Phương pháp kiểm thử các tính chất tương tranh trong biểu đồ tuần tự UML

Vũ Thị Đào, Phạm Ngọc Hùng, Nguyễn Việt Hà

***Tóm tắt*— Bài báo này giới thiệu phương pháp sinh các ca kiểm thử cho các ứng dụng tương tranh từ biểu đồ tuần tự UML. Phương pháp này không làm tăng quá nhanh số lượng các ca kiểm thử do việc chọn lựa điểm hoán đổi trong các luồng đồng thời nên có thể phát hiện các lỗi bao gồm: lỗi đồng bộ, deadlock,… trong các ứng dụng tương tranh. Bài báo đề xuất các tiêu chuẩn bao phủ tương tranh để có thể sinh các ca kiểm thử theo các tiêu chuẩn đó. Kết quả thực nghiệm cho thấy, các ca kiểm thử được sinh ra bằng phương pháp trên có khả năng phát hiện lỗi vượt trội so với thuật toán tìm kiếm theo chiều rộng (BFS) và tìm kiếm theo chiều sâu (DFS).**

***Abstract*— This paper presents an automated test case generation method from UML sequence diagrams for concurrent applications. This method avoids the number of test cases explosion by selecting switch point in concurrent threads. Therefore, it can uncover errors (such as synchronization, deadlocks and more) in concurrent applications. Moreover, the method also proposes concurrent coverage criteria in order to test cases generation according to the criteria. By the experimental results, the test cases are generated by our algorithm are superior as compared to breadth-first search and depth-first search algorithms.**

Từ khóa— ca kiểm thử; biểu đồ tuần tự UML; hệ thống đồng thờ; kịch bản kiểm thử.

I. MỞ ĐẦU

Trong những năm gần đây, các ứng dụng tương tranh đã được nghiên cứu nhiều, cả về lý thuyết và thực tiễn, cùng với sự thực thi trong xử lý đa luồng, đa tiến trình. Ứng dụng này cho phép cải thiện khả năng tính toán hiệu quả và tận dụng các nguồn tài nguyên tốt hơn. Tuy nhiên, các chương trình tương tranh có thể không xác định và cho các kết quả đầu ra khác nhau khi chạy với cùng một đầu vào trong các lần chạy khác nhau. Các chương trình ứng dụng này có các mức độ không xác định mà các chương trình tuần tự không có. Do đó, kiểm thử trong các ứng dụng tương tranh mà sử dụng các kỹ thuật truyền thống là một công việc gặp nhiều khó khăn và thách thức.

Một trong những cách tiếp cận để kiểm thử ứng dụng tương tranh là kiểm thử dựa trên các mô hình UML [1]. Các đặc tả UML trong phát triển phần mềm không những chỉ ra cách phát triển, mô hình hoá các yêu cầu được đưa ra từ giai đoạn phân tích đến thiết kế, mà còn cả trong tiến trình kiểm thử và tìm lỗi ở mức thiết kế. Mô hình thiết kế UML bao gồm các biểu đồ, trong đó mỗi biểu đồ miêu tả các khía cạnh khác nhau của hệ thống. Hành vi thực thi đồng thời trong các ứng dụng tương tranh được biểu diễn trong biểu đồ tuần tự sử dụng toán tử song song (parallel fragment) và các thông điệp bất đồng bộ.

Có rất nhiều hướng tiếp cận đã được đưa ra để giải quyết vấn đề trên. Việc sinh ra các ca kiểm thử bán tự động từ biểu đồ tuần tự UML sử dụng phân tích cấu trúc điều khiển được đưa ra bởi Lie và Lin [2], song vấn đề đồng bộ chưa được đề cập. Trong [4,6] sử dụng thuật toán tìm kiếm theo chiều sâu, vì vậy không thể khai thác hết được việc chen ngang vào giữa tuần tự các thông điệp trong toán tử song song của biểu đồ tuần tự. Phương pháp trong [3] sinh ra tất cả các kịch bản có thể xảy ra, nên sẽ dẫn đến việc tăng quá nhanh về số lượng ca kiểm thử trong nhiều trường hợp.

Trong bài báo này, chúng tôi trình bày một phương pháp sinh các ca kiểm thử từ biểu đồ tuần tự UML với mục đích các ca kiểm thử này dùng để kiểm tra tính tương trianh trong hệ thống.

Bố cục của bài báo như sau: Sau phần Mở đầu, Mục II sẽ đưa ra một tập các quy tắc chuyển đổi từ biểu đồ tuần tự thành mô hình trung gian và đưa ra các tiêu chuẩn bao phủ tương tranh. Mục III trình bày phương pháp sinh các ca kiểm thử cho các ứng dụng tương tranh từ biểu đồ tuần tự UML. Để có thể sinh ra các ca kiểm thử theo các tiêu chuẩn bao phủ đó, chúng tôi sử dụng thuật toán tìm kiếm hàng đợi đồng thời. Minh họa ví dụ trong việc áp dụng phương pháp đó được trình bày trong Mục IV. Kết quả đạt được là các ca kiểm thử sinh ra theo phương pháp do chúng tôi đề xuất có khả năng tìm lỗi tốt hơn. Do đã chọn điểm hoán đổi (swich point) nên số lượng các ca kiểm thử đã kiểm soát được. Cuối cùng là Mục Kết luận.

II. CHUYỂN ĐỔI BIỂU ĐỒ TUẦN TỰ VÀ CÁC TIÊU CHUẨN BAO PHỦ TƯƠNG TRANH

A. Một vài khái niệm cơ bản

Một ứng dụng tương tranh: là một hệ thống có tính chất bao gồm các tiến trình tính toán được thực thi trùng nhau về mặt thời gian, trong đó các

tính toán chạy đồng thời này có thể chia sẻ các tài nguyên dùng chung.

Mô hình trung gian: sự chuyển đổi biểu đồ tuần tự vào một mô hình trung gian là bắt buộc để sinh các kịch bản kiểm thử. Mô hình trung gian phải giúp cho việc tìm kiếm các luồng điều khiển thông điệp, để việc sinh các ca kịch bản kiểm thử được dễ dàng. Các mô hình trung gian gồm:

• Biểu đồ hoạt động UML [9].

• Mạng Petri là mô hình hình thức có thể hỗ trợ các ứng dụng tương tranh [10].

• Đồ thị luồng điều khiển (hỗ trợ phân tích luồng điều khiển của biểu đồ tuần tự) [8].

Trong biểu đồ hoạt động UML: nút fork là nút điều khiển biểu thị sự phân tách thành nhiều luồng đồng thời [9]. Nút join là nút điều khiển để đồng bộ nhiều luồng đồng thời [9].

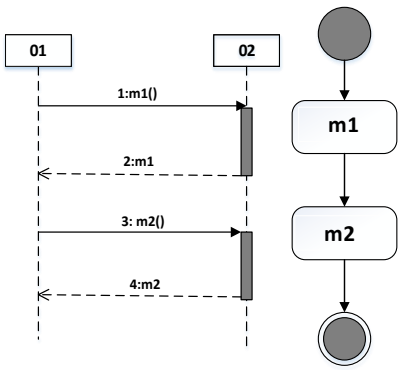
B. Chuyển đổi biểu đồ tuần tự thành mô hình trung gian cho các luồng tương tranh

Quá trình sinh các tuần tự kiểm thử từ các mô hình UML cần có một dạng biểu diễn trung gian, trong phương pháp đưa ra chọn mô hình trung gian là biểu đồ hoạt động UML.

Chuyển biểu đồ tuần tự thành mô hình trung gian: đầu tiên, các thông điệp được chuyển đổi thành các nút tương ứng, sự liên tiếp giữa hai thông điệp theo trình tự thời gian được tạo thành một cạnh. Tiếp theo, tuỳ thuộc vào từng toán tử của biểu đồ thì mô hình trung gian sẽ được xây dựng tương ứng.

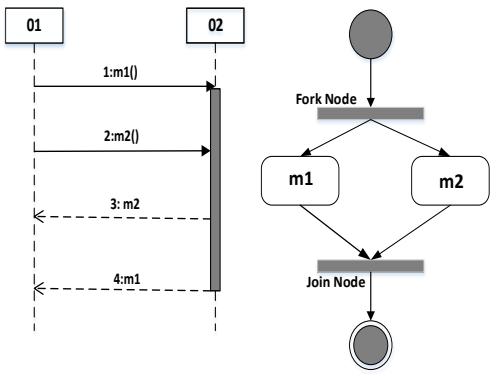
Thủ tục sau miêu tả chi tiết sự chuyển đổi của thông điệp đồng bộ, các thông điệp bất đồng bộ và toán tử song song trong biểu đồ tuần tự.

Thông điệp đồng bộ: là một cặp thông điệp bao gồm thông điệp gọi và thông điệp trả lời (ví dụ trên Hình 1, thông điệp gọi 1: m1() và thông điệp trả lời 2: m1 là một cặp thông điệp), sau khi thông điệp 1: m1() gọi xong phải đợi thông điệp trả lời 2:m1 thì mới có thể thực hiện gọi tiếp một công việc khác (chẳng hạn 3:m2()). Một cặp thông điệp đồng bộ được biểu diễn bởi một nút trong đồ thị (1: m1() và 2: m1 chuyển thành nút m1 tương ứng). Do đó, chỉ xây dựng một nút tương ứng trong đồ thị đối với một cặp thông điệp đồng bộ.



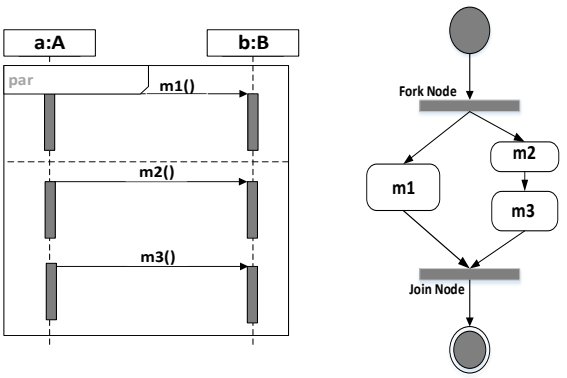
Hình 1. Các thông điệp đồng bộ

Thông điệp bất đồng bộ: là thông điệp được gửi tới một đối tượng, mà đối tượng gửi thông điệp đó không chờ đợi trả lời. Việc gọi đối tượng bắt đầu đến đối tượng kế tiếp được thực hiện ngay lập tức. Thông điệp bất đồng bộ đưa ra việc thực thi đồng thời của quá trình gọi đối tượng và tiếp nhận của đối tượng được gọi. Do đó, thông điệp bất đồng bộ được chuyển thành nút fork của mô hình trung gian. Sự bắt đầu của thông điệp bất đồng bộ trong biểu đồ tuần tự được mô hình hoá bằng nút fork. Thông điệp trả lời của thông điệp bất đồng bộ trong biểu đồ tuần tự được mô hình hoá bởi nút join tương ứng của mô hình. Một luồng chỉ ra luồng điều khiển của quá trình đối tượng được gọi và một luồng khác chỉ ra quá trình gọi đối tượng (Hình 2).



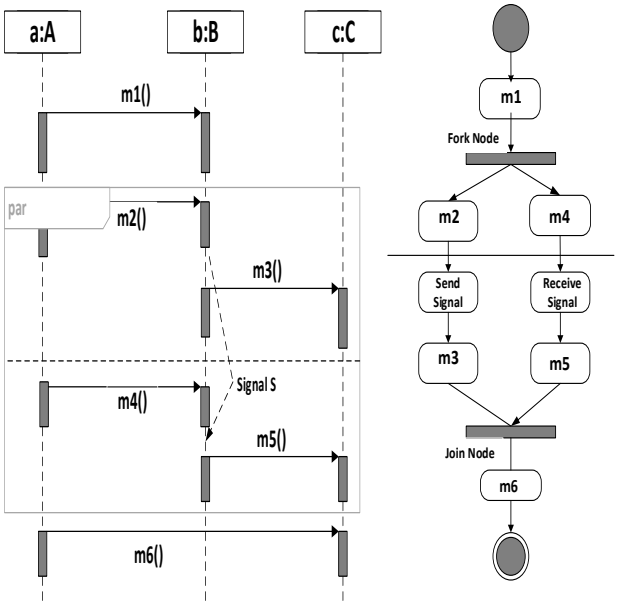
Hình 2. Các thông điệp bất đồng bộ

Toán tử song song: được sử dụng cho việc thực thi các toán hạng đồng thời. Mỗi luồng toán hạng đồng thời được chuyển đổi thành luồng cấu trúc fork-join tương ứng trong mô hình trung gian (Hình 3). Các thông điệp trong từng toán hạng giữ thứ tự thực thi của chúng, trong khi đó các luồng thông điệp giữa các toán hạng được chen ngang theo bất kỳ thứ tự nào. Toán tử song song được biểu diễn theo cấu trúc nút fork tại thời điểm bắt đầu và nút join tại thời điểm kết thúc. Mỗi toán hạng là một luồng được tạo tương ứng (Hình 3).



Hình 3. Toán tử song song

Thứ tự chung: trong toán tử song song, thứ tự bất kỳ trong các toán hạng giữa các thông điệp được đưa ra là thứ tự chung trong biểu đồ tuần tự. Các thông điệp trong các toán hạng khác nhau có thể liên quan với nhau, nó được nhấn mạnh bởi đường kẻ nét đứt với mũi tên.



Hình 4. Toán tử song song với quá trình gửi và nhận tín hiệu

Một vài thông điệp cần được thực thi bởi thứ tự được chỉ rõ. Trong toán tử song song, điểm bắt đầu của đường kẻ nét đứt là gửi tín hiệu (send signal) và điểm kết thúc của đường kẻ nét đứt là nhận tín hiệu (receive signal). Việc gửi và nhận tín hiệu cũng được chuyển đổi thành các nút tương ứng theo đúng thứ tự gửi và nhận của chúng giữa các toán hạng (Hình 4).

C. Các tiêu chuẩn bao phủ tương tranh

Tiêu chuẩn tương tranh yếu: là kịch bản kiểm thử được đưa ra để bao phủ một tuần tự khả thi của các luồng đồng thời giữa cặp nút fork và join mà không xét đến sự chen ngang của các thông điệp giữa các luồng này.

Tiêu chuẩn tương tranh trung bình: là kịch bản kiểm thử được đưa ra để bao phủ tất cả các tuần tự khả thi của các luồng đồng thời giữa cặp nút fork và join mà không xét đến sự chen ngang của các thông điệp giữa các luồng này.

Tiêu chuẩn tương tranh mạnh: là kịch bản kiểm thử được đưa ra để bao phủ tất cả các tuần tự khả thi của các thông điệp và các luồng đồng thời giữa các cặp nút fork và join, mà có xét đến sự chen ngang của các thông điệp giữa các luồng này.

Tất cả các tiêu chuẩn bao phủ tương tranh trên đều yêu cầu các kịch bản kiểm thử đưa ra để bao phủ từng luồng đồng thời thực thi ít nhất một lần. Trong cả tiêu chuẩn bao phủ tương tranh yếu và trung bình, kịch bản kiểm thử của các thông điệp và luồng các thông điệp trong tiến trình đồng thời thực hiện một cách tuần tự, trong khi đó tiêu chuẩn bao phủ tương tranh mạnh xem xét sự chen ngang của các thông điệp và tuần tự các thông điệp từ các luồng này.

D. Độ phân tích đột biến

Độ phân tích đột biến là kỹ thuật kiểm tra lỗi dựa trên giả thiết là một số loại lỗi có thể được đưa vào chương trình và sau đó thiết kế các trường hợp kiểm thử nhằm vào việc phát hiện các lỗi như vậy [11]. Phân tích đột biến cũng đã được sử dụng rộng rãi để đánh giá hiệu quả của các kỹ thuật kiểm thử khác nhau. Nó áp dụng các toán tử đột biến để đưa vào chương trình kiểm thử, do đó phát sinh một tập các phiên bản lỗi, gọi là các đột biến. Sau khi tạo ra các đột biến, một tập hợp các ca kiểm thử được thực hiện trên chúng. Khi một ca kiểm thử thực hiện tạo ra các đột biến khác nhau, đầu ra không đúng khi so sánh với chương trình gốc thì có nghĩa là đột biến bị tiêu diệt. Độ đo đột biến (Mutation Score) biểu diễn chất lượng khả năng tìm lỗi của một tập các ca kiểm thử, được xác định theo công thức sau:

$$MS\ (p,t) = \frac{N_k}{N_m - N_e}$$

Trong đó p là chương trình kiểm thử, t là các bộ kiểm thử, N_k là số lượng đột biến bị tiêu diệt, N_m là tổng số đột biến, và N_e là tổng số đột biến tương đương. Trong hệ thống cần kiểm thử, độ đo *MS* tốt xác định hiệu quả của kỹ thuật kiểm thử. Vì vậy, chúng ta sử dụng *MS* để đánh giá chất lượng các ca kiểm thử sinh ra theo cách tiếp cận được đưa ra.

III. PHƯƠNG PHÁP SINH CÁC CA KIỂM THỬ

Việc sinh các tuần tự kiểm thử từ mô hình trung gian cần duyệt đồ thị sử dụng thuật toán DFS hoặc BFS. Các đường dẫn tuần tự được sinh ra theo các thuật toán đó thích hợp cho việc bao phủ các nút và các cạnh của đồ thị, nhưng không giải quyết được vấn đề đồng bộ và an toàn dữ liệu. Để sinh các ca kiểm thử theo các tiêu chuẩn bao phủ tương tranh đã được đưa ra, chúng tôi sử dụng phương pháp thuật toán tìm kiếm hàng đợi đồng thời (CQS) để có thể tìm các lỗi đồng bộ tương tranh và an toàn dữ liệu. Thuật toán CQS gồm hai giai đoạn (chi tiết thuật toán được miêu tả trong Thuật toán 1). Thuật toán sinh ra một tập các đường dẫn kiểm thử, một đường từ nút bắt đầu đến nút kết thúc.

Giai đoạn 1: Thuật toán CQS xây dựng một hàng đợi cho từng luồng thực thi. Các hàng đợi đồng thời được sinh ra từ mô hình trung gian. Khi bắt đầu, các nút trong đồ thị được xếp vào hàng đợi chính. Để xử lý các luồng đồng thời, tại nút fork các hàng đợi con mới được sinh ra theo từng luồng đi ra từ nút đó. Mỗi hàng đợi được xếp hàng với các nút trong các luồng thực thi tương ứng cho đến khi gặp nút join của mô hình trung gian. Sau nút join, các nút còn lại được xếp vào hàng đợi chính.

Giai đoạn 2: Việc tìm kiếm bắt đầu từ hàng đợi chính. Tại mỗi bước, thuật toán lấy lần lượt từng nút thông điệp từ hàng đợi chính và thêm vào đường dẫn kiểm thử. Khi nút đó là nút fork thì các hàng đợi con biểu diễn cho từng luồng được chọn đến. Thuật toán sẽ chọn ngẫu nhiên một hàng đợi con, lần lượt lấy các nút cho đến nút release lock hoặc send signal, receive signal từ hàng đợi con đó và cập nhật vào đường dẫn. Sau đó, thuật toán sẽ duyệt sang các hàng đợi con khác cho đến khi gặp nút join. Khi tất cả các hàng đợi con đồng thời mà tín hiệu đi ra từ nút fork là rỗng thì thuật toán sẽ tiếp tục chuyển sang lấy các nút từ hàng đợi chính cho đến nút kết thúc của đồ thị và cập nhật tiếp vào đường dẫn. Tập hợp các đường dẫn được sinh ra chính là các kịch bản kiểm thử cho việc kiểm thử các lỗi tương tranh trong biểu đồ tuần tự.

Thuật toán 1: Sinh ra các ca kiểm thử từ mô hình trung gian G

Input: G với nút bắt đầu *in* và các nút kết thúc fn_i.

Output: T là tập hợp các kịch bản kiểm thử, t là một đường dẫn kiểm thử.

```
1.  T =∅;
2.  main_queue=∅;
3.  curNode =in;   //nút hiện tại là nút bắt đầu
4.  repeat
5.        move to next node
6.        if curNode == message node then
7.           main_queue.append(message);
8.        end if
9.        if curNode ==fork node then
10.          Tạo các sub_queue cho từng luồng tín hiệu đi ra từ nút fork.
11.          Đưa các nút message vào từng luồng tương ứng cho đến khi gặp nút join
12.       end if
13. until graph end
14. t = ϕ;
15. repeat
16.       message =main_queue.dequeue();
17.       if message == message node then
18.          t.append(message);
19.       end if
20.       if curNode ==fork node then
21.          active all sub_queue của fork node;
22.          repeat
23.             Chọn ngẫu nhiên sub_queue;
                Cập nhật các message vào path t cho đến khi gặp release lock or trước khi send signal hoặc sau receive signal
24.          until all sub_queue là rỗng
25.       end if
26.       if curNode == fn then
27.          T = T + { t}
28.       end if
29. until end of main queue
```

Trong kịch bản đồng thời, việc chọn các điểm hoán đổi thích hợp cho các thông điệp đan xen giữa các hàng đợi là quan trọng. Nếu không có điểm hoán đổi nào cho các luồng đó, thì các thông điệp sẽ thực thi tuần tự từng cái một. Tuần tự sinh ra theo cách này thì khả năng tìm lỗi sẽ khó hơn tại các điểm đồng bộ và điểm chia sẻ dữ liệu trong các luồng. Nếu điểm hoán đổi là từng thông điệp trong hàng đợi thì số lượng các đường kiểm thử sẽ tăng lên theo hàm mũ. Trong trường hợp các luồng đồng thời không chia sẻ bất kỳ dữ liệu chung hoặc không gửi hay nhận các tín hiệu giữa các luồng khác nhau, thì việc sinh ra tuần tự có thể

chen ngang theo bất kỳ thứ tự nào. Do đó, thuật toán sẽ không tìm ra các lỗi đồng bộ hay chia sẻ dữ liệu nào. Những loại luồng này gọi là luồng đồng thời và bất kỳ biểu diễn đan xen của các luồng là đủ tốt để kiểm tra thực thi của tất cả các thông điệp đồng thời. Các luồng đồng thời chia sẻ dữ liệu hoặc cần một thứ tự giữa các luồng khác nhau, mà việc đan xen được thực hiện một cách nghiêm ngặt được gọi là các luồng đồng bộ. Những luồng đồng bộ này cần chọn điểm hoán đổi thích hợp trong các hàng đợi để sinh các kịch bản kiểm thử có khả năng tìm lỗi tốt.

Việc chọn điểm hoán đổi thích hợp sẽ phát sinh các tuần tự kiểm thử khả thi trong biểu diễn tương tranh. Điểm chia sẻ dữ liệu giữa các luồng khác nhau cần được truy cập đồng bộ. Các điểm chia sẻ dữ liệu này được truy cập trong một đoạn găng. Đoạn găng được thực thi bằng việc sử dụng cơ chế khoá. Để bắt các điểm lỗi an toàn dữ liệu trong các luồng đồng bộ, ta thay thế điểm hoán đổi trước và sau khi khoá. Trong thuật toán đưa ra, điểm hoán đổi được chọn là trước khi release lock. Điểm hoán đổi sẽ cố gắng bắt các lỗi an toàn dữ liệu trong khi thực thi các luồng đồng bộ. Tuần tự kiểm thử phát sinh bởi thuật toán này cho việc chia sẻ dữ liệu có nghĩa là: Nếu tuần tự kiểm thử này được thực thi với kết quả mong đợi, thì ca kiểm thử thành công. Nếu tuần tự kiểm thử không thực thi hoặc thực thi với kết quả mong đợi bị sai thì ca kiểm thử lỗi. Thứ tự bất kỳ giữa các thông điệp trong các luồng khác nhau được thực thi như là việc gửi tín hiệu từ một luồng này sang một luồng khác. Để bắt các lỗi trong thứ tự bất kỳ, điểm hoán đổi có thể được xác định ở trong các thời điểm sau: trước và sau khi gửi tín hiệu, trước và sau khi nhận tín hiệu. Trong thuật toán đưa ra, các điểm hoán đổi được chọn là trước khi gửi tín hiệu và sau khi nhận tín hiệu. Những điểm hoán đổi này sẽ bắt các lỗi thứ tự trong việc thực thi các luồng đồng bộ.

IV. MỘT SỐ VÍ DỤ ÁP DỤNG

Trong [5] và [7] luồng điều khiển của biểu đồ tuần tự được đưa ra theo cơ chế ngược. Những luồng điều khiển này có thể được sử dụng để sinh tuần tự kiểm thử. Trong phương pháp đưa ra, tuần tự kiểm thử được sinh ra để kiểm thử các lỗi đồng bộ trong hệ thống sử dụng biểu đồ tuần tự. Chúng tôi đưa ra thuật toán CQS để tìm lỗi trong các ứng dụng tương tranh. Những tuần tự kiểm thử được sinh ra bởi thuật toán BFS và DFS tạo nên các kịch bản của hệ thống. Các kịch bản dựa trên thuật toán DFS thì đạt độ bao phủ tương tranh trung bình, còn kịch bản theo thuật toán CQS được đưa ra ở trên thì đạt độ bao phủ tương tranh mạnh. Do đó, khả năng tìm lỗi bao phủ cho một thứ tự bất kỳ trong các luồng và lỗi an toàn dữ liệu sẽ tốt hơn. Xem xét các ví dụ sau để so sánh chất lượng của các tuần tự kiểm thử sinh ra bởi thuật toán DFS, BFS và thuật toán CQS về độ bao phủ và khả năng bao phủ các lỗi đồng bộ.

Ví dụ 1: Hình 4 đã nêu ở trên biểu diễn các thực thi đồng bộ. Thực thi này bao gồm thứ tự ngẫu nhiên giữa các thông điệp trong thực thi đồng bộ được đưa vào theo thứ tự chung trong biểu đồ tuần tự, và trong việc gửi hoặc nhận tín hiệu trong biểu đồ tuần tự. Trong kịch bản có một thứ tự bất kỳ, kịch bản kiểm thử có thể phát hiện được các lỗi đồng bộ. Thực thi của các kịch bản nên duy trì thứ tự chung được đưa ra trong biểu đồ tuần tự. Cách khả thi để kiểm tra lỗi đó là lấy tuần tự thực thi không có ý nghĩa, để nó vi phạm thứ tự thông thường. Nếu tuần tự kiểm thử không có ý nghĩa, thì thực thi không theo thứ tự thông thường trong bản thiết kế.

Các kịch bản kiểm thử được sinh ra bởi thuật toán BFS, DFS và thuật toán CQS (Hình 4) được đưa ra trong Bảng 1. Thuật toán BFS và DFS sinh ra các kịch bản kiểm thử bao gồm một tuần tự có ý nghĩa và một tuần tự không có ý nghĩa. Tuy nhiên, khó có thể tìm được một tuần tự theo thứ tự thông thường có lỗi.

Thuật toán CQS sinh ra cả hai kịch bản đều là các tuần tự không có ý nghĩa. Vì vậy, khả năng tìm lỗi thứ tự trong thuật toán CQS cao hơn thuật toán BFS và DFS.

BẢNG 1. CÁC KỊCH BẢN KIỂM THỬ ĐƯỢC SINH RA, KHẢ NĂNG TÌM LỖI (CHO HÌNH 4)

Thuật toán	**Các tuần tự kiểm thử**	**Khả năng tìm lỗi thứ tự**	**Độ bao phủ tương tranh**
DFS	m_1-m_2-S_s- m_3-m_4-R_s-m_5-m_6 **(T1)**	Không	Trung bình
	m_1-m_4-R_s- m_5-m_2-S_s-m_3-m_6 **(T2)**	Có	Trung bình
BFS	m_1-m_2-m_4-S_s-R_s- m_3-m_5 –m_6 **(T3)**	Không	Mạnh
	m_1-m_4-m_2-R_s-S_s- m_5-m_3 –m_6 **(T4)**	Có	Mạnh
CQS	m_1-m_2-m_4-R_s-S_s- m_5-m_3 –m_6 **(T5)**	Có	Mạnh
	m_1-m_4-R_s -m_2-S_s-m_5-m_3 –m_6 **(T6)**	Có	Mạnh

Áp dụng độ đo đột biến cho từng bộ kiểm thử để phân tích khả năng dò tìm lỗi của chúng. Để thấy hiệu quả của thuật toán CQS, chúng ta lần

lượt lấy số ca kiểm thử trên một bộ kiểm thử lần lượt là 2/10/15/30. Từ Bảng 2 biểu diễn kết quả MS cho từng kịch bản, theo mức phương thức và lớp.

BẢNG 2. ĐỘ ĐO ĐỘT BIẾN(MS) CHO TỪNG TUẦN TỰ KIỂM THỬ THEO CÁC THUẬT TOÁN TRÊN (ĐƠN VỊ %)

Mức độ	**Số các ca kiểm thử**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**
Phương thức	2	30,4	31,2	40,7	42,5	61,5	61,5
	10/ /15/30	40,2	30,5	42,5	45,5	62,3	60,5
Lớp	2/10/ 15/30	50,3	52,3	54,2	60,5	74,5	81,5

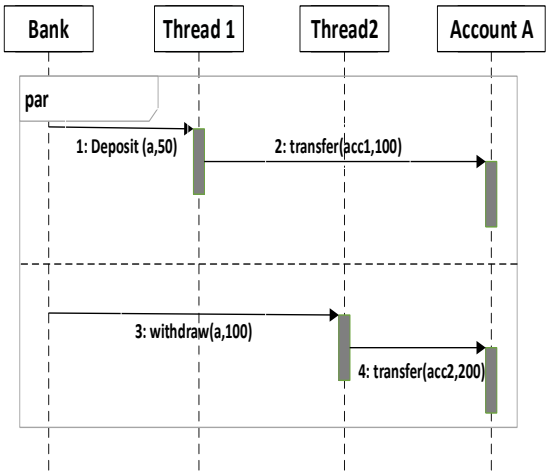
Từ kết quả Bảng 2, chúng ta thấy kết quả và có những nhận xét sau:

(i) Với khả năng tìm lỗi cả mức phương thức và mức lớp, các bộ kiểm thử được sinh ra (T5,T6) theo thuật toán CQS có khả năng dò tìm lỗi tốt hơn so với các bộ kiểm thử theo DFS (T1,T2) và BFS (T3,T4).

(ii) Khả năng dò tìm lỗi của các ca kiểm thử sinh ra trong mức độ lớp cao hơn so với mức độ phương thức.

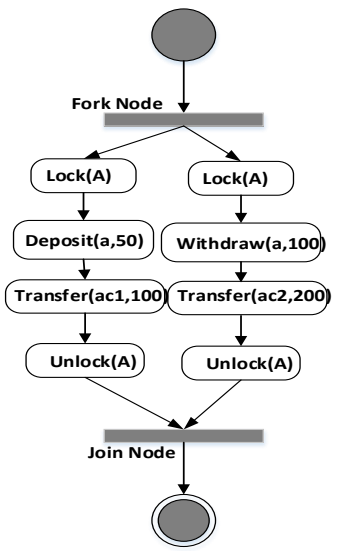
(iii) Các bộ kiểm thử khác nhau có khả năng dò tìm lỗi khác nhau.

Ví dụ 2: Trong Hình 5 dưới đây biểu diễn: chức năng giao dịch ngân hàng gồm gửi tiền (deposit), chuyển tiền (transfer) và rút tiền (withdraw) của tài khoản A. Vì các tiến trình đồng thời đều thực hiện với tài khoản A, nên sẽ có điểm truy cập chia sẻ dữ liệu. Các kịch bản kiểm thử sinh ra bởi thuật toán CQS có khả năng tìm lỗi an toàn dữ liệu, nên việc thực thi đồng thời có khả năng dò tìm các trạng thái không nhất quán trong chia sẻ dữ liệu nhờ chen ngang các thông điệp được chỉ rõ của quá trình thực thi. Cách khả thi để tạo ra sự chen ngang là hoán đổi thực thi giữa các luồng bên trong đoạn găng. Tuần tự kiểm thử được tạo ra bởi các điểm hoán đổi, vì vậy, khi kiểm tra dữ liệu không nhất quán, CQS sẽ phát hiện được các lỗi về an toàn dữ liệu.



Hình 5. Biểu đồ tuần tự của chức năng Giao dịch trong hệ thống ngân hàng (cùng thực hiện với Tài khoản A)

Các kịch bản tuần tự sinh ra bởi thuật toán DFS, BFS và CQS (cho Hình 6) được chỉ ra trong Bảng 3. Việc thực thi là tuần tự trong các kịch bản kiểm thử được sinh ra bởi thuật toán DFS không cho phép chen ngang các tuần tự thông điệp, do đó các ca kiểm thử theo thuật toán này đạt độ bao phủ tương tranh trung bình, nó không có khả năng tìm lỗi an toàn dữ liệu. Thuật toán BFS và CQS sinh ra các kịch bản kiểm thử mà nó có khả năng tìm ra lỗi về an toàn dữ liệu. Việc thực thi có thể chen ngang giữa các thông điệp, vì vậy nó đạt độ bao phủ tương tranh mạnh. Thuật toán CQS chọn điểm hoán đổi là trước và sau khi chia sẻ dữ liệu nên số lượng ca kiểm thử và khả năng tìm lỗi sẽ tốt hơn thuật toán BFS (xem Bảng 3).



Hình 6. Mô hình trung gian của giao dịch trong hệ thống ngân hàng

BẢNG 3. CÁC KỊCH BẢN KIỂM THỬ ĐƯỢC SINH RA, KHẢ NĂNG TÌM LỖI CHIA SẺ DỮ LIỆU CHO CHỨC NĂNG GIAO DỊCH TRONG HỆ THỐNG NGÂN HÀNG

TT	Các tuần tự kiểm thử	Khả năng tìm lỗi chia sẻ dữ liệu	Độ bao phủ tương tranh
DFS	Lock(A)- Deposit(50)- Transfer(100)-Unlock(A)- Lock(A)-Withdraw(100)- Transfer(200)-Unlock(A) **(T1)**	Không	Trung bình
	Lock(A)-Withdraw(100)- Transfer(200)-Unlock(A)- Lock(A)- Deposit(50)- Transfer(100)-Unlock(A) **(T2)**	Không	Trung bình
BFS	Lock(A) - Lock(A)- Deposit(50)- Withdraw(100)- Transfer(100) Transfer(200) -Unlock(A) - Unlock(A) **(T3)**	Có	Mạnh
	Lock(A) - Lock(A)- Withdraw(100)- Deposit(50)- Transfer(200) Transfer(100) -Unlock(A) - Unlock(A) **(T4)**	Có	Mạnh
CQS	Lock(A) - Deposit(50) Lock(A) -Withdraw(100)- Transfer(100) Transfer(200) -Unlock(A) - Unlock(A) **(T5)**	Có	Mạnh
	Lock(A) - Withdraw(100)- Lock(A) - Deposit(50)- Transfer(200) Transfer(100) -Unlock(A) - Unlock(A) **(T6)**	Có	Mạnh
	Lock(A) - Deposit(50)- Transfer(100) -Lock(A) - Withdraw(100)- Transfer(200) -Unlock(A) - Unlock(A) **(T7)**	Có	Mạnh
	Lock(A) Withdraw(100)- Transfer(200)- Lock(A) Deposit(50)- Transfer(100)- Unlock(A)-Unlock(A)(**T8**)	Có	Mạnh

Tương tự như Ví dụ 1, chúng ta sử dụng độ đo MS để đo khả năng tìm lỗi của các bộ kiểm thử T1, T2, T3, T4, T5, T6, T7, T8 được sinh ra theo từng cách tiếp cận của thuật toán DFS, BFS so với thuật toán CQS đưa ra (chọn số ca kiểm thử trên một bộ kiểm thử là 15/20/40) (Bảng 4).

Tương tự, số liệu từ Bảng 4 cho thấy độ đo *MS* của các tuần tự kiểm thử T5, T6, T7, T8 được sinh ra do thuật toán CQS cao hơn so với T1, T2, T3 và T4. Do đó, khả năng tìm lỗi của các bộ kiểm thử do CQS đưa ra tốt hơn so với các thuật toán DFS và BFS.

BẢNG 4. ĐỘ ĐO ĐỘT BIẾN (MS) CHO TỪNG TUẦN TỰ KIỂM THỬ THEO CÁC THUẬT TOÁN TRÊN (ĐƠN VỊ %)

Mức độ	T1	T2	T3	T4	T5	T6	T7	T8
Phương thức	32,4	31,2	41,7	43,5	61,5	61,5	59,5	60,5
	41,5	30,5	43,5	44,5	62,3	60,5	60,3	59,7
Lớp	51,5	51,3	52,2	59,5	74,5	78,5	76,5	80,3

Kết quả thu được từ Ví dụ 1 và Ví dụ 2 minh chứng tính hiệu quả của thuật toán CQS đưa ra tốt hơn so với thuật toán BFS và DFS cả về độ bao phủ tương tranh và khả năng dò tìm lỗi của các bộ kiểm thử được sinh ra.

V. KẾT LUẬN

Trong bài báo, các tác giả đã trình bày thuật toán tìm kiếm hàng đợi đồng thời (CQS) cho việc sinh ra các kịch bản kiểm thử từ biểu đồ tuần tự UTM trong các ứng dụng tương tranh. Thuật toán này cung cấp kịch bản với mục đích tìm các lỗi an toàn dữ liệu trong các ứng dụng tương tranh và các lỗi thứ tự giữa các thông điệp trong các toán hạng giữa các luồng đồng thời. Các quy tắc chuyển đổi từ biểu đồ tuần tự thành mô hình trung gian cũng dễ dàng được thực thi trong phương pháp đã đưa ra. Hơn nữa, với việc dựa vào thuật toán CQS, các ca kiểm thử sinh ra đã thoả mãn tiêu chuẩn bao phủ tương tranh mạnh và khả năng dò tìm lỗi tốt, khi so sánh với thuật toán tìm BFS và DFS.

Trong thời gian tới, việc cải tiến phương pháp đưa ra và cài đặt chương trình thực nghiệm sẽ được nhóm nghiên cứu tiếp tục phát triển. Hướng phát triển này có thể áp dụng với các loại biểu đồ UML khác hoặc kết hợp nhiều loại biểu đồ để các ca kiểm thử có độ bao phủ tốt hơn. Việc 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 nhóm nghiên cứu 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]. Mark Utting and Bruno Legeard, *"*Practical Model-Based Testing: A Tools Approach". Morgan Kaufmann Publishers Inc.,San Francisco, CA, USA, 2006.

[2]. Bao-Lin Li, Zhi-shu Li, Li Qing, and Yan-Hong Chen, "Test Case Automate Generation from UML Sequence Diagram and OCL Expression. In *"*Proceedings of the 2007 International Conference on Computational Intelligence and Security" (CIS'07). IEEE Computer Society, Washington, DC, USA, pp. 1048-1052, 2007.

[3]. Khandai, M.; Acharya, A.A.; Mohapatra, "D.P. A novel approach of test case generation for concurrent systems using UML Sequence Diagram". Electronics Computer Technology (ICECT), 3rd International Conference", vol. 1, pp. 157-161, 2011.

[4]. M. Dhineshkumar and Galeebathullah, "An Approach to Generate Test Cases from Sequence Diagram. In Proceedings of the 2014 International Conference on Intelligent Computing Applications (ICICA '14)". IEEE Computer Society, Washington, DC, USA, pp. 345-349, 2014.

[5]. V. Garousi, L. Briand, and Y. Labiche, Control Flow Analysis of UML 2.0 Sequence Diagrams. In A. Hartman and D. Kreische, editors, "Model Driven Architecture -Foundations and Applications, volume 3748 of LNCS", pp. 160-174. Springer, 2005.

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

[7]. A. Rountev, O. Volgin, and M. Reddoch. "Static Control-Flow Analysis for Reverse Engineering of UML Sequence Diagrams". SIGSOFT Software Engineeing Notes, pp. 96-102, September 2005.

[8]. E. Cartaxo, F. Neto, and P. Machado, "Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems". In IEEE International Conference on Systems, Man and Cybernetics, ISIC, pp. 1292-1297, 2007.

[9]. OMG, "UML 2.0 Superstructure Specification", OMG Adopted Specification.

[10]. C. Eichner, H. Fleischhack, U. Schrimpf, and C. Stehno. "Compositional Semantics for UML 2.0 Sequence Diagrams Using Petri Nets". In 12th Int. SDL Forum of LNCS, vol. 3530, pp. 133-148. Springer, 2005.

[11]. Sun C-A, Wang G, Cai K-Y, Chen TY, "Distribution-aware mutation analysis". Proceedings of 9th IEEE International Workshop on Software Cybernetics (IWSC 2012) , IEEE Computer Society, Izmir, Turkey, pp. 170-175, 2012.

SƠ LƯỢC VỀ TÁC GIẢ

ThS. Vũ Thị Đào

Đơn vị công tác: Khoa Công nghệ thông tin, Học viện Kỹ thuật mật mã, Ban Cơ yếu Chính phủ, Hà Nội

E-mail: vuthidao@gmail.com

Nhận bằng Thạc sĩ Công nghệ thông tin, chuyên ngành Công nghệ Phần mềm tại Trường Đại học Công nghệ năm 2008. Đang là nghiên cứu sinh tại trường Đại học Công nghệ từ tháng 11/2010 - chuyên ngành Công nghệ phần mềm.

Hướng nghiên cứu hiện nay: Kiểm thử phần mềm dựa trên mô hình, kiểm thử tự động phần mềm.

PGS. TS. Phạm Ngọc Hùng

Đơn vị công tác: Đại học Công nghệ, Đại học Quốc gia Hà Nội, Hà Nội.

Email: hungpn@vnu.edu.vn

Tốt nghiệp tại Đại học Công nghệ năm 2002. Nhận bằng Thạc sĩ và Tiến sĩ tại Viện Khoa học và Công nghệ Tiên tiến Nhật Bản - JAIST năm 2006 và 2009. Được phong hàm Phó Giáo Sư ngành CNTT năm 2015.

Hướng nghiên cứu hiện nay: Kiểm chứng và kiểm thử phần mềm, Kiểm chứng đảm bảo giả định, Tiến hoá phần mềm.

PGS.TS. Nguyễn Việt Hà

Đơn vị công tác: Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội, Hà Nội.

E-mail: hanv@vnu.edu.vn

Nhận bằng Cử nhân, Thạc sỹ và bảo vệ luận án Tiến sĩ tại Đại học Takushoku, Nhật Bản lần lượt vào các năm 1997, 1999 và 2002. Được phong hàm Phó Giáo sư ngành CNTT năm 2009.

Lĩnh vực nghiên cứu hiện nay: Kiểm chứng, kiểm thử phần mềm, Kiến trúc phần mềm.