*Kỷ yếu Hội nghị Quốc gia lần thứ VIII về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR); Hà Nội, ngày 9-10/7/2015*
*DOI: 10.15625/vap.2015.000198*

# PHƯƠNG PHÁP SINH TỰ ĐỘNG CA KIỂM THỬ TỪ MÔ HÌNH CA SỬ DỤNG

**Chu Thị Minh Huệ[1, 2], Đặng Đức Hạnh[2], Nguyễn Ngọc Bình[3]**

[1] Khoa Công nghệ thông tin, Trường Đại học Sư phạm Kỹ thuật Hưng Yên

[2] Khoa Công nghệ thông tin, Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội

[3] Viện Quốc tế Pháp ngữ - Đại học Quốc gia Hà Nội

*Huectm@gmail.com, hanhdd@vnu.edu.vn, nnbinh@vnu.edu.vn*

***TÓM TẮT*** *- Mô hình ca sử dụng (Use Case) nắm bắt các chức năng mà hệ thống phần mềm đáp ứng. Hiện nay, mô hình ca sử dụng thông thường được biểu diễn bằng biểu đồ biểu đồ Use Case như trong UML và tài liệu hóa các đặc tả của từng ca sử dụng (Use Case Specification) ở dạng văn bản. Đặc tả ca sử dụng được xây dựng trong pha đặc tả yêu cầu phần mềm và nó có thể được sử dụng cho việc tạo các ca kiểm thử (Test Case) ở mức kiểm thử hệ thống (System Testing). Sử dụng mô hình ca sử dụng để thiết kế kiểm thử (Test Design) ở giai đoạn sớm trong vòng đời phát triển phần mềm sẽ làm giảm chi phí cho phát triển hệ thống. Đặc tả ca sử dụng thường được tài liệu hóa bằng ngôn ngữ tự nhiên. Vì vậy việc sinh tự động các ca kiểm thử từ các kịch bản của ca sử dụng vẫn còn là một khoảng cách lớn. Trong bài báo này, chúng tôi đề xuất một phương pháp cho đặc tả ca sử dụng bằng một mô hình và hướng dẫn sinh tự động các ca kiểm thử tự động từ mô hình này. Trong đó chúng tôi đề xuất một ngôn ngữ mô hình để mô hình hóa đặc tả ca sử dụng USL (Use Case Specification Language) và sinh tự động các ca kiểm thử. Ngôn ngữ USL được mở rộng từ biểu đồ hoạt động trong UML và thêm vào khái niệm cam kết (Contract) cho phép đặc tả chi tiết các hành động và điều kiện gác trong luồng chuyển. Với cách tiếp cận này, chúng tôi xây dựng một MetaModel mô tả cú pháp trừu tượng của ngôn ngữ USL. Từ đó đem lại khả năng chuyển trực tiếp từ mô hình đặc tả ca sử dụng sang các ca kiểm thử.*

***Từ khóa*** *- Sinh các ca kiểm thử tự động, đặc tả ca sử dụng, cam kết, USL, Use Case Specification, Automatic Test Generation, Contract.*

## I. GIỚI THIỆU

Đặc tả đặc tả ca sử dụng nắm bắt các nghiệp vụ của ca sử dụng, đặc tả này thường được mô tả bằng ngôn ngữ tự nhiên theo định dạng chuẩn như đề xuất trong [ 2 ]. Trong UML đặc tả ca sử dụng được mô tả bằng ngôn ngữ tự nhiên và được liên kết lỏng lẻo với biểu đồ ca sử dụng. Sử dụng ngôn ngữ tự nhiên để mô tả cho phép các chuyên gia miền và người sử dụng không hiểu biết kỹ thuật dễ dàng hiểu được đặc tả nghiệp vụ của ca sử dụng, tuy nhiên hạn chế của ngôn ngữ tự nhiên là khó tự động hóa. Trong phát triển phần mềm đặc tả ca sử dụng được sử dụng để xác định các kịch bản kiểm thử và các ca kiểm thử cho kiểm thử mức hệ thống, thường kiểm thử viên tiến hành đọc đặc tả và xác định các ca kiểm thử thủ công. Một giải pháp tốt cho công nghiệp phần mềm là có thể tự động chuyển trực tiếp từ đặc tả ca sử dụng sang các kịch bản kiểm thử và các ca kiểm thử. Vì vậy đặc tả ca sử dụng cần hướng tới một phương pháp đặc tả hình thức để máy có thể hiểu được. Trong bài báo này chúng tôi đề xuất một ngôn ngữ mô hình hóa USL cho đặc tả ca sử dụng với mục đích sinh kịch bản kiểm thử và ca kiểm thử từ mô hình.

Trong [ 2 ], chuỗi hành động trong đặc tả ca sử dụng là một chuỗi hành động tương tác giữa tác nhân (Actor) và hệ thống (system) để thực hiện một ca sử dụng. Chuỗi hành động tương tác gồm có một chuỗi tương tác chính đúng đắn (Basic Flow) và các chuỗi tương tác rẽ nhánh ngoại lệ (Alternate Flows). Rất nhiều nghiên cứu chỉ dẫn phương pháp hình thức cho đặc tả ca sử dụng với mục đích sinh tự động ca kiểm thử. Một số nghiên cứu tập trung vào đặc tả chuỗi hành động tương tác như trong [11] sử dụng ngôn ngữ hữu hạn trạng thái để đặc tả (Abstract State Machine Language - ASML), trong [6] sử dụng biểu đồ hoạt động để đặc tả bằng cách sử dụng stereotypes để đặc tả riêng hành động của tác nhân và hành động của hệ thống, trong [14] sử dụng biểu đồ tuần tự để đặc tả, trong [15] sử dụng cấu trúc File XML để đặc tả, trong [5,7] đề xuất một cấu Contract để đặc tả tiền điều kiện và hậu điều kiện của ca sử dụng bằng biểu thức logic ràng buộc trên các hành động. Các nghiên cứu trên chủ yếu tập chung vào mô hình hóa các chuỗi hành động tương tác bằng các phương pháp khác nhau, nhưng chưa đặc tả được chính xác chi tiết cho từng hành động của chuỗi tương tác. Nên các nghiên cứu có đề xuất phương pháp sinh tự động các ca kiểm thử từ mô hình chỉ dừng lại ở bước sinh kịch bản kiểm thử hoặc sinh được các ca kiểm thử từ kịch bản kiểm thử ở mức tổng quát, chưa chỉ rõ được tập các dữ liệu đầu vào, đầu ra mong đợi và miền giá trị cụ thể cho từng ca kiểm thử.

Chúng tôi đề xuất lý thuyết cho ngôn ngữ mô hình hóa USL cho phép đặc tả ca sử dụng bằng mô hình. Ngôn ngữ cho phép đặc tả chính xác các chuỗi hành động tương tác và ngữ nghĩa đầy đủ cho các hành động, điều kiện gác trên các luồng chuyển. Điều này cho phép hướng dẫn sinh các kịch bản kiểm thử, ca kiểm thử một cách đầy đủ từ mô hình. Các khái niệm của ngôn ngữ USL được mở rộng từ biểu đồ hoạt động trong UML và thêm vào các Contract cho phép đặc tả chi tiết của từng hành động, từ đó cho phép hướng dẫn sinh các ca kiểm thử đầy đủ. Như hành động cung cấp dữ liệu đầu vào là gì? Điều kiện để hành động xảy ra? Đầu ra mong đợi của hành động là gì? Các ràng buộc được biểu diễn bằng một biểu thức Logic được xây dựng trên ngôn ngữ ràng buộc đối tượng OCL. Các mối quan hệ giữa các ca sử dụng cũng được biểu diễn. Với hướng tiếp cận này chúng tôi xây dựng một MetaModel cho ngôn ngữ đặc tả ca sử dụng.

## II. VÍ DỤ MINH HỌA

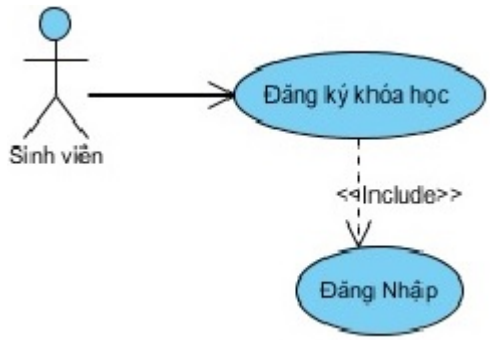
Phần này, qua ví dụ minh họa chúng tôi trình bày khái niệm cơ bản về mô hình ca sử dụng và kỹ thuật đang được sử dụng để tạo một cách thủ công các ca kiểm thử từ mô hình ca sử dụng, những thách thức đặt ra cho việc tự động hóa hoạt động này.

### *A. Biểu diễn ca sử dụng*

Trong [1] định nghĩa một ca sử dụng là một trường hợp sử dụng của hệ thống phần mềm cung cấp cho tác nhân bên ngoài của hệ thống thực hiện. Ca sử dụng mô tả các hành vi của hệ thống trong các điều kiện khác nhau, đáp ứng yêu cầu của tác nhân, tác nhân có thể là người, một hệ thống bên ngoài hoặc một thiết bị.

Các ca sử dụng của hệ thống thông thường được biểu diễn dưới dạng đồ họa như ví dụ minh họa trong hình 1 và các mô tả chi tiết cho từng ca sử dụng dưới dạng văn bản như ví dụ minh họa trong bảng 1.

Hình 1 là ví dụ về biểu đồ ca sử dụng, trong đó tác nhân chính là Sinh viên gồm có hai ca sử dụng "Đăng ký khóa học" và ca sử dụng "Đăng nhập", hai ca sử dụng này có mối quan hệ <<Include>> với nhau. Tác nhân chính là Sinh viên.




**Hình 1.**Ví dụ biểu đồ ca sử dụng trong UML

Trong bảng 1 là đặc tả ca sử dụng "Đăng nhập". Trong đó Pre-Condition là tiền điều kiện, là điều kiện cần đảm bảo trước khi thực hiện ca sử dụng. Pos-Condition là hậu điều kiện, là điều kiện đảm bảo sau khi thực hiện ca sử dụng. Trigger là sự kiện kích hoạt ca sử dụng. Basic Flow là luồng chính, khi ca sử dụng thực hiện tương ứng với trường hợp mọi điều kiện đều đúng đắn. Alternate Flow là các luồng rẽ nhánh khi thực hiện ca sử dụng, tương ứng với các trường hợp rẽ nhánh và ngoại lệ xảy ra khi thực hiện ca sử dụng. Một ca sử dụng gồm một luồng chính và nhiều luồng rẽ nhánh như minh họa trong bảng 1.

**Bảng 1.** Ví dụ về đặc tả ca sử đăng nhập bằng ngôn ngữ tự nhiên

| | |
|---|---|
| **1. Pre-Condition** | |
| | Sinh viên truy cập vào Website của trung tâm X |
| **2. Pos-Condition** | |
| | Nếu đăng nhập hệ thống thành công, hệ thống hiển thị các chức năng cho phép sinh viên thực hiện trên Website, nếu không thành công đưa ra các thông báo cho sinh viên biết. |
| **3. Trigger** | |
| | Use Case này được thực hiện khi Sinh viên click vào nút "Đăng nhập" trên Website của trung tâm X |
| **Basic flow** | |
| 1 | Hệ thống hiển thị giao diện đăng nhập |
| 2 | Sinh viên nhập User Name và PassWord |
| 3 | Sinh viên Click vào Button Đăng Nhập" |
| 4 | Hệ thống kiểm tra tính hợp lệ của UserName và PassWord |
| 5 | Hệ thống kiểm tra tài khoản của sinh viên có trong hệ thống không |
| 6 | Hệ thống hiển thị các chức năng mà sinh viên có quyền thực hiện trên Website |
| **Alternate flow** | |
| 1 | 1.1. UserName và PassWord không hợp lệ<br>Hệ thống hiển thị thông báo "UserName trong khoản từ 8 đến 16 ký tự, PassWord là 8 ký tự, yêu cầu nhập lại" |

| | | |
|---|---|---|
| 2 | Nếu tài khoản của sinh viên không có trong hệ thống, số lần đăng nhập chưa quá 3<br>2.1. Hệ thống đếm số lần đăng nhập<br>2.2. Hệ thống hiển thị thông báo "Tài khoản không tồn tại, đăng nhập lại"<br>2.3 Hệ thống kiểm tra số lần đăng nhập nếu nhỏ hơn 3 thì quay về luồng chính, ngược lại chuyển luồng phụ 3 | |
| 3 | 3.1. Nếu tài khoản của sinh viên không có trong hệ thống và số lần đăng nhập bằng 3<br>Hệ thống hiển thị thông báo: "Tài khoản không tồn tại, số lần đăng nhập tối đa là 3" vô hiệu hóa chức năng đăng nhập và kết thúc | |

### *B. Tạo ca kiểm thử từ đặc tả ca sử dụng*

Với phương pháp kiểm thử dựa trên đặc tả ca sử dụng như trong [3], kiểm thử viên tiến hành đọc đặc tả ca sử dụng từ đó xác định thủ công các ca kiểm thử theo các bước như sau:

- *Xác định các kịch bản kiểm thử:* Dựa theo các luồng điều khiển đã mô tả trong luồng chính và luồng rẽ nhánh trong đặc tả ca sử dụng, chúng ta sẽ xác định được các đường đi khác nhau khi thực hiện một ca sử dụng mỗi một đường đi là một kịch bản ca sử dụng. Tương ứng với mỗi kịch bản ca sử dụng là một kịch bản kiểm thử.
- *Xác định ca kiểm thử:* Mỗi kịch bản kiểm thử, kiểm thử viên sẽ xác định các bộ giá trị của dữ liệu đầu vào khác nhau thỏa mãn và dữ liệu đầu ra mong đợi. Các dữ liệu đầu vào và các bộ giá trị thỏa mãn khác nhau của đầu vào được xác định thông qua đọc đặc tả luồng để nhận biết các dữ liệu đầu vào và các giá trị theo phương pháp kết hợp các điều kiện trong các phép toán quan hệ and, or. Tương ứng với một kịch bản kiểm thử, kiểm thử viên có thể xác định được nhiều ca kiểm thử thỏa mãn một kịch bản kiểm thử.
- *Xác định các giá trị cụ thể cho từng ca kiểm thử:* khi thực thi kiểm thử, kiểm thử viên sẽ nhập các giá trị cụ thể ứng với các trường hợp làm cho dữ liệu đầu vào thỏa mãn, không thỏa mãn theo miền giá trị.

Bảng 2 là ví dụ về các kịch bản kiểm thử xác định được từ ca sử dụng "Đăng nhập" được <<Include>> trong ca sử dụng "Đăng ký khóa học" được đặc tả trong bảng 1. Bảng 3 là ví dụ về các ca kiểm thử xác định được từ các kịch bản kiểm thử trong bảng 2. Trong bảng 3, từ cột 3 đến cột 6 là dữ liệu đầu vào của ca kiểm thử, cột 7 là kết quả mong đợi của ca kiểm thử. Giá trị trong dữ liệu đầu vào V là giá trị đúng, I giá trị sai, N/A là không cần xác định.

**Bảng 2.** Các kịch bản kiểm thử ca sử dụng "Đăng nhập"

| Mã kịch bản | Tên kịch bản | Luồng bắt đầu | Rẽ nhánh |
|---|---|---|---|
| SC1 | Nhập UseName & Pass 1 lần hợp lệ và đăng ký khóa học thành công | Basic Flow | |
| SC2 | Nhập UserName và PassWord không hợp lệ | Basic Flow | A1 |
| SC3 | Tài khoản đăng nhập không có trong hệ thống, số lần đăng nhập nhỏ hơn 3 | Basic Flow | A2 |
| SC4 | Tài khoản đăng nhập không có trong hệ thống, số lần đăng nhập bằng 3 | Basic Flow | A3 |

**Bảng 3.** Các ca kiểm thử của ca sử dụng "Đăng nhập"

| Test Case ID | Kịch bản ID | User Name | Pass Word | Số lần đăng nhập | Tài khoản có trong hệ thống | Kết quả mong đợi |
|---|---|---|---|---|---|---|
| TC001 | SC1 | V | V | N/A | True | Hiển thị các chức năng sinh viên được thực hiện |
| TC002 | SC2 | V | I | N/A | N/A | Thông báo: "Tài khoản không hợp lệ UserName trong khoảng từ 8 đến 16 ký tự, PassWord là 8 ký tự, yêu cầu nhập lại" |
| TC003 | SC2 | I | V | N/A | N/A | Thông báo: "Tài khoản không hợp lệ UserName trong khoảng từ 8 đến 16 ký tự, PassWord là 8 ký tự, yêu cầu nhập lại" |
| TC004 | SC3 | V | V | <3 | Fall | Thông báo: "Tài khoản không tồn tại, đăng nhập lại" |
| TC005 | SC4 | V | V | >=3 | Fall | Thông báo: "Tài khoản không tồn tại, Số lần đăng nhập tối đa là 3", vô hiệu hóa chức năng đăng nhập |

### *C. Khả năng tự động hóa tạo ca kiểm thử*

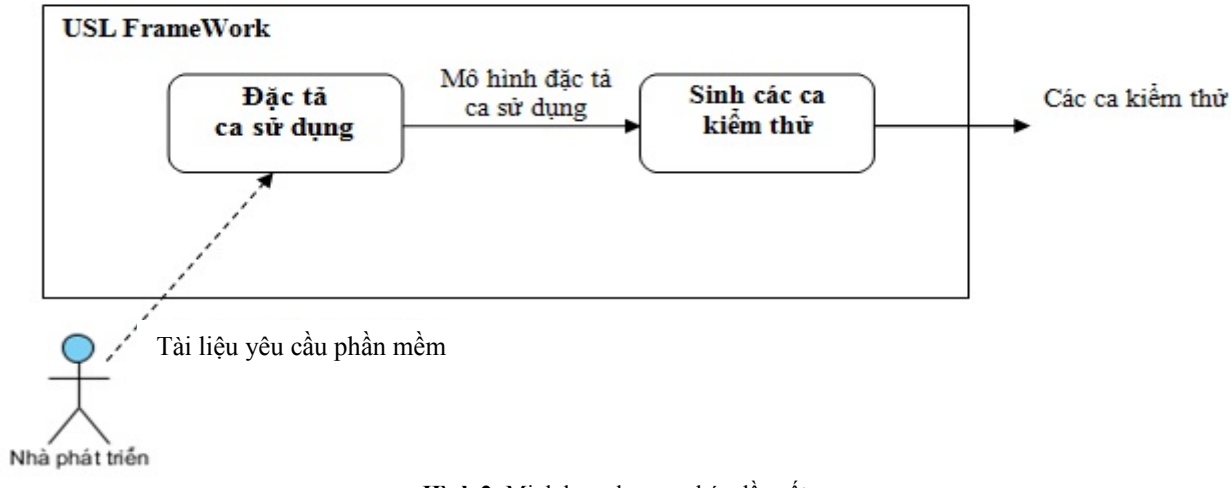
Hoạt động thiết kế ca kiểm thử từ đặc tả ca sử dụng được thực hiện thủ công. Khi yêu cầu phần mềm thay đổi, kiểm thử viên phải thực hiện thiết kế lại các ca kiểm thử. Để giảm chi phí cho hoạt động này, giải pháp đưa ra là cần tự động hóa trong thiết kế ca kiểm thử từ đặc tả ca sử dụng. Nhưng với đặc tả ca sử dụng bằng ngôn ngữ tự nhiên, việc tự động hóa gặp nhiều khó khăn do kỹ thuật xử lý ngôn ngữ tự nhiên là rất khó hơn nữa một vấn đề trong ngôn ngữ tự nhiên có thể được diễn đạt bằng nhiều cách khác nhau. Vì vậy để tự động hóa hoạt động này, ca sử dụng cần được đặc tả bằng một ngôn ngữ hình thức để máy có thể hiểu được đặc tả và tự động sinh các ca kiểm thử.

Ngôn ngữ đặc tả hình thức cho đặc tả ca sử dụng cần biểu diễn chính xác các đặc tả ca sử dụng, tương ứng với mỗi ca sử dụng thì chuỗi hành động nào xảy ra, trên các hành động đó dữ liệu đầu vào như thế nào để ca sử dụng thực hiện theo kịch bản, kết thúc đầu ra là gì. Hiện nay có rất nhiều nghiên cứu đề xuất phương pháp đặc tả ca sử dụng như chúng tôi đã trình bày trong phần mở đầu. Tuy nhiên các phương pháp đó cho phép đặc tả chưa đầy đủ. Chưa có phương pháp nào cho phép đặc tả các ràng buộc phải thỏa mãn cho các dữ liệu đầu vào trên từng hành động, điều kiện gác trên luồng, điều này làm cho các chỉ dẫn sinh tự động các ca kiểm thử của các nghiên cứu chưa hoàn chỉnh.

Nghiên cứu của chúng tôi đề xuất ngôn ngữ USL, ngôn ngữ có đủ các khái niệm cho phép đặc tả chi tiết ca sử dụng và các đặc tả này có thể là đầu vào cho mục đích hướng dẫn sinh tự động các ca kiểm thử. Chi tiết của phương pháp này chúng tôi sẽ trình bày trong phần tiếp theo.

## III. TỔNG QUAN VỀ PHƯƠNG PHÁP

Để giải quyết các vấn đề đặt ra trong phần trên chúng tôi đề xuất một ngôn ngữ cho phép đặc tả chi tiết ca sử dụng bằng mô hình và từ mô hình này có thể sử dụng để sinh tự động các ca kiểm thử như minh họa trong hình 2 dưới đây.




**Hình 2.** Minh họa phương pháp đề xuất

Đề xuất của chúng tôi là xây dựng một FrameWork cho ngôn ngữ đặc tả ca sử dụng, FrameWork này cho phép nhà phát triển đặc tả ca sử dụng theo tài liệu yêu cầu và đầu ra là các ca kiểm thử được sinh tự động từ đặc tả. Trong FrameWork gồm có hai phần.

- *Phần 1:* Ngôn ngữ đặc tả mô hình cho phép nhà phát triển đặc tả ca sử dụng bằng một mô hình. Mô hình này sẽ là đầu vào cung cấp cho bộ chuyển mô hình với mục đích sinh các ca kiểm thử tự động. Ngôn ngữ đặc tả này được mở rộng từ biểu đồ hoạt động trong UML và thêm vào khái niệm contract cho phép đặc tả chi tiết từng hành động và điều kiện gác của luồng. Ngôn ngữ được phát triển dựa trên kỹ thuật phát triển hướng mô hình với hướng tiếp cận là mô hình hóa miền chuyên biệt (Domain Specific Modelling).
- *Phần 2:* Bộ chuyển mô hình cho phép sinh tự động các ca kiểm thử từ mô hình đặc tả ca sử dụng bằng cách áp dụng các thuật toán sinh các ca kiểm thử trên chuyển mô hình. Bộ chuyển mô hình được xây dựng dựa trên kỹ thuật chuyển mô hình (Model Transformation), trong đó mô hình được chuyển sang tài liệu dựa trên kỹ thuật chuyển mô hình sang văn bản (Model to Text).

Trong bài báo này chúng tôi tập trung vào trình bày về đề xuất ngôn ngữ đặc tả và các hướng dẫn sinh các ca kiểm thử từ đặc tả. Chi tiết chúng tôi trình bày trong phần tiếp theo.

## IV. BIỂU DIỄN CHÍNH XÁC ĐẶC TẢ CA SỬ DỤNG BẰNG USL

Trong phần này chúng tôi trình bày cách thức mở rộng biểu đồ hoạt động của UML, đề xuất cấu trúc cho đặc tả chi tiết các hành động và điều kiện gác trên chuyển và phương pháp sinh ca kiểm thử tự động từ mô hình của ngôn ngữ USL mà chúng tôi đề xuất.

### *A. Mở rộng biểu đồ hoạt động để biểu diễn ca sử dụng*

Trong đặc tả ca sử dụng, chúng tôi chủ yếu tập chung vào đặc tả các luồng thực hiện của ca sử dụng (luồng chính và các luồng thay thế) và các luật nghiệp vụ ràng buộc trên các hành động trong chuỗi tương tác của luồng. Các luồng, các luật này là cơ sở để xác định các kịch bản kiểm thử và các ca kiểm thử. Để biểu diễn các chuỗi tương tác chúng tôi sử dụng biểu đồ hoạt động trong UML để biểu diễn, tuy nhiên để biểu đồ hoạt động phù hợp với mục đích đặc tả ca sử dụng, chúng tôi định nghĩa lại các siêu khái niệm (Meta Concept) cho mục đích đặc tả này. Các siêu khái niệm xác định được cho mục đích đặc tả ca sử dụng được xác định như sau:

- Hành động của tác nhân (Actor Action) là hành động của đối tượng bên ngoài hệ thống thực hiện các tương tác với hệ thống bên trong, hành động của tác nhân nhằm cung cấp các dữ liệu đầu vào hoặc yêu cầu hệ thống thực hiện các xử lý. Hành động của tác nhân được biểu diễn bằng hình chữ nhật.
- Hành động của hệ thống (System Action) là hành động của hệ thống nhằm xử lý các yêu cầu của hành động tác nhân. Khi hành động của hệ thống thực hiện có thể cung cấp dữ liệu lấy về từ hệ thống hoặc dẫn đến sự thay đổi trạng thái của hệ thống. Hành động của hệ thống được biểu diễn bằng hình chữ nhật bo cung ở góc.
- Nút quyết định (Decision Node) là các điểm kiểm tra điều kiện và tùy theo điều kiện mà luồng điều khiển thực hiện rẽ ở các nhánh khác nhau. Nút quyết định được biểu diễn bằng hình chữ nhật viền đậm và nét đứt.
- Luồng (Flow) là luồng chỉ dẫn chiều của hành động tiếp theo. Luồng được biểu diễn bằng đường thẳng có hướng mũi tên chỉ chiều hành động tiếp theo. Ngoài ra chúng tôi còn sử dụng luồng để biểu diễn mối quan hệ giữa các ca sử dụng. Luồng này được biểu diễn bằng đường thẳng nét đứt có hướng mũi tên chỉ chiều quan hệ và phía trên là nhãn của quan hệ gồm <<include>>, <<extend>>, <<use>>. Trên các luồng có thể có các điều kiện gác là điều kiện cho luồng xảy ra, điều kiện này sẽ được đặc tả chi tiết bằng khái niệm contract mà chúng tôi đề xuất trong phần sau.
- Nút bắt đầu (Node Start) biểu diễn điểm bắt đầu của ca sử dụng, được biểu diễn bằng hình tròn tô đen. Một ca sử dụng chỉ có duy nhất một nút bắt đầu.
- Nút kết thúc (Node End) biểu diễn điểm kết thúc của ca sử dụng, được biểu diễn bằng hai hình tròn lồng nhau. Một ca sử dụng có thể có nhiều nút kết thúc.

Để mô tả chi tiết cho từng hành động như: hành động cung cấp dữ liệu đầu vào nào, dữ liệu đầu vào đó do tác nhân cung cấp, hay hệ thống cung cấp; để hành động đó xảy ra thì các dữ liệu đầu vào phải thỏa mãn các ràng buộc như thế nào; sau khi thực hành động đó hệ thống có trả về kết quả mong đợi nào. Để mô tả khái niệm này chúng tôi đề xuất khái niệm cam kết (Contract) cho phép đặc tả chi tiết từng hành động. Ngoài ra khái niệm này có thể được sử dụng để mô tả ràng buộc trên luồng. Cấu trúc khái niệm cam kết chúng tôi sẽ mô tả chi tiết trong phần tiếp theo.

### *B. Cấu trúc mô tả chi tiết cho hành động và điều kiện gác trên luồng*

Để mô tả chi tiết cho hành động và điều kiện gác trên luồng, chúng tôi đề xuất một cấu trúc Contract như sau:

```
Contract <name>
InputA
   object: Type
InputS
   object: Type
Pre
[OCL_Condition]
Out
Description
```

Trong đó:

- <name>: Là tên của Contract.
- Đối tượng object khai báo trong InputA là các giá đầu vào do hành động tác nhân cung cấp.
- Đối tượng object khai báo trong InputS là các giá trị đầu vào lấy ra từ hệ thống.
- Biểu thức OCL_Condition khai báo trong Pre là biểu thức Logic mô tả các ràng buộc trên các tập dữ liệu đầu vào phải thỏa mãn để hành động có thể xảy ra. Biểu thức ràng buộc này được viết bằng ngôn ngữ ràng buộc đối tượng OCL.
- Mô tả Description khai báo trong Out mô tả kết quả mong đợi mà hệ thống trả về khi thực hiện hành động.

*Ví dụ 1:* Đặc tả chi tiết của hành động "nhập UserName và PassWord" của tác nhân :

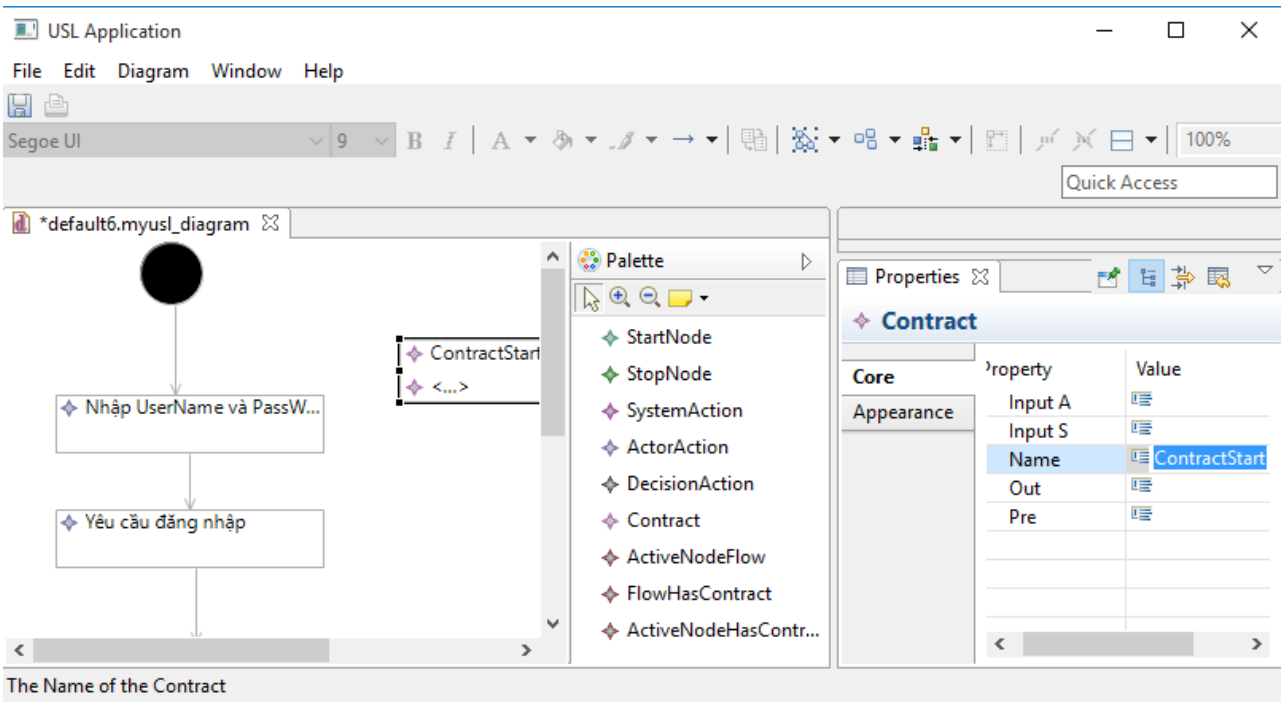
| | |
|---|---|
| Contract NhapUserpass<br>InputA<br>us, pass: String | Ngữ nghĩa Contract này cho biết hành động "nhập UserName và PassWord" cung cấp dữ liệu đầu vào là us, pass được khai báo trong InputA. |

*Ví dụ 2:* Đặc tả chi tiết của hành động kiểm tra "Hiển thị các chức năng sinh viên được thực hiện" của hệ thống:

| | |
|---|---|
| Contract KTTKhoan<br>Pre<br>[AcountExist=true]<br>Out<br>"Hiển thị các chức năng sinh viên được thực hiện" | Ngữ nghĩa Contract này cho biết để thực hiện hành động "Hiển thị các chức năng sinh viên được thực hiện" thì điều kiện AcountExist bằng true và đầu ra mong đợi của hệ thống sau khi thực hiện hành động là: "Hiển thị các chức năng sinh viên được thực hiện". |

### *C. Ngôn ngữ đặc tả USL*

Một ngôn ngữ đồ họa được phát triển bằng cách sử dụng một siêu mô hình (MetaModel) để định nghĩa cú pháp trừu tượng (Abstract Systax) của ngôn ngữ. Siêu mô hình là mô hình của một ngôn ngữ mô hình hóa, nó bao gồm các lớp nguyên thủy, các quan hệ để tạo nên ngôn ngữ mô hình hóa. Siêu mô hình định nghĩa cú pháp (Syntax) và ngữ nghĩa (Sematic) của mô hình, trong đó mỗi mô hình là một thể hiện của MetaModel [1]. Hình 3 dưới đây là FrameWork Plugin trên Eclipse mà chúng tôi đã xây dựng dựa trên siêu mô của ngôn ngữ USL đã đề xuất cho đặc tả ca sử dụng.




**Hình 3.** FrameWork soạn thảo của ngôn ngữ USL

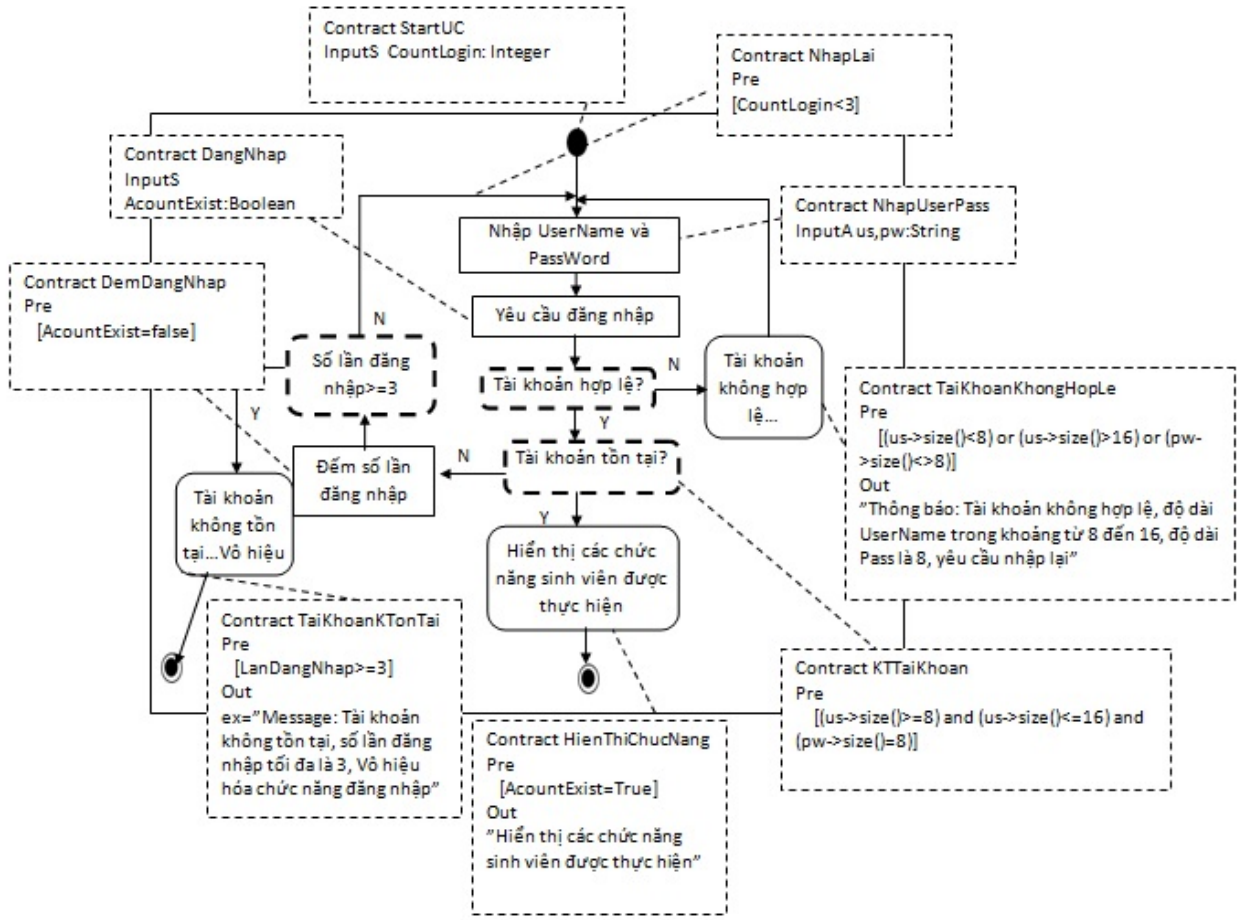
Các ký hiệu đồ họa (Notation) cho trình diễn cú pháp cụ thể của ngôn ngữ chúng tôi đề xuất được trình bày trong bảng 4 dưới đây.

**Bảng 4.** Các ký hiệu đồ họa của các khái niệm trong ngôn ngữ USL

| Ký hiệu | Khái niệm |
|---|---|
| ● | Bắt đầu ca sử dụng |
| ◉ | Kết thúc ca sử dụng |
| <Tên hành động> | Hành động của tác nhân |
| <Tên hành động> | Hành động của hệ thống |

| | |
|---|---|
| <Điều kiện kiểm tra> | Nút quyết định |
| Điều kiện gác | Luồng điều khiển |
| <<include>> | Quan hệ Include giữa các ca sử dụng |
| <<Extend>> | Quan hệ Extend giữa các ca sử dụng |
| <<Use>> | Quan hệ Use giữa các ca sử dụng |
| | Đường bao ca sử dụng |
| Contract <Name> | Contract |
| | Quan hệ giữa hành động hoặc luồng với Contract |

Hình 4 dưới đây là mô hình đặc tả ca sử dụng "Đăng nhập" được mô hình hóa bằng ngôn ngữ USL mà chúng tôi đã đề xuất. Mô hình này là một thể hiện của MetaModel của ngôn ngữ USL trong hình 3.




**Hình 4.** Mô hình đặc tả ca sử dụng Đăng nhập bằng ngôn ngữ mô hình hóa USL

## V. SINH CÁC CA KIỂM THỬ TỪ MÔ HÌNH ĐẶC TẢ CA SỬ DỤNG

Để minh họa cho phương pháp sinh các ca kiểm thử tự động trong phần này chúng tôi sẽ sử dụng ví dụ đặc tả ca sử dụng đăng nhập trong hình 4. Trong đó chúng tôi ký hiệu các hành động và Contract như sau:

- *Ký hiệu các hành động:* A1: Bắt đầu ca sử dụng, A2: Nhập User và PassWord, A3: yêu cầu đăng nhập, A4: Kiểm tra tài khoản hợp lệ, A5: Kiểm tra tài khoản tồn tại, A6: Hiển thị các chức năng sinh viên được thực hiện, A7: Thông báo: Tài khoản không hợp lệ, A8: Đếm số lần đăng nhập, A9: Kiểm tra số lần đăng nhập >=3, A10: Thông báo: Tài khoản không tồn tại, A11: kết thúc ca sử dụng.

- *Ký hiệu các contract:* C1: StartUC, C2: NhapUsserPass, C3: DangNhap, C4: KiemTraTaiKhoan, C5: HienThiChucNang, C6: TaiKhoanKhongHopLe, C7: DemDangNhap, C8: TaiKhoanKhongTonTai.

Ca sử dụng được đặc tả trong USL có thể được sử dụng cho mục đích sinh tự động các ca kiểm thử bằng một bộ chuyển mô hình. Các bước sinh các ca kiểm thử như sau:

- *Bước 1:* Sinh các kịch bản kiểm thử và xác định các dữ liệu đầu vào của các ca kiểm thử. Chúng tôi sẽ xây dựng một thuật toán cho phép duyệt đồ thị để lấy về các đường đi trong mô hình và các dữ liệu đầu vào cung cấp từ các hành động. Khi đó ứng với mỗi đường đi sẽ xác định được một chuỗi các hành động và một tập các Contract đặc tả chi tiết cho các hành động và điều kiện gác. Mỗi một đường đi tương ứng với một kịch bản kiểm thử. Bảng 5 dưới đây là các đường đi xác định được từ mô hình đặc tả ca sử dụng "Đăng nhập" trong hình 4 dựa vào thuật toán. Đồng thời xác định được các dữ liệu đầu vào trong phần khai báo InputA và InputS trong các Contract. Ví dụ với các Contract xác định được trong ví dụ này là ContLogin, us, pw, AcountExist.

**Bảng 5.** Các kịch bản xác định được từ mô hình

| Kịch bản kiểm thử | Chuỗi các hành động thực hiện | Tập các Contract |
|---|---|---|
| SC1 | A1, A2, A3, A4, A5, A6, A11 | C1, C2, C3, C4, C5 |
| SC2 | A1, A2, A3, A4, A7 | C1, C2, C3, C6 |
| SC3 | A1, A2, A3, A4, A5, A8, A9 | C1, C2, C3, C4, C7 |
| SC4 | A1, A2, A3, A4A5, A8, A9, A10, A11 | C1, C2, C3, C4, C7, C8 |

- *Bước 2:* Sinh các ca kiểm thử từ các kịch bản kiêm thử bằng cách duyệt qua tập các Contract trong kịch bản để: xác định được các trường hợp kiểm thử khác nhau căn cứ vào biểu thức logic ràng buộc trong khai báo pre, mỗi trường hợp kiểm thử tương ứng với một bộ giá trị đầu vào làm cho biểu thức Logic thỏa mãn, nếu dữ liệu đầu vào nào không có ràng buộc thì tương ứng với trường hợp không cần xác định giá trị; Xác định đầu ra mong đợi trong phần khai báo Out. Ví dụ với kịch bản kiểm thử SC2 trong tập các Contract có Contract C6 có mệnh đề Pre. Ràng buộc trên hai biến là us và ps bằng sẽ xác đinh phép toán or, các Contract khác không có ràng buộc như vậy với các đầu vào khác không cần xác định giá trị. Các ca kiểm thử xác định được trong kịch bản SC2 như trong bảng 6.

**Bảng 6.** Các ca kiểm thử của kịch bản SC2 của ca sử dụng Đăng nhập

| ca kiểm thử | Kịch bản kiểm thử | User Name | Pass Word | Count Login | Acount Exist | Kết quả mong đợi |
|---|---|---|---|---|---|---|
| TC2.1 | SC2 | us->size()<8, !(us->size()>16) | !(us->size()<>8) | N/A | N/A | Thông báo: "Tài khoản không hợp lệ UserName trong khoảng từ 8 đến 16 ký tự, PassWord là 8 ký tự, yêu cầu nhập lại". |
| TC2.2 | SC2 | !(us->size()<8), us->size()>16 | !(us->size()<>8) | N/A | N/A | Thông báo: "Tài khoản không hợp lệ UserName trong khoảng từ 8 đến 16 ký tự, PassWord là 8 ký tự, yêu cầu nhập lại". |
| TC2.3 | SC2 | !(us->size()<8), ! (us->size()>16) | us->size()<>8 | N/A | N/A | Thông báo: "Tài khoản không hợp lệ UserName trong khoảng từ 8 đến 16 ký tự, PassWord là 8 ký tự, yêu cầu nhập lại". |
| TC2.4 | SC2 | us->size()<8, us->size()>16 | us->size()<>8 | N/A | N/A | Thông báo: "Tài khoản không hợp lệ UserName trong khoảng từ 8 đến 16 ký tự, PassWord là 8 ký tự, yêu cầu nhập lại". |

## VI. CÁC NGHIÊN CỨU LIÊN QUAN

Có rất nhiều các nghiên cứu đề xuất phương pháp tiếp cận cho sinh ca kiểm thử từ đặc tả dựa trên mô hình của đặc tả ca sử dụng đã tồn tại. Như nghiên cứu [5, 6,13, 14 ,17] đã đề xuất phương pháp hình thức cho đặc tả ca sử dụng cho mục đích sinh test tự động. Các nghiên cứu đề xuất phương pháp cho phép đặc tả các chuỗi tương tác trong các luồng điều khiển thực hiện ca sử dụng.

Trong [5] Clémentine Nebut và các cộng sự đã đề xuất cấu trúc Contract cho đặc tả ca sử dụng bằng các mệnh đề logic trong ngôn ngữ ràng buộc OCL trong pre (tiền điều kiện), (post) hậu điều kiện các contract này mô tả tiền điều kiện và hậu điều kiện của ca sử dụng chứ không phải của từng hành động trong ca sử dụng. Nhóm tác giả mới dừng lại ở bước sinh kịch bản test từ contract và biểu đồ tuần tự trong UML.

Trong [6] Alexander và các cộng sự đã đề xuất sử dụng steorio type trong UML để đặc tả hành động của tác nhân và hành động của hệ thống bằng biểu đồ hoạt động, đề xuất của nhóm tác giả mới chỉ dừng lại ở bước đặc tả ca sử dụng.

Trong [13] Simona Vasilache đề xuất lý thuyết cho xây dựng FrameWork cho phép sinh ca kiểm thử dựa trên đặc tả bằng biểu đồ tuần tự và biểu đồ phụ thuộc giữa các kịch bản. Trong đó bước đầu tiên sinh kịch bản test từ biểu đồ tuần tự, bước thứ hai sẽ tiến hành mô hình hóa các kịch bản bằng biểu đồ phụ thuộc và sinh ca kiểm thử từ biểu đồ này, với phương pháp này nhóm tác giả chỉ dừng lại đưa ra các trường hợp Test khác nhau chứ chưa sinh ra được các ca kiểm thử cụ thể gồm các tập dữ liệu đầu vào và đầu ra mong đợi.

Trong [14] Tho T. Quan đề xuất phương pháp đặc tả ca sử dụng bằng cấu trúc File XML và xây dựng FrameWork trực quan hỗ trợ cho đặc tả ca sử dụng và từ đó sinh ra các kịch bản kiểm thử.

Trong [17] Tomasz Straszak và các cộng sự đề xuất phương pháp một ngôn ngữ mô hình hóa TSL cho phép đặc tả các kịch bản của ca kiểm thử bằng ngôn văn bản với cú pháp theo SVO-O (Subject Verb Object – indirect Object). Hướng tiếp cận cho phép sinh các kịch bản kiểm thử còn việc sinh các ca kiểm thử còn nhiều hạn chế và chỉ dừng lại ở các ca kiểm thử trừu tượng chưa xác định rõ được từng dữ liệu đầu vào cũng như tập giá trị đầu vào cho các ca kiểm thử.

Các nghiên cứu trên chủ yếu dừng ở bước đề xuất giải pháp đặc tả yêu cầu của ca sử dụng bằng các giải pháp khác nhau như sử dụng biểu đồ hoạt động trong UML, biểu đồ tuần tự, Contract, biểu đồ phụ thuộc và đề xuất phương pháp sinh kịch bản kiểm thử. Một số nghiên cứu đề xuất sinh ca kiểm thử, tuy nhiên các ca kiểm thử xác định được chỉ là các trường hợp Test khác nhau chứ chưa xác định được bộ dữ liệu và đầu ra mong đợi cụ thể cho từng ca kiểm thử. Giải pháp của chúng tôi cho phép sinh được các kịch bản kiểm thử cụ thể và bộ ca kiểm thử hoàn chỉnh với tập các dữ liệu đầu vào, đầu ra mong đợi. Ngôn ngữ đặc tả USL được phát triển dựa trên mở rộng biểu đồ hoạt động trong UML bổ sung thêm khái niệm Contract, trong đó ngôn ngữ thừa kế được từ biểu đồ hoạt động trong UML đã được kiểm chứng tính đúng đắn của biểu đồ, các khái niệm của ngôn ngữ là các khái niệm thuộc miền đặc tả ca sử dụng phù hợp và thân thiện với mục đích đặc tả.

## VII. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Chúng tôi đã đề xuất lý thuyết cho phương pháp sinh các ca kiểm thử từ đặc tả ca sử dụng với hướng tiếp cận mô hình hóa miền chuyên biệt cho đặc tả ca sử dụng. Trong đó chúng tôi đã đưa ra được cú pháp trừu tượng cũng như cú pháp cụ thể cho ngôn ngữ USL, bằng cách mở rộng biểu đồ hoạt động trong UML và thêm vào các Contract cho phép đặc tả chi tiết các hành động, điều kiện gác cho mục đích sinh ca kiểm thử từ mô hình. Phần sinh tự động ca kiểm thử cũng như phát triển FrameWork hỗ trợ cho ngôn ngữ chúng tôi sẽ trình bày trong các bài báo tiếp theo.

Trong các bài báo tiếp theo chúng tôi sẽ tập chung vào trình bày các thuật toán sinh kịch bản ca kiểm thử và các ca kiểm thử. Xây dựng bộ chuyển mô hình áp dụng các thuật toán trên cho phép sinh các ca kiểm thử tự động từ ngôn ngữ mà chúng tôi đã đề xuất. Đồng thời tiếp tục phát triển FrameWork cho phép mô hình hóa trực quan đồ họa cho ngôn ngữ bằng dự án EMF và GMF của Eclipse và xây dựng bộ sinh kịch bản kiểm thử và ca kiểm thử cho ngôn ngữ.

## VIII. TÀI LIỆU THAM KHẢO

[1] Marco and Cabot, Jordi and Wimmer, Manuel Brambilla, *Model-Driven Software Engineering in Practice*.: Morgan \& Claypool Publishers, 2012.

[2] Alistair Cockburn, *Writing Effective Use Cases*, 1st ed.: Addison-Wesley Longman Publishing Co., Inc., 2000.

[3] Jim Heumann. (2001) http://www.ibm.com/. [Online]. HYPERLINK "http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf". http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf.

[4] Francisca and Matteo, Alfredo and Pastor, Oscar Isabel and Losavio, "A Specification Pattern for Use Cases," *Inf. Manage.*, Nov 2004.

[5] Clémentine Nebut and Franck Fleurey and Yves Le Traon and Jean-marc Jézéquel, "Automatic Test Generation: A Use Case Driven Approach," *IEEE Transactions on Software Engineering*, 2006.

[6] Alexander and Six, Hans-Werner Lorenz, "Tailoring UML Activities to Use Case Modeling for Web Application Development," in *IBM Corp.*, Toronto, Ontario, Canada, 2006.

[7] Bertrand Meyer, "Applying "Design by Contract"," *Computer*, Oct 1992.

[8] Micha and Bojarski, Jacek and Nowakowski, Wiktor and Ambroziewicz, Albert and Straszak, Tomasz mia\llek, "Complementary Use Case Scenario Representations Based on Domain Vocabularies," in *Proceedings of the 10th International Conference on Model Driven Engineering Languages and Systems*, 2007.

[9] Jacek Bojarski, Wiktor Nowakowski, Tomasz Straszak Michal Smialek, "Scenario Construction Tool Based on Extended UML Metamodel," in *Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems*, 2005.

[10] Frantisek and Mencl, Vladimir Plasil, "Getting 'Whole Picture' Behavior In A Use Case Model," *J. Integr. Des. Process Sci.*, Dec 2003.

[11] Wolfgang Grieskamp and Wolfram Schulte, "Testable Use Cases in the Abstract State Machine Language," in *In Proceedings of APAQS'01, Dec 10 – 11, 2001, Hong Kong*, 2001.

[12] Sanna Sivonen, "Domain-specific modelling language and code generator for developing repository-based Eclipse plug-ins", 2008.

[13] Simona Vasilache, "De PAT: A Framework for Specification-Based Automatic Test Generation", in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2015 Vol I*, Hong Kong, 2015.

[14] Tho T.Quan, Nhuan S.Lai, Thuan D.le Vu Y.Nguyen, "A Framework for Automatic Construction of Test Scenarios from Use-cases", in *Ho Chi Minh City Software Testing Conference Jan2015*, HCM, 2015.

[15] Dang, D.H.: Triple Graph Grammars and OCL for Validating System Behavior. In Hartmut Ehrig, Reiko Heckel, G.R., Taentzer, G., eds.: Proc. 4rd Int. Conf. Graph Transformations (ICGT'2008), Springer, Berlin, LNCS (2008).

[16] Richard C. Gronback, "Eclipse Modeling Project A Domain-Specific Language", United States of America, 2009.

[17] Tomasz Straszak and Michal Smialek, "Automating Acceptance Testing with tool support", in *straszak:automating, 2014.*

# A METHOD TO GENERATE TEST CASES FORM USE CASE

**Chu Thi Minh Hue, Dang Duc Hanh, Nguyen Ngoc Binh**

**ABSTRACT** - *Use cases have achieved wide use for capturing and structuring software requirements. Nowadays, the use case model usually is presented by the use case diagrams in UML and use cases specification is documented in text format. Use case specifications are created in during the analysis phase to specify software system's requirements and can also be used for creating system level test cases. Using use cases to get system tests has several benefits including test design at early stages of software development life cycle that reduces over all development cost of the system. It is usually documented by natural language. So there is a large gap to bridge between use case specification and concrete test cases. In this paper, we propose a method to specify use case by a model and guide automatically generated test cases from this model. In which, we propose a language to modeling use case specification USL for modeling the use case specification and automatic generate test cases. USL are extended form the UML activity diagram and add Contract concept, which allows specify details a action and a guard condition in a flow. With this approach, we develop a metamodel for presenting the abstract syntax of USL.Our approach allows to transform directly from the use case specification to test case scenario, test cases.*