ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN THU TRANG

ĐỊNH VỊ VÀ SỬA LỖI TỰ ĐỘNG CHO CÁC HỆ THỐNG PHẦN MỀM DÒNG SẢN PHẨM

Luận án Tiến sĩ

Chuyên ngành: Kỹ thuật Phần mềm

Hà Nội - 2024

Tóm tắt

Hệ thống phần mềm dòng sản phẩm (Software Product Lines - SPL) đang trở nên phổ biến và được sử dụng rộng rãi để phát triển các dự án lớn trong công nghiệp. Tuy nhiên, các đặc tính biến đổi vốn có của chúng đặt ra những thách thức lớn trong việc đảm bảo chất lượng của các hệ thống này. Mặc dù việc gỡ lỗi tự động cho hệ thống phần mềm đơn đã được nghiên cứu chuyên sâu, nhưng việc gỡ lỗi hệ thống SPL hầu như vẫn chưa có các nghiên cứu chuyên sâu. Trong thực tế, các hoạt động gỡ lỗi trong hệ thống SPL thường được thực hiện thủ công. Luận án đề xuất các giải pháp hỗ trợ gỡ lỗi tự đông cho các hệ thống SPL bằng cách tập trung vào ba nhiệm vụ cơ bản, bao gồm phát hiện *sản phẩm đúng giả* (false-passing products), định vị lỗi biến đổi và sửa chữa lỗi biến đổi.

Đầu tiên, *Luận án cải thiện độ tin cậy của kết quả kiểm thử bằng cách phát hiện các sản phẩm đúng giả trong hệ thống SPL.* Với một tập hợp các sản phẩm đã được kiểm thử của hệ thống SPL, cách tiếp cận của Luận án, Clap, là thu thập các dấu hiệu lỗi ở các sản phẩm thất bại dựa trên việc cài đặt và chất lượng bộ kiểm thử của chúng. Đối với một sản phẩm, Clap đánh giá các dấu hiệu này và các dấu hiệu càng mạnh thì sản phẩm đó càng có nhiều khả năng là *sản phẩm đúng giả*. Cụ thể, khả năng một sản phẩm là *sản phẩm đúng giả* được đánh giá dựa trên việc liệu nó có một số lượng lớn các câu lệnh khả nghi, đó là các câu lệnh đã gây ra lỗi cho các sản phẩm thất bại và liệu bộ các ca kiểm thử của nó có chất lượng thấp hơn so với bộ các ca kiểm thử của các sản phẩm bị lỗi hay không.

Thứ hai, *Luận án đề xuất* VarCop*, một phương pháp định vị lỗi biến đổi mới và hiệu quả cho hệ thống SPL.* Đối với hệ thống SPL bị lỗi do lỗi biến đổi, VarCop sẽ khoanh vùng các câu lệnh khả nghi bằng cách phân tích kết quả kiểm thử của các sản phẩm được lấy mẫu và mã nguồn của chúng. Các câu lệnh khả nghi là các câu lệnh liên quan đến sự tương tác giữa các tính năng cần thiết để các lỗi trong hệ thống có thể được thể hiện ra. Trong VarCop, mức độ khả nghi của từng câu lệnh riêng biệt được đánh giá dựa trên cả kết quả kiểm thử tổng thể của các sản phẩm chứa câu lệnh cũng như kết quả chi tiết của từng ca kiểm thử được thực thi bởi câu lệnh trong các sản phẩm này.

Thứ ba, *Luận án đề xuất hai cách tiếp cận hướng sản phẩm và hướng hệ thống để sửa các lỗi biến đổi trong hệ thống SPL nhằm khắc phục lỗi của các sản phẩm thất bại và không phá vỡ các hành vi đúng của các sản phẩm thành công.* Đối với phương pháp *hướng sản phẩm*, mỗi sản phẩm bị lỗi sẽ được sửa riêng lẻ và các bản vá thu được sau đó sẽ được áp dụng và kiểm chứng trên các sản phẩm khác của hệ thống. Đối với phương pháp *hướng hệ thống*, tất cả các sản phẩm đều được sửa chữa đồng thời. Các bản vá được tạo và kiểm chứng bởi tất cả các sản phẩm lấy mẫu của hệ thống trong mỗi lần sửa chữa. Ngoài ra, để cải thiện hiệu suất sửa chữa của cả hai phương pháp, Luận án cũng giới thiệu một số quy

tắc để đánh giá *điều hướng các điểm sửa đổi* và *lựa chọn các sửa đổi phù hợp*. Các quy tắc này sử dụng kết quả thử nghiệm trung gian của các chương trình trong quá trình sửa lỗi làm phản hồi để tinh chỉnh kết quả định vị lỗi và đánh giá tính phù hợp của các sửa đổi trước khi thực sự áp dụng và kiểm chứng chúng bằng cách thực hiện kiểm thử.

Để đánh giá các phương pháp đề xuất, Luận án đã tiến hành một số thực nghiệm trên một tập dữ liệu công khai lớn về các hệ thống SPL có chứa lỗi biến đổi. Kết quả thử nghiệm cho thấy Clap có thể phát hiện hiệu quả các *sản phẩm đúng giả* và *sản phẩm đúng thật* với độ chính xác trung bình trên 90%. Đặc biệt, độ chính xác khi phát hiện *sản phẩm đúng giả* bằng Clap lên tới 96%. Điều này có nghĩa là trong số mười sản phẩm được dự đoán là *sản phẩm đúng giả*, có hơn chín sản phẩm được phát hiện chính xác.

Về vấn đề định vị lỗi biến đổi, VarCop cải thiện hiệu quả của hai kỹ thuật tiên tiến lần lượt là 33% và 50% trong việc xếp hạng các câu lệnh lỗi trong các hệ thống chứa một lỗi duy nhất. Trong khoảng 2/3 trường hợp, VarCop xếp hạng chính xác các câu lệnh lỗi ở vị trí Top-3 trong danh sách xếp hạng. Đối với các trường hợp hệ thống có nhiều lỗi, VarCop vượt trội hơn các phương pháp tiếp cận tiên tiến hai lần và mười lần về tỷ lệ lỗi được định vị chính xác.

Ngoài ra, về vấn đề sửa các lỗi biến đổi, kết quả của Luận án cho thấy rằng phương pháp *hướng sản phẩm* tốt hơn khoảng 20 lần so với phương pháp *hướng hệ thống* về số lần sửa lỗi chính xác. Đáng chú ý, các quy tắc đề xuất có thể cải thiện hiệu suất của cả hai phương pháp bằng cách tăng 30-150% số lần sửa đúng và giảm 30-50% số lần thử sửa đổi.

Keywords: *Phần mềm dòng sản phẩm*, *Lỗi biến đổi*, *Sản phẩm đúng giả*, *Định vị lỗi*, *Tự động sửa lỗi*

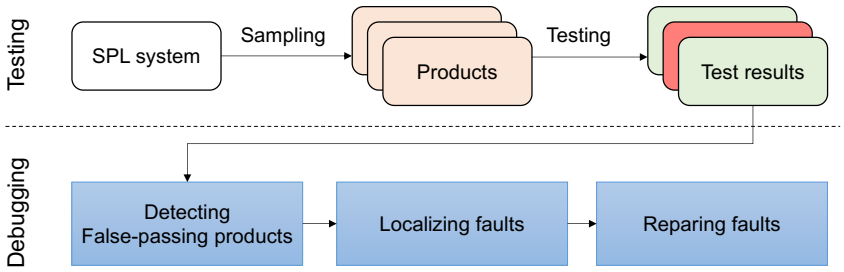
Chương 1

Giới thiệu

Ngày nay, các hệ thống Phần mềm dòng sản phẩm (SPL) (hoặc Hệ thống có thể cấu hình nói chung) đang trở nên phổ biến và được sử dụng rộng rãi để phát triển các dự án lớn trong công nghiệp. Hệ thống SPL là một họ sản phẩm chứa một tập hợp các sản phẩm có chung bộ mã nguồn cơ sở. Mỗi sản phẩm được xác định bằng các tính năng đã chọn. Nói cách khác, một dự án áp dụng phương pháp SPL có thể tuỳ chỉnh các đặc tính chức năng và phi chức năng của nó cho phù hợp với yêu cầu của người dùng. Điều này có thể được thực hiện bằng cách sử dụng một số lượng rất lớn lựa chọn (*options*) để kiểm soát các tính năng (*features*). Một tập hợp *lựa chọn* gồm tất cả các tính năng được gọi là một cấu hình, (*configurations*). Cấu hình định nghĩa một phiên bản của hệ thống (*variant*) hay còn gọi là một sản phẩm (*product*). Ví dụ, Linux hỗ trợ hàng nghìn tính năng được kiểm soát bởi các hơn 12.000 tuỳ chọn. Từ đó, nhà phát triển có thể tạo ra hàng tỉ phiên bản phần mềm Linux khác nhau.

Mặc dù tính biến đổi của hệ thống SPL tạo ra nhiều lợi ích trong quá trình phát triển phần mềm, nhưng đặc điểm này cũng gây ra các thách thức cho việc đảm bảo chất lượng phần mềm. So với kỹ thuật phát triển hệ thống đơn truyền thống, việc phát hiện lỗi, định vị lỗi và sửa chữa thông qua kiểm thử trong hệ thống SPL gặp nhiều khó khăn hơn. Lý do vì lỗi có thể có tính chất biến đổi (được gọi là *variability bug*), lỗi này chỉ có thể được thể hiện ra ngoài dưới sự kết hợp nhất định của một số tính năng của hệ thống. Nói cách khác, tồn tại một tập hợp các tính năng phải được *bật* và *tắt cùng nhau* để kích hoạt hành vi không đúng của lỗi này. Do sự xuất hiện/không xuất hiện của các *tương tác* giữa các tính năng đó, các câu lệnh lỗi có thể biểu hiện khác nhau trong các sản phẩm khác nhau. Do đó, *những câu lệnh lỗi chỉ có thể bộc lộ lỗi ở một số sản phẩm nhất định, nhưng lại không thể hiện điều này ở những sản phẩm khác.* Đặc biệt trong hệ thống SPL, các lỗi biến đổi chỉ gây ra lỗi ở một số sản phẩm nhất định, trong khi những sản phẩm khác vẫn thành công ở tất cả các ca kiểm thử của chúng.

Mặc dù việc gỡ lỗi tự động trong phát triển hệ thống phần mềm đơn đã được



Hình 1.1: Quy trình gỡ lỗi tự động đề xuất trong hệ thống phần mềm dòng sản phẩm

nghiên cứu chuyên sâu, việc gỡ lỗi các hệ thống SPL hầu như vẫn chưa được khám phá. Luận án nghiên cứu việc gỡ lỗi tự động các hệ thống SPL trong ba nhiệm vụ chính, bao gồm phát hiện sản phẩm *sản phẩm đúng giả*, định vị các lỗi biến đổi và sửa chữa các lỗi đó trong hệ thống SPL. Hình 1.1 thể hiện quy trình gỡ lỗi tự động cho hệ thống SPL được đề xuất bởi Luận án.

Luận án đề xuất các phương pháp tự động gỡ lỗi cho các hệ thống SPL bị thất bại gây ra bởi lỗi biến đổi. Để cải thiện độ tin cậy của kết quả kiểm thử, Luận án đề xuất Clap, một phương pháp phát hiện *sản phẩm đúng giả*. Tiếp theo, Luận án trình bày VarCop, một phương pháp định vị mới chuyên dùng cho các lỗi biến đổi của hệ thống SPL. Cuối cùng, Luận án giới thiệu hai phương pháp *hướng sản phẩm* và *hướng hệ thống* để tự động sửa chữa các lỗi biến đổi.

Đầu tiên, *Luận án giới thiệu* Clap, *một phương pháp tiếp cận* ***phát hiện sản phẩm đúng giả*** *của hệ thống SPL.* Ý tưởng chính trong cách tiếp cận của Clap là đối với hệ thống SPL có chứa lỗi, các sản phẩm có thể có chung một số chức năng. Nếu các hành vi không mong muốn của các chức năng được phát hiện qua các kết quả kiểm thử ở một số sản phẩm, các sản phẩm khác có chức năng tương tự cũng có thể bị lỗi do những hành vi không mong muốn đó. Trong Clap, các *sản phẩm đúng giả* có thể được phát hiện dựa trên các *dấu hiệu lỗi* được thu thập bằng cách xem xét *mã nguồn* và *chất lượng bộ kiểm thử* của các sản phẩm không thành công. Để đánh giá khả năng một sản phẩm là *sản phẩm đúng giả*, Luận án đề xuất một số *thuộc tính có thể đo lường* để đánh giá độ mạnh của các dấu hiệu lỗi này trong sản phẩm. Kết quả đánh giá của các thuộc tính càng cao thể hiện khả năng sản phẩm đó là *sản phẩm đúng giả* càng cao.

Thứ hai, *Luận án đề xuất* VarCop, *một cách tiếp cận mới để* ***định vị các lỗi biến đổi***. Ý tưởng chính VarCop là các lỗi biến đổi được định vị dựa trên (i) sự

tương tác giữa các tính năng cần thiết để phát hiện ra lỗi và (ii) mức độ khả nghi của các câu lệnh được thể hiện thông qua cả kết quả kiểm thử tổng thể của sản phẩm và kết quả kiểm thử chi tiết của từng ca kiểm thử.

Thứ ba, *Luận án đề xuất hai cách tiếp cận hướng sản phẩm và hướng hệ thống để* ***tự động sửa các lỗi biến đổi*** *của hệ thống SPL.* Bên cạnh đó, *Luận án cũng giới thiệu một số quy tắc để cải thiện hiệu suất của hai phương pháp trong việc sửa chữa hệ thống SPL có lỗi.* Luận án bắt đầu từ việc quan sát rằng, để sửa lỗi một cách chính xác và hiệu quả, công cụ tự động sửa lỗi phải quyết định chính xác (i) vị trí cần sửa và (ii) các thay đổi phù hợp. Các quy tắc của Luận án tập trung vào việc nâng cao độ chính xác của các tác vụ này bằng cách sử dụng các kết quả thử nghiệm trung gian của quá trình sửa lỗi.

Tóm lại, Luận án có những đóng góp chính sau:

- Định nghĩa bài toán phát hiện *sản phẩm đúng giả* trong hệ thống SPL và giới thiệu một bộ dữ liệu để đánh giá các kỹ thuật phát hiện *sản phẩm đúng giả*.
- CLAP: một cách tiếp cận hiệu quả để phát hiện các *sản phẩm đúng giả* trong hệ thống SPL và giảm thiểu tác động tiêu cực của chúng đối với hiệu quả định vị lỗi biến đổi. Công cụ đề xuất được triển khai tại: *https://ttrangnguyen.github.io/CLAP/*.
- Định nghĩa tập tính năng cần thiết mà trong đó sự tương tác giữa các tính năng này là nguyên nhân thất bại của sản phẩm gây ra bởi các lỗi biến đổi trong hệ thống SPL.
- VARCOP: Một cách tiếp cận/công cụ mới, hiệu quả để định vị các lỗi biến đổi trong hệ thống SPL. Công cụ đề xuất, VARCOP, được triển khai tại: *https://ttrangnguyen.github.io/VARCOP/*.
- Quy tắc để điều hướng các điểm sửa đổi và lựa chọn các sửa đổi phù hợp để cải thiện hiệu suất của các công cụ tự động sửa lỗi.
- Phương pháp *hướng sản phẩm* và *hướng hệ thống* để sửa các lỗi biến đổi trong mã nguồn của hệ thống SPL. Các phương pháp đề xuất được triển khai tại: *https://github.com/ttrangnguyen/SPLRepair*.
- Thử nghiệm đánh giá các giải pháp đề xuất cho thấy kết quả vượt trội của chúng so với các giải pháp khác.

Phần còn lại của Luận án được tổ chức như sau. Chương 2 giới thiệu kiến thức cơ sở và khảo sát các nghiên cứu liên quan. Phương pháp được đề xuất để phát hiện các *sản phẩm đúng giả* được giới thiệu trong Chương 3. Phương pháp đề xuất để định vị các lỗi biến đổi được mô tả trong Chương 4. Chapter 5 trình bày hai cách tiếp cận *hướng sản phẩm* và *hướng hệ thống* của Luận án để tự động sửa các lỗi biến đổi trong hệ thống SPL. Cuối cùng, Chương 6 tóm tắt và kết luận Luận án.

Chương 2

Kiến thức cơ sở và nghiên cứu liên quan

Phần này giới thiệu kiến thức cơ sở và các khái niệm được sử dụng trong các phần tiếp theo của Luận án. Đầu tiên, Chương 2 giới thiệu các khái niệm chính của hệ thống SPL, các phương pháp kiểm thử chính, kỹ thuật định vị lỗi và tự động sửa lỗi. Tiếp theo, Chương này khảo sát các nghiên cứu liên quan. Cuối cùng, chương này giới thiệu các bộ dữ liệu phổ biến để đánh giá các phương pháp kiểm thử và gỡ lỗi của hệ thống SPL.

Với kỹ thuật phát triển phần mềm dòng sản phẩm, thay vì phân tích và triển khai từng sản phẩm, các nhà phát triển nhắm tới mục tiêu phát triển cùng lúc *nhiều sản phẩm tương tự* nhau nhưng *không giống nhau hoàn toàn*. Để đạt được mục tiêu này, quá trình phát triển hệ thống SPL xem xét hai yếu tố quan trọng: *tính biến đổi* và *tái sử dụng*. Trong quy trình phát triển hệ thống SPL, có hai quy pha chính: Kỹ thuật miền và Kỹ thuật ứng dụng. Nhìn chung, SPL là một dòng sản phẩm bao gồm một bộ sản phẩm có chung mã nguồn cơ sở. Các sản phẩm này khác biệt với các sản phẩm khác về một số tính năng (*features*). Không giống như các sản phẩm phần mềm đơn truyền thống, lỗi trong hệ thống SPL có thể là *lỗi biến đổi* và chỉ gây ra lỗi ở một số sản phẩm nhất định.

Để đảm bảo chất lượng của phần mềm, kiểm thử là phương pháp phổ biến nhất. Mặc dù việc kiểm thử có thể giúp phát hiện ra lỗi do các hành vi sai sót được phát hiện thông qua quan sát kết quả, nhưng việc tìm và sửa chúng lại là một vấn đề hoàn toàn khác. Định vị lỗi hay xác định vị trí của câu lệnh lỗi chương trình, rất quan trọng trong việc gỡ lỗi chương trình, nhưng thường được coi là một hoạt động tẻ nhạt, tốn thời gian và rất tốn kém chi phí. Để định vị lỗi phần mềm hiệu quả, nhiều phương pháp định vị lỗi hỗ trợ tìm lỗi bán tự động hoặc hoàn toàn tự động đã được đề xuất. Các phương pháp định vị lỗi này thường được phân loại thành tám nhóm theo kỹ thuật của chúng, bao gồm các kỹ thuật dựa trên lát cắt, dựa trên phổ, dựa trên thống kê, dựa trên trạng thái chương trình, dựa trên học máy, dựa trên khai phá dữ liệu, dựa trên mô hình và các kỹ thuật khác.

Trong số các kỹ thuật này, định vị lỗi dựa trên phổ (Spectrum-based fault localiza-

tion) được cho là phổ biến nhất do tính hiệu quả và hữu hiệu của nó. Cụ thể, định vị lỗi dựa trên phổ là một kỹ thuật phân tích chương trình tự động sử dụng thông tin thực thi kiểm thử để tính toán điểm khả nghi của các thành phần trong mã nguồn như câu lệnh, khối, phương thức, v.v. Ý tưởng chính là, trong một chương trình, một câu lệnh thực thi qua nhiều ca kiểm thử thất bại và càng ít ca kiểm thử thành công thì câu lệnh đó càng khả nghi là câu lệnh lỗi. Câu lệnh có điểm khả nghi cao hơn có nhiều khả năng bị lỗi hơn.

Để giảm thiểu các chi phí bảo trì phần mềm, nhiều kỹ thuật tự động sửa lỗi đã được đề xuất. Cách tiếp cận tự động sửa lỗi phổ biến nhất là *sửa lỗi chương trình dựa trên bộ các ca kiểm thử*, ví dụ như GenProg, Nopol và Cardumen. Các kỹ thuật này sử dụng bộ kiểm thử như đặc tả hành vi mong muốn của chương trình. Để sửa chữa một chương trình bị lỗi, các phương pháp tự động sửa lỗi này cố gắng tạo ra các bản vá tiềm năng. Sau đó, các ca kiểm thử có sẵn sẽ được sử dụng để kiểm tra xem các bản vá được tạo có thể khắc phục được lỗi chương trình hay không.

Chương 3

Xác định các sản phẩm đúng giả

Để đảm bảo chất lượng của chương trình việc kiểm thử thường được yêu cầu thực hiện kỹ lưỡng. Tuy nhiên, việc tiến hành kiểm thử kỹ lưỡng trong thực tế thường khó khăn, tẻ nhạt và tốn thời gian. Các bộ ca kiểm thử có thể bỏ qua nhiều lỗi khác nhau vì việc bao quát tất cả các hành vi của chương trình là cực kỳ khó khăn. Hơn nữa, có những loại lỗi rất khó phát hiện do khó có thể lan truyền tính không chính xác của chúng đến kết quả đầu ra. Do đó, ngay cả khi ca kiểm thử thực thi qua các câu lệnh chứa lỗi, ca kiểm thử vẫn có thể thu được kết quả thành công. Nói một cách khác, các ca kiểm thử như vậy là các ca kiểm thử *trùng hợp đúng/ngẫu nhiên thành công* (coincidentially correct). Trong thực tế, tính đúng ngẫu nhiên là một vấn đề phổ biến trong kiểm thử phần mềm và hiện tượng này gây ra các tác động tiêu cực đến hiệu suất định vị lỗi.

Tương tự như việc kiểm thử mã nguồn của chương trình đơn truyền thống, hiện tượng tính đúng ngẫu nhiên cũng xảy ra trong hệ thống SPL và gây khó khăn trong việc tìm ra lỗi trong các hệ thống này. Cụ thể, đối với hệ thống SPL, một số sản phẩm thường được lấy mẫu để kiểm thử. Mỗi sản phẩm được lấy mẫu bao gồm một tập hợp các tính năng của hệ thống và được thực hiện kiểm thử với một bộ các ca kiểm thử riêng giống như một chương trình đơn. Đối với hệ thống SPL có lỗi, lỗi có thể xảy ra ở một hoặc nhiều sản phẩm. Lý tưởng nhất là nếu một sản phẩm có lỗi, thì bộ các ca kiểm thử của sản phẩm đó sẽ phát hiện ra lỗi. Nói cách khác, sau khi kiểm thử phải có ít nhất một ca kiểm thử thất bại. Tuy nhiên, nếu bộ kiểm thử của một sản phẩm có lỗi không hiệu quả trong việc phát hiện lỗi thì kết quả kiểm thử tổng thể của sản phẩm vẫn có thể được đánh giá là thành công. Ví dụ, bộ kiểm thử không bao bao phủ các câu lệnh lỗi của sản phẩm hoặc các trường hợp kiểm thử đó có thể thực thi các câu lệnh lỗi nhưng không thể truyền trạng thái không chính xác của hệ thống đến kết qủa đầu ra, sản phẩm vẫn vượt qua tất cả các ca kiểm thử. Một sản phẩm như vậy thực sự là một sản phẩm có lỗi, nhưng kết quả kiểm thử thể hiện không chính xác. Sản phẩm như vậy chính là *sản phẩm đúng giả*. Do kết quả kiểm thử không đáng tin cậy nên các *sản phẩm*

đúng giả này có thể tác động tiêu cực đến hiệu suất định vị lỗi. Đặc biệt, hiệu suất của hai chiến lược định vị lỗi chính dựa trên phổ của chương trình (SBFL) trong hệ thống SPL, *hướng sản phẩm* và *hướng ca kiểm thử*, bị ảnh hưởng nghiêm trọng.

Đầu tiên, kỹ thuật định vị lỗi *hướng sản phẩm* đánh giá mức độ khả nghi của một câu lệnh trong hệ thống SPL dựa trên tần suất xuất hiện của câu lệnh đó trong các sản phẩm thất bại. Đặc biệt, ý tưởng chính để tìm ra lỗi trong hệ thống SPL là một câu lệnh xuất hiện trong nhiều sản phẩm thất bại hơn và ít xuất hiện trong các sản phẩm thành công hơn sẽ có nhiều khả năng là câu lệnh lỗi hơn các câu lệnh khác của hệ thống. Việc đánh giá một sản phẩm có lỗi là sản phẩm thành công sẽ làm giảm độ chính xác trong việc tính số lượng sản phẩm thành công và số lượng sản phẩm thất bại có chứa câu lệnh lỗi. Do đó, câu lệnh có lỗi được coi là ít khả nghi hơn mức độ thực tế của nó.

Thứ hai, kỹ thuật định vị lỗi *hướng ca kiểm thử* đo lường điểm khả nghi của các câu lệnh dựa trên số ca kiểm thử thất bại và số ca kiểm thử thành công thực thi câu lệnh. Thật vậy, các *sản phẩm đúng giả* có thể dẫn đến việc đếm thiếu số lượng các ca kiểm thử (thực sự) thất bại và đếm quá mức số ca kiểm thử thành công được thực hiện bởi các câu lệnh lỗi. Nguyên nhân là do *sản phẩm đúng giả* có lỗi nhưng không có ca kiểm thử thất bại nào. Trong các *sản phẩm đúng giả* này, các câu lệnh lỗi không được thực thi bởi bất kỳ ca kiểm thử nào hoặc chúng vẫn có thể được thực thi bởi một số ca kiểm thử, tuy nhiên các ca kiểm thử đó *ngẫu nhiên* thành công. Trong cả hai trường hợp, bộ kiểm thử có độ bao phủ thấp và các ca kiểm thử được đánh giá thành công ngẫu nhiên như vậy đều có thể gây ra đánh giá không chính xác cho các câu lệnh có lỗi.

Chương này giới thiệu CLAP, một phương pháp phát hiện *sản phẩm đúng giả* dành cho các hệ thống SPL. Ý tưởng chính của CLAP là đối với hệ thống SPL, các sản phẩm có thể có chung một số chức năng. Nếu các hành vi không mong muốn của các chức năng được phát hiện qua các ca kiểm thử ở một số sản phẩm (gây ra ít nhất một ca kiểm thử thất bại cho sản phẩm), các sản phẩm khác có chức năng tương tự cũng có thể bị lỗi do những hành vi không mong muốn đó. Trong CLAP, các *sản phẩm đúng giả* có thể được phát hiện dựa trên các *dấu hiệu lỗi* được thu thập bằng cách xem xét *mã nguồn* và *chất lượng bộ kiểm thử* của các sản phẩm không thành công. Để đánh giá khả năng một sản phẩm là *sản phẩm đúng giả*, Luận án đề xuất một số *thuộc tính có thể đo lường* để đánh giá độ mạnh của các

dấu hiệu lỗi này trong sản phẩm. Các dấu hiện này càng rõ ràng thì khả năng sản phẩm đó là *sản phẩm đúng giả* càng cao.

Các thuộc tính được đề xuất thuộc về hai khía cạnh: *mã nguồn của sản phẩm* và *chất lượng bộ kiểm thử* (tính đầy đủ và hiệu quả của bộ kiểm thử). Các thuộc tính liên quan đến *mã nguồn sản phẩm* phản ánh khả năng sản phẩm có chứa câu lệnh lỗi. Nếu sản phẩm có nhiều câu lệnh khả nghi được thực thi bởi các ca kiểm thử không thành công trong các sản phẩm bị lỗi của hệ thống thì sản phẩm có nhiều khả năng chứa lỗi hơn. Đối với *chất lượng bộ kiểm thử* của sản phẩm, *tính đầy đủ* phản ánh cách bộ kiểm thử bao phủ và kiểm tra các thành phần mã nguồn như câu lệnh, nhánh hoặc đường thực thi. Bộ kiểm thử có phạm vi bao phủ thấp có thể bỏ qua các phần tử không chính xác trong sản phẩm có lỗi. Do đó, sản phẩm có bộ kiểm thử có độ bao phủ thấp hơn có nhiều khả năng là *sản phẩm đúng giả* hơn. Trong khi đó, *tính hiệu quả* phản ánh độ mạnh của bộ kiểm thử trong việc xác minh các hành vi của sản phẩm cũng như khả năng phát hiện các hành vi không mong muốn của sản phẩm. Ý tưởng chính là nếu sản phẩm được kiểm thử bởi bộ kiểm thử kém hiệu quả hơn, kết quả kiểm thử tổng thể của nó kém tin cậy hơn. Khi đó, sản phẩm có nhiều khả năng là *sản phẩm đúng giả* hơn.

Bên cạnh đó, Luận án cũng thảo luận về một số chiến lược nhằm giảm thiểu tác động tiêu cực của các *sản phẩm đúng giả* tới hiệu suất của các phương pháp định vị lỗi. Do các tác động tiêu cực này chủ yếu gây ra bởi kết quả kiểm thử không đáng tin cậy nên mục tiêu của Luận án là cải thiện *độ tin cậy* của kết quả kiểm thử bằng cách nâng cao chất lượng kiểm thử dựa trên các dấu hiệu lỗi. Hơn nữa, độ tin cậy của kết quả kiểm thử cũng có thể được cải thiện bằng cách bỏ qua các kết quả kiểm thử không đáng tin cậy ở cấp độ sản phẩm hoặc cấp độ ca kiểm thử.

Luận án đã tiến hành một số thực nghiệm trên một tập dữ liệu lớn về các lỗi biến đổi, trong đó có 823 phiên bản có lỗi của sáu hệ thống SPL được sử dụng rộng rãi trong các nghiên cứu liên quan. Tổng cộng có 14.191 *sản phẩm đúng giả* và 22.555 *sản phẩm đúng thật*. Kết quả thực nghiệm cho thấy Clap đạt được hơn 90% *độ chính xác* trong việc phát hiện các *sản phẩm đúng giả* và *sản phẩm đúng thật*. Luận án cũng đánh giá khả năng của Clap trong việc giảm thiểu tác động tiêu cực của các *sản phẩm đúng giả* đối với hiệu suất định vị lỗi. Kết quả thử nghiệm cho thấy Clap có thể giảm thiểu đáng kể tác động tiêu cực của các *sản phẩm đúng giả* trong việc định vị các lỗi biến đổi và giúp tăng tốc độ tìm lỗi.

Chương 4

Định vị lỗi biến đổi

Tính biến đổi vốn có của hệ thống SPL là một thách thức lớn đối với các hoạt động đảm bảo chất lượng phần mềm. So với kỹ thuật phát triển hệ thống đơn, việc phát hiện và định vị lỗi thông qua kết quả kiểm thử trong các hệ thống SPL gặp nhiều vấn đề phức tạp hơn, vì lỗi có thể chỉ được phát hiện dưới sự kết hợp của *một vài* tính năng của hệ thống. Cụ thể, tồn tại một tập hợp các tính năng phải được bật và tắt cùng nhau để có thể phát hiện ra lỗi. Do sự xuất hiện/không xuất hiện của sự *tương tác* giữa các tính năng đó, các câu lệnh lỗi có thể có các hành vi khác nhau trong các sản phẩm khác nhau. Do đó, *những câu lệnh lỗi chỉ có thể bộc lộ lỗi ở một số sản phẩm cụ thể, nhưng không thể thể hiện lỗi ở những sản phẩm khác.* Đặc biệt, trong hệ thống SPL, các lỗi biến đổi chỉ gây ra thất bại ở một số sản phẩm nhất định, còn những sản phẩm khác vẫn thành công trong tất cả các ca kiểm thử của chúng. Tính chất biến đổi này gây ra những khó khăn đáng kể cho việc định vị loại lỗi này trong hệ thống SPL.

Mặc dù việc *định vị lỗi biến đổi* là rất cần thiết, tuy nhiên các phương pháp định vị lỗi hiện tại không được thiết kế cho loại lỗi này. Những kỹ thuật này được thiết kế chuyên biệt cho việc định vị lỗi trong một sản phẩm cụ thể. Ví dụ, để định vị các câu lệnh gây ra lỗi trong nhiều sản phẩm của một hệ thống SPL, các phương pháp dựa trên lát cắt chương trình có thể được sử dụng để xác định tất cả các lát cắt liên quan đến lỗi cho từng sản phẩm một cách độc lập với các sản phẩm khác. Do đó, có nhiều bộ (số lượng lớn) câu lệnh riêng biệt cần được kiểm tra để tìm ra lỗi. Điều này khiến các phương pháp dựa trên lát cắt trở nên không khả thi trong các hệ thống SPL.

Ngoài ra, lập trình viên cũng có thể sử dụng kỹ thuật định vị lỗi dựa trên phổ chương trình (Spectrum-based fault localization-SBFL) để tính điểm khả nghi cho các câu lện dựa trên thông tin thực thi kiểm thử (tức là phổ chương trình) của từng sản phẩm của hệ thống một cách riêng biệt. Đối với mỗi sản phẩm, phương pháp này tạo ra một danh sách xếp hạng các câu lệnh khả nghi. Kết quả là, có thể có nhiều danh sách xếp hạng được tạo cho một hệ thống SPL thất bại do lỗi

biến đổi gây ra. Từ nhiều danh sách này, các nhà phát triển rất khó để xác định điểm bắt đầu để kiểm tra nguyên nhân gây ra lỗi. Do đó, việc tìm ra các lỗi biến đổi bằng cách sử dụng phương pháp định vị lỗi dựa trên phổ để xếp hạng các câu lệnh khả nghi trong nhiều sản phẩm một cách riêng biệt là không hiệu quả.

Một phương pháp khác để áp dụng phương pháp định vị lỗi dựa trên phổ trong hệ thống SPL là lập trình viên có thể coi toàn bộ hệ thống như một chương trình duy nhất. Điều này có nghĩa là cơ chế kiểm soát sự bật/tắt của các tính năng trong hệ thống (ví dụ, các chỉ thị tiền xử lý `#ifdef`) sẽ được coi như các câu lệnh điều kiện `if-then` tương ứng trong quá trình định vị lỗi. Bằng cách điều chỉnh này, một danh sách xếp hạng duy nhất của các câu lệnh trong chương trình có thể được tạo ra tùy theo mức độ khả nghi của chúng. Trong chương này, Luận án xem xét các sản phẩm được kiểm thử theo hướng sản phẩm. Nói cách khác, mỗi sản phẩm được kiểm thử bằng một bộ kiểm thử riêng biệt. Ngoài ra, một ca kiểm thử được thiết kế để thử nghiệm một tính năng trong kỹ thuật miền, có thể được cụ thể hóa cho nhiều trường hợp kiểm thử theo yêu cầu của từng sản phẩm trong kỹ thuật ứng dụng. Bằng cách sử dụng phương pháp định vị này, mức độ khả nghi của các câu lệnh được đo lường dựa trên tổng số ca kiểm thử thành công và không thành công do nó thực hiện trong tất cả các sản phẩm được kiểm thử của hệ thống. Trong khi đó, các đặc điểm như sự tương tác giữa các tính năng của hệ thống và tính biến đổi của lỗi giữa các sản phẩm cũng rất hữu ích để định vị các lỗi biến đổi trong hệ thống SPL. Tuy nhiên, những loại thông tin quan trọng này không được sử dụng trong các phương pháp hiện có.

Chương này đề xuất VarCop, một phương pháp định vị lỗi mới cho các lỗi biến đổi. Ý tưởng chính của VarCop là các lỗi biến đổi được định vị dựa trên (i) sự tương tác giữa các tính năng cần thiết để phát hiện ra lỗi và (ii) mức độ khả nghi của các câu lệnh được phản ánh qua cả kết quả kiểm thử tổng thể của sản phẩm và kết quả kiểm thử chi tiết của từng ca kiểm thử trong sản phẩm.

Với một hệ thống SPL có lỗi, VarCop xác định tập các tính năng cần được chọn bật/tắt cùng nhau để khiến hệ thống thể hiện ra lỗi bằng cách phân tích kết quả kiểm thử tổng thể (tức là trạng thái thành công ở tất cả các ca kiểm thử hoặc không thành công ở ít nhất một ca kiểm thử) của sản phẩm. Luận án gọi mỗi bộ lựa chọn tính năng này là một BPC. Sau đó, VarCop phân tích sự tương tác giữa các tính năng trong BPC này để xác định các câu lệnh khả nghi. Trong VarCop,

mức độ khả nghi của từng câu lệnh được đánh giá dựa trên hai tiêu chí. Tiêu chí đầu tiên dựa trên kết quả kiểm thử tổng thể của các sản phẩm có chứa câu lệnh. Theo tiêu chí này, một câu lệnh xuất hiện trong càng nhiều sản phẩm thất bại và càng ít sản phẩm thành công thì câu lệnh đó càng khả nghi. Trong khi đó, tiêu chí thứ hai đánh giá mức độ khả nghi của các câu lệnh trong mỗi sản phẩm không thành công. Đặc biệt, ở mỗi sản phẩm thất bại, mức độ khả nghi của câu lệnh được đo lường dựa trên kết quả chi tiết của từng ca kiểm thử của sản phẩm. Ý tưởng là nếu trong một sản phẩm thất bại, câu lệnh có độ khả nghi cao thì câu lệnh đó cũng có độ khả nghi cao trên toàn hệ thống.

Luận án đã tiến hành thực nghiệm để đánh giá độ hiệu quả VarCop trong cả các hệ thống có duy nhất một lỗi và các hệ thống có nhiều lỗi trên tập dữ liệu gồm 1.570 phiên bản (trường hợp) có chứa (các) lỗi biến đổi. Luận án đã so sánh VarCop với các phương pháp tiên tiến nhất bao gồm phương pháp định vị dựa trên phổ, sự kết hợp giữa phương pháp lát cắt chương trình và phương pháp dựa trên phổ, và định vị lỗi hướng tính năng, trên 30 chỉ số xếp hạng phổ biến nhất. Kết quả thực nghiệm cho thấy VarCop vượt trội đáng kể so với các giải pháp này trong tất cả các chỉ số được xem xét.

Chương 5

Tự động sửa lỗi biến thể

Trong thực tế, lỗi là một vấn đề không thể tránh khỏi trong các chương trình phần mềm. Các nhà phát triển thường cần dành khoảng 50% thời gian của mình để giải quyết các lỗi phần mềm. Việc phát hiện và sửa lỗi trong hệ thống SPL có thể rất phức tạp do tính chất biến đổi của chúng. Echeverría và cộng sự đã tiến hành một nghiên cứu thực nghiệm để đánh giá hành vi của các kỹ sư trong việc sửa lỗi và áp dụng các bản vá sửa lỗi cho các sản phẩm khác nhau trong hệ thống SPL công nghiệp. Họ chỉ ra rằng việc sửa các hệ thống SPL có nhiều lỗi là một thách thức, đặc biệt đối với các hệ thống lớn. Thật vậy, trong hệ thống SPL, mỗi sản phẩm bao gồm một bộ tính năng khác nhau. Do sự tương tác của các tính năng khác nhau, một lỗi trong hệ thống SPL có xuất hiện ở một số sản phẩm của hệ thống nhưng không xuất hiện ở các sản phẩm khác, nên được gọi là *lỗi biến đổi*. Để khắc phục các lỗi biến đổi, chúng ta cần tìm bản vá lỗi không chỉ áp dụng được cho một sản phẩm mà còn cần áp dụng được cho tất cả các sản phẩm của hệ thống, tức là chúng ta cần sửa các hành vi không đúng của tất cả *sản phẩm bị lỗi* và không làm ảnh hưởng tới các hành vi đúng của *sản phẩm thành công* .

Để giảm thiểu chi phí bảo trì phần mềm và giảm bớt gánh nặng của các hoạt động gỡ lỗi thủ công, nhiều phương pháp tự động sửa lỗi trong chương trình đã được đề xuất trong những thập kỷ gần đây. Các phương pháp này sử dụng các kỹ thuật khác nhau để tổng hợp các bản vá một cách tự động (tức là không cần sự can thiệp của con người) giúp loại bỏ lỗi chương trình và thu được kết quả đầy hứa hẹn. Tuy nhiên, các phương pháp này tập trung vào việc sửa lỗi trong một hệ thống đơn truyền thống.

Trong bối cảnh của hệ thống SPL, có một số nghiên cứu cố gắng giải quyết các lỗi biến đổi ở các cấp độ khác nhau, chẳng hạn như mô hình hoặc cấu hình. Ví dụ, Arcaini và cộng sự cố gắng sửa lỗi trong các mô hình biến thể. Weiss và cộng sự đề xuất phương pháp sửa lỗi các cấu hình sai của hệ thống SPL. Tuy nhiên, việc sửa các lỗi biến đổi ở cấp độ mã nguồn vẫn chưa được nghiên cứu.

Trong chương này, Luận án đề xuất hai phương pháp tiếp cận *hướng sản phẩm*

và *hướng hệ thống* để sửa chữa các hệ thống SPL có lỗi ở cấp độ mã nguồn. Đối với cách tiếp cận *hướng sản phẩm* ($ProdBased_{basic}$), mỗi sản phẩm bị lỗi của hệ thống sẽ được sửa riêng lẻ và sau đó các bản vá thu được sẽ được áp dụng và kiểm chứng trên các sản phẩm khác của hệ thống. Đối với phương pháp *hướng hệ thống* ($SysBased_{basic}$), thay vì sửa từng sản phẩm riêng lẻ tại một thời điểm, tất cả các sản phẩm đều được xem xét để sửa đồng thời. Cụ thể, các bản vá được tạo ra và sau đó được kiểm chứng bởi tất cả các sản phẩm của hệ thống trong mỗi lần sửa. Đối với cả hai phương pháp, các bản vá hợp lệ là các bản vá khiến tất cả các ca kiểm thử của tất cả các sản phẩm lấy mẫu của hệ thống đều thành công.

Ngoài ra, Luận án cũng giới thiệu một số quy tắc để cải thiện hiệu suất của hai phương pháp trong việc sửa các lỗi biến đổi của hệ thống SPL. Luận án bắt đầu từ việc quan sát rằng, để sửa lỗi một cách hiệu quả, các công cụ tự động sửa lỗi phải quyết định chính xác (i) vị trí cần sửa và (ii) những thay đổi phù hợp. Các quy tắc của Luận án tập trung vào việc nâng cao độ chính xác của các tác vụ này bằng cách sử dụng các kết quả thử nghiệm trung gian của quá trình sửa lỗi.

Đối với việc *điều hướng các vị trí sửa đổi*, các công cụ tự động sửa lỗi thường sử dụng *điểm khả nghi*, các điểm này thể hiện khả năng một câu lệnh là câu lệnh lỗi. Những điểm số này thường được tính một lần trước quá trình sửa chữa bằng các kỹ thuật định vị lỗi. Tuy nhiên, có rất nhiều thông tin hữu ích có thể thu được trong quá trình sửa lỗi, chẳng hạn như kết quả kiểm thử của các chương trình đã sửa đổi. Những thông tin như vậy có thể cung cấp phản hồi có giá trị để liên tục tinh chỉnh việc điều hướng các điểm sửa đổi. Vì vậy, trong nghiên cứu này, bên cạnh điểm khả nghi, *điểm sửa lỗi* của các câu lệnh cũng được sử dụng để điều hướng các vị trí sửa lỗi trong mỗi lần sửa chữa. Các điểm này thể hiện khả năng chương trình có thể được sửa bằng cách thay đổi mã nguồn tại các vị trí tương ứng. Điểm sửa lỗi được đo lường và cập nhật liên tục theo kết quả kiểm thử trung gian của các chương trình sửa đổi. Ý tưởng của Luận án là, *nếu việc thay đổi mã nguồn tại một vị trí mp khiến một hoặc một số ca kiểm thử thất bại ban đầu trở nên thành công thì mp có thể là vị trí chính xác của lỗi hoặc có liên quan đến lỗi.* Mặt khác, nếu vị trí đó không liên quan đến lỗi, việc thay đổi mã nguồn của nó sẽ không thể thay đổi kết quả của các ca kiểm thử thất bại. Các vị trí có điểm sửa lỗi cao và điểm nghi ngờ cao nên được ưu tiên kiểm thử trong mỗi lần sửa lỗi tiếp theo.

Sau khi chọn vị trí sửa đổi, các công cụ tự động sửa lỗi sẽ tạo và *chọn các thay đổi*

phù hợp cho vị trí đó và đánh giá chúng bằng cách thực hiện các ca kiểm thử. Việc kiểm chứng động này tốn nhiều thời gian và tiêu tốn một lượng lớn tài nguyên. Để giảm thiểu lãng phí thời gian kiểm chứng các thay đổi không chính xác, Luận án giới thiệu quy tắc *đo mức độ phù hợp của thay đổi* giúp nhanh chóng đánh giá và loại bỏ các thay đổi không phù hợp. Tính phù hợp của một thay đổi tại vị trí *mp* được đánh giá bằng sự tương đồng của thay đổi đó với mã nguồn ban đầu và với các sửa đổi đã thử trước đó tại *mp*. Thực tế, *sự thay đổi chính xác tại mp thường giống với mã ban đầu của nó và những sửa đổi thành công khác tại vị trí này, trong khi những thay đổi tương tự với những thay đổi thất bại trước đó thường không không phải là bản vá chính xác*. Do đó, một thay đổi càng giống với mã gốc và các sửa đổi thành công và càng ít giống với các sửa đổi thất bại thì sửa đổi đó càng phù hợp hơn để thử tại *mp*.

Luận án cài đặt các quy tắc này vào cả hai cách tiếp cận *hướng sản phẩm* và *hướng hệ thống*, đồng thời các phiên bản nâng cao lần lượt được gọi là $ProdBased_{enhanced}$ và $SysBased_{enhanced}$.

Luận án đã thực hiện một số thực nghiệm với các giải pháp đề xuất trên tập dữ liệu gồm 318 phiên bản lỗi của 5 hệ thống SPL (tức là 318 lỗi biến đổi). Kết quả thử nghiệm cho thấy phương pháp *hướng sản phẩm* tốt hơn đáng kể so với phương pháp *hướng hệ thống* từ **12 đến 30 lần** về số bản vá hợp lệ và khoảng **20 lần** về số bản vá đúng. Điều thú vị là các quy tắc đề xuất của Luận án có thể giúp tăng hiệu suất của cả hai phương pháp *hướng sản phẩm* và *hướng hệ thống* lên tới **200%**. Ví dụ, bằng cách sử dụng công cụ Cardumen, $ProdBased_{basic}$ và $SysBased_{basic}$ có thể **sửa chính xác lần lượt là 13 và 0 hệ thống**, trong khi $ProdBased_{enhanced}$ và $SysBased_{enhanced}$ **sửa chính xác lần lượt là 40 và 1 hệ thống**. Bên cạnh đó, hiệu suất sửa lỗi có thể bị ảnh hưởng tiêu cực bởi các công cụ tự động định vị lỗi do các vị trí sửa đổi được chọn dựa trên kết quả định vị lỗi thường không hoàn hảo. Để giảm thiểu tác động của công cụ định vị lỗi, Luận án đánh giá tính hiệu quả của các phương pháp sửa lỗi trong tình huống kết quả định vị lỗi chính xác được cung cấp. Trong thí nghiệm này, Luận án nhận thấy rằng phương pháp *hướng sản phẩm* tốt hơn phương pháp *hướng hệ thống* từ **3 lần** tới **9 lần**. Ngoài ra, các quy tắc được đề xuất giúp tăng **30-150%** số bản vá đúng và giảm **30-70%** số lần thử thay đổi của các phương pháp cơ bản tương ứng.

Chương 6

Kết luận

Các đóng góp của Luận án: Hệ thống SPL đã đạt sự phát triển và chú ý trong ngành công nghiệp phần mềm. Bằng cơ chế cho phép tuỳ chỉnh các cấu hình và tái sử dụng các tính năng của hệ thống, kỹ thuật SPL cho phép các nhà phát triển tạo ra nhiều sản phẩm phù hợp với yêu cầu của từng khách hàng một cách nhanh chóng và dễ dàng. Điều này giúp giảm chi phí và cải thiện hiệu suất của quá trình phát triển phần mềm. Tuy nhiên, do tính biến đổi vốn có của các hệ thống SPL, việc kiểm tra và gỡ lỗi các hệ thống này là rất khó khăn. Mặc dù việc gỡ lỗi tự động trong các hệ thống phần mềm đơn đã được nghiên cứu chuyên sâu, việc gỡ lỗi các hệ thống SPL vẫn chưa được đầu tư nghiên cứu.

Luận án hướng tới mục tiêu đảm bảo chất lượng cho các hệ thống SPL và tự động gỡ lỗi trong các hệ thống này. Luận án tập trung vào ba nhiệm vụ chính: phát hiện *sản phẩm đúng giả*, định vị lỗi biến đổi và sửa chữa lỗi biến đổi. Những đóng góp của Luận án có thể kết luận như sau:

Đầu tiên, *Luận án đề xuất* Clap*, một phương pháp phát hiện sản phẩm đúng giả của hệ thống SPL.* Trong Chương 3, Luận án đã định nghĩa bài toán phát hiện *sản phẩm đúng giả*. Để giải quyết vấn đề này, Luận án đã đề xuất sáu *thuộc tính có thể đo lường* để đánh giá mức độ của các dấu hiệu lỗi trong sản phẩm. Những dấu hiệu này quan tâm đến *mã nguồn* và *chất lượng bộ kiểm thử*; dấu hiệu càng mạnh thì sản phẩm càng có nhiều khả năng là *sản phẩm đúng giả*.

Kết quả thực nghiệm của Luận án cho thấy Clap đạt được hơn 90% *độ chính xác* trong việc phát hiện các *sản phẩm đúng giả* và *sản phẩm đúng thật*. Điều này có nghĩa là trong số 10 sản phẩm được Clap dự đoán là *sản phẩm đúng giả* thì có 9 sản phẩm thực sự là *sản phẩm đúng giả*. Luận án cũng đánh giá khả năng của Clap trong việc giảm thiểu tác động tiêu cực của các *sản phẩm đúng giả* đối với hiệu suất định vị lỗi. Luận án đã tiến hành thực nghiệm nghiệm hai phương pháp định vị lỗi biến đổi tiên tiến nhất với năm chỉ số xếp hạng lỗi phổ biến. Điều thú vị là Clap có thể cải thiện đáng kể hiệu suất của các giải pháp này trong việc xếp hạng các câu lệnh có lỗi lên tới 30%. Điều này cho thấy Clap có thể giảm thiểu

đáng kể tác động tiêu cực của các *sản phẩm đúng giả* trong việc định vị lỗi và giúp các nhà phát triển tìm ra lỗi nhanh hơn. Mã nguồn của CLAP được cung cấp tại: *https://ttrangnguyen.github.io/CLAP/*.

Thứ hai, *Luận án đề xuất* VARCOP*, một cách tiếp cận mới để định vị các lỗi biến đổi.* Chương 4 trình bày những quan sát của Luận án về khả năng hiển thị/ẩn của loại lỗi này trong hệ thống SPL. Luận án đã định nghĩa các điều kiện để hiển thị lỗi biến đổi trong các sản phẩm của hệ thống SPL và giới thiệu các tính chất quan trọng để xác định các điều kiện này. Đối với hệ thống SPL có chứa lỗi biến thể, VARCOP định vị các lỗi bằng cách xác định tập các tính năng cần thiết để thu hẹp không gian tìm kiếm. Sau đó, VARCOP xem xét cả kết quả kiểm thử tổng thể của sản phẩm và kết quả kiểm thử chi tiết của các ca kiểm thử để tìm ra vị trí của lỗi.

Kết quả kiểm thử cho thấy VARCOP vượt trội đáng kể so với các giải pháp khác trong tất cả các độ đo đánh giá. Đối với các trường hợp chứa một câu lệnh lỗi (lỗi đơn), kết quả thực nghiệm cho thấy rằng VARCOP vượt trội hơn đáng kể so với S-SBFL, SBFL và giải pháp của Arrieta trong **tất cả 30/30** độ đo lần lượt là **33%**, **50%** và **95%**. Bên cạnh đó, VARCOP đã xếp hạng chính xác các lỗi ở vị trí Top-3 trong **+65%** trong số các trường hợp thử nghiệm. Ngoài ra, VARCOP đã xếp hạng hiệu quả các câu lệnh có lỗi ở vị trí đầu tiên trong khoảng **30%** trường hợp, điều này **gấp đôi** kết quả của SBFL.

Với hệ thống có nhiều hơn một câu lệnh lỗi, sau khi kiểm tra câu lệnh đầu tiên trong danh sách xếp hạng do VARCOP tạo ra, lập trình viên có thể tìm thấy tới **10%** lỗi trong hệ thống. Kết quả này cao gấp **2 lần** và **10 lần** so với S-SBFL và SBFL. Đặc biệt, kết quả thực nghiệm cũng cho thấy rằng trong **22**% và **65%** các trường hợp, VARCOP đã định vị lỗi hiệu quả ít nhất một câu lệnh lỗi của hệ thống ở vị trí Top-1 và Top-5. Từ đó, các nhà phát triển có thể lặp lại quá trình tìm lỗi, sửa lỗi và kiểm thử hồi quy để nhanh chóng sửa tất cả các lỗi và đảm bảo chất lượng của hệ thống SPL. Mã nguồn của VARCOP được cung cấp tại: *https://ttrangnguyen.github.io/VARCOP/*.

Thứ ba, *Luận án đề xuất phương pháp hướng sản phẩm và hướng hệ thống để tự động sửa các lỗi biến đổi.* Luận án đã giới thiệu các thuật toán chi tiết của hai phương pháp này và các phiên bản nâng cao của chúng với một số quy tắc được đề xuất. Để cải thiện hiệu suất sửa lỗi, các quy tắc của Luận án tận dụng thông tin trung gian thử nghiệm việc sửa lỗi để điều hướng các vị trí sửa đổi và lựa chọn

các thay đổi phù hợp.

Kết quả thực nghiệm cho thấy phương pháp *hướng sản phẩm* tốt hơn đáng kể so với phương pháp *hướng hệ thống* từ **12 đến 30 lần** về số bản vá hợp lệ và khoảng **20 lần** về số bản vá đúng. Điều thú vị là các quy tắc đề xuất của Luận án có thể giúp tăng hiệu suất của cả hai phương pháp *hướng sản phẩm* và *hướng hệ thống* lên tới **200%**. Ví dụ, bằng cách sử dụng công cụ Cardumen, $ProdBased_{basic}$ và $SysBased_{basic}$ có thể **sửa chính xác lần lượt là 13 và 0 hệ thống**, trong khi $ProdBased_{enhanced}$ và $SysBased_{enhanced}$ **sửa chính xác lần lượt 40 và 1 hệ thống**. Hơn nữa, hiệu suất sửa lỗi có thể bị ảnh hưởng tiêu cực bởi các công cụ định vị do các vị trí sửa đổi được chọn dựa trên kết quả định vị thường không hoàn hảo. Để giảm thiểu tác động của công cụ định vị lỗi, Luận án đánh giá tính hiệu quả của các phương pháp sửa lỗi nếu kết quả định vị chính xác được cung cấp. Trong thực nghiệm này, Luận án nhận thấy rằng phương pháp *hướng sản phẩm* tốt hơn phương pháp *hướng hệ thống* về từ **3 lần** đến **9 lần** về độ hiệu quả. Ngoài ra, các quy tắc đề xuất giúp tăng **30-150%** số bản vá đúng và giảm **30-70%** số lần thử sửa đổi của các phương pháp cơ bản tương ứng. Mã nguồn của các công cụ này được cung cấp tại: *https://github.com/ttrangnguyen/SPLRepair*.

Hạn chế của Luận án: Đối với mỗi phương pháp đề xuất, Luận án đã phân tích kỹ sự đóng góp của từng thành phần trong các phương pháp tiếp cận đối với toàn bộ hiệu quả của giải pháp, cũng như mức độ nhạy cảm của các phương pháp tiếp cận với các đầu vào khác nhau để tìm ra điểm yếu của chúng. Có thể kể đến một số hạn chế như:

- Mặc dù tập dữ liệu thực nghiệm là các hệ thống được sử dụng rộng rãi trong các nghiên cứu liên quan, tập dữ liệu này chỉ chứa các lỗi nhân tạo của hệ thống SPL viết bằng ngôn ngữ Java, vì vậy Luận án không thể kết luận các kết quả tương tự đối với các lỗi thực xuất hiện trong thực tế.

- Tất cả các hệ thống bộ dữ liệu thực nghiệm đều được phát triển bằng Java. Vì thế, Luận án không thể khẳng định rằng các kết quả tương tự sẽ được kết luận trong các chương trình được phát triển bằng ngôn ngữ hoặc công nghệ khác.

- Để đảm bảo độ tin cậy của kết quả kiểm thử hệ thống SPL, vấn đề ca kiểm thử có kết quả không ổn định vẫn còn là thách thức và chưa được giải quyết.

Hướng phát triển trong tương lai: Từ những kết quả đạt được trong Luận án cũng như những hạn chế còn tồn tại, Luận án đề xuất một số hướng phát triển trong tương lai.

- *Thu thập các lỗi biến đổi trong thực tế từ các hệ thống SPL lớn hơn để đánh giá các giải pháp đề xuất kỹ lưỡng hơn.* Abal và cộng sự đã thu thập và biểu diễn một bộ dữ liệu gồm 98 lỗi biến đổi trong thế giới thực trong các hệ thống lớn như Linux, Apache , BusyBox và Marlin. Những lỗi này rất cần thiết để đánh giá các công cụ đảm bảo chất lượng phần mềm của hệ thống SPL. Tuy nhiên, hầu hết các lỗi này là lỗi biên dịch và chúng không được cung cấp bộ kiểm thử. Do đó, tập dữ liệu này không phù hợp với các phương pháp sử dụng thông tin kiểm thử như định vị dựa trên phổ, Clap hoặc VarCop, v.v. Trong thực tế, việc thu thập các lỗi biến thể trong thực tế là rất khó khăn. Vì vậy, công việc này đòi hỏi phải phân tích và thiết kế chuyên sâu để có thể thu thập lỗi một cách có hệ thống và tự động. Trong các nghiên cứu trong tương lai, Luận án dự định xây dựng bộ dữ liệu gồm các lỗi trong thực tế bằng cách phân tích và kiểm tra các bản vá lỗi cũng như báo cáo về lỗi từ các hệ thống. Từ những bản và báo cáo này, Luận án có thể truy ngược lại để tìm các sửa đổi gây ra lỗi và sau đó là các phiên bản có lỗi của hệ thống.

- *Mở rộng thực nghiệm với nhiều công cụ tự động sửa lỗi hơn.* Luận án đã đánh giá hiệu suất của việc sửa lỗi biến đổi với jGenProg và Cardumen. Những công cụ này có thể sửa lỗi trong chương trình ở các cấp độ khác nhau, bao gồm cấp độ câu lệnh và biểu thức. Tuy nhiên, có nhiều công cụ tự động sửa lỗi khác, đặc biệt là với sự phát triển của các mô hình ngôn ngữ lớn và trí tuệ nhân tạo, nhiều công cụ tự động sửa lỗi mới đã được giới thiệu. Trong nghiên cứu tiếp theo, Luận án dự định tiến hành nhiều thực nghiệm hơn với các công cụ một cách đa dạng để đánh giá kỹ lưỡng sự đóng góp của các quy tắc đề xuất và mở rộng kết luận của Luận án.

- *Xử lý vấn đề ca kiểm thử có kết quả không ổn định để cải thiện chất lượng của bộ kiểm thử.* Đối với các phương pháp gỡ lỗi sử dụng kết quả kiểm thử, chất lượng của bộ kiểm thử là một yếu tố quan trọng. Các bộ kiểm thử chất lượng thấp có thể dẫn đến cả sự đúng ngẫu nhiên và các vấn đề về kiểm thử không ổn định. Sự đúng ngẫu nhiên dẫn đến việc tính thiếu các ca kiểm thử thất bại và tính quá mức các ca kiểm thử thành công, ảnh hưởng tiêu cực đến hiệu suất

của các phương pháp định vị lỗi. Trong luận án này, CLAP đã được giới thiệu để giải quyết hiện tượng này ở cấp độ sản phẩm. Trong khi đó, các ca kiểm thử có kết quả không ổn định mang lại cả kết quả thành công và kết quả thất bại mặc dù không có thay đổi nào đối với mã nguồn. Sự không đáng tin cậy của kết quả kiểm thử này cung cấp các chỉ dẫn không chính xác cho kỹ thuật định vị lỗi và tự động sửa lỗi. Do đó làm giảm hiệu suất của chúng. Trong tương lai, Luận án dự định phân tích dấu hiệu của các ca kiểm thử không ổn định và thiết kế một phương pháp chuyên biệt để phát hiện các kiểm thử này trong hệ thống SPL.

Danh mục các công bố liên quan

NTT1 . Nguyen, Thu-Trang, Kien-Tuan Ngo, Son Nguyen, and Hieu Dinh Vo. "A variability fault localization approach for software product lines." IEEE Transactions on Software Engineering 48, no. 10 (2021): ISSN 0098-5589, DOI: https://doi.org/10.1109/TSE.2021.3113859, ISI/Q1.

NTT2 . Nguyen, Thu-Trang, and Hieu Dinh Vo. "Detecting Coincidental Correctness and Mitigating Its Impacts on Localizing Variability Faults." In 2022 14th International Conference on Knowledge and Systems Engineering (KSE), pp. 1-6. IEEE, 2022.

NTT3 . Nguyen, Thu-Trang, Kien-Tuan Ngo, Son Nguyen, and Hieu Dinh Vo. "Detecting false-passing products and mitigating their impact on variability fault localization in software product lines." Information and Software Technology 153 (2023): ISSN 0950-5849, volume 153, DOI: https://doi.org/10.1016/j.infsof.2022.107080, ISI/Q1.

NTT4 . Nguyen, Thu-Trang, Xiao-Yi Zhang, Paolo Arcaini, Fuyuki Ishikawa, and Hieu Dinh Vo. "Automated Program Repair for Variability Bugs in Software Product Line Systems." Journal of Systems and Software. ISI/Q1 (accepted).

Danh sách này có 04 công trình.