BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC ĐÀ NẴNG

LÊ THANH LONG

KIỂM THỬ TỰ ĐỘNG CHO CÁC ỨNG DỤNG ĐA PHƯƠNG THỨC TƯƠNG TÁC

Chuyên ngành: Khoa học máy tính

Mã số: 62 48 01 01

LUẬN ÁN TIẾN SĨ

Đà Nẵng 2017

Công trình được hoàn thành tại:

ĐẠI HỌC ĐÀ NẴNG

Tập thể hướng dẫn: GS. TS. Ioannis Parissis

PGS. TS. Nguyễn Thanh Bình

Phản biện1:……

Phản biện 2:……

Phản biện 3:……

Luận án được bảo vệ trước Hội đồng chấm luận án cấp Đại học Đà Nẵng họp tại : Đại học Đà Nẵng

Vào hồi …..giờ…..ngày:.....tháng……năm…….

Có thể tìm hiểu luận án tại:

- Thư viện quốc gia Việt Nam
- Trung tâm thông tin và học liệu, Đại học Đà Nẵng

GIỚI THIỆU

1. Động cơ nghiên cứu

Ứng dụng đa phương thức tương tác (Interactive multimodal application – IMA) xử lý hai hoặc nhiều hơn các phương thức đầu vào (ví dụ như lời nói, ánh mắt, cử chỉ và chuyển động của cơ thể) trong một cách phối hợp với đầu ra của ứng dụng. Các ứng dụng này làm tăng sự tương tác người - máy bởi vì chúng trực quan, tự nhiên, hiệu quả và mạnh mẽ. Sự trực quan và tự nhiên thể hiện qua tính tương tác đa phương thức giữa người và máy tính, ví dụ các ứng dụng thực tại ảo. Tính hiệu quả khi người dùng sử dụng các phương thức tương đương cho cùng một nhiệm vụ. Tính mạnh mẽ là kết quả của sự tích hợp các dữ liệu đầu vào dựa vào tính bổ trợ các phương thức.

Kiểm thử các IMA đã và đang được quan tâm bởi nhiều nghiên cứu khác nhau. Phương pháp hình thức hóa ICO (Interactive Cooperative Object) dựa trên mạng Petri. Phương pháp kiểm thử dựa trên cách tiếp cận đồng bộ cũng đã được đề xuất trong [16], nhờ vào các giả thuyết đồng bộ cơ bản, các đặc tả và xác minh của chương trình trở nên đơn giản và dễ dàng hơn. Phương pháp kiểm thử này sử dụng môi trường kiểm thử Lutess và một phần đặc tả hành vi người dùng được cung cấp như là một tập hợp các biểu thức Lustre. Laya Madani và đồng nghiệp đã đề xuất phương pháp kiểm thử dữ liệu dựa trên cây nhiệm vụ và mô hình hợp nhất. Một phương pháp được sử dụng rộng rãi là kiểm thử dựa trên ngôn ngữ mô hình hóa [1]. Phương pháp này yêu cầu mô hình hóa hành vi người dùng IMA.

Các phương pháp trên sử dụng các mô hình khác nhau để xây dựng mô hình kiểm thử: *mô hình hành vi ứng dụng, mô hình nhiệm vụ tương tác, hồ sơ hoạt động và đặc tả các phương thức*. Nhiều mô hình đã làm cho việc mô hình hóa các quá trình trở nên khó khăn. Vì vậy, trong luận án "Kiểm thử tự động cho các ứng dụng đa phương thức tương tác", mục tiêu của chúng tôi là định nghĩa một ngôn ngữ mô hình hóa kiểm thử để biểu diễn duy nhất và nhất quán các đặc tính, quá trình của ứng dụng đa phương thức tương tác.

Luận án xây dựng phương pháp sinh dữ liệu thử tự động dựa trên ngôn ngữ mô hình hóa. Phương pháp này đặc tả các sự kiện đa phương thức, các thuộc tính và kiểm tra tính hợp lý của các thuộc tính CARE. Từ đó, luận án phát triển một công cụ tự động hóa kiểm thử cho các ứng dụng đa phương thức tương tác.

2. Các đóng góp chính của luận án

(1) Dựa vào việc phân tích các đặc điểm của cây nhiệm vụ tương tranh CTT (ConcurTaskTrees), một mô hình phổ biến đặc tả các ứng dụng đa phương thức tương tác, chúng tôi đã mở rộng CTT với các xác suất có điều kiện. Chính xác hơn, hành vi của người dùng trên ứng dụng đa phương thức tương tác thường bị ảnh hưởng bởi các điều kiện của ứng dụng và môi trường làm việc.

(2) Một ngôn ngữ đặc tả kiểm thử mới cho các ứng dụng đa phương thức tương tác được đề xuất, gọi là ngôn ngữ TTT (Task Tree based Test). Ngôn ngữ TTT đặc tả các kịch bản kiểm thử và hồ sơ hoạt động có điều kiện của các ứng dụng đa phương thức tương tác.

(3) Xác định các luật chuyển đổi từ CTT sang mô hình kiểm thử của ngôn ngữ TTT.

(4) Xây dựng giải pháp sinh dữ liệu kiểm thử tự động cho các ứng dụng đa phương thức tương tác.

(5) Đặc tả các sự kiện đa phương thức tương tác và kiểm tra tính hợp lý của các thuộc tính CARE trên các ứng dụng đa phương thức tương tác.

(6) Công cụ TTTEST được phát triển để tự động kiểm thử các ứng dụng đa phương thức tương tác.

3. Cấu trúc của luận án

Chương 1 trình bày nền tảng của các IMA bao gồm các khái niệm chính, các đặc điểm của tương tác đa phương thức. Các phương pháp kiểm thử cho các IMA được tóm tắt trong chương này.

Trong chương 2, luận án trình bày chi tiết phương pháp kiểm thử mà chúng tôi tập trung vào, bao gồm nền tảng về cây nhiệm vụ, máy trạng thái hữu hạn, tương tác đa phương thức, các thuộc tính CARE, hồ sơ hoạt động và xác suất có điều kiện.

Trong chương 3, chúng tôi đề xuất một ngôn ngữ mô hình hóa kiểm thử, gọi là TTT, biểu diễn các kịch bản và hồ sơ hoạt động có

điều kiện cho các IMA. Các phương pháp chuyển đổi từ CTT đến ngôn ngữ TTT cũng được trình bày trong chương này.

Chương 4 trình bày công cụ TTTEST để kiểm thử các IMA. Chúng tôi cũng trình bày các IMA và mô tả cách kiểm thử các ứng dụng này.

Chương 1. ỨNG DỤNG ĐA PHƯƠNG THỨC TƯƠNG TÁC

Trong chương một, luận án trình bày nền tảng của IMA trong đó bao gồm các định nghĩa đa phương thức và các tính năng quan trọng của sự tương tác đa phương thức. Các phương pháp kiểm thử cho IMA cũng được tóm tắt trong chương này.

1.1. Đa phương thức

Phương thức là một kênh liên lạc giữa người dùng và máy tính, một phương pháp tương tác mà người dùng sử dụng để đạt được một mục tiêu. Phương thức như một trong những giác quan, qua đó người dùng có thể nhận được kết quả đầu ra của máy tính; ngược lại, phương thức cũng là thiết bị đầu vào cho phép máy tính nhận được thông tin từ người dùng. Một phương thức tương tác bao gồm một thiết bị vật lý và một ngôn ngữ tương tác. Thiết bị vật lý có thể là một thiết bị nhập, xuất của hệ thống. Ngôn ngữ tương tác được xác định bằng cách định rõ các biểu thức mang ý nghĩa, chẳng hạn như một tập hợp các ký hiệu thông thường.

IMA xử lý hai hoặc nhiều hơn các kết hợp dữ liệu đầu vào (ví dụ như lời nói, ánh mắt, cử chỉ, đầu và chuyển động của cơ thể) trong một cách phối hợp với đầu ra của ứng dụng. Chúng là một lớp các ứng dụng có mục đích nhận dạng ngôn ngữ tự nhiên và hành vi của con người, bằng cách kết hợp một hoặc nhiều công nghệ nhận dạng khác nhau (ví dụ như lời nói, ánh mắt, cử chỉ).

1.2. Đặc điểm của tương tác đa phương thức

Do các phương thức có đặc điểm khác nhau và sự hợp nhất đa phương thức thích ứng với thời gian thực nên IMA có những đặc điểm như sau:

- Hỗ trợ khả năng nhận thức và giao tiếp của người sử dụng;

- Tích hợp các kỹ năng tính toán của máy tính trong thực tế nhằm cung cấp tương tác với con người tự nhiên hơn;
- Tăng cường việc xử lý mạnh mẽ do kết hợp các phương thức khác nhau;
- Cá nhân hóa dựa trên người dùng và bối cảnh;
- Liên quan đến nhiều người dùng và tương tác di động.

Các đặc điểm của IMA có nhiều khác biệt với các đặc điểm ứng dụng có giao diện đồ họa: Ứng dụng giao diện đồ họa quản lý các sự kiện đầu vào theo một cách tuần tự, trong khi đó, IMA có khả năng xử lý song song các sự kiện đầu vào như lời nói, cử chỉ, cảm xúc v.v.

1.3. Hợp nhất đa phương thức

Quá trình tích hợp thông tin từ các phương thức đầu vào khác nhau và kết hợp chúng thành một lệnh hoàn chỉnh được gọi là hợp nhất đa phương thức. Hợp nhất sớm bao gồm việc kết hợp các kết quả của từng bộ nhận dạng phương thức. Các cơ chế hợp nhất sớm bao gồm kỹ thuật tích hợp thống kê, lý thuyết tác tử, các mô hình Markov ẩn và các mạng nơ ron nhân tạo. Hợp nhất trễ sáp nhập các thông tin ngữ nghĩa chiết xuất bằng cách sử dụng các thủ tục hợp nhất hộp thoại để mang lại sự giải thích đầy đủ, chẳng hạn như Melting pots và khung ngữ nghĩa.

1.4. Không gian thiết kế và khung lý thuyết

Đối với mô hình thiết kế, các nhà nghiên cứu đã khái niệm hóa các mối quan hệ giữa các phương thức đầu vào và đầu ra. TYCOON là khung lý thuyết thiết kế IMA gồm hai trục: *trục X biểu diễn năm loại kết hợp của các phương thức, trục Y biểu diễn các mục tiêu đạt được khi các kết hợp thực hiện.* Mô hình CASE [24] phân loại IMA theo hai tính năng chính của chúng: xử lý đồng thời và hợp nhất dữ liệu. Mô hình các thuộc tính CARE [22] được dùng nhằm đánh giá khả năng sử dụng và tính năng trung tâm của IMA.

1.5. Kiểm thử phần mềm

Kiểm thử phần mềm là một cuộc kiểm tra được tiến hành để cung cấp cho các bên liên quan thông tin về chất lượng của một ứng

dụng phần mềm. Trong kỹ thuật kiểm thử không chỉ giới hạn ở việc thực hiện một chương trình hoặc ứng dụng với mục đích đi tìm các lỗi phần mềm mà còn là một quá trình phê chuẩn và xác minh một ứng dụng phần mềm [24].

1.5.1 Kiểm thử dựa vào mô hình

Mô hình là một mô tả đơn giản của một ứng dụng dựa trên các yêu cầu chức năng. Mô hình giúp chúng ta hiểu biết và dự đoán hành vi của ứng dụng. Kiểm thử dựa trên mô hình là một kỹ thuật kiểm thử phần mềm, trong đó trường hợp kiểm thử được sinh ra từ một mô hình của các ứng dụng. Có rất nhiều lợi ích của kiểm thử dựa trên mô hình: *phát hiện lỗi, giảm chi phí và thời gian kiểm thử, phát hiện các khiếm khuyết của yêu cầu và phát triển các yêu cầu*. Tuy nhiên, kiểm thử dựa trên mô hình không dễ dàng áp dụng trong thực tế: *khó xây dựng mô hình chính xác, yêu cầu cao cho các kiểm thử viên và tạo ra giá trị đầu ra mong đợi cho các ca kiểm thử là một vấn đề khó khăn*.

1.5.2 Kiểm thử dựa trên hồ sơ hoạt động

Kiểm thử dựa trên hồ sơ hoạt động là một kỹ thuật kiểm thử tập trung vào phân phối sử dụng chương trình ứng dụng [49]. Kỹ thuật này kết hợp kiểm thử dựa trên mô hình, kiểm thử ngẫu nhiên và kiểm thử dựa trên chuyển tiếp trạng thái. Hồ sơ hoạt động bao gồm một máy trạng thái và những phân bố xác suất. Máy trạng thái biểu diễn hành vi dự kiến của ứng dụng. Các phân phối xác suất đại diện cho các đặc điểm sử dụng mong đợi của ứng dụng.

1.5.3 Kiểm thử dựa trên yêu cầu

Kiểm thử dựa trên yêu cầu được chia thành hai giai đoạn: xem xét sự mơ hồ trong các yêu cầu và dùng kỹ thuật biểu đồ nhân quả để sinh các trường hợp kiểm thử. Xem xét sự mơ hồ là một kỹ thuật để xác định các mơ hồ trong các yêu cầu chức năng để nâng cao chất lượng những yêu cầu. Biểu đồ nhân quả là một kỹ thuật thiết kế các kiểm thử với số lượng tối thiểu các trường hợp kiểm thử nhằm bao phủ 100% các yêu cầu chức năng.

1.6. Kiểm thử các ứng dụng đa phương thức tương tác

IMA tương tác hỗ trợ giao tiếp với người dùng thông qua những phương thức khác nhau. Đa phương thức mang lại mối quan hệ tự nhiên giữa máy và người dùng nhưng cũng làm tăng độ phức tạp của hệ thống, dễ có rủi ro sinh ra lỗi trong suốt quá trình phát triển, vì vậy IMA cần phải có sự kiểm thử khắt khe. Các phương pháp kiểm thử IMA bao gồm: *Phương pháp ICO, Phương pháp đại số, Phương pháp Event B, Phương pháp đồng bộ và phương pháp sử dụng cây nhiệm vụ và mô hình hợp nhất.*

1.7. Kết luận

Chương này đã trình bày nền tảng cơ bản của tương tác đa phương thức. Các đặc điểm chính cũng như cơ sở nhận thức của sự tương tác đa phương thức đã nhấn mạnh thêm những khả năng mới của sự tương tác người - máy. Các phương pháp kiểm thử cũng được giới thiệu trong chương này.

Chương 2. NGÔN NGỮ MÔ HÌNH HÓA CTT VỚI XÁC SUẤT CÓ ĐIỀU KIỆN

Trong chương hai, luận án trình bày cây nhiệm vụ, máy trạng thái hữu hạn, tương tác đa phương thức, thuộc tính CARE, và các hồ sơ hoạt động. Chúng là nền tảng liên quan của ngôn ngữ mô hình hóa kiểm thử mới của chúng tôi. Chúng tôi đề xuất bổ sung điều kiện cho các phép toán trong cây nhiệm vụ và các phép toán là một phần cần thiết của ngôn ngữ mô hình kiểm thử.

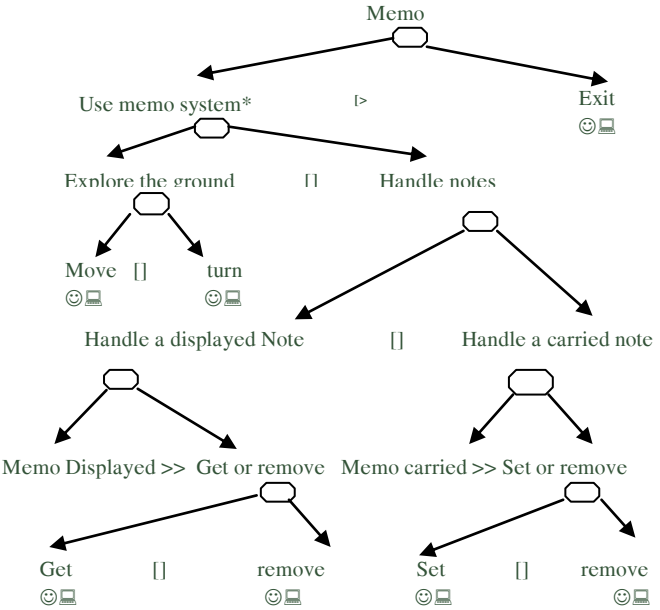
2.1. Cây nhiệm vụ

Cây nhiệm vụ [11] thường được dùng trong việc thiết kế các IMA. Chúng được sử dụng để hỗ trợ cho các giai đoạn khác nhau trong quá trình phát triển phần mềm. Trong mô hình này, các nhiệm vụ được biểu diễn phân cấp, một nhiệm vụ có những nhiệm vụ con kết hợp với nhau bởi yếu tố thời gian. Mô hình nhiệm vụ được biểu diễn bởi cây nhiệm vụ tương tranh CTT [27]. Cây CTT chứa những thông tin về hành vi của người dùng mong muốn hay những phản ứng dự kiến sẽ diễn ra của hệ thống, do đó nó cung cấp một mô hình hành vi

của người dùng và hỗ trợ cho việc mô hình hóa kiểm thử. CTT gồm bốn loại nhiệm vụ: *nhiệm vụ người dùng, nhiệm vụ ứng dụng, nhiệm vụ tương tác và nhiệm vụ trừu tượng.*

2.2. Ứng dụng đa phương thức tương tác Memo

Memo [4] là một IMA nhằm chú thích địa điểm vật lý bằng các ghi chú kỹ thuật số được trình bày như trong Hình 2.1. Người dùng có thể kéo thả một ghi chú đến một vị trí vật lý. Ghi chú sau đó có thể được đọc/lưu trữ/gỡ bỏ bởi người dùng di động khác. Memo được trang bị một hệ thống định vị GPS và một khí cụ đo từ cho phép hệ thống tính toán vị trí của mình và định hướng. Người dùng Memo được mang một đầu nghe gắn kết hiển thị (Head Mounted Display - HMD).



Hình 2.1. Cây CTT cho ứng dụng Memo.

Memo cung cấp ba chức năng chính: (1) Định hướng và định vị người dùng di động, để từ đó hệ thống có thể hiển thị các ghi chú theo vị trí và định hướng hiện tại của người dùng; (2) Các thao tác trên một ghi chú bao gồm: nhận (get), cài đặt (set) và hủy bỏ (remove); (3)

Thoát khỏi hệ thống (exit). Hình 2.1 cho thấy CTT cho các ứng dụng Memo (nhiệm vụ tương tác được biểu diễn bởi biểu tượng ☺💻).

2.3. Hồ sơ hoạt động

Hồ sơ hoạt động [28] cung cấp thông tin về cách sử dụng hiệu quả của ứng dụng. Đặc biệt, chúng có thể được sử dụng để định hướng quá trình kiểm thử. Đối với các IMA, hồ sơ hoạt động có thể dễ dàng được xác định bằng cách gán xác suất xảy ra với một số hành động được mô tả trong ứng dụng. Laya Madani và các đồng nghiệp đã đề xuất để nâng cao cây nhiệm vụ với hồ sơ hoạt động bằng cách gán xác suất cho các hành động người dùng theo cú pháp sau:

$T :: t \mid T[]_{pr1,pr2}\, T \mid T|||_{pr1,pr2}\, T \mid T[]|_{pr1,pr2}\, T \mid T|=|_{pr1,pr2}\, T \mid T[>_{pr} T \mid T >> T \mid T[]>> T \mid T>_{pr} T \mid T^* \mid T(n) \mid [T]_{pr} \mid$

Trong đó, *t* là một nhiệm vụ cơ bản bao gồm nhiệm vụ ứng dụng hoặc nhiệm vụ người dùng.

2.4. Máy trạng thái hữu hạn

Máy trạng thái hữu hạn [20] (Finite State Machine - FSM) được sử dụng rộng rãi để mô hình hóa hành vi của ứng dụng tương tác. Mô hình này bao gồm các trạng thái, các hành động, các chuyển tiếp và được trình bày bởi một sơ đồ trạng thái. Khi một ứng dụng tương tác được đặc tả bởi một máy trạng thái hữu hạn, các trạng thái đại diện cho một sự trừu tượng hóa tình trạng của các hoạt động trong ứng dụng tương tác và hầu hết các hoạt động này liên quan đến trạng thái kế tiếp. Sau khi ứng dụng tương tác được mô hình hóa bởi các FSMs, bước tiếp theo là sinh ca kiểm thử từ các mô hình này. Mỗi ca kiểm thử là một chuỗi các chuyển tiếp bắt đầu từ trạng thái đầu và kết thúc khi gặp một trong những trạng thái cuối.

2.5. Các mức độ trừu tượng kiểm thử

Mức độ trừu tượng kiểm thử phụ thuộc vào các tính chất và các thành phần của IMA nhận dữ liệu thử nghiệm. Có 3 mức độ trừu tượng của các sự kiện tương tác như sau:

– *Mức 1*: Các sự kiện sẽ được gửi đến thành phần tương tác logic. Trong trường hợp này, dữ liệu kiểm thử là các chuỗi sự kiện phụ

thuộc ở mức độ thấp như số lần nhấp chuột hoặc lựa chọn các nút bằng cách sử dụng chuột.

- *Mức 2:* Các sự kiện gửi đến các cơ chế hợp nhất phương thức, ví dụ <chuột, mở file> hoặc <lời nói, mở file>.
- *Mức* 3: Các sự kiện sẽ được gửi đến các bộ điều khiển hộp thoại. Chúng tương ứng để hoàn thành lệnh, độc lập của các phương thức sử dụng, ví dụ <mở file>.

2.6. Sinh dữ liệu thử tại mức điều khiển hộp thoại

Để tạo ra dữ liệu thử nghiệm ở cấp độ điều khiển hộp thoại, Laya Madani và các đồng nghiệp [20] đã sử dụng cây nhiệm vụ để mô hình hóa hành vi ứng dụng và người dùng. Các tác giả đã tiến hành chuyển đổi cây CTT thành một máy trạng thái hữu hạn xác suất vào ra. Dữ liệu kiểm thử đã được tạo ra một cách tự động từ máy này. Để thực hiện được việc chuyển đổi, với một nhiệm vụ T bất kỳ trên cây CTT, các tác giả đã định nghĩa một máy trạng thái hữu hạn vào/ra:

$M_T = (Q_T, qi_T, q_{fT}, I_T, O_T, trans_T, P_T)$. Trong đó:

- Q_T là một tập trạng thái.
- qi_T là trạng thái ban đầu, trạng thái mà nhiệm vụ T bắt đầu.
- qf_T là trạng thái cuối cùng, đây là trạng thái mà nhiệm vụ T kết thúc.
- I_T là một tập các đầu vào cho nhiệm vụ T.
- O_T là một tập các đầu ra áp dụng bởi nhiệm vụ T.
- $trans_T \subseteq Q_T \times (I_T \cup \{\mu\}) \times O_T \times Q_T$ là tập các chuyển tiếp tương ứng với nhiệm vụ T. Đầu vào μ là một đầu vào rỗng (không có tác động của người dùng). Nếu $(q_T, a, b, s_T) \in trans_T$, thì thể viết $q_T \xrightarrow{a/b} s_T$.
- Đôi khi, dữ liệu đầu vào và ra của sự chuyển đổi được bỏ qua: $q_T \xrightarrow{c} s_T$ với $c = a / b$.
- P_T là hàm xác suất: P_T: $trans_T \rightarrow [0 .. 1]$. Hàm P_T có các tính chất: Đối với mọi trạng thái, tổng các xác suất của quá trình chuyển đổi trạng thái này là bằng 1:

$$\forall q \in Q_T \setminus \{qf_T\}: \Sigma_{c,q'} P_T (q \xrightarrow{c} q') = 1.$$

2.7. Tính đến xác suất có điều kiện

Cây CTT mở rộng giúp tự động hóa sinh dữ liệu thử theo xác suất. Tuy nhiên, CTT mở rộng chưa được định nghĩa để sử dụng các điều kiện, tức là một xác suất xảy ra được gán với một sự kiện theo một điều kiện. Điều này làm khó khăn cho các nhà thiết kế trong việc mô tả đầy đủ các điều kiện của hành vi người dùng trên các IMA. Do đó, CTT cần được giải quyết theo hướng mở rộng thêm điều kiện để sử dụng các hồ sơ hoạt động. Chúng tôi đã đề xuất gán xác suất có điều kiện cho các phép toán trên CTT và được trình bày tổng hợp như trong Bảng 2.1.

Bảng 2.1. Phép toán CTT được bổ sung xác suất có điều kiện

Phép toán CTT	Phép toán CTT với xác suất có điều kiện
$T_1[]T_2$	$T_1\ []_{(PA1,PB1\|cond1),(PA2,\ PB2\|cond2),\ldots,\ (PAn,\ PBn\|condn),(PA0,PB0)}\ T_2$
$T_1\|\|\|T_2$	$T_1\ \|\|\|_{(PA1,PB1\|cond1),(PA2,\ PB2\|cond2),\ldots,\ (PAn,\ PBn\|condn),(PA0,PB0)}\ T_2$
$T_1\|[]\|T_2$	$T_2\ \|[]\|_{(PA1,PB1\|cond1),(PA2,\ PB2\|cond2),\ldots,\ (PAn,\ PBn\|condn),(PA0,PB0)}\ T_2$
$T_1\|=\|T_2$	$T_2\ \|=\|_{(PA1,PB1\|cond1),(PA2,\ PB2\|cond2),\ldots,\ (PAn,\ PBn\|condn),(PA0,PB0)}\ T_2$
$T_1[>T_2$	$T_1\ [>_{(PA1\|cond1),(\ PA2\|cond2),\ldots,\ (PAn\|condn),(\ PA0)}\ T_2$
$T_1\|>T_2$	$T_1\ \|>_{(PA1\|cond1),(\ PA2\|cond2),\ldots,\ (PAn\|condn),(\ PA0)}\ T_2$
$[T]$	$[T]_{(PA1\|cond1),(\ PA2\|cond2),\ldots,\ (PAn\|condn),(\ PA0)}$

2.8. Đánh giá kết quả mở rộng cây CTT với xác suất có điều kiện

Hồ sơ hoạt động cung cấp thông tin về việc sử dụng hiệu quả của một ứng dụng. Trong [4], Laya Madani và các đồng nghiệp đề nghị bổ sung xác suất vào các phép toán trên cây CTT để sinh dữ liệu thử theo hồ sơ hoạt động. Tuy nhiên, một phần mở rộng như vậy là không đầy đủ và hợp lý với một số trạng thái của ứng dụng.

Ví dụ, ở cột 2 của Bảng 2.2, các dữ liệu thử được tạo ra với xác suất cho các ứng dụng Memo, các hành động xóa ghi chú (remove) được tạo ra nhưng không có ghi chú nào (tại thời điểm 5), vì trong thời điểm 2, người dùng có được một ghi chú nhưng tại thời điểm 4,

người dùng đã xóa ghi chú này. Vì vậy, không hợp lý nếu chỉ mở rộng CTT với xác suất không điều kiện.

Bảng 2.2. Dữ liệu thử được sinh với xác suất và xác suất có điều kiện

Time	Dữ liệu thử được sinh với xác suất không điều kiện	Dữ liệu thử được sinh với xác suất có điều kiện
1	Move	Move
2	Get	Get
3	Move	Move
4	Remove	Remove
5	**Remove**	**Get**
6	Remove	Remove

Trong cột 3 của Bảng 2.2, các dữ liệu thử được tạo ra với xác suất có điều kiện, hành động "get" được tạo ra thay vì hành động "remove" vào thời điểm đó là 5. Điều đó là hợp lý, vì vậy CTT cần được mở rộng với các xác suất có điều kiện.

2.9. Kết luận

Trong chương này, cây nhiệm vụ được mở rộng với các xác suất để có thể đặc tả các kịch bản tương tác khác nhau. Sau đó, các phép toán trên cây CTT chuyển thành một FSM xác suất để mô hình hóa các hành vi người dùng. Như một sự cải tiến bổ sung cho công tác nghiên cứu trước đây, trong luận án này, chúng tôi đã mở rộng thêm các phép toán của cây CTT với các xác suất có điều kiện.

Chương 3. TTT: NGÔN NGỮ MÔ HÌNH HÓA KIỂM THỬ MỚI ĐẶC TẢ KIỂM THỬ CÁC ỨNG DỤNG ĐA PHƯƠNG THỨC TƯƠNG TÁC

3.1. Giới thiệu

Các phương pháp kiểm thử IMA nêu trên đã sử dụng nhiều mô hình khác nhau để xây dựng các mô hình kiểm thử: *mô hình hành vi* của ứng dụng được biểu diễn qua mô hình cây nhiệm vụ, *hồ sơ hoạt động* đã được chú thích trên cây nhiệm vụ, *máy trạng thái hữu hạn*

biểu diễn các trạng thái và chuyển tiếp của ứng dụng. Nhiều mô hình làm cho việc mô hình hóa các thuộc tính và các quá trình trở nên khó khăn. Vì vậy, chúng tôi đã đề xuất ngôn ngữ mô hình hóa kiểm thử TTT để biểu diễn nhất quán các thuộc tính và các quá trình này.

Mô hình được viết bởi ngôn ngữ TTT có thể biểu diễn các yếu tố sau trong một cú pháp duy nhất và nhất quán:

- Các kịch bản cho các ứng đa phương thức tương tác.
- Các xác suất có điều kiện được gán cho các nhiệm vụ.
- Các "dấu vết" của các hành động người dùng và các hàm truy vấn trên các dấu vết.
- Tự động hóa sinh các sự kiện đa phương thức.
- Kiểm tra tính đúng đắn của các thuộc tính CARE.

3.2. Dấu vết của hành vi người dùng

Các dấu vết của các hoạt động người dùng là tập dữ liêu của các hành động cụ thể của người dùng trong quá trình tương tác của mình với ứng dụng. Nó bao gồm các thuộc tính như thời gian và các hành động như lời nói, cử chỉ, ánh mắt, chuột, bàn phím... Những dấu vết được thiết kế như một bảng dữ liệu có thể tạo ra bằng ngôn ngữ TTT. Các dữ liệu hành vi người dùng cũng có thể chèn, cập nhật, xóa và truy vấn dữ liệu từ bảng. Bảng này được tạo trong thời gian người dùng bắt đầu tương tác với ứng dụng, và nó được hủy khi việc kiểm thử kết thúc.

Các mục đích của việc lưu trữ các dấu vết của các hoạt động sử dụng là: (1) để xác định trạng thái của mô hình, (2) tính các hành động dùng để xây dựng các điều kiện xác định các hành động tiếp theo của người dùng, (3) để tạo ra hàm chỉ đọc để tính toán xác suất có điều kiện, và (4) để xây dựng một bộ phán xét kiểm thử nhằm đánh giá tính hợp lý của các thuộc tính CARE.

3.3. Định nghĩa của ngôn ngữ TTT

Chúng tôi sử dụng một văn phạm phi ngữ cảnh đặc tả ngôn ngữ TTT. Một văn phạm phi ngữ cảnh có thể được xác định hình thức bởi một bộ bốn *(N, Σ, P, S):*

- *N* là một tập hợp hữu hạn các ký hiệu không kết thúc;

- Σ là một tập hữu hạn các kí hiệu két thúc;
- S là ký hiệu bắt đầu
- P là một tập hữu hạn các luật sản xuất với hình thức $v \rightarrow w$, trong đó v là ký hiệu không kết thúc, $v \in N$, $w \in (\Sigma \cup N)^*$.

Các biểu thức đơn giản trong ngôn ngữ TTT có thể được mô tả bởi các biểu thức chính quy, nhưng các lệnh phức tạp phải được mô tả bởi văn phạm phi ngữ cảnh.

3.4. Cấu trúc cơ bản của ngôn ngữ TTT

Trong Bảng 3.1, chúng tôi định nghĩa cú pháp và ngữ nghĩa của ngôn ngữ TTT bởi văn phạm phi ngữ cảnh.

Bảng 3.1. Cú pháp của ngôn ngữ TTT

```
1. <tttmodel> ::= <function> ;
2. <function> ::= <function> <statement> ;
3. <statement> ::= ';'
            | <expr> ';'
            | TESTCTT VAR ';'
            | output '(' VAR ')' ';'
            | input '(' VAR ')' ';'
            | VAR '=' <expr> ';
            | begin <statement_list> end;
4. <statement_list> ::= <statement>
                        | <statement_list>
<statement>
5. <expr> ::=  <expr> '+' <expr>
            | <expr> '-' <expr>
            | <expr> '*' <expr>
            | <expr> '/' <expr>
            | <expr> '<' <expr>
            | <expr> '>' <expr>
            | <expr> GE <expr>
            | <expr> LE <expr>
            | <expr> EQ <expr>
            | <expr> AND <expr>
            | <expr> OR <expr>
            | NOT <expr>
            |'(' <expr> ')'
```

Cấu trúc cơ bản của ngôn ngữ TTT bao gồm nhiều khối *TESTCTT* và một hoặc nhiều *function*. *TESTCTT* được định nghĩa bởi tập các mệnh đề và hình thức của *TESTCTT* được trình bày trong Bảng 3.2.

Bảng 3.2. Cấu trúc của khối *TESTCTT*

```
TESTCTT name;
sets
    basic_types;
var
    local_variables;
init
    initial_state;
begin
invar(i1,i2,...,in);
ctt_operators;
sql_statements;
end;
```

Cấu trúc của một hàm được trình bày trong Bảng 3.3.

Bảng 3.3. Cấu trúc của hàm FUNCTION.

```
function name(<inputs>)returns(<outputs>);
var
    local_ variables;
begin
    statements;
end;
```

Các *function* có thể không có hoặc có nhiều tham số đầu vào và có một tham số đầu ra. Một hàm bao gồm ít nhất một lệnh. Các tham số được đưa vào bởi các <*inputs*>, các tham số cách nhau bằng dấu phẩy, mỗi tham số trong danh sách có một kiểu dữ liệu theo sau bởi một định danh: <*variable-identifier*>.

3.5. Đặc tả các phép toán CTT có bổ sung xác xuất có điều kiện

Nhằm đặc tả các phép toán CTT có bổ sung xác suất có điều kiện, chúng tôi định nghĩa các phép toán có ngữ nghĩa phù hợp và giúp tự động sinh dữ liệu thử cho IMA.

Table 3.4. The CTT Syntax

```
<ctt_operator> ::= <choice>|<concur>|<indepen>|<deact>|
                   <suspend>|<option>|<enabling>
<choice> ::= choice(<expression>,<expression>,<expression>,
```

```
        <expression>, <expression>,<expression>, <expression>)
<concur> ::= concur(<expression>,<expression>,<expression>,
         <expression>, <expression>,<expression>, <expression>)
<indepen> ::= indepen(<expression>, <expression>, <expression>,
         <expression>, <expression>,<expression>, <expression>)
<deact>::=   deact(<expression>,<expression>,<expression>)
<suspend>::= suspend(<expression>,<expression>,<expression>)
<option>::=  option(<expression>,<expression>,<expression>)
<enabling>::=enabling(<expression>, <expression>)
<expression> ::= <assignment_expression>|<expression>
<assignment_expression> ::= <conditional_expression>
                      |<expression> = <assignment_expression>
<conditional_expression> ::= <logical_or_expression> |
                               <logical_and_expression>
<logical_or_expression> ::=
         logical_or_expression> or <logical_or_expression>
<logical_and_expression> ::=
   |<logical_ and_expression> and <inclusive_and_expression>
```

3.6. Hỗ trợ định nghĩa trạng thái

Trạng thái của mô hình liên quan đến đầu vào lần trước và đầu ra của mô hình. Để ngôn ngữ TTT định nghĩa được trạng thái, Chúng tôi phải lưu dữ liệu về hành vi người dùng. Chúng tôi kế thừa và rút gọn các lệnh SQL để lưu trữ và truy vấn dữ liệu này. Trong Bảng 3.5, chúng tôi đặc tả các lệnh này bằng văn phạm phi ngữ cảnh.

Table 3.5. Cú pháp các lênh tựa SQL

```
   <sql_statement> ::= <create_table> | <alter_table> |
   <drop_table> | <insert> | <delete> | <update>| <select>
   <create_table>::= create table <name> (<variable>);
   <drop_table>::= drop table <name>;
   <insert>::= insert into
   <name>(<variable>)values(<variable>)
   <update>::=update <name> set <expression_list>
              where <expression_list>
   <delete>::=delete from <name> [where <expression_list>]?
```

```
<select>::=select <name> from
<name>[where<expression_list>]?
<expression_list>::=<expression>|<expression_list>,
            <expression>
```

3.7. Hỗ trợ đa phương thức

Trong một IMA, các phương thức có thể được sử dụng độc lập hoặc kết hợp (sự hợp nhất của phương thức). Việc sử dụng kết hợp các phương thức bị giới hạn bởi những ràng buộc thời gian. Các phương thức có thể được thực hiện tuần tự hoặc đồng thời [5] trong một cửa sổ thời gian *TW*.

Các phương thức của một tập *M* được sử dụng đồng thời nếu chúng được sử dụng cùng khoảng thời gian. Các phương thức của một tập *M* được sử dụng tuần tự trong một cửa sổ thời gian *TW*, nếu có nhiều nhất một phương thức hoạt động ở một thời điểm. Nếu tất cả các phương thức trong tập *M* này được sử dụng trong cửa sổ thời gian *TW*, thì tính đồng thời và trình tự thể hiện một ràng buộc trên không gian tương tác.

Trong luận án, chúng tôi xây dựng được các thuật toán: *Sinh các sự kiện đa phương thức và kiểm tra các thuộc tính CARE: thuộc tính Equivalence, thuộc tính Redundancy-Equivalence và thuộc tính Complementarity.*

3.8. Các qui tắc chuyển đổi từ CTT sang mô hình TESTCTT

Hành vi của ứng dụng tương tác thường được mô tả bằng cách sử dụng CTT. Vì vậy, chúng tôi đề xuất các quy tắc chuyển đổi sau đây để ánh xạ các nhiệm vụ từ CTT sang các lệnh của ngôn ngữ TTT. Chúng tôi tóm tắt các quy tắc chuyển đổi trong Bảng 3.6

Bảng 3.6. Các qui tắc chuyển đổi từ CTT sang TTT

Qui tắc	CTT	Mô hình TestCTT
1	Nhiệm vụ tương tác	Đầu ra
2	Nhiệm vụ ứng dụng	Đầu vào
3	Nhiệm vụ trừu tượng	Các biến trạng thái
4	Các phép toán CTT	Các hàm

3.9. Mô hình hóa ứng dụng đa phương thức tương tác Memo

Để giải thích rõ hơn cách xây dựng mô hình kiểm thử bằng ngôn ngữ TTT, chúng tôi xây dựng một mô hình kiểm thử cho ứng dụng đa phương thức tương tác Memo. Mô hình TESTCTT cho ứng dụng Memo được xây dựng thông qua bốn bước:

Bước 1. Chọn một mục tiêu kiểm thử cho các ứng dụng Memo

Bước 2. Xác định ký hiệu cho các hoạt động trong mô hình

Bước 3. Xác định các biến trạng thái và các loại dữ liệu

Bước 4. Viết cho từng hoạt động kiểm thử

Bảng 3.7. Mô hình TESTCTT cho ứng dụng Memo

```
1. TESTCTT Memo;
2. VAR
3. q0, q1, q2, q3 : bool;
4. T, Tout: char;
5. tw : integer;
6. begin
7. INIT (Tout='D')
8. do
9. begin
10.     q0=(Tout<>'D' and Tout <> 'C' and T <>'o');
11.     q1=(T=='o')or(Tout=='G'and T =='g') or
12.        (Tout=='R'and T =='r')or(Tout=='S'and T =='s');
13.     q2 = (Tout=='D');
14.     q3 = (Tout=='C');
15.  if (q0)
16.  begin
17.      T = Choice('o','',0.5, note_nb()=0,1,note_nb()>=5,0.1);
18.      insert into u_actions(input) values(T);
19.  end
20.  if (q1)
21.  begin
22.     T= Choice(('o' ,'',0.5, note_nb()=0,1,note_nb()>=5,0.1);
23.     insert into u_actions(input) values(T);
24.  end
25.  if (q2)
26.    begin
```

```
27.    T= choice('g','r',0.8,note_nb()=0,1,
28.       note_nb()>=5,0.1);
29.       If T =’g’
30.         begin
31.          tw=1;
32.          do
33.          begin
34.           Modalities(Speech_get,Mouse_get,0.3,0.7,
35.                note_nb()=0 ,0,0, note_nb()>=5,0.2,0.8);
36.           Tout = call_Memo(T);
37.           Insert into u_actions(M1,M2,input,ouput)
38.                values(Speech_get, Mouse_get,T,Tout);
39.           Tw =tw+1;
40.           end
41.           while (tw <=3)
42.        TestRedundantEquivalenceEarly(Speech,Mouse,get,3)
43.             end
44.           else
45.       begin
46.           Tw=1;
47.           do
48.           begin
49.           Modalities(Speech_remove,Mouse_remove,0.8,0.9,
50.                note_nb()=0,0,0, note_nb()>=5,0.9,0.7);
51.           Tout = call_Memo(T);
52.           insert into u_actions(M1,M2, input,output)
                 values(Speech_remove, Mouse_remove,T,Tout);
53.           Tw=tw+1;
54.           end
55.              while (tw<=3);
56.            TestEquivalence (Speech, Mouse, get,3)
57.            TestRedundantEquivalenceEarly
58.                              (Speech, Mouse,remove,3);
59.          TestComplementaryEarly(Mouse,Speech,remove, 3);
60.         end;
61.     while (T<>’E’);
62.     end
```

```
63. function note_nb() returns (note_nb: int);
64. Var get_nb, remove_nb :int;
65. begin
66. get_nb= select count(*) from u_actions where input =”g”;
67. remove_nb=select count(*)from u_actions where input =”r”;
68. note_nb=get_nb –remove_nb
69. end
```

Trong Bảng 3.7, chúng tôi định nghĩa các biến trạng thái q_i (dòng 10,11,12,13,14) dựa trên các hành động đầu vào, đầu ra và trạng thái của ứng dụng. Dòng 10 có nghĩa là: nếu người dùng đang di chuyển (*T='o'*) hoặc ứng dụng tương tác đang xuất hiện một ghi chú mà người dùng đã nhận ghi chú này (*Tout=='G' and T =='g'*) hoặc ứng dụng tương tác đang xuất hiện một ghi chú mà người dùng đã xóa ghi chú này (*Tout=='R' and T =='r'*) hoặc ứng dụng tương tác đang mang một ghi chú mà người dùng đã cài đặt ghi chú này (*Tout=='S' and T =='s'*) thì mô hình đang ở trạng thái *q1*.

Nếu mô hình hiện tại đang ở trạng thái nào (dòng 15,21,27) thì sinh dữ liệu *T* tương ứng với trạng thái đó và lưu dữ liệu vào bảng *u_actions*. Tiếp đến, mô hình gửi dữ liệu *T* sang ứng dụng tương tác Memo (dòng 53) và nhận kết quả trả về *Tout* của ứng dụng này.

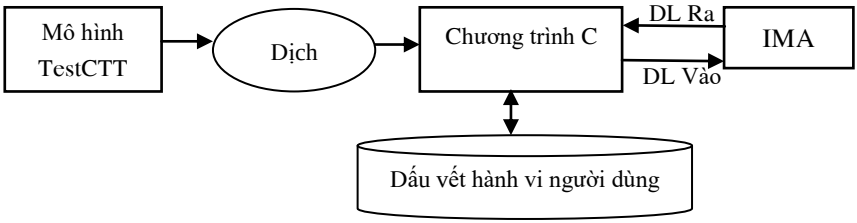
Chương 4. TTTEST: CÔNG CỤ HỖ TRỢ KIỂM THỬ CÁC ỨNG DỤNG ĐA PHƯƠNG THỨC TƯƠNG TÁC

4.1. Giới thiệu

Trong chương này, luận án trình bày công cụ TTTEST hỗ trợ sinh dữ liệu kiểm thử cho các sự kiện đa phương thức và kiểm tra tính hợp lý của các thuộc tính CARE của IMA.

4.2. Môi trường thực thi kiểm thử

Để kiểm thử các IMA, chúng tôi xây dựng một môi trường kiểm thử TTTEST (TTT-based Test), trong Hình 4.1.



Hình 4.1. Môi trường kiểm thử TTTEST.

Môi trường kiểm thử *TTTEST* gồm bốn thành phần cơ bản: mô hình *TESTCTT* được đặc tả bằng ngôn ngữ TTT, chương trình C được biên dịch từ mô hình *TESTCTT,* ứng dụng tương tác cần kiểm thử, bộ lưu dấu vết hành động của người dùng. Hoạt động của môi trường *TTTEST* được mô tả như sau:

Tại Bước 0, mô hình TESTCTT được đặc tả bằng ngôn ngữ TTT và được biên dịch thành chương trình C. Bước 1, chương trình C tạo dữ liệu đầu ra X từ một trạng thái nội bộ, các trạng thái này được định nghĩa dựa trên dữ liệu đầu vào, đầu ra và trạng thái hiện tại của ứng dụng. Bước 2, chương trình C chuyển đầu ra *X* cho các ứng dụng tương tác. Dữ liệu *X* thực chất là các thao tác của người dùng đối với ứng dụng. Vì vậy, *X* là các dữ liệu kiểm thử đối với ứng dụng tương tác. Bước 3, ứng dụng tương tác xử lý thao tác *X* và cho kết quả đầu ra *Y*. Bước 4, chương trình C nhận *Y* làm dữ liệu đầu vào, cập nhật biến trạng thái nội bộ của mô hình và tiếp tục quay lại Bước 1. Toàn bộ các dữ liệu đầu vào, đầu ra đều được lưu vào bộ lưu dấu vết hành động người dùng. Các điều kiện sinh dữ liệu thử từ chương trình C được tính toán từ bộ lưu dấu vết này.

4.3. Công cụ TTTEST

4.4. Chuyển đổi mô hình TESTCTT sang chương trình C

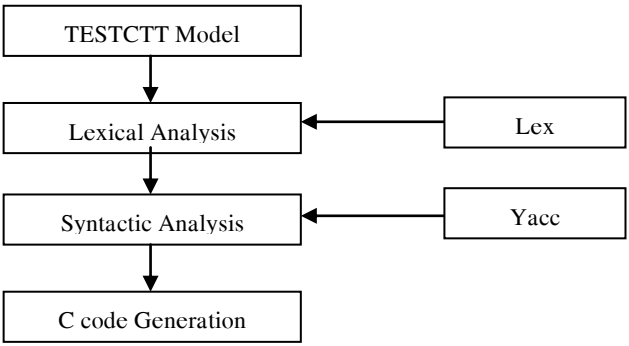
4.4.1. Các vấn đề chuyển đổi

Dịch từ ngôn ngữ TTT sang ngôn ngữ C liên quan đến nhiều vấn đề khác nhau. Vấn đề đầu tiên là phân tích từ vựng. Vấn đề thứ hai phân tích cú pháp. Một số cú pháp TTT có cấu trúc tương đương trong C, nhưng với một thứ tự từ khóa khác nhau. Cấu trúc ở mức độ

trừu tượng cao các phép toán CTT phải được chuyển đổi thành các cấu trúc với các mức cụ thể hơn trong ngôn ngữ C. Cuối cùng, các câu lệnh tựa SQL từ ngôn ngữ TTT được chuyển vào chương trình C.

4.4.2. Giải pháp tự động chuyển đổi

Để chuyển đổi tự động từ mô hình TESTCTT sang ngôn ngữ C, chúng tôi sử dụng hai công cụ Lex và Yacc. Quy trình tự động chuyển đổi từ mô hình TESTCTT sang chương trình C được trình bày trong Hình 4.2.



Hình 4.2. Chuyển đổi mô hình TESTCTT sang chương trình C.

4.5. Các thử nghiệm

Để giải thích rõ hơn cách tự động hóa kiểm thử các IMA bằng ngôn ngữ TTT, chúng tôi xét ba thử nghiệm về mô hình hóa và kiểm thử các ứng dụng tương tác: NotePad, Memo và MapNavigator. NotePad là một ứng dụng tương tác, còn Memo và Map Navigator là những ứng dụng đa phương thức tương tác.

4.6. Đánh giá kết quả các ca kiểm thử

Mục đích của các thử nghiệm sau đây là để so sánh và đánh giá thời gian kiểm thử ứng dụng Memo bằng các cách thực hiện thủ công, viết một chương trình C và viết một mô hình bằng ngôn ngữ TTT. Chúng tôi so sánh thời gian của ba phương pháp theo cả về hai vấn đề khác nhau khi kiểm thử ứng dụng Memo: (1) tạo ra các dữ liệu thử

cho các sự kiện đa phương thức, và (2) kiểm tra tính hợp lý của các thuộc tính CARE.

Để sinh các sự kiện đa phương thức, cả ba phương pháp đều dùng một bộ dữ liệu thử. Các hành động của người dùng như *move, get, set, remove* được xây dựng lặp đi lặp lại 100 lần và gửi đến ứng dụng Memo. Để kiểm tra tính hợp lý của các thuộc tính CARE, hành động của người dùng như *move, get, set, remove* được thực hiện liên tục trong một cửa sổ thời gian *TW* của ứng dụng Memo.

Chúng tôi đã tiến hành ba thử nghiệm như sau:

Thử nghiệm 1: Chúng tôi kiểm thử ứng dụng Memo bằng thủ công. Chúng tôi thu thập thời gian hoàn thành của việc kiểm thử này. Bảng 4.1 là kết quả của thử nghiệm 1.

Thử nghiệm 2: Chúng tôi kiểm thử ứng dụng Memo bằng cách viết chương trình C. Chương trình C tạo ra dữ liệu thử và kiểm tra tự động tính hợp lý của các thuộc tính của CARE. Chúng tôi thu thập được thời gian hoàn thành của thử nghiệm. Bảng 4.2 là kết quả của thử nghiệm 2.

Thử nghiệm 3: Chúng tôi kiểm thử các ứng dụng Memo bằng cách viết các mô hình bằng ngôn ngữ TTT. Mô hình được dịch sang chương trình C, chương trinh được thực thi tạo ra dữ liệu thử và kiểm tra tự động tính hợp lý của các thuộc tính CARE. Chúng tôi thu thập được thời gian hoàn thành của thử nghiệm. Bảng 4.3 là kết quả của thử nghiệm 3.

Table 4.1. Kết quả của thử nghiệm 1

Thời gian hoàn thành (phút)			
Sinh dữ liệu thử	Kiểm tra tính hợp lý của các thuộc tính CARE	Báo cáo	Tổng cộng
30	123	35	188

Table 4.2. Kết quả thử nghiệm 2

Thời gian hoàn thành (phút)		
Viết chương trình C	Chạy chương trình C	Tổng cộng
545	2	547

Table 4.3. Kết quả thử nghiệm 3

Thời gian hoàn thành (phút)			
Viết mô hình	Chuyển sang	Chạy chương	Tổng cộng

TestCTT	chương trình C	trình C	
173	2	3	178

Trong Bảng 4.1, thời gian hoàn thành của thử nghiệm 1 là 188 phút, trong khi trong Bảng 4.2, thời gian hoàn thành của thử nghiệm 2 là 547 phút. Trong bảng 4.3, thời gian hoàn thành của các thử nghiệm 3 là 178 phút. Tóm lại, kết quả của ba thử nghiệm xác nhận rằng ngôn ngữ TTT có thể giúp những kiểm thử viên kiểm thử ứng dụng Memo nhanh hơn.

Mặc dù thử nghiệm với ngôn ngữ TTT dường như chỉ hơi nhanh hơn so với kiểm tra thủ công, nhưng việc sử dụng ngôn ngữ TTT có thể kiểm thử nhiều lần như mong muốn (khi gỡ lỗi các ứng dụng) mà không cần nỗ lực bổ sung trong khi kiểm thử thủ công đòi hỏi phải kiểm thử lại một lần nữa.

Tất nhiên, để đánh giá tốt hơn, thử nghiệm này nên được lặp lại với những người dùng khác. Trong tương lai, chúng tôi cần thêm nhiều thử nghiệm để so sánh thời gian kiểm thử các IMA bằng thủ công, bằng cách viết chương trình C và viết các mô hình bằng ngôn ngữ TTT.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết quả của luận án

Các ứng dụng đa phương thức tương tác đã đáp ứng tốt phần nào về sự tiện lợi, tính hữu dụng đối với người dùng và trở thành một phần mềm ứng dụng phổ biến quan trọng. Tuy nhiên các ứng dụng đa phương thức tương tác phát triển ngày càng nhanh cả về số lượng lẫn kích thước và độ phức tạp. Do đó, việc đảm bảo tính đúng đắn cho các ứng dụng đa phương thức tương tác là cần thiết vì vậy tác giả đã nghiên cứu và đề xuất phương pháp, cùng với công cụ hỗ trợ cho kiểm thử tự động cho các ứng dụng này. Tác giả đã có một số đóng góp về mặt lý thuyết cũng như thực nghiệm trong lĩnh vực kiểm thử các ứng dụng đa phương thức tương tác :

– Dựa vào việc phân tích các đặc điểm của cây nhiệm vụ tương tranh CTT (ConcurTaskTrees), một mô hình phổ biến đặc tả các IMA, chúng tôi đã mở rộng CTT với các xác suất có điều kiện. Chính xác hơn, hành vi của người dùng trên IMA thường bị ảnh hưởng bởi các điều kiện của ứng dụng và môi trường làm việc.

– Một ngôn ngữ đặc tả kiểm thử mới cho các IMA được đề xuất, gọi là ngôn ngữ TTT (Task Tree based Test). Ngôn ngữ TTT đặc tả các kịch bản kiểm thử và hồ sơ hoạt động có điều kiện của các IMA.

– Xác định các luật chuyển đổi từ CTT sang mô hình kiểm thử của ngôn ngữ TTT.

– Xây dựng giải pháp sinh dữ liệu kiểm thử tự động cho các ứng dụng tương tác.

– Đặc tả các sự kiện đa phương thức tương tác và các thuộc tính CARE. Sự đặc tả này bao gồm hai vấn đề quan trọng trong kiểm thử ứng dụng đa phương thức tương tác: Sinh dữ liệu kiểm thử đa phương thức và kiểm tra tính hợp lý của các thuộc tính CARE trên các ứng dụng đa phương thức tương tác.

– Công cụ TTTEST được phát triển để tự động kiểm thử các ứng dụng đa phương thức tương tác.

Các kết quả trên đã được trình bày tại các hội nghị chuyên ngành (quốc gia và quốc tế) và các tạp chí khoa học trong nước và quốc tế, thể hiện độ tin cậy của kết quả.

Hướng phát triển

– Phương pháp kiểm thử dựa vào ngôn ngữ TTT cần nhiều thời gian để xây dựng các mô hình cho các IMA. Vì vậy, chúng tôi hướng tới việc tự động sinh các mô hình bằng các kỹ thuật khám phá tự động IMA.

– Chúng tôi sẽ nghiên cứu và cải tiến các thuật toán sinh dữ liệu tự động và kiểm thử tính hợp lý của các thuộc tính CARE.

– Cải tiến công cụ TTTEST để trong tương lai sinh các dữ liệu kiểm thử tự động cho các thuộc tính an toàn của các ứng dụng đa phương thức tương tác.

CÁC CÔNG TRÌNH ĐÃ CÔNG BỐ

1. Le Thanh Long, Nguyen Thanh Binh, Ioannis Parissis, "A New Test Modeling Language for Interactive Applications Based on Task Trees", In Proceedings of the 4th International Symposium on Information and Communication Technology (SoICT 2013), ACM Publisher, ISBN: 978-1-5403-2454-0, pp.285-293.
2. Le Thanh Long, Nguyen Thanh Binh, Ioannis Parissis, "TTT: A Test Modeling Language for Interactive Applications Based on Task Trees", in Proceedings of 16th National Conference: Selected Problems about IT and Telecommunication (@ 2013), ISBN: 978-604-67-0251-1, pp.333-338.
3. Le Thanh Long, Nguyen Thanh Binh, Ioannis Parissis, "A solution of generate test data for interactive applications", In Proceedings of the 7th National Conference on Fundamental and Applied Information Technology Research (FAIR 2014), ISBN: 978-604-913-300-8, pp 134-143.
4. Le Thanh Long, Nguyen Thanh Binh, Ioannis Parissis, "Model-to-C program translation in TTTEST", in Proceedings of 17th National Conference: Selected Problems about IT And Telecommunication, Ho Chi Minh city (@ 2015), 05-06/11/2015, ISBN: 978-604-67-0645-8, pp 142-149.
5. Le Thanh Long, Nguyen Thanh Binh, Ioannis Parissis, "Testing Multimodal Interactive Applications By Means of The TTT Language", Domain Specific Model-Based Approaches To Verification And Validation - Amaretto 2016, In conjunction with the 4th International Conference on Model-Driven Engineering and Software Development - MODELSWARD 2016, 19 February, 2016 - Rome, Italy, ISBN: 978-989-758-166-3, pp 23-32.
6. Le Thanh Long, Nguyen Thanh Binh, Ioannis Parissis, "TTTEST : The Tool Support For Testing Interactive Multimodal Applications", In Proceedings of the International Conference on Electronic, Information and Communication (ICEIC 2016). 27-30/01.2016, IEEE Publisher, pp 78-81.
7. Le Thanh Long, Nguyen Thanh Binh, Ioannis Parissis, "Automatic Testing of Interactive Multimodal Applications", in Communications in Computer and Information Science (Scopus indexed), 2017, pp 93-113