Phương pháp sinh các ca kiểm thử tự động từ các mô hình thiết kế UML và ngôn ngữ ràng buộc đối tượng OCL

An Approach for Automated Test Case Generation from UML Models and OCL

Vũ Thị Đào, Tô Văn Khánh và Nguyễn Việt Hà

Abstract: *In software development, testing is the crucial and integral process to produce a reliable and high quality system. Automated test case generation plays a significant role in practice and a lot of researches on it has been investigated in recent years. The paper proposes an automated test case generation approach based on UML sequence diagrams, class diagrams and Object Constraint Language (OCL). Comparing with other test case generation, the approaches* [7,8,9] *have achieved message paths coverage, pre- and postcondition coverage while our approach also gain extra boundary coverage and association-end multiplicity coverage. In addition, the UML sequence diagrams can contain other nested sequence diagrams and apply to interactive operators such as alternative, option, break, sequence, negative, strict, and ignore. Our approach is aimed at high coverage of testing and reducing the number of test cases for generation.*

Keyword: *Test case, pre- and post-condition, Unified Modelling Language, UML sequence diagram, class diagram, Object Contraint Language, Finite state machine, predicate.*

I. MỞ ĐẦU

Trong phát triển các dự án phần mềm, kiểm thử phần mềm là giai đoạn quan trọng và thực sự cần thiết để tạo ra một hệ thống phần mềm có độ tin cậy cao, có chất lượng tốt [1,2]. Công việc thiết kế các ca kiểm thử là bước khó và thách thức nhất trong giai đoạn kiểm thử, đặc biệt đối với các hệ thống lớn vốn đã phức tạp để kiểm thử mà còn đòi hỏi một số lượng lớn các ca kiểm thử được tạo ra. Kiểm thử viên tốn rất nhiều thời gian và công sức để thiết kế các ca kiểm thử có độ bao phủ tốt và có thể tìm được nhiều lỗi của hệ thống. Vì vậy, quá trình sinh các ca kiểm thử tự động trở nên thực sự cần thiết, nhất là đối với những phần mềm lớn và phức tạp. Quá trình này có thể làm giảm giá thành phát triển phần mềm, cũng như tiết kiệm thời gian, nâng cao chất lượng phần mềm, tăng độ tin cậy, và độ bao phủ các yêu cầu phần mềm [3].

Trong cách tiếp cận kiểm thử dựa trên mô hình có ba hướng: kiểm thử từ máy hữu hạn trạng thái, kiểm thử từ mô hình các đặc tả (như Z,B,Spec#...) và kiểm thử từ các mô hình UML [4].

- Kiểm thử dựa trên máy hữu hạn trạng thái là một loại kiểm thử dựa trên mô hình trong đó mỗi nút của máy trạng thái tương ứng với một trạng thái cụ thể của hệ thống và các cạnh tương ứng là các hành động, vì vậy quá trình sinh các ca kiểm thử dựa vào việc duyệt tuần tự trong máy hữu hạn trạng thái. Hạn chế của mô hình là chọn trạng thái trong hệ thống như thế nào cho thích hợp và rất dễ bùng nổ không gian trang thái. Để khắc phục trường hợp bùng nổ không gian trạng thái, có thể mở rộng máy hữu hạn trạng thái bằng cách biểu diễn trạng thái dưới dạng các *trạng thái tượng trưng* (symbolic state) và điều kiện chuyển đổi giữa các

trạng thái [4]. Hướng tiếp cận này thường thích hợp hơn với các dự án vừa và nhỏ.

- Trong hướng tiếp cận thứ hai, giai đoạn đầu tiên sẽ chọn ngôn ngữ đặc tả và các tiêu chuẩn bao phủ chẳng hạn như bao phủ điều kiện (condition coverage), sau đó các ca kiểm thử được sinh ra dựa trên các độ bao phủ này. Một trong những công cụ thành công theo hướng tiếp cận này là Spec Explorer của Microsoft, công cụ này được tích hợp với kiến trúc Microsoft.net và sử dụng phiên bản mở rộng của C#, gọi là Spec# [4]. Hạn chế theo hướng này là kiểm thử viên phải biết và thành thạo các ngôn ngữ đặc tả được sử dụng.
- Hướng tiếp cận cuối cùng là kiểm thử dựa trên các mô hình UML. UML là một ngôn ngữ mô hình hóa chuẩn để thiết kế phần mềm hướng đối tượng [5]. Mô hình trạng thái UML có thể mở rộng được từ mô hình hữu hạn trạng thái với các trường hợp máy trạng thái lồng nhau hoặc máy trạng thái song song [4]. Hơn nữa, UML có thể phản ánh các khía cạnh khác nhau của hệ thống bằng các loại biểu đồ khác nhau, và có thể sử dụng cùng với ngôn ngữ ràng buộc đối tượng OCL [6]. OCL là ngôn ngữ chuẩn, được chấp nhận rộng rãi để viết các ràng buộc trong các mô hình UML; ví dụ: có thể viết ràng buộc cho các thuộc tính trong biểu đồ lớp, *các bất biến* (invariants) của các trạng thái, *các bảo vệ* (guards) của sự chuyển đổi các trạng thái, ràng buộc trong biểu đồ tuần tự, tiền và hậu điều kiện của các phương thức. OCL giúp khắc phục những thiếu sót và hạn chế của các biểu đồ, nó giúp biểu diễn các đặc tả, các ràng buộc mà nhiều khi biểu đồ không thể biểu diễn hết được. Do vậy, chúng tôi đã chọn theo hướng tiếp cận này.

Hiện nay, hướng tiếp cận dựa vào các mô hình UML và ngôn ngữ ràng buộc đối tượng OCL đang được quan tâm và nghiên cứu rộng rãi. Theo [7, 8, 9] từ biểu đồ tuần tự chuyển đổi sang mô hình trung gian là cây kịch bản [7], đồ thị xoắn [9] và đồ thị tham số biến [8]; sau đó áp dụng thuật toán tìm kiếm theo chiều sâu hoặc chiều rộng [7, 9] hoặc giải pháp ràng buộc và thực thi tượng trưng [8] để sinh các ca kiểm thử đạt được độ bao phủ luồng *thông điệp* (message) và điều kiện. Một hướng khác theo [10] và [11] xây dựng mô hình trung gian là hệ thống chuyển đổi nhãn trạng thái [10] hoặc đồ thị điều khiển cấu trúc[11] và độ bao phủ là tất cả các luồng thông điệp.

Từ những phương pháp hiện tại, bài báo đề xuất phương pháp sinh các ca kiểm thử ngoài việc đạt được độ bao phủ luồng thông điệp và điều kiện mà còn tạo ra dữ liệu kiểm thử và làm tăng độ bao phủ trong các ca kiểm thử. Chúng tôi áp dụng phương pháp biến thay thế đối với các *vị từ* (predicates) để sinh ra dữ liệu kiểm thử và kết hợp với biểu đồ lớp và OCL để làm tăng độ bao phủ của chúng. Phương pháp này có thể áp dụng cho các biểu đồ tuần tự lồng nhau và cho các toán tử tương tác như: thay thế, lựa chọn, ngắt, tuần tự, phủ định, chặt chẽ và bỏ qua.

Trong bài báo này chúng tôi đưa ra phương pháp sinh các ca kiểm thử tự động dựa trên biểu đồ tuần tự UML, biểu đồ lớp và ngôn ngữ ràng buộc đối tượng OCL. Từ biểu đồ tuần tự UML, xây dựng đồ thị tuần tự, sau đó duyệt đồ thị dựa vào thuật toán tìm kiếm theo chiều sâu hoặc tìm kiếm theo chiều rộng để lựa chọn *các vị từ* (predicates) và sinh các kịch bản khác nhau. Các vị từ được chuyển đổi thành các hàm vị từ và áp dụng kỹ thuật hàm nhỏ nhất [17] để sinh dữ liệu kiểm thử cho từng kịch bản tương ứng. Chúng được kiểm tra với các ràng buộc tiền và hậu điều kiện của từng phương thức. Vì vậy các kịch bản kiểm thử đã thỏa mãn các độ bao phủ luồng thông điệp, bao phủ biên và bao phủ tiền và hậu điều kiện. Theo các kịch bản được sinh ra, chọn các lớp liên quan, từ đó xây dựng biểu đồ lớp có các mối quan hệ và các bản số quan hệ của các lớp. Sau đó, sử dụng công cụ USE (UML Specification Environment) để kiểm tra sự thỏa mãn của các bất biến viết bằng OCL trong các kịch bản đó. Do vậy, các ca kiểm thử được sinh ra sẽ thỏa mãn các độ bao phủ cao: độ bao phủ các luồng thông điệp, các ràng buộc tiền và hậu điều kiện, bao phủ biên và bao phủ bản số của mối quan hệ giữa các thực thể trong biểu đồ lớp.

Bài báo bao gồm các phần như sau: phần II sẽ đưa ra một số khái niệm cơ bản, định nghĩa về độ bao phủ và một số độ bao phủ liên quan. Phần III trình bày cách tiếp cận và phương pháp đưa ra để sinh các ca kiểm thử từ các mô hình thiết kế UML và OCL. Minh họa ví dụ trong việc áp dụng phương pháp đưa ra được trình bày trong phần IV, và cuối cùng là kết luận.

II. CƠ SỞ LÝ THUYẾT

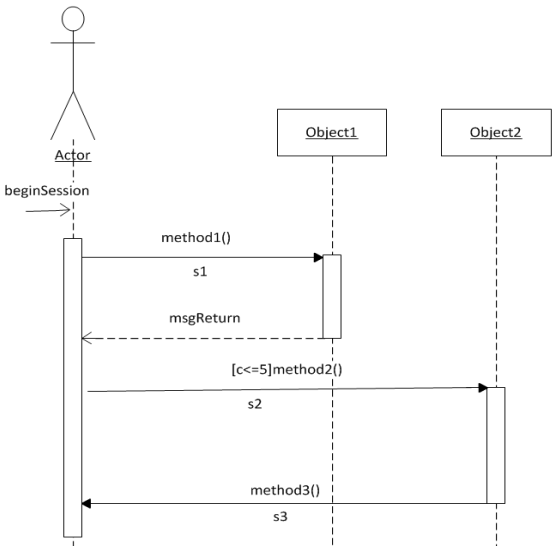
Sau đây là một số khái niệm liên quan đến quá trình sinh các ca kiểm thử từ các mô hình UML (biểu đồ tuần tự, biểu đồ lớp) và OCL, sau đó định nghĩa về độ bao phủ kiểm thử và đưa ra các độ bao phủ liên quan.

II.1. Một vài khái niệm cơ bản

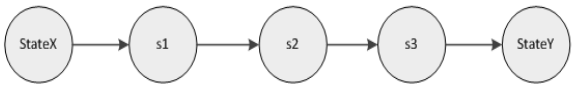
Ca kiểm thử: Một ca kiểm thử (test case) là một bộ [I,D,O] trong đó I là trạng thái ban đầu của hệ thống nơi mà dữ liệu kiểm thử được đưa vào, D là dữ liệu kiểm thử và O là kết quả mong đợi của hệ thống [2,12]. Với từng ca kiểm thử cụ thể, kết quả mong đợi sẽ được so sánh với kết quả thực tế của các kịch bản khi thực thi phần mềm.

*Đồ thị tuần tự***:** Một đồ thị tuần tự G= (V,E) trong đó V là tập hợp các đỉnh của đồ thị G và E là tập hợp các cạnh. Trong G, mỗi đỉnh biểu diễn một thông điệp tương tác giữa hai đối tượng và điều kiện (nếu có) trong biểu đồ tuần tự, và cạnh được nối giữa hai đỉnh tồn tại nếu có một thông điệp tương ứng xảy ra sau một thông điệp khác theo trình tự thời gian, ví dụ: từ biểu đồ tuần tự Hình 1 chuyển sang đồ thị tuần tự Hình 2. Đỉnh của thông điệp được khởi tạo một kịch bản được gọi là gốc của đồ thị, các đỉnh lá tương ứng là kết thúc trong tuần tự các thông điệp [7].

Trong biểu đồ tuần tự có thể được lồng các biểu đồ tuần tự khác, các biểu đồ tuần tự con chuyển đổi thành các đồ thị tuần tự con, các thông điệp hay các đỉnh trong đồ thị tuần tự con là các đỉnh con và đỉnh vào của đồ thị con gọi là đỉnh phức hay đỉnh cha trong đồ thị.



Hình 1. Biểu đồ tuần tự



Hình 2. Đồ thị tuần tự

Luồng (path): Một luồng P từ đỉnh s_i đến s_k là một chuỗi tuần tự các đỉnh s_i, s_{i+1},...,s_k trong đó các cặp đỉnh liền kề (s_{i+j}, s_{i+j+1}) tương ứng với một cạnh trong đồ thị G với $0 \leq j < k-i$

Miền của một luồng (path domain): Xem xét một luồng P trong đồ thị tuần tự, điều kiện trên luồng P là tập hợp tất cả các vị từ gắn với các đỉnh trong đồ thị P. Ví dụ, trong Hình 2, $c \leq 5$ trong đỉnh s2. Miền của một luồng P là tập hợp tất cả các giá trị dữ liệu đưa vào cho luồng P mà thỏa mãn tất cả các điều kiện vị từ trên P.

Biên: Miền trong mọi luồng đều được giới hạn bởi các đường biên. Một đường biên được xác định là một tập hợp các điểm biên mà các điểm này làm cho các điều kiện vị từ chuyển từ giá trị đúng sang giá trị sai và ngược lại [13]. Trong Hình 2, 5 là một điểm biên vì là điểm dữ liệu làm cho vị từ $c \leq 5$ chuyển từ giá trị đúng sang sai và ngược lại.

II.2. Các tiêu chuẩn bao phủ

Một tiêu chuẩn thích hợp trong kiểm thử có thể được sử dụng để xác định tính hiệu quả của phương pháp sinh các ca kiểm thử được đưa ra. Nó giúp việc

xem xét liệu tập các ca kiểm thử là đúng về chất lượng và đủ số lượng chưa cho giai đoạn kiểm thử phần mềm. Một tiêu chuẩn kiểm thử là một quy tắc hoặc một tập các quy tắc để tập các các ca kiểm thử sinh ra thỏa mãn yêu cầu phần mềm và bao phủ mô hình đưa ra [4]. Sau đây là một vài tiêu chuẩn bao phủ liên quan đạt được trong phương pháp đưa ra.

II.2.1. Tiêu chuẩn luồng thông điệp

Một luồng tuần tự các thông điệp biểu diễn hành vi được kiểm thử và mô tả tương tác giữa các đối tượng cần thiết tương ứng với chức năng hệ thống [14]. Tiêu chuẩn này đưa ra một tập các ca kiểm thử sao cho mỗi các ca kiểm thử tạo ra một luồng thông điệp có xảy ra trong biểu đồ tuần tự và được thực thi ít nhất một lần [15]. Một luồng tuần tự các thông điệp bắt đầu từ đỉnh vào và kết thúc ở đỉnh cuối cùng trong luồng của đồ thị đó.

II.2.2. Tiêu chuẩn kiểm thử biên được xác định theo [13]

Một tiêu chuẩn kiểm thử biên được thỏa mãn cho các vùng ranh giới không bằng nhau, nếu một vùng b được lựa chọn để kiểm thử bởi hai điểm (ON-OFF) của miền giá trị đầu vào, một điểm sẽ cho kết quả của vị từ q được chọn là đúng, còn điểm kia sẽ cho kết quả q là sai. Do vậy, các điểm cũng thỏa mãn luồng P xác định kết hợp với b và các điểm của miền giá trị đầu vào này phải gần các điểm khác và gần ranh giới nhất có thể.

Tiêu chuẩn kiểm thử biên là tiêu chuẩn để chắc chắn rằng vùng biên được kiểm thử tương xứng và phù hợp. Thay thế việc phát sinh một loạt các giá trị dữ liệu kiểm thử, chỉ kiểm thử ranh giới được xác định bởi các vị từ đơn giản. Các ca kiểm thử được sinh ra đạt được chuẩn bao phủ cao [16].

II.2.3. Tiêu chuẩn thỏa mãn bản số liên kết các thực thể trong biểu đồ lớp [4]

Sự kết hợp (association) là một mối quan hệ cấu trúc mô tả một tập các mối liên kết giữa các đối tượng và các bản số thể hiện số các đối tượng tham gia liên kết. Tiêu chuẩn này đòi hỏi từng thể hiện của các cặp bản số phải được tạo ra trong mỗi bộ kiểm thử. Ví dụ, nếu lớp A có quan hệ với một lớp khác B với bản số 1..1 ở phía lớp A và 0..4 ở phía lớp B thì các cặp kết hợp bản số giữa lớp A và lớp B được kiểm thử là: (1,0),(1,1),(1,4).

II.2.4. Tiêu chuẩn tiền và hậu điều kiện

Xem xét các các bất biến trong các lớp, nó được liên quan bởi việc thực thi các luồng thông điệp trong biểu đồ tuần tự. Ràng buộc nhấn mạnh vào các bất biến của các lớp, các tiền và hậu điều kiện của các phương thức và được viết bằng OCL. Để thỏa mãn tiêu chuẩn kiểm thử này thì phải đưa ra một tập hợp các ca kiểm thử từ biểu đồ tuần tự mà các ràng buộc sẽ được thực thi ít nhất một lần.

III. PHƯƠNG PHÁP SINH CÁC CA KIỂM THỬ DỰA TRÊN CÁC MÔ HÌNH THIẾT KẾ UML VÀ NGÔN NGỮ RÀNG BUỘC ĐỐI TƯỢNG OCL

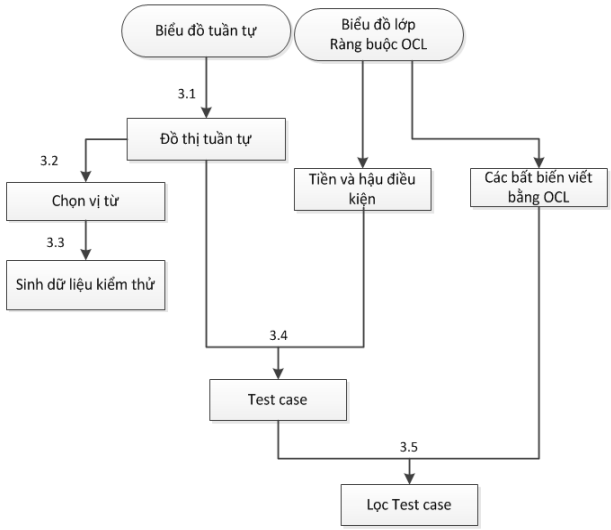
Kế thừa từ các phương pháp hiện tại sinh tự động các ca kiểm thử từ các mô hình tuần tự UML đạt được độ bao phủ luồng thông điệp, điều kiện trong các phương thức. Chúng tôi kết hợp với biểu đồ lớp để có thể đạt được độ bao phủ ràng buộc về bản số của các mối quan hệ trong biểu đồ này, và dùng phương pháp biến thay thế khi sinh tự động dữ liệu kiểm thử để đạt độ bao phủ biên của các ca kiểm thử. Thêm vào đó, kết hợp với OCL nhằm biểu diễn được các đặc tả mà mô hình không thể biểu diễn được, với mục đích kiểm tra các bất biến, các điều kiện ràng buộc đối tượng. Phương pháp này có thể áp dụng cho các biểu đồ tuần tự lồng nhau và cho các toán tử tương tác như: thay thế, lựa chọn, ngắt, tuần tự, phủ định, chặt chẽ và bỏ qua.

Trong phần này, chúng tôi đề xuất phương pháp để sinh các ca kiểm thử từ biểu đồ tuần tự, biểu đồ lớp và OCL. Hình 3 chỉ ra các bước cơ bản để sinh các ca kiểm thử. Phương pháp được chia làm 5 bước:

- Bước đầu tiên chuyển đổi biểu đồ tuần tự thành đồ thị tuần tự.
- Sau đó duyệt qua đồ thị để chọn các vị từ, mỗi vị từ đã chọn được chuyển thành các hàm vị từ.
- Sinh dữ liệu kiểm thử trong các ca kiểm thử từ các hàm vị từ.

- Từ đồ thị tuần tự kết hợp với tiền và hậu điều kiện sẽ được kiểm tra trong các kịch bản đó.
- Sử dụng công cụ USE để kiểm tra các bất biến viết bằng OCL theo các kịch bản đã sinh ra và các ràng buộc về bản số của biểu đồ lớp.

Phương thức sinh các ca kiểm thử này đạt được độ bao phủ cao: độ bao phủ các luồng thông điệp, các ràng buộc tiền và hậu điều kiện, bao phủ biên và bao phủ về bản số của các mối quan hệ trong các thực thể của biểu đồ lớp.



Hình 3. Các bước cơ bản sinh các ca kiểm thử

III.1. Chuyển đổi biểu đồ tuần tự thành đồ thị tuần tự

Sau khi có biểu đồ tuần tự, chúng ta biểu diễn phương thức chuyển đổi từ biểu đồ tuần tự sang đồ thị tuần tự. Để công thức hóa phương thức chuyển đổi, ta định nghĩa một kịch bản là một bộ bốn gồm có: {ScID; StartState; MessageSet; EndState} trong đó ScID là số xác định duy nhất trong từng kịch bản, StartState là điểm bắt đầu của kịch bản ScID, MessageSet chỉ ra một tập các sự kiện xảy ra trong một kịch bản, EndState là trạng thái mà hệ thống đến sau khi hoàn thành kịch bản hay là trạng thái kết thúc. Trong một đồ thị tuần tự có duy nhất một trạng thái bắt đầu và có một hoặc nhiều hơn trạng thái kết thúc phụ thuộc vào các kịch bản khác nhau.

Một sự kiện trong MessageSet được biểu diễn bởi bộ ba: {messageName; fromObject; toObject [/guard]} trong đó, messageName là tên của thông điệp với ký hiệu của nó, fromObject là đối tượng gửi thông điệp đi, toObject là đối tượng nhận thông điệp đó và có thể có điều kiện bảo vệ (guard) là điều kiện để sự kiện sẽ xảy ra. Vì vậy khi chuyển đổi, mỗi đỉnh trong đồ thị tuần tự tương ứng với một bộ ba ở trên, sự kiện này xảy ra sau sự kiện khác tạo ra hai đỉnh liên tiếp thành một cạnh trong đồ thị. Các đỉnh có thể có hoặc không có điều kiện bảo vệ hoặc vị từ giữa các đối tượng nhận và gửi thông qua thông điệp. Thông tin cần thiết được lưu ở các đỉnh tương ứng trong đồ thị tuần tự.

Khi biểu đồ tuần tự lồng nhau thì các đỉnh có thể là điểm vào của một đồ thị con, và đỉnh đó gọi là các đỉnh phức (đỉnh cha), các đỉnh trong đồ thị con gọi là các đỉnh con.

III.2. Chọn các vị từ và chuyển thành các hàm vị từ

Để chọn các vị từ, chúng ta thực hiện duyệt đồ thị tuần tự. Thuật toán duyệt đồ thị có thể dùng thuật toán tìm kiếm theo chiều rộng hoặc tìm kiếm theo chiều sâu để chắc chắn rằng mọi đỉnh đều được đi qua để chọn vị từ trong đồ thị. Trong cách tiếp cận này sử dụng thuật toán tìm kiếm theo chiều sâu, khi đó sẽ duyệt từ đỉnh bắt đầu và phát triển xa nhất có thể theo mỗi nhánh của đồ thị, do đó tạo ra các đường đi có thể từ điểm bắt đầu đến đỉnh kết thúc của đồ thị nên các kịch bản sinh ra cũng dễ dàng thỏa mãn độ bao phủ luồng thông điệp. Tất cả các đỉnh được xem như các đỉnh đơn trong quá trình duyệt. Nếu có gặp đỉnh phức, việc duyệt sẽ bắt đầu từ đỉnh khởi tạo hoặc đỉnh vào của đồ thị con. Trong suốt quá trình duyệt đồ thị, chúng ta sẽ tìm kiếm vị từ tại mỗi đỉnh, từ các vị từ được chọn để sinh dữ liệu kiểm thử tương ứng.

Trước khi sinh dữ liệu kiểm thử, chúng ta phải thực hiện chuyển đổi các vị từ thành các hàm vị từ. Xem xét tập dữ liệu khởi tạo I_0 , ở đây I_0 bao gồm tất cả các giá trị biến mà ảnh hưởng đến vị từ q trên luồng P trong đồ thị tuần tự. Như đề cập ở trên, chúng ta chia hai điểm ON và OFF cho ranh giới đưa ra thỏa mãn

tiêu chuẩn kiểm thử biên. Thực hiện chuyển đổi biểu thức quan hệ của các vị từ thành các hàm F (gọi là hàm vị từ).

Mục đích chuyển vị từ thành hàm F: để hàm phụ thuộc vào các biến (chính là các dữ liệu kiểm thử), phương pháp này thay đổi giá trị của các biến để tìm ra các bộ giá trị dữ liệu trên vùng biên và gần vùng biên nhất có thể (để thỏa mãn tiêu chuẩn kiểm thử biên).

Nếu vị từ q có dạng: (E_1 op E_2), trong đó E_1, E_2 là các biểu thức toán học (với các phép toán +, -, *, / và mod) và op là toán tử quan hệ

thì F = (E_1-E_2) hoặc (E_2-E_1) phụ thuộc vào liệu hàm F có giá trị dương khi thỏa mãn dữ liệu I_0

Các toán tử quan hệ và hàm F chỉ ra theo Bảng 1. (Hàm abs là hàm tính giá trị tuyệt đối)

Bảng 1. Toán tử quan hệ và hàm F.

Vị từ	*Hàm vị từ F*
$E_1 > E_2$	$E_1 - E_2$
$E_1 \geq E_2$	$E_1 - E_2$
$E_1 < E_2$	$E_2 - E_1$
$E_1 \leq E_2$	$E_2 - E_1$
$E_1 = E_2$	$E_1 - E_2$
$E_1 \neq E_2$	abs($E_1 - E_2$)

Việc thay đổi dữ liệu đầu vào I_0 để hàm F giảm dần và cuối cùng đạt được giá trị âm. Khi hàm F đạt giá trị âm, nó tương ứng thay thế kết quả của vị từ. Do đó, kết quả của việc chuyển đổi này, là tìm được các điểm dữ liệu làm cho kết quả của vị từ thay đổi, nó tương ứng với vấn đề tìm giá trị nhỏ nhất trong hàm F tương ứng. Giá trị nhỏ nhất có thể đạt được thông qua việc thay đổi các giá trị dữ liệu đầu vào.

Việc xác định giá trị của hàm F trên các luồng thông điệp tạo ra các bộ dữ liệu kiểm thử để các ca kiểm thử đạt được độ bao phủ biên.

III.3. Sinh dữ liệu kiểm thử

Các dữ liệu kiểm thử được sinh ra từ mỗi vị từ tương ứng với giá trị đúng hoặc sai của các vị từ thỏa mãn trên một luồng P trong đồ thị tuần tự. Thủ tục tìm kiếm cơ bản mà chúng ta sử dụng cho quá trình tìm kiếm giá trị nhỏ nhất của hàm vị từ là phương thức thay thế giá trị biến[13]. Phương thức này được dựa trên việc làm giá trị hàm F nhỏ nhất với lần lượt thay đổi giá trị đầu vào. Giá trị dữ liệu đầu vào ban đầu có thể chọn ngẫu nhiên, và ở mỗi bước các giá trị dữ liệu này được tăng hoặc giảm trong khi các giá trị dữ liệu khác được giữ nguyên.

Giả sử có hai giá trị dữ liệu I_{in} (trong miền biên giới hạn) và I_{out} (ngoài miền biên giới hạn) được sinh ra bằng cách sử dụng cách tìm kiếm như sau:

- Hai điểm dữ liệu này nằm ở hai phía khác nhau của đường biên giới hạn.
- Để tìm kiếm hai điểm dữ liệu này, một loạt các bước di chuyển được tạo ra trong cùng một chiều xác định bởi thủ tục tìm kiếm và giá trị của hàm F được tính toán lại sau mỗi lần di chuyển.
- Kích thước của từng bước di chuyển sẽ tăng gấp đôi sau khi di chuyển thành công, việc này nhằm cho phương thức tìm kiếm dữ liệu kiểm thử nhanh hơn. Mỗi lần di chuyển thành công thì giá trị của các hàm vị từ được giảm xuống.

Khi hàm F đạt được giá trị âm (hoặc bằng 0) thì giá trị dữ liệu đưa ra I_{in} và I_{out} được ghi lại. Các điểm này được lọc để tạo ra dữ liệu kiểm thử, nó tương ứng với việc làm giá trị của hàm vị từ nhỏ nhất. Việc lọc sẽ được thực hiện bởi việc giảm kích thước của mỗi bước và so sánh giá trị của hàm F với giá trị trước đó. Do vậy, khoảng cách giữa các điểm dữ liệu cũng nhỏ nhất bởi việc giảm kích thước này.

Với từng vị từ trong luồng kịch bản của đồ thị tuần tự, sẽ sinh ra dữ liệu kiểm thử tương ứng. Phương pháp này được lặp lại với các vị từ khác. Kịch bản và các dữ liệu tương ứng được lưu vào trong một tệp.

III.4. Thuật toán để sinh các ca kiểm thử

Trong mục này chúng tôi trình bày thuật toán để sinh các ca kiểm thử từ đồ thị tuần tự.

Input: Đồ thị tuần tự (SDG)

SDG = { S, $\sum$, q_0 , F }

Trong đó, S là tập tất cả các đỉnh của đồ thị tuần tự. $\sum$ là tập tất cả các cạnh, mỗi cạnh biểu diễn chuyển trạng thái từ đỉnh này sang đỉnh khác.

q_0 là đỉnh bắt đầu của đồ thị tuần tự.

F là tập tất cả các đỉnh kết thúc của đồ thị tuần tự.

Trong mỗi đỉnh $n_i \in S$ của đồ thị là một sự kiện e_i gồm một bộ : {messageName; fromObject; toObject [/guard]} trong đó messageName (m) là một thông điệp, có đối tượng gửi fromObject và đối tượng nhận toObject và điều kiện guard c (nếu có).

Với mỗi message m_i trong một đỉnh của SDG đều có tiền và hậu điều kiện ($preC_i$ và $postC_i$).

Output: Các ca kiểm thử T, độ bao phủ mỗi thông điệp, độ bao phủ luồng thông điệp và độ bao phủ tiền và hậu điều kiện.

Giải thích: Từ đồ thị tuần tự sẽ sinh ra các ca kiểm thử (kịch bản) tức là sinh ra tất cả các đường đi có thể từ đỉnh bắt đầu đến đỉnh kết thúc mà vẫn thỏa mãn tiền và hậu điều kiện của mỗi thông điệp và ràng buộc của nó.

Thuật toán chi tiết được mô tả bằng giả mã như sau:

Begin
// liet ke tat ca cac duong tu dinh bat dau den dinh ket thuc
//trong do thi
P= EnumerateAllPaths (SDG)
// voi moi duong di trong do thi
For each path $P_i \in P$ do
//bat dau tu dinh n_x va n_j la dinh hien tai
$n_j = n_x$
// $preC_i$ la tien dieu kien cua kich ban Sc_i chua trong
// dinh n_x
$preC_i$ = FindPreCond (n_x)
//Ca kiem thu tuong ung kich ban Sc_i khoi tao rong
$t_i \leftarrow \phi$
For each node n_j of path P_i do
//Su kien e_j tuong ung voi nut n_j va duoc dua ra
//thong diep m tu doi tuong gui a den doi tuong
//nhan b va dieu kien c
e_j = (m,a,b,c)
//neu khong co dieu kien tuong ung
If c no guard condition then
t = {preC,I(a_1,a_2,…,a_1),O(d_1,d_2,…,d_m), postC}
//preC:la tien dieu kien cua phuong thuc m
// I(a_1,a_2,…,a_1):tap cac gia tri dau vao cua m() tu
//doi tuong gui
//O(d_1,d_2,…,d_m):tap cac ket qua trong đối tượng
// nhận khi phuong thuc m() duoc thuc thi
//postC: la hau dieu kien cua phuong thuc m()
End If
If c has guard condition then
//tap hop cac dieu kien tren duong P_i
c(v) = (c_1,c_2,…,c_l)
t = {preC,I(a_1,a_2,…,a_l),O(d_1,d_2,…,d_m), c(v), postC}
EndIf
$t_i = t_i \cup t$
End For
$T \leftarrow T \cup t_i$
End For
Return (T)
End.

Tập T chính là các luồng thông điệp (kịch bản) thỏa mãn các tiền và hậu điều kiện, bao gồm tập I và O trong khái niệm ca kiểm thử. Thêm vào đó, theo III.2 và III.3 có thể sinh ra các dữ liệu kiểm thử trong các ca kiểm thử này, chính là tập D. Do đó, sinh ra được I, O và D (theo khái niệm ca kiểm thử trong II.1).

III.5. Kiểm tra các kịch bản thỏa mãn các bất biến viết bằng OCL

Theo từng luồng thông điệp, chọn các lớp và các thuộc tính liên quan, từ đó xây dựng biểu đồ lớp và các ràng buộc đối tượng viết bằng OCL. Biểu đồ lớp có mối quan hệ và bản số của mối quan hệ giữa các thực thể. Sử dụng công cụ USE để tạo ra các snapshot tương ứng với các kịch bản ở trên và vẫn thỏa mãn được độ bao phủ về bản số của mối quan hệ trong biểu đồ lớp. Từ đó có thể kiểm tra được theo các kịch bản tạo ra ở trên có thỏa mãn các bất biến viết bằng OCL , tức là các ca kiểm thử có thỏa mãn các bất biến đưa ra. Như vậy, các ca kiểm thử được sinh ra đạt được độ bao phủ cao: độ bao phủ các luồng thông điệp, thỏa mãn các ràng buộc tiền và hậu điều kiện, bao phủ biên

và bao phủ bản số liên kết các thực thể trong biểu đồ lớp.

Phương pháp sinh các ca kiểm thử trên có thể áp dụng cho các biểu đồ tuần tự mà các biểu đồ tuần tự này có các toán tử tương tác: thay thế, lựa chọn, ngắt, tuần tự, phủ định, chặt chẽ và bỏ qua. Trong mỗi toán tử này đều có các toán hạng, đó là các biểu thức logic và điều kiện cho các biểu thức logic ấy là đúng thì các biểu đồ tuần tự bên trong toán hạng đó được thực thi. Trong một biểu đồ tuần tự có thể có nhiều loại toán tử tương tác khác nhau, và các biểu đồ tuần tự lồng nhau biểu diễn nên một biểu đồ có cấu trúc phức tạp. Do đó, khi chuyển sang đồ thị tuần tự tương ứng cũng tạo nên đồ thị lồng nhau ở nhiều mức, các điều kiện trên sẽ gắn với các đỉnh (có thể là đỉnh con hoặc đỉnh cha) tương ứng.

Như vậy, so với kết quả đạt được từ các bài báo [7,8,9] các ca kiểm thử được sinh ra trong phương pháp đưa ra này đã thỏa mãn thêm độ bao phủ biên (khi tạo dữ liệu kiểm thử) và độ bao phủ về bản số trong biểu đồ lớp.

IV. VÍ DỤ ÁP DỤNG

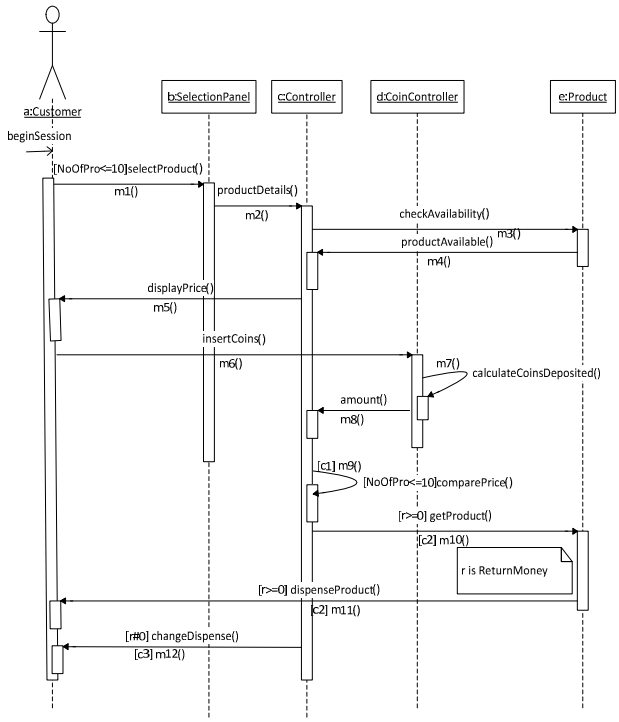
Trong phần này chúng tôi minh họa phương pháp và thuật toán đề xuất bằng ví dụ với máy bán hàng tự động.

Trong máy bán hàng tự động bán các loại nước uống và khách hàng sẽ cho các đồng xu vào máy. Hình 4(a) minh họa biểu đồ tuần tự UML của máy bán hàng tự động. Khi máy bán hàng tự động bật lên, khách hàng bắt đầu thực hiện giao dịch, thông tin các loại sản phẩm khác nhau có sẵn được hiển thị trên màn hình của máy, ở đây giả sử có hai loại nước uống là nước ngọt và cafe. Khi khách hàng chọn một loại trên màn hình thì các chi tiết về sản phẩm đó như giá thành, số lượng tối đa được mua trong một giao dịch được hiển thị. Khách hàng có thể chọn loại sản phẩm cũng như số lượng sản phẩm cần mua. Giả sử điều kiện số lượng mỗi sản phẩm trong một lần mua không quá 10 sản phẩm (NoOfPro ≤ 10), máy sẽ không phân phối nhiều hơn 10 sản phẩm mỗi loại trong một giao dịch. Khi lựa chọn xong về loại sản phẩm và số lượng yêu cầu thì khách hàng phải cho đồng xu vào máy.

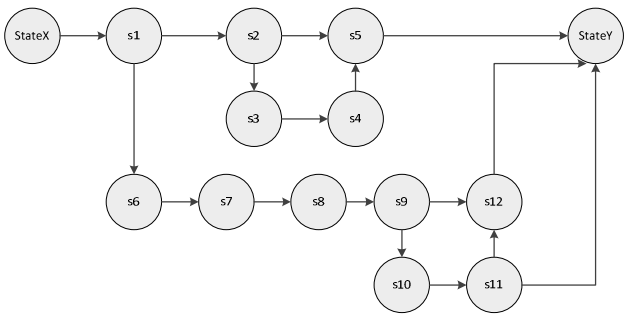
IV.1. Chuyển biểu đồ tuần tự của máy bán hàng tự động thành đồ thị tuần tự và sinh các kịch bản.

Theo phương thức đã được trình bày trong mục III.1. Việc chuyển đổi biểu đồ tuần tự sang đồ thị tuần tự của máy bán hàng tự động minh họa ở Hình 4.

Hình 4(a) đưa ra biểu đồ tuần tự của máy bán hàng tự động và đồ thị tuần tự được chuyển đổi được chỉ ra trong Hình 4(b). Hình 4(c) minh họa năm kịch bản được sinh ra từ đồ thị tuần tự.



Hình 4(a). Biểu đồ tuần tự của máy bán hàng tự động



Hình 4(b). Đồ thị tuần tự của máy bán hàng tự động

<Sc1	<Sc2	<Sc3	<Sc4	<Sc5
StateX	StateX	StateX	StateX	StateX
s1: (m1,a,b)|c1	s1: (m1,a,b)|c1	s1:(m1,a,b)| c1	s1: (m1,a,b)|c1	s1: (m1,a,b)|c1
s2: (m2,b,c)	s2: (m2,b,c)	s6: (m6,a,d)	s6: (m6,a,d)	s6: (m6,a,d)
s5: (m5,c,a)	s3: (m3,c,e)	s7: (m7,d,d)	s7: (m7,d,d)	s7: (m7,d,d)
StateY>	s4: (m4,e,c)	s8: (m8,d,c)	s8: (m8,d,c)	s8: (m8,d,c)
	s5: (m5,c,a)	s9: (m9,c,c)|c1	s9: (m9,c,c)|c1	s9: (m9,c,c)|c1
	StateY>	s12: (m12,c,a)|c3	s10: (m10,c,e)|c2	s10: (m10,c,e)|c2
		StateY>	s11: (m11,e,a)|c2	s11: (m11,e,a)|c2
			s12: (m12,c,a)|c3	StateY>
			StateY>	

Hình 4(c). Năm kịch bản sinh ra, được biểu diễn trong dạng bộ bốn

IV.2. Sinh dữ liệu kiểm thử từ các vị từ được chọn từ đồ thị

Từ đồ thị tuần tự được chuyển đổi, thực hiện thuật toán tìm kiếm theo chiều sâu trên đồ thị để chọn các vị từ gắn với các đỉnh của đồ thị. Giả sử từ đồ thị chúng ta chọn điều kiện c2 nghĩa là ReturnMoney ≥ 0, ở đây ReturnMoney là số tiền máy tự động sẽ trả lại cho khách hàng nếu số tiền khách hàng cho vào máy bán hàng(amount) nhiều hơn số tiền mua sản phẩm và

ReturnMoney = amount – totalMoney,trong đó totalMoney = NoOfPro1*Price1 + NoOfPro2*Price2 với NoOfPro1 là số lượng loại sản phẩm thứ 1 khách hàng yêu cầu; NoOfPro2 là số lượng loại sản phẩm thứ 2 khách hàng yêu cầu; Price1: giá của một sản phẩm trong loại thứ 1; Price2: giá của một sản phẩm trong loại thứ 2.

Xem xét đường biên kết hợp với vị từ (ReturnMoney ≥ 0). Giả sử I_0 là dữ liệu khởi tạo: I_0 = [(5,5), 200], trong đó (NoOfPro1=NoOfPro2=5 <10 và amount = 200).

Hàm F sẽ được biểu diễn như sau: F = ReturnMoney = amount-totalMoney

Vì vậy $F(I_0)$ =50. Thực hiện thay đổi kết quả logic của vị từ ReturnMoney ≥ 0. Đầu tiên, tăng giá trị của biến NoOfPro trong các bước khác nhau.

1. Trong bước đầu tiên, (NoOfPro1= NoOfPro2=5 và amount = 400), totalMoney = 5x15 + 5x15= 150 và ReturnMoney = 400-150=250. Từ biểu thức hàm F = amount -(NoOfPro1*Price1 + NoOfPro2*Price2) , thấy rằng khi F giảm thì NoOfPro tăng.
2. Bước tiếp theo, tăng kích thước của bước dịch chuyển lên gấp đôi tức là giá trị NoOfPro lên thêm 2 nữa, bây giờ NoOfPro là 7. Do vậy, [(NoOfPro1, NoOfPro2),amount] =[(7,7),400). Suy ra, F giảm xuống ReturnMoney= 400- 2x(7x15)= 190 > 0.
3. Tiếp tục tăng gấp đôi bước dịch chuyển và NoOfPro sẽ là 7 + 2x2 =11, do đó nó bị vi phạm ràng buộc NoOfPro$\leq$ 10 của một loại sản phẩm được chọn trong một giao dịch. Vì vậy, theo thuật toán chúng ta lại giảm kích thước di chuyển xuống là 2. Bây giờ thay thế NoOfPro = 11, nó sẽ là NoOfPro=9, và [(NoOfPro1, NoOfPro2),amount] = [(9,9),400]. Hàm F tính được sẽ là: ReturnMoney = 400- 2x(9x15) = 130, giá trị hàm F vẫn dương. Ta lại tiếp tục giảm kích thước bước dịch chuyển xuống một nửa thì giá trị giảm sẽ là 1. Vì vậy, NoOfPro = 9+1= 10 và F = 400- 2x(15x10) = 100. Nhưng hàm F $\neq 0$, hàm vẫn chưa đạt giá trị nhỏ nhất [17].
4. Tiếp tục xét biến tiếp theo của hàm F và lại tăng hoặc giảm các bước dịch chuyển để thực hiện giảm hàm F. Bây giờ giá trị NoOfPro giữ nguyên không đổi là 10 và số lượng tiền cho vào máy bán hàng sẽ thực hiện giảm đi theo các bước sau.

Với [(NoOfPro1, oOfPro2),amount)=[(10,10),399], F sẽ có giá trị là: F= (399- 2x(10x15)= 99. Sau đó, tiếp tục lặp lại để giảm số tiền amount như sau: [(10,10),397], [(10,10),393], [(10,10),385], [(10,10),369], [(10,10),337], [(10,10),273], bởi vì kích thước của mỗi bước dịch chuyển sẽ tăng gấp đôi trong mỗi lần lặp, tại dữ liệu [(10,10),273] thì hàm F = 273 – 2 x (10x15) = -27, tức là hàm F sẽ có giá trị âm tại điểm dữ liệu [(10,10),273].

5. Khi hàm F có giá trị âm với kích thước của bước dịch chuyển là 64; chúng ta sẽ lấy hai điểm dữ liệu khởi tạo là I_{in}: [(10,10),337] and I_{out}: [(10,10),273].
6. Sau đó, tiếp tục giảm kích thước dịch chuyển xuống một nửa từ 64 chuyển xuống 32. Vì vậy, amount = 337 – 32 = 305, với điểm dữ liệu [(10,10),305] thì F có giá trị là 5, giá trị này vẫn

dương. Do đó, chúng ta thay thế I_{in} là [(10,10),305**]** cho giá trị [(10,10),337]. Quá trình này lại được lặp lại, giảm kích thước dịch chuyển từ 32 xuống 16 sẽ là 305-16 = 289 và hàm F = 289- 2x(10x15) = -11. Hàm F đã chuyển sang giá trị âm, vì vậy lại thay thế I_{out} là [(10,10),289] cho giá trị dữ liệu [(10,10),273].

7. Giá trị dữ liệu I_{in} và I_{out} có được nhờ việc giảm giá trị cho mỗi bước dịch chuyển. Hàm F = 0, khi dữ liệu là[(10,10),300]; F = 300 – 2x(10x15) =0. Trong trường hợp này, ta có dữ liệu kiểm thử là I_{in}: [(10,10),305], I_{out}: [(10,10),289] và $B_{boundary}$: [(10,10),300].

8. Từ các kịch bản trong hình 4 (c) với dữ liệu kiểm thử sinh tương ứng với vị từ (ReturnMoney $\geq$0) là:

<Sc3>: (stateX, s1, s6, s7, s8, s9,s12,stateY, [(10,10),289])

<Sc4>: (stateX, s1, s6, s7, s8, s9,s10,s11,s12,stateY, [(10,10),305])

<Sc5>: (stateX, s1, s6, s7, s8, s9,s10,s11,stateY, [(10,10),300])

Chỉ có hai điểm dữ liệu kiểm thử I_{in} và I_{out} , chúng là các điểm dữ liệu nhỏ nhất để thỏa mãn vị từ (ReturnMoney $\geq$0). Tương tự dữ liệu kiểm thử sẽ được sinh ra thỏa mãn tất cả các vị từ gắn với các đỉnh khi duyệt đồ thị tuần tự. Do đó, các điểm dữ liệu tương ứng với các giá trị logic khác nhau thỏa mãn các điều kiện. Thủ tục sinh dữ liệu kiểm thử sẽ được lặp lại cho tất cả các vị từ tương ứng với các kịch bản trong đồ thị tuần tự. Khi sinh ra các kịch bản kiểm thử thì các điều kiện tiền và hậu điều kiện của các phương thức cũng được kiểm tra. Vì vậy, các ca kiểm thử sinh ra sẽ có độ bao phủ luồng thông điệp, độ bao phủ biên và độ bao phủ tiền và hậu điều kiện.

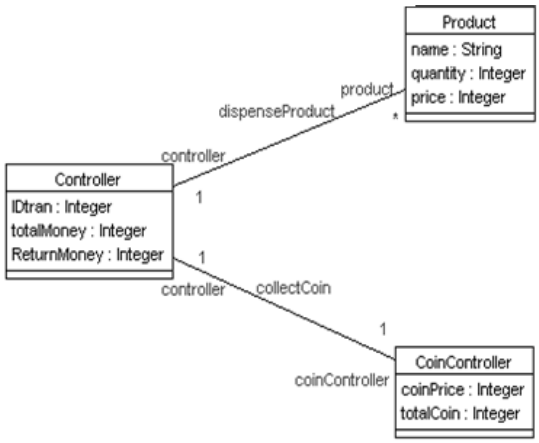
IV.3. Kiểm tra các kịch bản sinh ra thỏa mãn các bất biến viết bằng OCL

Theo từng luồng thông điệp chọn các lớp và các thuộc tính liên quan, từ đó xây dựng biểu đồ lớp và các ràng buộc viết bằng OCL. Giả sử chọn một kịch bản con trong máy bán hàng tự động:

(s6, s7, s8, s9,s10,s11,s12) với dữ liệu kiểm thử [(NoOfPro1, NoOfPro2),amount] là[(10,10),305])

Chọn các lớp liên quan đến kịch bản này gồm có: Controller, Product và CoinController.

Từ đó xây dựng biểu đồ lớp có bản số của mối quan hệ như Hình 5.



Hình 5. Biểu đồ lớp liên quan với các đối tượng trong luồng thông điệp được chọn

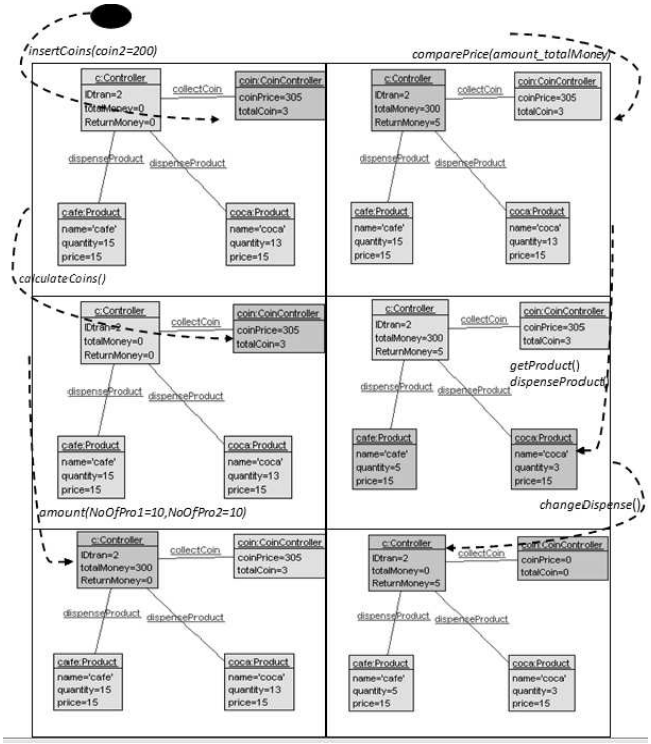
Sử dụng công cụ USE để tạo ra biểu đồ đối tượng theo kịch bản trên (xem Hình 6) và bản số của các mối quan hệ giữa các lớp cũng được thỏa mãn. Từ đó kiểm tra các bất biến, giả sử kiểm tra ràng buộc: số lượng sản phẩm bán ra trong một giao dịch không được vượt quá 20 và số lượng coinController chỉ có 1, được viết bằng OCL như sau:

Controller.allInstances.forAll(d| d.product $\rightarrow$size() <=20) và

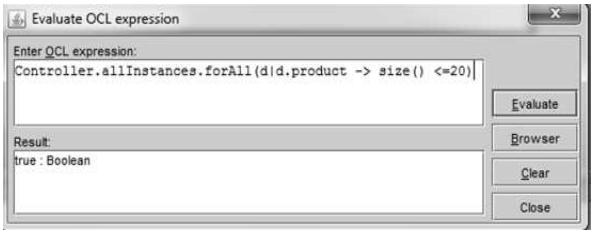
Controller.allInstances.forAll(d| d.coinController $\rightarrow$size() =1)

Minh họa kết quả kiểm tra các bất biến này ở Hình 7.

Vì vậy, kịch bản trên với dữ liệu được sinh ra đã thỏa mãn được các bất biến đưa ra. Từ phương pháp trên có thể kiểm tra tương tự với các ràng buộc khác, do đó các ca kiểm thử đạt được các độ bao phủ: độ bao phủ các luồng thông điệp, các ràng buộc tiền và hậu điều kiện, bao phủ biên và bao phủ bản số liên kết các thực thể trong biểu đồ lớp.



Hình 6. Biểu đồ đối tượng liên quan trong kịch bản được chọn



Hình 7. Kiểm tra các bất biến viết bằng OCL

V. KẾT LUẬN

Bài báo đã đề xuất một phương pháp sinh các ca kiểm thử từ các biểu đồ tuần tự UML, biểu đồ lớp và ngôn ngữ ràng buộc đối tượng OCL. Điểm mới của phương pháp này là sinh ra dữ liệu kiểm thử tự động trong các kịch bản kiểm thử. Kết quả cho thấy các ca kiểm thử sinh ra có độ bao phủ tốt hơn trong các bài báo [7,8,9], nó đạt độ bao phủ luồng thông điệp, các ràng buộc tiền và hậu điều kiện, độ bao phủ biên, độ bao phủ về bản số trong biểu đồ lớp và có thể áp dụng cho các biểu đồ tuần tự lồng nhau, có các toán tử tương tác: thay thế, lựa chọn, ngắt, tuần tự, phủ định, chặt chẽ và bỏ qua.

Kết quả mới trong phương thức đưa ra là áp dụng độ bao phủ biên để sinh dữ liệu kiểm thử từ các vị từ được duyệt từ đồ thị. Và cũng từ biểu đồ lớp, kiểm tra các luồng kịch bản sinh ra ở trên thỏa mãn các bất biến viết bằng OCL bằng cách sử dụng công cụ USE. Sự kiểm tra này sẽ lọc các ca kiểm thử để đạt được bao phủ về bản số trong mối liên kết giữa các thực thể trong biểu đồ lớp. Chúng đã tôi minh họa sự hiệu quả của phương pháp đề xuất bằng ví dụ về máy bán hàng tự động.

Trong tương lai, việc tiếp tục phát triển sinh các ca kiểm thử từ các loại biểu đồ UML khác hoặc kết hợp các loại biểu đồ khác để các ca kiểm thử có độ bao phủ tốt hơn, cũng có thể sử dụng các giải thuật tìm kiếm từ các ràng buộc để tạo ra dữ liệu kiểm thử tự động là các vấn đề mà chúng tôi sẽ giải quyết trong những nghiên cứu tiếp theo.

LỜI CẢM ƠN

Công trình này được tài trợ một phần từ đề tài KHCN cấp ĐHQGHN mã số QGTĐ 13.01.

TÀI LIỆU THAM KHẢO

[1] R.V. BINDER, *Testing object-oriented software: a survey*, Software Testing Verification and Reliability, 6(3/4), 1996, 125-252.

[2] R. MALL, *Fundamentals of Software Engineering*, Prentice Hall, 3th edition, 2009.

[3] S. GHOSE, R. FRANCE, C. BRAGANZA, N. KAWANE, A. ANDREWS, O. PILSKALNS, *"Test adequacy assessment for UML design model testing"*, Proceeding of the International Symposium on Software Reliability Engineering, Denver, CO, 2003 , pp. 332-343.

[4] M. UTTING and B. LEGEARD, *Practical Model-Based Testing: A Tools Approach*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[5] Object Management Group, *The Unified Modeling Language UML 1.5 Technical Report formal/03-03-01*, The Object Management Group (OMG) , 2003.

[6] http:/ / www.omg.org/ spec/ OCL/ 2.2/

[7] B. LI, Z. LI, L. QING, and Y. CHEN, *"Test Case Automate Generation from UML Sequence Diagram and OCL Expression"*, International Conference on Computational Intelligence and Security, 2007, pp. 1048-1052.

[8] T. DINH-TRONG, S. GHOSH, and R. FRANCE, *"A Systematic Approach to Generate Inputs to Test UML Design Models"*, Proceedings of the 17th International Symposium on Software Reliability Engineering, ISSRE '06, 2006, pp. 95-104.

[9] H. ZHOU, Z. HUANG, and Y. ZHU, *"Polymorphism Sequence Diagrams Test Data Automatic Generation Based on OCL"*, 9th International Conference for Young Computer Scientists, 2008, ICYCS 2008, November 2008, pp. 1235 -1240.

[10] E. CARTAXO, F. NETO, AND P. MACHADO, *"Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems"*, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, ISIC, 2007, pp. 1292-1297.

[11] A. NAYAK and D. SAMANTA, *"Automatic Test Data Synthesis using UML Sequence Diagrams"*, Journal of Object Technology, 2010, vol. 9, no. 2, pp. 115-144.

[12] J. OFFUTT, A. ABDURAZIK, *Generating tests from UML specifications*, Proceedings of 2nd International Conference UML, Lecture Notes in Computer Science,1999, pp. *416 – 429*.

[13] A. HAJNAL, I. FORGACS, *"An applicable test data generation algorithm for domain errors"*, ACM SIGSOFT Software Engineering Notes, Proc. ACM SIGSOFT Int. Symp. Software Testing and Analysis, 1998.

[14] BERNHARD K. AICHERNIG, *"Test Design through Abstraction: a Systematic Approach Based in the Refinement Calculus"*, Journal of Universal Computer Science, August 2001, 7(8):710 – 735.

[15] G. BOOCH, J. RUMBAUGH, I. JACOBSON, *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

[16] B. JENG, EJ. WEYUKER, *"A simplified domain-testing strategy"*, ACMTrans. Software. Eng. Methodology, 1994, 3(3), pp.*254 – 270*.

[17] RANJITA SWAIN, VIKAS PANTHI, KUMAR BEHERA, DURGA PRASAD MOHAPATRA, *"Automatic test case generation from UML state chart diagrams*", International Journal of Computer Applications, March 2012, Vol 42- No.7, pp.26-36.

Nhận bài ngày: 10/1/2014

SƠ LƯỢC VỀ TÁC GIẢ

VŨ THỊ ĐÀO

Sinh ngày 01 tháng 05 năm 1982.

Nhận bằng thạc sỹ CNTT, chuyên ngành Công nghệ Phần mềm tại trường ĐH Công nghệ – ĐH Quốc gia Hà Nội năm 2008, làm nghiên cứu sinh chuyên ngành Công nghệ phần mềm cũng tại trường này từ tháng 11/2010.

Hiện công tác tại Khoa CNTT, Học viện Kỹ thuật Mật mã

Lĩnh vực nghiên cứu: Kiểm thử phần mềm dựa trên mô hình, kiểm chứng phần mềm.

Điện thoại: 0982151982

E-mail: vuthidao@gmail.com

TÔ VĂN KHÁNH

Sinh ngày: 04 tháng 03 năm 1982.

Tốt nghiệp Đại học và Cao học tại trường ĐH Công nghệ, ĐH Quốc gia Hà Nội. Nhận bằng Tiến sỹ tại Viện KH&CN tiên tiến Nhật Bản - JAIST năm 2013.

Hiện là giảng viên Bộ môn Công nghệ Phần mềm, trường ĐH Công nghệ, ĐH Quốc gia Hà Nội.

Lĩnh vực nghiên cứu: Kiểm chứng hình thức, kiểm thử phần mềm; SAT, SMT và các ứng dụng trên SAT, SMT.

Điện thoại: 098 543 5604

E-mail: khanhtv@vnu.edu.vn

NGUYỄN VIỆT HÀ

Sinh năm 1974.

Nhận bằng Cử nhân, Thạc sỹ và bảo vệ luận án Tiến sỹ tại Đại học Takushoku, Nhật Bản lần lượt vào các năm 1997, 1999 và 2002.

Hiện là Phó Giáo sư, công tác tại Bộ môn Công nghệ Phần mềm, Khoa CNTT, trường ĐH Công nghệ, ĐHQG HN.

Các lĩnh vực nghiên cứu quan tâm hiện nay: Kiểm chứng, kiểm thử phần mềm, Kiến trúc phần mềm.

Điện thoại: (04)37546575

E-mail: hanv@vnu.edu.vn