ngắn (từ 1-4 tuần). Trong mỗi phân đoạn, nhóm phát triển thực hiện việc lập kế hoạch, phân tích yêu cầu, thiết kế, triển khai, kiểm thử để cho ra các phần nhỏ của sản phẩm.

Tính tăng trưởng và tiến hóa (Incremental & Evolutionary): Cuối các phân đoạn, nhóm cho ra các phần nhỏ của sản phẩm cuối cùng, có khả năng chạy tốt, được kiểm thử cẩn thận xem sản phẩm đó có tốt để sử dụng. Theo thời gian, phân đoạn này tiếp nối phân đoạn kia, các phần chạy được này sẽ được tích lũy, lớn dần lên cho tới khi toàn bộ yêu cầu của khách hàng được thỏa mãn.

Tính thích nghi (adaptive): Do các phân đoạn chỉ kéo dài trong một khoảng thời gian ngắn nên các yêu cầu thay đổi công nghệ, thay đổi định hướng về mục tiêu v.v.) đều có thể được đáp ứng theo cách thích hợp thích nghi với doanh nghiệp và thị trường.

Nhóm tự tổ chức và liên chức năng: Các cấu trúc nhóm này tự phân công công việc mà không dựa trên các mô tả cứng về chức danh hay làm việc dựa trên một sự phân cấp rõ ràng trong tổ chức. Nhóm tự tổ chức đã đủ các kỹ năng cần thiết để có thể được trao quyền tự ra quyết định, tự quản lí và tổ chức lấy công việc của chính mình để đạt được hiệu quả cao nhất.

Quản lý tiến trình thực nghiệm (Empirical Process Control): Các nhóm Agile ra các quyết định dựa trên các dữ liệu thực tiễn. Agile rút ngắn vòng phản hồi để dễ dàng thích nghi và tăng tính linh hoạt nhờ đó có thể kiểm soát được tiến trình, và nâng cao năng suất lao động.

Giao tiếp trực diện (face-to-face communication): Agile không phản đối việc tài liệu hóa, nhưng đánh giá cao hơn việc giao tiếp trực diện. Agile khuyến khích nhóm phát triển trực tiếp trao đổi với khách hàng, các thành viên trong nhóm, để nắm bắt được nhu

cầu khách hàng, trao đổi và thống nhất với nhau về thiết kế của hệ thống và cùng nhau triển khai thành các chức năng theo yêu cầu.

Phát triển dựa trên giá trị (value-based development): Dựa trên nguyên tắc “sản phẩm chạy tốt chính là thước đo của tiến độ”. Nhóm Agile thường cộng tác trực tiếp và thường xuyên với khách hàng để biết yêu cầu nào có độ ưu tiên cao hơn, mang lại giá trị cao nhất cho dự án.

9. Chọn kỹ thuật mô hình hay biểu diễn thích hợp

9.1. Kỹ thuật mô hình

* Khái niệm

- Kỹ thuật mô hình : là cách mô hình hóa các yêu cầu dưới dạng văn bản và hình ảnh để xây dựng 1 bức tranh đầy đủ về hệ thống dự định.

* Một số phương pháp :

- Mô hình hóa: Mô hình hóa là quá trình biểu diễn một hệ thống thông tin dưới dạng các mô hình hoặc biểu đồ để hiểu rõ cấu trúc, chức năng và tương tác của các thành phần trong hệ thống. Các loại mô hình thường được sử dụng bao gồm mô hình luồng dữ liệu, mô hình quan hệ, và mô hình hoạt động.

- Mô hình luồng dữ liệu (DFD - Data Flow Diagram): DFD là một biểu đồ mô hình hóa dữ liệu và luồng dữ liệu trong hệ thống thông tin. Nó biểu thị các quá trình xử lý dữ liệu, các thành phần dữ liệu, và cách chúng tương tác với nhau.

- Mô hình quan hệ (ERD - Entity-Relationship Diagram): ERD được sử dụng để biểu diễn các thực thể (entities) trong hệ thống và các quan hệ giữa chúng. Đây là một công cụ quan trọng để thiết kế cơ sở dữ liệu.

- Mô hình hoạt động (Activity Diagram): Mô hình hoạt động được sử dụng để mô tả các hoạt động và luồng công việc trong hệ thống thông tin. Nó thường được sử dụng để hiểu quy trình làm việc và tương tác giữa các thành phần.

- Mô hình trạng thái (State Diagram): Mô hình trạng thái được sử dụng để biểu diễn các trạng thái khác nhau mà một đối tượng hoặc hệ thống có thể tồn tại và các sự kiện chuyển đổi giữa các trạng thái này.

- Mô hình phân rã (Decomposition Models): Mô hình phân rã giúp chia nhỏ hệ thống thành các phần nhỏ hơn để dễ quản lý và phát triển. Các mô hình như mô hình phân rã

thành phần (Component Decomposition) và mô hình phân rã chức năng (Functional Decomposition) được sử dụng để thực hiện việc này.

- Mô hình hóa dự án (Project Modeling): Mô hình hóa dự án giúp quản lý và lập kế hoạch cho quy trình phát triển hệ thống thông tin. Các biểu đồ Gantt, mô hình kế hoạch (PERT/CPM), và mô hình phân tích tiền điều kiện (What-If Analysis) là một số ví dụ.

* **Mục đích sử dụng :**

- Kỹ thuật mô hình thường được dùng với các nhóm phân tích và thiết kế hệ thống chuyên nghiệp

- Khi cần hiểu rõ cấu trúc và hành vi của hệ thống thông tin

- Mục tiêu là phân tích kỹ thuật, thiết kế chi tiết, và triển khai hệ thống.

- Mô phỏng các khía cạnh phức tạp của hệ thống để đảm bảo hiểu rõ ràng và chi tiết.

9.2. Biểu diễn thích hợp

* **Khái niệm :**

- Biểu diễn thích hợp : là biểu diễn thông tin một cách rõ ràng và dễ hiểu cho các bên liên quan, bao gồm cả người dùng cuối, nhóm phát triển, và các nhóm quản lý.

* **Một số cách biểu diễn thích hợp :**

- Biểu đồ: Sử dụng các loại biểu đồ như biểu đồ cơ sở dữ liệu (ERD), biểu đồ luồng dữ liệu (DFD), biểu đồ hoạt động (Activity Diagrams), biểu đồ trạng thái (State Diagrams), và biểu đồ Gantt để mô tả các khía cạnh khác nhau của hệ thống.

- Sơ đồ: Sử dụng sơ đồ để biểu diễn cấu trúc và tương tác của các thành phần trong hệ thống. Sơ đồ có thể là sơ đồ hệ thống (system diagram), sơ đồ tương tác (interaction diagram), sơ đồ kiến trúc (architecture diagram), và sơ đồ luồng công việc (workflow diagram).

- Biểu đồ UML (Unified Modeling Language): UML là một ngôn ngữ mô hình hóa phổ biến được sử dụng trong phân tích và thiết kế hệ thống thông tin. Nó cung cấp các phần tử và biểu diễn chuẩn để mô tả cấu trúc và hành vi của hệ thống.

- Biểu đồ gốc dự án (Project Charter): Biểu đồ này bao gồm các thông tin cơ bản về dự án như mục tiêu, phạm vi, lịch trình, nguồn lực, và các người tham gia. Đây là một công cụ quan trọng để trình bày cho các bên liên quan về mục tiêu và phạm vi của dự án.

- Biểu đồ tư duy (Mind Maps): Sử dụng biểu đồ tư duy để tóm tắt ý tưởng, thông tin, và mối quan hệ giữa chúng. Đây có thể là công cụ hữu ích trong quá trình phân tích và thiết kế để hiểu cấu trúc tổ chức và mối tương tác.
- Biểu đồ sơ đồ dữ liệu (Data Flow Diagrams): Sử dụng biểu đồ DFD để biểu diễn luồng dữ liệu và quá trình xử lý dữ liệu trong hệ thống thông tin. Điều này giúp làm rõ cách thông tin di chuyển qua hệ thống.
- Biểu đồ kiến trúc hệ thống (System Architecture Diagram): Sử dụng biểu đồ này để mô tả kiến trúc tổng quan của hệ thống, bao gồm các thành phần phần mềm và phần cứng, giao tiếp giữa chúng, và các giao diện người dùng.

* **Mục đích sử dụng**

- Khi cần trình bày thông tin cho các bên liên quan không chuyên về công nghệ, bao gồm cả khách hàng và các nhóm quản lý.
- Mục tiêu là trình bày thông tin về dự án, tiến độ, kế hoạch và mục tiêu một cách dễ hiểu.
- Muốn tạo sự tham gia và hiểu biết từ phía người dùng cuối.
- Khi dự án đòi hỏi trình bày thông tin tổng quan và quản lý dự án.

9.3. So sánh kỹ thuật mô hình và biểu diễn thích hợp

* **Mục tiêu chính:**

- Kỹ thuật mô hình: Mục tiêu chính của kỹ thuật mô hình là tạo ra các biểu đồ, mô hình, hoặc biểu diễn trừu tượng để mô tả cấu trúc, hành vi, và tương tác của các thành phần trong hệ thống thông tin. Mô hình là công cụ hữu ích để hiểu sâu hơn về hệ thống.
- Biểu diễn thích hợp: Mục tiêu chính của biểu diễn thích hợp là truyền đạt thông tin một cách rõ ràng và hiệu quả cho các bên liên quan đến dự án. Biểu diễn thích hợp tập trung vào cách biểu diễn thông tin để nó trở nên dễ hiểu cho mọi người.

* **Loại công cụ:**

- Kỹ thuật mô hình: Sử dụng công cụ và kỹ thuật mô hình hóa như biểu đồ DFD, biểu đồ luồng dữ liệu, biểu đồ hoạt động, biểu đồ ERD, và các phần mềm mô hình hóa như Visio, Lucidchart, hoặc các công cụ UML.

- Biểu diễn thích hợp: Sử dụng các công cụ và kỹ thuật để tạo biểu đồ, sơ đồ, và trình bày dự án một cách dễ hiểu. Các công cụ bao gồm PowerPoint, Keynote, Prezi, và các công cụ đồ họa khác.

*** Phạm vi ứng dụng:**

- Kỹ thuật mô hình: Kỹ thuật mô hình thường được sử dụng trong quá trình phân tích chi tiết và thiết kế kỹ thuật hệ thống thông tin. Nó giúp các nhà phân tích và nhà thiết kế hiểu rõ cấu trúc và chức năng của hệ thống.
- Biểu diễn thích hợp: Biểu diễn thích hợp thường được sử dụng để trình bày thông tin về dự án, tiến độ, kế hoạch, và mục tiêu cho các bên liên quan, bao gồm cả khách hàng và các nhóm quản lý.

*** Mức độ trừu tượng:**

- Kỹ thuật mô hình: Kỹ thuật mô hình thường sử dụng trừu tượng cao hơn để biểu diễn các khía cạnh phức tạp của hệ thống. Nó tập trung vào cấu trúc và hành vi của hệ thống mà không quan tâm đến chi tiết triển khai cụ thể.
- Biểu diễn thích hợp: Biểu diễn thích hợp thường sử dụng biểu đồ và sơ đồ cụ thể để trình bày thông tin một cách dễ hiểu và cụ thể. Nó có thể bao gồm các chi tiết về kế hoạch, lịch trình, và nguồn lực.

*** Đối tượng mục tiêu:**

- Kỹ thuật mô hình: Kỹ thuật mô hình thường dành cho các chuyên gia trong lĩnh vực phân tích và thiết kế hệ thống thông tin, cũng như các nhóm phát triển hệ thống.
- Biểu diễn thích hợp: Biểu diễn thích hợp hướng đến mọi người trong tổ chức, bao gồm cả người dùng cuối và các bên liên quan không chuyên về công nghệ.

9.4. Chọn kỹ thuật mô hình hay biểu diễn thích hợp

Việc nên sử dụng kỹ thuật mô hình hay biểu diễn thích hợp phụ thuộc vào mục tiêu và ngữ cảnh của dự án hoặc tình huống cụ thể. Thường thì sẽ kết hợp cả hai để đảm bảo hiệu suất và hiểu biết tốt nhất về hệ thống thông tin đang quản lý.

Ví dụ cụ thể với hệ thống hệ thống thương mại điện tử (E-commerce):

*** Khi nào dùng Kỹ thuật mô hình trong hệ thống thương mại điện tử (E-commerce):**

- Thiết kế Cơ sở Dữ liệu: có thể sử dụng mô hình hóa cơ sở dữ liệu để mô tả các thực thể như sản phẩm, đơn hàng, người dùng, và quan hệ giữa chúng.

- Quy trình đặt hàng: Khi muốn hiểu quy trình đặt hàng, có thể sử dụng biểu đồ luồng dữ liệu hoặc biểu đồ hoạt động để biểu diễn cách thông tin và dữ liệu di chuyển qua các bước từ lúc khách hàng chọn sản phẩm đến lúc đơn hàng được xử lý.

- Kiến trúc hệ thống: Khi muốn hiểu kiến trúc tổng quan của hệ thống e-commerce, bạn có thể sử dụng biểu đồ kiến trúc hệ thống để mô tả các thành phần phần mềm, cơ sở dữ liệu, và giao tiếp giữa chúng.

* **Khi nào dùng Biểu diễn thích hợp trong hệ thống thương mại điện tử (E-commerce):**

- Báo cáo tiến độ dự án: Khi cần thông báo tiến độ phát triển dự án e-commerce cho ban quản lý hoặc các bên liên quan, bạn có thể sử dụng biểu đồ Gantt hoặc biểu đồ tương tác để trình bày các milestone và kế hoạch thực hiện.

- Chiến lược tiếp thị: Khi muốn trình bày chiến lược tiếp thị và kế hoạch quảng cáo cho sản phẩm e-commerce của bạn, có thể sử dụng biểu đồ marketing hoặc bảng điều khiển trực quan để thể hiện dữ liệu liên quan đến lượt truy cập, chuyển đổi, và doanh số bán hàng.

- Họp gấp khách hàng: Khi muốn trình bày sản phẩm và tính năng cho khách hàng hoặc đối tác, bạn có thể sử dụng slide trình bày kết hợp với hình ảnh và biểu đồ để minh họa và trình bày thông tin một cách dễ hiểu.

10. Trình bày ngắn gọn 14 biểu đồ UML:

10.1. Structural diagram

Sơ đồ cấu trúc hiển thị cấu trúc tĩnh của hệ thống và các bộ phận của nó ở các mức độ trừu tượng và triển khai khác nhau cũng như cách các bộ phận đó liên quan với nhau. Các thành phần trong sơ đồ cấu trúc thể hiện các khái niệm có ý nghĩa của một hệ thống và có thể bao gồm các khái niệm trừu tượng, thế giới thực và triển khai.

Sơ đồ cấu trúc không sử dụng các khái niệm liên quan đến thời gian, không hiển thị chi tiết về hành vi động. Tuy nhiên, chúng có thể thể hiện mối quan hệ với hành vi của các bộ phận loại được trình bày trong sơ đồ cấu trúc.

| Diagram | Mục đích | Yếu tố |
|------------------------------------|---|---|
| <u>Class diagram</u> | Hiển thị cấu trúc của hệ thống, hệ thống con hoặc thành phần được thiết kế dưới dạng các lớp và giao diện liên quan, với các tính năng, ràng buộc và mối quan hệ của chúng - liên kết, khai quát hóa, phụ thuộc, v.v. | <u>class</u> , <u>interface</u> , <u>feature</u> , <u>constraint</u> , <u>association</u> , <u>generalization</u> , <u>dependency</u> . |
| <u>Object diagram</u> | Biểu đồ của các đối tượng, bao gồm các đối tượng và giá trị dữ liệu. Sơ đồ đối tượng tĩnh là một thể hiện của sơ đồ lớp; nó hiển thị ảnh chụp nhanh về trạng thái chi tiết của hệ thống tại một thời điểm. Nó cũng tuyên bố rằng sơ đồ đối tượng là "sơ đồ lớp có đối tượng và không có lớp". | <u>instance specification</u> , <u>object</u> , <u>slot</u> , <u>link</u> . |
| <u>Package diagram</u> | Hiển thị package và mối quan hệ giữa các package | <u>package</u> , <u>packageable element</u> , <u>dependency</u> , <u>element import</u> , <u>package import</u> , <u>package merge</u> . |
| <u>Composite structure diagram</u> | dùng sơ đồ để biểu diễn: <u>Cấu trúc bên trong của một bộ phân loại</u> <u>Hành vi hợp tác</u> | |
| <u>Component diagram</u> | Hiển thị các thành phần và sự phụ thuộc giữa chúng. Loại sơ đồ này được sử dụng cho Phát triển dựa trên thành phần (CBD), để mô tả các hệ thống có Kiến trúc hướng dịch vụ (SOA).. | <u>component</u> , <u>interface</u> , <u>provided interface</u> , <u>required interface</u> , <u>class</u> , <u>port</u> , <u>connector</u> , <u>artifact</u> , <u>component realization</u> , <u>usage</u> . |

| | | |
|---------------------------|---|---|
| <u>Deployment diagram</u> | <p>Hiển thị kiến trúc của hệ thống dưới dạng <u>triển khai</u> (phân phối) <u>các tạo phẩm</u> phần mềm cho <u>các mục tiêu triển khai</u>.</p> <p>Lưu ý rằng <u>các thành phần</u> đó đã được triển khai trực tiếp tới các nút trong sơ đồ triển khai UML 1.x. Trong UML 2.x, <u>các tạo phẩm</u> được triển khai tới các nút và các tạo phẩm có thể <u>biểu hiện</u> (trình khai) các thành phần. Các thành phần được triển khai gián tiếp đến các nút thông qua các tạo phẩm.</p> <p><u>Sơ đồ triển khai mức đặc tả</u> (còn gọi là cấp loại) hiển thị một số tổng quan về <u>việc triển khai</u> <u>các tạo phẩm</u> cho <u>các mục tiêu triển khai</u> mà không tham chiếu đến các phiên bản cụ thể của các tạo phẩm hoặc nút.</p> <p><u>Sơ đồ triển khai cấp phiên bản</u> hiển thị <u>việc triển khai</u> các phiên bản của <u>thành phần</u> là cho các phiên bản cụ thể của <u>mục tiêu triển khai</u>. Ví dụ: nó có thể được sử dụng để hiển thị sự khác biệt trong việc triển khai tới môi trường phát triển, dàn dựng hoặc sản xuất với tên/id của máy chủ hoặc thiết bị xây dựng hoặc triển khai cụ thể.</p> | <u>deployment</u> , <u>artifact</u> , <u>deployment target</u> , <u>node</u> , <u>device</u> , <u>execution</u> <u>environment</u> , <u>communication path</u> , <u>deployment specification</u> , |
| <u>Profile diagram</u> | <p>Sơ đồ UML phụ trợ cho phép xác định các khuôn mẫu tùy chỉnh, các giá trị được gắn thẻ và các ràng buộc như một cơ chế mở rộng nhẹ cho tiêu chuẩn UML. Cấu hình cho phép điều chỉnh siêu mô hình UML cho các mục đích khác nhau</p> | <u>profile</u> , <u>metaclass</u> , <u>stereotype</u> , <u>extension</u> , <u>reference</u> , <u>profile</u> <u>application</u> . |

| | | |
|--|---|--|
| | <p>nền tảng (chẳng hạn như J2EE hoặc .NET) hoặc các miền (chẳng hạn như mô hình hóa quy trình kinh doanh hoặc thời gian thực).</p> <p>Sơ đồ hồ sơ lần đầu tiên được giới thiệu trong UML 2.0.standard. Profiles allow to adapt the UML metamodel for different platforms (such as J2EE or .NET), or domains (such as real-time or business process modeling).</p> <p>Profile diagrams were first introduced in UML 2.0.</p> | |
|--|---|--|

10.2. Behavior Diagrams

Sơ đồ hành vi cho thấy hành vi động của các đối tượng trong hệ thống, có thể được mô tả như một chuỗi các thay đổi đối với hệ thống theo thời gian .

| Diagram | Mục đích | Yếu tố |
|-------------------------|---|--|
| <u>Use case diagram</u> | Mô tả một tập hợp các hành động (<u>trường hợp sử dụng</u>) mà một số hệ thống hoặc các hệ thống (chủ thẻ) nên hoặc có thể thực hiện cùng với một hoặc nhiều người dùng bên ngoài của hệ thống (<u>tác nhân</u>) để cung cấp một số kết quả có giá trị và có thể quan sát được cho các tác nhân hoặc các bên liên quan khác của hệ thống. (các) hệ thống. | <u>use case</u> , <u>actor</u> , <u>subject</u> , <u>extend</u> , <u>include</u> , <u>association</u> . |
| <u>Activity diagram</u> | Hiển thị trình tự và điều kiện để phối hợp các hành vi cấp thấp hơn, thay vì trình phân loại nào sở hữu | <u>activity</u> , <u>partition</u> , <u>action</u> , <u>object</u> , <u>control</u> , <u>activity edge</u> . |

| | | |
|------------------------------|--|--|
| | các hành vi đó. Chúng thường được gọi là mô hình luồng điều khiển và luồng đối tượng | |
| <u>State machine diagram</u> | Được sử dụng để mô hình hóa hành vi rời rạc thông qua chuyển đổi trạng thái hữu hạn. Ngoài việc thể hiện hành vi của một phần hệ thống, máy trạng thái còn có thể được sử dụng để thể hiện giao thức sử dụng của một phần hệ thống. Hai loại máy trạng thái này được gọi là <u>máy trạng thái hành vi</u> và <u>máy trạng thái giao thức</u> . | |
| Interaction diagram | Sơ đồ tương tác bao gồm một số loại sơ đồ khác nhau: <u>Sơ đồ trình tự</u> , <u>Sơ đồ giao tiếp</u> (được gọi là sơ đồ cộng tác trong UML 1.x), <u>Sơ đồ thời gian</u> <u>Sơ đồ tổng quan về tương tác</u> | |
| <u>Sequence diagram</u> | Loại sơ đồ tương tác phổ biến nhất tập trung vào việc trao đổi thông điệp giữa <u>các dây cứu sinh</u> (đối tượng). | <u>lifeline</u> , <u>execution specification</u> , <u>message</u> , <u>combined fragment</u> , <u>interaction use</u> , <u>state invariant</u> , <u>destruction occurrence</u> . |

| | | |
|--|--|--|
| <u>Communication diagram</u> (a.k.a. Collaboration diagram in UML 1.x) | Tập trung vào sự tương tác giữa các dây cáu sinh trong đó kiến trúc của cấu trúc bên trong và cách thức này tương ứng với việc truyền thông điệp là trọng tâm. Trình tự của các thông điệp được đưa ra thông qua sơ đồ đánh số thứ tự . | <u>lifeline</u> , <u>message</u> . |
| <u>Timing diagram</u> | Hiển thị các tương tác khi mục đích chính của sơ đồ là lý giải về thời gian. Biểu đồ thời gian tập trung vào các điều kiện thay đổi bên trong và giữa các đường dây cáu sinh dọc theo trục thời gian tuyến tính. | <u>lifeline</u> , <u>state or condition</u> <u>timeline</u> , <u>destruction event</u> , <u>duration constraint</u> , <u>time constraint</u> . |
| <u>Interaction overview diagram</u> | Xác định các tương tác thông qua một biến thể của <u>sơ đồ hoạt động</u> theo cách thúc đẩy tổng quan về luồng điều khiển. Sơ đồ tổng quan về tương tác tập trung vào tổng quan về luồng điều khiển trong đó các nút là tương tác hoặc <u>sử dụng</u> <u>tương tác</u> . Dây cáu sinh và thông báo không xuất hiện ở cấp độ tổng quan này. | <u>initial node</u> , <u>flow final node</u> , <u>activity final node</u> , <u>decision node</u> , <u>merge node</u> , <u>fork node</u> , <u>join node</u> , <u>interaction</u> , <u>interaction use</u> , <u>duration constraint</u> , <u>time constraint</u> . |

Mức độ trừu tượng phù hợp cho việc phân tích yêu cầu, một số mô hình:

Class diagrams, để hiển thị các lớp đối tượng liên quan đến miền ứng dụng; của họ thuộc tính, hành vi và thuộc tính; và mối quan hệ giữa các lớp, cũng có thể được sử dụng để mô hình hóa dữ liệu, nhưng ứng dụng hạn chế này không khai thác đầy đủ các khả năng ngữ nghĩa của sơ đồ lớp.

Usecase diagrams, mô tả quan hệ giữa các tác nhân bên ngoài hệ thống và các trường hợp sử dụng mà chúng tương tác,

Activitydiagrams, để chỉ ra cách các dòng khác nhau đan xen trong một ca sử dụng hoặc vai trò nào thực hiện một số hành động nhất định như trong sơ đồ làn đường bối hoặc để mô hình hóa dòng chảy của quy trình kinh doanh.

State diagram, để thể hiện các trạng thái khác nhau mà một hệ thống hoặc đối tượng dữ liệu có thể đảm nhận và các chuyển đổi được phép giữa các trạng thái.

11. Trình bày các đặc trưng phi chức năng hay thuộc tính chất lượng phần mềm

Vài đặc điểm của sản phẩm có thể được gọi là thuộc tính chất lượng, mặc dù hầu hết các nhóm dự án đều chỉ cần xem xét cẩn thận một số ít trong số họ. Nếu nhà phát triển biết đặc điểm nào trong số này quan trọng nhất để thành công, họ có thể lựa chọn các phương pháp thiết kế và xây dựng phù hợp để đạt được mục tiêu chất lượng. Các thuộc tính chất lượng được phân loại theo nhiều tiêu chí khác nhau các sơ đồ. Một số tác giả đã xây dựng hệ thống phân cấp rộng rãi để nhóm các thuộc tính liên thành một số loại chính. Một cách để phân loại các thuộc tính chất lượng là phân biệt những đặc điểm có thể nhận thấy được thông qua việc thực thi phần mềm (chất lượng bên ngoài) từ những phần mềm không đảm bảo (chất lượng bên trong). Các yếu tố chất lượng bên ngoài chủ yếu quan trọng đối với người sử dụng, trong khi phẩm chất bên trong có ý nghĩa hơn đối với nhân viên phát triển và bảo trì. Chất lượng bên trong thuộc tính gián tiếp góp phần vào sự hài lòng của khách hàng bằng cách làm cho sản phẩm dễ nâng cấp hơn, sửa, kiểm tra và di chuyển sang nền tảng mới.

- **External quality(chất lượng bên ngoài):** Thuộc tính chất lượng bên ngoài mô tả các đặc điểm được quan sát thấy khi phần mềm đang thực thi. Chúng ảnh hưởng sâu sắc đến trải nghiệm người dùng và nhận thức của người dùng về chất lượng hệ thống.
- **Availability:** Tính sẵn sàng là thước đo thời gian hoạt động theo kế hoạch trong đó các dịch vụ của hệ thống có sẵn cho sử dụng và vận hành đầy đủ. Về mặt hình thức, tính khả dụng bằng tỷ lệ thời gian hoạt động trên tổng thời gian hoạt động và thời gian xuống. Còn chính thức hơn, tính khả dụng bằng với thời gian trung bình giữa các lần hỏng hóc (MTBF) của hệ thống chia cho tổng MTBF và thời gian sửa chữa trung bình (MTTR) của hệ thống sau khi xảy ra lỗi đang gặp phải. Thời gian bảo trì theo lịch trình cũng ảnh

hướng đến tính khả dụng. Sự sẵn có có liên quan chặt chẽ với độ tin cậy và bị ảnh hưởng mạnh mẽ bởi tiêu thể loại khả năng bảo trì của khả năng sửa đổi

- **Installability:** Phần mềm không hữu ích cho đến khi nó được cài đặt trên thiết bị hoặc nền tảng thích hợp. Một số ví dụ về cài đặt phần mềm là: tải ứng dụng về điện thoại hoặc máy tính bảng; chuyển phần mềm từ PC sang máy tính máy chủ web; cập nhật hệ điều hành; cài đặt một hệ thống thương mại lớn, chẳng hạn như một doanh nghiệp công cụ hoạch định nguồn lực; tải xuống bản cập nhật chương trình cơ sở vào hộp giải mã truyền hình cáp; và cài đặt một ứng dụng của người dùng cuối trên PC. Khả năng cài đặt mô tả mức độ dễ dàng thực hiện các thao tác này một cách chính xác. Việc tăng khả năng cài đặt của hệ thống giúp giảm thời gian, chi phí, sự gián đoạn của người dùng, tần suất lỗi, và trình độ kỹ năng cần thiết cho hoạt động cài đặt. Khả năng cài đặt giải quyết các hoạt động sau:

Cài đặt ban đầu

- Khôi phục sau quá trình cài đặt không đầy đủ, không chính xác hoặc do người dùng hủy bỏ
- Cài đặt lại phiên bản tương tự
- Cài đặt phiên bản mới
- Hoàn nguyên về phiên bản trước
- Cài đặt các thành phần hoặc bản cập nhật bổ sung
- Gỡ cài đặt.
- **Integrity:** Tính toàn vẹn liên quan đến việc ngăn ngừa mất thông tin và duy trì tính chính xác của dữ liệu được nhập vào hệ thống. Yêu cầu về tính toàn vẹn không có sai sót: dữ liệu ở trạng thái tốt và được bảo vệ, hoặc không. Dữ liệu cần được bảo vệ chống lại các mối đe dọa như mất mát vô tình hoặc hỏng hóc, các bộ dữ liệu có vẻ giống hệt nhau nhưng không khớp, hư hỏng vật lý đối với phương tiện lưu trữ, vô tình xóa tập tin hoặc ghi đè dữ liệu bởi người dùng. Các cuộc tấn công có chủ ý cố tình Dữ liệu bị hỏng hoặc bị đánh cắp cũng là một rủi ro. Bảo mật đôi khi được coi là một tập hợp con của tính toàn vẹn, bởi vì một số yêu cầu bảo mật nhằm ngăn chặn việc người dùng trái phép truy cập vào dữ liệu. Chính trực yêu cầu phải đảm bảo rằng dữ liệu nhận được từ các hệ thống khác khớp với dữ liệu được gửi và ngược lại. Bản thân các phần mềm thực thi có thể bị tấn công, do đó tính toàn vẹn của chúng cũng phải được đảm bảo được bảo vệ.

Tính toàn vẹn dữ liệu cũng đề cập đến tính chính xác và định dạng phù hợp của dữ liệu. Cái này bao gồm các mối quan tâm như định dạng các trường cho ngày tháng, hạn chế các trường ở loại dữ liệu chính xác hoặc độ dài, đảm bảo rằng các phần tử dữ liệu có giá trị hợp lệ, kiểm tra mục nhập thích hợp trong một trường khi một trường khác có một giá trị nhất định, v.v. Sau đây là một số yêu cầu về tính toàn vẹn mẫu:

- **Interoperability:** Khả năng tương tác cho biết hệ thống có thể trao đổi dữ liệu và dịch vụ với phần mềm khác dễ dàng như thế nào hệ thống và mức độ dễ dàng tích hợp với các thiết bị phần cứng bên ngoài. Để đánh giá khả năng tương tác, bạn cần biết những ứng dụng nào khác mà người dùng sẽ sử dụng cùng với sản phẩm của bạn và dữ liệu nào họ mong đợi trao đổi. Người sử dụng Hệ thống Theo dõi Hóa chất đã quen với để vẽ các cấu trúc hóa học bằng một số công cụ thương mại, vì vậy họ đã trình bày những điều sau đây:
 - **Performance:** Hiệu suất là một trong những thuộc tính chất lượng mà người dùng thường đưa ra một cách tự nhiên. Hiệu suất thể hiện khả năng đáp ứng của hệ thống đối với các yêu cầu và hành động khác nhau của người dùng. Hiệu suất kém gây khó chịu cho người dùng đang chờ truy vấn hiển thị kết quả. Nhưng các vấn đề về hiệu suất cũng có thể gây ra những rủi ro nghiêm trọng đối với an toàn, chẳng hạn như khi một quy trình thời gian thực hệ thống điều khiển bị quá tải. Yêu cầu nghiêm ngặt về hiệu năng ảnh hưởng mạnh mẽ đến thiết kế phần mềm chiến lược và lựa chọn phần cứng, từ đó xác định mục tiêu hiệu suất phù hợp với hoạt động
 - **Reliability:** Xác suất để phần mềm thực thi không bị lỗi trong một khoảng thời gian cụ thể đã được biết như độ tin cậy. Các vấn đề về độ tin cậy có thể xảy ra do đầu vào không đúng, sai sót trong bản thân mã phần mềm, các thành phần không có sẵn khi cần và lỗi phần cứng. Sự mạnh mẽ và tính sẵn sàng có liên quan chặt chẽ đến độ tin cậy. Các cách xác định và đo lường phần mềm độ tin cậy bao gồm tỷ lệ phần trăm các hoạt động được hoàn thành chính xác, độ dài trung bình thời gian hệ thống chạy trước khi bị lỗi (thời gian trung bình giữa các lần lỗi hoặc MTBF) và thời gian tối đa xác suất xảy ra sự cố có thể chấp nhận được trong một khoảng thời gian nhất định. Thiết lập độ tin cậy định lượng yêu cầu dựa trên mức độ ảnh hưởng nghiêm trọng nếu xảy ra lỗi và liệu chi phí tối đa hóa độ tin cậy là hợp lý. Các hệ thống yêu cầu độ tin cậy cao cũng cần được thiết kế cho khả năng xác minh cao để giúp dễ dàng tìm ra các khiếm khuyết có thể ảnh hưởng đến độ tin cậy.

- **Robustness:** Tính vững chắc là mức độ mà hệ thống tiếp tục hoạt động bình thường khi gặp phải đầu vào không hợp lệ, lỗi trong các thành phần phần cứng hoặc phần mềm được kết nối, sự tấn công từ bên ngoài hoặc các điều kiện hoạt động không mong muốn. Phần mềm mạnh mẽ phục hồi một cách khéo léo sau các tình huống có vấn đề và tha thứ cho những sai sót của người dùng. Nó phục hồi sau các lỗi nội bộ mà không ảnh hưởng xấu đến trải nghiệm của người dùng cuối. Lỗi phần mềm được xử lý theo cách mà người dùng cho là hợp lý, không gây khó chịu.
- **Safety:** Yêu cầu an toàn giải quyết nhu cầu ngăn chặn hệ thống gây thương tích cho con người hoặc thiệt hại về tài sản. Yêu cầu an toàn có thể được quyết định bởi các quy định của chính phủ hoặc các quy tắc kinh doanh khác và các vấn đề pháp lý hoặc chứng nhận có thể liên quan đến với việc đáp ứng những yêu cầu đó. Yêu cầu an toàn thường được viết dưới dạng điều kiện hoặc hành động mà hệ thống không được phép xảy ra.
- **Security:** Bảo mật liên quan đến việc chặn truy cập trái phép vào các chức năng hoặc dữ liệu của hệ thống, đảm bảo rằng phần mềm được bảo vệ khỏi các cuộc tấn công của phần mềm độc hại, v.v. Bảo mật là vấn đề lớn với Internet phần mềm. Người dùng hệ thống thương mại điện tử muốn thông tin thẻ tín dụng của họ được bảo mật. Web người lướt sóng không muốn thông tin cá nhân hoặc hồ sơ về các trang web họ truy cập bị sử dụng không thích hợp. Các công ty muốn bảo vệ trang web của họ khỏi các cuộc tấn công từ chối dịch vụ hoặc hack. Như với yêu cầu về tính toàn vẹn, yêu cầu về bảo mật không có khả năng chấp nhận lỗi. Sau đây là một số những cân nhắc cần kiểm tra khi đưa ra các yêu cầu bảo mật.
- **Usability:** Cân bằng khả năng sử dụng tối ưu cho toàn bộ đối tượng người dùng chứ không chỉ cho một cộng đồng duy nhất. Điều này có thể có nghĩa là một số người dùng nhất định không hài lòng với kết quả như họ mong muốn. Người dùng các tùy chọn tùy chỉnh có thể mở rộng sức hấp dẫn của ứng dụng.
- **Efficiency:** Hiệu quả có liên quan chặt chẽ với thuộc tính chất lượng bên ngoài là hiệu suất. Hiệu quả là thước đo của hệ thống sử dụng dung lượng bộ xử lý, dung lượng ổ đĩa, bộ nhớ hoặc băng thông liên lạc tốt như thế nào. Nếu như một hệ thống tiêu tốn quá nhiều tài nguyên sẵn có, người dùng sẽ gặp phải tình trạng hiệu suất bị suy giảm. Hiệu quả—và do đó là hiệu suất—là yếu tố thúc đẩy trong kiến trúc hệ thống, ảnh hưởng đến cách thức một nhà thiết kế chọn phân phối các tính toán và chức năng trên các thành phần hệ thống. Hiệu quả yêu cầu có thể làm tổn hại đến việc đạt được các thuộc tính chất lượng khác. Xem xét tối thiểu cấu hình phần cứng khi xác định mục tiêu hiệu quả,

năng lực và hiệu suất. Cho phép lợi nhuận kỹ thuật cho các điều kiện không lường trước được và sự tăng trưởng trong tương lai (do đó ảnh hưởng đến khả năng mở rộng).

- **Modifiability:** Tính có thể sửa đổi đề cập đến việc các thiết kế và mã phần mềm có thể được hiểu, thay đổi, và mở rộng. Khả năng sửa đổi bao gồm một số thuật ngữ thuộc tính chất lượng khác có liên quan đến các hình thức bảo trì phần mềm khác nhau. Nó liên quan chặt chẽ đến khả năng kiểm chứng. Nếu các nhà phát triển dự kiến thực hiện nhiều cải tiến, họ có thể chọn các phương pháp thiết kế phù hợp tối đa hóa khả năng sửa đổi của phần mềm. Khả năng sửa đổi cao là rất quan trọng đối với các hệ thống sẽ trải qua sửa đổi thường xuyên, chẳng hạn như những bản sửa đổi được phát triển bằng cách sử dụng vòng đời tăng dần hoặc lặp lại.
- **Portability:** Nỗ lực cần thiết để di chuyển phần mềm từ môi trường hoạt động này sang môi trường hoạt động khác là thước đo khả năng tính di động. Một số người thực hiện bao gồm khả năng quốc tế hóa và nội địa hóa một sản phẩm theo tiêu đề của tính di động. Các phương pháp thiết kế làm cho phần mềm có thể di chuyển được cũng tương tự như các phương pháp thiết kế làm cho nó có thể tái sử dụng được. Tính di động ngày càng trở nên quan trọng vì các ứng dụng phải chạy trên nhiều nền tảng các môi trường như Windows, Mac và Linux; iOS và Android; và PC, máy tính bảng và điện thoại.
- **Reusability:** Khả năng sử dụng lại cho biết nỗ lực tương đối cần thiết để chuyển đổi một thành phần phần mềm để sử dụng cho mục đích khác các ứng dụng. Phần mềm có thể tái sử dụng phải có tính mô-đun, được ghi chép đầy đủ, độc lập với một phần mềm cụ thể. Ứng dụng và môi trường hoạt động, và có phần chung chung về khả năng. Nhiều dự án tạo ra các phần mềm cung cấp tiềm năng tái sử dụng, bao gồm các yêu cầu, kiến trúc, thiết kế, mã, kiểm thử, quy tắc kinh doanh, mô hình dữ liệu, mô tả lớp người dùng, hồ sơ các bên liên quan và thuật ngữ thuật ngữ. Việc làm cho phần mềm có thể tái sử dụng được tạo điều kiện thuận lợi bằng cách đặc tả các yêu cầu và thiết kế, tuân thủ nghiêm ngặt các tiêu chuẩn mã hóa, duy trì bộ hồi quy các trường hợp thử nghiệm và một thư viện tiêu chuẩn được duy trì gồm các thành phần có thể tái sử dụng.
- **Scalability:** Yêu cầu về khả năng mở rộng đề cập đến khả năng ứng dụng phát triển để đáp ứng nhiều người dùng hơn, dữ liệu, máy chủ, vị trí địa lý, giao dịch, lưu lượng mạng, tìm kiếm và các dịch vụ khác mà không cần ảnh hưởng đến hiệu suất hoặc tính chính xác. Khả năng mở rộng có ý nghĩa cả về phần cứng và phần mềm. Mở rộng quy mô hệ thống có thể có nghĩa là có được máy tính nhanh hơn, thêm bộ nhớ hoặc dung lượng ổ đĩa, thêm máy chủ, cơ sở dữ liệu phản chiếu hoặc tăng dung lượng mạng. Các phương

pháp tiếp cận phần mềm có thể bao gồm phân phối tính toán lên nhiều bộ xử lý, nén dữ liệu, tối ưu hóa thuật toán, và các kỹ thuật điều chỉnh hiệu suất khác. Khả năng mở rộng có liên quan đến khả năng sửa đổi và độ bền, bởi vì một loại độ bền có liên quan đến cách hệ thống hoạt động khi có giới hạn về công suất đã đạt tới hoặc vượt quá.

- **Verifiability:** Được gọi một cách hẹp hơn là khả năng kiểm thử, khả năng kiểm chứng để cập đến mức độ tốt của các thành phần phần mềm hoặc sản phẩm tích hợp có thể được đánh giá để chứng minh liệu hệ thống có hoạt động như mong đợi hay không. Thiết kế để có thể kiểm chứng là rất quan trọng nếu sản phẩm có các thuật toán và logic phức tạp hoặc nếu nó chứa mối quan hệ chức năng tinh tế. Khả năng xác minh cũng rất quan trọng nếu sản phẩm sẽ được sửa đổi thường xuyên, bởi vì nó sẽ trải qua quá trình kiểm tra hồi quy thường xuyên để xác định xem những thay đổi làm hỏng bất kỳ chức năng hiện có nào. Các hệ thống có khả năng kiểm chứng cao có thể được kiểm tra một cách hiệu quả và một cách hiệu quả. Thiết kế phần mềm có khả năng kiểm chứng có nghĩa là làm cho phần mềm dễ dàng được đưa vào hệ thống trạng thái mong muốn trước khi thử nghiệm, để cung cấp dữ liệu thử nghiệm cần thiết và để quan sát kết quả thử nghiệm.

12. Lớp và quan hệ lớp từ phân tích đến thiết kế. Code tương ứng. Cho ví dụ.

Định nghĩa Lớp:

Trong phương pháp hướng đối tượng, các đối tượng có chung một số thuộc tính và phương thức được nhóm thành một lớp. Tương tác giữa các đối tượng trong hệ thống sẽ được biểu diễn thông qua mối quan hệ giữa các lớp.

Các lớp (bao gồm cả các thuộc tính và phương thức) cùng với các mối quan hệ sẽ tạo thành biểu đồ lớp (class diagram). Biểu đồ lớp là thể hiện mô hình tĩnh của hệ thống nhằm mô tả khung nhìn về một hệ thống bằng các khái niệm lớp, các thuộc tính, phương thức của lớp và mối quan hệ giữa chúng với nhau.

Ví dụ:

Lớp User:

| User |
|-----------------------|
| -name : Name |
| -dob : Date |
| -sex : boolean |
| -phoneNumber : String |
| -email : String |
| -address : Address |
| -account : Account |

Lớp Account:

| Account |
|--------------------|
| -username : String |
| -password : String |

Lớp Name:

| Name |
|---------------------|
| -firstName : String |
| -midName : String |
| -lastName : String |

Lớp Address:

| Address |
|--------------------|
| -section : String |
| -road : String |
| -town : String |
| -district : String |
| -city : String |

Biểu đồ lớp phân tích là một công cụ trong quy trình phân tích hệ thống trong quy trình phát triển phần mềm. Biểu đồ này giúp mô hình hóa các thực thể (entities), thuộc tính và quan hệ giữa chúng trong quá trình phân tích hệ thống. Nó thường được sử dụng ở giai đoạn đầu của quy trình phát triển để hiểu rõ yêu cầu của khách hàng và xây dựng một cơ sở cho quá trình thiết kế.

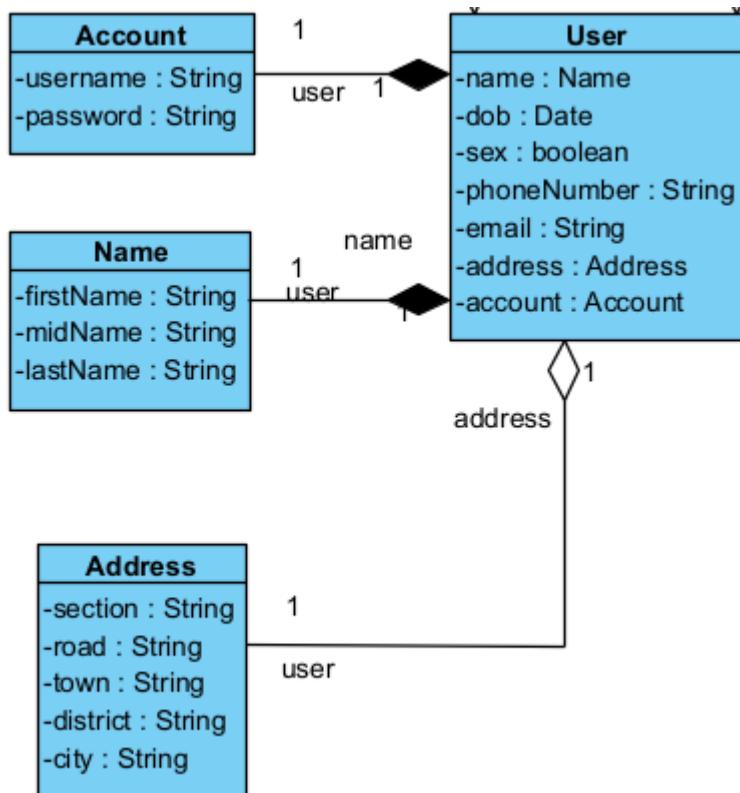
Một biểu đồ lớp phân tích có thể chứa các yếu tố sau:

- Lớp (Class):** Đại diện cho một thực thể trong hệ thống, ví dụ như đối tượng, người dùng, hoặc quá trình.
- Thuộc tính (Attribute):** Mô tả các đặc điểm của một lớp, ví dụ như tên, địa chỉ, hoặc các thuộc tính khác liên quan.

3. **Phương thức (Method):** Mô tả các hành vi hoặc chức năng của một lớp, thường được biểu diễn dưới dạng các hàm, phương thức, hoặc thủ tục.
4. **Quan hệ (Relationship):** Liên kết giữa các lớp, mô tả mức độ tương tác và phụ thuộc giữa chúng. Các quan hệ thường được mô tả bằng cách sử dụng các mũi tên hoặc đường kết nối giữa các lớp.

Biểu đồ lớp phân tích thường không mô tả các chi tiết thiết kế cụ thể của mã nguồn, mà tập trung vào hiểu rõ vấn đề và yêu cầu của hệ thống từ góc độ phân tích. Sau khi biểu đồ lớp phân tích được hoàn thành, nó có thể được sử dụng như một cơ sở để phát triển biểu đồ lớp thiết kế trong giai đoạn thiết kế hệ thống.

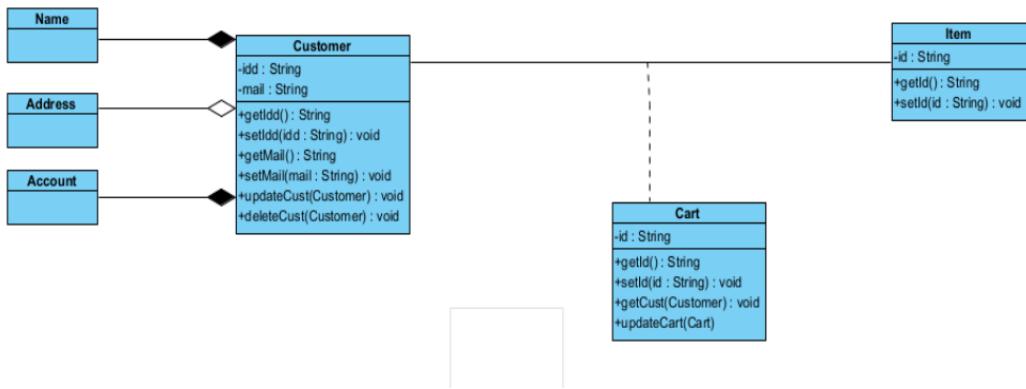
Ví dụ:



BUỚC 1: Thêm phương thức **method** vào lớp. Nhiều người cho rằng Bước này nên tạo trong pha PHÂN TÍCH hợp lý hơn. Hợp lý nghĩa là nó thể hiện quan điểm của khách hàng. Lớp Cart gọi là lớp liên kết (association class)

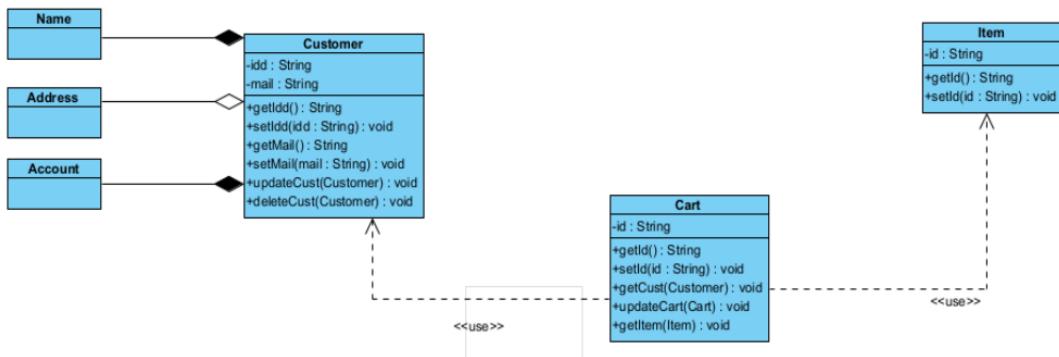
Chú ý: có 2 loại method:

- Method cho chính bản thân lớp đó. Ví dụ `updateCust(Customer)` trong lớp `Customer`, `updateCart(Cart)` trong lớp `Cart`
- Method liên kết với lớp khác. Ví dụ `getCust(Customer)`, `getItem(Item)` trong lớp `Cart`



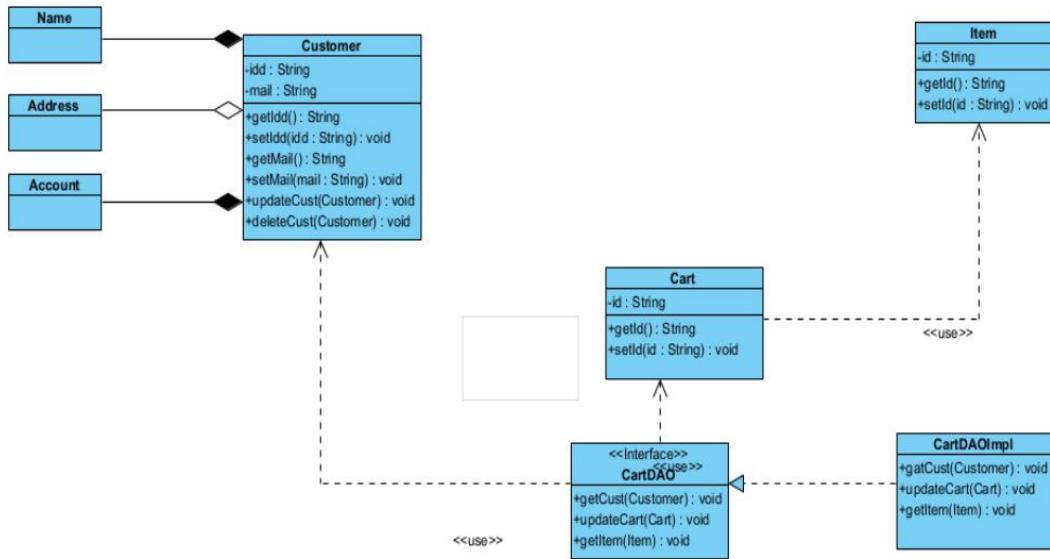
BUỚC 2

Thay quan hệ association bởi quan hệ use. Quan hệ use trong thiết kế nghĩa là khi lớp A có phương thức `method(B)` sử dụng thuộc tính của B



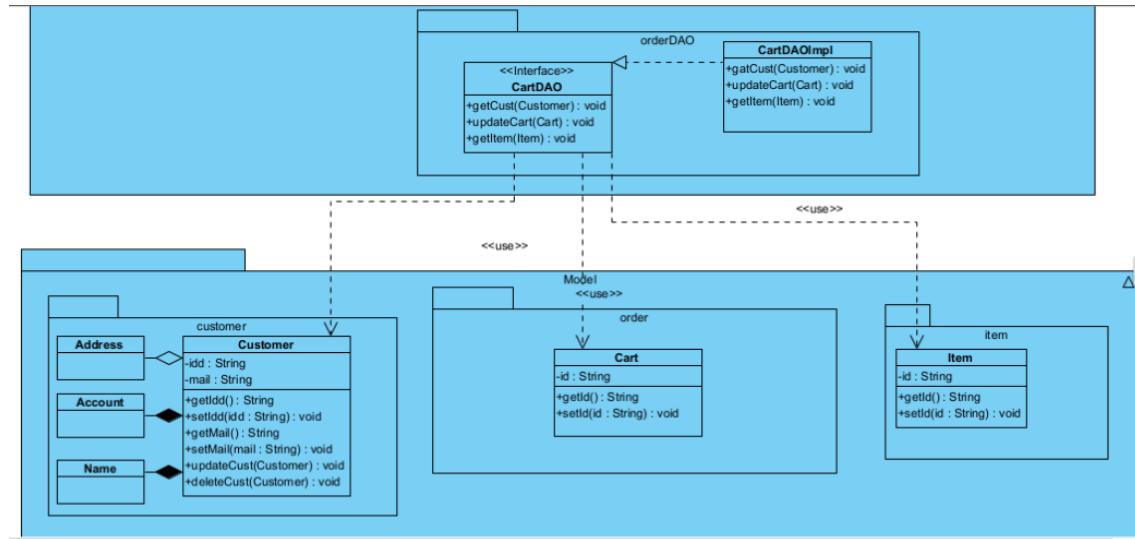
BUỚC 3:

Sử dụng DAO



BUỚC 4

Sử dụng architecture pattern MVC. Mô hình MVC cho hệ e-commerce thường gồm 3 layer: model, controller, interface. Bạn cần tiếp tục DAO cho lớp *Customer* và *Item* như lớp *Cart*. Package order sẽ chứa các lớp *Order*, *Pagement*, *Shipment*. Ta có thể tách gói cho Payment tùy quan điểm thiết kế. Các lớp *Name*, *Adddressss*, *Account* có thể để trong tầng model mà không có methods. Nếu bạn muốn thiết kế các method như *changePass(Customer)* thì có thể đặt trong *CustomerDAO* hay thiết kế riêng *AccountDAO*.



Bước 5: Code tương ứng

// I.java

```
public interface I {
```

```
    String getidd();
```

```
void setIdd(String idd);

String getMail();

void setMail(String mail);

void updateCust(Customer customer);

void deleteCust(Customer customer);

}
```

```
// Item.java

public class Item {

    private String id;

    public String getId() {

        return id;
    }
}
```

```
public void setId(String id) {

    this.id = id;
}

}
```

```
// Customer.java

public class Customer implements I {

    private String idd;

    private String mail;
```

```
// Getters and setters for id and mail

public void updateCust(Customer customer) {
    // Update customer logic
}
```

```
public void deleteCust(Customer customer) {
    // Delete customer logic
}
```

// Cart.java

```
public class Cart {
    private String id;

    public String getId() {
        return id;
    }
```

```
    public void setId(String id) {
        this.id = id;
    }
}
```

// CartDAO.java

```
public interface CartDAO {
    void getCust(Customer customer);
```

```
void updateCart(Cart cart);

void getItem(Item item);

}

// CartDAOImpl.java

public class CartDAOImpl implements CartDAO {

    public void getCust(Customer customer) {

        // Retrieve customer logic

    }

    public void updateCart(Cart cart) {

        // Update cart logic

    }

    public void getItem(Item item) {

        // Retrieve item logic

    }

}

// OrderDAO.java

public interface OrderDAO {

    // Declare necessary methods

}

// Main.java

public class Main {

    public static void main(String[] args) {
```

```
// Initialize necessary objects  
// Use the methods defined in the interfaces and their implementations  
}  
}
```

13. Khảo sát trên mạng các Hệ thống sau đây:

A. Hệ quản lý thư viện (Library Management System LibMaS)

13a.1. Trình bày bằng ngôn ngữ tự nhiên và biểu đồ context cho các hệ thống

Hệ quản lý thư viện (Library Management System - LibMaS) là một hệ thống phần mềm được thiết kế để quản lý và tổ chức các hoạt động trong một thư viện. Nó cung cấp các công cụ và chức năng để quản lý thông tin về sách, độc giả, mượn/trả sách, và các hoạt động khác liên quan đến quản lý thư viện.

Hệ thống LibMaS là một hệ thống phân tán, bao gồm các thành phần chính như máy chủ chính, cơ sở dữ liệu, giao diện người dùng và các thiết bị ngoại vi. Máy chủ chính là nơi lưu trữ và xử lý dữ liệu của thư viện, bao gồm thông tin về sách, thông tin độc giả, lịch sử mượn/trả sách và các tài liệu khác. Cơ sở dữ liệu được sử dụng để lưu trữ thông tin chi tiết về sách, tác giả, danh mục, độc giả và các thông tin liên quan khác.

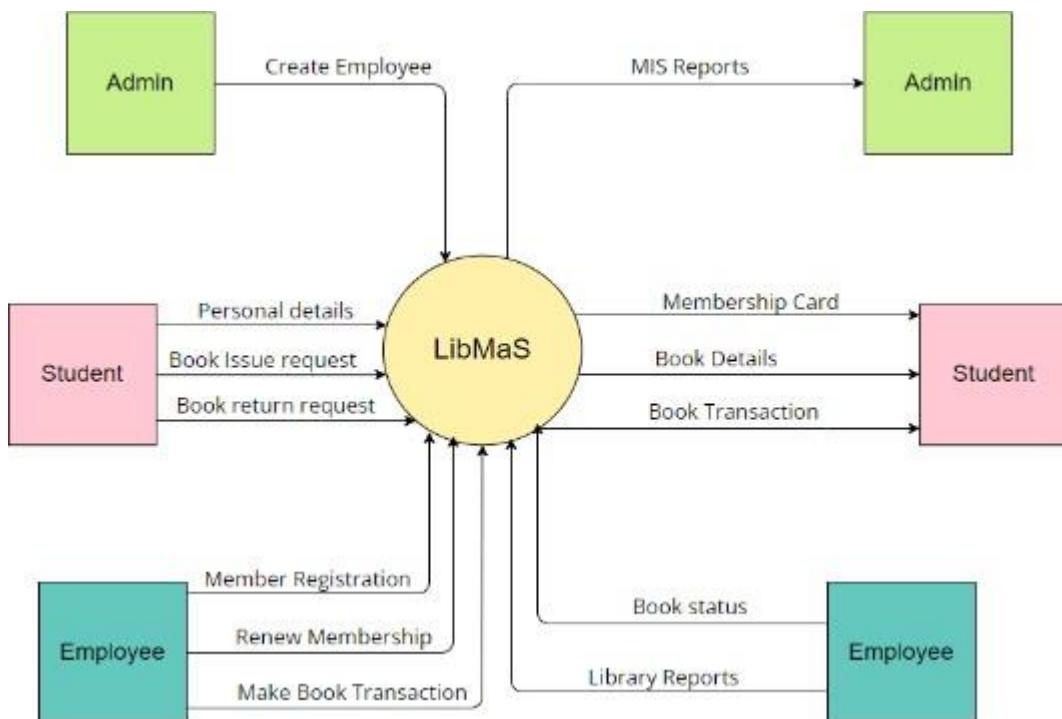
Người dùng tương tác với hệ thống thông qua giao diện người dùng, cung cấp các chức năng để tìm kiếm sách, mượn/trả sách, đăng ký thành viên mới, và quản lý tài khoản độc giả. Giao diện người dùng có thể được truy cập thông qua các thiết bị như máy tính cá nhân, máy tính xách tay, điện thoại di động hoặc máy tính bảng.

Hệ thống LibMaS cũng liên kết với các thiết bị ngoại vi như máy in, máy quét mã vạch và máy đọc thẻ để hỗ trợ các hoạt động như in phiếu mượn sách, quét mã vạch sách và thẻ độc giả, giúp tăng cường hiệu quả và độ chính xác trong quá trình quản lý và giao dịch sách.

Trong biểu đồ context, hệ thống LibMaS nằm ở trung tâm và bao gồm các thành phần của nó. Nó cũng truy cập vào cơ sở dữ liệu để lưu trữ và truy xuất thông tin liên quan đến quản lý thư viện:

- **Quản lý tài liệu và danh mục:** LibMaS cho phép người quản trị quản lý thông tin chi tiết về các tài liệu trong thư viện, bao gồm sách, báo, tạp chí và nhiều loại tài liệu khác. Các danh mục tài liệu giúp tổ chức chúng một cách có trật tự và dễ dàng tìm kiếm.

- **Quản lý độc giả và Mượn/Trả tài liệu:** Hệ thống giúp theo dõi thông tin về độc giả, quản lý quá trình mượn và trả tài liệu. Người sử dụng có thể tạo tài khoản, đăng ký thẻ thư viện và thực hiện các giao dịch mượn trả một cách thuận tiện.
- **Tìm kiếm và Tra cứu:** LibMaS cung cấp công cụ tìm kiếm mạnh mẽ, cho phép người dùng tra cứu thông tin về tài liệu, tác giả, chủ đề và nhiều yếu tố khác, giúp họ tìm thấy tài liệu một cách nhanh chóng.
- **Quản lý Nhân viên và Thống kê:** Hệ thống cho phép quản lý thông tin về nhân viên thư viện và thực hiện các chức năng liên quan như quản lý lịch làm việc, tính lương. Ngoài ra, LibMaS cung cấp các công cụ thống kê về việc sử dụng tài liệu và hoạt động thư viện.



13a.2. Mô tả Actor hệ thống

Primary actor

- Bạn đọc - Member (bao gồm những người đọc như sinh viên, giảng viên,... và bất kỳ ai có thẻ bạn đọc, tài khoản member hệ thống). Hoạt động chính:
 - + Đăng ký và đăng nhập vào hệ thống để sử dụng tài liệu.
 - + Tìm kiếm sách
 - + Mượn sách

- + Trả sách
- + Gia hạn thời gian mượn sách
- Thủ thư – Librarian. Đây là những người quản lý và điều hành thư viện. Họ cũng có thể hỗ trợ bạn đọc trong quá trình tìm kiếm và sử dụng tài liệu của thư viện. Hoạt động chính:
 - + Đăng nhập vào hệ thống để quản lý và điều hành các hoạt động mượn trả tài liệu
 - + Xác nhận các yêu cầu mượn sách
 - + Cập nhật, thêm, sửa, xóa thông tin mượn trả sách
 - + Kiểm tra các thông tin mượn trả sách
- Quản trị viên - Admin: Đây là người quản lý toàn bộ hệ thống. Họ có quyền cao nhất trong hệ thống. Hoạt động chính:
 - + Thêm, sửa, xóa sách vào trong hệ thống
 - + Quản lý tài khoản người dùng
 - + Thực hiện các tác vụ quản trị khác của hệ thống

Secondary actor

- Thanh toán trung gian – Bank: Thực hiện thu phí quản lý hoặc các dịch vụ liên quan đến tài chính trong hệ thống như thu tiền phạt của bạn đọc bị phạt
- Hệ Thống Thông Báo (Notification System): Hệ thống thông báo là một actor phụ, có nhiệm vụ gửi thông báo và nhắc nhở đến người mượn, bao gồm thông báo đặt sách trước, thông báo trễ hạn, và các thông báo khác.
- Hệ thống hiển thị: Hệ thống hiển thị trong hệ thống quản lý thư viện là một phần quan trọng để giúp người dùng và nhân viên thư viện tương tác và quản lý thông tin thư viện một cách hiệu quả.

13a.3 Mô tả yêu cầu chức năng và phi chức năng của hệ thống

Hệ thống quản lý thư viện là một ứng dụng phần mềm được thiết kế để quản lý các hoạt động và tài liệu trong một thư viện. Hệ thống này cung cấp một cách hiệu quả để tổ chức, lưu trữ, tìm kiếm và quản lý thông tin về sách, tài liệu, và các tài liệu khác trong thư viện

Các yêu cầu chức năng:

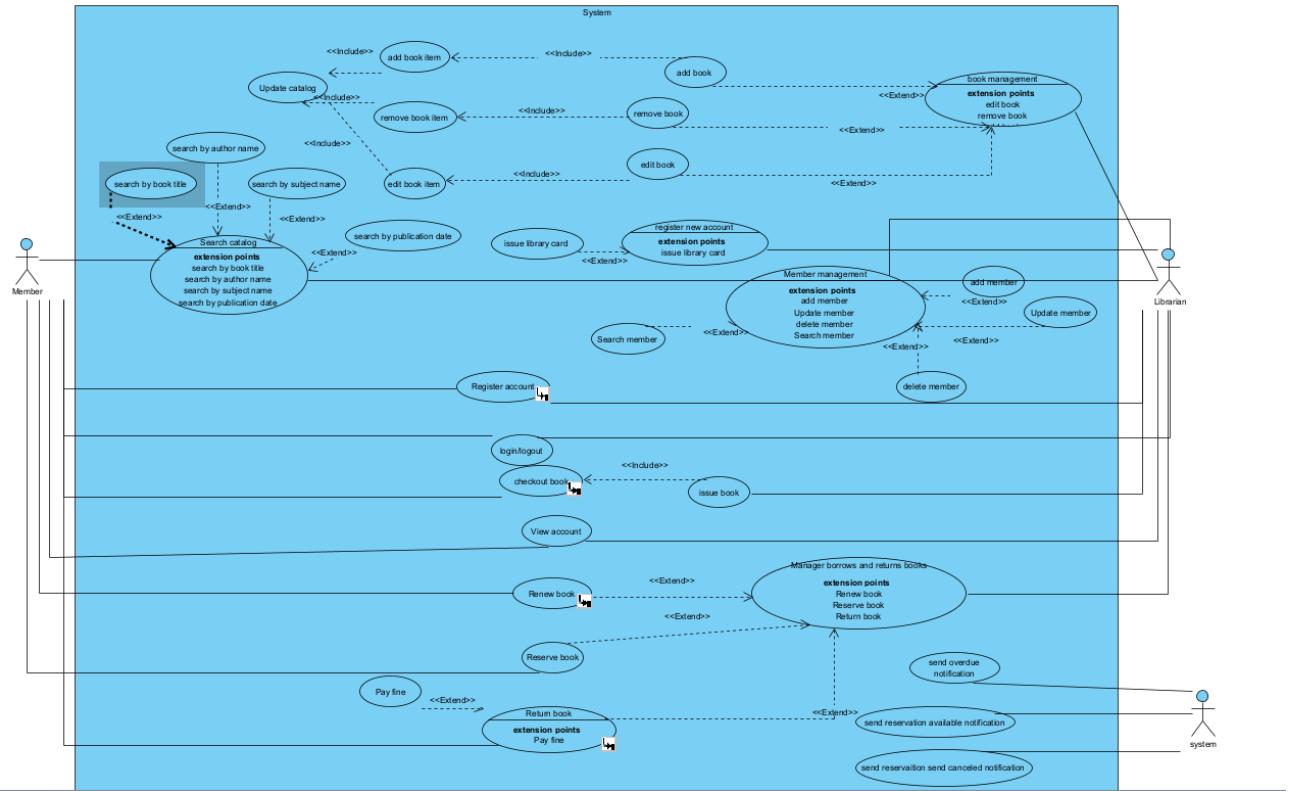
- Cho phép quản lý thư viện đăng nhập vào hệ thống, đổi mật khẩu, lấy lại mật khẩu
- Cho phép quản lý thư viện thực hiện:
 - + Quản lý Tài liệu:
 - Thêm, chỉnh sửa, và xóa thông tin về sách, báo, tạp chí, tài liệu điện tử, và các tài liệu khác.
 - Tạo mã số và mã vạch cho tài liệu.
 - Xác định vị trí lưu trữ của tài liệu trong thư viện.
 - Theo dõi số lượng tài liệu còn lại và tự động cập nhật khi mượn/trả sách. + Quản lý Người dùng:
 - Đăng ký thành viên mới và cấp phát thẻ thư viện.
 - Chỉnh sửa thông tin cá nhân và lịch sử mượn trả của người dùng.
 - Quản lý loại người dùng (học sinh, giáo viên, bạn đọc, v.v.). .
 - + Quản lý Sự Kiện và Chương Trình: Tạo và quản lý sự kiện, chương trình đọc sách, buổi thảo luận, triển lãm, và các hoạt động khác của thư viện.
 - + Báo Cáo và Thông Kê: Tạo báo cáo về hoạt động của thư viện, bao gồm số lượng sách mượn, số lượng người đến thư viện, và các dữ liệu thống kê khác.
 - + Quản lý Tài khoản Nhân viên: Quản lý tài khoản của nhân viên thư viện, đặc biệt là về quyền truy cập vào hệ thống.
 - + Quản lý Lịch Trình: Đặt lịch mở cửa, lịch nghỉ, và lịch sự kiện thư viện.
 - Cho phép thủ thư đăng nhập vào hệ thống, đổi mật khẩu, lấy lại mật khẩu
 - Cho phép thủ thư xem thông tin của mình: Hệ thống phải có tính năng cho phép thủ thư xem và sửa thông tin cá nhân của mình.
 - Cho phép thủ thư có khả năng thêm và quản lý sách: Thủ thư cần có khả năng thêm, sửa đổi và xóa sách trong thư viện. Họ cũng cần quản lý thông tin về sách bao gồm tiêu đề, tác giả, ngày xuất bản, vị trí, số lượng tồn, v.v.
 - Cho phép người dùng tìm kiếm sách dựa trên tiêu đề, ngày xuất bản, tác giả, v.v., và tìm vị trí của chúng trong thư viện: Hệ thống cần cung cấp chức năng tìm kiếm sách cho

người dùng. Người dùng có thể tìm sách theo nhiều tiêu chí như tiêu đề, ngày xuất bản, tác giả, và nhận thông tin về vị trí của sách trong thư viện.

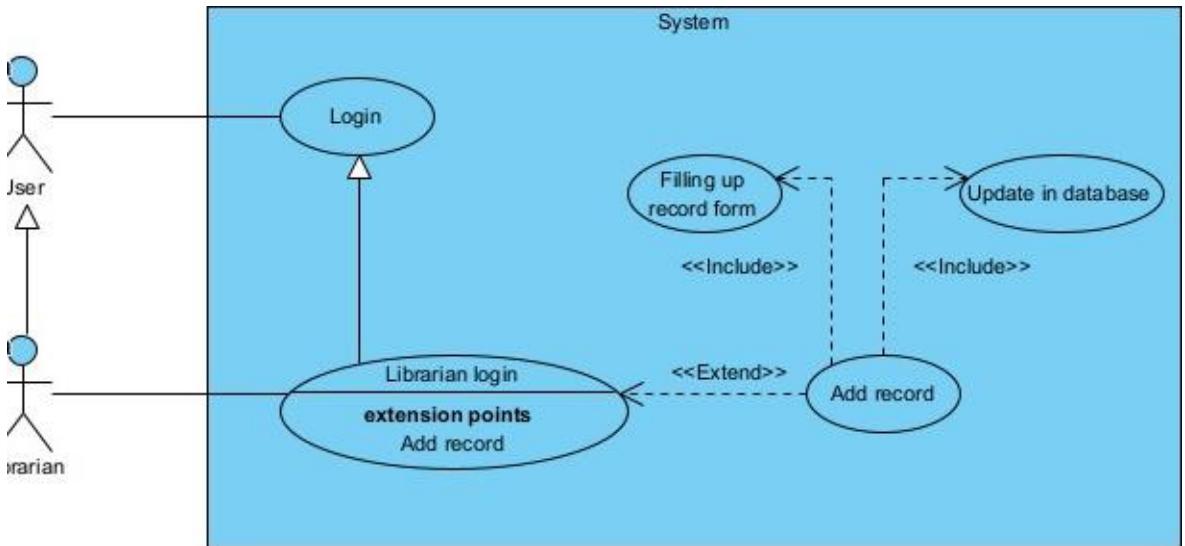
- Người dùng có thể yêu cầu, đặt trước hoặc gia hạn một cuốn sách: Hệ thống phải hỗ trợ người dùng trong việc yêu cầu mượn sách, đặt trước sách đã mượn hoặc gia hạn thời gian mượn sách.
- Hệ thống cần thông báo cho người dùng và thủ thư về sách quá hạn: Hệ thống cần gửi thông báo cho người dùng và thủ thư khi sách mượn quá hạn. Thông báo này bao gồm thông tin về sách quá hạn và mức phạt tương ứng.
- Hệ thống tính toán phạt cho sách quá hạn khi được trả lại: Khi người dùng trả lại sách quá hạn, hệ thống phải tự động tính toán mức phạt dựa trên thời gian quá hạn và các quy tắc phạt được định trước.
- Khả năng Thanh toán bằng Ví điện tử: Hệ thống cần hỗ trợ khả năng thanh toán bằng ví điện tử. Người dùng phải có thể liên kết tài khoản ví điện tử của họ và sử dụng nó để thanh toán các khoản phí mượn sách hoặc phí phạt.
- Liên kết Tài khoản Ví điện tử: Hệ thống cần cho phép người dùng liên kết tài khoản ví điện tử của họ với tài khoản trong hệ thống quản lý thư viện. Điều này có thể yêu cầu xác thực và xác minh thông tin tài khoản ví điện tử.
- Xử lý Thanh toán Ví điện tử: Hệ thống phải kết nối với các cổng thanh toán ví điện tử và xử lý các giao dịch thanh toán từ ví điện tử của người dùng. Điều này bao gồm việc tính toán tổng số tiền cần thanh toán và thực hiện giao dịch.
- Thông báo Kết quả Thanh toán: Hệ thống cần thông báo cho người dùng kết quả của giao dịch thanh toán bằng ví điện tử, bao gồm cả thông báo về thanh toán thành công hoặc thất bại.
- Lưu trữ Lịch sử Thanh toán: Hệ thống cần lưu trữ lịch sử các giao dịch thanh toán bằng ví điện tử để có thể theo dõi và xem lại.

13a.4. Trình bày Biểu đồ Usecase THEO 2 LEVEL

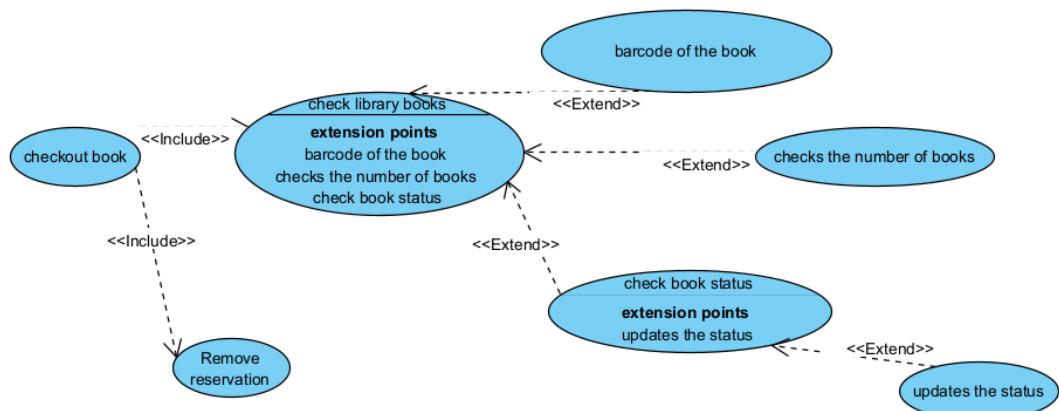
- Usecase tổng quan:



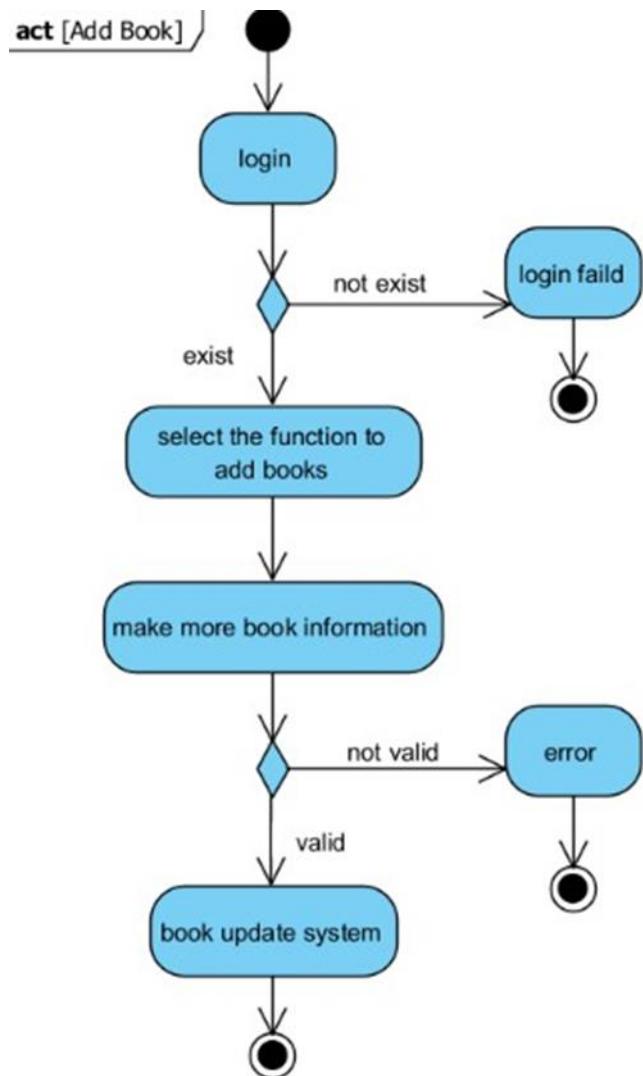
- Usecase thêm sách

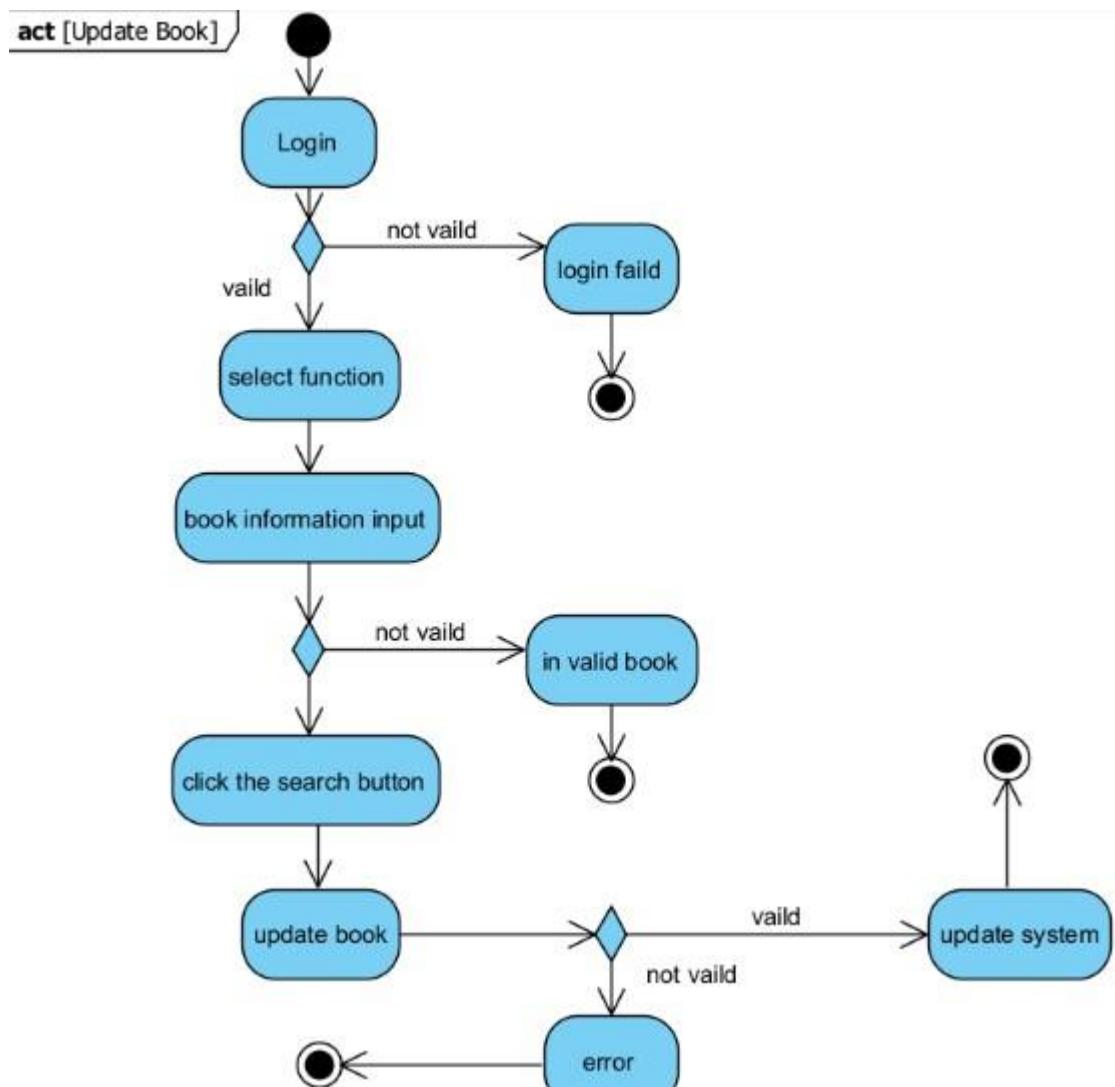


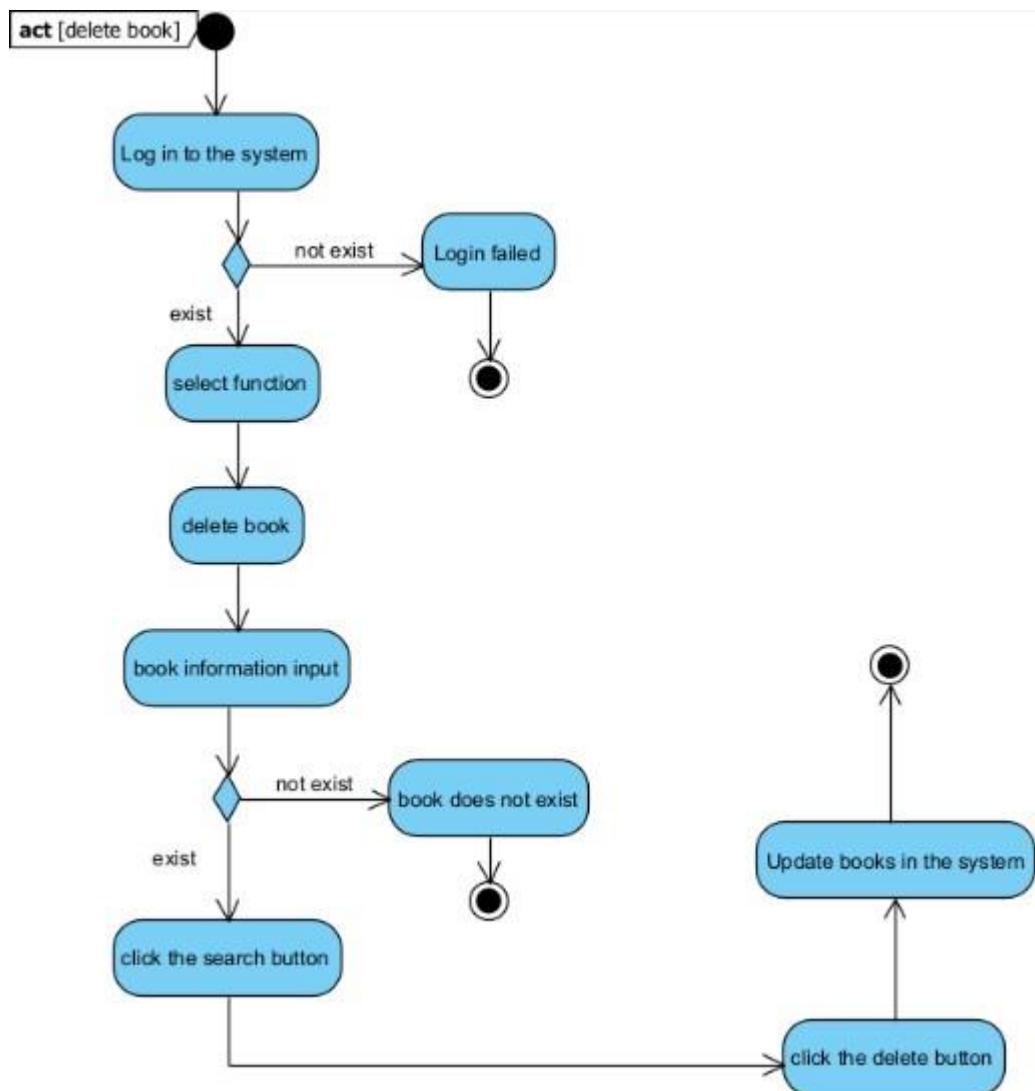
- Checkout Book:



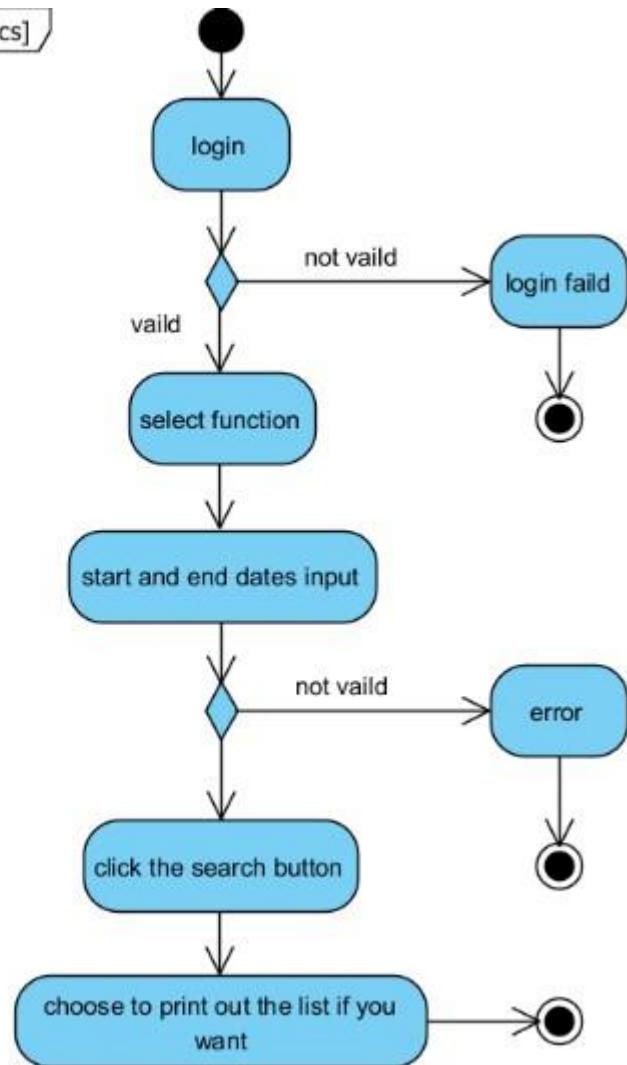
13a.5. Biểu đồ activity hệ thống





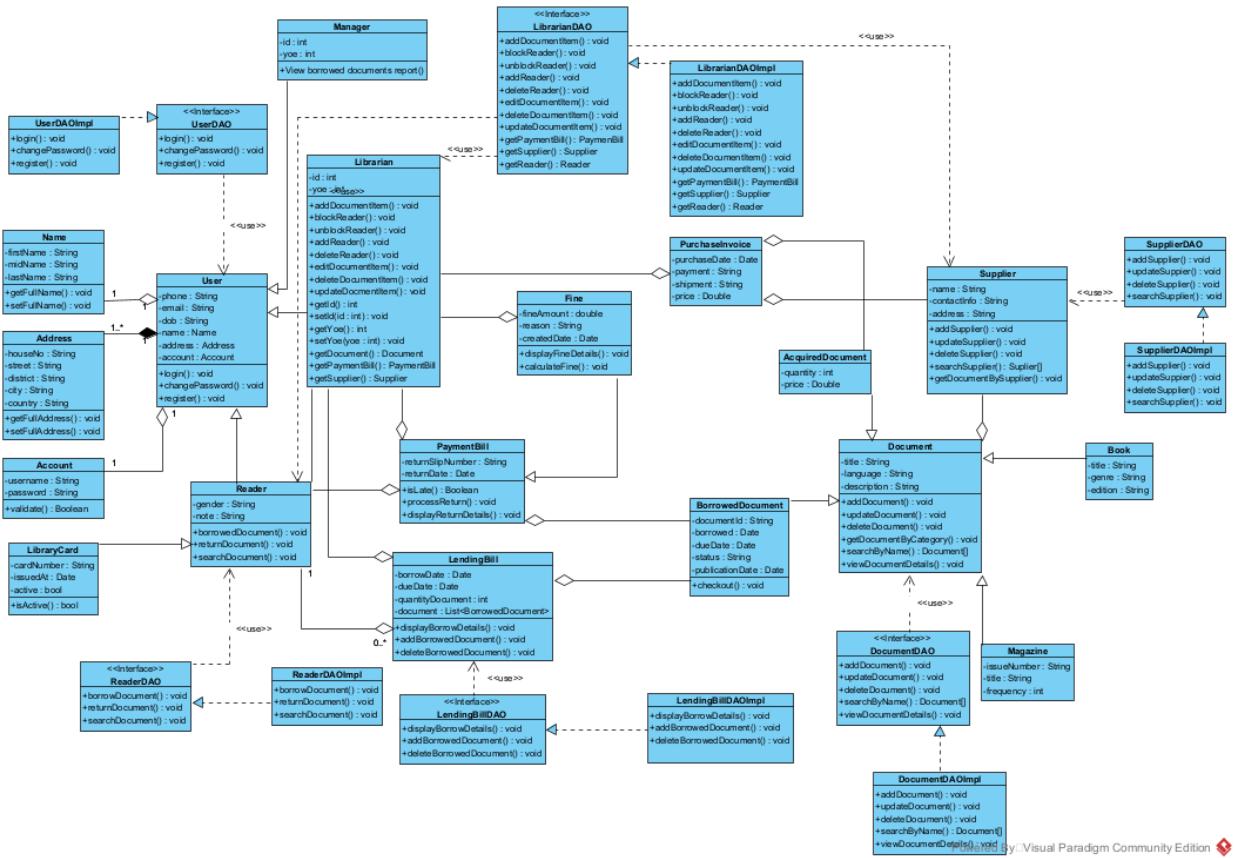


act [Borrowed book statistics]

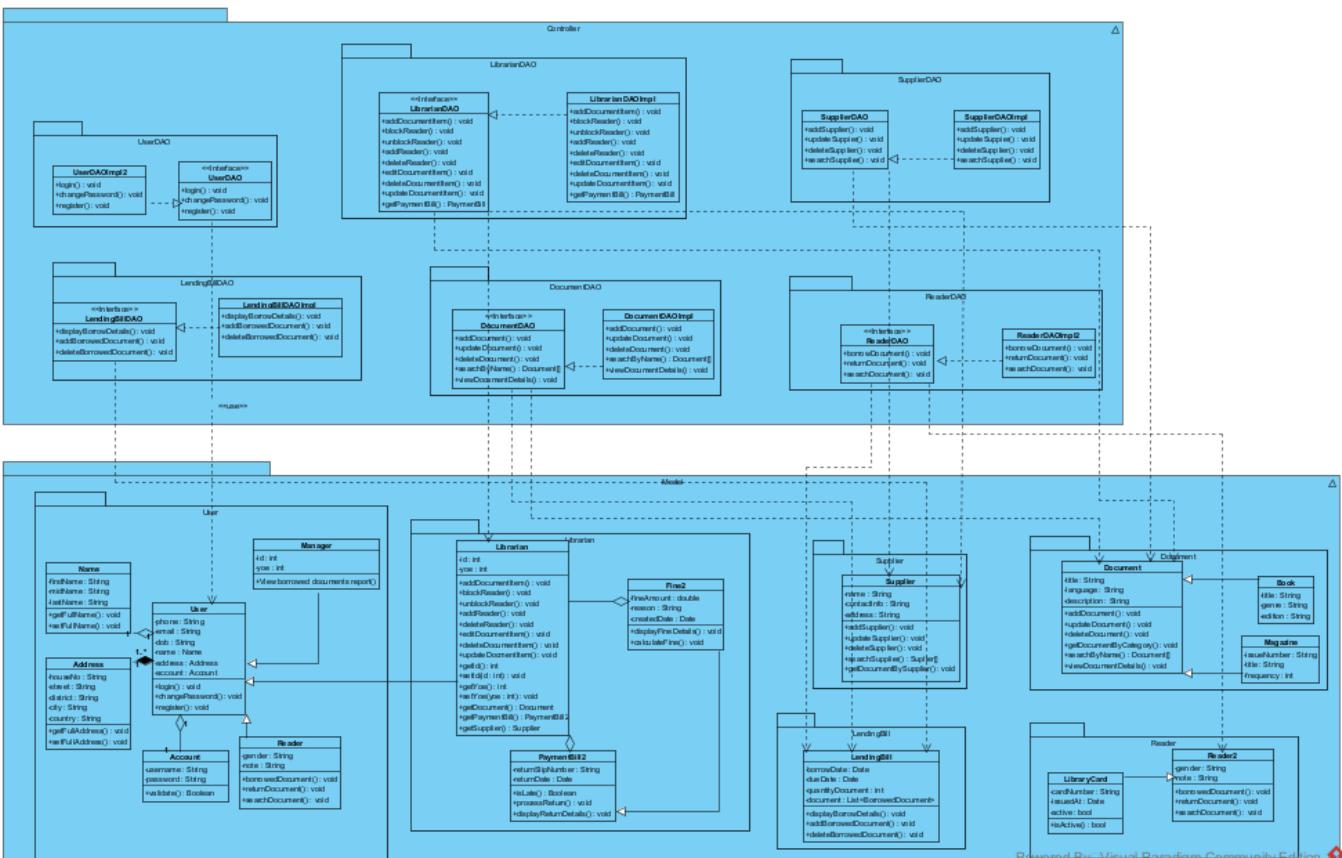


13a.6 Trình bày Biểu đồ LÓP THIẾT KẾ với DAO và layer MVC cho các Hệ thống

Sử dụng kỹ thuật DAO



Mô hình MVC



B. HỆ THỐNG THƯƠNG MẠI ĐIỆN TỬ E-COMMERCE

13b.1. Trình bày bằng ngôn ngữ tự nhiên và biểu đồ context cho các hệ thống

Hệ thống Quản lý Thương mại Điện tử (E-commerce Management System), còn được gọi là EcoMaS, là một nền tảng phần mềm đa nhiệm và tích hợp, được thiết kế để quản lý và điều hành các hoạt động trong môi trường thương mại điện tử. EcoMaS được phát triển để hỗ trợ các doanh nghiệp và cá nhân trong việc quản lý các khía cạnh khác nhau của việc kinh doanh trực tuyến.

Hệ thống này cung cấp một loạt các chức năng quản lý bao gồm quản lý sản phẩm, quản lý đơn hàng, quản lý khách hàng, quản lý thanh toán và giao hàng. Nó cho phép người dùng tạo và cập nhật thông tin sản phẩm, theo dõi tình trạng đơn hàng, quản lý thông tin khách hàng và thực hiện các giao dịch thanh toán an toàn.Thêm vào đó, EcoMaS cung cấp cơ chế tùy chỉnh cho giao diện người dùng, giúp doanh nghiệp tạo ra trải nghiệm mua sắm trực tuyến độc đáo và thú vị.

Hệ thống cũng hỗ trợ việc quản lý dữ liệu và thống kê kinh doanh, cung cấp thông tin về doanh số bán hàng, lượng truy cập trang web, tỷ lệ chuyển đổi và nhiều chỉ số quan trọng khác. Điều này giúp người dùng có cái nhìn tổng quan về hiệu suất kinh doanh của họ và đưa ra các quyết định dựa trên dữ liệu.

Tóm lại mục đích chính của hệ thống Quản lý Thương mại Điện tử (EcoMaS) là cung cấp một nền tảng phần mềm tích hợp và hiệu quả để quản lý, vận hành và tối ưu hóa các hoạt động trong môi trường thương mại điện tử. EcoMaS được thiết kế để hỗ trợ doanh nghiệp và cá nhân trong việc xây dựng, quản lý và phát triển doanh nghiệp trực tuyến một cách hiệu quả.

Phạm vi của hệ thống EcoMaS:

Phạm vi của hệ thống EcoMaS bao gồm một loạt các chức năng và tính năng để đáp ứng nhu cầu quản lý thương mại điện tử của các doanh nghiệp và người dùng cá nhân:

Quản lý Sản phẩm:

- Tạo, cập nhật và xóa thông tin sản phẩm, bao gồm hình ảnh và mô tả.
- Phân loại sản phẩm và quản lý danh mục để dễ dàng tìm kiếm và quản lý.

Quản lý Đơn hàng:

- Ghi nhận chi tiết đơn hàng, bao gồm thông tin sản phẩm, giá, số lượng và khách hàng.
- Theo dõi trạng thái xử lý đơn hàng từ khi đặt hàng đến khi giao hàng.

Quản lý Khách hàng:

- Lưu trữ thông tin cá nhân của khách hàng, lịch sử mua hàng và thông tin liên hệ.
- Hỗ trợ quản lý tài khoản khách hàng và đăng nhập.

Quản lý Thanh toán và Giao hàng:

- Hỗ trợ nhiều phương thức thanh toán an toàn và thuận tiện.
- Tích hợp tính năng tính phí vận chuyển và lựa chọn giao hàng.

Tùy chỉnh Giao diện:

- Cho phép tùy chỉnh giao diện cửa hàng trực tuyến để phản ánh thương hiệu và phong cách riêng.

Thông kê Kinh doanh:

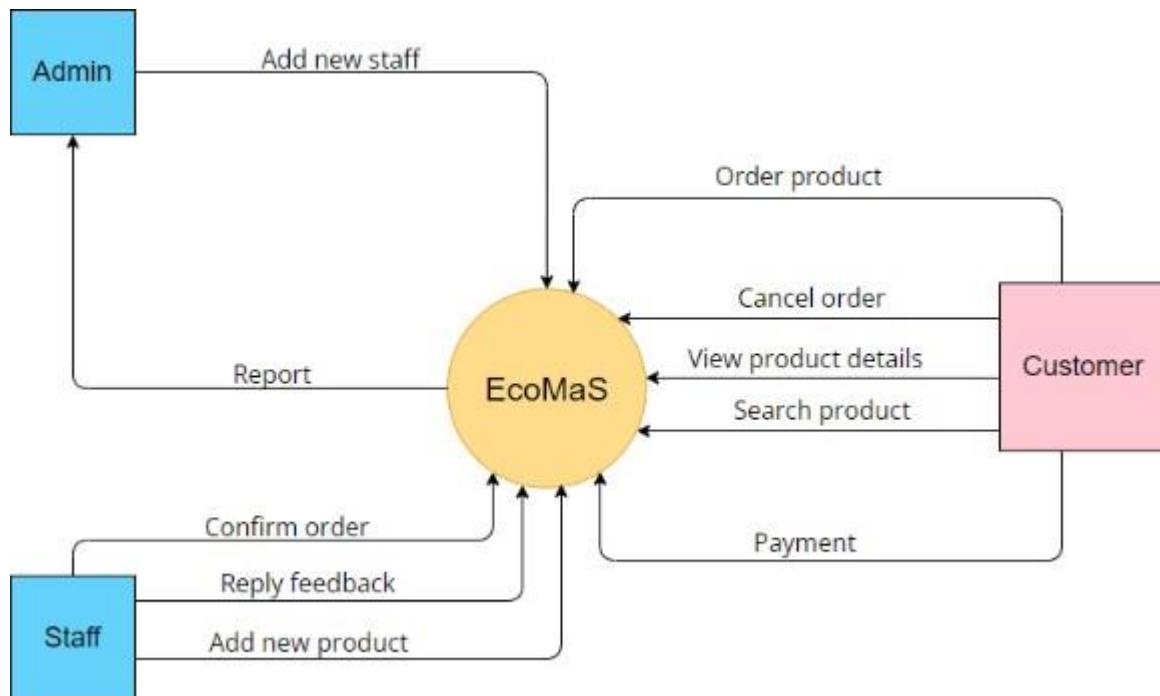
- Cung cấp thông tin chi tiết về doanh số bán hàng, lượng truy cập, tỷ lệ chuyển đổi và các chỉ số quan trọng khác.

Tích hợp và Mở rộng:

- Khả năng tích hợp với các hệ thống bên ngoài như quản lý kho và dịch vụ vận chuyển.
- Khả năng mở rộng và cải tiến hệ thống để đáp ứng nhu cầu mới trong tương lai.

Hệ thống EcoMaS có phạm vi rộng và toàn diện, nhằm mục tiêu hỗ trợ việc quản lý và vận hành các hoạt động thương mại điện tử một cách hiệu quả và linh hoạt.

Context Diagram



13b.2 Mô tả các actor

Primary actor

- Khách hàng - Customer. Đây là những người tham gia vào hoạt động mua bán và tìm kiếm sản phẩm trên hệ thống. Họ được xem như là người dùng cuối cùng của hệ thống.

Hoạt động chính:

- + Thực hiện đăng ký tài khoản và đăng nhập trên hệ thống.
- + Xem sản phẩm trên hệ thống.
- + Tìm kiếm các sản phẩm muốn mua.
- + Đặt hàng.

- + Thanh toán.
- + Theo dõi hoặc hủy đơn hàng.
- + Thực hiện đánh giá đơn hàng.

- Nhân viên hệ thống – Staff(Admin phân quyền). Đây là những người thực hiện quản lý và kiểm tra các đơn hàng trên hệ thống cũng như các dịch vụ liên quan đến khách hàng.
- Hoạt động chính:
 - + Đăng nhập bằng tài khoản nhân viên trên hệ thống.
 - + Xác nhận các đơn đặt hàng.
 - + Thêm, sửa, xóa thông tin sản phẩm trên hệ thống.
 - + Xác nhận hủy đơn hàng khi gấp vận đề.
 - + Xem và tư vấn các đánh giá của khách hàng.
- Quản trị viên của hệ thống - Admin. Họ là những người quản lý và vận hành hệ thống.
Hoạt động chính:
 - + Quản lý tài khoản người dùng trên hệ thống.
 - + Quản lý tài khoản nhân viên trên hệ thống.
 - + Thêm nhân viên mới.
 - + Xóa nhân viên đã nghỉ.
 - + Xem các báo cáo trên hệ thống.
 - + Quản lý tất cả các sản phẩm ở trên hệ thống.

Secondary actor

- Thanh toán trung gian – bank: Đây là người thực hiện các yêu cầu thanh toán có trên hệ thống từ thanh toán online hoặc trả tiền mặt. Họ có nhiệm vụ thực hiện thanh toán và xuất hóa đơn cho khách hàng
- Nhà Cung Cấp (Supplier):

Nhà Cung Cấp là một trong các actor quan trọng trong hệ thống Quản lý Thương

mại Điện tử (EcoMaS). Nhà Cung Cấp tương tác với hệ thống để cung cấp thông tin sản phẩm, quản lý lượng tồn kho và xác nhận xử lý đơn hàng từ khách hàng.

- Nhân Viên Giao Hàng (Delivery Personnel):

- Là người thực hiện việc giao hàng tới khách hàng.
- Nhận thông tin về đơn hàng cần giao và cập nhật trạng thái giao hàng.
- Tương tác với chức năng Quản lý Đơn hàng và Cập nhật trạng thái giao hàng.

- Người Phân Tích Kinh Doanh (Business Analyst):

- Là người dùng có nhiệm vụ phân tích dữ liệu kinh doanh và tạo báo cáo.
- Truy cập vào thông tin thống kê kinh doanh và tạo báo cáo để đưa ra quyết định liên quan đến chiến lược kinh doanh.
- Tương tác với chức năng Thống kê Kinh doanh và Tạo báo cáo.

13b.3 Mô tả yêu cầu các chức năng và phi chức năng của hệ thống bằng ngôn ngữ tự nhiên và dạng Bảng

| | |
|-------|--|
| Admin | Đăng nhập Đổi mật khẩu |
| | Lấy lại mật khẩu |
| | Quản lý Sản phẩm: <ul style="list-style-type: none">- Tạo, cập nhật và xóa thông tin sản phẩm, bao gồm hình ảnh và mô tả.- Phân loại sản phẩm và quản lý danh mục để dễ dàng tìm kiếm và quản lý. |
| | Quản lý Đơn hàng: <ul style="list-style-type: none">- Ghi nhận chi tiết đơn hàng, bao gồm thông tin sản phẩm, giá, số lượng và khách hàng.- Theo dõi trạng thái xử lý đơn hàng từ khi đặt hàng đến khi giao hàng. |

| | |
|------------|--|
| | <p>Quản lý tài khoản:</p> <ul style="list-style-type: none"> - Lưu trữ thông tin cá nhân của khách hàng, lịch sử mua hàng và thông tin liên hệ. <p>Hỗ trợ quản lý tài khoản khách hàng và đăng nhập.</p> <p>Tạo và xem báo cáo thống kê kinh doanh</p> |
| | <p>Đăng nhập Đổi mật khẩu</p> <p>Lấy lại mật khẩu</p> <p>Duyệt và Tìm Kiếm Sản Phẩm:</p> <ul style="list-style-type: none"> - Khách Hàng có khả năng duyệt qua danh sách sản phẩm, sử dụng các bộ lọc và tìm kiếm để tìm sản phẩm cụ thể. <p>Thêm Sản Phẩm vào Giỏ Hàng:</p> <ul style="list-style-type: none"> - Khách Hàng có thể thêm sản phẩm vào giỏ hàng bằng cách chọn số lượng và nhấn nút "Thêm vào giỏ hàng". <p>Xem Giỏ Hàng: Khách Hàng có thể xem danh sách sản phẩm trong giỏ hàng, điều</p> |
| Khách hàng | |

| | |
|--------------------|---|
| | <p>chỉnh số lượng hoặc xóa sản phẩm khỏi giỏ.</p> |
| | <p>Đặt Hàng: Sau khi xem giỏ hàng và thực hiện thanh toán,</p> <p>Khách Hàng hoàn tất việc đặt hàng và gửi yêu cầu đến hệ thống.</p> |
| | <p>Theo Dõi Đơn Hàng: Khách Hàng có thể xem tình trạng đơn hàng, từ khi đặt hàng đến khi giao hàng thành công</p> |
| | <p>Xem Lịch Sử Mua Hàng: Khách Hàng có thể xem lịch sử các đơn hàng đã thực hiện trong quá khứ</p> |
| | <p>Liên Hệ Hỗ Trợ: Khách Hàng có khả năng liên hệ với dịch vụ hỗ trợ khách hàng để giải quyết các vấn đề hoặc đặt câu hỏi liên quan đến sản phẩm và đơn hàng.</p> |
| Khách hàng | <p>Đăng ký</p> |
| | <p>Đăng nhập Đổi mật khẩu</p> |
| | <p>Lấy lại mật khẩu</p> |
| | <p>Đặt trước các cuốn sách mượn mượn</p> |
| | <p>Gia hạn thời gian mượn sách</p> |
| | <p>Yêu cầu mượn 1 cuốn sách mới</p> |
| | <p>Trả tiền phạt khi trả sách muộn</p> |
| Hệ thống thông báo | <p>Gửi thông báo đến khách hàng, bao gồm thông báo về sản phẩm mới có thể khách hàng sẽ thích, thông báo đơn hàng</p> |

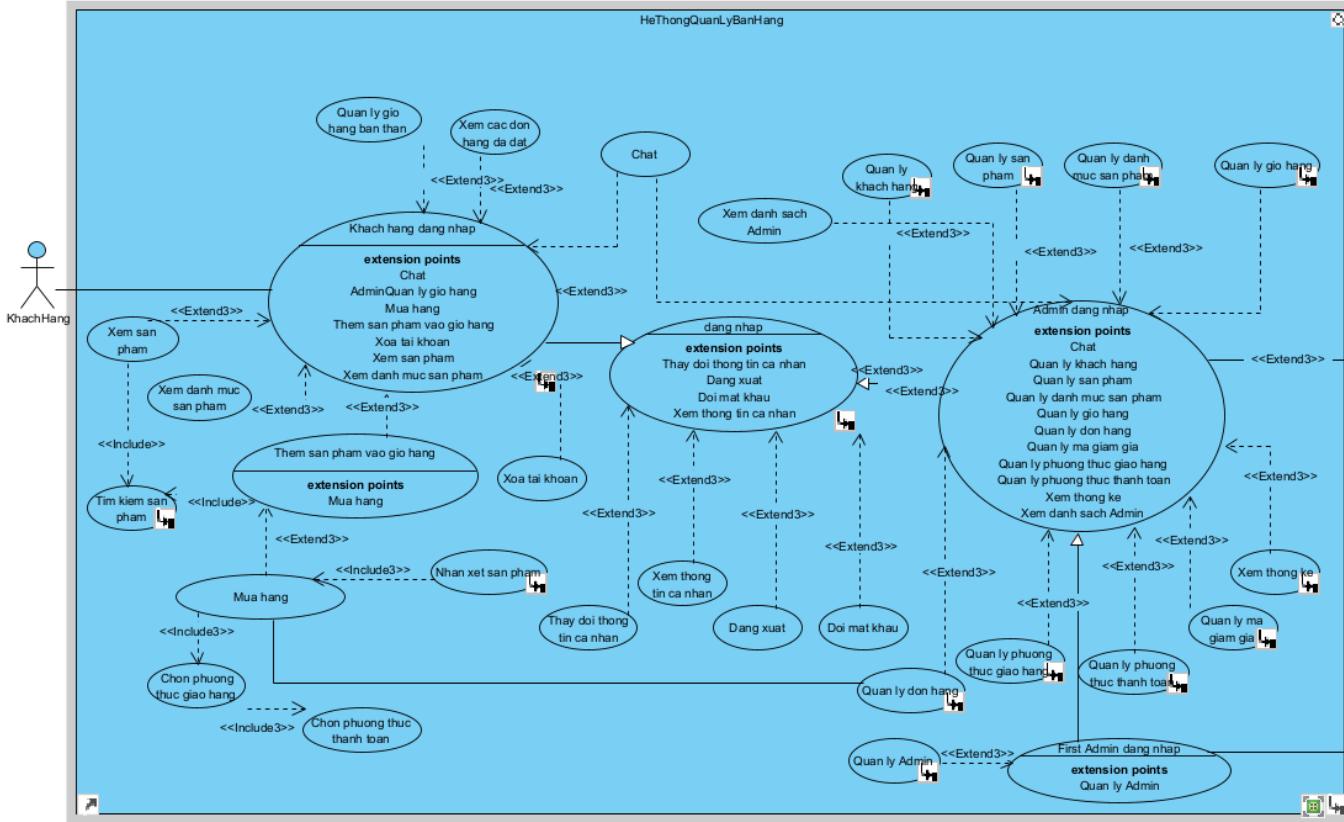
| | |
|---------------------|---|
| | chuẩn bị được giao tới nơi,... |
| Hệ thống thanh toán | Xử lý các giao dịch thanh toán từ người mượn, bao gồm thanh toán tiền mượn và tiền phạt |
| Hệ thống hiển thị | Hiển thị giao diện người dùng |

Yêu cầu phi chức năng quan trọng cho hệ thống thương mại điện tử:

- Bảo mật:

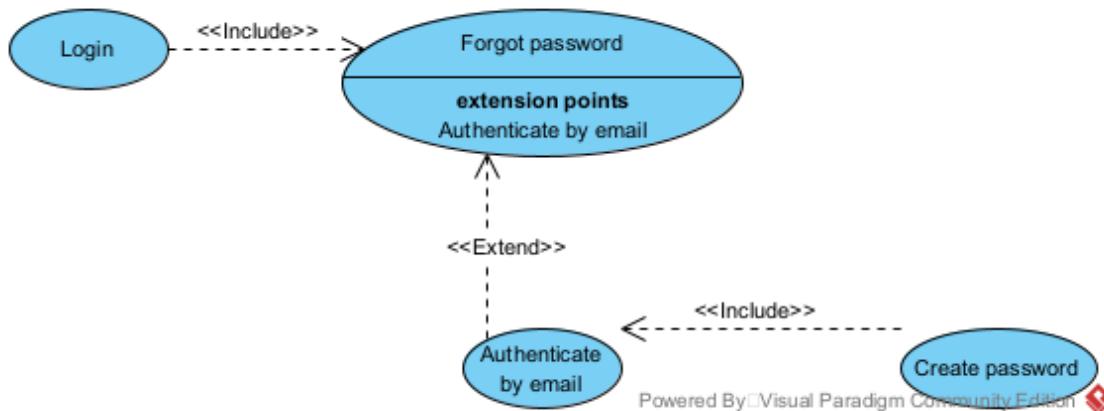
- + Bảo vệ dữ liệu cá nhân và tài khoản người dùng.
- + Bảo vệ trước các cuộc tấn công mạng như tấn công DDoS hoặc SQL injection.
- Hiệu suất: Đảm bảo hệ thống có khả năng chịu tải để xử lý số lượng lớn người dùng và giao dịch cùng một lúc, đặc biệt trong các sự kiện thời trang hoặc khuyến mãi.
- Tích hợp dễ dàng: Hỗ trợ các API mở để tích hợp với các hệ thống bên ngoài, chẳng hạn như hệ thống thanh toán, hệ thống vận chuyển, hoặc cơ sở dữ liệu khách hàng.
- Dễ sử dụng: Giao diện người dùng thân thiện và dễ hiểu.
- Hỗ trợ tìm kiếm sản phẩm dễ sử dụng và hệ thống lọc sản phẩm.
- Thời gian hoạt động 24/7: Hệ thống phải có thời gian hoạt động liên tục mà không cần tắt để bảo trì.
- Phân quyền người dùng: Điều này đảm bảo rằng người dùng có quyền truy cập và thực hiện các chức năng dựa trên vai trò của họ (quản trị viên, người dùng cuối, người bán, v.v.).

13b.4. Trình bày Biểu đồ use case THEO 2 LEVEL cho các Hệ thống

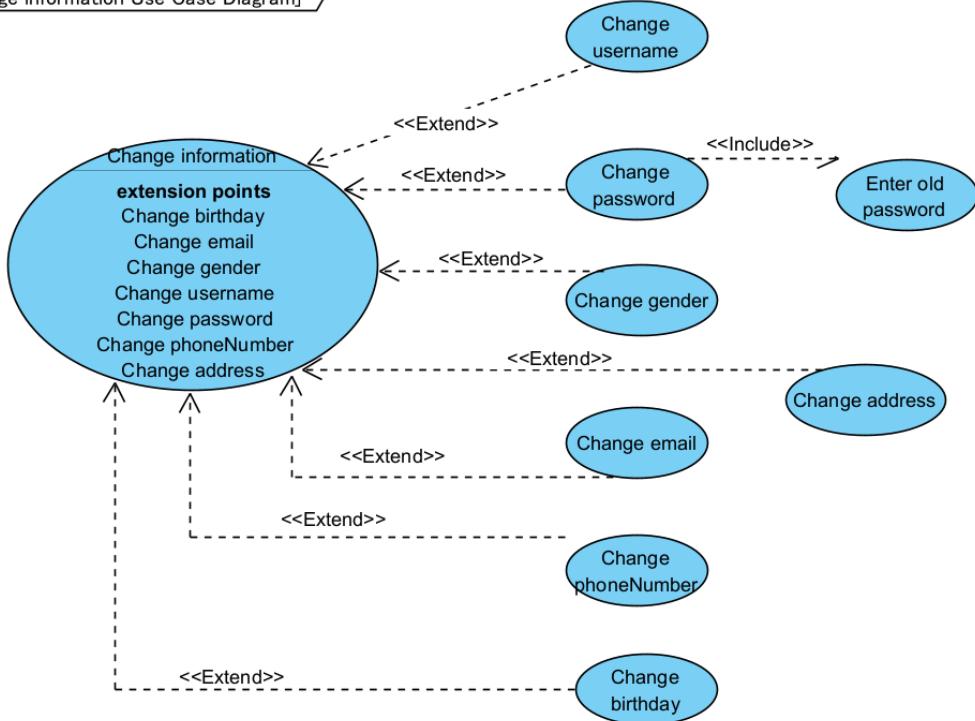


Admin:

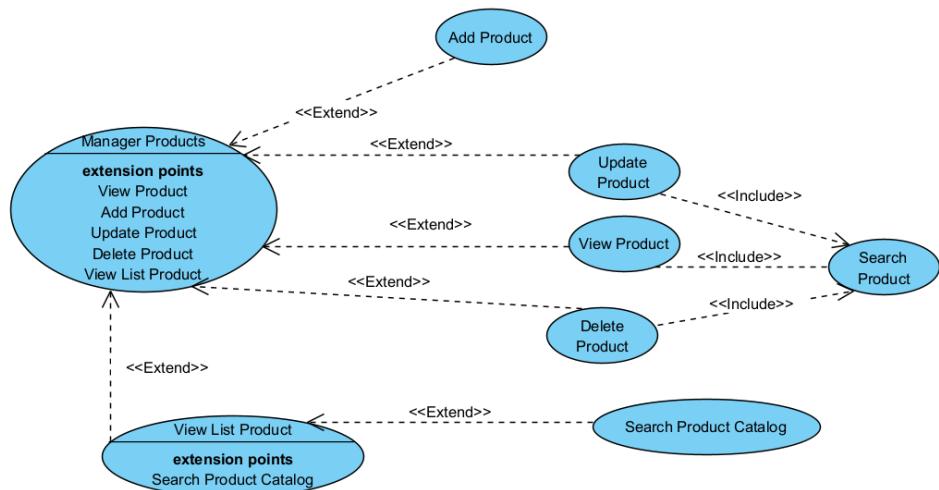
Forgot password usecase diagram:



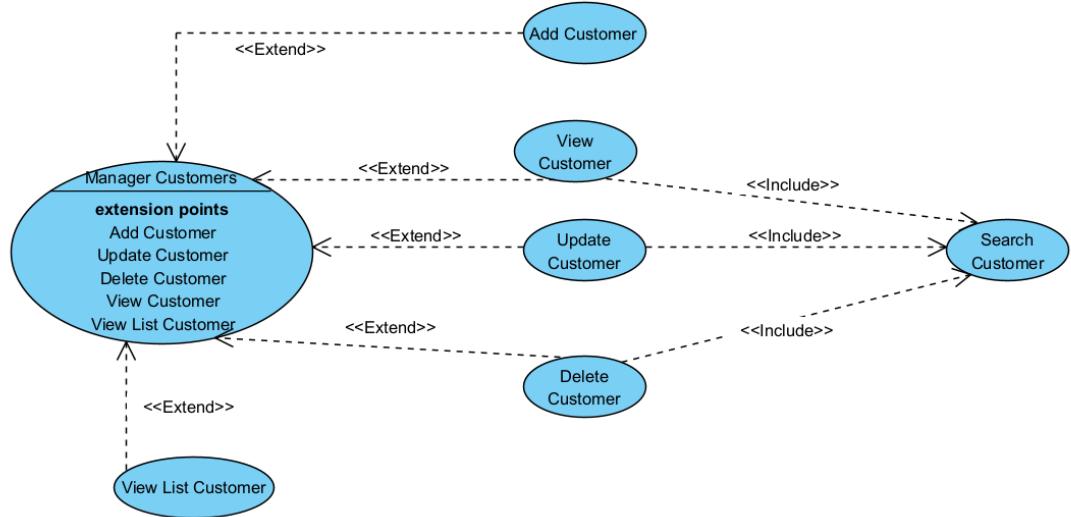
uc [Change information Use Case Diagram]



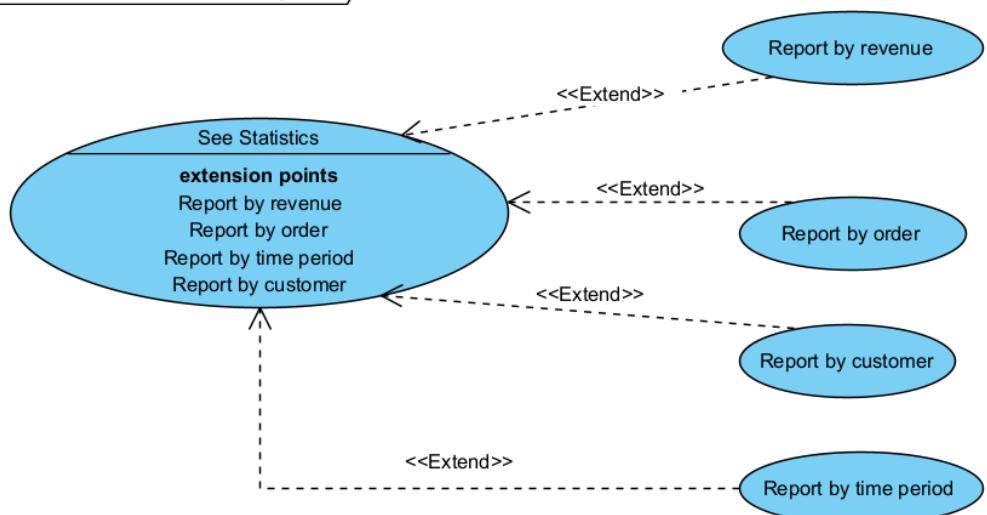
uc [Manager Products Use Case Diagram]



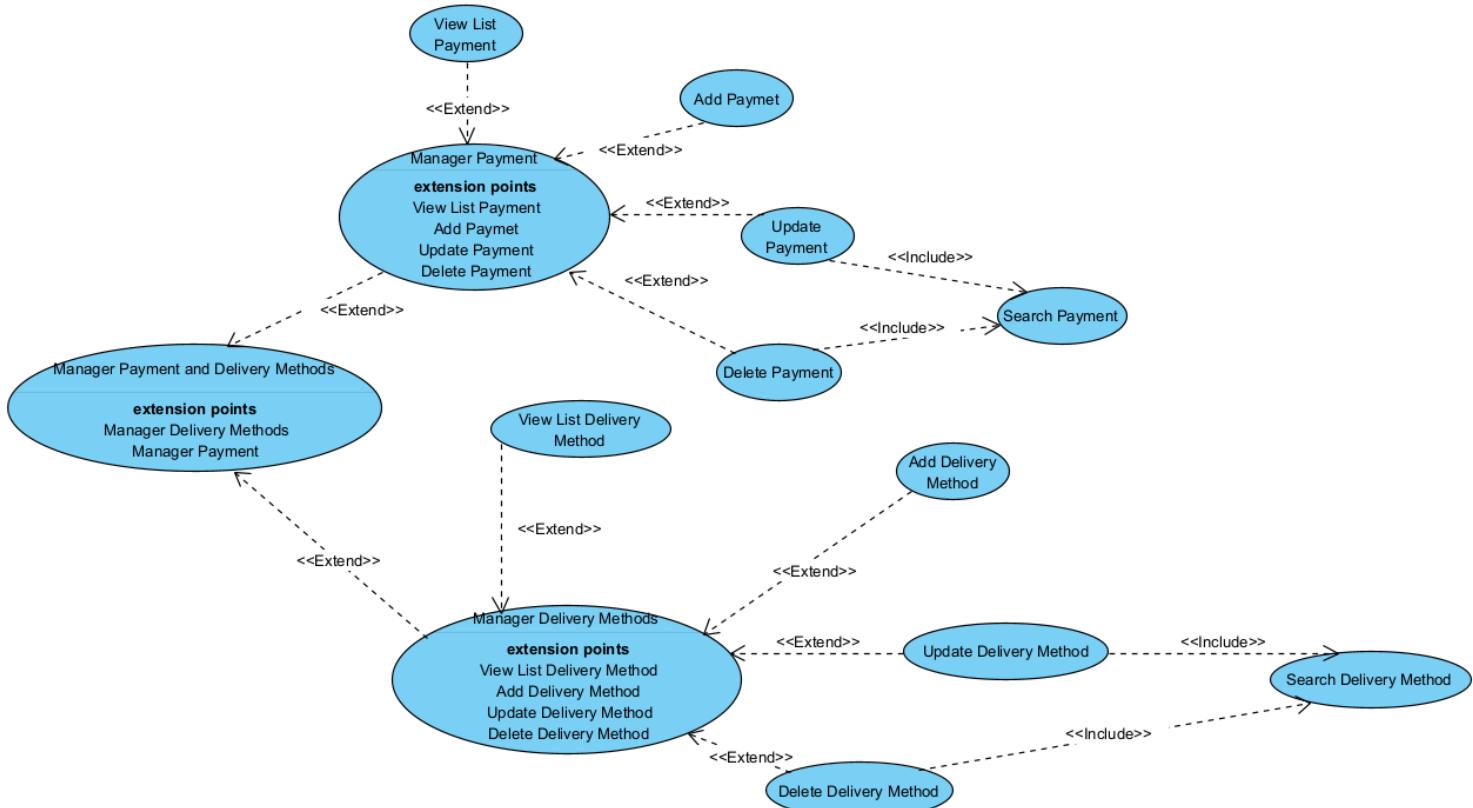
uc [Manager Customers Use Case Diagram]



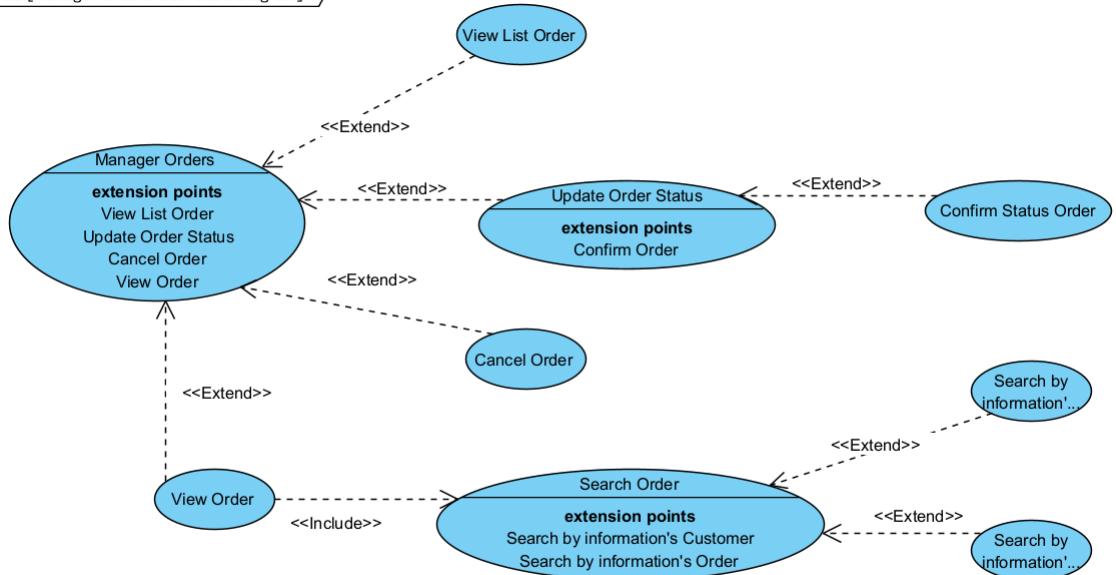
uc [See Statistics Use Case Diagram]



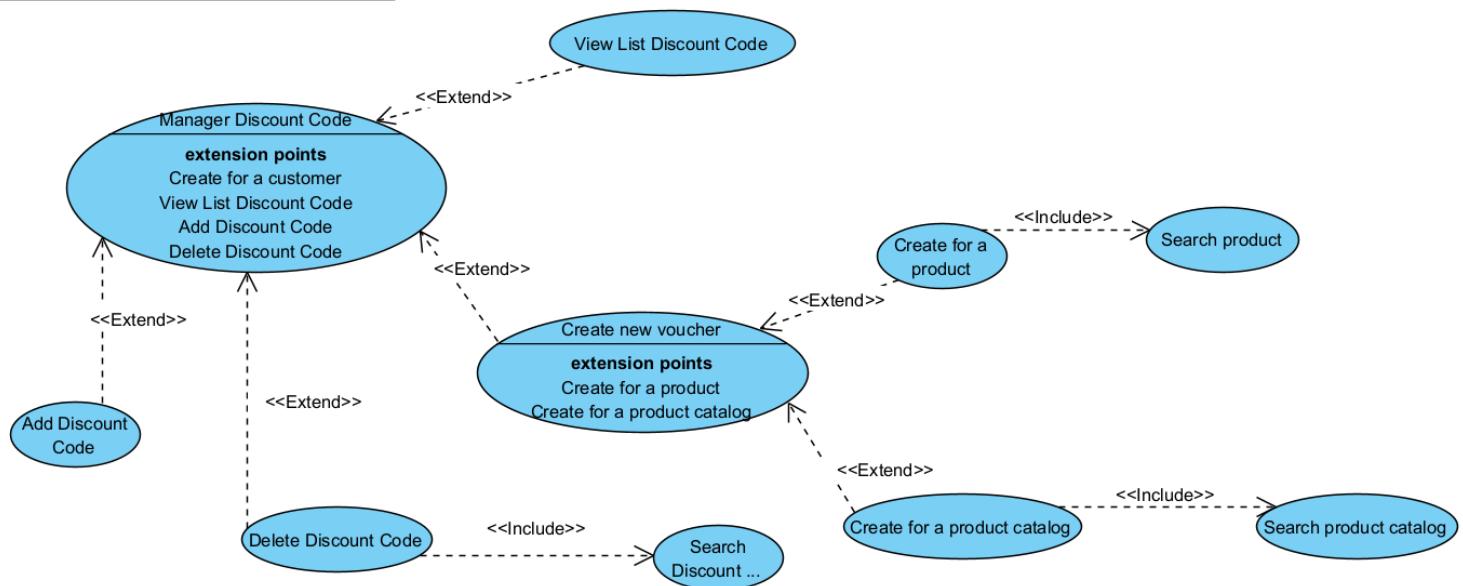
uc [Manager Payment and delivery methods Use Case Diagram]



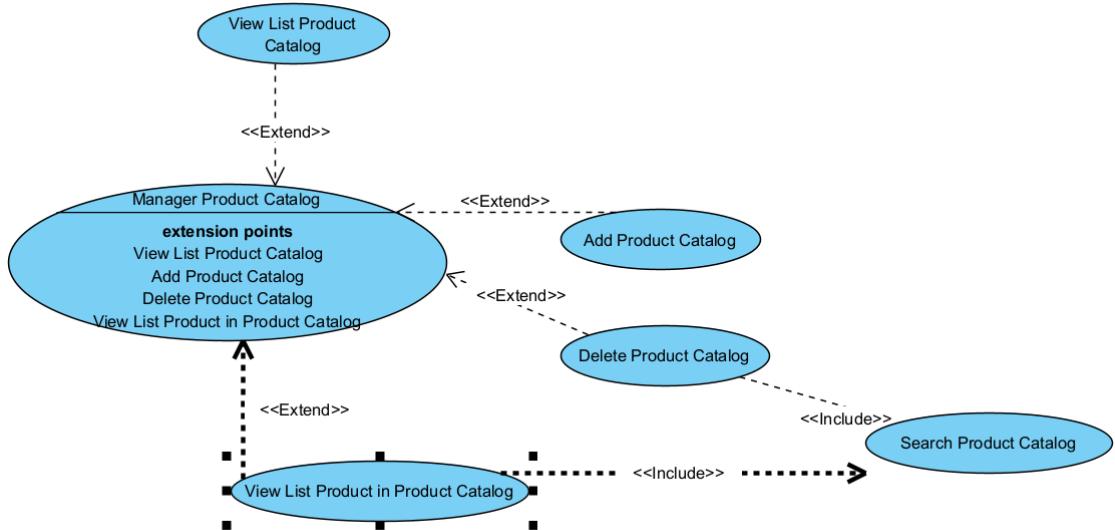
uc [Manager Orders Use Case Diagram]



uc [Create Discount Code Use Case Diagram]

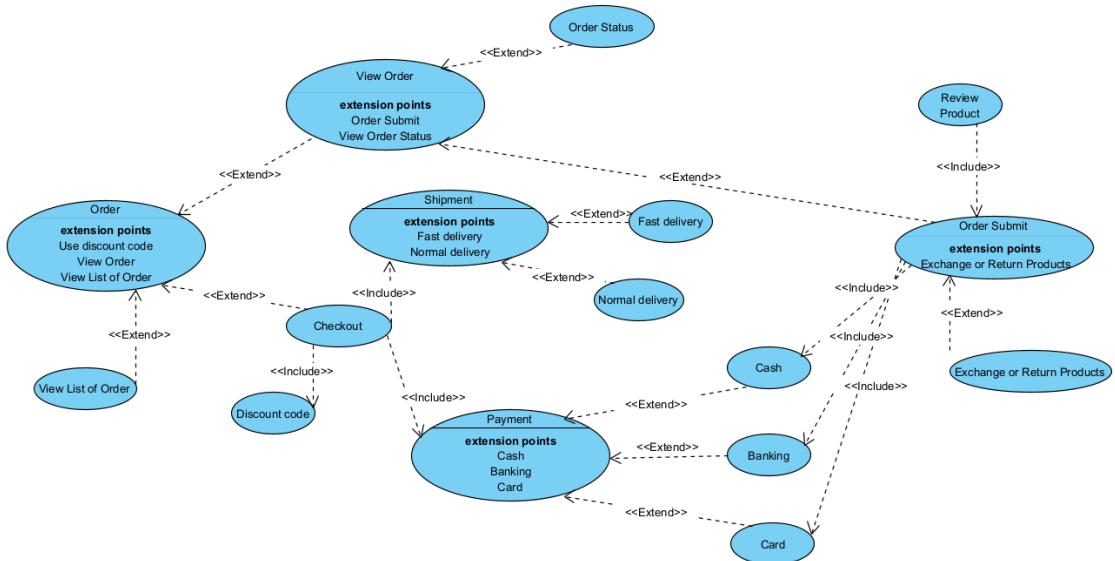


uc [Manager Product Catalog Use Case Diagram]

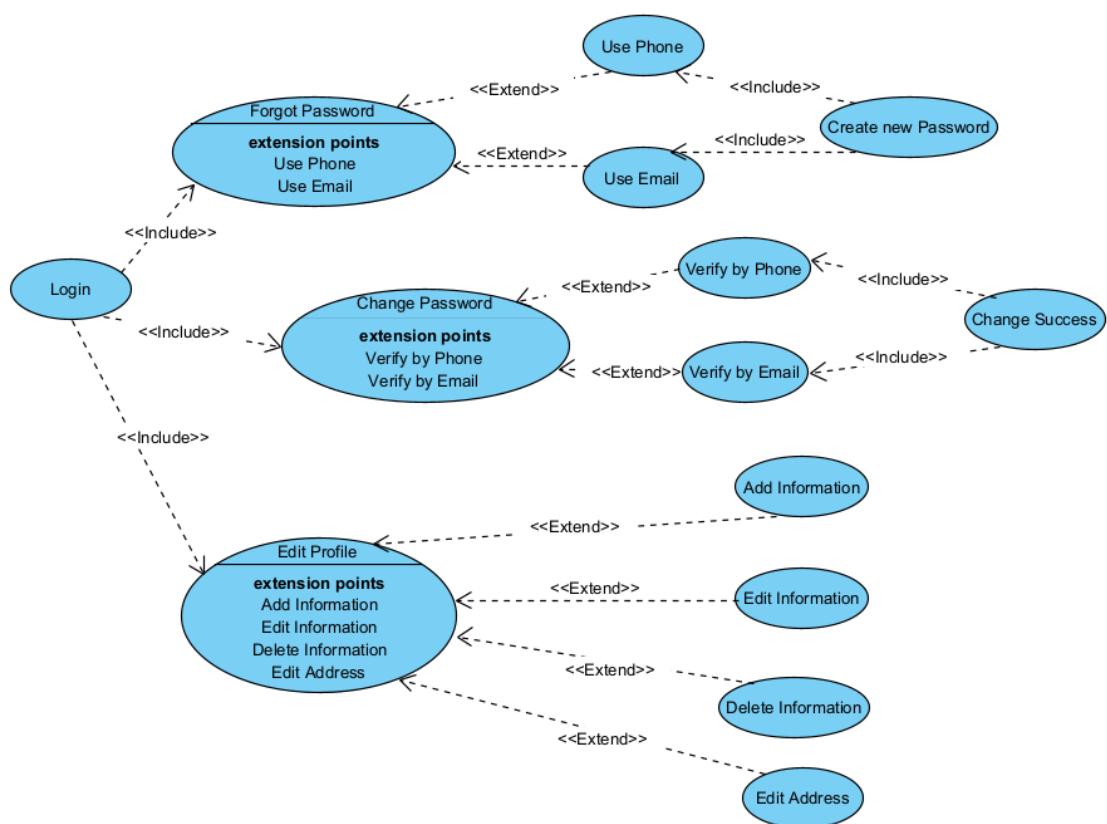


Customer:

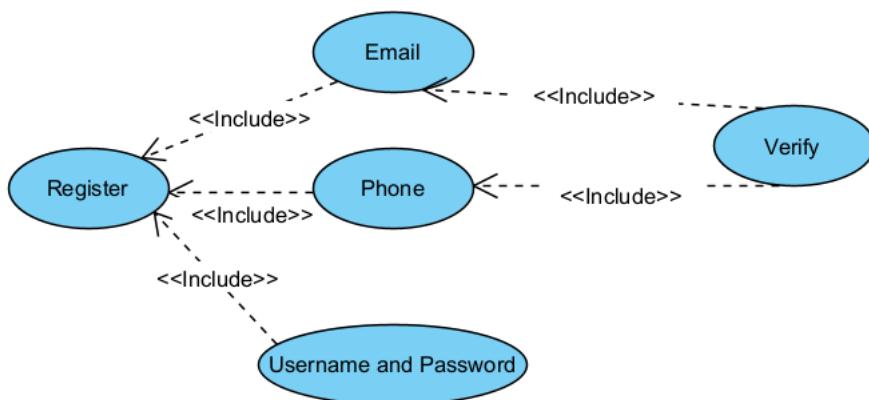
uc [Order Use Case Diagram]



uc [Login Use Case Diagram]



uc [Register Use Case Diagram]

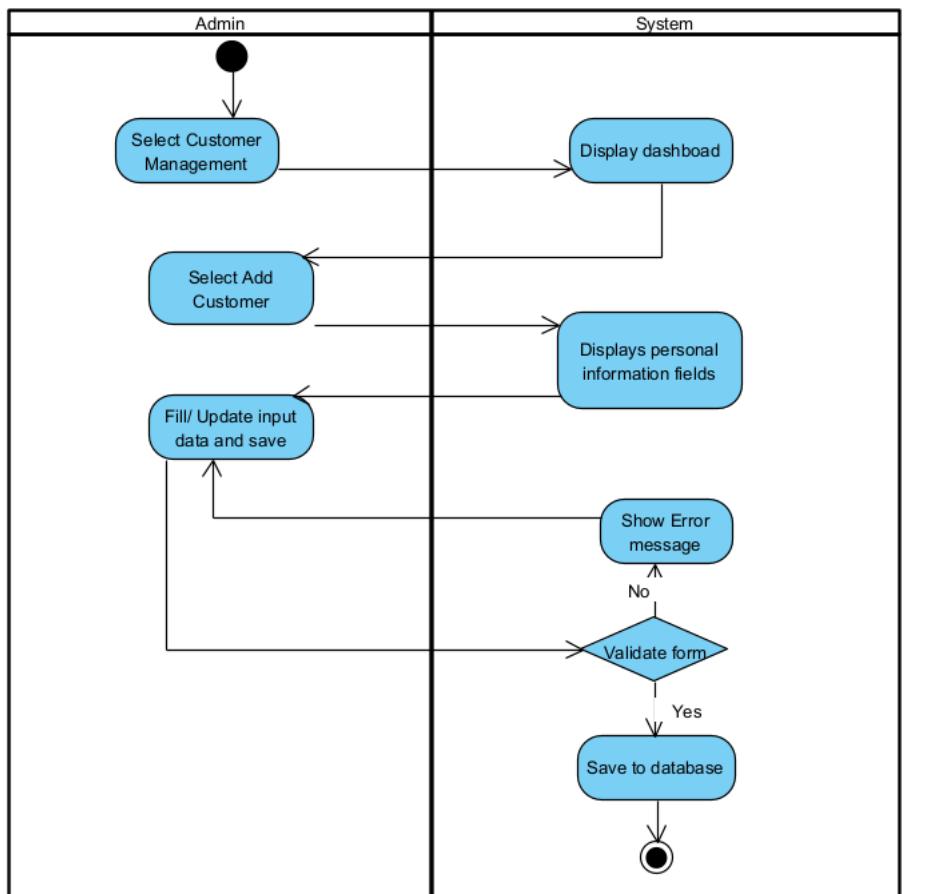


13b.5 Biểu đồ activity

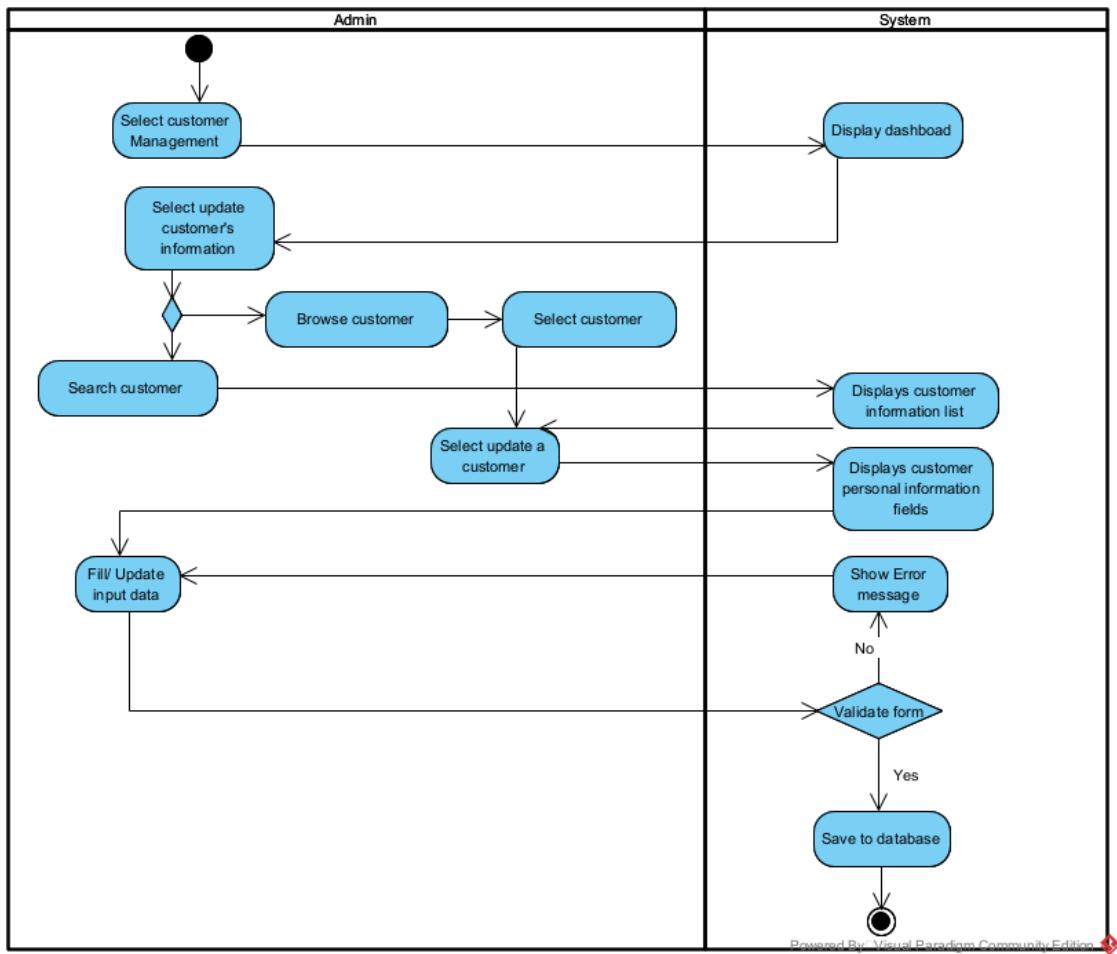
Admin:

Add customer

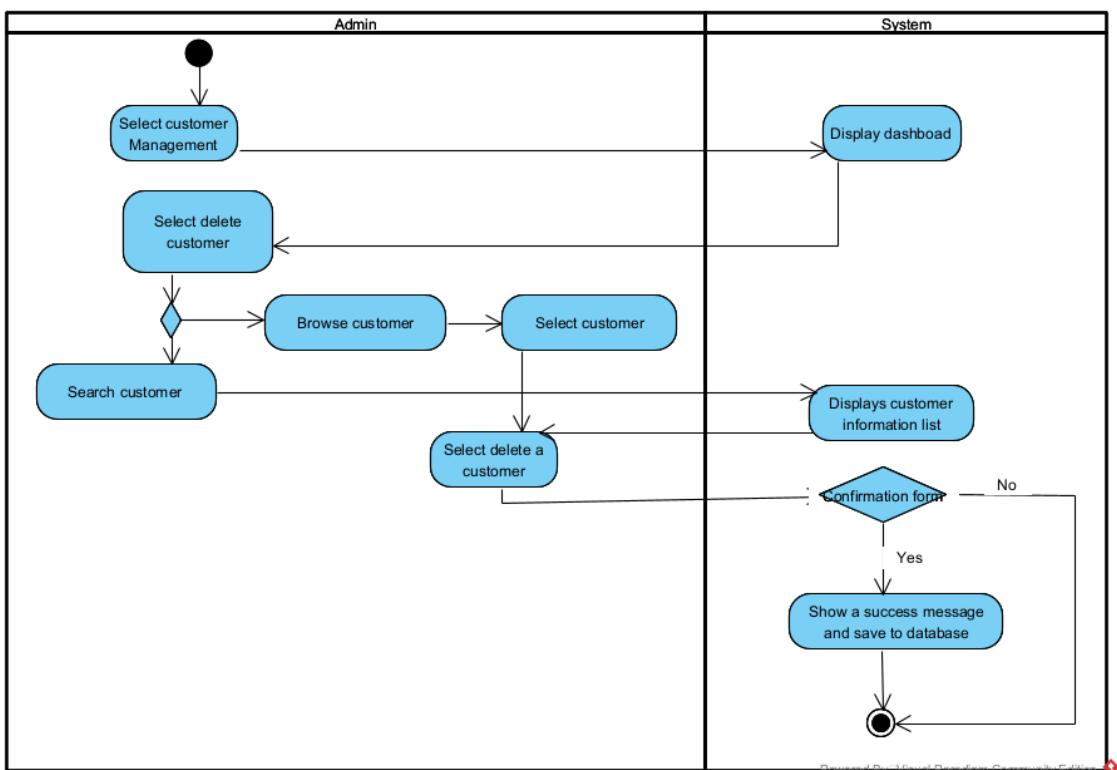
act [Add customer]



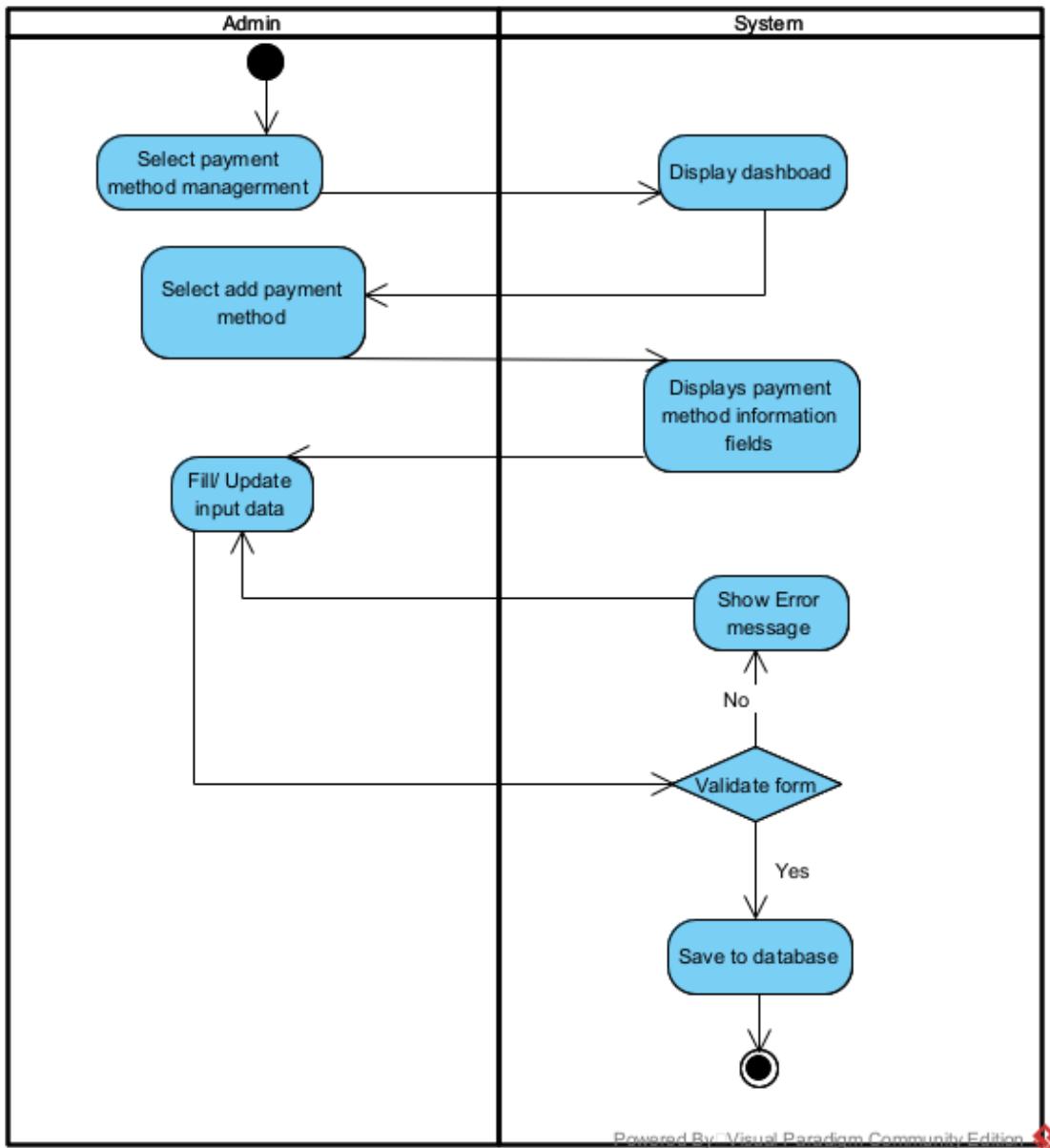
Sửa thông tin khách hàng:



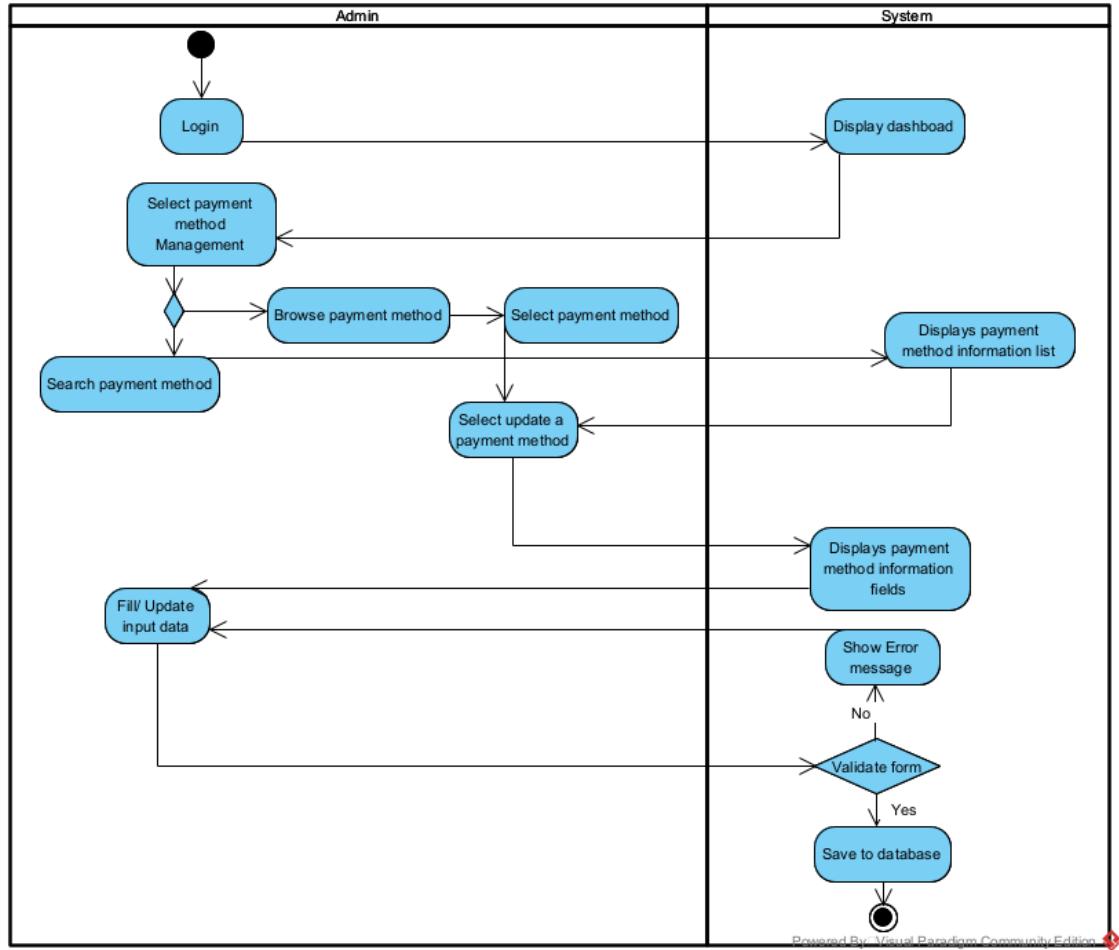
Xóa khách hàng



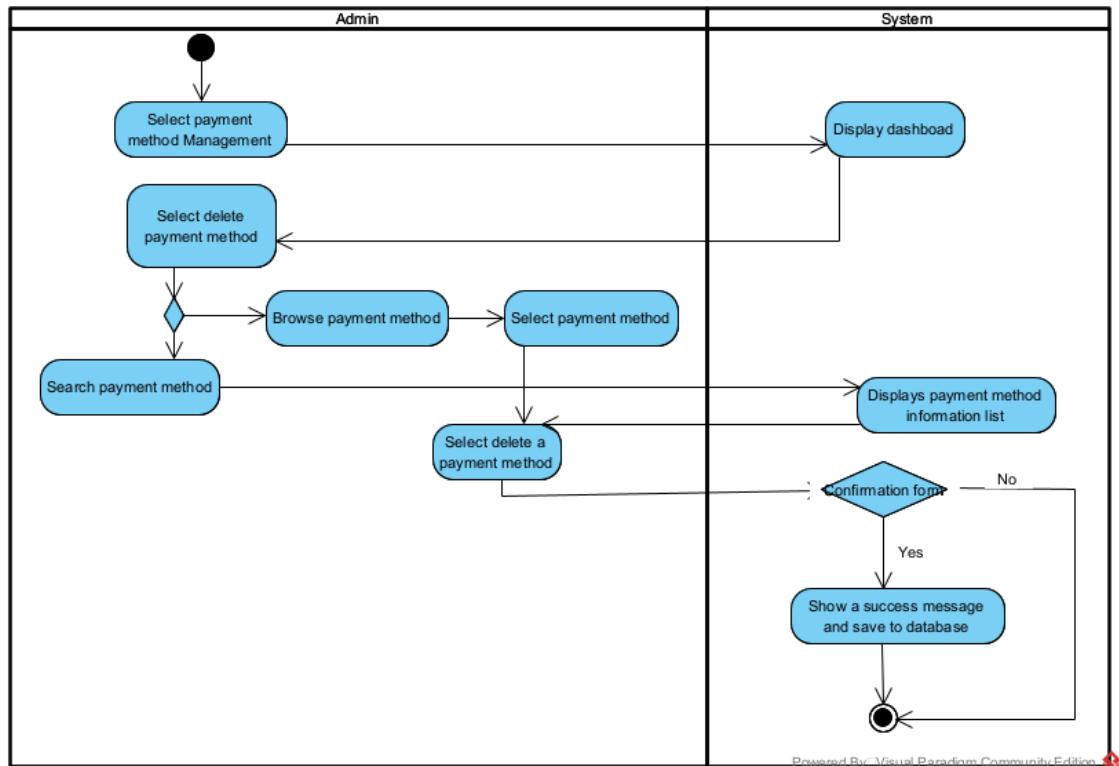
Thêm phương thức thanh toán



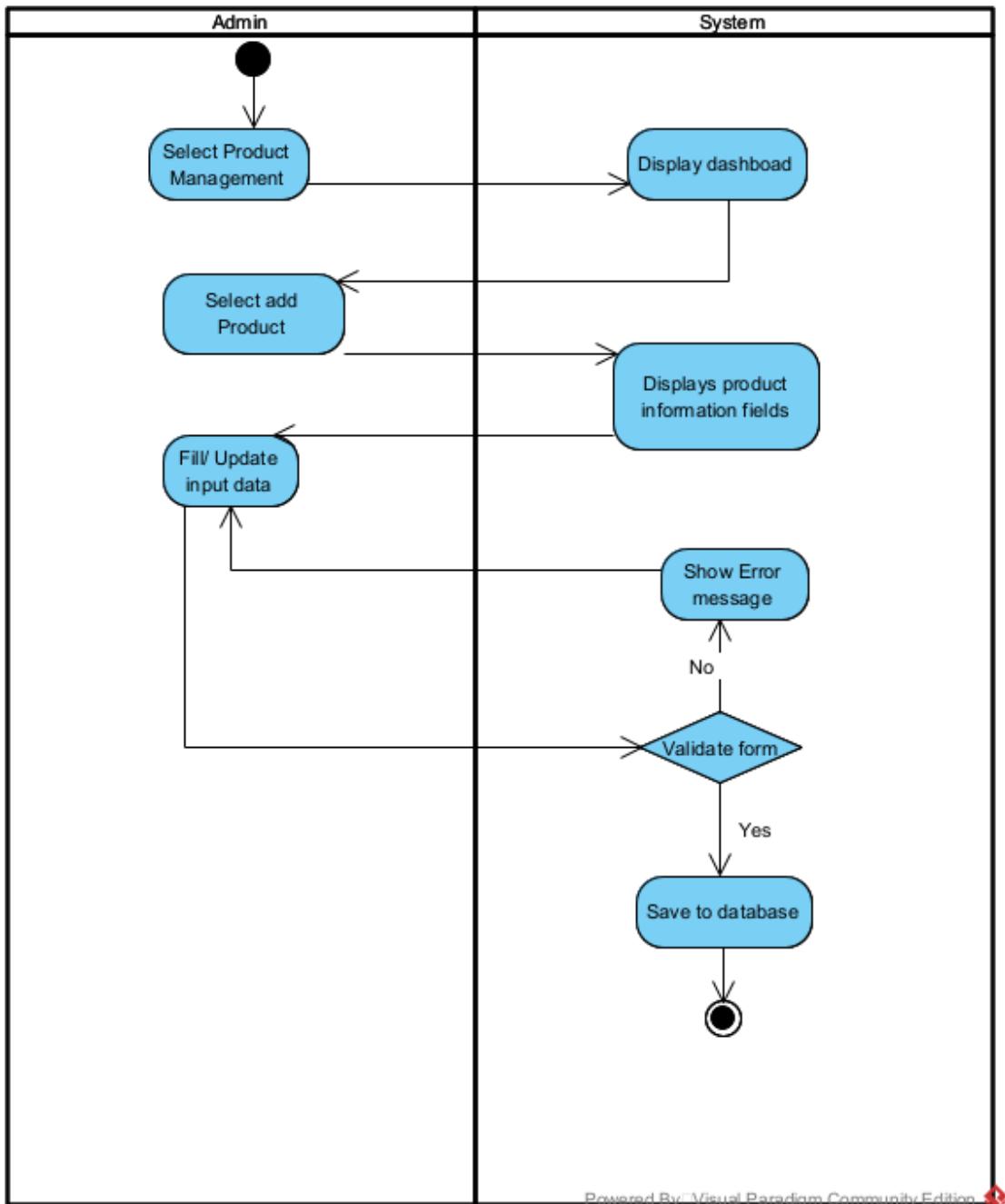
Sửa phương thức thanh toán



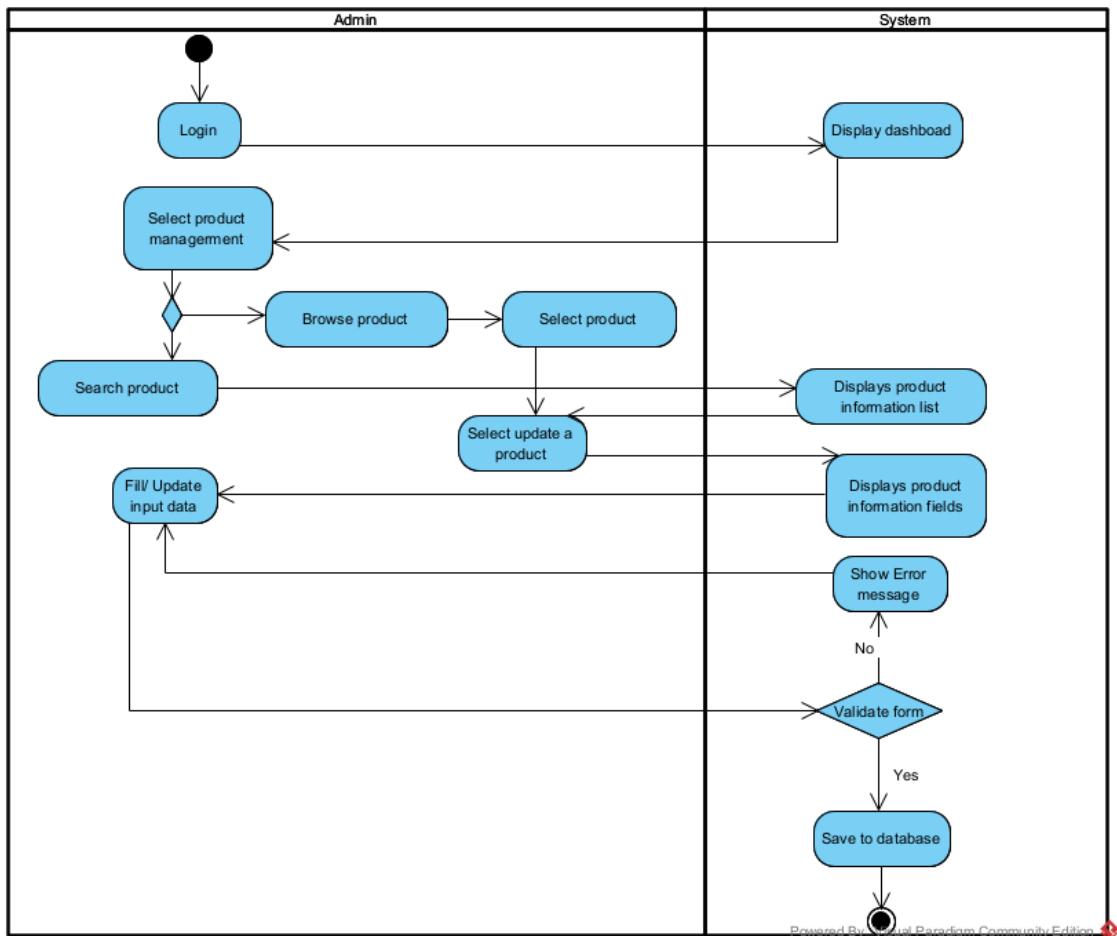
Xóa phương thức thanh toán:



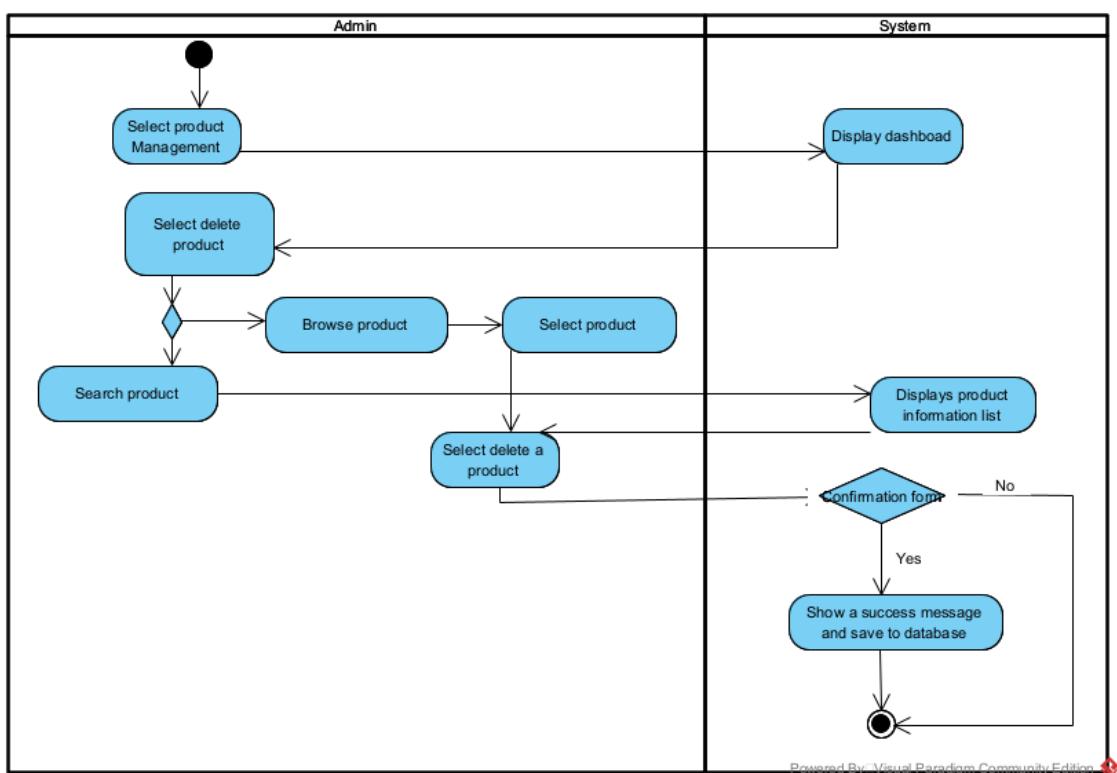
Thêm sản phẩm:



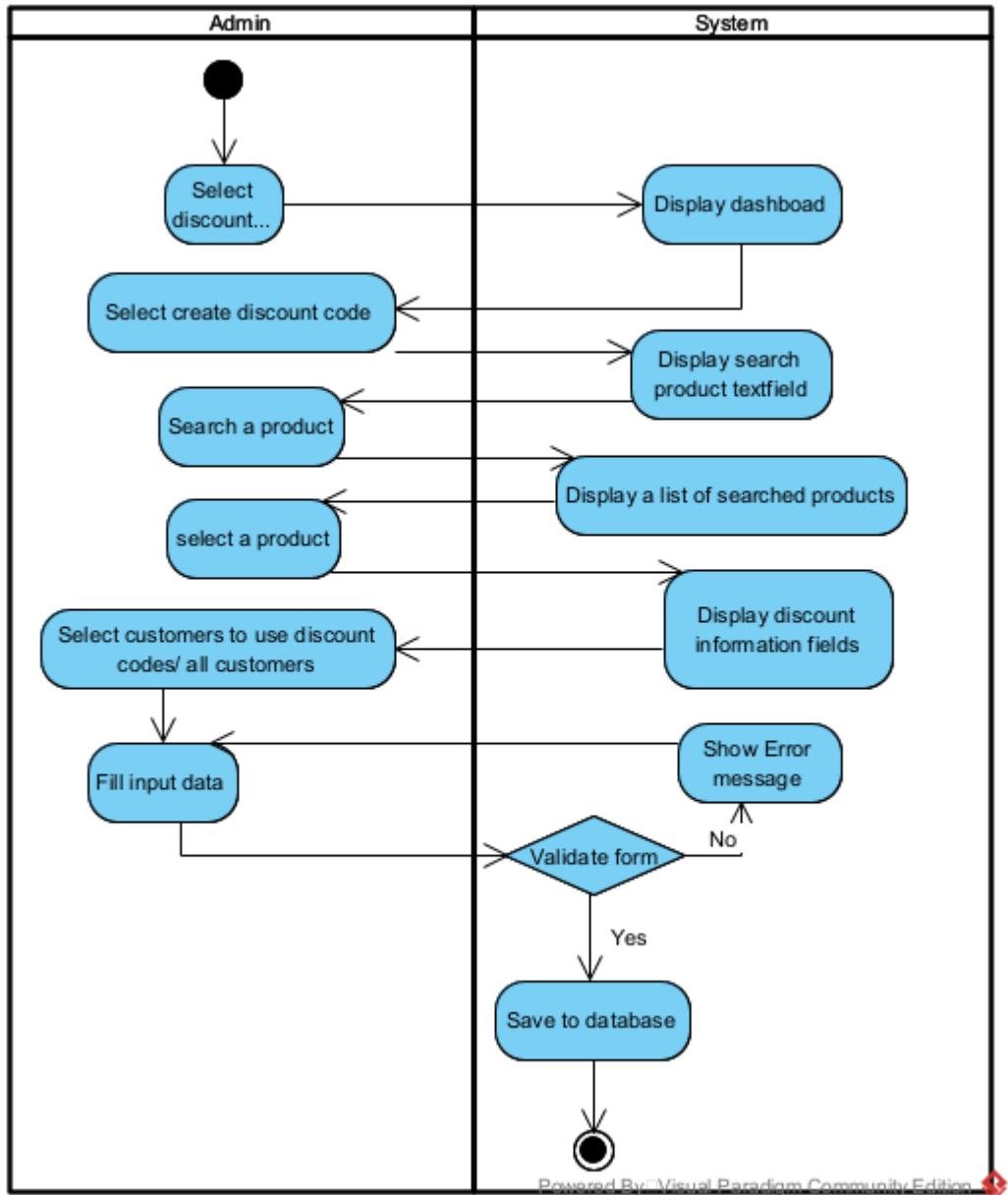
Sửa thông tin sản phẩm:



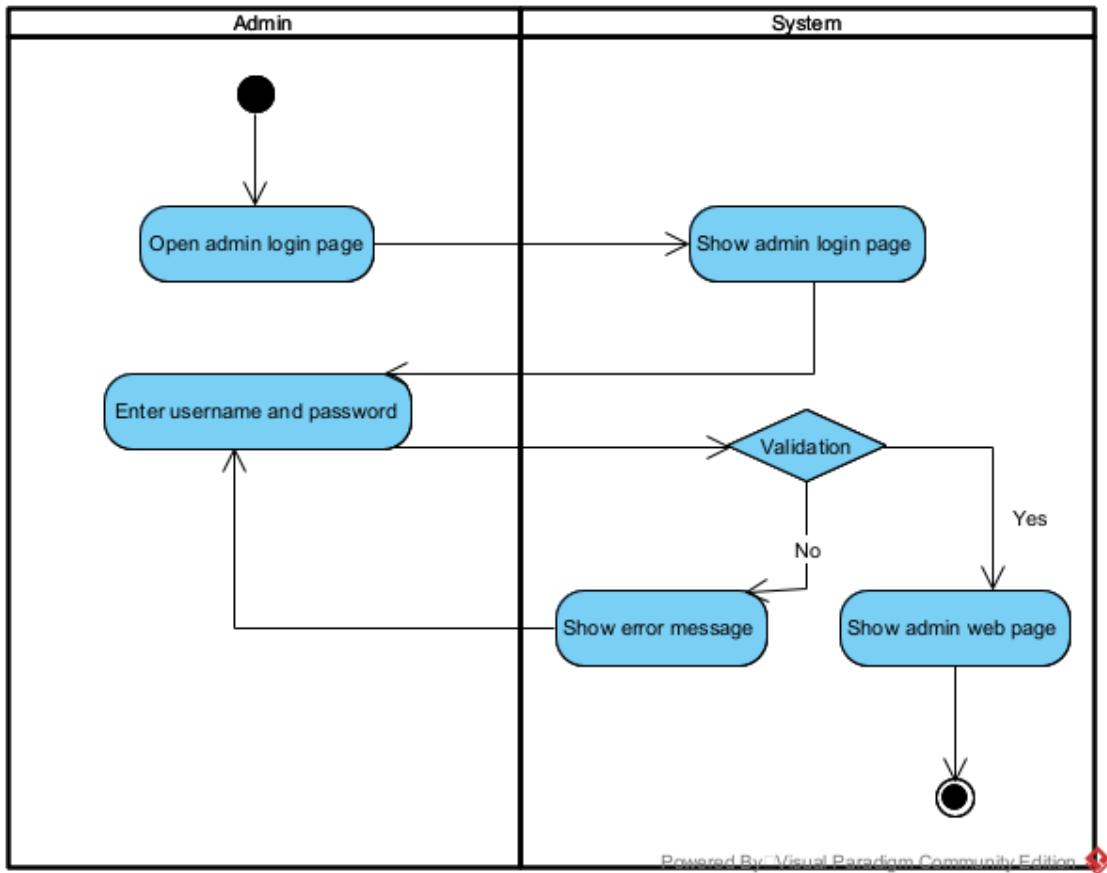
Xóa Sản phẩm



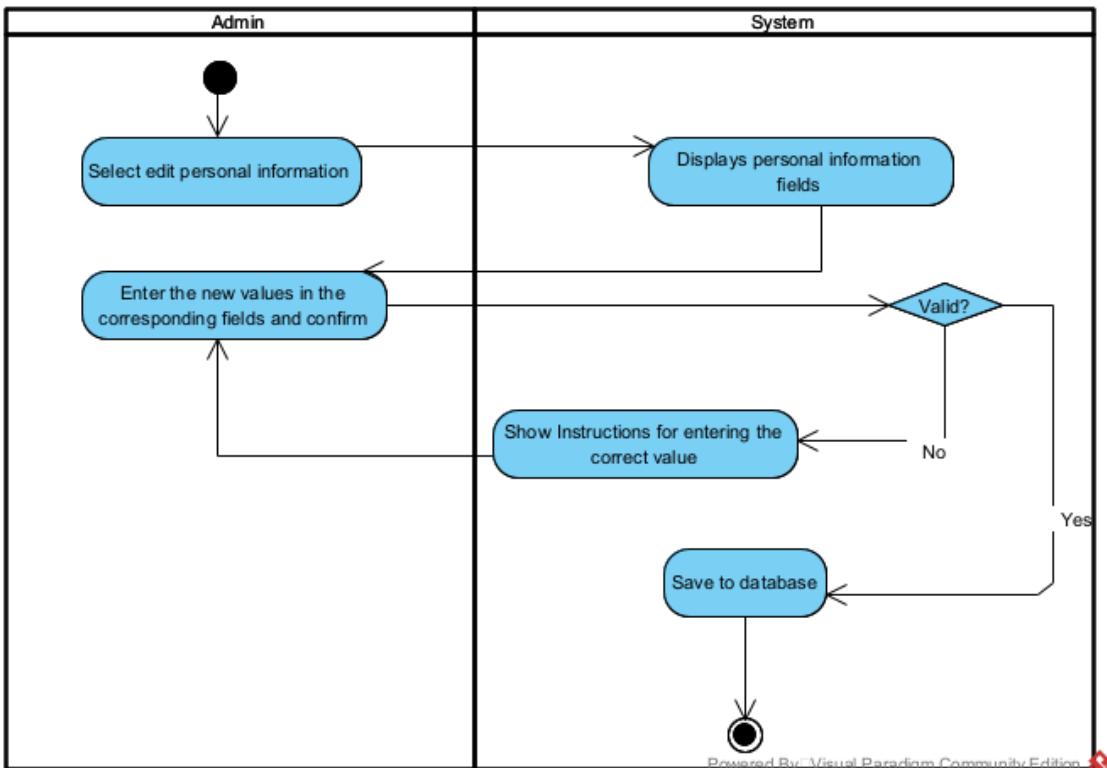
Thêm mã giảm giá



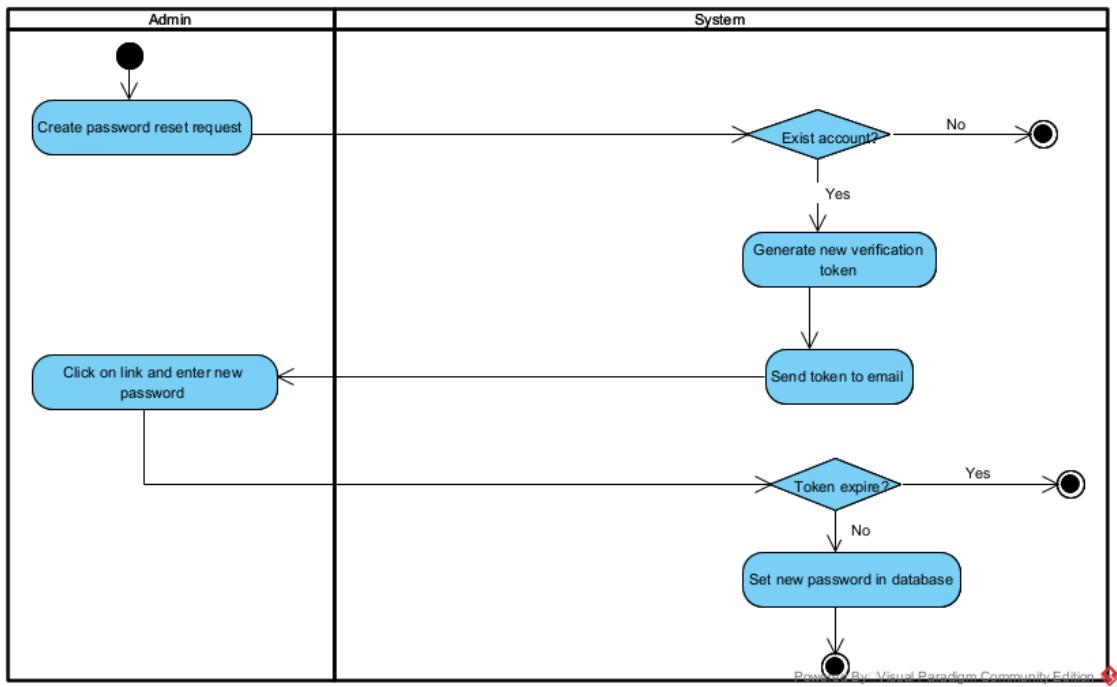
Login



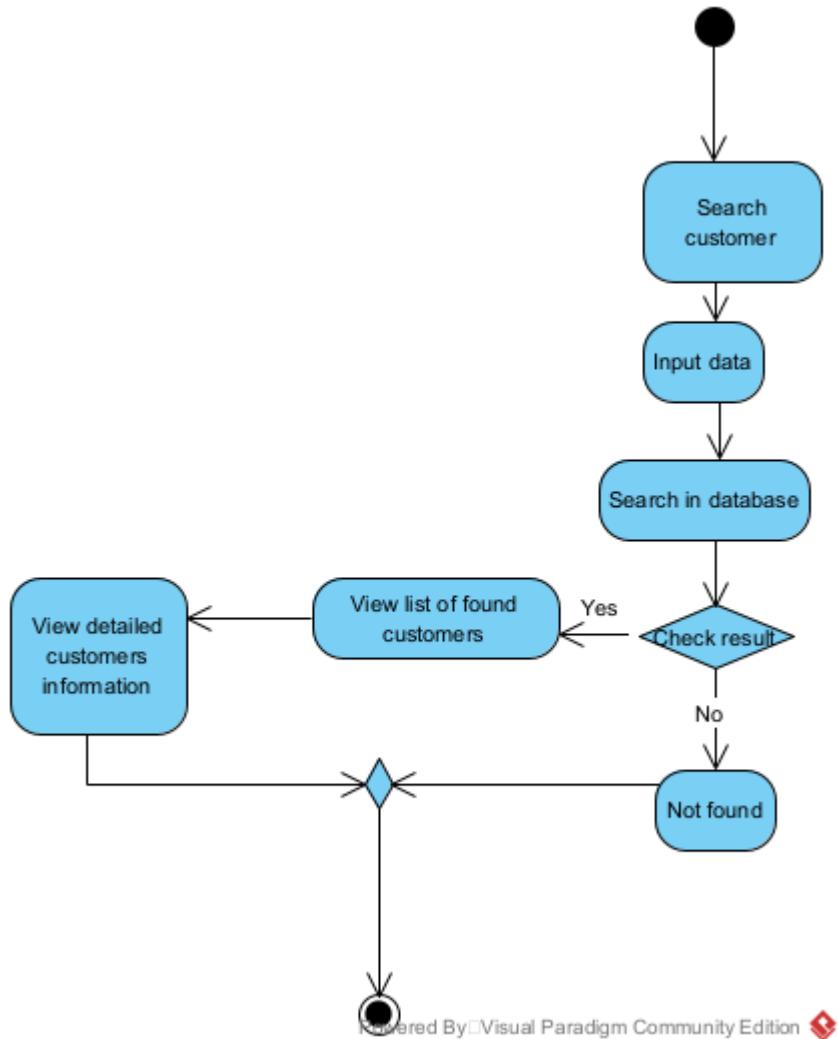
Change admin's information



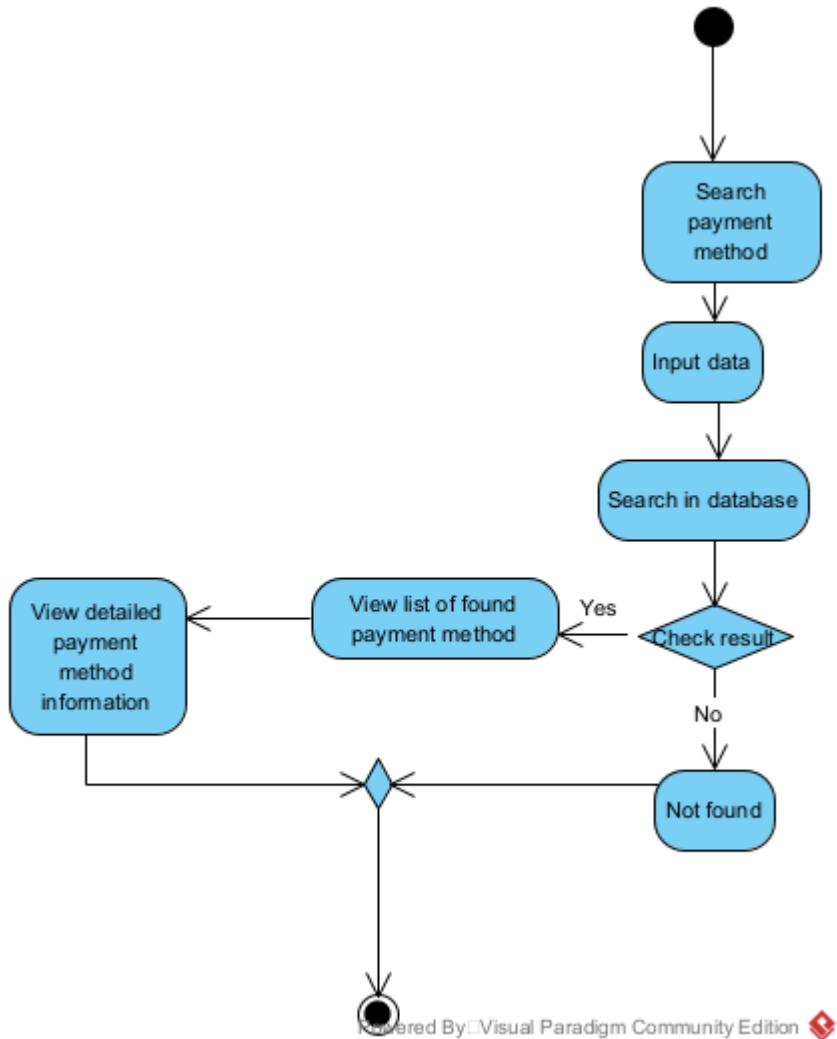
Forgot password



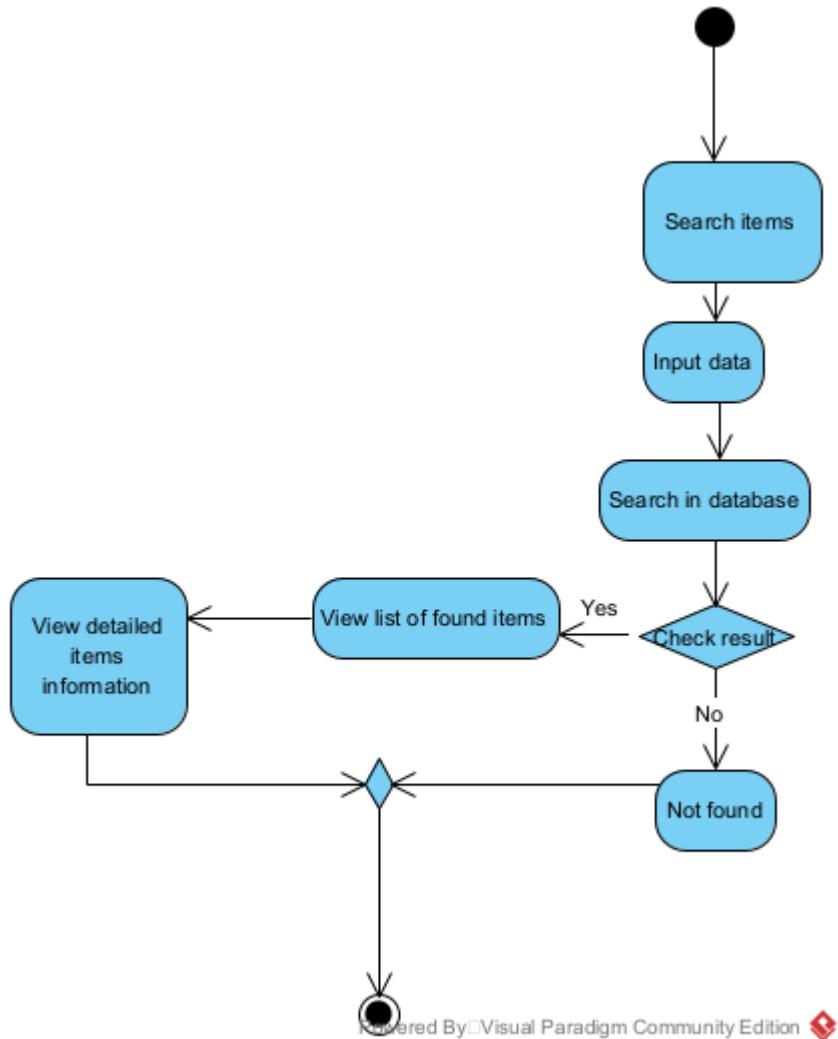
Search customer



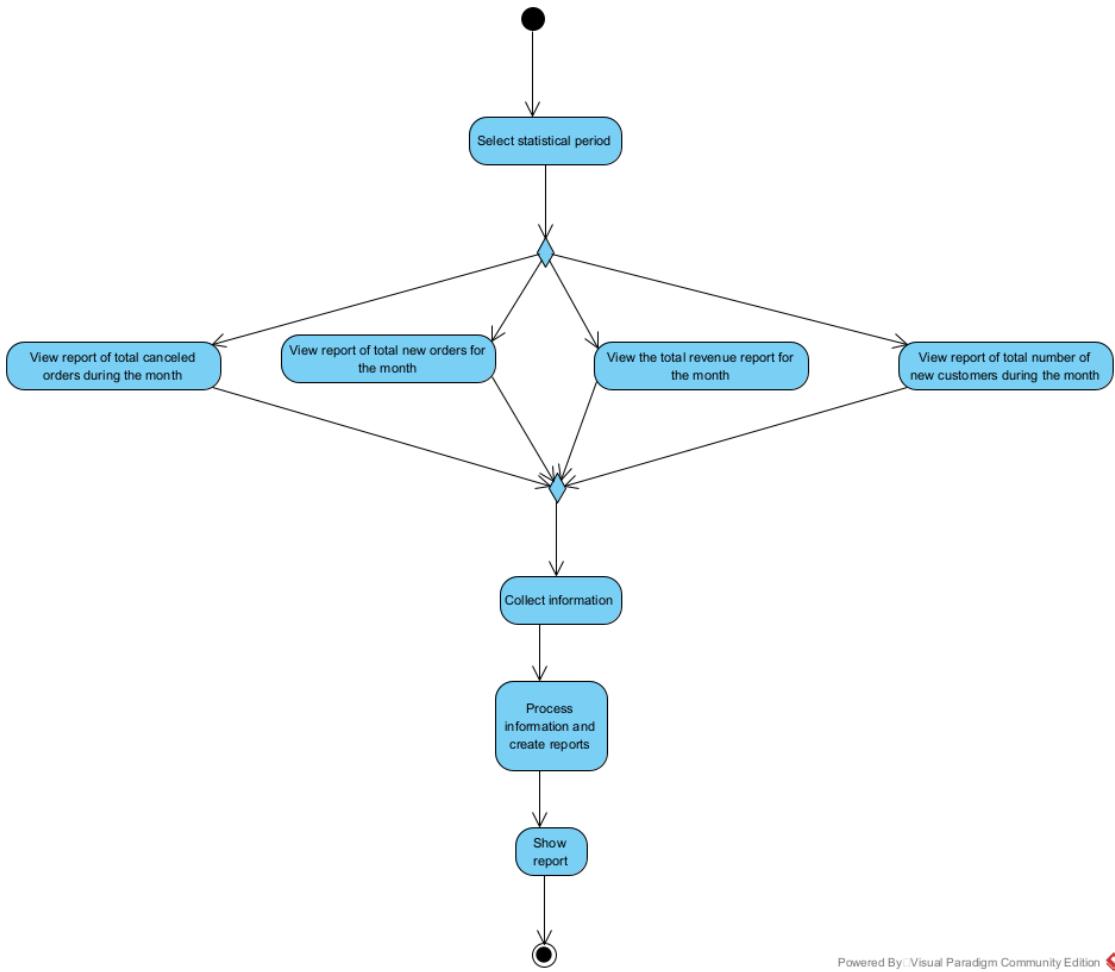
Search payment method



Search product

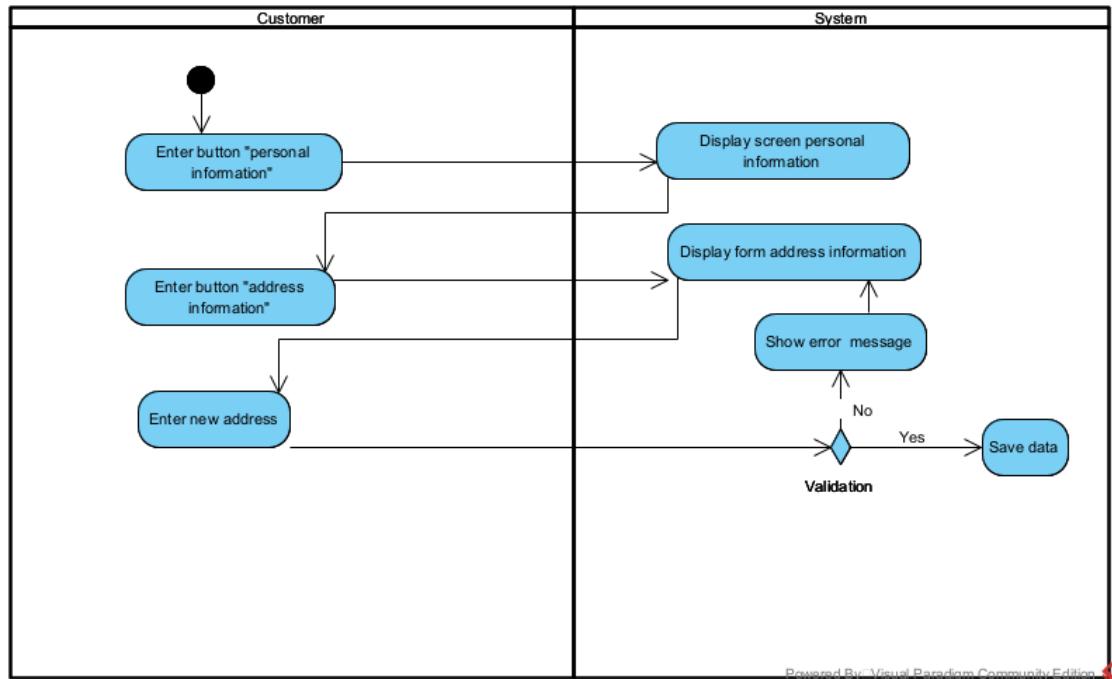


View report



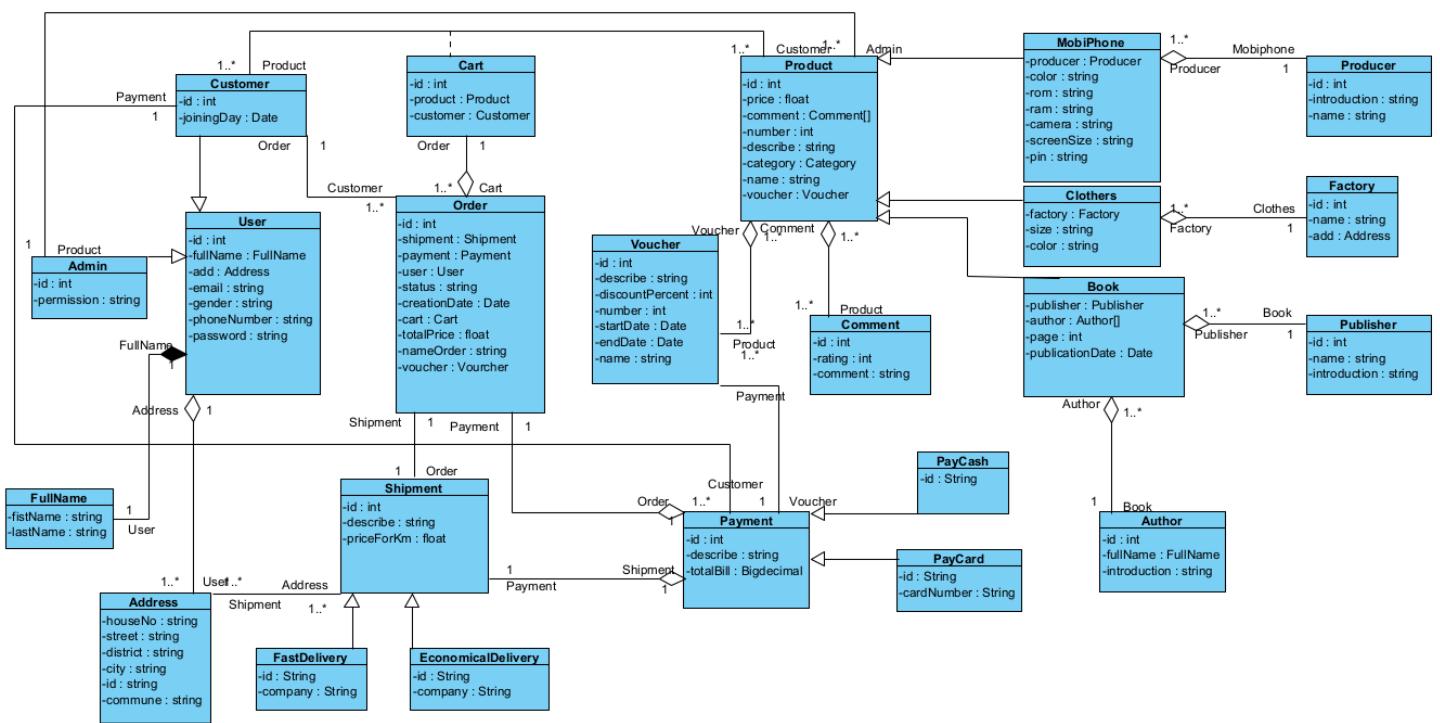
Customer:

Thêm nhiều địa chỉ

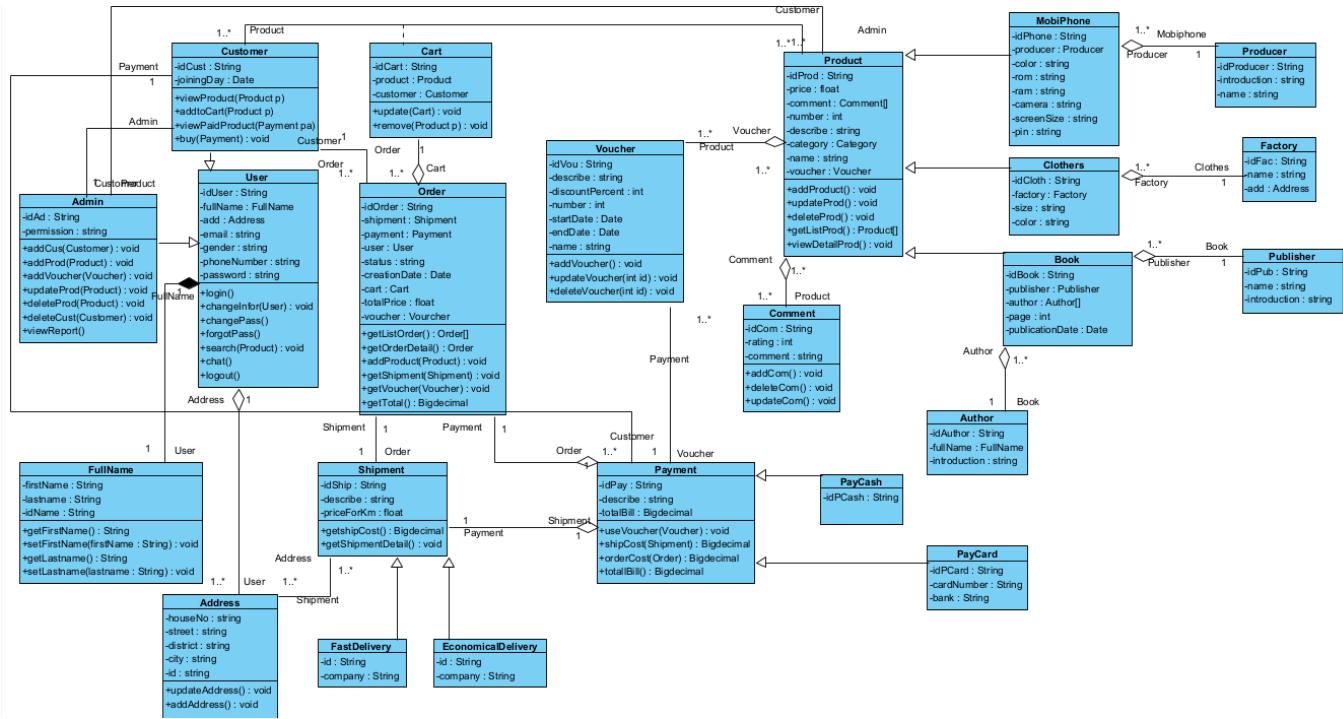


13b.6 Trình bày Biểu đồ LỚP THIẾT KẾ với DAO và layer MVC cho Hệ thống

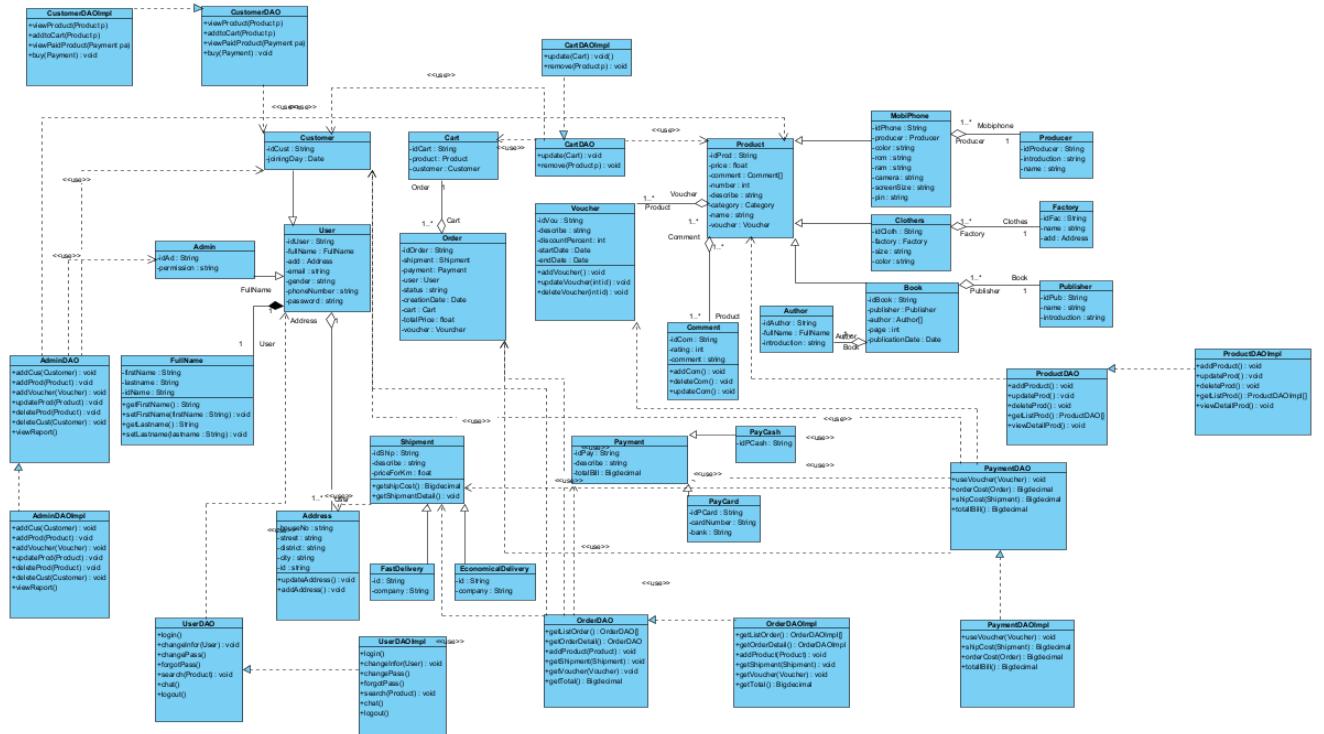
Biểu đồ lớp



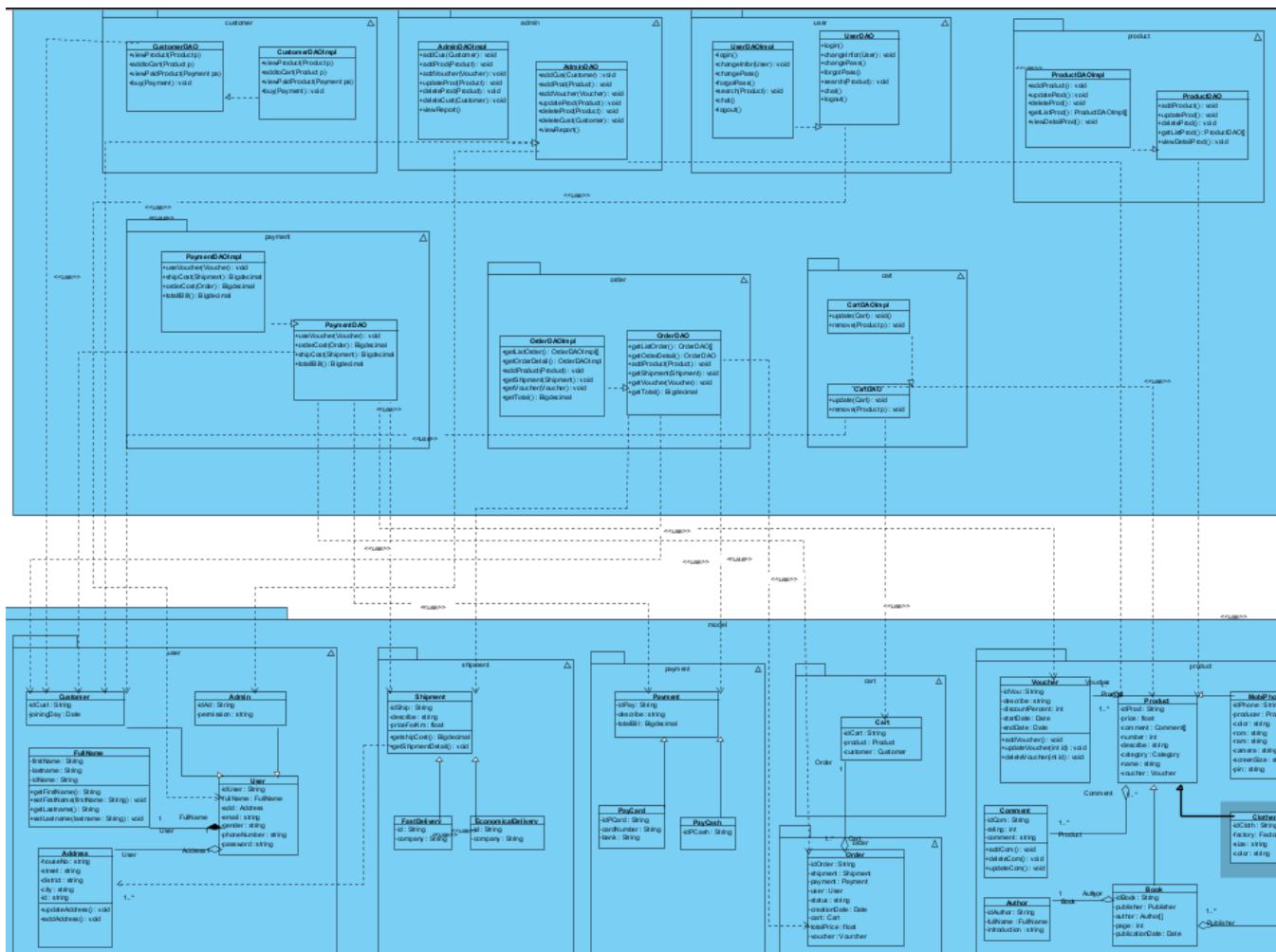
Thêm phương thức:



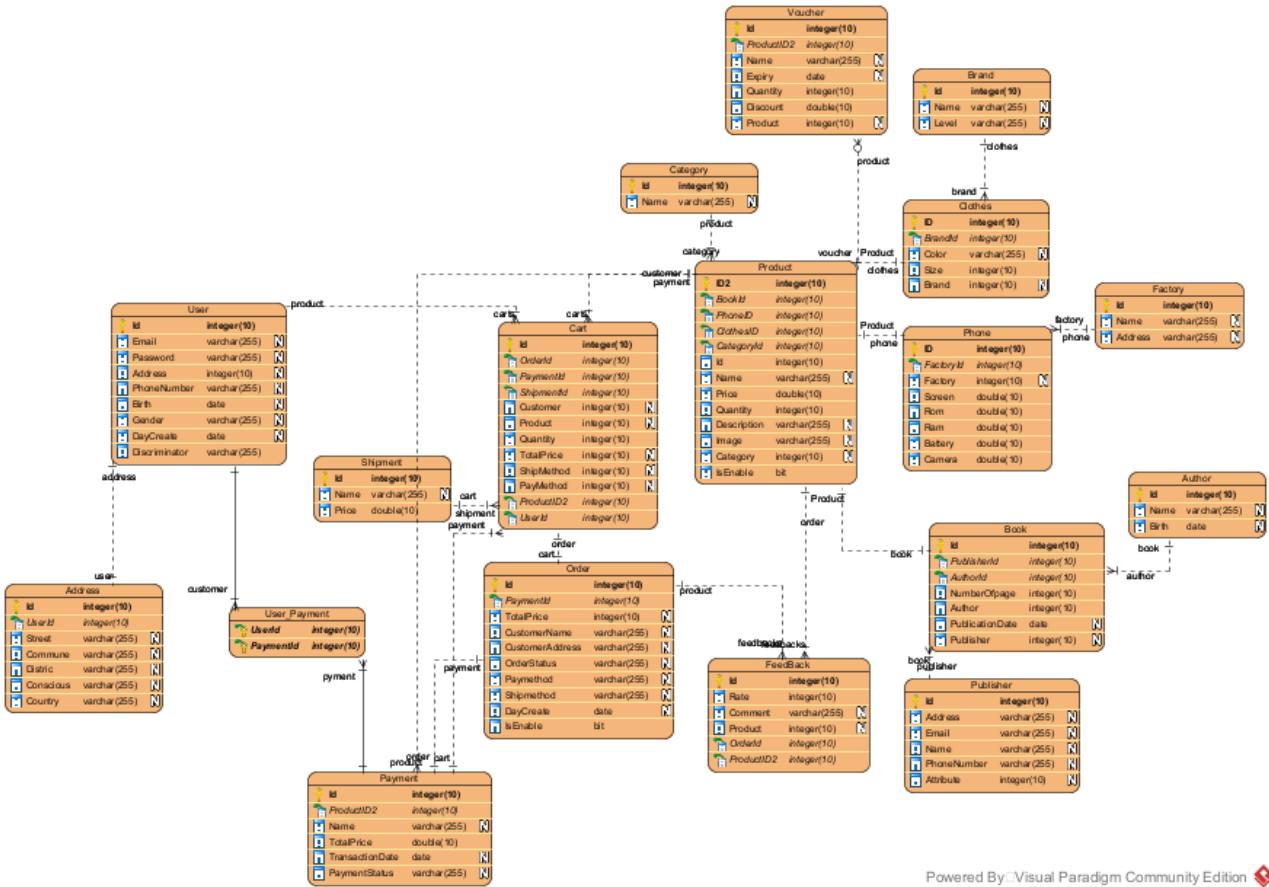
Sử dụng kỹ thuật DAO



Mô hình MVC



13b.7 Thiết kế Bảng và sinh ra cơ sở dữ liệu với mySQL



Powered By Visual Paradigm Community Edition

Sinh ra cơ sở dữ liệu

use EcoMas;

```
CREATE TABLE Product (ID2 int(10) NOT NULL AUTO_INCREMENT, BookId
int(10) NOT NULL, PhoneID int(10) NOT NULL, ClothesID int(10) NOT NULL,
CategoryId int(10) NOT NULL, Id int(10) NOT NULL, Name varchar(255), Price
double NOT NULL, Quantity int(10) NOT NULL, Description varchar(255), Image
varchar(255), Category int(10), IsEnable bit(1) NOT NULL, PRIMARY KEY (ID2));
```

```
CREATE TABLE Clothes (ID int(10) NOT NULL AUTO_INCREMENT, BrandId
int(10) NOT NULL, Color varchar(255), `Size` int(10) NOT NULL, Brand int(10),
PRIMARY KEY (ID));
```

```
CREATE TABLE Phone (ID int(10) NOT NULL AUTO_INCREMENT, FactoryId
int(10) NOT NULL, Factory int(10), Screen double NOT NULL, Rom double NOT
```

NULL, Ram double NOT NULL, Battery double NOT NULL, Camera double NOT NULL, PRIMARY KEY (ID));

CREATE TABLE Book (Id int(10) NOT NULL AUTO_INCREMENT, PublisherId int(10) NOT NULL, AuthorId int(10) NOT NULL, NumberOfpage int(10) NOT NULL, Author int(10) NOT NULL, PublicationDate date, Publisher int(10), PRIMARY KEY (Id));

CREATE TABLE Category (Id int(10) NOT NULL AUTO_INCREMENT, Name varchar(255), PRIMARY KEY (Id));

CREATE TABLE `User` (Id int(10) NOT NULL AUTO_INCREMENT, Email varchar(255), Password varchar(255), Address int(10), PhoneNumber varchar(255), Birth date, Gender varchar(255), DayCreate date, Discriminator varchar(255) NOT NULL, PRIMARY KEY (Id));

CREATE TABLE Cart (Id int(10) NOT NULL AUTO_INCREMENT, OrderId int(10) NOT NULL, PaymentId int(10) NOT NULL, ShipmentId int(10) NOT NULL, Customer int(10), Product int(10), Quantity int(10) NOT NULL, TotalPrice int(10), ShipMethod int(10), PayMethod int(10), ProductID2 int(10) NOT NULL, UserId int(10) NOT NULL, PRIMARY KEY (Id));

CREATE TABLE `Order` (Id int(10) NOT NULL AUTO_INCREMENT, PaymentId int(10) NOT NULL, TotalPrice int(10), CustomerName varchar(255), CustomerAddress varchar(255), OrderStatus varchar(255), Paymethod varchar(255), Shipmethod varchar(255), DayCreate date, IsEnable bit(1) NOT NULL, PRIMARY KEY (Id));

CREATE TABLE Voucher (Id int(10) NOT NULL AUTO_INCREMENT, ProductID2 int(10) NOT NULL, Name varchar(255), Expiry date, Quantity int(10) NOT NULL, Discount double NOT NULL, Product int(10), PRIMARY KEY (Id));

CREATE TABLE Shipment (Id int(10) NOT NULL AUTO_INCREMENT, Name varchar(255), Price double NOT NULL, PRIMARY KEY (Id));

CREATE TABLE Payment (Id int(10) NOT NULL AUTO_INCREMENT, ProductID2 int(10) NOT NULL, Name varchar(255), TotalPrice double NOT NULL, TransactionDate date, PaymentStatus varchar(255), PRIMARY KEY (Id));

```
CREATE TABLE FeedBack (Id int(10) NOT NULL AUTO_INCREMENT, Rate int(10) NOT NULL, Comment varchar(255), Product int(10), OrderId int(10) NOT NULL, ProductID2 int(10) NOT NULL, PRIMARY KEY (Id));

CREATE TABLE Factory (Id int(10) NOT NULL AUTO_INCREMENT, Name varchar(255), Address varchar(255), PRIMARY KEY (Id));

CREATE TABLE Brand (Id int(10) NOT NULL AUTO_INCREMENT, Name varchar(255), Level varchar(255), PRIMARY KEY (Id));

CREATE TABLE Author (Id int(10) NOT NULL AUTO_INCREMENT, Name varchar(255), Birth date, PRIMARY KEY (Id));

CREATE TABLE Publisher (Id int(10) NOT NULL AUTO_INCREMENT, Address varchar(255), Email varchar(255), Name varchar(255), PhoneNumber varchar(255), Attribute int(10), PRIMARY KEY (Id));

CREATE TABLE Address (Id int(10) NOT NULL AUTO_INCREMENT, UserId int(10) NOT NULL, Street varchar(255), Commune varchar(255), Distric varchar(255), Conscious varchar(255), Country varchar(255), PRIMARY KEY (Id));

CREATE TABLE User_Payment (UserId int(10) NOT NULL, PaymentId int(10) NOT NULL, PRIMARY KEY (UserId, PaymentId));

ALTER TABLE Cart ADD CONSTRAINT FKCart829174 FOREIGN KEY (ProductID2) REFERENCES Product (ID2);

ALTER TABLE Cart ADD CONSTRAINT FKCart424263 FOREIGN KEY (UserId) REFERENCES `User` (Id);

ALTER TABLE FeedBack ADD CONSTRAINT FKFeedBack188717 FOREIGN KEY (OrderId) REFERENCES `Order` (Id);

ALTER TABLE FeedBack ADD CONSTRAINT FKFeedBack925561 FOREIGN KEY (ProductID2) REFERENCES Product (ID2);

ALTER TABLE Product ADD CONSTRAINT FKProduct365970 FOREIGN KEY (CategoryId) REFERENCES Category (Id);

ALTER TABLE Book ADD CONSTRAINT FKBook666529 FOREIGN KEY (AuthorId) REFERENCES Author (Id);
```

```
ALTER TABLE Book ADD CONSTRAINT FKBook221707 FOREIGN KEY  
(PublisherId) REFERENCES Publisher (Id);  
  
ALTER TABLE Phone ADD CONSTRAINT FKPhone539846 FOREIGN KEY  
(FactoryId) REFERENCES Factory (Id);  
  
ALTER TABLE Clothes ADD CONSTRAINT FKClothes84051 FOREIGN KEY  
(BrandId) REFERENCES Brand (Id);  
  
ALTER TABLE Address ADD CONSTRAINT FKAddress555376 FOREIGN KEY  
(UserId) REFERENCES `User` (Id);  
  
ALTER TABLE Voucher ADD CONSTRAINT FKVoucher387523 FOREIGN KEY  
(ProductID2) REFERENCES Product (ID2);  
  
ALTER TABLE Cart ADD CONSTRAINT FKCart112749 FOREIGN KEY  
(ShipmentId) REFERENCES Shipment (Id);  
  
ALTER TABLE Cart ADD CONSTRAINT FKCart453015 FOREIGN KEY  
(PaymentId) REFERENCES Payment (Id);  
  
ALTER TABLE Cart ADD CONSTRAINT FKCart92330 FOREIGN KEY (OrderId)  
REFERENCES `Order` (Id);  
  
ALTER TABLE User_Payment ADD CONSTRAINT FKUser_Payme429008  
FOREIGN KEY (UserId) REFERENCES `User` (Id);  
  
ALTER TABLE User_Payment ADD CONSTRAINT FKUser_Payme400256  
FOREIGN KEY (PaymentId) REFERENCES Payment (Id);  
  
ALTER TABLE Payment ADD CONSTRAINT FKPayment709128 FOREIGN KEY  
(ProductID2) REFERENCES Product (ID2);  
  
ALTER TABLE `Order` ADD CONSTRAINT FKOrder92127 FOREIGN KEY  
(PaymentId) REFERENCES Payment (Id);  
  
ALTER TABLE Product ADD CONSTRAINT FKProduct838605 FOREIGN KEY  
(ClothesID) REFERENCES Clothes (ID);  
  
ALTER TABLE Product ADD CONSTRAINT FKProduct528032 FOREIGN KEY  
(PhoneID) REFERENCES Phone (ID);  
  
ALTER TABLE Product ADD CONSTRAINT FKProduct263447 FOREIGN KEY  
(BookId) REFERENCES Book (Id);
```

C.HỆ ĐĂNG KÍ HỌC THEO TÍN CHỈ Ở TRƯỜNG ĐẠI HỌC

13c.1 Trình bày bằng ngôn ngữ tự nhiên và biểu đồ context cho các hệ thống

Hệ thống đăng kí tín chỉ của một trường đại học là một phần quan trọng của hệ thống quản lý học tập và sinh viên của trường. Hệ thống này giúp sinh viên đăng kí và quản lý tín chỉ của họ trong suốt thời gian học tập tại trường. Dưới đây là mô tả tổng quan về cách một hệ thống đăng kí tín chỉ của trường đại học có thể hoạt động:

1. Giao diện người dùng:

- Hệ thống cung cấp giao diện người dùng trực quan cho sinh viên để họ đăng kí tín chỉ và quản lý lịch học của mình. Giao diện này cần thân thiện, dễ sử dụng và có thể truy cập từ mọi thiết bị có kết nối internet.

2. Đăng kí tín chỉ:

- Sinh viên có thể xem danh sách các khóa học có sẵn và đăng kí tín chỉ thông qua giao diện trực tuyến.
- Hệ thống cung cấp thông tin về các khóa học, bao gồm tên khóa học, mô tả, giảng viên, ngày giờ, và số lượng tín chỉ.

3. Lựa chọn khóa học:

- Sinh viên có thể lựa chọn các khóa học mà họ muốn đăng ký và thêm vào lịch học của họ.

- Hệ thống cần kiểm tra xem các khóa học này có sự trùng lặp về thời gian hoặc có những yêu cầu tiên quyết không.

4. Xác nhận đăng ký:

- Sau khi lựa chọn khóa học, sinh viên cần xác nhận đăng ký của họ, và hệ thống sẽ gửi xác nhận đến họ.

5. Quản lý lịch học:

- Hệ thống cho phép sinh viên xem lịch học của họ, bao gồm thời gian, địa điểm, và tên giảng viên.

- Nếu có sự thay đổi trong lịch học, hệ thống cần thông báo cho sinh viên.

6. Thông tin tín chỉ:

- Hệ thống cung cấp thông tin về số tín chỉ đã đăng ký, số tín chỉ đã hoàn thành, và số tín chỉ còn thiếu để hoàn thành chương trình học.

7. Hỗ trợ và thông kê:

- Hệ thống cần có tính năng hỗ trợ trực tuyến hoặc tư vấn cho sinh viên khi họ cần giúp đỡ về việc đăng ký tín chỉ.

- Cung cấp các thông kê về đăng ký tín chỉ cho các phòng ban của trường đại học để quản lý tài nguyên giảng dạy.

8. Điều chỉnh và hủy đăng ký:

- Sinh viên cần có khả năng điều chỉnh hoặc hủy đăng ký tín chỉ trong khoảng thời gian được cho phép.

9. Bảo mật và quyền truy cập

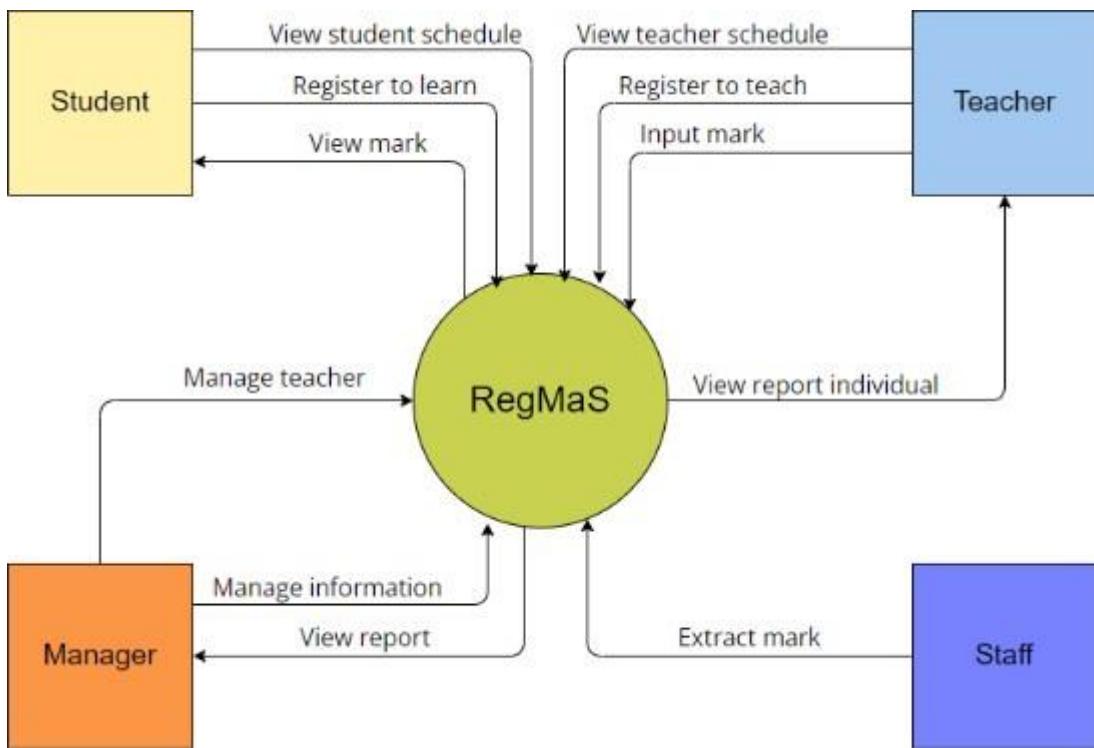
- Hệ thống cần có các biện pháp bảo mật mạnh để đảm bảo an toàn thông tin cá nhân và đăng ký của sinh viên.

- Quản trị viên của trường đại học cần có quyền truy cập để kiểm tra và quản lý đăng ký tín chỉ của sinh viên.

10. Tích hợp với hệ thống quản lý sinh viên:

- Hệ thống đăng ký tín chỉ cần tích hợp chặt chẽ với hệ thống quản lý sinh viên để cập nhật thông tin sinh viên và kết quả học tập.

Biểu đồ context:



13c.2 Mô tả các actor

Primary actors:

- Sinh viên - Student. Đây có thể xem là người dùng chính của hệ thống, họ tham gia vào việc sử dụng hệ thống để đăng ký môn học theo tín chỉ phù hợp với mục tiêu học tập của bản thân. Hoạt động chính:

- + Đăng nhập vào hệ thống.
- + Thay đổi mật khẩu của tài khoản được cấp.
- + Thực hiện đăng ký môn học.
- + Xem thời khóa biểu học tập.
- + Xem điểm cá nhân.
- + Xuất bản điểm cá nhân.

- Giảng viên - Teacher. Cung cấp thông tin môn học, lịch học tập và theo dõi tiến độ học tập của sinh viên và các vấn đề liên quan đến môn học mà mình đã đăng ký. Hoạt động chính:

- + Đăng ký các môn học để giảng dạy.
 - + Xem thời khóa biểu giảng dạy.
 - + Nhập điểm cho sinh viên.
 - + Xem các đánh giá và báo cáo của môn học.
- Nhân viên giáo vụ - TQA staff. Thực hiện các chức năng liên quan đến điểm số đối với sinh viên. Hoạt động chính: Xuất bản điểm cá nhân cho sinh viên khi có yêu cầu.
- Quản lý - Manager. Thực hiện quản lý giảng viên, sinh viên và các thông tin khác trên hệ thống. Hoạt động chính:
- + Quản lý giảng viên.
 - + Quản lý sinh viên.
 - + Xem các báo cáo và thống kê trên hệ thống.
 - + Quản lý các thông tin trên hệ thống.

- + Quản lý các lịch giảng dạy và học tập trên hệ thống.

Secondary actors

- Kế toán trường:** Kế toán trường có thể liên quan đến việc xác nhận thanh toán học phí sau khi sinh viên đăng ký tín chỉ.
- Hệ thống gửi thông báo:** Một hệ thống tự động có thể gửi thông báo đến sinh viên và giảng viên liên quan đến việc đăng ký, thay đổi lịch, hay các thông tin quan trọng khác.
- Phòng đào tạo:** Phòng đào tạo có thể tham gia trong việc xác nhận và duyệt các yêu cầu đăng ký đặc biệt, như đăng ký quá hạn.
- Nhân viên giáo vụ:** Nhân viên giáo vụ có thể tham gia vào nhiều khía cạnh của quá trình đăng ký tín chỉ. Họ có thể hỗ trợ sinh viên trong việc đăng ký, cung cấp thông tin về lịch học, giúp giải quyết vấn đề liên quan đến việc đăng ký và thay đổi tín chỉ.
- Nhân viên khảo thí:** Nhân viên khảo thí có thể liên quan đến việc quản lý và tổ chức các kỳ thi và khảo thí. Họ có thể nhập điểm sau khi kết thúc một kỳ thi và đảm bảo rằng kết quả được cập nhật đúng cách trong hệ thống.

13c.3 Mô tả yêu cầu các chức năng và phi chức năng của hệ thống bằng ngôn ngữ tự nhiên và dạng bảng

Dưới đây là mô tả chi tiết về các chức năng của mỗi vai trò trong hệ thống đăng ký tín chỉ:

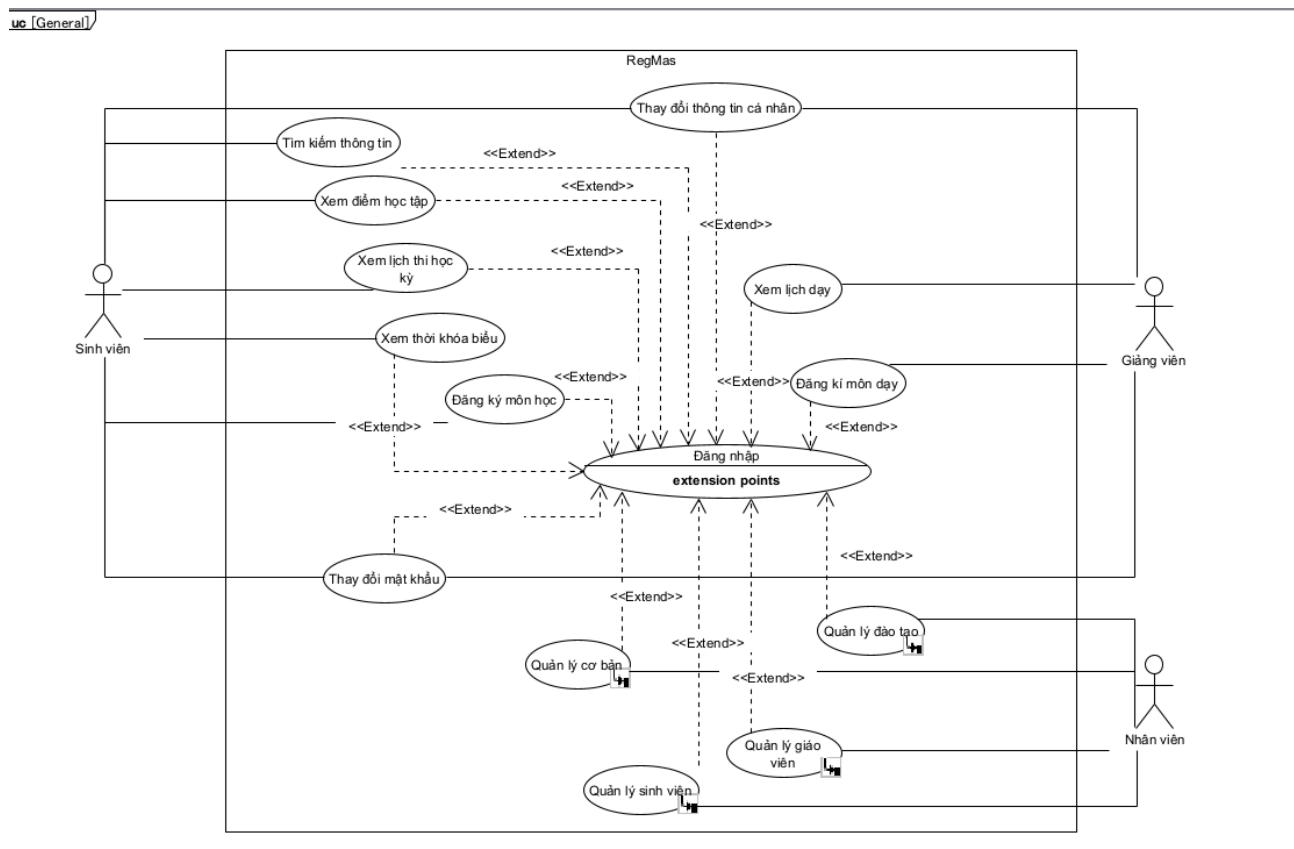
1. Đăng ký học: Sinh viên có thể tìm kiếm, chọn và đăng ký các khóa học từ danh sách có sẵn. Họ có khả năng xem thông tin chi tiết về các khóa học và sau đó đăng ký tín chỉ cho các khóa học mà họ muốn tham gia.
2. Sửa thông tin đăng ký: Sinh viên có thể điều chỉnh hoặc hủy đăng ký tín chỉ trong khoảng thời gian cho phép. Họ có khả năng thêm hoặc xóa các khóa học từ lịch học của mình.
3. Xem lịch học: Sinh viên có thể xem lịch học của họ, bao gồm thời gian, địa điểm, và tên giảng viên. Họ cần biết khi nào và ở đâu học các khóa học đã đăng ký.
4. Đăng ký dạy: Giảng viên có thể đăng ký dạy các khóa học mà họ được giao. Họ cung cấp thông tin về khóa học, bao gồm lịch giảng dạy và thông tin liên quan.
5. Sửa thông tin đăng ký dạy: Giảng viên có khả năng cập nhật thông tin về khóa học mà họ dạy, bao gồm lịch giảng dạy và nội dung khóa học.
6. Xem lịch dạy: Giảng viên có thể xem lịch dạy của họ, bao gồm thời gian, địa điểm, và tên khóa học mà họ đang dạy.
7. Xem thống kê liên quan đến các lớp mình dạy: Giảng viên có khả năng xem các thống kê về sự tham gia của sinh viên
8. Quản lý thông tin sinh viên: Nhân viên giáo vụ có thể thêm, xóa hoặc sửa thông tin sinh viên theo yêu cầu từ sinh viên. Họ cần duyệt đăng ký và hỗ trợ sinh viên về thông tin liên quan đến học tập.
11. Quản lý thông tin giảng viên: Họ cung cấp và quản lý thông tin giảng viên theo yêu cầu từ giảng viên. Họ đảm bảo rằng thông tin về giảng viên là chính xác và cập nhật.
12. Quản lý thông tin môn học: Nhân viên giáo vụ quản lý thông tin về các môn học, bao gồm tên, mô tả, và số tín chỉ. Họ cập nhật danh sách các khóa học có sẵn cho đăng ký.
13. Quản lý thông tin lớp học phần: Họ quản lý thông tin về các lớp học phần, bao gồm lịch học, địa điểm, và sự phân chia giảng viên.
14. Xem các loại thống kê: Nhân viên quản lý có quyền xem các loại thống kê toàn diện

liên quan đến hoạt động của hệ thống đăng ký tín chỉ. Điều này bao gồm thông kê về đăng ký, thông kê về số lượng sinh viên và giảng viên, và thông kê về tiến trình học tập.

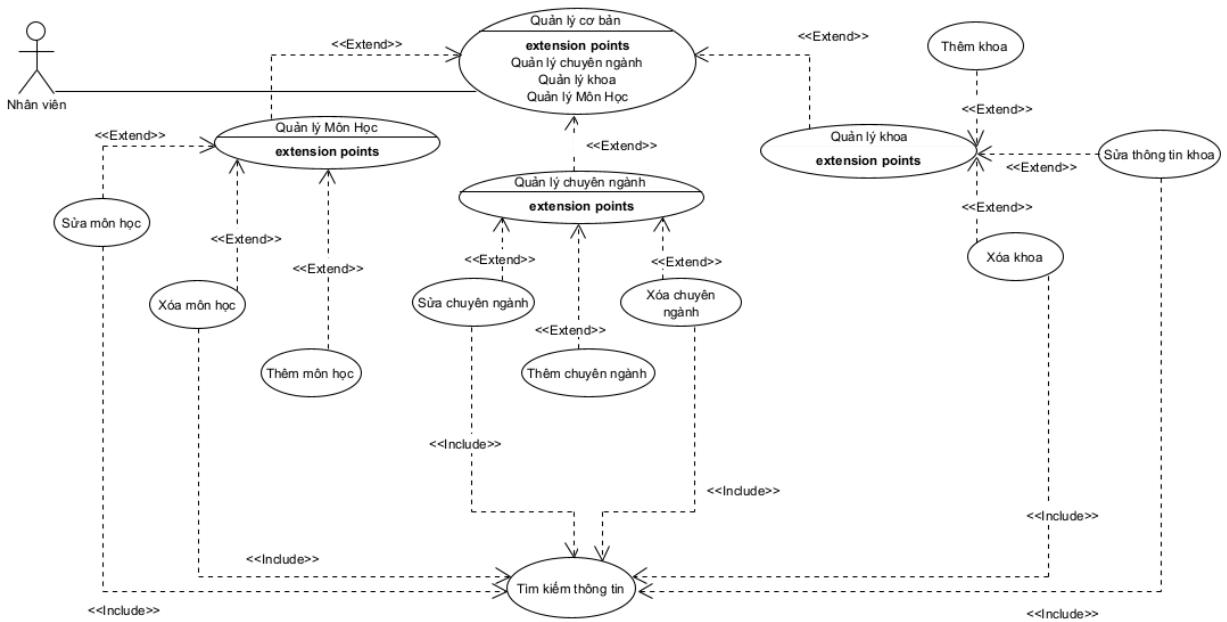
Mô tả phi chức năng:

- Tính bảo trì: hệ thống nên có thời gian bảo trì ngắn
- Tính bảo mật: hệ thống bảo mật thông tin của khách hàng
- Hiệu suất: đảm bảo hiệu suất cao khi có lượng lớn truy nhập vào hệ thống
- Độ tin cậy: hệ thống phải có khả năng sao lưu dữ liệu để tránh bị mất mát dữ liệu

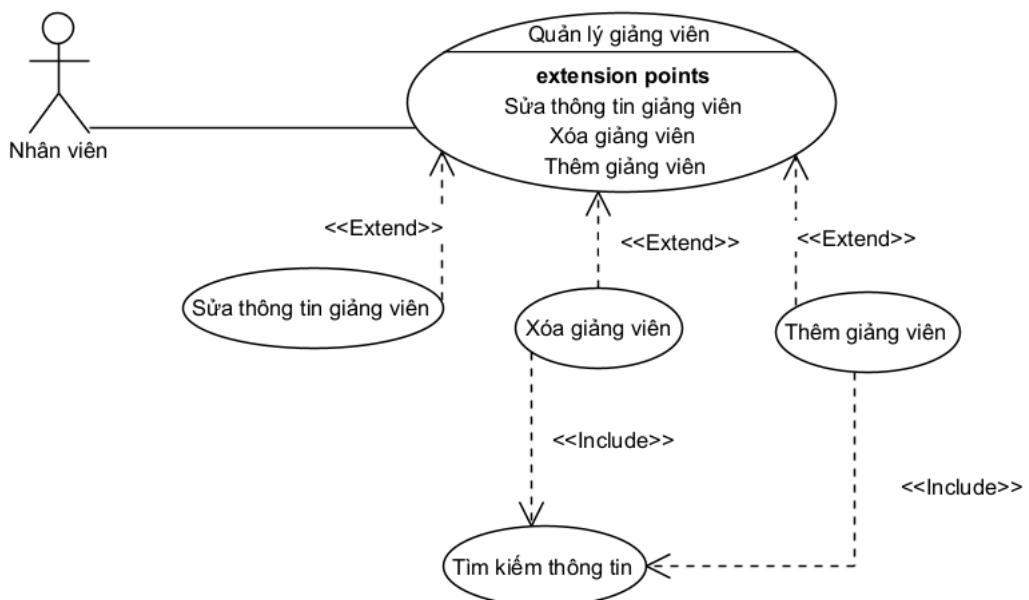
13c.4 Trình bày Biểu đồ use case THEO 2 LEVEL cho các Hệ thống



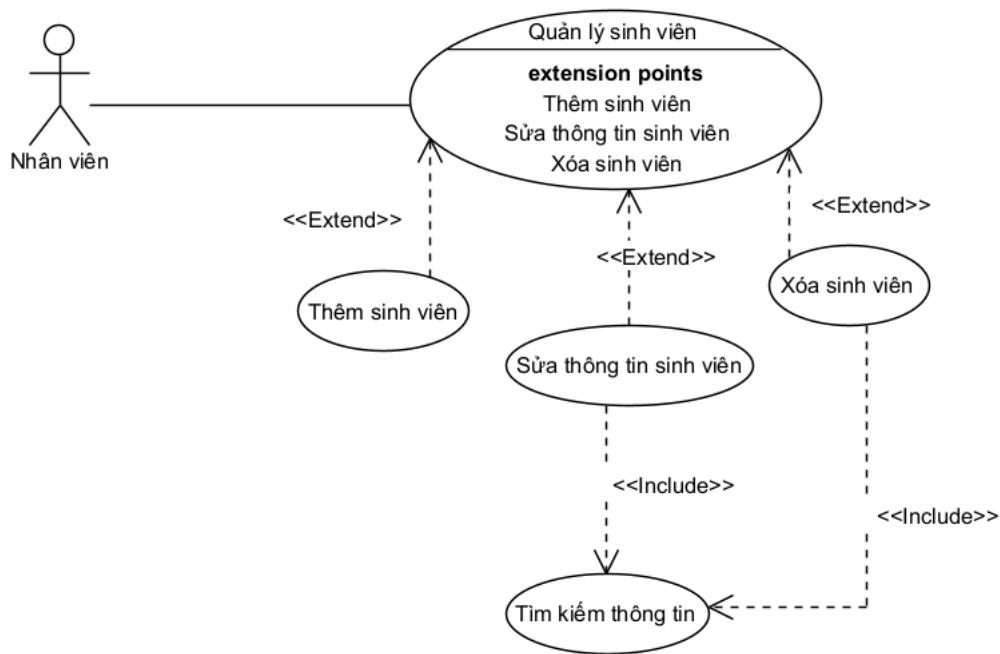
uc [Quản lý cơ bản Use Case Diagram]



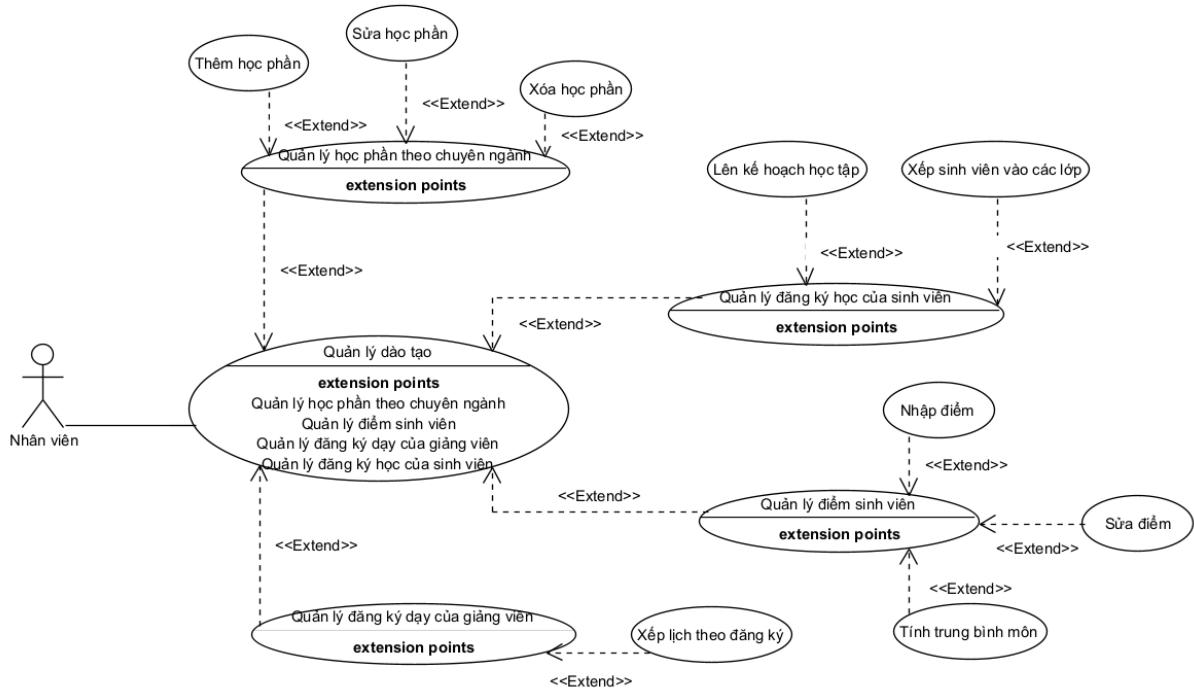
uc [Quản lý giáo viên Use Case Diagram]



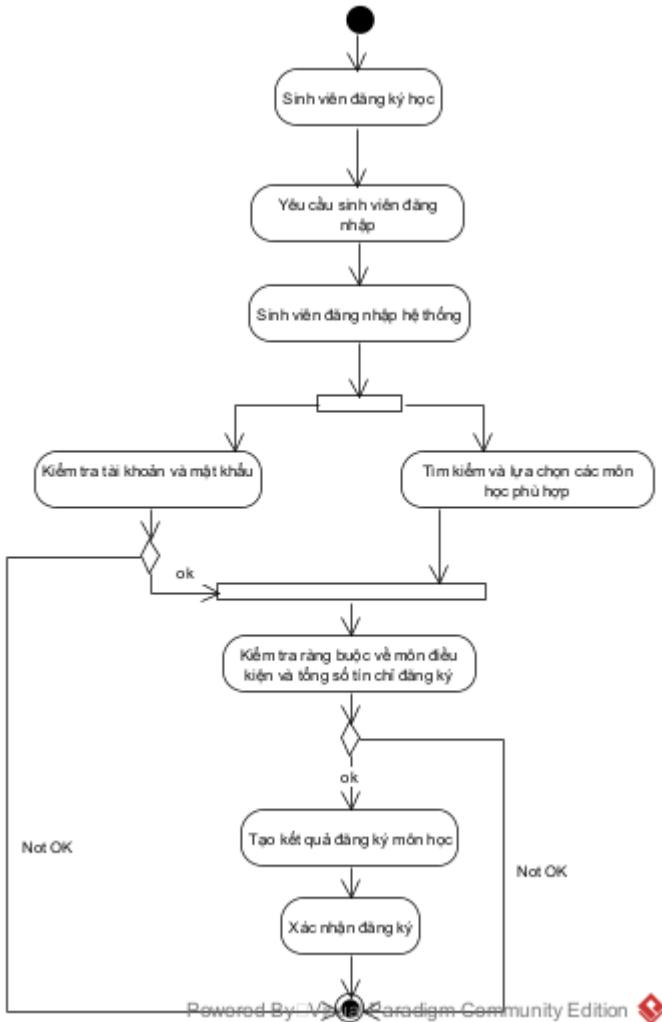
uc [Quản lý sinh viên Use Case Diagram]



uc [Quản lý đào tạo Use Case Diagram]



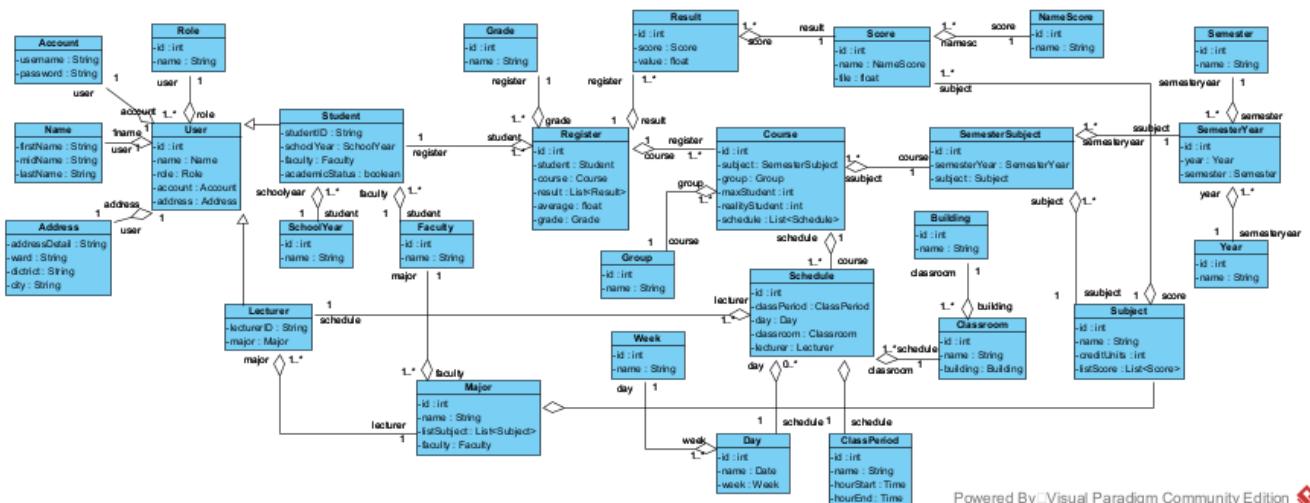
13c.5. Biểu đồ activity



Powered By: Visual Paradigm Community Edition

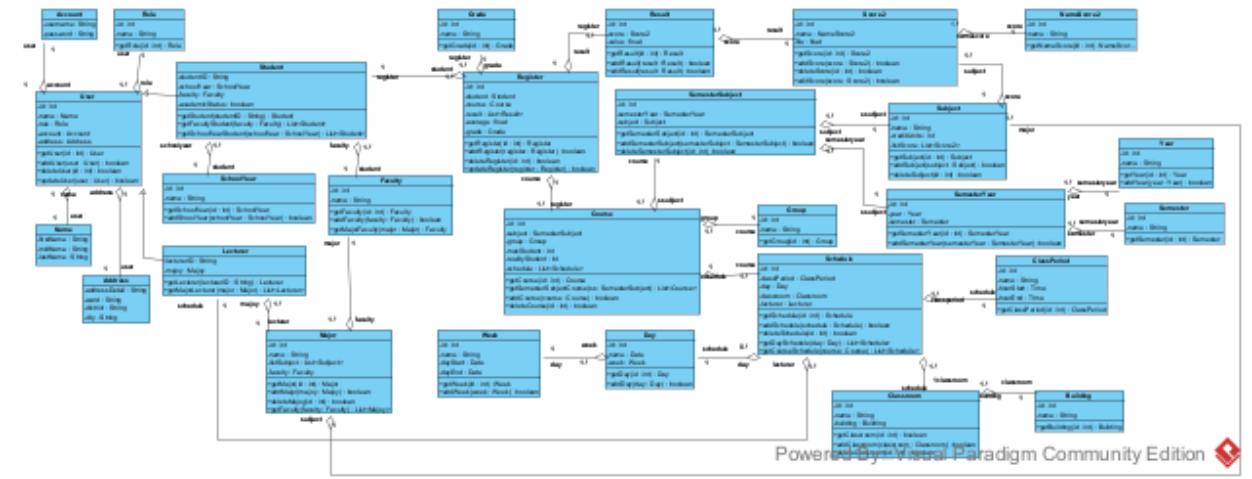
13c.6. Trình bày Biểu đồ LỚP THIẾT KẾ với DAO và layer MVC cho các Hệ thống

Biểu đồ lớp



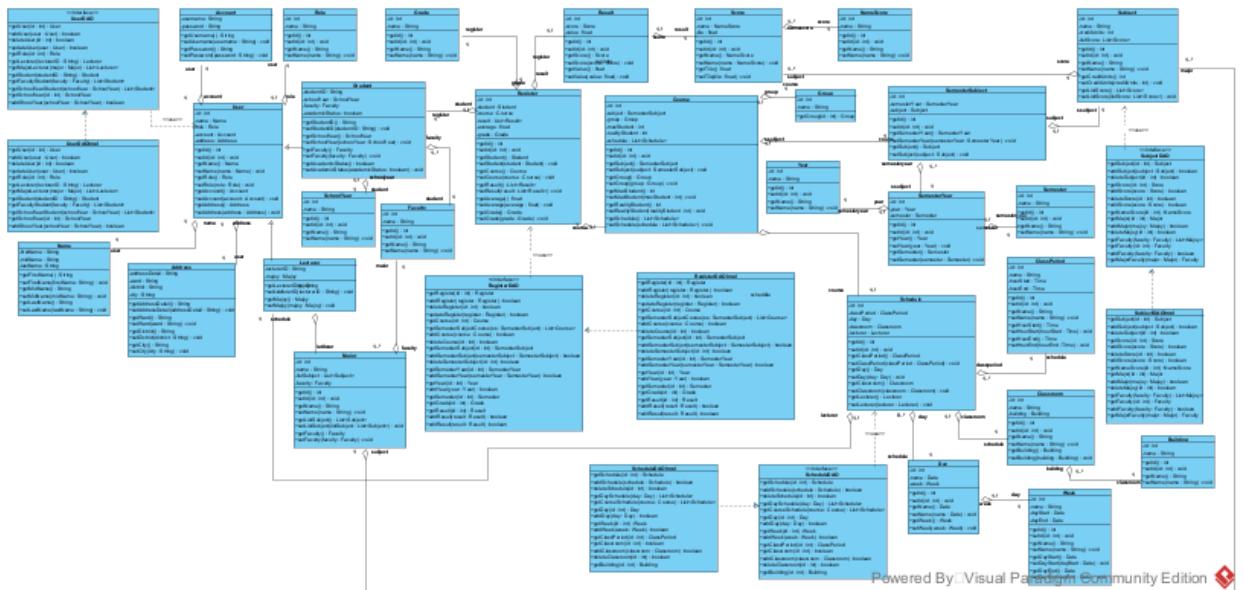
Powered By: Visual Paradigm Community Edition

Thêm phương thức:



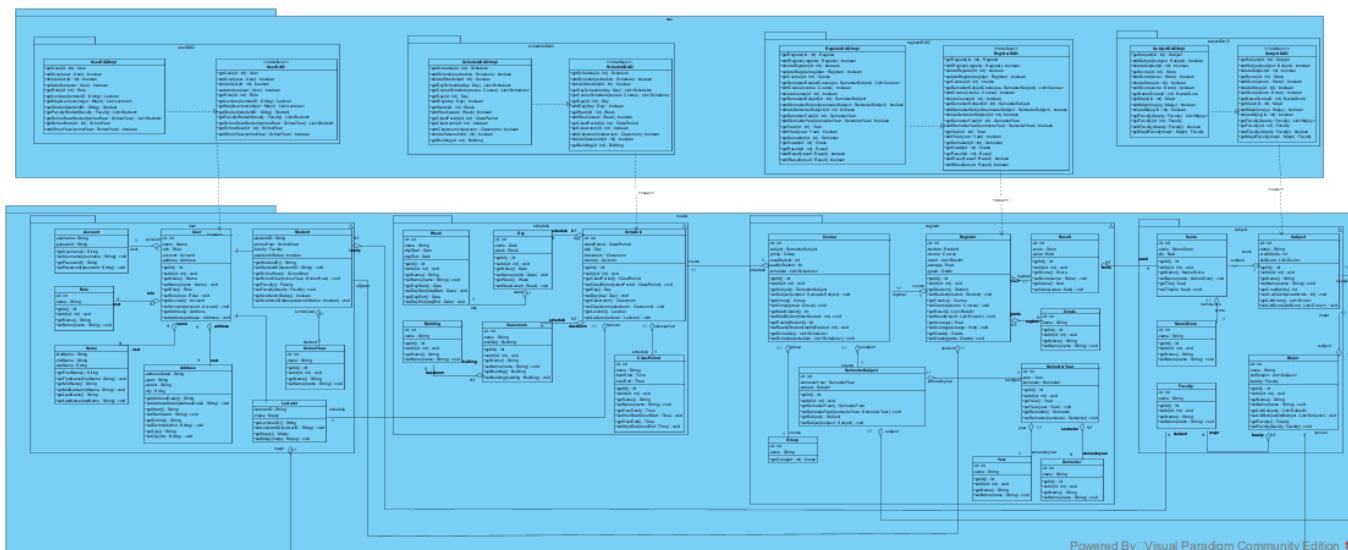
Powered By Visual Paradigm Community Edition

Sử dụng kỹ thuật DAO



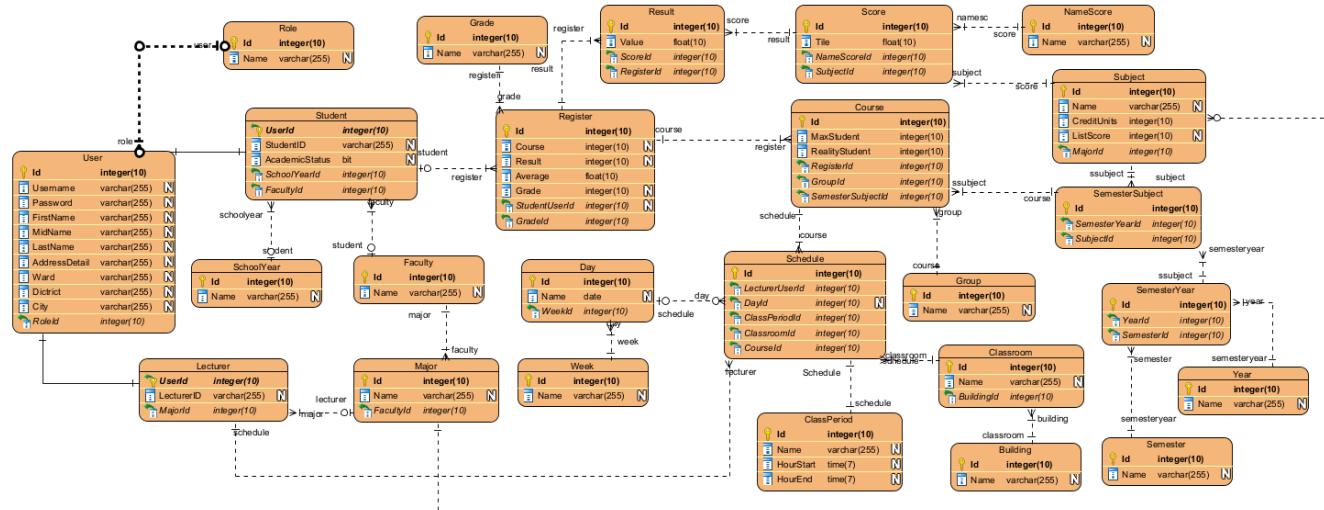
Powered By Visual Paradigm Community Edition

Mô hình MVC



Powered By: Visual Paradigm Community Edition

13c.7 Thiết kế bảng và sinh ra cơ sở dữ liệu với mysql



| | |
|------------------|--|
| Quản lý thư viện | <p>Đăng nhập Đổi mật khẩu</p> |
| | <p>Lấy lại mật khẩu</p> |
| | <p>Quản lý Người dùng: Đăng ký thành viên mới và cấp phát thẻ thư viện. - Chính sửa thông tin cá nhân và lịch sử mượn trả của người dùng. Quản lý loại người dùng (học sinh, giáo viên, bạn đọc, v.v.).</p> |
| | <p>Quản lý Sự Kiện và Chương Trình: Tạo và quản lý sự kiện, chương trình đọc sách, buổi thảo luận, triển lãm, và các hoạt động khác của thư viện.</p> |
| | <p>Báo Cáo và Thống Kê: Tạo báo cáo về hoạt động của thư viện, bao gồm số lượng sách mượn, số lượng người đến thư viện, và các dữ liệu thống kê khác.</p> |
| | <p>Quản lý Tài khoản Nhân viên: Quản lý tài khoản của nhân viên thư viện, đặc biệt là về quyền truy cập vào hệ thống</p> |
| | <p>Quản lý Lịch Trình: Đặt lịch mở cửa, lịch nghỉ, và lịch sự kiện thư viện.</p> |
| | <p>Đăng nhập Đổi mật khẩu Lấy lại mật khẩu</p> |
| | Xử lý mượn trả |

| | |
|---------------------|--|
| Nhân viên thư viện | Tìm kiếm sách dựa trên tiêu đề, ngày xuất bản, tác giả |
| | Xác định vị trí sách |
| Người mượn | Đăng ký |
| | Đăng nhập Đổi mật khẩu |
| | Lấy lại mật khẩu |
| | Đặt trước các cuốn sách muộn mượn |
| | Gia hạn thời gian mượn sách |
| | Yêu cầu mượn 1 cuốn sách mới |
| Hệ thống thông báo | Trả tiền phạt khi trả sách muộn |
| | Gửi thông báo và nhắc nhở đến người mượn, bao gồm thông báo đặt sách trước, thông báo trễ hạn, và các thông báo khác |
| Hệ thống thanh toán | Xử lý các giao dịch thanh toán từ người mượn, bao gồm thanh toán tiền mượn và tiền phạt |
| Hệ thống hiển thị | Hiển thị giao diện người dùng |

Các yêu cầu phi chức năng:

- Giao diện thân thiện với người dùng để dễ dàng điều hướng và sử dụng: Giao diện người dùng phải thân thiện và dễ sử dụng, giúp người dùng dễ dàng tìm kiếm sách và sử dụng các tính năng một cách thuận tiện.
- Hiệu suất cao và khả năng mở rộng để xử lý lượng dữ liệu lớn: Hệ thống phải có hiệu suất cao và khả năng mở rộng để có thể xử lý một lượng lớn dữ liệu mà thư viện có thể

chứa.

- Bảo mật và bảo vệ dữ liệu để đảm bảo sự riêng tư và bảo mật của độc giả thư viện và thông tin của họ: Hệ thống phải đảm bảo an ninh và bảo mật thông tin của người đọc và thông tin liên quan đến hoạt động mượn sách.
- Tương thích với nhiều hệ điều hành và thiết bị: Hệ thống phải tương thích với nhiều hệ điều hành và thiết bị khác nhau, bao gồm máy tính cá nhân, máy tính bảng và điện thoại di động.
- Khả năng xử lý nhiều người dùng và truy cập đồng thời vào hệ thống: Hệ thống phải có khả năng xử lý nhiều người dùng và truy cập đồng thời vào hệ thống mà không gây sự cố hoặc giảm hiệu suất.
- Tuân thủ các luật và quy định liên quan đến quản lý thư viện và bảo vệ dữ liệu riêng tư: Hệ thống phải tuân thủ tất cả các luật và quy định có liên quan đến quản lý thư viện và bảo vệ thông tin riêng tư của người đọc.
- Cập nhật định kỳ và bảo trì để đảm bảo tính hoạt động và bảo mật của hệ thống theo thời gian: Hệ thống phải được cập nhật và bảo trì định kỳ để đảm bảo bảo tính hoạt động và bảo mật của nó theo thời gian

