ISSN 2354-0575

NGÔN NGỮ ĐẶC TẢ CA SỬ DỤNG

Chu Thị Minh Huệ, Trần Đỗ Thu Hà, Hoàng Quốc Việt
Trường Đại học Sư phạm Kỹ thuật Hưng Yên

Ngày nhận: 28/1/2016
Ngày xét duyệt: 09/3/2016

Tóm tắt:

Mô hình ca sử dụng 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 ca sử dụng như trong ngôn ngữ mô hình hóa UML và tài liệu hóa các đặc tả của từng ca sử dụng ở dạng văn bản. Mô hình ca sử dụng được xây dựng trong pha đặc tả yêu cầu, mô hình nắm bắt toàn bộ các hành vi của hệ thống và các tương tác giữa hệ thống và các tác nhân bên ngoài. Đặc tả ca sử dụng là đầu vào cho hoạt động thiết kế các ca kiểm thử ở mức kiểm thử hệ thống và xây dựng các mô hình thiết kế như mô hình tương tác trong UML. Để giảm chi phí cho phát triển phần mềm, giải pháp tự động hóa là một trong các giải pháp tốt và đem lại lợi ích lớn trong phát triển phần mềm. Đặc tả ca sử dụng thường được tài liệu hóa bằng ngôn ngữ tự nhiên vì vậy giải pháp sinh tự động tài liệu thiết kế kiểm thử hoặc mô hình thiết kế từ đặc tả ca sử dụng vẫn còn nhiều thách thức đặt ra. Trong bài báo này, chúng tôi đề xuất một phương pháp cho đặc tả hình thức ca sử dụng bằng một mô hình. Mô hình này cho phép đặc tả chính xác ca sử dụng bằng một ngôn ngữ mô hình hóa USL. Ngôn ngữ USL được mở rộng từ biểu đồ hoạt động trong UML và thêm và các cam kết cho phép đặc tả chi tiết các hành động và các ràng buộc trên các hành động, các điều kiện trên luồng chuyển. Với cách tiếp cận này, chúng tôi xây dựng một ngôn ngữ mô hình hóa gồm có một siêu mô hình mô tả cú pháp trừu tượng và đề xuất các ký hiệu cho cú pháp cụ thể của ngôn ngữ. Từ đó đem lại khả năng chuyển trực tiếp từ mô hình đặc tả ca sử dụng sang các mô hình thiết kế hoặc sinh tự động các ca kiểm thử.

Từ khoá: *Đặc tả ca sử dụng.*

1. Giới thiệu

Theo định dạng chuẩn trong tài liệu đặc tả phần mềm (SRS) [6] một mô hình ca sử dụng (use case) là một mô hình hướng đối tượng, nó được sử dụng để xác định các hành vi của hệ thống. Mô hình Use case gồm các biểu đồ Use case và các đặc tả. Các đặc tả thường được mô tả bằng ngôn ngữ tự nhiên. Sử dụng ngôn ngữ tự nhiên để mô tả giúp khách hàng và người dùng có thể dễ dàng hiểu và tham gia tích cực vào hoạt động phân tích yêu cầu. Tuy nhiên nhược điểm của ngôn ngữ tự nhiên là khó tự động hóa. Để có thể tự động hóa đặc tả ca sử dụng cần được đặc tả bằng một phương pháp hình thức mà máy có thể hiểu được và chuyển tự động từ đặc tả hình thức sang các mô hình khác theo yêu cầu. Trong bài báo này chúng tôi đề xuất lý thuyết phương pháp cho đặc tả ca sử dụng. Phương pháp của chúng tôi hướng tới một ngôn ngữ mà máy có thể hiểu được cho đặc tả ca sử dụng.

Mô hình ca sử dụng gồm các tác nhân (actors), các ca sử dụng (use cases), và các liên kết (associations) và được mô tả trong biểu đồ ca sử dụng. Theo cockburn [4] Mỗi một use case, trình diễn các phần chức năng hoàn chỉnh của hệ thống phần mềm và được mô tả bằng một đặc tả ca sử dụng gồm: Luồng chính của ca sử dụng, các luồng thay thế, các bên liên quan, các điều kiện, và tham chiếu tới các quy tắc nghiệp vụ của ca sử dụng. Trong [4,7,11] một đường đi trong luồng thực hiện Use case là một kịch bản thực hiện Use case, một kịch bản là một chuỗi các hành động của tác nhân và phản ứng của hệ thống để thực hiện ca sử dụng. Để đặc tả hình thức cho các luồng thực hiện của use case, có một số giải pháp như trong [1] sử dụng ngôn ngữ Z object để đặc tả, trong [8] sử dụng biểu đồ hoạt động và stereotypes để đặc tả riêng rẽ cho các hành động của tác nhân, hành động của hệ thống. Trong [9,10] đề xuất một cấu trúc 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 như hành động của tác nhân, hành động của hệ thống cung cấp các dữ liệu đầu vào như thế nào? các ràng buộc nghiệp vụ trên các dữ liệu đầu vào đó.

Chúng tôi đề xuất lý thuyết cho ngôn ngữ mô hình hóa USL (Use Case Specification Language) 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 chuyển tự động đầy đủ từ đặc tả ca sử dụng sang các mô hình thiết kế chi tiết hoặc sinh

các kịch bản kiểm thử, dữ liệu kiểm thử một cách đầy đủ từ mô hình. Ngôn ngữ USL chúng tôi đề xuất bằng cách kết hợp các ý tưởng của các nghiên cứu [8,9,10], 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.

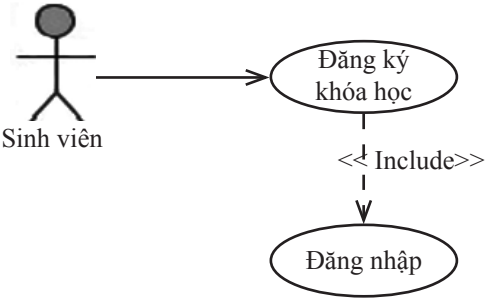
2. Kiến thức cơ sở

2.1. Ca sử dụng (Use Case)

Trong [3] 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 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.

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 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. *Biểu đồ ca sử dụng*

Trong Bảng 1 là đặc tả ca sử dụng "Đăng nhập". Trong đó tiền điều kiện (Pre-Condition), là điều kiện cần đảm bảo trước khi thực hiện ca sử dụng. Hậu điều kiện (Pos-Condition), là điều kiện đảm bảo sau khi thực hiện ca sử dụng. Sự kiện kích hoạt (Trigger) là sự kiện làm cho ca sử dụng thực hiện. Luồng chính (Basic Flow) là luồng thực hiện đúng đắn khi thực hiện ca sử dụng. Các luồng rẽ nhánh (Alternate Flow) là các luồng thực hiện trong các trường hợp ngoại lệ khi thực hiện ca sử dụng. Các luật nghiệp vụ (Business Rules), là các luật ràng buộc nghiệp vụ trên các hành động. Một ca sử dụng gồm một luồng chính và nhiều luồng rẽ nhánh như minh họa trong Bảng 1.

Bảng 1. *Đặc tả ca sử dụng Đăng nhập*

1. Tiền điều kiện	
	Sinh viên truy cập vào Website của trung tâm X
2. Hậu điều kiện	
	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. Kích hoạt	
	Use Case này được thực hiện khi Sinh viên click vào nút "Đăng nhập" trên Website của trung tâm X
Luồng chính	
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

Luồng phụ	
1	1.1. UserName và PassWord không hợp lệ
	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
	2.1. Hệ thống đếm số lần đăng nhập 2.2. Hệ thống hiển thị thông báo "Tài khoản không tồn tại, đăng nhập lại" 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
	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
Các luật nghiệp vụ	
1. Độ dài UserName từ 8 đến 16 ký tự 2. Độ dài Password là 8 ký tự	

2.2. Siêu mô hình (MetaModel)

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 siêu mô hình. Như trong Hình 3 là siêu mô hình của ngôn ngữ mô hình hóa USL mà chúng tôi đã đề xuất.

3. Xác định ca kiểm thử từ ca sử dụng

3.1. Xác định các kịch bản kiểm thử

Việc xác định các kịch bản thực hiện ca sử dụng cho phép xác định các đường đi thực hiện ca sử dụng khác nhau khi thực hiện ca sử dụng. Trên cơ sở đó trong phát triển phần mềm mỗi một kịch bản thực hiện ca sử dụng sẽ tương ứng với một kịch bản kiểm thử, hoặc trong thiết kế chi tiết mỗi một kịch bản thực hiện ca sử dụng tương ứng với một mô hình tương tác như biểu đồ hoạt động trong UML.

Để xác các kịch bản thực hiện ca sử dụng nhà phát triển sẽ tiến hành đọc đặc tả ca sử dụng trong tài liệu dưới dạng ngôn ngữ tự nhiên như ví dụ trên Bảng 1. Dựa trên các nhánh thực hiện chương trình khác nhau sẽ tiến hành xác định các đường đi khác nhau tương ứng với mỗi nhánh đó. Hoạt động này được thực hiện hoàn toàn thủ công. Theo đặc tả trong Bảng 1, chúng ta có thể xác định được các kịch bản thực hiện ca sử dụng "Đăng nhập" được <<include>> trong ca sử dụng "Đăng ký khóa học" như trong Bảng 2 dưới đây.

Bảng 2. *Các kịch bản thực hiện ca sử dụng "Đăng nhập"*

Tên kịch bản	Luồng
Nhập UseName & Pass 1 lần hợp lệ và đăng ký khóa học thành công	Basic Flow
Nhập UserName và PassWord không hợp lệ	Basic Flow, A1
Tài khoản đăng nhập không có trong hệ thống, số lần đăng nhập nhỏ hơn 3	Basic Flow, A2
Tài khoản đăng nhập không có trong hệ thống, số lần đăng nhập bằng 3	Basic Flow, A3

3.2. Khả năng tự động hóa từ mô hình ca sử dụng

Các hoạt động xây dựng mô hình thiết kế, xây dựng ca kiểm thử từ đặc tả ca sử dụng được thực hiện thủ công. Do đặc tả ca sử dụng trong UML không có mô hình hỗ trợ mà đặc tả bằng ngôn ngữ tự nhiên. Điều này dẫn đến khi yêu cầu phần mềm thay đổi, nhà phát triển phải thực hiện thiết kế lại mô hình thiết kế và ca kiểm thử. Để giảm chi phí cho hoạt động này, giải pháp đưa ra là cần tự động hóa trong xây dựng các mô hình thiết kế và xây dựng 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 mô hình thiết kế hoặc các ca kiểm thử.

Ngôn ngữ đặc tả hình thức cho đặc tả ca sử dụng cần biểu diễn chính xác các luồng thực hiện trong ca sử dụng, các ràng buộc trên luồng và các dữ liệu đầu vào, đầu ra trên các hành động. 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 các hành động và các ràng buộc trên các dữ liệu đầu vào được cung cấp bởi hành động, điều này làm cho các chỉ dẫn sinh các ca kiểm thử tự động 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 tự động hóa.

4. Biểu diễn 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.

4.1. Mở rộng biểu đồ hoạt động để biểu diễn ca sử dụng

Trong đặc tả ca sử dụng, chúng tôi chủ yếu tập trung 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 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) được biểu diễn bằng hình chữ nhật.

• Hành động của hệ thống (System Action) được biểu diễn bằng hình chữ nhật bo cung ở góc.

• Nút quyết định (Decision Node) 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.

4.2. 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: Đặc tả chi tiết của hành động kiểm tra "Hiển thị các chức năng sinh viên được thực hiện" của hệ thống:

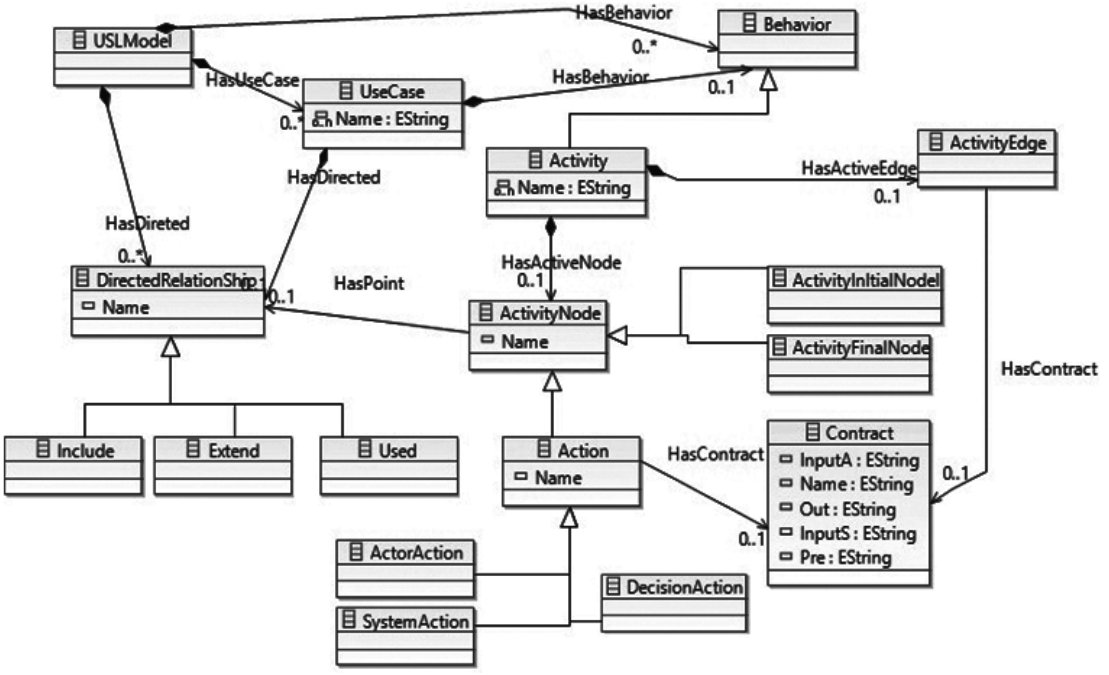
```
Contract KTTKhoan
Pre
[AcountExist=true]
Out
"Hiển thị các chức năng sinh viên được thực hiện"
```

Ngữ nghĩa Contract này cho biết để thực hiện Hành động "Hiển thị các chức năng sinh viên

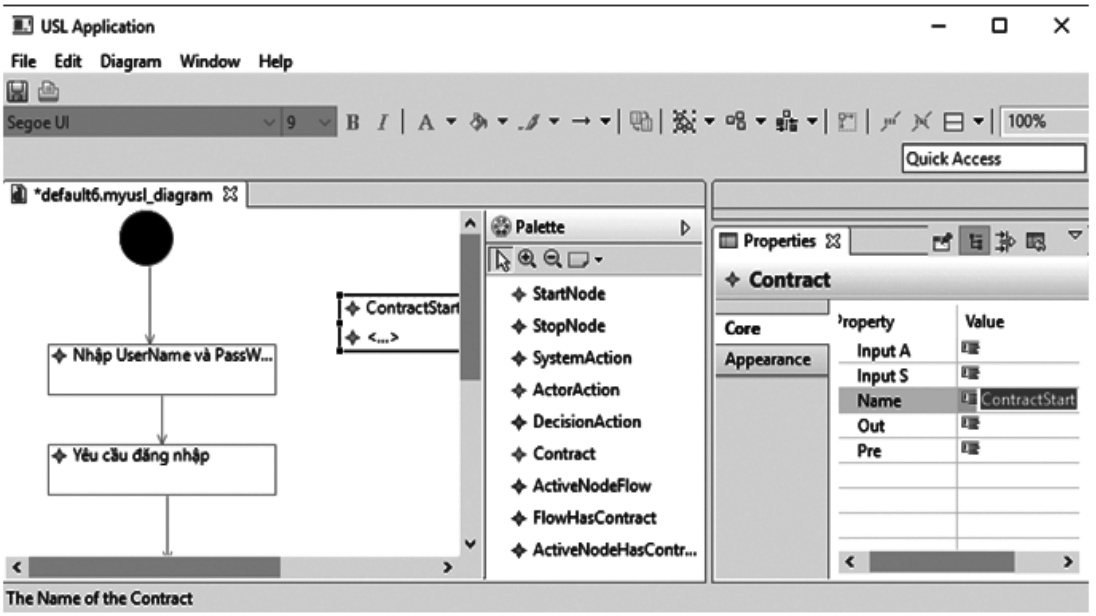
được thực hiện" thì điều kiện AcountExist bằng true, và đầu ra mong đợi của hệ thống sau khi thực hiện hành động là: "Hiển thị các chức năng sinh viên được thực hiện".

5. Ngôn ngữ đặc tả USL

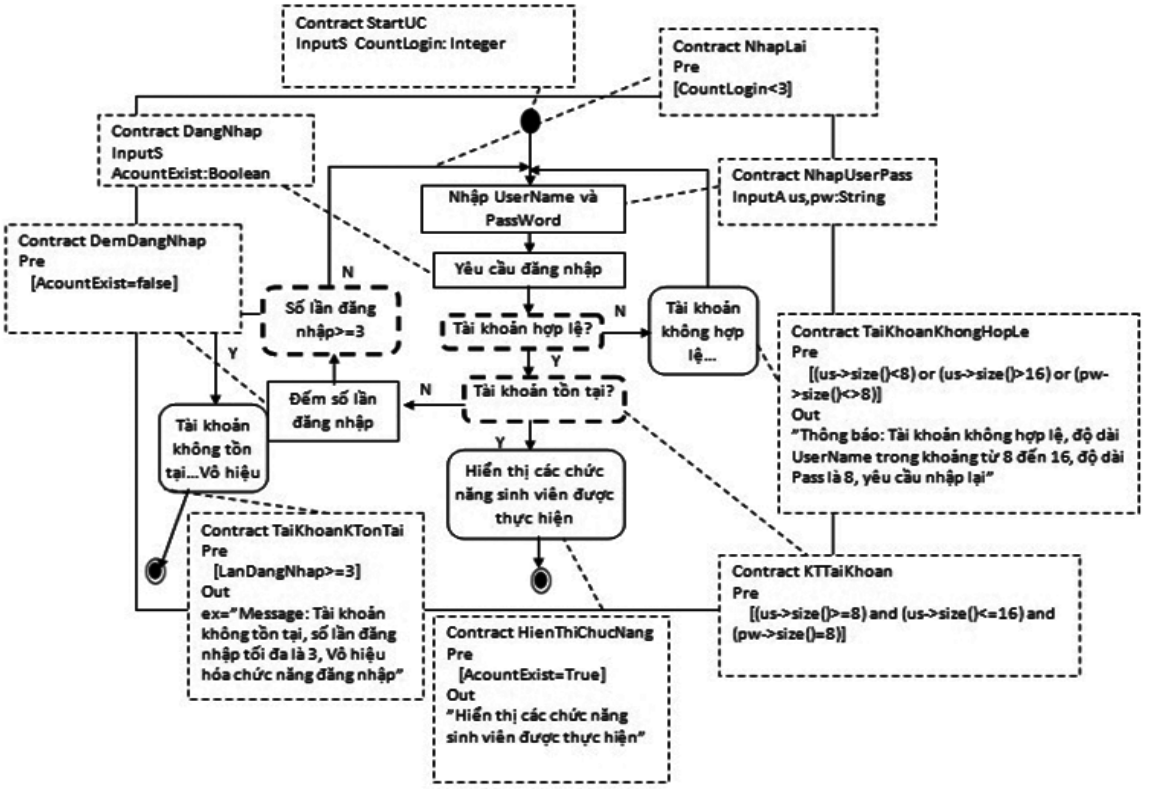
Trong Eclipse có các dự án cho phép xây dựng ngôn ngữ mô hình hóa đồ họa. Như dự án EMF cho phép xây dựng cú pháp trừu tượng bằng cách định nghĩa một siêu mô hình. Dự án GMF cho phép phát triển cú pháp cụ thể của ngôn ngữ. Một số dự án cho phép thực hiện các phép chuyển mô hình sang văn bản như Xpand, Xtext, Aceleo…. Chúng tôi đã phát triển được công cụ cho mô hình hóa của ngôn ngữ USL bằng cách phát triển cú pháp trừu tượng và cú pháp cụ thể cho ngôn ngữ USL, công cụ xây dựng như trong Hình 4. Quy trình và phương pháp xây dựng ngôn ngữ mô hình hóa đồ họa trên Eclipse không thuộc phạm vi trình bày của bài báo này. Hình 5 là mô hình đặc tả chi tiết ca sử dụng "Đăng nhập" như ví dụ trong mục 3 được tạo bằng công cụ mô hình hóa USL.



Hình 3. *Siêu mô hình của ngôn ngữ USL*



Hình 4. *FrameWork cho ngôn ngữ USL*



Hình 5. *Ví dụ mô hình đặc tả ca sử dụng bằng USL*

Kết luận

Chúng tôi đã đề xuất lý thuyết cho phương pháp đặc tả ca sử dụng với hướng tiếp cận mô hình hóa miền chuyên biệt. 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 Constract cho phép đặc tả chi tiết các hành động, điều kiện gác cho mục đích tự động hóa phát triển phần mềm từ đặc tả ca sử dụng. Và đã xây dựng FrameWork cho ngôn ngữ. Trong các bài báo tiếp theo chúng tôi sẽ tập chung vào trình bày các giải pháp cho phát triển tự động từ đặc tả ca sử dụng bằng ngôn ngữ USL đã xây dựng.

Tài liệu tham khảo

[1]. *Formalization of UML Use Case Diagram*—A Z Notation Based Approach, 2006.

[2]. Jesus M. Almendros-Jimenez and Luis Iribarne, *Describing Use cases with Activity Chart*, In Uffe Kock Wiil, editor, Metainformatics. Springer, LNCS 3511, 2004.

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

[4]. Alistair Cockburn, *Writing Effective Use cases*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 2000.

[5]. F. G. Dias, E. A. Schmitz, M. L. M. Campos, A. L. Correa, and A. J. Alencar, *Elaboration of Use case Specifications: An Approach Based on Use Case Fragments*, In Proceedings of the 2008 ACM Symposium on Applied Computing, SAC '08, pages 614–618, New York, NY, USA, 2008. ACM.

[6]. IEEE, *IEEE Recommended Practice for Software Requirements Specifications*, ANSI/IEEE Standard 830-1998, 1998.

[7]. Ivar Jacobson and Magnus Christerson, *A Growing Consensus on Use Cases*, JOOP, 1995.

[8]. Alexander Lorenz and Hans-Werner Six, *Tailoring UML Activities to Use case Modeling for Web Application Development*, In Hakan Erdogmus, Eleni Stroulia, and Darlene A. Stewart, editors, CASCON, pages 333–338. IBM, 2006.

[9]. Bertrand Meyer, *Applying "Design by Contract"*, Computer, 25(10):40–51, October 1992.

[10]. Clémentine Nebut, Franck Fleurey, Yves Le Traon, and Jean marc Jézéquel, *Automatic Test Generation: A Use case Driven Approach*, IEEE Transactions on Software Engineering, 32:140–155, 2006.
[11]. James Rumbaugh, Ivar Jacobson, and Grady Booch, *Unified Modeling Language Reference Manual*, The (2Nd Edition). Pearson Higher Education, 2004.
[12]. Nhuan D. Lai Vu Y.Nguyen, Tho T. Quan and Thuan D. Le, *A Framework for Automatic Construction of Test Scenarios from Use-cases*, In Ho Chi Minh City Software Testing Conference January 2015, 2015.

USE CASE SPECIFICATION LANGUAGE

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 and construction of the design model such as interactive models in UML. 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 tranform directly from the use case specification to test case scenario, test cases or design models.

Keywords: *Use case specification.*