Mục lục

Đặt vấn đề

Như chúng ta đã biết nhu cầu sử dụng các thiết bị di động thông minh của con người ngày càng cao, số lượng các nhà sản xuất thiết bị cũng ngày càng nhiều và đa dạng hơn về chủng loại, mẫu mã. Mỗi chiếc điện thoại thông minh ngày nay không đơn thuần chỉ để nghe, gọi và nhắn tin như trước, mà nó giống như một máy tính để bàn thu nhỏ, chúng ta có thể lướt web, chat wifi, mua hàng trực tuyến, tìm kiếm thông tin, xử lý thông tin mạng, kết nối thiết bị ngoại vi, điều khiển ô tô, điều khiển robot, giải quyết công việc với đối tác ở bất kì nơi đâu và vô vàn những lợi ích lớn lao khác.

Để các thiết bị di động có được sức mạnh như vậy trước hết là nhờ các công ty phát triển phần mềm mà cụ thể ở đây là Google với Android, Apple với iOS và Microsoft với Windows Phone. Các công ty này đã tập trung lớn nguồn lực của họ vào việc phát triển các nền tảng di động kể trên để đưa chúng lên một tầm cao hơn trước đây, mà ngày nay chúng ta thường hay gọi là "HỆ ĐIỀU HÀNH".

Trong số các hệ điều hành cho các thiết bị di động, Android hiện đang là hệ điều hành phổ biến và lớn mạnh nhất. Với tính chất nguồn mở, Android thu hút nhiều nhà sản xuất thiết bị trên thế giới như Sony, Samsung, LG, HTC, v.v… Ngày nay Android không chỉ được sử dụng là hệ điều hành

trên điện thoại thông minh và máy tính bảng, mà còn được ứng dụng vào các dự án khác như: đồng hồ thông minh (smartwatch), nhà thông minh (smart home), tivi thông minh (smart tivi), robot, điều khiển ô tô, kính thực thể ảo …

Theo số liệu thống kê, Android hiện đang nắm giữ 86% [1] tổng thị phần cho hệ điều hành di động. Sự phổ biến ngày càng tăng của các thiết bị Android có tác động trực tiếp đến cửa hàng ứng dụng Google Play. Đây là cửa hàng ứng dụng lớn nhất thế giới và có 3.3 triệu ứng dụng (tháng 9/2017) có sẵn để tải xuống. Chỉ riêng trong quý 2 năm 2017, đã có gần 17 tỷ lượt tải xuống các ứng dụng từ Google Play. Với những con số thống kê như trên, có thể thấy việc xây dựng các ứng dụng cho thiết bị Android đã, đang và sẽ vẫn là một xu hướng phát triển mạnh mẽ và bền vững.

Trong vòng đời phát triển phần mềm, kiểm thử là một hoạt động quan trọng và không thể bỏ qua đối với phần mềm nói chung và các ứng dụng Android nói riêng. Hoạt động kiểm thử có thể tiến hành một cách thủ công tuy nhiên điều này sẽ mất thời gian, tốn chi phí và đôi khi không mang lại hiệu quả cao. Thậm chí trong một vài những phương pháp kiểm thử, hoạt động kiểm thử thủ công là không thể thực hiện được. Do đó đòi hỏi phải có một hình thức kiểm thử tự động hỗ trợ. Tuy nhiên kiểm thử tự động cũng có nhiều kỹ thuật khác nhau với

các mức độ tự động khác nhau. Đối với nhiều các công cụ kiểm thử, vẫn cần có sự tham gia của kiểm thử viên vào trong quá trình. Kiểm thử viên sẽ phải xây dựng các kịch bản kiểm thử hoàn toàn thủ công để các công cụ kiểm thử có thể thực thi được từ các kịch bản đó. Đây là một công việc không hề đơn giản, tốn thời gian và nhân lực. Vậy một câu hỏi được đặt ra, làm sao để hoạt động kiểm thử hoàn toàn tự động, từ thao tác sinh ra các kịch bản kiểm thử cho đến việc thực thi những kịch bản kiểm thử đó. Đã có rất nhiều các nghiên cứu về các kỹ thuật sinh dữ liệu kiểm thử tự động. Và trong nội dung luận văn này cũng sẽ tìm hiểu về các kỹ thuật sinh dữ liệu kiểm thử tự động, và cụ thể nó được áp dụng vào quá trình kiểm tra tự động cho các ứng dụng Android ra sao.

Chương 1. Nền tảng Android

1.1. Giới thiệu chung về Android

Android là hệ điều hành mã nguồn mở, dựa trên Linux, được tạo ra cho một loạt các thiết bị và yếu tố hình thức bao gồm các thành phần chính:

- Tầng hạt nhân Linux
- Lớp trừu tượng phần cứng (Hardware Abstraction Layer - HAL)
- Thời gian chạy Android (Thời gian chạy Android – ART)
- Tầng thư viện C/C++
- Tầng khung Java API

1.2. Bản kê khai ứng dụng AndroidManifest

1.2.1. Hoạt động (activity)

Hoạt động là thành phần phụ trách giao diện người dùng của ứng dụng. Một hoạt động là một giao diện người dùng trực quan mà người dùng có thể thực hiện trên đó mỗi khi được kích hoạt. Một ứng dụng có thể có nhiều hoạt động và chúng có thể gọi đến nhau và chuyển giữa các hoạt động với nhau. Mỗi hoạt động là một dẫn xuất của lớp android.app.Activity.

1.2.2. Dịch vụ (service)

Một dịch vụ (service) là các đoạn mã được thực thi ngầm bởi hệ thống mà người sử dụng không thấy được. Mỗi dịch vụ đều được mở rộng từ lớp cơ sở là dịch vụ trong gói android.app.

1.2.3. Bộ nhận quảng bá (Broadcast Receiver)

Bộ nhận quảng bá là một thành phần không làm gì cả nhưng nó nhận và phản hồi lại các thông báo quảng bá.

1.2.4. Trình cung cấp nội dung (Content Provider)

Trình cung cấp nội dung có chức năng cung cấp một tập hợp các phương thức cho phép một ứng dụng có thể lưu trữ và lấy dữ liệu được quản lý bởi trình cung cấp nội dung đó.

Chương 2. Sinh đầu vào kiểm thử tự động

2.1. Phương pháp kiểm thử Fuzz (Fuzzing)

2.1.1. Kiểm thử Fuzz là gì??

Kiểm thử Fuzz là một kỹ thuật phát hiện lỗi của phần mềm bằng cách tự động hoặc bán tự động sử dụng phương pháp lặp lại thao tác sinh dữ liệu sau đó chuyển cho hệ thống xử lý.

2.1.2. Các giai đoạn của kiểm thử Fuzz

- Bước 1: Xác định mục tiêu (Indentify target)
- Bước 2: Xác định đầu vào (Indentify inputs)
- Bước 3: Sinh dữ liệu mờ hay còn gọi là tạo các ca kiểm thử (generate fuzzed data)
- Bước 4: Thực thi dữ liệu mờ (execute fuzzed data)
- Bước 5: Giám sát dữ liệu mờ (monitor for execution fuzzed data)
- Bước 6: Xác định khả năng khai thác (determine exploitability)

2.1.3. Phân loại kiểm thử Fuzz

- Kiểm thử Fuzz dựa trên đột biến (Mutation based Fuzzing)
- Kiểm thử mờ dựa trên thế hệ (Generation based Fuzzing)

2.1.4. Các lỗ hổng được phát hiện bởi kiểm thử Fuzz

Kiểm thử Fuzz làm việc tốt nhất đối với các vấn đề mà gây ra lỗi của một chương trình như là: tràn bộ nhớ (buffer overflow), kịch bản hóa chéo trang(XSS), từ chối dịch vụ (DoS), lỗi chuỗi định dạng (Format String Errors), chèn câu truy vấn (SQL Injection) v.v... Vì thế với kiểm thử Fuzz người ta có thể kiểm tra sự an toàn của bất kỳ quá trình, các dịch vụ, thiết bị, hệ thống, hoặc mạng máy tính v.v…

2.1.5. Ưu nhược điểm của kiểm thử Fuzz

2.1.6. Một số công cụ kiểm thử Fuzz

2.2. Phương pháp dựa trên mô hình (Model based Testing)

2.2.1. Kiểm thử dựa trên mô hình là gì?

2.2.1.1. Mô hình là gì?

2.2.1.2. Kiểm thử dựa trên mô hình

Kiểm thử dựa trên mô hình là một kỹ thuật kiểm tra, nơi mà hành vi thời gian chạy của một phần mềm kiểm thử được kiểm tra để chống lại những dự đoán được thực hiện bởi một đặc tả hoặc mô hình chính thức.

2.2.2. Các loại MBT

- Offline/ a priori: Sinh ra các bộ kiểm thử trước khi thực thi chúng. Bộ kiểm thử chính là tập hợp của các ca kiểm thử
- Online/ on-the-fly: Sinh ra các bộ kiểm thử ngay trong khi thực thi kiểm thử.

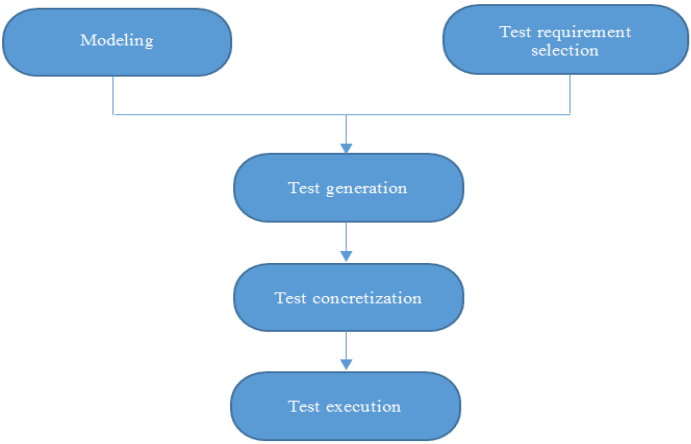
2.2.3. Các mô hình khác nhau trong kiểm thử

2.2.1.1. Máy trạng thái hữu hạn

2.2.1.2. Biểu đồ trạng thái

2.2.1.3. Ngôn ngữ mô hình hóa thống nhất (UML)

2.2.4. Tiến trình kiểm thử dựa trên mô hình



Hình 2.12: Các bước trong kiểm thử dựa trên mô hình

2.2.4.1. Mô hình hóa

Tiến hành việc xây dựng một mô hình cho hệ thống kiểm thử dựa trên các nền tảng là các yêu cầu

2.2.4.2. Lựa chọn yêu cầu kiểm thử:

Các thao tác được thực hiện trong bước này là:

- Các tiêu chí lựa chọn trường hợp kiểm thử được xác định
- Các tiêu chí lựa chọn trường hợp kiểm thử sau đó được chuyển đổi thành các đặc tả ca kiểm thử

Một số các tiêu chí bao phủ chính sử dụng để đánh giá cho bộ kiểm thử được sinh ra:

- Tiêu chí bao phủ cấu trúc
- Tiêu chí bao phủ dữ liệu
- Tiêu chí dựa trên lỗi
- Tiêu chí bao phủ yêu cầu

2.2.4.3. Sinh kiểm thử

Một khi mô hình và đặc tả ca kiểm thử đã được xác định, một bộ kiểm thử trừu tượng sẽ được tạo ra

2.2.4.4. Cụ thể hóa kiểm thử (chuyển đổi)

Trong bước này, thực hiện cụ thể hóa những bộ kiểm thử trừu tượng được sinh ở bước trên thành các kịch bản kiểm thử có thể thực thi được bằng công cụ.

2.2.4.5. Thực thi kiểm thử

Thực thi kịch bản kiểm thử, kết quả thực tế đầu ra sẽ được so sánh với các kết quả mong đợi từ đó mà đưa ra được những kết quả Pass, Fail cho từng kịch bản kiểm thử.

2.2.5. Ưu nhược điểm của kiểm thử dựa trên mô hình

2.2.6. Một số công cụ kiểm thử dựa trên mô hình

Chương 3. Một số công cụ sinh đầu vào kiểm thử tự động cho ứng dụng Android

3.1. Công cụ kiểm thử ngẫu nhiên – Monkey tool

3.1.1. Tổng quan chung về Monkey tool

Monkey là một phần của Android SDK được phát triển bởi Google sử dụng cho việc kiểm thử tự động các ứng dụng Android. Với việc tích hợp sẵn trong Android Studio, Monkey là một công cụ hữu ích cho các lập trình viên trong việc kiểm tra ứng dụng ngay trong quá trình phát triển. Nó sử dụng kỹ thuật ngẫu nhiên/ mờ trong việc sinh ra các sự kiện người dùng làm đầu vào cho quá trình kiểm thử.

3.1.2. Kiểm thử Fuzz với Monkey

Bước 1: Xác định hệ thống mục tiêu: truyền các tham số về gói và danh mục của ứng dụng cần thực thi kiểm thử vào trong lệnh chạy.

Bước 2: Xác định đầu vào: lựa chọn và thiết lập tỉ lệ các sự kiện mong muốn sinh ra trong quá trình thực thi kiểm thử

Bước 3: Sinh dữ liệu kiểm thử: Monkey sẽ tiến hành việc sinh các đầu vào kiểm thử là các sự kiện tương ứng với yêu cầu được đưa ra.

Bước 4: Thực thi kiểm thử: Các lệnh ADB này được truyền tới thiết bị kiểm thử.

Bước 5: Giám sát hành vi hệ thống: các sự kiện sinh ra đều được lưu lại dưới dạng các tệp tin log

Bước 6: Đăng lỗi và phân tích

3.2. Công cụ kiểm thử dựa trên mô hình – DroidBot

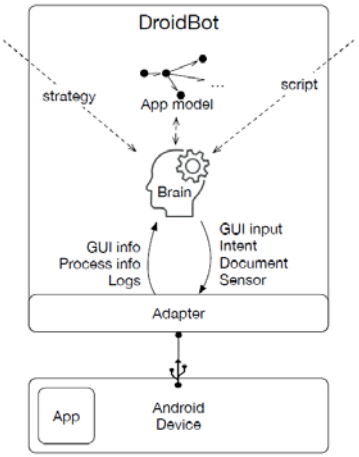
3.2.1. Tổng quan chung về DroidBot

DroidBot là một công cụ sinh đầu vào kiểm thử mã nguồn mở dạng nhẹ dựa trên UI cho các ứng dụng Android, được phát triển bởi Yuanchun Li, nghiên cứu sinh của Học viện phần mềm, Đại học Bắc Kinh.

DroidBot cung cấp bộ sinh đầu vào theo hướng dẫn UI dựa trên mô hình chuyển đổi trạng thái được tạo ra khi đang chạy. Sau đó nó sẽ sinh ra đầu vào kiểm thử theo hướng dẫn UI dựa trên mô hình chuyển tiếp.

Thành phần đầu tiên của DroidBot ở đây là mô đun Adapter dùng để cung cấp tính trừu tượng của thiết bị và ứng dụng kiểm thử.

Mô đun Brain nhận thông tin của thiết bị và ứng dụng được tạo ra từ Adapter trong thời gian chạy và gửi các đầu vào kiểm thử được sinh ra đến Adapter



Hình 3.3. Kiến trúc tổng quan của DroidBot

3.2.3. Kiểm thử dựa trên mô hình với DroidBot

Bước 1: Mô hình hóa

Để thực hiện được việc mô hình hóa, DroidBot lấy các thông tin GUI từ ứng dụng kiểm thử: đối với mỗi UI, DroidBot ghi lại ảnh chụp màn hình và cây phân cấp UI

Bước 2: Lựa chọn yêu cầu kiểm thử

DroidBot tích hợp bốn thuật toán thăm dò khác nhau là naive depth-first, naive breadth-first, greedy depth-first và greedy breadth-first bên cạnh lựa chọn khám phá bằng Monkey

Bước 3: Sinh kiểm thử

Các loại đầu vào kiểm thử được DroidBot hỗ trợ bao gồm đầu vào UI, các intent, tài liệu tải lên và các dữ liệu cảm biến (tín hiệu GPS, v.v…)

Bước 4: Cụ thể hóa kiểm thử

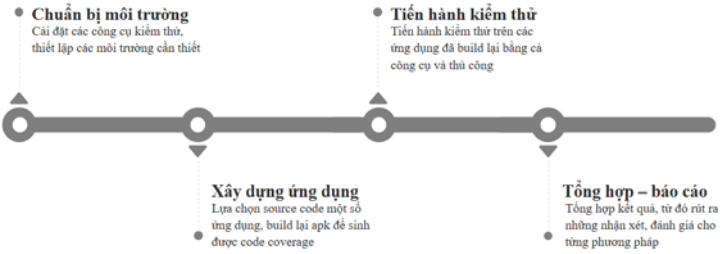
DroidBot tìm nạp các thông tin của thiết bị/ ứng dụng từ thiết bị và gửi đầu vào kiểm thử tới thiết bị thông qua ADB

Bước 5: Thực thi kiểm thử

Các sự kiện sinh ra bởi DroidBot thông qua ADB sẽ được thực thi trên thiết bị kiểm thử

Chương 4: Nghiên cứu thực nghiệm

Trong chương này, chúng ta tiến hành thực hiện một thực nghiệm nhỏ: kiểm thử một số ứng dụng Android với 2 công cụ kiểm thử tự động được giới thiệu ở chương 3 là Monkey và DroidBot và kiểm thử thủ công.



Hình 4.1: Quy trình tiến hành thực nghiệm

4.1. Thiết lập môi trường thực nghiệm

4.1.1. Chuẩn bị công cụ kiểm thử

Cài đặt Monkey và DroidBot, chuẩn bị kiểm thử thủ công

4.1.2. Chuẩn bị thiết bị kiểm thử

Samsung Galaxy Note 5 (N920I), hệ điều hành Android Nougat 7.0

4.2. Xây dựng ứng dụng kiểm thử

Bước 1: Tải mã nguồn ứng dụng

Các ứng dụng này được lấy mã nguồn từ https://f-droid.org/

Bước 2: Sử dụng Jacoco, build lại apk để đo độ bao phủ mã nguồn

4.3. Tiến hành kiểm thử

Bước 1: Cài đặt các ứng dụng và lần lượt thực hiện kiểm tra tự động với Monkey và Droidbot, kiểm tra thủ công bởi người dùng.

Bước 2: lấy tập tin .ec và sinh báo cáo cho độ bao phủ mã nguồn

Bước 3: Tổng hợp số liệu và phân tích kết quả

4.4. Kết quả thực nghiệm

4.5. Phân tích – đánh giá

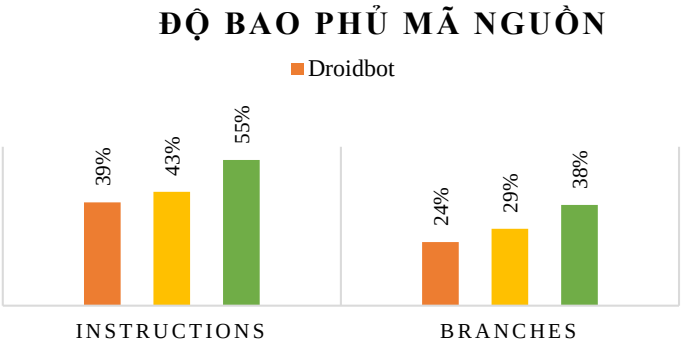
4.5.1. Tính hiệu quả trong việc phát hiện lỗi

Droibot và Monkey đều khá hiệu quả trong việc tìm ra lỗi so với kiểm thử thủ công

4.5.2. Tính hiệu quả trong chiến lược khám phá

Khi thực thi với số lượng sự kiện nhỏ thì khả năng bao phủ mã nguồn của Monkey sẽ tốt hơn so với DroidBot. Mặc dù

vậy, các công cụ tự động đều có hạn chế khi gặp phải những giao diện có trường nhập thông tin chứa các yêu cầu đặc biệt.



4.5.3. Tính khả dụng

Cả Monkey và DroidBot đều là các công cụ chạy bằng dòng lệnh, việc cài đặt và sử dụng không quá phức tạp. Tuy nhiên, với cùng một số lượng sự kiện, thời gian thực hiện của DroidBot lớn hơn rất nhiều so với Monkey

Kết luận

Sau quá trình nghiên cứu và tìm hiểu về đề tài "Nghiên cứu một số phương pháp sinh đầu vào kiểm thử tự động cho Android", các kết quả mà luận văn đã đạt được là:

Đầu tiên, luận văn đã giúp đưa ra một cái nhìn tổng quan về kiểm thử tự động dành cho phần mềm nói chung và kiểm thử tự động cho các ứng dụng Android nói riêng.

Từ cái nhìn tổng quan về kiểm thử tự động, luận văn đã giúp đưa ra khái niệm chi tiết hơn về sinh đầu vào kiểm thử tự động là gì cùng với các kỹ thuật phổ biến đang được sử dụng để sinh đầu vào kiểm thử tự động: phương pháp kiểm thử Fuzz và phương pháp kiểm thử dựa trên mô hình. Đưa ra các ưu, nhược điểm của các phương pháp này để từ đó giúp người đọc có được những đánh giá, so sánh và đưa ra lựa chọn một phương pháp phù hợp cho mục đích sử dụng của mình. Luận văn cũng đã đưa ra những tìm hiểu về một số hướng tiếp cận các phương pháp trên áp dụng cho các ứng dụng Android

Để có cái nhìn cụ thể và chi tiết hơn về hai phương pháp sinh đầu vào kiểm thử tự động được trình bày ở trên, luận văn đã lựa chọn hai công cụ tự động tiêu biểu tương ứng cho hai phương pháp là DroidBot và Monkey để tìm hiểu. Bên cạnh

việc tìm hiểu về lý thuyết, đã tiến hành làm thực nghiệm để so sánh với nhau đồng thời cũng so sánh với việc kiểm thử thủ công. Sau thực nghiệm đã thu được kết quả về số lượng lỗi, độ bao phủ mã nguồn, thời gian thực thi của mỗi công cụ. Từ những kết quả thu được đó đã giúp đưa ra được những so sánh, phân tích và đánh giá cho tính hiệu quả của từng phương pháp kiểm thử.

Tuy nhiên luận văn còn có hạn chế trong việc tiến hành thực nghiệm: số lượng các công cụ kiểm thử còn hạn chế, số lượng ứng dụng lựa chọn chưa phong phú

Với những hạn chế nêu trên, một số hướng mở rộng nghiên cứu và tìm hiểu trong tương lai:

- Mở rộng thực nghiệm với số lượng các công cụ lựa chọn lớn hơn, tiến hành kiểm tra với số lượng ứng dụng nhiều hơn và có độ phức tạp cao hơn, đồng thời kiểm tra với số lượng sự kiện lớn hơn nữa.

- Có kế hoạch cho việc lựa chọn các công cụ thích hợp để cải tiến và phát triển, để áp dụng thực tế vào công việc kiểm thử phần mềm cho các ứng dụng Android tại Samsung.

Tài liệu tham khảo

[1] "Statista," [Online]. Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/.

[2] "Android Developer," [Online]. Available: https://developer.android.com/guide/platform/index.html.

[3] "Android Developer," [Online]. Available: https://developer.android.com/reference/android/app/Activity.html.

[4] "Android Developer," [Online]. Available: https://developer.android.com/guide/components/services.html.

[5] "Android Developer," [Online]. Available: https://developer.android.com/guide/topics/manifest/receiver-element.html.

[6] "Android Developer," [Online]. Available: https://developer.android.com/guide/topics/providers/content-providers.html.

[7] Abel Méndez-Porras, Christian Quesada-López, and Marcelo Jenkins, "Automated Testing of Mobile Applications: A Systematic Map and Review," p. 1.

[8] Hitesh Tahbildar, Bichitra Kalita, "Automated software test data generation: Direction of research," in *International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2*, 2011, pp. 2-3.

[9] WANG Tao, LI Yanling, MA Yingli, GUO Wei, "Research and Application of a New Fuzz-test Framework," p. 2.

[10] "VIBLO," [Trực tuyến]. Available: https://viblo.asia/p/tim-hieu-ve-fuzz-testing-YWOZrDzv5Q0.

[11] "Guru99," [Online]. Available: https://www.guru99.com/fuzz-testing.html.

[12] P. Garg, "Fuzzing - mutation vs generation".*INFOSEC Institute.*

[13] "Tutorials point," [Online]. Available: https://www.tutorialspoint.com/software_testing_dictionary/fuzz_testing.htm.

[14] “Khoa CNTT, Đại học Duy Tân,” [Trực tuyến]. Available: http://kcntt.duytan.edu.vn/Home/ArticleDetail/vn/128/2461/bai-01-so-luoc-ve-fuzzing-testing.

[15] "OWASP," [Online]. Available: https://www.owasp.org/index.php?title=File:CLASP_Vulnerabilities_SDLC_Phases.gif&setlang=en.

[16] "Guru99," [Online]. Available: https://www.guru99.com/fuzz-testing.html.

[17] R. T. M. N. Aravind MacHiry, "Dynodroid: An Input Generation System for Android Apps".

[18] J. R. Raimondas Sasnauskas, "Intent Fuzzer: Crafting Intents of Death".

[19] ANTOJOSEPH, "DROID-FF – THE ANDROID FUZZING FRAMEWORK".

[20] "Guru99," [Online]. Available: https://www.guru99.com/model-based-testing-tutorial.html.

[21] Zoltan Micskei, Istvan Majzik, "Model-based test generation," *Software and Systems Verification (VIMIMA01),* pp. 13-17.

[22] E. Karaman, "Model Based Software Testing," *SWE550 Boğaziçi University,* pp. 7-17.

[23] P. Ana, "Model – based testing," *MDSE – Model Driven Software Engineering,* pp. 3-12.

[24] Shauvik Roy Choudhary, Alessandra Gorla, Alessandro Orso, "Automated Test Input Generation for Android: Are We There Yet?".

[25] "Android Developer," [Online]. Available: https://developer.android.com/studio/test/monkey.html.

[26] Yuanchun Li, Ziyue Yang, Yao Guo, Xiangqun Chen, "DroidBot: A Lightweight UI-Guided Test Input Generator for Android".

[27] [Online]. Available: https://f-droid.org/.