FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Injeção de Defeitos em Aplicações Android

**Liliana Filipa Lobo Ribeiro**


Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ana Cristina Ramada Paiva

July 24, 2017

# Injeção de Defeitos em Aplicações Android

**Liliana Filipa Lobo Ribeiro**

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: João Carlos Pascoal Faria
External Examiner: José Creissac Campos
Supervisor: Ana Cristina Ramada Paiva

July 24, 2017

# Abstract

The number of Android applications is rising at a rate of more than a thousand applications a day in the Android App Store. The problem is that the quality is sometimes neglected in this kind of application, which results in defective software being frequently used. In order to improve the quality of the software it is necessary to create test cases that are adequate to cover all the implementation requirements. However this task is not as trivial as it seems, and for this reason mutation testing techniques are important as they can be useful to assess the quality of the test cases.

This research aims to extend the research work performed in the SE lab in which a tool was developed to test Android applications (iMPAcT Tool).This tool executes test strategies that aim to check whether the guidelines for Android programming are being employed or not. The goal of this work is to analyse the faults that originate the failures detected by the iMPAcT tool and define a set of mutators that can be applied over Android applications and finally assess if the test suites used are effective in finding those failures.

The ten mutation operators that were defined were applied to the source code of several Android applications and then the applications were tested using the iMPAcT tool. After, the results were analysed to verify if the iMPAcT tool was or not able to detect the errors inserted. Even though the iMPAcT tool was not able to detect all the errors inserted, the problem was not related with the set of mutation operators but with the algorithms of exploration of applications and detection of differences in two screens implemented by the iMPAcT tool.

# Resumo

O número de aplicações Android está a aumentar a uma taxa de mais de mil aplicações por dia na loja de aplicações Android. O problema é que a qualidade é, por vezes, negligenciada neste tipo de aplicações, o que resulta no uso de software com defeitos. Para se conseguir melhorar a qualidade do software é necessário que se crie testes que sejam adequados para cobrir todos os requisitos da implementação. Porém esta tarefa não é tão trivial como parece, por isso é que as técnicas de teste de mutação são importantes uma vez que estas são uteis para avaliar a qualidade de um conjunto de testes.

Esta pesquisa tem como objetivo complementar o trabalho de pesquisa realizado no laboratório de SE, no qual foi desenvolvida uma ferramenta para testar aplicações Android (iMPAcT Tool). Esta ferramenta executa estratégias de testes com o objetivo de verificar se as boas práticas da programação em Android estão a ser utilizadas ou não. Assim, o objetivo deste trabalho é analisar as falhas que originam os erros detetados pela iMPAcT Tool e definir um conjunto de operadores de mutação que possam ser aplicados a aplicações Android. E, por fim, verificar se os testes que estão a ser usados são ou não eficazes na deteção desses erros.

Os dez operadores de mutação definidos foram aplicados ao código de diferentes aplicações Android e depois as aplicações foram testadas usando a iMPAcT tool. Os resultados desses testes foram analisados para se perceber se a iMPAcT tool foi ou não capaz de detetar os erros inseridos. Embora a iMPAcT tool não tenha sido capaz de detetar os erros inseridos, o problam não está relacionado com os testes que são executados pela ferramenta mas pelos algoritmos de exploração de aplicações e de deteção de diferenças entre dois ecras que são implementados pela iMPAcT tool.

# Acknowledgements

I would like to thank my family and friends for all the love and support that they constantly give me. My friends who were always there to listen to me even though most of the time they had no idea of what I was talking. And my family, especially my mom, for being a role model and a constant source of strength.

I would also like to thank my supervisor, Ana Paiva, for her guidance and support during this work, that helped me to overcome the difficulties encountered during the work.

Liliana Filipa Lobo Ribeiro

*"Creativity is intelligence having fun"*

Albert Einstein

# Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| $A^2A^2$ | Android Automatic Testing Tool |
| ABS | Absolute Value Insertion |
| APD | Activity Permission Deletion |
| APK | Android Application Package |
| BWD | Button Widget Deletion |
| ECR | OnClick Event Replacement |
| EDS | Event Driven Software |
| ETR | OnTouch Event Replacement |
| GUI | Graphical User Interface |
| IPR | Intent Payload Replacement |
| ITR | Intent Target Replacement |
| MS | Mutation Score |
| MT | Mutation Testing |
| TWD | EditText Deletion |
| UI | User Interface |
| XML | Extensible Markup Language |

# Chapter 1

# Introduction

The smartphone has been taking control of our lifes for the past years, it is estimated that the number of smartphone users is about 2.32 billion and that number is expected to grow to 2.38 billion by 2020 [Sta17]. This growth has to be followed by an increase in the quality of the mobile applications but, unfortunately that is not been the case so far.

Nowadays there are almost three million applications available at the Google Store and about 12% of those application are classifed has having poor quality [App16]. In average, approximately eleven thousand low-quality applications are added each month to the Android store.

Google is aware of these quality problems and is trying to combat them with its new initiative, *Android Vitals*. *Android Vitals* displays to the developers several metrics about their application to help them improve the stability and performance of the application [Devc]. These metrics are more related with the bad behaviour of an application, like freezing the screen or when the application stops responding. With this initiative Google understands the current problem of the quality of the Android applications and it is trying to help the developers to create applications that are more responsive. However, *Android Vitals* does not cover every aspect of the quality of applications, only some performance related issues.

## 1.1 Context

With the increase popularity and usage of Android applications it is crucial that the quality of these applications is guaranteed. Mobile applications, namely Android applications, are becoming more complex and critical to every day life.

To ensure the quality of an application, the application has to go through a testing process, so that errors can be found and corrected. Testing the application will not prove that there are no errors, but will increase the confidence on the quality of said application.

It is in this context that the *iMPAcT tool* works, see section 2.3. By using reverse engineering and pattern identification techniques it automates the recognition and testing of UI patterns. The

set of tests is created automatically according to the patterns identified. Since the patterns were defined taking into consideration several Android programming guidelines, executing the tests will give an idea if those guidelines were followed or not. By having an automated and black-box tool to test Android applications the process of testing becomes simpler. There is no need to provide the source code of an application to test it, or to spend as much manual labour as it would be needed without the *iMPAcT tool*. However, sometimes black-box techniques cannot find all the errors and it can be necessary to use other techniques.

Using the *iMPAcT tool* as an example, if, after executing the tool against an Android application, no errors are detected does it mean that there are really no errors left in the code? Or just that the test set is not enough to uncover them? That's one of the most important questions when it comes to validating the testing process. A test passes if the application works as expected, but there is no guaranty that every possible scenario is being tested or that the test is relevant/crucial to find the errors.

This work will validate the results of the *iMPAcT tool* by determining if the test cases that are applied are enough to detect errors related to the defined patterns in Android applications or not.

## 1.2 Motivation and Goals

Considering that Android applications are present in our daily activities it is important that they work as expected, and that they have the maximum of quality that they can achieve. This is the biggest issue associated with Android applications nowadays, and quality is still low in most cases.

It is important to understand that this low quality does not necessarily mean that there were no testing process when developing the applications. Sometimes there are tests, but they are not enough to find the errors in the code.

One way to solve these problems is to assess the efficiency and quality of test sets. This will not only increase the confidence in the tests and the testing process but also increase the quality of the applications under test.

In this work, by using mutation testing to evaluate the set of tests provided by the *iMPAcT tool*, it will be possible to check if those tests are really enough to find the errors that they are expected to find. So, by assessing the quality of the tests and verifying their effectiveness, this work will lead to an increase in quality of Android applications.

The solution proposed is to define mutation operators that are specific to Android applications and which are tested by the *iMPAcT tool*. Then, from those mutation operators, mutants will be created and tested using the *iMPAcT tool* to determine whether or not the faults injected can be detected by the tool. As a result, it will be possible to determine if the *iMPAcT tool* is able to find the errors it was designed to find or not.

The project is divided into four phases:

1. Analyse typical Android programming errors;
2. Define mutation operators;
3. Evaluate the effectiveness of the test set generated by the *iMPAcT tool*;
4. Automation of the insertion of the mutants.

The first phase consists in analysing source code of Android applications, and identifying the errors associated with a bad implementation of the guidelines for the Android programming. In the second phase, using the errors that were identified earlier, the mutation operators will be defined. These mutation operators have to be specific to Android applications and take into consideration the scope that is covered by the test set from the *iMPAcT tool*. In the third phase, the mutation operators will be used to evaluate the test set and validate the results. The last phase, consists in the automation of the insertion of the mutants, however not all the mutants will be automated.

## 1.3 Structure of the document

The remaining of the document is structured as follow. Chapter 2 presents the current state of mobile testing, android programming and mutation testing. Chapter 3 talks about the selection of the applications, their errors and the definition of the mutants of each pattern and the theory behind their definition. In this chapter it is also presented a tool to automate the insertion of some of the mutation operators. Chapter 4 presents the results of the insertion of the mutants to each selected application. Chapter 5 presents the conclusions of this research and the future work.

# Chapter 2

# State of the Art

This chapter is divided into four main sections: in section 2.1 is presented the state of the art of the process of testing mobile applications, in section 2.2 some unique aspects related with Android programming are presented, in section 2.3 the *iMPAcT tool* is presented since it will be used in this project, and section 2.4 summarizes the current state of mutation testing in general and also summarizes the current state of mutation testing on Android applications and existing tools. The final section serves as a conclusion of this literature review.

## 2.1 Testing Mobile Applications

A mobile application is an application that runs on a mobile device (e.g. smartphone, tablet). Nowadays, a mobile application can run on Android, IOS, Windows Phone, among others.

These applications have some limitations when compared to traditional desktop applications, mainly because of the hardware that is used [ZSG16]. They have limited resources, less than available in current computers, they are context aware (the application is aware of the environment it is running, to adapt to it), most of these applications work based on user input, or on information from hardware sensors. The small screens, the use of touch screen instead of mouse and keyboard, connectivity (GPS, Wi-Fi, Bluetooth), possibility to rotate the screen, among others, all of this makes the process of testing mobile applications unique, and different from testing traditional software.

There are two ways of testing these applications, using emulators or using real devices. The use of emulators makes testing in different devices (simulated devices) easier and cheaper, as there is no need to invest in buying several devices, but can compromise the results of testing, especially in terms of performance [BV08]. Also, testing using emulators can be slower than using real

devices, depending on the computer that is emulating the mobile device.

When testing a mobile application one must test every aspect that is specific to this type of applications. According to [MDE12], when testing a mobile application one must consider:

**GUI testing** It is crucial to test the GUI in a mobile application because it is the only way the user has to interact with the application. If it has some error it will stop the user from working with the application as expected. There has been some research in trying to adapt pattern based GUI testing from web applications to mobile applications, as described in [CPN14]. The *iMPAcT tool*, described in 2.3. is an example of a GUI testing tool. Another example is the $A^2A^2$ (Android Automatic Testing Tool) [AFT11]. The $A^2A^2$ is a tool that uses a GUI crawling based technique for running crash testing and regression testing on Android applications. Crash testing has as objective detecting faults in applications caused by runtime exceptions. The objective of the tool is to find runtime crashes and visible faults on different versions of the same application.

**Performance and reliability testing** These two aspects are related to the resources and other internal characteristics of the devices. In [BV08] the author brings attention to the fact that the results of performance testing can be different when testing an application using an emulator or a real device.

**Security Testing** The assurance of security and privacy is critical for these applications, because the leak of private and sensitive information (private networks, passwords, location) can be devastating not only for the user but also for the developers of the applications, which will lose credibility. In [MEK+12] the authors define a framework to detect automatically security vulnerabilities in an application.

**Device multitude testing** Today there is a wide range of mobile devices, especially when talking about Android devices, so the applications have to adapt and work correctly on each of these devices. One approach to this type of tests in described in [ZGCU15], where the algorithm of k-means is used to cluster different types of devices and prioritize the ones in which the application should be tested on first.

**Memory and energy testing** Because of the limited resources, and low autonomy of some devices, it is crucial to perform these tests to prevent memory leaks. Some work was done to estimate the power consumption of an application by Thompson et al. in "Optimizing mobile application performance with model— driven engineering" as cited in [MDE12]. Palin et al. "Selection and execution of user-level test cases for energy cost evaluation of smartphones" as cited in [MDE12] present a methodology to select user test cases to perform the evaluation of the energy cost of an application.

### 2.1.1 Problems

The problems associated with mobile testing can derive from the unique features and functionalities of mobile devices and mobile applications, or from the testing tool that are used.

One of the biggest problems related with mobile testing is the fact that it is not possible to test every existent combination of software and hardware, which means that is not possible to test every scenario in which an application may be used. Each mobile device has different technical specifications such as the amount of available, network connectivity settings, and different hardware specifications such as CPU or cameras and sensors [Ram13]. Mobile applications rely on inputs from various sensors ( e.g. brightness, touch, orientation, motion) and connectivity ( e.g. bluetooth, Wi-Fi, 3G,). To test whether an application will behave as expected having in consideration all of these inputs and all the values they can take is not feasible. Since these mobile devices and consequently the mobile applications deal with personal and critical data that has to the stored and shared, security testing is not only important but also necessary [Cig]. The poor autonomy of the devices is also and important aspect to have in consideration when testing. So performance, memory and energy testing should also be performed during the testing process. The process of testing involves the test of each of the unique features of mobile applications, this will increase the effort and time spent in the process.

Another problem is that, since mobile applications are different from a traditional desktop application [ZSG16], it is not possible to apply testing techniques and methodologies that were developed for desktop applications in mobile applications without modifying them. These mobile applications need a new and effective approach that is capable of ensuring that the application is of high quality and reliable [ZSG16]. This can be achieved with the development of new tools and techniques or with the modification of tools and techniques from traditional software testing [MDE12].Since mobile applications are considered event-driven most tool available for event driven software (EDS) are still applicable to test mobile applicatinos. However, the tools will have to be adapted so they can perform cost effective tests in the Android environment [AFT11].

## 2.2 Android Programming

Android is an operating system based on the linux kernel that targets touch screen mobile devices [Dev13]. Android applications are mainly written using the Java programming language, although Java is not the only language that can be used. Recently, the language Kotlin is being used more in the developement of Android applications. This is manly because kotlin can work side by side with the Java code, so applications can migrate to Kotlin feauture by feature [Devf]. In this work, Kotlin code will not be used only Java code.

When developing and Android application there are several components that can compose that application [Dev13]:

### Activities

An activity represents an application's screen and its user interface. It is the entry point for the interaction of the application with the user [Dev13]. An activity can be in different states: *created*, *started*, *resumed*, *paused*, *stopped*, or *destroyed*. The state of an application is controlled by callback methods, as seen in Figure 2.1:

- onCreate() – is fired when the system first creates the activity. This callback must always be implemented, and it is here that the basic logic of the application should be initialized [Dev13]. The callback is associated with the state *created*, but the activity does not continue in this state for long, since as soon as the initialization is done the state is changed to the *started* state and the onStart() callback is fired.
- onStart() – is fired once the activity changes state to the *started* state. Once this callback is fired the activity will be visible to the user. This callback ends, normally, very quickly and once is finished the activity changes to the *resumed* state [Dev13].
- onResume() – is fired as soon as the activity changes to the *resumed* state. Here, the activity starts to interact with the user. The activity remains in this state until some event takes the focus out of the activity. For example, receiving a phone call, the user navigating to another activity or application.
- onPause() – this callback is fired at the first indication that the user is leaving the application and it should be used to pause certain operations that do not have to continue when the activity is in the background (in the *paused* state) [Dev13].
- onStop() – when the activity stops being visible to the user, its state changes to the *stopped* state and this callback is called. Here the resources that are not needed when the activity does not have focus should be released [Dev13].
- onDestroy() – fired right before the activity is destroyed, last callback to be fired. All the resources that have not been release yet have to be release here [Dev13].

When developing an Android application it is important to have a good understanding on how the life cycle of an activity works, because the bad implementation of these callbacks can lead to the loss of information when an user leaves an application and then returns to it, can cause the application to crash.

### Services

A service is executed in the background and it can perform long running operations or operations for remote processes. Unlike the activities, the services do not provide an UI [Dev13]. An example of a service can be the authentication of an user in an applications, it is done in the background and the user does not see the process, only the result afterwards.

### Broadcast Receivers

A broadcast receiver enables the communication between the system and the application, by allowing the application to respond to system-wide broadcast announcements. [Dev13] the

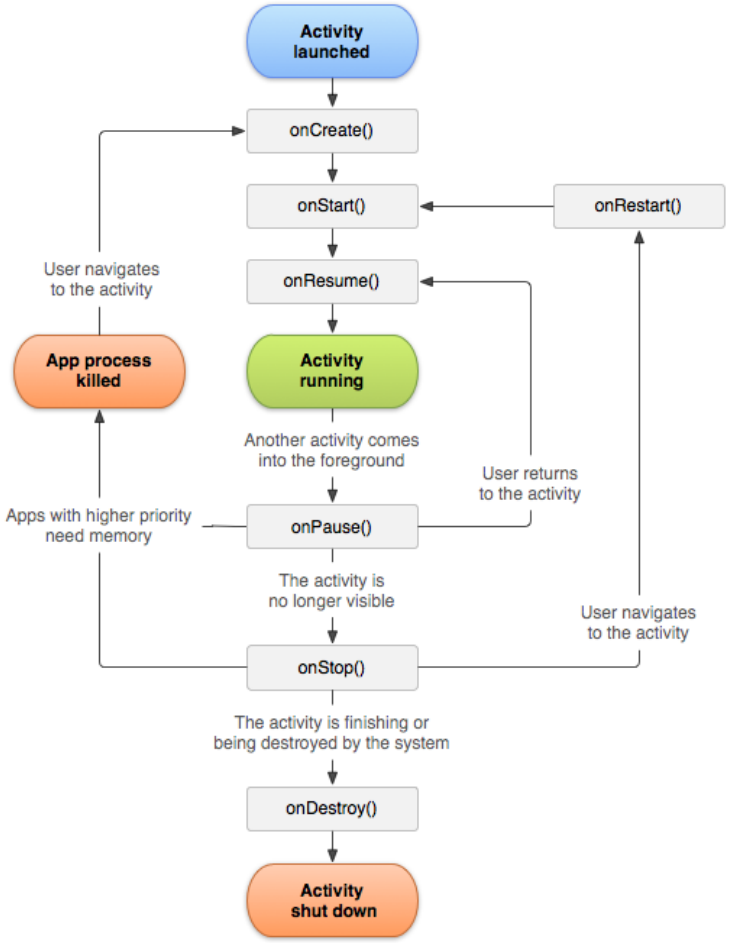


Figure 2.1: Activity lifecycle, [Dev13]

broadcasts can be initiated by the system, for example when the battery is running low or the screen is turned off, or initiated by an applications, for example to warn other applications that their data has been downloaded and is ready to be used.

**Content Providers**

A content provider is responsible for the management of application data that can be stored in a database, in the file system or on the web or any persistent storage location [Dev13]. By using a content provider the applications can read and modify data if they have the right permissions to do so.

One unique feature of the Android applications is that they can start components from another applications [Dev13]. For example, if an application needs to take a picture it does not have to implement that feature, because it can start the activity in the camera application (that is already present in the smartphone) and capture the photo that will then be returned to the first application [Dev13]. This is done with the help of intents. An intent allows the connection and communication between different Android applications. Intents can be explicit, where the application that should

perform the action is explicitly defined, or implicit, where only the action is specified and any application that can do that application can respond to that call.

## 2.3 *iMPAcT tool*

The *iMPAcT tool* uses reverse engineering and pattern identification techniques to automate the testing of UI patterns on Android applications, [MP15].

The Android application under test is automatically explored with the objective of finding the UI patterns, and to test them using the test strategies that are associated with each of the patterns.

Some patterns that are currently implemented in the tool, according to [MP15], are:

- **Side or Navigation drawer pattern** – The side drawer is a menu that displays the application's main menu on the left side of the screen. It is hidden most of the time, and can be uncovered by swiping the screen from the left edge of the screen to the middle, or by clicking the the icon of the menu. This pattern tests the height of the side drawer, that should occupy the full height of the screen [Coi17].
- **Orientation pattern** – The mobile devices can be in one of two possible orientations: landscape or portrait. When the mobile device rotates the layout of the application typically changes too, and it is important that no information is lost during that change. That is what this pattern tests [Coi17]:
  - No user input is lost during the rotation
  - No widget disappears during the rotation

  To do this the *iMPAcT tool* compares the screen before and after the rotation of the mobile device, in order to verify if the screens are identical. An error is detected when [Coi17]:
  - one screen is pop-up and the other is not
  - one screen has a side drawer and the does not
  - a widget is present in one screen but not in the other
  - the user inserted data is present in the first screen but not in the other
- **Resources dependency pattern** – since several applications are dependant of resources like GPS or Wifi, it is important to guarantee that an application continues responding and does not crash when those resources are not available [Coi17]. To do this the *iMPAcT tool* first determines if the resource is being used by the application or not, and if it is, that resource is turned off and the application is tested to check whether that causes and error or not.
- **Tab** – the presence of tabs makes it easier to navigate and application by switching between different views. The *iMPAcT tool* tests the tabs based on some guidelines for their correct

implementation. So, for the tab pattern to be correctly implemented the following most be true [Coi17] :

- there is only one set of tabs per screen
- the tabs should be on the upper part of the screen, for android applications
- by swiping the screen horizontally the selected tab should change and nothing else

The execution of the *iMPAcT tool* is divided into four phases, [MPF]:

1. **Exploration** — Exploration of the application under test and the visible screen to detect the possible events that can be fired. One of those events is chosen at random to be executed. This phase consists of four steps [Coi17]:
   - Explore of the current state of the application
   - Identify of the events that can be fired
   - Decide which event should be fired
   - Fire the selected event
2. **Pattern Matching (or reverse engineering)** — The *iMPAcT tool* analyses the current screen, after the event is fired, to determine whether a pattern is present or not.
3. **Tester** — After the pattern has been identified, the associated test strategy is applied. If the test fails then the pattern is not correctly implemented, if the test passes it means that the pattern is correctly implemented [Coi17]. If no pattern is detected in the previous phase, then this one is skipped.
4. **Artefacts** – At the end of the execution two artefacts are created, the report of the exploration and the model of the behaviour of the application [Coi17].

The tool has four modes of exploring the events [MP16]:

- the *execute once* – each event is only fired once, after an event is fired the first time it stops being considered as a valid event.
- the *priority to not executed* – the events that were not fired yet have priority over the ones that had already been fired. If all the possible event have already been fired, then the algorithm chooses the one that has the highest possibility of leading to a new screen that was never tested [Coi17].
- the *priority to not executed and list items* – this is similar to the *priority to not executed*, but also gives higher priority to the events that are associated with lists. Because this can help to explore all the contents of a screen before changing to another one [Coi17].

- the *all events* – every event will be tested.

The execution time changes depending on the exploration mode that is being used [MP16].

The UI patterns were defined having in consideration several Android programming guidelines, so the execution of the tests is able to check whether or not the guidelines were followed when developing the application under test or not.

The *iMPAcT tool* will be the object of study of this work, the test set will be evaluated using mutation testing techniques.

## 2.4 Mutation Testing

Mutation testing is a fault based testing technique that is used to assess the effectiveness of a test set by injecting faults into the original code and checking if the provided test can detect them or not [JH11].

Its basic principle is that the faults that are inserted are able to represent the mistakes that a programmer would make [JH11]. These faults are simple syntactic changes, like changing the operator < for the operator >, or the deletion of a statement. These are called mutation operators.

Mutation testing works on the basis of two theoretical principles, [Off89], [JH11]:

**The Competent Programmer:** assumes that the programmer will create code that is very close to the correct version. The incorrect version will only differ from the correct one as a consequence of small syntatic errors, [JH11], [Off92].

**The Coupling Effect:** says that test data that is able to detect small, simple faults will also be able to detect more complex ones, [JH11], [Off92].

The process of mutation testing starts with the original program and the set of tests, and it is divided into several steps:

- The tests are executed against the original code. If the tests fail then the original code needs to be corrected before continuing;
- The mutants are created according to the selected mutation operators;
- The set of tests is executed against the mutated code and the results are analysed;
- If the results are different from the original code, the mutants are killed and the tests are considered adequate;
- If the results cannot be distinguished between the original and the mutated code, then the mutants are still alive and the tests need to be rewritten and the process of mutation testing needs to begin again;

- In some cases there is no test case that is able to kill the mutant, these are called equivalent mutants and should be discarded.

Figure 2.2 shows the flow of a traditional process of mutation testing. The steps that are represented with solid boxes are automated and steps, and the steps represented with the dashed boxes are manual steps.

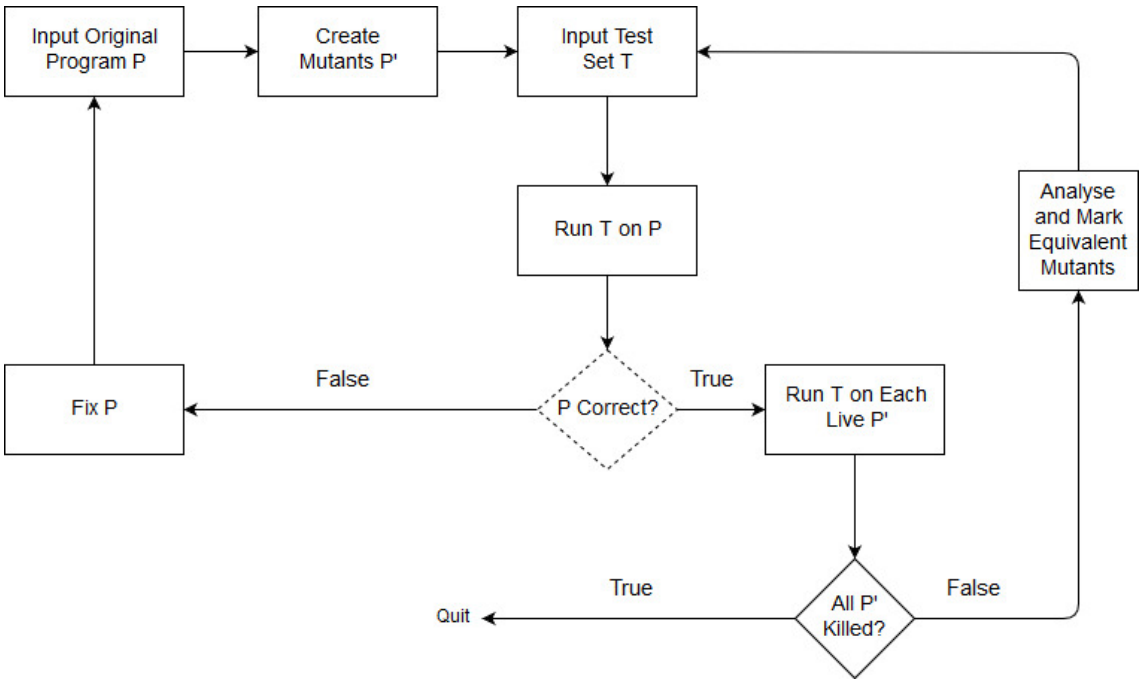


Figure 2.2: Traditional process of mutation testing, adpated from [JH11]

After this process, it is possible to calculate the Mutation Score, also known as the Mutation Adequacy Score:

$$MS = MutantsKilled/(TotalMutants - EquivalentMutants)$$

By analysing the mutation score it is possible to assess the effectiveness of a test set when it comes to detecting the faults that were introduced. The value will be a number between 0 and 1, 1 meaning that all mutants were killed, and 0 that all mutants are still alive. The closer the mutation score is to 1 the better the quality of the test set. If a test set gets a low mutation score it means it is not enough to detect the majority of errors of the code, and the set should be improved. In practice, it is nearly impossible to achieve a mutation score of 1, because that would be very expensive and time consuming.

### 2.4.1 Problems

One of the problems of mutation testing is the large number of mutants that will be created. This happens because a mutation operator can appear several times in the code and in different locations, for each one of those locations a mutant will be created [ND14]. With a big number of mutants, and because the test cases will be executed against all of them, mutation testing has a high execution cost, making the process sometimes very slow and computationally heavy. For this reason, it is not feasible to use mutation operators that cover all the code, as it would be very expensive.

In order to reduce the number of mutation operators without compromising its effectiviness in [JH09] the authors propose the use of higher order mutation operators. These higher order operators are created by combining simple faults that mask each other, making the higher order mutant harder to detect. This will reduce the set of mutation, and therefore the effort and time needed to perform mutation testing [JH09]. However, it raises a new challenge: how to select the higher order mutation operators. In [JH09] are search-based optimization is used to find which combinations are worth using, meaning which combinations will result is hard to kill mutants.

Other problem is that some of those mutants are in fact equivalent mutants and cannot be killed. If these cases are not detected early in the process, time and effort will be spent trying to modify the test set when there is no need for it. Over the years there has been some research in trying to find what makes a mutant equivalent. In [YHJ14] the authors found that certain mutation operators will always create more equivalent mutants than others. For example, it was detected that the ABS class of operators were responsible of the majority of equivalent mutants created during their experiment. In [OP94] the author presents a technique that uses mathematical constraints to detect equivalent mutants. In their work a tool name Equivalencer was developed in order to automate the task of identifying equivalent mutants in software, its detection rate is above 45% [OP94].

Nowadays, the detection of equivalent mutants is still a manual task ([JH11]), and is very slow, tiresome and error prone, despite all the research that is being performed in this topic. The tool developed in [OP94] is only a proof of concept and it is not used by practitioners.

Another problem is the fact that mutation testing is based on the injection of generated faults and not real faults. If the faults that are introduced are not very representative of the real ones then the findings of mutation testing will not be the correct ones. This problem is very important because it can render useless the process of mutation testing, this is why there have been some research in the topic of whether mutation testing can be used as a substitute for real faults. In [JJI+14] and [ABL05] the authors came to similar conclusions, that if the mutation operators are well selected, and if the equivalent mutants are identified and removed, then the faults injected by mutation testing can be used instead of real faults.

## 2.5 Mutation Testing on Android

The use of techniques of mutation testing in an Android environment is not very common, however there are some research already made in this field. In [DMAO15] and [DOAM17] the authors present some mutation operators that can be used in Android applications. In [Wei] the author presents a mutation testing tool for Android applications.

### 2.5.1 Mutation Operators

The mutation operators defined in [DMAO15] and in [DOAM17] are the same, and represent specific aspects of android programming. They are divided into four categories: Intent Mutation Operators, Event Handler Mutation Operators, Activity Life cycle Mutation Operators and XML Mutation Operators.

#### 2.5.1.1 Intent Mutation Operators

In these categories there are two types of mutation operators defined in [DOAM17]: *Intent Payload Replacement* (IPR) and *Intent Target Replacement* (ITR).

*Intent Payload Replacement* consists in changing the value of the payload (data that is sent in the intent) to a default value. This mutation operator will test whether the tester is verifying that the value passed by the payload is the correct one or not.

*Intent Target Replacement* consists in changing the target that will be executed (component that should be started). The target is replaced with all existing classes in the directory. This mutation operator forces the tester to test if the desired component is really started after the intent is executed.

#### 2.5.1.2 Event Handler Mutation Operators

Since Android is event based this type of mutation operators are very important. The authors defined two mutation operators in [DOAM17], the *OnClick Event Replacement* (ECR) and the *OnTouch Event Replacement* (ETR).

The *OnClick Event Replacement* mutation operator replaces each handler of the OnClick events with each other. The resulting mutant can only be killed if the tester test each component that has a OnClick event and checks if the execution is the one expected.

The *OnTouch Event Replacement* mutation operator works in a similar way to the ECR mutation operator, but replaces the OnTouch handlers instead. This mutation operator, like the ECR, forces the tester to test each component that has a OnTouch event and to check if the code executed is the expected one.

#### 2.5.1.3 Activity Life cycle Mutation Operators

There is only one mutation operator defined in [DOAM17] for this category, it is called the *Activity Lifecycle* Mutation operator.

The *Activity Lifecycle* Mutation Operator deletes the methods that override the transition between the states of the application lifecycle. By deleting those methods, the default methods will be executed instead of the ones created by the developers. This mutation operator forces the tester to test that the application is in the expected state after each execution.

#### 2.5.1.4 XML Mutation Operators

The operators defined in this category modify one of the various XML files that are used for configuring the Android application. The author defined in [DOAM17] three mutation operators, the *Button Widget Deletion* (BWD) , the *EditText Widget Deletion* (TWD), and the *Activity Permission Deletion* (APD).

The *Button Widget Deletion* mutation operator deletes the buttons in the XML file that defines the layout of the UI, one button is deleted at a time. This mutation operator forces the tester to write tests that will ensure that all the buttons are successfully displayed.

The *EditText Widget Deletion* mutation operator deletes the EditText widgets, that are widgets that allow the user enter values. Like the BWD, these widgets are removed from the XML file one at a time. The mutant that results from this mutation operator is only killed if the tester creates tests that use each one of these widgets.

The *Activity Permission Deletion* mutation operator is a little different from the other two. This operator deletes the permissions from the manifest file, one at a time. The only way to kill this mutant is if one of the tests uses a functionality that needs that kind of permission.

The authors present eight mutation operators that are specific for Android applications, these operators should be viewed as a small subset of all the mutation operators that can be used in

Android. The mutation operators presented only cover a small part of what can and should be tested in Android applications.

### 2.5.2 Problems of mutation testing on Android

One of the biggest problems is the fact that running tests on Android is very slow [DOAM17], this problem is common to all testing in Android not only on mutation testing. However, this problem is aggravated when talking about mutation testing, because mutation testing involves the creation of several modified copies of the original code. So each test will be run against not only the original code, but also against all the mutants.

### 2.5.3 Existing Tools

In [Wei] the author presents *muDroid*, a mutation testing tool for Android applications. This tool uses six mutation operators, that are not specific to Android applications but are rather general mutation operators.

*muDroid* has three main phases, [Wei], the phases are also visible in Figure 2.3 :

- **Mutant Generator** — Main goal is to generate the APK mutants from the APK file under test. Each one of the APK mutants will contain a unique fault. The steps of this phase are:

  1. APK is decompressed using the apktool, giving origin to Smali files
  2. Each Smali file is scanned in order to identify the patterns of each mutant. After a pattern is detected the mutant is generated using the pre defined rules
  3. Number of mutants is reduced using mutant selection strategies

  These APK mutants will be sent to the Interaction Simulator.

- **Interaction Simulator** — This phase has as input the APK mutants and as outputs screenshots of the application. The user behaviour is simulated and the screenshots are taken at pre-defined intervals.

- **Result Analyser** — Is in this phase that the results of the tests show if the mutants were killed by the tests or not. This phase is divided into two steps:

  – Screenshots are analysed to check if the mutants are killed or not
  – An HTML formatted report is generated with details of each mutant, in order to facilitate the visualization of the results

The fact that this tool does not need to have the source code of an application to insert the mutants is an advantage, but can also bring some disadvantages. For example, the manipulation

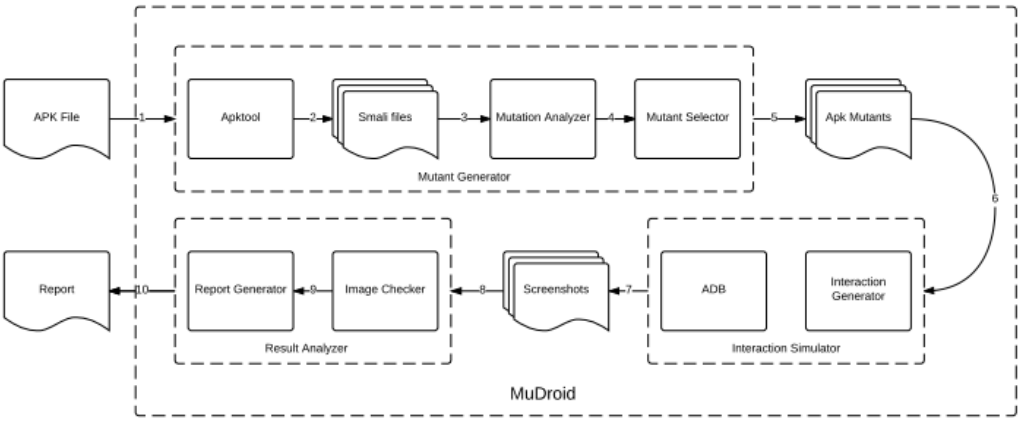


Figure 2.3: *muDroid* System Architecture, [Wei]

of Smali code is not as efficient as the manipulation of the Android bytecode directly. It also can increase the testing time, because the APK needs to be decompiled before the alterations are made, and after it has to be compiled again.

The source code of this tool is available in a GitHub repository [1], but there is almost no information about how to use the tool (apart from simple commands) or about what parameters have to be passed to the tool or how those will influence the results.

## 2.6 Conclusions

Nowadays, the topic of mutation testing on Android applications is still not as well developed and researched as mutation testing on traditional software.

Even though *muDroid* is a tool that uses mutation testing techniques on Android applications, the mutation operators that are used are not specific to Android programming so it does not test the applications having in consideration their nature and unique aspects. As far as we know, until this moment there is no tool that performs mutation testing on Android applications and that uses mutation operators that are specific to Android programming. As far as we know, the only mutation operators defined, at the moment, for Android applications are the ones presented in [DOAM17], and these only cover four of the specific aspects of Android programming (intent payload, event handlers, life-cycle of an application and XML files). These operators will not be used in this project because the operators are outside of the scope of the *iMPAcT tool*.

So, at this moment, there are no specific mutants that are able to verify if the testing tools are capable of detecting whether the good practices of Android programming were followed during the development or not. This is were this research work will make its contribution, by providing a set of mutants that will be able to assess the effectiveness of the *iMPAcT tool*.

[1] https://github.com/Yuan-W/muDroid

# Chapter 3

# iMPAcT Tool – Mutation Testing

The objective of this chapter is to define the mutation operators that will be used in the case study. To do this, first the set of applications that will be studied and used was selected and tested using the *iMPAcT tool*. Then, and having as basis the errors that were detected in each application and the guidelines of Android programming, the mutation operators are defined. Later, a tool in Java is developed to automate the insertion of some of the mutation operators into the Android applications.

## 3.1 Selection of Android Applications

The selection of the applications was done by browsing the Android projects available at github[1] and f-droid[2].

On github the query "topic:android" was used and the results were sorted by most stars. From this query, 9566 applications were found but only 136 applications were selected. The other applications did not met the criteria presented in Table 3.1. It was evident that this query did not return all the Android projects on github, but it would have been impossible to explore all the projects that are available on github in order to find all the applications present.
The f-droid repository is smaller than the github repository, so all the applications available on the website were considered, all 2130 applications. From this search only 193 applications followed the criteria defined in Table 3.1.

[1] http://github.com/
[2] http://f-droid.org/

Table 3.1: Criteria for the selection of applications

| Criteria | |
|---|---|
| Be available on the Google Store | application has to be available in the Google store |
| Be available in Portugal | the application has to be available in Portugal to access its information |
| Have the source code available | access to the source code is needed to analyse the code and to insert the mutants |
| Have a Google store rating >= 3.5 | to analyse only the applications that are have some degree of quality according to its users |
| Have a number of ratings >= 100 | to insure that the rating of the application is a representation of the experience of several users and not only a small group |
| Use gradle | to simplify the build of the application |
| Be Android Native | because the aim of this work is to define mutation operators that are specific to Android |
| Have a GUI | the *iMPAcT tool* tests the GUI so it is important that the applications have one and are more than just a launcher or a keyboard |
| Be in an Western European Language | to facilitate the understanding of the UI of the application |

### 3.1.1 Compiling the Applications

In this phase several applications were removed from the list because it was not possible to build the APK, since the application was only a widget or an extension to another application or because to fully use the applications credentials to specific entities were needed (Mensa Card, credentials to the JUET[3] or JIIT-128[4] )

From the 329 applications that resulted from the selection phase, only 167 applications were successfully built and worked on the smartphone used for the tests. Among the building/compiling errors the more common were:

[3]Jaypee University of Engineering and Technology
[4]Jaypee Institute of Information Technology

- Generic build failed error message, missing configurations or build files.
- Missing google-services.json file.
- Gradle sync failed (missing files).

During this phase the source code of some applications had to be altered in order to compile it ( update the version of some dependencies that were used, remove some building options that needed to connect the application with external sources associated with building the release version of the application ). The IDE used was Android Studio 2.3.

All of the top categories of applications on the Google store, according to [App] are represented in the set of 167 applications, except the business category where no application was found.

Table 3.2: Category distribution

| **Category** | **Number of applications** |
|---|---|
| Board | 1 |
| Books and Reference | 7 |
| Comics | 2 |
| Comunication | 12 |
| Education | 5 |
| Entertainment | 2 |
| Finance | 2 |
| Game Strategy | 1 |
| Health and Fitness | 1 |
| Libraries and Demo | 8 |
| Lifestyle | 2 |
| Maps and Navigation | 4 |
| Medical | 1 |
| Music and Audio | 6 |
| News and Magazines | 7 |
| Personalization | 3 |
| Photography | 2 |
| Productivity | 21 |
| Puzzle | 5 |
| Social | 6 |
| Tools | 58 |
| Travel and Local | 7 |
| Video Players | 4 |
| Total | 167 |

## 3.2 Tests

All of the 167 applications were tested using the *iMPAcT tool*, each one was tested two times. If the results of the tests differed or an error that was visible in the application was not detected by the *iMPAcT tool*, then a third test was performed. Because of time constraints, all the patterns were tested at the same time (orientation, tabs, side drawer and resource dependency). Testing each pattern separately would increase the number of tests without decreasing significantly the time spent in each test.

### 3.2.1 Orientation

This pattern envolves two test strategies [Coi17]:

- Detect of missing widget after rotation:

```
Goal: "UI main components are still present"
P: {"UIP is present and TP not applied to current activity"}
V: {}
A: [observation, rotate screen, observation, scroll screen, observation]
C: {"widgets still present"}
```

- Detect of missing user inserted data after rotation:

```
Goal: "Data unchanged when screen rotates"
P: {"UIP is present and user data was entered and TP not applied to this
    element"}
V: {}
A: [observation, rotate screen, observation, scroll screen, observation]
C: {"user entered data was not lost"}
```

If an application fails one of the two test strategies it will be considered to have the orientation pattern incorrectly implemented.

According to the results of the *iMPAcT tool*, from the 167 applications tested: 61 had the orientation pattern correctly implemented; 103 had the pattern not correctly implemented; and in 3 applications the pattern was not detected. This is represented in Figure 3.1.

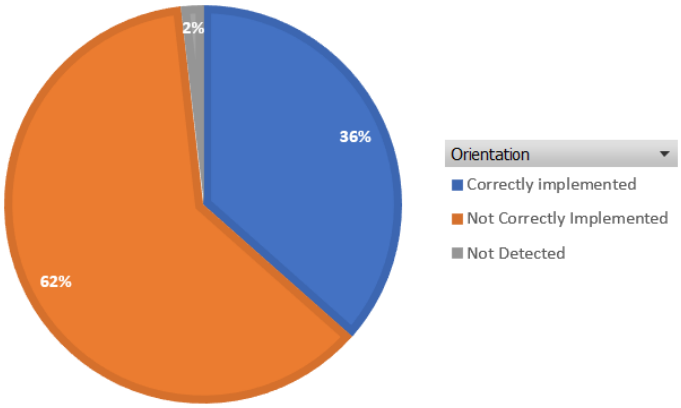


Figure 3.1: Comparison of the applications with the Orientation pattern correctly and not correctly implemented

Since one of the applications (*Timber*) had a visible error in the UI that the *iMPAcT tool* was not able to identify, the source code of the *iMPAcT tool* has slightly changed, instead of rotating the screen once and verifying if any components disappeared now it rotates the screen two times so that the screen is in the beginning position when the verification of the components occurs.

At that moment 11 applications had already been tested, so they were retested this time only for the orientation pattern using the modified *iMPAcT tool*. This resulted in 3 more applications being recognized has having the orientation pattern incorrectly implemented.

From the 61 applications that have the orientation pattern correctly implemented, 21 applications do not allow rotation. This means that the application only works in portrait mode or landscape mode. These applications are not good examples of the correct implementation of the orientation so they will not be used in the study. So, there are only 40 applications left that have the orientation pattern correctly implemented. The Table 3.3 shows some of the applications that have the pattern correctly implemented in two sets: the ones that allow the rotation of the screen and the ones that do not allow the rotation of the screen. The complete set of applications that have the pattern correctly implemented and that allow rotation can be found in Table A.2.

Table 3.3: Orientation: Applications with pattern correctly implemented that allow rotation

| **Allows rotation** | **Does not allow rotation** |
|---|---|
| ObservableScrollView demo | Lottie |
| wallsplash \| wallpaper app | williamchart |
| MaterialViewPager | MusicDNA |
| Advanced RecyclerView | My Diary(unofficial) |
| ForkHub for GitHub | Habitica: Gamify Your Tasks |
| Simple Gallery | 2048 (Ads Free) |
| Bewegungsmelder | Gobandroid Go Material |
| Hubble Gallery | Critical Maps |
| Chroma Doze | MHGen Database |
| Calendar Notifications | Simple Flashlight |
| Hacker News | ARChon Packager |
| Anuto TD | 2048 |
| Blokish | Simple Camera |
| Simple File Manager | Hex |
| iBeacon Detetor | Mongo Explorer |
| Lucid Browser | Movian Remote |
| BatteryFu | Note Crypt |
| Kaleidoscope | Olam |
| Multitouch Test | RGB Tool |
| Notes | TV Kill |
| Open Link With | Wifi Walkie Talkie |
| ... | ... |

Figure 3.2 displays a comparison between the number of applications with the orientation pattern correctly implemented and the applications with the orientation pattern not correctly implemented grouped by their Google Store categories. In most categories the number of applications that have the pattern incorrectly implemented is superior to the number of applications that have the pattern correctly implemented. With the exception of some categories in which the inverse is visible or the number is the same.

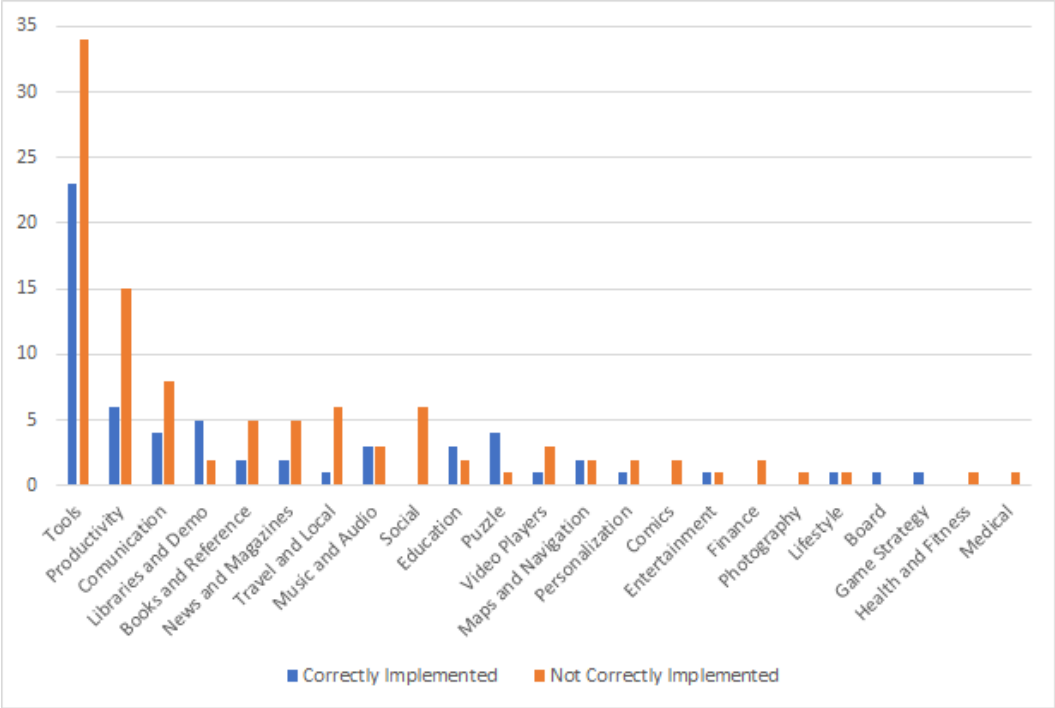


Figure 3.2: Orientation Pattern: Comparison between the categories of the applications and the *iMPAcT tool* results

### 3.2.2 Tab

The tab pattern consists of three test strategies [Coi17]:

- Verify if there is only one set of tabs present in the current screen:

```
Goal: "Only one set of patterns"
P: {"UIP && TP not applied to current activity"}
V: {}
A: [observation]
C: {"there is only one set of tabs at the same time"}
```

- Verify if the tab is in upper part of the screen:

```
Goal: "Tabs position"
P: {"UIP && TP not applied to current activity"}
V: {}
A: [observation]
C: {"Tabs are on the upper part of the screen"}
```

- Verify if swiping the screen will change the selected tab:

```
Goal: "Horizontally scrolling the screen should change the selected tab"
P: {"UIP && TP not applied to current activity"}
V: {}
A: [observation, swipe screen horizontally, observation]
C: {"the selected tab changed"}
```

If an application fails one of the three test strategies it will be considered to have the tab pattern incorrectly implemented.

According to the results of the *iMPAcT tool*, from the 167 applications that were tested: 11 applications had the tab pattern correctly implemented; 15 had the pattern not correctly implemented; and in 141 the pattern was not detected. These results are visible in Figure 3.3.

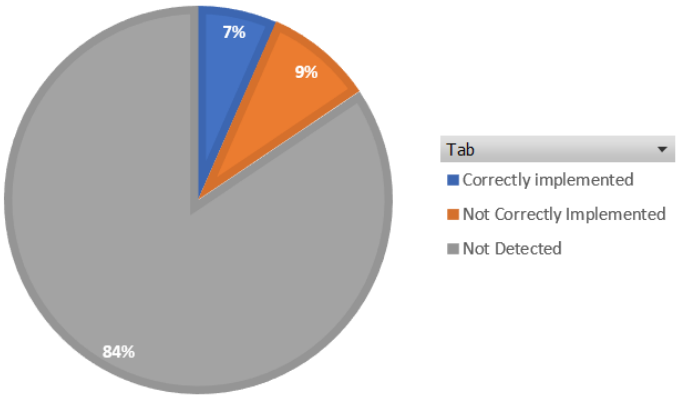


Figure 3.3: Comparison of the applications with the Tab pattern correctly and not correctly implemented

One of the applications was classified as having the pattern correctly implemented but, in reality, it does not have the pattern implemented. The errors detected by the *iMPAcT tool* in this set of applications were all related with the test strategy of verifying if the selected tab changes when swiping the screen.

Table 3.4: Tab Pattern: classification of applications

| Correctly Implemented | Not Correctly Implemented |
|---|---|
| Timber | MaterialViewPager |
| Conversations (Jabber / XMPP) | Materialistic - Hacker News |
| WordPress | Slide for Reddit |
| Antox | Habitica: Gamify Your Tasks |
| FOSDEM Companion | GDG - News and Events |
| MHGen Database | Bodyweight Fitness Pro |
| OctoDroid | AntennaPod |
| OI File Manager | One Bus Away |
| Open Tasks | AntennaPode |
| Poet Assistant | Numix Calculator |
| Forkhub | pMetro |
| | Privacy Week Schedule |
| | Quick Dice Roller |
| | Timer Droid |
| | TV Kill |
| | Vlille Checker |

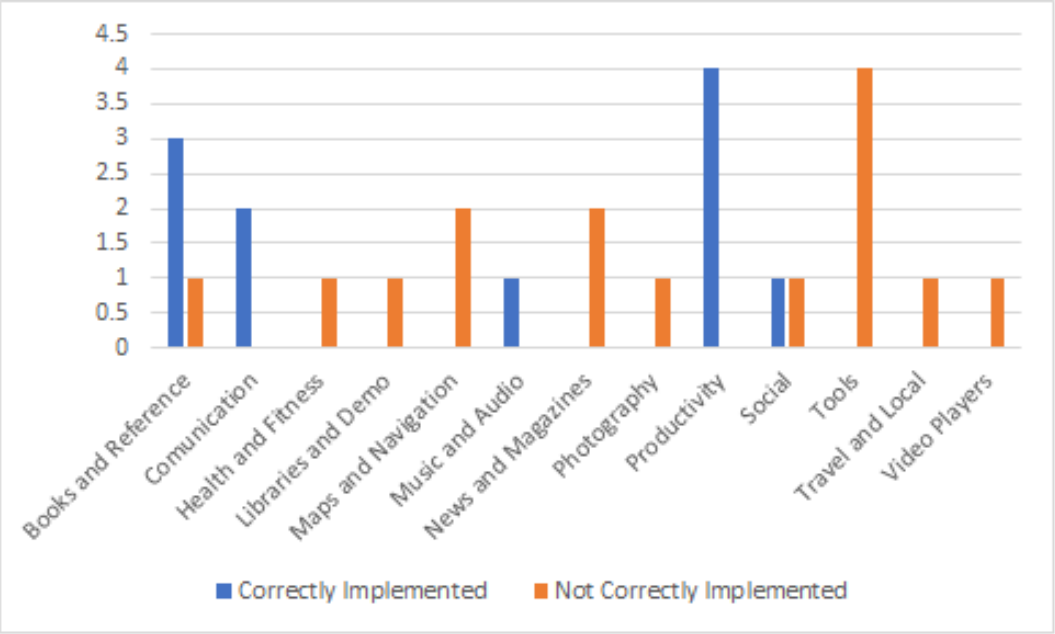


Figure 3.4: Tab Pattern: Comparison between the categories of the applications and the *iMPAcT tool* results

Figure 3.4 displays a comparison between the number of applications with the tab pattern correctly implemented and the applications with the tab pattern not correctly implemented grouped

by their Google Store categories. In this pattern it is visible that most applications that have the pattern correctly implemented are concentrated in only five categories, and the applications that have the pattern incorrectly implemented are more scattered.

### 3.2.3 Side Drawer

The side drawer pattern consist of only one test strategy [Coi17]:

- Verify if the side drawer occupies the full height of the screen:

```
Goal: "Side Drawer occupies full height"
P: {"UIP present and side drawer available and TP not applied to current
    activity"}
V: {}
A: [open side drawer, observation]
C: {"covers the screen in full height"}
```

According to the *iMPAcT tool* results, from the 167 applications that were tested: 22 applications had the side drawer pattern correclty implemented; 19 had the pattern not correctly implemented; and in 126 applications the pattern was not detected. These results are represented in Figure 3.9.

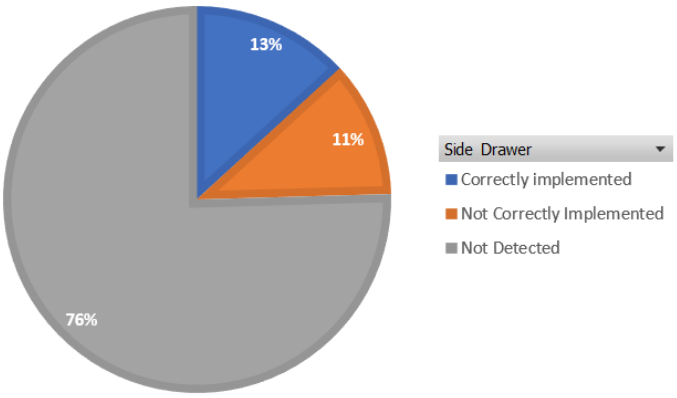


Figure 3.5: Comparison of the applications with the Side Drawer pattern correctly and not correctly implemented

The Table 3.5 shows all the applications in which the pattern was detected as correctly implemented and as not correctly implemented.

Table 3.5: Side Drawer Pattern: classification of applications

| **Correctly Implemented** | **Not Correctly Implemented** |
|---|---|
| Timber | ForkHub for GitHub |
| MusicDNA | Materialistic - Hacker News |
| Yaaic - IRC Client | Slide for Reddit |
| GDG - News and Events | RedReader |
| Polar Dashboard Sample | Antox |
| Etar - OpenSource Calendar | Orgzly: Notes and To-Do Lists |
| wallabag | Gobandroid Go Material |
| Twittnuker for Twitter | Critical Maps |
| Equate | Hubble Gallery |
| FOSDEM Companion | Bubble |
| AnkiDroid | iFixit |
| Lightning | Mongo Explorer |
| File Manager | My Expenses |
| OctoDroid | PassAndroid |
| QR Scanner | pOT-Droid |
| Quick Lyrics | RF Analyzer |
| Riot | S Tools + |
| SMSsync | SealNote |
| Toffed | Shader Editor |
| Unit Converter Ultimate | |
| Web Opac | |
| WiFiAnalyzer | |

Figure 3.6 displays a comparison between the number of applications with the side drawer pattern correctly implemented and the applications with the side drawer pattern not correctly implemented grouped by their Google Store categories. In this pattern there is not a big difference between the number of applications that have the pattern correctly implemented and applications that have the pattern incorrectly implemented. Most categories only have applications that have the pattern incorrectly implemented, or have almost the same number of applications with the pattern correctly and incorrectly implemented.

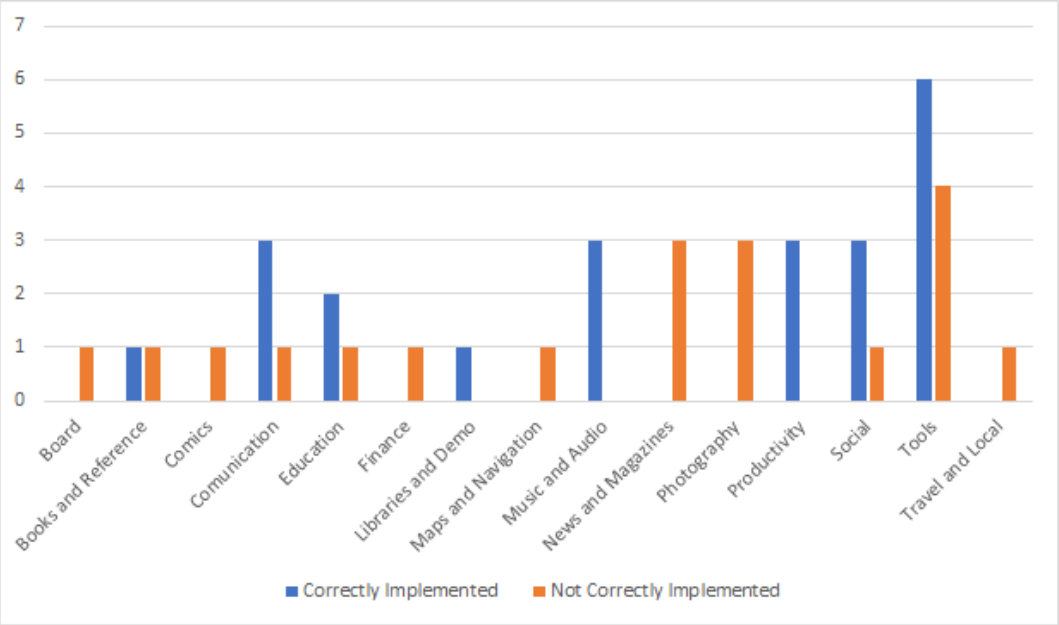


Figure 3.6: Side Drawer Pattern: Comparison between the categories of the applications and the *iMPAcT tool* results

### 3.2.4 Resource Dependency

The resource dependency pattern consists of only one test strategy [Coi17]:

- Verify if the application crashes when the resource is not available:

```
Goal: "Application does not crash when resource is made unavailable"
P: {"UIP && TP not applied to current activity"}
V: {"resource", resource_name}
A: [observation, turn resource off, observation]
C: {"application did not crash"}
```

At the moment, the only resource that is tested by the *iMPAcT tool* is the WiFi, and if an application fails the test strategy it means that it does not have this pattern correctly implemented.

There was no application that had this pattern implemented incorrectly. This is, mostly likely, related to the fact that when programming an Android application using Android Studio, or Eclipse, it is mandatory by the IDE to surround most calls to function that throw exceptions with a try and catch. So even if the developer is not conscientiously creating an application that can deal with the lack of the WiFi resource this is ensured by the IDE. The only exception to this are the runtime exceptions, that can be thrown by functions and not enforce that the calling of those functions has to be surrounded by try and catch clauses.

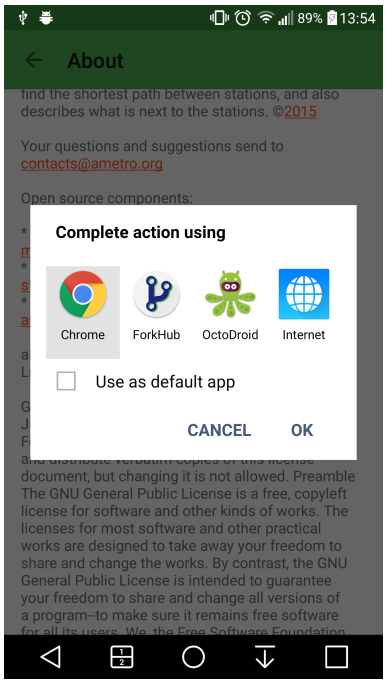


Figure 3.7: Default application pre selected

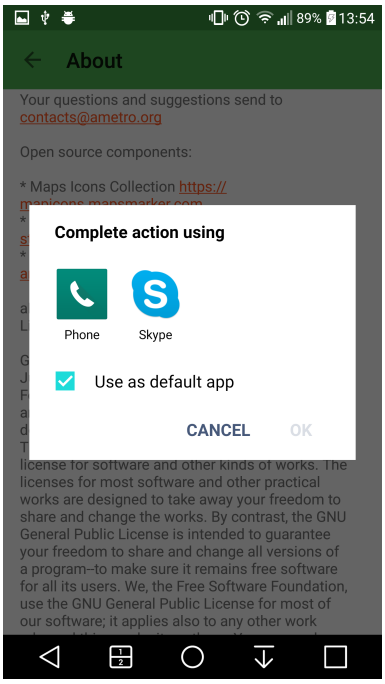


Figure 3.8: No default application pre selected

At first the *iMPAcT tool* detected 18 applications that did not have the pattern correctly implemented, but after analysing the source code of each application no error was detected. More tests had to be executed in order to understand why the detected errors in the pattern when there were no errors present. The problem was associated with the popup to select an application to perform a certain action. If an application was pre selected as default them the "OK" button will be clickable as is visible in 3.7, otherwise only the "Cancel" button will be clickable as shown in 3.8, what induced the *iMPAcT tool* in error. After selecting default applications to perform specific actions that were used by the 18 applications, the tests were executed again and this time no error was detected.

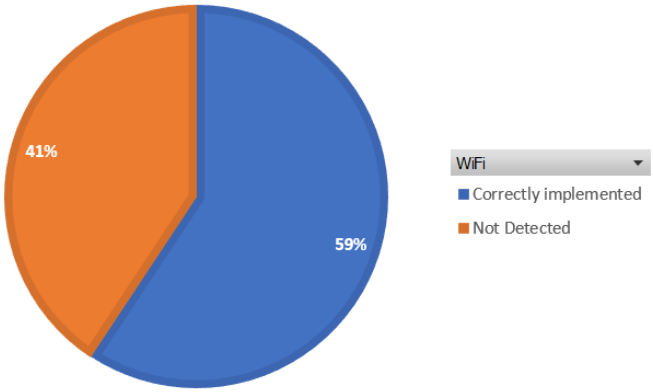


Figure 3.9: Comparison of the applications with the Resource Dependency pattern correctly and not correctly implemented

According to the *iMPAcT tool* results, from the 167 applications that were tested: 99 had the pattern correctly implemented and in 68 applications the pattern was not detected. These results are visible in Figure 3.9.

Figure 3.10 displays the distribution of applications that have the pattern correctly implemented by the Google Store categories, since there was no application with the pattern incorrectly implemented.

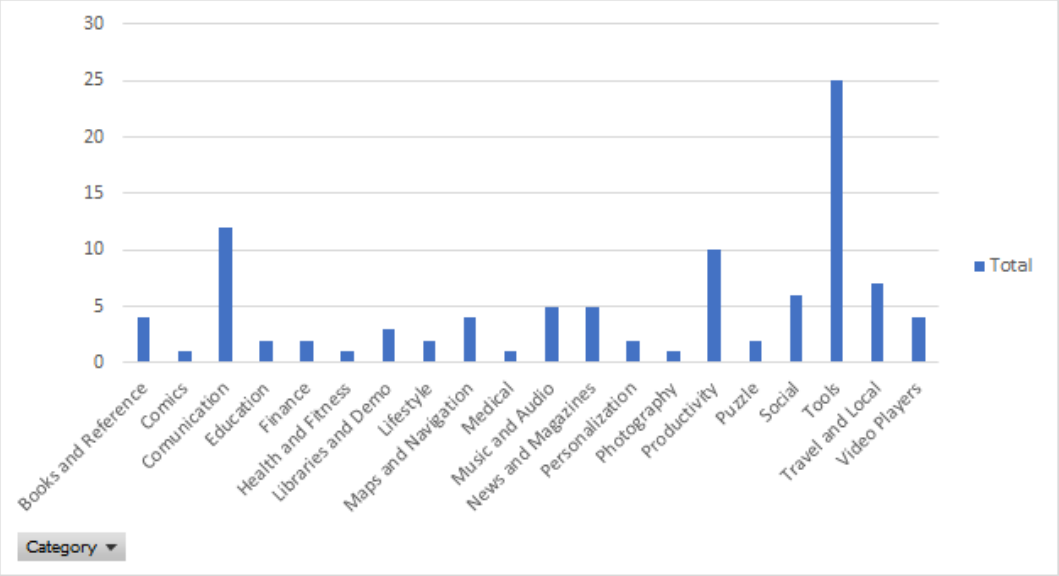


Figure 3.10: Resource Dependency Pattern: Comparison between the categories of the applications and the *iMPAcT tool* results

## 3.3 Mutant definition

In this section the mutation operators are defined, and the reasoning behind their definition is explained. The mutation operators are defined for each pattern, except for the resource dependency pattern because no application that had this pattern not correctly implemented was found. For the orientation pattern, 4 mutation operators were defined, for the tab pattern only one mutation operator was defined and for the side drawer pattern 5 mutation operators were defined.

### 3.3.1 Orientation

Since this pattern is not connected with one specific widget, like the tab and the side drawer patterns, it was difficult to find a common ground for all the errors detected. The only common aspect between all the errors was the fact that the presence of the error implies that the developers did not take into consideration the lifecycle of an Android application when they were developing their application.

When the rotation of the screen is detected the activity is restarted and the following methods are called, in order:

- onDestroy() – last method called before the activity is destroyed
- onCreate() – called when the activity is starting (or restarting)

By restarting, the application is trying to adapt to the new configurations automatically. It reloads the alternative resources so that the ones that are more compatible with those configurations start being used instead. There can be different layouts for the activities depending on the mode of the screen (landscape or portrait mode). Those layouts are saved inside folders that define their target configuration, for example every layout that is inside the "layout" folder will be used when the phone is in portrait mode, and every layout that is inside the folder "layout-land" will be used when the phone changes to landscape mode. Unless the programmer specifies otherwise, the change of the layout used will be done automatically as soon as a change in orientation in noticed.

When the activity restarts, it loses all the information that is not saved in a persistent memory. To prevent the loss of information, that information has to be saved before the activity restarts (before the onDestroy() ) and restored once the activity restarts (after or during the onCreate() ). Two methods exist to help with the task of saving and restoring the information:

- onSaveInstanceState() – called by the system before destroying the activity [Devb]. In this example, adapted from [Devb], the state of two variables is being saved using this method.

```
static final String STATE_POINTS = "2";
static final String STATE_PLAYER_NAME = "smith";
...

@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    // To save the user's current game state so it can be restored later
    savedInstanceState.putInt(STATE_POINTS, mCurrentPoints);
    savedInstanceState.putInt(STATE_PLAYER_NAME, mCurrentPlayerName);

    // Superclass should always be called so it can save the view hierarchy
        state
    super.onSaveInstanceState(savedInstanceState);
}
```

- onRestoreInstanceState() – called when the activity is being re-initialized from a previously saved state [Deva]. In this snippet, the state of the same two variables is being restored.

```
public void onRestoreInstanceState(Bundle savedInstanceState) {
```

```
    // Superclass should be always called to restore the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);

    // Restore sthe variables from saved instance
    mCurrentPoints = savedInstanceState.getInt(STATE_POINTS);
    mCurrentPlayerName = savedInstanceState.getInt(STATE_PLAYER_NAME);
}
```

- onCreate() – instead of restoring the state using the onRestoreInstanceState() method, this method can be used. Since this method is also called when the activity is created for the first time, it is necessary to verify if there is a previously saved state before trying to restore it.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  // Always call the superclass first
    super.onCreate(savedInstanceState);

    // Restore a previous saved state
    if (savedInstanceState != null) {
        // Restore value of members from saved state
        mCurrentPoints = savedInstanceState.getInt(STATE_POINTS);
        mCurrentPlayerName = savedInstanceState.getInt(STATE_PLAYER_NAME);
    } else {
        // activity is starting for the first time
    }
    ...
}
```

In the examples, the state is saved using a *Bundle*, but it is also possible to save the state using the *SharedPreferences* class, or a file saved in the internal or external memory. The process is the same regardless of the method chosen to save the state.

If an application deals with a lot of information saving the state of that huge amount of data will be costly and can jeopardize the performance of the application. In this cases there are three solutions:

**Retain an object during a configuration change**

To do this it is necessary to extend the *Fragment* class, and call the setRetainInstance(Boolean)

```
public class RetainedFragment extends Fragment {
    // object with the data the has to be saved
    private HugeDataClass data;

    // this method is only called once for this fragment
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // retain the fragment
        setRetainInstance(true);
    }

    public void setData(HugeDataClass data) {
        this.data = data;
    }

    public HugeDataClass getData() {
        return data;
    }
}
```

This way the data will not be lost when the activity restarts, regardless of the reason.

**Handle the configuration yourself**

Specify in the *AndroidManifest.xml* that when the orientation changes it will be handled by a function and not with the normal behaviour

```
android:configurationChanged="orientation|screenSize"
```

and then specify what to do when the orientation changes in the onConfigurationChanged( Configuration newConfig). In this case it is the programmers responsability to change the resources that will be used having in cosideration the new configurations. Code adapted from [And]:

```
  @Override
  public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    // deal with the orientation change
    // for example, changing the layout that is being used
  }
```

**Stop the application from rotating the screen**

- To stop an activity from rotating, the programmer has to specify in the *AndroidManifest.xml*, for each activity that should not rotate:

```
android:screenOrientation="portrait"
```

or

```
android:screenOrientation="landscape"
```

- Or add the following to the onCreate() function of the corresponding activity:

```
setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_PORTRAIT);
```

- Or specify in the *AndroidManifest.xml* file, and inside the target activity:

```
android:configurationChanged="orientation|screenSize"
```

And add the following method to the file that defined the activity:

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
  super.onConfigurationChanged(newConfig);

  // Checks the orientation of the screen
  if (newConfig.orientation == Configuration.ORIENTATION\_LANDSCAPE) {
     //define what to do when the phone is rotated to landscape
     //to
  } else
   if (newConfig.orientation == Configuration.ORIENTATION\_PORTRAIT){

   }
}
```

There are some widgets that save their own state by default, like the *EditText*, or the *Spinner*. When this widgets are used, the developers do not have to explicitly save their state. However it is possible to remove this behaviour by calling the method "setSaveEnabled(boolean)" with the parameter "false".

Most applications use a combination of all of these solutions, so it is difficult to define a general rule that is easily applied to all. The mutants were defined to simulate the most common errors in the applications.

1. If the activity saves the state of the variables using a *Bundle*:

- In the file that defines the activity, before restoring the state of the variables set the *Bundle* to null

```
savedInstanceState = null;
if (savedInstanceState != null) {
  ...
}
```

- If the activity that uses the *EditText* has removed the default behaviour of the application when a configuration changes

```
android:configurationChanged="orientation"
```

then remove from the *AndroidManifest.xml* file, the "orientation" from the attribute, or remove the whole line

2. If the application uses a *EditText* or a subclass:

- find the initialization *EditText* and add:

```
private EditText descriptionText;
...
descriptionText = finder.find(R.id.et_gist_description);
descriptionText.setSaveEnabled(false);
```

- If the activity that uses the *EditText* has removed the default behaviour of the application when a configuration changes :

```
android:configurationChanged="orientation"
```

then remove from the *AndroidManifest.xml* file, the "orientation" from the attribute, or remove the whole line

3. If the application uses a *Spinner* or subclass:

- find the initialization *Spinner* and add to that file:

```
Spinner s1 = (Spinner) findViewById(bitlength);
s1.setSaveEnabled(false);
```

- If the activity that uses the *Spinner* has removed the default behaviour of the application when a configuration changes:

```
android:configurationChanged="orientation"
```

then remove from the *AndroidManifest.xml* file, the "orientation" from the attribute, or remove the whole line

4. If the application does not save the state of the variables, or uses a *EditText* or subclass, but has a popup menu or handles user input, then the "orientation" has to be removed from this attribute (*AndroidManifest.xml* file), or the whole line has to be removed:

```
android:configurationChanged="orientation"
```

The process behind the definition of the mutant 1 can be easily adapted to work with *SharedPreferences* or a file instead of the *Bundle*. Unfortunately, none of the applications that had the pattern correctly implemented used any of those methods, so the mutant would not be tested even if defined.

The mutant 2 is used in order to simulate the usage of other widgets that do not have the default behaviour of saving their own state. In every application analysed that used the *EditText*, it was used correctly. Since each programmer develops their application in their own way, and because the pattern of orientation is related with more than just one or two widgets, it is simpler to simulate the bad implementation of a widget, than to define errors that are too specific to one application and that can be reused in another one.

The error associated with the mutant 4 was the most common amongst the applications tested. However, the code:

```
android:configurationChanged="orientation"
```

as the only way to prevent orientation errors is not the perfect solution, and should only be used as temporary solution.

#### 3.3.1.1 Visible Effects

The insertion of the mutant 1 will force the data that is inserted by the user to disappear when the orientation of the phone changes. In Figure 3.11 shows the application after the user inserts the word "Question" into one of the *TextView* widget. Figure 3.12 shows at happens to the application after the orientation is changed, the data inserted disappears.

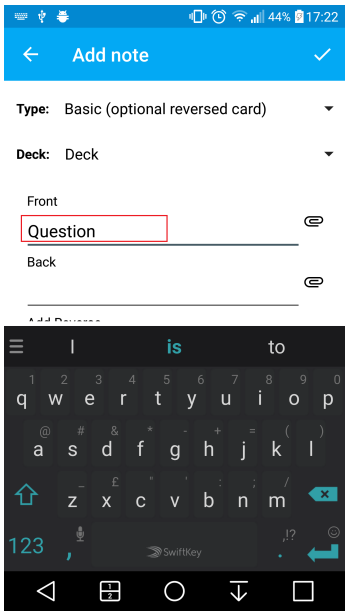


Figure 3.11: *AnkiDroid*: Mutant 1, before rotation

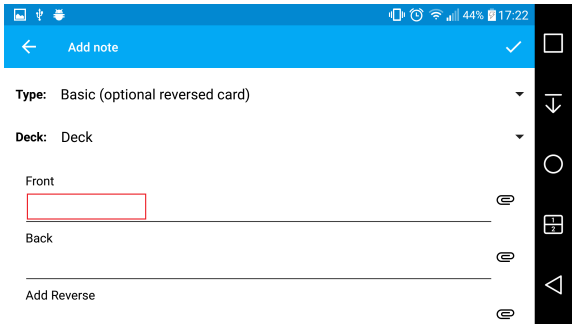


Figure 3.12: *AnkiDroid*: Mutant 1, after rotation

The insertion of the mutant 2 will force the data that is inserted by the user to disappear when the orientation of the phone changes. In Figure 3.13 shows the application after the user inserts the email "email@gmail.com" into the *EditText* widget. Figure 3.14 shows at happens to the application after the orientation is changed, the data inserted disappears.

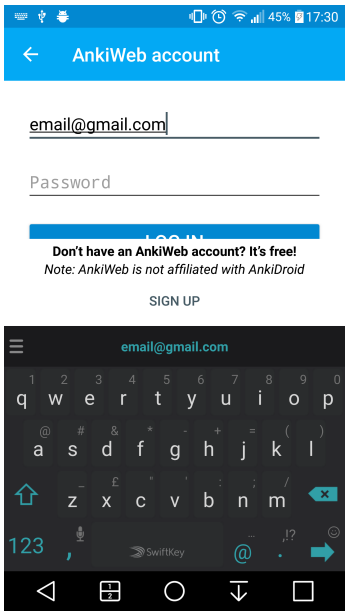


Figure 3.13: *AnkiDroid*: Mutant 2, before rotation

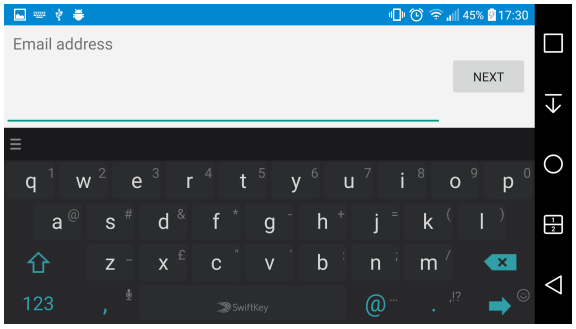


Figure 3.14: *AnkiDroid*: Mutant 2, after rotation

The insertion of the mutant 3 will force the visible *Spinner* widget to go back to its hidden

state when the orientation changes. In Figure 3.15 shows the application after the user clicks on the hidden *Spinner* widget making it visible in the activity. Figure 3.16 shows at happens when the orientation changes, the *Spinner* is no longer visible.

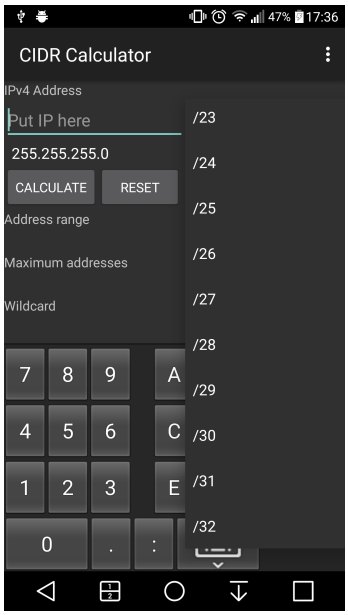


Figure 3.15: *CIDR Calculator*: Mutant 3, before rotation

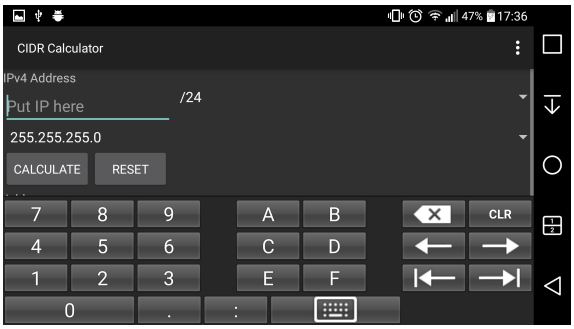


Figure 3.16: *CIDR Calculator*: Mutant 3, after rotation

The insertion of the mutant 4 will force any data that is not being save to be lost, this can be data that was inserted by the user, or a popup dialog. In Figure 3.17 shows the application after the user clicks a button that shows a dialog to select a date. Figure 3.18 shows at happens when the orientation changes, the dialog disappears.

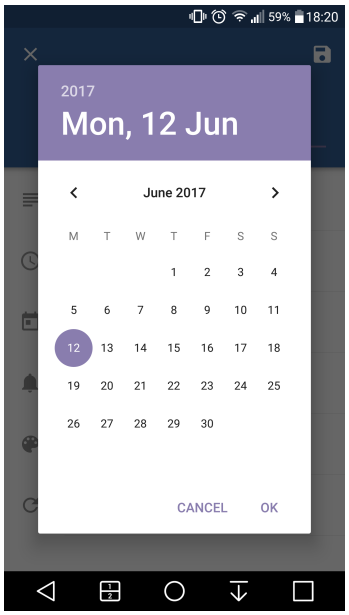


Figure 3.17: *Recurrence*: Mutant 4, before rotation

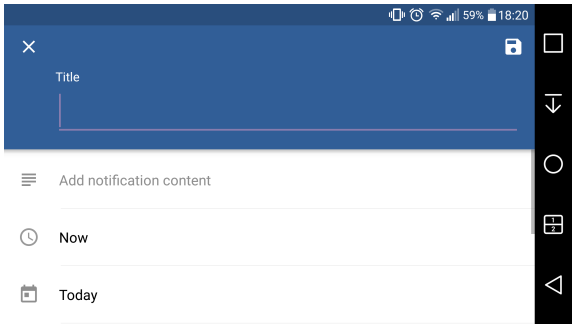


Figure 3.18: *Recurrence*: Mutant 4, after rotation

### 3.3.2 Tab

The most common way to create tabs in Android is to use the ViewPager widget [Devd]. The *ViewPager* widget has built-in swipe gestures

By using this widget the default behaviour of swiping the screen is to change the tab that is visible according with the direction of the swipe. So, when an application does not allow the swiping of tabs it is because the developers have explicitly deactivated the functionality in the code.

The *ViewPager* has two methods that are responsible for handling touch events [Devh]:

**onTouchEvent(MotionEvent)** – handles touch screen motion events

**onInterceptEvent(MotionEvent** – intercepts all touch screen motion events

If the developer wants to modified the way the tab in their application responds to the touch gestures, the they have to define a custom *ViewPager* subclass, that overrides those methods and performs the necessary actions. If the goal of the developer is to disable the ability to change the tabs with the swipe gesture, then they have to change the return value of those methods to "false" and remove any other actions performed in those methods if they exist.

It is possible to implement tabs in Android without using the *ViewPager* and using a *TabLayout*. However, by choosing this solution the tabs will not change with the swipe gesture only with by touching the tab name. So, an application that does not use the *ViewPager* will always be considered as having the tab pattern incorrectly implemented by the *iMPAcT tool*.

Only one mutant was defined for this pattern, but there are different ways of injecting it depending on how the target application was developed.

1. If the application does not extend the *ViewPager* class:

   - Create a new class that extends the *ViewPager* class, for example *CustomViewPager*

```
public CustomViewPager(Context context, AttributeSet attrs) {
      super(context, attrs);
  }
```

   - Override the onTouchEvent (MotionEvent) and the onInterceptTouchEvent (MotionEvent) functions with a return value of "false"

```
@Override
  public boolean onTouchEvent(MotionEvent ev) {
    return false;
  }
@Override
  public boolean onInterceptTouchEvent(MotionEvent ev) {
    return false;
  }
```

   - Class constructor also has to be created

```
public CustomViewPager(Context context, AttributeSet attrs) {
    super(context, attrs);
}
```

   - Import packages needed (if this process is done with the help of an IDE the imports will be inserted automatically by the IDE)

```
import android.support.v4.view.ViewPager
import android.util.AttributeSet
import android.view.MotionEvent
import android.content.Context
```

   - Change all references of the default *ViewPager* to new class created in the XML layout files, for example:

```
<android.support.v4.view.ViewPager
android:id="@+id/pager"
```

```
android:layout_width="match_parent"
android:layout_height="0px"
android:layout_weight="1"
android:background="@android:color/white"/>
```

becomes

```
<com.exampleApp.CustomViewPager
android:id="@+id/pager"
android:layout_width="match_parent"
android:layout_height="0px"
android:layout_weight="1"
android:background="@android:color/white"/>
```

2. If the applications already extends the *ViewPager* class:

   (a) If it overrides the onTouchEvent (MotionEvent) and the OnInterceptTouchEvent (MotionEvent):

   - Change the value that is returned from the function to "false"

```
@Override
  public boolean onTouchEvent(MotionEvent ev) {
    return false;
  }
@Override
  public boolean onInterceptTouchEvent(MotionEvent ev) {
    return false;
  }
```

   (b) If it does not override the functions:

   - Override the functions with the return value of "false"

```
@Override
  public boolean onTouchEvent(MotionEvent ev) {
    return false;
  }
@Override
  public boolean onInterceptTouchEvent(MotionEvent ev) {
    return false;
  }
```

   - It can be necessary to add the import of the MotionEvent package

```
```

```
1 import android.view.MotionEvent;
```

#### 3.3.2.1 Visible effects

All of the different approaches defined here have the same visible effect when injected in an application. It will stop the tabs from changing when the screen is swiped.

### 3.3.3 Side Drawer

The mutants defined for this pattern are related with the height of the side drawer, because that is the only thing that is tested by the *iMPAcT tool*. The dimensions of the widgets is commonly defined in XML files, it is also possible to set the dimensions in the Java code but this is not as common and none of the applications selected did this.

There are several ways to create a side drawer, but a *DrawerLayout* is always needed because it is the widget that allows the existence of an interactive drawer [Deve]. The widgets inside the *DrawerLayout* can be:

- *FrameLayout* for the main content and a *ListView* for the drawer content

```
<android.support.v4.widget.DrawerLayout
  android:id="@+id/drawer_layout"
  ...>
  <FrameLayout
    android:id="@+id/main_content"
    ... />
  <ListView
    android:id="@+id/left_drawer"
        .../>

</android.support.v4.widget.DrawerLayout>
```

- *FrameLayout* for the main content and another *FrameLayout* for the drawer content

```
<android.support.v4.widget.DrawerLayout
  android:id="@+id/drawer_layout"
  ...>
  <FrameLayout
    android:id="@+id/main_content"
    ... />
  <FrameLayout
    android:id="@+id/left_drawer"
```

```
    ... />
</android.support.v4.widget.DrawerLayout>
```

- *FrameLayout* for the main content and a *NavigationView* for the drawer content

```
<android.support.v4.widget.DrawerLayout
  android:id="@+id/drawer_layout"
  ...>
  <FrameLayout
    android:id="@+id/main_content"
    ... />
  <android.support.design.widget.NavigationView
        android:id="@+id/navigation_view"
        ... />
</android.support.v4.widget.DrawerLayout>
```

Instead of using the classes *DrawerLayout* or *FrameLayout*, a subclass of these classes can be used.

Each of these widgets has a set of attributes with values that can be altered in order to configure them to the desired layout. These attributes can be layout_height, layout_width, backgroud, fitsSystemWindows, among others

The ones that are tested in the side drawer pattern of the *iMPAcT tool* are the ones related with the height of the drawer. This involves the layout_height and the fitsSystemWindows attributes. The layout_height attribute is the one that defines the height of the widget associated, most common values are: match_parent, widget occupies the same space as the parent widget; wrap_content, the height of the widget depends on the content that is placed there; and an exact value for the height, value can be in pixels (px), density-independent pixels (dp), scaled pixels (sp), inches (in) or milllimeters (mm) [Devg]. The attribute fitsSystemWindows adjusts the layout of the view/widget based on system windows like the notification bar, and has as values either "true" or "false". If the value is "true" then the view will span the full height of the screen, including behind the notification bar, content that is behind the notification bar will be visible but will be a little bit darker than it should be. If the value is "false" then the view will start only after the notification bar.

The mutants were defined having this in mind. They affect either the *DrawerLayout*, the *NavigationView* or the *FrameLayout* (if it belongs to the side drawer).

1. If a *DrawerLayout* is present:
    - Change the value of the attribute layout_ height to some value that is smaller than the screen size but bigger than the middle of the screen

```
<android.support.v4.widget.DrawerLayout
  android:id="@+id/drawer_layout"
  android:layout_height="1400px"
  ...>
  ...
</android.support.v4.widget.DrawerLayout>
```

2. If *NavigationView* is present:

   (a) Attribute android:fitsSystemWindows

   - If android:fitsSystemWindows="true", change the value to "false"

```
<android.support.design.widget.NavigationView
  android:id="@+id/navigation_view"
  android:fitsSystemWindows="false"
  ... />
```

   - If the attribute android:fitsSystemWindows is not set, add the attribute with value "false"

```
<android.support.design.widget.NavigationView
  android:id="@+id/navigation_view"
  android:fitsSystemWindows="false"
  ... />
```

   (b) Change the value of the attribute layout_ height to some value that is smaller than the screen size but bigger than the middle of the screen

```
<android.support.design.widget.NavigationView
  android:id="@+id/navigation_view"
  android:layout_height="1400px"
  ... />
```

3. If a *FrameLayout* that has as a parent a *DrawerLayout* is present (is not necessary that the *DrawerLayout* and the *FrameLayout* have to be in the same file, the *DrawerLayout* can import a *FrameLayout* that is defined in another file):

   (a) Attribute android:fitsSystemWindows

   - If android:fitsSystemWindows="true", change the value to "false"

```
<android.support.v4.widget.DrawerLayout
```

```
  ...>
  <FrameLayout
    android:id="@+id/left_drawer"
    android:fitsSystemWindows="false"
    ... />
</android.support.v4.widget.DrawerLayout>
```

- If the attribute android:fitsSystemWindows is not set, add the attribute with value "false"

```
<android.support.v4.widget.DrawerLayout
  ...>
  <FrameLayout
    android:id="@+id/left_drawer"
    android:fitsSystemWindows="false"
    ... />
</android.support.v4.widget.DrawerLayout>
```

(b) Change the value of the attribute layout_ height to some value that is smaller than the screen size but bigger than the middle of the screen

```
<android.support.v4.widget.DrawerLayout
  ...>
  <FrameLayout
    android:id="@+id/left_drawer"
    android:layout_height="1400px"
    ... />
</android.support.v4.widget.DrawerLayout>
```

#### 3.3.3.1 Visible effects

The mutant 1 reduces the height of the parent of the *NavigationView* or the *FrameLayout*, forcing the height of the drawer layout to be smaller then the screen even if the child's height is set to "match_parent". Figure 3.19 shows the effect of the mutant in the application *OctoDroid*. The *DrawerLayout* ends before the end of the screen, so anything that will be placed inside that will also end before the screen.

The mutants 2b and 3b works similarly to the mutant 1, but reduces the height of the NavigationView or the FrameLayout, respectively, instead of the height of parent view. Figure 3.20 shows the visible effect of the injection of the mutant 2b to the *OctoDroid* application. Here, only the side drawer is smaller, the rest of the application continues with the same height as before. And Figure 3.24 shows the visible effect of the injection of the mutant 3b on the *FOSDEM* application.

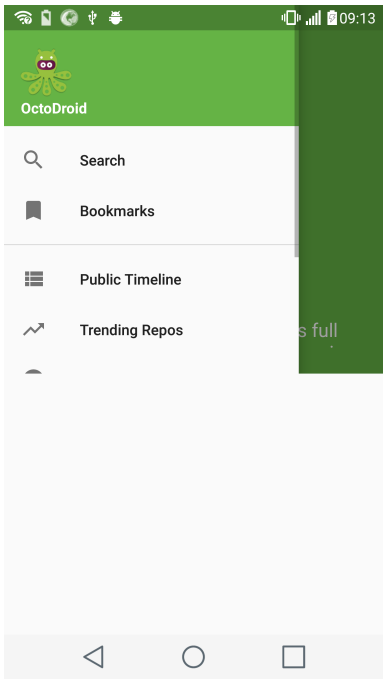


Figure 3.19: Example of mutant 1, side Drawer

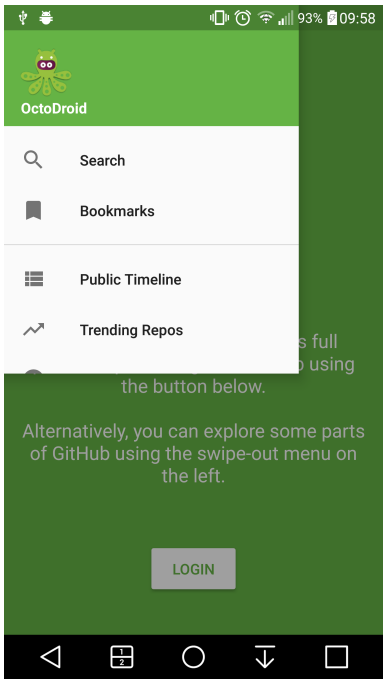


Figure 3.20: Example of mutant 2b, side Drawer

By comparing these figures, 3.20 and 3.24, it is possible to verify that although the mutants target different widgets the visible effect is the same.

The mutants 2a and 3a have the same visible effect, the side drawer is drawn underneath the notification bar, as is possible to verify by comparing Figures 3.21 and 3.23. Figure 3.21 shows the visible effect of the mutant 2a when injected in the *OctoDroid* application, and Figure 3.23 shows the visible effect of the mutant 3a when injected in the *FOSDEM* application.

Figure 3.22 shows how the side drawer of the application *OctoDroid* should appear when no mutant is injected. The side drawer is still visible even behind the notification bar.

### 3.3.4 Resource Dependency

Since no application that had this pattern implemented incorrectly was found, it was impossible to define mutants for the resource dependency pattern.

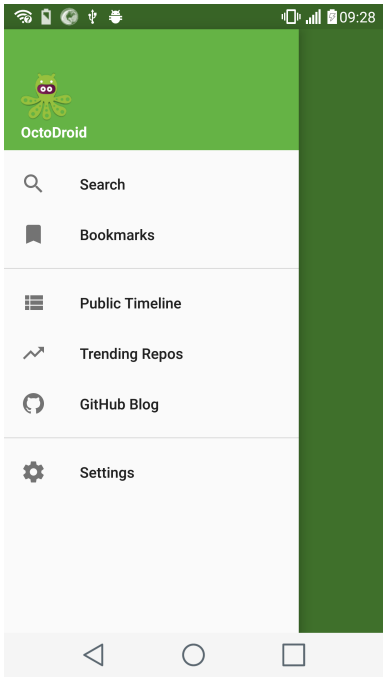


Figure 3.21: Example of mutant 2a, side drawer

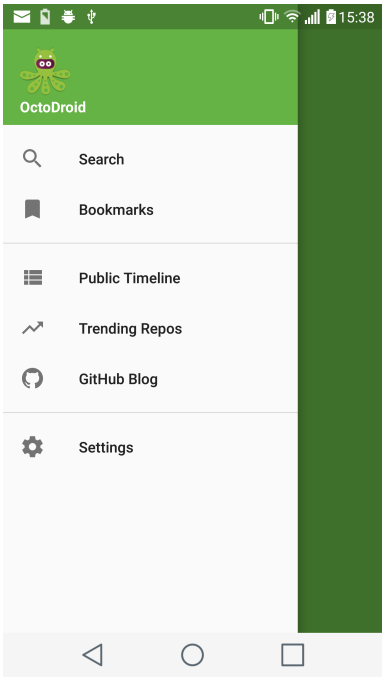


Figure 3.22: Example side drawer with no errors

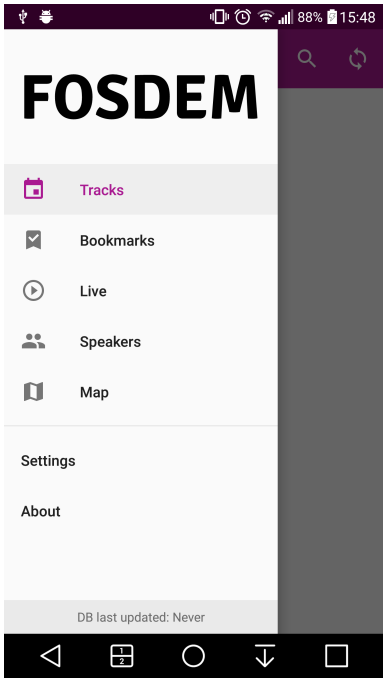


Figure 3.23: Example of mutant 3a, side drawer

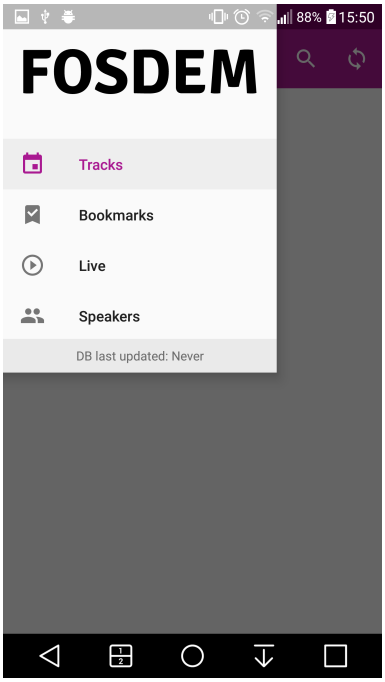


Figure 3.24: Example of mutant 3b, side drawer

## 3.4 Automation

Only the insertion of the mutants related with the Tab pattern and the Side Drawer pattern were automate. The mutants of the Orientation pattern were not automated because there was no time to do it. The orientation mutants are more difficult to automate because they involve several files of the same activity. A Java application was created to do this, and it assumes that the code of the target Android application has no compiling errors and uses gradle. Since the mutation operators related to each of the pattern involve different type of files, the approach for each of the patterns is explained separately.

### 3.4.1 Tab

There is only one mutant to be automated, but there are different approaches depending on how the code was developed. In two approaches only the Java files are used, and in the other one only XML files are used and a Java class has to be created. The code is based on some assumptions:

- The directory with the source code of the target application has to be in the same directory as the tool
- The name of the directory has to be defined in the code, variable "DIRECTORY"
- The root directory of the java code has to defined in the variable "DIRECTORY_BASE_SRC". Example using the application Conversations:

```
private static final String DIRECTORY_BASE_SRC = "\\src\\main\\java\\eu\\
    siacs\\conversations\\";
```

- The name of the package of the application has to be defined in the variable "PACKAGE_NAME"

```
private static final String PACKAGE_NAME = "eu.siacs.conversations"
```

The tool is divided into 3 steps:

**1. The directory of the target application is copied to a new folder**

All the files and folders are copied except the ones that start with ".", or are inside folders that are generated automatically when the application builds

**2. The files are analysed to check if there is a subclass of the ViewPager class in the application**

If there is a subclass of the *ViewPager* class:

- If the class overrides the methods onTouchEvent( MotionEvent) and the onInterceptEvent( MotionEvent) – the only thing to do is to change the return value, approach 2a

```
// each line of the file
while ( (currentLine = br.readLine() ) != null){
  Matcher imp = onTouch.matcher(currentLine);
  Matcher m = onIntercept.matcher(currentLine);
  if ( imp.find()){ // if found the declaration of the onTouchEvent(
      Motion) method
    bw.write(currentLine);
    bw.write("\\n");
    bw.write("return false;");
    bw.write("\\n");
    foundOnTouch = true;
  }else if ( m.find()){ // if found the declaration of the
      onInterceptEvent(Motion) method
      bw.write(currentLine);
      bw.write("\\n");
      bw.write("return false;");
      bw.write("\\n");
      foundOnIntercept = true;
    }else {
      bw.write(currentLine);
      bw.write("\\n");
  }
}
```

- If the class does not override the methods, the only thing to do is to insert the methods with a return value of "false". These are inserted in the next line after the definition of the class, approach 2b:

```
if ( !foundOnTouch && !touchDone){
  bw.write("  @Override");
  bw.write("\\n");
  bw.write("  public boolean onTouchEvent(MotionEvent ev) {");
  bw.write("\\n");
  bw.write("    return false;");
  bw.write("\\n");
  bw.write("  }");
  bw.write("\\n");
  touchDone = true;
}
if ( !foundOnIntercept && !interceptDone ){
  bw.write("  @Override");
  bw.write("\\n");
  bw.write("  public boolean onInterceptTouchEvent(MotionEvent ev) {");
  bw.write("\\n");
```

```
  bw.write("    return false;");
  bw.write("\\n");
  bw.write("  }");
  bw.write("\\n");
  interceptDone = true;
}
```

If the application does not extend the *ViewPager* class, approach 1:

- Creates the new file with the subclass of the *ViewPager*, name of the class is defined in the variable "VIEWPAGER_NAME_DEFAULT". This file is created in the root directory of the source (defined in a variable).
- All the XML files are selected
- From those files, the ones that have a *ViewPager* reference are selected
- Each XML file is parsed to replace the references to the *ViewPager* for the new class and saved with the original name

```
while ( (currentLine = br.readLine() ) != null){
  Matcher begin = viewPagerBeginTag.matcher(currentLine);
  Matcher end = viewPagerEndTag.matcher(currentLine);

  if ( begin.find()){
    if ( begin.group(1) != null)
      bw.write("<"+PACKAGE\_NAME+"."+VIEWPAGER\_NAME + "  " + begin.
          group(1));
    else
      bw.write("<"+PACKAGE\_NAME+"."+VIEWPAGER\_NAME);
    bw.write("\\n");
  }else if(end.find()){
    if ( end.group(1) != null)
      bw.write("</"+PACKAGE\_NAME+"."+VIEWPAGER\_NAME+ end.group(1));
    else
      bw.write("</"+PACKAGE\_NAME+"."+VIEWPAGER\_NAME+">");
    bw.write("\\n");
  }else{
    bw.write(currentLine);
    bw.write("\\n");
  }
}
```

**3. The mutated APK is created**

Since all the applications use gradle, only one command is needed to build the APK

```
private static void createAPK(String mutDirectory) {
  String cmd = "cmd /c start /wait cmd.exe /K \"cd " +  MUT_DIRECTORY + " &&
      gradlew assemble";
  Utils.executeCmd(cmd);
}
```

### 3.4.2 Side Drawer

All the mutants defined for this pattern are related with XML files, so the approach is a little different than the one use for the automation of the mutant for the tab pattern. The code is based on some assumptions:

- The directory with the source code of the target application has to be in the same directory as the tool
- The name of the directory has to be defined in the code, variable "DIRECTORY"
- The mutant to be applied has to be defined in the code, variable "MUT". This variable can take
  - "drawer_1" – Mutant 1
  - "navigation_1" – Mutant 2a
  - "navigation_2" – Mutant 2b
  - "frameLayout_1" – Mutant 3a
  - "frameLayout_2" – Mutant 3b

The tool is divided into 4 steps:

**1. The directory of the target application is copied to a new folder**

All the files and folders are copied except the ones that start with ".", or are inside folders that are generated automatically when the application builds

**2. XML files are selected**

Since the mutants are applied only to XML files there is no need to parse the Java files, so only the XML files will be parsed

**3. Each XML file is parsed**

If the file has the tag that corresponds with the target mutant, then the mutant is applied and the file is saved with the same name. To check if the tag exists in the file, each node is analysed in order to find the one that has the tag:

```
private static Node findNode(NodeList doc, Pattern tag_to_find) {
  for (int i = 0; i < doc.getLength(); i++){

    Node tmp = doc.item(i);

    if ( tmp.getNodeType() == Node.ELEMENT_NODE){
      Matcher mat = tag_to_find.matcher(tmp.getNodeName());
      if ( mat.find()){
        return tmp;
      }
    }
    ...
```

After having the target file selected, the mutant is applied to the attribute that it pertains. The next snippet is an example of how the mutants are applied, in this case the mutants are related with the *NavigationView*:

```
...
  if ( MUT.split("_")[0].equals("navigation")){
    tmp = findNode(doc_tmp.getChildNodes(), NAVIGATION_VIEW);
    tmp2 = tmp;
    if ( tmp == null)
      return null;
    if ( MUT.split("_")[1].equals("1")){
      if ( tmp2.getAttributes().getNamedItem("android:fitsSystemWindows") !=
          null){
        tmp2.getAttributes().getNamedItem("android:fitsSystemWindows").
            setNodeValue("false");
        altered = true;
      }else {
        ((Element)tmp2).setAttribute("android:fitsSystemWindows", "false");
        altered = true;
      }
    }if ( MUT.split("_")[1].equals("2")){
      tmp2.getAttributes().getNamedItem("android:layout_height").setNodeValue
          ("1400px");
      altered = true;
    }
    doc_tmp.getChildNodes().item(counter).replaceChild(tmp2, tmp);
  }
...
```

**4. The mutated APK is created**

Since all the applications selected used gradle, only one command is needed to build the APK

```
private static void createAPK(String mutDirectory) {
  String cmd = "cmd /c start /wait cmd.exe /K \"cd " +  MUT + " && gradlew
      assemble";
  Utils.executeCmd(cmd);
}
```

The APK will be created inside the folder "outputs", that is inside the "build" folder.

### 3.4.3 Limitations

There are some limitations associated with this automation:

- It was only tested with some applications – there is no guarantee that it will work with every application.
- No UI – uses a lot of hardcoded variables.
- Can be slow – depending on the size of the application, and the files.
- Only works with the default classes – does not work with subclasses of those classes.
- Does not verify if the code that is being altered is reachable.
- In order to create the APK the application has to use gradle. Otherwise, the mutants will still be inserted but the APK will have to be created manually.

## 3.5 Conclusion

In this chapter ten mutation operators are defined, these operators are distributed between 3 of the patterns tested by the *iMPAcT tool* and are specific for Android applications. Since the mutation operators were defined having in consideration what is tested by the *iMPAcT tool* (the development of applications following the guidelines of Android programming) and the most common errors detected in real applications, they can be used to assess the effectiveness of the *iMPAcT tool*.

From these ten mutation operators, six were automated. This means that the mutation operators related with the tab and the side drawer pattern can be automatically inserted into an application. If the mutation operator selected can be inserted in the target application a new APK will be created with the necessary modifications. The mutation operators related with the orientation pattern were not automated, because of lack of time. These mutation operators will not be easy to automated because there is a lot of dependencies that have to be verified before the operator can be applied.

Since it was not possible to find any Android application that had the resource dependency incorrectly implemented, no mutation operators were defined for this pattern. Consequently, this pattern will not be referred in Chapter 4.

# Chapter 4

# Case study

In this chapter it is presented the result of the insertion of the mutants defined in Section 3.3 and the results given by the *iMPAcT tool* for each application and mutant. The set of applications that were considered by the *iMPAcT tool* as having each of the patterns correctly implemented will be used in this phase. However, not everyone of those applications was used, each pattern explains with detail why some applications were rejected from this phase. The last section of this chapter describes how the automation of the insertion of the mutants was developed.

This chapter is divided by the different patterns, and in each pattern it is described the criteria for the selection of the applications, which mutants were inserted in each application and the results.

## 4.1 Orientation

Even thought the number of applications that had the orientation pattern correctly implemented was very big, only a few were used in this phase. This was because most of the applications were too simple, and did met the requirements for the mutants to be inserted. While other applications were mostly developed using Kotlin or Scala, which made impossible to insert the mutants. So from the 40 applications that have the pattern correctly implemented and that allow rotation, only 14 were used in this phase, these applications are shown in Table 4.1.

Table 4.1: Orientation Pattern: Mutants inserted

| Application | Mutant |
|---|---|
| ForkHub for GitHub | Mutant 2 |
| CIDR Calculator | Mutants 2 and 3 |
| AnkiDroid | Mutants 1, 2 and 4 |
| Bewegungsmelder | Mutant 4 |
| MIFARE Classic Tool | Mutants 2 and 4 |
| Cadroid | Mutant 2 |
| Lightning | Mutants 4 and 2 |
| NetGuard | Mutant 4 |
| Network Monitor | Mutants 1 and 2 |
| NoNonsense Notes | Mutants 1 and 2 |
| RasPi Check | Mutants 2 and 3 |
| Recurrence | Mutant 4 |
| Reddinator | Mutants 4 and 2 |
| Weechat Android | Mutant 2 |

Table B.1 shows the reason why each application was removed from this phase. When an application has as reason to be excluded the fact that there was nothing to test, it means that the application was very simple with only one activity, or that the application did not require any input from the user, or popup dialogs. Some of those applications were games, that had no settings menu, and only the game screen that consisted in drawings.

### 4.1.1 *iMPAcT tool* results

The Table 4.2 shows the results of the insertion of the mutants for the selected applications.

One problem encountered with the testing of the orientation pattern is that the *iMPAcT tool* is not able to detect errors related with the search widget. The *iMPAcT tool* is not able to detect when the query of a search disappears when the screen rotates. This had already happen in the initial tests with the application *Timber*. But at that time it was thought that the problem could be related with the nature of the application, since it is a music player and the vast majority of the time spent in testing was spent in the activity that play the track. However after inserting the mutant 4 in other applications and the error being visible but not recognized by the *iMPAcT tool* it was clear

that the problem was not related with the applications being tested but related with the tool that was testing it.

Table 4.2: Orientation Pattern: Mutants results

| Application | Mutant 1 | Mutant 2 | Mutant 3 | Mutant 4 |
|---|---|---|---|---|
| ForkHub for GitHub | Not Tested | YES | Not Tested | NO [1] |
| CIDR Calculator | Not Tested | YES | YES | Not Tested |
| AnkiDroid | YES | YES | Not Tested | YES |
| Bewegungsmelder | Not Tested | Not Tested | Not Tested | YES |
| MIFARE Classic Tool | Not Tested | YES | Not Tested | YES |
| Cadroid | Not Tested | YES | Not Tested | Not Tested |
| Lightning | Not Tested | YES | Not Tested | NO[1]/YES [2] |
| NetGuard | Not Tested | Not Tested | Not Tested | NO [1]/YES [2] |
| Network Monitor | YES | NO [3] | Not Tested | Not Tested |
| NoNonsense Notes | YES | YES | Not Tested | Not Tested |
| RasPi Check | Not Tested | YES | YES | Not Tested |
| Recurrence | Not Tested | Not Tested | Not Tested | YES |
| Reddinator | Not Tested | Not Tested | Not Tested | YES |
| Weechat Android | Not Tested | YES | Not Tested | Not Tested |

## 4.2 Tab

From all the applications that were classified by the *iMPAcT tool* as having the tab pattern correctly implemented, two were excluded from this phase. The *OI File Manager* application was excluded because it did not contain tabs even though the *iMPAcT tool* detected one. No tab was detected when analysing the code or by manually exploring the application. The *Antox* application was excluded because it was developed using mainly Scala.

All of the mutated applications present the same visible effect: when swiping the screen the selected tab does not change. They are divided into 3 approaches because of how they are inserted into the code of the application.

[1] Does not detect the disappearance of the search query
[2] Detected the disappearance of a popup
[3] iMPAcT tool does not reach the screen

Table 4.3: Tab Pattern: Mutants inserted

| Application | Approach |
|---|---|
| Timber | Approach 2b |
| Conversations (Jabber / XMPP) | Approach 1 |
| Forkhub | Approach 2b |
| WordPress | Approach 2a |
| FOSDEM Companion | Approach 1 |
| MHGen Database | Approach 1 |
| OctoDroid | Approach 1 |
| Open Tasks | Approach 1 |
| Poet Assistant | Approach 1 |

### 4.2.1 *iMPAcT tool* results

As Table 4.4 shows, the *iMPAcT tool* was able to detect the inserted errors in all the applications. The *iMPAcT tool* was not able to detect the error in all the tests to the application *Conversations*, only one test was able to detect the error (see B.7).

Table 4.4: Tab Pattern: Mutants results

| Application | Detected |
|---|---|
| Timber | Yes |
| Conversations (Jabber / XMPP) | No[1] |
| WordPress | Yes |
| FOSDEM Companion | Yess |
| MHGen Database | Yes |
| OctoDroid | Yes |
| Open Tasks | Yes |
| Poet Assistant | Yes |
| Forkhub | Yes |

This is related to the fact that in the first and third tests the *iMPAcT tool* did not reach the screen in which the mutation was inserted even though the exploration mode used in the *iMPAcT*

[1] iMPAcT tool does not reach the screen

*tool* was to test "all events". This was a recurrent problem during the tests, sometimes the *iMPAcT tool* does not test all the possible screens.

## 4.3 Side Drawer

Only one application was excluded from this phase. The *File Manager* application was excluded because of their implementation of the side drawer, it did not occupy the whole height but the *iMPAcT tool* was not able to detected. Figure 4.1 shows the side drawer of the application.

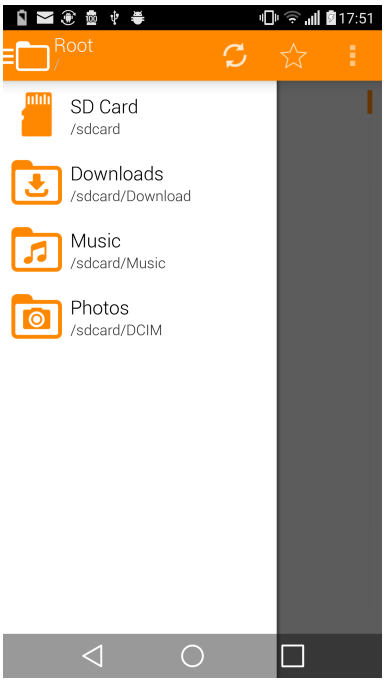


Figure 4.1: Side Drawer example of the *File Manager* application

The mutants were applied to all the other applications, depending on the way that each application implemented the side drawer.

Table 4.5: Side Drawer Pattern: Mutants inserted

| Application | Mutants |
|---|---|
| Timber | Mutants 1, 2a and 2b |
| MusicDNA | Mutants 1, 2a and 2b |
| Yaaic - IRC Client | Mutants 1 , 3a and 3b |
| GDG - News and Events | Mutants 1, 2a and 2b |
| Polar Dashboard Sample | Mutants 1, 2a and 2b |
| Etar - OpenSource Calendar | Mutants 1, 2a and 2b |
| wallabag | Mutants 1, 2a and 2b |
| Twittnuker for Twitter | Mutants 1, 2a and 2b |
| Equate | Mutants 1, 2a and 2b |
| FOSDEM Companion | Mutants 1, 3a and 3b |
| AnkiDroid | Mutants 1, 2a and 2b |
| Lightning | Mutants 1, 3a and 3b |
| OctoDroid | Mutants 1, 2a and 2b |
| QR Scanner | Mutants 1, 2a and 2b |
| Quick Lyrics | Mutants 1, 3a and 3b |
| Riot | Mutants 1, 2a and 2b |
| SMSsync | Mutants 1, 2a and 2b |
| Toffed | Mutants 1, 2a and 2b |
| Unit Converter Ultimate | Mutants 1, 2a and 2b |
| Web Opac | Mutants 1, 2a and 2b |
| WiFiAnalyzer | Mutants 1, 2a and 2b |

### 4.3.1 *iMPAcT tool* results

In Figure 4.6 shows the results given by the *iMPAcT tool* for each mutant inserted and application. The *iMPAcT tool* was able to detect every error that was injected. The combination of each mutant and application was tested three times, and the error was detected every time. Different exploration modes of the *iMPAcT tool* were used in different tests. The first and second tests used the "All events" exploration and the third used the "Priority to not executed" exploration. The errors were detected regardless of the exploration mode used, because once the side drawer was open the error was immediately detected.

Table 4.6: Side Drawer Pattern: Mutants results

| Application | Mutant 1 | Mutant 2b | Mutant 2a | Mutant 3b | Mutant 3a |
|---|---|---|---|---|---|
| Timber | YES | YES | YES | Not tested | Not tested |
| MusicDNA | YES | YES | YES | Not tested | Not tested |
| Yaaic - IRC Client | YES | Not tested | Not tested | YES | YES |
| GDG - News and Events | YES | YES | YES | Not tested | Not tested |
| Polar Dashboard Sample | YES | YES | YES | Not tested | Not tested |
| Etar - OpenSource Calendar | YES | YES | YES | Not tested | Not tested |
| wallabag | YES | YES | YES | Not tested | Not tested |
| Twittnuker for Twitter | YES | YES | YES | Not tested | Not tested |
| Equate | YES | YES | YES | Not tested | Not tested |
| FOSDEM Companion | YES | Not tested | Not tested | YES | YES |
| AnkiDroid | YES | YES | YES | Not tested | Not tested |
| Lightning | YES | Not tested | Not tested | YES | YES |
| OctoDroid | YES | YES | YES | Not tested | Not tested |
| QR Scanner | YES | YES | YES | Not tested | Not tested |
| Quick Lyrics | YES | Not tested | Not tested | YES | YES |
| Riot | YES | YES | YES | Not tested | Not tested |
| SMSsync | YES | YES | YES | Not tested | Not tested |
| Toffed | YES | YES | YES | Not tested | Not tested |
| Unit Converter Ultimate | YES | YES | YES | Not tested | Not tested |
| Web Opac | YES | YES | YES | Not tested | Not tested |
| WiFiAnalyzer | YES | YES | YES | Not tested | Not tested |

## 4.4 Conclusion

This chapter exposes the results given by the *iMPAcT tool* when testing the various mutated applications. In most cases the *iMPAcT tool* was able to detect the insertion of the mutation, but in some applications they were not detect. Next are presented the results by pattern and mutation operator:

**Orientation:**

- Mutant 1 was inserted into three applications and was detected by the *iMPAcT tool* in all applications.
- Mutant 2 was inserted into ten applications, but was only detected by the *iMPAcT tool* in nine applications.
- Mutant 3 was inserted into two applications and was detected by the *iMPAcT tool* in all applications
- Mutant 4 was inserted into eight applications but was only fully detected by the*iMPAcT tool* in five applications.

**Tab:**

- The mutant was inserted into nine applications, but was only detected by the *iMPAcT tool* in eight applications.

**Side Drawer:**

- Mutant 1 was inserted into twenty-one different applications and detected by the *iMPAcT tool* in all the applications.
- Mutant 2b was inserted into seventeen different applications and detected by the *iMPAcT tool* in all the applications.
- Mutant 2a was inserted into seventeen different applications and detected by the *iMPAcT tool* in all the applications.
- Mutant 3b was inserted into four different applications and detected by the *iMPAcT tool* in all the applications.
- Mutant 3a was inserted into four different applications and detected by the *iMPAcT tool* in all the applications.

By focusing in analysing the execution and the logs of the *iMPAcT tool* in the cases that it was not able to detect the mutants, two problems were found:

- The *iMPAcT tool* is not prepared to detect the loss of information in the search widget. Figures 4.2 and 4.3 show the loss of information of the search widget in the *Forkhub* application. This loss is not detected by the *iMPAcT tool*.
- The *iMPAcT tool* is not always able to reach all the screens/activities of an application, even when the "all events" exploration mode is used.

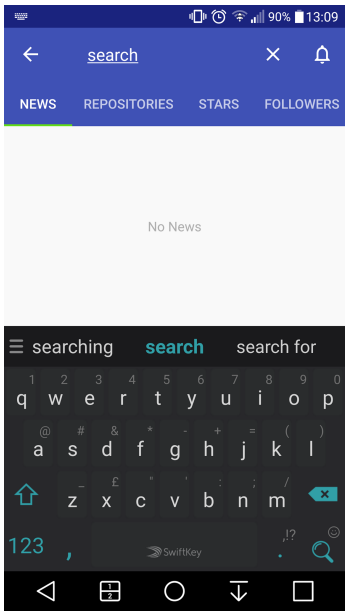


Figure 4.2: *Forkhub* with mutant 4 before rotation

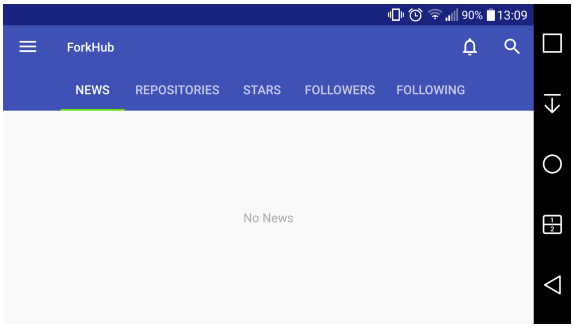


Figure 4.3: *Forkhub* with mutant 4 after rotation

Figures 4.2 and 4.3 show the error inserted into the application *Forkhub* (insertion of mutant 4), after rotating the screen the search query is lost. Even though this error is visible in the application it was not detected by the *iMPAcT tool* in any test.

The first problem is associated with the technique used by the *iMPAcT tool* to detect the loss of information after rotating the screen, while the second problem is related with the algorithm used by the *iMPAcT tool* to explore each application. These problems are still present even when the exploration mode of the *iMPAcT tool* is changed.

# Chapter 5

# Conclusion and Future Work

In this work several Android specific mutation operators were defined. The mutation operators were related to three patterns: four mutation operators for the orientation pattern; one mutation operator for the tab pattern; and five for the side drawer pattern. No mutation operator was defined for the resource dependency pattern. These mutation operators were defined having in consideration the areas that are tested by the *iMPAcT tool* and the android guidelines in which the *iMPAcT tool* is based.

These mutation operators were inserted in several Android applications, and then tested using the *iMPAcT tool*. Since the *iMPAcT tool* is based on the exploration of the applications, the tests were executed with two different exploration modes: "all events" and "priority to not executed". The time spent in each test was around 20 minutes, and there was not a big difference in time spent executing the tests with the "all events" exploration mode when compared with the time spent with the priority to not executed.

During this work, a tool was developed to automate the insertion of the defined mutation operators. Because of time constraints it was only automated the insertion of six mutation operators, the operators are relative to the tab and the side drawer patterns. This tool was developed in Java, and it capable of insert the mutation operators into the source code of an application and then creating a new and mutated APK.

When analysing the results of testing the mutated applications using the *iMPAcT tool*, two problems were found with the *iMPAcT tool*. One error related with the algorithm used to detect the loss of information when rotating the screen and the other related with the algorithm used to explore the applications.

The *iMPAcT tool* is not capable of detecting errors related with the search widget, even when its behaviour is similar to a *TextView* widget that holds information inserted by the user. There was

one application (*Timber*) that had an error in the search widget (query was lost when the screen rotated) that the *iMPAcT tool* classified as having the orientation pattern correctly implemented. In other applications (*Forkhub*, *Lightning* and *NetGuard*) the error was inserted by the mutation operators, and once again the *iMPAcT tool* did not detect it. Another problem was detected in the *iMPAcT tool*, and this one was not related with a specific pattern or mutation operator: some screens are not reached by the *iMPAcT tool* even though the screen is reachable within the application. No difference was encountered when testing those applications, namely the *Network Monitor* application, with different exploration modes. The application Conversations had several tests in which the *iMPAcT tool* did not detect the error, and only one in which the tool was able to detect the error.

Regarding the mutation operators related with the side drawer pattern, those were always detected by the *iMPAcT tool*. All applications that were tested had the side drawer implemented in their main activity, which made easier the identification of the existence of a side drawer by the *iMPAcT tool*.

## 5.1 Goal Satisfaction

The main goal of this work was to define a set of mutation operators that were specific to Android programming and that could be used to assess the effectiveness of the *iMPAcT tool* in detecting errors related with the bad implementation of the Android guidelines. This goal was successfully met, even though it was not possible to define mutation operators related with the resource dependency pattern. The insertion of these mutation operators was also automated to make the process easier, except the four mutation operators that are related with the orientation pattern.

With those mutation operators it was possible to assess the effectiveness of the *iMPAcT tool*. By analysing if the *iMPAcT tool* was able to detect the insertion of the mutation operators or not, it was possible to find two problems that are associated with the *iMPAcT tool*: the tool is not able to detect errors in the search widget; and the tool is not able to reach all the possible screens of an application (depends on the size of the application). Because of the last problem some errors that can be present in an application may never be found by the *iMPAcT tool*, on the grounds that the screen in which the error is visible will never be reached during the tests.

## 5.2 Future work

Some of the mutation operators defined for the orientation pattern were tested against a small number of applications. In the future more applications should be analysed in order to find more errors, define other mutation operators for this pattern and to aid the validation of the mutation operators defined here.

In this work it was not possible to define any mutation operator related with the resource dependency pattern (in this case the WiFi resource) because no application was found that had the pattern incorrectly implemented. More time should be invested in finding applications that have this pattern implemented incorrectly in order to define the mutation operators for the pattern.

Since the automation of the insertion of the mutation operators is only implemented to two patterns, it would be beneficial to extend this automation to the orientation pattern and its mutation operators. The development of a GUI to help the configuration of the tool would also improve it, since, at the moment, all the configuration is done with the use of hardcoded variables.

# References


[ABL05] J H Andrews, L C Briand, and Y Labiche. Is Mutation an Appropriate Tool for Testing Experiments? In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 402–411, New York, NY, USA, 2005. ACM.

[AFT11] Domenico Amalfitano, Anna Rita Fasolino, and Porfirio Tramontana. A GUI crawling-based technique for android mobile application testing. In *Proceedings - 4th IEEE International Conference on Software Testing, Verification, and Validation Workshops, ICSTW 2011*, pages 252–261. IEEE, mar 2011.

[And] Android. Handling Configuration Changes | Android Developers. https://developer.android.com/guide/topics/resources/runtime-changes.html.

[App] AppBrain. Top categories on Google Play. https://www.appbrain.com/stats/android-market-app-categories.

[App16] AppBrain. Number of available Android applications. http://www.appbrain.com/stats/number-of-android-apps, 2016.

[BV08] Mabel Vazquez Briseno and Pierre Vincent. Observations on performance of client-server mobile applications. In *Proceedings of the 2008 1st International Conference on Information Technology, IT 2008*, pages 1–4. IEEE, may 2008.

[Cig] Cigniti. Top 10 Mobile Testing Problems and How to Avoid Them - Software Testing Blog. http://www.cigniti.com/blog/top-10-mobile-testing-problems-and-how-to-avoid-them/.

[Coi17] Inês Coimbra Morgado. *Automated Pattern-Based Testing of Mobile Applications*. PhD thesis, 2017.

[CPN14] Pedro Costa, Ana C R Paiva, and Miguel Nabuco. Pattern based GUI testing for mobile applications. In *Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology, QUATIC 2014*, pages 66–74. IEEE, sep 2014.

[Deva] Android Developers. Activity | Android Developers. https://developer.android.com/reference/android/app/Activity.html.

[Devb] Android Developers. Android Developers API Guides - The Activity Lifecycle. https://developer.android.com/guide/components/activities/activity-lifecycle.html.

[Devc] Android Developers. Android Vitals | Android Developers. https://developer.android.com/topic/performance/vitals/index.html.

[Devd] Android Developers. Creating Swipe Views with Tabs | Android Developers. http://developer.android.com/intl/es/training/implementing-navigation/lateral.html.

[Deve] Android Developers. DrawerLayout | Android Developers. https://developer.android.com/reference/android/support/v4/widget/DrawerLayout.html.

[Devf] Android Developers. Kotlin and Android | Android Developers. https://developer.android.com/kotlin/index.html.

[Devg] Android Developers. ViewGroup.LayoutParams | Android Developers. https://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html.

[Devh] Android Developers. ViewPager | Android Developers. https://developer.android.com/reference/android/support/v4/view/ViewPager.html.

[Dev13] Android Developers. Application Fundamentals - Android Developers, http://developer.android.com/guide/components/fundamentals.html. https://developer.android.com/guide/components/fundamentals.html, 2013.

[DMAO15] L Deng, N Mirzaei, P Ammann, and J Offutt. Towards mutation analysis of Android apps. In *Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on*, pages 1–10, 2015.

[DOAM17] Lin Deng, Jeff Offutt, Paul Ammann, and Nariman Mirzaei. Mutation operators for testing Android apps. *Information and Software Technology*, 81:154–168, 2017.

[JH09] Yue Jia and Mark Harman. Higher Order Mutation Testing. *Information and Software Technology*, 51(10):1379–1393, 2009.

[JH11] Yue Jia and Mark Harman. An Analysis and Survey of the Development of Mutation Testing. *IEEE Trans. Softw. Eng.*, 37(5):649–678, sep 2011.

[JJI+14] René Just, Darioush Jalali, Laura Inozemtseva, Michael D Ernst, Reid Holmes, and Gordon Fraser. Are Mutants a Valid Substitute for Real Faults in Software Testing? In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, pages 654–665, New York, NY, USA, 2014. ACM.

[MDE12] Henry Muccini, Antonio Di Francesco, and Patrizio Esposito. Software Testing of Mobile Applications: Challenges and Future Research Directions. In *Proceedings of the 7th International Workshop on Automation of Software Test*, AST '12, pages 29–35, Piscataway, NJ, USA, 2012. IEEE Press.

[MEK+12] Sam Malek, Naeem Esfahani, Thabet Kacem, Riyadh Mahmood, Nariman Mirzaei, and Angelos Stavrou. A framework for automated security testing of android applications on the cloud. In *Proceedings of the 2012 IEEE 6th International Conference on Software Security and Reliability Companion, SERE-C 2012*, pages 35–36. IEEE, jun 2012.

[MP15] Ines Coimbra Morgado and Ana C R Paiva. The iMPAcT Tool: Testing UI Patterns on Mobile Applications. pages 876–881, 2015.

[MP16] Inês Coimbra Morgado and Ana C R Paiva. Impact of Execution Modes on Finding Android Failures. In *Procedia Computer Science*, volume 83, pages 284–291, 2016.

# REFERENCES


[MPF] Inês Coimbra Morgado, Ana C R Paiva, and João Pascoal Faria. Automated Pattern-Based Testing of Mobile Applications.

[ND14] Yi-Shuai Niu and Tao Pham Dinh. Advanced Computational Methods for Knowledge Engineering. *Advances in Intelligent Systems and Computing*, 282:37 – 63, 2014.

[Off89] A. Jefferson Offutt. The Coupling Effect: Fact or Fiction. *ACM SIGSOFT Software Engineering Notes*, 14(8):131–140, 1989.

[Off92] A. Jefferson Offutt. Investigations of the software testing coupling effect. *ACM Transactions on Software Engineering and Methodology*, 1(1):5–20, jan 1992.

[OP94] A Jefferson Offutt and Jie Pan. *Using constraints to detect equivalant mutants*. PhD thesis, 1994.

[Ram13] Ravi Ramchandra Nimbalkar. Mobile Application Testing and Challenges. *International Journal of Science and Research*, 2(7):2319–7064, 2013.

[Sta17] Statista. Number of smartphone users worldwide 2014-2020 | Statista. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, 2017.

[Wei] Yuan Wei. *MuDroid: Mutation Testing for Android Apps Undergraduate Final Year Individual Project*. PhD thesis.

[YHJ14] Xiangjuan Yao, Mark Harman, and Yue Jia. A Study of Equivalent and Stubborn Mutation Operators Using Human Analysis of Equivalence. *Proceedings of the 36th International Conference on Software Engineering*, pages 919–930, 2014.

[ZGCU15] Tao Zhang, Jerry Gao, Jing Cheng, and Tadahiro Uehara. Compatibility Testing Service for Mobile Applications. *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on*, (April):179–186, mar 2015.

[ZSG16] Samer Zein, Norsaremah Salleh, and John Grundy. A systematic mapping study of mobile application testing techniques. *Journal of Systems and Software*, 117:334–356, 2016.

# REFERENCES

# Appendix A

# Results Initial Tests

This appendix displays the result of testing the original applications with the *iMPAcT tool*.

## A.1 Results all applications

Table A.1: Intial test results

| **Application** | **Category** | **Downloads** | **Rating** | **Votes** | **Orientation** | **Tab** | **Side Drawer** | **WiFi** |
|---|---|---|---|---|---|---|---|---|
| Lottie | Libraries and Demo | 5 000 – 10 000 | 4.9 | 168 | Correctly implemented | Not Detected | Not Detected | Correctly implemented |
| Material Dialogs Library Demo | Libraries and Demo | 10 000 – 50 000 | 4.8 | 659 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Observable ScrollView demo | Libraries and Demo | 10 000 – 50 000 | 4.7 | 365 | Correctly implemented | Not Detected | Not Detected | Not Detected |
| wallsplash \| wallpaper app | Personalization | 50 000 – 100 000 | 4.6 | 1558 | Correctly implemented | Not Detected | Not Detected | Correctly implemented |
| Material ViewPager | Libraries and Demo | 5 000 – 10 000 | 4.5 | 109 | Correctly implemented | Not Correctly Implemented | Not Detected | Correctly implemented |
| Recycler ViewAnimators | Libraries and Demo | 10 000 – 50 000 | 4.5 | 126 | Not Detected | Not Detected | Not Detected | Not Detected |
| uCrop | Photography | 10 000 – 50 000 | 4.3 | 102 | Not Detected | Not Detected | Not Detected | Not Detected |
| K-9 Mail | Comunication | 5 000 000 – 10 000 000 | 4.2 | 85408 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| Advanced Recycler View | Libraries and Demo | 10 000 – 50 000 | 4.8 | 254 | Correctly implemented | Not Detected | Not Detected | Not Detected |
| williamchart | Libraries and Demo | 5 000 – 10 000 | 4.8 | 192 | Correctly implemented | Not Detected | Not Detected | Not Detected |
| Timber | Music and Audio | 100 000 – 500 000 | 4.3 | 2669 | Correctly implemented | Correctly implemented | Correctly implemented | Correctly implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| Conversations (Jabber / XMPP) | Comunication | 10 000 – 50 000 | 4.5 | 790 | Not Correctly Implemented | Correctly implemented | Not Detected | Correctly implemented |
| ForkHub for GitHub | Productivity | 50 000 – 100 000 | 4.4 | 1121 | Correctly implemented | Correctly implemented | Not Correctly Implemented | Correctly implemented |
| MusicDNA | Music and Audio | 10 000 – 50 000 | 4 | 148 | Correctly implemented | Not Detected | Correctly implemented | Correctly implemented |
| WordPress | Social | 5 000 000 – 10 000 000 | 4.2 | 95299 | Not Correctly Implemented | Correctly implemented | Not Detected | Correctly implemented |
| My Diary (unofficial) | Lifestyle | 50 000 – 100 000 | 4.9 | 2705 | Correctly implemented | Not Detected | Not Detected | Correctly implemented |
| Materialistic - Hacker News With Account | News and Magazines | 10 000 – 50 000 | 4.7 | 1540 | Not Correctly Implemented | Not Correctly Implemented | Not Correctly Implemented | Correctly implemented |
| Loop - Acompanhador de Hábitos | Productivity | 500 000 – 1 000 000 | 4.7 | 7485 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| ConnectBot | Comunication | 1 000 000 – 5 000 000 | 4.6 | 41831 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| Slide for Reddit | News and Magazines | 100 000 – 500 000 | 4.5 | 4857 | Not Correctly Implemented | Not Correctly Implemented | Not Correctly Implemented | Correctly implemented |
| RedReader | News and Magazines | 50 000 – 100 000 | 4.6 | 2052 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Correctly implemented |
| Antox | Comunication | 10 000 – 50 000 | 3.9 | 318 | Not Correctly Implemented | Correctly implemented | Not Correctly Implemented | Correctly implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| ML Manager: APK Extractor | Tools | 10 000 – 50 000 | 4.6 | 942 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| LibreTorrent | Video Players | 10 000 – 50 000 | 4.3 | 401 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| Habitica: Gamify Your Tasks | Productivity | 500 000 – 1 000 000 | 4.3 | 7210 | Correctly implemented | Not Correctly Implemented | Not Detected | Correctly implemented |
| Yaaic - IRC Client | Comunication | 100 000 – 500 000 | 4.1 | 1854 | Not Correctly Implemented | Not Detected | Correctly implemented | Correctly implemented |
| GDG - News & Events | Social | 10 000 – 50 000 | 4.4 | 586 | Not Correctly Implemented | Not Correctly Implemented | Correctly implemented | Correctly implemented |
| Polar Dashboard Sample (Demo) | Libraries and Demo | 1 000 – 5 000 | 4.9 | 106 | Not Correctly Implemented | Not Detected | Correctly implemented | Correctly implemented |
| Etar - Open-Source Calendar | Productivity | 5 000 – 10 000 | 4.6 | 119 | Not Correctly Implemented | Not Detected | Correctly implemented | Not Detected |
| Orgzly: Notes & To-Do Lists | Productivity | 10 000 – 50 000 | 4.6 | 695 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Correctly implemented |
| Simple Calendar | Tools | 100 000 – 500 000 | 4.4 | 658 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Plumble - Mumble VOIP | Comunication | 10 000 – 50 000 | 4.5 | 801 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| OneBusAway | Maps and Navigation | 500 000 – 1 000 000 | 4.3 | 7583 | Not Correctly Implemented | Not Correctly Implemented | Not Detected | Correctly implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| wallabag | Productivity | 10 000 – 50 000 | 4.3 | 331 | Not Correctly Implemented | Not Detected | Correctly implemented | Correctly implemented |
| Bodyweight Fitness Pro | Health and Fitness | 1 000 – 5 000 | 4.9 | 329 | Not Correctly Implemented | Not Correctly Implemented | Not Detected | Correctly implemented |
| Memento - Birthdays & Namedays | Lifestyle | 10 000 – 50 000 | 4.5 | 577 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| 2048 (Ads Free) | Puzzle | 1 000 000 – 5 000 000 | 4.5 | 9666 | Correctly implemented | Not Detected | Not Detected | Not Detected |
| Gobandroid Go Material | Board | 100 000 – 500 000 | 4.4 | 889 | Correctly implemented | Not Detected | Not Correctly Implemented | Not Detected |
| Twittnuker for Twitter | Social | 10 000 – 50 000 | 4 | 1593 | Not Correctly Implemented | Not Detected | Correctly implemented | Correctly implemented |
| Simple Gallery | Tools | 100 000 – 500 000 | 4.4 | 1954 | Correctly implemented | Not Detected | Not Detected | Not Detected |
| Critical Maps | Maps and Navigation | 5 000 – 10 000 | 4.8 | 167 | Correctly Implemented | Not Detected | Not Correctly Implemented | Correctly Implemented |
| Easy xkdc | Comics | 10 000 – 50 000 | 4.8 | 418 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| Equate | Tools | 10 000 – 50 000 | 4.8 | 176 | Not Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |
| FOSDEM Companion | Books and Reference | 5 000 – 10 000 | 4.8 | 204 | Not Correctly Implemented | Correctly Implemented | Correctly Implemented | Correctly Implemented |
| MHGen Database | Books and Reference | 100 000 – 500 000 | 4.8 | 3496 | Correctly Implemented | Correctly Implemented | Not Detected | Not Detected |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| Bewegungs melder | Travel and Local | 10 000 – 50 000 | 4.7 | 115 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| AndBible | Books and Reference | 100 000 – 500 000 | 4.6 | 5333 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| AntennaPod | Video Players | 100 000 – 500 000 | 4.6 | 11931 | Not Correctly Implemented | Not Correctly Implemented | Not Detected | Correctly Implemented |
| CIDR Calculator | Tools | 50 000 – 100 000 | 4.6 | 991 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Clip Stack | Productivity | 100 000 – 500 000 | 4.6 | 4343 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Debatekeeper | Tools | 10 000 – 50 000 | 4.6 | 447 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Simple Draw | Tools | 10 000 – 50 000 | 4.6 | 111 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Fast App Search Tool | Tools | 10 000 – 50 000 | 4.6 | 496 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Hubble Gallery | Education | 50 000 – 100 000 | 4.6 | 852 | Correctly Implemented | Not Detected | Not Correctly Implemented | Not Detected |
| AnkiDroid | Education | 1 000 000 – 5 000 000 | 4.5 | 28789 | Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |
| Chroma Doze | Music and Audio | 50 000 – 100 000 | 4.5 | 812 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Clementine Remote | Music and Audio | 100 000 – 500 000 | 4.5 | 2924 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| CPU Stats | Tools | 100 000 – 500 000 | 4.5 | 3298 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| DO Swimmer | Productivity | 10 000 – 50 000 | 4.5 | 1084 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Ametro | Maps and Navigation | 100 000 – 500 000 | 4.4 | 3058 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| Bubble | Comics | 10 000 – 50 000 | 4.4 | 278 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Not Detected |
| Calendar Notifications | Productivity | 10 000 – 50 000 | 4.4 | 159 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Simple Flashlight | Tools | 5 000 – 10 000 | 4.4 | 186 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| flym | News and Magazines | 10 000 – 50 000 | 4.4 | 1116 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| French Calendar | Personalization | 5 000 – 10 000 | 4.4 | 117 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Hacker News | News and Magazines | 50 000 – 100 000 | 4.4 | 1346 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| LifeCounter | Tools | 10 000 – 50 000 | 4.4 | 539 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| MIFARE Classic Tool | Tools | 100 000 – 500 000 | 4.4 | 610 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Minimal | Productivity | 10 000 – 50 000 | 4.4 | 698 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| Anuto TD | Game Strategy | 5 000 – 10 000 | 4.3 | 125 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| ARChon Packager | Tools | 100 000 – 500 000 | 4.3 | 1608 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Blokish | Puzzle | 100 000 – 500 000 | 4.3 | 2037 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Droir Beard | Video Players | 10 000 – 50 000 | 4.3 | 202 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| Simple File Manager | Tools | 10 000 – 50 000 | 4.3 | 225 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Glucosio | Medical | 5 000 – 10 000 | 4.3 | 157 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| iFixit | Books and Reference | 1 000 000 – 5 000 000 | 4.3 | 9036 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Correctly Implemented |
| 2048 | Puzzle | 1 000 000 – 5 000 000 | 4.2 | 55197 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| AndStatus | Social | 5 000 – 10 000 | 4.2 | 107 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Cadroid | Tools | 10 000 – 50 000 | 4.2 | 184 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Commons | Photography | 5 000 – 10 000 | 4.2 | 120 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| Lightning | Comunication | 500 000 – 1 000 000 | 4.2 | 11634 | Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| AcDisplay | Personalization | 1 000 000 – 5 000 000 | 4.1 | 63502 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| BankDroid | Finance | 100 000 – 500 000 | 4.1 | 5062 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| Simple Camera | Tools | 10 000 – 50 000 | 4.1 | 280 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Chess | Puzzle | 100 000 – 500 000 | 4.1 | 3623 | Not Correctly Implemented | Not Detected | Not Detected | Correctly implemented |
| File Manager | Tools | 5 000 – 10 000 | 4.1 | 143 | Not Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |
| Gapps Browser | Comunication | 10 000 – 50 000 | 4.1 | 286 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| GPS Logger | Travel and Local | 500 000 – 1 000 000 | 4.1 | 3941 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| iBeacon Detetor | Tools | 10 000 – 50 000 | 4.1 | 135 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Logical Defence | Education | 50 000 – 100 000 | 4.1 | 447 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Lucid Browser | Comunication | 10 000 – 50 000 | 4.1 | 688 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Alldebrid Android | Tools | 10 000 – 50 000 | 3.8 | 191 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| BatteryFu | Tools | 100 000 – 500 000 | 3.8 | 1610 | Correctly Implemented | Not Detected | Not Detected | Not Detected |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| Blue Mono Sound | Tools | 100 000 – 500 000 | 3.7 | 422 | Not Detected | Not Detected | Not Detected | Not Detected |
| CycleStreets | Travel and Local | 50 000 – 100 000 | 3.7 | 490 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Hex | Puzzle | 50 000 – 100 000 | 3.7 | 1636 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Hosts Editor | Tools | 5 000 000 – 10 000 000 | 3.7 | 48437 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Kaleidoscope | Entertainment | 10 000 – 50 000 | 3.7 | 289 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Missed Notifications Reminder | Tools | 10 000 – 50 000 | 4.1 | 143 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Mongo Explorer | Tools | 10 000 – 50 000 | 4.2 | 262 | Correctly Implemented | Not Detected | Not Correctly Implemented | Correctly Implemented |
| Movian Remote | Video Players | 1 000 – 5 000 | 4.5 | 111 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| MTG Familiar | Tools | 500 000 – 1 000 000 | 4.6 | 10932 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Multitouch Test | Tools | 10 000 – 50 000 | 4.3 | 147 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Music Player | Tools | 10 000 – 50 000 | 4.3 | 272 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| My Expenses | Finance | 500 000 – 1 000 000 | 4.4 | 6339 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Correctly Implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| MyHackerSpace | Tools | 10 000 – 50 000 | 4.3 | 78 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| NetGuard | Tools | 100 000 – 500 000 | 4.2 | 4634 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Network Monitor | Tools | 50 000 – 100 000 | 4.3 | 432 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| NoNonsense Notes | Productivity | 100 000 – 500 000 | 4.4 | 1797 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Note Crypt | Productivity | 5 000 – 10 000 | 4.4 | 191 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Notes | Tools | 10 000 – 50 000 | 4.5 | 180 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Notify | Productivity | 5 000 – 10 000 | 4.1 | 112 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Nounours and Friends | Entertainment | 10 000 – 50 000 | 3.8 | 371 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Numix Calculator | Tools | 50 000 – 100 000 | 4.1 | 1972 | Not Correctly Implemented | Not Correctly Implemented | Not Detected | Correctly Implemented |
| OctoDroid | Productivity | 50 000 – 100 000 | 4.5 | 1498 | Not Correctly Implemented | Correctly Implemented | Correctly Implemented | Correctly Implemented |
| OI File Manager | Productivity | 5 000 000 – 10 000 000 | 4.2 | 52469 | Not Correctly Implemented | Correctly Implemented | Not Detected | Not Detected |
| Olam | Books and Reference | 100 000 – 500 000 | 4.3 | 3234 | Correctly Implemented | Not Detected | Not Detected | Not Detected |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| Open Link With | Tools | 10 000 – 50 000 | 4.5 | 461 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| OpenTasks | Productivity | 100 000 – 500 000 | 4.2 | 1439 | Not Correctly Implemented | Correctly Implemented | Not Detected | Not Detected |
| OSM tracker | Travel and Local | 100 000 – 500 000 | 4.3 | 1251 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| OSRS Helper | Tools | 10 000 – 50 000 | 4.5 | 188 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| ownCloud News | News and Magazines | 1 000 – 5 000 | 4.5 | 177 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| PassAndroid | Travel and Local | 1 000 000 – 5 000 000 | 4.1 | 4885 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Correctly Implemented |
| PasswordStore | Productivity | 10 000 – 50 000 | 4.5 | 240 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| pMetro | Travel and Local | 10 000 – 50 000 | 4.2 | 164 | Not Correctly Implemented | Not Correctly Implemented | Not Detected | Correctly Implemented |
| Poet Assistant | Books and Reference | 50 000 – 100 000 | 4.6 | 472 | Not Correctly Implemented | Correctly Implemented | Not Detected | Not Detected |
| Port Authority | Tools | 50 000 – 100 000 | 4.3 | 306 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Port Knocker | Tools | 10 000 – 50 000 | 4.6 | 139 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| pOT-Droid | Social | 1 000 – 5 000 | 5 | 166 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Correctly Implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| primitive ftpd | Tools | 5 000 – 10 000 | 4.4 | 160 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Privacy Week Schedule | Books and Reference | 10 000 – 50 000 | 4.8 | 317 | Not Correctly Implemented | Not Correctly Implemented | Not Detected | Correctly Implemented |
| qBittorrent Client Pro | Tools | 100 000 – 500 000 | 3.9 | 811 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| QR Scanner | Tools | 10 000 – 50 000 | 4.4 | 101 | Correctly Implemented | Not Detected | Correctly Implemented | Not Detected |
| Quick Dice Roller | Tools | 100 000 – 500 000 | 4.2 | 1268 | Not Correctly Implemented | Not Correctly Implemented | Not Detected | Not Detected |
| Quick Lyrics | Music and Audio | 100 000 – 500 000 | 4.2 | 3014 | Not Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |
| RasPi Check | Tools | 100 000 – 500 000 | 4.5 | 904 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Recurrence | Productivity | 10 000 – 50 000 | 4.5 | 731 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Reddinator | News and Magazines | 50 000 – 100 000 | 4.3 | 476 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| RF Analyzer | Tools | 10 000 – 50 000 | 4.3 | 413 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Correctly Implemented |
| RGB Tool | Tools | 10 000 – 50 000 | 4.3 | 186 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Riot | Comunication | 10 000 – 50 000 | 4.6 | 322 | Not Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| S Tools + | Tools | 10 000 – 50 000 | 4.2 | 599 | Correctly Implemented | Not Detected | Not Correctly Implemented | Not Detected |
| SealNote | Productivity | 50 000 – 100 000 | 4.5 | 2049 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Not Detected |
| Secret Codes | Tools | 100 000 – 500 000 | 4.1 | 2092 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Shader Editor | Tools | 10 000 – 50 000 | 4.7 | 378 | Not Correctly Implemented | Not Detected | Not Correctly Implemented | Not Detected |
| Simple Last fm Scrobbler | Music and Audio | 500 000 – 1 000 000 | 4.2 | 14896 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Slide | Productivity | 1 000 – 5 000 | 4.7 | 110 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| SMSsync | Tools | 10 000 – 50 000 | 4.1 | 159 | Not Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |
| Taskbar | Tools | 100 000 – 500 000 | 4.4 | 1541 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| TimerDroid | Tools | 10 000 – 50 000 | 3.9 | 215 | Not Correctly Implemented | Not Correctly Implemented | Not Detected | Not Detected |
| Toffed | Social | 10 000 – 50 000 | 4.1 | 860 | Not Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |
| Traccar Client | Tools | 50 000 – 100 000 | 4.5 | 416 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Tradutor Softcatala | Tools | 100 000 – 500 000 | 4.3 | 744 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| Tricky Tipper | Travel and Local | 5 000 – 10 000 | 4.6 | 125 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| TV Kill | Tools | 10 000 – 50 000 | 4 | 100 | Correctly Implemented | Not Correctly Implemented | Not Detected | Not Detected |
| Unit Converter Ultimate | Tools | 500 000 – 1 000 000 | 4.5 | 12189 | Not Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |
| Vlille Checker | Maps and Navigation | 5 000 – 10 000 | 4.7 | 111 | Correctly Implemented | Not Correctly Implemented | Not Detected | Correctly Implemented |
| WaveUp | Tools | 100 000 – 500 000 | 4.5 | 1562 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Web Opac | Education | 50 000 – 100 000 | 4.3 | 613 | Not Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |
| Weechat Android | Comunication | 50 000 – 100 000 | 4.3 | 718 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Wi-FI Privacy Police | Tools | 50 000 – 100 000 | 4.2 | 1180 | Not Correctly Implemented | Not Detected | Not Detected | Not Detected |
| WiFi Automatic | Tools | 1 000 000 – 5 000 000 | 4.1 | 9009 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| Wifi Walkie Talkie | Comunication | 50 000 – 100 000 | 3.9 | 270 | Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |
| WiFi Analyzer | Tools | 1 000 000 – 5 000 000 | 4.4 | 4208 | Not Correctly Implemented | Not Detected | Correctly Implemented | Correctly Implemented |
| Wigle Wifi Wardriving FOSS | Tools | 500 000 – 1 000 000 | 4.2 | 3021 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |

Table A.1 (continued)

| Application | Category | Downloads | Rating | Votes | Orientation | Tab | Side Drawer | WiFi |
|---|---|---|---|---|---|---|---|---|
| WordPower Made Easy | Education | 10 000 – 50 000 | 4 | 254 | Correctly Implemented | Not Detected | Not Detected | Not Detected |
| Writeily Pro | Productivity | 5 000 – 10 000 | 4.3 | 118 | Not Correctly Implemented | Not Detected | Not Detected | Correctly Implemented |

## A.2 Results rotation allowed

In Table A.2 distinguishes the applications that have the orientation pattern correctly implemented and that allow rotation from the applications that have the orientation pattern correctly implemented but do not allow rotation. If the application does not allow rotation then it will not have any of the problems associated with the changing of orientation because they only allow or the portrait or the landscape mode.

Table A.2: Applications with orientation pattern correctly implemented: rotation allowed or not

| **Allows rotation** | **Does not allow rotation** |
|---|---|
| ObservableScrollView demo | Lottie |
| wallsplash \| wallpaper app | williamchart |
| MaterialViewPager | MusicDNA |
| Advanced RecyclerView | My Diary(unofficial) |
| williamchart | Habitica: Gamify Your Tasks |
| Timber | 2048 (Ads Free) |
| ForkHub for GitHub | Gobandroid Go Material |
| Simple Gallery | Critical Maps |
| Bewegungsmelder | MHGen Database |
| CIDR Calculator | Simple Flashlight |
| Hubble Gallery | ARChon Packager |
| AnkiDroid | 2048 |
| Chroma Doze | Simple Camera |
| Calendar Notifications | Hex |
| Simple Flashlight | Mongo Explorer |
| Hacker News | Movian Remote |
| MIFARE Classic Tool | Note Crypt |
| Anuto TD | Olam |
| Blokish | RGB Tool |
| Simple File Manager | TV Kill |
| AndStatus | Wifi Walkie Talkie |
| Cadroid | |
| Lightning | |
| iBeacon Detetor | |

Table A.2 (continued)

| Allows rotation | Does not allow rotation |
|---|---|
| Lucid Browser | |
| BatteryFu | |
| Kaleidoscope | |
| Mongo Explorer | |
| Multitouch Test | |
| NetGuard | |
| Network Monitor | |
| NoNonsense Notes | |
| Notes | |
| Open Link With | |
| QR Scanner | |
| RasPi Check | |
| Recurrence | |
| Reddinator | |
| S Tools + | |
| Secret Codes | |
| Traccar Client | |
| Vlille Checker | |
| Weechat Android | |
| WiFiAnalyzer | |
| WordPower Made Easy | |

# Appendix B

# Results Mutants

## B.1 Orientation

Table B.1 displays the reason for the exclusion from the case study of each application that had the orientation pattern correctly implemented and that allowed the screen rotation. Figure B.1 presents an example of the interface of a game application (*Blokish*), which made it impossible to use the application in the case study.

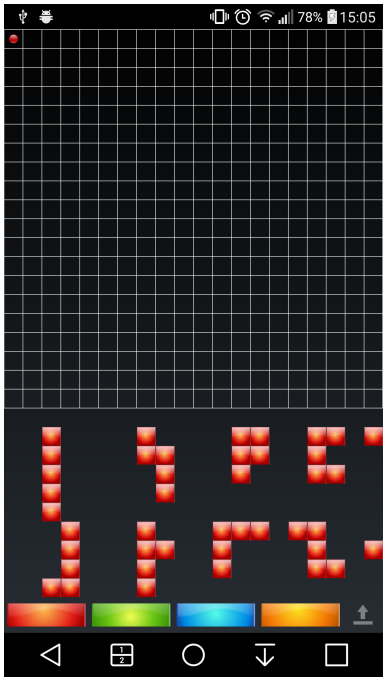


Figure B.1: Example of a game interface, *Blokish* application


Figure B.2: Example of a simple interface, *Simple Flashlight* application

The mutant 1 was only inserted into three applications. Each application was tested three

Table B.1: Orientation Pattern: Excluded applications

| Application | Reason for exclusion |
|---|---|
| ObservableScrollView demo | There is nothing to test in this application |
| wallsplash \| wallpaper app | There is nothing to test in this application |
| MaterialViewPager | There is nothing to test in this application |
| Advanced RecyclerView | There is nothing to test in this application |
| williamchart | There is nothing to test in this application |
| Timber | There is nothing to test in this application. Has an error in the search widget, that is not detected by the iMPAcT tool |
| Simple Gallery | Uses kotlin |
| Hubble Gallery | There is nothing to test in this application |
| Chroma Doze | There is nothing to test in this application |
| Calendar Notifications | Uses kotlin |
| Simple Flashlight | There is nothing to test in this application |
| Hacker News | There is nothing to test in this application |
| Anuto TD | It is a game, only one screen. Nothing to test |
| Blokish | It is a game, only one screen. Nothing to test |
| Simple File Manager | There is nothing to test in this application |
| AndStatus | There is nothing to test in this application |
| iBeacon Detetor | There is nothing to test in this application |
| Lucid Browser | Screen that contains information in which the mutants could be inserted, is not reachable because the settings menu is hidden |
| BatteryFu | There is nothing to test in this application |
| Kaleidoscope | There is nothing to test in this application |
| Mongo Explorer | There is nothing to test in this application, needed connection to mongo server to explore further |
| Multitouch Test | There is nothing to test in this application |
| Notes | Uses kotlin |
| Open Link With | There is nothing to test in this application |
| QR Scanner | There is nothing to test in this application |
| S Tools + | There is nothing to test in this application |
| Secret Codes | There is nothing to test in this application |
| Traccar Client | It is an hidden application |
| Vlille Checker | Application contained an error in the settings menu, that was not previously detected by the *iMPAcT tool* |
| WiFiAnalyzer | There is nothing to test in this application |
| WordPower Made Easy | There is nothing to test in this application |

times, using two different exploration modes of the *iMPAcT tool* ( "priority to not executed" and "all events"). The errors were detected in every test as is shown in Table B.2.

Table B.2: Orientation Mutant 1: extended results

| | **Mutant 1** | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| AnkiDroid | YES | YES | YES |
| Network Monitor | YES | YES | YES |
| NoNonsense Notes | YES | YES | YES |

The mutant 2 was only inserted into ten applications. Each application was tested three times, using two different exploration modes of the *iMPAcT tool* ( "priority to not executed" and "all events"). The *iMPAcT tool* was not able to detect the error in the *Network Monitor* application, because it never reached the screen in which the mutant was inserted. Even when using the exploration mode of "all events", the error was not detected.

Table B.3: Orientation Mutant 2: extended results

| | **Mutant 2** | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| ForkHub for GitHub | YES | YES | YES |
| CIDR Calculator | YES | YES | YES |
| AnkiDroid | YES | YES | YES |
| MIFARE Classic Tool | YES | YES | YES |
| Cadroid | YES | YES | YES |
| Lightning | YES | YES | YES |
| Network Monitor | NO [3] | NO [3] | NO [3] |
| NoNonsense Notes | YES | YES | YES |
| RasPi Check | YES | YES | YES |
| Weechat Android | YES | YES | YES |

The mutant 3 was only inserted into two applications. Each application was tested three times, using two different exploration modes of the *iMPAcT tool* ( "priority to not executed" and "all events"). The errors were detected in every test as is shown in Table B.4.

Table B.4: Orientation Mutant 3: extended results

| | Mutant 3 | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| CIDR Calculator | YES | YES | YES |
| RasPi Check | YES | YES | YES |

The mutant 4 was inserted into eight applications. Each application was tested three times, using two different exploration modes of the *iMPAcT tool* ( "priority to not executed" and "all events"). The *iMPAcT tool* was not able to detect all the errors, in some applications the tool detected the disappearance of popups but it did not detect any of the errors that were inserted in the search widget. This happened in two applications: *Lightning* and *NetGuard*. In the *ForkHub* application there was no errors related with popup only with the search widget and the *iMPAcT tool* did not detect them.

Table B.5: Orientation Mutant 4: extended results

| | Mutant 4 | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| ForkHub for GitHub | NO [1] | NO [1] | NO [1] |
| AnkiDroid | YES | YES | YES |
| Bewegungsmelder | YES | YES | YES |
| MIFARE Classic Tool | YES | YES | YES |
| Lightning | NO[1]/YES [2] | NO[1]/YES [2] | NO[1]/YES [2] |
| NetGuard | NO[1]/YES [2] | NO[1]/YES [2] | NO[1]/YES [2] |
| Recurrence | YES | YES | YES |
| Reddinator | YES | YES | YES |

## B.2 Tab

Table B.9 shows the results of testing each application after the insertion of the mutants. In all the applications, with the exception of the *Conversations* application, the iMPAcT tool was able to detect the insertion of the mutants. When testing the *Conversations* application the *iMPAcT tool* had difficulties in detecting the error, only detected the error in one test. This happened because the *iMPAcT tool* did not reach the screen with the mutation in most tests, making it impossible to detect the error. In the test that it did reach the desired screen, the *iMPAcT tool* was able to detect the error.

Table B.6: Tab Pattern: Mutants results

| | Tests | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| Timber | Yes | Yes | Yes |
| Conversations (Jabber / XMPP) | No | Yes | No |
| WordPress | Yes | Yes | Yes |
| FOSDEM Companion | Yes | Yes | Yes |
| MHGen Database | Yes | Yes | Yes |
| OctoDroid | Yes | Yes | Yes |
| Open Tasks | Yes | Yes | Yes |
| Poet Assistant | Yes | Yes | Yes |
| Forkhub | Yes | Yes | Yes |

Since the*iMPAcT tool* was having difficulties in detecting the mutation operator inserted in the application *Conversations*, more tests were executed against this application, and this time the four possible configurations were used at least once. Only one test was able to detect the error, the others did not reach the desired screen.

Table B.7: *Conversation* application: Mutants extended results

| **Configuration used** | **Error detected** |
|---|---|
| "priority to not executed and list items" | No |
| "priority to not executed" | No |
| "execute once" | No |
| "all events" | No |
| "priority to not executed" | No |
| "execute once" | No |
| "all events" | No |
| "priority to not executed" | No |
| "execute once" | No |
| "all events" | Yes |

## B.3 Side Drawer

The mutant 1 was applied to twenty-one applications. Each application was tested three times using two different exploration modes of the *iMPAcT tool* ( "priority to not executed" and "all

events"). The errors were detected in every test as is shown in Table B.8.

Table B.8: Side Drawer Mutant 1: extended results

| | Mutant 1 | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| Timber | YES | YES | YES |
| MusicDNA | YES | YES | YES |
| Yaaic - IRC Client | YES | YES | YES |
| GDG - News and Events | YES | YES | YES |
| Polar Dashboard Sample | YES | YES | YES |
| Etar - OpenSource Calendar | YES | YES | YES |
| wallabag | YES | YES | YES |
| Twittnuker for Twitter | YES | YES | YES |
| Equate | YES | YES | YES |
| FOSDEM Companion | YES | YES | YES |
| AnkiDroid | YES | YES | YES |
| Lightning | YES | YES | YES |
| OctoDroid | YES | YES | YES |
| QR Scanner | YES | YES | YES |
| Quick Lyrics | YES | YES | YES |
| Riot | YES | YES | YES |
| SMSsync | YES | YES | YES |
| Toffed | YES | YES | YES |
| Unit Converter Ultimate | YES | YES | YES |
| Web Opac | YES | YES | YES |
| WiFiAnalyzer | YES | YES | YES |

The mutant 2b was only applied to seventeen applications. Each application was tested three times using two different exploration modes of the *iMPAcT tool* ( "priority to not executed" and "all events"). The errors were detected in every test as is shown in Table B.9.

Table B.9: Side Drawer Mutant 2b: extended results

| | Mutant 2b | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| Timber | YES | YES | YES |
| MusicDNA | YES | YES | YES |
| GDG - News and Events | YES | YES | YES |
| Polar Dashboard Sample | YES | YES | YES |
| Etar - OpenSource Calendar | YES | YES | YES |
| wallabag | YES | YES | YES |
| Twittnuker for Twitter | YES | YES | YES |
| Equate | YES | YES | YES |
| AnkiDroid | YES | YES | YES |
| OctoDroid | YES | YES | YES |
| QR Scanner | YES | YES | YES |
| Riot | YES | YES | YES |
| SMSsync | YES | YES | YES |
| Toffed | YES | YES | YES |
| Unit Converter Ultimate | YES | YES | YES |
| Web Opac | YES | YES | YES |
| WiFiAnalyzer | YES | YES | YES |

The mutant 2a was only applied to seventeen applications. Each application was tested three times using two different exploration modes of the *iMPAcT tool* ( "priority to not executed" and "all events"). The errors were detected in every test as is shown in Table B.10.

Table B.10: Side Drawer Mutant 2a: extended results

| | Mutant 2a | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| Timber | YES | YES | YES |
| MusicDNA | YES | YES | YES |
| GDG - News and Events | YES | YES | YES |
| Polar Dashboard Sample | YES | YES | YES |
| Etar - OpenSource Calendar | YES | YES | YES |
| wallabag | YES | YES | YES |
| Twittnuker for Twitter | YES | YES | YES |
| Equate | YES | YES | YES |
| AnkiDroid | YES | YES | YES |
| OctoDroid | YES | YES | YES |
| QR Scanner | YES | YES | YES |
| Riot | YES | YES | YES |
| SMSsync | YES | YES | YES |
| Toffed | YES | YES | YES |
| Unit Converter Ultimate | YES | YES | YES |
| Web Opac | YES | YES | YES |
| WiFiAnalyzer | YES | YES | YES |

The mutant 3b was only applied to four applications. Each application was tested three times using two different exploration modes of the *iMPAcT tool* ( "priority to not executed" and "all events"). The errors were detected in every test as is shown in Table B.11.

Table B.11: Side Drawer Mutant 3b: extended results

| | Mutant 3b | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| Yaaic - IRC Client | YES | YES | YES |
| FOSDEM Companion | YES | YES | YES |
| Lightning | YES | YES | YES |
| Quick Lyrics | YES | YES | YES |

The mutant 3a was only applied to four applications. Each application was tested three times using two different exploration modes of the *iMPAcT tool* ( "priority to not executed" and "all

events"). The errors were detected in every test as is shown in Table B.12.

Table B.12: Side Drawer Mutant 3a: extended results

| | Mutant 3a | | |
|---|---|---|---|
| **Application** | **"priority to not executed"** | **"all events"** | **"all events"** |
| Yaaic - IRC Client | YES | YES | YES |
| FOSDEM Companion | YES | YES | YES |
| Lightning | YES | YES | YES |
| Quick Lyrics | YES | YES | YES |