ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN MAI HƯƠNG

PHÂN TÍCH ĐỘT BIẾN TRONG KIỂM THỬ PHẦN MỀM VÀ ÁP DỤNG TRONG KIỂM THỬ ỨNG DỤNG ANDROID

LUẬN VĂN THẠC SĨ NGÀNH HỆ THỐNG THÔNG TIN

HÀ NỘI - 2019

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN MAI HƯƠNG

PHÂN TÍCH ĐỘT BIẾN TRONG KIỂM THỬ PHẦN MỀM VÀ ÁP DỤNG TRONG KIỂM THỬ ỨNG DỤNG ANDROID

Chuyên ngành: Hệ Thống Thông Tin
Mã số: 60480104

LUẬN VĂN THẠC SĨ NGÀNH HỆ THỐNG THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS. TS. HÀ QUANG THỤY

HÀ NỘI - 2019

LỜI CẢM ƠN

Lời đầu tiên, tôi xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới PGS. TS Hà Quang Thụy, đã tận tình hướng dẫn và chỉ bảo tôi trong suốt quá trình thực hiện luận văn tốt nghiệp.

Tôi xin chân thành cảm ơn các thầy, cô trong trường đại học Công Nghệ - đại học Quốc gia Hà Nội đã cho tôi nền tảng kiến thức tốt và tạo mọi điều kiện thuận lợi cho tôi học tập và nghiên cứu.

Tôi cũng xin gửi lời cảm ơn đến các thầy cô, các anh chị và các bạn trong phòng thí nghiệm DS&KTLab đã hỗ trợ tôi rất nhiều về kiến thức chuyên môn trong quá trình thực hiện luận văn. Tôi xin cảm ơn tất cả mọi ngườiđã ủng hộ và khuyến khích tôi trong suốt suốt quá trình học tập tại trường.

Cuối cùng, tôi xin được gửi lời cám ơn vô hạn tới gia đình và bạn bè, những người đã luôn bên cạnh, giúp đỡ và động viên tôi trong quá trình học tập cũng như trong suốt quá trình thực hiện luận văn.

Tôi xin chân thành cảm ơn!

Hà Nội, ngày tháng năm 2019

Học viên

Nguyễn Mai Hương

PHÂN TÍCH ĐỘT BIẾN TRONG KIỂM THỬ PHẦN MỀM VÀ ÁP DUNG TRONG KIỂM THỬ ỨNG DỤNG ANDROID

Nguyễn Mai Hương

Khóa K23, chuyên ngành Hệ Thống Thông Tin

Tóm tắt Luận văn tốt nghiệp:

Hiện nay, do việc sử dụng rộng rãi các thiết bị Android, các ứng dụng của Android có nhiều phiên bản, có thể tải xuống các ứng dụng cho bất kỳ thiết bị di động nào. Nên tạo ra mối lo ngại về chất lượng của phần mềm. Vì vậy việc kiểm thử để nâng cao chất lượng phần mềm là vấn đề thiết yếu.

Tuy nhiên, việc kiểm thử các ứng dụng Android không được kiểm thử từ các chương trình Java truyền thống do tính độc đáo cấu trúc chương trình và các tính năng mới của ứng dụng. Các nhà phát triển phần mềm cho thấy sự quan tâm không nhỏ trong việc phát triển Android tốt hơn tuy nhiên vẫn còn thiếu các kỹ thuật kiểm thử và có thể sử dụng để đánh giá các chiến lược kiểm thử.

Kiểm thử đột biến là một phương pháp kiểm thử phần mềm trong đó chương trình hoặc mã nguồn được thay đổi có chủ ý. Khi một chương trình được sửa đổi, nó sẽ thực thi chương trình theo đột biến vừa được tạo ra, để kiểm tra xem phần mềm có thực hiện đúng hành vi của mình hay không. Chính vì vậy kiểm thử đột biến giúp phân tích xem một tập hợp các chiến lược kiểm thử có đủ để đảm bảo rằng sản phẩm đáp ứng các yêu cầu chất lượng hay không.

Trong luận văn này, tôi tìm hiểu những kiến thức về kiểm thử phần mềm [9] và phân tích nghiên cứu về các phương pháp, kỹ thuật kiểm thử đột biến [1-7]. Từ đó áp dụng những kỹ thuật trên vào kiểm thử ứng dụng trên Android.

Từ khóa: Android, Kiểm thử phần mềm, Kiểm thử đột biến

LỜI CAM ĐOAN

Tôi xin cam đoan phân tích đột biến trong kiểm thử phần mềm và áp dụng trong kiểm thử ứng dụng Androidvà thực nghiệm được trình bày trong luận văn là do tôi đề ra và thực hiện dưới sự hướng dẫn của PGS. TS Hà Quang Thụy.

Tất cả các tài liệu tham khảo từ các nghiên cứu liên quan đều có nguồn gốc rõ ràng từ danh mục tài liệu tham khảo trong luận văn. Trong luận văn, không có việc sao chép tài liệu, công trình nghiên cứu của người khác mà không chỉ rõ về tài liệu tham khảo.

Hà Nội, ngày tháng năm 2019

Học viên

Nguyễn Mai Hương

MỤC LỤC

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG BIỂU

DANH SÁCH CÁC TỪ VIẾT TẮT

STT	Tên viết tắt	Cụm từ đầy đủ
1	XML	Extensible Markup Language
2	IDE	Integrated Development Environment
3	APK	Android application package
4	IPR	Intent Payload Replacement
5	ITR	Intent Target Replacement
6	MDL	Lifecycle Method Deletion
7	ECR	OnClick Event Replacement
8	ETR	OnTouch Event Replacement
9	BWD	Button Widget Deletion
10	TWD	EditText Widget Deletion
11	BWS	Button Widget Switch

MỞ ĐẦU

Android mobile là một chương trình phần mềm chạy trên thiết bị di động chẳng hạn như điện thoại thông minh hoặc máy tính bảng. Số lượng ứng dụng di động được phát triển nhiều khi nền tảng trở nên có sẵn, và chất lượng là một vấn đề nghiêm trọng đang gia tăng. Các ứng dụng Android được xây dựng hoàn toàn từ phần mềm truyền thống và được sử dụng cấu trúc, kết nối dữ liệu kiểu mới [1]. Cấu trúc của ứng dụng di động khác với cấu trúc các loại phần mềm khác, vì vậy việc kiểm thử cũng sẽ khác nhau nên cần có các phương pháp khác nhau để kiểm thử phần mềm trên di động. Phương pháp tiếp cận là phương pháp để kiểm thử các ứng dụng di động [2]và kiểm thử đột biến thường được sử dụng như một cách để đánh giá mức độ đầy đủ của các trường hợp khi kiểm thử [8].

Khi kiểm thử được tất cả các trường hợp xảy ra sẽ giảm thiểu được rủi ro khi bàn giao sản phẩm cũng như dễ dàng trong việc vận hành, bảo trì phần mềm trong giai đoạn sau. Từ đó cải thiện được chất lượng phần mềm. Sản phẩm được phát triển theo yêu cầu của khách hàng, đáp ứng mục tiêu mong muốn đồng thời đảm bảo chất lượng cao và đáng tin cậy hơn.

Trong luận văn này tôi áp dụng một số kỹ thuật kiểm thử đột biến hiện có cho phần mềm, ứng dụng di động sử dụng nền tảng Android. Chi tiết về phương pháp và các kỹ thuật này sẽ được trình bày chi tiết trong luận văn này. Nội dung của luận văn chia thành các chương như sau:

Chương 1: Luận văn trình bày về ứng dụng Android và bài toán kiểm thử ứng dụng trên thiết bị di động.

Chương 2: Luận văn trình bày về phương pháp và các kỹ thuật toán tử đột biến trong kiểm thử ứng dụng Android trên thiết bị di động.

Chương 3: Luận văn trình bày về mô hình đề nghị để áp dụng kiểm thử.

Chương 4: Luận văn trình bày thực nghiệm cho việc kiểm thử ứng dụng Android.

Chương 1. ỨNG DỤNG ANDROID VÀ BÀI TOÁN KIỂM THỬ ỨNG DỤNG TRÊN THIẾT BỊ DI ĐỘNG

Trong chương này, chúng tôi giới thiệu chung về những kiến thức kiểm thử phần mềm chung nhất [10], khái niệm ứng dụng Android trên thiết bị di động, những thách thức trong kiểm thử ứng dụng trên Android, phương pháp toán tử đột biến trong kiểm thử phần mềm và ý nghĩa của bài toán.

1.1. Giới thiệu chung về kiểm thử phần mềm

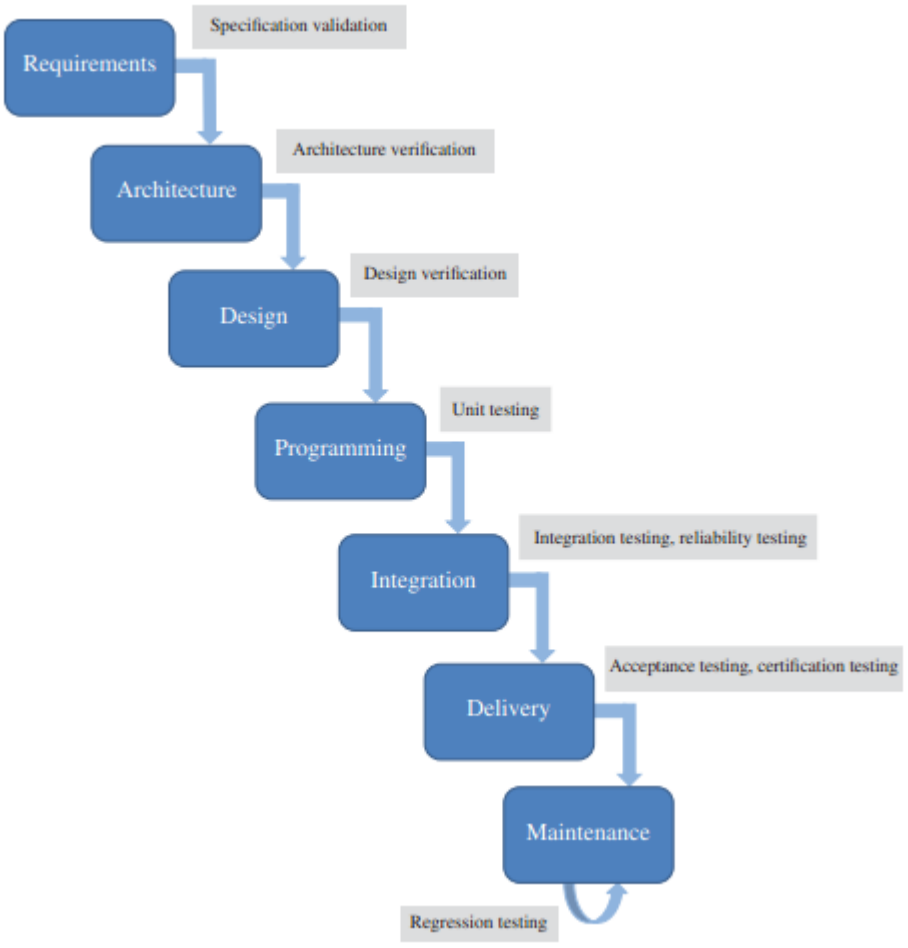
Kiểm thử [10] là quá trình thực hiện một sản phẩm phần mềm trên dữ liệu đầu vào mẫu và phân tích đầu ra của nó. Sản phẩm phần mềm không giống như các sản phẩm kỹ thuật khác, thường không có lỗi, hoặc có rất ít lỗi. Nhưng các sản phẩm phần mềm dễ bị lỗi, do sự tích lũy các lỗi trong tất cả các giai đoạn trong vòng đời của chúng (thông số kỹ thuật bị lỗi, thiết kế bị lỗi, thực hiện bị lỗi, v.v.). Ngoài ra, không giống như các sản phẩm kỹ thuật khác, sản phẩm phần mềm còn có lỗi do hao mòn sản phẩm trong quá trình sử dụng. Lỗi phát sinh trong các sản phẩm phần mềm hầu hết là lỗi thiết kế, được cung cấp cùng với sản phẩm mới.

Kiểm thử phần mềm cũng cung cấp mục tiêu, cái nhìn độc lập về phần mềm, điều này cho phép việc đánh giá và hiểu rõ các rủi ro khi thực thi phần mềm. Kiểm thử phần mềm tạo điều kiện cho kiểm thử viên tận dụng tối đa tư duy đánh giá và sáng tạo để có thể phát hiện ra những điểm mà người khác chưa nhìn thấy. Vì mỗi kiểm thử viên ngoài các cách nhìn chung và tổng quát, từng kiểm thử viên còn có cách nhìn của riêng mình. Điều này làm cho từng kiểm thử viên có thể phát hiện được nhiều lỗi ở các góc nhìn khác nhau.

Mục đích của kiểm thử phần mềm là tìm ra lỗi chưa được phát hiện, và tìm một cách sớm nhất để bảo đảm rằng lỗi sẽ được sửa. Mục tiêu của kiểm thử phần mềm là thiết kế tài liệu kiểm thử một cách có hệ thống như thiết kế các trường hợp kiểm thử và đi cụ thể chi tiết từng trường hợp kiểm thử. Từ đó thực hiện nó sao cho có hiệu quả, nhưng tiết kiệm được thời gian, công sức và chi phí.

1.1.1. Vòng đời phát triển của phần mềm

Vòng đời của phát triển phần mềm được trải qua các giai đoạn tiến hành tuần tự từ giai đoạn phân tích yêu cầu đến giai đoạn vận hành và bảo trì sản phẩm như Hình 1. 1[10].



Hình 1.1 Các giai đoạn phát triển của phần mềm

- Requirements (phân tích yêu cầu): Phân tích yêu cầu là quan trọng nhất trong các giai đoạn của vòng đời phát triển phần mềm. Ở giai đoạn này, người phân tích thiết kế sẽ xác định yêu cầu cụ thể của phần mềm. Từ đó cho phép người sử dụng và các lập trình viên, kiểm thử viên có cái nhìn tổng quát nhất về phần mềm.
- Architecture (cấu trúc phần mềm): Trong giai đoạn này người thiết kế và phân tích sẽ xác định thành phần bên trong và mô tả chi tiết các thành phần bên trong

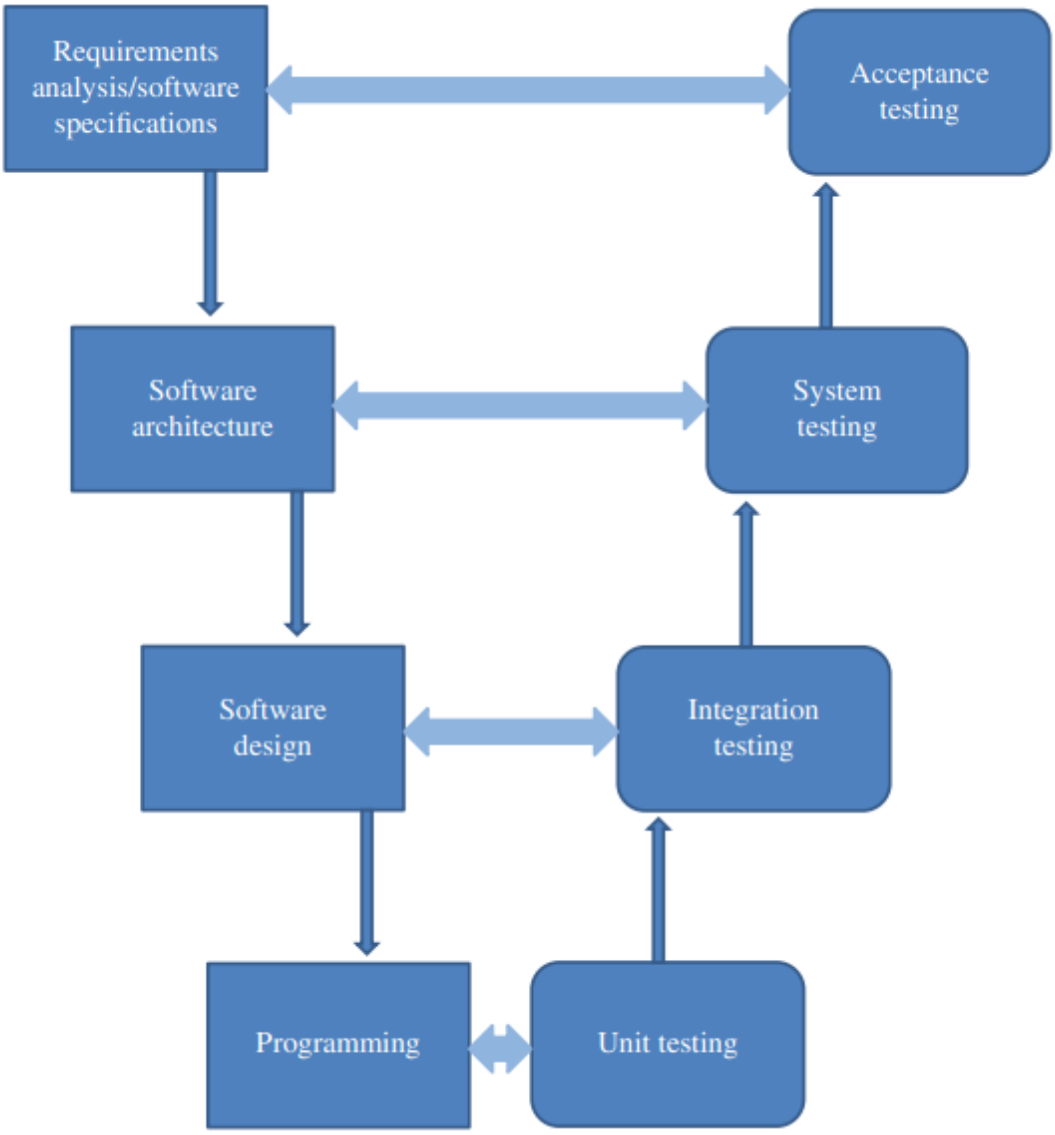
của phần mềm, cũng như cách hệ thống sẽ được triển khai, cách chia sẻ dữ liệu, v.v.

- Design (thiết kế phần mềm): Trong giai đoạn thiết kế này người thiết kế và phân tích sẽ đưa ra các quyết định thiết kế toàn bộ hệ thống. Các thiết kế liên quan đến cấu trúc dữ liệu, hiển thị dữ liệu, các thuật toán và chia tách các chức năng phần mềm thành các đơn vị nhỏ để các lập trình viên phát triển độc lập.
- Programming (lập trình): Giai đoạn lập trình được thực hiện bởi các lập trình viên dựa trên yêu cầu thiết kế và kiến trúc phần mềm đã được xác định trước đó, Trong giai đoạn này, các kiểm thử viên đã có thể thiết kế các trường hợp kiểm thử đơn vị để sử dụng kiểm thử đơn vị để kiểm thử trong giai đoạn này.
- Integration (tích hợp): Khi tất cả các đơn vị của phần mềm đã được phát triển, các đơn vị hoặc các chức năng sẽ được tích hợp với nhau để tạo nên một phần mềm hoàn chỉnh. Trong giai đoạn này, các kiểm thử viên sẽ thiết kế các trường hợp kiểm thử tích hợp và sử dụng kiểm thử tích hợp để kiểm thử trong giai đoạn này.
- Delivery (bàn giao sản phẩm) : Khi phần mềm đã trải qua kiểm thử tích hợp thì các kiểm thử viên sẽ kiểm thử chấp nhận. Kiểm thử chấp nhận là kiểm thử toàn bộ phần mềm. Kiểm thử trong giai đoạn này có thể được thiết kế các trường hợp kiểm thử với góc nhìn của người sử dụng. Khi kiểm thử chấp nhận được hòa thành thì sản phẩm đã có thể được bàn giao.
- Maintenance (vận hành và bảo trì) : Giai đoạn vận hành bảo trì là giai đoạn cuối cùng trong vòng đời phát triển phần mềm. Nếu trong giai đoạn vận hành, sản phẩm phần mềm bị lỗi hoặc có yêu cầu thay đổi, thì thao tác bảo trì được bắt đầu để thay đổi. Khi hoàn thành thao tác bảo trì, hệ thống phần mềm hoạt động bảo trì có thể chỉ liên quan đến một phần giới hạn của mã nguồn hoặc chỉ tập trung vào một vài chức năng hệ thống. Kiểm thử trong giai đoạn này được gọi là kiểm thử hồi quy.

1.1.2. Vòng đời của kiểm thử phần mềm

Hiện nay có rất nhiều mô hình được đưa ra trong phát triển phần mềm. Trong phần này sẽ trình về vòng đời của kiểm thử phần mềm cụ thể ở đây là vòng đời kiểm thử phần mềm của mô hình chữ V. Mặc dù kiểm thử thường được coi là một pha, và là pha cuối cùng của vòng đời phần mềm. Nhưng nó thực sự được xem là hoạt động diễn

ra đồng thời trong tất cả các giai đoạn của vòng đời, từ đầu đến cuối. Trong mô hình chữ V, việc minh họa bản chất của kiểm thử là một hoạt động đang diễn ra trong suốt vòng đời phần mềm và việc kiểm thử có thể được lên kế hoạch từng bước một, mỗi cặp pha được kết nối bởi một mũi tên nằm ngang trong Hình 1. 2 [10]và có mối liên hệchặt chẽ với nhau. Trong mô hình chữ V pha hiệu chỉnh bên nhánh bên phải, và cái sau kiểm tra tính hợp lệ của cái trước.



Hình 1.2 Kiểm thử phần mềm tương ứng với mô hình chữ V

- Unit testing (kiểm thử đơn vị): Unit test diễn ra trong quá trình lập trình được lập trình viên hoặc kiểm thử viên thực hiện. Trong pha này đơn vị là phần có thể kiểm thử nhỏ nhất như hàm hoặc các thành phần đơn lẻ.

- Integration testing (kiểm thử tích hợp): Kiểm thử tích hợp là phương pháp tích hợp các thành phần cơ sở, sau đó bổ sung thêm các thành phần chức năng được bắt đầu từ đơn vị thấp nhất hoặc trong cùng của ứng dụng và dần dần di chuyển lên trên. Kiểm thử tích hợp được bắt đầu từ module thấp nhất và dần dần tiến tới các module cao hơn của ứng dụng. Sự tích hợp này tiếp tục cho tới khi tất cả các module được kiểm thử hoàn tất.
- System testing (kiểm thử hệ thống): Kiểm thử hệ thống cho phép kiểm thử viên theo dõi và đánh giá hành vi của hệ thống, hệ thống được hoàn chỉnh và tích hợp đầy đủ các chức năng như yêu cầu chức năng đã được xác định trước.
- Acceptance testing (kiểm thử chấp nhận): Kiểm thử chấp nhận thường được diễn ra cuối cùng và xác nhận lại sự tin tưởng của hệ thống, các đặc tính thuộc về chức năng hoặc phi chức năng của hệ thống. Ngoài ra kiểm thử chấp nhận thường được kiểm thử viên nhìn nhận và đánh giá phần mềm dưới góc nhìn của người sử dụng. Vì vậy những lỗi phát hiện ra ở giai đoạn này thường mang ý nghĩa quan trọng.

1.2. Ứng dụng Android trên thiết bị di động và kiểm thử ứng dụng Android

1.2.1. Ứng dụng Android trên thiết bị di động

Hiện nay khi các ứng dụng di động ngày càng được sử dụng rộng rãi và các nhà phát triển cho phép người dùng có thể kết nối với nhau theo nhiều cách, thì công việc của các nhà phát triển ứng dụng ngày càng trở nên quan trọng trong một nền kinh tế toàn cầu. Các ứng dụng di động mà chúng ta sử dụng hàng ngày đã thay đổi cách chúng ta tiến hành kinh doanh, giao tiếp, giải trí, làm việc và học thêm những điều mới.

Các thiết bị và ứng dụng Android hiện nay đang chiếm phần lớn trên thị trường. Đem lại nhiều tính ứng dụng cao trong đời sống và cho người sử dung. Hệ điều hành Android không chỉ là hệ điều hành trên điện thoại di động được sử dụng rộng rãi nhất, mà nhiều ứng dụng đã được phát hành và tải xuống cho Android nhiều hơn bất kỳ hệ điều hành nào khác. Android là một trong các hệ điều hành được ưa chuộng nhất hiện nay. Với ưu thế là mã nguồn mở và được đông đảo cộng đồng yêu thích, Android đã thu hút rất nhiều nhà phát triển từ khắp mọi nơi trên thế giới và đang dần khẳng định vị thế.

Nhờ Android mà hàng loạt các ứng dụng games, ứng dụng di động gia tăng một cách nhanh chóng. Khi các ứng dụng di động ngày càng được sử dụng rộng rãi và cho phép người dùng có thể kết nối với nhau theo nhiều cách, thì chất lượng phần mềm lại được đặt lên hàng đầu. Vì thế để giải quyết bài toán chất lượng, thì kiểm thử phần mềm đang và đã được các nhà phát triển, các lập trình viên và các kiểm thử viên quan tâm.

Hình 1.3 Minh họa các ứng dụng Android trên Google Play

1.2.2. Kiểm thử ứng dụng Android

Các thiết bị và ứng dụng Android hiện nay đang chiếm phần lớn trên thị trường. Đem lại nhiều tính ứng dụng cao trong đời sống và cho người sử dung. Vì vậy kiểm thử ứng dụng Android trên thiết bị di động là cần thiết. Và phân tích đột biến là một phương pháp kiểm thử phần mềm tiềm năng, đặc biệt đối với việc kiểm thử các ứng dụng trên Android. Như trong tài liệu [1-7] đã đưa ra những toán tử đột biến để kiểm thử ứng dụng Android [1]. Và dẫn đến hiệu quả của việc phân tích đột biến trong ứng dụng Android được đề cập trong [2]. Sử dụng đột biến để thiết kế kiểm thử cho các mô hình định hướng theo khía cạnh được nói đến trong tài liệu [3]. Các mô hình định hướng khía cạnh để kiểm thử xuyên suốt được đưa ra ở tài liệu[4]. Bên cạnh đó kiến trúc phát triển của các toán tử đột biến ở tài liệu [6]. Và xu hướng kiểm thử phần mềm trong tài liệu[7].

Do việc sử dụng rộng rãi các thiết bị Android, Android application (ứng dụng) có nhiều phiên bản phát hành, khách hàng và người sử dụng có thể mua hàng và tải ứng dụng cho bất kỳ thiết bị di động nào [1]. Đề xuất việc sử dụng các đột biến theo định hướng các mô hình để kiểm tra mối quan tâm xuyên suốt. Trong kiểm thử đột biến, một phần mềm tạo ra như một chương trình hoặc một mô hình được sửa đổi luân phiên, những chương trình hoặc mô hình này thường gặp rất nhiều lỗi, và các phiên bản được gọi là đột biến [4]. Tổng số ứng dụng Android có sẵn trong cửa hàng Google Play đã vượt quá 2, 8 triệu ứng dụng. Tuy nhiên, chất lượng ứng dụng Android là một vấn đề nghiêm trọng và cần phải được giải quyết. Nghiên cứu cho thấy nhiều ứng dụng có lỗi nghiêm trọng đã dẫn đến việc gặp lỗi thường xuyên trong quá trình sử dụng[2]. Ý tưởng về kiểm thử đột biến là bắt buộc các nhà phát triển phải thiết kế các trường hợp kiểm thử để khám phá các hành vi hệ thống mà để phát hiện ra các lỗi thường gặp. Các loại lỗi khác nhau mà người ta có thể sử dụng có thể tìm ra, dẫn đến các trường hợp kiểm thử với các thuộc tính khác nhau. Chúng được thiết kế lại và có sai số về lập trình, ranh giới về điều kiện kiểm thử, và mô phỏng các mục tiêu kiểm thử của các tiêu chí để kiểm tra các kết cấu và kiến trúc khác nhau của phần mềm.

Nói chung, phương pháp này có đủ khả năng để nó có thể được điều chỉnh kiểm thử hầu hết mọi trường hợp mà các nhà phát triển muốn kiểm tra [8]. Từ đó ta có thể thấy kiểm thử phần mềm như một phần không thể tách rời trong quy trình đảm bảo chất lượng phần mềm. Vì vậy có thể xem kiểm thử phần mềm như một phần của chiến lược toàn diện về đảm bảo chất lượng phần mềm [9] [10].

1.3. Những thách thức trong kiểm thử Android

Đối với kiểm thử android, mặc dù một phần mã nguồn của ứng dụng Android được viết bằng Java, nhưng khi kiểm thử ứng dụng Android sẽ có những thách thức đòi hỏi kiểm thử viên phải nắm rõ để vượt qua. Dưới đây là những thách thức mà các kiểm thử viên phải nắm rõ được.

1. Vòng đời độc đáo của các thành phần Android: Tất cả các thành phần chính của ứng dụng Android cần phải hoạt động theo vòng đời được chỉ định trước. Nếu xử lý không đúng cách vòng đời của một thành phần đang hoạt động có thể gây ra sự cố không mong muốn. Ví dụ khi phát triển ứng dụng chơi trò chơi, việc xử lý liên tục các sự kiện xảy ra rất quan trọng đối với người dùng. Khi đang có cuộc gọi đến, người dung nghe và kết thúc

cuộc gọi thì trò chơi có được tiếp tục hay không hoặc nếu trò chơi không được tiếp tục diễn ra thì phần mềm sẽ xử lý hành động gì tiếp theo.

2. Sử dụng ngôn ngữ đánh dấu XML: Các chương trình Java truyền thống với các giao diện người dùng đồ họa hiếm khi sử dụng các tệp XML. Ngoài ra, không có tiêu chí về phạm vi kiểm thử để đo mức độ bao phủ cho các tệp XML. Rõ ràng, việc không kiểm tra các tệp XML của ứng dụng Android có thể dẫn đến các lỗi không mong muốn. Vì vậy kiểm thử viên cần phải chú ý đến các tệp XML trong quá trình kiểm thử.
3. Đặc điểm về nhận thức ngữ cảnh: Ứng dụng Android là ứng dụng nhận thức ngữ cảnh bởi vì nó nhận được đầu vào từ các thiết bị vật lý. Những loại dữ liệu này không được người dùng trực tiếp cung cấp, nhưng chúng được phân loại thông qua các cảm biến như gia tốc kế, cảm biến nhiệt độ, cảm biến môi trường và cảm biến từ trường, v.v. Các kỹ thuật kiểm thử hiện tại không xem xét các loại đầu vào này nhưng dữ liệu được lấy từ các đầu vào này lại có thể ảnh hưởng đến phần mềm khi hoạt động và khi kiểm thử.
4. Màn hình trong thiết bị di động: Thông thường các thiết bị di động trên Android có hai loại hướng màn hình là màn hình ngang và màn hình dọc. Nhiều ứng dụng Android có giao diện người dùng khác nhau để thích ứng với từng sự kiện khi chuyển màn hình từ ngang sang dọc hoặc ngược lại. Vì vậy, khi kiểm thử ứng dụng Android, kiểm thử viên cần phải xem xét tính năng này, bởi vì nó rất có thể gây ra lỗi khi chuyển màn hình từ ngang sang dọc.
5. Ứng dụng Android là ứng dụng chạy các chương trình dựa trên các sự kiện, vì vậy luồng thực thi của chúng phụ thuộc vào hành động của người dùng, chẳng hạn như khi nhấp hoặc nhấn vào một nút nào đó. Các sự kiện này được xử lý bởi các trình xử lý sự kiện khác nhau. Ngoài ra, mọi thiết bị Android đều được trang bị ba nút hệ thống trên màn hình: Back, Home và Recents. Nên các nút hệ thống này có thể làm gián đoạn luồng thực thi thông thường và thường không nằm trong luồng thực thi thường được mong đợi, vì thế kiểm thử viên không kiểm thử tác động của các nút hệ thống.

6. Hầu hết các thiết bị Android được trang bị nhiều dạng kết nối mạng, kết nối phổ biến nhất là kết nối dữ liệu di động và kết nối WiFi. Bất cứ khi nào có kết nối WiFi thì hệ thống Android sẽ cố gắng truyền dữ liệu qua WiFi trước. Nếu không có WiFi, hệ thống Android sẽ cố gắng chuyển sang kết nối dữ liệu di động, kết nối di động sẽ tốn kém chi phí hơn và sử dụng nhiều pin hơn. Việc chuyển đổi này có thể gây ra một số lỗi khi đang chạy ứng dụng trên Android.
7. Hầu hết các thiết bị di động có nguồn pin thực sự hạn chế. Và có những ứng dụng làm tăng mức tiêu thụ năng lượng pin trên Android vì vậy dẫn đến có ảnh hưởng đến thiết bị trong quá trình sử dụng.

1.4. Khái niệm và mô hình kiểm thử dựa trên đột biến

Kiểm thử đột biến là kiểm thử dựa trên cú pháp giúp người kiểm tra thiết kế của phần mềm một cách hiệu quả và đạt được chất lượng cao. Kiểm thử viên có thể chèn lỗi vào chương trình đang kiểm thử để kiểm thử hoạt động của chương trình.

Mô hình định hướng theo khía cạnh [4] là cách tiếp cận được đề xuất để tránh các mô hình hành vi quá phức tạp, sử dụng mô hình định hướng định hướng cho các mối quan tâm xuyên suốt. Một mối quan tâm xuyên suốt được áp dụng trong toàn bộ quá trinh phần mềm và có rất quan trọng đối với độ tin cậy, hiệu suất, bảo mật của hệ thống. Các ví dụ điển hình bao gồm các sự kiện đòi hỏi sự chú ý ngay lập tức. Mối quan tâm xuyên suốt có xu hướng đảo lộn các mô hình, dẫn đến các mô hình phức tạp và khó phân tích. Trong mô hình định hướng theo khía cạnh, mối quan tâm xuyên suốt được mô hình hóa theo mô hình aspects, được tách ra khỏi hành vi thông thường, do đó tạo ra mô hình định hướng aspect-oriented model (AOM). Ý tưởng chung với AOM là mô hình hóa hành vi bình thường của hệ thống trong mô hình cơ sở, để lại các mối quan tâm xuyên suốt được mô tả trong các mô hình khía cạnh riêng biệt. Bằng cách này mô hình hóa các mối quan tâm một cách riêng biệt và sau đó tự động áp dụng vào mô hình sau đó, các mô hình sau đó trở nên sạch hơn và ít lộn xộn hơn.

Phát triển dựa trên mô hình đang được sử dụng rộng rãi trong ngành công nghiệp phần mềm ngày nay. Các mô hình cung cấp một cái nhìn trực quan về phần mềm. Ngoài ra, một số loại mô hình nhất định, ví dụ như biểu đồ trạng thái, lưới Petri và tự động hóa thời gian rất hữu ích cho mục đích phân tích các mô hình. Hơn nữa, các mô

hình hành vi có thể được sử dụng để tạo các trường hợp kiểm thử bao trùm phần mềm. Do đó, mô hình hóa hành vi phần mềm có thể giúp các nhà phát triển hiểu và phân tích những hành vi phức tạp bên trong của hệ thống. Sự mô tả rõ rang của một mô hình thường phụ thuộc vào mức độ chi tiết mà nó được mô hình hóa. Tuy nhiên, các mô hình chi tiết cũng thường phức tạp hơn. Vì vậy, các mô hình chi tiết thường khó hiểu và khó để duy trì được. Việc sử dụng kiểm thử đột biến theo định hướng mô hình để kiểm tra mối quan tâm xuyên suốt. Trong kiểm thử đột biến, một phần mềm như chương trình hoặc mô hình được sửa đổi để tạo ra các phiên bản thay thế, thường bị lỗi, được gọi là các trình biến đổi được áp dụng một cách có hệ thống, là các quy tắc để thay đổi các yếu tố cú pháp được thiết kế để khiến các đột biến có hành vi khác với phiên bản gốc gọi là hủy các đột biến. Các toán tử đột biến bắt chước các lỗi lập trình điển hình hoặc thực hiện các thay đổi và khuyến khích người thử nghiệm thiết kế các đầu vào có giá trị đặc biệt.

1.5. Về phương pháp toán tử đột biến trong kiểm thử phần mềm

Phần này giới thiệu về các phương pháp toán tử đột biến đã được L. Deng và cộng sự trình bày [2]. Kiểm thử đột biến được bắt đầu từ năm 1978 và đây là một kỹ thuật kiểm tra dựa trên cú pháp giúp người kiểm tra thiết kế một cách hiệu quả kiểm tra chất lượng cao, được coi là một tiêu chí hiệu quả vượt trội cho cả thiết kế và đánh giá thử nghiệm. Quá trình chung kiểm thử đột biến bao gồm các bước được mô tả như sau đây.

Khi tiến hành thực thi kiểm thử lần lượt chương trình gốc P và đột biến P’ của P với một dữ liệu kiểm thử T, sẽ có hai kịch bản khác nhau có thể xảy ra:

- Một là, đột biến P’ được gọi là bị diệt bởi dữ liệu kiểm thử T.

- Hai là, chương trình gốc P và đột biến P’ cho ra kết quả hoàn toàn giống nhau, đột biến P’ được cho là còn sống.

Các đột biến tương đương là các đột biến của chương trình gốc nhưng hoạt động hoàn toàn giống với chương trình gốc và cho ra kết quả giống với chương trình gốc trong mọi trường hợp kiểm thử. Việc sử dụng đột biến của các mô hình định hướng theo khía cạnh để kiểm tra các mối quan tâm xuyên suốt. Trong kiểm thử đột biến, một phần mềm như chương trình hoặc mô hình được sửa đổi để tạo ra các phiên bản thay thế, thường bị lỗi, được gọi là đột biến. Các đột biến được tạo ra bằng cách áp dụng

một cách có hệ thống các toán tử đột biến, đó là các quy tắc để thay đổi các yếu tố cú pháp.

Các kiểm thử sau đó được thiết kế để khiến các đột biến có hành vi khác với phiên bản gốc, được gọi là hủy các toán tử đột biến. Việc này khuyến khích các kiểm thử viên thiết kế các đầu vào kiểm thử có giá trị đặc biệt. Điểm tương xứng đột biến là một tiêu chí, như độ bao phủ của câu lệnh và luồng dữ liệu, nhưng được phát hiện là mạnh hơn các tiêu chí đã biết khác và do đó thường được gọi là tiêu chuẩn vàng của đột biến và là tiêu chí quan trọng nhất trong số các tiêu chí. Chương 2 sẽ giới thiệu chi tiết hơn về các toán tử đột biến [1].

Tóm tắt chương 1

Trong chương 1, luận văn đã trình bày tổng quan về kiểm thử phần mềm, ứng dụng Android trên thiết bị di động và phương pháp toán tử đột biến trong kiểm thử phần mềm. Luận văn cũng đã nêu lên được vấn đề tại sao nên sử dụng kiểm thử và cụ thể ở đây là kiểm thử đột biến. Ngoài ra, luận văn cũng đã nêu lên ý nghĩa của bài toán này.

Chương tiếp theo trong luận văn này sẽ chi tiết hóa nền tảng kiến thức liên quan về kiểm thử đột biến và kiểm thử đột biến trên ứng dụng Android. Đồng thời các kỹ thuật thực hiện kiểm thử đột biến sẽ được trình bày chi tiết ở những chương sau.

Chương 2. KỸ THUẬT TOÁN TỬ ĐỘT BIẾN TRONG KIỂM THỬ ỨNG DỤNG ANDROID TRÊN THIẾT BỊ DI ĐỘNG

Chương 2 luận văn tập trung khảo sát các kỹ thuật toán tử đột biến để kiểm thử ứng dụng Android của L. Deng và cộng sự [1] bao gồm các kỹ thuật toán tử đột biến ý định, vòng đời hoạt đông của toán tử đột biến, xử lý toán tử đột biến và toán tử đột biến XML, v.v. Các kỹ thuật này sẽ được sử dụng trong mô hình thực nghiệm của luận văn.

2.1. Phương pháp toán tử đột biến trong kiểm thử phần mềm

Việc sử dụng đột biến của các mô hình định hướng theo khía cạnh để kiểm tra các mối quan tâm xuyên suốt của phần mềm. Trong kiểm thử đột biến, một chương trình phần mềm hoặc mô hình được sửa đổi để tạo ra các phiên bản thay thế, thường bị lỗi, được gọi là đột biến. Các đột biến được tạo ra bằng cách áp dụng một cách có hệ thống các toán tử đột biến đó là các quy tắc để thay đổi các yếu tố cú pháp. Theo đó toán tử đột biến được chèn vào trong hệ thống đang kiểm thử đã dẫn đến ý tưởng và phát triển cho nhiều loại cấp độ thử nghiệm, nhiều loại ngôn ngữ lập trình khác nhau. Các công trình đầu tiên về kiểm thử đột biến nhắm vào các chức năng của chương trình Fortran. Fortran là một ngôn ngữ lập trình biên dịch tĩnh kiểu mệnh lệnh được phát triển từ thập niên 1950 và vẫn được dùng nhiều trong tính toán khoa học hay phương pháp số cho đến hơn nửa thế kỷ sau đó.

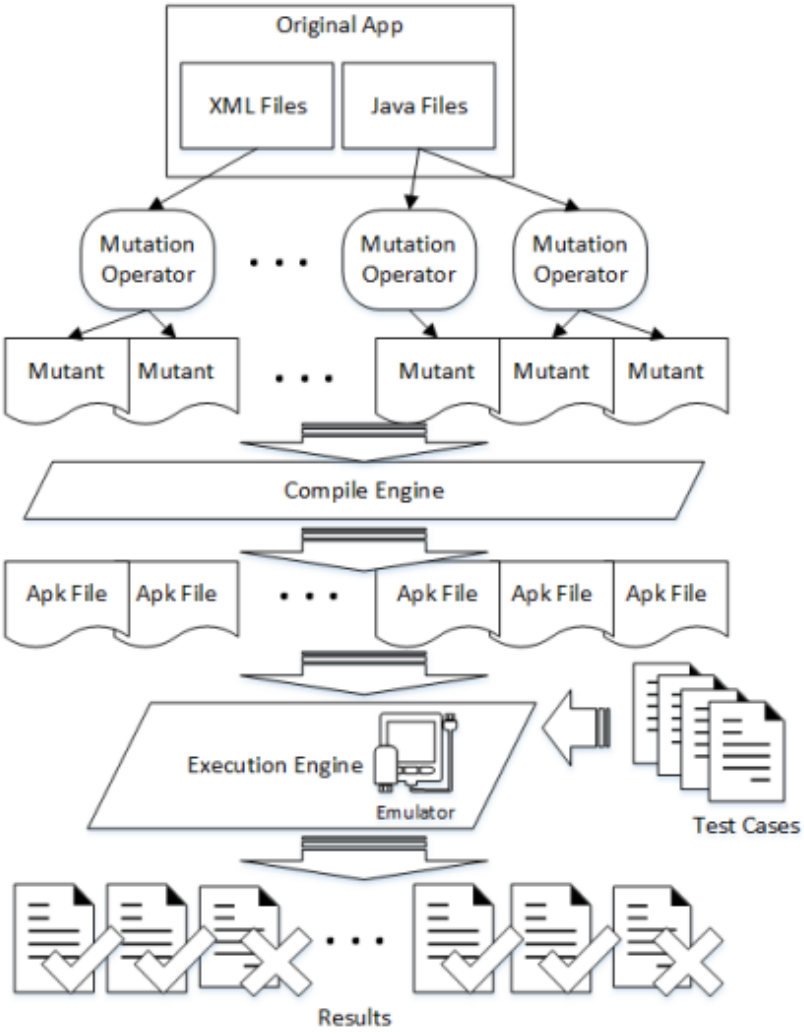
Trong một lần kiểm thử L. Deng và cộng sự[1] đã đề xuất các toán tử đột biến cụ thể cho các đối tượng để kiểm thử tích hợp đã được phát triển. Sau đó, L. Deng và cộng sự đã xác định 58 toán tử đột biến để kiểm thử các hệ thống đa lớp ở các cấp độ tích hợp khác nhau, và đã đề xuất các toán tử cụ thể cho một số ngôn ngữ tạo chương trình như C, C#, C++, Python hoặc PHP. Các ngôn ngữ này cũng có các toán tử đột biến cho các cơ sở dữ liệu quan hệ.

L. Deng và cộng sự [1]chỉ ra các phương pháp và các kỹ thuật đột biến có thể được sử dụng để tạo các kịch bản kiểm thử cho các mô hình theo hướng khía cạnh. Kiểm thử đột biến không thể được thực hiện theo cách tương tự cho các ứng dụng Android như đối với các chương trình Java truyền thống.

Đầu tiên, trong khi các công cụ phân tích đột biến Java chỉ đột biến các tệp Java, thì các nhà phát triển Android cũng thay đổi các bố cục của các tệpXML. Thứ

hai, ứng dụng Android yêu cầu xử lý bổ sung trước khi được triển khai. Do vậy các công cụ đột biến Java thường biến đổi mã nguồn, sau đó biên dịch thành các tệp lớp bytecode hoặc biên dịch thành mã byte, sau đó biến đổi mã byte. Các tệp bytecode Java sau đó được liên kết động bởi hệ thống ngôn ngữ trong khi thực thi. Các ứng dụng Android có yêu cầu bổ sung rằng mỗi đột biến Android phải được biên dịch dưới dạng gói ứng dụng Android (APK) để có thể cài đặt và thực thi trên thiết bị di động và trình giả lập (Emulator). Điều này có ý nghĩa tác động đến việc thiết kế các công cụ phân tích đột biến cho ứng dụng Android. Minh họa cách công cụ phân tích đột biến như Hình 2.1 [1]là các bước để tiến hành phân tích đột biến trên các ứng dụng Android.

1. Bước 1: Kiểm thử viên chọn toán tử đột biến nào sẽ được sử dụng trong quá trình kiểm thử. Đồng thời công cụ phân tích đột biến Android và sử dụng một phần của công cụ tạo ra đột biến muJava để thực hiện các toán tử đột biến này.
2. Bước 2: Đối với các toán tử làm thay đổi mã Java (cả Android mới và các toán tử muJava truyền thống), hệ thống sẽ sửa đổi mã nguồn Java ban đầu và biên dịch chúng thành các tệp lớp bytecode.
3. Bước 3: Các toán tử đột biến XML được áp dụng trực tiếp vào các tệp XML, tạo ra một bản sao mới của tệp cho mỗi đột biến. Chúng được hoán đổi vị trí để liên kết động khi các tệp APK được tạo ra.
4. Bước 4: Đối với mỗi tệp lớp Java và tệp XML bị đột biến, tệp đột biến hệ thống sẽ tạo ra tệp APK bị đột biến bằng cách tạo ra mã nguồn bị đột biến và tệp dự án khác. Một số đột biến có thể gây ra lỗi biên dịch (stillborn) sẽ được loại bỏ ngay lập tức và không được sử dụng trong kết quả cuối cùng.



Hình 2.1 Thực hiện phân tích đột biến trên ứng dụng Android [1]

5. Bước 5: Sử dụng khung kiểm thử Android mở rộng JUnit để hỗ trợ kiểm thử các loại thành phần Android khác nhau. Ngoài ra, kiểm thử viên có thể viết các trường hợp kiểm thử với sự hỗ trợ của các khung tự động kiểm thử Android bên ngoài, chẳng hạn như Robotium. Đây là công cụ phân tích đột biến Android đểtriển khai để chạy cả hai loại trường hợp kiểm thử ở trên. Các trường hợp kiểm thử được thiết kế bởi người thử nghiệm để nhắm mục tiêu các đột biến hoặc có thể sử dụng một bộ thử nghiệm được tạo bên ngoài.
6. Bước 6: Sau khi tạo đột biến và biên dịch chúng thành tệp APK, hệ thống sẽ tải phiên bản gốc (không bị đột biến) của ứng dụng đang được kiểm thử vào trình giả lập hoặc trên thiết bị di động. Sau đó, hệ thống sẽ thực thi tất cả các trường hợp kiểm thử trên ứng dụng gốc và ghi lại kết quả đầu ra bị vô hiệu hóa. Kết quả đột biến được so sánh với kết quả của ứng dụng gốc để xác định đột biến nào bị loại bỏ.

7. Bước 7: Mỗi đột biến được tải vào trình giả lập hoặc lên thiết bị di động để thực thi kiểm thử. Hệ thống đột biến thực hiện tất cả các trường hợp kiểm thử đối với các đột biến và lưu trữ kết quả đầu ra như kết quả thực tế. Với công cụ hiện tại kiểm thử viên chạy các trường hợp kiểm thử bằng Robotium rất tốn thời gian. Theo đó khi dùng Robotium, tốc độ thực hiện kiểm thử cao hơn có thể làm cho việc thực thi không ổn định trên trình giả lập. Khi thực thi các trường hợp kiểm thử với trình giả lập, mỗi lần kiểm thử cần hàng giờ để chạy với tất cả các đột biến.
8. Bước 8: Sau khi thu thập tất cả các kết quả hệ thống đột biến so sánh kết quả mong đợi với kết quả thực tế. Nếu kết quả thực tế trong lần kiểm thử khác với kết quả dự kiến trong cùng một lần kiểm thử, thì đột biến đó được đánh dấu là đã bị hủy bỏ bởi kiểm thử đó.
9. Bước 9: Cuối cùng điểm đột biến được tính bằng tỷ lệ phần trăm của các đột biến bị hủy bỏ bởi các quá trình kiểm thử. Hiện tại, công cụ này không triển khai bất kỳ phương pháp phỏng đoán nào để giúp xác định các đột biến tương đương, vì vậy tất cả chúng phải được đánh giá bằng phương pháp thủ công.

2. 1. 1. Hướng dẫn đột biến và thế hệ đột biến

M. P. Usaola và cộng sự [6] nêu rõ hành vi của một đột biến có thể bao gồm trong việc thông qua mọi toán tử đột biến và yêu cầu nó lấy đột biến của lớp để đột biến. Giả sử chỉ có thể thay đổi các cấu trúc và phương thức xây dựng, toán tử sẽ thực hiện mọi thao tác trong lớp. Với mỗi thao tác nó lần lượt đi qua tất cả các hướng dẫn của nó để xác định xem nó có thể hoặc không thể thay đổi phương pháp.

Let be c the class to mutate
Let be mutant = Ø
Let be mutableMethods = Ø
For each method m in c
If m is mutable then
mutableMethods = mutableMethods U { m}
End
End
For each method m in muatbleMethods
mutants = mutansmutate (c, m)

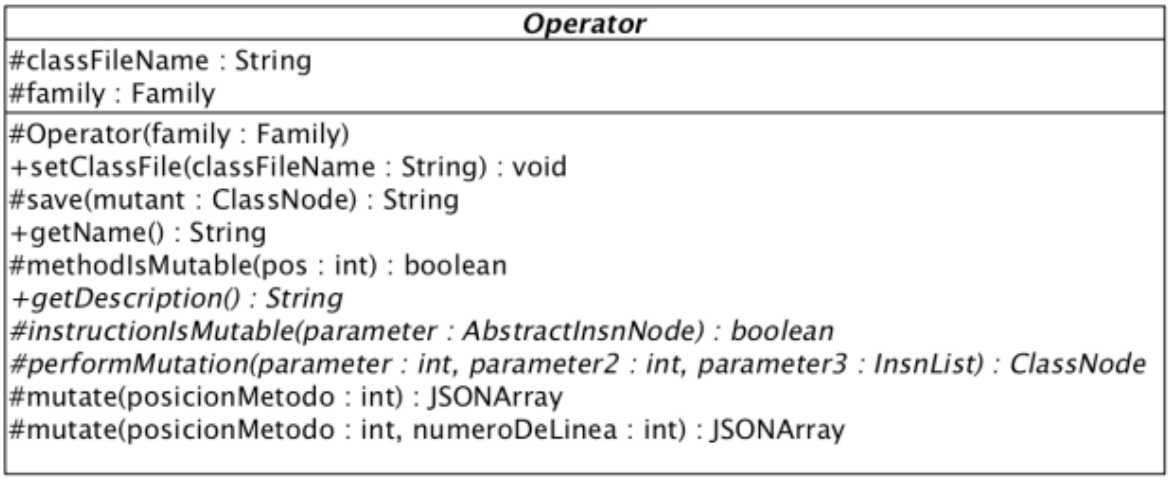
End

Ví dụ trên cho thấy mã giả của một phương thức có thể thực hiện của phương thức GenerMutants (c: Class) thuộc về lớp Toán tử: như đã quan sát, nó bổ sung vào một phương thức có thể thay đổi tập hợp tất cả các phương thức trong c mà nó có thể biến đổi. Đối với mọi phương thức có thể thay đổi, nó gọi một hàm đột biến bổ sung (c: Class, m: Phương thức), áp dụng toán tử đột biến cho phương thức được truyền dưới dạng tham số.

2. 1. 2. Xác định cấu trúc toán tử

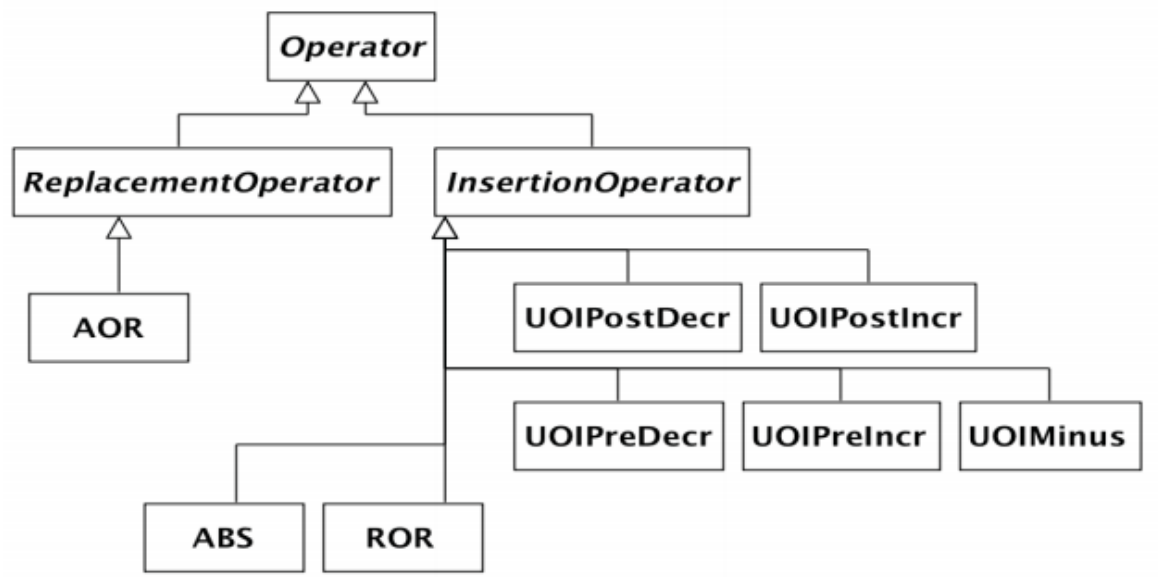
Mục tiêu của việc xác định một cấu trúctoán tử là có thể tái sử dụng và dễ dàng thực hiện các toán tử đột biến. Vì vậy để xác định một lớp toán tử trừu tượng như Hình 2.2 [6]sẽ chứa nhiều phương thức cụ thể:

- Mỗi toán tử có hai trường: tên tệp lớp (được sử dụng để xử lý mã byte của nó) và cụm, được sử dụng để nhóm các nhà khai thác theo danh mục trong giao diện người dùng web.
- Muốn cung cấp cho công cụ một kiến trúc plugin (tức là các toán tử mới có thể được thêm, tải khi áp dụngtại thời gian chạy), hàm tạo của lớp được bảo vệ vàkhông thể nhìn thấy từ bên ngoài. Để khởi tạo, nó sẽ gọi hàm tạo của nó bằng một phản xạ tới phương thức newInstance (bên trong java. lang. Class).
- Hàm getName trả về tên toán tử lớp và nó là từ viết tắt được hiển thị trong giao diện người dùng. Ví dụ: nếu toán tử AOR được thực hiện trong tệp AOR. Class, nó phản hồi trả về "AOR"string.
- getDescriptionphương thức là trừu tượng và nó trả về một mô tả văn bản của toán tử. Ví dụ, đối với AOR, nó trả về "Thay thế toán tử số học".
- Phương thức đột biến biểu thị việc thực hiện các nhiệm vụ được mô tả, có hai phương thức sau đây: (i)instructionIsMutable là trừu tượng, vì vậy việc triển khai của nó phụ thuộc vào toán tử cụ thể, (ii) performMutationđược sử dụng để sửa đổi phương thức và hướng dẫn có chỉ mục được truyền dưới dạng tham số. Đây là phương thức thực sự xây dựng từng đột biến, biến nó thành đối tượng ClassNode với bytecode của nó và sử dụng một danh sách (như minh họa tại Hình 2.2).



Hình 2.2 Cấu trúc của toán tử trừu tượng [6]

Như M. P. Usaola và cộng sự [6] chỉ ra, một số toán tử như AOR, bao gồm việc thay thế đơn giản một lệnh bytecode bằng một số lệnh khác, Trong khi những yêu cầu khác như chèn các dòng bytecode tại một vị trí cụ thể. Thì đối với điều này toán tử có hai phân vùng trực tiếp, trừu tượng được hiển thị trong hình 2.3 [6] (InsertsOperator và Replace-mentOperator).



Hình 2.3 Thiết kế của các toán tử truyền thống [6]

Hình 2.3 cũng cho thấy hệ thống phân cấp các toán tử trong số các toán tử đột biến "truyền thống" được tạo ra trong Bacterio Web.

AOR, ABS: Là những toán tử số học được thay thế.

ROR : Toán tử thay thế quan hệ.

UOI: Toán tử chèn được thực hiện trong một số lớp.

2.2. Toán tử đột biến trong kiểm thử phần mềm

Phần này trình bày chi tiết về những kỹ thuật kiểm thử đột biến từ [1] dành cho những ứng dụng được chạy trên nền tảng Android. Để vận dụng những kỹ thuật này trong thực tế tôi sẽ chạy lại thực nghiệm kỹ thuật này trong chương 4. Ứng dụng được kiểm thử là những ứng dụng chạy trên điện thoại di động đang được sử dụng hệ điều hành Android.

2.2.1. Toán tử đột biến ý định

Toán tử đột biến ý định (Intent Mutation Operators) là biểu thị một hoạt động được thực hiện giữa các thành phần của Android. Chúng thường được sử dụng để khởi chạy một hoạt động truyền dữ liệu hoặc truyền thông điệp giữa các hoạt động của ứng dụng với nhau.

2.2.2. Thay thế trọng tải ý định

Thay thế trọng tải ý định (Intent Payload Replacement : IPR)là một ý định có thể mang các loại dữ liệu khác nhau được gọi là tải trọng dưới dạng các cặp khóa và giá trị. Ví dụ phương thức TheputExtra () lấy tên khóa làm tham số đầu tiên và giá trị làm tham số thứ hai. Toán tử IPR biến đổi tham số thứ hai thành giá trị mặc định phụ thuộc vào kiểu dữ liệu cơ bản như int, short, long, string…. Các giá trị mặc định này được liệt kê như trong bảng 2.1 [1]:

Original Type	Default Value
Int, short, long, float, double, char	0
Boolean	True/false
String	“”(String) null
Array	(Array) null
Others	(Others) null

Bảng 2.1 Giá trị mặc định của IPR [1]

Đối với các đối tượng có kiểu số nguyên thủy, chẳng hạn như int, short, long, v.v. sẽ được thay thế bằng giá trị 0. Các biến boolean được thay thế bằng cả true và false. Các đối tượng chuỗi được thay thế bằng các chuỗi rỗng và giá trị null. Mảng và

các loại đối tượng khác được thay thế bằng các giá trị null thành các loại thích hợp. Đối tượng String được thay thế bằng một String rỗng (các câu lệnh gốc và biến đổi được in đậm). Đột biến IPR thách thức kiểm thử viên thiết kế các trường hợp kiểm thử để đảm bảo giá trị được truyền bởi một đối tượng ý định là chính xác.

Chương trình gốc:

```
public void test (View view)
{
Intent intent = new Intent (this, DisplayMessageActivity.class);
EditTex teditText=(EditText) findViewById (R.id.editmessage);
String message = editText.getText().toString();
intent.putExtra (EXTRA MESSAGE, message);
startActivity (intent);
}
```

Chương trình sau khi biến đổi:

```
public void test (View view)
{
Intent intent = new Intent (this, DisplayMessageActivity.class);
EditText editText = (EditText) findViewById (R.id.editmessage);
String message = editText.getText().toString();
intent.putExtra (EXTRA MESSAGE, \");
startActivity (intent);
}
```

2.2.3. Mục tiêu thay thế

Mục tiêu thay thế (Intent Target Replacement: ITR) là việc sử dụng ý định ẩn ý để chỉ định thành phần nào sẽ được bắt đầu bằng cách khai báo ý định với tên của thành phần đích trong ứng dụng. Toán tử ITR đầu tiên tìm kiếm tất cả các lớp trong cùng một gói của lớp hiện tại, và sau đó thay thế mục tiêu của mỗi ý định bằng tất cả các lớp có thể. Điều này thách thức kiểm thử viên thiết kế các trường hợp kiểm thử để kiểm tra xem hoạt động được khởi chạy thành công sau khi ý định được thực thi.

Chương trình gốc:

```
public void startActivityB (View v)
{
Intent intent = new Intent (ActivityA.this, ActivityB.class);
startActivity (intent);
}
```

Chương trình sau khi biến đổi:

```
public void startActivityB (View v)
{
Intent intent = new Intent (ActivityA.this, ActivityC.class);
startActivity (intent);
}
```

2.3. Vòng đời hoạt động của toán tử đột biến

Vòng đời hoạt đôngcủa các toán tử đôt biến được thể hiện cụ thể trước đến sau bởi các thành phần chính. Các thành phần sử dụng các phương pháp để chuyển tiếp giữa các trạng thái khác nhau trong vòng đời. Toán tử này sẽ sửa đổi các phương pháp đó.

2.3.1. Phương pháp xóa vòng đời

Phương pháp xóa vòng đời (Lifecycle Method Deletion: MDL)là việc ghi đè các phương thức chuyển đổi để chuyển đổi giữa các trạng thái và công việc của kỹ thuật này. MDL xóa từng phương thức ghi đè để buộc Android gọi phiên bảntrong supper class. Điều này đòi hỏi kiểm thử viên phải thiết kế các trường hợp kiểm thử để đảm bảo ứng dụng ở trạng thái mong đợi chính xác. Toán tử MDL tương tự như toán tử đột biến Overriding Method Deletion (IOD) trong muJava, nhưng chỉ xem xét các phương thức liên quan đến vòng đời hoạt đông.

2.4. Xử lý toán tử đột biến

Khi xử lý toán tử đột biến các ứng dụng Android hoạt động dựa trên trình xử lý sự kiện, vì vậy trình xử lý sự kiện thường được sử dụng để nhận biết và phản hồi các sự kiện. Các hành động phổ biến của người dùng là nhấp và chạm, mỗi hành động đó sẽ tạo ra một sự kiện. Do đó, hai toán tử đột biến cho các trình xử lý sự kiện là toán tử thay thế sự kiện OnClick (ECR) và toán tử thay thế sự kiện OnTouch (ETR).

2.4.1. Thay thế sự kiện bằng onClick

Thay thế sự kiện bằng onClick (OnClick Event Replacement :ECR)là khi ECR đầu tiên được tìm kiếm và lưu trữ tất cả các trình xử lý sự kiện phản hồi các OnClick events trong lớp hiện tại. Sau đó, nó thay thế mỗi trình xử lý bằng một trình xử lý tương thích khác. Trong đó trình xử lý sự kiện cho nút mPrepUp đã được thay thế bằng trình xử lý sự kiện cho nút mPrepDown. Để tiêu diệt các đột biến ECR, mỗi sự kiện OnClick của ứng dụng phải được thực hiện bằng ít nhất một lần.

Chương trình gốc:

```
mPrepUp.setOnClickListener (new OnClickListener()
{
public void onClick (View v){
incrementPrepTime();
}
});
mPrepDown.setOnClickListener (new OnClickListener()
{
public void onClick (View v){
decrementPrepTime();
}
});
```

Chương trình sau khi biến đổi:

```
mPrepUp.setOnClickListener (new OnClickListener()
{
public void onClick (View v){
decrementPrepTime();
}
});
mPrepDown.setOnClickListener (new OnClickListener()
{
public void onClick (View v){
decrementPrepTime();
}
```

});

2.4.2. Thay thế sự kiện bằng onTouch

Toán tử này thay thế các trình xử lý (OnTouch Event Replacement: ETR) sự kiện cho mỗi OnTouchevent. Nó hoạt động giống như toán tử đột biến ECR (Thay thế sự kiện bằng onClick).

2.5. Toán tử đột biến XML

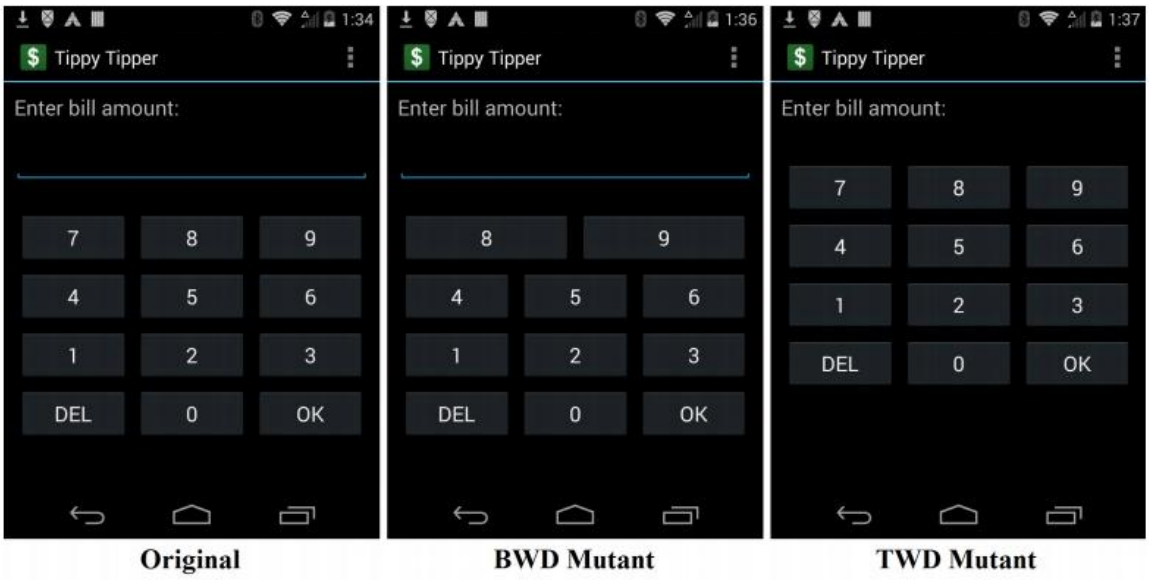
Đối với kiểm thử viên, khi sử dụng toán tử đột biến XML thì cần biết trong Android sử dụng nhiều tệp XML không chỉ là tệp kê khai. Tệp XML được sử dụng trong giao diện người dung để lưu trữ dữ liệu độ tin cậy như quyền, hoạt động khởi chạy mặc định và nhiều chức năng hơn thế nữa. Toán tử này khác nhau ở chỗ chúng không sửa đổi mã thực thi nhưng XML là tĩnh.

2.5.1. Xóa nút trên giao diện

Xóa một nút (Button Widget Deletion : BWD) tại một thời điểm khỏi bố cục XML trên giao diện. Việc hủy bỏ các đột biến BWD yêu cầu đảm bảo rằng mọi nút phải được hiển thị thành công. Hình 2.1 [1] cho thấy một màn hình gốc ở bên trái và hai đột biến ở bên phải. Màn hình ở giữa là đột biến BWD trong đó nút "7" bị xóa khỏi giao diện. Toán tử đột biến này buộc kiểm thử viên phải thiết kế các trường hợp kiểm thử từng nút theo đúng hoạt động của nó.

2.5.2. Sửa đổi thành phần

Sửa đổi thành phần (EditText Widget Deletion: TWD) được sử dụng để hiển thị văn bản cho người sử dụng. Toán tử đột biến TWD loại bỏ từng widget EditText. Màn hình ngoài cùng bên phải trong Hình 2.1 [1] cho thấy một ví dụ đột biến TWD trong đó số tiền hóa đơn không được hiển thị.

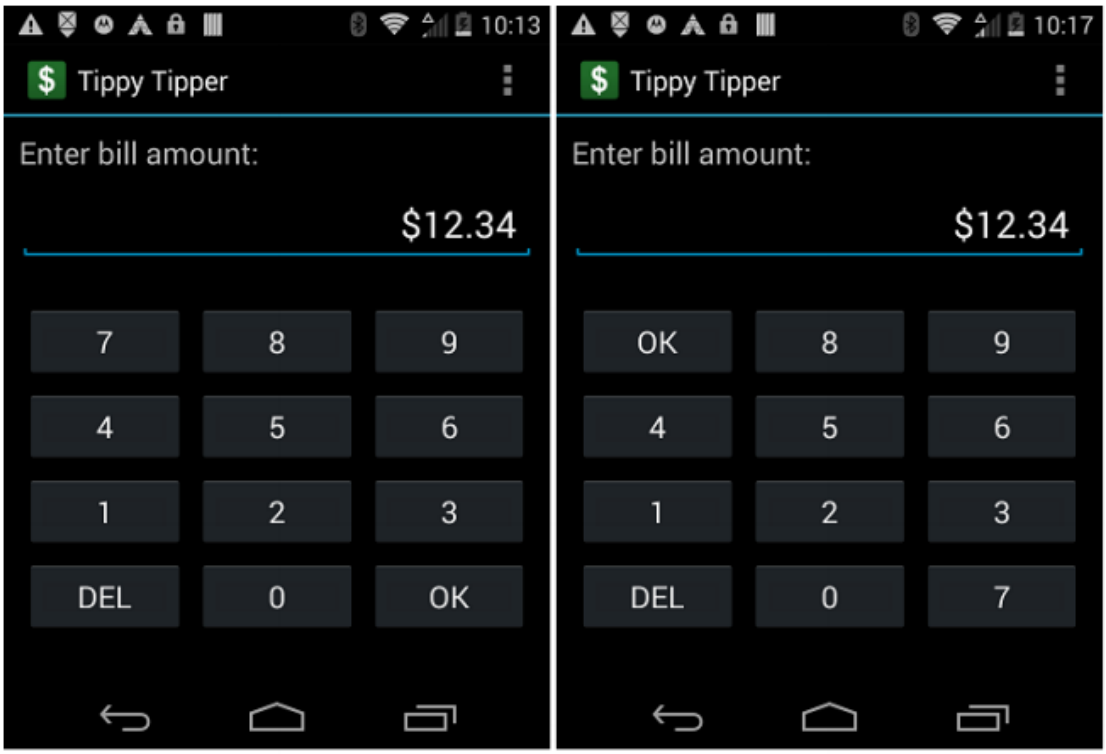


Hình 2.4 Ví dụ về BWD và TWD [1]

2.5.3. Chuyển đổi nút trên giao diện

Chuyển đổi nút trên giao diện (Button Widget Switch: BWS) là chuyển đổi các nút xuất hiện trên màn hình. Kiểm thử viên phải thiết kế các trường hợp kiểm thử để đảm bảo ứng dụng hoạt động như mong đợi. Mặc dù các nút đã được thay đổi nhưng hoạt động đối với các yêu cầu chức năng của nó vẫn được hoạt động đúng yêu cầu. Tuy nhiên, các ứng dụng Android hoạt động dựa trên sự kiện, điều đó có nghĩa là cần thiết để hiển thị cấu trúc giao diện một cách thích hợp, cũng như xử lý các sự kiện của người dùng.

Không giống như BWD, BWS không xóa nút mà chuyển đổi vị trí của hai nút trên cùng một màn hình. Theo cách này, chức năng của một nút sẽ không được chú ý, nhưng bố cục màn hình lại trông khác với bản gốc. BWS yêu cầu kiểm thử viên thiết kế các trường hợp kiểm thử có chủ ý và kiểm thử vị trí (tương đối hoặc tuyệt đối) của một nút. Hình 2.2 [1] minh họa một ví dụ về đột biến BWS. Đột biến ở phía bên phải thể hiện sự chuyển đổi vị trí của các đột biến BWS ở nút " 7 "và " OK".



Hình 2.5 Ví dụ về Button Widget Switch [1]

Kiểm thử viên cần thiết kế các trường hợp vị trí của một widget và so sánh nó với một giá trị dự kiến hoặc vị trí của các widget khác, để đảm bảo vị trí chính xác của nó. Ví dụ, đoạn mã nguồn này so sánh vị trí của hai nút để đảm bảo nút OK được hiển thị ở bên trái nút Hủy.

Button okButton = (Button) solo. getView (R. id. ok);

Button cancelButton = (Button) solo. getView (R. id. cancel);

int [ ] locationOfOK = new int [2];

int [ ] locationOfCancel = new int [2];

okButton. getLocationInWindow (locationOfOK);

cancelButton. getLocationInWindow (locationOfCancel);

assertTrue (\OK button is on the left of Cancel", locationOfOK [0]<locationOfCancel [0]);

2.6. Ý tưởng áp dụng trong luận văn

Từ những kỹ thuật trên, chúng tôi đưa ra ý tưởng để áp dụng vào trong luận văn. Ở mỗi phần 2.3;2.4; 2.5, chúng tôi sẽ đưa ra một thí nghiệm để minh họa một kỹ thuật

trong mỗi phần. Và ứng dụng được lựa chọn để kiểm thử là ứng dụng Flashair, ứng dụng này kết nối với thẻ nhớ Flashair và được chạy trên nền tảng Android. Mô tả chi tiết của ứng dụng này được đề cập đến chương 4.

Trong ứng dụng Flashair, chúng tôi sẽ lựa chọn một số chức năng chính để sử dụng vào việc kiểm thử đột biến như: chức năng chỉnh sửa ảnh, chức năng xem ảnh trong màn hình chính và chức năng cài đặt.

Về công cụ để thực hiện kiểm thử, chúng tôi sử dụng công cụ kiểm thử Robotium kết hợp vào trong Android studio để kiểm thử. Chi tiết và tính năng của hai công cụ này sẽ được đề cập ở chương 3 và chương 4 của luận văn này.

2.7. Tóm tắt chương 2

Trong chương 2, luận văn đã cụ thể hóa phương pháp kiểm thử đột biến trên ứng dụng Android. Cụ thể ở đây là đột biến với những toán tử và các công việc cần làm trong mỗi trường hợp kiểm thử đột biến. Ngoài ra, chương này cũng trình bày cách kiểm thử cấu trúc chương trình bên trong của ứng dụng cũng như kiểm thử về giao diện của ứng dụng. Trong phần tiếp theo của luận văn, chúng tôi sẽ đưa ra mô hình đề nghị để thực hiện việc kiểm thử đột biến.

Chương 3. MÔ HÌNH ĐỀ NGHỊ, THỰC NGHIỆM VÀ ĐÁNH GIÁ

3.1. Phát biểu bài toán

Trong phần này, chúng tôi sẽ nói rõ hơn về mô hình được áp dụng để kiểm thử trong luận văn. Môi trường phát triển của Android bao gồm phần kiểm thử riêng mở rộng JUnit. JUnit là một framework đơn giản dùng cho việc tạo các unit testing tự động, và chạy các trường hợp kiểm thử lặp đi lặp lại. Do đó JUnit là một framework đơn giản thường được dùng để viết Unit Test trên môi trường Java. JUnit giúp người lập trình viên phải lặp lại nhiều lần việc kiểm thử bằng cách tách biệt mã kiểm thử ra khỏi mã chương trình, đồng thời tự động hóa việc tổ chức và thi hành các bộ số liệu kiểm thử. Hiện nay trong Android studio đã tích hợp sẵn công cụ JUnit và sau khi tích hợp Robotium vào Android studio, kiểm thử viên có thể sử dụng Robotium để kiểm thử đơn vị, kiểm thử hệ thống và kiểm thử chấp nhận của người dùng.

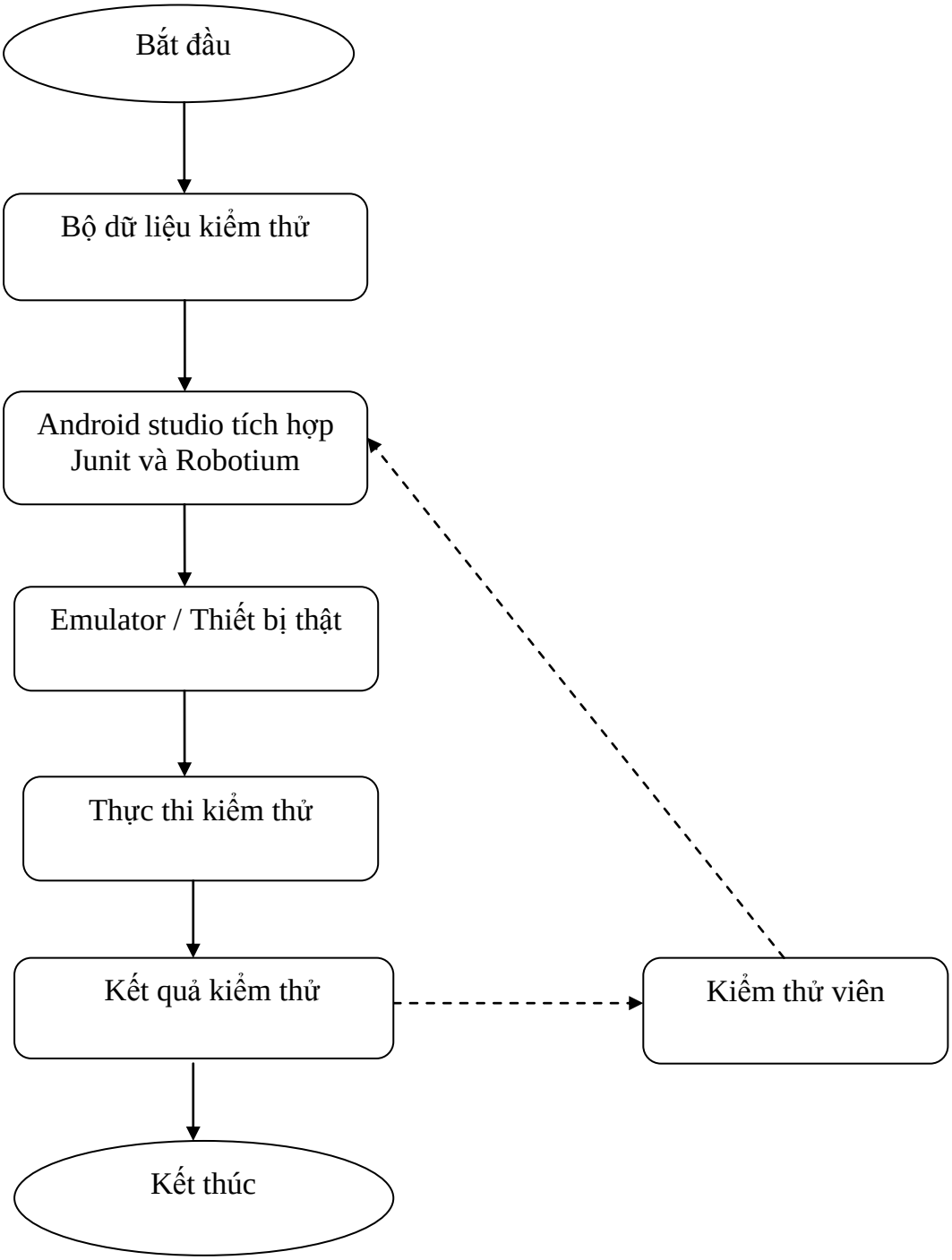
Robotium là lựa chọn để tiến hành kiểm thử ứng dụng Android. Robotium là khung nền nguồn mở và được xem như là một khung nền mở rộng Android. Sử dụng Robotium, lập trình viên có thể viết các trường hợp kiểm thử tự động cho ứng dụng Android. Tuy nhiên, lập trình viên có thể viết các chức năng, hệ thống và các kịch bản kiểm thử chấp nhận, là cầu nối giữa các hoạt động Android. Nhờ các API của nó cho các thành phần giao diện của Android bằng cách liên kết thời gian chạy tất cả các dữ liệu kiểm được sử dụng trong luận văn này đều được thiết kế bằng Robotium. Có thể sử dụng chương trình giả lập như emulator để chạy chương trình kiểm thử. Tuy nhiên trong luậnvăn này chúng tôi sẽ kết nối với thiết bị thật.

3.2. Mô hình đề nghị và các bước thực hiện

3.2.1. Mô hình thực hiện kiểm thử

Đối với mô hình này kiêm thử viên sẽ thiết kế sẵn bộ dữ liệu kiểm thử bao gồm các trường hợp kiểm thử và đưa vào Android Studio để thực hiện kiểm thử. Trong Android studio sẽ được tích hợp sẵn Junit và Robotium, chi tiết công cụ này sẽ được nói rõ hơn trong phần sau. Android Studio sẽ thực thi kiểm thử trên thiết bị thật hoặc một trình giả lập Emulator. Emulator(trình giả lập) là một chương trình phần mềm cho phép thiết bị điện thoại di động của bạn bắt chước các tính năng của một máy tính hoặc phần mềm di động khác mà bạn muốn chúng bắt chước bằng cách cài đặt vào máy tính

hoặc thiết bị di động của mình. Sau khi việc thực thi kết thúc, chương trình sẽ thông báo cho kiểm thử viên về kết quả kiểm thử. Kết quả trường hợp kiểm thử đó có thể là thành công hay thất bại, nếu kết quả kiểm thử thất bại thì kiểm thử viên sẽ báo cáo với lập trình viên để sửa những lỗi đó. Mô hình sẽ được cụ thể như sau:



Hình 3.1 Mô hình thực hiện kiểm thử

3.2.2. Các bước thực hiện kiểm thử

Bước 1:Cài đặt thành công Android Studio trên máy tính.

Android Studio là môi trường phát triển tích hợp (IDE) chính thức dành cho phát triển các ứng dụng chạy trên nền tảng Android.Junitđã được tích hợp sẵn trong bộ cài Android Studio và để thực hiện được kiêm thử thì cần tích hợp Robotium vào Android Studio. chi tiết về tích hợp sẽ được nói rõ hơn trong phần thực nghiệm.

Bước 2: Tạo bộ dữ liệu kiểm thử

Các kiểm thử viên tạo bộ dữ liệu kiểm thử để kiểm thử bằng các kỹ thuật như black box testing, white box testing và một số kỹ thuật khác. Việc tạo bộ dữ liệu kiểm thử theo các kỹ thuật đã nói như trên sẽ bao phủ được các trường hợp kiểm thử để tránh tình trạng thừa hoặc thiếu trường hợp kiểm thử, hoặc tạo ra những trường hợp kiểm thử không cần thiết khi kiểm thử.

Bước 3: Tạo Test Project trong Android studio

Trong bước này, kiểm thử viên sẽ tạo một project test và các trường hợp kiểm thử sau khi đã thiết kế các trường hợp kiểm thử. Các trường hợp kiểm thử trong mô hình này sẽ được viết bằng ngôn ngữ Java. Cấu trúc project gồm những thành phần như sau:

- manifests: Chứa tệp AndroidManifest. xml.
- java: Chứa các tệp mã nguồn Java, bao gồm mã kiểm tra JUnit.
- res: Chứa tất cả các tài nguyên không phải là mã nguồn, chẳng hạn như XML layouts, UI stringsvà hình ảnh bitmap.

Bước 4: Cài đặt trên thiết bị thật hoặc ảotrên máy tính

Để chạy các trường hợp kiểm thửtrong Android Studio, kiểm thử viên có thể tự tạo những thiết bị ảo bằng những công cụ hiện nay mà các nhà phát triển cung cấp hoặc có thể tạo thiết bị ảo ngay trên Android Studio. Đối với thiết bị ảo, kiểm thử viên có thể tạo ra bất cứ thiết bị ảo nào với hệ điều hành tương ứng (Ví dụ: Kiểm thử viên tạo ra một thiết bị ảo là điện thoại Nexus sử dụng hệ điều hành Android 6.0). Kiểm thử viên có thể kiểm thử phần mềm trên nhiều thiết bị

khác nhau. Trong trường hợp kiểm thử viên muốn kiểm thử trên thiết bị thật thì có thể kết nối thiết bị thật với máy tính để kiểm thử.

Bước 5: Thực thi Test Project trong Android Studio

Sau khi thiết kế và viết các trường hợp kiểm thử trên Android Studio và kết nối thiệt bị ảo hoặc thật. Kiểm thử viên sẽ chạy các trường hợp kiểm thử của mình một các dễ dàng bằng cách chọn lệnh Run trên hệ thống thanh công cụ. Khi đó chương trình sẽ hiển thị hộp thoại cho phép bạn tuỳ chỉnh.Khi chạy, ứng dụng sẽ hiển thị thông báo kết nối với thiết bị nào để chạy.

Bước 6: Đọc kết quả kiểm thử trên màn hình

Sau khi chạy chương trình kiểm thử hoàn tất. Màn hình sẽ thông báo kết quả kiểm thử. Kết quả thông báo sẽ có hai giá trị là fail và passed. Fail tương ứng với trường hợp kiểm thử thất bại. Passed tương ứng với trường hợp kiểm thử thành công. Đối với những trường hợp kiểm thử thất bại, kiểm thử viên sẽ báo cáo kết quả cho các lập trình viên, phân tích nguyên nhân gây ra thất bại của trường hợp kiểm thử đó. Từ đó các lập trình viên sẽ khắc phục những trường hợp thất bại, chất lượng phần mềm được nâng cao.

3.2.3. Đánh giá

Khi tích hợp và sử dụng Robotium vào Android studio để kiểm thứ ứng dụng thì có nhiều ưu điểm như sau:

- Android Studio làmột IDE được thiết kế và phát triển đặc biệt để dành cho việc xây dựng các ứng dụng Android. Nó có tốc độ rất nhanh và hiệu quả, có thể thiết lập một dự án Android mới cho nhiều loại ứng dụng Android khác nhau chỉ trong vòng vài giây hoặc có thể tạo ra nhiều các thiết bị ảo phiên bản khác nhau (emulator) khác nhau. Ví dụ như có thể tạo các phiên bản Android 5.0, Androi 6.0, Andoid 8.0….. Trước đây, khi Android mới được khởi chạy, việc phát triển ứng dụng Android được thực hiện bằng Eclipse và Android Developer Tools plugin. Tuy nhiên, hiện nay đã được thay thế bởi Android Studio.

- Khi dựa trên nền tảng của JUnit test sẽ quen thuộc và dễ sử dụng. Kèm theo đó JUnit được phát triển cho mục đích kiểm thửnên các thành phần trong ứng dụng. Vì vậy nên dễ cài đặt và tốc độ chạy kiểm thử ứng dụng sẽ nhanh. Bên cạnh đó hệ thống thư viện phong phú, Framework và tool được hỗ trợ bởi Google, tương thích với tất cả các phiên bản Android và được cập nhật thường xuyên.
- Robotium xử lý từng hoạt động một vì vậy có thể sử dụng những trường hợp khó và phức tạp. Bên cạnh đó khi sử dụng Robotium thì code ngắn hơn và dễ viết hơn, nên thời gian tối thiểu để viết các trường hợp kiểm thử được rút ngắn và không ảnh hưởng đến nền tảng Android.

Tuy nhiên, bên cạnh những ưu điểm này thì các công cụ được sử dụng trong mô hình cũng có những nhược điểm sau đây:

- Các nhà phát triển Android Studio luôn phát triển và cho ra những phiên bản mới cải tiến hơn so với phiên bản cũ. Vì vậy những phiên bản mới có tốc độ khởi động và chạy tương đối lâu. Khi sử dụng Android Studio kết hợp với trình giả lập emulator cũng làm mất nhiều bộ nhớ của máy tính.
- Đối với Robotiumviệc thực thi các trường hợp kiểm thử gồm nhiều bước phức tạp là tương đối khó. Vì vậy Robotium được áp dụng để chạy những trường hợp kiểm thử không quá phức tạp.

Trong luận văn này chúng tôi thực hiện ba thí nghiệm: thí nghiệm 1 là sử kỹ thuật Intent Payload Replacement (IPR). Thí nghiệm 2 là sử dụng kỹ thuật OnClick Event Replacement (ECR). Và thí nghiệm 3 là sử dụng kỹ thuật thuật Button Widget Switch (BWS). Dựa vào kết quả thực nghiệm, chúng tôi tiến hành đánh giá nhận xét rút ra kết luận và đưa ra định hướng tiếp theo.

3.3. Môi trường thực nghiệm

3.3.1. Cấu hình phần cứng

Bảng 3.1. cung cấp thông tin cấu hình phần cứng tiến hành thực nghiệm trong luận văn.

Bảng 3.1 Cấu hình máy tính thực nghiệm

Thành phần	Chỉ số
CPU	Intel® Core™ i5-8265U
RAM	4GB
HDD	500GB
OS	Window 7 (32bit)

3.3.2. Công cụ phần mềm

Trong quá trình thực nghiệm, chúng tôi sử dụng một số công cụ và phần mềm mã nguồn mở được liệt kê trong Bảng 3.2

Bảng 3.2 Danh sách phần mềm sử dụng trong thực nghiệm

STT	Tên phầm mềm	Tác giả	Nguồn
1	Android-studio-ide-182. 5314842		https://developer. android. com/studio#downloads
2	Robotium5. 6. 3		http://robotiumsolo. blogspot. com/p/downloads. html
3	MS-Excel trong bộ MS-Office 2013	Microsoft	http://www. microsoft. com
4	Flashair		https://github. com/FlashAirDevelopers/FlashAirFileManager

Dưới đây là mô tả về các phần mềm chính trong thực nghiệm:

- **Android-studio-ide-182. 5314842**

Android Studio là Môi trường phát triển tích hợp (IDE) chính thức để phát triển ứng dụng Android, dựa trên IntelliJ IDEA. Ngoài các công cụ phát triển và chỉnh sửa

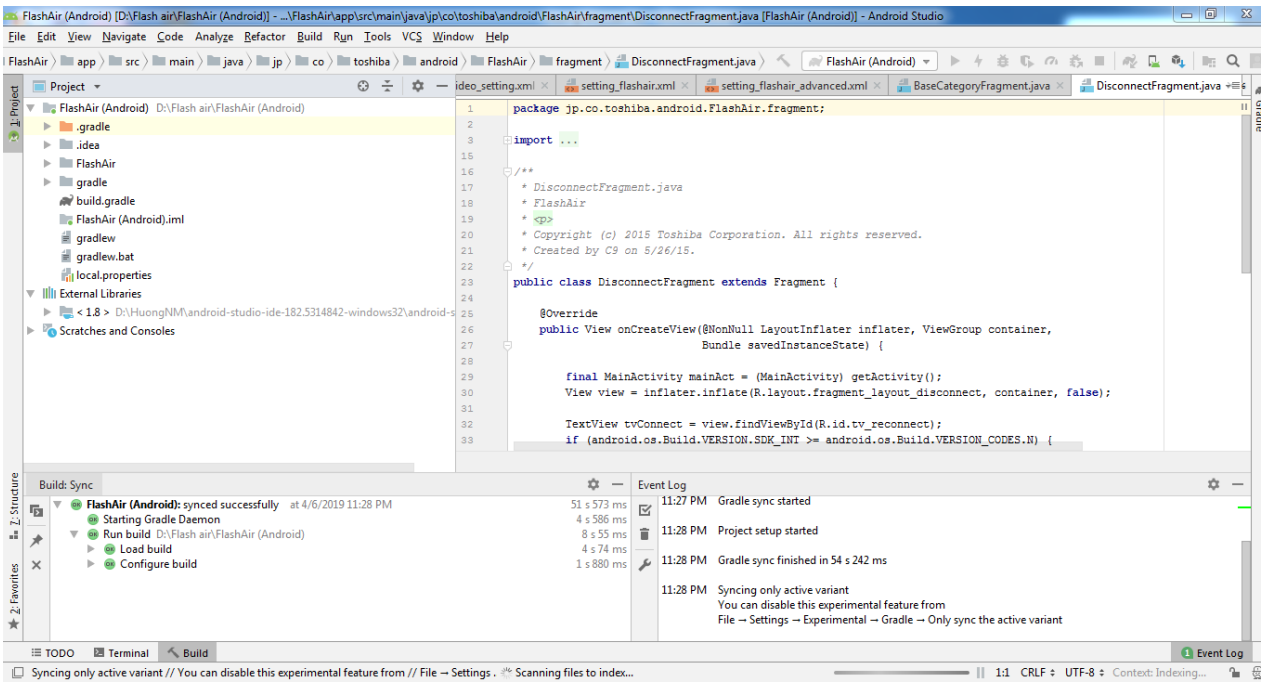
mã mạnh mẽ của IntelliJ, Android Studio còn cung cấp nhiều tính năng hơn giúp nâng cao năng suất khi xây dựng các ứng dụng Android.

Mỗi dự án trong Android Studio chứa một hoặc nhiều mô-đun với tệp mã nguồn và tệp tài nguyên (Hình 4.1). Các loại mô-đun bao gồm:

- Android app modules
- Library modules
- Google App Engine modules

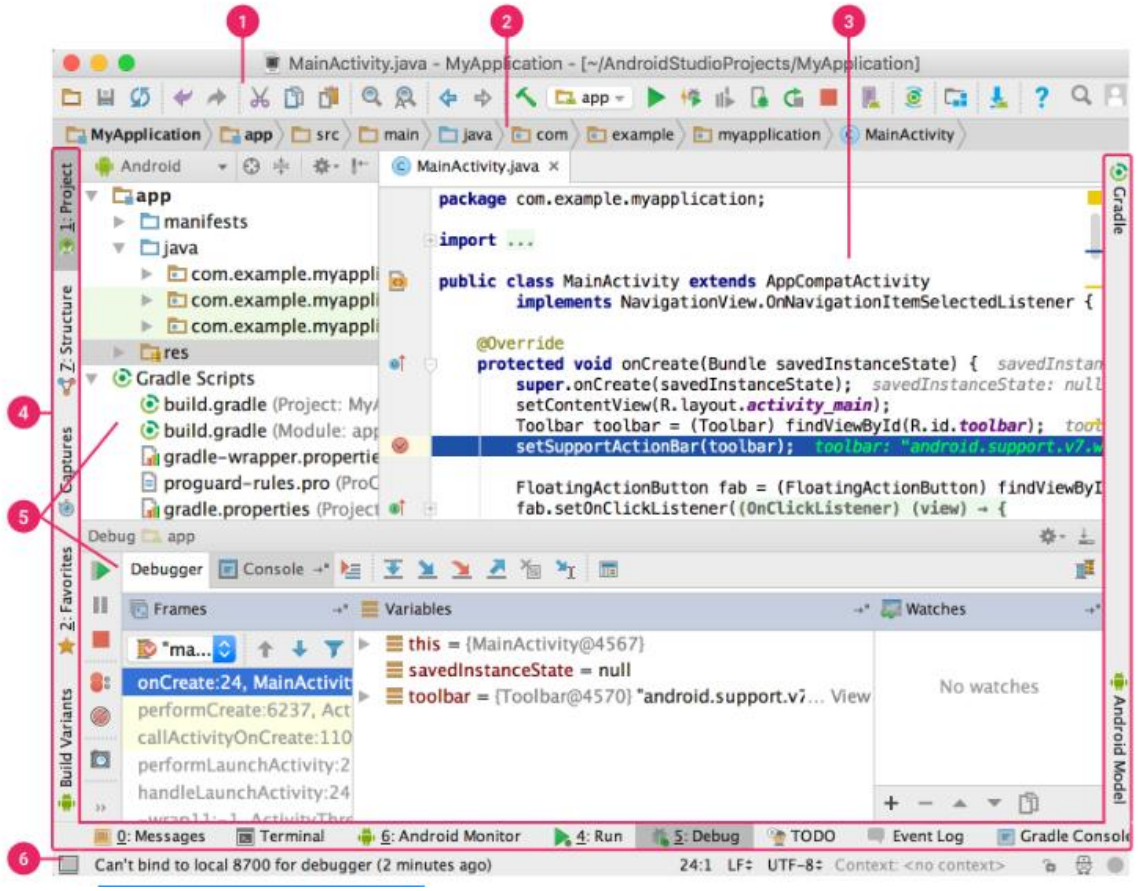
Tất cả các tệp xây dựng được hiển thị ở cấp cao nhất trong Gradle Scriptsvà mỗi mô-đun ứng dụng chứa các thư mục sau:

- **manifests:** Chứa tập tin AndroidManifest.xml. Đây là tập tin khai báo tất cả các Activity được sử dụng trong project. Ngoài ra, tập tin này còn có chức năng cấp quyền cho ứng dụng, chẳng hạn cấp quyền truy cập internet, …
- **java:** Chứa tất cả các tập tin java. Mỗi một Activity được tạo ra sẽ tương ứng cho một tập tin java và tập tin này sẽ được chứa ở đây. Đây cũng là nơi lập trình viên viết code cho ứng dụng.
- **assets:** Mặc định thư mục này chưa tồn tại. Lập trình viên phải tạo thư mục này. Trong thư mục này có thể chừa hình, txt, …
- **res:** Chứa layout, menu, animation, …Khi một Activity được tạo, đồng thời Android sẽ tạo 2 tập tin. Một là tập tin java, chứa code. Một là tập tin xml, chứa layout. Ngoài ra, khi chúng ta tạo menu hoặc tạo animation (anim), cũng sẽ được thực hiện ở đây.



Hình 3.2 Các tập tin trong Android studio

Giao diện của Android Studio được tạo thành từ một số khu vực logic được xác định trong Hình 4.2.



Hình 3.3 Giao diện chính của Android studio

1. Thanh công cụ cho phép thực hiện một loạt các hành động, bao gồm chạy ứng dụng và khởi chạy các công cụ Android.
2. Thanh điều hướng giúp điều hướng qua dự án và mở các tệp để chỉnh sửa. Nó cung cấp một cái nhìn nhỏ gọn hơn về cấu trúc có thể nhìn thấy trong cửa sổ Project.
3. Cửa sổ soạn thảo là nơi tạo và sửa đổi mã. Tùy thuộc vào loại tệp hiện tại, trình chỉnh sửa có thể thay đổi. Ví dụ, khi xem tệp bố cục, trình chỉnh sửa sẽ hiển thị trình chỉnh sửa bố cục.
4. Thanh cửa sổ công cụ chạy xung quanh bên ngoài cửa sổ IDE và chứa các nút cho phép mở rộng hoặc thu gọn các cửa sổ công cụ riêng lẻ.
5. Các cửa sổ công cụ cung cấp quyền truy cập vào các tác vụ cụ thể như quản lý dự án, tìm kiếm, kiểm soát phiên bản và có thể mở rộng chúng và thu gọn chúng.
6. Thanh trạng thái hiển thị trạng thái của dự án và chính IDE, cũng như bất kỳ cảnh báo hoặc thông báo nào.

- **Tích hợp Robotium vào Android Studio**

Robotium là một công cụ viết bằng mã nguồn mở dùng để kiểm thử hộp xám tự động dành cho các ứng dụng trên điện thoại hệ điều hành android. Với sự hỗ trợ của Robotium, chúng ta có thể viết các trường hợp kiểm thử về test chức năng, hệ thống, và, bao phủ các tính năng của Android. Robotium thể được sử dụng cả cho các ứng dụng test sử dụng mã nguồn có sẵn và các ứng dụng mà chỉ có sẵn file apk.

Sử dụng Robotium để tiến hành thử nghiệm trên ứng dụng Android. Để đảm bảo chất lượng ứng dụng Android, thông thường sẽ theo quy trình dưới đây:

1. Thiết kế kiểm tra đặc điểm kỹ thuật
2. Xây dựng chương trình kiểm tra
3. Thực thi Test Case trên thiết bị
4. Thu thập kết quả kiểm tra

Để chạy Robotium trong dự án kiểm thửAndroid, kiểm thử viên sẽ thêm dòng sau vào dependencies của tệp build. gradle bên trong (tệp này được đặt ở cùng cấp với thư mục src), thay phiên bản của robotium cho phù hợp với bản cài trên máy tính:

```
dependencies {
```

```
// Unit testing dependencies
androidTestCompile 'com. jayway. android. robotium:robotium-solo:5.6.3'
androidTestCompile 'junit:junit:4.12'

}
```

Tại thời điểm viết bài này, phiên bản Robotium được sử dụnglà 5.6.3

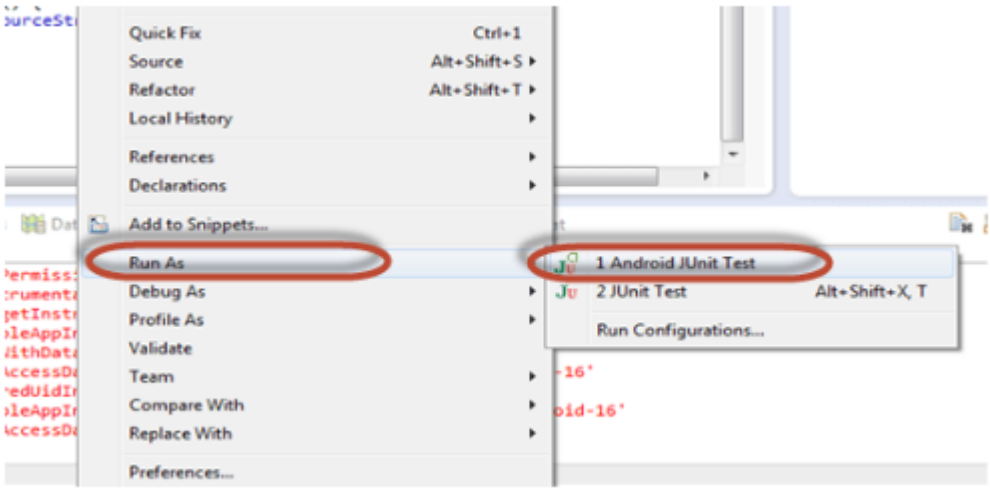
Ví dụ về kiểm thử Android khi dùng Robotium có tại Hình 4.3.

```
/* test Target application contains a text display "Hello World!"*/
public void testSearchText() {
    assertEquals(resourceString,(String) mView.getText());
}

/* test HelloAndroid Activity on target application is exist*/
public void testCurrentActivity() throws Exception  {
    solo.assertCurrentActivity("wrong activity", HelloAndroid.class);
}

/* test Application UI contains "Start" button */
/* send event click button to target application */
public void testSearchButton() throws Exception {
    boolean found = solo.searchButton("Start");
    solo.clickOnButton("Start");
    assertTrue(found);
}
```

Hình 3.4 Ví dụ về kiểm thử Android khi dùng Robotium

Sau khi viết xong chương trình kiểm tra, và chạy thử bằng các bước bên dưới. Kết nối thiết bị Android với PC (hoặc khởi độngtrình giả lập như emulator trong trường hợp không có thiết bị thực).

Trong IDE, nhấp chuột phải vào Run as ->Android JUnit Test



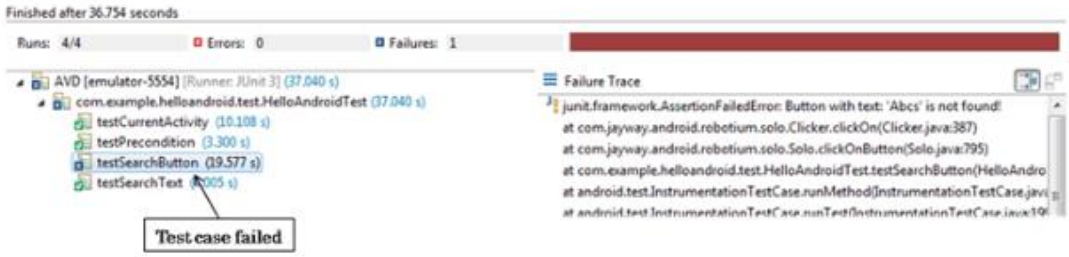
Hình 3.5 Chạy chương trình thử nghiệm Robotium

Sau khi thực hiện kiểm tra, kiểm thử viên sẽnhận được kết quả kiểm tra.

Trong chương trình thử nghiệm này, như Hình 3.6 thì tất cả các trường hợp thử nghiệm được thông qua.

Hình 3.6 Trường hợp thử nghiệm thành công

Trong trường hợp thử nghiệm thất bại như trong Hình 3.7 đầu ra được hiển thị và chothấy trường hợp thử nghiệm nào thất bại.



Hình 3.7 Trường hợp kiểm thử thất bại.

- **Flashair**

Flashair là ứng dụng chạy trên nền tảng Android và IOS. Trong luận văn này chỉ đề cập đến việc chạy ứng dụng này trên nền tảng Android. Ứng dụng này đi kèm với thẻ nhớ có tên là Flashair.

Ứng dụng này có những chức năng như sau:

- Kết nối với thẻ nhớ Flashair
- Xem ảnh, video, nghe nhạc trực tiếp từ thẻ nhớ
- Tải ảnh từ ứng dụng về thiết bị
- Chỉnh sửa ảnh
- Cài đặt mật khẩu cho thẻ

- Chia sẻ ảnh thông qua việc phát wifi và thông qua các mạng xã hội

3.4. Dữ liệu thực nghiệm

Trong thực nghiệm, chúng tôi sử dụng kiểm thử các chức năng của ứng dụng Flashair trên thiết bị di động Android. Hình 3.8 minh họa giao diện Flashair.

Hình 3.8 Giao diện ứng dụng Flashair

3.5. Thử nghiệm trên dữ liệu thực tế

Chúng tôi thực hiện hai thí nghiệm sau với mục đích làm rõ vai trò của việc sử dụng luật toán tử đột biến khi kiểm thử ứng dụng

Thí nghiệm 1

Áp dụng kỹ thuật toán tử đột biến ý định, cụ thể ở đây là Intent Mutation Operators (Toán tử đột biến ý định). Khi kiểm thử chức năng chỉnh sửa ảnh, ảnh sau khi được chỉnh sửa sẽ có chức năng Save. Người sử dụng sẽ nhận được dialog thông báo có muốn lưu ảnh sau khi chỉnh sửa hay không.

Mô tả testcase ở thí nghiệm 1:

Mô tả	Các bước thực hiện	Kết quả mong đợi
Kiểm tra dialog thông báo lưu ảnh sau khi chạy chương trình biến đổi.	1: Mở ứng dụng Flashair. 2: Click ảnh bất kỳ. 3: Chọn chức năng chỉnh sửa ảnh. 4: Chỉnh sửa ảnh và lưu lại.	4. Không hiển thị nội dung trong dialog lưu ảnh.

Hàm save như sau:

```
public void onActivityBackPressed () {
Intent intent = new Intent ();
intent. putExtra (IS_DOWNLOAD_IMAGE_FROM_EDIT, Successfully saved);
mActivity. setResult (Activity. RESULT_OK, intent);
this. mIsResetAsyncTask = true;
}
```

Sử dụng kỹ thuật Intent Payload Replacement (IPR) để kiểm thử:

```
public void onActivityBackPressed () {
Intent intent = new Intent ();
intent. putExtra (IS_DOWNLOAD_IMAGE_FROM_EDIT, ””);
mActivity. setResult (Activity. RESULT_OK, intent);
this. mIsResetAsyncTas = true;
}
```

Thí nghiệm 2

Áp dụng kỹ thuật Xử lý toán tử đột biến, cụ thể ở đây làOnClick Event Replacement (ECR). Trong ứng dụng Flashair, Chức năng cài đặt có thể ON/OFF nhiều chức năng nhỏ khác.

Để kiểm thử chức năng này chúng tôi sử dụng kỹ thuật OnClick Event Replacement (ECR). Trong màn hình chính có các chế độ xem theo Ngày/ Tháng/ Năm. Để kiểm thử chức năng dựa trên kỹ thuật này chúng tôi sẽ thay thế trình sự kiện

của nút Ngày bằng trình sự kiện của nút Năm, theo đó sau khi kiểm thử xong bấm vào nút Ngày sẽ hiển thị toàn bộ thông tin ảnh của năm đó, và được hiện thị theo các Năm khác nhau.

Mô tả testcase ở thí nghiệm 2:

Mô tả	Các bước thực hiện	Kết quả mong đợi
Kiểm tra hoạt động của các chức năng lọc ảnh theo Ngày/Tháng/Năm khi chạy chương trình biến đổi.	1: Mở ứng dụng Flashair. 2: Chọn chức năng lọc ảnh theo Ngày. 3: Kiểm tra chức năng lọc ảnh theo Ngày.	3. Hiển thị tất cả những ảnh được lọc theo Năm

```
public DisplayImage ()
{
public void onClick (ViewDay) {
int id = v. getId ();
ViewImageDay ();
}
```

Chương trình sau khi kiểm thử đột biến:

```
public DisplayImage ()
{
public void onClick (ViewDay) {
int id = v. getId ();
ViewImageYear ();
}
```

Thí nghiệm 3

Áp dụng kỹ thuật toán tử đột biến XML, cụ thể ở đây là kỹ thuật Button Widget Switch (BWS).

Mô tả testcase ở thí nghiệm 3:

Mô tả	Các bước thực hiện	Kết quả mong đợi
Kiểm tra hoạt động của chức năng Album khi chạy chương trình biến đổi.	1: Mở ứng dụng Flashair. 2: Click Album ở phía trên bên phải của màn hình. 3: Kiểm tra chức năng Album	3. Chức năng Album vẫn hoạt động bình thường.

Button SelectButton = (Button) solo. getView (id. Select);

Button AlbumButton = (Button) solo. getView (id. Album);

int [ ] locationOfSelect = new int [2];

int [ ] locationOfAlbum = new int [2];

SelectButton. getLocationInWindow (locationOfSelect);

AlbumButton. getLocationInWindow (locationOfAlbum);

assertTrue (Select button is on the left of AlbumlocationOfSelect[0]>locationOfAlbum[0]);

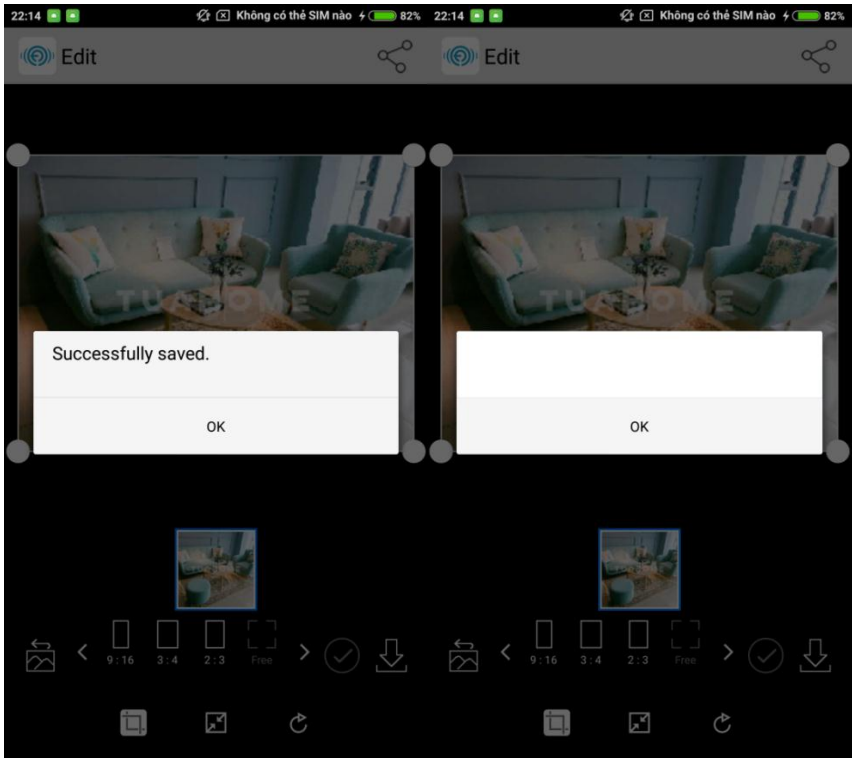
3.6. Kết quả và Đánh giá

3.6.1. Kết quả sau khi chạy thực nghiệm

Sau khi thực nghiệm với hai thí nghiệm 1,2 và 3 chúng tôi thu được kết quả như sau đây.

Đối với thí nghiệm 1

Trước khi kiểm thử, bấm vào biểu tượng lưu hình ảnh, dialog sẽ hiển thị lên màn hình. Sau khi kiểm thử, bấm vào biểu tượng lưu hình ảnh, dialog sẽ không hiển thị lên màn hình.

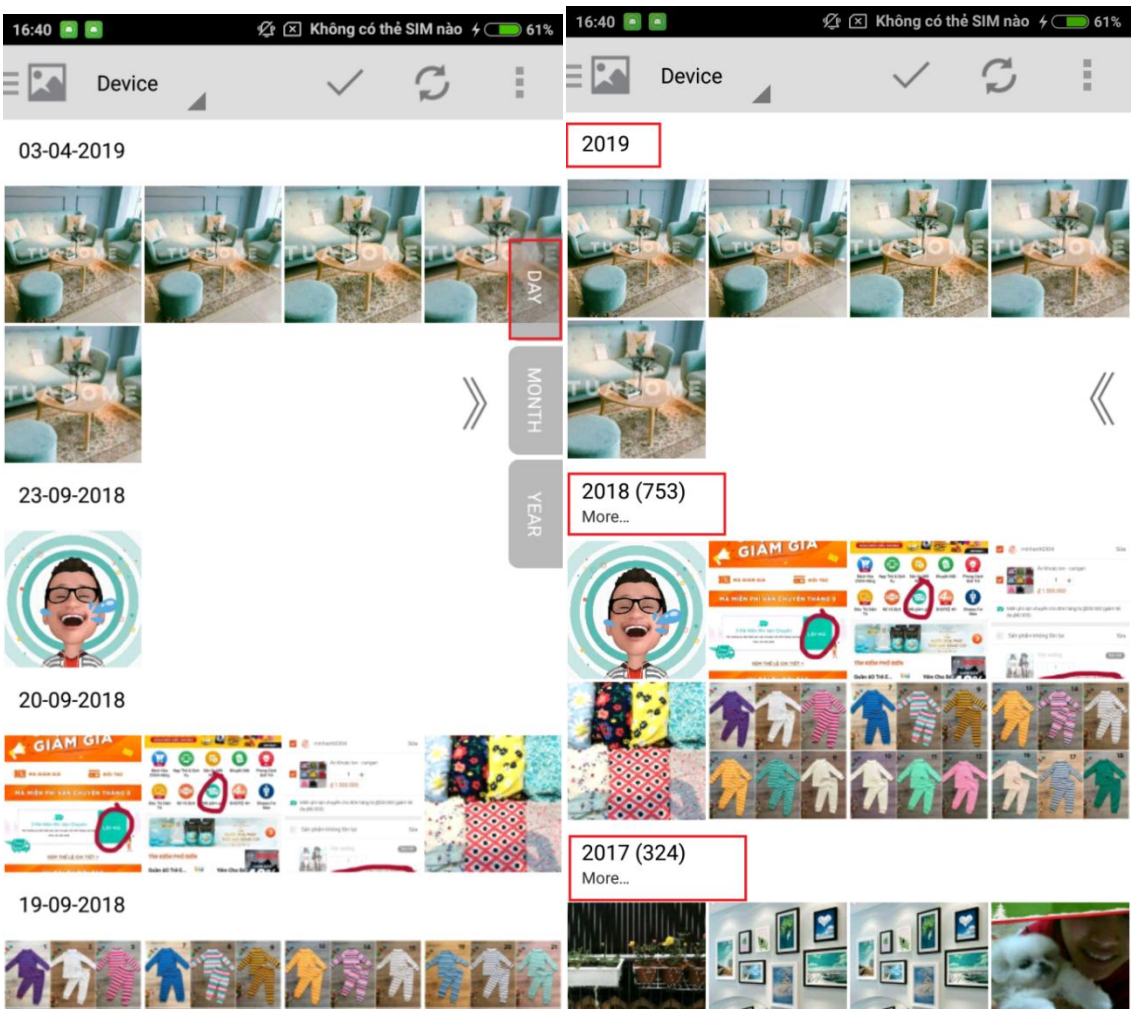


Hình 3.9 Chức năng chỉnh sửa ảnh trước và sau khi kiểm thử.

Kết luận: Trường hợp kiểm thử này đã thành công sau khi áp dụng kỹ thuật đột biến Intent Mutation Operators.

Đối với thí nghiệm 2

Trước khi kiểm thứ, người sử dụng sẽ bấm vào nút DAY để xem ảnh theo từng ngày. Tuy nhiên sau khi kiểm thử, người sử dung bấm DAY để xem ảnh theo từng năm.

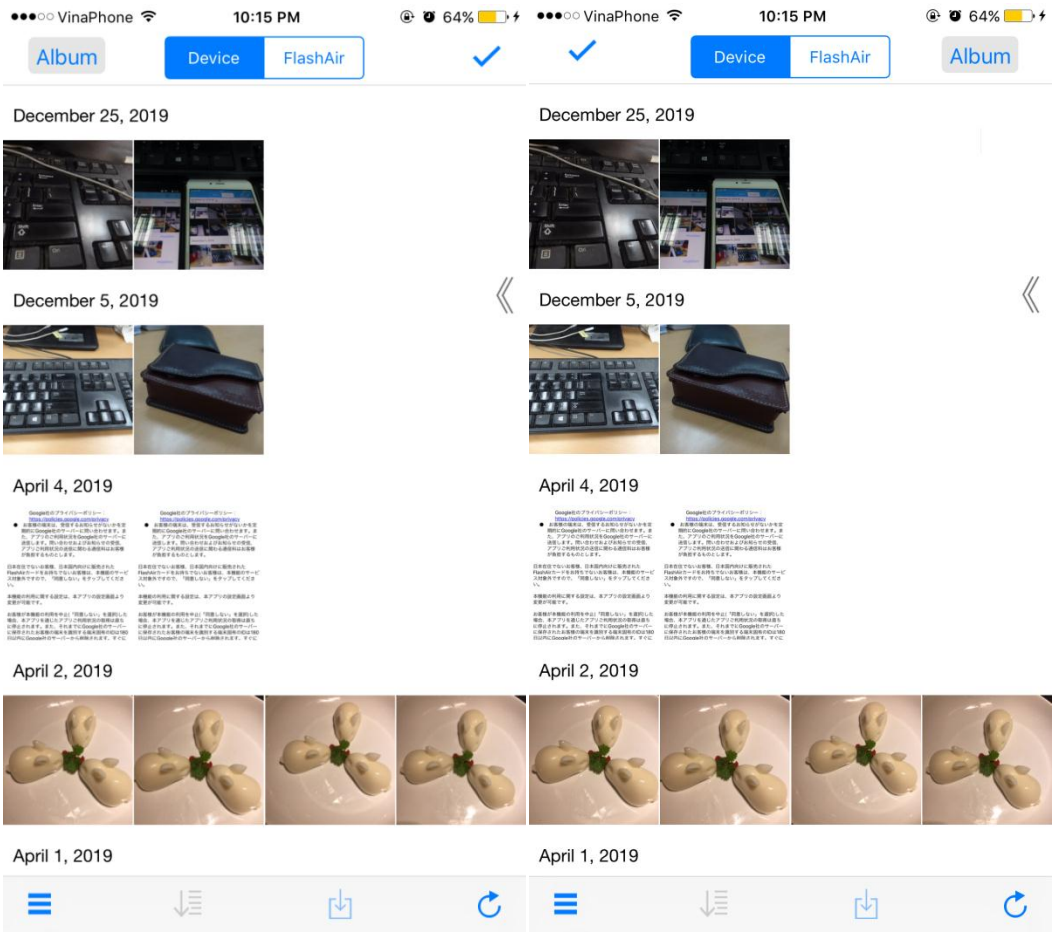


Hình 3.10 Chức năng xem ảnh theo Ngày/Tháng/ Năm và sau khi kiểm thử

Kết luận: Trường hợp kiểm thử này đã thành công sau khi áp dụng kỹ thuật đột biến OnClick Event Replacement.

Đối với thí nghiệm 3

Sau khi đổi chỗ nút Select và Album, hành vi và cách thức hoạt động của hai nút không có sự khác biệt so với ban đầu.



Hình 3.11 Chức năng chọn ảnh và Album trước và sau khi kiểm thử

Kết luận: Trường hợp kiểm thử này đã thành công sau khi áp dụng kỹ thuật đột biến Button Widget Switch.

3.6.2. Đánh giá kết quả các thực nghiệm

Sau khi áp dụng các phương pháp và kỹ thuật kiểm thử đột biến vào thực tế, chúng ta nhận thấy có nhiều sự đổi mới so với những phương pháp kiểm thử trước đây. Hơn thế nữa khi kiểm thử đột biến, các kiểm thử viên có thể tạo ra nhiều trường hợp kiểm thử khác nhau và kiểm thử sâu hơn về chức năng của ứng dụng.

3.7. Tóm tắt chương 3

Trong chương 3, luận văn đã ứng dụng các phương pháp và kỹ thuật về kiểm thử đột biến như đã trình bày ở chương 2 vào trong thực tế. Phần mềm Flashair đã được phát triển trong thời gian dài vì vậy khi kiểm thử có một số mặt hạn chế như các trường hợp kiểm thử thất bại thường khó xảy ra. Tuy vậy các trường hợp kiểm thử trong chương 3 đều được mô tả rõ và ứng dụng các phương pháp, kỹ thuật kiểm thử đột biến một cách tốt nhất.

KẾT LUẬN VÀ ĐỊNH HƯỚNG NGHIÊN CỨU TIẾP THEO

Qua tìm hiểu về kiểm thử phần mềm và kiểm thử đột biến trên ứng dụng Android, luận văn đã áp dụngkiến thức về toán tử đột biến và những phương pháp về toán tử đột biến để kiểm thử phần mềm [1] để nhằm nâng cao chất lượng cho ứng dụng Android. Luận văn đã đạt được các kết quả sau đây:

- Áp dụng kỹ thuật toán tử đột biến, phương pháp xử lý vòng đời toán tử đột biến và sử dụng kỹ thuật toán tử đột biến XML để kiểm thử ứng dụng Android.
- Kiểm thử chức năng của phần mềm Flashair như chỉnh sửa ảnh, đồng ý nhận thông báo từ nhà phát triển và chức năng chọn ảnh, xem album từ màn hình chính.

Tuy nhiên do hạn chế về thời gian nên luận văn vẫn còn một số hạn chế sau như:

- Chưa kiểm thử được nhiều chức năng của phần mềm.
- Chưa kiểm thứ nhiều phần mềm chạy trên nền tảng Android.

Trong thời gian sắp tới, chúng tôi sẽ thực hiện với nhiều dữ liệu, cụ thể ở đây là nhiều ứng dụng chạy trên nền tảng Android để nâng cao kết quả thực nghiệm.

TÀI LIỆU THAM KHẢO

[1] Lin Deng, Jeff Offutt, Paul Ammann, Nariman Mirzaei. *Mutation operators for testing Android apps*. Information & Software Technology 81: 154-168 (2017).

[2] Lin Deng, Jeff Offutt, David Samudio. *Is Mutation Analysis Effective at Testing Android Apps?* QRS 2017: 86-93.

[3] Birgitta Lindström, Jeff Offutt, Daniel Sundmark, Sten F. Andler, Paul Pettersson. *Using mutation to design tests for aspect-oriented models*. Information & Software Technology 81: 112-130 (2017)

[4] Birgitta Lindström, Sten F. Andler, Jeff Offutt, Paul Pettersson, Daniel Sundmark. *Mutating aspect-oriented models to test cross-cutting concerns*. ICST Workshops 2015: 1-10

[5] Reyhaneh Jabbarvand, Sam Malek. *µDroid: an energy-aware mutation testing framework for Android*. ESEC/SIGSOFT FSE 2017: 208-219

[6] Macario Polo Usaola, Gonzalo Rojas, Isyed Rodriguez, Suilen Hernandez. *An Architecture for the Development of Mutation Operators*. ICST Workshops 2017: 143-148

[7] Hrushikesha Mohanty, J. R. Mohanty, Arunkumar Balakrishnan. Trends in Software Testing. Springer Singapore, 2017

[8] Mike Papadakis, Marinos Kintis, Jie Zhang, Yue Jia, Yves Le Traon, Mark Harman. Mutation Testing Advances_An Analysis and Survey
http://orbilu. uni. lu/bitstream/10993/31612/1/Mutation-Survey-06-17. pdf

[9] Paul Ammann, Jeff Offutt. Introduction to Software Testing, Cambridge University Press, 2017

[10] Ali Mili, Fairouz Tchier. Software Testing_ Concepts and Operations. Wiley, 2015