

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

NGUYỄN TIẾN TÙNG

**NGHIÊN CỨU CÔNG CỤ HỖ TRỢ ĐÁM BẢO CHÍNH SÁCH
QUYỀN TRUY CẬP TRONG MỘT SỐ QUY TRÌNH NGHIỆP VỤ
NGÂN HÀNG THƯƠNG MẠI**

LUẬN VĂN THẠC SĨ KỸ THUẬT PHẦN MỀM

Hà Nội – Năm 2019

LỜI CAM ĐOAN

Tôi là Nguyễn Tiến Tùng, học viên lớp Cao học K22 - Trường Đại học Công nghệ - ĐHQGHN – cam kết Luận văn tốt nghiệp là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của TS. Đặng Đức Hạnh, Khoa Công nghệ Thông tin, Trường Đại học Công nghệ - ĐHQGHN. Các kết quả trong Luận văn tốt nghiệp là trung thực, không sao chép toàn văn của bất kỳ công trình nào khác.

LỜI CẢM ƠN

Em xin bày tỏ lòng biết ơn sâu sắc tới Thầy giáo, TS. Đặng Đức Hạnh, Khoa Công nghệ Thông tin – Trường Đại học Công nghệ - ĐHQGHN. Trong quá trình từ khi được Thầy giảng dạy hướng dẫn, Thầy vẫn luôn ủng hộ và động viên em rất nhiều. Nhờ sự quan tâm chỉ bảo và những ý kiến đóng góp quý báu của Thầy, em mới có thể tiếp tục và hoàn thành luận văn này.

Tôi xin chân thành cảm ơn tập thể các Giảng viên Trường Đại học Công nghệ nói chung và Khoa Công Nghệ Thông Tin nói riêng đã tận tình giảng dạy truyền đạt cho tôi kiến thức, kinh nghiệm quý báu trong suốt những năm học vừa qua.

Tôi cũng xin cảm ơn các bạn học viên Khóa 22 đã cùng tôi tiếp cận nghiên cứu, tìm hiểu nhiều lĩnh vực hữu ích và xu hướng công nghệ mới để hoàn thành luận văn và phục vụ trong công việc.

Cuối cùng tôi xin chân thành cảm ơn gia đình, người thân đã hết lòng giúp đỡ, hỗ trợ về vật chất lẫn tinh thần giúp tôi yên tâm học tập và nghiên cứu trong suốt quá trình học tập và thực hiện luận văn.

MỤC LỤC

LỜI CAM ĐOAN	i
LỜI CẢM ƠN	ii
MỤC LỤC	iii
DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT	v
DANH MỤC HÌNH VẼ	vii
MỞ ĐẦU	1
CHƯƠNG 1 KIẾN THỨC NỀN TẢNG.....	3
1.1. Giới thiệu tổng quan về quy trình nghiệp vụ	3
1.1.1. Khái niệm quy trình nghiệp vụ.....	3
1.1.2. Mô hình quy trình nghiệp vụ BPMN.....	4
1.1.2.1. Lịch sử phát triển của BPMN.....	4
1.1.2.2. Các phần tử (element) của BPMN.....	5
1.1.2.3. Các mô hình thành phần của BPMN.....	7
1.1.2.4. Các loại biểu đồ BPMN	9
1.2. Mô hình điều khiển truy cập	10
1.2.1. Khái niệm điều khiển truy cập	10
1.2.2. Cơ chế điều khiển truy cập - MAC/DAC	10
1.2.3. Mô hình dựa trên định danh và danh sách - IBAC/ACLs	11
1.2.4. Mô hình dựa trên vai trò - RBAC	11
1.2.5. Mô hình dựa trên thuộc tính - ABAC	12
1.3. Bộ công cụ hỗ trợ Activiti	13
1.3.1. Mô tả tổng quan.....	13
1.3.2. Cơ chế thực thi - Activiti Engine.....	14
1.3.3. Một số ưu và nhược điểm của công cụ Activiti	15
1.3.3.1. So sánh Activiti và JBPMP	16
1.3.3.2. So sánh Activiti và BonitaSoft	16
1.3.3.3. Tóm lược công cụ Activiti	17
1.4. Tổng kết chương	17
CHƯƠNG 2 PHƯƠNG PHÁP XÂY DỰNG MÔ HÌNH ABAC VÀ CÔNG CỤ HỖ TRỢ.....	18
2.1. Mô hình điều khiển truy cập ABAC	18
2.1.1. Cơ chế điều khiển trong mô hình ABAC	18
2.1.2. Ưu điểm của mô hình ABAC	19
2.2. Thiết kế mô hình ABAC.....	20
2.3. Tích hợp mô hình ABAC vào công cụ Activiti	23
2.3.1. Cơ chế hoạt động của công cụ Activiti	23
2.3.1.1. Các thành phần chính công cụ Activiti.....	23
2.3.1.2. Module Activiti UI.....	25
2.3.2. Ý tưởng tích hợp mô hình ABAC vào công cụ Activiti.....	28
2.3.3. Thiết kế tích hợp mô hình ABAC vào thành phần Activiti	28
2.3.4. Cài đặt thiết kế tích hợp.....	29

2.4. <i>Tổng kết chương</i>	33
CHƯƠNG 3 VẬN DỤNG VÀ THỰC NGHIỆM.....	34
3.1. <i>Bài toán nghiệp vụ “Phê duyệt hồ sơ tín dụng”</i>	34
3.2. <i>Yêu cầu về chính sách truy cập</i>	36
3.3. <i>Xây dựng mô hình ABAC</i>	38
3.4. <i>Xây dựng và thực thi mô hình quy trình trên Activiti</i>	39
3.4.1. <i>Cài đặt Activiti trên Webserver</i>	39
3.4.2. <i>Xây dựng mô hình quy trình “Phê duyệt hồ sơ tín dụng” trên Activiti</i>	41
3.4.2.1. <i>Biểu diễn mô hình quy trình</i>	41
3.4.2.2. <i>Triển khai mô hình quy trình</i>	48
3.4.2.3. <i>Thiết lập tập các quy tắc kiểm soát thẩm quyền</i>	50
3.4.3. <i>Thực thi quy trình trên Activiti</i>	50
3.5. <i>Kết quả thực nghiệm</i>	52
3.6. <i>Đánh giá kết quả vận dụng và thực nghiệm</i>	56
3.7. <i>Tổng kết chương</i>	57
KẾT LUẬN.....	58
TÀI LIỆU THAM KHẢO	60
PHỤ LỤC.....	62

DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT

Tên viết tắt	Tên đầy đủ	Ý nghĩa
OMG	Object Management Group	Tổ chức quản lý đối tượng
BPEL	Business Process Execution Language	Ngôn ngữ thực thi quy trình nghiệp vụ. Ngôn ngữ BPEL sẽ định nghĩa quy trình cũng như các tác vụ thực hiện trên quy trình đó.
BPMN	Business Process Modelling and Notation	Ngôn ngữ luồng công việc cho phép mô hình hóa các tiến trình nghiệp vụ ở mức cao (mức phân tích và thiết kế nghiệp vụ). Nó bao gồm sơ đồ (biểu diễn hướng người dùng), và văn bản (để lưu trữ và xử lý tự động, sử dụng ngôn ngữ XML).
BPMI	Business Process Management Initiative	Tổ chức Sáng kiến quản lý quy trình nghiệp vụ
BPML	Business Process Modeling Language	Ngôn ngữ mô hình hóa quy trình nghiệp vụ

BPMS	Business Process Management System	Hệ thống quản lý quy trình nghiệp vụ
DAC	Discretionary Access Control	Điều khiển truy nhập tùy ý. Sự điều khiển được gọi là tùy ý (discretion) theo nghĩa là một chủ thể đã có một số quyền truy nhập nào đó thì có thể chuyển quyền đó (một cách trực tiếp hay gián tiếp) cho bất kỳ chủ thể khác.
MAC	Mandatory Access Control	Điều khiển truy nhập bắt buộc. Bắt buộc (mandatory) với nghĩa là các quyền truy nhập đã bị quy định cứng bởi hệ thống, và nó không thể bị thay đổi bởi người dùng hoặc bởi chương trình của người dùng.
RBAC	Role-Based Access Control	Mô hình điều khiển truy nhập dựa trên vai trò.
ABAC	Attribute-Based Access Control	Mô hình điều khiển truy nhập dựa trên thuộc tính.
CNTT	Công nghệ thông tin	

DANH MỤC HÌNH VẼ

- Hình 1.1. Minh họa về mô hình hóa quy trình nghiệp vụ
- Hình 1.2. Minh họa về Quy trình nghiệp vụ riêng
- Hình 1.3. Minh họa quy trình nghiệp vụ công khai
- Hình 1.4. Minh họa về Quy trình nghiệp vụ cộng tác
- Hình 1.5. Ví dụ về chức năng các điểm kiểm soát truy cập
- Hình 1.6. Tổng quan công cụ Activiti
- Hình 1.7. Minh họa chuyển trạng thái trong Activiti Engine

- Bảng 1.1 – Danh sách các phần tử mô hình hóa cơ bản và ký hiệu
- Bảng 1.2 - Các thành phần trong công cụ Activiti
- Bảng 1.3 - Các điểm khác nhau giữa Activiti và jBPM

- Hình 2.1. Cơ chế cốt lõi của ABAC
- Hình 2.2. Ví dụ về truy cập chéo
- Hình 2.3. Kịch bản cơ bản của mô hình ABAC
- Hình 2.4. Thiết kế chi tiết mô hình
- Hình 2.5. Các thành phần bộ công cụ Activiti
- Hình 2.6. Chi tiết luồng xử lý khi Users đăng nhập
- Hình 2.7. Minh họa luồng xử lý xem một task
- Hình 2.8. Minh họa xử lý cho event Complete
- Hình 2.9. Minh họa xử lý cho luồng xử lý ABAC trong Activiti
- Hình 2.10. Quan hệ giữa các thực thể
- Hình 2.11. Bổ sung tập chính sách (P)
- Hình 2.12. Minh họa quản lý ủy quyền

- Hình 3.1. Quy trình phê duyệt hồ sơ tín dụng
 - Hình 3.2. Minh họa gán vai trò xác định trước
 - Hình 3.3. Minh họa vi phạm phê duyệt theo thẩm quyền hạn mức
 - Hình 3.4. Minh họa màn hình Activiti App
 - Hình 3.5. Minh họa màn hình Kickstart App
 - Hình 3.6. Minh họa màn hình Tasks
-

- Hình 3.7. Minh họa màn hình Identity management
- Hình 3.8. Minh họa tạo mới BPM
- Hình 3.9. Minh họa màn hình thiết kế mô hình
- Hình 3.10. Minh họa quy trình được mô hình hóa
- Hình 3.11. Minh họa form nhập liệu trong quy trình
- Hình 3.12. Minh họa gán yêu cầu cho KSV
- Hình 3.13. Minh họa điều khiển hướng quy trình
- Hình 3.14. Minh họa khi yêu cầu bị từ chối
- Hình 3.15. Minh họa các màn hình xem thông tin
- Hình 3.16. Minh họa phê duyệt cuối của quy trình
- Hình 3.17. Minh họa việc tạo màn hình dữ liệu
- Hình 3.18. Minh họa việc tạo màn hình dữ liệu
- Hình 3.19. Minh họa việc thiết kế màn hình dữ liệu
- Hình 3.20. Minh họa các đối tượng trên màn hình nhập dữ liệu
- Hình 3.21. Minh họa màn hình xem dữ liệu
- Hình 3.22. Minh họa việc tạo App cho mô hình
- Hình 3.23. Minh họa chọn mô hình cho App
- Hình 3.24. Minh họa publish cho App
- Hình 3.25. Thông tin quy tắc kiểm soát theo thẩm quyền
- Hình 3.26. Minh họa tạo người sử dụng/ nhóm người sử dụng
- Hình 3.27. Minh họa tạo mới quy trình
- Hình 3.28. Minh họa nhập liệu cho quy trình
- Hình 3.29. Minh họa việc hoàn thiện phê duyệt theo thẩm quyền
- Hình 3.30. Minh họa hoàn thiện phê duyệt
- Hình 3.31. Minh họa ủy quyền phê duyệt
- Hình 3.32. Minh họa tự gán Task
- Hình 3.33. Minh họa kết quả thực nghiệm
- Bảng 3.1 Bảng phân quyền chức năng người sử dụng
- Bảng 3.2 Minh họa kết quả các testcase
-

MỞ ĐẦU

“Quy trình phải nhất quán, con người luôn tuân thủ” – hai yếu tố quan trọng đảm bảo cho hoạt động vận hành thông suốt và hiệu của một tổ chức, doanh nghiệp trong môi trường kinh tế - xã hội không ngừng biến đổi. Quy trình cần phải đảm bảo tính thống nhất, đầy đủ thể hiện chiến lược cũng như tầm nhìn của tổ chức. Đồng thời, quy trình cũng cần uyển chuyển, dễ dàng đáp ứng việc tự động hóa và thay đổi trong quản lý... Con người là nhân tố sống còn, tối quan trọng trong bất cứ hoạt động kinh tế - xã hội nào. Trong hoàn cảnh đó, việc quản lý quy trình sao cho vận hành hiệu quả tối ưu là yếu tố quan trọng với mỗi tổ chức. Quản lý quy trình nghiệp vụ (Business Process Management - BPM) là một phương pháp được thiết kế để cải thiện các quy trình nghiệp vụ thông qua sự kết hợp của công nghệ và nghiệp vụ, là một mô hình làm việc kết hợp giữa các bộ phận kinh doanh, nghiệp vụ và CNTT cùng nỗ lực để làm cho các quy trình nghiệp vụ hiệu quả và tối ưu hơn. Trong quy trình cần có ít nhất hai người hoặc ứng dụng tham gia vào các công việc. Khi thông tin được truyền đạt từ người này sang người khác xuất hiện khả năng mất mát thông tin. Khả năng này càng tăng khi có nhiều cá nhân hoặc ứng dụng tham gia vào luồng công việc, hậu quả đem tới còn đặc biệt nghiêm trọng khi có sự tấn công và/hoặc mất an ninh bảo mật. Do đó, việc điều khiển truy cập đảm bảo “chính sách quyền truy cập” có vai trò quan trọng trong quy trình nghiệp vụ, nhất là trong lĩnh vực Tài chính – Ngân hàng. Theo đó, luận văn tập trung vào **“Nghiên cứu công cụ hỗ trợ đảm bảo Chính sách quyền truy cập trong một số quy trình nghiệp vụ Ngân hàng thương mại”**. Luận văn xác định các vấn đề cần giải quyết là tìm hiểu mô hình điều khiển truy cập theo thuộc tính hay mô hình ABAC, vận dụng vào bài toán nghiệp vụ “Phê duyệt hồ sơ tín dụng” trong Ngân hàng thương mại. Trước tiên, các kiến thức nền tảng được đề cập nhằm làm rõ hướng tiếp cận của luận văn là vận dụng mô hình truy cập trên công cụ hỗ trợ Activiti để giải quyết bài toán. Tiếp theo, quá trình phát triển tích hợp mô hình vào công cụ cho thấy việc cụ thể hóa hướng tiếp cận đã trình bày trước đó. Cuối cùng, với các kết quả thực nghiệm, luận văn chứng minh hướng đi đúng. Tuy nhiên, còn một số vấn đề mở vẫn cần tiếp tục phát triển trong thời gian tới là việc nâng cấp mô hình và việc phát triển hoàn thiện công cụ nhằm mục đích tạo ra một sản phẩm hoàn chỉnh có thể đưa vào sử dụng thực tế trong Ngân hàng thương mại.

Luận văn được tổ chức gồm 05 phần:

Giới thiệu. Lý do thực hiện đề tài, mục tiêu cần đạt

Chương 1. Kiến thức nền tảng về các mô hình và các công cụ hỗ trợ. Chương này giới thiệu tổng quan về quy trình nghiệp vụ, tiêu chuẩn mô hình BPMN và cuối cùng là giới thiệu sơ lược bộ công cụ mã nguồn mở Activiti.

Chương 2. Mô hình điều khiển truy cập ABAC về mô hình truy cập và cách tích hợp vào công cụ Activiti. Chương 2 đề cập về việc xây dựng mô hình điều khiển truy cập theo thuộc tính ABAC. Cuối chương, luận văn trình bày cách xây dựng và cài đặt mô hình tích hợp vào công cụ mở Activiti.

Chương 3. Vận dụng và thực nghiệm về bài toán trong Ngân hàng thương mại và phát triển thực tế. Chương này trình bày quá trình giải quyết bài toán “Phê duyệt hồ sơ tín dụng” áp dụng mô hình điều khiển truy cập theo thuộc tính ABAC. Phần cuối trình bày các kết quả đạt được khi thực nghiệm.

Kết luận. Kết quả đạt được và hướng cho tương lai.

CHƯƠNG 1

KIẾN THỨC NỀN TẢNG

Chương này giới thiệu tổng quan về quy trình nghiệp vụ và kỹ thuật mô hình hóa BPMN. Tiếp đó là phần diễn giải về mô hình truy cập thuộc tính ABAC. Cuối cùng chương giới thiệu về công cụ hỗ trợ việc quản lý mô hình BPMN và các cơ chế điều khiển truy cập.

1.1. Giới thiệu tổng quan về quy trình nghiệp vụ

Thông thường trong thời gian hàng ngày, các hoạt động của các cá nhân đều là một phần của các quy trình khác nhau. Ví dụ, khi ta đặt mua một cuốn sách trong một cửa hàng sách online thì khi đó một quy trình được thực thi gồm việc thanh toán, đóng gói và tới việc vận chuyển sách cho người mua. Phần đầu chương trình bày về các khái niệm quy trình nghiệp vụ cùng với các cách thức để mô hình quy trình trong nghiệp vụ thực tế.

1.1.1. Khái niệm quy trình nghiệp vụ

Quy trình – Process – được định nghĩa là một loạt các hành động/hoạt động có kết thúc; một loạt các hoạt động liên tiếp nhau hoặc các ứng xử đặc biệt trong quá trình sản xuất. Quy trình nghiệp vụ tồn tại song hành cùng quá trình kinh doanh/nghiệp vụ - Business - của một doanh nghiệp, một tổ chức. Quy trình nghiệp vụ thường được đề cập tới việc cách tổ chức các hoạt động tạo ra giá trị.

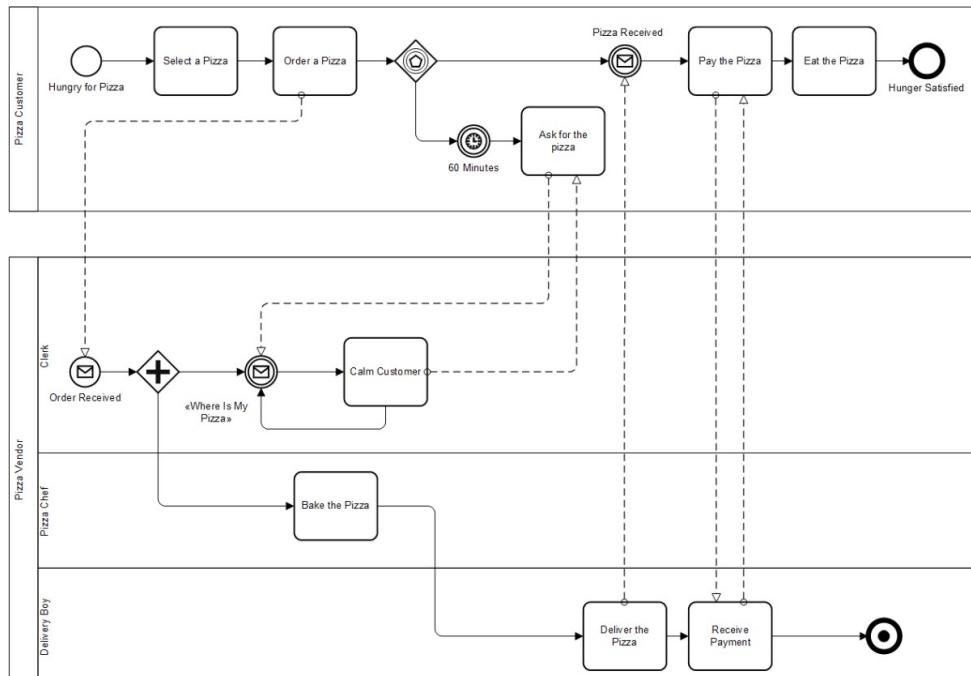
Để quản lý các quy trình nghiệp vụ của một tổ chức nói chung, cần thiết phải mô tả và tài liệu hóa. Có rất nhiều cách thức để thực hiện, tuy nhiên, cách dễ dàng và đơn giản nhất là sử dụng mô tả dạng văn bản hay dạng bảng. Các biểu đồ luồng thường được tạo ra bằng cách sử dụng các phần mềm về trình diễn và đồ họa. Các biểu đồ này hầu hết đều chứa các hình hộp và các mũi tên, không tuân theo một phương thức cụ thể nào. Do đó, dẫn đến không đáp ứng được các yêu cầu về việc biểu diễn các quy trình theo các khía cạnh như quy tắc, sự kiện, các đơn vị tổ chức, luồng dữ liệu...

Tuy nhiên, khi mô hình các quy trình nghiệp vụ các tác nhân thực hiện nghiệp vụ, người phân tích quy trình nghiệp vụ, người phát triển kỹ thuật và người quản lý nghiệp vụ gặp khó khăn trong việc hiểu ý tưởng của nhau [9]. 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) chung 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 về mô hình hóa quy trình nghiệp vụ theo đồ họa xác định các biểu tượng cho các phần tử quy trình nghiệp vụ, ý nghĩa cũng như các khả năng kết hợp của chúng.

1.1.2. Mô hình quy trình nghiệp vụ BPMN

Tiêu chuẩn Ký hiệu và mô hình hóa quy trình nghiệp vụ (BPMN) với mục đích chính là làm cầu nối khoảng cách về thông tin giữa các bên liên quan thường xuyên xảy ra trong việc thiết kế và triển khai quy trình nghiệp vụ [9], đã và đang được sử dụng rộng rãi để mô hình hóa quy trình nghiệp vụ 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 nghiệp vụ. Một cách đơn giản, ta hãy quan sát một ví dụ Hình 1.1.



Hình 1.1. Minh họa về mô hình hóa quy trình nghiệp vụ.

1.1.2.1. Lịch sử phát triển của BPMN

Ban đầu, BPMN được phát triển bởi Tổ chức Sáng kiến quản lý quy trình nghiệp vụ (BPMI), một tổ chức gồm các công ty về phần mềm [9]. 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 (BPML). So với BPEL, BPML được sử

dụng để xác định các mô tả quy trình có thể được thực thi bởi một BPMS, BPML không được tiếp tục phát triển nữa [9][4].

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 (OMG). Tổ chức 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.

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.

1.1.2.2. Các phần tử (element) của BPMN

Các phần tử của BPMN được phân thành 5 loại cơ bản sau [9]:

- Các đối tượng luồng (Flow Objects): là các phần tử đồ họa chính định nghĩa hành vi của một Quy trình nghiệp vụ. Có ba đối tượng luồng gồm Sự kiện (Event); Hoạt động (Activity); Cổng (Gateway). Activity tập trung trả lời câu hỏi *làm gì*. Tức là mô tả tất cả các công việc trong quy trình. Activity gồm 04 loại: Task - là từng việc chi tiết, tập các task thành một quy trình lớn; Transaction - là các giao dịch, gồm nhiều task mà các task này liên hệ logic với nhau; Sub-Process - là các quy trình con, hiểu đơn giản là quy trình nhỏ trong quy trình lớn; Call Activity - là hàm gọi, thực hiện gọi một sub process nào đó. Thành phần Gateways là bộ phận logic mà luồng của hệ thống sẽ thay đổi tùy vào các điều kiện khác nhau.

- Swimlanes: có hai cách thức để nhóm các phần tử mô hình hóa chính thông qua Swimlanes là Pool và Lane, trong đó, Pool là biểu diễn đồ họa của một Thành phần tham gia còn Lane là một phân vùng thuộc một Process (đôi khi thuộc một Pool). Đây là linh hồn của BPMN, hiểu một cách khác: Pool thể hiện một tổ chức, một bộ phận, một phòng ban, một vai trò hay một hệ thống nào đó. Lane thể hiện là một cá nhân, một chủ thể riêng lẻ, người sẽ thực hiện các hoạt động cụ thể nào đó.

- Dữ liệu (Data): đây là thành phần quan trọng của bất cứ quy trình nào. Data được biểu diễn với bốn phần tử là Đối tượng dữ liệu (Data Object) – như là tài liệu, email, form; Đầu vào (Data Input) – dữ liệu để hoàn thành một hành động nào đó;

Đầu ra (Data Output) – dữ liệu trả ra của một hành động; Kho dữ liệu (Data Object Collection) – thể hiện một tập, một loạt hay một danh sách thông tin.

- Đối tượng kết nối (Connecting Object): Có bốn cách kết nối các Đối tượng luồng với nhau hoặc với thông tin khác, cụ thể gồm: Luồng tuần tự (Sequence Flow) – thể hiện luồng đi của quy trình; Luồng thông điệp (Message Flow) – luồng thông tin được trao đổi giữa các Lane hoặc các Pool; Liên kết (Association); Liên kết dữ liệu (Data Association).

- Artifacts: được sử dụng để cung cấp thông tin bổ sung về Quy trình. Có hai artifact tiêu chuẩn nhưng những nhà mô hình hóa hay công cụ mô hình hóa có thể tự do thêm các Artifact khi cần thiết. Hiện tại, tập artifact gồm: Group và Text Annotation.

BPMN v2.0 xác định các phần tử mô hình hóa cơ bản và ký hiệu của chúng như Bảng 1.1.

Bảng 1.1 – Danh sách các phần tử mô hình hóa cơ bản và ký hiệu

STT	Phần tử	Ký hiệu
1.	Sự kiện (event)	
2.	Hoạt động (activity)	
3.	Cổng (gateway)	
4.	Luồng tuần tự (sequence flow)	
5.	Luồng thông điệp (message flow)	
6.	Liên kết (association)	
7.	Làn/ phân vùng (pool)	
8.	Làn/ phân vùng (lane)	
9.	Đối tượng dữ liệu (data object)	

10.	Thông điệp (message)	
11.	Nhóm (group – hộp nhóm các đối tượng cùng loại)	
12.	Chú thích (notation – đi kèm liên kết)	

Ngoài các phần tử mô hình hóa cơ bản nói trên, BPMN v2.0 còn có một số phần tử mô hình hóa mở rộng, tham khảo thêm tại Mục 7.2.2 trong [11].

1.1.2.3. Các mô hình thành phần của BPMN

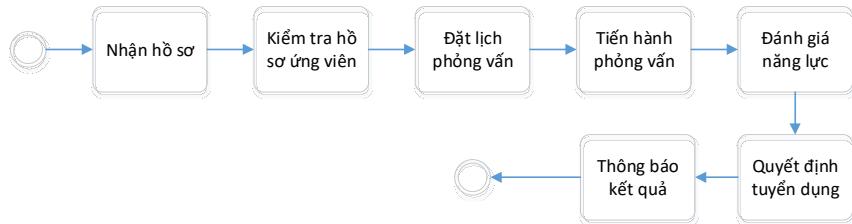
Quy trình (Process) là một khái niệm cơ bản trong BPMN. Một Process mô tả một chuỗi hay một dòng gồm nhiều Hoạt động (Activity) trong một tổ chức với mục đích thực hiện công việc. Trong BPMN, một Quy trình được mô tả là một hình ảnh về chuỗi các Phần tử (Element) chứa một tập các Hoạt động, Sự kiện (Event), Cổng (Gateway) và là chuỗi có trình tự xác định ngữ nghĩa thực thi. Các Quy trình có thể được định nghĩa ở mức độ bất kỳ, có thể là Quy trình mức cao có phạm vi toàn tổ chức hay cũng có thể là Quy trình mức thấp và được thực hiện bởi một cá nhân. Các Quy trình mức thấp có thể được nhóm lại với nhau để đạt được một mục tiêu nghiệp vụ chung.

Mô hình hóa Business Process (Quy trình nghiệp vụ) được sử dụng để truyền tải một lượng lớn các thông tin đến nhiều đối tượng người đọc khác nhau. BPMN được thiết kế bao gồm nhiều kiểu mô hình hóa và cho phép việc tạo ra các Quy trình nghiệp vụ điểm-điểm. Các phần tử có cấu trúc của BPMN cho phép nhiều người đọc có thể hiểu dễ dàng sự khác biệt giữa các phần của biểu đồ BPMN. Có 03 kiểu mô hình thành phần cơ bản trong mô hình BPMN điểm-điểm:

- Processes hay Orchestration (Điều phối), bao gồm:
 - + Quy trình nghiệp vụ riêng (nội bộ) là những quy trình nội bộ của một tổ chức cụ thể. Những quy trình này có thể được gọi chung là luồng công việc (workflow) hay quy trình BPM. Một từ đồng nghĩa thường được sử dụng trong các dịch vụ Web là Điều phối (Orchestration) các dịch vụ. Có 2 loại quy trình nghiệp vụ riêng là: Quy trình riêng không thể thực thi (là một quy trình được mô hình hóa phục vụ mục đích tài liệu hóa hành vi của quy trình ở mức chi tiết được xác định bởi người mô hình

hóa) và Quy trình nghiệp vụ riêng có thể thực thi (là một quy trình được mô hình hóa phục vụ mục đích được thực thi theo ngữ nghĩa xác định).

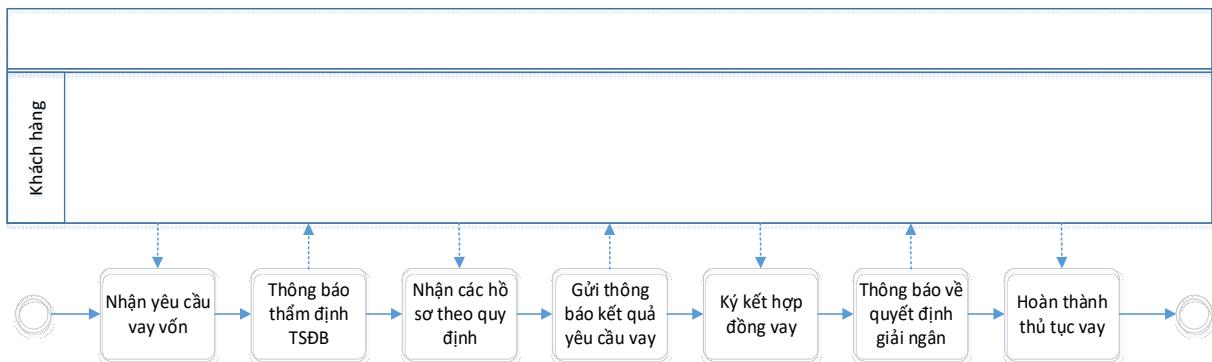
Ví dụ Hình 1.2. trình bày Quy trình nghiệp vụ riêng (nội bộ) của một công ty A về việc tuyển dụng nhân viên mới.



Hình 1.2. Minh họa về Quy trình nghiệp vụ riêng.

+ Quy trình công khai (public): thể hiện các tương tác giữa một Quy trình riêng với một quy trình khác hoặc với một thành phần tham gia (Participants). Chỉ những hoạt động (Activity) được sử dụng để giao tiếp với một thành phần tham gia khác mới được đưa vào Quy trình công khai. Tất cả các hoạt động nội bộ của quy trình riêng đều không được biểu diễn trong Quy trình công khai. Do vậy, Quy trình công khai hiển thị cho bên ngoài biết luồng thông điệp (Message Flow) và thứ tự của luồng thông điệp đó để phục vụ tương tác với chính quy trình.

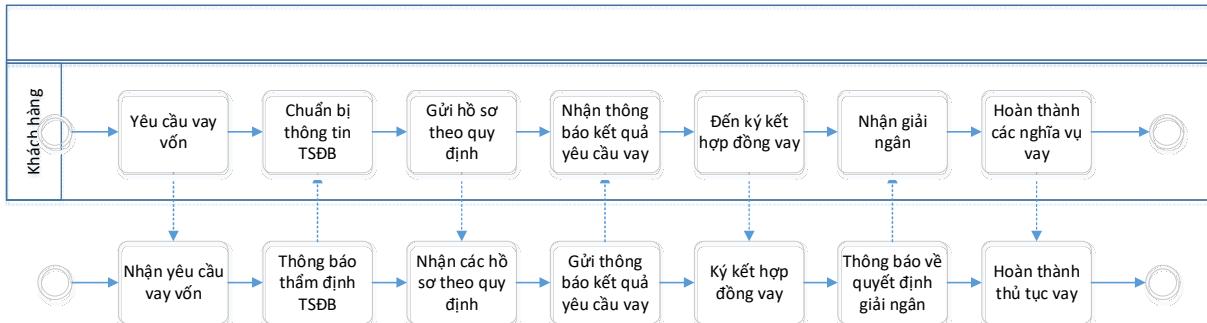
Ví dụ Hình 1.3. trình bày Quy trình nghiệp vụ công khai khi khách hàng thực hiện một yêu cầu vay tín dụng với Ngân hàng thương mại.



Hình 1.3. Minh họa quy trình nghiệp vụ công khai.

- Cộng tác (Collaborations): mô hình mô tả các tương tác giữa hai hay nhiều thực thể nghiệp vụ. Một mô hình cộng tác thường chứa hai hoặc nhiều Pool (biểu diễn đồ họa của một thành phần tham gia). Thông tin trao đổi giữa những người tham gia được thể hiện bởi một luồng thông điệp kết nối giữa hai Pool (hoặc 2 đối tượng trong Pools). Các thông điệp liên quan đến Luồng thông điệp có thể cũng sẽ được hiển thị. Mô hình cộng tác có thể hiển thị giống như hai hoặc nhiều quy trình công khai giao tiếp với nhau.

Ví dụ Hình 1.4. trình bày Quy trình nghiệp vụ cộng tác giữa khách hàng và Ngân hàng thương mại trong nghiệp vụ vay tín dụng.



Hình 1.4. Minh họa về Quy trình nghiệp vụ cộng tác.

- Choreography (Điều phối theo trình tự định sẵn với kết quả mong đợi): là một định nghĩa về hành vi được mong đợi, về cơ bản là một hợp đồng giữa các thành phần tham gia tương tác. Trong khi một mô hình Quy trình thông thường tồn tại trong một Pool thì một mô hình Choreography tồn tại giữa các Pool (hoặc các thành phần tham gia). Choreography có vẻ khá giống với một Quy trình nghiệp vụ riêng do nó bao gồm nhiều hoạt động, sự kiện và cổng (Gateway). Tuy nhiên, điểm khác biệt của một Choreography ở chỗ các hoạt động là những tương tác thể hiện một tập (một hoặc nhiều) trao đổi thông điệp liên quan đến hai hay nhiều thành phần tham gia. Ngoài ra, điểm khác biệt nữa so với một Quy trình thông thường là ở Choreography không có thành phần điều khiển trung tâm, không có thực thể chịu trách nhiệm cũng như không có người quan sát như Quy trình.

1.1.2.4. Các loại biểu đồ BPMN

Có rất nhiều loại biểu đồ về Quy trình nghiệp vụ có thể được tạo ra sử dụng BPMN v2.0, bao gồm:

- Các hoạt động Quy trình không thể thực thi được thể hiện ở mức cao;
- Quy trình nghiệp vụ có thể thực thi được thể hiện một cách chi tiết;
- Quy trình nghiệp vụ hiện tại;
- Quy trình nghiệp vụ tương lai (mục tiêu);
- Mô tả hành vi mong đợi giữa hai hay nhiều người tham gia (còn gọi là Choreography);
 - Quy trình nghiệp vụ riêng được thể hiện chi tiết (gồm cả Quy trình nghiệp vụ riêng có thể thực thi và Quy trình nghiệp vụ riêng không thể thực thi) kèm theo các tương tác với một hoặc nhiều thực thể bên ngoài (còn gọi là Quy trình hộp đen);
 - Hai hoặc nhiều Quy trình có thể thực thi tương tác được thể hiện chi tiết;

- Mỗi quan hệ chi tiết giữa Quy trình nghiệp vụ có thể thực thi với một Choreography;
- Hai hoặc nhiều Quy trình công khai;
- Mỗi quan hệ giữa Quy trình công khai với Choreography;
- Hai hay nhiều Quy trình nghiệp vụ có thể thực thi tương tác qua một Choreography;

1.2. Mô hình điều khiển truy cập

Ở phần trên, vấn đề quản lý quy trình đã được chỉ rõ trong mô hình thành phần và các quy trình của BPMN, từ đó ta có được cái nhìn nhất định. Tuy nhiên, để thực sự đóng vai trò là một hệ quản lý, hiện thực được quy trình nghiệp vụ tuân thủ đúng mục đích, vai trò thì còn thiếu một cơ chế điều khiển truy nhập đảm bảo an ninh dựa trên một số điều kiện như: Biểu diễn và quản lý được các ràng buộc về luồng công việc, gán vai trò, phân chia công việc, công việc liên quan, các ràng buộc tĩnh và ràng buộc động...

Tiếp theo, đề tài sẽ đi tới một mô hình điều khiển truy cập dựa trên thuộc tính. Cố gắng khắc phục các nhược điểm của các mô hình trước đó.

1.2.1. Khái niệm điều khiển truy cập

Cơ chế điều khiển truy cập (ACM-Access Control Mechanism) là cơ chế phụ trách việc tiếp nhận yêu cầu truy nhập từ chủ thẻ/ đối tượng tới việc quyết định, và thực thi quyết định truy nhập [7].

Những chức năng của ACM được mô tả trong logic của nhiều mô hình điều khiển truy cập. Những mô hình này thường cung cấp một khung làm việc hoặc một tập các khung điều kiện dựa trên các đối tượng, chủ thẻ, hành động và quy tắc từ đó đưa ra những quyết định điều khiển. Mỗi mô hình có ưu điểm và hạn chế nhất định, chính trong quá trình phát triển của các mô hình này đã đem đến những cải tiến tích cực, tạo nên tính linh động và uyển chuyển của mô hình ABAC.

1.2.2. Cơ chế điều khiển truy cập - MAC/DAC

Hai kỹ thuật điều khiển được áp dụng sớm nhất là Điều khiển truy cập bắt buộc (MAC - Mandatory Access Control) và Điều khiển truy cập tùy quyền (DAC - Discretionary Access Control) [7].

Hai cơ chế này có hơi khác nhau về nguyên tắc: trong khi MAC yêu cầu truy cập nghiêm khắc nhất, nó dựa trên nhãn và cấp độ, chỉ khi chủ thẻ có cấp cao hơn hoặc tương đương đối tượng mới được truy cập; ngược lại DAC lại ít hạn chế nhất, mọi

đối tượng đều có chủ sở hữu, chủ sở hữu có toàn quyền điều khiển ngay cả cấp quyền với đối tượng cho chủ thể khác.

1.2.3. Mô hình dựa trên định danh và danh sách - IBAC/ACLs

Cùng với sự phát triển của mạng (network), sự cần thiết phải hạn chế truy cập tới các đối tượng được bảo vệ đã thúc đẩy sự hình thành khả năng điều khiển truy cập dựa trên định danh (IBAC – Identity Based Access Control). IBAC sử dụng cơ chế giống như danh sách chính sách truy cập (ACLs – Access Control Lists) để bắt được định nghĩa/điều kiện cho phép truy cập tới đối tượng [7]. Nếu chủ thể đưa ra được các điều kiện phù hợp với điều kiện trong danh sách ACL thì chủ thể sẽ được gán quyền truy cập đối tượng. Mỗi quyền tương ứng mỗi hành động (đọc, ghi, sửa, xóa...) cơ bản được quản lý bởi chủ sở hữu. Mỗi đối tượng cần có danh sách ACL của riêng nó và tập quyền được gán cho mỗi chủ thể.

Trong mô hình IBAC, các quyết định ủy quyền được đưa ra trước bất kỳ yêu cầu truy cập cụ thể nào và dẫn đến việc chủ thể được thêm vào ACL. Đối với mỗi chủ thể được đặt trong ACL, chủ sở hữu đối tượng phải đánh giá danh tính, đối tượng và thuộc tính bối cảnh dựa trên chính sách điều chỉnh đối tượng và có quyết định thêm chủ thể vào ACL không. Quyết định cố định này cần có một quy trình thông báo để chủ sở hữu đánh giá lại và có thể xóa chủ thể khỏi ACL để thể hiện chủ thể, đối tượng hoặc thay đổi theo ngữ cảnh. Việc không xóa hay thu hồi quyền theo thời gian sẽ dẫn tới việc người dùng tích lũy đặc quyền.

1.2.4. Mô hình dựa trên vai trò - RBAC

Mô hình truy cập dựa trên vai trò sử dụng các vai trò được xác định trước theo một nhóm đặc quyền cụ thể được liên kết với chúng và chủ thể được gán [7]. Ví dụ, một chủ thể được gán vai trò của người quản lý sẽ có quyền truy cập vào một nhóm đối tượng khác với người được chỉ định vai trò của người phân tích.

Trong mô hình này, quyền truy cập được xác định trước một cách ngầm định bởi người gán vai trò cho từng cá nhân và rõ ràng bởi chủ sở hữu đối tượng khi xác định đặc quyền liên quan đến từng vai trò. Tại điểm của yêu cầu truy cập, cơ chế kiểm soát truy cập sẽ đánh giá vai trò được gán cho chủ thể yêu cầu quyền truy cập và tập hợp các hoạt động mà vai trò này được ủy quyền để thực hiện trên đối tượng khi kết xuất và thi hành quyết định truy cập.

Cần lưu ý rằng vai trò có thể được xem như một thuộc tính chủ thể được đánh giá bởi cơ chế kiểm soát quyền và xung quanh chính sách truy cập đối tượng này

được tạo. Khi các đặc tả RBAC trở nên phổ biến, nó nâng năng lực quản lý tập trung khả năng kiểm soát truy cập của doanh nghiệp/ tổ chức và giảm nhu cầu về ACLs.

1.2.5. Mô hình dựa trên thuộc tính - ABAC

Hai cơ chế ACL và RBAC theo cách hiểu đặc biệt cũng là trường hợp của ABAC về việc sử dụng các thuộc tính. ACL hoạt động dựa trên thuộc tính của việc “nhận định/ xác định”. RBAC lại hoạt động dựa trên thuộc tính của “Vai trò” [7].

Điểm khác biệt chính với ABAC là khái niệm về các chính sách thể hiện một bộ quy tắc logic phức tạp có thể đánh giá nhiều thuộc tính khác nhau [7]. Mặc dù có thể đạt được các mục tiêu của ABAC bằng cách sử dụng ACLs hoặc RBAC, nhưng việc chứng minh tuân thủ theo ACL thì khó khăn và tốn kém do mức độ trừu tượng cần thiết giữa một bên là yêu cầu điều khiển truy cập (AC) đối với mô hình ACL hoặc mô hình RBAC. Một vấn đề khác của mô hình ACL hoặc RBAC là nếu thay đổi yêu cầu AC có thể khó xác định tất cả các vị trí triển khai ACL hoặc RBAC cần cập nhật.

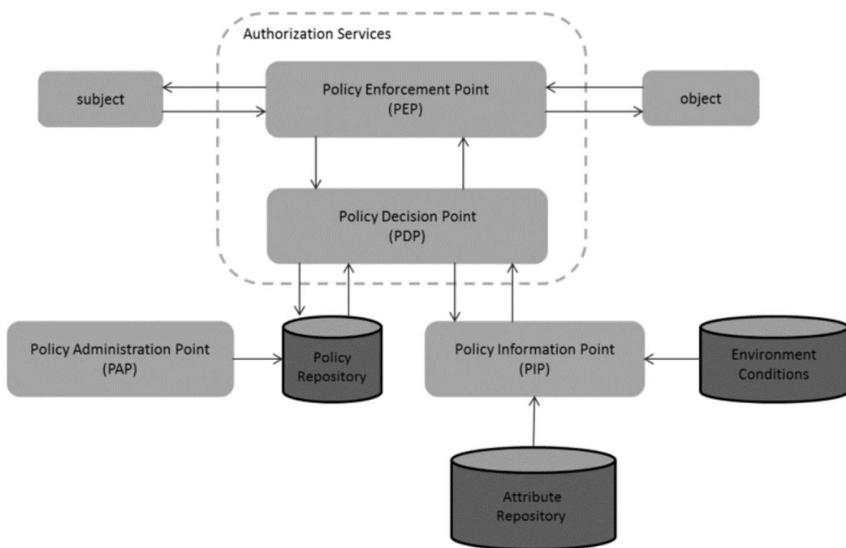
Một ví dụ về khung điều khiển truy cập với ABAC là Ngôn ngữ đánh dấu kiểm soát truy cập mở rộng (XACML - Extensible Access Control Markup Language). Mô hình XACML sử dụng các yếu tố như: quy tắc (rules), chính sách (policies), thuật toán kết hợp quy tắc-chính sách, các thuộc tính (chủ đề, đối tượng hoặc tài nguyên, điều kiện hành động và môi trường), nghĩa vụ... Kiến trúc tham chiếu của nó gồm các chức năng để điều khiển truy cập trong Hình 1.5.:

Điểm quyết định chính sách (PDPs - Policy Decision Points)

Điểm thực thi chính sách (PEPs - Policy Enforcement Points)

Điểm quản trị chính sách (PAPs - Policy Administration Points)

Điểm thông tin chính sách (PIPs - Policy Information Points)



Hình 1.5. Ví dụ về chức năng các điểm kiểm soát truy cập.

Nói chung là mô hình ABAC đã tránh khả năng gán trực tiếp các cặp tương quan (thao tác – đối tượng) cho chủ thể yêu cầu hoặc vai trò hoặc nhóm vai trò trước khi các yêu cầu được đưa ra. Thực tế, khi một chủ thể yêu cầu quyền truy cập, cơ chế ABAC có thể đưa ra quyết định kiểm soát truy cập dựa trên một loạt các tập: các thuộc tính được chỉ định của người yêu cầu, các thuộc tính được gán của đối tượng, các điều kiện môi trường và một chính sách được chỉ định theo các thuộc tính và điều kiện vừa nêu. Nhờ sự sắp xếp này, các chính sách được tạo ra và quản lý mà không cần tham chiếu trực tiếp đến nhiều đối tượng/ người dùng không liên quan.

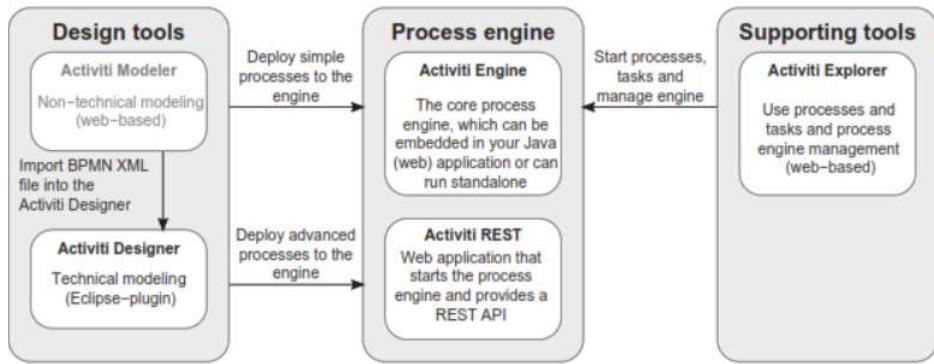
1.3. Bộ công cụ hỗ trợ Activiti

Trong phần trước của chương, quy trình nghiệp vụ và cách thức mô hình quy trình đã được đề cập một cách tổng quan. Trong doanh nghiệp và tổ chức, thực tế cần thiết là triển khai và thực thi được quy trình nghiệp vụ. Để giải quyết vấn đề đó, luận văn tiếp cận tới một phương pháp là thực nghiệm trên công cụ hỗ trợ mã nguồn mở Activiti.

1.3.1. Mô tả tổng quan

Thành phần cốt lõi của khung công cụ Activiti là engine xử lý. Engine này cung cấp các khả năng thực thi quy trình của ký hiệu mô hình hóa quy trình nghiệp vụ (BPMN – Business Process Model and Notation) 2.0, tạo mới các luồng làm việc và nhiều thành phần khác. Dự án Activiti bao gồm nhiều công cụ xoay quanh và hỗ trợ Activiti Engine [6]. Hình 1.6. sẽ cho ta cái nhìn tổng quan về bộ công cụ này với

trung tâm là engine xử lý, hai bên là các công cụ hỗ trợ về mô hình hóa (modeling), thiết kế (design) và quản lý:



Hình 1.6. Tổng quan công cụ Activiti.

Các thành phần của công cụ Activiti được mô tả chi tiết trong Bảng 1.2.

Bảng 1.2 - Các thành phần trong công cụ Activiti

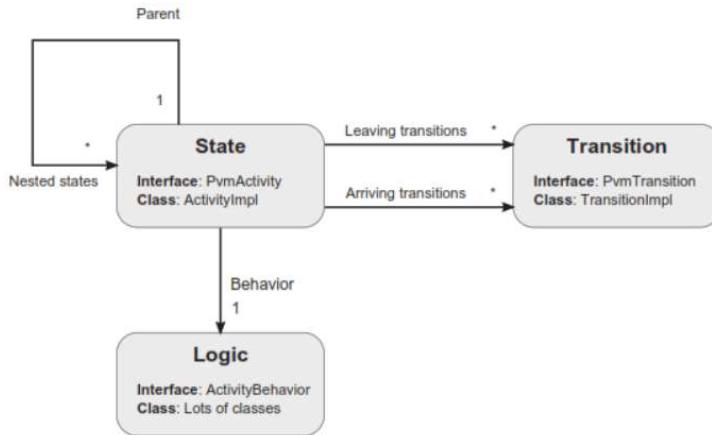
Thành phần	Mô tả
Activiti Engine	Thành phần cốt lõi của bộ công cụ, chịu trách nhiệm thực thi quy trình xử lý như thực thi quy trình BPMN 2.0, tạo luồng công việc
Activiti Modeler	Môi trường web để tạo mô hình BPMN
Activiti Designer	Một plugin Eclipse, sử dụng cho việc thiết kế quy trình BPMN để bổ sung vào Activiti, thậm chí cả việc kiểm thử bộ phận (unit test)
Activiti Explorer	Ứng dụng web cho người dùng thao tác như: tạo mới quy trình, xem/ thực hiện task được gán...
Activiti REST	Ứng dụng web cung cấp interface REST giao tiếp với Activiti Engine

1.3.2. Cơ chế thực thi - Activiti Engine

Activiti là một khung công cụ xử lý BPMN v2.0 thực hiện đặc tả BPMN v2.0. Nó có thể triển khai các định nghĩa quy trình, khởi tạo quy trình, thực thi tác vụ của người dùng và thực hiện các chức năng BPMN v2.0 khác [1] ...

Đặc tính nổi bật của Activiti là cơ chế quản lý trạng thái. Một quy trình BPMN v2.0 sẽ bao gồm nhiều yếu tố như các sự kiện (events), nhiệm vụ (tasks) và các cổng điều hướng (gateways) được nối với nhau qua các chuỗi tuần tự (arrows). Khi deploy một quy trình như vậy, các phần tử BPMN 2.0 sẽ được thực thi tuần tự. Quá trình thực thi này qua các bước sẽ được quản lý trạng thái —các trạng thái hoạt động - và

dựa trên các điều kiện sẽ có chuyển đổi giữa các trạng thái theo hướng luồng (arrows). Cơ chế này được thể hiện minh họa trong Hình 1.7. như sau:



Hình 1.7. Minh họa chuyển trạng thái trong Activiti Engine.

Trong Activiti Engine, hầu hết các thành phần của BPMN 2.0 đều được triển khai dưới dạng trạng thái. Chúng kết nối theo các chuỗi tuần tự, mỗi trạng thái hoặc phần tử BPMN 2.0 đều có logic thực thi và có trạng thái riêng. Như hình trên, giao diện logic ActivityBehavior được thực hiện bởi rất nhiều lớp logic. Đây là một công cụ hữu hiệu cho phép thực hiện hàng loạt các task (nhiệm vụ) của quy trình. Từ một nền tảng hỗ trợ BPMN v2.0, Activiti đã được mở rộng rất nhiều tính năng và khả năng như: cấu hình với JTA, Sprint, thực thi tốc độ nhanh, cơ chế (Engine) đơn giản dễ dàng tích hợp, hỗ trợ thực thi bất động bộ. Đặc biệt, Activiti có khả năng mở rộng tốt với việc tích hợp điện toán đám mây và hỗ trợ kiểm thử quá trình thực thi quy trình (execute process). Công cụ cho phép lựa chọn uyển chuyển giữa Engine API hoặc REST API, các luồng được thực hiện như các services...

1.3.3. Một số ưu và nhược điểm của công cụ Activiti

Activiti là một trong những nền tảng công cụ và phần mềm công việc mã nguồn mở tốt nhất được thiết kế đặc biệt cho các tổ chức kinh doanh và nhà phát triển. Nó rất nhẹ và kết hợp với một công cụ xử lý BPMN 2.0 siêu nhanh cho Java [13].

Ngoài ra, Activiti được tối ưu hóa rất nhiều để xử lý các khía cạnh kỹ thuật và phi kỹ thuật cụ thể là phân tích, mô hình hóa, tạo khả năng tương thích quy trình kinh doanh và hỗ trợ và tạo phần mềm tương ứng.

Trên thế giới mã nguồn mở, có rất nhiều nền tảng được chia sẻ và phát triển, có thể kể tới BonitaSoft, Red Hat Jboss BPM, Adobe LifeCycle, Modelio, Camunda, Orchestra, jBPM hay Jobget, jSonic BPM... Ta sẽ xem thông tin giữa Activiti và hai đối thủ lớn nhất của nó là Jboss BPM/jBPM và BonitaSoft. Jboss BPM hay jBPM là

nền tảng mở đầu tiên cho phép tùy chỉnh ngôn ngữ xử lý. BonitaSoft, một nền tảng mở cho BPMN2.0 với một tập cực lớn các thành phần và khả năng tích hợp cực tốt.

1.3.3.1. So sánh Activiti và JBoss BPM

Hai nền tảng này có rất nhiều điểm chung, thậm chí có thể nói Activiti còn là sản phẩm phát triển nên từ Jboss BPM (JBPM) [6]. Tuy nhiên các tính năng khác biệt lớn giúp ta nhận dạng được trong Bảng 1.3 là:

Bảng 1.3 - Các điểm khác nhau giữa Activiti và JBoss BPM

Mô tả	Activiti	jBPM
Cộng đồng mở	Cộng đồng từ nhân sự của tập đoàn Alfresco, các công ty SpringSource, FuseSoft, MulSoft và cộng đồng lập trình viên cá nhân...	Từ nhân sự của Jboss và cộng đồng lập trình viên cá nhân...
Hỗ trợ nền tảng Spring	Hỗ trợ hoàn toàn Spring giúp cho việc quản lý cực đơn giản	Không hoàn toàn hỗ trợ Spring, phải dùng add-in
Hỗ trợ nguyên tắc nghiệp vụ	Đã tích hợp với Drools engine hỗ trợ nguyên tắc nghiệp vụ (business rules)	Hỗ trợ tích hợp Drools engine tại mức độ nhất định
Công cụ hỗ trợ	Các quá trình mô hình hóa, thiết kế, sử dụng (chạy theo quy trình) đều được hỗ trợ dễ dàng trên nền tảng Web	Có hỗ trợ các quá trình nhưng trên giao diện Form
Dự án mã nguồn	Được cộng đồng liên tục ra mắt phiên bản mới (2 tháng/lần). Dự án được tách biệt làm 03 phần nhỏ: Engine, Designer và ứng dụng REST	Việc cập nhật phiên bản không thường xuyên. Các dự án tổ chức theo dạng plug-in cho Eclipse

1.3.3.2. So sánh Activiti và BonitaSoft

Luận văn đưa ra một số điểm khác biệt giữa hai nền tảng này:

Thứ nhất, Activiti là nền tảng hướng tới lập trình viên, cung cấp các hàm Java API rất dễ dàng để tích hợp với Activiti Engine. BonitaSoft theo hướng công cụ nhiều hơn, hỗ trợ việc kéo-thả để thao tác là chính.

Tiếp đó, với Activiti lập trình viên hoàn toàn kiểm soát tới từng dòng mã nguồn – code – của mình. Trong khi đó BonitaSoft, các dòng mã nguồn – code lại được tự sinh do công cụ.

Cuối cùng, BonitaSoft cung cấp sẵn cực kỳ nhiều các tùy chọn kết nối cho các hệ thống khác. Trong khi Activiti tập trung hướng vào lập trình viên, hỗ trợ hai cơ chế tích hợp là Mule và Camel.

1.3.3.3. *Tóm lược công cụ Activiti*

Như các so sánh trên, Activiti không phải công cụ mã nguồn mở duy nhất hỗ trợ tiêu chuẩn BPMN v2.0. Công cụ cũng có các nhược điểm nhất định như: người dùng phải tự lập trình và quản lý từng mã nguồn, việc tích nối kết hợp cần sự hiểu biết tương đối về nền tảng Java, chưa hỗ trợ cơ chế kéo-thả để thao tác... Tuy nhiên với các tiêu chí mã nguồn rõ ràng, tính linh động, tùy chỉnh mạnh mẽ, dễ tích hợp, Activiti giúp ta dễ dàng nhúng vào ứng dụng, kiểm soát và chạy trên mọi nền tảng mà ta mong muốn.

1.4. Tổng kết chương

Các kiến thức nền tảng trong luận văn đã được tóm lược trong chương này. Trước tiên, các khái niệm và thông tin về tiêu chuẩn ký hiệu mô hình hóa BPMN được đề cập. Mục đích là việc hiểu rõ tính quan trọng của việc mô hình hóa, để có thể đem tới cái nhìn chung nhất, thống nhất giữa các bộ phận nghiệp vụ bởi nó đại diện cho một thứ duy nhất: “Thể hiện được quy trình nghiệp vụ”. Tiếp theo, là việc nhắc tới các cơ chế, mô hình điều khiển truy cập. Mỗi mô hình sẽ cho biết ưu điểm, nhược điểm trong việc kiểm soát truy cập của người sử dụng trong hệ thống nhằm đảm bảo đúng vai trò của mỗi chủ thể trong quy trình nghiệp vụ, trong đó đặc biệt nhấn mạnh tới mô hình điều khiển dựa trên thuộc tính. Cuối cùng, công cụ hỗ trợ Activiti, một công cụ hỗ trợ quy trình và cơ chế kiểm soát truy cập được sử dụng trong luận văn cũng được đề cập tới.

CHƯƠNG 2

PHƯƠNG PHÁP XÂY DỰNG MÔ HÌNH ABAC VÀ CÔNG CỤ HỖ TRỢ

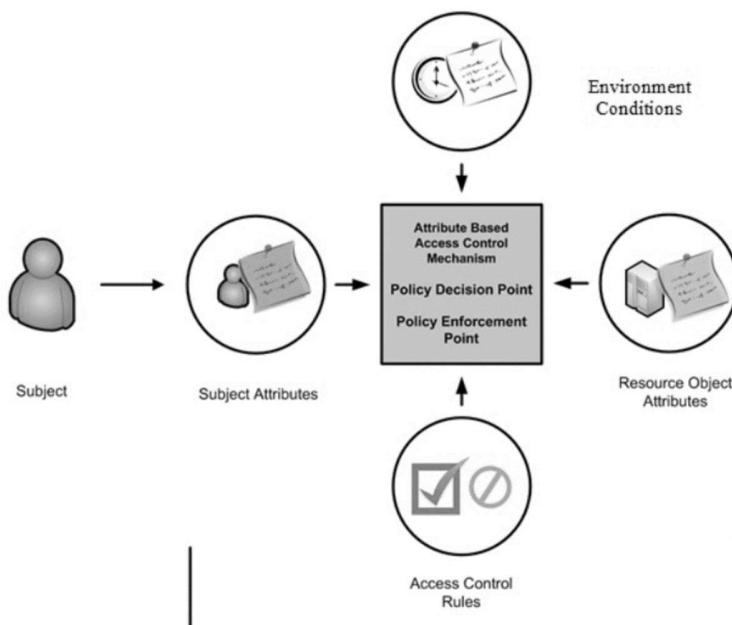
Chương này tập trung đề cập chi tiết tới mô hình ABAC và việc phát triển công cụ hỗ trợ, thông qua việc tùy biến và điều chỉnh trong bộ công cụ Activiti. Việc chỉnh sửa công cụ Activiti cũng sẽ mô tả chi tiết thông tin liên quan tới quy trình nghiệp vụ trong ngân hàng là nghiệp vụ cụ thể thực tế.

2.1. Mô hình điều khiển truy cập ABAC

Chương trước, luận văn đã trình bày tổng quan về mô hình điều khiển truy cập theo thuộc tính ABAC. Để giải quyết bài toán thực nghiệm và tích hợp được mô hình điều khiển này vào công cụ trước tiên cần định nghĩa được mô hình điều khiển này.

2.1.1. Cơ chế điều khiển trong mô hình ABAC

Về cơ bản, ABAC dựa vào việc đánh giá các thuộc tính của chủ thể, thuộc tính của đối tượng, điều kiện môi trường và các mối quan hệ, các quy tắc/chính sách, các hoạt động để định nghĩa hoạt động cho phép [6]. Tất cả các giải pháp ABAC đều chứa các tính chất cốt lõi cơ bản này để đánh giá các thuộc tính và thực thi các quy tắc hoặc mối quan hệ giữa các thuộc tính đó.



Hình 2.1. Cơ chế cốt lõi của ABAC.

Trong Hình 2.1. khi một truy cập điều khiển được tạo ra, quy tắc điều khiển truy cập và thuộc tính sẽ được đánh giá bởi “Cơ chế điều khiển truy cập dựa trên thuộc

tính” nhằm cung cấp một quyết định kiểm soát truy cập. Trong mô hình đơn giản nhất của ABAC, cơ chế điều khiển này bao gồm 02 điểm chính sách là PDPs và PEPs.

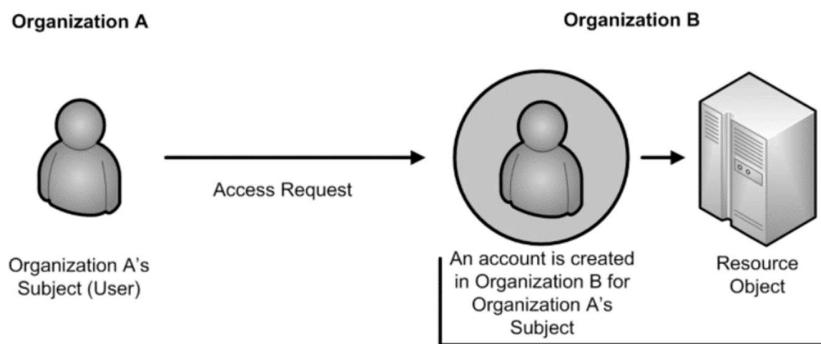
Không giống như mô hình RBAC dựa trên vai trò hay định danh người dùng để tạo nên mức độ kiểm soát, ABAC kết hợp cả chính sách quản trị và kiểm soát. Mô hình ABAC không chỉ định nghĩa “AI” được quyền truy cập “CÁI GÌ” mà còn kiểm soát (định nghĩa quy tắc kiểm soát) trên các điều kiện ngữ cảnh như “KHI NÀO”, “Ở ĐÂU”, “TẠI SAO” và “NHƯ THẾ NÀO” bởi một tập các thông tin trên chủ thẻ và các quan hệ giữa các chủ thẻ [3].

Mô hình ABAC có thể định nghĩa các truy cập dựa trên các đặc tính liên quan đến an ninh, hay gọi là các thuộc tính có thể chia thành 03 nhóm: thuộc tính của chủ thẻ, thuộc tính của tài nguyên, thuộc tính của môi trường:

- ✓ Thuộc tính của chủ thẻ (Subject Attributes): một chủ thẻ là một đối tượng thực hiện một hành động trên một tài nguyên nào đó. Mỗi chủ thẻ sẽ có các thuộc tính liên quan giúp xác định định danh và các đặc tính của chủ thẻ, ví như: mã số cá nhân, họ tên, nơi công tác, vị trí công tác... Do vậy vai trò của một chủ thẻ được coi như một thuộc tính của chủ thẻ đó.
- ✓ Thuộc tính của tài nguyên (Resource Attributes): một tài nguyên là một thực thể (ví như một web service, một cấu trúc dữ liệu, một cơ sở dữ liệu...) được sử dụng bởi một chủ thẻ. Tương tự như một chủ thẻ, một tài nguyên cũng có các thuộc tính nhằm bổ sung thêm thông tin cần thiết cho quá trình kiểm tra và đưa ra các quyết định về truy nhập.
- ✓ Thuộc tính của môi trường (Environment Attributes): giúp mô tả các khía cạnh cần thiết của môi trường hoạt động hay ngữ cảnh mà trong đó các yêu cầu truy cập xuất hiện. Các khía cạnh này rất đa dạng như: vận hành, kỹ thuật, thời gian, địa điểm...

2.1.2. Ưu điểm của mô hình ABAC

Trong nhiều hệ thống kiểm soát truy cập – AC (Access Control), các giải pháp kiểm soát truy cập logic chủ yếu dựa trên danh tính của một chủ th yêu cầu thực hiện một thao tác nào đó (như đọc/xem) tên một đối tượng (như một file). Ví dụ như IBAC hay RBAC, quyền truy cập tới đối tượng xác định được gán đơn lẻ hoặc khi quyền truy cập đã được cấp cho các vai trò của chủ thẻ. Cách tiếp cận này với AC thường khó quản lý. Trong ví dụ Hình 2.2. minh họa dưới đây là sự truy cập chéo giữa các tổ chức, việc xác thực các chủ thẻ bên ngoài đối tượng được khởi tạo trước và được điền trước vào danh sách truy cập.



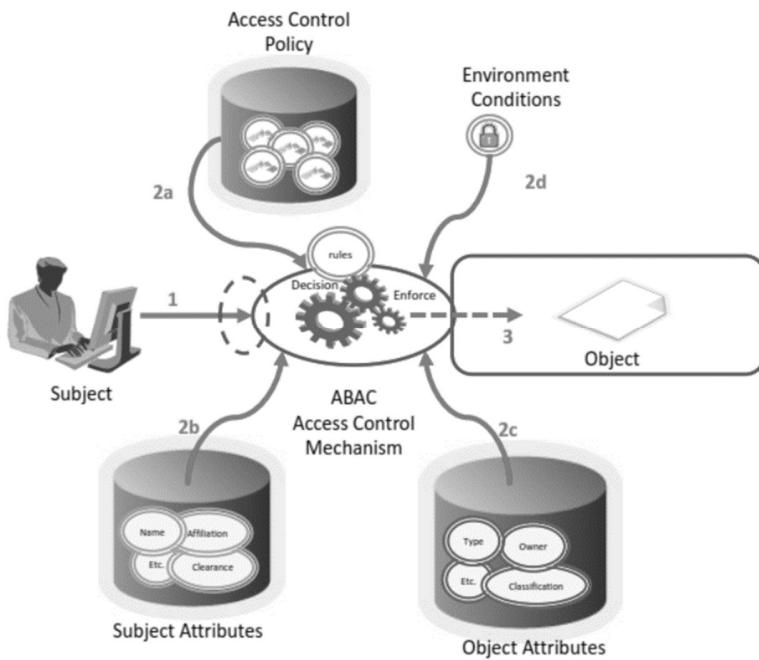
Hình 2.2. Ví dụ về truy cập chéo.

Ngoài ra, khi phân loại chủ thể như việc xác định danh tính và vai trò thường không đủ để thể hiện nhu cầu AC trong thế giới thực. Mô hình RBAC đưa ra quyết định dựa trên sự liên kết của chủ thể với vai trò [3]. RBAC không hỗ trợ quyết định “đa yếu tố” (như vị trí, chuyên ngành khi truy cập vào hồ sơ). Vai trò trong RBAC được gán dựa trên các “yếu tố tĩnh”, đây là thách thức bởi thực tế nhiều trường hợp cần các quyết định kiểm soát truy cập mang tính động. Do đó, cần thiết có cơ chế đưa ra quyết định AC khi mà không có kiến thức hay hiểu biết trước đó về chủ thể đưa ra yêu cầu truy cập.

Bằng cách dựa vào các khái niệm thuộc tính của chủ thể và đối tượng, ABAC tránh được xác thực chủ thể trước khi cấp quyền. Hơn nữa, mô hình này linh hoạt ở chỗ cho phép một doanh nghiệp/ tổ chức lớn đỡ tốn thời gian và giảm độ phức tạp khi quản lý một danh sách hay vai trò đồ sộ (cho một lượng lớn người sử dụng/ chủ thể).

2.2. Thiết kế mô hình ABAC

Ở mức tổng quát mô hình ABAC, như minh họa trong Hình 2.3., cơ chế kiểm soát truy cập sẽ xác định hành động nào của chủ thể có thể thực hiện đối với đối tượng gồm 03 bước [7].



Hình 2.3. Kịch bản cơ bản của mô hình ABAC.

Bước 1: Chủ thể yêu cầu truy cập vào đối tượng.

Bước 2: Cơ chế kiểm soát truy cập sẽ đánh giá 4 thành phần. Thứ nhất là các quy tắc – Rules, là những ràng buộc được quy định trước đó. Thứ hai là các thuộc tính của chủ thể – Subject Attributes, thể hiện những yếu tố liên quan đến chủ thể yêu cầu truy cập. Thứ ba là các thuộc tính đối tượng – Object Attributes, thể hiện những yếu tố liên quan đến đối tượng được truy cập tới. Cuối cùng là điều kiện môi trường, là các yếu tố liên quan tới thông số hệ thống. Từ đó sẽ đưa ra quyết định.

Bước 3: Chủ thể sẽ được truy cập đối tượng nếu như được xác thực đủ điều kiện.

Định nghĩa 1. (Mô hình ABAC) Một mô hình ABAC là một bộ gồm các thành phần $\{S, R, E, A_S, A_R, A_E, P\}$:

S: tập các chủ thể (subjects); $S = \{s_1, s_2, \dots, s_n\}$

R: tập các tài nguyên (resources); $R = \{r_1, r_2, \dots, r_m\}$

E: tập các yếu tố môi trường (environments); $E = \{e_1, e_2, \dots, e_k\}$

A_S, A_R, A_E : lần lượt là tập các thuộc tính của các chủ thể, các tài nguyên và của các yếu tố môi trường.

A_S : tập thuộc tính như user-name, roles, certificate (chứng chỉ)

A_R : tập thuộc tính như workflow-name, tasks, roles

A_E : tập thuộc tính như current-date, current-time, concurrent-user, max-thread

P: tập các quy tắc biểu diễn cho các chính sách (policies) của hệ thống.

Trong mô hình ABAC, mỗi quy tắc (rule) được xác định một điều kiện nếu thỏa mãn thì một chủ thể sẽ được truy cập vào một tài nguyên (đối tượng) trong một môi trường (hệ thống) cụ thể. Điều đó đồng nghĩa chủ thể (người sử dụng) chỉ có thể truy cập vào tài nguyên trong điều kiện môi trường nếu nó thỏa mãn quy tắc. Biểu thức biểu diễn quy tắc là:

r: granted (s, r, e) $\leftarrow A_1, A_2, \dots, A_n$ ($n \geq 0$)

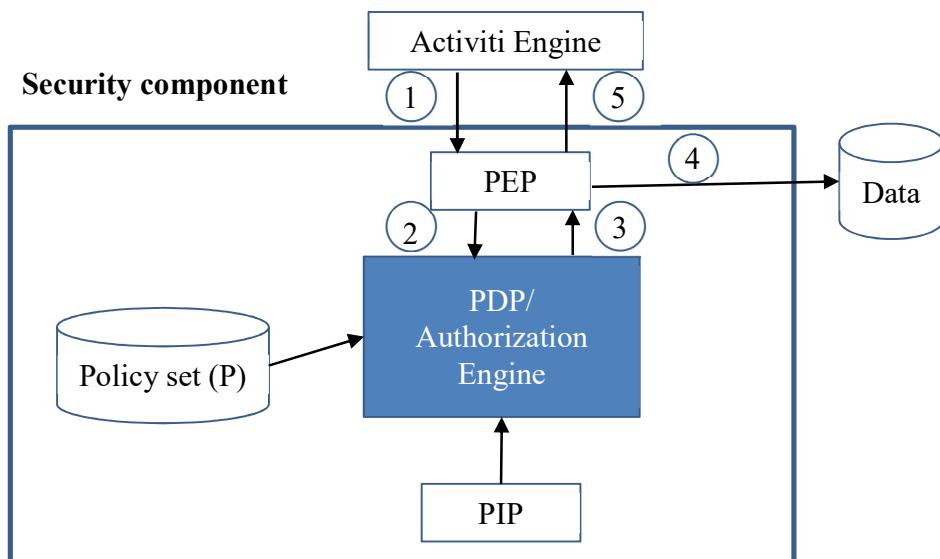
Trong đó: s, r, e lần lượt tương ứng cho một chủ thể, một tài nguyên và một yếu tố môi trường; A_1, A_2, \dots, A_n là các toán hạng logic.

Nếu giá trị về phải của biểu thức trên là đúng (TRUE) thì chủ thể s được gán quyền truy cập tài nguyên r trong ngữ cảnh e. Trong hàm trên có quy tắc, giá trị của các tham số s, r, e có thể NULL (ký hiệu là “_”) với ý nghĩa là với mọi giá trị. Ví dụ:

granted (s, r, _): chủ thể s có thể truy cập tài nguyên r trong mọi ngữ cảnh

granted (_, r, _): mọi chủ thể có thể truy cập tài nguyên r trong mọi ngữ cảnh.

Từ các yêu cầu của mô hình ABAC, ta đưa ra thiết kế chi tiết cho các module của mô hình, được minh họa như Hình 2.4.



Hình 2.4. Thiết kế chi tiết mô hình.

Thành phần trong thiết kế này là “Security component”, theo tiêu chuẩn kiến trúc của hệ thống ABAC – XACML – bao gồm 3 modules:

PEP (Policy Enforcement Point): module thực hiện các điều khiển truy cập gồm một số bước. Đầu tiên, (1) nó được gọi từ thân hàm Activiti Engine, sau đó nó chuyển sang cho module PDP (2).

PDP (Policy Decision Point): module đánh giá dựa các chính sách an ninh (P) sau khi tiếp nhận yêu cầu từ PEP. Sau khi có quyết định (từ chối, chấp nhận hay cấp quyền nào đó), PDP trả về cho PEP (3).

PIP (Policy Information Point): module này là một nguồn giá trị cho các thuộc tính, nó là một tập các hàm của hệ thống.

Tóm lại, các bước trong mô hình được diễn đạt như sau:

- (1) Từ một tiến trình trong Activiti Engine sẽ gọi tới thành phần an ninh, qua PEP
- (2) PEP tiếp nhận thông tin và lấy các thông tin cần thiết khác
- (3) PDP đưa ra quyết định trả lời cho PEP, quyết định này là chấp nhận gán quyền hoặc không. Nếu chấp nhận, thông tin sẽ được lưu trữ trong Data (4). Nếu từ chối sẽ trả lại trạng thái không chấp nhận.
- (4) Lưu thông tin trong trường hợp chấp nhận gán quyền truy cập.
- (5) Trả lại thông tin từ chối, kết thúc xử lý.

2.3. Tích hợp mô hình ABAC vào công cụ Activiti

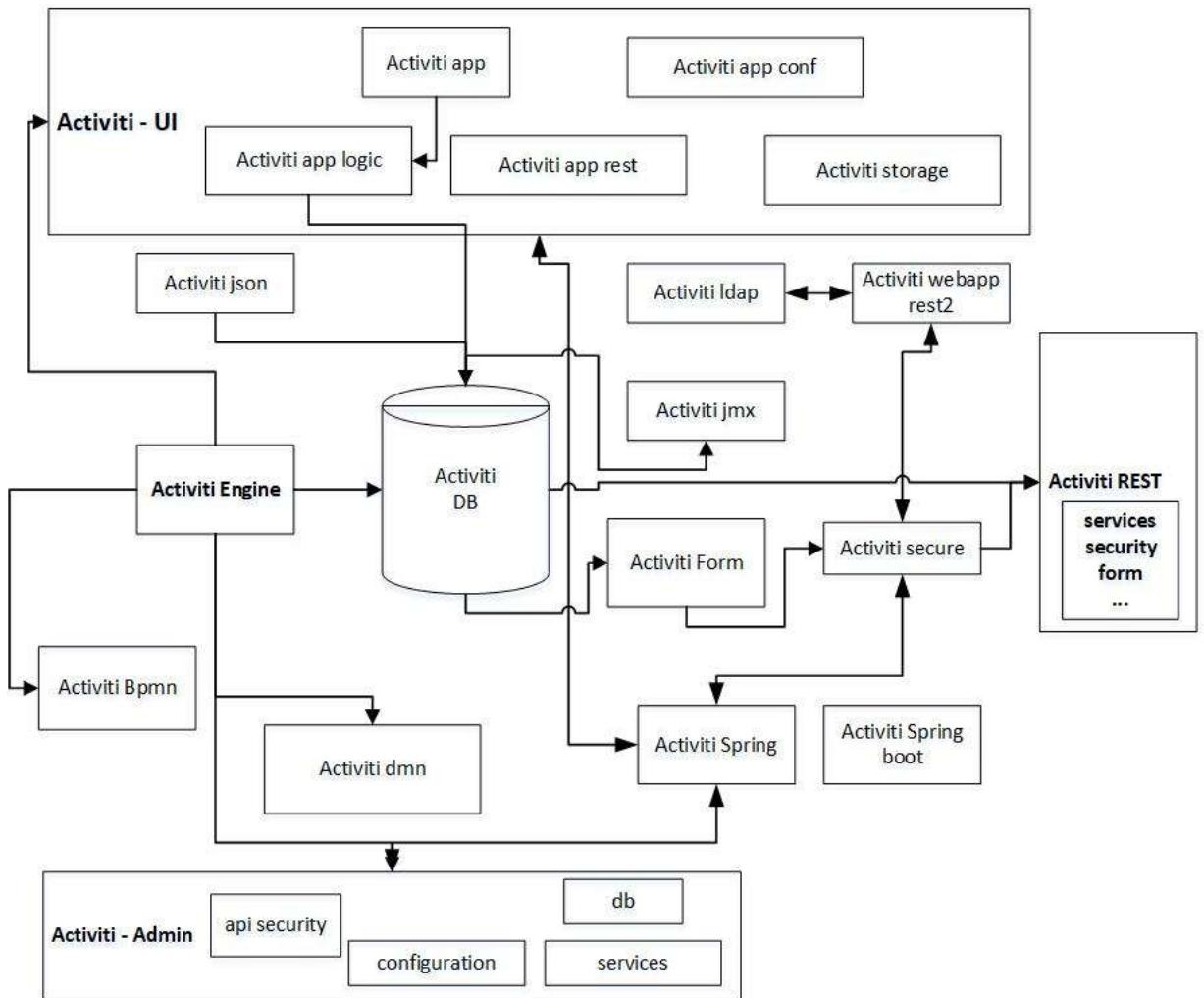
Sau khi định nghĩa mô hình điều khiển được trình bày rõ. Luận văn trình bày chi tiết các thao tác cần thiết phục vụ việc cài đặt, tối mức hoàn thiện thực thi mô hình trên công cụ Activiti.

2.3.1. Cơ chế hoạt động của công cụ Activiti

2.3.1.1. Các thành phần chính công cụ Activiti

Để hiểu rõ hơn việc phát triển trên công cụ Activiti, ta hãy nghiên cứu các thành phần (modules) của bộ công cụ này.

Dưới góc nhìn sâu hơn về kỹ thuật, lập trình viên có thể hiểu được luồng xử lý chính và giao tiếp giữa các thành phần. Qua đó thêm cơ sở để nắm bắt công nghệ và làm chủ công cụ.



Hình 2.5. Các thành phần bộ công cụ Activiti.

Trong Hình 2.5. kiến trúc phần mềm của công cụ được tổ chức với 05 thành phần rõ ràng: Database, Activiti Engine, Activiti Admin, Activiti REST và Activiti UI.

Cơ sở dữ liệu – database chịu trách nhiệm lưu trữ toàn bộ các thông tin liên quan tới cấu hình hệ thống, phiên bản (version) của công cụ. Ngoài ra, phần dữ liệu quan trọng nhất liên quan tới mô hình và quá trình vận hành :

- ✓ Dữ liệu định nghĩa mô hình (process)
- ✓ Dữ liệu quản lý phiên bản mô hình
- ✓ Dữ liệu định nghĩa, quản lý Form (đối tượng) sử dụng trong mô hình
- ✓ Dữ liệu vận hành/ quản lý trạng thái các bước thực thi trong mô hình
- ✓ Dữ liệu vận hành quá khứ của mô hình

Ngoài ra trong cơ sở dữ liệu có các thông tin quản trị khác như người sử dụng (user), nhóm người sử dụng (group), vai trò (role).

Thành phần tiếp theo, Activiti Engine như đã đề cập ở phần trước. Đây là trái tim của bộ công cụ Activiti, vận hành theo cơ chế quản lý trạng thái.

Thành phần Activiti Admin dạng web-based, kết nối qua REST, nhằm mục đích quản trị. Thành phần này cung cấp rất nhiều màn hình, góc nhìn cho phép theo dõi toàn bộ các thành phần liên quan tới mô hình đang hoạt động trong hệ thống. Đồng thời cho phép thiết lập một số tùy biến như chạy các công việc (jobs) định kỳ.

Người sử dụng (end-users) sẽ làm việc trực tiếp với hệ thống qua thành phần Activiti UI, hay còn gọi là giao diện ứng dụng. Các tính năng nổi bật:

- ✓ Quản lý người sử dụng (users management) : quản lý thông tin cá nhân, quản lý danh sách người sử dụng có trong hệ thống, quản lý danh sách nhóm người sử dụng trong hệ thống, phân quyền người sử dụng tới các nhóm quyền.
- ✓ Quản lý mô hình quy trình nghiệp vụ (business process models): quản lý quy trình, quản lý các form nhập liệu, quản lý instance của các quy trình. Các chức năng như tạo mới quy trình, tạo form sử dụng cho quy trình, cài đặt (publish processes) quy trình cho người sử dụng ; đồng thời với đó là chức năng chỉnh sửa, quản lý phiên bản (version).
- ✓ Quản lý quá trình thực thi quy trình như : quản lý các nhiệm vụ (tasks) của người sử dụng và quản lý trạng thái các quy trình (processes). Người sử dụng có thể theo dõi danh sách tasks của bản thân qua các tiêu chí: task được gán, task liên quan và task mà người sử dụng là ứng viên (candidate). Các processes liên quan tới người sử dụng cũng được quản lý qua các tiêu chí: đang mở, đang thực hiện hay đã hoàn thành.

Module Activiti REST có nhiệm vụ như cầu nối Activiti Engine với bên ngoài. Trong bộ công cụ ban đầu, chức năng cơ bản của REST là kết nối hai thành phần Activiti Admin và Activiti UI.

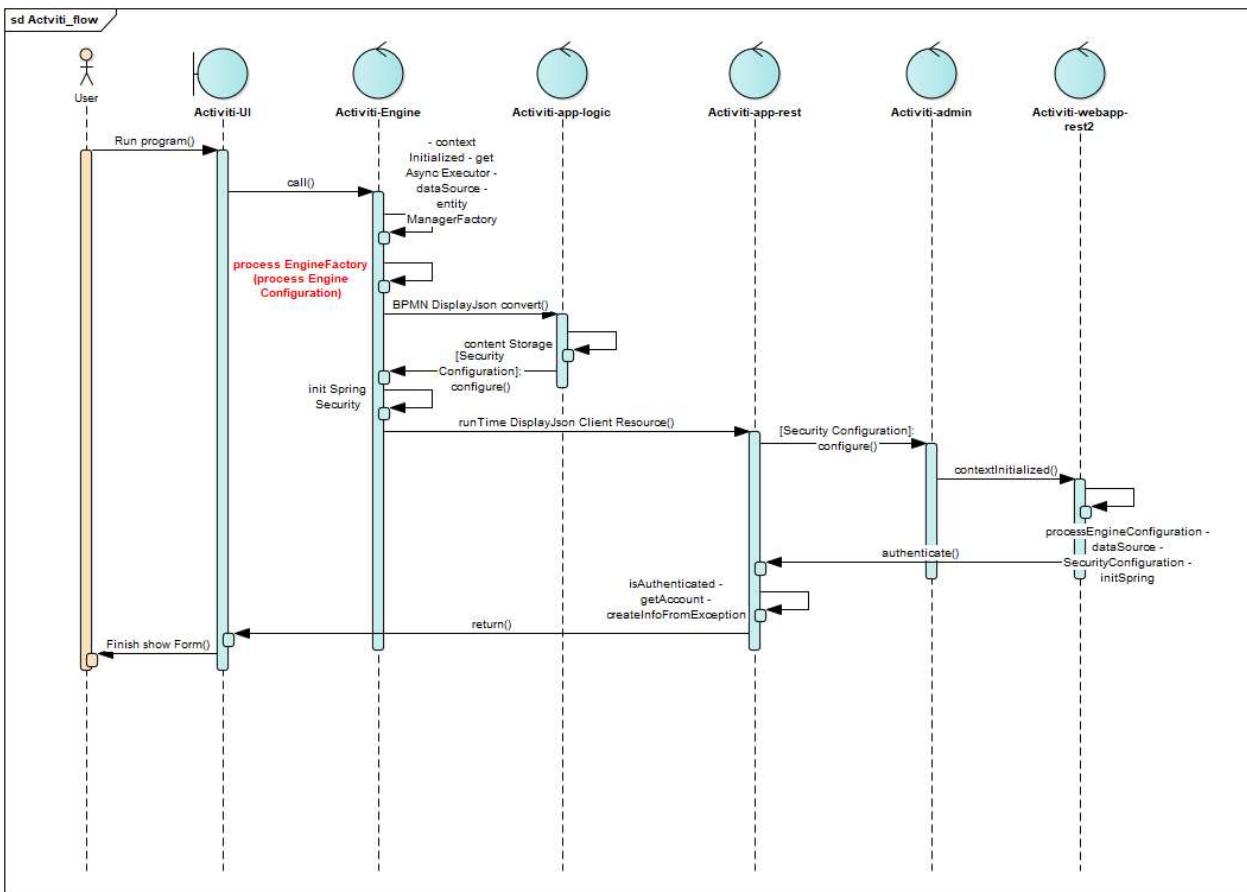
Ngoài ra, còn có rất nhiều thành phần hỗ trợ khác phục vụ cho từng mục đích, lần lượt là các kỹ thuật json, security, ldap, Spring. Trong phạm vi luận văn thực hiện can thiệp và mở rộng công cụ trong module Activiti UI.

2.3.1.2. *Module Activiti UI*

Tiếp theo, chúng ta sẽ đi sâu hơn thành phần Activiti UI của bộ công cụ Activiti để biết điểm bắt đầu (breakpoint) cho việc mở rộng công cụ hỗ trợ.

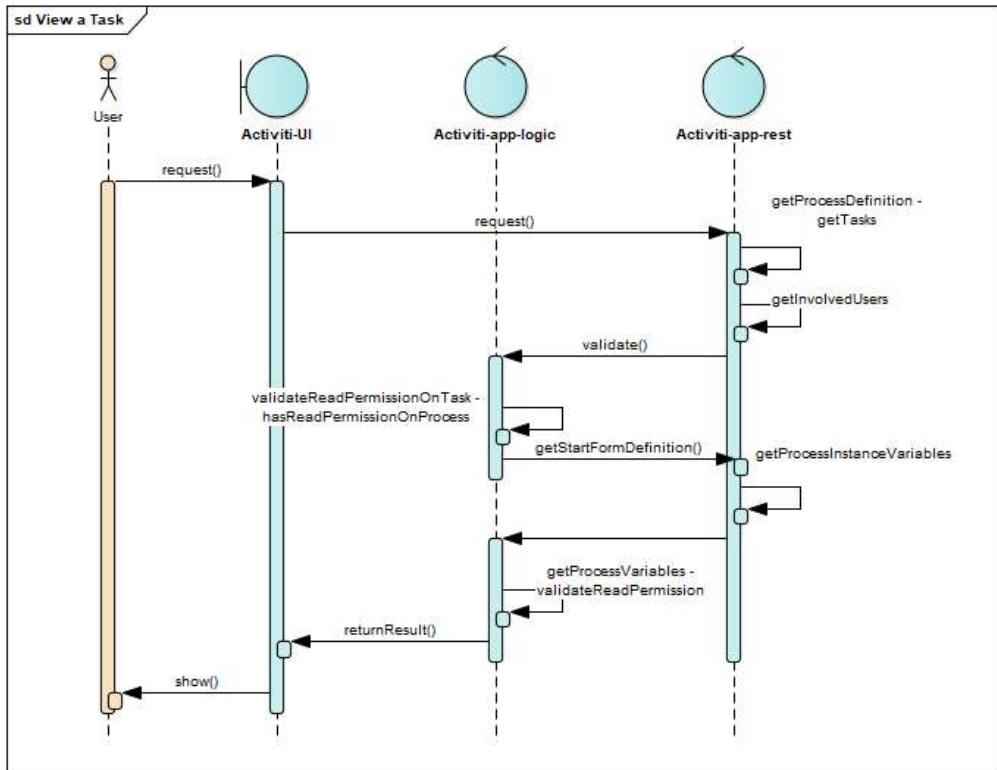
Theo thiết kế ban đầu, kiến trúc dự án Activiti được quản lý theo Maven, mã nguồn được phát triển trên ngôn ngữ Java. Mỗi thành phần nhỏ trong bộ công cụ

được thiết kế như một tiểu dự án (sub-project). Qua quá trình nghiên cứu, luận văn trình bày các tương tác qua lại của các thành phần.

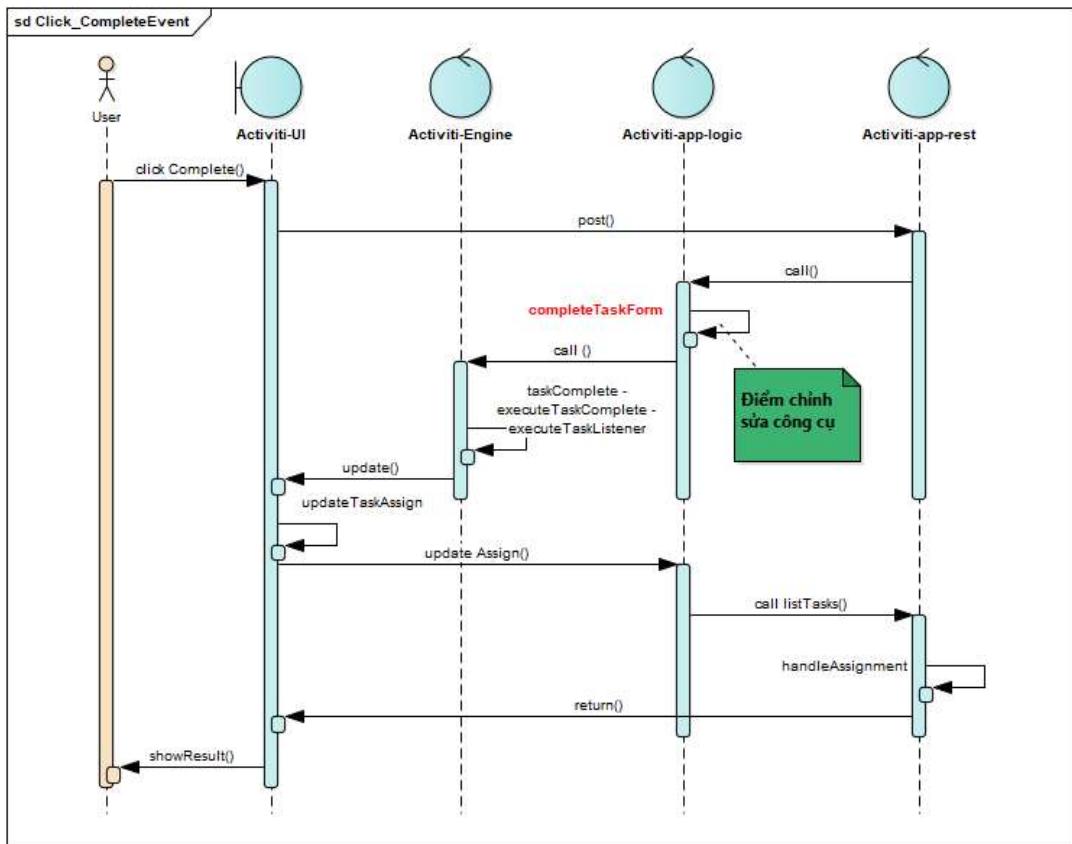


Hình 2.6. Chi tiết luồng xử lý khi Users đăng nhập.

Hình 2.6. minh họa khi người sử dụng (end-users) truy cập hệ thống trên giao diện Activiti UI, các thành phần của module sẽ tham gia xử lý. Trong đó, thành phần quan trọng nhất là ProcessEngineConfiguration (trong Activiti Engine). Đây là một java bean, thông số của nó được khai báo trong file `activiti.cfg.xml`. Bean này được sử dụng để xây dựng ProcessEngine, Activiti cung cấp nhiều lớp (class) sẵn có được sử dụng để định nghĩa processEngineConfiguration. Các lớp này đại diện cho các môi trường/ module khác nhau và mặc định định phải có. Cách thiết kế này giúp cho tối thiểu hóa số lượng thuộc tính cần thiết để cấu hình. Trong các trường hợp tiếp theo, luồng xử lý có thể có sự tham gia của các thành phần như : Activiti-app-logic, Activiti-app-rest, Activiti-admin, Activiti-webapp-rest2... Hình 2.7. và Hình 2.8. trình bày luồng xử lý với hai loại sự kiện (event) và tương tác giữa các thành phần trong hệ thống.



Hình 2.7. Minh họa luồng xử lý xem một task.

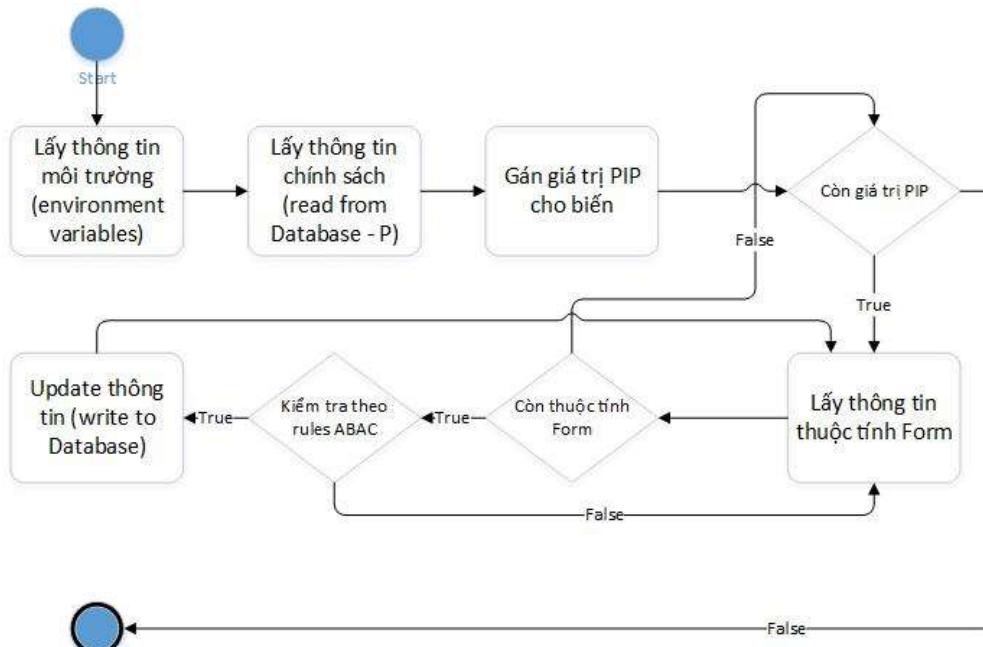


Hình 2.8. Minh họa xử lý cho event Complete.

2.3.2. Ý tưởng tích hợp mô hình ABAC vào công cụ Activiti

Activiti là một trong những nền tảng công cụ hướng tới người lập trình vì vậy cách tổ chức chương trình hết sức rõ ràng và minh bạch. Tại mỗi thành phần ứng dụng, chúng ta có thể dễ dàng theo dõi luồng xử lý. Trên cơ sở này, luận văn sẽ tích hợp mô hình ABAC vào một vài điểm (hook point) trong chương trình nhằm đáp ứng yêu cầu và giải quyết bài toán nghiệp vụ thực tế.

Trong việc hoạt động của công cụ Activiti, mỗi bước trong quy trình xử lý đều có các trạng thái được quản lý riêng biệt. Mỗi hành động tại một bước đều có thể thực hiện các chức năng cơ bản với dữ liệu là CRUD: Create – Tạo mới, Read/ View – Xem dữ liệu, Update – Cập nhật dữ liệu, Delete – Xóa dữ liệu. Sau khi hoàn thành việc nhập việc, cần thiết việc chuyển tiếp qua bước tiếp theo trong quy trình từ bước hiện tại, quá trình này thực hiện qua một sự kiện CompleteTask trong Activiti. Cụ thể, khi người sử dụng click nút “Complete” trên giao diện sử dụng, một sự kiện (event) completeTask sẽ được kích hoạt. Trong luận văn, đây chính là điểm bắt đầu để bổ sung phần hỗ trợ kiểm soát quyền truy cập.



Hình 2.9. Minh họa xử lý cho luồng xử lý ABAC trong Activiti.

2.3.3. Thiết kế tích hợp mô hình ABAC vào thành phần Activiti

Các bước để tích hợp ABAC vào module quản lý Task của Activiti gồm:

Bước 1: Bổ sung hàm xử lý PEP, được gọi bởi Activiti Engine. Trong luận văn sẽ xây dựng hàm này tại bước CompleteTask, lớp (class Java) ActivitiTaskFormService trong thành phần activiti-app-logic.

Bước 2: Xây dựng tập chính sách (policy), lưu các chính sách theo quy định của doanh nghiệp/ tổ chức. Tập chính sách này sẽ được thành phần PDP sử dụng để quyết định việc cấp quyền truy cập.

Bước 3: Trong hàm xử lý PEP, thực hiện phát triển thành phần PDP quyết định việc cấp quyền truy cập.

2.3.4. Cài đặt thiết kế tích hợp

Tạo hàm UpdateAssigneeABAC (PEP) trong project activiti-app-logic để Engine gọi. Trong thân hàm có thành phần PDP quyết định quy cập.

ActivitiTaskFormService.java (activiti-app-logic)

```
public void completeTaskForm(String taskId, CompleteFormRepresentation completeTaskFormRepresentation) {

    ...
    //Bo sung phan Assignee neu dang o buoc GDDVduyet
    if(task.getFormKey().equalsIgnoreCase("KSVduyet"))
        UpdateAssigneeABAC(task,task.getProcessInstanceId());
    }

    //Tungnt: Function added
    public void UpdateAssigneeABAC(Task task, String processInstanceId)
    {
        try {
            ...
            while (rsIm.next()) {
                property_ = rsIm.getString("PROPERTY_");
                limit_floor_ = rsIm.getString("LIMIT_FLOOR_");
                limit_ceiling_ = rsIm.getString("LIMIT_CEILING_");
                user_group_id_ = rsIm.getString("USER_GROUP_ID_");
                isuser_ = rsIm.getString("ISUSER_");
                //
                try {
                    for (String key : varOfForms.keySet()) {
                        ...
                        if (key.equalsIgnoreCase(property_)) {
                            if (property_.equalsIgnoreCase("Noidung")) {
                                if (limit_floor_.equals(conditionValue_)) {
                                    UpdateTaskAssign(task, isuser_, user_group_id_, processInstanceId, stmt);
                                    break;
                                }
                            } else {
                                //if ceiling & floor: not null
                                if (limit_ceiling_ == null) limit_ceiling_ = "0";
                                if (Long.parseLong(limit_floor_) < Long.parseLong(limit_ceiling_)) {
                                    //floor < formValue < ceiling
                                    if (Long.parseLong(limit_floor_) <= Long.parseLong(conditionValue_.toString())
                                        && Long.parseLong(limit_ceiling_) >= Long.parseLong(conditionValue_.toString())) {
                                        UpdateTaskAssign(task, isuser_, user_group_id_, processInstanceId, stmt);
                                        break;
                                    }
                                } else {
                                    //floor < formValue
                                    if (Long.parseLong(limit_floor_) <= Long.parseLong(conditionValue_.toString())) {
                                        UpdateTaskAssign(task, isuser_, user_group_id_, processInstanceId, stmt);
                                        break;
                                    }
                                }
                            }
                        } //end for key
                    }catch (Exception ex) {
                        isUpdate_ = false;
                    }
                ...
            }
        }
    }
}
```

Tạo hàm UpdateTaskAssign (PIP) thực hiện việc cập nhật dữ liệu truy cập (Data).

```

private void UpdateTaskAssign(Task task, String isUser, String groupOrUser, String processInstanceId, Statement stmt)
{
    String queryUpdate = "";
    try{
        if (isUser.equalsIgnoreCase("Y")) {
        ....
            queryUpdate = "Update ACT_RU_TASK set ASSIGNEE_ = '" + groupOrUser +"' where PROC_INST_ID_ = " +
processInstanceId;
            stmt.executeQuery(queryUpdate);
            queryUpdate = "Update ACT_HI_TASKINST set ASSIGNEE_ = '" + groupOrUser +"' where PROC_INST_ID_ = " +
processInstanceId;
            stmt.executeQuery(queryUpdate);
        ....
    }else
    {
        //Get maxID
        ....
        queryUpdate = "insert into ACT_RU_IDENTITYLINK (ID_,GROUP_ID_,TYPE_,TASK_ID_) values
(" +maxID+ "," +groupOrUser+ ", 'candidate', " + taskID_ +")";
        stmt.executeQuery(queryUpdate);
        queryUpdate = "insert into ACT_HI_IDENTITYLINK (ID_,GROUP_ID_,TYPE_,TASK_ID_) values
(" +maxID+ "," +groupOrUser+ ", 'candidate', " + taskID_ +")";
        stmt.executeQuery(queryUpdate);
    ....
    }
    catch (SQLException ex) {
        System.out.println("SMS:-----Error has raised!");
    }
}
}

```

Chỉnh sửa class để hiển thị thông tin phân quyền đã cấp.

TaskUtil.java (activiti-app-logic)

```

public static void fillPermissionInformation(TaskRepresentation taskRepresentation, TaskInfo task, User
currentUser,
IdentityService identityService, HistoryService historyService, RepositoryService repositoryService) {

    //Tungnt OLD: List<GroupRepresentation> groups2 = (List<GroupRepresentation>)new
UserRepresentation(currentUser).getGroups();
    //Change the way get GroupID of user
    UserRepresentation userRepresentation = new UserRepresentation(currentUser);
    List<Group> groups = identityService.createGroupQuery().groupMember(currentUser.getId()).list();
    List<GroupRepresentation> groups2 = new ArrayList<GroupRepresentation>();
    if (groups != null) {
        for (Group group : groups) {
            groups2.add(new GroupRepresentation(group));
        }
    }
}
}

```

Trong công cụ Activiti sẵn có các thực thể để quản lý người sử dụng – User. Tuy nhiên, để phục vụ áp dụng mô hình ABAC, ta bổ sung thêm bảng chính sách ACT_RU_USER_LIMIT. Bảng/ thực thể này sẽ ghi nhận các quy tắc chính sách theo các thuộc tính cần thiết để tham chiếu trong mô hình ABAC. Khi đó, quan hệ giữa các thực thể trong cơ sở dữ liệu sẽ như Hình 2.10 minh họa sau:



Hình 2.10. Quan hệ giữa các thực thể.

Trong cơ sở dữ liệu lưu trữ tập các chính sách, minh họa trong Hình 2.11.

```

-- Create table
create table ACT_RU_USER_LIMIT
(
    id_          NVARCHAR2(64) not null,
    process_id_  NVARCHAR2(255),
    priority_    INTEGER,
    user_group_id_ NVARCHAR2(255),
    property_    NVARCHAR2(255),
    limit_floor_ NVARCHAR2(255),
    limit_ceiling_ NVARCHAR2(255),
    isuser_      NVARCHAR2(255),
    notes_       NVARCHAR2(255),
    company_    NVARCHAR2(255)
)
  
```

Select * from ACT_RU_USER_LIMIT										
ID_	PROCESS_ID_	PRIORITY_	USER_GROUP_ID_	PROPERTY_	LIMIT_FLOOR_	LIMIT_CEILING_	ISUSER_	NOTES_	COMPANY_	
1	4	QuytrinhTindung	1	hotd	Noidung	Khẩn-VIP	N	Tờ trình khẩn cấp, độ ưu tiên cao nhất.		
2	1	QuytrinhTindung	1	dvgd50	Giatri_DX	50000000001	N	Hạn mức > 50 tỷ		
3	2	QuytrinhTindung	1	dvgd1050	Giatri_DX	10000000001	N	Hạn mức 10~50 tỷ		
4	3	QuytrinhTindung	1	dvgd0110	Giatri_DX	0	N	Hạn mức < 10 tỷ		

Hình 2.11. Bổ sung tập chính sách (P).

Trong bảng ACT_RU_USER_LIMIT, các trường dữ liệu sẽ quy định quy tắc chính sách như sau: với mỗi quy trình (process_id_), một quy tắc (rule) sẽ được định nghĩa cho từng thuộc tính (property_) sẽ có các giá trị quy định hoặc giá trị chặn dưới (limit_floor_) và giá trị chặn trên (limit_ceiling_), nếu thỏa mãn sẽ được gán quyền cho người dùng hoặc nhóm người dùng (user_group_id_).

Trong thực tế, ngoài yêu cầu phân quyền theo yếu tố thuộc tính động còn có yêu cầu quản lý ủy quyền. Để hiện thực điều này, trong luận văn cũng đã bổ sung module quản lý ủy quyền này. Cụ thể là bổ sung thực thể ACT_RU_USER_DELEGATE. Thực thể gồm các thuộc tính để quản lý việc ủy quyền từ một người (user_send_) tới một người dùng khác (user_get_), việc ủy quyền này được xét trong một khoảng thời gian, một giai đoạn nhất định (date_from_ và date_to_), chi tiết được minh họa trong Hình 2.12.

```
create table ACT_RU_USER_DELEGATE
(
    id          NVARCHAR2(64) not null,
    user_send_  NVARCHAR2(255),
    user_get_   NVARCHAR2(255),
    date_from_  NVARCHAR2(255),
    date_to_   NVARCHAR2(255),
    isactive_   NVARCHAR2(255)
)
```

```
select * from ACT_RU_USER_DELEGATE order by ID_ desc ;
```

ID_	USER_SEND_	USER_GET_	DATE_FROM_	DATE_TO_	ISACTIVE_
1	giamdoc1ty	uyquyen	20181201	20190420	Y

Hình 2.12. Minh họa quản lý ủy quyền.

Với việc quản lý ủy quyền, một vấn đề giải quyết là nảy sinh tính mơ hồ, không rõ ràng giữa người ủy quyền và người được ủy quyền. Cụ thể như trong cùng một thời điểm cả người ủy quyền và người được ủy quyền đều có thể thực hiện các thao tác theo vai trò được gán sẵn. Vấn đề đặt ra trong hoàn cảnh đó là phân biệt rõ trách

nhiệm liên quan hay việc phân định chủ thể xử lý trong thời điểm đó là người ủy quyền hay người được ủy quyền. Để giải quyết vấn đề này, luận văn đưa ra các giới hạn chặn về thời gian được ủy quyền và việc quản lý hiệu lực của ủy quyền. Trong giới hạn thời gian ủy quyền (từ thời gian nào đến thời gian nào) và chỉ khi ủy quyền có hiệu lực (active), người được ủy quyền mới có thể thực hiện các thao tác ủy quyền. Lưu ý rằng việc ủy quyền chỉ áp dụng cho quyền hạn phê duyệt/từ chối phê duyệt hồ sơ tín dụng, không áp dụng cho các phạm vi khác.

2.4. Tổng kết chương

Chương này đã đưa ra các thao tác cụ thể nhất tích hợp mô hình chính sách ABAC vào công cụ Activiti hỗ trợ BMPN. Thông thường các tính năng ban đầu của các hệ thống BPM như Oracle, Alfresco, IBM... hỗ trợ quản lý người sử dụng, nhóm người sử dụng theo vai trò (roles). Luận văn đã đưa ra cách tiếp cận cải tiến và mềm dẻo hơn là yêu cầu an ninh được đưa ra kết hợp roles và trên các tập thuộc tính trong các sự kiện yêu cầu truy cập (xem thông tin, sửa thông tin...). Hướng đi này là cần thiết bởi thực tế chứng minh, yêu cầu về an ninh trong vận hành quy trình nghiệp vụ rất đa dạng và yêu cầu sự linh động cao.

CHƯƠNG 3

VẬN DỤNG VÀ THỰC NGHIỆM

Chương này tập trung vận dụng giải pháp tích hợp ABAC vào bộ công cụ quản lý quy trình nghiệp vụ Activiti để giải quyết bài toán “Phê duyệt hồ sơ tín dụng” tại Ngân hàng thương mại.

Thực tế các hệ thống thông tin và các công cụ hỗ trợ như Activiti mới đang dùng lại ở việc hỗ trợ phân quyền và kiểm soát truy cập theo vai trò. Cụ thể các chủ thể có yêu cầu truy cập vào hệ thống đều được gán các vai trò được tạo trước đó. Điều này không đáp ứng được tiêu chí phân quyền linh động của bài toán nghiệp vụ. Trong chương 3 luận văn sẽ trình bày việc áp dụng mô hình điều khiển truy cập theo thuộc tính ABAC trên công cụ Activiti để đáp ứng nhu cầu có tính động của bài toán.

3.1. Bài toán nghiệp vụ “Phê duyệt hồ sơ tín dụng”

Thực tế quy trình “Phê duyệt hồ sơ tín dụng” có thể hiểu đầy đủ là “Hệ thống luân chuyển và quản lý tác nghiệp hồ sơ tín dụng” [14]. Để thực thi quy trình và quản lý được cần thiết một hệ thống, một hệ thống thông tin gồm nhiều chức năng xoay quanh việc xử lý hồ sơ bao gồm:

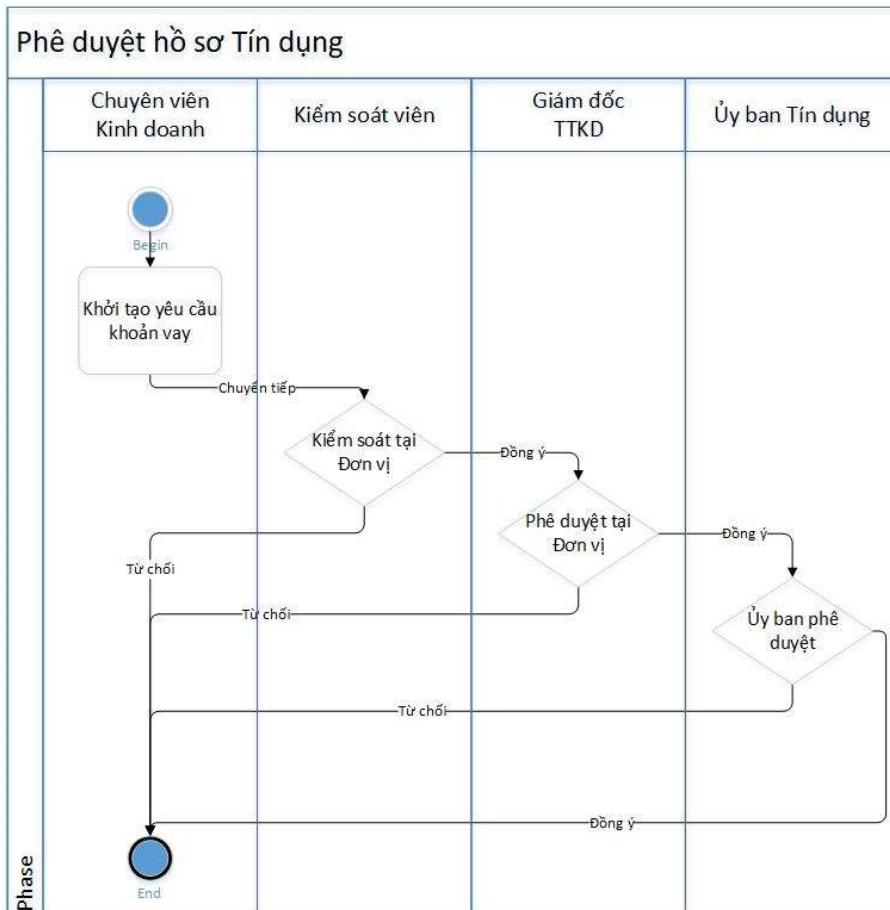
Tính năng đầu tiên đề cập tới là việc quản lý các quy trình luân chuyển và xử lý hồ sơ tín dụng. Việc này nhằm đảm bảo tính thống nhất, tập trung và chuẩn hóa bộ dữ liệu khách hàng; giúp tăng tính bảo mật thông tin và nâng cao năng lực quản trị rủi ro của Ngân hàng thương mại.

Các giao diện màn hình nhập liệu, các tờ trình, biểu mẫu được quản lý trên hệ thống và tương tác với người sử dụng theo phạm vi quyền hạn khi truy cập dữ liệu nhạy cảm như thông tin khách hàng, tài sản đảm bảo, hồ sơ tín dụng.

Bên cạnh đó, việc xử lý luân chuyển và giao nhận hồ sơ được tự động hóa nhằm giảm thiểu thời gian thao tác hành chính và giảm chi phí giấy tờ. Ngoài ra các tính năng như báo cáo tổng hợp, báo cáo chi tiết, các yêu cầu về quản trị hệ thống – quản trị người sử dụng... đều cần yêu cầu tuân thủ và đảm bảo trong hệ thống này.

Với tập chức năng đồ sộ như vậy, yêu cầu cho hệ thống thông tin để quản lý là rất lớn và khá phức tạp. Trong tổ chức ngân hàng thương mại, quy trình Tín dụng tác động tới rất nhiều đơn vị, bộ phận cùng tác nghiệp. Để đơn giản bài toán, luận văn chỉ tập trung vào mạch chính trong quy trình Luân chuyển hồ sơ tín dụng. Các đối tượng tham gia vào quy trình gồm có CVKD (chuyên viên quan hệ khách hàng),

KSVKD (kiểm soát viên trung tâm kinh doanh), GD TTKD (giám đốc trung tâm kinh doanh), UBTD (ủy ban tín dụng hội sở). Với mô hình đơn giản nhất, luồng quy trình thực hiện như Hình 3.1.



Hình 3.1. Quy trình phê duyệt hồ sơ tín dụng.

Bước 1: CVKD có trách nhiệm lập yêu cầu cấp tín dụng. Yêu cầu phải đảm đầy đủ các thông tin về Khách hàng (tên khách hàng, địa chỉ, vốn điều lệ, tỷ lệ vốn...), thông tin về nhu cầu cấp tín dụng (loại hình tín dụng, sản phẩm tín dụng, mục đích cấp tín dụng, số tiền đề nghị cấp tín dụng, thời hạn đề nghị, lãi suất cho vay, các loại phí, kỳ hạn trả nợ...), thông tin tài sản đảm bảo (loại hình tài sản đảm bảo, giá trị tài sản đảm bảo, tỷ lệ cấp tín dụng/ định giá tài sản)

Bước 2: KSV tiếp nhận hồ sơ từ CVKD và kiểm tra danh mục hồ sơ với các nội dung theo điều kiện, kiểm soát... trường hợp đồng ý với nội dung cấp tín dụng, thực hiện bước tiếp theo của quy trình. Trường hợp không đồng ý, phát hiện những điểm còn thiếu/ còn chưa phù hợp với phê duyệt, sản phẩm thì phải ghi chú lại và phản hồi lại để chỉnh sửa.

Bước 3: Tùy theo thẩm quyền phê duyệt, giám đốc TTKD sẽ nhận được các hồ sơ theo hạn mức. Các hạn mức này quy định rõ trong văn bản quy chế. Sau khi tiếp nhận, GD có trách nhiệm phản hồi và trao đổi lại để thống nhất những điểm thiếu sót/ chưa phù hợp (nếu có). Trong trường hợp đồng ý, hồ sơ cấp tín dụng sẽ được chuyển lên ủy ban tín dụng phê duyệt.

Bước 4: UBTD là cấp phê duyệt chuyên gia, tiếp nhận mọi yêu cầu không giới hạn hạn mức. UBTD hoặc người được ủy quyền/ cấp có thẩm quyền sẽ phê duyệt hồ sơ. Hồ sơ sau phê duyệt sẽ được thông báo cho CVKD để thực hiện các bước trong quy trình tiếp theo là Quy trình tác nghiệp tín dụng.

3.2. Yêu cầu về chính sách truy cập

Yêu cầu về quản trị người sử dụng, đảm bảo kiểm soát truy cập, an ninh bảo mật cũng là một phần không thể thiếu trong hệ thống “Phê duyệt hồ sơ tín dụng”. Đặc biệt, tại bước 03 trong quy trình nêu trên có việc kiểm soát theo hạn mức phê duyệt theo thẩm quyền. Ngoài ra, cần nhắc tới yếu tố đơn vị kinh doanh. Bởi vì, mỗi cấp phê duyệt TTKD chỉ có thể phê duyệt các hồ sơ cấp tín dụng của đơn vị mình quản lý. Các GD TTKD không thể xem hay truy cập hồ sơ cấp tín dụng của TTKD khác đang được trình để phê duyệt. Các quy định cụ thể được minh họa trong Bảng 3.1.

Bảng 3.1 – Bảng phân quyền chức năng người sử dụng

Người sử dụng	Chức năng	Quyền hạn	Ghi chú	
Cán bộ nhân viên (Cán bộ tín dụng)	Khởi tạo/ Chính sửa hồ sơ tín dụng	Nhập liệu/ chỉnh sửa hồ sơ tín dụng Truy vấn hồ sơ Thao tác trên hồ sơ mình khởi tạo		Đã đáp ứng
Trưởng, phó phòng (Kiểm soát viên) tại ĐVKD	Kiểm tra, kiểm soát hồ sơ tín dụng do cán bộ tín dụng khởi tạo	Chuyển tiếp hồ sơ cho Giám đốc Yêu cầu bổ sung (nếu có) Truy vấn dữ liệu, hồ sơ của đơn vị		Đã đáp ứng
Giám đốc/ Phó giám đốc ĐVKD	Phê duyệt/Từ chối phê duyệt các hồ sơ thuộc thẩm quyền	Phê duyệt/Từ chối phê duyệt các hồ sơ thuộc thẩm quyền Chuyển tiếp hồ sơ Yêu cầu bổ sung (nếu có)	Thẩm quyền phê duyệt theo hạn mức	Đáp ứng 1 phần
Thành viên Ủy ban tín dụng (Hội sở)	Tiếp nhận các hồ sơ thuộc thẩm quyền phê duyệt	Phê duyệt/Từ chối phê duyệt đối với các hồ sơ thuộc thẩm quyền Truy vấn dữ liệu trên toàn hệ thống		Đã đáp ứng

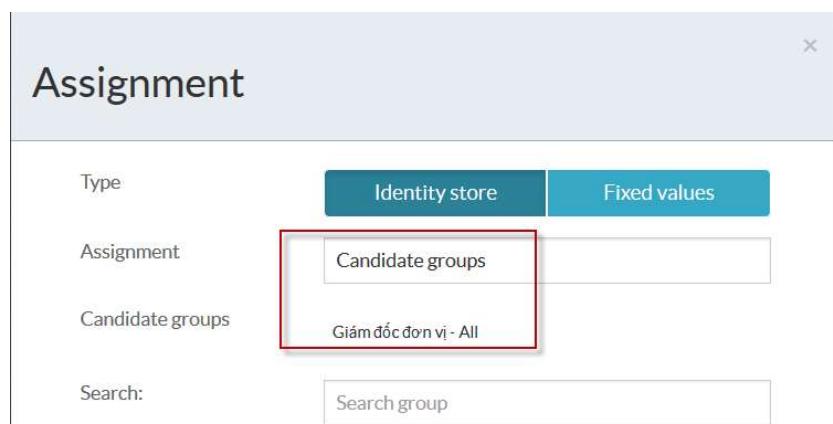
Ủy quyền	Tiếp nhận các hồ sơ thuộc thẩm quyền được ủy quyền	Phê duyệt/Từ chối phê duyệt đối với các hồ sơ thuộc thẩm quyền ủy quyền Truy vấn dữ liệu thuộc ủy quyền	Ủy quyền được quy định theo văn bản	Chưa đáp ứng
----------	--	--	-------------------------------------	--------------

Trong thực tế hiện nay, các chính sách truy cập trong hệ thống “Phê duyệt hồ sơ tín dụng” thông thường thực hiện được các yêu cầu chủ yếu là:

- Mô hình phân quyền: Tính năng → Vai trò → Người sử dụng/ Nhóm người sử dụng. Tức là, vai trò được thiết lập phân quyền thao tác trên các tính năng (menu), vai trò được quyền thực hiện 1 số hành động (xem/sửa/xóa) trên màn hình chức năng, mỗi người dùng hoặc nhóm người dùng được gán vào mỗi vai trò.
- Việc phân quyền cho vai trò được làm sẵn trước đó, rồi gán/bỏ sung người dùng hoặc nhóm người dùng vào vai trò.
- Phân quyền theo phạm vi dữ liệu được làm một phần, cụ thể với bài toán “Phê duyệt hồ sơ tín dụng” thì người dùng được phân quyền thao tác trên dữ liệu của đơn vị họ làm việc. Ví dụ, cán bộ tín dụng hay Kiểm soát viên ở chi nhánh Hà Nội chỉ được thao tác dữ liệu của đơn vị là chi nhánh Hà Nội.

Trong khi đó, các yêu cầu như phân quyền động, tức là phân quyền theo thuộc tính động chưa được xử lý. Có thể lấy một ví dụ như sau:

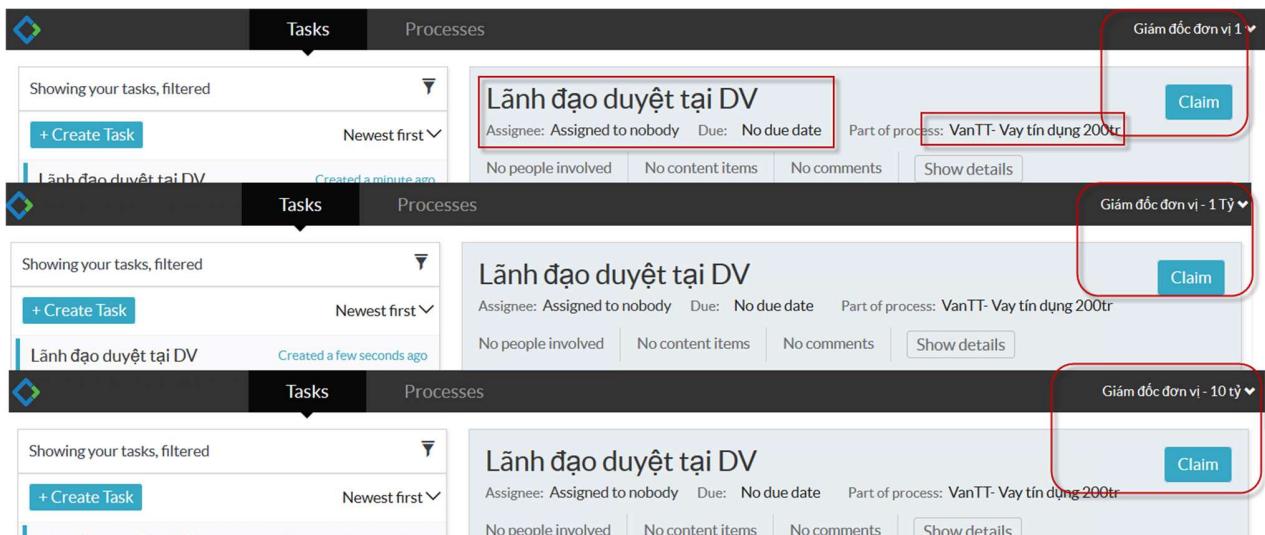
Trên hệ thống “Phê duyệt hồ sơ tín dụng”, có định sẵn vai trò là nhóm “Giám đốc” có quyền hạn như trong Bảng 3.2 và minh họa trong Hình 3.2. Một hồ sơ được cán bộ tín dụng tạo ra, sau khi kiểm soát viên phê duyệt sẽ được chuyển tiếp lên cấp xử lý tiếp theo là Giám đốc ĐVKD.



Hình 3.2. Minh họa gán vai trò xác định trước.

Tuy nhiên, quy định sẵn có là phân quyền cho vai trò nhóm “Giám đốc” thì với hồ sơ tín dụng này các giám đốc đều có thể xem, nhận task và phê duyệt hồ sơ này.

Điều này gây ra một vấn đề là chưa đáp ứng được mong muốn trong yêu cầu đảm bảo chính sách truy cập trong quy trình, yêu cầu đó là “phê duyệt theo thẩm quyền hạn mức”. Cụ thể thẩm quyền hạn mức là có các hạn mức theo đoạn giá trị ví dụ như: 0VNĐ – 1 tỷ VNĐ, từ 1 tỷ VNĐ – 5 tỷ VNĐ, từ 5 tỷ VNĐ trở lên... các Giám đốc được phê duyệt theo các hạn mức giá trị ở trên. Kết quả của việc chưa đảm bảo này được minh họa trong Hình 3.3. là với khoản vay 200 triệu VNĐ, các giám đốc đều có thể xem và phê duyệt hồ sơ tín dụng này.



Hình 3.3. Minh họa vi phạm phê duyệt theo thẩm quyền hạn mức.

Với việc bổ sung công cụ theo mô hình ABAC và bổ sung việc quản lý ủy quyền (như đã đề cập trong phần 2.2) các tính chất để đảm bảo chính sách quyền truy cập đã được đáp ứng và vận hành theo yêu cầu nghiệp vụ trong thực tế.

3.3. Xây dựng mô hình ABAC

Trong bài toán nghiệp vụ, mô hình ABAC được áp dụng như sau: một bộ gồm các thành phần $\{S, R, E, A_S, A_R, A_E, P\}$:

S: tập các chủ thể (subjects). Chủ thể ở đây là người tham gia vào quy trình. Thực hiện một nhiệm vụ theo phân công hoặc theo trách nhiệm (roles) cho trước. Tập các chủ thể được biểu diễn với $S = \{s_1, s_2, \dots, s_n\}$.

R: tập các tài nguyên (resources). Các yếu tố môi trường được quản lý như quy trình nghiệp vụ, các nhiệm vụ hay bước trong quy trình và dữ liệu trong mỗi nhiệm vụ đó. Tập tài nguyên được biểu diễn với $R = \{r_1, r_2, \dots, r_m\}$.

E: tập các yếu tố môi trường (environments). Đây là các yếu tố trong hệ thống thông tin như số lượng user, ngày giờ hệ thống. Tập hợp này sẽ là $S = \{e_1, e_2, \dots, e_k\}$.

P: tập các quy tắc biểu diễn cho các chính sách (policies) của hệ thống. Tập hợp này quy định các quy tắc cho trước như định lượng hóa hạn mức được phân quyền cho mỗi người sử dụng được thao tác phê duyệt.

Trong mô hình ABAC, mỗi quy tắc (rule) được xác định một điều kiện nếu thỏa mãn thì một chủ thể sẽ được truy cập vào một tài nguyên (đối tượng) trong một môi trường (hệ thống) cụ thể. Điều đó đồng nghĩa chủ thể (người sử dụng) chỉ có thể truy cập vào tài nguyên trong điều kiện môi trường nếu nó thỏa mãn quy tắc. Biểu thức biểu diễn quy tắc là:

r: granted (s, r, e) $\leftarrow A_1, A_2, \dots, A_n$ ($n \geq 0$)

Trong đó: s, r, e lần lượt tương ứng cho người sử dụng, các nhiệm vụ (task) và yếu tố thời gian hệ thống; A_1, A_2, \dots, A_n là các điều kiện logic.

Với bài toán nghiệp vụ ta áp dụng như sau:

granted (s, r, e): người sử dụng s có thể thực hiện nhiệm vụ r thỏa mãn yếu tố e.

granted (s, r, _): người sử dụng s có thể thực hiện nhiệm vụ r với mọi điều kiện.

granted (_, r, _): mọi người sử dụng đều thực hiện nhiệm vụ r với mọi điều kiện.

3.4. Xây dựng và thực thi mô hình quy trình trên Activiti

Để thực hiện một quy trình trên Activiti, ta cần thực hiện theo trình tự như sau:

Bước 1: Cài đặt Activiti trên web server

Bước 2: Mô hình hóa quy trình nghiệp vụ trên Activiti

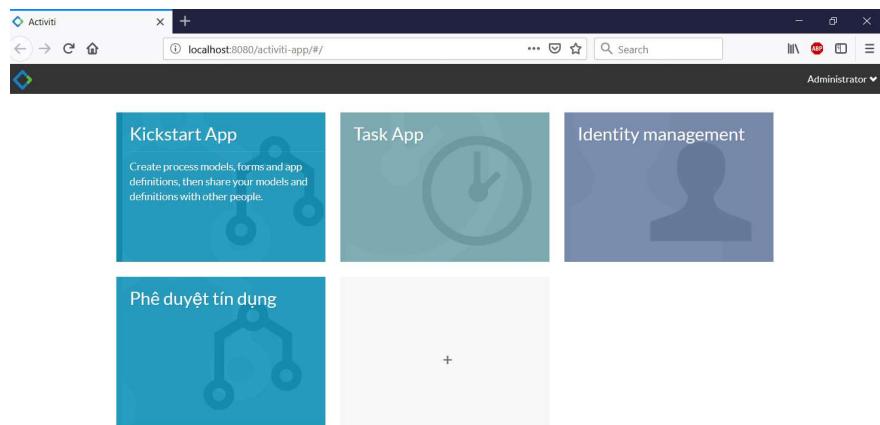
Bước 3: Thực thi quy trình trên Activiti

3.4.1. Cài đặt Activiti trên Webserver

Phiên bản Activiti mà luận văn sử dụng là Activiti 6.0. Phiên bản này hoàn toàn tích hợp BPMN trong hệ thống như đã trình bày trước đây. Thành phần thao tác sử dụng là Activiti App.

Bước 1: cài đặt Activiti 6.0 theo hướng dẫn hình ảnh [8].

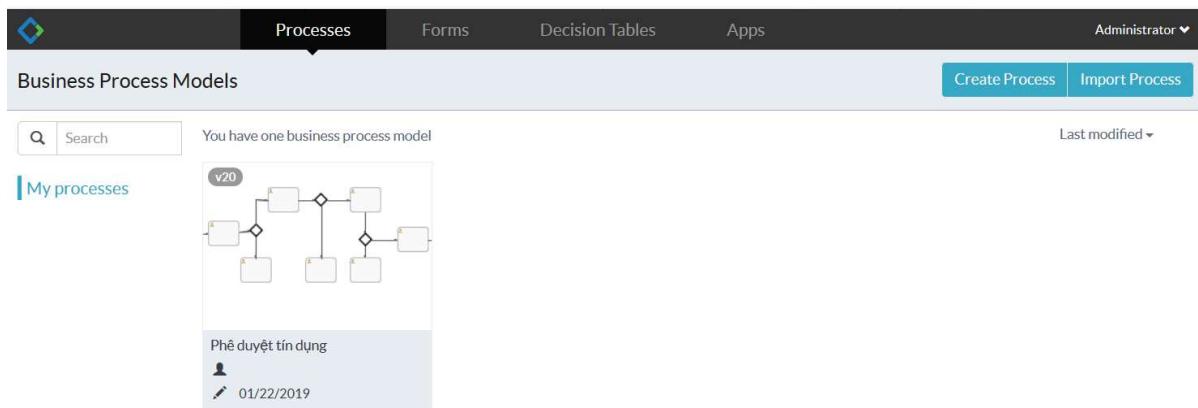
Bước 2: truy cập địa chỉ <http://localhost:8080/activiti-app/#/>. User và mật khẩu quản trị (admin) mặc định là test.



Hình 3.4. Minh họa màn hình Activiti App.

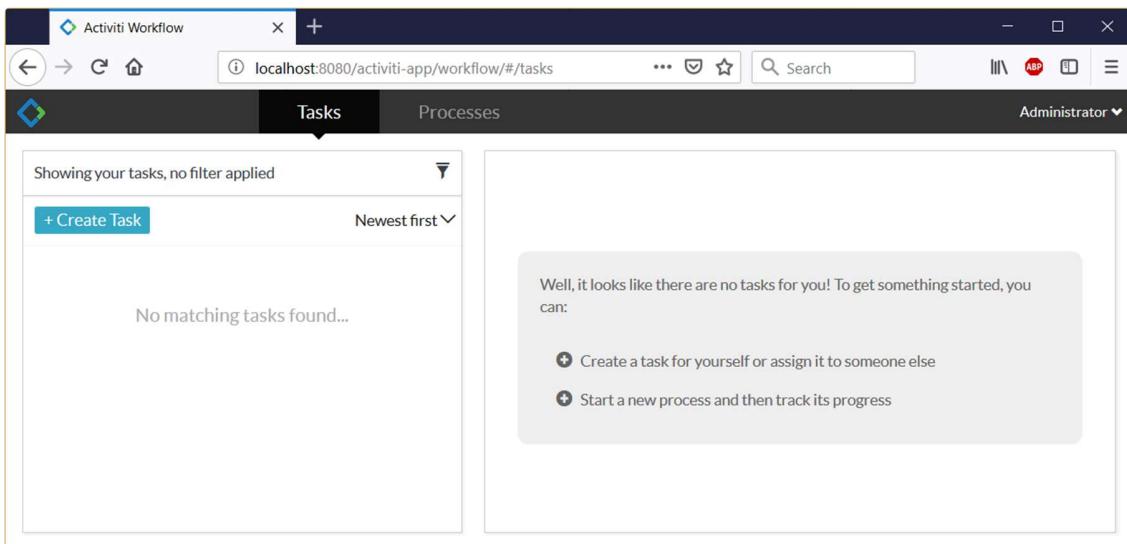
Sau khi đăng nhập, giao diện Activiti gồm 03 thành phần như trong hình 3.4.

Thành phần đầu tiên là Kickstart App. Đây là khung quản lý việc mô hình hóa quy trình như Hình 3.5. Thành phần này gồm các thẻ tab: Processes – quản lý mô hình hóa quy trình ; Forms – quản lý các màn hình nhập liệu, truy vấn dữ liệu ; Decision Tables – quản lý bảng quy tắc nghiệp vụ (business rule) ; Apps – quản lý trạng thái các mô hình như công khai (public) một mô hình để thực thi.



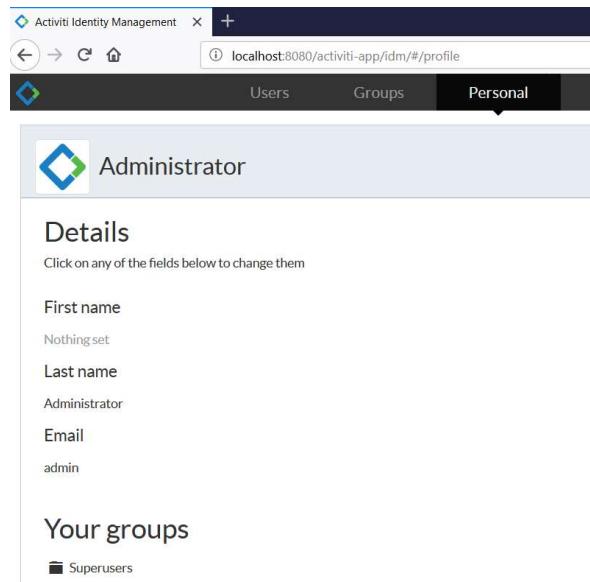
Hình 3.5. Minh họa màn hình Kickstart App.

Tiếp theo là thành phần Task App. Cung cấp tính năng quản lý việc thực thi các quy trình, quản lý các tasks như Hình 3.6. Người sử dụng theo phân quyền có thể xem hoặc tạo các task của bản thân, các task mình có liên quan, chủ động tự gán (claim) task cho mình...



Hình 3.6. Minh họa màn hình Tasks.

Cuối cùng là thành phần Identity management. Trong công cụ, thành phần này cho phép quản trị người sử dụng như Hình 3.7. Bao gồm các chức năng cơ bản như: tạo mới người sử dụng, nhóm người sử dụng, phân quyền người sử dụng/ nhóm người sử dụng...



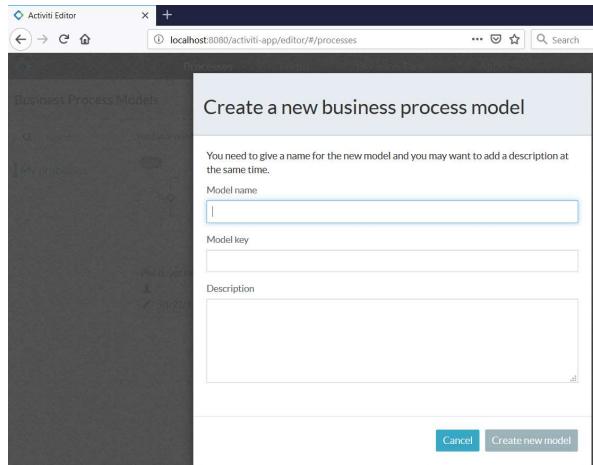
Hình 3.7. Minh họa màn hình Identity management.

3.4.2. Xây dựng mô hình quy trình “Phê duyệt hồ sơ tín dụng” trên Activiti

Sau khi tích hợp mô hình điều khiển ABAC vào công cụ Activiti, ta thực hiện việc mô hình hóa quy trình nghiệp vụ trên công cụ Activiti để quản lý và kiểm chứng. Quá trình gồm 2 giai đoạn là mô hình hóa quy trình và thực thi quy trình.

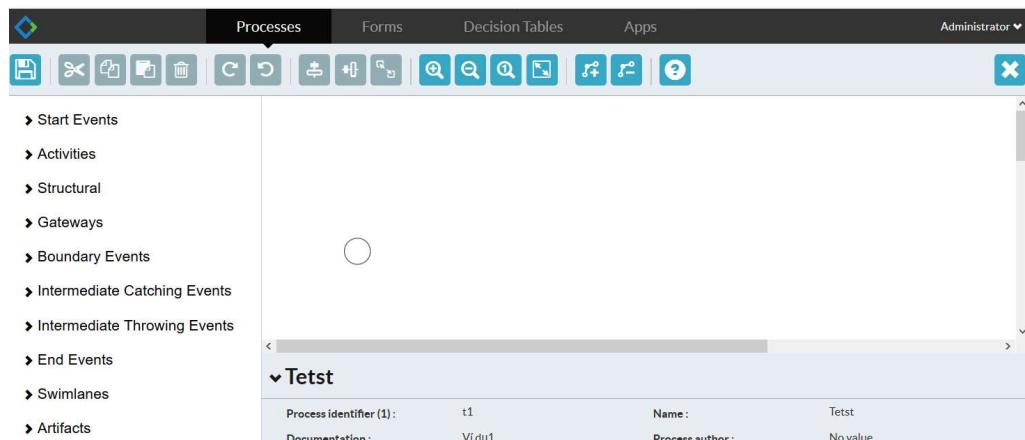
3.4.2.1. Biểu diễn mô hình quy trình

Để tạo một mô hình cho quy trình nghiệp vụ BPM trên công cụ Activiti, ta thao tác theo trình tự như sau: Kickstart App > Create Process > Edit model details, minh họa Hình 3.8.



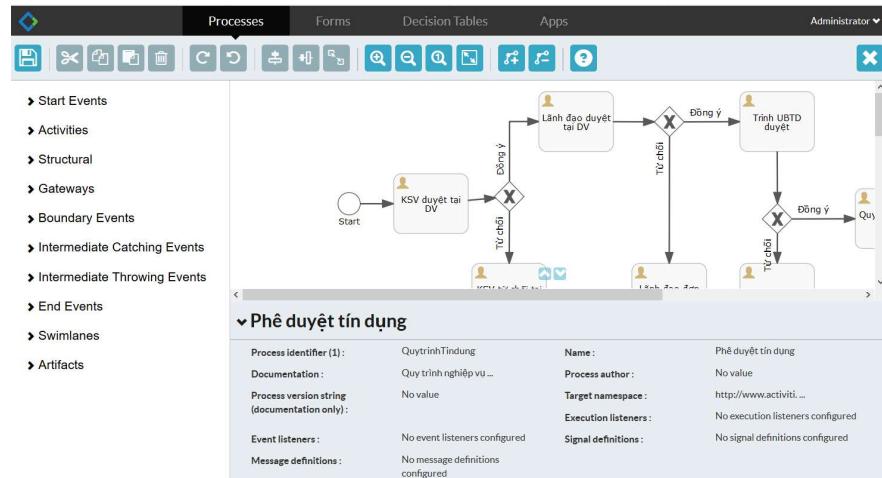
Hình 3.8. Minh họa tạo mới BPM.

Tiếp theo, nhập thông tin cho mô hình “Phê duyệt tín dụng” và chọn chức năng “Create new model”. Sau đó, màn hình designer (thiết kế) cho mô hình xuất hiện. Bên trái màn hình là các thành phần sử dụng cho việc mô hình hóa như: Event, Activities, Gateways, Swimslanes, Artifacts... cụ thể trong Hình 3.9.



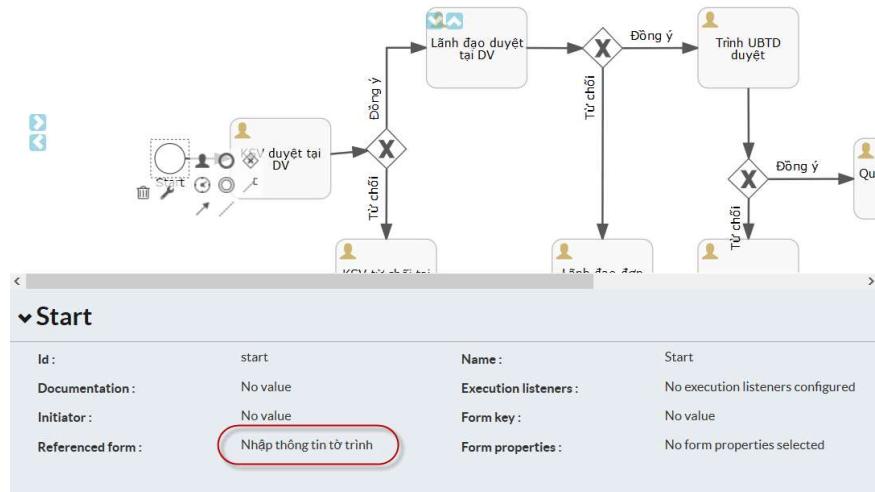
Hình 3.9. Minh họa màn hình thiết kế mô hình.

Việc mô hình hóa được thực hiện bằng cách kéo thả các đối tượng từ menu chức năng để thiết kế mô hình. Các thành phần trong quy trình sẽ lần lượt được mô hình hóa và trực quan như Hình 3.10.



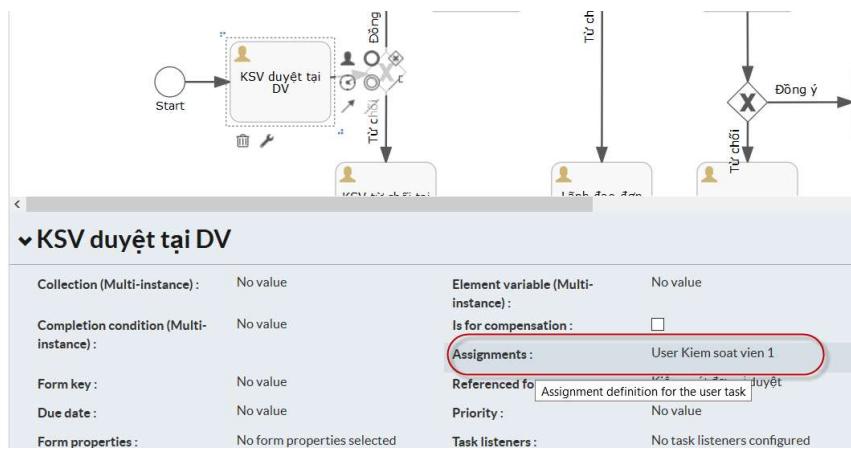
Hình 3.10. Minh họa quy trình được mô hình hóa.

Theo quy trình hồ sơ, tại bước đầu tiên yêu cầu CVKD phải nhập thông tin về hồ sơ, nên tại bước này ta chọn form nhập liệu là “Nhập thông tin tờ trình”, minh họa Hình 3.11. Form này sẽ được đề cập tới trong phần tiếp theo.



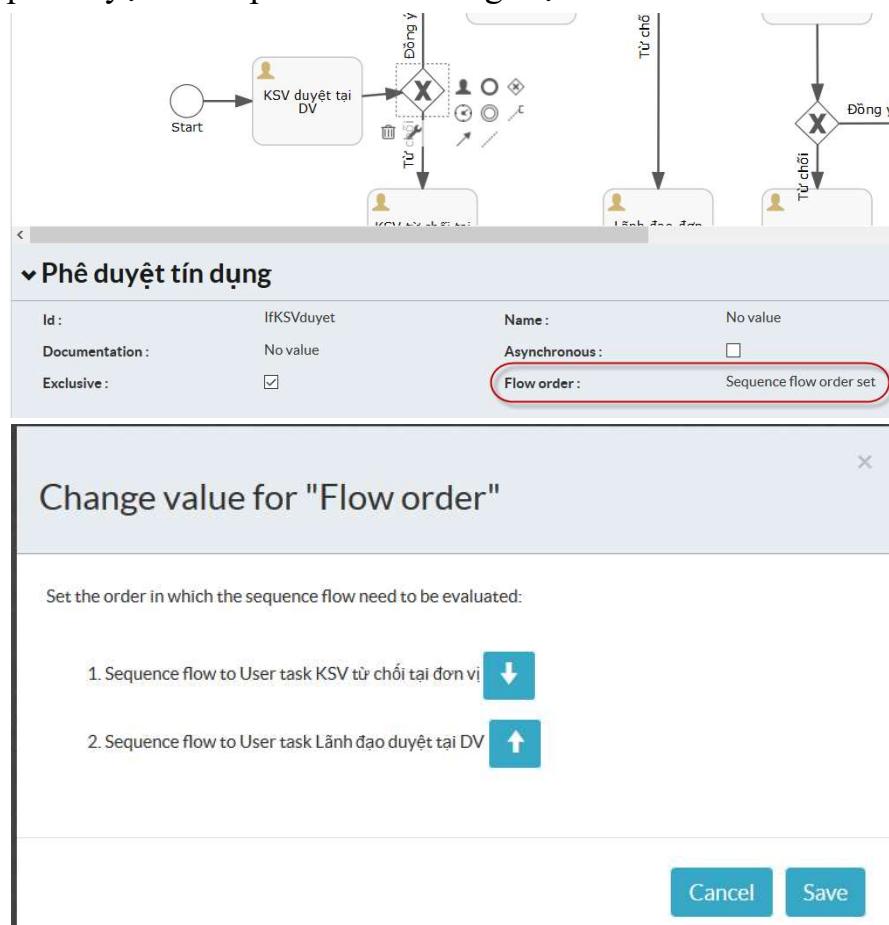
Hình 3.11. Minh họa form nhập liệu trong quy trình.

Tại bước tiếp theo trong quy trình yêu cầu bắt buộc là KSV kiểm duyệt việc tạo hồ sơ nên ta gán nhiệm vụ này cho KSV như Hình 3.12.



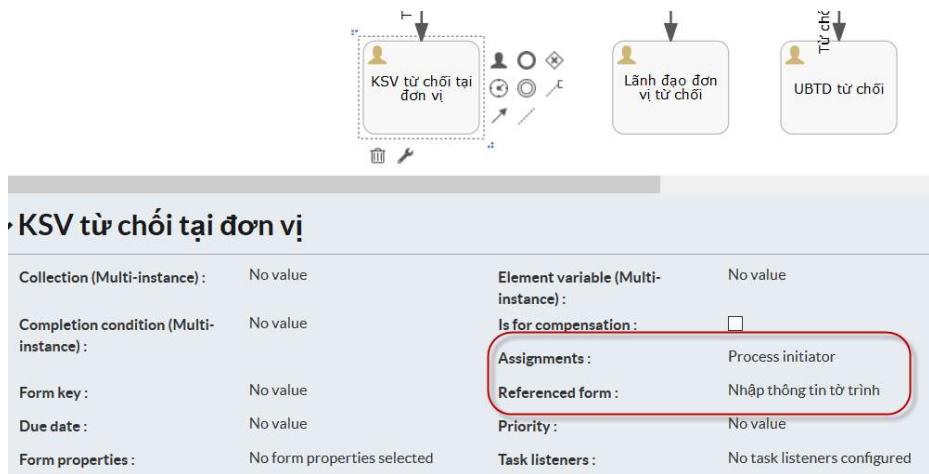
Hình 3.12. Minh họa gán yêu cầu cho KSV.

Trong quy trình có nhiều điểm điều khiển định hướng xử lý, Hình 3.13. mô tả quy trình theo phê duyệt của cấp kiểm soát trong mục Flow order.



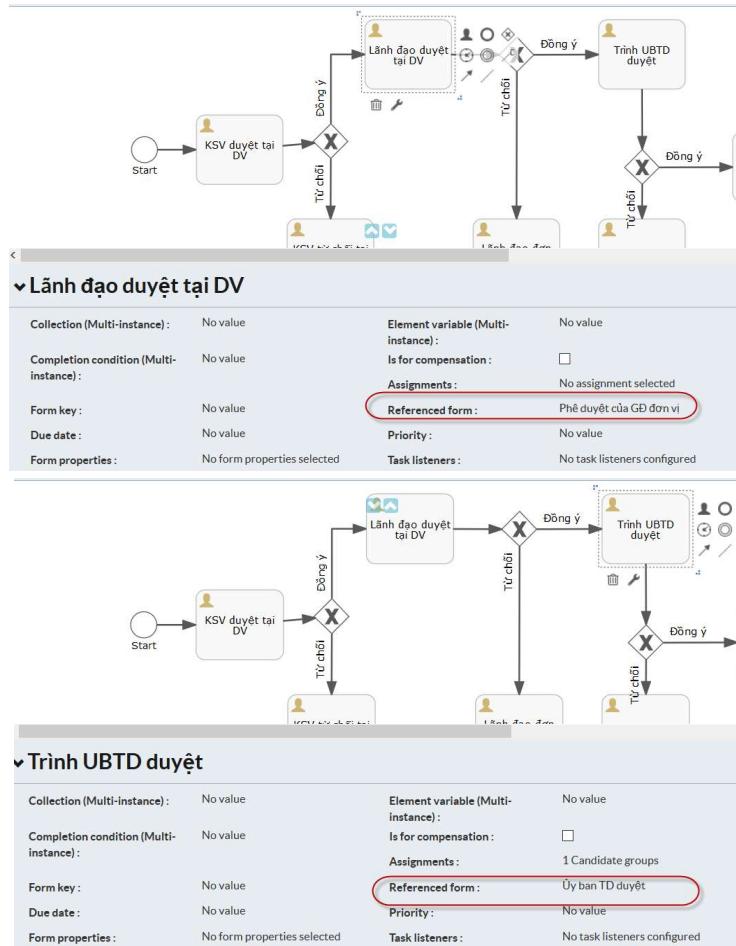
Hình 3.13. Minh họa điều khiển hướng quy trình.

Tại mục Flow order cho phép xác định hướng xử lý với các trường hợp đồng ý hay từ chối phê duyệt. Khi yêu cầu phê duyệt bị từ chối, yêu cầu sẽ gửi trả lại cho người khởi tạo quy trình (trường hợp này là CVKD) như Hình 3.14.



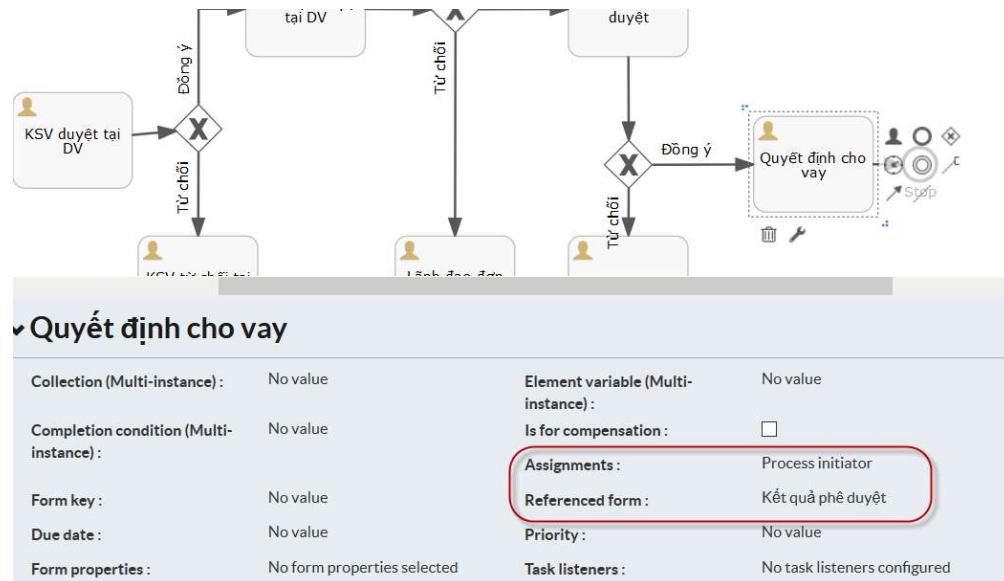
Hình 3.14. Minh họa khi yêu cầu bị từ chối.

Trong quy trình thực tế, mọi thông tin đều được thu thập và ghi lại để để người phê duyệt theo dõi và kiểm tra. Vì vậy, ta tạo ra các màn hình (form) xem dữ liệu tại các bước kiểm duyệt, phần Referenced form như Hình 3.15.



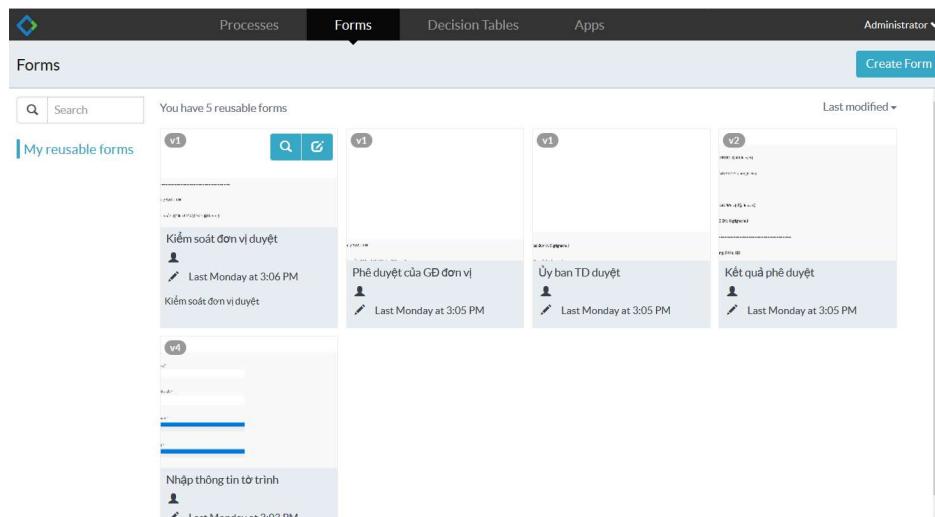
Hình 3.15. Minh họa các màn hình xem thông tin.

Kết thúc mỗi quy trình, hồ sơ sẽ được gán cho người khởi tạo quy trình để xem xét các kết luận, kết quả phê duyệt và thực hiện các thủ tục tiếp theo quy trình nghiệp vụ. Do đó yêu cầu sẽ quay trở lại cho người khởi tạo quy trình như Hình 3.16.



Hình 3.16. Minh họa phê duyệt cuối của quy trình.

Như đã đề cập trước đó, tại các bước trong quy trình đều cần nhập liệu và ghi nhận thông tin. Do đó, ta đã sử dụng một số form (màn hình) để nhập liệu và xem dữ liệu. Cách tạo các form như sau: vào tính năng Forms > Create Form > Nhập thông tin > Create new form như Hình 3.17. và Hình 3.18.



Hình 3.17. Minh họa việc tạo màn hình dữ liệu.

Create a new form

You need to give a name for the new form and you may want to add a description at the same time.

Form name

Form key

Description

Edit model details

Change any of the model properties below and then press Save to update the model.

Model name

Model key

Description

Cancel Create new form Save

Hình 3.18. Minh họa việc tạo màn hình dữ liệu.

Sau khi tạo form, sẽ có màn hình thiết kế form với các đối tượng : Text – ô nhập liệu dữ liệu dạng ký tự; Multiline text – ô nhập dữ liệu dạng ký tự, hỗ trợ nhiều dòng; Number – ô nhập dữ liệu dạng số ; Dropdown - ô nhập dữ liệu dạng lựa chọn và các đối tượng khác. Chi tiết thao tác như Hình 3.19, Hình 3.20. và Hình 3.21.

Nhập thông tin tờ trình

Version 4 Last updated by , Last Monday at 3:03 PM

Design Outcomes

Mã khách hàng*

Tổng giá trị đề xuất *

PGD/ Chi nhánh *

Đơn vị tiền tệ *

Cancel Create new form Save

Hình 3.19. Minh họa việc thiết kế màn hình dữ liệu.

Edit field 'Mã khách hàng'

Edit field 'PGD/ Chi nhánh'

General	Options
Label: Mã khách hàng	<input type="radio"/> Chọn đơn vị <input checked="" type="radio"/> Phòng giao dịch số 1 - Chi nhánh Hà Nội <input type="radio"/> Phòng giao dịch số 2 - Chi nhánh Hà Nội <input type="radio"/> Phòng giao dịch số 3 - Chi nhánh Hà Nội <input type="radio"/> Phòng giao dịch số 1 - Chi nhánh HCM <input type="radio"/> Phòng giao dịch số 2 - Chi nhánh HCM + Add a new option
Override id? <input checked="" type="checkbox"/>	
Id: Ma_KH	
Required <input checked="" type="checkbox"/>	
Placeholder:	

Hình 3.20. Minh họa các đối tượng trên màn hình nhập dữ liệu.

Kiểm soát đơn vị duyệt

Version 1 Last updated by , Last Monday at 3:06 PM

Design Outcomes

CSV duyệt*

Mã khách hàng: \${Ma_KH}

Tổng giá trị đề xuất: \${GiaTri_DX} (DVT: \${Tiente})

Thời hạn vay: \${Thoihanvay}

Đơn vị: \${pgdchinhanh}

Hình 3.21. Minh họa màn hình xem dữ liệu.

3.4.2.2. Triển khai mô hình quy trình

Sau khi mô hình hóa quy trình, để người sử dụng có thể sử dụng/ thực thi quy trình. Ta cần đưa mô hình ra màn hình chức năng qua thành phần Apps của Activiti, thao tác này gọi là công khai mô hình (hay publish mô hình).

Thao tác tạo một Apps cho mô hình bằng cách Apps > Create App > Nhập thông tin > Create new app definition. Chi tiết như Hình 3.22.

Create a new app definition

You need to give a name for the new app definition and you may want to add a description at the same time.

App definition name

App definition key

Description

Cancel **Create new app definition**

Edit model details

Change any of the model properties below and then press Save to update the model.

Model name
 Phê duyệt tín dụng

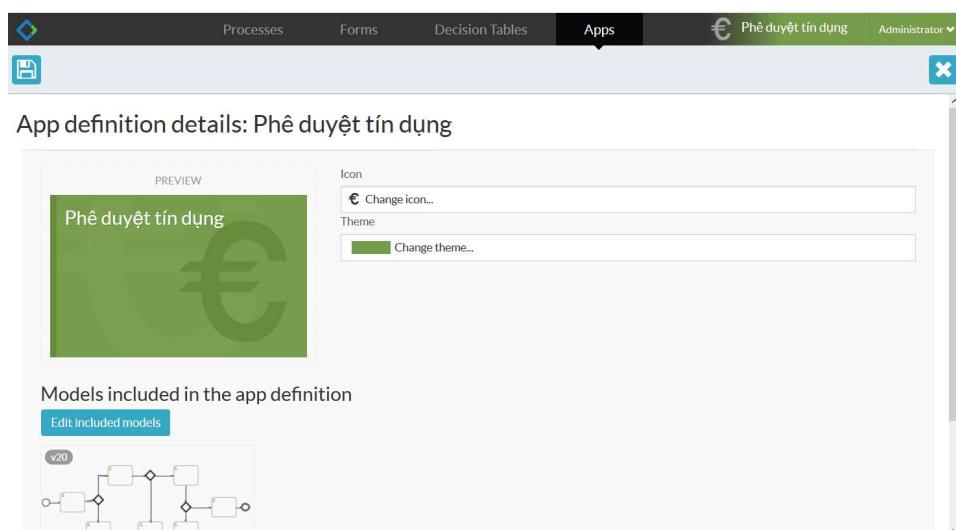
Model key
 PDTDapp

Description
 Quy trình phê duyệt tín dụng HO

Cancel **Save**

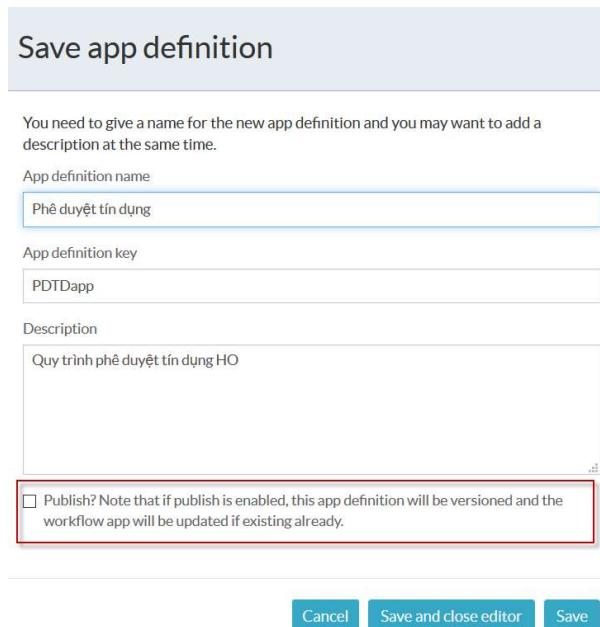
Hình 3.22. Minh họa việc tạo App cho mô hình.

Tiếp đến, ta thực hiện việc gán mô hình vào App: Edit included models > Close.
 Kết quả như Hình 3.23.



Hình 3.23. Minh họa chọn mô hình cho App.

Cuối cùng, publish App bằng cách: chọn Save > Publish hoặc trong màn hình App chọn Publish. Lưu ý khung viền đỏ trong Hình 3.24.



Hình 3.24. Minh họa publish cho App.

3.4.2.3. Thiết lập tập các quy tắc kiểm soát thẩm quyền

Nhập thông tin trong cơ sở dữ liệu để hàm PDP có thể sử dụng. Kết quả truy vấn từ cơ sở dữ liệu như Hình 3.25.

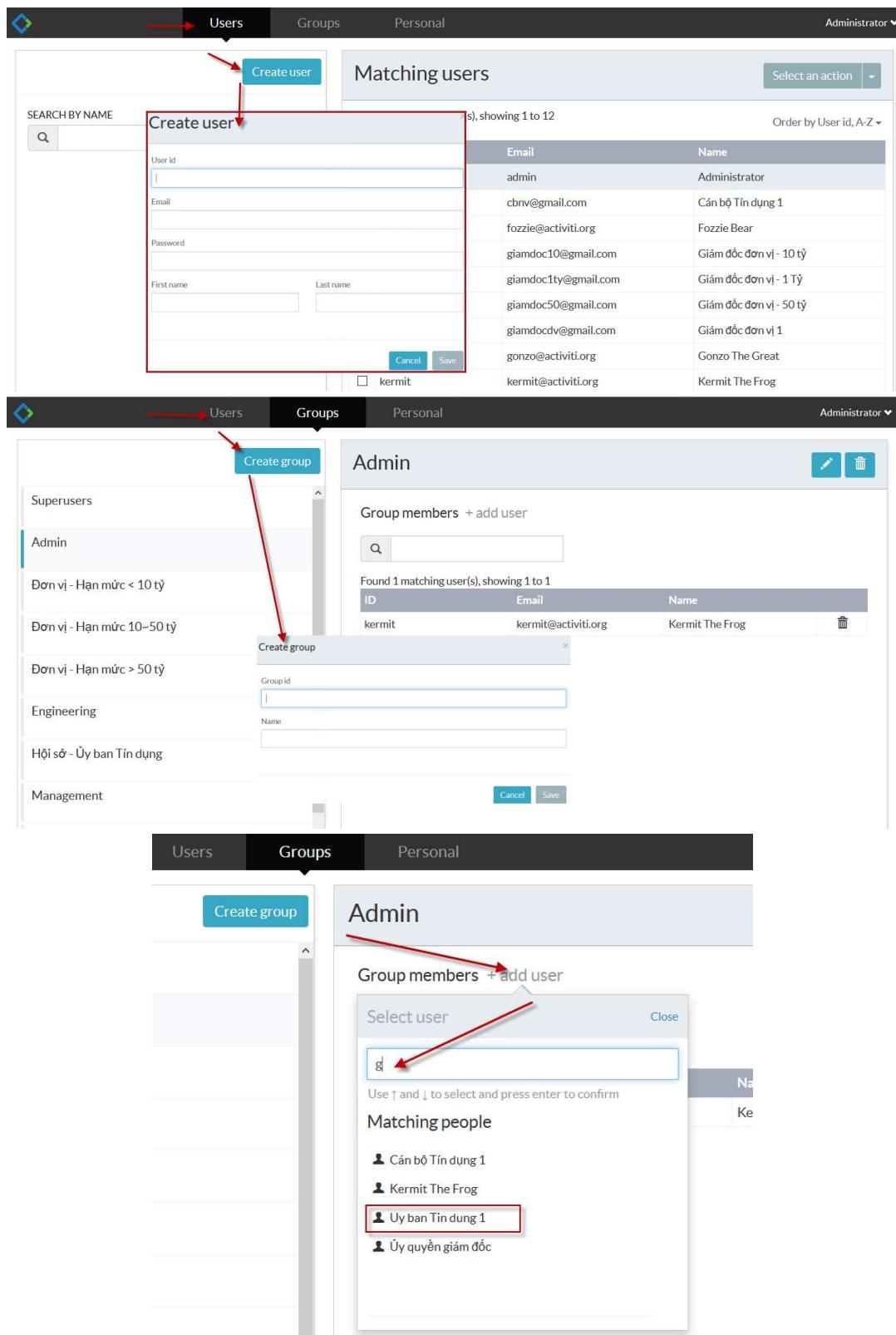
Select * from ACT_RU_USER_LIMIT												
	ID_	PROCESS_ID_	PRIORITY_	USER_GROUP_ID_	PROPERTY_	LIMIT_FLOOR_	LIMIT_CEILING_	ISUSER_	NOTES_	COMPANY_		
▶	1	4	QuytrinhTindung	...	1 hotd	... Noidung	... Khách-VIP	... N	... Tờ trình khách cấp, độ ưu tiên cao nhất.	
2	1	QuytrinhTindung	...	1 dvgd50	Giatri_DX	... 500000000001 N	... Hạn mức > 50 tỷ	
3	2	QuytrinhTindung	...	1 dvgd1050	Giatri_DX	... 100000000001	... 500000000000	... N	... Hạn mức 10~ 50 tỷ	
4	3	QuytrinhTindung	...	1 dvgd0110	Giatri_DX	... 0	... 100000000000	... N	... Hạn mức < 10 tỷ	

Hình 3.25. Thông tin quy tắc kiểm soát theo thẩm quyền.

3.4.3. Thực thi quy trình trên Activiti

Việc thực thi quy trình được triển khai như sau:

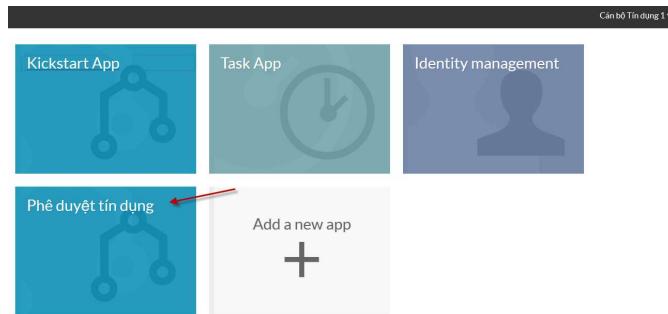
- ✓ Đăng nhập Activiti bằng user có quyền quản trị (admin)
- ✓ Tạo người sử dụng và các nhóm người sử dụng. Minh họa các bước lặp lượt trong Hình 3.26.



Hình 3.26. Minh họa tạo người sử dụng/ nhóm người sử dụng.

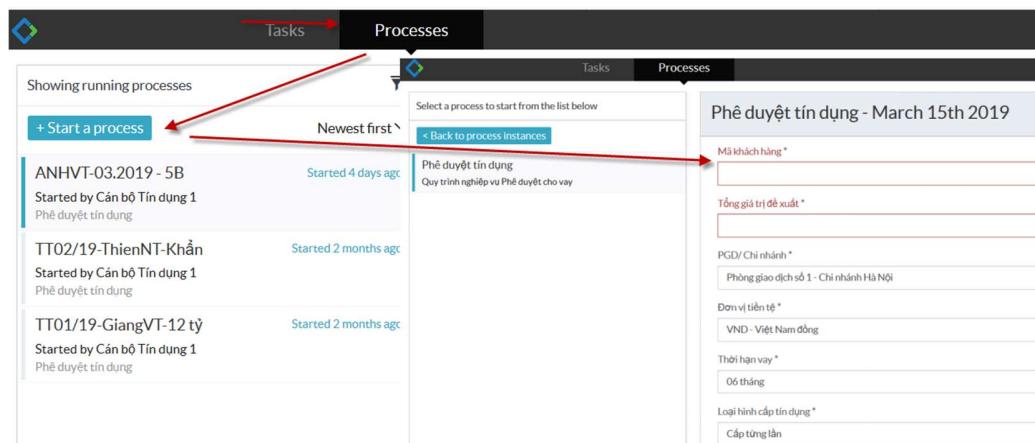
3.5. Kết quả thực nghiệm

Sau khi triển khai, quy trình sau khi mô hình hóa sẽ được thực hiện. Đầu tiên, người sử dụng được phân quyền trong tổ chức đăng nhập hệ thống có thể khởi tạo quy trình (Hình 3.27.). Thao tác: Apps > Phê duyệt tín dụng > Processes > Start a process.



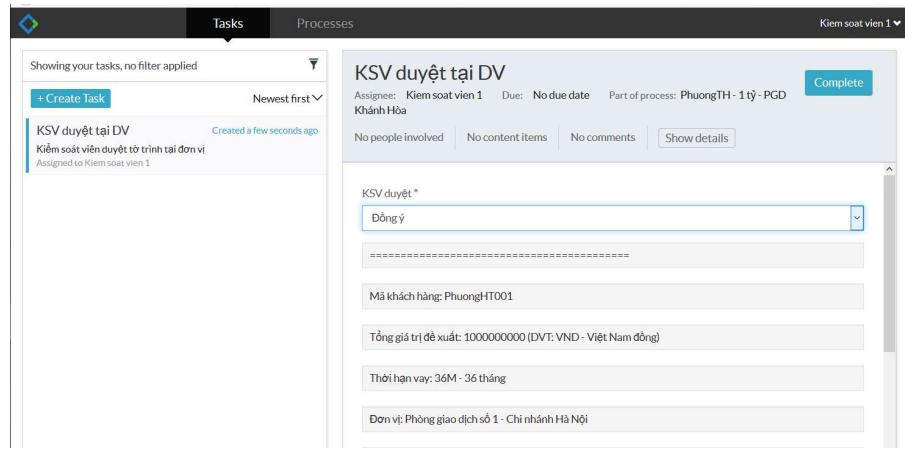
Hình 3.27. Minh họa tạo mới quy trình.

Tiếp theo, người sử dụng nhập các dữ liệu theo màn hình yêu cầu và xác định chuyển hồ sơ sang bước tiếp theo. Mặc định bước phê duyệt tiếp theo là Kiểm soát viên như Hình 3.28.



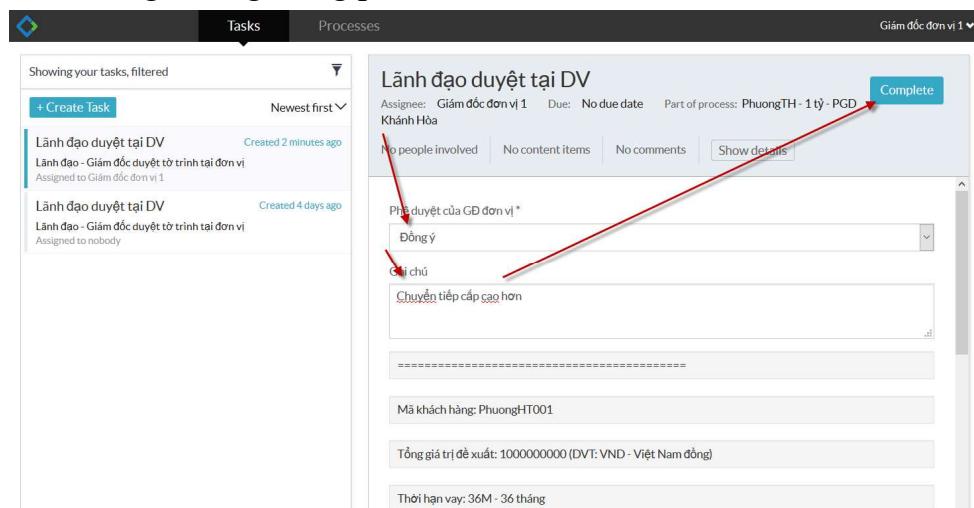
Hình 3.28. Minh họa nhập liệu cho quy trình.

Người sử dụng có quyền Kiểm soát viên sau khi đăng nhập, thực hiện phê duyệt qua màn hình nhiệm vụ của mình. Thao tác Tasks > Nhập nội dung phê duyệt > Complete (Hình 3.29., Hình 3.30.). Nếu không thấy Task mình cần thực hiện, người sử dụng có thể tìm kiếm task theo các tiêu chí: task được gán cho mình, task mà mình có liên quan, task mà mình là ứng viên xử lý.



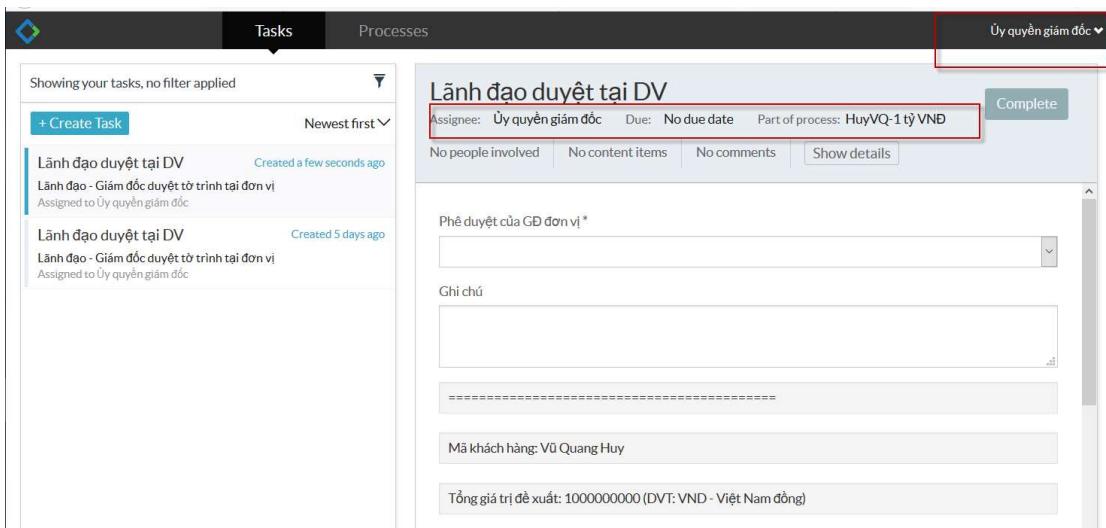
Hình 3.29. Minh họa việc hoàn thiện phê duyệt theo thẩm quyền.

Khi quy trình chuyển sang bước tiếp theo, chỉ người sử dụng có thẩm quyền phù hợp với các tiêu chí, thuộc tính dữ liệu mới có thể truy cập thông tin và thực hiện các thao tác tiếp theo trong quy trình. Về yêu cầu phê duyệt “theo thẩm quyền hạn mức” ta có thể nhìn thấy rõ ràng Giám đốc phê duyệt hạn mức 1 tỷ chỉ được phê duyệt hồ sơ với giá trị tương đương trong phạm vi.



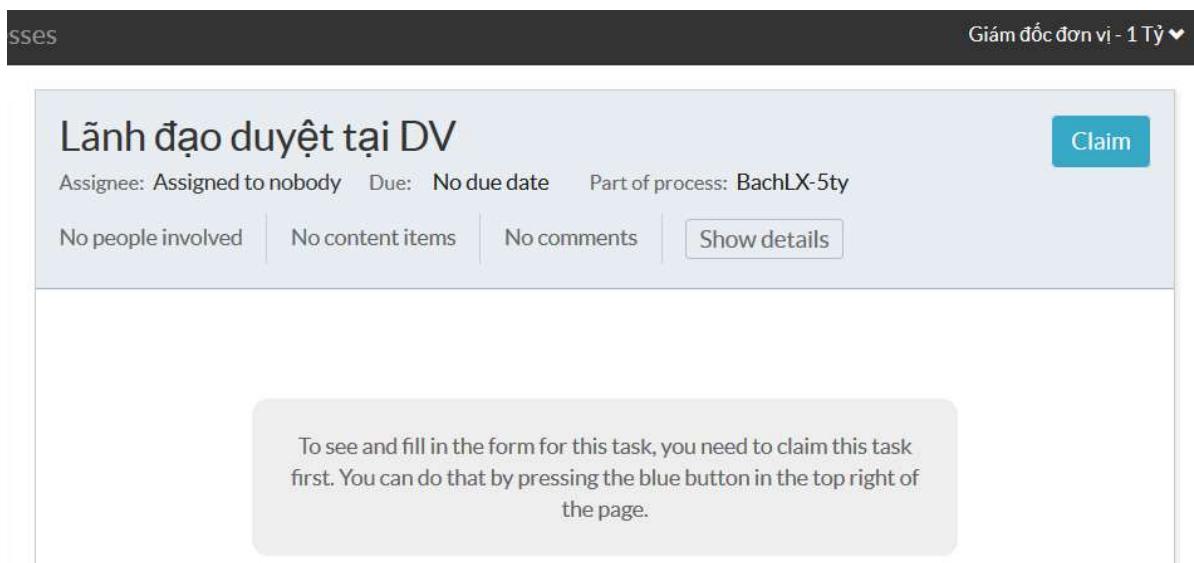
Hình 3.30. Minh họa hoàn thiện phê duyệt.

Trong trường hợp ủy quyền, người được ủy quyền sẽ được phân công phê duyệt nếu thỏa mãn bộ tiêu chí (ủy quyền còn hiệu lực trong khoảng thời gian cho phép và hồ sơ tín dụng thuộc thẩm quyền phê duyệt của người ủy quyền). Trong minh họa Hình 3.31. người ủy quyền (Ủy quyền giám đốc) có thể xem và phê duyệt 01 hồ sơ có giá trị 1 tỷ theo thẩm quyền của Giám đốc phê duyệt hạn mức 01 tỷ đồng.



Hình 3.31. Minh họa ủy quyền phê duyệt.

Người sử dụng được phân quyền phê duyệt ở bước sau có thể xem danh sách các Task trong quy trình đang xử lý. Nếu Task đó được gán cho nhóm mà người sử dụng liên quan, người sử dụng có thể tự nhận Task đó để xử lý, thao tác Processes > Claim như Hình 3.31.



Hình 3.32. Minh họa tự gán Task.

Sau khi quy trình hoàn thiện, thông tin sẽ được chuyển về cho người khởi tạo quy trình để hoàn thiện các công tác bên ngoài hệ thống. Người sử dụng khởi tạo cần truy cập mục Tasks và xem nội dung.

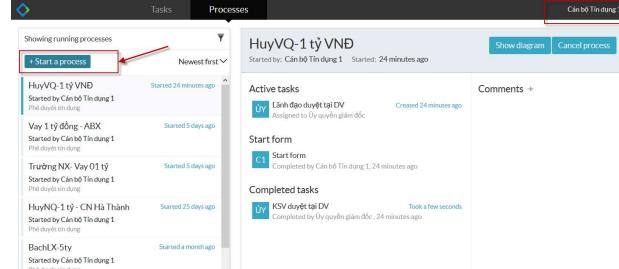
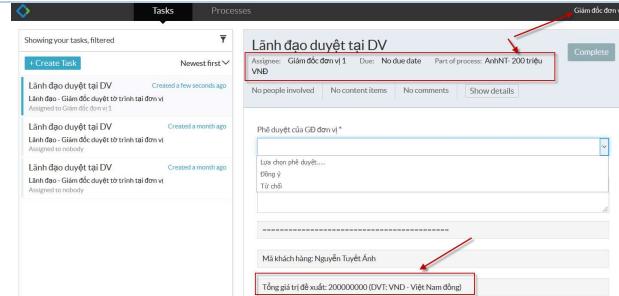
Thử nghiệm chương trình với các hồ sơ thực tế cho thấy, việc điều khiển quy trình đã thực hiện theo đúng luồng được thiết kế, với những thao tác đơn giản, dễ hiểu dễ thực hiện trên hệ thống. Số liệu thống kê được lấy từ cơ sở dữ liệu của Activiti như Hình 3.32.

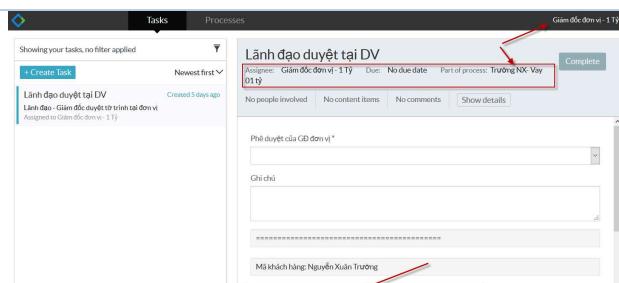
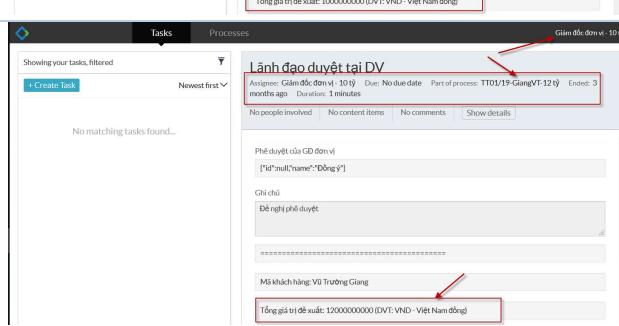
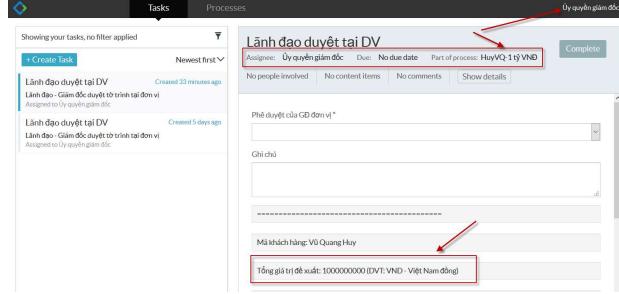
Select * from ACT_HI_TASKINST order by ID_desc ;										
ID_	PROC_DEF_ID_	TASK_DEF_KEY_	PROC_INST_ID_	EXECUTION_ID_	DESCRIPTION_	OWNER_	ASSIGNEE_	START_TIME_	END_TIME_	LAST_UPDATED_TIME_
1 112529	QuytrinhTindung:54:107509	KSVduyet	112507	112526	Kiểm soát viên duyệt tờ trình tại đơn vị	... kiemsoatvien	27-MAR-19 09.26.21.8
2 120145	Phéduyéttindung-Normal:4:120117	GDDuyet	120118	120137	Lãnh đạo - Giám đốc duyệt tờ trình tại đơn vị	... uyquyen	17-APR-19 05.07.20.98
3 132556	QuytrinhTindung:54:107509	GDDuyet	132529	132548	Lãnh đạo - Giám đốc duyệt tờ trình tại đơn vị	... uyquyen	22-APR-19 10.28.41.38
4 105185	QuytrinhTindung:52:40038	UBTDduyet	105143	105161	Ủy ban Tín dụng phê duyệt tờ trình tại đơn vị	... kiemsoatvien	22-JAN-19 11.21.18.85
5 115023	QuytrinhTindung:54:107509	KSVduyet	115001	115020	Kiểm soát viên duyệt tờ trình tại đơn vị	... kiemsoatvien	28-MAR-19 08.29.16.1
6 115028	QuytrinhTindung:54:107509	GDDuyet	115001	115020	Lãnh đạo - Giám đốc duyệt tờ trình tại đơn vị	... giamdoc1ty	28-MAR-19 08.29.31.3
7 120140	Phéduyéttindung-Normal:4:120117	KSVduyet111	120118	120137	Kiểm soát viên duyệt tờ trình tại đơn vị	... kiemsoatvien	17-APR-19 05.07.61
8 132584	QuytrinhTindung:54:107509	GDDuyet	132557	132576	Lãnh đạo - Giám đốc duyệt tờ trình tại đơn vị	... giamdocdv	22-APR-19 10.57.48.85
9 112505	QuytrinhTindung:54:107509	UBTDduyet	110001	110020	Ủy ban Tín dụng phê duyệt tờ trình tại đơn vị	... kiemsoatvien	27-MAR-19 09.11.53.6
10 110023	QuytrinhTindung:54:107509	KSVduyet	110001	110020	Kiểm soát viên duyệt tờ trình tại đơn vị	... kiemsoatvien	15-MAR-19 08.56.43.1
11 117523	QuytrinhTindung:54:107509	KSVduyet	117501	117520	Kiểm soát viên duyệt tờ trình tại đơn vị	... kiemsoatvien	17-APR-19 10.04.53.59
12 120028	QuytrinhTindung:54:107509	GDDuyet	120001	120020	Lãnh đạo - Giám đốc duyệt tờ trình tại đơn vị	... uyquyen	17-APR-19 03.46.57.31
13 132551	QuytrinhTindung:54:107509	KSVduyet	132529	132548	Kiểm soát viên duyệt tờ trình tại đơn vị	... uyquyen	22-APR-19 10.28.30.7C
14 105141	QuytrinhTindung:52:40038	KSVduyet	105120	105138	Kiểm soát viên duyệt tờ trình tại đơn vị	... kiemsoatvien	22-JAN-19 11.17.30.87
15 105164	QuytrinhTindung:52:40038	KSVduyet	105143	105161	Kiểm soát viên duyệt tờ trình tại đơn vị	... kiemsoatvien	22-JAN-19 11.18.41.32

Hình 3.33. Minh họa kết quả thực nghiệm.

Như vậy, với đa số các trường hợp thử nghiệm đã cho kết quả đảm bảo đúng yêu cầu chính sách truy cập mà luận văn đề ra, minh họa Bảng 3.2 cho ta thấy rõ hơn các kết quả kiểm thử testcase:

Bảng 3.2: Minh họa kết quả các testcase

Testcase	Mô tả	Kết quả	Ghi chú
Các vai trò được đảm bảo	Đảm bảo theo bảng phân quyền chức năng	- User canbonv: tạo hồ sơ/ chỉnh sửa hồ sơ - User kiemsoatvien: được phép kiểm tra, kiểm soát và chuyển tiếp - User giamdoc...: được phép phê duyệt/ từ chối phê duyệt - User uybantd: được phép phê duyệt/ từ chối phê duyệt	Đã đáp ứng
Tạo hồ sơ tín dụng	User canbonv có thể tạo hồ sơ, chỉnh sửa hồ sơ tín dụng		Đáp ứng 1 phần
Duyệt giao dịch giá trị dưới 1 tỷ	Chỉ có user giamdocdv có thể phê duyệt/ từ chối phê duyệt		Đã đáp ứng

Duyệt giao dịch giá trị đến 1 tỷ	Chỉ có user giamdoc1ty có thẻ phê duyệt/từ chối phê duyệt		Đã đáp ứng
Duyệt giao dịch giá trị trên 10 tỷ	Chỉ có user giamdoc10ty có thẻ phê duyệt/từ chối phê duyệt		Đã đáp ứng
Ủy quyền duyệt	User uyquyen được phê duyệt/từ chối phê duyệt hồ sơ theo thẩm quyền		Đã đáp ứng

3.6. Đánh giá kết quả vận dụng và thực nghiệm

Với đại đa số các trường hợp kiểm thử đạt kết quả đáp ứng, luận văn đã tích hợp thành công mô hình ABAC vào công cụ Activiti. Các trường hợp thử nghiệm đã cho thấy tính uyển chuyển trong việc kiểm soát truy cập tùy theo các thông tin thuộc tính hồ sơ tín dụng và thông tin quy tắc chính sách truy cập (trước đó). Điều này, chứng tỏ mô hình ABAC mà luận văn tìm hiểu và áp dụng đã được chứng minh là phù hợp với bài toán đặt ra và giải quyết được các yêu cầu mang tính động (khi phân quyền) trong nghiệp vụ thực tế.

Dù vậy, luận văn còn một số điểm hạn chế cần phải hoàn chỉnh. Đầu tiên là việc xử lý mang tính tổng quát, tức là khi số lượng thuộc tính tăng lên nhiều khi đó các luật xử lý yêu cầu phải phủ được hầu hết các trường hợp mà không cần phát triển (code) chi tiết cho từng luật. Tiếp theo, luận văn mới dừng lại ở việc tích hợp mô hình ABAC vào công cụ Activiti trong một vài sự kiện đặc trưng trong khi thực tế còn nhiều hoạt động/ sự kiện khác cần triển khai. Ngoài ra, để mang công cụ Activiti áp dụng trong hoạt động thực tiễn của Ngân hàng thương mại còn cần phải phát triển

và hoàn thiện nhiều tính năng bộ công cụ hơn nữa như: việc quản lý luân chuyển linh động hai chiều (chuyển tiếp và trả lại), việc quản lý các văn bản đính kèm, việc quản lý ủy quyền cần mềm dẻo hơn...

3.7. Tổng kết chương

Việc tích hợp thêm ABAC vào công cụ Activiti đã giải quyết được vấn đề kiểm soát truy cập dựa trên thuộc tính nói chung và bài toán Phê duyệt hồ sơ tín dụng (theo hạn mức phê duyệt) nói riêng. Các yêu cầu an ninh đã được mô hình hóa và cụ thể hóa từng thuộc tính khiến cho việc quản lý ma trận phân quyền trên thực tế rất phức tạp trở nên dễ dàng hơn. Cùng với việc cài đặt và triển khai công cụ Activiti khá trực quan và dễ dàng giúp việc áp dụng vào thực tế là rất khả thi và hiệu quả. Tuy nhiên, quá trình thực hiện cũng đã cho thấy các điểm còn hạn chế của công cụ và việc tích hợp ABAC phải thực hiện ở mức cao hơn và hoàn thiện hơn nữa để đáp ứng kỳ vọng đưa hệ thống vào sử dụng thực tế.

KẾT LUẬN

Quy trình nghiệp vụ trong tổ chức vốn có tính chất vô cùng quan trọng, ảnh hưởng tới kinh doanh và hoạt động sống còn của tổ chức. Tuy nhiên, tính chất phức tạp và nhiều ràng buộc trong quy trình làm nhiều công tác chưa thể quản lý bằng hệ thống thông tin. Đặc biệt trong lĩnh vực tài chính – ngân hàng, yêu cầu về tính tuân thủ và bảo mật lại càng được đặt yêu cầu cao. Với mong muốn xây dựng một công cụ hỗ trợ, luận văn đã thực hiện nghiên cứu và thực hiện được một số kết quả.

Luận văn đạt được các kết quả chính sau:

Thứ nhất, luận văn đã trình bày tổng quan về quy trình nghiệp vụ và tiêu chuẩn mô hình hóa nghiệp vụ bằng các khái niệm định nghĩa về BPMN. Cùng với đó là tiếp cận mô hình chính sách điều khiển truy cập thuộc tính (kiểm soát truy cập) ABAC để phân quyền truy cập cho người sử dụng hệ thống.

Thứ hai, với mục đích phát triển công cụ hỗ trợ, luận văn cũng đã nghiên cứu sử dụng và điều chỉnh bộ công cụ mã nguồn mở rất mạnh là Activiti. Việc mô hình hóa được quy trình nghiệp vụ BPMN, bộ công cụ Activiti là một lựa chọn tốt khi bắt đầu. Những tính năng sẵn có ở mức trung bình nhưng khả năng mở rộng của công cụ là rất lớn, giúp cho doanh nghiệp, tổ chức có thể dễ dàng tùy chỉnh cho phù hợp với mô hình của mình, đặc biệt ở việc tùy biến để đảm bảo an ninh, kiểm soát truy cập.

Thứ ba, bài toán “Phê duyệt hồ sơ tín dụng” là quy trình phức tạp trong nghiệp vụ ngân hàng. Luận văn đã hiện thực được mô hình hóa quy trình này, kết hợp với việc áp dụng mô hình điều khiển truy cập ABAC trên công cụ Activiti. Kết quả thực nghiệm với hàng chục trường hợp đặc trưng cho thấy việc tùy chỉnh công cụ đã đáp ứng được tính chất yêu cầu của nghiệp vụ này.

Cuối cùng, quá trình phát triển trên công cụ Activiti cũng cho thấy các điểm hạn chế của công cụ này (ngoài các ưu điểm đã biết). Với bài toán quy trình phức tạp trong thực tế, công cụ chắc chắn còn nhiều điểm phải khắc phục. Đầu tiên, với mô hình điều khiển ABAC, luận văn mới thực hiện kiểm soát với quy trình nghiệp vụ cơ bản nhất, ít thuộc tính; bài toán thực tế sẽ phức tạp hơn rất nhiều với nhiều trường hợp về yêu cầu truy cập như xem dữ liệu, thêm/bớt dữ liệu, chỉnh sửa dữ liệu... Việc phát triển công cụ cũng mới thực hiện cho một quy trình cụ thể là “Phê duyệt hồ sơ tín dụng”, muốn áp dụng chung cho các quy trình khác còn rất nhiều việc phải làm.

Với những kết quả đã đạt được, luận văn cũng đã có được hướng phát triển tiếp theo. Thứ nhất là hoàn thiện về mặt lý luận và tính khả thi cho mô hình điều khiển truy cập theo thuộc tính ABAC. Tiếp theo, hoàn thiện bộ công cụ Activiti khắc phục

các điểm hạn chế vốn có của một phiên bản mã nguồn mở cơ bản và bổ sung các yếu tố cần thiết theo nhu cầu thực tế: việc quản lý luân chuyển linh động hai chiều (chuyển tiếp và trả lại), việc quản lý các văn bản đính kèm, việc quản lý ủy quyền cần mềm dẻo hơn. Cuối cùng, việc tích hợp mô hình ABAC vào công cụ Activiti cần được thiết kế đầy đủ, logic; triển khai hoàn thiện hơn. Đây là cơ sở tốt cả về lý thuyết và thực tế, là hướng đi tương lai để có thể triển khai một cách toàn vẹn hệ thống đáp ứng tốt yêu cầu của doanh nghiệp/ tổ chức.

Trong quá trình học tập và thực hiện luận văn, tôi đã thu được rất nhiều kiến thức bổ ích về hệ thống quản lý quy trình nghiệp vụ trong doanh nghiệp cũng như kiến thức về các kỹ thuật phát triển phần mềm, ứng dụng công nghệ. Tuy nhiên, do kiến thức có hạn nên trong luận văn không thể tránh khỏi những sai sót, khiếm khuyết, tôi rất mong nhận được sự góp ý của quý Thầy, Cô để luận văn được hoàn thiện hơn.

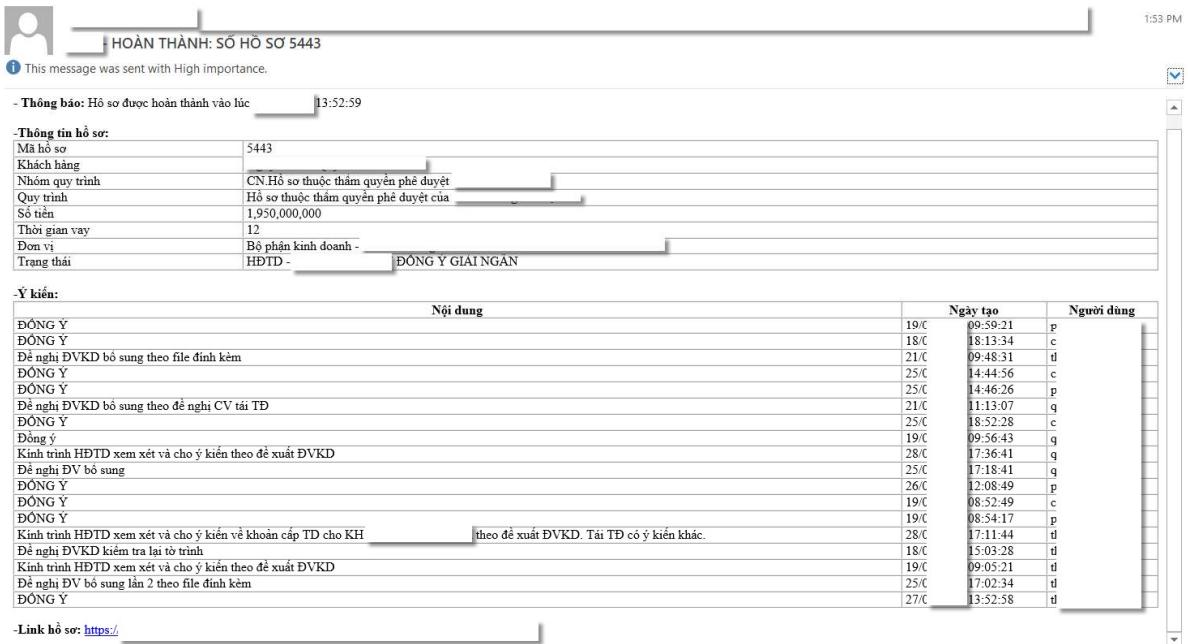
TÀI LIỆU THAM KHẢO

1. Dr. Zakir Laliwala, Irshad Mansuri (2014) “*Activiti 5.x Business Process Management Beginner's Guide*”, Packt Publishing, pp. 8-10, 109-119.
 2. Jan vom Brocke, Jörg Becker, Alessio Maria Braccini... (2010) “*Current and Future Issues in BPM Research: A European Perspective from the ERCIS Meeting*”, ERCIS Meeting 2010
 3. Milson Munakami (2016) “*Developing an ABAC-Based Grant Proposal Workflow Management System*”, Boise State University, pp. 4-22.
 4. OMG Document (2009) “*Business Process Model and Notation*”, OMG, pp. 14-33.
 5. Stephen A. White, Derek Miers (2008) “*BPMN Modeling and Reference Guide: Understanding and Using BPMN*”, Future Strategies Inc., Book Division, pp. 27-33.
 6. Tom Baeyens, Joram Barrez (2012) “*Activiti in Action - Executable business processes in BPMN 2.0*”, Manning Publications Co, pp. 4-9.
 7. Vincent C. Hu, David Ferraiolo, Rick Kuhn (2013) “*Guide to Attribute Based Access Control (ABAC) Definition and Considerations*”, NIST Special Publication, pp. 4-16.
 8. Sunil (2017), tài liệu hướng dẫn cài đặt Activiti 6.0 <https://www.youtube.com/watch?v=pNZGyMGEMd0>, truy cập lần cuối 02.2019
 9. Bộ Thông tin và Truyền thông – Cục Tin học hóa (2017), tài liệu về tiêu chuẩn BPMN, <http://aita.gov.vn/to%CC%89ng-quan-ve%CC%80-tieu-chuan-ky-hieu-va-mo-hinh-hoa-quy-trinh-nghiep-vu-business-process-modelling-and-notation-bpmn-phien-ban-2.0>, truy cập lần cuối 12.2018
 10. Redhat, tài liệu mã nguồn mở BPM, <https://www.jbpm.org/>, truy cập lần cuối 20.01.2019
 11. OMG, tài liệu mô tả BPMN, <https://www.omg.org/spec/BPMN/2.0/>, truy cập lần cuối 20.01.2019
 12. Activiti (2017), tài liệu hướng dẫn lập trình với Activiti project, <https://www.activiti.org/userguide/>, truy cập lần cuối 20.01.2019
 13. Softwaresuggest (2017), so sánh các hệ thống BPMN mã nguồn mở tốt nhất, <https://www.softwaresuggest.com/blog/top-free-open-source-bpm-software/>, truy cập lần cuối 21.03.2019
-

14. CMC SISG, thông tin hệ thống phê duyệt tín dụng, <https://www.cmcsisg.vn/giai-phap-dich-vu/giai-phap-cong-nghe-thong-tin/giai-phap-cntt-chuyen-nganh/giai-phap-chuyen-nganh-cho-mang-thi-truong-fsi/nhom-giai-phap-tu-dong-hoa-quy-trinh/he-thong-khoi-tao-phe-duyet-khoan-vay/>, truy cập lần cuối 21.03.2019
15. Sacombank (2019), thông tin hệ thống phê duyệt tín dụng, <https://www.sacombank.com.vn/company/Pages/Sacombank-chinh-thuc-trien-khai-he-thong-khoi-tao-phe-duyet-va-quan-ly-cap-tin-dung-LOS.aspx>, truy cập lần cuối 21.03.2019

PHỤ LỤC

- Yêu cầu quy trình của “Phê duyệt hồ sơ tín dụng”



- Activiti Form Services

```

public void completeTaskForm(String taskId, CompleteFormRepresentation completeTaskFormRepresentation) {

    // Get the form definition //Tungnt: Complete a task Start_2
    Task task = taskService.createTaskQuery().taskId(taskId).singleResult();

    //
    if (task == null) {
        throw new NotFoundException("Task not found with id: " + taskId);
    }

    FormDefinition formDefinition = formRepositoryService.getFormDefinitionById(completeTaskFormRepresentation.getFormId());

    User currentUser = SecurityUtils.getCurrentUserObject();

    if (!permissionService.isTaskOwnerOrAssignee(currentUser, taskId)) {
        if (!permissionService.validateIfUserIsInitiatorAndCanCompleteTask(currentUser, task)) {
            throw new NotPermittedException();
        }
    }

    // Extract raw variables and complete the task //TungNT: Lenh lay danh sach cac truong trong Form khi Complete
    Map<String, Object> variables = formService.getVariablesFromFormSubmission(formDefinition, completeTaskFormRepresentation.getValues(),
        completeTaskFormRepresentation.getOutcome());

    formService.storeSubmittedForm(variables, formDefinition, task.getId(), task.getProcessInstanceId());

    taskService.complete(taskId, variables); //Tungnt: Complete a task Start_3 (thuc su update query)
    //Bo sung phan Assignee neu dang o buoc GDDVduyet
    if(task.getFormKey().equalsIgnoreCase("KSVduyet")){
        UpdateAssigneeABAC(task,task.getProcessInstanceId());
    }
}

```

```

//Tungnt: Function added
public void UpdateAssigneeABAC(Task task, String processInstanceId)
{
    try {
        // thuc hien connect DB
        Connection dbConnection = null;
        PreparedStatement preparedStatement = null;
        System.out.println("Process file for: ");
        //Connect DB
        new ConnectDB().getConn();
        dbConnection = ConnectDB.conn;
        //Connect successed
        if (dbConnection != null) {
            Statement stmt = dbConnection.createStatement();
            dbConnection.setAutoCommit(false);
            // Get the form variables
            Map<String, Object> varOfForms = new HashMap<String, Object>();
            if (task.getProcessInstanceId() != null) {
                List<HistoricVariableInstance> variableInstances = historyService.createHistoricVariableInstanceQuery()
                    .processInstanceId(task.getProcessInstanceId())
                    .list();

                for (HistoricVariableInstance historicVariableInstance : variableInstances) {
                    varOfForms.put(historicVariableInstance.getVariableName(), historicVariableInstance.getValue());
                }
            }
            String process_Def_id =
task.getProcessDefinitionId().substring(0,task.getProcessDefinitionId().indexOf(":"));
            String queryUserLimit= "Select * from ACT_RU_USER_LIMIT where PROCESS_ID_ = '"+process_Def_id_+"' ORDER
BY PRIORITY_ ";
            String queryUpdate= "";
            String queryTaskVar= "";
            //Get user limit rule
            ResultSet rsIm= stmt.executeQuery(queryUserLimit);
            String property_ = "";
            String limit_floor_ = "";
            String limit_ceiling_ = "";
            String user_group_id_ = "";
            String isuser_ = "";
            boolean isUpdate_ = false;
            while (rsIm.next()) {
                property_ = rsIm.getString("PROPERTY_");
                limit_floor_ = rsIm.getString("LIMIT_FLOOR_");
                limit_ceiling_ = rsIm.getString("LIMIT_CEILING_");
                user_group_id_ = rsIm.getString("USER_GROUP_ID_");
                isuser_ = rsIm.getString("ISUSER_");
                //
                try {
                    for (String key : varOfForms.keySet()) {
                        Object conditionValue_ = varOfForms.get(key);
                        //String conditionValue_ = (String) value;

                        if (key.equalsIgnoreCase(property_)) {
                            if (property_.equalsIgnoreCase("Noidung")) {
                                if (limit_floor_.equals(conditionValue_)) {
                                    UpdateTaskAssign(task, isuser_, user_group_id_, processInstanceId, stmt);
                                    break;
                                }
                            } else {
                                //if ceiling & floor: not null
                                if (limit_ceiling_ == null) limit_ceiling_ = "0";
                                if (Long.parseLong(limit_floor_) < Long.parseLong(limit_ceiling_)) {
                                    //floor < formValue < ceiling
                                    if (Long.parseLong(limit_floor_) <= Long.parseLong(conditionValue_.toString())
                                        && Long.parseLong(limit_ceiling_) >= Long.parseLong(conditionValue_.toString())) {
                                        UpdateTaskAssign(task, isuser_, user_group_id_, processInstanceId, stmt);
                                        break;
                                    }
                                } else {
                                    //floor < formValue
                                    if (Long.parseLong(limit_floor_) <= Long.parseLong(conditionValue_.toString())) {
                                        UpdateTaskAssign(task, isuser_, user_group_id_, processInstanceId, stmt);
                                        break;
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    } catch (Exception ex) {
        isUpdate_ = false;
    }
    //Break if updated
    if (isUpdate_)
        break;
}

```

```

    } //end while userlimit

    //
    stmt.close();
    rsIm.close();
    dbConnection.close();
    new ConnectDB().closeConn();
}

} catch (SQLException ex) {
    System.out.println("SMS:-----Error has raised!");
}
finally {

}

}

private void UpdateTaskAssign(Task task, String isUser, String groupOrUser, String processInstanceId, Statement stmt)
{
    String queryUpdate = "";
    try{
        if (isUser.equalsIgnoreCase("Y")){
            String user_assignee = groupOrUser;
            //Get info's delegate, if any
            DateFormat dateFormat = new SimpleDateFormat("yyyyMMdd");
            Date date = new Date();
            String date_now = dateFormat.format(date);
            ResultSet rsDelegate= stmt.executeQuery("select * from ACT_RU_USER_DELEGATE where user_send_ = '" +
user_assignee +"' and date_from_ <=' "+date_now+"' and date_to_ >= '"+date_now+"'");

            String userDel_ = "";
            while (rsDelegate.next()) {
                userDel_ = rsDelegate.getString("USER_GET_");
                break;
            }
            if (!userDel_.isEmpty())
                user_assignee = userDel_;
            //Assignee for task
            queryUpdate = "Update ACT_RU_TASK set ASSIGNEE_ = '" + user_assignee +"' where PROC_INST_ID_ = " +
processInstanceId;
            stmt.executeQuery(queryUpdate);
            queryUpdate = "Update ACT_HI_TASKINST set ASSIGNEE_ = '" + user_assignee +"' where PROC_INST_ID_ = " +
processInstanceId;
            stmt.executeQuery(queryUpdate);
        }else
        {
            //Get maxID
            ResultSet rsIm= stmt.executeQuery("select max(id_) maxID from ACT_RU_IDENTITYLINK");
            String maxID_ = "";
            while (rsIm.next()) {
                maxID_ = rsIm.getString("MAXID");
                break;
            }
            //Get taskID
            String queryTaskID= "Select * from ACT_RU_TASK where PROC_INST_ID_ = '"+processInstanceId+"'";
            ResultSet rsImTask= stmt.executeQuery(queryTaskID);
            String taskID_ = "";
            while (rsImTask.next()) {
                taskID_ = rsImTask.getString("ID_");
            }
            //Insert
            Integer maxID = Integer.parseInt(maxID_);
            maxID = maxID + 1;
            queryUpdate = "insert into ACT_RU_IDENTITYLINK (ID_,GROUP_ID_,TYPE_,TASK_ID_) values
("+maxID+","+groupOrUser+",'candidate','"+ taskID_ +"')";
            stmt.executeQuery(queryUpdate);
            queryUpdate = "insert into ACT_HI_IDENTITYLINK (ID_,GROUP_ID_,TYPE_,TASK_ID_) values
("+maxID+","+groupOrUser+",'candidate','"+ taskID_ +"')";
            stmt.executeQuery(queryUpdate);
        }
    } catch (SQLException ex) {
        System.out.println("SMS:-----Error has raised!");
    }
}

```

• Chính sửa TaskUtil

```

public static void fillPermissionInformation(TaskRepresentation taskRepresentation, TaskInfo task, User
currentUser,
    IdentityService identityService, HistoryService historyService, RepositoryService repositoryService) {

    String processInstanceIdStartUserId = null;
    boolean initiatorCanCompleteTask = true;
    boolean isMemberOfCandidateGroup = false;
    boolean isMemberOfCandidateUsers = false;

```

```

if (task.getProcessInstanceId() != null) {

    HistoricProcessInstance historicProcessInstance =
    historyService.createHistoricProcessInstanceQuery().processInstanceId(task.getProcessInstanceId()).singleResult
    ();

    if (historicProcessInstance != null) {
        processInstanceIdStartUserId = historicProcessInstance.getStartUserId();
        BpmnModel bpmnModel = repositoryService.getBpmnModel(task.getProcessDefinitionId());
        FlowElement flowElement = bpmnModel.getFlowElement(task.getTaskDefinitionKey());
        if (flowElement != null && flowElement instanceof UserTask) {
            UserTask userTask = (UserTask) flowElement;
            List<ExtensionElement> extensionElements = userTask.getExtensionElements().get("initiator-can-
complete");
            if (CollectionUtils.isNotEmpty(extensionElements)) {
                String value = extensionElements.get(0).getElementText();
                if (StringUtils.isNotEmpty(value)) {
                    initiatorCanCompleteTask = Boolean.valueOf(value);
                }
            }

            Map<String, Object> variableMap = new HashMap<String, Object>();
            if ((CollectionUtils.isNotEmpty(userTask.getCandidateGroups()) &&
userTask.getCandidateGroups().size() == 1
            &&
userTask.getCandidateGroups().get(0).contains("${taskAssignmentBean.assignTaskToCandidateGroups('')"))
            || (CollectionUtils.isNotEmpty(userTask.getCandidateUsers()) &&
userTask.getCandidateUsers().size() == 1
            &&
userTask.getCandidateUsers().get(0).contains("${taskAssignmentBean.assignTaskToCandidateUsers('')")) {
                List<HistoricVariableInstance> processVariables =
                historyService.createHistoricVariableInstanceQuery().processInstanceId(task.getProcessInstanceId()).list();
                if (CollectionUtils.isNotEmpty(processVariables)) {
                    for (HistoricVariableInstance historicVariableInstance : processVariables) {
                        variableMap.put(historicVariableInstance.getVariableName(),
                        historicVariableInstance.getValue());
                    }
                }
            }

            if (CollectionUtils.isNotEmpty(userTask.getCandidateGroups())) {
                List<Group> groups = identityService.createGroupQuery().groupMember(currentUser.getId()).list();
                if (CollectionUtils.isNotEmpty(groups)) {
                    List<String> groupIds = new ArrayList<String>();
                    if (userTask.getCandidateGroups().size() == 1 &&
userTask.getCandidateGroups().get(0).contains("${taskAssignmentBean.assignTaskToCandidateGroups('')"))
                    {
                        String candidateGroupString = userTask.getCandidateGroups().get(0);
                        candidateGroupString =
candidateGroupString.replace("${taskAssignmentBean.assignTaskToCandidateGroups('', '')");
                        candidateGroupString = candidateGroupString.replace("'", execution)", "");
                        String groupsArray[] = candidateGroupString.split(",");
                        for (String group : groupsArray) {
                            if (group.contains("field(")) {
                                String fieldCandidate = group.trim().substring(6, group.length() - 1);
                                Object fieldValue = variableMap.get(fieldCandidate);
                                if (fieldValue != null && NumberUtils.isNumber(fieldValue.toString())) {
                                    groupIds.add(fieldValue.toString());
                                }
                            } else {
                                groupIds.add(group);
                            }
                        }
                    } else {
                        groupIds.addAll(userTask.getCandidateGroups());
                    }
                    for (Group group : groups) {
                        if (groupIds.contains(String.valueOf(group.getId()))) {
                            isMemberOfCandidateGroup = true;
                            break;
                        }
                    }
                }
            }

            if (CollectionUtils.isNotEmpty(userTask.getCandidateUsers())) {
                if (userTask.getCandidateUsers().size() == 1 &&
userTask.getCandidateUsers().get(0).contains("${taskAssignmentBean.assignTaskToCandidateUsers('')"))
                {
                    String candidateUserString = userTask.getCandidateUsers().get(0);

```

```

candidateUserString =
candidateUserString.replace("${taskAssignmentBean.assignTaskToCandidateUsers('', '')");
candidateUserString = candidateUserString.replace("'", execution)}", "");
String users[] = candidateUserString.split(",");
for (String user : users) {
    if (user.contains("field")) {
        String fieldCandidate = user.substring(6, user.length() - 1);
        Object fieldValue = variableMap.get(fieldCandidate);
        if (fieldValue != null && NumberUtils.isNumber(fieldValue.toString()) &&
String.valueOf(currentUser.getId()).equals(fieldValue.toString())) {

            isMemberOfCandidateGroup = true;
            break;
        }

    } else if (user.equals(String.valueOf(currentUser.getId()))) {
        isMemberOfCandidateGroup = true;
        break;
    }
}

} else if (userTask.getCandidateUsers().contains(String.valueOf(currentUser.getId()))) {
    isMemberOfCandidateUsers = true;
}
}

if (!isMemberOfCandidateGroup && !isMemberOfCandidateUsers) {
List<String> candidateGroupIds = new ArrayList<String>();
List<HistoricIdentityLink> links =
(List<HistoricIdentityLink>)historyService.getHistoricIdentityLinksForTask(task.getId());
for (HistoricIdentityLink historicIdentityLink : links) {
    if (!isMemberOfCandidateUsers &&
StringUtils.isNotEmpty((CharSequence)historicIdentityLink.getUserId()) &&
String.valueOf(currentUser.getId()).equals(historicIdentityLink.getUserId()) &&
"candidate".equalsIgnoreCase(historicIdentityLink.getType())) {
        isMemberOfCandidateUsers = true;
    }
    else {
        if (!StringUtils.isNotEmpty((CharSequence)historicIdentityLink.getGroupId()) ||

!"candidate".equalsIgnoreCase(historicIdentityLink.getType())) {
            continue;
        }
        candidateGroupIds.add(historicIdentityLink.getGroupId());
    }
}

//Tungnt OLD: List<GroupRepresentation> groups2 = (List<GroupRepresentation>)new
UserRepresentation(currentUser).getGroups();
//Change the way get GroupID of user
UserRepresentation userRepresentation = new UserRepresentation(currentUser);
List<Group> groups = identityService.createGroupQuery().groupMember(currentUser.getId()).list();
List<GroupRepresentation> groups2 = new ArrayList<GroupRepresentation>();
if (groups != null) {
    for (Group group : groups) {
        groups2.add(new GroupRepresentation(group));
    }
}
//End changed
if (groups2 != null) {
    for (GroupRepresentation group3 : groups2) {
        if (candidateGroupIds.contains(group3.getId().toString())) {
            isMemberOfCandidateGroup = true;
            break;
        }
    }
}
}

taskRepresentation.setProcessInstanceId(processInstanceId);
taskRepresentation.setInitiatorCanCompleteTask(initiatorCanCompleteTask);
taskRepresentation.setMemberOfCandidateGroup(isMemberOfCandidateGroup);
taskRepresentation.setMemberOfCandidateUsers(isMemberOfCandidateUsers);
}
}

```

- Tạo bảng chính sách trong cơ sở dữ liệu

```
-- Create table
create table ACT_RU_USER_LIMIT
(
    id_          NVARCHAR2(64) not null,
    process_id_  NVARCHAR2(255),
    priority_   INTEGER,
    user_group_id_ NVARCHAR2(255),
    property_   NVARCHAR2(255),
    limit_floor_ NVARCHAR2(255),
    limit_ceiling_ NVARCHAR2(255),
    isuser_     NVARCHAR2(255),
    notes_      NVARCHAR2(255),
    company_    NVARCHAR2(255)
)
tablespace USERS
pctfree 10
initrans 1
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
-- Create/Recreate indexes
create index ACT_IDX_USER_ID_LIMIT on ACT_RU_USER_LIMIT (USER_GROUP_ID_)
tablespace USERS
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
-- Create/Recreate primary, unique and foreign key constraints
alter table ACT_RU_USER_LIMIT
add primary key (ID_)
using index
tablespace USERS
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
-- Create table
create table ACT_RU_USER_DELEGATE
(
    id_          NVARCHAR2(64) not null,
    user_send_   NVARCHAR2(255),
    user_get_   NVARCHAR2(255),
    date_from_  NVARCHAR2(255),
    date_to_    NVARCHAR2(255),
    isactive_   NVARCHAR2(255)
)
tablespace USERS
pctfree 10
initrans 1
maxtrans 255
storage
(
    initial 64K
    next 1M
```

```
    minextents 1
    maxextents unlimited
);
-- Create/Recreate indexes
create index ACT_IDX_USER_SEND on ACT_RU_USER_DELEGATE (USER_SEND_)
  tablespace USERS
  pctfree 10
  initrans 2
  maxtrans 255
  storage
(
  initial 64K
  next 1M
  minextents 1
  maxextents unlimited
);
-- Create/Recreate primary, unique and foreign key constraints
alter table ACT_RU_USER_DELEGATE
  add primary key (ID_)
  using index
  tablespace USERS
  pctfree 10
  initrans 2
  maxtrans 255
  storage
(
  initial 64K
  next 1M
  minextents 1
  maxextents unlimited
);
```