

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



**TRẦN THU TRANG**

**SINH TỰ ĐỘNG CHẾ TÁC PHẦN MỀM  
TỪ MÔ HÌNH QUY TRÌNH NGHIỆP VỤ DỰA TRÊN  
CHUYỂN ĐỔI MÔ HÌNH**

**LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN**

**HÀ NỘI - 2022**

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**TRẦN THU TRANG**

**SINH TỰ ĐỘNG CHẾ TÁC PHẦN MỀM  
TỪ MÔ HÌNH QUY TRÌNH NGHIỆP VỤ DỰA TRÊN  
CHUYỂN ĐỔI MÔ HÌNH**

Ngành: Công nghệ thông tin  
Chuyên ngành: Kỹ thuật phần mềm  
Mã số: 8480103.01

**LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN**

**CÁN BỘ HƯỚNG DẪN: TS. ĐẶNG ĐỨC HẠNH**

**Hà Nội – 2022**

## TÓM TẮT

**Tóm tắt:** Phần mềm là một phần quan trọng trong các giải pháp dựa trên phần mềm (software-based solutions), trong đó yêu cầu phần mềm là một phần có ảnh hưởng rất lớn tới sự thành công của các dự án phần mềm. Yêu cầu phần mềm, bao gồm cả yêu cầu chức năng và phi chức năng, thường được xác định dựa trên mô hình quy trình nghiệp vụ. Thực tế cho thấy nhiều tổ chức, doanh nghiệp đã có sẵn các quy trình nghiệp vụ tương đối đầy đủ và có nhiều tương đồng với các yêu cầu phần mềm nên có thể dễ dàng cung cấp nền tảng cho việc chuyển đổi mô hình. Tuy nhiên cho đến nay, việc xác định và chuyển đổi này thường được thực hiện một cách thủ công, phát sinh nhiều hạn chế gây tốn kém nguồn lực và có thể làm chậm tiến độ phát triển phần mềm. Từ đó nảy sinh nhu cầu cần phải phát triển một giải pháp tự động hóa để thiết kế các chế tác phần mềm tốt hơn với hiệu quả cao. Luận văn này sẽ tập trung nghiên cứu và đề xuất một phương pháp tự động sử dụng mô hình hóa quy trình nghiệp vụ để tạo ra mô hình ca sử dụng phù hợp. Nguyên tắc dựa trên việc so sánh và đánh giá sự tương quan giữa các siêu mô hình để xây dựng bộ quy tắc ánh xạ, qua đó có thể cài đặt chương trình chuyển đổi. Kết quả có thể giúp tiết kiệm chi phí, có thể duy trì lùn vết giữa phần mềm - ở không gian giải pháp và nghiệp vụ - ở không gian vấn đề và đảm bảo các mô hình sản phẩm phản ánh chính xác hệ thống.

**Từ khóa:** *chuyển đổi mô hình, Model2Model, BPMN, mô hình ca sử dụng, metamodel*

## LỜI CẢM ƠN

Sau quá trình học tập và nghiên cứu, được sự hỗ trợ từ Khoa Công nghệ thông tin, Trường Đại học Công Nghệ - ĐHQGHN và sự tận tình chỉ bảo của thầy giáo hướng dẫn – TS. Đặng Đức Hạnh, tôi đã hoàn thành đề tài luận văn thạc sĩ “*Sinh tự động chế tác phần mềm từ mô hình quy trình nghiệp vụ dựa trên chuyển đổi mô hình*”.

Đầu tiên, tôi xin gửi lời cảm ơn sâu sắc đến giảng viên hướng dẫn, Tiến sĩ Đặng Đức Hạnh - đã tận tâm định hướng, chỉ dạy tôi và gợi ý giải quyết các khó khăn để tôi có thể hoàn thành luận văn này.

Tôi xin gửi lời cảm ơn chân thành đến các thầy cô giảng viên của Trường Đại học Công Nghệ – ĐHQGHN đã tận tình dạy dỗ và truyền đạt kiến thức giúp tôi có thêm nhiều hiểu biết và nhận thức về chuyên ngành cũng như định hướng phát triển sự nghiệp tương lai. Tôi cũng xin cảm ơn các thành viên trong nhóm nghiên cứu đã hỗ trợ và giải đáp rất nhiều thắc mắc của tôi trong khoảng thời gian vừa qua.

Cuối cùng, tôi muốn cảm ơn gia đình, bạn bè và người thân đã đồng hành cùng tôi trong cuộc sống, luôn tạo điều kiện và động viên giúp đỡ tôi vượt qua mọi khó khăn.

Tôi xin chân thành cảm ơn.

Hà Nội, tháng 08 năm 2022

Học viên

Trần Thu Trang

## **LỜI CAM ĐOAN**

Tôi là Trần Thu Trang, học viên cao học khóa K25 - CNPM của Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội. Tôi xin cam đoan đây là công trình nghiên cứu của tôi dưới sự đầu tư nghiên cứu nghiêm túc và tận tình giúp đỡ của Giảng viên hướng dẫn là Tiến sĩ Đặng Đức Hạnh và các bạn trong nhóm nghiên cứu. Những nội dung nghiên cứu và kết quả trong đề tài này là hoàn toàn trung thực. 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.

Hà Nội, tháng 08 năm 2022

Học viên thực hiện

Trần Thu Trang

## MỤC LỤC

<b>DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT .....</b>	<b>1</b>
<b>DANH MỤC BẢNG BIỂU .....</b>	<b>2</b>
<b>DANH MỤC HÌNH VẼ.....</b>	<b>3</b>
<b>CHƯƠNG 1. TỔNG QUAN .....</b>	<b>5</b>
1.1. Đặt vấn đề .....	5
1.2. Mục tiêu và phương pháp .....	8
1.3. Bố cục của luận văn .....	8
<b>CHƯƠNG 2. KIẾN THỨC NỀN TẢNG.....</b>	<b>9</b>
2.1. Kỹ nghệ hướng mô hình .....	9
2.1.1. Mô hình hóa (Modeling) .....	9
2.1.2. Chuyển đổi mô hình .....	11
2.2. Mô hình quy trình nghiệp vụ .....	14
2.2.1. Mô hình hóa quy trình nghiệp vụ.....	15
2.2.2. Tiêu chuẩn ký hiệu và mô hình hóa quy trình nghiệp vụ BPMN .....	16
2.3. Mô hình ca sử dụng .....	18
2.3.1. Ngôn ngữ mô hình thống nhất .....	18
2.3.2. Biểu đồ ca sử dụng.....	19
<b>CHƯƠNG 3. PHƯƠNG PHÁP SINH TỰ ĐỘNG CA SỬ DỤNG TỪ MÔ HÌNH QUY TRÌNH NGHIỆP VỤ .....</b>	<b>21</b>
3.1. Giới thiệu .....	21
3.2. Tổng quan về phương pháp BPMN2UseCase .....	22
3.3. Biểu diễn mô hình BPMN .....	24
3.4. Biểu diễn mô hình ca sử dụng .....	25
3.5. Xác định các luật chuyển đổi mô hình.....	27
3.5.1. Luật chuyển đổi cho các đối tượng (Rules for Elements).....	27
3.5.2. Luật chuyển đổi cho các quan hệ giữa các đối tượng (Rules for Association) .....	28
3.5.3. Luật chuyển đổi cho các quan hệ với dữ liệu (Rules for Data association).32	32
3.6. Thuật toán và thực thi chuyển đổi mô hình .....	35
3.7. Tổng kết chương .....	37
<b>CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNH GIÁ.....</b>	<b>38</b>
4.1. Công cụ, ngôn ngữ và môi trường hỗ trợ .....	38
4.1.1. Kỹ thuật biểu diễn quy trình nghiệp vụ.....	38
4.1.2. Kỹ thuật biểu diễn biểu đồ ca sử dụng.....	39

4.2. Nghiên cứu tình huống thực nghiệm .....	39
4.2.1. School Library System.....	39
4.2.2. Phương pháp đánh giá kết quả .....	43
4.2.3. Tổng kết thực nghiệm và đánh giá .....	47
4.3. Tổng kết chương .....	48
<b>CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>49</b>
5.1. Kết quả đạt được .....	49
5.2. Hướng phát triển .....	50
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>51</b>
<b>PHỤ LỤC .....</b>	<b>54</b>

## **DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT**

AMW	ATLAS Model Weaving
ATL	ATLAS Transformation Language
BPM	Business Process Management
BPMN	Business Process Model and Notation
CIM	Computer Independent Model
EMF	Eclipse Modeling Framework
M2M	Model-to-Model
M2T	Model-to-Text
MDE	Model-Driven Engineering
MOF	Meta Object Facility
OMG	Object Management Group
PIM	Platform Independent Model
QVT	Query/View/Transformation
RE	Requirement Engineering
UC	Use Case

**DANH MỤC BẢNG BIỂU**

Bảng 2.1. BPMN Flow Objects [23] .....	17
Bảng 2.2. BPMN Connecting Objects [23] .....	17
Bảng 2.3. BPMN Swimlanes [23] .....	18
Bảng 2.4. BPMN Artifacts [23].....	18
Bảng 2.5. Các khái niệm trong ca sử dụng.....	19
Bảng 4.1. Kết quả đánh giá tính đúng đắn .....	44
Bảng 4.2. Kết quả đánh giá tính đầy đủ .....	46

## DANH MỤC HÌNH VẼ

Hình 1.1. Ánh xạ các khái niệm UC từ quy trình nghiệp vụ [14]. .....	7
Hình 2.1. Mẫu mô hình hóa trong MDE [8].....	9
Hình 2.2. Chuyển đổi mô hình [25].....	11
Hình 2.3. Cơ chế chuyển đổi mô hình [24]. .....	13
Hình 2.4. Minh họa bài toán chuyển đổi “Families to Persons”.....	14
Hình 2.5. Mô hình chuyển đổi ATL “Families to Persons”.....	14
Hình 3.1. Mô hình quy trình nghiệp vụ Nobel Prize [9]. .....	21
Hình 3.2. Mô hình ca sử dụng Nobel Prize [9]. .....	22
Hình 3.3. Quy trình chuyển đổi mô hình đề xuất.....	23
Hình 3.4. Cơ chế chuyển đổi mô hình BPMN2UseCase. .....	23
Hình 3.5. Định dạng đầu vào *.bpmn.....	24
Hình 3.6. Định dạng đầu ra *.uml. ....	24
Hình 3.7. Siêu mô hình lược giản BPMN [8].....	25
Hình 3.8. Mô hình quy trình nghiệp vụ xử lý đơn hàng online.....	25
Hình 3.9. Siêu mô hình ca sử dụng đơn giản [13].....	26
Hình 3.10. Mô hình hoá ca sử dụng trong một nhà hàng.....	26
Hình 3.11. Luật RE1 - Ánh xạ Task sang UseCase. .....	27
Hình 3.12. Hàm chuyển đổi RE1 - Task2UseCase.....	27
Hình 3.13. Luật RE2 - Ánh xạ Participant sang Actor. .....	28
Hình 3.14. Hàm chuyển đổi RE2 - Participant2Actor. ....	28
Hình 3.15. Luật RA1 - Ánh xạ giữa Participant và Task sang Association. .....	28
Hình 3.16. Hàm chuyển đổi RA1 - Task2UseCaseAssociation. ....	29
Hình 3.17. Luật RA2 - Ánh xạ SequenceFlow sang Dependency. ....	29
Hình 3.18. Hàm chuyển đổi RA2 - SequenceFlow2Dependency. ....	29
Hình 3.19. Luật RA3 - Ánh xạ ExclusiveGateway sang quan hệ Extend.....	30
Hình 3.20. Hàm chuyển đổi RA3 - Gateway2Extend. ....	30
Hình 3.21. Luật RA4 - Ánh xạ MessageFlow sang Association.....	31
Hình 3.22. Hàm chuyển đổi RA4 - MessageFlow2Association.....	31
Hình 3.23. Luật RD1 - Ánh xạ đọc Data Store sang quan hệ include ReadData.....	32

Hình 3.24. Hàm chuyển đổi RD1 - DataStore2ReadData.....	32
Hình 3.25. Luật RD2 - Ánh xạ ghi Data Store sang quan hệ include WriteData.....	33
Hình 3.26. Hàm chuyển đổi RD2 – DataStore2WriteData. ....	33
Hình 3.27. Luật RD3 - Ánh xạ nhận Data Object sang quan hệ include ReceiveData.	33
Hình 3.28. Hàm chuyển đổi RD3 - DataObject2ReceiveData. ....	34
Hình 3.29. Luật RD4 - Ánh xạ gửi Data Object sang quan hệ include SendData. ....	34
Hình 3.30. Hàm chuyển đổi RD4 – DataObject2SendData. ....	34
Hình 3.31. Mã giả mô tả thuật toán sinh mô hình ca sử dụng.....	36
Hình 4.1. Giao diện công cụ Eclipse BPMN2 Modeler. ....	38
Hình 4.2. Giao diện công cụ Eclipse Papyrus. ....	39
Hình 4.3. Mô hình BPMN Purchase book [10].....	40
Hình 4.4. Mô hình BPMN Lend book [10]. ....	40
Hình 4.5. Mô hình BPMN Return book [10]. ....	40
Hình 4.6. Mô hình ca sử dụng Purchase book.....	41
Hình 4.7. Mô hình ca sử dụng Lend book.....	42
Hình 4.8. Mô hình ca sử dụng Return book. ....	42
Hình 4.9. Mô hình ca sử dụng School Library System [10]. ....	43

## CHƯƠNG 1. TỔNG QUAN

Trong chương này, luận văn sẽ giới thiệu và trình bày một cách tổng quát về đề tài nghiên cứu, bao gồm thực trạng phát triển hiện nay, các vấn đề thách thức gặp phải và các hướng tiếp cận liên quan. Từ đó, luận văn xác định được mục tiêu tìm hiểu và phương pháp nghiên cứu để xây dựng và phát triển giải pháp cho đề tài.

### 1.1. Đặt vấn đề

Trong phát triển phần mềm hiện nay, kỹ thuật yêu cầu (Requirement Engineering - RE) là một pha cực kỳ trọng yếu, là bước khởi đầu và có thể ảnh hưởng tới toàn bộ hoạt động phát triển phần mềm nếu không được thực hiện một cách đúng đắn. RE là chìa khóa cho sự thành công của các dự án phần mềm [19]. Thực tế, trong các hệ thống lớn và phức tạp, rất khó để phát triển được chính xác các yêu cầu phần mềm [19]. Một số nghiên cứu chỉ ra rằng: Các lỗi phát sinh trong pha yêu cầu thường là nghiêm trọng và khó khăn nhất để khắc phục. 70% các lỗi hệ thống là do đặc tả chưa đầy đủ và 30% là do các vấn đề thiết kế [1, 18]. Có nhiều nguyên nhân dẫn đến các vấn đề trên, có thể được phân loại thành các nhóm chính sau [18]:

- Vấn đề về phạm vi: Ranh giới của hệ thống không rõ ràng, dẫn đến việc các thông tin không quan trọng có thể được đưa vào thiết kế trong khi các thông tin cần thiết có thể bị thiếu trong đặc tả yêu cầu.
- Vấn đề về nhận thức và hiểu rõ hệ thống: phát sinh khi người phân tích/phát triển không xác định đúng nhu cầu của người yêu cầu hoặc khi bản thân người yêu cầu và các bên liên quan không thể hiện rõ được nhu cầu của chính họ. Bên cạnh đó, sự thiếu trao đổi hoặc hiểu sai trong giao tiếp giữa bên yêu cầu và bên phân tích/phát triển cũng góp phần làm sinh ra các yêu cầu thiếu hoàn chỉnh, chính xác hoặc còn mơ hồ, không nhất quán.
- Vấn đề về đáp ứng sự thay đổi: Theo thời gian, các yêu cầu cần luôn được cập nhật và thay đổi dựa trên sự thay đổi về nhu cầu và mục đích của bên yêu cầu, hay do sự phát triển không ngừng của các công nghệ phần mềm. Do đó, sự chậm trễ trong việc thích ứng hay sự giải quyết không thỏa đáng các xung đột phát sinh có thể làm nảy sinh nhiều vấn đề.

Nhiều nghiên cứu được đề xuất để nâng cao hiệu quả của pha yêu cầu và khắc phục các lỗi trên để có thể tạo ra được những chế tác yêu cầu phần mềm chính xác, hoàn thiện và hiệu quả. Trong đó, hướng tiếp cận mô hình quy trình nghiệp vụ là một giải pháp phù hợp bởi ít nhất một nửa sự phát triển phần mềm công nghiệp được kết nối với sự phát triển ứng dụng kinh doanh [5] và việc hiểu rõ quy trình hoạt động là chìa khóa để xác định, phân tích các yêu cầu hệ thống cần thiết cho quá trình phát triển. Thực tế cũng cho thấy nhiều tổ chức, doanh nghiệp đã có sẵn các mô hình quy trình nghiệp vụ tương đối đầy đủ và chi tiết để có thể cung cấp nền tảng cho việc mô hình hóa yêu cầu phần mềm. Ví dụ, một nghiên cứu tại một ngân hàng quốc tế lớn chỉ ra rằng 350 trên tổng số 1000 quy trình kinh doanh dùng trong ngân hàng đã được mô

tả rõ ràng và sử dụng để tham khảo trong hoạt động kinh doanh hàng ngày [14]. Bên cạnh đó, mô hình quy trình nghiệp vụ có thể thể hiện hoạt động của tổ chức một cách rõ ràng dễ hiểu qua các biểu đồ đồ họa, góp phần làm giảm các lỗi phát sinh do “vấn đề về nhận thức và hiểu rõ hệ thống”. Mô hình quy trình nghiệp vụ cũng giải quyết được “vấn đề về phạm vi” khi xây dựng đặc tả yêu cầu bởi việc thể hiện cách thức mà tổ chức thực hiện và sắp xếp công việc cũng như việc quy định rõ ràng cấu trúc và mục đích của hệ thống.

Theo các nghiên cứu [7, 13, 14], một dạng chế tác phần mềm có sự tương đồng rất lớn với mô hình quy trình nghiệp vụ đó là ca sử dụng. Điều này khá rõ ràng khi xem xét đến hai khái niệm của ca sử dụng và quy trình kinh doanh: Một ca sử dụng UML chỉ định một chuỗi các hành động, bao gồm các biến thể mà hệ thống có thể thực hiện, mang lại kết quả giá trị có thể quan sát được cho một tác nhân cụ thể [14]. Một quy trình kinh doanh là một tập hợp các hoạt động có cấu trúc và được cân nhắc, được thiết kế để tạo ra đầu ra cụ thể cho một khách hàng hoặc một thị trường cụ thể [12]. Dựa trên các định nghĩa này, cả quy trình nghiệp vụ và ca sử dụng đều tập trung vào các hoạt động (hành động), đều được sắp xếp theo một cách nào đó và đều nhằm mục đích tạo ra một kết quả cụ thể cho một đối tượng nhất định (tác nhân hoặc khách hàng). Không chỉ tương đồng rõ rệt về mặt khái niệm, các quy trình nghiệp vụ còn có thể được mô tả bởi chính mô hình ca sử dụng [15]. Từ đó có thể nhận định việc chuyển đổi từ quy trình kinh doanh sang ca sử dụng là hoàn toàn có thể và hợp lý.Thêm vào đó, việc chuyển đổi này có thể mang lại lợi ích cụ thể, bởi thông thường, để xây dựng ca sử dụng, các tổ chức phải thực hiện khá nhiều cuộc phỏng vấn và phân tích đánh giá trong khi quy trình kinh doanh lại thường có sẵn dưới dạng các hướng dẫn công việc hoặc sổ tay quản trị [14]. Sử dụng chính những nguồn có sẵn này để sinh ra ca sử dụng có thể tiết kiệm được nhiều thời gian, công sức và đảm bảo phản ánh đúng nhu cầu nghiệp vụ của hệ thống.

Hiện nay, nhiều nghiên cứu đã đưa ra các phương pháp để giải quyết vấn đề trong bài toán này, có thể chia thành 3 hướng tiếp cận chính như sau: thứ nhất là phương pháp thủ công chuyển đổi giữa BPM và UC, thứ hai là chuyển đổi tự động mô hình UC từ BPM bằng cách xây dựng thuật toán chuyển đổi và cuối cùng là chuyển đổi tự động mô hình UC từ BPM bằng cách áp dụng MDE.

- **Chuyển đổi thủ công giữa mô hình BPM và UC**

Ở cách tiếp cận đầu tiên, Cruz và các đồng tác giả đã xây dựng một tập các luật để tạo ra mô hình ca sử dụng bao gồm cả biểu đồ và đặc tả từ mô hình quy trình nghiệp vụ [9]. Nghiên cứu tập trung ưu tiên vào quá trình sinh ra đặc tả giàu thông tin sử dụng nhiều luật tương quan hơn. Các đặc tả thể hiện qua ngôn ngữ tự nhiên (Natural Language – NL) bởi một tập hợp các cấu trúc câu đã được quy định trước. Trong khi đó, nội dung của biểu đồ ca sử dụng được tạo ra bởi 5 luật chuyển đổi, áp dụng tương ứng cho các đối tượng Participant, Lane, Activity, quan hệ giữa Participant và Pool, quan hệ giữa Participant và Activity

từ mô hình BPMN. Do đó, biểu đồ tạo ra tương đối đơn giản, chỉ bao gồm những yếu tố thành phần cơ bản là Actor, Use Case và quan hệ giữa Actor và Use Case, không bao gồm các quan hệ giữa các Use Case như quan hệ mở rộng, bao gồm hay quan hệ tổng quát.

- **Chuyển đổi tự động mô hình UC từ BPM bằng cách xây dựng thuật toán chuyển đổi**

Nghiên cứu [13, 14] mô tả một thuật toán để chuyển đổi mô hình quy trình nghiệp vụ thành đặc tả yêu cầu chức năng, cụ thể là biểu đồ ca sử dụng. Thuật toán hoạt động dựa trên việc tạo ra các siêu mô hình của quy trình nghiệp vụ và yêu cầu chức năng ca sử dụng, từ đó so sánh các khái niệm, ý nghĩa của các khái niệm và đưa ra được các ánh xạ tương quan giữa các mô hình. Chi tiết việc xây dựng khái niệm tương quan này được trình bày cụ thể trong [14] (Hình 1.1).

Business Process Concept	Use Case Concept
Role	Actor
Step	Use Case
Association between Role and Step	Association between Actor and Use Case
Task	Interaction
Task in a Step	Interaction in a Use Case
Transition between Tasks in the same Step	Ordering between Interactions in the same Use Case
Guard on Transition	Constraint on Interaction
Alternative Path through a Branch	Alternative Path Description of a Use Case, or Extending Use Case

Hình 1.1. Ánh xạ các khái niệm UC từ quy trình nghiệp vụ [14].

Trong quá trình thực nghiệm, phương pháp đã đánh giá 6 quy trình nghiệp vụ và tạo ra được 42 ca sử dụng. Trong đó 17 ca sử dụng có cấu trúc chưa chính xác và vẫn còn sai sót, một tỉ lệ lỗi tương đối cao [13].

- **Chuyển đổi tự động mô hình UC từ BPM bằng cách áp dụng MDE**

Trong [21], một giải pháp tự động hóa được đưa ra để chuyển đổi từ mức mô hình độc lập tính toán (CIM) sang mức mô hình độc lập với nền tảng (PIM). Cách tiếp cận sử dụng biểu đồ cộng tác và biểu đồ quy trình BPMN đại diện cho mô hình nghiệp vụ tiêu chuẩn để thể hiện ở mức CIM. Từ đó thu được biểu đồ ca sử dụng quy định chức năng của hệ thống và cuối cùng là biểu đồ lớp để mô hình các lớp và các mối quan hệ giữa chúng. Để thực hiện việc chuyển đổi, tác giả sử dụng công cụ dựa trên ngôn ngữ QVT, theo kiến trúc hướng mô hình (MDA). Một nghiên cứu khác cũng theo

hướng MDE là [6], đề xuất một công cụ BPMN2UC sử dụng ngôn ngữ ATL (ATLAS Transformation Language) để sinh ra biểu đồ UC từ các khái niệm BPMN. Đồng thời phương pháp cũng cho phép tạo ra đặc tả ca sử dụng cho các ca sử dụng phức tạp bằng cách sử dụng Acceleo.

Có thể thấy các phương pháp tiếp cận như trên đều mang lại những hiệu quả nhất định, tuy nhiên vẫn còn tồn tại một số nhược điểm cụ thể. Phần lớn các giải pháp đều được thực hiện một cách thủ công, độ chính xác chưa cao và phát sinh tốn kém nhân lực và chi phí. Cùng với đó, khả năng bao trùm các loại phần tử hay khái niệm còn yếu kém, dẫn đến việc đầu ra còn chưa đầy đủ và nghèo thông tin. Việc thực hiện chuyển đổi còn phức tạp hay sự dư thừa trong quá trình xử lý cũng là những hạn chế cần được khắc phục.

## 1.2. Mục tiêu và phương pháp

Mục đích cơ bản của luận văn là xây dựng một phương pháp sinh tự động mô hình ca sử dụng từ mô hình quy trình nghiệp vụ, nhằm nâng cao hiệu quả của pha yêu cầu phần mềm và gia tăng tự động hóa trong phát triển phần mềm. Giải pháp cần đơn giản, có thể áp dụng dễ dàng và phản ánh tương đối chi tiết và chính xác các hoạt động nghiệp vụ hệ thống.

Để đạt được mục tiêu này, luận văn đã đề xuất phương thức tiếp cận như sau:

- Nghiên cứu siêu mô hình của quy trình nghiệp vụ và ca sử dụng, từ đó so sánh, phân tích và đánh giá để tìm ra sự tương quan giữa các yếu tố trong mô hình. Từ đó, xây dựng một bộ quy tắc chuyển đổi mô hình.
- Phát triển một chương trình chuyển đổi mô hình – BPMN2UseCase, cài đặt dựa trên bộ luật nền trên và tự động sinh ca sử dụng từ mô hình nghiệp vụ đầu vào.
- Thủ nghiệm và đánh giá chương trình sau khi áp dụng vào tình huống cụ thể và bộ test-cases. Chương trình được đánh giá dựa trên các tiêu chí: tính ổn định, tính đầy đủ và tính đúng đắn.

## 1.3. Bộ cục của luận văn

**Chương 1:** Giới thiệu, trình bày vấn đề nghiên cứu, mục tiêu và phương pháp đề xuất.

**Chương 2:** Kiến thức nền tảng, trình bày cơ sở lý thuyết về kỹ nghệ hướng mô hình, mô hình quy trình nghiệp vụ và mô hình ca sử dụng.

**Chương 3:** Tìm hiểu về bài toán chuyển đổi mô hình quy trình nghiệp vụ sang ca sử dụng và các công trình nghiên cứu liên quan. Từ đó đề xuất phương pháp BPMN2UseCase sinh tự động mô hình ca sử dụng từ mô hình quy trình nghiệp vụ.

**Chương 4:** Cách cài đặt chương trình, ứng dụng thực nghiệm giải pháp vào các tình huống cụ thể. Xây dựng bộ test-cases để kiểm tra đánh giá phương pháp.

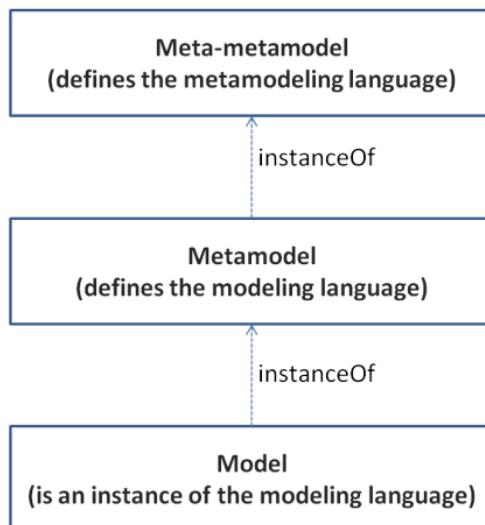
**Chương 5:** Kết quả đạt được và hướng phát triển trong tương lai.

## CHƯƠNG 2. KIẾN THỨC NỀN TẢNG

Trong chương này, luận văn sẽ trình bày chi tiết các kiến thức cần thiết để giúp hiểu rõ được phương pháp tiếp cận đề tài, cụ thể bao gồm: kỹ nghệ hướng mô hình và các khái niệm quan trọng, mô hình quy trình nghiệp vụ và tiêu chuẩn ký hiệu BPMN cùng với mô hình ca sử dụng và biểu đồ ca sử dụng UML. Đây là những nền tảng cơ bản cần phải nắm được để có thể ứng dụng xây dựng giải pháp cho bài toán chuyển đổi mô hình.

### 2.1. Kỹ nghệ hướng mô hình

Mô hình là một khái niệm rất quan trọng trong việc nghiên cứu phát triển các phần mềm phức tạp. Kỹ nghệ hướng mô hình (Model-Driven Engineering - MDE) là một hướng tiếp cận đầy hứa hẹn sử dụng mô hình để hỗ trợ các giai đoạn khác nhau trong vòng đời phát triển phần mềm. MDE dựa trên các nguyên tắc liên quan đến các khái niệm mô hình, siêu mô hình, siêu-siêu mô hình và chuyển đổi mô hình để cung cấp một quy trình cho phép phát triển tự động một hệ thống [8]. Trong ngữ cảnh của MDE, một mô hình là một thể hiện trừu tượng của một hệ thống được xác định bởi ngôn ngữ mô hình hóa, trong đó ngôn ngữ mô hình hóa là phương tiện thể hiện hệ thống một cách đơn giản và chính xác bằng cách sử dụng các biểu đồ, ký hiệu, số, chữ cái, ... Ba khái niệm mô hình, siêu mô hình và siêu-siêu mô hình tạo thành một mẫu mô hình ba cấp [8] (Hình 2.1). Mô hình ở cấp càng cao thì mức độ trừu tượng càng cao.



Hình 2.1. Mẫu mô hình hóa trong MDE [8].

#### 2.1.1. Mô hình hóa (Modeling)

Mô hình (Model) là một thể hiện dạng trừu tượng hóa một đối tượng hay một hệ thống thực, cụ thể là một hình ảnh hay một biểu diễn phản ánh hệ thống thực, qua đó diễn tả hệ thống:

- Ở một mức trừu tượng hóa nhất định

- Theo một quan điểm hay một góc nhìn nào đó
- Bởi một hình thức diễn tả có thể hiểu được (đồ thị, bảng, văn bản,...)

Mô hình được dùng để nghiên cứu và tìm hiểu hoạt động của hệ thống, mô hình hóa là việc thay thế các đối tượng thực của hệ thống bằng mô hình hay thiết lập và sử dụng mô hình, từ đó thu được các thông tin quan trọng cần thiết của hệ thống bằng cách tiến hành thực nghiệm trên mô hình. Các mục đích chính của quá trình mô hình hóa đó là:

- *Để hiểu:* Hiểu là hình thành được hình ảnh xác thực và giản lược về hệ thống cần được tìm hiểu. Nhờ vào việc sử dụng mô hình, các vấn đề được nhận thức dễ dàng và nhanh chóng hơn.
- *Để trao đổi:* Mô hình có thể coi như một loại ngôn ngữ để trao đổi giữa những người quan tâm đến hệ thống.
- *Để hoàn chỉnh:* Mô hình với sự minh bạch, rõ ràng có thể giúp người dùng đánh giá hệ thống có đầy đủ, chặt chẽ hay phù hợp với yêu cầu chưa, từ đó có thể hoàn thiện hơn.

### ***Ngôn ngữ mô hình hóa (Modeling language)***

Ngôn ngữ mô hình hóa là ngôn ngữ nhân tạo được sử dụng để diễn tả thông tin hoặc hệ thống trong một cấu trúc xác định bởi một bộ quy tắc nhất quán, trong đó các quy tắc dùng để giải thích ý nghĩa của các thành phần trong cấu trúc. Dựa trên yêu cầu và miền sử dụng, các ngôn ngữ mô hình hóa có thể chia thành bốn loại: mô hình hóa hệ thống, mô hình hóa đối tượng, mô hình hóa dữ liệu và mô hình hóa thực tế ảo. Xét về ngữ cảnh, một ngôn ngữ mô hình hóa có thể là đồ họa hoặc văn bản.

- *Ngôn ngữ mô hình hóa đồ họa:* sử dụng kỹ thuật biểu diễn sơ đồ với các ký hiệu được đặt tên để đại diện cho các khái niệm, các đường kết nối đại diện cho các quan hệ và nhiều dạng ký hiệu đồ họa khác để biểu diễn các ràng buộc. Một số ví dụ về dạng đồ họa này là Business Process Modeling Notation (BPMN) - mô hình hóa quy trình, EXPRESS - mô hình hóa dữ liệu, Unified Modeling Language (UML) - mô hình hóa cho các hệ thống phần mềm chuyên sâu, hỗ trợ 13 kỹ thuật biểu đồ khác nhau...
- *Ngôn ngữ mô hình hóa văn bản:* sử dụng các từ khóa được chuẩn hóa cùng với các tham số hoặc thuật ngữ để tạo ra các biểu thức máy tính có thể hiểu được.

### ***Lưu trữ mô hình***

Để lưu trữ mô hình, công nghệ phổ biến được sử dụng là tiêu chuẩn eXtensible Markup Language Metadata Interchange (XMI). Theo [3], XMI – giao thức trao đổi dữ liệu đặc tả dựa trên ngôn ngữ đánh dấu mở rộng là một tiêu chuẩn được phát triển và duy trì bởi Tổ chức Quản lý Đối tượng (OMG). Mục tiêu của XMI là cho phép trao đổi dữ liệu đặc tả dễ dàng giữa các công cụ mô hình hóa dựa trên UML và lưu trữ dữ liệu đặc tả dựa trên MOF (Meta Object Facility - MOF là công nghệ nền tảng để mô tả các mô hình đặc tả) trong các môi trường không đồng nhất. XMI được sử dụng

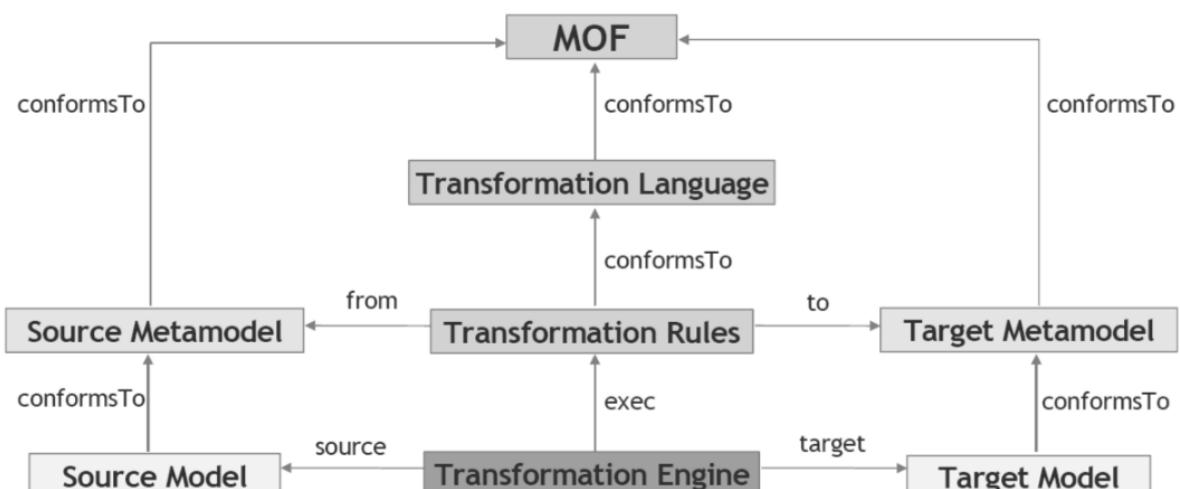
rong rãi trong định dạng trao đổi XML, XMI định nghĩa các vấn đề sau trong việc mô tả các đối tượng trong XML:

- Biểu diễn các đối tượng dưới dạng các phần tử và các thuộc tính XML.
- Các cơ chế chuẩn cho các đối tượng trong cùng một tập tin hay trên nhiều tập tin.
- Xác định đối tượng, cho phép các đối tượng được tham chiếu từ các đối tượng khác dưới dạng các định danh (identifier - ID) hoặc các định danh toàn thể duy nhất (universally unique identifier – UUID).

XMI cung cấp một phương thức chuẩn cho người lập trình trao đổi thông tin về dữ liệu đặc tả. Cụ thể XMI hướng đến giúp người lập trình sử dụng UML với các ngôn ngữ và các công cụ phát triển khác để trao đổi các mô hình dữ liệu với nhau. Đặc biệt, XMI chuẩn hóa cách thức tập dữ liệu đặc tả được mô tả và yêu cầu người dùng trên nhiều lĩnh vực và nhiều môi trường hoạt động thống nhất cách mô tả dữ liệu.

### 2.1.2. Chuyển đổi mô hình

Trong kỹ nghệ hướng mô hình MDE, chuyển đổi mô hình là quá trình biến đổi một mô hình sang một mô hình khác của cùng một hệ thống [26], nói cách khác là một cách tự động tạo ra mô hình đích từ một mô hình nguồn dựa trên một định nghĩa chuyển đổi. Trong đó, một định nghĩa chuyển đổi là một tập hợp các quy tắc chuyển đổi để mô tả cách một hay nhiều cấu trúc của mô hình nguồn có thể được chuyển thành một hay nhiều cấu trúc của mô hình đích như thế nào [25]. Hình 2.2 thể hiện các khái niệm cơ bản của chương trình chuyển đổi mô hình, với đầu vào là một mô hình nguồn tạo nên siêu mô hình nguồn và đầu ra là một mô hình khác tạo nên một siêu mô hình đích. Chương trình chuyển đổi bao gồm một bộ các quy tắc, bản thân nó có thể được coi là một mô hình dựa trên một siêu mô hình và sử dụng một ngôn ngữ chuyển đổi để thực hiện.



Hình 2.2. Chuyển đổi mô hình [25].

## Các hướng chuyển đổi mô hình

Một cách tổng quát, cách tiếp cận chuyển đổi mô hình có thể được chia thành hai hướng là mô hình-thành-mô hình - M2M (model-to-model) và mô hình-thành-văn bản - M2T (model-to-text). Điểm khác biệt cơ bản đó là M2M tạo ra một mô hình đích với cú pháp trùu tượng, còn kết quả sinh ra của M2T được thể hiện bằng cú pháp cụ thể, có dạng chuỗi ký tự, văn bản, mã nguồn,... Theo [27], chuyển đổi M2T được phân thành hai loại: “visitor-based” và “template-based”. Trong khi đó, M2M gồm năm loại chính [25]:

- Thao tác trực tiếp (direct-manipulation): cung cấp một đại diện mô hình nội bộ và một số APIs để thao tác với nó. Nó cũng thường được thực hiện như một khung hướng đối tượng và người dùng phải xây dựng các quy tắc chuyển đổi, lên kế hoạch, truy vết,... bằng một ngôn ngữ lập trình.
- Quan hệ (relational): được xem như một hình thức giải quyết các ràng buộc. Ý tưởng cơ bản là xác định các mối quan hệ giữa các phần tử nguồn và đích bằng cách sử dụng các ràng buộc không thể thực thi.
- Dựa trên chuyển đổi biểu đồ (graph-transformation-based): mô hình nguồn và mô hình đích đều được thể hiện dưới dạng đồ thị và thực hiện chuyển đổi mô hình bằng phép biến đổi đồ thị - lấy đồ thị cú pháp trùu tượng của mô hình và biến đổi nó theo quy tắc nhất định.
- Tiếp cận hoạt động (operational): tương tự như thao tác trực tiếp nhưng cung cấp nhiều hỗ trợ sâu hơn cho chuyển đổi mô hình. Một giải pháp điển hình là mở rộng hình thức siêu mô hình hóa với những điều kiện để thể hiện tính toán.
- Kết hợp (hybrid): kết hợp các kỹ thuật khác nhau nêu trên.

Để hỗ trợ việc tiếp cận các hướng nghiên cứu trên, nhiều kỹ thuật chuyển đổi mô hình đã được phát triển và đưa vào sử dụng, ví dụ M2T có Eclipse Acceleo, M2M có hai kỹ thuật nổi bật là ATL (ATLAS Transformation Language) và QVT (Query/View/Transformation).

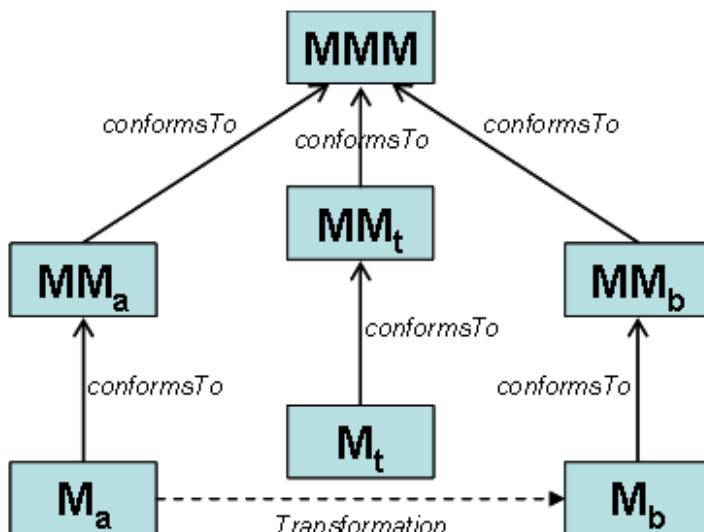
- ATL là một bộ công cụ và ngôn ngữ chuyển đổi mô hình. Vì ATL cung cấp cả hai cấu trúc khai báo và mệnh lệnh, nó được coi là một chuyển đổi mô hình kết hợp (hybrid). Kiến trúc bao gồm ba lớp: ATLAS Model Weaving (AMW), ATL và máy ảo ATL (Virtual Machine). AMW có thể được sử dụng như một ngôn ngữ đặc tả chuyển đổi mức trùu tượng cao hơn. Các chương trình chuyển đổi được viết chủ yếu với ATL, và ATL VM sẽ biên dịch các chương trình ATL đó. ATL cung cấp các phương thức để tạo ra một tập hợp các mô hình đích từ một tập hợp các mô hình nguồn.
- QVT là một bộ ngôn ngữ tiêu chuẩn để thực hiện chuyển đổi mô hình được xác định bởi OMG, có khả năng thực hiện các truy vấn và các phép biến đổi mô hình. QVT tích hợp tiêu chuẩn ngôn ngữ ràng buộc đối tượng (Object Constraint Language – OCL). Đây cũng là một loại chuyển đổi kết hợp (hybrid) với cả hai cấu trúc khai báo và mệnh lệnh.

Luận văn được định hướng xây dựng theo hướng M2M và sử dụng kỹ thuật ATL nên tập trung tìm hiểu sâu hơn về công cụ này.

### *Công cụ chuyển đổi ATL*

Trong MDE, ATL là một ngôn ngữ và công cụ hiệu quả, cho phép tạo ra các mô hình đích từ một (hoặc một tập hợp) các mô hình nguồn. Được phát triển dựa trên nền tảng Eclipse, môi trường tích hợp ATL (ATL Integrated Environment) cung cấp các công cụ tiêu chuẩn (debug, hỗ trợ cú pháp, ...) giúp cho việc phát triển thuận tiện hơn.

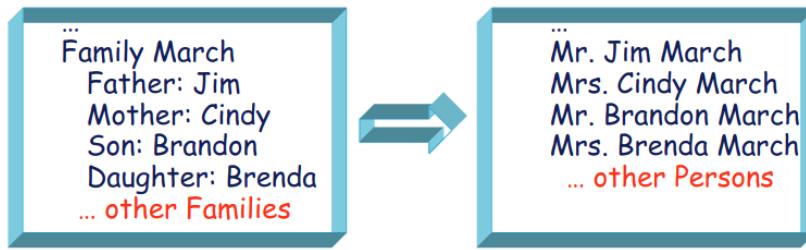
Về mặt hình thức, một phép chuyển đổi mô hình đơn giản phải xác định cách tạo ra một mô hình Mb, phù hợp với siêu mô hình MMb từ mô hình Ma, phù hợp với siêu mô hình MMA. Bản thân phép biến đổi này cũng được định nghĩa là một mô hình, và phù hợp với một siêu mô hình chuyển đổi xác định ngữ nghĩa của việc chuyển đổi. Phép biến đổi được xác định bởi mô hình chuyển đổi Mt, phù hợp với siêu mô hình MMt. Siêu mô hình MMt này, cùng với MMA và MMb phải tuân theo siêu-siêu mô hình MMM [24]. Lược đồ Hình 2.3 đã tóm tắt toàn bộ quá trình chuyển đổi mô hình từ mô hình Ma sang mô hình Mb.



Hình 2.3. Cơ chế chuyển đổi mô hình [24].

Hiện nay, có một số công nghệ siêu-siêu mô hình được đưa ra và áp dụng thực tế. Trong đó hai công nghệ được hỗ trợ sử dụng trong ATL đó là: MOF (Meta Object Facilities) định nghĩa bởi OMG và Ecore định nghĩa bởi EMF (Eclipse Modeling Framework). Như vậy, trong ATL, metamodel MMM ở hình được thể hiện bằng cách sử dụng ngữ nghĩa của MOF hoặc Ecore [24].

Để có thể nắm được cơ chế hoạt động cơ bản của ATL, luận văn sẽ trình bày một ví dụ đơn giản “Families to Persons”, với đầu vào là một danh sách các gia đình và yêu cầu đầu ra là một danh sách cá nhân với đầy đủ tên họ và danh xưng phù hợp (Hình 2.4).



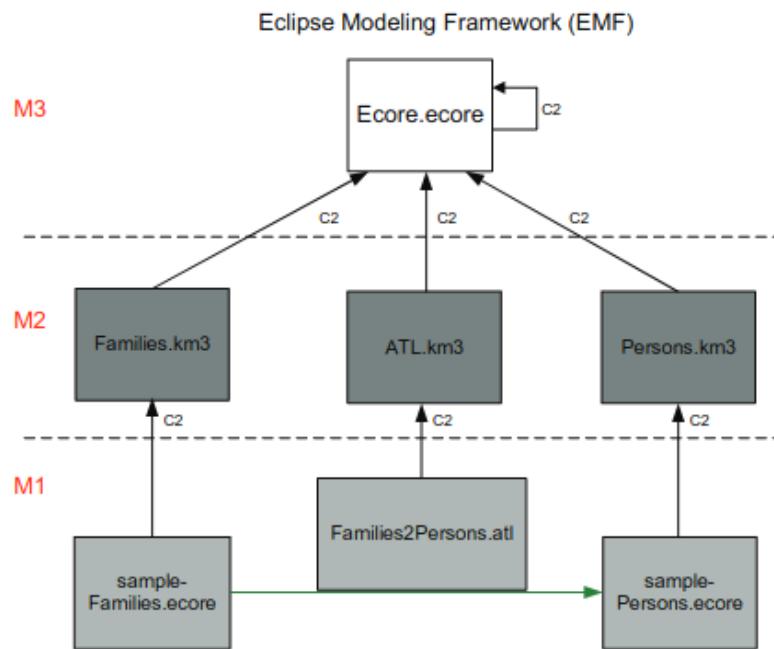
Hình 2.4. Minh họa bài toán chuyển đổi “Families to Persons”.

Áp dụng chuyển đổi ATL cho bài toán được thể hiện trong Hình 2.5. Giả thiết quá trình cài đặt môi trường ATL đã thành công, các thành phần sau đã có sẵn: định nghĩa của ngôn ngữ ATL (ATL.km3) và định nghĩa của siêu-siêu mô hình (Ecore.ecore). Để thực hiện việc chuyển đổi, cần xác định các thành phần như sau:

- Mô hình nguồn thể hiện dưới dạng XMI của đầu vào (sample-Families.ecore).
- Siêu mô hình nguồn (Families.km3) và siêu mô hình đích (Persons.km3): sử dụng ngôn ngữ đặc tả miền DSL.
- Mô hình chuyển đổi ATL (Families2Persons.atl): sử dụng ngôn ngữ ATL, xây dựng các hàm chức năng thực hiện chuyển đổi từ đối tượng nguồn sang đối tượng đích.

Sau khi quá trình chuyển đổi ATL được thực thi, đầu ra thu được là:

- Mô hình đích thể hiện dưới dạng XMI (sample-Persons.ecore)



Hình 2.5. Mô hình chuyển đổi ATL “Families to Persons”.

## 2.2. Mô hình quy trình nghiệp vụ

Khái niệm quy trình nghiệp vụ là quá trình diễn ra các công việc trong một tổ chức, doanh nghiệp, thể hiện được các thành phần chính của quá trình kinh doanh.

Quy trình nghiệp vụ cũng cho thấy các bước từ đầu đến cuối của quy trình để đảm bảo đáp ứng các mục tiêu của tổ chức đề ra.

Quản lý quy trình nghiệp vụ (Business Process Management - BPM) là một lĩnh vực trong hoạt động quản lý doanh nghiệp với phương pháp quản lý và tối ưu hóa các quy trình nghiệp vụ trong doanh nghiệp. Có thể nói BPM là giải pháp tiềm năng giúp các tổ chức không ngừng cải tiến dịch vụ và nâng cao chất lượng phục vụ khách hàng của mình. Với BPM, các doanh nghiệp có một hướng mới để triển khai ứng dụng công nghệ thông tin. Mặc dù mới ra đời hơn một thập kỷ, nhưng BPM đã phát triển rất nhanh trên nhiều mặt: phương pháp luận, phương pháp và công cụ, và cũng đã có nhiều kinh nghiệm triển khai thành công. Nhờ những ưu điểm vượt trội về tính đơn giản, dễ triển khai so với phương pháp ứng dụng công nghệ thông tin truyền thống, BPM đã được nhiều tổ chức doanh nghiệp trên toàn thế giới vận dụng hiệu quả và phổ biến.

### **2.2.1. Mô hình hóa quy trình nghiệp vụ**

Song song với quá trình khảo sát tìm hiểu về vấn đề hệ thống thì cách tiếp cận nghiệp vụ là phương pháp có hệ thống nhất để nắm bắt các yêu cầu của các ứng dụng nghiệp vụ. Theo [2], mô hình hóa quy trình nghiệp vụ là một kỹ thuật để tìm hiểu quy trình nghiệp vụ của một tổ chức và xác định các quy trình nghiệp vụ nào được hỗ trợ bởi hệ thống. Các hệ thống phần mềm càng phức tạp thì cách vận dụng các kỹ thuật mô hình hóa và phương thức mô hình hóa trực quan càng trở nên quan trọng hơn. Có nhiều yếu tố ảnh hưởng đến sự thành công của một dự án, trong đó việc có một tiêu chuẩn ngôn ngữ mô hình hóa chặt chẽ là một trong những yếu tố quan trọng nhất. Mục đích đầu tiên của mô hình hóa nghiệp vụ là lập mô hình những tổ chức thế giới thực: tạo ra các đối tượng (mô hình) để có thể thiết kế những chương trình máy tính thông qua hiện tượng thế giới thực như: người, đối tượng làm việc và cách thức thực hiện các tác vụ. Bên cạnh đó, thực hiện mô hình hóa quy trình nghiệp vụ cũng mang lại rất nhiều lợi ích cho tổ chức, doanh nghiệp:

- Hiểu được cấu trúc và các hoạt động của tổ chức được triển khai hệ thống.
- Hiểu được các vấn đề hiện tại trong tổ chức và xác định các vấn đề cần cải tiến.
- Bảo đảm các khách hàng, người dùng cuối và các nhà phát triển có sự hiểu biết chung về tổ chức.
- Thiết lập các yêu cầu hệ thống nhằm hỗ trợ tổ chức.

Phạm vi ảnh hưởng của mô hình hóa nghiệp vụ có thể biến đổi tùy theo nhu cầu và hệ thống nghiệp vụ cụ thể. Có thể đơn giản chúng ta chỉ nhắm vào việc tăng năng suất bằng cách cải tiến những quy trình đã tồn tại, hoặc đang tạo ra những sự cải tiến có ảnh hưởng lớn bằng cách thay đổi đáng kể những quy trình nghiệp vụ dựa trên sự phân tích kỹ lưỡng các mục tiêu và khách hàng của tổ chức. Để đạt được những mục đích trên, luồng công việc mô hình hóa nghiệp vụ mô tả một bức tranh tổng quát về tổ chức, từ đó xác định các quy trình (process), các vai trò (role), và các trách

nhiệm của tổ chức này trong mô hình ca sử dụng nghiệp vụ (business use-case model) và mô hình đối tượng nghiệp vụ (business object model) [2].

### **2.2.2. Tiêu chuẩn ký hiệu và mô hình hóa quy trình nghiệp vụ BPMN**

Theo [1], vấn đề xảy ra trong quá trình tin học hóa các quy trình nghiệp vụ đó là các tác nhân thực hiện nghiệp vụ, người phân tích quy trình nghiệp vụ, người phát triển kỹ thuật (người chịu trách nhiệm tin học hóa các quy trình nghiệp vụ) và người quản lý nghiệp vụ (người giám sát và quản lý quy trình nghiệp vụ) gặp khó khăn trong việc hiểu ý tưởng của nhau. Hơn nữa, chính những nhà phân tích nghiệp vụ của các tổ chức khác nhau, nhiều khi cũng không thể giao tiếp trong quá trình liên thông các quy trình nghiệp vụ với nhau. Để giải quyết vấn đề này, cần thiết phải có các ký hiệu (notation) tiêu chuẩn tương ứng biểu diễn các phần tử nghiệp vụ như các sự kiện, hoạt động, luồng dữ liệu, các đơn vị tổ chức,... Một tập các ký hiệu đồ họa về mô hình hóa quy trình nghiệp vụ xác định các biểu tượng (symbol) cho các phần tử quy trình nghiệp vụ, ý nghĩa cũng như các khả năng kết hợp của chúng.

Tiêu chuẩn ký hiệu và mô hình hóa quy trình nghiệp vụ (Business Process Model and Notation - BPMN) phát triển với mục đích chính là làm cầu nối khoảng cách về thông tin giữa các bên liên quan thường xuyên xảy ra vấn đề trong việc thiết kế và triển khai quy trình nghiệp vụ, đã và đang được sử dụng rộng rãi trong nhiều tổ chức. BPMN hỗ trợ cho cả người dùng kỹ thuật và người dùng nghiệp vụ trong việc quản lý các quy trình nghiệp vụ bằng cách đưa ra một tập các ký hiệu chung, có tính trực quan và dễ hiểu cho người dùng.

#### **2.2.2.1. Lịch sử phát triển của BPMN**

Ban đầu, BPMN được phát triển bởi Tổ chức Sáng kiến quản lý quy trình nghiệp vụ (Business Process Management Initiative – BPMI), một tổ chức gồm các công ty về phần mềm [1]. Ở giai đoạn khởi đầu, mục tiêu là cung cấp một tập các ký hiệu đồ họa mô tả quy trình được thể hiện trong Ngôn ngữ mô hình hóa quy trình nghiệp vụ (Business Process Modeling Language – BPML). BPML được sử dụng để xác định các mô tả quy trình có thể được thực thi bởi một hệ thống quản lý quy trình nghiệp vụ (Business Process Management System).

Phiên bản đầu tiên của BPMN được phát triển bởi nhóm của Stephen A. White thuộc IBM năm 2004. Trong thời gian này, BPMI đã trở thành một nhóm thuộc Tổ chức quản lý đối tượng (Object Management Group – OMG). OMG là một tổ chức nổi tiếng về các tiêu chuẩn phần mềm, đặc biệt là UML. Năm 2006, BPMN phiên bản 1.0 chính thức được chấp nhận là một tiêu chuẩn của tổ chức OMG [1].

Sau đó, OMG công bố phiên bản BPMN v1.1 vào tháng 01/2008 và công bố BPMN v1.2 vào tháng 01/2009 với một số thay đổi nhỏ. Phiên bản BPMN v2.0 với nhiều thay đổi và mở rộng so với các phiên bản cũ, đã được OMG công bố vào tháng 01/2011. Phiên bản gần đây nhất là BPMN v2.0.2 được OMG công bố tháng 12/2013. Nội dung phiên bản BPMN v2.0.2 không khác biệt nhiều so với BPMN v2.0, chỉ chỉnh

sửa một số lỗi nhỏ về văn bản. Trong năm 2013, BPMN cũng chính thức trở thành tiêu chuẩn quốc tế ISO/IEC 19510:2013.

### 2.2.2.2. Các phần tử (element) của BPMN

- **Flow Objects:** là tập hợp của các phần tử chính, được gọi là các đối tượng luồng.

Bảng 2.1. BPMN Flow Objects [23]

Event	<p>Event được đại diện bởi một hình tròn và là một điều "diễn ra" trong quá trình kinh doanh. Những Events ảnh hưởng đến luồng của quy trình và thường có nguyên nhân (kích hoạt) hoặc một tác động (kết quả).</p> <p>Có ba loại Event, dựa trên thời điểm chúng ảnh hưởng đến luồng: Start, Intermediate và End (thứ tự tương ứng với các hình bên phải).</p>	
Activity	<p>Activity được biểu thị bằng một hình chữ nhật được bo góc và là một thuật ngữ chung cho hoạt động mà công ty thực hiện. Các loại Activity là: Task và Sub-Process.</p>	
Gateway	<p>Gateway thể hiện bằng hình thoi và được sử dụng để kiểm soát sự phân kỳ và hội tụ của chuỗi hoạt động. Tại đây các hoạt động sẽ được kết hợp hoặc phân nhánh thành các luồng xử lý trong quy trình.</p>	

- **Connecting Objects:** Các Flow Objects được kết nối với nhau trong một biểu đồ để tạo ra cấu trúc cơ bản của một quy trình nghiệp vụ. Có ba loại Connecting Objects cung cấp chức năng này.

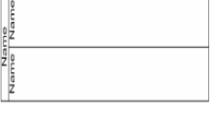
Bảng 2.2. BPMN Connecting Objects [23]

Sequence Flow	Được biểu thị bằng một đường mũi tên liền nét và được sử dụng để hiển thị thứ tự (trình tự) mà các hoạt động sẽ được thực hiện trong một quy trình.	
Message Flow	Được thể hiện bằng một đường mũi tên đứt nét và sử dụng để hiển thị luồng tin nhắn giữa những người tham gia quá trình (các thực thể kinh doanh hoặc vai trò kinh doanh).	
Association	Được biểu thị bằng một đường mũi tên chấm chấm và sử dụng để liên kết dữ liệu, text và các	

	artifacts khác với các đối tượng luồng. Các liên kết hiển thị các đầu vào và đầu ra của các hoạt động.	
--	--	--

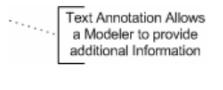
- **Swimlanes:** Nhiều phương pháp mô hình hóa quy trình sử dụng khái niệm Swimlane như một cơ chế để tổ chức các hoạt động thành các danh mục trực quan riêng biệt để minh họa các chức năng khác nhau. BPMN hỗ trợ swimlanes với hai cấu trúc chính:

Bảng 2.3. BPMN Swimlanes [23]

Pool	Đại diện cho một người tham gia quá trình. Nó cũng thể hiện như một vùng chứa để phân vùng một tập hợp các hoạt động với các Pool khác.	
Lane	Một phân vùng con trong một Pool và có cùng chiều dài của Pool. Lane được sử dụng để tổ chức và phân loại các hoạt động.	

- **Artifacts:** BPMN được thiết kế để cho phép sự linh động trong việc mở rộng các ký hiệu và cung cấp khả năng thêm ngữ cảnh phù hợp với một tình huống mô hình cụ thể. Bất kỳ số lượng Artifacts nào cũng có thể được thêm vào biểu đồ, phù hợp với bối cảnh của các quy trình kinh doanh đang được mô hình hóa. Hiện tại phiên bản của đặc tả BPMN chỉ xác định trước ba loại Artifacts.

Bảng 2.4. BPMN Artifacts [23]

Data Object	Một cơ chế để chỉ ra cách thức dữ liệu được yêu cầu hoặc tạo ra bởi các hoạt động. Nó được kết nối với các hoạt động thông qua Associations.	
Group	Được sử dụng cho mục đích tài liệu hóa hoặc phân tích, nhưng không ảnh hưởng đến Sequence Flow	
Annotation	Một cơ chế để cung cấp các thông tin văn bản bổ sung của một biểu đồ BPMN	 Text Annotation Allows a Modeler to provide additional information

## 2.3. Mô hình ca sử dụng

### 2.3.1. Ngôn ngữ mô hình thống nhất

UML là viết tắt của Unified Modeling Language, là một ngôn ngữ mô hình hóa hình ảnh tiêu chuẩn để mô hình hóa nghiệp vụ và các quy trình tương tự, phân tích,

thiết kế và triển khai các hệ thống dựa trên phần mềm. UML được tạo ra bởi Tổ chức Quản lý Đối tượng (OMG) và phiên bản UML 1.0 đã được đề xuất và thông qua bởi OMG năm 1997.

Khác với các ngôn ngữ lập trình thông thường khác như C++, Java, COBOL,... UML là một ngôn ngữ hình ảnh được sử dụng để tạo các bản thiết kế phần mềm, được mô tả như một ngôn ngữ mô hình trực quan có mục đích chung để trực quan hóa, xây dựng và lập tài liệu hệ thống phần mềm. Mặc dù UML thường được sử dụng để mô hình hóa các hệ thống phần mềm, nhưng không bị giới hạn trong ranh giới này. UML có thể được sử dụng để mô hình hóa các hệ thống không phải phần mềm ví dụ quy trình trong một đơn vị sản xuất,...

“Một bức tranh có giá trị bằng một nghìn từ” - điều này hoàn toàn phù hợp để mô tả UML. Có nhiều mục tiêu để phát triển UML nhưng quan trọng nhất là xác định một ngôn ngữ mô hình hóa chung mà tất cả các nhà lập mô hình đều có thể sử dụng, đơn giản, dễ hiểu và dễ sử dụng. Bên cạnh đó, biểu đồ UML không chỉ dành cho các nhà phát triển mà còn cho người dùng doanh nghiệp, những người bình thường và bất kỳ ai quan tâm đến việc hiểu hệ thống. Tóm lại, mục tiêu của UML có thể được định nghĩa là một cơ chế mô hình hóa đơn giản để mô hình hóa tất cả các hệ thống thực tế có thể có trong môi trường phức tạp ngày nay.

UML được liên kết với thiết kế và phân tích hướng đối tượng. UML sử dụng các phần tử và hình thức liên kết giữa chúng để tạo thành biểu đồ. Các biểu đồ trong UML có thể được phân loại như sau:

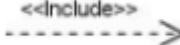
- Biểu đồ cấu trúc - Chụp các khía cạnh hoặc cấu trúc tĩnh của hệ thống và các bộ phận của nó ở các mức độ trừu tượng, thực hiện khác nhau và cách chúng có liên quan với nhau. Các yếu tố trong biểu đồ cấu trúc đại diện cho các khái niệm có ý nghĩa của một hệ thống. Biểu đồ cấu trúc bao gồm: Biểu đồ thành phần, Biểu đồ đối tượng, Biểu đồ lớp và Biểu đồ triển khai.
- Biểu đồ hành vi - Chụp các khía cạnh động hoặc hành vi của hệ thống hay hành 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. Biểu đồ hành vi bao gồm: Biểu đồ ca sử dụng, Biểu đồ trạng thái, Biểu đồ hoạt động và Biểu đồ tương tác.

### 2.3.2. Biểu đồ ca sử dụng

Biểu đồ ca sử dụng mô tả tổng thể hệ thống một cách đơn giản, mô hình hóa các chuỗi hành động và xác định tương tác giữa người dùng và hệ thống.

Bảng 2.5. Các khái niệm trong ca sử dụng

Khái niệm	Ký hiệu	Ý nghĩa
Actor		Người dùng hệ thống hay một hệ thống khác. Actor có thể chỉ cung cấp thông tin cho hệ thống, chỉ lấy thông tin từ hệ thống hoặc nhận

		thông tin từ hệ thống và cung cấp thông tin cho hệ thống.
Use Case		Một khái niệm chức năng được thực hiện bởi hệ thống để mang lại một kết quả có giá trị đối với một Actor cụ thể.
Relationship		Quan hệ giữa các phần tử trong mô hình, bao gồm kết hợp (association), tổng quát hóa (generalization).
Include		Một Use Case có thể có chức năng của 1 Use Case khác.
Extend		Dùng để chỉ các hành vi không bắt buộc (có thể có hoặc không), các hành vi theo điều kiện nhất định

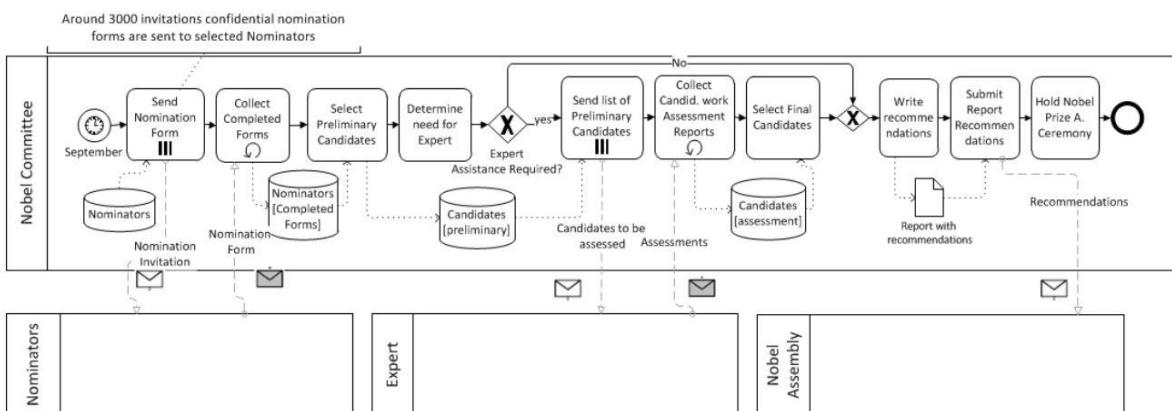
## CHƯƠNG 3. PHƯƠNG PHÁP SINH TỰ ĐỘNG CA SỬ DỤNG TỪ MÔ HÌNH QUY TRÌNH NGHIỆP VỤ

Trong chương này, luận văn sẽ đề xuất phương pháp BPMN2UseCase - tự động chuyển đổi mô hình quy trình nghiệp vụ sang ca sử dụng và trình bày cụ thể về quy trình, cơ chế hoạt động chương trình cũng như bộ quy tắc chuyển đổi và thuật toán thực thi giải pháp.

### 3.1. Giới thiệu

Trong quá trình phát triển các hệ thống hiện nay, vấn đề chuyển đổi mô hình quy trình nghiệp vụ sang mô hình ca sử dụng có thể mang một ý nghĩa ứng dụng rất lớn và đem lại nhiều lợi ích cụ thể. Đây có thể coi là một công cụ hữu dụng hỗ trợ cho phát triển phần mềm, đặc biệt là khâu xác định chức năng phần mềm. Với ưu thế tận dụng các mô hình nghiệp vụ sẵn có, hướng tiếp cận này có thể giảm thiểu được các bước không cần thiết và tiết kiệm được nguồn lực, thời gian và chi phí. Mô hình đầu ra rõ ràng, trực quan giúp cho các bên liên quan có thể nắm bắt được hệ thống một cách nhanh chóng, bao gồm phạm vi của hệ thống, các tác nhân trong hệ thống và các tình huống sử dụng với các tương tác giữa tác nhân và hệ thống.

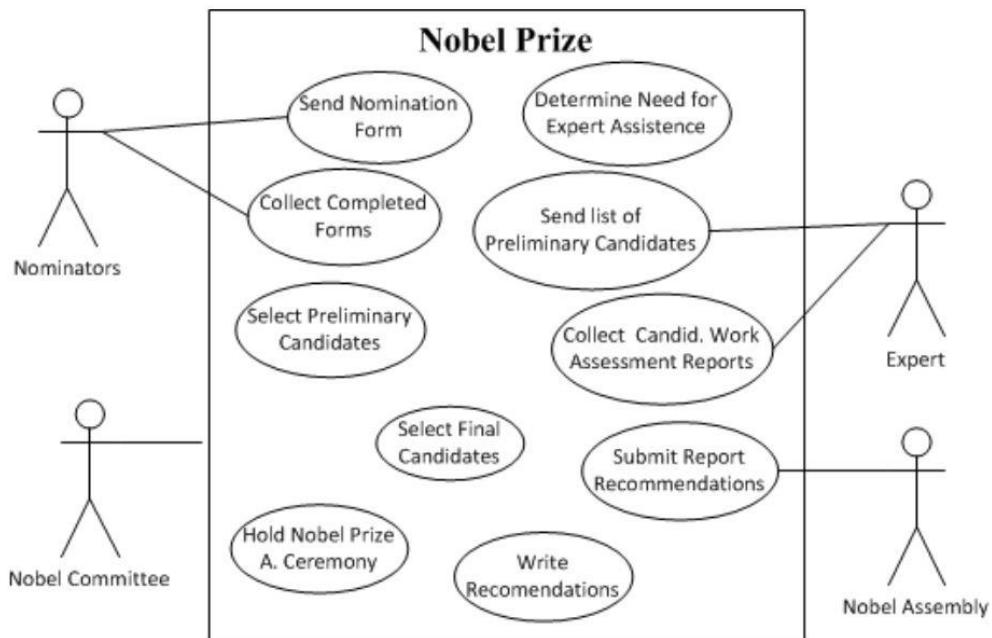
Tuy nhiên, bài toán chuyển đổi này cũng xuất hiện một số khó khăn và thách thức. Cụ thể, hai mô hình quy trình nghiệp vụ và ca sử dụng tuy có những sự tương đồng về mặt khái niệm tổng quát nhưng phần lớn các siêu khái niệm của các phần tử đều tương đối khác biệt. Điều này đòi hỏi sự phân tích và nghiên cứu kỹ đối với từng phần tử để có thể tìm ra sự tương quan để có thể xây dựng các quy tắc chuyển đổi. Bên cạnh đó, quy trình nghiệp vụ nếu chưa được chuẩn hóa hay không sử dụng các tiêu chuẩn mô hình ký hiệu phù hợp có thể sẽ đem lại nhiều khó khăn và vướng mắc khi đánh giá và trích xuất các phần tử của mô hình đầu vào. Một ví dụ minh họa cho quy trình nghiệp vụ thực tế là Nobel Prize, biểu diễn trong Hình 3.1.



Hình 3.1. Mô hình quy trình nghiệp vụ Nobel Prize [9].

Đây là một mô hình giàu thông tin và bao gồm tương đối đầy đủ các loại phần tử hay siêu khái niệm phổ biến nhất của quy trình nghiệp vụ. Tuy nhiên, do gấp phai

khó khăn trong việc phân tích và xác định các luật chuyển đổi, mô hình ca sử dụng đầu ra còn sơ sài, chưa đầy đủ và thiếu nhiều thông tin [9] (Hình 3.2).

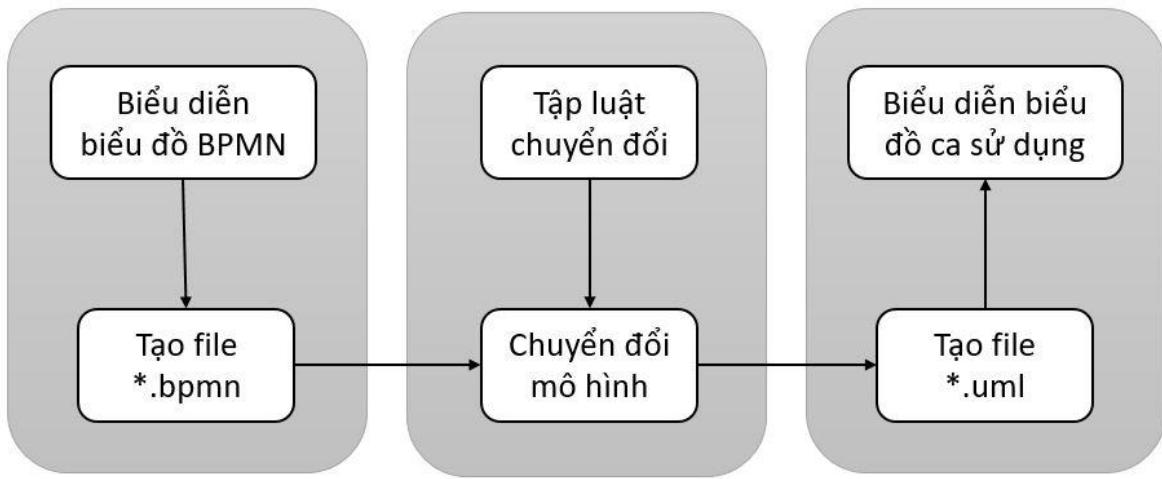


Hình 3.2. Mô hình ca sử dụng Nobel Prize [9].

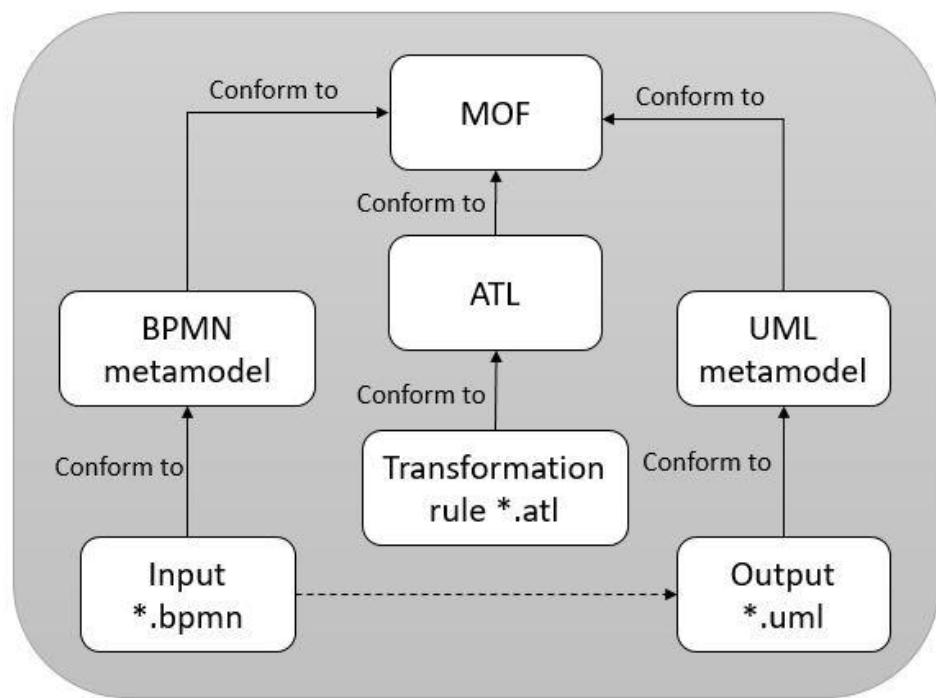
Sau quá trình nghiên cứu tìm hiểu, luận văn đề xuất phương pháp sinh tự động biểu đồ ca sử dụng từ mô hình quy trình nghiệp vụ dựa trên chuyển đổi mô hình. Nguyên lý nền tảng của giải pháp là sử dụng các siêu mô hình metamodels của cả đầu vào và đầu ra, dựa trên sự tương quan về định nghĩa và ý nghĩa của các khái niệm giữa hai siêu mô hình này để xây dựng các luật chuyển đổi. Nội dung của các mô hình được thể hiện qua định dạng giao thức XMI - một tiêu chuẩn được sử dụng rộng rãi để lưu trữ và trao đổi các dữ liệu đặc tả. Từ đó, việc biểu diễn biểu đồ các mô hình trở nên dễ dàng hơn với sự hỗ trợ của các công cụ mô hình hóa UML phổ biến hiện nay.

### 3.2. Tổng quan về phương pháp BPMN2UseCase

Để thực hiện sinh tự động biểu đồ ca sử dụng từ mô hình quy trình nghiệp vụ, luận văn đã phát triển chương trình BPMN2UseCase sử dụng ngôn ngữ chuyển đổi mô hình ATL và nền tảng mô hình hóa Eclipse (EMF). Hình 3.3 thể hiện quy trình thực hiện của phương pháp, bao gồm 3 bước riêng biệt: biểu diễn mô hình BPMN đầu vào, chuyển đổi BPMN2UseCase dựa trên tập luật định sẵn và biểu diễn mô hình UseCase đầu ra.



Hình 3.4 mô tả tổng quan cách thức hoạt động của chương trình được cài đặt, tương ứng với ba bước thực hiện quy trình trên.



Một ưu điểm quan trọng của cách tiếp cận là dựa trên ký hiệu mô hình tiêu chuẩn, nên có thể dễ dàng tích hợp vào các nền tảng và công cụ sẵn có. Trong chương trình này, mô hình quy trình nghiệp vụ được thiết kế và sử dụng bởi công cụ *Eclipse BPMN2 Modeler* và tạo ra file đầu vào chuẩn định dạng .bpn như ví dụ Hình 3.5. Sau quá trình chuyển đổi với ATL, chương trình sẽ tạo ra file đầu ra chuẩn XMI định dạng .uml như Hình 3.6. Sau khi đưa vào công cụ *Eclipse Papyrus*, biểu đồ ca sử dụng sẽ được tạo ra một cách trực quan và dễ tiếp cận hơn.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- origin at X=0.0 Y=0.0 -->
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:bpmn2="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:dc="http://www.omg.org/spec/DC/20100524/DC" xmlns:ext="http://www.omg.org/spec/DC/20100524/DC" xmlns:ext="http://org.eclipse.bpmn2/ext" id="Definitions_1" exporter="org.eclipse.bpmn2.modeler.core" exporterVersion="1.1.2.201502162150" targetNamespace="http://org.eclipse.bpmn2/default/collaboration">
  <bpmn2:collaboration id="Collaboration_1" name="Default Collaboration">
    <bpmn2:participant id="Participant_3" name="Khách hàng" processRef="Process_3"/>
    <bpmn2:participant id="Participant_4" name="Agent đặt xe" processRef="Process_4"/>
    <bpmn2:participant id="Participant_5" name="Lái xe" processRef="Process_5"/>
    <bpmn2:participant id="Participant_6" name="Process 3 Pool" processRef="Process_3">
      <bpmn2:extensionElements>
        <ext:style/>
      </bpmn2:extensionElements>
    </bpmn2:participant>
    <bpmn2:messageFlow id="MessageFlow_1" sourceRef="Task_1" targetRef="Task_2"/>
    <bpmn2:messageFlow id="MessageFlow_2" sourceRef="Task_6" targetRef="IntermediateThrowEvent_2"/>
    <bpmn2:messageFlow id="MessageFlow_3" name="Thông báo agent" sourceRef="Task_8" targetRef="Task_6"/>
    <bpmn2:messageFlow id="MessageFlow_4" name="Điều phòi lái xe" sourceRef="Task_6" targetRef="Task_7"/>
  </bpmn2:collaboration>
  <bpmn2:process id="Process_3" name="Khách hàng Process" definitionalCollaborationRef="Collaboration_1" isExecutable="false">
    <bpmn2:startEvent id="StartEvent_1" name="Khởi tạo">
      <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
    </bpmn2:startEvent>
    <bpmn2:task id="Task_1" name="Đặt xe">
      <bpmn2:incoming>SequenceFlow_1</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_12</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:intermediateThrowEvent id="IntermediateThrowEvent_2" name="Chi tiết đặt xe">
      <bpmn2:incoming>SequenceFlow_12</bpmn2:incoming>
      <bpmn2:messageEventDefinition id="MessageEventDefinition_2"/>
    </bpmn2:intermediateThrowEvent>
    <bpmn2:sequenceFlow id="SequenceFlow_1" sourceRef="StartEvent_1" targetRef="Task_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_12" sourceRef="Task_1" targetRef="IntermediateThrowEvent_2"/>
  </bpmn2:process>
  <bpmn2:process id="Process_4" name="Agent đặt xe Process" definitionalCollaborationRef="Collaboration_1" isExecutable="false">
```

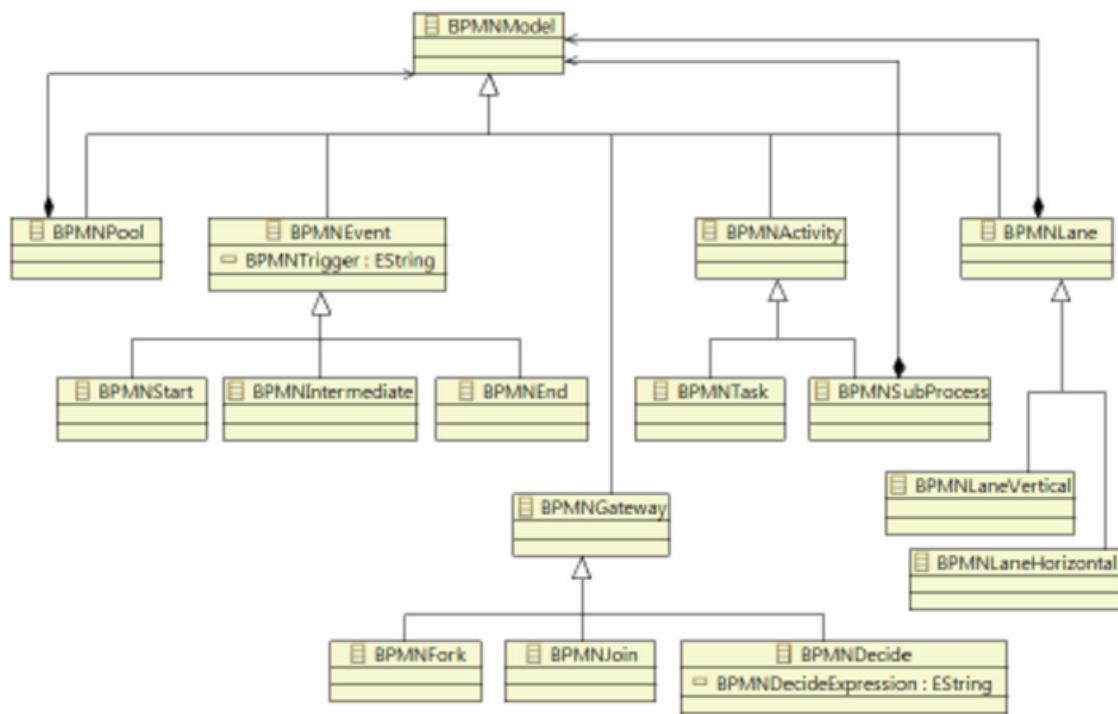
Hình 3.5. Định dạng đầu vào \*.bpmn.

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmi:version="20131001" xmlns:xmi="http://www.omg.org/spec/XMI/20131001" xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" xmlns:uml="http://www.eclipse.org/uml2/2.0.0/UML" id="DksRwOChEey_TNo0V40hYg" name="uct">
  <packageImport xmi:type="uml:PackageImport" xmi:id="_Dox340ChEey_TNo0V40hYg">
    <importedPackage xmi:type="uml:Model" href="pathmap:/UML_LIBRARIES/UMLPrimitiveTypes.library.uml#_0"/>
  </packageImport>
  <packagedElement xmi:type="uml:Actor" xmi:id="_BgbG00ChEey_TNo0V40hYg" name="Actor"/>
  <packagedElement xmi:type="uml:UseCase" xmi:id="_HYC3Q0ChEey_TNo0V40hYg" name="UseCase"/>
  <packagedElement xmi:type="uml:Association" xmi:id="_gL4xs01EEey9hcEaMSS6Cw" memberEnds="_gL5Ywui1EEey9hcEaMSS6Cw _gL5Yw-1EEey9hcEaMSS6Cw">
    <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_gL5Yw01EEey9hcEaMSS6Cw" source="org.eclipse.papyrus">
      <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_gL5Ywe1EEey9hcEaMSS6Cw" key="nature" value="UML_Nature"/>
    </eAnnotations>
    <ownedEnd xmi:type="uml:Property" xmi:id="_gL5Ywui1EEey9hcEaMSS6Cw" name="usecase" type="_HYC3Q0ChEey_TNo0V40hYg" association="_gL4xs01EEey9hcEaMSS6Cw" />
    <ownedEnd xmi:type="uml:Property" xmi:id="_gL5Yw-1EEey9hcEaMSS6Cw" name="actor" type="_BgbG00ChEey_TNo0V40hYg" association="_gL4xs01EEey9hcEaMSS6Cw" />
  </packagedElement>
  <packagedElement xmi:type="uml:UseCase" xmi:id="_M34l4010EEey9hcEaMSS6Cw" name="UseCase 1"/>
  <packagedElement xmi:type="uml:UseCase" xmi:id="_OBYNM010EEey9hcEaMSS6Cw" name="UseCase 2"/>
  <packagedElement xmi:type="uml:Dependency" xmi:id="_rym_w010EEey9hcEaMSS6Cw" name="&lt;&lt;precede&gt;>" client="_M34l4010EEey9hcEaMSS6Cw" supplier="_OBYNM010EEey9hcEaMSS6Cw" />
  <packagedElement xmi:type="uml:UseCase" xmi:id="_0Kzyo10EEey9hcEaMSS6Cw" name="UseCase 1">
    <extend xmi:type="uml:Extend" xmi:id="_BO1ck01QEey9hcEaMSS6Cw" extendedCase="_12PZw010EEey9hcEaMSS6Cw" extensionLocation="_BO1po01QEey9hcEaMSS6Cw"/>
  </packagedElement>
```

Hình 3.6. Định dạng đầu ra \*.uml.

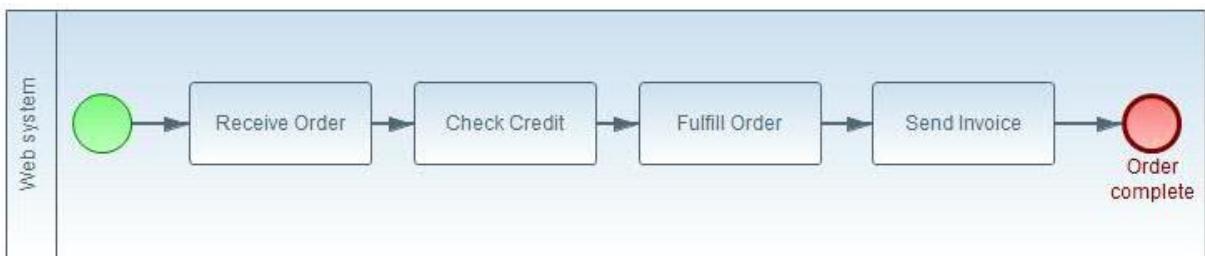
### 3.3. Biểu diễn mô hình BPMN

Theo [14], rất nhiều kỹ thuật và công cụ đã được phát triển để sử dụng cho việc mô hình hóa quy trình nghiệp vụ, mà các kỹ thuật này phần lớn đều tập trung vào mô hình hóa các khía cạnh khác ngoài khía cạnh hành vi, ví dụ như cấu trúc tổ chức hay các đối tượng dữ liệu. Trong khi đó, biểu đồ ca sử dụng thuộc loại biểu đồ hành vi, không có các khái niệm khác ngoài hành vi. Do đó, các mô hình này không có sự tương quan và rất khó để tạo ra sự chuyển đổi phù hợp. Trong luận văn này, BPMN đã được lựa chọn sử dụng cho bước mô hình khái niệm của quy trình nghiệp vụ, với các định nghĩa về hành vi là Flow Objects và các mối quan hệ kết nối giữa chúng là Sequence Flow, Message Flow và Association. Hình 3.7 thể hiện metamodel BPMN đơn giản.



Hình 3.7. Siêu mô hình lược giản BPMN [8].

Minh họa cho quy trình nghiệp vụ đơn giản là quá trình xử lý đơn hàng online của một webshop thể hiện trong Hình 3.8. Đây là một luồng công việc cơ bản thông thường của hệ thống bán hàng sau khi nhận được đơn hàng online từ người dùng. Đối tượng thực hiện ở đây là Web system, tương ứng với siêu khái niệm BPMNPool - đại diện cho một người tham gia quy trình. Quy trình gồm bốn bước theo thứ tự bao gồm: Receive Order, Check Credit, Fulfill Order và Send Invoice và được thể hiện bằng các siêu khái niệm BPMNTask - đại diện cho các hoạt động cần thực hiện. Sự kiện bắt đầu và kết thúc quá trình lần lượt tương ứng với BPMNStart và BPMNEnd nằm trong siêu khái niệm BPMNEvent - đại diện cho các sự kiện diễn ra trong suốt quy trình nghiệp vụ. Đây là một ví dụ đơn giản để có thể giúp hình dung rõ hơn về quy trình nghiệp vụ thực tế và cách thể hiện các siêu khái niệm tương ứng trong siêu mô hình BPMN.

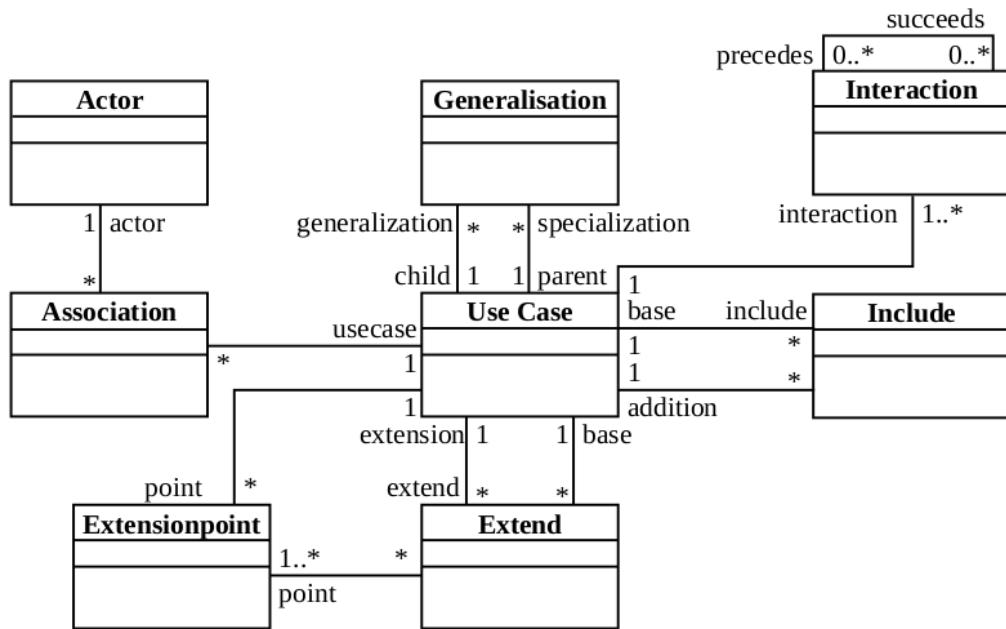


Hình 3.8. Mô hình quy trình nghiệp vụ xử lý đơn hàng online.

### 3.4. Biểu diễn mô hình ca sử dụng

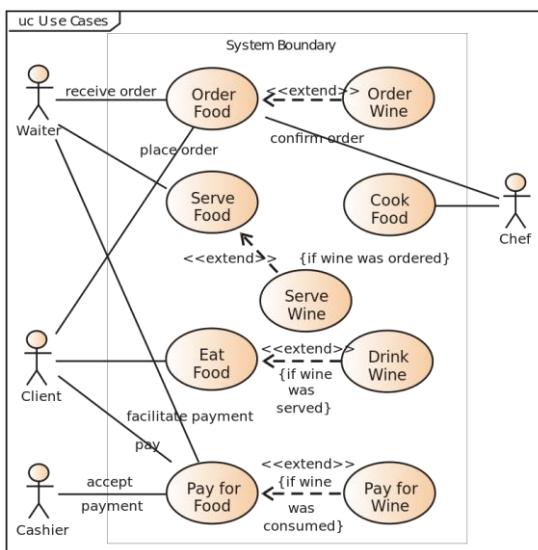
Một biểu đồ ca sử dụng mô tả cách một thực thể sử dụng hay tương tác với hệ thống như thế nào. Để thể hiện điều đó, biểu đồ ca sử dụng thường có các yếu tố thành phần là Actor và Use Case. Trong đó Actor là thể hiện của thực thể tác động

tới hệ thống, Use Case là tập hợp các tương tác giữa các Actor và hệ thống, từ đó mô tả một phần hành vi của hệ thống. Hình 3.9 thể hiện siêu mô hình của ca sử dụng dưới dạng đơn giản.



Hình 3.9. Siêu mô hình ca sử dụng đơn giản [13].

Một ví dụ mô hình ca sử dụng được minh họa trong Hình 3.10, thể hiện các hành vi hoạt động của một nhà hàng. Biểu đồ xác định 4 tác nhân chính của hệ thống là Client, Waiter, Cashier và Chef, tương ứng với siêu khái niệm Actor. 9 ca sử dụng bao gồm Order Food, Serve Food, Pay for Food, ... tương ứng với siêu khái niệm Use Case là những hoạt động chính xảy ra tại nhà hàng. Quan hệ giữa tác nhân và ca sử dụng chính là những Association và giữa các ca sử dụng với nhau là Include, Extend, ... Với cấu trúc biểu đồ đơn giản, rõ ràng, mô hình ca sử dụng giúp người dùng có thể nắm rõ cách thức hoạt động của hệ thống và tương tác của người dùng với hệ thống một cách dễ dàng hơn.



Hình 3.10. Mô hình hóa ca sử dụng trong một nhà hàng.

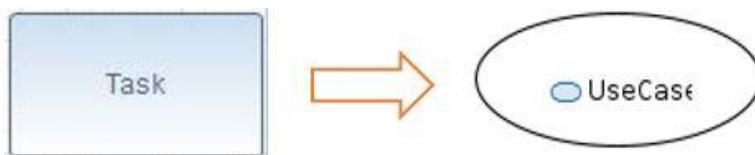
### 3.5. Xác định các luật chuyển đổi mô hình

Ở phần này, luận văn đưa ra một tập hợp các mối tương quan giữa các đối tượng và các quan hệ của hai mô hình trên (dưới dạng XMI), từ đó cho phép xây dựng phương pháp để chuyển đổi mô hình một cách hợp lý. Cụ thể, tập luật được chia thành ba nhóm: chuyển đổi đối tượng, chuyển đổi quan hệ giữa các đối tượng và chuyển đổi quan hệ với dữ liệu. Dựa trên việc phân tích và trích xuất các yếu tố thành phần tương ứng với các luật chuyển đổi đã được xác định, luận văn sẽ phát triển các hàm ATL thực hiện quá trình chuyển đổi, xác định bởi từ khóa cú pháp rule.

#### 3.5.1. Luật chuyển đổi cho các đối tượng (Rules for Elements)

##### Luật RE1: Ánh xạ Task sang UseCase [9]

Trong BPMN, một Task biểu diễn một tương tác người dùng với hệ thống. Trong khi đó, một ca sử dụng thể hiện một mục tiêu người dùng muốn đạt được khi sử dụng hệ thống [16]. Do đó, có thể chuyển đổi một Task trong BPMN sang một Use Case, tên của Use Case chính là tên của Task.



Hình 3.11. Luật RE1 - Ánh xạ Task sang UseCase.

Siêu khái niệm Task được chuyển đổi sang siêu khái niệm UseCase, trong đó thuộc tính name sẽ được giữ nguyên và gán tương ứng cho đối tượng UseCase mới được tạo ra. Hình 3.12 mô tả cấu trúc cài đặt của luật RE1.

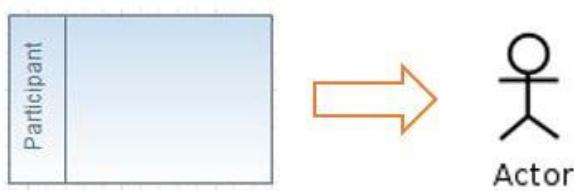
```

29 rule Task2UseCasenAssociation {
30   from
31     s : BPMN!Task
32   to
33     t : UML!UseCase (
34       name <- s.name
35     )
  
```

Hình 3.12. Hàm chuyển đổi RE1 - Task2UseCase.

##### Luật RE2: Ánh xạ Participant sang Actor [7, 9]

Trong BPMN, bất kỳ ai có hoạt động để thực hiện, tương tự như một quy trình, được gọi là một Participant [23]. Trong biểu đồ ca sử dụng, một Actor thể hiện cho một người dùng của hệ thống. Từ đó, một Participant trong quy trình nghiệp vụ có thể ánh xạ sang một Actor trong ca sử dụng.



Hình 3.13. Luật RE2 - Ánh xạ Participant sang Actor.

Siêu khái niệm Participant được chuyển đổi sang siêu khái niệm Actor, trong đó thuộc tính name sẽ được giữ nguyên và gán tương ứng cho đối tượng Actor mới được tạo ra. Hình 3.14 mô tả cấu trúc cài đặt của luật RE2.

```

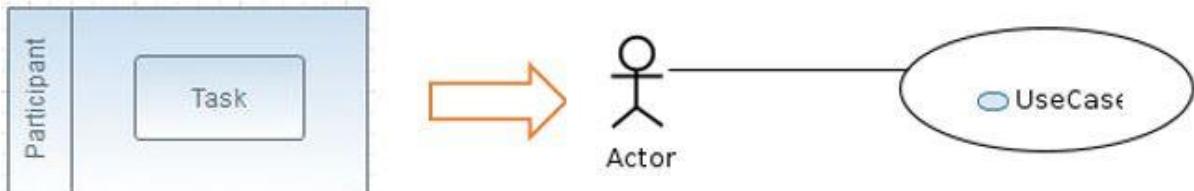
23 rule Participant2Actor {
24   from s : BPMN!Participant
25   to t : UML!"uml::Actor" (
26     name <- s.name
27   )
28 }
```

Hình 3.14. Hàm chuyển đổi RE2 - Participant2Actor.

### 3.5.2. Luật chuyển đổi cho các quan hệ giữa các đối tượng (Rules for Association)

**Luật RA1:** Ánh xạ quan hệ giữa Participant và Task sang Association giữa Actor và UseCase [7, 9]

Từ 2 luật RE1 và RE2, một Participant và một Task trong quy trình nghiệp vụ có thể tương ứng sinh ra một Actor và một Use Case trong biểu đồ ca sử dụng. Do đó, liên kết giữa Participant và Task sẽ tương ứng với liên kết Actor và Use Case trên.



Hình 3.15. Luật RA1 - Ánh xạ giữa Participant và Task sang Association.

Quan hệ giữa hai siêu khái niệm Task và Participant ánh xạ sang siêu khái niêm Association với thuộc tính ownedEnd xác định Use Case và Actor. Để thực hiện việc này, mỗi đối tượng ownedEnd được gán name và type của UseCase và Actor tương ứng với khái niêm nguồn. Chi tiết cài đặt được thể hiện trong Hình 3.16.

```

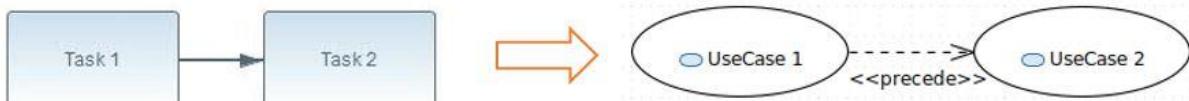
29 rule Task2UseCasenAssociation {
30   from
31     s : BPMN!Task
32   to
33     t : UML!UseCase (
34       name <- s.name
35     )
36     , a:UML!Association (
37       ownedEnd <- Set {endUsecase, endActor}
38     )
39     , endUsecase: UML!Property (
40       name <- s.name
41       , type <- thisModule.getUsecase(s.name)
42     )
43     , endActor : UML!Property (
44       name <- thisModule.getParticipantName(s)
45       , type <- thisModule.getActor(thisModule.getParticipantName(s))
46     )
47 }

```

Hình 3.16. Hàm chuyển đổi RA1 - Task2UseCasenAssociation.

### Luật RA2: Ánh xạ SequenceFlow sang Dependency

Trong BPMN, dòng trình tự (SequenceFlow) được sử dụng để thể hiện thứ tự thực hiện các tác vụ trong một quy trình [17]. Khái niệm này tương đồng với quan hệ Dependency trong mô hình ca sử dụng [11]. Luận văn bổ sung thêm mô tả <>precede>> để thể hiện một ca sử dụng hoàn thành trước khi ca sử dụng tiếp theo xảy ra. Hình 3.17 biểu diễn quy tắc chuyển đổi này.



Hình 3.17. Luật RA2 - Ánh xạ SequenceFlow sang Dependency.

Siêu khái niệm SequenceFlow thể hiện luồng thực hiện giữa hai Task được chuyển đổi thành quan hệ Dependency với thuộc tính name là <>precedes>> để thể hiện thứ tự của các UseCase sinh ra.

```

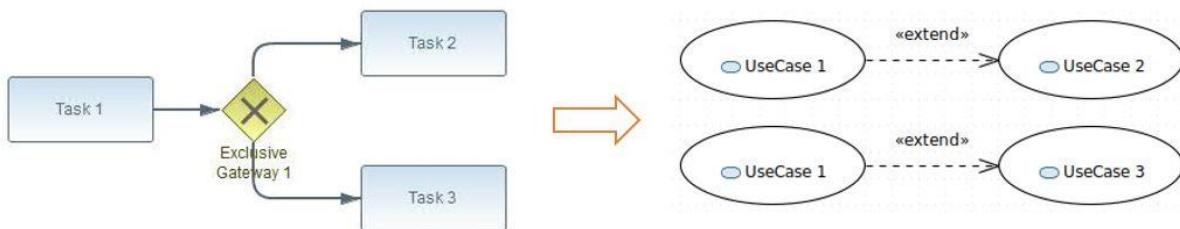
48 rule SequenceFlow2Dependency {
49   from s: BPMN!SequenceFlow (
50     s.sourceRef.oclIsKindOf(BPMN!Task) and
51     s.targetRef.oclIsKindOf(BPMN!Task)
52   )
53   to
54     t: UML!"uml::Dependency" (
55       name <- '<>precedes>>'
56       , client <- UML!UseCase.allInstances() -> select(
57         u | u.name=s.sourceRef.name) -> first()
58       , supplier <- UML!UseCase.allInstances() -> select(
59         u | u.name=s.targetRef.name) -> first()
60     )
61 }

```

Hình 3.18. Hàm chuyển đổi RA2 - SequenceFlow2Dependency.

### Luật RA3: Ánh xạ ExclusiveGateway sang quan hệ Extend [9, 22]

Trong BPMN, một Exclusive Gateway liên kết các Task và việc thực hiện các Task đích của Gateway là không bắt buộc và phụ thuộc vào điều kiện của Gateway. Khái niệm này tương đồng với quan hệ <<extend>> trong mô hình ca sử dụng. Use Case ban đầu được mở rộng và thêm chức năng cho hệ thống, còn Use Case mở rộng phụ thuộc vào Use Case ban đầu, thông thường là không bắt buộc và được sử dụng có điều kiện. Biểu diễn cụ thể của luật chuyển đổi này được thể hiện trong Hình 3.19.



Hình 3.19. Luật RA3 - Ánh xạ ExclusiveGateway sang quan hệ Extend.

Luồng rẽ nhánh qua ExclusiveGateway được chuyển đổi thành các quan hệ Extend giữa các UseCase. Cụ thể, UseCase nguồn tương ứng với Task nguồn sẽ có thuộc tính ExtensionPoint, còn các UseCase đích tương ứng với các Task đích sẽ có thuộc tính Extend, trỏ extendedCase tới UseCase nguồn và extensionLocation tới ExtensionPoint của UseCase nguồn.

```

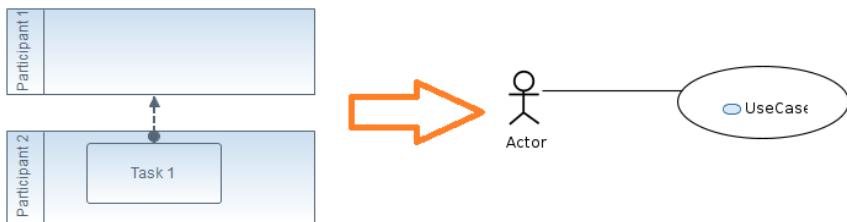
63 unique lazy rule ExtensionPoint {
64   from s: BPMN!SequenceFlow
65   to t: UML!ExtensionPoint (
66     name <- s.id
67   )
68 }
69 unique lazy rule Extend {
70   from s: BPMN!SequenceFlow
71   to t: UML!Extend (
72     extendedCase <- thisModule.getUsecase(
73       s.sourceRef.incoming->first().sourceRef.name)
74     ,extensionLocation <- UML!ExtensionPoint.allInstances()->select(
75       e|e.name=s.id)->first()
76   )
77 }

168 endpoint rule Gateway2Extend() {
169   do {
170     for (gw in BPMN!ExclusiveGateway.allInstances()) {
171       gw.id.debug();
172       if (gw.incoming.size() = 1) {
173         if (gw.incoming->first().sourceRef.oclIsKindOf(BPMN!Task)) {
174           for (out in gw.outgoing) {
175             if (out.targetRef.oclIsKindOf(BPMN!Task)) {
176               thisModule.getUsecase(gw.incoming->first()
177                 .sourceRef.name).extensionPoint
178                 <- thisModule.ExtensionPoint(out);
179               thisModule.getUsecase(out.targetRef.name)
180                 .extend <- thisModule.Extend(out);
181             }
182           }
183         }
184       }
185     }
186   }
187 }
```

Hình 3.20. Hàm chuyển đổi RA3 - Gateway2Extend.

### **Luật RA4: Ánh xạ MessageFlow sang Association [7, 9]**

Một Actor - đại diện cho một Participant bên ngoài - có quan hệ Association với Use Case - đại diện cho Task mà Participant trao đổi thông điệp.



Hình 3.21. Luật RA4 - Ánh xạ MessageFlow sang Association.

Siêu khái niệm MessageFlow từ Participant đến Task ánh xạ đến Association giữa Actor và UseCase tương ứng được tạo ra. Tương tự như cách cài đặt luật RA1, Association có các thuộc tính ownedEnd để trỏ tới Actor và UseCase, với thuộc tính name và type được gán tương ứng cho đối tượng này.

```

134 rule MessageFlow2Association1 {
135     from s: BPMN!MessageFlow (
136         s.sourceRef.oclIsKindOf(BPMN!Task)
137             and s.targetRef.oclIsKindOf(BPMN!Participant))
138     )
139     to t:UML!Association (
140         ownedEnd <- Set {endUsecase, endActor}
141     )
142     , endUsecase: UML!Property (
143         name <- s.sourceRef.name
144         , type <- thisModule.getUsecase(s.sourceRef.name)
145     )
146     , endActor : UML!Property (
147         name <- s.targetRef.name
148         , type <- thisModule.getActor(s.targetRef.name)
149     )
150 }
151 rule MessageFlow2Association2 {
152     from s: BPMN!MessageFlow (
153         s.sourceRef.oclIsKindOf(BPMN!Participant)
154             and s.targetRef.oclIsKindOf(BPMN!Task))
155     )
156     to t:UML!Association (
157         ownedEnd <- Set {endUsecase, endActor}
158     )
159     , endUsecase: UML!Property (
160         name <- s.targetRef.name
161         , type <- thisModule.getUsecase(s.targetRef.name)
162     )
163     , endActor : UML!Property (
164         name <- s.sourceRef.name
165         , type <- thisModule.getActor(s.sourceRef.name)
166     )
167 }

```

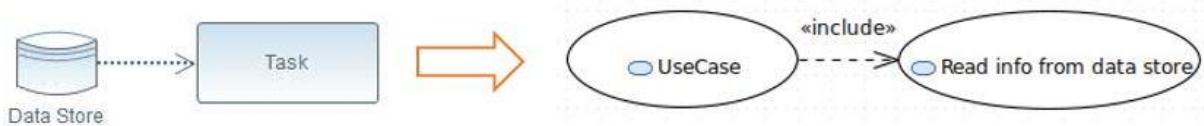
Hình 3.22. Hàm chuyển đổi RA4 - MessageFlow2Association.

### 3.5.3. Luật chuyển đổi cho các quan hệ với dữ liệu (Rules for Data association)

Quan hệ <<include>> trong biểu đồ ca sử dụng cho thấy hành vi của Use Case được bao gồm là bắt buộc và là một phần của Use Case cơ bản. Bên cạnh đó, Use Case cơ bản không thể hoàn thành nếu không có Use Case bao gồm. Hành vi này tương tự với trường hợp các Task của quy trình nghiệp vụ kết nối với các đối tượng dữ liệu, bao gồm các hành động: đọc/ghi thông tin Data Store và gửi/nhận thông tin Data object. Cách chuyển đổi tương ứng với các hành động này được thể hiện trong bốn luật sau đây.

**Luật RD1:** Ánh xạ đọc Data Store sang quan hệ include ReadData

Việc một Task đọc thông tin từ Data Store có thể chuyển thành quan hệ <<include>> giữa hai Use Case như Hình 3.23.



Hình 3.23. Luật RD1 - Ánh xạ đọc Data Store sang quan hệ include ReadData.

Với quan hệ Task đọc dữ liệu từ Data Store, chương trình sẽ tạo mới một UseCase có thuộc tính name là ‘Read information from ‘+ Data Store name. UseCase sinh tương ứng với Task sẽ có thuộc tính include và trỏ tới UseCase ReadData mới được tạo ra.

```

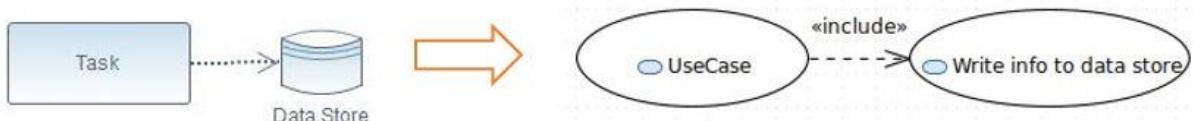
128 unique lazy rule Include {
129     from s: BPMN!Task
130     to t: UML!Include (
131         addition <- thisModule.getUsecase(s.name)
132     )
133 }

92 rule DataStore2ReadData{
93     from s: BPMN!DataInputAssociation (
94         s.sourceRef->first().oclIsTypeOf(BPMN!DataStoreReference) )
95     to t: UML!UseCase (
96         name <- 'Read information from ' + s.sourceRef->first().name
97     )
98     do {
99         if (s.refImmediateComposite().oclIsKindOf(BPMN!Task) )
100             thisModule.getUsecase(s.refImmediateComposite().name)
101             .include <- thisModule.Include(t);
102     }
103 }
  
```

Hình 3.24. Hàm chuyển đổi RD1 - DataStore2ReadData.

**Luật RD2:** Ánh xạ ghi Data Store sang quan hệ include WriteData

Việc một Task ghi thông tin vào Data Store có thể chuyển thành quan hệ <<include>> giữa hai Use Case như Hình 3.25.



Hình 3.25. Luật RD2 - Ánh xạ ghi Data Store sang quan hệ include WriteData.

Với quan hệ Task ghi dữ liệu vào Data Store, chương trình sẽ tạo mới một UseCase có thuộc tính name là ‘Write information to ‘+ Data Store name. UseCase sinh tương ứng với Task sẽ có thuộc tính include và trỏ tới UseCase WriteData mới được tạo ra.

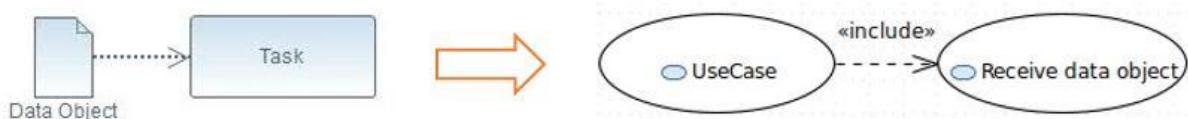
```

79 rule DataStore2WriteData {
80     from s: BPMN!DataOutputAssociation (
81         s.targetRefoclIsTypeOf(BPMN!DataStoreReference))
82     to t: UML!UseCase (
83         name <- 'Write information to ' + s.targetRef.name
84     )
85     do {
86         if (s.refImmediateComposite().oclIsKindOf(BPMN!Task))
87             thisModule.getUsecase(s.refImmediateComposite().name)
88             .include <- thisModule.Include(t);
89     }
90 }
91

```

Hình 3.26. Hàm chuyển đổi RD2 – DataStore2WriteData.

**Luật RD3:** Ánh xạ nhận Data Object sang quan hệ include ReceiveData  
Việc một Task nhận thông tin Data Object có thể chuyển thành quan hệ <<include>> giữa hai Use Case như Hình 3.27.



Hình 3.27. Luật RD3 - Ánh xạ nhận Data Object sang quan hệ include ReceiveData.

Với quan hệ Task nhận dữ liệu Data Object, chương trình sẽ tạo mới một UseCase có thuộc tính name là ‘Receive ’+ Data Object name. UseCase sinh tương ứng với Task sẽ có thuộc tính include và trỏ tới UseCase ReceiveData mới được tạo ra.

```

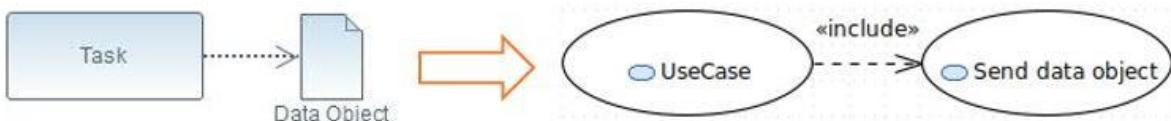
116 rule DataObject2ReceiveData {
117   from s: BPMN!DataInputAssociation (
118     s.sourceRef->first().oclIsTypeOf(BPMN!DataObject))
119   to t: UML!UseCase (
120     name <- 'Receive ' + s.sourceRef->first().name
121   )
122   do {
123     if (s.refImmediateComposite().oclIsKindOf(BPMN!Task))
124       thisModule.getUsecase(s.refImmediateComposite().name)
125         .include <- thisModule.Include(t);
126   }
127 }

```

Hình 3.28. Hàm chuyển đổi RD3 - DataObject2ReceiveData.

**Luật RD4:** Ánh xạ gửi Data Object sang quan hệ include SendData

Việc một Task gửi thông tin Data Object có thể chuyển thành quan hệ <<include>> giữa hai Use Case như Hình 3.29.



Hình 3.29. Luật RD4 - Ánh xạ gửi Data Object sang quan hệ include SendData.

Với quan hệ Task gửi dữ liệu Data Object, chương trình sẽ tạo mới một UseCase có thuộc tính name là 'Send ' + Data Object name. UseCase sinh tương ứng với Task sẽ có thuộc tính include và trỏ tới UseCase SendData mới được tạo ra.

```

104 rule DataObject2SendData{
105   from s: BPMN!DataOutputAssociation (
106     s.targetRef.oclIsTypeOf(BPMN!DataObject))
107   to t: UML!UseCase (
108     name <- 'Send ' + s.targetRef.name
109   )
110   do {
111     if (s.refImmediateComposite().oclIsKindOf(BPMN!Task))
112       thisModule.getUsecase(s.refImmediateComposite().name)
113         .include <- thisModule.Include(t);
114   }
115 }

```

Hình 3.30. Hàm chuyển đổi RD4 – DataObject2SendData.

Một cách tổng quát, phương pháp đã trình bày một bộ quy tắc gồm 10 luật chuyển đổi để thực hiện quá trình sinh ca sử dụng từ quy trình nghiệp vụ. Trong đó, các luật chuyển đổi được đề xuất dựa trên các nghiên cứu đã có là RE1, RE2, RA1, RA3, RA4 [7, 9, 22]. Đây là những phần tử đối tượng và quan hệ cơ bản nhất trong mô hình và lý thuyết chuyển đổi là tương tự nhau giữa các nghiên cứu và phương pháp đề xuất BPMN2UseCase. Bên cạnh đó, để có thể truyền tải được nhiều thông tin hơn từ

mô hình quy trình nghiệp vụ, luận văn đã tìm hiểu thêm các luật RA2, RD1, RD2, RD3, RD4. Từ đó, quan hệ giữa các đối tượng với nhau và với dữ liệu có thể được bổ sung vào ca sử dụng, giúp cho mô hình đầu ra phong phú và giàu thông tin hơn.

### 3.6. Thuật toán và thực thi chuyển đổi mô hình

Dựa trên bộ quy tắc chuyển đổi đối tượng trên, luận văn đã xây dựng thuật toán BPMN2UseCase thực hiện việc chuyển đổi mô hình thể hiện ở Hình 3.31. Các yếu tố chính của thuật toán được trình bày cụ thể như sau:

- **Đầu vào:**

Chương trình BPMN2UseCase sử dụng đầu vào là một mô hình quy trình nghiệp vụ BPMN thể hiện dưới dạng cú pháp trùu tượng XMI để có thể phân tích và bóc tách các đối tượng cần thiết cho chuyển đổi.

- **Đầu ra:**

Sau khi thực hiện thuật toán, chương trình sẽ sinh ra một tệp XMI thể hiện nội dung mô hình ca sử dụng. Với nội dung mới này, sử dụng các công cụ hỗ trợ có thể dễ dàng tạo ra dạng đồ họa của biểu đồ ca sử dụng một cách thuận tiện và trực quan.

- **Luồng thực hiện**

Thứ tự thực hiện các hàm chuyển đổi được sắp xếp lần lượt như sau:

- Chuyển đổi các đối tượng cơ bản của mô hình (Use Case và Actor) tương ứng với các bước 1-2. Đây là các bước đầu tiên và bắt buộc cần thực hiện.
- Chuyển đổi các đối tượng quan hệ trong mô hình (Association, Flow, ...) tương ứng với các bước 3-10. Các bước này là không bắt buộc, nếu trong mô hình có các đối tượng đầu vào tương ứng thì sẽ được thực hiện, nếu không sẽ được bỏ qua.

Algorithm: BPMN2UseCase

Input: BPMN2UseCase

Output: UseCase Model

Begin

1. Map Task to UseCase (**RE1**)  
UseCase.name = Task.name
2. Map Participant to Actor (**RE2**)  
Actor.name = Participant.name
3. Map relationship between Task and Participant to Association between UseCase to Actor (**RA1**)  
Association.ownedEnd = Set {UseCase, Actor}

4. Map Sequence Flow between Task to Dependency between UseCase (**RA2**)  
 Dependency.name = <<precedes>>  
 Dependency.client = sourceUseCase  
 Dependency.supplier = targetUseCase
5. Map ExclusiveGateway to Extend (**RA3**)
  - 5.1. Rule ExtensionPoint (SequenceFlow)  
 ExtensionPoint.name = SequenceFlow.id
  - 5.2. Rule Extend  
 Extend.extendedCase = sourceUseCase  
 Extend.extensionLocation = sourceExtensionPoint
  - 5.3. ExclusiveGateway2Extend  
 sourceUseCase.extensionPoint = ExtensionPoint(sourceSequenceFlow)  
 targetUseCase.extend = Extend ()
6. Map MessageFlow between Participant and Task to Association between Actor and UseCase (**RA4**)  
 Association.ownedEnd = Set {UseCase, Actor}
7. Map relationship from Data Store to Task to Include association ReadData (**RD1**)  
 Include.addition = UseCase  
 TargetUseCase.name = "Read information from " + Data Store.name  
 SourceUseCase.include = Include(TargetUseCase)
8. Map relationship from Task to Data Store to Include association WriteData (**RD2**)  
 Include.addition = UseCase  
 TargetUseCase.name = "Write information to " + Data Store.name  
 SourceUseCase.include = Include(TargetUseCase)
9. Map relationship from Data Object to Task to Include association ReceiveData (**RD3**)  
 Include.addition = UseCase  
 TargetUseCase.name = "Receive " + Data Object.name  
 SourceUseCase.include = Include(TargetUseCase)
10. Map relationship from Task to Data Object to Include association SendData (**RD4**)  
 Include.addition = UseCase  
 TargetUseCase.name = "Send " + Data Object.name  
 SourceUseCase.include = Include(TargetUseCase)

End

*Hình 3.31. Mô hình ca sử dụng.*

Để đánh giá hiệu quả của thuật toán trong việc giải quyết chuyển đổi mô hình, ta căn cứ vào hai tiêu chuẩn dưới đây:

- ***Tính đúng đắn của thuật toán***

Về mặt lý thuyết, thuật toán luôn đảm bảo tạo ra được mô hình đầu ra theo yêu cầu của bài toán. Một mô hình quy trình nghiệp vụ dù đơn giản nhất cũng luôn có các phần tử cơ bản bao gồm Task và Participant, do đó các bước 1-2 luôn đảm bảo được thực hiện để sinh ra mô hình ca sử dụng với Use Case và Actor tương ứng. Để kiểm tra và xác định tính đúng đắn trong thực hành, chương trình sẽ được cài đặt và chạy thực nghiệm với một số bộ dữ liệu mẫu, lấy kết quả thu được so sánh với kết quả từ phương pháp thủ công. Chi tiết của việc đánh giá thực nghiệm sẽ được trình bày trong chương sau.

- ***Độ phức tạp của thuật toán***

Giả thiết mô hình đầu vào có n phần tử. Thuật toán sẽ duyệt lần lượt và kiểm tra từng phần tử, nếu thoả mãn yêu cầu của bất kỳ một quy tắc chuyển đổi nào thì phần tử đó sẽ được tiến hành chuyển đổi. Do đó, độ phức tạp ở đây có thể hiểu là  $O(n)$ . Như vậy, xét trong thực tế đối với các hệ thống có quy mô lớn và quy trình nghiệp vụ phức tạp (n lớn), thuật toán hoàn toàn có thể đáp ứng tốt về thời gian và tốc độ xử lý.

### 3.7. Tổng kết chương

Trong chương này, luận văn đã trình bày một quy trình hoàn chỉnh sinh tự động mô hình ca sử dụng từ mô hình quy trình nghiệp vụ BPMN. Áp dụng việc phân tích và đánh giá các định nghĩa và khái niệm của hai siêu mô hình, luận văn đã xây dựng được các luật chuyển đổi tương ứng và đề xuất một thuật toán thực hiện quá trình chuyển đổi. Từ đó, sử dụng các kỹ thuật biểu diễn mô hình UML và tạo ra được biểu đồ ca sử dụng theo mong muốn của người dùng.

## CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Trong chương này, luận văn sẽ trình bày cụ thể chương trình sinh tự động ca sử dụng dựa trên chuyển đổi mô hình ATL cũng như các bộ công cụ và kỹ thuật biểu diễn mô hình cần thiết để sử dụng cho phương pháp đã đề xuất. Sau đó, áp dụng quy trình trên vào nghiên cứu tình huống cụ thể. Luận văn cũng xây dựng bộ test-cases để chạy và kiểm tra tính đúng đắn của chương trình. Từ đó, luận văn có thể đánh giá và nhận xét về hiệu quả cũng như ưu nhược điểm của giải pháp.

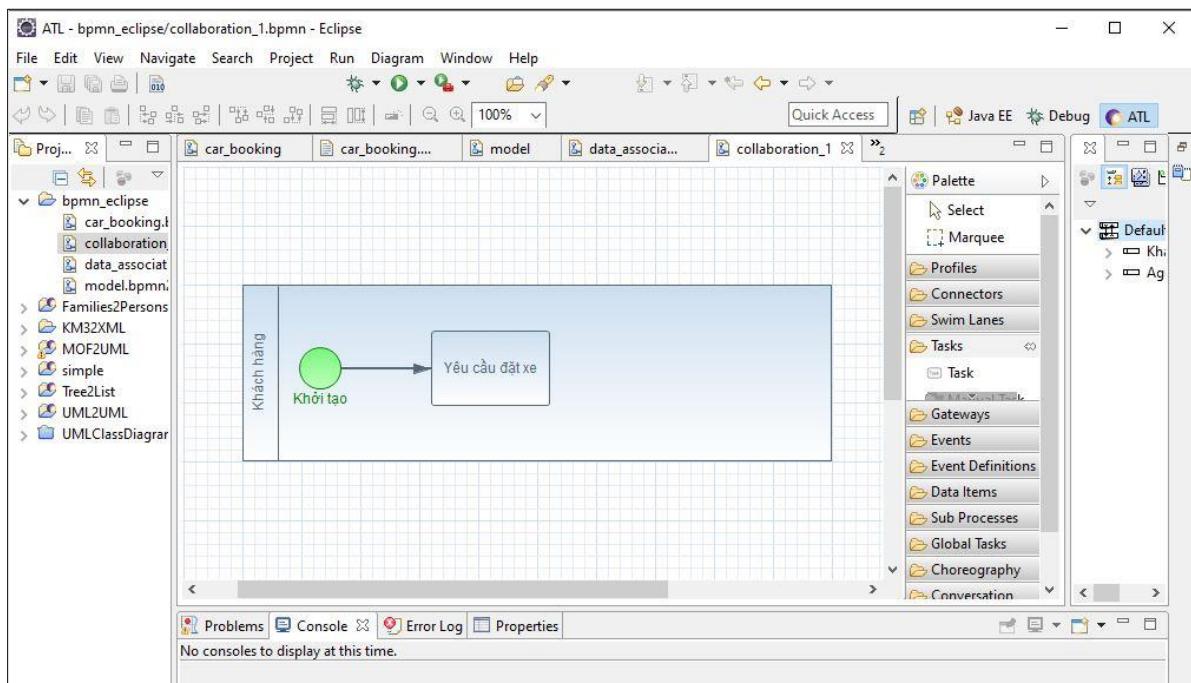
### 4.1. Công cụ, ngôn ngữ và môi trường hỗ trợ

Tương ứng với các bước trong quy trình tự động được thể hiện trong Hình 3.3, luận văn đã sử dụng các công cụ/môi trường để cài đặt chương trình như sau.

#### 4.1.1. Kỹ thuật biểu diễn quy trình nghiệp vụ

BPMN là một tiêu chuẩn đã được sử dụng rộng rãi và hiện nay đã được hỗ trợ bởi rất nhiều công cụ mô hình quy trình nghiệp vụ phổ biến như *Eclipse BPMN2 Modeler*, *Bizagi modeller* hay *Enterprise Architect*. Trong luận văn này, công cụ *Eclipse BPMN2 Modeler* đã được lựa chọn sử dụng vì các lý do sau:

- Công cụ miễn phí cung cấp đầy đủ các tính năng cần thiết, có thể cài đặt plugin đơn giản trong bộ công cụ Eclipse.
- Thiết kế biểu đồ BPMN một cách thuận tiện với đồ họa đơn giản dễ nhìn và kéo thả các thành phần dễ dàng.
- Lưu mô hình dưới định dạng bpmn chuẩn và hỗ trợ xuất ra file ảnh (.png, \*.jpg, ...) cho biểu đồ.

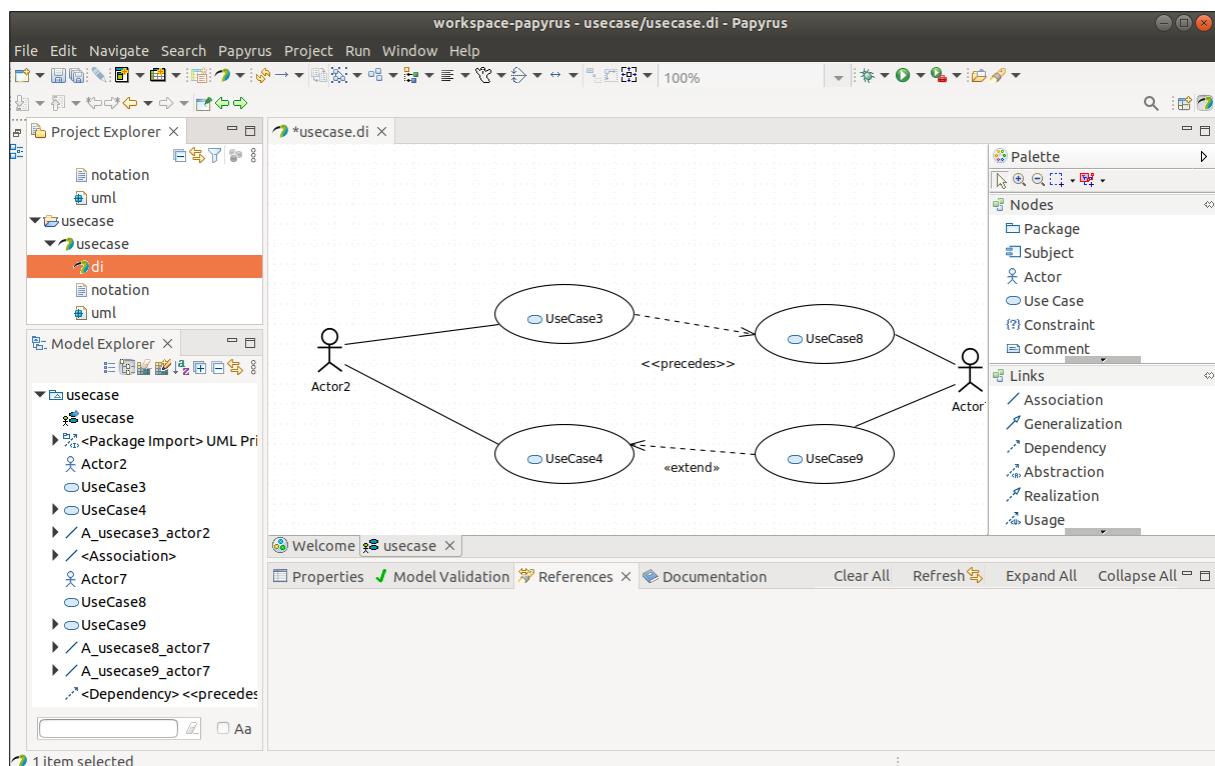


Hình 4.1. Giao diện công cụ *Eclipse BPMN2 Modeler*.

### 4.1.2. Kỹ thuật biểu diễn biểu đồ ca sử dụng

Sau quá trình chuyển đổi mô hình trên, đầu ra thu được là một file XMI chứa đầy đủ nội dung thể hiện mô hình ca sử dụng. Để biểu diễn hình ảnh biểu đồ ca sử dụng này một cách trực quan, có nhiều công cụ có thể sử dụng bao gồm *Eclipse Papyrus*, *UML Designer*, *Sirius*, ... Để thống nhất môi trường sử dụng cho toàn bộ giải pháp, luận văn đã sử dụng *Eclipse Papyrus* với các ưu điểm như sau:

- Công cụ mã nguồn mở cho kỹ thuật dựa trên mô hình (Model-Based Engineering), cài đặt và sử dụng dễ dàng với bộ cài riêng hoặc plugin trên bộ công cụ Eclipse.
- Cho phép biểu diễn biểu đồ UML dễ dàng từ các tệp XMI chuẩn hóa.

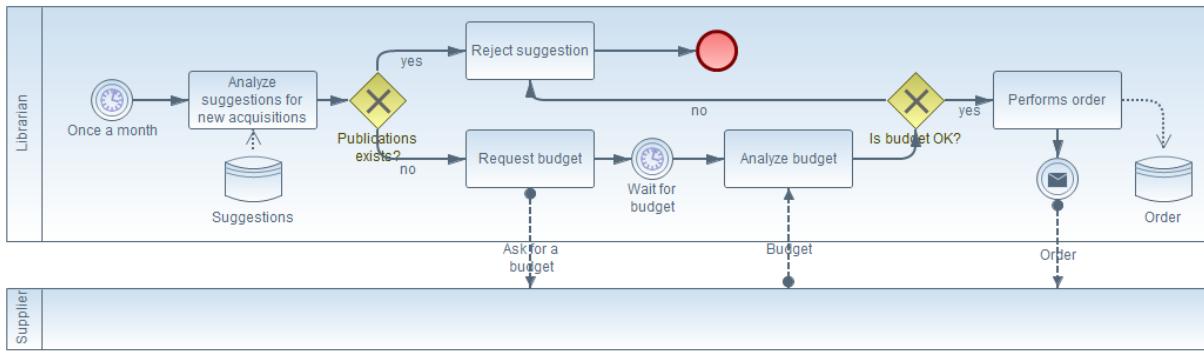


Hình 4.2. Giao diện công cụ *Eclipse Papyrus*.

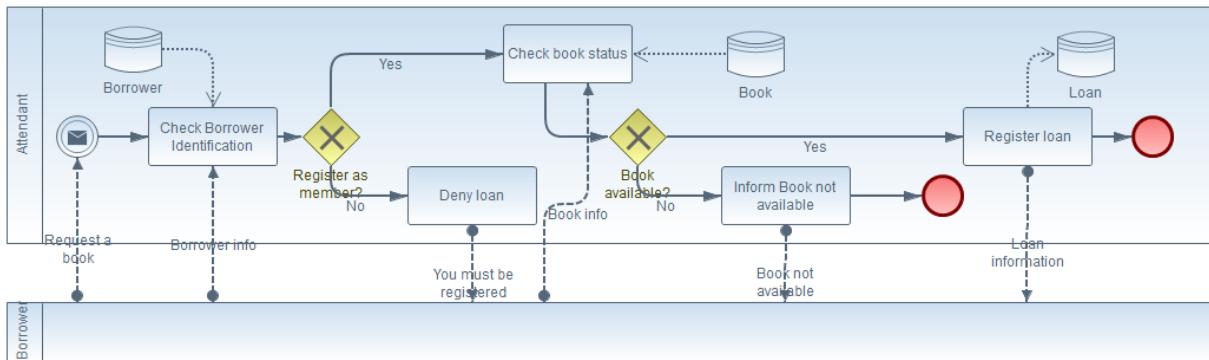
### 4.2. Nghiên cứu tình huống thực nghiệm

#### 4.2.1. School Library System

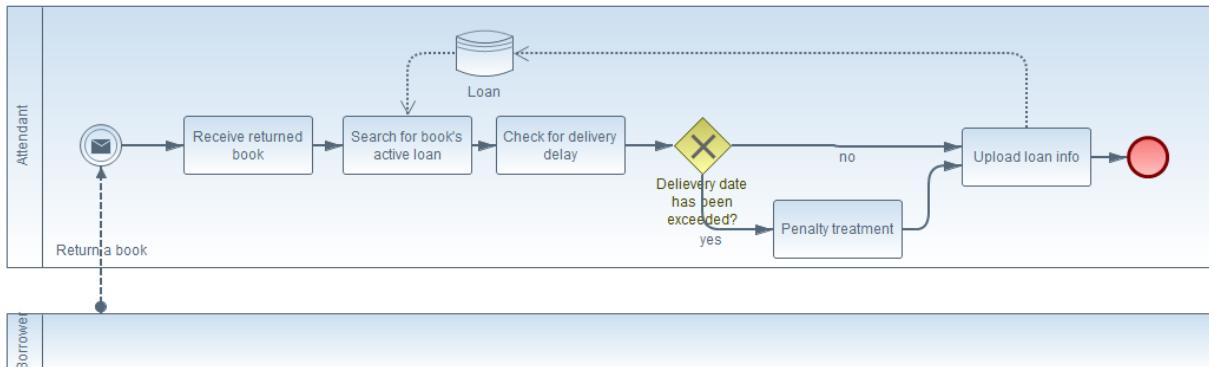
Trong phần này, luận văn sẽ nghiên cứu và áp dụng phương pháp BPMN2UseCase vào mô hình quy trình nghiệp vụ School Library System được đề xuất và thử nghiệm trong [10]. Mô hình này được chia thành ba mô hình nhỏ: Purchase book, Lend book và Return book, được thể hiện trong các Hình 4.3, 4.4 và 4.5. Sau đó, từ kết quả mô hình ca sử dụng sinh ra có thể so sánh và đánh giá hiệu quả của cả hai phương pháp.



Hình 4.3. Mô hình BPMN Purchase book [10].



Hình 4.4. Mô hình BPMN Lend book [10].

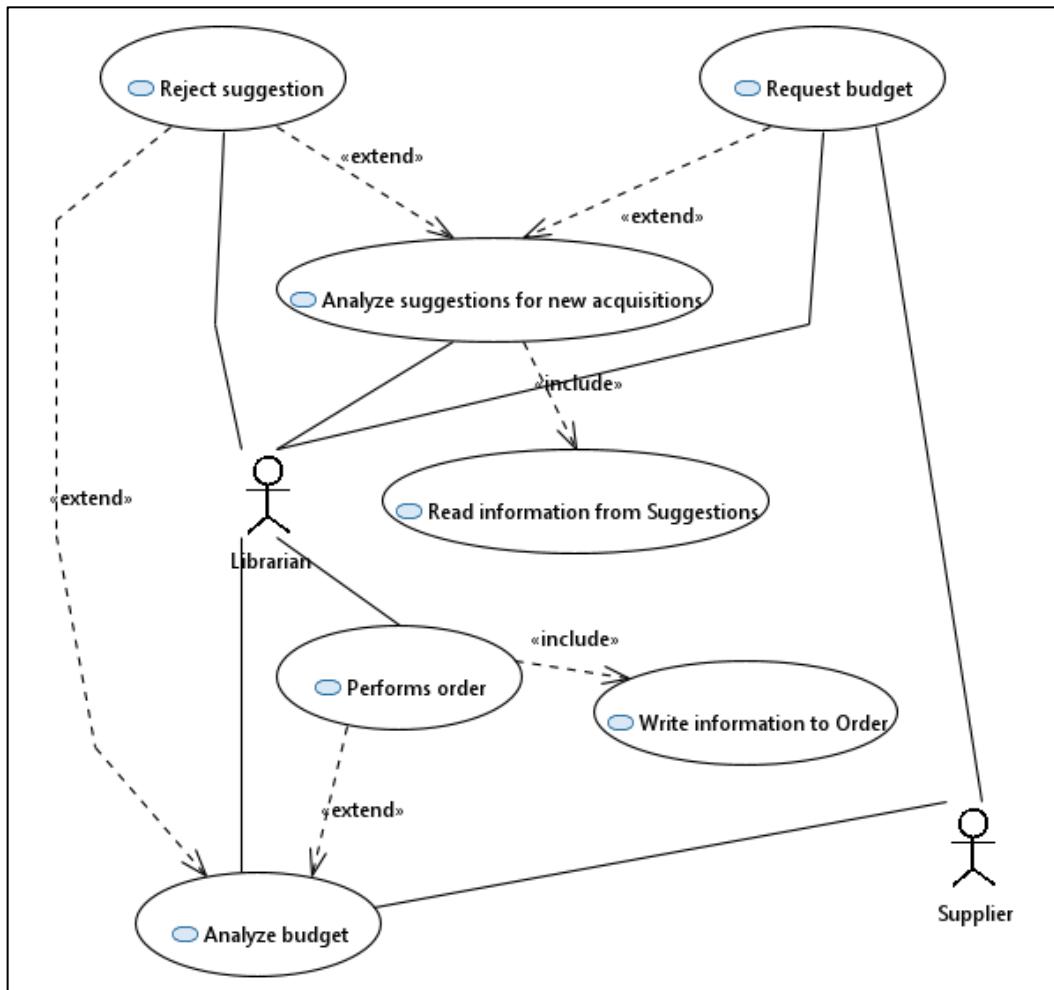


Hình 4.5. Mô hình BPMN Return book [10].

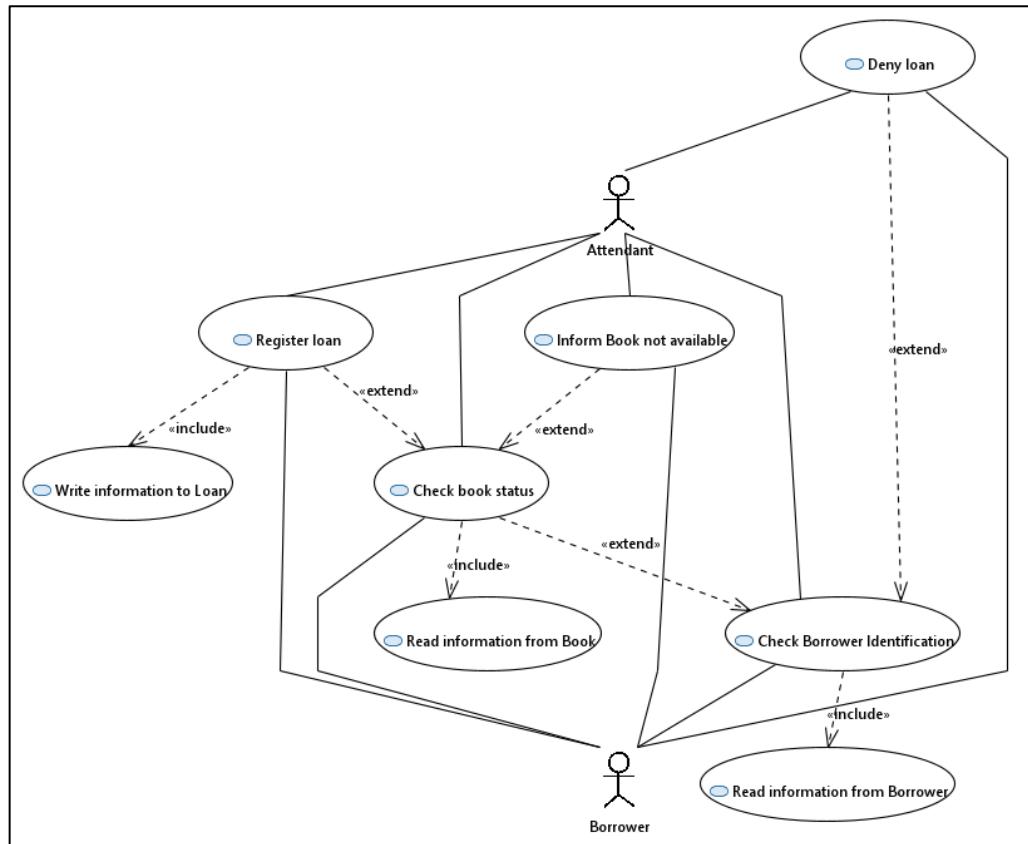
Áp dụng các luật chuyên đổi BPMN2UseCase được đề xuất ở trên, ta có thể xây dựng được biểu đồ ca sử dụng như Hình 4.6, 4.7 và 4.8.

- Luật RE1: 15 UseCase sinh ra từ 15 Task tương ứng (Analyze suggestions for new acquisitions, Reject suggestion, Request budget,...)
- Luật RE2: 4 Actor sinh ra từ 4 Participant tương ứng (Librarian, Supplier, Attendant, Borrower)
- Luật RA1: Liên kết Association tương ứng giữa 4 Actor với các UseCase tương ứng sinh ra từ quan hệ giữa 4 Participant và các Task của từng Participant (Librarian và Analyze suggestions for new acquisitions, Attendant và Check borrower identification, ...)
- Luật RA2: 3 Dependency <<precedes>> sinh ra từ 3 SequenceFlow giữa 2 Task (Receive returned book đến Search for book's active loan, Penalty treatment đến Upload Loan info, ...)

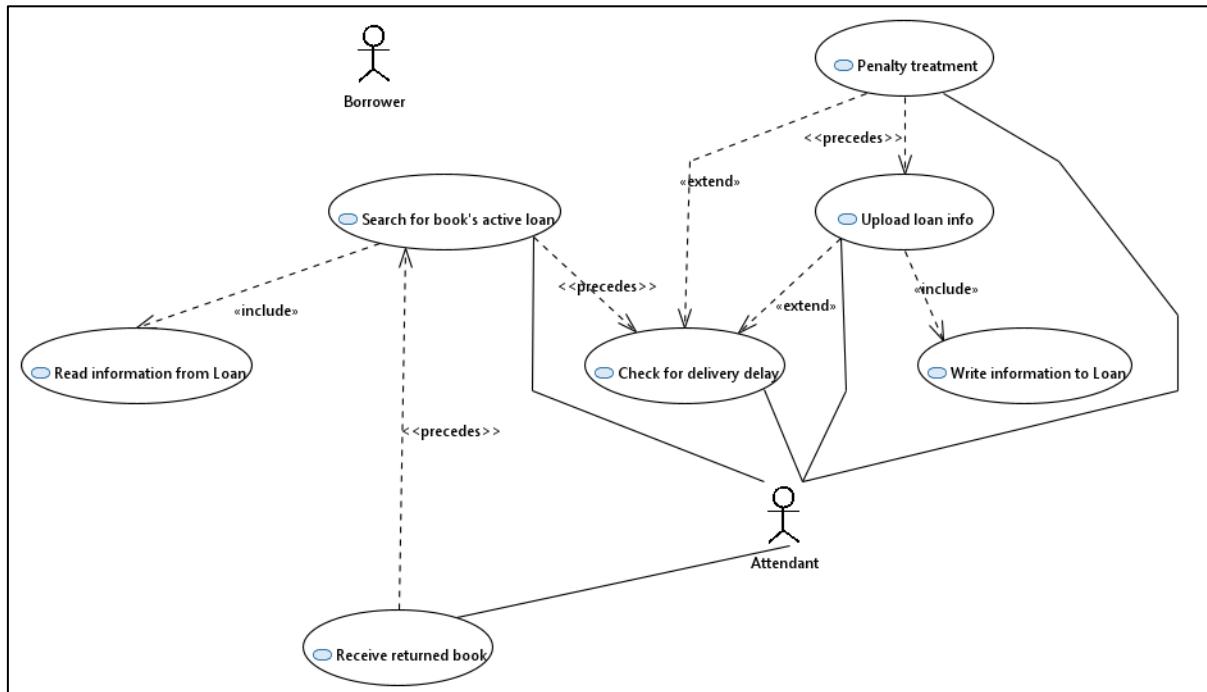
- Luật RA3: 8 Extend association giữa các UseCase sinh ra từ 4 ExclusiveGateway (Check book status đến Register loan và đến Inform Book not available, ...)
- Luật RA4: 7 Association sinh ra từ 7 MessageFlow giữa các Participant và Task (Request budget đến Supplier, Borrower đến Check book status, ...)
- Luật RD1: 5 quan hệ Include ReadData sinh ra từ 5 quan hệ đọc dữ liệu từ Data Store (Suggestions đến Analyze suggestions for new acquisitions, Book đến Check book status, ...)
- Luật RD2: 3 quan hệ Include WriteData sinh ra từ 3 quan hệ ghi dữ liệu vào Data Store (Perform order ghi vào Order, Register loan ghi vào Loan...)



Hình 4.6. Mô hình ca sử dụng Purchase book.



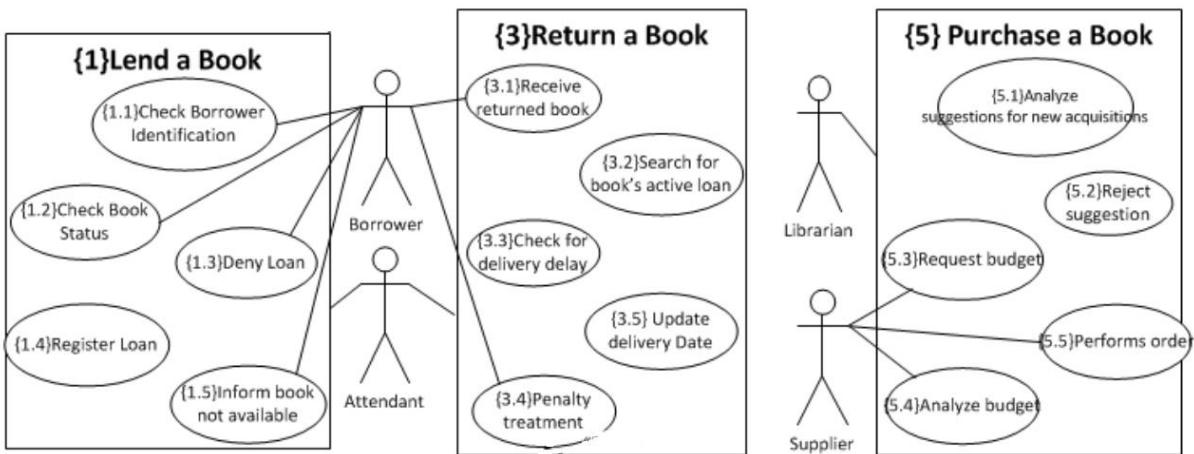
Hình 4.7. Mô hình ca sử dụng Lend book.



Hình 4.8. Mô hình ca sử dụng Return book.

Theo kết quả thu được, biểu đồ bao gồm 4 Actor, 22 UseCase, 22 Association giữa Actor và UseCase, 20 Association giữa các UseCase thuộc 3 loại (include, extend, precedes).

Trong khi đó, biểu đồ ca sử dụng sinh ra theo phương pháp [10] tương đối đơn giản, bao gồm 4 Actor, 15 UseCase, 10 Association giữa Actor và UseCase và 0 Association giữa các UseCase (Hình 4.9).



Hình 4.9. Mô hình ca sử dụng School Library System [10].

#### 4.2.2. Phương pháp đánh giá kết quả

Để kiểm tra và đánh giá hiệu quả cũng như tính chính xác của phương pháp, luận văn đã đề xuất một bộ gồm 14 test-cases để thử nghiệm, bao gồm 3 test-cases School Library System được trình bày ở Mục 4.2.1. Các test-cases là những ví dụ về mô hình quy trình nghiệp vụ được sưu tầm online [28], chi tiết (đầu vào và đầu ra) sẽ được thể hiện trong phần Phụ lục. Sau quá trình chạy thực nghiệm, kết quả sẽ được sử dụng để đánh giá phương pháp dựa trên ba tiêu chí: tính ổn định, tính đúng đắn và tính đầy đủ.

##### 4.2.2.1. Đánh giá tính ổn định

Trong mục này, luận văn sẽ chạy và kiểm tra chương trình có hoạt động ổn định, đảm bảo ra được kết quả và đúng định dạng hay không. Bên cạnh đó, một tiêu chí đánh giá quan trọng nhất đó là chương trình cài đặt có đảm bảo thực hiện và truyền tải đúng và đủ các quy tắc chuyển đổi đã được đề xuất trong Mục 3.5 hay không. Để thực hiện việc này, hai mô hình ca sử dụng sẽ được tạo ra bởi hai phương thức: BPMN2UseCase và cách thủ công. Bất kỳ sai lệch nào giữa hai kết quả đầu ra này sẽ thể hiện được tỉ lệ sai sót trong quá trình cài đặt chương trình. Chương trình được chạy với 14 ca kiểm thử và đánh giá dựa trên 4 mục:

- Chương trình có lỗi khi đang chạy hay không?
- Sau khi chạy xong, chương trình có tạo ra được kết quả không?
- Kết quả sinh ra có đúng định dạng cho trước không?
- Tỉ lệ sai sót khi truyền tải bộ luật chuyển đổi như thế nào?

Kết quả thu được cho thấy phương pháp chạy tương đối ổn định, không có lỗi phát sinh và đảm bảo được đầu ra tuân theo đúng bộ quy tắc chuyển đổi mô hình.

#### 4.2.2.2. Đánh giá tính đúng đắn

Đối với các yếu tố thành phần cơ bản (UseCase, Actor và Association), phương thức đánh giá đó là thực hiện chuyển đổi mô hình quy trình nghiệp vụ sang mô hình ca sử dụng bởi hai phương pháp: thủ công và tự động sử dụng BPMN2UseCase. Từ đó có thể xác định tính đúng đắn của phương pháp đề xuất trong trường hợp xét riêng các thành phần cơ bản này.

Bảng 4.1. Kết quả đánh giá tính đúng đắn

	Phương pháp thủ công	Phương pháp BPMN2 UseCase	Tỉ lệ đúng
Auction Service			
UseCase from Task	5	5	100%
Actor from Participant	2	2	100%
Association from Participant-Task	5	5	100%
Book Selling			
UseCase from Task	8	8	100%
Actor from Participant	5	2	40%
Association from Participant-Task	8	8	100%
Car booking			
UseCase from Task	8	8	100%
Actor from Participant	3	3	100%
Association from Participant-Task	8	8	100%
Employment application			
UseCase from Task	12	12	100%
Actor from Participant	2	2	100%
Association from Participant-Task	12	12	100%
Incident management			
UseCase from Task	12	12	100%
Actor from Participant	5	5	100%
Association from Participant-Task	12	12	100%
IT help			
UseCase from Task	10	10	100%

<b>Actor from Participant</b>	3	3	100%
<b>Association from Participant-Task</b>	10	10	100%
<b>Online shopping</b>			
<b>UseCase from Task</b>	4	4	100%
<b>Actor from Participant</b>	2	2	100%
<b>Association from Participant-Task</b>	4	4	100%
<b>Payment process</b>			
<b>UseCase from Task</b>	7	7	100%
<b>Actor from Participant</b>	4	2	50%
<b>Association from Participant-Task</b>	7	7	100%
<b>Pizza store</b>			
<b>UseCase from Task</b>	8	8	100%
<b>Actor from Participant</b>	5	2	40%
<b>Association from Participant-Task</b>	8	8	100%
<b>Shopping process</b>			
<b>UseCase from Task</b>	8	8	100%
<b>Actor from Participant</b>	2	2	100%
<b>Association from Participant-Task</b>	8	8	100%
<b>Nobel Prize</b>			
<b>UseCase from Task</b>	8	8	100%
<b>Actor from Participant</b>	2	2	100%
<b>Association from Participant-Task</b>	8	8	100%
<b>Purchase book</b>			
<b>UseCase from Task</b>	8	8	100%
<b>Actor from Participant</b>	2	2	100%
<b>Association from Participant-Task</b>	8	8	100%
<b>Lend book</b>			
<b>UseCase from Task</b>	8	8	100%
<b>Actor from Participant</b>	2	2	100%
<b>Association from Participant-Task</b>	8	8	100%

Return book			
UseCase from Task	8	8	100%
Actor from Participant	2	2	100%
Association from Participant-Task	8	8	100%
<b>Average</b>	<b>95.9%</b>		

Trong phần lớn các trường hợp, phương pháp BPMN2UseCase đã sinh được đầu ra trùng khớp 100% với cách thực hiện thủ công. Tuy nhiên, sai khác được phát hiện ở các test-cases Book selling, Payment process và Pizza store, ở phần số lượng Actor được tạo ra. Sau khi xem xét lại mô hình BPMN đầu vào, ta nhận thấy với trường hợp một Participant bao gồm nhiều Lane bên trong, sau quá trình chuyển đổi BPMN2UseCase sẽ chỉ sinh ra 1 Actor tương ứng với Participant chính. Trong khi đó, phương pháp thủ công sẽ tạo ra thêm các Actor tương ứng với từng Lane, dẫn đến số lượng Actor tổng cộng sẽ tăng lên. Để khắc phục được lỗi này, phương pháp cần nghiên cứu tìm hiểu thêm về Lane và xem xét khả năng chuyển đổi trong hướng phát triển tiếp theo.

#### 4.2.2.3. Đánh giá tính đầy đủ

Trong phần này, luận văn sẽ xem xét khả năng bao trùm của phương pháp BPMN2UseCase đối với các loại ký hiệu được sử dụng trong mô hình quy trình nghiệp vụ.

Bảng 4.2. Kết quả đánh giá tính đầy đủ

Mô hình BPMN	Số ký hiệu được chuyển đổi	Số ký hiệu không được chuyển đổi	Loại ký hiệu không được chuyển đổi
Auction Service	5	2	SequenceFlow giữa 2 Gateway SequenceFlow giữa Gateway và Event
Book Selling	5	0	
Car booking	5	0	
Employment application	7	0	
Incident management	6	1	SequenceFlow giữa 2 Gateway
IT help	4	0	
Online	4	1	SequenceFlow giữa Event và

shopping			Gateway
Payment process	5	0	
Pizza store	5	0	
Shopping process	4	2	SequenceFlow giữa 2 Gateway SequenceFlow giữa Gateway và Event
Nobel Prize	8	1	SequenceFlow giữa 2 Gateway
Purchase book	6	0	
Lend book	5	0	
Return book	6	0	

Theo nhận xét từ Bảng 4.2, có hai loại ký hiệu BPMN chưa được chuyển đổi sang biểu đồ ca sử dụng:

- SequenceFlow giữa hai Gateway: Gateway ảnh hưởng đến luồng thực hiện nghiệp vụ của mô hình đầu vào. Việc hai Gateway liên tiếp có xảy ra trong một số trường hợp cụ thể và dữ liệu này có thể được phân tích và đề xuất chuyển đổi thành Pre-condition trong phần đặc tả ca sử dụng.

- SequenceFlow giữa Gateway và Event: Một Event có ý nghĩa quyết định quy trình kết thúc hay được thực thi, trong khi Gateway có ý nghĩa trong việc phân luồng thực hiện quy trình. Liên kết giữa hai loại ký hiệu này có thể được xem xét chuyển thành Post-condition trong phần đặc tả ca sử dụng.

#### 4.2.3. Tổng kết thực nghiệm và đánh giá

Sau khi cài đặt phương pháp BPMN2UseCase, luận văn đã tiến hành thực nghiệm và áp dụng giải pháp trên mô hình quy trình nghiệp vụ cụ thể School Library System gồm 3 mô hình nhỏ. Kết quả sinh ra cho thấy biểu đồ ca sử dụng thu được chi tiết và giàu thông tin hơn so với kết quả của nghiên cứu liên quan trước đó [10], đặc biệt được bổ sung thêm nhiều quan hệ Association giữa các đối tượng Actor và Use Case.

Nhằm kiểm tra hiệu quả của giải pháp BPMN2UseCase, luận văn đã đề xuất phương pháp đánh giá dựa trên ba tiêu chí: tính ổn định, tính đúng đắn và tính đầy đủ. Để thực hiện điều này, luận văn đã xây dựng bộ test-cases gồm 14 test-cases là những mô hình quy trình nghiệp vụ thực tế. Dựa trên kết quả chạy thực nghiệm, có thể phân tích và đưa ra các nhận xét như sau:

- Tính ổn định: Quá trình chạy bộ test-cases cho thấy tỉ lệ lỗi hay sai sót nghiêm trọng là 0%, luôn ra được kết quả đúng định dạng chứng tỏ chương trình chạy ổn định, đảm bảo đầu ra tuân theo đúng quy tắc chuyển đổi mô hình.

- Tính đúng đắn: So sánh việc chuyển đổi các yếu tố thành phần cơ bản giữa hai phương pháp BPMN2UseCase và thủ công, tỉ lệ đúng trung bình là 95.9%. Vấn đề xảy ra trong trường hợp một Participant có nhiều Lane bên trong, dẫn đến số Actor sinh ra có sự sai khác.
- Tính đầy đủ: Phương pháp bao trùm và chuyển đổi được hầu hết các loại ký hiệu BPMN, có hai loại ký hiệu chưa thể chuyển đổi sẽ được xem xét và đưa vào phát triển trong hướng nghiên cứu sắp tới.

### **4.3. Tổng kết chương**

Trong chương này, quá trình cài đặt, áp dụng và đánh giá phương pháp sinh tự động ca sử dụng từ mô hình quy trình nghiệp vụ đã được trình bày chi tiết và cụ thể. Luận văn đã nêu rõ các bộ công cụ, môi trường hỗ trợ và các kỹ thuật biểu diễn mô hình được sử dụng cho phương pháp. Bộ quy tắc chuyển đổi mô hình đề xuất ở Mục 3.5 được cài đặt trong chương trình BPMN2UseCase sử dụng ngôn ngữ ATL: xây dựng các hàm chính và hàm hỗ trợ chuyển đổi thực hiện từng quy tắc. Khi việc cài đặt hoàn thiện, chương trình được thử nghiệm và áp dụng trên các tình huống cụ thể với bộ test-cases để có thể phân tích đánh giá và rút ra nhận xét về hiệu quả của giải pháp.

## CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong quá trình phát triển phần mềm ứng dụng nói chung, đặc biệt trong các tình huống thay đổi quy trình nghiệp vụ, việc sử dụng và sinh ra các chế tác phần mềm từ quy trình có sẵn là một nhu cầu đòi hỏi nhiều nghiên cứu và tìm hiểu. Đổi với những hệ thống lớn và phức tạp, việc chuyển đổi tự động có thể mang lại nhiều lợi ích cụ thể: mang lại hiệu quả cao, tiết kiệm nhiều nguồn lực và dễ dàng thích nghi với thay đổi. Dựa trên nhu cầu đó, luận văn đã cung cấp một phương pháp sinh tự động ca sử dụng từ mô hình quy trình nghiệp vụ và đạt được một số kết quả cụ thể. Bên cạnh đó, luận văn cũng xác định những điểm hạn chế để xem xét khắc phục và phát triển trong hướng nghiên cứu tiếp theo.

### 5.1. Kết quả đạt được

Sau quá trình tìm hiểu và nghiên cứu, luận văn đã đạt được các đóng góp như sau:

- Luận văn đã tìm hiểu tổng quan về kỹ nghệ hướng mô hình (MDE), khái niệm siêu mô hình, nghiên cứu các mô hình ca sử dụng, mô hình quy trình nghiệp vụ và ký pháp BPMN2 cùng các công cụ hỗ trợ xây dựng mô hình hóa.
- Luận văn đã phân tích, tìm hiểu và diễn giải các thành phần của siêu mô hình quy trình nghiệp vụ và siêu mô hình ca sử dụng. Từ đó có thể đánh giá sự tương quan/ánh xạ và xây dựng được bộ luật chuyển đổi mô hình. Bộ quy tắc chuyển đổi gồm 10 luật được chia thành ba nhóm: luật chuyển đổi cho các đối tượng cơ bản, luật chuyển đổi cho các quan hệ giữa các đối tượng và luật chuyển đổi cho các quan hệ giữa đối tượng và dữ liệu.
- Luận văn đã biểu diễn mô hình quy trình nghiệp vụ dưới dạng biểu đồ và cú pháp bpmn qua công cụ Eclipse BPMN2 Modeler. Sau đó sử dụng ngôn ngữ Eclipse ATL để xây dựng các mã cài đặt chương trình BPMN2UseCase, sinh tự động uml file chứa cú pháp trừu tượng của ca sử dụng từ file bpmn đầu vào. Cuối cùng, hiển thị mô hình ca sử dụng dưới dạng biểu đồ hình vẽ trực quan bằng cách đưa file uml đầu ra vào công cụ Eclipse Papyrus và tạo ra biểu đồ.
- Luận văn đã áp dụng phương pháp BPMN2UseCase vào bài toán thực tế Nobel Prize và so sánh kết quả thu được với nghiên cứu đã công bố trước đó. Bên cạnh đó, xây dựng bộ test-cases gồm 14 test (là các bài toán/mô hình quy trình nghiệp vụ thực tế) để kiểm tra, phân tích và đánh giá hiệu quả cũng như tính đúng đắn của phương pháp.

Với các kết quả trên, phương pháp có thể được ứng dụng trong thực tiễn như một công cụ hỗ trợ để gia tăng tự động hóa trong phát triển phần mềm, cụ thể là ở khâu xác định các chức năng phần mềm. Đặc biệt, trong các tổ chức, doanh nghiệp đã có sẵn các mô hình quy trình nghiệp vụ tương đối đầy đủ và chi tiết, việc áp dụng phương pháp giúp cho việc mô hình hóa yêu cầu phần mềm trở nên nhanh chóng, phản

ánh được nhu cầu và hoạt động nghiệp vụ của hệ thống một cách chính xác hơn và hạn chế được một số nhược điểm của cách thức thủ công.

## 5.2. Hướng phát triển

Việc phát triển phương pháp BPMN2UseCase sinh tự động ca sử dụng từ mô hình quy trình nghiệp vụ BPMN đã đạt được các kết quả sơ bộ và có thể mang lại một số lợi ích nhất định trong ứng dụng. Tuy nhiên, phương pháp vẫn còn các vấn đề hạn chế và có thể phát triển thêm để hoàn thiện hơn. Luận văn xin được đề xuất một số hướng nghiên cứu tiếp theo như sau:

- Phân tích và tìm hiểu thêm về trường hợp Participant có nhiều Lane và các loại ký pháp BPMN chưa được chuyển đổi khi sử dụng phương pháp để xuất và xem xét tìm phương án chuyển đổi.
- Nghiên cứu phương pháp sinh ra đặc tả ca sử dụng để có thể truyền tải và phản ánh quy trình nghiệp vụ một cách chi tiết và rõ ràng hơn. Xây dựng bộ luật chuyển đổi tương ứng và xem xét tìm hiểu ngôn ngữ Acceleo (Eclipse) để cài đặt chương trình.
- Mở rộng phương pháp BPMN2UseCase để có thể sinh tự động nhiều mô hình UML từ mô hình quy trình nghiệp vụ hơn, ví dụ biểu đồ hoạt động (UML Activity diagram) hoặc biểu đồ tuần tự (UML Sequence diagram).

## TÀI LIỆU THAM KHẢO

### Tiếng Việt

- [1] Bộ Thông tin và Truyền thông – Cục Tin học hóa (2017), Tổng quan về Tiêu chuẩn ký hiệu và mô hình hóa quy trình nghiệp vụ (Business Process Modelling and Notation - BPMN), phiên bản 2.0.
- [2] Phạm Nguyễn Cường, Hồ Tường Vinh, *Phân tích thiết kế hệ thống hướng đối tượng bằng UML*, Đại học KHTN-TP. HCM, ASIA-ITC 1.
- [3] Trần Việt Cường, *Tiêu chuẩn XMI*, Cục Tin học hóa.

### Tiếng Anh

- [4] Beichter F. W., Herzog O., Petzsch H. (1984), "SLAN-4-A software specification and design language.", *IEEE transactions on software engineering* 2, pp. 155-162.
- [5] Bider I. (2002), *State-oriented business process modeling: principles, theory and practice*.
- [6] Bouzidi A., Haddar N., Abdallah M. B., Haddar K. (2017), "Deriving Use Case Models from BPMN Models", *IEEE/ACS 14th International Conference on Computer Systems and Applications*, Hammamet, pp. 238-243.
- [7] Bouzidi, A., Haddar, N., Ben-Abdallah, M. and Haddar, K. (2020), "Toward the Alignment and Traceability between Business Process and Software Models", *Proceedings of the 22nd International Conference on Enterprise Information Systems*, Vol. 2, pp. 701-708.
- [8] Cetinkaya D., Verbraeck A., and Seck M. D. (2011), "MDD4MS: A Model Driven Development Framework for Modeling and Simulation", *Proceedings of the 2011 Summer Computer Simulation Conference*, The Hague, Netherlands.
- [9] Cruz E. F., Machado R. J., and Santos M. Y. (2014), "From Business Process Models to Use Case Models: A systematic approach", *Advances in Enterprise Engineering VIII*. Springer International Publishing, pp. 167-181.
- [10] Cruz E. F., Machado R. J., and Santos M. Y. (2015), "Bridging the gap between a set of interrelated business process models and software models", *17th International Conference on Enterprise Information Systems*, pp. 338–345.
- [11] Cruz E. F., Cruz A. M. R (2018), "Deriving Integrated Software Design Models from BPMN Business Process Models", *Proceedings of the 13th International Conference on Software Technologies*, pp. 571-582.

- [12] Davenport T. H. (1993), *Process Innovation: Reengineering Work through Information Technology*, Harvard Business School Press, Boston.
- [13] Dijkman R. M., Joosten S. M. (2002), *An algorithm to derive use case diagrams from business process models*, the 6th Intern. Conf. on Software Engineering and Applications, US.
- [14] Dijkman R. M., Joosten S. M. (2002), *Deriving use case diagrams from business process models*, CTIT Tech. Rep., Enschede, The Netherlands.
- [15] Nurcan S., Grosz G., and Souveyet C. (1998), "Describing business processes with a guided use case approach", *Proceedings of the 1998 Conference on Advanced Information Systems Engineering*, Vol. 1413 of Lecture Notes in Computer Science, pp. 339–362, Berlin.
- [16] Object Management Group (2001), *OMG Unified Modeling Language Specification version 1.4*, OMG Specification formal/2001, pp. 09-67.
- [17] Object Management Group (2011), *Business process model and notation (BPMN) version 2.0*, tech. rep., Object Management Group.
- [18] Rajagopal P., Lee R., Ahlswede T., Chiang C., Karolak D. (2005), "A new approach for software requirements elicitation." *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, IEEE.
- [19] Rehman T., Khan M. N. A., Riaz N. (2013), "Analysis of Requirement Engineering Processes, Tools/Techniques and Methodologies", *International Journal of Information Technology and Computer Science*, pp. 40-48.
- [20] Rodríguez A., García-Rodríguez de Guzmán I. (2007), "Obtaining Use Cases and Security Use Cases from Secure Business Process through the MDA Approach", *Proceedings of the 5th International Workshop on Security in Information Systems*, pp. 209-219.
- [21] Rhazali Y., Hadi Y., Mouloudi A. (2014), "Transformation Method CIM to PIM: From Business Processes Models Defined in BPMN to Use Case and Class Models Defined in UML", *World Academy of Science, Engineering and Technology – International Journal of Computer, Electrical, Automation, Control and Information Engineering*, Vol. 8 (No. 8), pp. 1467–1471.
- [22] Turkman S. and Taweel A. (2019), "Business Process Model Driven Automatic Software Requirements Generation", *Shishkov, B. (ed.) BMSD 2019. LNIP*, vol. 356, pp. 270–278.
- [23] White S. A. (2004), *Introduction to BPMN*, IBM Cooperation.

- [24] <https://wiki.eclipse.org/ATL/Concepts> (truy cập ngày 20/07/2022)
- [25] Di Ruscio D., Eramo R., Pierantonio A. (2012), “Model transformations”, *Formal Methods for Model-Driven Engineering*, pp. 91–136, Springer.
- [26] Object Management Group (2003), *MDA Guide version 1.0.1*, OMG Document/2003-06-01.
- [27] Czarnecki K, Helsen S (2003), “Classification of model transformation approaches”, *Proceedings of the 2nd OOPSLA workshop on generative techniques in the context of the model driven architecture*, Vol. 45 (No. 3), pp 1–17.
- [28] <https://www.edrawmax.com/templates/> (truy cập ngày 09/06/2022)

## PHỤ LỤC

Chi tiết xây dựng bộ test-cases gồm mô hình quy trình nghiệp vụ đầu vào và kết quả sau khi chạy chương trình BPMN2UseCase là các biểu đồ ca sử dụng.

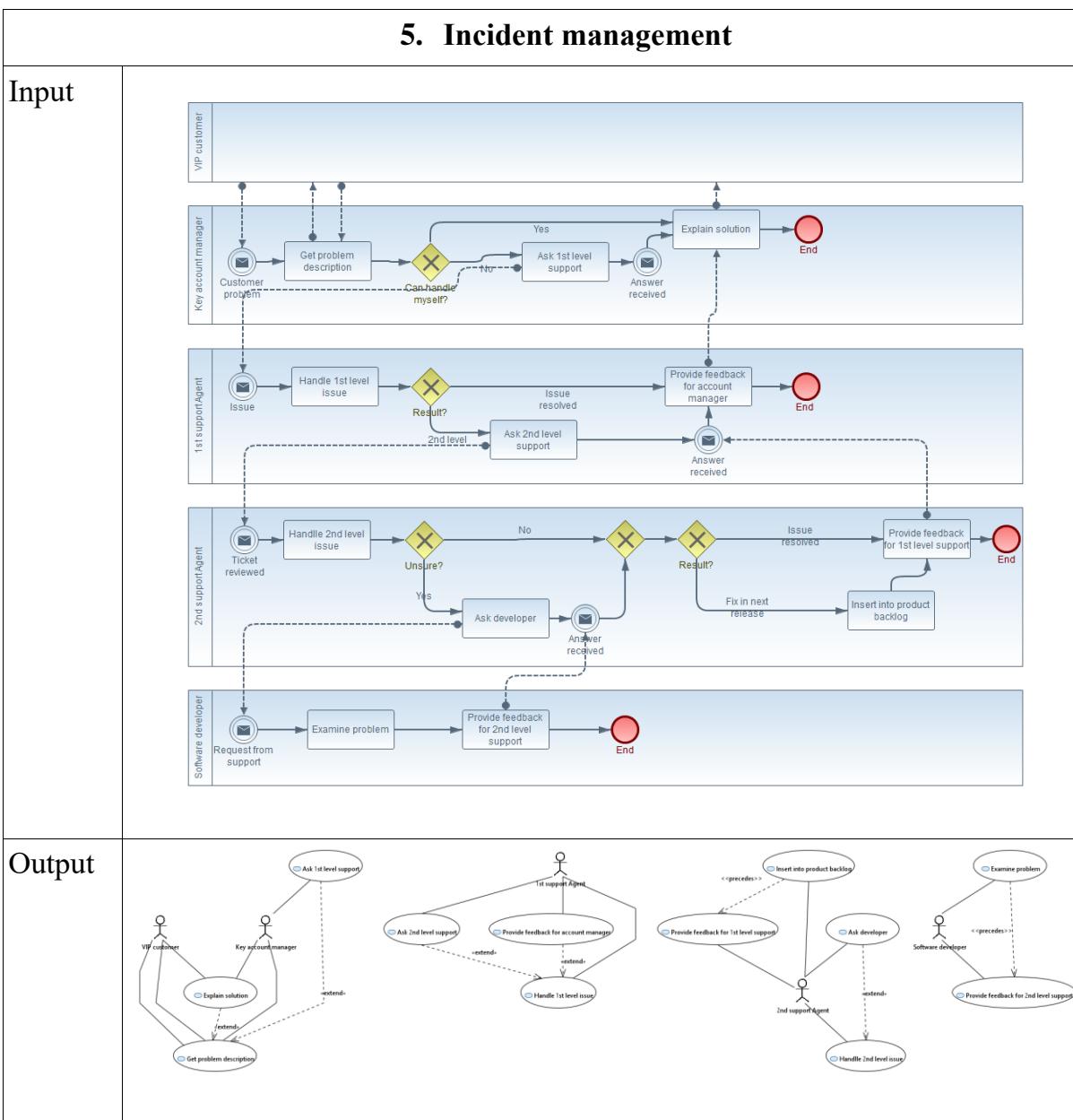
<b>1. Auction Service</b>	
Input	
	<pre> graph TD     subgraph Seller [Seller]         Start((Start)) --&gt; SA[Send Auction Creation Request]         SA --&gt; DR{ }         DR -- No --&gt; AR[Account Creation Request]         AR --&gt; SR[Send Registration Info]         SR --&gt; AC[ ]         AC --&gt; SC[Send payment]         SC --&gt; End1(((End)))     end      subgraph AuctioningCenter [Auctioning Center]         DR -- Yes --&gt; SCR[Second Creation Request]         SCR --&gt; AR2[Second Creation Request]         AR2 --&gt; SR2[Send Registration Info]         SR2 --&gt; AC2[ ]         AC2 --&gt; SC2[Send payment]         SC2 --&gt; End2(((End)))     end      AC --&gt; SCR     AC2 --&gt; SCR     SCR --&gt; AR     SCR --&gt; AR2     AR --&gt; SR     AR2 --&gt; SR2     SR --&gt; AC     SR2 --&gt; AC2     AC --&gt; SC     AC2 --&gt; SC2     SC --&gt; End1     SC2 --&gt; End2 </pre>
Output	
	<pre> usecase SendPayment as SP usecase SecondCreationRequest as SCR usecase SendAuctionCreationRequest as SACR usecase SendRegistrationInfo as SRI usecase ReceiveRequests as RR usecase SecondCreationConfirmation as SCC usecase SendRequest as SR  class Seller     SR     SACR     SRI     &lt;&lt;include&gt;&gt; SR     SCR  class AuctioningCenter     RR     &lt;&lt;include&gt;&gt; RR     SCC </pre>
<b>2. Book selling</b>	
Input	
	<pre> graph TD     subgraph Customer [Customer]         Start((Start)) --&gt; RB[Request for books]         RB --&gt; DR{ }         DR --&gt; PA[Pay the order]         PA --&gt; RBK[Receive books]         RBK --&gt; AR[Acknowledge receipt]     end      subgraph Status [Status]         DR --&gt; UR{ }         UR -- Unavailable --&gt; End1(((End)))         UR -- Available --&gt; CI[Charging issue]         CI --&gt; CP[Confirm payment]         CP --&gt; I[Issue invoice]         I --&gt; DR     end      subgraph Financing [Financing]         DR --&gt; DR     end      subgraph Deliver [Deliver]         DR --&gt; DR         DR --&gt; DB[Deliver books]         DB --&gt; CD[Confirm delivery]         CD --&gt; End2(((End)))     end </pre>

<b>Output</b> <pre> graph TD     subgraph Customer         Start((Start)) --&gt; RequestBooking[Request booking]         RequestBooking --&gt; BookingDetails[Booking details]     end      subgraph TravelAgent         GetBookingRequest[Get booking request] --&gt; CheckAvailability{Check availability}         CheckAvailability -- Yes --&gt; ProposeBookingStatus[Propose booking status]         ProposeBookingStatus -- Accepted --&gt; ConfirmBooking[Confirm booking]         ConfirmBooking --&gt; End((End))         ProposeBookingStatus -- Not accepted --&gt; End         GetBookingRequest -.-&gt; GetAlternativeTime[Get alternative time]         GetAlternativeTime -.-&gt; ProposeBookingStatus     end      subgraph CarDriver         PickupCustomer[Pickup customer] --&gt; CompleteAssignment[Complete assignment]         CompleteAssignment -- wait 1 hour --&gt; PickupCustomer     end </pre> <p>The diagram illustrates a car booking process involving three participants: Customer, Travel Agent, and Car Driver. The Customer starts by requesting booking, which triggers a booking details step. The Travel Agent handles the booking request, checking availability. If available, they propose booking status; if not, they get alternative time. The proposed booking status leads to confirming the booking, which ends the process. If the booking is not accepted, the process ends. The Travel Agent also assigns the car operator and notifies the agent. The Car Driver picks up the customer and completes the assignment after a one-hour wait.</p>	
<b>3. Car booking</b>	
<b>Input</b> <pre> graph TD     subgraph Customer         Start((Start)) --&gt; RequestBooking[Request booking]         RequestBooking --&gt; BookingDetails[Booking details]     end      subgraph TravelAgent         GetBookingRequest[Get booking request] --&gt; CheckAvailability{Check availability}         CheckAvailability -- Yes --&gt; ProposeBookingStatus[Propose booking status]         ProposeBookingStatus -- Accepted --&gt; ConfirmBooking[Confirm booking]         ConfirmBooking --&gt; End((End))         ProposeBookingStatus -- Not accepted --&gt; End         GetBookingRequest -.-&gt; GetAlternativeTime[Get alternative time]         GetAlternativeTime -.-&gt; ProposeBookingStatus     end      subgraph CarDriver         PickupCustomer[Pickup customer] --&gt; CompleteAssignment[Complete assignment]         CompleteAssignment -- wait 1 hour --&gt; PickupCustomer     end </pre> <p>This section shows the input activity diagram for the car booking process, which is identical to the output diagram above. It defines the flow from customer request through travel agent processing and car driver pickup.</p>	<b>Output</b> <pre> graph TD     subgraph Customer         Customer((Customer)) --&gt; RequestBooking[Request booking]     end      subgraph TravelAgent         GetBookingRequest[Get booking request] --&gt; CheckAvailability{Check availability}         CheckAvailability --&gt; ProposeBookingStatus[Propose booking status]         ProposeBookingStatus -- "extend" --&gt; GetAlternativeTime[Get alternative time]         ProposeBookingStatus -- "&lt;&lt;precedes&gt;&gt;" --&gt; ConfirmBooking[Confirm booking]         ConfirmBooking -- "extend" --&gt; PickupCustomer[Pickup customer]         PickupCustomer --&gt; CompleteAssignment[Complete assignment]         CompleteAssignment -- "extend" --&gt; PickupCustomer     end </pre> <p>The diagram shows the car booking process from the perspective of the Travel Agent. It includes steps for getting a booking request, checking availability, proposing booking status, getting alternative time, confirming booking, and picking up the customer. Relationships include 'extend' from 'Propose booking status' to 'Get alternative time', 'extend' from 'Propose booking status' to 'Confirm booking', and 'extend' from 'Confirm booking' to 'Pickup customer'. Additionally, there are '&lt;&lt;precedes&gt;&gt;' relationships from 'Check availability' to 'Propose booking status' and from 'Propose booking status' to 'Confirm booking'. The Travel Agent also extends the 'Pickup customer' step to include a one-hour wait and a complete assignment step.</p>

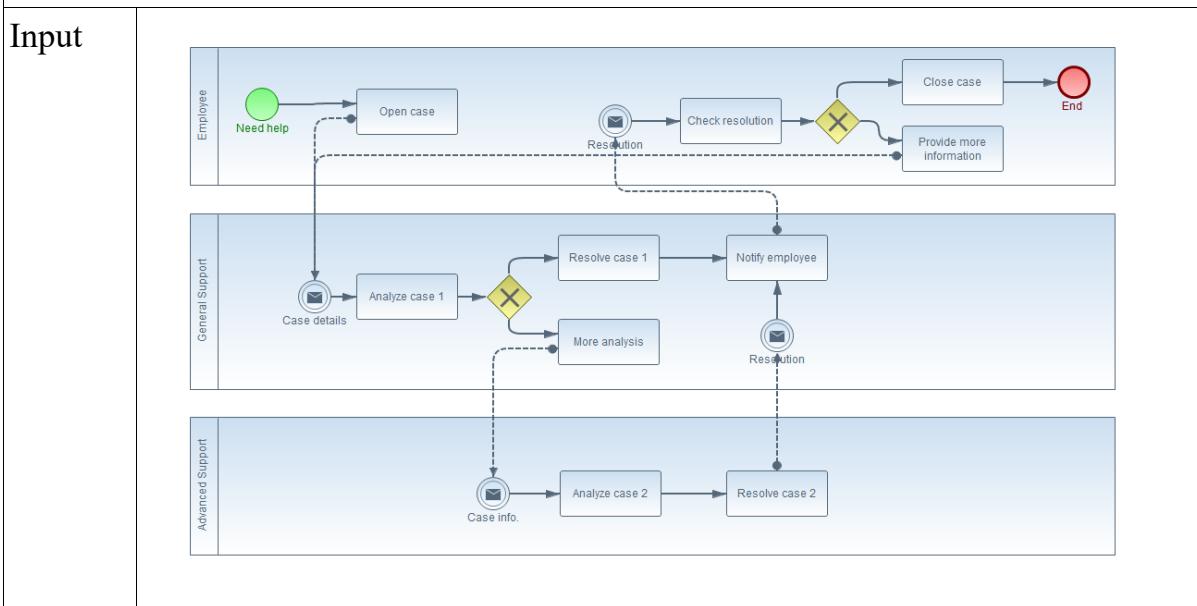
#### 4. Employment application

<b>Input</b>	<pre>     graph TD         subgraph Candidate [Candidate]             Start((Start)) --&gt; Submit[Submit Application]             Submit --&gt; Valid{Valid?}             Valid -- No --&gt; End1((End))             Valid -- Yes --&gt; Interview[Interview]             Interview --&gt; ReviewOffer[Review offer]             ReviewOffer --&gt; Satisfied{Satisfied?}             Satisfied -- Yes --&gt; GetHired((Get hired))             Satisfied -- No --&gt; Reject[Reject]             Reject --&gt; End2((End))         end         subgraph Employer [Employer]             Application[Application] --&gt; ReviewApp[Review Application]             ReviewApp --&gt; Accept{Accept?}             Accept -- No --&gt; End3((End))             Accept -- Yes --&gt; PlanInterview[Plan interview]             PlanInterview --&gt; InterviewCandidates[Interview candidates]             InterviewCandidates --&gt; JudgeDecide[Judge and decide]             JudgeDecide --&gt; Offer[Send offer]             Offer --&gt; AcceptOffer{Accept?}             AcceptOffer -- No --&gt; RejectCandidate[Reject Candidate]             RejectCandidate --&gt; End4((End))             AcceptOffer -- Yes --&gt; SendOffer[Send offer]             SendOffer --&gt; HireCandidate[Hire candidate]             HireCandidate --&gt; End5((End))         end         Interview &lt;--&gt; ReviewOffer         ReviewOffer &lt;--&gt; Satisfied         Satisfied &lt;--&gt; GetHired         Reject &lt;--&gt; End2         Application &lt;--&gt; ReviewApp         ReviewApp &lt;--&gt; Accept         Accept &lt;--&gt; End3         PlanInterview &lt;--&gt; InterviewCandidates         InterviewCandidates &lt;--&gt; JudgeDecide         JudgeDecide &lt;--&gt; Offer         Offer &lt;--&gt; AcceptOffer         AcceptOffer &lt;--&gt; RejectCandidate         AcceptOffer &lt;--&gt; SendOffer         SendOffer &lt;--&gt; HireCandidate     </pre>
<b>Output</b>	<pre>     graph TD         Candidate --&gt; Submit[Submit Application]         Submit --&gt; Interview[Interview]         Interview --&gt; ReviewOffer[Review offer]         ReviewOffer --&gt; Satisfied{Satisfied?}         Satisfied -- Yes --&gt; GetHired((Get hired))         Satisfied -- No --&gt; Reject[Reject]                  Employer --&gt; ReviewApp[Review Application]         ReviewApp --&gt; Accept{Accept?}         Accept -- No --&gt; End1((End))         Accept -- Yes --&gt; PlanInterview[Plan interview]         PlanInterview --&gt; InterviewCandidates[Interview candidates]         InterviewCandidates --&gt; JudgeDecide[Judge and decide]         JudgeDecide --&gt; Offer[Send offer]                  Offer --&gt; AcceptOffer{Accept?}         AcceptOffer -- No --&gt; RejectCandidate[Reject Candidate]         RejectCandidate --&gt; End2((End))         AcceptOffer -- Yes --&gt; SendOffer[Send offer]         SendOffer --&gt; HireCandidate[Hire candidate]         HireCandidate --&gt; End3((End))                  Interview &lt;--&gt; ReviewOffer         ReviewOffer &lt;--&gt; Satisfied         Satisfied &lt;--&gt; GetHired         Reject &lt;--&gt; End1         Application &lt;--&gt; ReviewApp         ReviewApp &lt;--&gt; Accept         Accept &lt;--&gt; End2         PlanInterview &lt;--&gt; InterviewCandidates         InterviewCandidates &lt;--&gt; JudgeDecide         JudgeDecide &lt;--&gt; Offer         Offer &lt;--&gt; AcceptOffer         AcceptOffer &lt;--&gt; RejectCandidate         AcceptOffer &lt;--&gt; SendOffer         SendOffer &lt;--&gt; HireCandidate     </pre>

## 5. Incident management



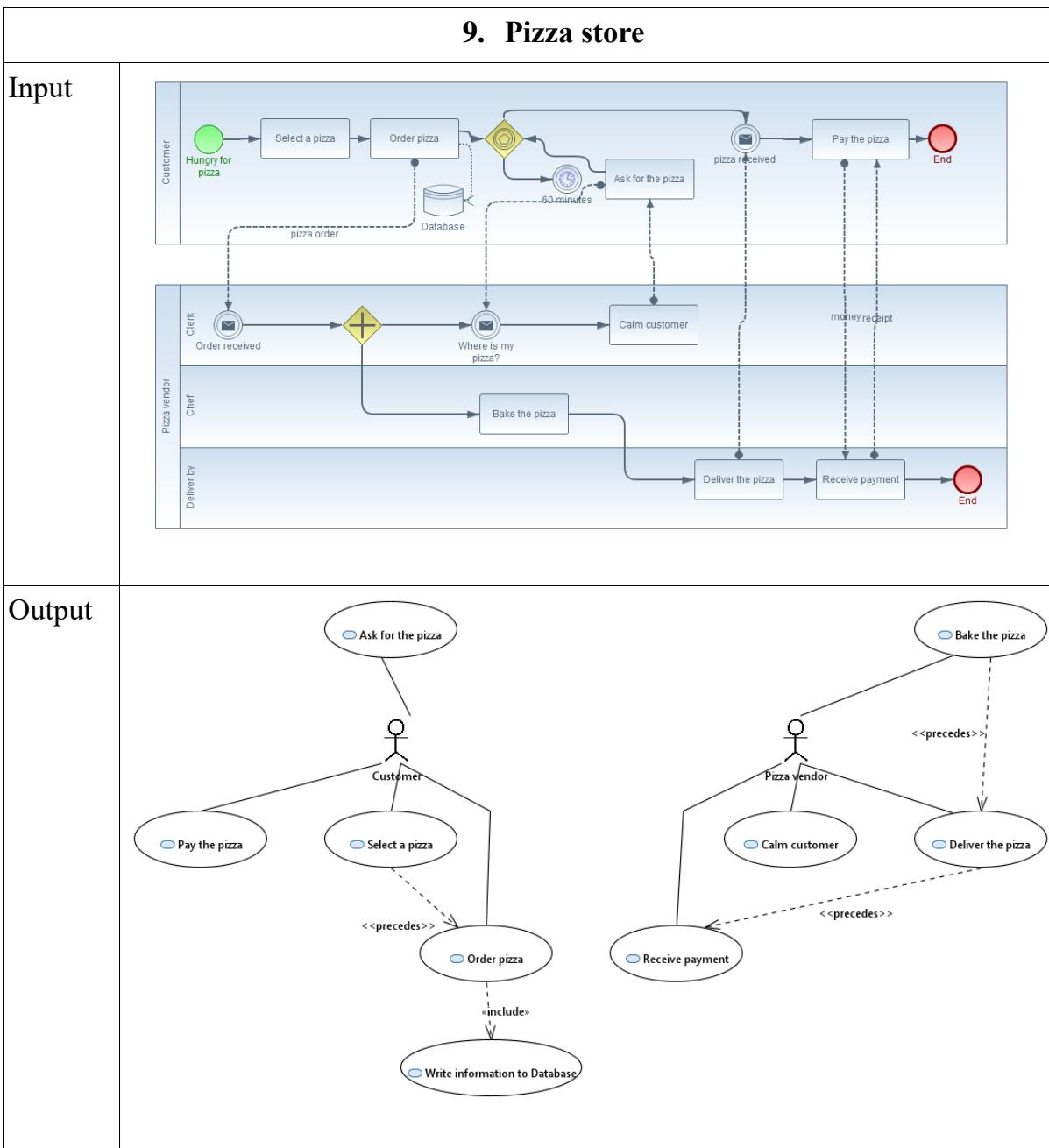
## 6. IT help



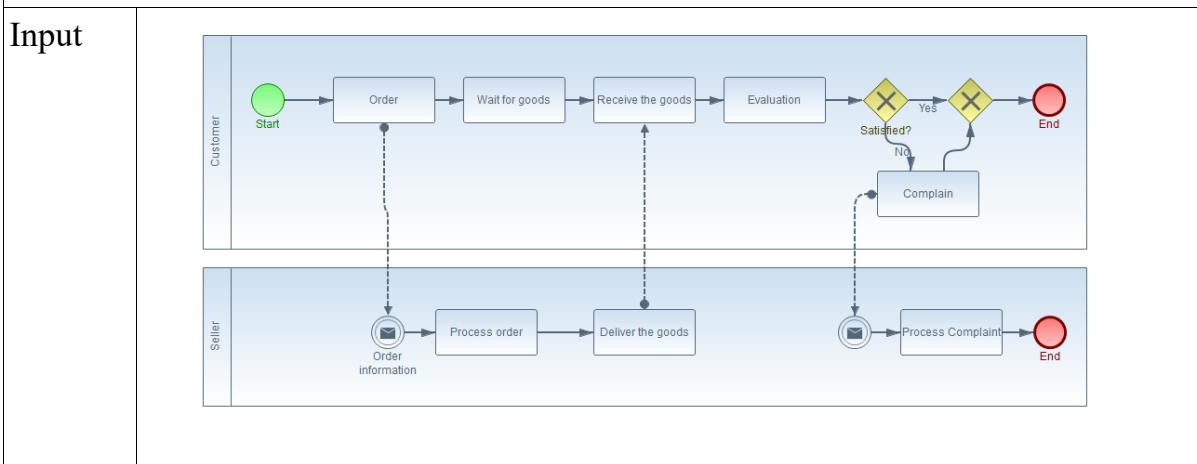
<b>Output</b> <pre> graph TD     subgraph General_Support [General Support]         R1[Resolve case 1] --&gt; N1[Notify employee]         N1 --&gt; A1[Analyze case 1]         A1 --&gt; M1[More analysis]         M1 --&gt; CR[Check resolution]     end     subgraph Advanced_Support [Advanced Support]         A2[Analyze case 2] --&gt; R2[Resolve case 2]         E1[Employee] --&gt; P1[Provide more information]         P1 --&gt; CR     end     N1 --&gt; A1     A1 --&gt; M1     M1 --&gt; CR     A2 --&gt; R2     R2 --&gt; P1     P1 --&gt; CR </pre>
<b>7. Online shopping</b>
<b>Input</b> <pre> graph TD     Start((Start)) --&gt; Choose[Choose a commodity]     Choose --&gt; Submit[Submit Order]     Submit --&gt; Order((Order))     Order --&gt; CheckX{Check Stock}     CheckX -- available --&gt; Prepare[Prepare the commodity]     Prepare --&gt; Deliver[Deliver the commodity]     Deliver --&gt; Notice[Shipment Notice]     CheckX -- stock out --&gt; StockOut[Indicate Stock out]     StockOut --&gt; Throw3{Intermediate Throw Event 3}     Throw3 --&gt; Notice </pre>
<b>Output</b> <pre> graph TD     C1[Choose a commodity] --&gt; C2[Customer]     S1[Indicate Stock out] --&gt; S2[Shopping Site]     S2 --&gt; S3[Prepare the commodity]     S3 --&gt; S4[Deliver the commodity]     S3 --&gt; S5[Deliver the commodity]     S5 --&gt; C3[Deliver the commodity]     S3 --&gt; S6[Deliver the commodity]     S6 --&gt; C4[Deliver the commodity]     S3 --&gt; S7[Deliver the commodity]     S7 --&gt; C5[Deliver the commodity] </pre>

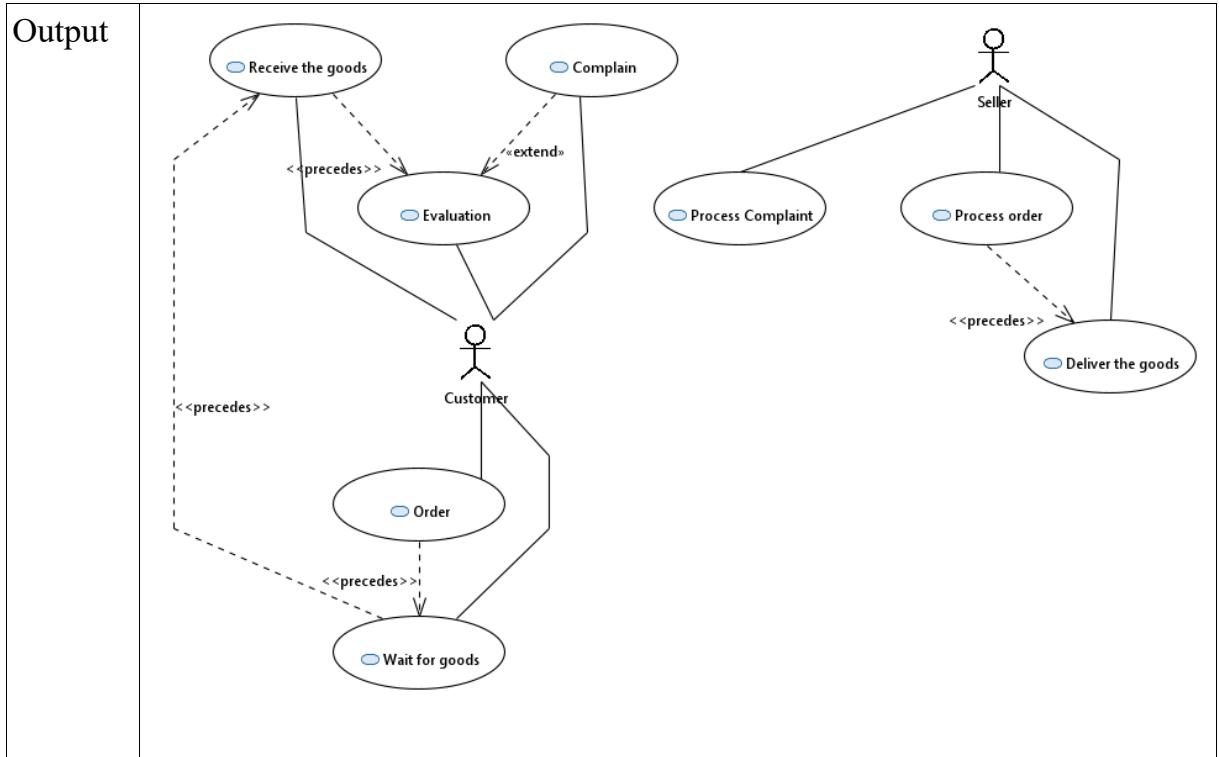
<b>8. Payment process</b>	
Input	
	<p>The BPMN diagram illustrates the Payment process across three vertical layers:</p> <ul style="list-style-type: none"> <li><b>Sales:</b> Starts at a green circle labeled "Start". It flows through "Identify Payment Method" to a decision diamond. If "Check or Cash" is chosen, it leads to "Accept Check or Cash", then "Prepare package for Customer", and finally "Deliver package to Customer", ending at "End Event 1". If "Credit Card" is chosen, it leads to "Authorize Credit card", then "Process Credit card", and then "Accept Check or Cash", followed by the same sequence as the cash path.</li> <li><b>Distribution:</b> A horizontal dashed line connects "Accept Check or Cash" and "Process Credit card". From "Accept Check or Cash", it continues to "Prepare package for Customer" and "Deliver package to Customer", leading to "End Event 1". From "Process Credit card", it goes to "Authorize Payment", which then leads to "End".</li> <li><b>Financial institution:</b> A separate column on the right shows "Process Credit card" connected to "Authorize Payment", which then leads to "End".</li> </ul>
Output	<p>The UML Activity Diagram details the interactions between the Retailer and the Financial Institution:</p> <ul style="list-style-type: none"> <li><b>Retailer Activities:</b> "Identify Payment Method", "Accept Check or Cash", "Authorize Credit card", "Prepare package for Customer", and "Deliver package to Customer".</li> <li><b>Financial Institution Activities:</b> "Process Credit card" and "Authorize Payment".</li> <li><b>Relationships:</b> <ul style="list-style-type: none"> <li>"Identify Payment Method" has a solid arrow pointing to "Accept Check or Cash" and "Authorize Credit card", with a label "«extend»".</li> <li>"Accept Check or Cash" and "Authorize Credit card" both have dashed arrows pointing to "Prepare package for Customer", with a label "«precedes»".</li> <li>"Prepare package for Customer" has a solid arrow pointing to "Deliver package to Customer".</li> <li>"Deliver package to Customer" has a solid arrow pointing to "End".</li> <li>"Process Credit card" has a dashed arrow pointing to "Authorize Payment".</li> <li>"Authorize Payment" has a solid arrow pointing to "End".</li> </ul> </li> </ul>

## 9. Pizza store



## 10. Shopping process





## 11. Nobel Prize

