ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

TRẦN TRUNG HÀ

NGHIÊN CỨU NGÔN NGỮ ĐẶC TẢ YÊU CẦU THEO HƯỚNG CHUYÊN BIỆT MIỀN

LUẬN VĂN THẠC SĨ: CÔNG NGHỆ PHẦN MỀM

HÀ NỘI - 2019

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Trần Trung Hà

NGHIÊN CỨU NGÔN NGỮ ĐẶC TẢ YÊU CẦU THEO HƯỚNG CHUYÊN BIỆT MIỀN

Ngành: Kỹ thuật phần mềm

Chuyên ngành: Kỹ thuật phần mềm

Mã số: 8480103.01

LUẬN VĂN THẠC SĨ: CÔNG NGHỆ PHẦN MỀM

Cán bộ hướng dẫn: TS. Đặng Đức Hạnh

HÀ NỘI - 2019

LỜI CẢM ƠN

Đầu tiên, tôi xin được gửi lời cảm ơn sâu sắc tới Tiến sĩ Đặng Đức Hạnh – giảng viên bộ môn Công nghệ Phần mềm – người đã dành nhiều thời gian và công sức trong suốt năm vừa qua để hướng dẫn tôi hoàn thành luận văn này. Thầy đã giúp tôi từ những bước đầu tiên, từ việc lựa chọn đề tài phù hợp với mình đến chia sẻ các phương pháp nghiên cứu, kinh nghiệm làm việc, giao tiếp,... những kĩ năng cần thiết không chỉ trong chính luận văn này mà còn trong cuộc sống, sự nghiệp tương lai của tôi.

Tôi cũng xin gửi lời cảm ơn chân thành đến các thành viên trong nhóm nghiên cứu đã hỗ trợ tôi rất tận tình trong khoảng thời gian vừa qua. Các anh chị em trong nhóm đã biểu hiện một tình thần đoàn kết cao, tương trợ lẫn nhau trong các công việc lớn nhỏ, cùng thảo luận, đóng góp ý kiến với mỗi vấn đề của mỗi thành viên. Đó chắc chắn sẽ là những kỉ niệm khó quên đối với mỗi người trong nhóm, đặc biệt là với tôi.

Ngoài ra, tôi xin gửi lời cảm ơn đến các thầy cô giảng viên của Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội. Những kiến thức chuyên môn, nghiệp vụ và cả các kĩ năng mềm mà các thầy cô đã dạy cho tôi trong suốt khóa học đã trở thành nền tảng để tôi phát triển và xây dựng luận văn này. Tôi cũng xin cảm ơn sự hỗ trợ của đề tài QG.18.61 của Đại học Quốc gia Hà Nội.

Cuối cùng, tôi xin cảm ơn gia đình, bạn bè và người thân đã đồng hành cùng tôi trong cuộc sống, cung cấp cho tôi ý chí và nghị lực để luôn vươn lên trong cuộc sống.

LỜI CAM ĐOAN

Tôi là Trần Trung Hà, học viên khóa K24CNPM thuộc chương trình đào tạo Thạc sĩ của Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội. Tôi xin cam đoan rằng những nghiên cứu trong luận văn này là của tôi, được hướng dẫn bởi Tiến sĩ Đặng Đức Hạnh. Những nghiên cứu này chưa từng được báo cáo hoặc sử dụng ở bất kì nơi nào khác, bởi bất kì ai khác. Tôi xin cam đoan không sao chép, sử dụng tài liệu, công trình nghiên cứu nào của người khác mà không chú thích, trích dẫn cụ thể. Công cụ FRSL là chương trình phần mềm do nhóm nghiên cứu của Tiến sĩ Đặng Đức Hạnh tự phát triển, không sao chép mã nguồn của người khác.

Hà Nội, ngày ... tháng ... năm 2019

Học viên

Trần Trung Hà

MỤC LỤC

MỞ ĐẦU 3
CHƯƠNG 1. Kiến thức nền tảng 5
1.1. Đặc tả yêu cầu 5
1.2. Ca sử dụng 7
1.3. Ngôn ngữ mô hình hóa chuyên biệt miền 9
1.3.1. Mô hình hóa chuyên biệt miền 9
1.3.2. Khái niệm về ngôn ngữ mô hình hóa chuyên biệt miền 12
1.3.3. Xây dựng ngôn ngữ mô hình hóa chuyên biệt miền 14
1.4. Một số công cụ hỗ trợ 15
1.4.1. Công cụ ANTLR 15
1.4.2. Công cụ PlantUML 17
1.5. Tổng kết chương 18
CHƯƠNG 2. Ngôn ngữ đặc tả ca sử dụng FRSL 19
2.1. Giới thiệu 19
2.2. Miền vấn đề đặc tả ca sử dụng 19
2.3. Cú pháp trừu tượng FRSL 23
2.4. Cú pháp cụ thể FRSL 25
2.5. Một số chuyển đổi từ đặc tả FRSL 27
2.6. Các công việc liên quan 28
2.7. Tổng kết chương 31
CHƯƠNG 3. Cài đặt và Thực nghiệm 32
3.1. Giới thiệu 32
3.2. Công cụ hỗ trợ 32
3.3. Bài toán vận dụng 34
3.4. Đánh giá 36
3.5. Tổng kết chương 37
KẾT LUẬN 38

DANH SÁCH KÝ HIỆU, CHỮ VIẾT TẮT

ANTLR	Another Tool for Language Recognition
AST	Abstract Syntax Tree
FRSL	Functional Requirement Specification Language
RUCM	Restricted Use Case Modeling
UML	Unified Modeling Language

DANH SÁCH BẢNG BIỂU, HÌNH VẼ

Hình 1.1: Sơ đồ ca sử dụng. 8
Hình 1.2: Sơ đồ hoạt động của ca sử dụng. 9
Hình 1.3: Hiệu quả của mô hình hóa chuyên biệt miền so với đa tính năng. 10
Hình 1.4: Hướng tiếp cận của mô hình hóa chuyên biệt miền so với UML. 11
Hình 1.5: Cấu trúc hướng phát triển metamodel. 13
Hình 1.6: Mối quan hệ giữa mô hình và metamodel. 14
Hình 1.7: Cây phân tích cú pháp xây dựng bởi ANTLR. 17
Hình 1.8: Một biểu đồ đơn giản của công cụ PlantUML. 18
Bảng 2.1: Ví dụ về một khuôn mẫu mô tả ca sử dụng. 22
Hình 2.2: Hệ thống metamodel của FRSL. 24
Hình 2.3: Cú pháp cụ thể dưới dạng văn bản của FRSL. 26
Hình 2.4: Ca sử dụng Rút tiền dưới dạng văn bản cú pháp cụ thể FRSL. 27
Bảng 2.5 (a)(b)(c): Bộ luật giới hạn của RUCM. 29
Bảng 2.6: Khuôn mẫu của RUCM. 30
Hình 3.1: Giao diện của công cụ FRSL. 33
Hình 3.2: Menu plugin của công cụ FRSL. 34
Hình 3.3: Văn bản đặc tả ca sử dụng rút gọn. 35
Hình 3.4: Sơ đồ ca sử dụng được sinh từ văn bản đặc tả. 36

TÓM TẮT

Đặc tả yêu cầu là một trong những bước quan trọng nhất của quá trình phát triển phần mềm. Ca sử dụng là một trong những phương pháp thể hiện yêu cầu được sử dụng phổ biến nhất, giúp tăng khả năng diễn đạt, tính dễ đọc và nhiều lợi ích khác cho người dùng. Tuy nhiên, vẫn còn nhiều vấn đề hạn chế trong quá trình thiết kế và phân tích ca sử dụng. Vì vậy, cần phải phát triển một phương pháp đặc tả ca sử dụng tốt hơn, với hiệu quả và chất lượng cao hơn. Luận văn hướng tới xây dựng một ngôn ngữ đặc tả ca sử dụng, phát triển theo hướng mô hình hóa chuyên biệt miền nhằm đem đến kết quả tốt hơn cho quá trình đặc tả. Được xây dựng trên miền vấn đề ca sử dụng, tư tưởng chính của ngôn ngữ là phân tích và mô hình hóa những mô tả ca sử dụng viết dưới dạng văn bản thành các mô hình dựa trên hệ thống metamodel đã được dựng sẵn. Mô hình tổng hợp được sau đó có thể được chuyển sang các dạng khác để hoàn thành các mục tiêu khác nhau. Ngôn ngữ này sau đó được phát triển cùng bộ công cụ hỗ trợ, giúp người dùng dễ dàng tiếp cận và mở rộng các tính năng dựa theo yêu cầu cụ thể.

MỞ ĐẦU

Những năm gần đây, chúng ta đã chứng kiến sự bùng nổ của khoa học công nghệ. Với cuộc cách mạng công nghiệp lần thứ tư đang từng bước diễn ra, tất cả các lĩnh vực trong đời sống đều đang ngày càng phát triển mãnh mẽ hơn trên toàn thế giới. Trong đó, ngành công nghiệp phần mềm là một trong những ngành tổng hợp và đa dạng nhất, đóng vai trò thiết yếu trong tốc độ phát triển của ứng dụng công nghệ. Một trong những bước quan trọng của quá trình phát triển phần mềm là khâu đặc tả yêu cầu. Cấu trúc và yêu cầu của các phần mềm rất phức tạp, đặc biệt là đối với những dự án lớn. Những người đưa ra yêu cầu thường không phải là người xây dựng phần mềm, vì vậy nên để những người phát triển hiểu được chính xác yêu cầu của khách hàng, cần phải đưa ra một bản đặc tả yêu cầu phần mềm chi tiết.

Một đặc tả tốt cần phải giúp cho người xem hiểu được vấn đề, xác định được những phạm vi và khả năng của hệ thống. Người phát triển phần mềm phải nắm được những hành vi mà người sử dụng có thể thực hiện, những tính năng mà hệ thống cung cấp, cùng những ràng buộc trong hệ thống. Ngoài ra, quá trình phân tích và đặc tả yêu cầu thường gặp nhiều khó khăn. Những sai sót trong đặc tả sẽ ảnh hưởng trực tiếp đến sản phẩm phần mềm. Việc định hướng trước được những tính năng trong tương lai là rất khó, đồng thời trong thời gian phát triển các yêu cầu có thể thay đổi nhiều. Những người đưa ra yêu cầu ít khi hiểu biết sâu về phần mềm, và những người xây dựng phần mềm thường không hiểu được những vấn đề chuyên ngành được nói tới. Vì vậy, đặc tả cần phải thể hiện sao cho cả hai bên đều hiểu, chỉ ra cho người dùng biết rõ họ thực sự muốn gì và mức độ khả thi thế nào, và giúp nhà phát triển xác định rõ làm sao để đưa những điều đó vào chương trình phần mềm.

Một trong những phương pháp thể hiện yêu cầu được sử dụng phổ biến là ca sử dụng (use case). Ca sử dụng được định nghĩa là thể hiện của những trình tự hành vi bao gồm cả những hành vi thay thế và trình tự lỗi, qua đó đặc tả được những yêu cầu chức năng của hệ thống. Đây là những hành vi tương tác giữa hệ thống và các yếu tố bên ngoài để tạo nên giá trị sử dụng [10]. Ưu điểm của đặc tả ca sử dụng là tính dễ hiểu và dễ dùng, giúp cho người đưa ra yêu cầu có thể tiếp cận và sử dụng. Tuy nhiên, nhược điểm là mô hình ca sử dụng là thường có một số thành phần không rõ ràng, gây khó khăn cho việc phát triển theo hướng mô hình. Với sự phát triển công nghệ mạnh mẽ trong cuộc cách mạng công nghệ 4.0, các yêu cầu ngày càng trở nên phức tạp hơn, và vì thế các ca sử dụng cũng ngày càng khó diễn đạt hơn và cần được phân tích, đặc tả một cách chính xác hơn.

Nhiều nghiên cứu đã đề cập và cung cấp phương pháp giải quyết vấn đề này, trong đó [2] đã đưa ra bộ ngôn ngữ USL để đặc tả ca sử dụng. Mô hình USL thể hiện các thành phần liên quan chặt chẽ của đặc tả ca sử dụng bao gồm các luồng, bước, hành vi hệ thống (system action), hành vi người dùng (actor action), các mối quan hệ, luồng điều khiển và các ràng buộc. Tuy nhiên, USL chỉ cung cấp cú pháp cụ thể dưới dạng đồ họa, khiến cho việc tiếp cận của người sử dụng để tạo những mô hình USL còn nhiều hạn chế. Ngoài ra, cấu trúc của USL khá phức tạp và được xây dựng theo một khuôn khổ chặt chẽ, khó có khả năng mở rộng và đáp ứng với các thay đổi sau này.

Nhằm mục tiêu giải quyết các vấn đề của quá trình đặc tả ca sử dụng – đặc tả yêu cầu phần mềm, đồng thời cải thiện các hạn chế của các nghiên cứu trước đây, luận văn sẽ đề xuất một ngôn ngữ mô hình hóa chuyên biệt miền có tên là Functional Requirement Specification Language (FRSL) để đặc tả ca sử dụng. Hướng tiếp cận này bao gồm quá trình xác định các khái niệm của miền đặc tả ca sử dụng, từ đó xây dựng cú pháp trừu tượng cho ngôn ngữ dưới dạng metamodel. FRSL cung cấp cú pháp cụ thể ở dạng văn bản để làm ngữ pháp cho việc xây dựng các mô hình. Mô hình FRSL sau đó có thể được chuyển đổi sang các dạng khác như ca kiểm thử, mô hình thiết kế, bản mẫu phần mềm,... Luận văn cũng hướng tới xây dựng một công cụ hỗ trợ cho quá trình tạo, sử dụng và lưu trữ các mô hình FRSL, đồng thời xây dựng công cụ dựa trên kiến trúc plugin, cho phép khả năng nâng cấp và mở rộng trong tương lai một cách dễ dàng. Những người sử dụng trong cộng đồng lập trình viên cũng sẽ có thể đóng góp những plugin do họ tự phát triển để phục vụ cho từng nhu cầu cụ thể.

Luận văn được trình bày theo năm phần:

Mở đầu: Giới thiệu vấn đề, mục tiêu của đề tài

Chương 1: Trình bày các kiến thức nền tảng về đặc tả yêu cầu và ngôn ngữ mô hình hóa chuyên biệt miền. Giới thiệu sơ lược về các công cụ hỗ trợ sử dụng trong quá trình xây dựng ngôn ngữ.

Chương 2: Diễn giải về miền đặc tả ca sử dụng. Trình bày về cú pháp trừu tượng và cú pháp cụ thể của FRSL. Giới thiệu về một số chuyển đổi từ đặc tả FRSL.

Chương 3: Áp dụng và xây dựng công cụ hỗ trợ. Vận dùng và trình bày kết quả thực nghiệm.

Kết luận: Kết quả đạt được và hướng phát triển.

CHƯƠNG 1. Kiến thức nền tảng

Chương này sẽ giới thiệu về cơ sở lý thuyết cần thiết của luận văn. Để xác định rõ các vấn đề về đặc tả yêu cầu và cách giải quyết chúng, luận văn cần phải nghiên cứu kĩ về các khía cạnh của đặc tả yêu cầu, tập trung vào yêu cầu chức năng thể hiện qua hệ thống ca sử dụng. Đồng thời, luận văn nghiên cứu theo hướng tiếp cận chuyên biệt miền, giúp thu hẹp phạm vi vấn đề để có thể đưa ra giải pháp đạt độ tối ưu và chất lượng cao hơn.

Phần đầu của chương sẽ trình bày những định nghĩa, tính chất, vai trò của yêu cầu và đặc tả yêu cầu, đồng thời nhấn mạnh vào những khó khăn trong quá trình xây dựng đặc tả yêu cầu. Luận văn sẽ tập trung vào yêu cầu chức năng và cách thể hiện yêu cầu chức năng thông qua mô hình ca sử dụng, cùng những đặc tả hỗ trợ ràng buộc.

Phần kế tiếp là những khái niệm về chuyên biệt miền, mô hình hóa chuyên biệt miền và ngôn ngữ mô hình hóa chuyên biệt miền. Những kiến thức và kĩ thuật áp dụng ngôn ngữ mô hình hóa chuyên biệt miền được đề cập đến để phục vụ cho quá trình nghiên cứu và xây dựng trong các phần tiếp theo.

Ngoài ra, chương này cũng sẽ giới thiệu một số công cụ hỗ trợ cho quá trình xây dựng ngôn ngữ đặc tả ca sử dụng. Mỗi công cụ sẽ được trình bày những đặc điểm và vai trò cụ thể, cùng với phương pháp áp dụng và tích hợp vào hệ thống, kèm theo những ví dụ minh họa.

1.1. Đặc tả yêu cầu

Yêu cầu có thể được hiểu là một mệnh đề dùng để thể hiện một nhu cầu sử dụng và các ràng buộc, điều kiện liên quan của nó. Theo [6] thì kỹ nghệ yêu cầu là "một chức năng liên ngành làm trung gian giữa các lĩnh vực của bên mua và nhà cung cấp để thiết lập và duy trì các yêu cầu được đáp ứng bởi hệ thống, phần mềm hoặc dịch vụ sử dụng". Kỹ nghệ yêu cầu chú trọng đến việc phát hiện, khám phá, phát triển, phân tích, xác định phương pháp xác minh, xác thực, truyền đạt, ghi chép và quản lý các yêu cầu. Kết quả đạt được của kỹ nghệ yêu cầu là một hệ thống phân cấp các yêu cầu sao cho đạt được sự hiểu biết chung giữa các bên liên quan, ví dụ như người đặt hàng phần mềm, nhà cung cấp, nhà điều hành, người dùng,... Ngoài ra, hệ thống phải được xác thực so với những yêu cầu của thế giới thực để xác nhận khả năng thực hiện.

Các bên liên quan (stakeholders) có thể khác nhau giữa các dự án phần mềm trong bối cảnh kỹ nghệ yêu cầu. Một nhóm tối thiểu các bên liên quan sẽ bao gồm hai thành phần là người sử dùng và người đặt hàng / mua phần mềm. Hai vai trò này có thể không phải là cùng một đối tượng. Các dự án phức tạp có thể bao gồm nhiều người sử dụng và nhiều người mua, mỗi bên có những mối quan tâm khác nhau. Ngoài ra, các yêu cầu của dự án có thể bao gồm thêm hai nhóm khác vào thành phần các bên liên quan tối thiểu. Nhóm đầu tiên là nhóm tổ chức phát triển, bảo trì hoặc vận hành hệ thống hoặc phần mềm có lợi ích hợp pháp trong việc hưởng lợi từ hệ thống. Nhóm thứ hai là các cơ quan quản lý với các yêu cầu theo luật định, yêu cầu ngành nghề hoặc các yêu cầu bên ngoài khác đòi hỏi những phân tích cẩn thận.

Quá trình chuyển hóa và xác định các yêu cầu bắt đầu với những ý định của các bên liên quan (gọi là nhu cầu, mục đích hoặc mục tiêu), được phát triển thành một mệnh đề chính thức hơn trước khi trở thành những yêu cầu hợp lệ của các bên liên quan. Ý định ban đầu của các bên liên quan không thể được sử dụng như các yêu cầu của các bên liên quan vì chúng thường thiếu những định nghĩa, phân tích và đôi khi là tính nhất quán và tính khả thi. Yêu cầu kĩ thuật sử dụng những khái niệm hoạt động để hỗ trợ khả năng hiểu ý định của các bên liên quan ở cấp độ có tổ chức, và các khái niệm vận hành hệ thống từ góc độ hệ thống, từ đó chuyển hóa giúp các bên liên quan từ những ý định ban đầu thành các mệnh đề yêu cầu có cấu trúc và chính thức hơn.

Tài liệu đặc tả yêu cầu thể hiện các yêu cầu phần mềm dưới một dạng cụ thể, hỗ trợ cho quá trình giao tiếp, sao chép, lưu trữ,... Tài liệu đặc tả yêu cầu cần phải đề cập đến mục tiêu và phạm vi của phần mềm, với các diễn đạt về những việc mà phần mềm sẽ làm, những lợi ích, đối tượng và mục đích mà phần mềm được ứng dụng. Bên cạnh đó là những thông tin quan hệ với các hệ thống khác, kết nối bên ngoài với những phần mềm là một phần của các hệ thống lớn hơn, những ràng buộc phần mềm về mặt hệ thống, người dùng, phần cứng, giao diện giao tiếp, bộ nhớ,... Ngoài những trình bày về các yêu cầu, tài liệu đặc tả cũng cần cung cấp những thông tin khác như hạn chế của phần mềm, các lệ thuộc bên ngoài có thể ảnh hưởng tới yêu cầu, các chuẩn dữ liệu và thủ tục được sử dụng,...

Có thể chia yêu cầu thành hai loại: yêu cầu chức năng và yêu cầu phi chức năng. Trong kỹ thuật phần mềm, một yêu cầu phi chức năng là một yêu cầu phần mềm được sử dụng không phải để diễn tả phần mềm sẽ làm gì, mà sẽ diễn tả phần mềm sẽ làm điều đó như thế nào. Yêu cầu phi chức năng có nhiều thành phần, trong đó có thể kể đến các yêu cầu về hiệu năng phần mềm, yêu cầu giao diện bên

ngoài, các ràng buộc thiết kế phần mềm, các thuộc tính đánh giá chất lượng phần mềm,... Các yêu cầu phi chức năng thường rất khó để kiểm chứng, vì vậy nên chúng thường được đánh giá chủ quan. Yêu cầu phi chức năng có vai trò quan trọng đối với chất lượng phần mềm, ảnh hưởng trực tiếp đến tính nhất quán của phần mềm và mức độ thỏa mãn của người sử dụng.

Ngược lại với yêu cầu phi chức năng là các yêu cầu chức năng. Yêu cầu chức năng nắm bắt những hành vi dự kiến của hệ thống. Những hành vi này có thể được thể hiện dưới dạng dịch vụ, tác vụ hoặc chức năng mà hệ thống được yêu cầu thực hiện [10]. Các yêu cầu chức năng có thể liên quan đến tính toán, chi tiết kỹ thuật, điều chỉnh và xử lý dữ liệu, và các chức năng cụ thể khác để mô tả chi tiết những mục tiêu của hệ thống. Các yêu cầu chức năng và các yêu cầu phi chức năng thường hỗ trợ lẫn nhau, giúp áp đặt các ràng buộc đối với thiết kế hoặc triển khai, đảm bảo những yêu cầu về hiệu suất, bảo mật hoặc độ tin cậy.

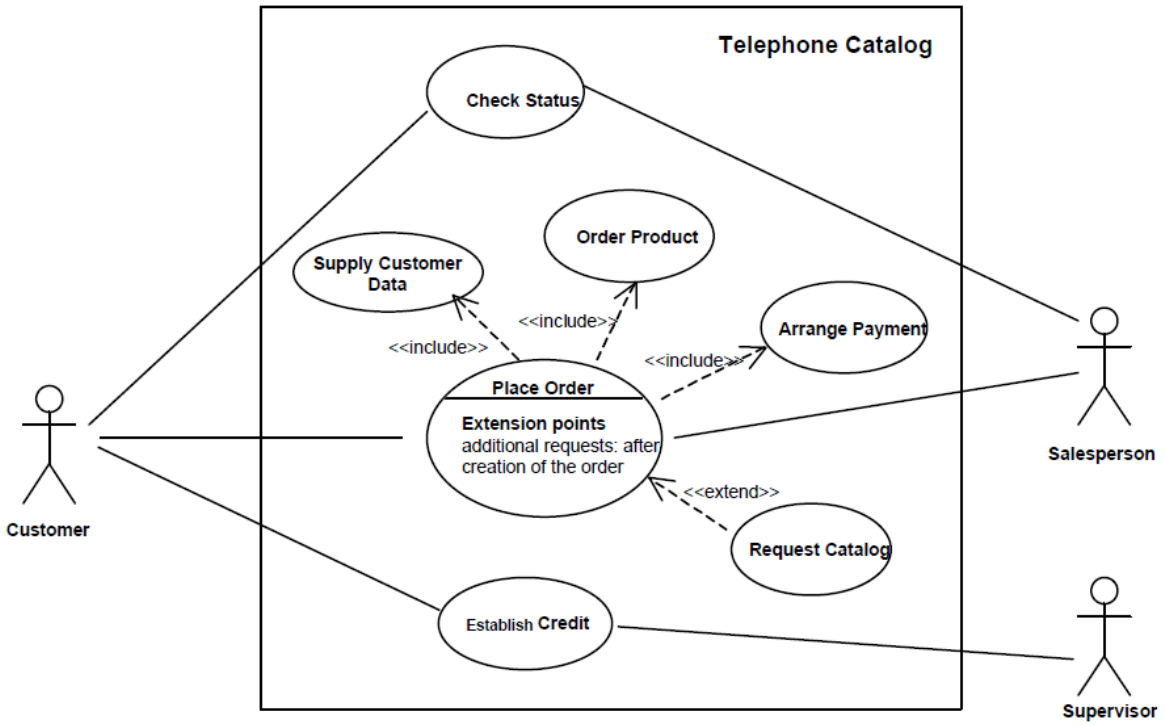
1.2. Ca sử dụng

Ca sử dụng (use case) đã và đang đóng vai trò là một thông lệ phổ biến để nắm bắt các yêu cầu chức năng [10]. Chúng đặc biệt phát huy trong cộng đồng hướng đối tượng, nơi khởi nguồn của ca sử dụng, nhưng khả năng ứng dụng của chúng không chỉ giới hạn ở các hệ thống hướng đối tượng. Ca sử dụng xác định một tập hợp các tương tác hướng mục tiêu giữa các tác nhân bên ngoài và hệ thống đang được xây dựng. Các tác nhân (actors) là các thành phần bên ngoài tương tác với hệ thống. Một tác nhân có thể là một lớp người dùng, vai trò mà người dùng có thể sử dụng, hoặc các hệ thống khác. Các tác nhân được phân biệt giữa hai loại là tác nhân chính (primary actor) và tác nhân thứ cấp (secondary actor). Tác nhân chính là người có mục tiêu cần sự trợ giúp của hệ thống. Tác nhân thứ cấp là người mà hệ thống cần sự trợ giúp.

Một ca sử dụng được bắt đầu bởi một người dùng có mục tiêu cụ thể, và kết thúc thành công khi mục tiêu đó được thỏa mãn. Nó mô tả chuỗi tương tác cần thiết giữa các tác nhân và hệ thống để cung cấp dịch vụ thỏa mãn mục tiêu. Ca sử dụng cũng bao gồm nhiều chuỗi thay thế có thể có, ví dụ như các chuỗi tương tác khác nhau cùng có thể thỏa mãn mục tiêu, cũng như các chuỗi có thể dẫn đến việc không hoàn thành dịch vụ do những hành vi ngoại lệ, xử lý lỗi. Các tương tác với hệ thống, bao gồm các phản hồi của hệ thống, được cảm nhận từ bên ngoài hệ thống. Vì vậy, ca sử dụng sẽ thể hiện rằng ai tương tác như thế nào với hệ thống, với mục đích gì, mà không phải giải thích chi tiết về các thành phần bên trong của

hệ thống. Một tập đầy đủ các ca sử dụng trình bày tất cả các cách khác nhau để sử dụng hệ thống, và từ đó xác định tất cả các hành vi cần thiết của hệ thống, giới hạn phạm vi của hệ thống.

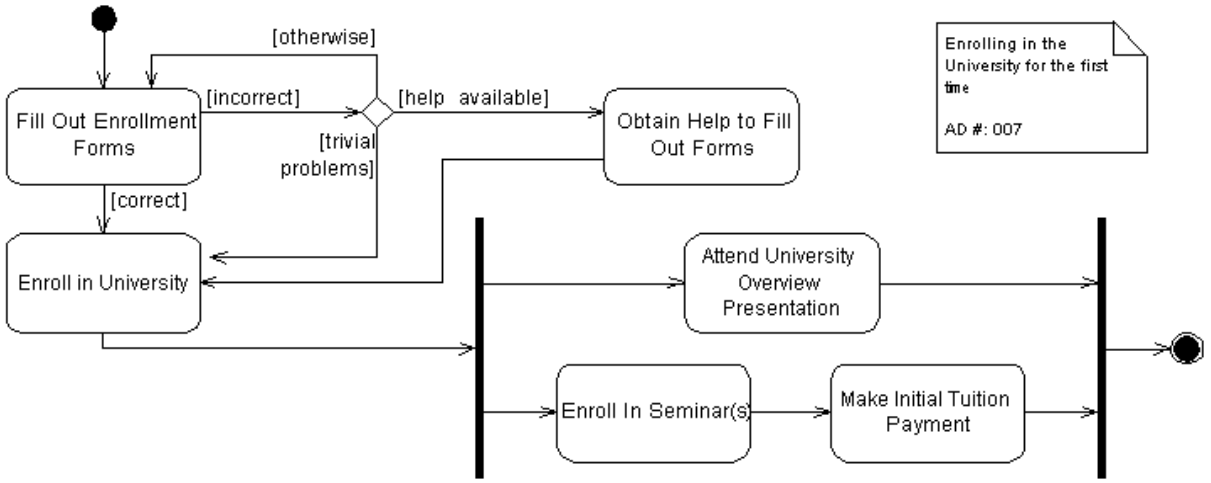
Ca sử dụng thường được viết theo cấu trúc dễ hiểu, kết hợp với sử dụng các từ vựng của miền. Điều này giúp cho người dùng có thể dễ dàng theo dõi và xác thực các thông tin của ca sử dụng, khuyến khích người dùng tham gia tích cực vào việc xác định và đặc tả các yêu cầu. Các ca sử dụng được thể hiện thông qua các kịch bản (scenario), trình bày một quy trình tương tác từ đầu đến cuối của ca sử dụng. Người dùng có thể xây dựng một kịch bản cho luồng chính đi qua ca sử dụng, và các kịch bản khác cho từng biến thể của luồng tương tác ca sử dụng, được kích hoạt bởi các tùy chọn, điều kiện lỗi, vi phạm bảo mật,... Các kịch bản có thể được mô tả bằng các sơ đồ trình tự (sequence diagram) và sơ đồ hoạt động (activity diagram).



Hình 1.1: Sơ đồ ca sử dụng.

Hình 1.1 là một ví dụ về ca sử dụng được thể hiện thông qua một sơ đồ ca sử dụng. Trong đó, các đối tượng "Customer", "Salesperson" và "Supervisor" là các tác nhân, thể hiện bằng biểu tượng hình người. Các hình elip như "Check status" hay "Place Order" là các ca sử dụng, và hình vuông bao bọc "Telephone Catalog" thể hiện hệ thống phần mềm. Các đường nối từ tác nhân tới ca sử dụng thể hiện mối quan hệ giữa hai đối tượng rằng tác nhân này sẽ là người yêu cầu /

kích hoạt ca sử dụng đó. Những mũi tên đứt đoạn là các liên kết phụ thuộc giữa các ca sử dụng với nhau, với từ khóa <<include>> và <<extend>> xác định loại quan hệ phụ thuộc.



Hình 1.2: Sơ đồ hoạt động của ca sử dụng.

Hình 1.2 là một ví dụ về kịch bản / luồng hành vi của ca sử dụng được thể hiện bằng một sơ đồ hoạt động. Dấu tròn đen là điểm khởi đầu của luồng, đánh dấu vị trí bắt đầu để duyệt qua sơ đồ. Những hình chữ nhật tròn góc là các bước hành động, ví dụ như "Fill Out Enrollment Form" và "Enroll in University". Hình thoi thể hiện bước điều kiện, với mỗi nhánh đi ra từ hình thoi tượng trưng cho hành động tiếp theo của luồng dựa vào kết quả được đánh giá. Nét gạch đậm là các bước xảy ra đồng thời, với hai hoặc nhiều hơn các chuỗi hành động được vẽ song song với nhau. Cuối cùng là dấu tròn đen có viền ngoài thể hiện điểm kết thúc, đóng lại chuỗi hoạt động của một luồng hành vi.

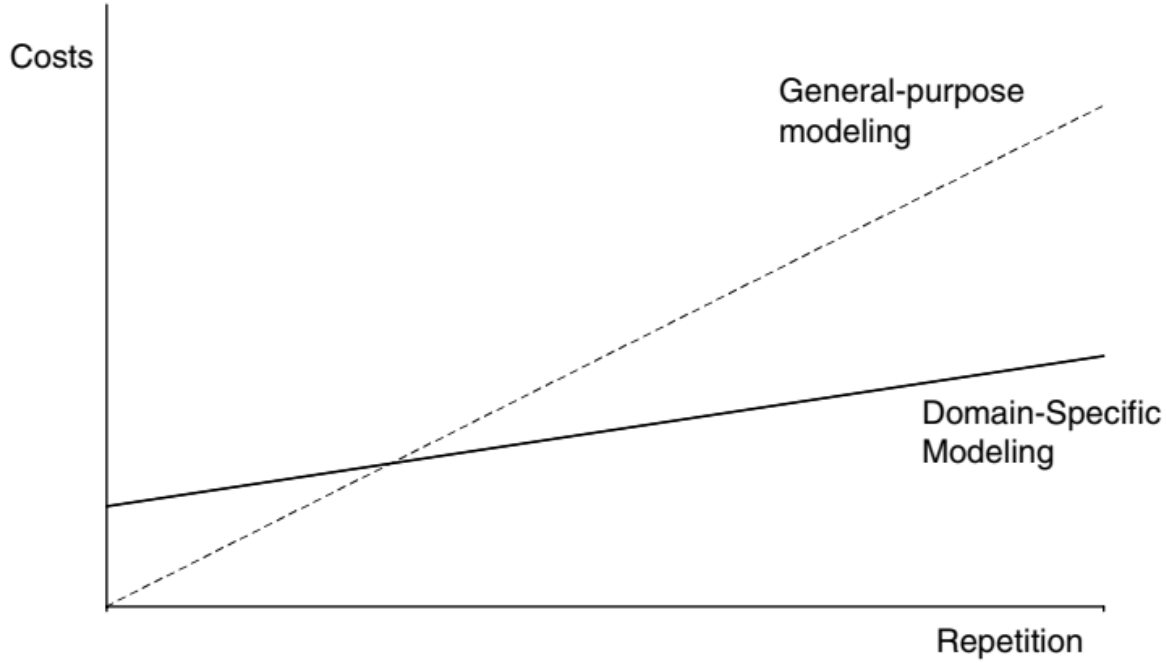
1.3. Ngôn ngữ mô hình hóa chuyên biệt miền

Nghiên cứu theo hướng chuyên biệt miền giúp thu hẹp phạm vi vấn đề, từ đó xây dựng giải pháp giải quyết vấn đề một cách chi tiết và chuyên sâu hơn. Đối với vấn đề phân tích và đặc tả yêu cầu, luận văn tập trung trình bày các khái niệm về ngôn ngữ mô hình hóa chuyên biệt miền, với các phương pháp thiết kế và xây dựng các thành phần chính của ngôn ngữ.

1.3.1. Mô hình hóa chuyên biệt miền

Mô hình hóa chuyên biệt miền (Domain-specific modeling – DSM) là một phương pháp kỹ thuật phần mềm được sử dụng để thiết kế và phát triển các hệ

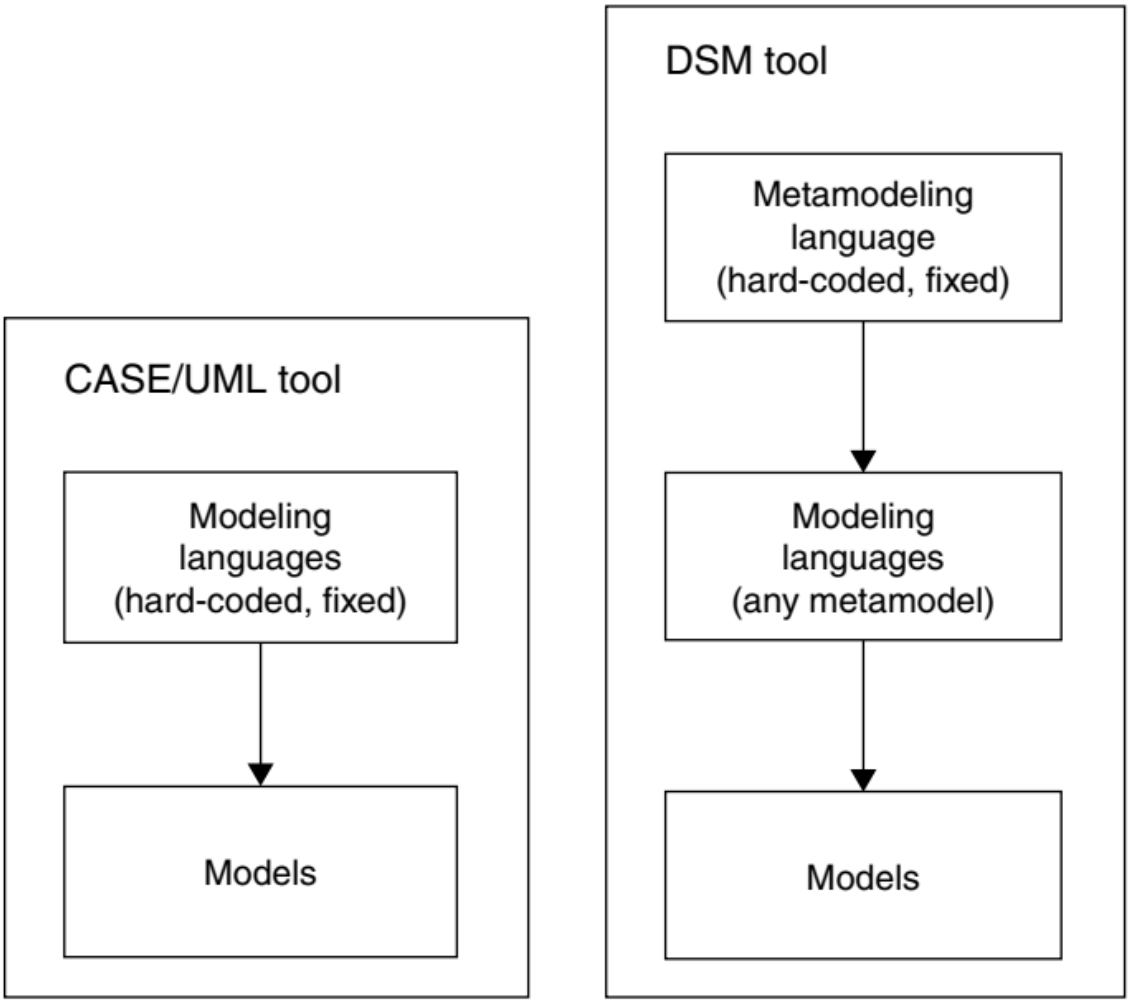
thống phần mềm máy tính [8]. Phương pháp này tập trung vào tự động hóa phát triển phần mềm trong một phạm vi miền vấn đề hẹp. Như ta có thể thấy từ tên của phương pháp, mô hình hóa chuyên biệt miền tập trung vào một hoặc một số miền cụ thể có liên quan đến nhau, chứ không nhắm tới giải quyết mục đích chung. Chi phí phát triển theo hướng mô hình hóa chuyên biệt miền cao hơn so với thông thường, tuy nhiên khả năng tái sử dụng nhiều lần bởi việc tập trung vào một phạm vi hẹp sẽ giảm chi phí phát sinh về lâu dài (Hình 1.3). Càng thu hẹp và hạn chế phạm vi, phương pháp lại càng dễ dàng hơn trong việc cung cấp hỗ trợ cho công việc đặc tả và tự động hóa các công việc lập trình thủ công. Không giống như cách tiếp cận mục đích tổng quát, mô hình hóa chuyên biệt miền có thể hỗ trợ các tác vụ phát triển phần mềm, do ngôn ngữ mô hình hóa biết về miền vấn đề và các hệ thống sinh tự động có thể chu cấp cho miền giải pháp, hỗ trợ phía triển khai.



Hình 1.3: Hiệu quả của mô hình hóa chuyên biệt miền so với đa tính năng.

Đối với đa số các lập trình viên, một phạm vi hẹp có thể được xác định bằng cách cung cấp một ngôn ngữ hoạt động trên các khái niệm đã biết có liên quan đến tên miền cụ thể và có các luật hướng dẫn các lập trình viên trong việc thực hiện đặc tả. Ví dụ, một ngôn ngữ có thể ngăn chặn các thiết kế sai lệch hoặc chất lượng kém bằng một cách đơn giản là không cho phép người dùng chỉ định chúng. Điều này giúp ngăn ngừa lỗi sớm trong quá trình phát triển phần mềm, khi những lỗi này tốn kém ít nhất để sửa. Điều đó có thể được thực hiện trong mỗi bước xây dựng mô hình bằng cách kiểm tra các mô hình đó tuân theo một metamodel (siêu mô hình) hoặc trong một quy trình kiểm tra mô hình riêng biệt.

Với metamodel, ngôn ngữ có thể ngăn việc tạo các kết nối bất hợp pháp giữa các thành phần mô hình nhất định hoặc buộc người lập mô hình phải chỉ định những dữ liệu nhất định. Đối với quy trình kiểm tra riêng biệt, kiểm tra mô hình có thể báo cáo các cấu trúc bất hợp pháp hoặc các thiết kế không hoàn chỉnh.



Hình 1.4: Hướng tiếp cận của mô hình hóa chuyên biệt miền so với UML.

Việc thu hẹp phạm vi giúp tăng mức độ trừu tượng vượt ra ngoài các ngôn ngữ lập trình hiện tại bằng cách chỉ định giải pháp trực tiếp thông qua sử dụng các khái niệm miền. Sự thay đổi tăng dần về độ trừu tượng nói chung dẫn đến tăng năng suất tương ứng. Năng suất được cải thiện không chỉ ở thời gian và nguồn lực cần thiết từ bước đặc tả đầu tiên mà còn liên quan đến giai đoạn bảo trì. Ví dụ: các thay đổi yêu cầu thường đến thông qua miền vấn đề, không phải miền triển khai, vì vậy những thay đổi đó được chỉ định một cách tự nhiên nhất bằng cách sử dụng cùng những thuật ngữ tên miền.

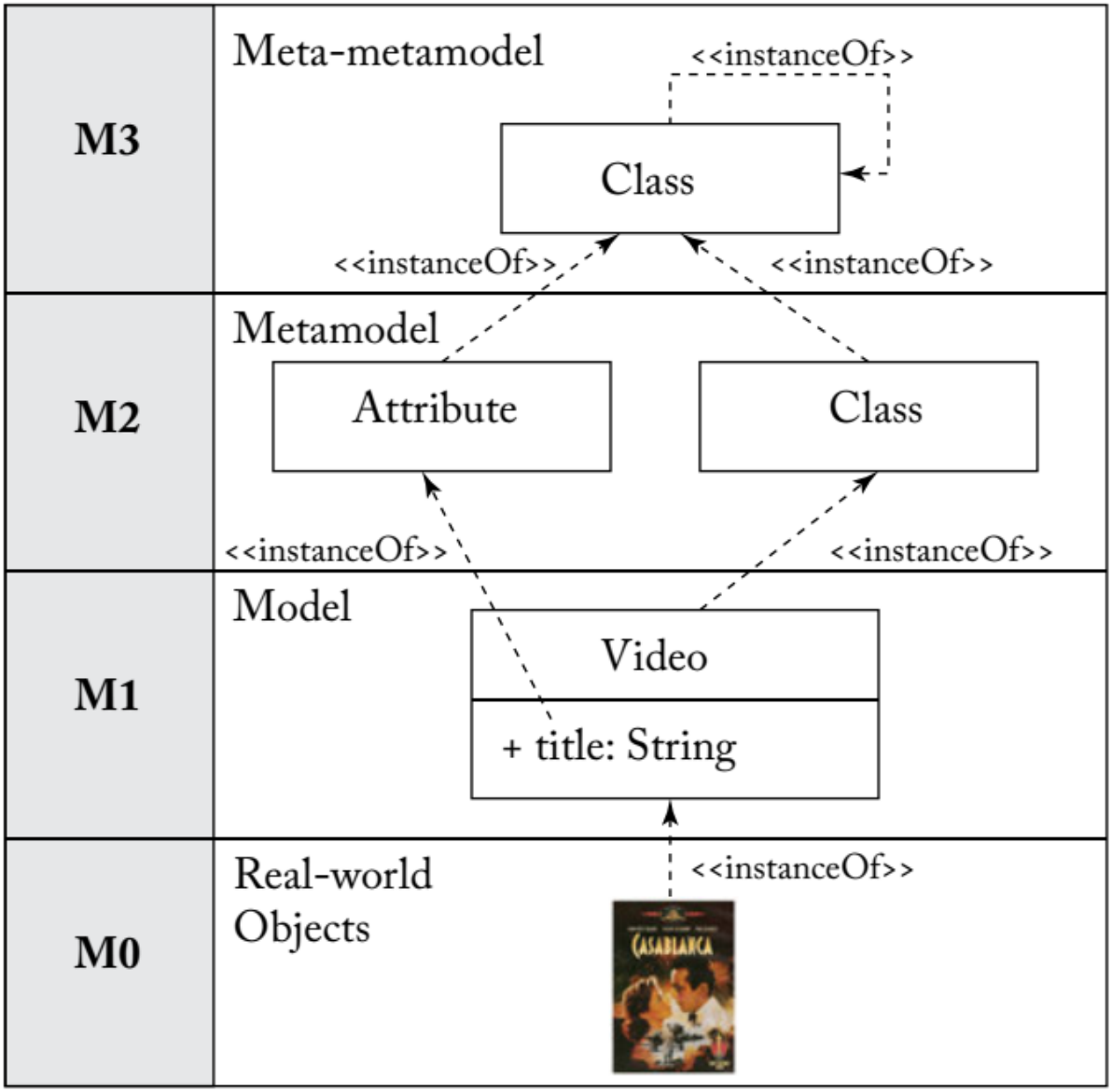
Mô hình hóa chuyên biệt miền về cơ bản làm tăng mức độ trừu tượng đồng thời thu hẹp không gian thiết kế, thường chỉ tập trung vào một phạm vi sản phẩm

nhỏ như là cho một công ty duy nhất. Với mô hình hóa chuyên biệt miền, vấn đề chỉ cần được giải quyết một lần bằng cách mô hình hóa trực quan giải pháp bằng việc sử dụng các khái niệm miền quen thuộc. Các sản phẩm cuối cùng sau đó có thể được tạo tự động từ các đặc tả cấp cao này với các trình sinh mã nguồn cụ thể theo miền. Trong phần lớn các quy trình phát triển phần mềm, các ngôn ngữ mô hình hóa mục đích chung không thể hoàn toàn phát triển theo hướng mô hình hóa, vì các mô hình cốt lõi ở mức độ trừu tượng tương đương các ngôn ngữ lập trình đang được sử dụng. Lợi ích của mô hình hóa trực quan không bù đắp được các tài nguyên cần được sử dụng để giữ cho tất cả các mô hình và mã nguồn được đồng bộ hóa vì chỉ hỗ trợ bán tự động. Vì vậy nên với một phạm vi cụ thể, các ngôn ngữ chuyên biệt miền luôn hoạt động tốt hơn các ngôn ngữ có mục đích chung.

1.3.2. Khái niệm về ngôn ngữ mô hình hóa chuyên biệt miền

Ngôn ngữ chuyên biệt miền (Domain-specific Language) là ngôn ngữ lập trình hoặc ngôn ngữ đặc tả thực thi tập trung vào và thường bị giới hạn trong một miền vấn đề cụ thể, thông qua các ký hiệu và mức độ trừu tượng thích hợp [1] [9]. Ngôn ngữ chuyên biệt miền được thiết kế để hoạt động chuyên sâu cho một nhóm nhiệm vụ cụ thể. Những ngôn ngữ đi theo hướng mô hình hóa được gọi là ngôn ngữ mô hình hóa chuyên biệt miền. Những lợi thế tiềm năng của ngôn ngữ chuyên biệt miền bao gồm giảm thời gian hoàn thành sản phẩm, giảm chi phí bảo trì, tăng tính di động, độ tin cậy, tối ưu hóa và có khả năng kiểm tra cao hơn.

Một trong những điều kiện tiên quyết để thiết kế ngôn ngữ chuyên biệt miền là khâu phân tích chi tiết và xây dựng cấu trúc cho miền ứng dụng. Những hướng tiếp cận và áp dụng để thu thập được những thông tin đó được cung cấp bởi ngành nghiên cứu phân tích miền (domain analysis), chuyên nghiên cứu về các cách mô hình hóa miền. Những nghiên cứu này kiểm tra các nhu cầu và yêu cầu của một bộ sưu tập các hệ thống có những sự giống nhau nhất định. Phân tích miền bắt nguồn từ nghiên cứu tái sử dụng phần mềm và có thể được ứng dụng để xây dựng các thư viện, nền tảng, ngôn ngữ hoặc dòng sản phẩm theo hướng chuyên biệt miền. Kết quả quan trọng nhất của phân tích miền là một mô hình tính năng (feature model). Một mô hình tính năng sẽ chỉ ra các tính tương đồng và tính biến đổi của các thành phần phần mềm, cũng như những mối quan hệ phụ thuộc giữa các tính năng dễ thay đổi. Mô hình tính năng thu thập và lưu trữ thông tin về tính năng, các bên liên quan, các ràng buộc (ví dụ như các tính năng có thể loại trừ lẫn nhau), các vị trí kết nối và các ưu tiên.

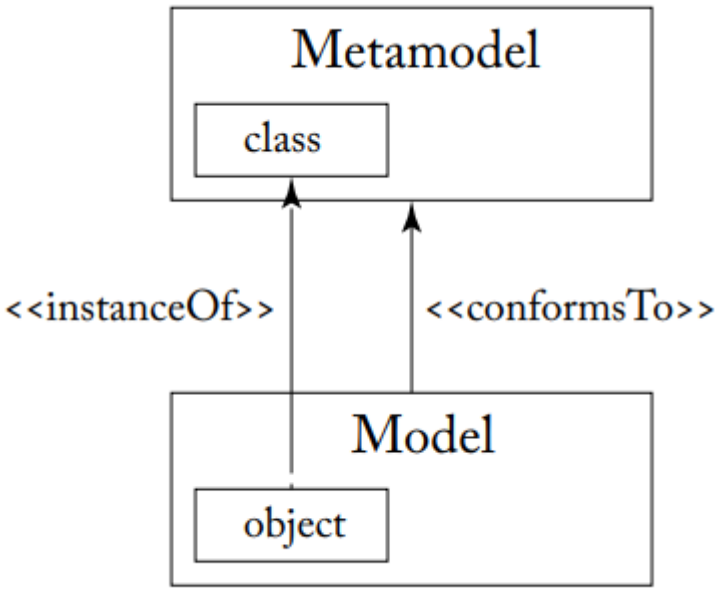


Hình 1.5: Cấu trúc hướng phát triển metamodel.

Một thành phần quan trọng đối với ngôn ngữ mô hình hóa chuyên biệt miền là metamodel (siêu mô hình) [8]. Trong phát triển theo hướng mô hình hóa, một cách để định nghĩa các mô hình là thể hiện chính các mô hình đó như là các phiên bản của các loại mô hình trừu tượng hơn. Do đó, cũng giống như cách ta định nghĩa một mô hình là sự trừu tượng của các hiện tượng trong thế giới thực, ta có thể định nghĩa một metamodel là một mức trừu tượng hóa cao hơn, làm nổi bật các thuộc tính của chính mô hình đó. Metamodel sẽ tạo thành định nghĩa của ngôn ngữ mô hình hóa, vì nó cung cấp một mô tả của toàn bộ lớp mô hình có thể được biểu thị bằng ngôn ngữ đó (Hình 1.5).

Về mặt lý thuyết, ta có thể định nghĩa vô hạn mức độ (meta)model, tức là ta có thể mô hình hóa metamodel thành meta-metamodel để mô tả những tính chất của metamodel, và cứ tiếp tục như vậy. Nhưng trên thực tế, các áp dụng đã chứng

minh được rằng meta-metamodel có thể tự định nghĩa dựa trên chính nó, và do đó mức độ trừu tượng thường chỉ dừng lại ở metamodel hoặc meta-metamodel và thường không có ý nghĩa khi vượt quá mức độ trừu tượng này. Một mô hình tuân theo metamodel của nó giống như cách một chương trình máy tính tuân theo ngữ pháp của ngôn ngữ lập trình mà nó được viết. Cụ thể hơn, một mô hình tuân theo metamodel của nó khi tất cả các phần tử của mô hình đó có thể được biểu diễn dưới dạng các thể hiện của các lớp metamodel tương ứng (Hình 1.6).



Hình 1.6: Mối quan hệ giữa mô hình và metamodel.

1.3.3. Xây dựng ngôn ngữ mô hình hóa chuyên biệt miền

Một ngôn ngữ chuyên biệt miền được cấu tạo từ hai thành phần chính: cú pháp trừu tượng và cú pháp cụ thể. Trong đó, cú pháp trừu tượng đóng vai trò thể hiện khả năng diễn đạt của ngôn ngữ, phạm vi tổng hợp và mô tả nội dung mà ngôn ngữ tập trung trình bày. Cú pháp cụ thể là thành phần ngữ pháp của ngôn ngữ, đề ra những quy tắc, giới hạn và khuôn mẫu mà người sử dụng ngôn ngữ cần phải tuân theo. Để xây dựng hai thành phần trên của ngôn ngữ chuyên biệt miền, trước tiên cần phải xác định được miền vấn đề. Quá trình xác định miền được thực hiện bằng cách định nghĩa các khái niệm miền (meta-concepts), các yếu tố về cấu trúc điển hình của miền. Các khái niệm này sẽ làm cơ sở để thể hiện các thành phần thực tế của vấn đề dưới dạng các thành phần dữ liệu, giúp phân loại và đóng gói các đối tượng, và thể hiện các ràng buộc giữa các đối tượng. Sau khi đã định nghĩa được miền vấn đề, ta sẽ xây dựng bộ cú pháp trừu tượng và cụ thể từ những khái niệm miền.

Các thành phần dữ liệu thông tin của ngôn ngữ mô hình hóa chuyên biệt miền sẽ được xây dựng thành các mô hình, vậy nên cú pháp trừu tượng sẽ cần phải thể hiện được cấu trúc thông tin của các mô hình đó và vì thế cú pháp trừu tượng sẽ được xây dựng theo hướng metamodel. Để thể hiện các thành phần của miền như mô tả văn bản, hành vi thể hiện hay các ràng buộc, ta coi chúng như là các toán tử của một mô hình hướng đối tượng. Từ đó, ta có thể phân loại và liên kết chúng với nhau, đặt những điều kiện ràng buộc và tổng hợp lại thành một cấu trúc thống nhất, và dựa vào đó xây dựng hệ thống metamodel trên môi trường tích hợp.

Để xây dựng cú pháp cụ thể cho ngôn ngữ mô hình hóa chuyên biệt miền, cụ thể hơn là một cú pháp cụ thể dưới dạng văn bản, ta cần xây dựng được hai thành phần: một khuôn mẫu mô tả và một bộ luật giới hạn. Khuôn mẫu sẽ đóng vai trò định hướng cấu trúc của hệ thống ngữ pháp, chỉ ra các thành phần mở đầu, kết thúc của từng mục, cấu trúc phân cấp và quan hệ. Các thành phần trong khuôn mẫu sẽ chỉ định những thông tin có thể được cung cấp trong văn bản đặc tả. Bộ luật giới hạn được sử dụng để chính xác hóa các thông tin của văn bản đặc tả, ví dụ như kiểu dữ liệu, thành phần bắt buộc, không bắt buộc, các từ khóa và phương pháp sử dụng chúng,... Đối với cú pháp cụ thể dưới dạng văn bản, bộ luật cũng có thể bao gồm các hạn chế trong cách sử dụng ngôn ngữ tự nhiên.

1.4. Một số công cụ hỗ trợ

Luận văn đã nghiên cứu và áp dụng một số công cụ có sẵn để hỗ trợ cho quá trình xây dựng ngôn ngữ đặc tả ca sử dụng. Đầu tiên là công cụ ANTLR, một chương trình sinh bộ phân tích dữ liệu dưới dạng văn bản. ANTLR được sử dụng để xây dựng cú pháp cụ thể cho ngôn ngữ đặc tả, cung cấp khả năng phân loại và chuyển hóa dữ liệu đầu vào thành các thành phần mô hình cụ thể. Công cụ thứ hai là PlantUML, với tác dụng biểu diễn dữ liệu dưới dạng văn bản thành các sơ đồ ca sử dụng. PlantUML cũng góp phần chứng minh khả năng mở rộng và tích hợp nhiều loại công cụ khác vào hệ thống, phát triển thêm các tính năng và đáp ứng nhiều nhu cầu khác nhau.

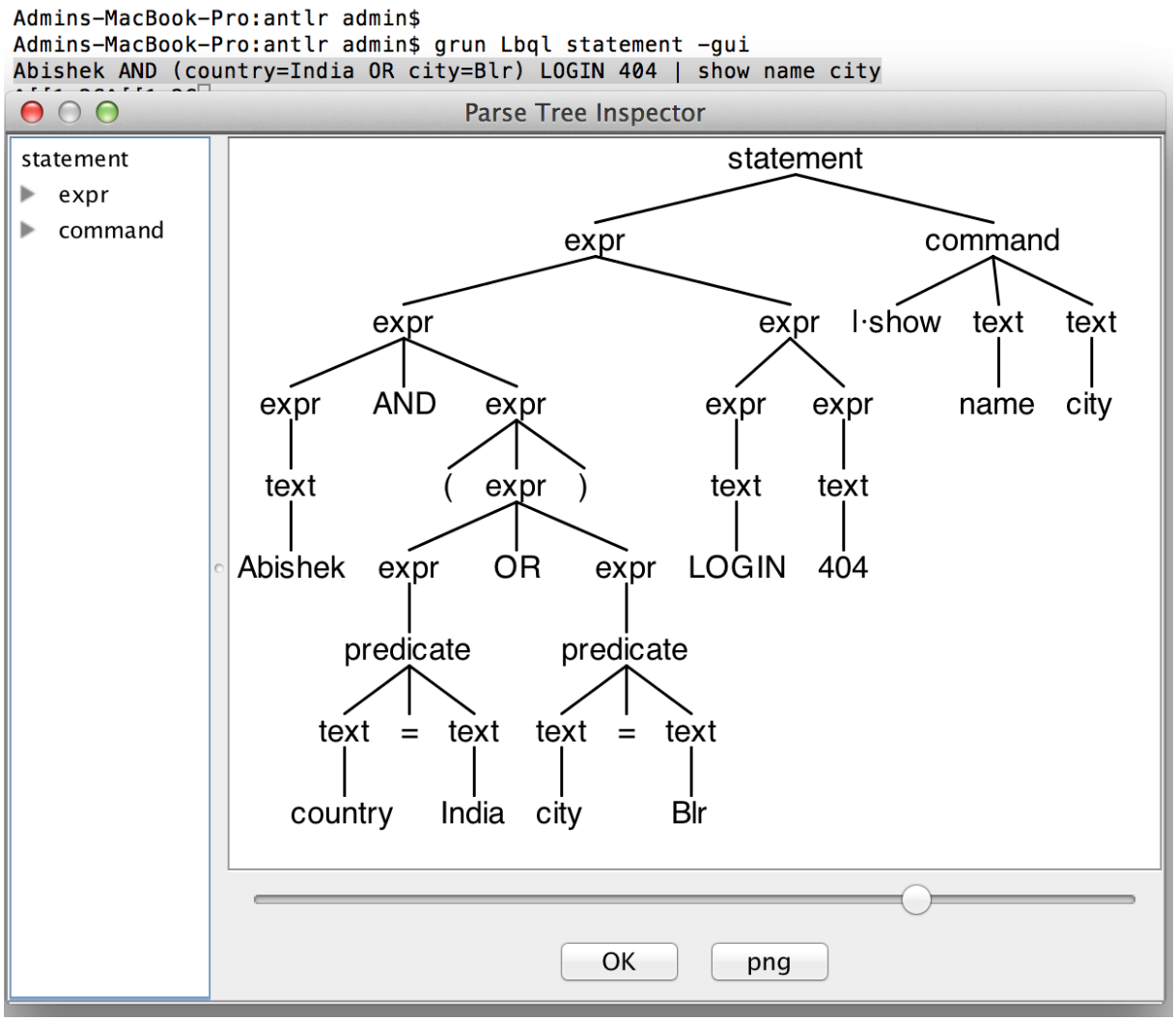
1.4.1. Công cụ ANTLR

Công cụ ANTLR là viết tắt của Another Tool for Language Recognition, là một chương trình tự động sinh những bộ phân tích cú pháp dùng để đọc, xử lý,

thực thi hoặc dịch những thông tin dưới dạng văn bản hoặc nhị phân. Từ những ngữ pháp được xây dựng trước, ANTLR sẽ xây dựng nên các trình phân tích nhằm tổng hợp dữ liệu thành các cây cú pháp trừu tượng và duyệt chúng. ANTLR được sử dụng rộng rãi trong nghiên cứu và các lĩnh vực xây dựng công cụ, ngôn ngữ hay các nền tảng. Theo những thống kê trên trang web chính thức của ANTLR, Twitter sử dụng ANTLR để phân tích hơn 2 tỉ truy vấn mỗi ngày. ANTLR cũng được sử dụng trong các ngôn ngữ như Hive và Pig, và trong hệ cơ sở dữ liệu Hadoop. Oracle sử dụng ANTLR trong bộ IDE cho nhà phát triển SQL, và NetBeans phân tích C++ với ANTLR.

ANTLR cho phép người phát triển tự định nghĩa bộ luật để đọc văn bản đầu vào, sử dụng hai thành phần luật chính của ANTLR là luật từ vựng (lexer rule) và luật phân tích (parser rule). Luật từ vựng phân tích từng kí tự văn bản đầu vào để phân loại chúng ra thành những dấu hiệu (token) dựa trên những thuộc tính chung của các kí tự. Ví dụ, người dùng có thể định nghĩa các kí tự "[0-9]" là các chữ số, và khi nhiều chữ số đứng liên tiếp nhau "[0-9]+" sẽ tạo thành dấu hiệu Số tự nhiên. Với một bộ luật đầy đủ, người phát triển sẽ xác định được tất cả các loại dấu hiệu có thể xuất hiện trong văn bản, phát hiện các từ khóa. Luật từ vựng cũng có thể được dùng để tạo ra các ràng buộc về mặt ngữ pháp, vị dụ như nếu đã có dấu mở ngoặc thì sẽ phải có dấu đóng ngoặc.

Bộ luật thứ hai của ANTLR là luật phân tích. Luật từ vựng đóng vai trò xác định các dấu hiệu từ đoạn văn bản, và luật phân tích sẽ dựa vào vị trí, thứ tự của các dấu hiệu đó để phân loại văn bản thành các câu, các lệnh. Chương trình sẽ phát hiện sự sắp xếp của các từ khóa và dấu hiệu phù hợp để xác định xem câu đó thuộc vào luật nào đã được định nghĩa. Mỗi luật phân tích sau đó sẽ được ANTLR sinh ra thành mã nguồn Java tương ứng, hỗ trợ người dùng trong việc thực hiện những biến đổi, xử lý dữ liệu cần thiết tùy theo đó là loại câu lệnh nào. Các luật phân tích được lồng vào với nhau, giúp cho kết quả thu được từ quá trình duyệt văn bản luôn phân cấp theo một cây hoàn chỉnh.



Hình 1.7: Cây phân tích cú pháp xây dựng bởi ANTLR.

Văn bản đầu vào sau khi được phân tích bởi bộ luật của ANTLR sẽ được xuất ra dưới dạng cây cú pháp trừu tượng (abstract syntax tree). Đó là tổng hợp của các thông tin theo cú pháp trừu tượng được phân cấp theo dạng cây. ANTLR cung cấp cho người dùng một bộ duyệt cây (visitor) được sinh ra thành một lớp của ngôn ngữ Java, giúp cho quá trình duyệt cây và thực thi các lệnh trên từng nút của cây được dễ dàng hơn. Qua đó, ANTLR có khả năng tích hợp vào các hệ thống khác cao và rất tiện lợi. Phiên bản mới nhất hiện nay của công cụ ANTLR là ANTLR 4.

1.4.2. Công cụ PlantUML

PlantUML là một công cụ được sử dụng để vẽ các biểu đồ UML, với đầu vào là những đoạn văn bản mang thông tin của biểu đồ theo chuẩn mà công cụ đề ra. PlantUML sử dụng phần mềm Graphviz để sắp xếp và phân tầng các thành

phần trong biểu đồ. Ngôn ngữ văn bản mà công cụ sử dụng có cấu trúc ngữ pháp khá chắc chắn những lại đơn giản và cho phép người dùng có thể dễ dàng đọc. PlantUML được phát triển theo hướng mã nguồn mở và được sử dụng rộng rãi trong các ứng dụng của nhiều lĩnh vực như Eclipse, Google Docs, NetBeans, LibreOffice, Microsoft Word,... Công cụ này đã giúp nhiều người, trong đó có nhiều lập trình viên, làm quen và tự thiết kế các biểu đồ UML. Với đặc tính dễ sử dụng và tích hợp như vậy, PlantUML được dùng để làm ứng dụng đầu tiên cho mô hình đặc tả ca sử dụng sinh ra bởi ngôn ngữ FRSL. Công cụ cũng góp phần chứng minh tính đúng đắn và khả năng mở rộng của mô hình, tạo tiền đề cho nhiều ứng dụng khác sẽ được phát triển sau này.

Hình 1.8: Một biểu đồ đơn giản của công cụ PlantUML.

1.5. Tổng kết chương

Các cơ sở lý thuyết và kiến thức nền tảng cần thiết cho luận văn đã được giới thiệu trong chương vừa rồi. Những trình bày về tính chất, vai trò và khó khăn của yêu cầu và đặc tả yêu cầu đã góp phần làm rõ mục tiêu và phạm vi của luận văn, cụ thể là tập trung vào giải quyết các vấn đề của yêu cầu chức năng thể hiện thông qua mô hình ca sử dụng. Các khái niệm và kĩ thuật về chuyên biệt miền, mô hình hóa chuyên biệt miền và ngôn ngữ mô hình hóa chuyên biệt miền đã trình bày về hướng tiếp cận mà luận văn hướng tới, những vai trò, ưu nhược điểm sẽ được áp dụng và khắc phục trong quá trình nghiên cứu. Bên cạnh đó, các công cụ hỗ trợ cũng đã được giới thiệu với những đặc điểm và vai trò cụ thể, định hướng được khả năng áp dụng để xây dựng ngôn ngữ mô hình hóa chuyên biệt miền cho chương sau.

CHƯƠNG 2. Ngôn ngữ đặc tả ca sử dụng FRSL

Chương này sẽ trình bày về quá trình phát triển ngôn ngữ đặc tả ca sử dụng FRSL (Functional Requirement Specification Language). Từ những khái niệm của miền vấn đề đặc tả ca sử dụng, ngôn ngữ FRSL được xây dựng với cú pháp trừu tượng theo hệ thống metamodel, và cú pháp cụ thể ở dạng văn bản. Đặc tả hoàn thiện của ngôn ngữ sẽ được áp dụng để chuyển đổi thành các dạng khác để phục vụ từng nhu cầu cụ thể. Luận văn cũng sẽ đưa ra những đánh giá sơ bộ và so sánh với các ngôn ngữ, phương pháp đặc tả ca sử dụng khác đã có.

2.1. Giới thiệu

Để có thể sử dụng một cách có hiệu quả các ca sử dụng, người dùng cần có những phương pháp hỗ trợ quá trình phân tích và đặc tả ca sử dụng. Một trong số những phương pháp đó là phát triển và sử dụng một ngôn ngữ đặc tả chuyên biệt về miền vấn đề ca sử dụng. Nhiều nghiên cứu như [2] [4] [12] đã đi theo hướng này và xây dựng những ngôn ngữ tập trung vào một số thành phần của ca sử dụng. Luận văn cũng nghiên cứu và đề xuất ngôn ngữ FRSL được tiếp tục phát triển từ ngôn ngữ RUCM, giúp hoàn thiện cú pháp cho RUCM bằng cách xây dựng cú pháp trừu tượng ở dạng metamodel và cú pháp cụ thể ở dạng văn bản. Ngôn ngữ đặc tả chuyên biệt cho miền ca sử dụng FRSL được phát triển qua bốn bước. Bước đầu là xác định miền, định nghĩa và diễn giải các siêu khái niệm của mô hình miền. Bước thứ hai là thực hiện đặc tả ngôn ngữ và từ đó xây dựng cú pháp trừu tượng dưới dạng metamodel. Bước thứ ba là xây dựng cú pháp cụ thể cho ngôn ngữ ở dạng văn bản, đồng thời định hướng để có thể phát triển cú pháp dưới dạng đồ họa trong tương lai. Bước cuối cùng là xây dựng bộ công cụ hỗ trợ để trực quan hóa các mô hình FRSL. Bước cuối cùng sẽ được trình bày trong chương tiếp theo.

2.2. Miền vấn đề đặc tả ca sử dụng

Xác định miền vấn đề là bước đầu tiên trong quá trình xây dựng một ngôn ngữ chuyên biệt miền. Việc nắm chắc các thông tin, đặc tính của các thành phần trong miền sẽ giúp tăng mức độ chất lượng và hiệu quả của quá trình giải quyết các vấn đề miền, tránh được những sai sót ngay từ những khâu đầu tiên. Theo [10], mô hình miền của miền đặc tả ca sử dụng có các khái niệm sau:

- Ca sử dụng (use case): xác định một tập hợp các tương tác hướng mục tiêu (goal-oriented) giữa các tác nhân bên ngoài và hệ thống đang được xét đến. Ca sử dụng sẽ thể hiện rằng ai (tác nhân) làm gì (tương tác) với hệ thống, cho mục đích gì (mục tiêu). Một tập hợp đầy đủ các ca sử dụng sẽ tổng hợp lại tất cả các cách khác nhau để sử dụng hệ thống và do đó xác định được hết các hành vi cần thiết của hệ thống mà không cần đào sâu vào cấu trúc bên trong của hệ thống.

- Tác nhân (actor): là các thành phần bên ngoài hệ thống có tương tác với hệ thống. Họ có thể là người dùng hoặc các hệ thống khác. Mỗi tác nhân xác định một tập hợp các vai trò mà người dùng của hệ thống có thể thực hiện. Tác nhân được phân biệt giữa các tác nhân chính và phụ. Tác nhân chính là một đối tượng cần đạt được một mục đích từ hệ thống. Tác nhân thứ cấp là đối tượng mà hệ thống cần sự trợ giúp để đáp ứng mục tiêu đó.

- Kịch bản (scenario): là một thể hiện của ca sử dụng và sẽ diễn giải một luồng hành động mà ca sử dụng có thể thực hiện. Một ca sử dụng có thể có một kịch bản cho luồng chính và các kịch bản khác cho từng luồng thay thế mà ca sử dụng có, sinh ra từ những điều kiện lỗi, hành vi không hợp lệ, vi phạm bảo mật,...

- Hệ thống (system): là hệ thống phần mềm đang được xét đến. Hệ thống cung cấp các tính năng giúp thỏa mãn một số mục tiêu nhất định của người sử dụng, thông qua những tương tác với người dùng, các hệ thống khác và cả trong chính bản thân nó.

Ngoài ra, trong từng ca sử dụng cũng có những thuộc tính, tương tác riêng, mô tả chi tiết thông tin của ca sử dụng đó, bao gồm:

- Tên ca sử dụng (use case name): Tên của ca sử dụng, thường bắt đầu bằng một động từ. Thông tin này sẽ luôn nhất quán giữa các sơ đồ và các biểu hiện khác.

- Mô tả tóm tắt (description): Tóm tắt ngắn gọn nội dung, vai trò, mục đích của ca sử dụng.

- Tiền điều kiện (precondition): chỉ những điều kiện của một ca sử dụng mà cần phải được thực hiện hoặc đáp ứng trước khi ca sử dụng bắt đầu. Đây là một ràng buộc về khả năng bắt đầu của ca sử dụng, không phải là sự kiện kích hoạt ca sử dụng.

- Tác nhân chính (primary actor): là đối tượng sẽ khởi đầu cho ca sử dụng này, đối tượng có một mục tiêu cần thỏa mãn.
- Tác nhân thứ cấp (secondary actors): Những đối tượng không trực tiếp khởi đầu ca sử dụng, nhưng hệ thống sẽ lệ thuộc vào họ để hoàn thành ca sử dụng.
- Quan hệ phụ thuộc (dependency): Chứa những thông tin về các mối quan hệ bao gồm (include) và mở rộng (extend) của ca sử dụng này tới các ca sử dụng khác. Trong đó, quan hệ bao gồm chỉ ra rằng các hành vi của ca sử dụng được bao gồm là một phần của ca sử dụng gốc, cho phép khả năng tái sử dụng các hành vi thường gặp và đơn giản hóa các hành vi phức tạp. Quan hệ mở rộng để chỉ các ca sử dụng mở rộng từ ca sử dụng gốc và thêm chức năng vào cho hệ thống. Ca sử dụng mở rộng sẽ phụ thuộc vào nội dung, dữ liệu của ca sử dụng gốc, và thường không thể được sử dụng riêng lẻ.
- Quan hệ tổng quát (generalization): Chứa thông tin về mối quan hệ tổng quát, kế thừa giữa ca sử dụng đó và các ca sử dụng khác. Các hành vi trong ca sử dụng cha sẽ được kế thừa bởi các ca sử dụng con cháu.
- Luồng chính (basic flow): mô tả một chu trình thành công chính theo mong đợi, thỏa mãn lợi ích của các bên liên quan. Luồng chính này thường không bao gồm các so sánh điều kiện hoặc phân nhánh. Các so sánh điều kiện và phân nhánh nên được mô tả riêng trong các luồng thay thế.
- Luồng thay thế (alternative flows): Các luồng thay thế mô tả tất cả các kịch bản hoặc nhánh khác, bao gồm cả các nhánh thành công và thất bại. Luồng thay thế luôn phụ thuộc vào một điều kiện xảy ra trong một hoặc một số bước cụ thể trong luồng tham chiếu (reference flow) và luồng tham chiếu đó có thể là luồng chính hoặc một luồng thay thế khác. Điều kiện phân nhánh sẽ được chỉ định trong luồng tham chiếu. Luồng thay thế có thể được chia ra làm ba loại: Luồng thay thế cụ thể (specific alternative flow) là luồng thay thế được rẽ nhánh ra từ một bước cụ thể trong luồng tham chiếu. Luồng thay thế giới hạn (bounded alternative flow) là luồng có thể được rẽ nhanh từ nhiều hơn một bước trong luồng tham chiếu, có thể là các bước liên tiếp hoặc không liên tiếp. Luồng thay thế toàn cục là luồng thay thế có thể rẽ nhánh xảy ra trên bất kỳ bước nào trong luồng tham chiếu.

Use Case Name	Withdraw Fund	
Brief Description	ATM customer withdraws a specific amount of funds from a valid bank account.	
Precondition	The system is idle. The system is displaying a Welcome message.	
Primary Actor	ATM customer	
Secondary Actors	None	
Dependency	INCLUDE USE CASE Validate PIN.	
Generalization	None	
Basic Flow	**Steps**	
	1	INCLUDE USE CASE Validate PIN.
	2	ATM customer selects Withdrawal through the system
	3	ATM customer enters the withdrawal amount through the system.
	4	ATM customer selects the account number through the system.
	5	The system VALIDATES THAT the account number is valid.
	6	The system VALIDATES THAT ATM customer has enough funds in the account.
	7	The system VALIDATES THAT the withdrawal amount does not exceed the daily limit of the account.
	8	The system VALIDATES THAT the ATM has enough funds.
	9	The system dispenses the cash amount.
	10	The system prints a receipt showing transaction number, transaction type, amount withdrawn, and account balance.
	11	The system ejects the ATM card.
	12	The system displays Welcome message.
	Postcondition	ATM customer funds have been withdrawn.
Bounded Alternative Flows	**RFS Basic Flow 5-7**	
	1	The system displays an apology message MEANWHILE the system ejects the ATM card.
	2	The system shuts down.
	3	ABORT.
	Postcondition	ATM customer funds have not been withdrawn. The system is shut down.
Global Alternative Flows	IF ATM customer enters Cancel THEN	
	1	The system cancels the transaction MEANWHILE the system ejects the ATM card.
	2	ABORT.
	ENDIF	
	Postcondition	ATM customer funds have not been withdrawn. The system is idle. The system is displaying a Welcome message.
Specific Alternative Flows	**RFS Basic Flow 8**	
	1	The system displays an apology message MEANWHILE the system ejects the ATM card.
	2	ABORT.
	Postcondition	ATM customer funds have not been withdrawn. The system is idle. The system is displaying a Welcome message.

Bảng 2.1: Ví dụ về một khuôn mẫu mô tả ca sử dụng

Mỗi luồng hành vi, chính hoặc thứ cấp, bao gồm những thành phần sau:

- Bước hành động (action step): thể hiện một hành động được thực hiện bởi một tác nhân hoặc bởi hệ thống. Các bước hành động thường được đánh số để thể hiện trình tự diễn ra của các hành động, đồng thời cho biết các bước trước sẽ cần phải được hoàn thành trước khi bước sau được bắt đầu.

- Bước điều kiện (condition step): mô tả rằng hệ thống sẽ cần phải đánh giá một số điều kiện để từ đó quyết định bước hành động tiếp theo dựa trên kết quả vừa đánh giá.
- Bước xảy ra đồng thời (fork step): được sử dụng để mô tả hai hoặc nhiều hành động có thể xảy ra đồng thời.
- Hậu điều kiện (postcondition): chỉ những điều kiện cần phải đúng và đã được đáp ứng sau khi luồng hành vi và ca sử dụng kết thúc.

Bảng 2.1 thể hiện một ví dụ về các khái niệm của miền vấn đề ca sử dụng thông qua ca sử dụng Rút tiền (Withdraw Fund) thông qua một khuôn mẫu mô tả ca sử dụng. Các khái niệm về thuộc tính của ca sử dụng được thể hiện trong phần đầu tiên, bao gồm các dữ liệu về Tên ca sử dụng, Mô tả tóm tắt, Tiền điều kiện, Tác nhân chính, Tác nhân phụ, cùng các mối Quan hệ phụ thuộc và Quan hệ tổng quát. Luồng hành động chính và các luồng thứ cấp được mô tả thông qua từng cụm các bước, với các Bước hành động được đánh số theo thứ tự tăng dần, Bước điều kiện được thể hiện với các từ khóa “IF” và “ENDIF”, và Bước xảy ra đồng thời với từ khóa “MEANWHILE”. Cuối mỗi luồng hành vi là một Hậu điều kiện, được viết bằng các câu văn bản thể hiện yêu cầu cần được đáp ứng.

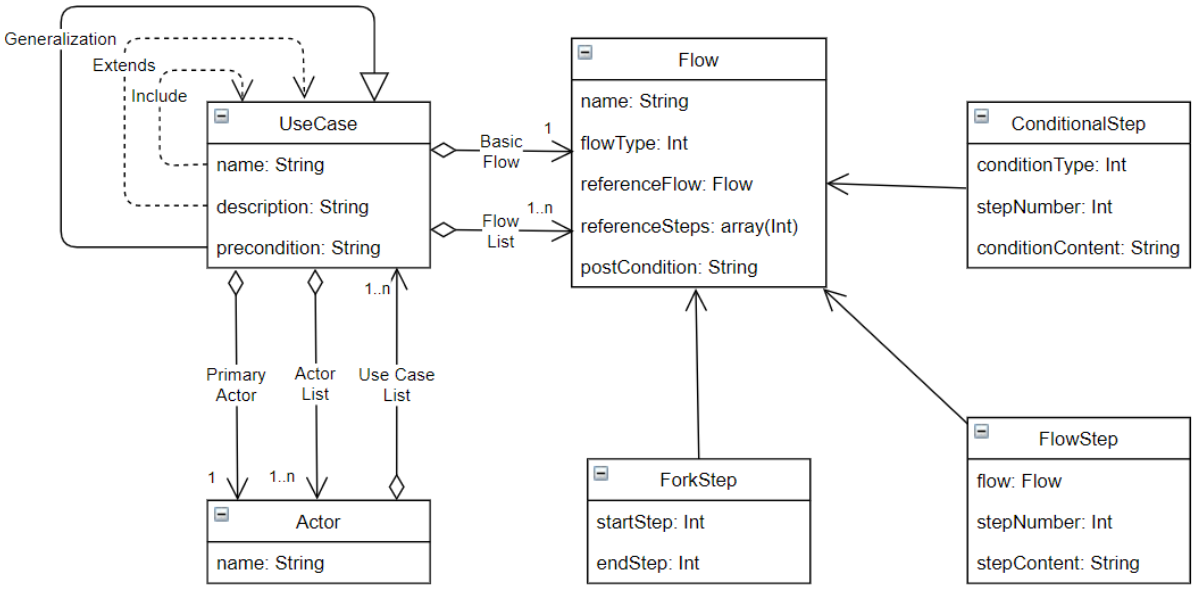
2.3. Cú pháp trừu tượng FRSL

Cú pháp trừu tượng của ngôn ngữ FRSL được xây dựng theo hướng tiếp cận metamodel, với các thành phần của metamodel được ánh xạ tương ứng với các khái niệm của miền vấn đề ca sử dụng. Hình 2.2 mô tả hệ thống metamodel của FRSL. Có thể chia metamodel này ra làm ba phần chính: Ca sử dụng (UseCase), Tác nhân (Actor) và Luồng thực thi (Flow). Trong đó:

- Ca sử dụng: là đối tượng bao quát của mô hình. Mỗi ca sử dụng sẽ có các thuộc tính tên, mô tả và tiền điều kiện. Các ca sử dụng có thể có những liên kết <<include>> và <<extend>> đối với các đối tượng ca sử dụng khác, đồng thời cũng có thể kế thừa các ca sử dụng khác theo phân cấp. Ca sử dụng sẽ có liên kết với một hoặc nhiều đối tượng tác nhân, trong đó sẽ phải có một và chỉ một tác nhân đóng vai trò là tác nhân chính của ca sử dụng. Một hoặc nhiều luồng thực thi cũng sẽ được liên kết với đối tượng ca sử dụng, trong đó luôn luôn có một luồng chính.

- Tác nhân: Chứa thông tin về tên của tác nhân và những liên kết giữa tác nhân với ca sử dụng. Mỗi tác nhân có thể liên kết tới một hoặc nhiều ca sử dụng khác nhau.

- Luồng thực thi: Chứa các thông tin về tên luồng, loại luồng thực thi (luồng chính, luồng thay thế cụ thể, luồng thay thế giới hạn, luồng thay thế toàn cục), các thông tin về luồng tham chiếu và bước phân nhánh của luồng tham chiếu nếu có, và hậu điều kiện cần thỏa mãn sau khi hoàn thành các bước trong luồng. Mỗi luồng thực thi sẽ chứa một tập hợp các hành động, được biểu diễn dưới dạng các bước. Các bước được chia nhỏ hơn thành ba loại: Bước thực thi (FlowStep) chỉ một hành động thông thường sẽ được người sử dụng thực hiện trên hệ thống, hoặc hệ thống tự thực hiện. Bước điều kiện (ConditionalStep) chỉ rằng hệ thống sẽ đánh giá những tham số ở thời điểm hiện tại, sau đó quyết định câu lệnh thực thi tiếp theo hoặc rẽ nhánh dựa trên kết quả thu được. Loại cuối cùng là bước phân tách (ForkStep) được sử dụng để mô tả những hành động xảy ra đồng thời.



Hình 2.2: Hệ thống metamodel của FRSL.

Một bộ luật giới hạn được áp dụng để tăng độ chính xác và tính đúng đắn của metamodel, gồm có:

- Tên của ca sử dụng và tác nhân phải gồm ít nhất một kí tự và độc nhất trong toàn bộ mô hình.

- Một ca sử dụng có một và chỉ một tác nhân chính.

- Một ca sử dụng chỉ có một luồng chính.
- Mọi luồng đều phải xác định loại luồng.
- Bước thực thi luôn phải có số bước.
- Bước điều kiện phải được xác định loại điều kiện.

2.4. Cú pháp cụ thể FRSL

Cú pháp cụ thể cho FRSL được xây dựng dựa trên khuôn mẫu và bộ luật giới hạn của RUCM, với những mở rộng phù hợp hơn cho khả năng diễn đạt đặc tả và chuyển hóa thành mô hình. Cú pháp cụ thể được cải tiến từ dạng bán văn bản của RUCM thành dạng văn bản như sau:

- Văn bản đặc tả sẽ chứa một danh sách các ca sử dụng, đơn vị ngăn cách của các câu là dấu xuống dòng.
- Phần đặc tả của một ca sử dụng bắt đầu bằng tên của ca sử dụng đó, với từ khóa "Use Case Name".
- Các thông tin về mô tả tóm tắt, tiền điều kiện, tác nhân chính, tác nhân thứ cấp, quan hệ phụ thuộc, quan hệ tổng quát được ghi trên từng dòng riêng biệt, với các từ khóa tương ứng "Brief Description", "Precondition", "Primary Actor", "Secondary Actor(s)", "Dependency", "Generalization".
- Một luồng hành vi sẽ bắt đầu bằng loại luồng ("Basic Flow", "Specific Alternative Flow", "Bounded Alternative Flow", "Global Alternative Flow") và kết thúc bằng hậu điều kiện với từ khóa "Postcondition". Một luồng có thể kéo dài trong nhiều dòng, với mỗi bước trong luồng ở một dòng riêng.
- Các bước điều kiện được bắt đầu với các từ khóa như "IF", "ELSE", "DO", "UNTIL" kết hợp với mệnh đề điều kiện nếu có.
- Các bước hành động được bắt đầu bằng một con số chỉ số bước, kèm theo nội dung của bước, trong đó có thể có các từ khóa thể hiện bước phân tách.

Hình 2.3 là biểu diễn chi tiết cú pháp cụ thể của FRSL.

```
grammar UsecaseReader;

usecaseList: (usecase|EOL)+ ;
usecase: ucName EOL (flow|(briefDesc|precondition|actor|dependency|generalization)EOL)+ ;
ucName: 'Use Case Name ' statement ;
briefDesc: 'Brief Description ' statement ;
precondition: 'Precondition ' statement ;
actor: actorType WS ('Actor'|'Actors') WS statement ;
actorType: 'Primary'|'Secondary' ;
dependency: 'Dependency ' specialKeyword? statement ;
generalization: 'Generalization ' specialKeyword? statement ;
flow: (flowType WS ('Flow'|'Flows') EOL) rfs? steps postCondition ;
flowType: ('Basic'|'Bounded Alternative'|'Global Alternative'|'Specific Alternative') ;
rfs: 'RFS ' flowType WS ('Flow'|'Flows') WS Number ('-'Number)? EOL ;
steps: (step|conditionalLogic)+ ;
step: Number WS sentence EOL ;
sentence: statement? specialKeyword? statement? ;
specialKeyword: ('INCLUDE USE CASE'|'EXTENDED BY USE CASE'|'MEANWHILE'|'VALIDATES THAT') ;
conditionalLogic: ('ELSE'|'ENDIF'|'DO'|'UNTIL'|('IF'|'ELSEIF') statement 'THEN') EOL ;
postCondition: 'Postcondition ' statement EOL ;
statement: (Word|Number|Punctuation|WS)+ ;
Word: [a-zA-Z]+ ;
Number: [0-9]+ ;
Punctuation: [,.\-] ;
EOL: '\r'? '\n' ;
WS : [ \t]+ ;
```

Hình 2.3: Cú pháp cụ thể dưới dạng văn bản của FRSL.

Hình 2.4 là ví dụ minh họa cú pháp cụ thể FRSL của một ca sử dụng Rút tiền (Withdraw Fund) của hệ thống máy ATM. Tên của ca sử dụng “Withdraw Fund” là thông tin đánh dấu sự mở đầu đặc tả ca sử dụng. Tiếp theo đó là các thông tin Brief Description, Precondition, Primary Actor, Secondary Actors, Dependency và Generalization được viết trên từng dòng. Phần còn lại của mô tả là các luồng hành vi của ca sử dụng, trong đó có một luồng chính “Basic Flow” và ba luồng thứ cấp. Luồng chính bao gồm 12 bước, mô tả một chu trình hoàn chỉnh của hành động Rút tiền. Các luồng phụ được sử dụng để xử lý lỗi và trường hợp khách hàng hủy giao dịch. Mỗi luồng được kết thúc bằng một Postcondition là hậu điều kiện mà hệ thống cần thỏa mãn mỗi khi thực hiện xong một luồng hành động bất kì.

```
Use Case Name Withdraw Fund
Brief Description ATM customer withdraws a specific amount of funds from a valid bank account
Precondition The system is idle. The system is displaying a Welcome message
Primary Actor ATM customer
Secondary Actors None
Dependency INCLUDE USE CASE Validate PIN
Generalization None
Basic Flows
1 INCLUDE USE CASE Validate PIN
2 ATM customer selects Withdrawal through the system
3 ATM customer enters the withdrawal amount through the system
4 ATM customer selects the account number through the system
5 The system VALIDATES THAT the account number is valid
6 The system VALIDATES THAT ATM customer has enough funds in the account
7 The system VALIDATES THAT the withdrawal amount does not exceed the daily limit of the account
8 The system VALIDATES THAT the ATM has enough funds
9 The system dispenses the cash amount
10 The system prints a receipt showing transaction number, transaction type, amount withdrawn, and account balance
11 The system ejects the ATM card
12 The system displays Welcome message
Postcondition ATM customer funds have been withdrawn
Bounded Alternative Flows
RFS Basic Flow 5-7
1 The system displays an apology message MEANWHILE the system ejects the ATM card
2 The system shuts down
3 ABORT
Postcondition ATM customer funds have not been withdrawn. The system is shut down
Global Alternative Flows
IF ATM customer enters Cancel THEN
1 The system cancels the transaction MEANWHILE the system ejects the ATM card
2 ABORT
ENDIF
Postcondition ATM customer funds have not been withdrawn. The system is idle. The system is displaying a Welcome message
Specific Alternative Flows
RFS Basic Flow 8
1 The system displays an apology message MEANWHILE the system ejects the ATM card
2 ABORT
Postcondition ATM customer funds have not been withdrawn. The system is idle. The system is displaying a Welcome message
```

Hình 2.4: Ca sử dụng Rút tiền dưới dạng văn bản cú pháp cụ thể FRSL.

2.5. Một số chuyển đổi từ đặc tả FRSL

Từ mô hình đặc tả FRSL, ta có thể áp dụng các phương pháp chuyển đổi mô hình để chuyển hóa dữ liệu, từ đó tự động sinh các loại chế tác tùy theo mục đích sử dụng. Một số nghiên cứu đã nhắm tới hướng tiếp cận này, ví dụ như [4] đã thực hiện chuyển đổi từ mô hình miền ca sử dụng sang mô hình kiểm thử ca sử dụng (Use Case Test Model - UMTG). Kết hợp với các ràng buộc OCL, mô hình kiểm thử ca sử dụng sinh tự động các đầu vào kiểm thử tương ứng với các kịch bản của ca sử dụng. Sau đó, họ thiết kế một bảng ánh xạ để ánh xạ các đầu vào kiểm thử với các chức năng điều khiển cụ thể. Các ca kiểm thử sẽ được sinh tự động thông qua bảng ánh xạ đó, và khi hệ thống thay đổi trong tương lai thì chỉ cần cập nhật lại bảng ánh xạ.

Không chỉ có ca kiểm thử, nhiều chế tác khác cũng có thể được sinh tự động như bản mẫu, mô hình thiết kế, các sơ đồ UML,... Luận văn đã xây dựng chức năng sinh tự động biểu đồ ca sử dụng từ mô hình FRSL để thử nghiệm khả năng chuyển đổi của mô hình. Sau khi được tổng hợp từ văn bản đặc tả theo cấu trúc metamodel, mô hình FRSL được chuyển đổi thành dạng văn bản đầu vào của công cụ PlantUML, sử dụng ánh xạ từ khái niệm miền vấn đề tương ứng. Các

thành phần của biểu đồ được sắp xếp dựa trên các mối quan hệ phụ thuộc giữa chúng.

Thuật toán thực thi chuyển đổi được viết bằng ngôn ngữ Java. Chương trình sử dụng vòng lặp để duyệt qua danh sách các tác nhân và các ca sử dụng trong mô hình FRSL, sắp xếp và xuất ra văn bản khai báo tương ứng với mỗi đối tượng. Sau khi đã tổng hợp được danh sách các thành phần của sơ đồ ca sử dụng, chương trình tiếp tục duyệt đến các mối quan hệ liên kết giữa chúng. Các liên kết tác nhân – ca sử dụng được lấy từ danh sách tác nhân trong dữ liệu của mỗi ca sử dụng, được biểu diễn bằng văn bản khai báo một mũi tên liền nét từ tác nhân đến ca sử dụng, sắp xếp sao cho các tác nhân sẽ hiển thị ở phía trên của sơ đồ, còn các ca sử dụng sẽ ở phần dưới. Tiếp theo đó là các liên kết giữa các ca sử dụng với nhau. Tùy theo loại liên kết, đối với include và extend, văn bản khai báo sẽ là mũi tên nét đứt, bắt đầu từ ca sử dụng gọi liên kết và trỏ tới ca sử dụng đích của liên kết. Ca sử dụng gốc sẽ nằm ở phía trên ca sử dụng đích. Đối với quan hệ generalization, chương trình sẽ thực hiện khai báo một liên kết mũi tên liền đầu tam giác, nối từ ca sử dụng con đến ca sử dụng cha. Ca sử dụng kế thừa sẽ nằm phía dưới ca sử dụng cha.

Sau khi những dữ liệu cần thiết đã được tổng hợp từ thông tin tương ứng trong mô hình FRSL theo đúng cấu trúc đầu vào của PlantUML, chương trình kích hoạt công cụ và truyền vào đoạn văn bản vừa tổng hợp được. PlantUML sẽ tiến hành vẽ ra sơ đồ ca sử dụng và xuất ra dưới dạng tệp ảnh. Chương trình sẽ đọc và hiển thị ảnh kết quả lên cửa sổ hiển thị chính. Thuật toán chuyển đổi không yêu cầu thêm thông tin gì của đặc tả ca sử dụng ngoài những dữ liệu mà mô hình FRSL đã cung cấp, từ đó cho thấy rằng mô hình FRSL lưu trữ và thể hiện được đầy đủ các dữ liệu cần thiết để sinh tự động biểu đồ ca sử dụng.

2.6. Các công việc liên quan

Ngôn ngữ FRSL được nghiên cứu và mở rộng từ ngôn ngữ RUCM . RUCM (Restricted Use Case Modeling – Mô hình hóa ca sử dụng hạn chế) là một cách tiếp cận mô hình hóa ca sử dụng [12] và mục tiêu của nó là hạn chế cách người dùng có thể ghi lại các bản đặc tả ca sử dụng để giảm sự mơ hồ, cải thiện tính dễ hiểu của các mô hình ca sử dụng và tạo điều kiện cho phân tích tự động để lấy ra các mô hình phân tích khởi nguồn, ví dụ như các đồ thị lớp, đồ thị tương tác, các ràng buộc,... RUCM bao gồm một bộ các luật giới hạn được định nghĩa chặt chẽ và một khuôn mẫu ca sử dụng dưới dạng bảng – văn bản. Các luật giới hạn và

khuôn mẫu ca sử dụng này nên được áp dụng từ giai đoạn xác định các yêu cầu phát triển phần mềm để có thể tạo ra những mô hình ca sử dụng chính xác và rõ ràng nhất mức có thể.

#	Description	Explanation
R1	The subject of a sentence in basic and alternative flows should be the system or an actor.	Enforce describing flows of events correctly. These rules conform to our use case template (the five interactions).
R2	Describe the flow of events sequentially.	
R3	Actor-to-actor interactions are not allowed.	
R4	Describe one action per sentence. (Avoid compound predicates.)	Otherwise it is hard to decide the sequence of multiple actions in a sentence.
R5	Use present tense only.	Enforce describing what the system does, rather than what it will do or what it has done.
R6	Use active voice rather than passive voice.	Enforce explicitly showing the subject and/or object(s) of a sentence.
R7	Clearly describe the interaction between the system and actors without omitting its sender and receiver.	

(a)

#	Description	Explanation
R8	Use declarative sentence only. "Is the system idle?" is a non-declarative sentence.	Commonly required for writing UCSs.
R9	Use words in a consistent way.	Keep one term to describe one thing.
R10	Don't use modal verbs (e.g., *might*)	Modal verbs and adverbs usually indicate uncertainty; therefore metrics should be used if possible.
R11	Avoid adverbs (e.g., *very*).	
R12	Use simple sentences only. A simple sentence must contain only one subject and one predicate.	Facilitate automated NL parsing and reduce ambiguity.
R13	Don't use negative adverb and adjective (e.g., *hardly*, *never*), but it is allowed to use *not* or *no*.	
R14	Don't use pronouns (e.g. *he*, *this*)	
R15	Don't use participle phrases as adverbial modifier. For example, the italic-font part of the sentence "ATM is idle, *displaying a Welcome message*", is a participle phrase.	
R16	Use "the system" to refer to the system under design consistently.	Keep one term to describe the system; therefore reduce ambiguity.

(b)

#	Description	#	Description
R17	INCLUDE USE CASE	R22	VALIDATE THAT
R18	EXTENDED BY USE CASE	R23	DO-UNTIL
R19	RFS	R24	ABORT
R20	IF-THEN-ELSE-ELSEIF-ENDIF	R25	RESUME STEP
R21	MEANWHILE	R26	Each basic flow and alternative flow should have their own postconditions.

(c)

Bảng 2.5 (a)(b)(c): Bộ luật giới hạn của RUCM.

Khuôn mẫu RUCM tích hợp các thành phần thông tin cơ bản của các ca sử dụng thường gặp trong các khuôn mẫu thông thường, đồng thời tìm cách xác định

rõ hơn cấu trúc của các luồng sự kiện của đặc tả ca sử dụng, giúp xây dựng một cấu trúc chặt chẽ, dễ viết và dễ đọc cho văn bản đặc tả ca sử dụng. Bộ luật giới hạn của RUCM gồm 26 luật, được phân thành hai nhóm: những luật hạn chế trong sử dụng ngôn ngữ tự nhiên và những luật bắt buộc sử dụng các từ khóa cụ thể để chỉ định cấu trúc điều khiển. Các luật về hạn chế ngôn ngữ tự nhiên chú trọng vào các bước hành động, yêu cầu người dùng phải sử dụng các câu đơn giản, thì hiện tại, sử dụng câu chủ định, tránh các từ ngữ không rõ ý, tránh nói lan man và tập trung vào các tương tác chính. Các luật về từ khóa đặc biệt yêu cầu người sử dụng phải hiểu rõ và sử dụng chính xác từ khóa trong từng trường hợp, giúp cho bản mô tả rõ nghĩa và không bị lẫn giữa các câu bình thường với các câu lệnh điều khiển.

Use Case Name	The name of the use case. It usually starts with a verb.	
Brief Description	Summarizes the use case in a short paragraph.	
Precondition	What should be true before the use case is executed.	
Primary Actor	The actor which initiates the use case.	
Secondary Actors	Other actors the system relies on to accomplish the services of the use case.	
Dependency	Include and extend relationships to other use cases.	
Generalization	Generalization relationships to other use cases.	
Basic Flow	Specifies the main successful path, also called "happy path".	
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the basic flow executes.
Specific Alternative Flows	Applies to one specific step of the basic flow.	
	RFS	A reference flow step number where flow branches from.
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the alternative flow executes.
Global Alternative Flows	Applies to all the steps of the basic flow.	
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the alternative flow executes.
Bounded Alternative Flows	Applies to more than one step of the basic flow, but not all of them.	
	RFS	A list of reference flow steps where flow branches from.
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the alternative flow executes.

Bảng 2.6: Khuôn mẫu của RUCM.

RUCM là một ngôn ngữ thể hiện dưới dạng bán văn bản và vì vậy nên thường phải sử dụng các phương pháp xử lý ngôn ngữ tự nhiên để lấy dữ liệu từ mô tả ca sử dụng. Ngoài ra, RUCM không phát triển theo hướng mô hình hóa, không chuyển hóa các diễn đạt đặc tả thành mô hình mà hầu hết đều thực hiện phân tích trực tiếp trên dữ liệu đầu vào. RUCM không có cú pháp trừu tượng riêng mà phụ thuộc vào cách xử lý của từng ứng dụng. Điều này có thể có lợi trong những trường hợp cố định đã được tính toán từ trước, nhưng không hỗ trợ khả năng mở rộng phạm vi ứng dụng của ngôn ngữ. FRSL đã giải quyết được các vấn đề này, với việc cung cấp một bộ cú pháp cụ thể trọn vẹn dưới dạng văn bản, tạo

thuận lợi cho quá trình phân tích và xây dựng thành mô hình đặc tả FRSL, đáp ứng tính năng dễ chuyển đổi để ứng dụng cho nhiều nhu cầu khác nhau.

Nghiên cứu [2] cũng đề xuất một ngôn ngữ đặc tả ca sử dụng USL. Đây là một ngôn ngữ đi sâu vào thể hiện những thông tin về mặt cấu trúc và hành vi của các ca sử dụng. USL tập trung vào các luồng hành vi và luồng sự kiện, với một hệ thống phân tích, phân loại và diễn giải rất chi tiết về các loại hành động và mối liên kết giữa chúng. Tuy nhiên, mức độ chuyên sâu cao khiến cho USL không xây dựng được cú pháp cụ thể dưới dạng văn bản, khiến cho ngôn ngữ khó có thể tiếp cận được với người dùng. So với USL, ngôn ngữ FRSL thể hiện đặc tả ở mức tổng quát và dễ sử dụng hơn. Bên cạnh đó, công cụ FRSL được xây dựng theo cấu trúc plugin, cung cấp khả năng mở rộng và phân tích sâu hơn ở mức độ mà người dùng mong muốn.

2.7. Tổng kết chương

Chương này đã trình bày về các khái niệm của miền vấn đề đặc tả ca sử dụng, từ đó làm cơ sở cho cú pháp trừu tượng và cú pháp cụ thể của ngôn ngữ đặc tả ca sử dụng FRSL. Cú pháp trừu tượng của FRSL được xây dựng theo hướng metamodel, với các thành phần được liên kết chặt chẽ theo miền vấn đề. Từ đó, luận văn đã xây dựng cú pháp cụ thể dưới dạng văn bản dựa trên một bộ luật chặt chẽ và một khuôn mẫu phát triển từ RUCM. Các mô hình được tổng hợp và xây dựng theo metamodel, sau đó có thể được sử dụng để làm đầu vào cho các phép chuyển đổi mô hình để thực hiện các mục đích khác.

CHƯƠNG 3. Cài đặt và Thực nghiệm

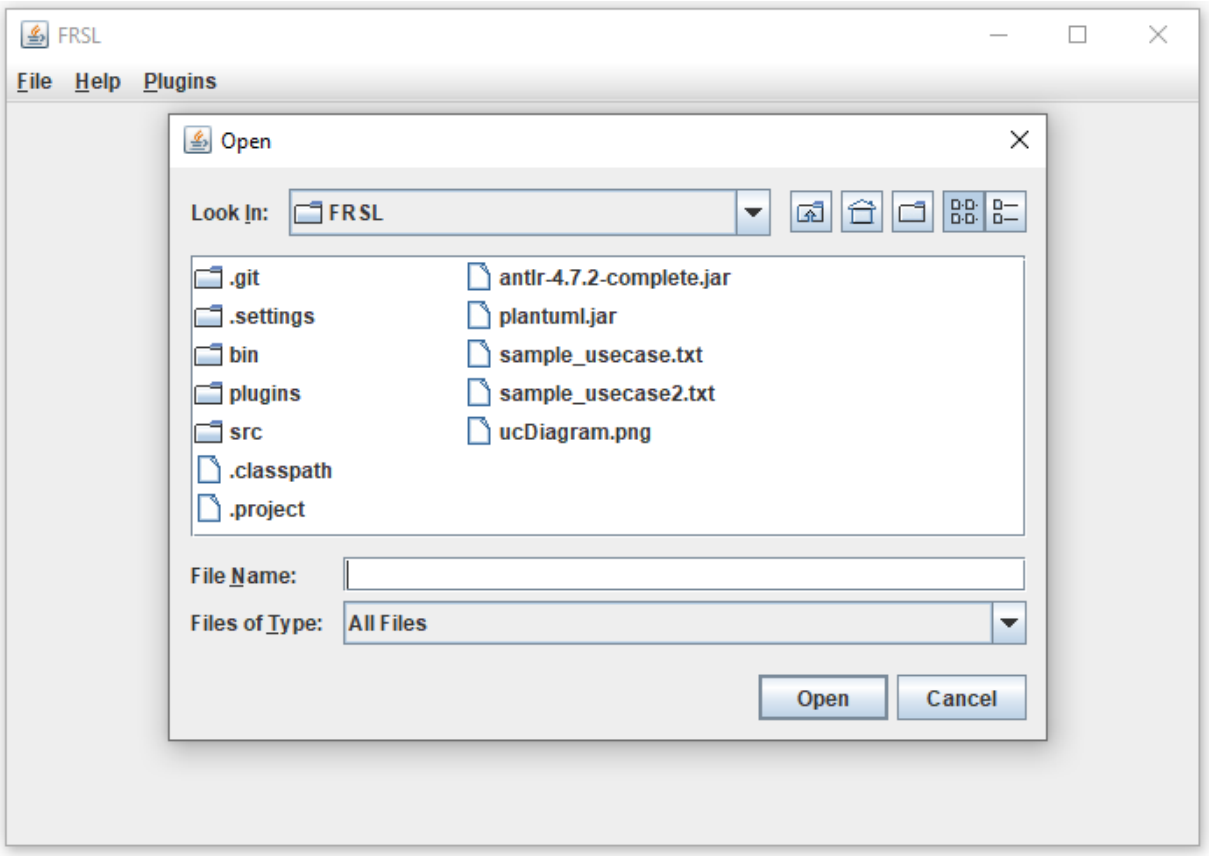
Trong chương này, luận văn sẽ mô tả bộ công cụ FRSL để hỗ trợ cho việc xây dựng và áp dụng các mô hình FRSL. Sau đó, bộ công cụ được đưa vào một bài toán vận dụng nhằm đánh giá các thông tin của mô hình FRSL, đồng thời kiểm chứng tính khả dụng của công cụ hỗ trợ. Từ đó, luận văn sẽ đưa ra một số nhận xét về hiệu quả cùng các ưu nhược điểm của ngôn ngữ.

3.1. Giới thiệu

Để có thể ứng dụng ngôn ngữ vào các vấn đề thực tế, đồng thời tăng khả năng tiếp cận của người dùng, luận văn đã phát triển bộ công cụ hỗ trợ FRSL. Phần đầu của chương sẽ trình bày về cấu trúc của chương trình, với mục tiêu cung cấp hệ thống giao diện giúp người dùng dễ dàng tương tác và sử dụng các đặc tả ca sử dụng. Bộ công cụ được xây dựng theo kiến trúc plugin, cho phép khả năng mở rộng phát triển trong tương lai. Trong phần tiếp theo, luận văn sẽ áp dụng ngôn ngữ đặc tả và bộ công cụ vào một bài toán thực tế về các ca sử dụng chính của hệ thống máy bán hàng tự động POS, từ đó đánh giá kết quả đạt được của thực nghiệm.

3.2. Công cụ hỗ trợ

Bộ công cụ hỗ trợ FRSL được xây dựng bằng ngôn ngữ Java, với phần đồ họa sử dụng thư viện Java Swing. Cú pháp trừu tượng của FRSL là hệ thống metamodel, được xây dựng trên các chế tác của ngôn ngữ Java. Cú pháp cụ thể của ngôn ngữ được xây dựng bằng bộ luật viết từ công cụ ANTLR. Chức năng sinh tự động sơ đồ ca sử dụng được hỗ trợ bởi công cụ PlantUML, với việc chuyển hóa mô hình từ mô hình FRSL sang dạng văn bản của PlantUML được thực hiện trong mã nguồn Java. Chương trình cung cấp hai chức năng chính. Chức năng đầu tiên là tải tệp đặc tả, cho phép người dùng tải lên một tệp văn bản có chứa những thông tin đặc tả các ca sử dụng, viết theo đúng cú pháp của ngôn ngữ FRSL. Tệp sẽ được đọc và đưa vào công cụ ANTLR để xử lý thành cây cú pháp, sau đó được duyệt với ANTLR visitor để xây dựng mô hình FRSL dựa trên metamodel đã được định nghĩa sẵn. Mô hình được sinh ra sẽ được lưu trong bộ nhớ và sẵn sàng để được sử dụng cho các mục đích khác.

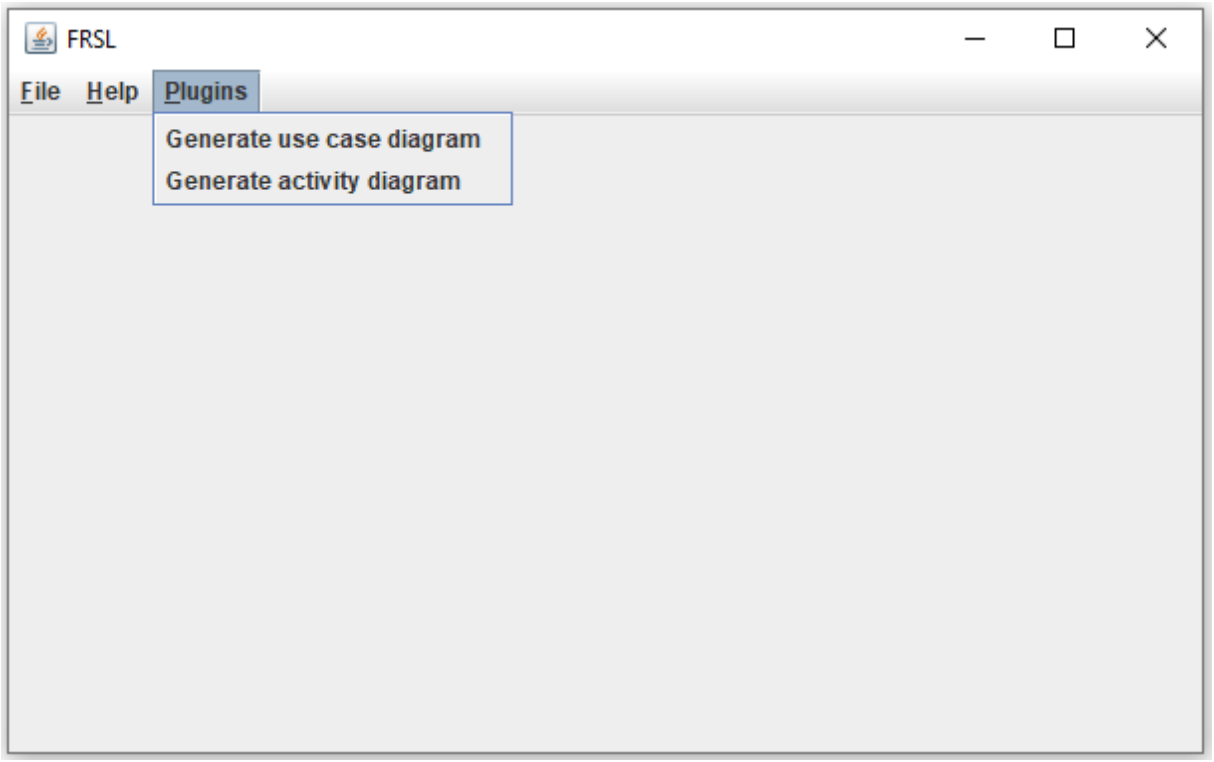


Hình 3.1: Giao diện của công cụ FRSL.

Cú pháp cụ thể từ ANTLR được viết trong một tệp .g4. Chương trình được cấu hình bằng công cụ Maven, với tích hợp ANTLR để dịch tệp đó thành các giao diện (interface) và lớp (class) của ngôn ngữ Java mỗi khi thực hiện build. Một lớp FRSLVisitor được tạo kế thưa từ Visitor mới được sinh ra, đóng vai trò là thành phần duyệt văn bản đặc tả và xử lý logic chính của quá trình chuyển hóa và xây dựng mô hình. Sau khi đã chiết suất được nội dung tệp đặc tả từ giao diện người dùng, FRSLVisitor sẽ đọc và phân tách văn bản thành cây cú pháp trừu tượng, sau đó duyệt cây và thực hiện các logic tính toán để nạp vào mô hình FRSL, kiểm tra điều kiện ràng buộc và xác định lỗi. FRSLVisitor thực hiện kiểm tra toàn cục một lần nữa sau khi đã duyệt hết cây, đảm bảo thỏa mãn các ràng buộc đã được định nghĩa trước khi trả về một mô hình hoàn chỉnh.

Chức năng thứ hai của chương trình là xử lý và áp dụng các plugin. Bộ công cụ FRSL được phát triển theo kiểu kiến trúc plugin, cung cấp khả năng mở rộng phạm vi ứng dụng của mô hình FRSL với nhiều nhu cầu khác nhau. Hệ thống đọc các tệp jar trong thư mục, sau đó plugin runtime sẽ xây dựng các điểm kích hoạt để gọi đến các đoạn mã trong từng plugin. Các điểm kích hoạt này được gắn vào menu chính của giao diện thông qua các menu phụ được tạo tương ứng. Từ đó, người sử dụng có thể quyết định sử dụng chức năng tùy theo mong muốn.

Chương trình cung cấp sẵn một plugin đầu tiên là plugin sinh tự động sơ đồ ca sử dụng.



Hình 3.2: Menu plugin của công cụ FRSL.

Các plugin được khởi tạo bằng một lớp MainPluginRuntime có nhiệm vụ đọc và ghi danh (register) các plugin từ các tệp jar. Để đọc các tệp jar, chương trình gọi đến lớp ParsePlugin để tìm và đọc tệp khai báo trong tệp jar, sau đó phân tích thành một đối tượng dữ liệu, trong đó có chứa thông tin về các kiện (package) và lớp chính của plugin, với các điểm kết nối được dùng để gắn lên menu chính của chương trình và chạy các thao tác của plugin khi cần thiết. Sau khi đã đọc ra thành đối tượng dữ liệu, MainPluginRuntime sẽ ghi danh đối tượng đó vào danh sách các plugin có thể sử dụng và chạy các tác vụ khởi tạo của plugin nếu có. Phần giao diện người dùng của chương trình sau đó sẽ đọc từ danh sách này và tạo các menu tương ứng để gắn kết các điểm kết nối của plugin (Hình 3.2).

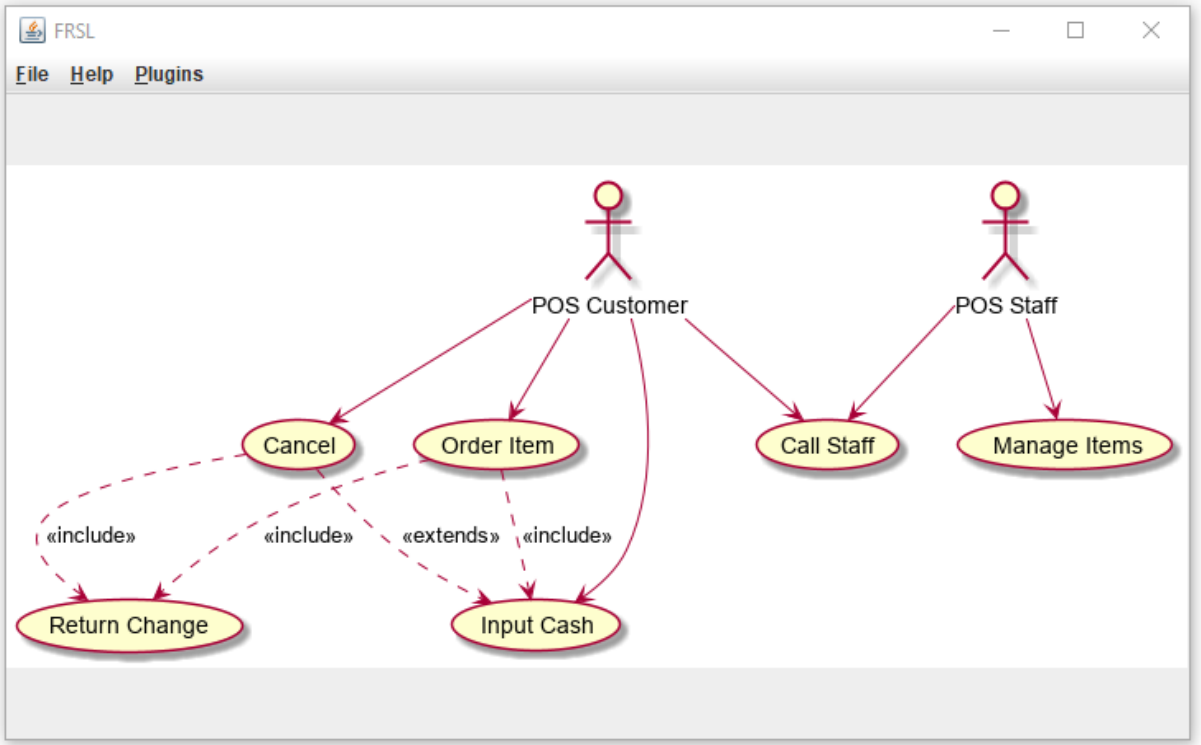
3.3. Bài toán vận dụng

Để kiểm chứng tính khả dụng của ngôn ngữ FRSL và bộ công cụ hỗ trợ, một bài toán đặc tả ca sử dụng của hệ thống máy bán hàng tự động (POS) đã được sử dụng làm đầu vào cho chương trình. Hệ thống bao gồm hai tác nhân, sáu ca sử dụng, và bốn mối quan hệ giữa các ca sử dụng. Hình 3.3 là phiên bản ngắn gọn của văn bản đặc tả của các ca sử dụng.

```
Use Case Name Input Cash
Brief Description POS Customer put bill into cash slot
Precondition The system is in idle state or select item state
Primary Actor POS Customer
Secondary Actors None
Dependency None
Generalization None
...

Use Case Name Order Item
Brief Description Select an item that it available
Precondition The system is in select item state
Primary Actor POS Customer
Dependency INCLUDE USE CASE Input Cash
Dependency INCLUDE USE CASE Return Change
...

Use Case Name Cancel
Brief Description ATM customer input PIN to gain access to the corresponding bank account
Precondition The system is in select item state
Primary Actor POS Customer
Dependency EXTENDED BY USE CASE Input Cash
Dependency INCLUDE USE CASE Return Change
...

Use Case Name Return Change
Brief Description The system return the change to POS Customer
Precondition The system is in delivered state
...

Use Case Name Call Staff
Brief Description POS Customer press the help button to call for a staff
Precondition The system is in any state
Primary Actor POS Customer
Secondary Actors POS Staff
...

Use Case Name Manage Items
Brief Description POS Staff can open the POS without trigger warnings
Precondition The system is in idle state
Primary Actor POS Staff
...
```

Hình 3.3: Văn bản đặc tả ca sử dụng rút gọn.

Hình 3.4 là sơ đồ ca sử dụng được sinh bởi chức năng sinh tự động mà chương trình cung cấp từ những thông tin trong văn bản đặc tả ca sử dụng. Kết quả này giúp đánh giá được rằng văn bản đặc tả đã được đọc và phân tích thành mô hình FRSL một cách đầy đủ và chính xác. Quá trình ứng dụng và chuyển đổi mô hình không gặp vấn đề gì, các thông tin và quan hệ giữa các ca sử dụng được bảo toàn. Ngoài ra, bài toán cũng giúp chứng minh khả năng hoạt động của kiến trúc plugin thông qua quá trình xây dựng tự động sơ đồ ca sử dụng. Sản phẩm của plugin cũng đã được tích hợp và hiển thị trên giao diện chính của công cụ.



Hình 3.4: Sơ đồ ca sử dụng được sinh từ văn bản đặc tả.

3.4. Đánh giá

Các ca sử dụng đã được thể hiện rõ ràng và dễ hiểu hơn thông qua những xử lý của ngôn ngữ đặc tả ca sử dụng FRSL, chuyển hóa từ dạng văn bản thành các dạng khác trong đó có sơ đồ ca sử dụng với mức độ diễn đạt cao. Điều này cho thấy hướng phát triển đặc tả ca sử dụng theo ngôn ngữ chuyên biệt miền là đúng đắn và có khả năng áp dụng cao. Ngôn ngữ FRSL cung cấp đầy đủ các thành phần cú pháp trừu tượng và cú pháp cụ thể, giúp đáp ứng hầu hết các trường hợp thể hiện ca sử dụng cơ bản trong yêu cầu phần mềm. Cú pháp cụ thể dưới dạng văn bản làm tăng tính dễ dùng và tránh được các lỗi không đáng có, giúp người sử dụng dễ dàng thao tác mà không cần nhiều thời gian làm quen.

Luận văn đã xây dựng thành công ngôn ngữ đặc tả ca sử dụng FRSL với kết quả tốt khi áp dụng vào bài toán thực tế. Mô hình FRSL được xây dựng chi tiết từ văn bản đặc tả, với khả năng áp dụng cho nhiều mục tiêu khác nhau, kết hợp với khả năng mở rộng của kiến trúc plugin đã đem lại một phạm vi diễn đạt rộng, đặc tả cụ thể các yêu cầu chức năng của bài toán. Tuy nhiên, ngôn ngữ vẫn còn một số điểm hạn chế cần phải khắc phục. Các thành phần đặc tả đang được xử lý một cách rời rạc, khiến cho việc phân tích chuyên sâu vào các thành phần khó đạt được độ chi tiết nếu không áp dụng các kĩ thuật xử lý ngôn ngữ tự nhiên

cao cấp. Ngoài ra, bộ công cụ hỗ trợ chưa cung cấp được nhiều chức năng để có thể áp dụng phổ biến và sẽ cần nhiều đóng góp từ cộng đồng để mở rộng dần dần.

3.5. Tổng kết chương

Ngôn ngữ FRSL đã được tích hợp để xây dựng thành một bộ công cụ FRSL hỗ trợ cho quá trình đọc và xử lý văn bản đặc tả đầu vào thành các mô hình, đồng thời áp dụng mô hình đó vào các chuyển đổi để đáp ứng nhiều nhu cầu khác nhau. Bộ công cụ được xây dựng theo kiểu kiến trúc plugin, giúp tăng khả năng mở rộng của các chức năng chương trình bằng các đóng góp của cộng đồng lập trình viên. Thông qua bộ công cụ này, ngôn ngữ FRSL đã được vận dụng vào thực nghiệm với một bài toán thực tế, từ đó đánh giá được khả năng diễn đạt khá tốt và khả năng áp dụng đa dạng của ngôn ngữ. Tuy nhiên, ngôn ngữ FRSL vẫn cần phải khắc phục thêm một số thành phần còn hạn chế để có thể đưa vào sử dụng phổ biến.

KẾT LUẬN

Phân tích và đặc tả yêu cầu là một trong những quy trình quan trọng nhất của phát triển phần mềm, đóng vai trò chủ chốt, ảnh hưởng đến toàn bộ các bước phía sau của quá trình xây dựng phần mềm. Tuy nhiên, việc phân tích yêu cầu thường gặp rất nhiều khó khăn, kết hợp với sự đa dạng và dễ thay đổi của yêu cầu phần mềm khiến cho đặc tả yêu cầu một cách có hiệu quả trở thành những thách thức. Nhiều nghiên cứu đã được phát triển để hỗ trợ người dùng đặc tả yêu cầu, trong đó các nghiên cứu tập trung vào UML và ca sử dụng đã đem đến những hiệu quả rất tốt về khả năng diễn đạt đặc tả. Thế nhưng vẫn còn có nhiều hạn chế trong việc tiếp cận và phân tích các ca sử dụng, vì vậy nên luận văn đã thực hiện nghiên cứu và đề xuất một ngôn ngữ đặc tả ca sử dụng phát triển theo hướng chuyên biệt miền nhằm mục đích hỗ trợ người dùng mô tả và áp dụng các thành phần của ca sử dụng.

Luận văn đã đạt được các kết quả chính như sau:

- Giới thiệu tổng quan về đặc tả yêu cầu, yêu cầu chức năng, ca sử dụng, các khái niệm và định hướng của mô hình hóa chuyên biệt miền và ngôn ngữ mô hình hóa chuyên biệt miền cùng các công cụ hỗ trợ xây dựng ngôn ngữ.
- Phân tích và diễn giải các thành phần của miền vấn đề ca sử dụng, từ đó xây dựng hệ thống metamodel đóng vai trò là cú pháp trừu tượng của ngôn ngữ. Xây dựng cú pháp cụ thể dưới dạng văn bản và đóng gói thành ngôn ngữ đặc tả ca sử dụng FRSL. Từ mô hình FRSL tổng hợp được, nghiên cứu chuyển đổi thành các dạng khác nhau cho từng mục đích sử dụng, trong đó tập trung vào sinh tự động sơ đồ ca sử dụng.
- Vận dụng ngôn ngữ và xây dựng bộ công cụ hỗ trợ FRSL. Bộ công cụ được xây dựng theo kiến trúc plugin, tăng khả năng mở rộng cho các lĩnh vực khác nhau và nhận đóng góp từ cộng đồng lập trình viên.
- Áp dụng ngôn ngữ vào bài toán thực tế về đặc tả ca sử dụng của hệ thống máy bán hàng tự động POS. Đưa ra đánh giá về khả năng diễn tả và áp dụng của mô hình FRSL, cùng những ưu nhược điểm của cách tiếp cận này.

Với những kết quả trên, luận văn xin được đề xuất một số hướng phát triển tiếp theo:

- Phát triển cú pháp cụ thể cho ngôn ngữ FRSL dưới dạng hình học, tăng khả năng diễn đạt đặc tả của người dùng và tính dễ dàng tiếp cận của ngôn ngữ.
- Xây dựng giao diện của công cụ hỗ trợ dưới dạng môi trường phát triển (IDE), giúp lập trình viên có thể tạo, sửa, xóa trực tiếp thông tin đặc tả và nhận kết quả một cách nhanh chóng.
- Xây dựng các plugin cho công cụ để chuyển hóa mô hình FRSL sang các nhu cầu khác như sinh tự động ca kiểm thử, mã nguồn, bản mẫu,...

TÀI LIỆU THAM KHẢO

[1] Arie van Deursen and Paul Klint (2002), "Domain-Specific Language Design Requires Feature Descriptions", *Journal of Computing and Information Technology – CIT*, pp. 1–17.

[2] Chu Thi Minh Hue, Dang Duc Hanh, Nguyen Ngoc Binh and Le Minh Duc (2018), "USL: A Domain-Specific Language for Precise Specification of Use Cases and Its Transformations", *Informatica*, volume 42, pp. 325–343.

[3] Chu Thi Minh Hue, Dang Duc Hanh, Nguyen Ngoc Binh and Truong Anh Hoang (2019), "USLTG: Test Case Automatic Generation by Transforming Use Cases", *Int. Journal of Software Engineering and Knowledge Engineering*, volumne 29, pp. 1313–1345.

[4] Chunhui Wang, Fabrizio Pastore, Arda Goknil, Lionel Briand, Zohaib Iqbal (2015), "Automatic Generation of System Test Cases from Use Case Specifications", *ISSTA 2015*, pp. 385–396.

[5] Hans J. Köhler, Ulrich Nickel, Jörg Niere, Albert Zündorf (2000), "Integrating UML Diagrams for Production Control Systems", *Proceedings - International Conference on Software Engineering*, pp. 241–251.

[6] IEEE (2011), *Systems and software engineering – Life cycle processes – Requirements engineering*, ISO/IEC/IEEE.

[7] Le Duc Minh, Dang Duc Hanh and Nguyen Viet Ha, "On Domain Driven Design Using Annotation-Based Domain Specific Language", *Computer Languages, Systems & Structures*, volumn 54, pp. 199–235.

[8] Marco Brambilla, Jordi Cabot, Manuel Wimmer (2017), *Model-Driven Software Engineering in Practice – Second Edition*, Morgan & Claypool Publishers.

[9] Richard C. Gronback (2009), *ECLIPSE MODELING PROJECT A Domain-Specific Language Toolkit*, Addison-Wesley.

[10] Ruth Malan and Dana Bredemeyer (2001), *Functional Requirements and Use Cases*, Bredemeyer Consulting.

[11] Steven Kelly, Juha-Pekka Tolvanen, *DOMAIN-SPECIFIC MODELING Enabling Full Code Generation*, Wiley-Interscience Publication, JOHN WILEY & SONS, INC.

[12] Tao Yue (2010), *Restricted Use Case Modeling Approach*, Simula Research Laboratory.