PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

BENCHMARK PARA APOIAR EXPERIMENTOS COM TESTES FUNCIONAIS DE APLICAÇÕES MÓVEIS

LARISSA LORENA EVANGELISTA DAS NEVES

Orientador: Arilo Claudio Dias Neto, DSc.

Manaus
2019

LARISSA LORENA EVANGELISTA DAS NEVES

BENCHMARK PARA APOIAR EXPERIMENTOS COM TESTES FUNCIONAIS DE APLICAÇÕES MÓVEIS

Proposta de Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, PPGI, da Universidade Federal do Amazonas como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Arilo Claudio Dias Neto, DSc.

Manaus
2019

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Neves, Larissa Lorena Evangelista das
N518b Benchmark para apoiar experimentos com testes funcionais de aplicações móveis / Larissa Lorena Evangelista das Neves. 2019
82 f.: il. color; 31 cm.

Orientador: Arilo Claudio Dias Neto
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Benchmark. 2. Aplicações Móveis. 3. Teste de Aplicações Móveis. 4. Teste Funcional. 5. Engenharia de Software Experimental. I. Dias Neto, Arilo Claudio II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Benchmark para Apoiar Experimentos com Testes Funcionais de Aplicações Móveis"

LARISSA LORENA EVANGELISTA DAS NEVES

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Arilo Claudio Dias Neto

Prof. Arilo Cláudio Dias Neto - PRESIDENTE

Eduardo Luzeiro Feitosa

Prof. Eduardo Luzeiro Feitosa - MEMBRO INTERNO

Prof. André Takeshi Endo - MEMBRO EXTERNO

Manaus, 19 de Setembro de 2019

Aos meus pais, meu irmão e ao meu namorado pelo carinho e apoio em todos os momentos.

Agradecimentos

À Deus por me dar saúde e ser meu guia nesse caminho, por me ensinar a ser perseverante, sem Ele nada teria sido possível.

Aos meus pais Edson Cabral e Linda Maria, por nunca ter desistido de mim, por me motivarem e por nunca medirem esforços em me ajudar ao longo de toda a minha trajetória acadêmica.

Ao meu irmão Edson Junior, pela companhia e apoio.

Ao meu namorado Guibson Souza, pelo amor, cumplicidade, dedicação, incentivo. Certamente sem seu apoio eu não teria conseguido. Obrigado por não ter me deixado desistir, por ser meu suporte, amigo e companheiro em todo momento.

Ao meu orientador professor Arilo, pelos ensinamentos, incentivos e por suas valiosas orientações no decorrer da minha vida acadêmica.

A todos os membros do grupo ExperTS (Experimentação e Teste de Software) pela amizade, contribuições, pela troca de ideias e experiências, pela cooperação e por tornarem nosso laboratório tão amistoso e divertido.

Aos meus amigos e colegas do PPGI pelo apoio, grupos de estudos, trocas de experiencias e momentos de descontração. A caminhada se tornou mais leve com a companhia de vocês.

A todos os amigos (que também chamava carinhosamente de coorientadores) que contribuíram de certa forma no meu mestrado, sendo com orientações, revisões do texto, debates sobre a pesquisa, indicando uma solução quando todas minhas ideias já se esgotaram ou um ombro amigo quando eu achava estar tudo perdido... certamente eu estaria perdida sem a ajuda de vocês.

Aos amigos externos a UFAM que contribuíram com os momentos de descontração com as saídas, noites de jogos, os jogos de handebol, treinos e principalmente muita risada.

Aos professores membros da minha banca, Eduardo Feitosa e André Endo, por terem aceito fazer parte e contribuir para a conclusão dessa etapa.

Aos professores e ao corpo administrativo do PPGI pelo suporte e acompanhamento durante esta fase da minha formação acadêmica.

À CAPES pelo apoio financeiro para a execução desta pesquisa.

Resumo da Dissertação apresentada à UFAM/AM como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática (M.Sc.)

BENCHMARK PARA APOIAR EXPERIMENTOS COM TESTES FUNCIONAIS DE APLICAÇÕES MÓVEIS

Larissa Lorena Evangelista das Neves

Setembro de 2019

Orientador: Arilo Claudio Dias Neto

Com o aumento da importância das aplicações móveis no cotidiano das pessoas, são necessárias novas abordagens de Engenharia de Software para garantir a qualidade de tais aplicações antes de sua disponibilização ao usuário final, e isso inclui novas técnicas de teste de software. A pesquisa em teste de aplicações móveis precisa de diretrizes para melhorar os processos de validação, por meio de experimentos das técnicas propostas como resultados de pesquisas científicas. No entanto, atualmente, não existe um *benchmark* detalhado e abrangente de aplicações móveis de referência para avaliar experimentalmente técnicas de teste para esta categoria de aplicações. Os estudos encontrados mostram a falta de informações mais detalhadas das aplicações. Com isso, a proposta deste trabalho é a construção de um *benchmark* de auxílio para pesquisador avaliar experimentalmente técnicas de teste em aplicações móveis composto por conjunto de aplicações móveis, *scripts* de teste e métricas referentes às aplicações. Como resultado, foi utilizado um processo de seleção de aplicações móveis para o *benchmark*. Após seleção do conjunto inicial de aplicações que compõe o benchmark, foram realizados dois estudos como forma de investigar se o benchmark proposto auxilia os pesquisadores em experimentos de técnicas de teste de aplicações móveis. As aplicações foram selecionadas de acordo com suas características e as necessidades do experimento. Com isso, em ambos estudos foram utilizados 67% das aplicações do benchmark.

Palavras-chave: *Benchmark*, Aplicações Móveis, Engenharia de Software Experimental, Teste de Aplicações Móveis, Teste Funcional.

Abstract of Thesis presented to UFAM/AM as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

BENCHMARK TO SUPPORT EXPERIMENTS WITH FUNCTIONAL TEST FOR MOBILE APPLICATIONS

Larissa Lorena Evangelista das Neves

September 2019

Advisor: Arilo Claudio Dias Neto

With the increasing importance of mobile applications in people's daily lives, new software engineering approaches are required to ensure the quality of such mobile applications before their publication in application stores, and this includes new software testing techniques. Research on mobile application testing needs guidelines to improve the validation processes, through experiments, of techniques proposed as results of scientific research. However, there is currently no detailed and comprehensive benchmark of applications for evaluating experimental testing techniques for mobile applications. The studies found present the lack of more detailed information of the applications. Thus, this work proposes a benchmark of support to researchers which experimentally evaluates test techniques for mobile applications. This benchmark is composed of set of mobile applications, test scripts and metrics for applications. As a result of the definition, selection process was obtained for mobile application benchmark. After selection the initial benchmark, two studies were conducted to investigate if the proposed benchmark provides support to researchers to experimentally evaluate test techniques for mobile applications. The applications were selected according to their characteristics and the needs of experiment. Thus, in both studies, 67% of benchmark applications were used.

***Keywords**:* Benchmark, Mobile Applications, Experimental Software Engineering, Mobile Application Testing, Functional Testing

ÍNDICE

LISTA DE FIGURAS xi

LISTA DE TABELAS xii

1. INTRODUÇÃO 1

1.1 Contextualização e Motivação 1

1.2 Descrição do Problema 2

1.3 Hipótese 3

1.4 Objetivos 3

1.4.1 Objetivo Geral 3

1.4.2 Objetivos Específicos 3

1.5 Metodologia 4

1.6 Estrutura do Documento 5

2. REFERENCIAL TEÓRICO 6

2.1 Teste em Aplicações Móveis 6

2.2 Engenharia de Software Experimental 7

2.3 Experimentos em Teste de Aplicações Móveis 9

2.3.1 Critério de Escolha das Aplicações 11

2.3.2 Origem das Aplicações 12

2.3.3 Quantidade de Aplicações 12

2.3.4 Métricas de Avaliação 13

2.3.5 Considerações Sobre Experimentos em Teste de Aplicações Móveis 14

2.4 Visão Geral Sobre Benchmark 14

2.5 Benchmark com Aplicações Móveis 15

2.6 Considerações Finais 18

3. *BENCHMARK* DE APLICAÇÕES MÓVEIS PARA EXPERIMENTAÇÃO DE TÉCNICAS DE TESTE 19

3.1 Componentes do Benchmark 19

3.2 Seleção das Aplicações para Compor o Benchmark 20

3.2.1 Realização dos Filtros para a Seleção das Aplicações 22

3.2.2 Execução dos Scripts de Teste 23

3.3 Benchmark Composto 28

3.4 Classificação das Aplicações 30

3.5 Considerações Finais 32
4. ESTUDOS DE AVALIAÇÃO DO *BENCHMARK* DE APLICAÇÕES MÓVEIS 33
4.1 Avaliação do Benchmark de Aplicações Móveis 33
4.1.1 Auxílio no Planejamento do Experimento 33
4.1.2 Auxílio na Execução do Experimento 38
4.2 Ameaças à Validade 41
4.3 Considerações Finais 41
5. CONCLUSÕES 43
5.1 Considerações Finais 43
5.2 Contribuições 43
5.3 Limitações 44
5.4 Trabalhos Futuros 44
REFERÊNCIAS 45
APÊNDICE A - EXPERIMENTOS SOBRE TESTE FUNCIONAL UTILIZANDO APLICAÇÕES MÓVEIS RETIRADOS DE ZEIN *ET AL.* (2016) 49
APÊNDICE B - BENCHMARKS UTILIZANDO APLICAÇÕES MÓVEIS 51
APÊNDICE C – INFORMAÇÕES OBTIDAS DE CADA APLICAÇÃO 53

LISTA DE FIGURAS

Figura 1. Metodologia aplicada na Pesquisa. 4
Figura 2. Processo de Experimentação (Wohlin et al., 2012). 8
Figura 3. Quantidade de trabalhos de acordo com os critérios de escolha das aplicações. 12
Figura 4. Quantidade de trabalhos de acordo com as origens das aplicações. 12
Figura 5. Quantidade de trabalhos de acordo com as quantidades de aplicações utilizadas. 13
Figura 6. Quantidade de trabalhos de acordo com as métricas de avaliação utilizadas no experimento. 13
Figura 7. Benchmark. 19
Figura 8. Processo de seleção das aplicações para compor o benchmark. 21
Figura 9. Filtro dos projetos que apresentam alguma evidencia de testes automatizados. 23
Figura 10. Visualização do relatório de teste gerado. 25
Figura 11. Visualização do relatório de cobertura gerado. 26
Figura 12. Resultado da utilização do processo de seleção das aplicações para compor o benchmark. 28
Figura 13. Interface da ferramenta Thor com as aplicações do benchmark inseridas. 39

LISTA DE TABELAS

Tabela 1. Experimentos sobre teste funcional utilizando aplicações móveis retiradas de (Zein et al., 2016). 10
Tabela 2. Benchmarks de aplicações móveis. 15
Tabela 3. Trabalhos apresentados por Silva (2017) que investigam conjuntos de aplicações open-source. 16
Tabela 4. Projetos que apresentam alguma evidência de testes automatizados. . 24
Tabela 5. Quantidade de testes e falhas de cada projeto informado pelo relatório. 25
Tabela 6. Informações do relatório de cobertura de cada projeto. 27
Tabela 7. Relatórios gerados de cada projeto. 27
Tabela 8. Métricas que compõem o benchmark. 29
Tabela 9. Informações do projeto “de.danoeh.antennapod”. 30
Tabela 10. Ranking de acordo com a quantidade de testes existentes na aplicação. 31
Tabela 11. Ranking de acordo com o percentual de cobertura de código. 31
Tabela 12. Lista de permissão com sua descrição e fontes relacionadas retirada de Usman et al. (2018). 34
Tabela 13. Características das aplicações móveis testadas no trabalho de Usman et al. (2018) com os dados atualizados utilizando o Prof.MApp e retirados da Google Play. 35
Tabela 14. Recursos de conectividade dos projetos utilizados no trabalho de Usman et al. (2018). 35
Tabela 15. Sensores dos projetos utilizados no trabalho de Usman et al. (2018). 36
Tabela 16. Recursos de conectividade e sensores dos projetos presentes no benchmark. 37
Tabela 17. Características das aplicações móveis testadas no trabalho de Adamsen et al. (2015). 38
Tabela 18. Execução da ferramenta Thor com as aplicações do benchmark. 40

1. INTRODUÇÃO

Neste capítulo serão apresentados o contexto, a motivação e o problema que será tratado neste trabalho. Também serão apresentados a hipótese, os objetivos, a metodologia de pesquisa adotada e a organização deste trabalho.

1.1 Contextualização e Motivação

As aplicações móveis avançaram utilizando os recursos existentes nos novos dispositivos, tais como processadores mais potentes, memórias com espaço de endereçamento maior, sensores mais ricos, GPS, acesso a dados de alta velocidade por meio de Wi-Fi e assim por diante (DELAMARO *et al.*, 2017). Com este avanço tecnológico, aliado ao aumento da importância das aplicações móveis no cotidiano das pessoas, servindo também em cenários altamente críticos, como sistemas bancários ou médicos, são necessárias novas abordagens de Engenharia de Software para garantir a qualidade de tais aplicações antes de sua disponibilização aos usuários.

O teste de software é uma atividade importante para garantir a qualidade da aplicação móvel. Apesar da sua importância, o teste de aplicações móveis enfrenta vários problemas, desafios e necessidades, como a fragmentação[1], dependência de software externo, frequente comunicação externa, usuário e contexto, evolução rápida, recursos limitados, inovação, limitações relacionadas à implementação da plataforma, dentre outros (SAMUEL & PFAHL, 2016). Diversos aspectos da aplicação móvel podem ser considerados na hora do teste, como o seu comportamento, a interação do usuário com a aplicação e aspectos de conectividade e mobilidade, visto que essas são características fundamentais em relação a uma aplicação móvel.

As características específicas de aplicações móveis frente a aplicações de software tradicionais exigem técnicas e métodos de teste especializados. O desenvolvimento de técnicas e ferramentas para identificar características de aplicações móveis e apoiar o teste de software pode ser fundamental para melhorar a produtividade de uma equipe e aumentar a qualidade da aplicação (SILVA, 2017).

Embora os testes automatizados sejam aprimorados para poupar tempo e agilizar o processo, a literatura técnica identifica características específicas de

[1] Fragmentação é definida pela falta de uniformidade. Denominada também como problema de incapacidade para desenvolver uma aplicação independente de contexto e obter o mesmo comportamento em todos os dispositivos/contextos (KAMRAN *et al.*, 2016).

aplicações móveis que se tornam fatores desafiadores para o teste de software. Os desafios encontrados podem ser agrupados e categorizados como: conectividade, GUIs (do inglês, *Graphical User Interface*), sensores e múltiplas configurações (SILVA, 2017).

Para analisar adequadamente a eficiência e desempenho de técnicas, métodos e ferramentas de teste, estudos experimentais são necessários para que estes sejam rigorosamente avaliados e comparados com outros existentes. A experimentação pode ser utilizada para avaliar a influência de fatores nos processos de software, a utilização de determinada ferramenta por um conjunto de usuários, entre outros fatores (SOUZA *et al.*, 2015).

Uma das principais vantagens em realizar um estudo experimental é que se ele for configurado corretamente, podem ser obtidas conclusões sobre a relação entre a causa e o efeito (WOHLIN *et al.*, 2012). Os estudos experimentais permitem testar uma hipótese de pesquisa, analisando a relação entre as variáveis (dependentes e independentes) envolvidas no ambiente do estudo. Outra vantagem em realizar estudos experimentais é a transferência de conhecimentos da academia para a indústria, podendo mostrar a qualidade dos produtos desenvolvidos pela comunidade científica como produtos de qualidade para aprimorar processos e produtos do setor industrial (FREIRE, 2015).

Em teste de aplicações móveis, existem várias abordagens na literatura técnica que diferem a estratégia que utilizam para explorar o comportamento da aplicação em teste. No entanto, ainda não está claro quais aplicações móveis utilizar nos experimentos para verificar os pontos fortes e fracos das abordagens e o quão eficazes são em geral (CHOUDHARY *et al.*, 2015). Todas essas necessidades motivam fortemente este trabalho como uma forma de obter uma visão geral sobre os trabalhos realizados e encontrados na literatura técnica. Na seção a seguir será detalhada essa problemática.

1.2 Descrição do Problema

Os pesquisadores desejam um método padronizado de avaliação para demonstrar a força e a fraqueza de sua abordagem em relação a outras (KAMRAN *et al.*, 2016). Allix *et al.* (2016) descrevem que analisar aplicações em larga escala é um desafio. Neste contexto, os pesquisadores coletam pequenos conjunto de dados limitados, levando a estudos que podem ser tendenciosos e experimentos que muitas vezes não são reproduzíveis.

De acordo com o mapeamento sistemático sobre técnicas de teste de aplicações móveis realizado por Zein *et al.* (2016), quase todos os experimentos reportados utilizam aplicações escolhidas pelos pesquisadores, o que seria uma ameaça à validade do estudo (ZEIN *et al.*, 2016). Os experimentos realizados não são uniformes experimentalmente, o que dificulta a análise da técnica. Assim, não é possível analisar adequadamente a sua efetividade.

A pesquisa em teste de aplicações móveis precisa de diretrizes para melhorar os processos de validação, por meio de experimentos, das técnicas propostas como resultados de pesquisas científicas. Tramontana *et al.* (2019) relatam que é muito difícil estabelecer um conjunto de dados usados para comparar o desempenho de ferramentas ou técnicas. Esse conjunto de dados base é conhecido como *benchmark* (SIM *et al.*, 2003). Por exemplo, Choudhary *et al.* (2015) formaram um *benchmark* de teste considerando um conjunto de aplicações que foi previamente testado nos artigos apresentando as ferramentas em comparação. Eles coletaram 68 aplicações móveis, mas admitiram que apenas 51 deles resultaram como executáveis em cada uma das sete ferramentas de teste consideradas.

Além disso, não há métricas ou critérios de avaliação padronizados para escolha das aplicações, e sem esse conjunto é difícil comparar qualitativamente e quantitativamente os pontos fortes e as limitações das técnicas existentes ou futuras. Além da demanda de tempo e esforço para construir um ambiente para avaliação padronizado, existem outros problemas que devem ser considerados sobre os diferentes contextos que cada pesquisador pode usar e aplicar a sua técnica de teste (KAMRAN *et al.*, 2016). Todas essas necessidades motivam fortemente um conjunto de *benchmarks* de avaliação com a finalidade de avaliar as técnicas de teste de aplicações móveis.

Portanto, tais problemas são argumentos para justificar a construção e disponibilização de um *benchmark* para funcionar como um ambiente padrão de referência na avaliação de técnicas de teste de aplicações móveis.

1.3 Hipótese

A hipótese definida para este trabalho considera o seguinte cenário:

- A definição de um *benchmark* de aplicações móveis contribui na seleção dos sujeitos em experimentos de técnicas de teste de aplicações móveis.

1.4 Objetivos

Nesta seção serão apresentados os objetivos desta pesquisa, que foram definidos a partir dos problemas apresentados na ***Seção 1.2***.

1.4.1 Objetivo Geral

Construir um *benchmark* composto por um conjunto de aplicações móveis, *scripts* de teste e métricas, a fim de auxiliar na fase de planejamento, na etapa de seleção dos sujeitos de um experimento para avaliar técnicas de teste funcional no contexto de aplicações móveis.

1.4.2 Objetivos Específicos

Para atingir o objetivo geral desta pesquisa, pretende-se alcançar os seguintes objetivos intermediários:

- Definição do processo necessário para a concepção do *benchmark*;
- Identificação de um conjunto de aplicações móveis a partir do processo de seleção;
- Definição das métricas de avaliação relevantes no teste de aplicações móveis;
- Categorização do conjunto de aplicações móveis, composta por *scripts* de teste e métricas de avaliação relevantes no teste de aplicações móveis.

1.5 Metodologia

A metodologia de pesquisa adotada neste trabalho é constituída das seguintes fases: Revisão da Literatura, Definição do *Benchmark*, realização da Prova de Conceitos e Estudo de caso, conforme ilustrado na ***Figura 1***.

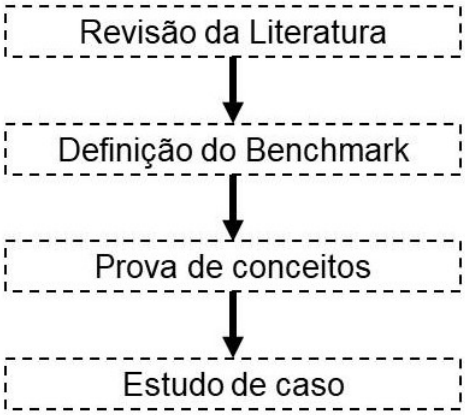


Figura 1. Metodologia aplicada na Pesquisa.

Cada etapa da metodologia de pesquisa aplicada neste trabalho pode ser descrita como:

- Revisão da literatura:
  - Sobre experimentos em Teste de Aplicações Móveis. A extração realizada nesta etapa resulta nas métricas utilizadas nos experimentos, conjunto de aplicações utilizadas, contribuição dos estudos, dentre outros resultados. Essas informações são apresentadas na ***Seção 2.3***.
  - Sobre *benchmark* utilizando aplicações móveis. Essa etapa visou compreender os pontos positivos dos *benchmarks* já existentes e quais são as limitações para o uso em experimentação de técnicas de teste de aplicações móveis. Esses estudos são apresentados na ***Seção 2.5***.
- Definição dos componentes que estruturam o *benchmark*. Para isso, foi definido um processo que extrairia todos os componentes apresentados na ***Seção 3.2***:

    - Definição do conjunto de aplicações móveis. Essa etapa visa definir o conjunto de aplicações móveis.
    - Definição das métricas relevantes de experimentação de teste funcional de aplicações móveis.
    - Definição dos *scripts* de teste das aplicações para a realização de teste funcional.
- Realização da prova de conceitos com o objetivo de verificar como o *benchmark* se comporta dentro de um cenário de experimentação na etapa de planejamento, apresentados na ***Seção 4.1.1***.
- Realização do estudo de caso com o objetivo de verificar como o *benchmark* se comporta dentro de um cenário de experimentação na etapa de planejamento e execução, apresentados na ***Seção 4.1.2***.

1.6 Estrutura do Documento

Este primeiro capítulo apresentou o contexto, no qual este trabalho está inserido, a motivação para realizá-lo, o objetivo a ser atingido e a metodologia a ser seguida. Além deste capítulo, a organização do texto está estruturada em mais três capítulos, descritos a seguir:

- **Capítulo 2:** aborda sobre o referencial teórico necessário para este trabalho. Apresentando sobre teste em aplicações móveis, engenharia de software experimental e experimentos em teste de aplicações móveis. Além de uma visão geral sobre benchmark e os trabalhos relacionados sobre *benchmark* com aplicações móveis.
- **Capítulo 3:** neste capítulo é apresentada a visão geral do *benchmark* feito neste trabalho. Descreve o processo de seleção das aplicações móveis para compor o benchmark e a classificação das aplicações.
- **Capítulo 4:** neste capítulo são apresentados os estudos realizados para verificar como o *benchmark* se comporta dentro de cenários reais de experimentação. Mostra as avaliações realizadas no *benchmark* e as ameaças a validade.
- **Capítulo 5:** neste capítulo são apresentadas as considerações finais deste trabalho juntamente com suas limitações e trabalhos futuros.

2. REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os conceitos relacionados a Teste em Aplicações Móveis, Engenharia de Software Experimental e Experimentos em Teste de Aplicações Móveis. Além disso uma visão geral sobre benchmark e trabalhos que, em especial, envolvem benchmark em aplicações móveis.

2.1 Teste em Aplicações Móveis

Uma aplicação móvel é um software projetado para ser executado em dispositivos móveis (ex: *smartphones* e *tablets*) considerando informações contextuais de entrada. Essas informações podem ser fornecidas pelo usuário ou por recursos dos dispositivos, tais como câmera, contatos, localização, microfone, calendário e rede móvel (REIS, 2016).

As aplicações móveis deixaram de ser usadas apenas para entretenimento e hoje são parte do cotidiano das pessoas para diversas outras atividades, sendo aplicadas em domínios mais críticos, como saúde, negócios, movimentações financeiras, viagem, educação, varejo, dentre outras finalidades, tornando as aplicações cada vez mais complexas e requerendo uma maior qualidade antes da publicação destas aplicações no mercado (DELAMARO *et al.*, 2017).

À medida que as aplicações móveis foram desenvolvidas para abordar domínios cada vez mais críticos, elas não estão apenas se tornando mais complexas para desenvolver, mas também são mais difíceis de serem testadas e avaliadas (CHOUDHARY *et al.*, 2015).

Os testes são indicadores da qualidade do produto que visa avaliar o software e reduzir o risco de falhas. Apesar de, em geral, não ser possível provar por meio de testes que um programa está correto, estes contribuem para aumentar a confiança de que o software desempenha as funções especificadas (MYERS, 2004).

A literatura técnica estabelece significados específicos para termos relacionados, tais como (ISTQB, 2011):

- **Falha:** comportamento anormal ocorrido que impede o funcionamento com relação a especificação.
- **Caso de teste:** consiste em um conjunto de valores de entrada, pré-condições de execução, resultados esperados e pós-condições de execução, desenvolvidos para cobrir certas condições de teste.
- ***Script* de teste:** formam uma sequência de execução de teste que definem a ordem em que vários procedimentos são executados.

Especificamente teste de software para aplicações móveis se refere aos diferentes tipos de teste a serem aplicados para os diferentes tipos de aplicações móveis (nativa, híbrida e web). Esses testes são executados sobre a plataforma móvel usando métodos e ferramentas de teste de software bem definidos a fim de garantir a qualidade em funções, comportamentos, desempenho e qualidade de serviço, assim como atributos como mobilidade, usabilidade, interoperabilidade, conectividade, segurança e privacidade (GAO *et al.*, 2014). O teste de aplicação móvel pode ser realizado de forma automatizada ou manual.

O teste para aplicações móveis tem focado em soluções para problemas técnicos específicos da plataforma. Com isso, os tipos de teste para aplicações móveis podem ser classificados em Teste Funcional e Comportamental, Teste Estrutural, Validação de Requisitos de Qualidade de Serviço (*Quality of Service*), Teste de Usabilidade, Teste de Compatibilidade e Teste de Conectividade (GAO *et al*., 2014; DELAMARO *et al*., 2017).

A seguir, será feita uma breve descrição de teste funcional e comportamental, como uma forma de delimitar o escopo deste trabalho apenas para teste funcional e comportamental.

Teste Funcional e Comportamental: visa validar funções de serviços, comportamento de sistemas externos, interfaces do usuário (UIs – *User Interfaces*) e seus gestos, funções baseadas em localização, perfis de usuários, dados de sistema e dados dos usuários. Segundo Gao *et al*. (2014), é possível encontrar abordagens que apoiem a elaboração e execução de testes funcionais para essas aplicações (GAO *et al.*, 2014). Tais abordagens visam avaliar diferentes aspectos comportamentais (DELAMARO *et al.*, 2017).

Segundo Silva (2017), existem características específicas de aplicações móveis que se tornam fatores desafiadores para o teste de software. Os desafios encontrados na literatura foram agrupados e categorizados como (SILVA, 2017):

- **Conectividade:** as informações de permissões de segurança foram utilizadas para identificar os componentes de conectividade acessados por cada aplicação.
- **GUIs:** foram contabilizados os componentes presentes na aplicação, tais como *layout*, *widgets* e eventos de interface para identificar os componentes de GUI.
- **Sensores:** foi necessário identificar os sensores habilitados no arquivo de configuração declarados no código-fonte da aplicação.
- **Múltiplas configurações:** para identificar foram utilizados o suporte a diferentes resoluções de tela e as configurações de otimização de tela.

2.2 Engenharia de Software Experimental

A Experimentação em Engenharia de Software tem como objetivo caracterizar, avaliar, prever, controlar ou melhorar tanto os produtos, como também

os processos, recursos, modelos ou teorias (DELAMARO *et al.*, 2017). Segundo Wohlin *et al*. (2012), a experimentação pode proporcionar uma base de conhecimentos para reduzir incertezas sobre quais teorias, ferramentas e métodos são adequados. Desta forma, por meio de experimentos é possível avaliar um objeto de estudo (p. ex., técnica, método, ferramenta, etc.) (WOHLIN *et al.*, 2012).

A apresentação de novas abordagens, técnicas ou métodos deveria ser avaliada por meio de experimentação, visando provar sua eficácia ou aceitação. Apesar de importante, a atividade de experimentação é negligenciada ou conduzida de maneira inadequada, comprometendo os resultados obtidos. Nesse sentido, a Engenharia de Software Experimental busca sistematizar o processo de experimentação por meio da proposição de diferentes atividades que devem ser conduzidas durante o processo. Essas atividades permitem organizar melhor o experimento, controlar o andamento e a replicação dos experimentos em diferentes contextos. Na ***Figura 2*** é possível visualizar as atividades do processo de experimentação proposto por Wohlin *et al*. (2012).

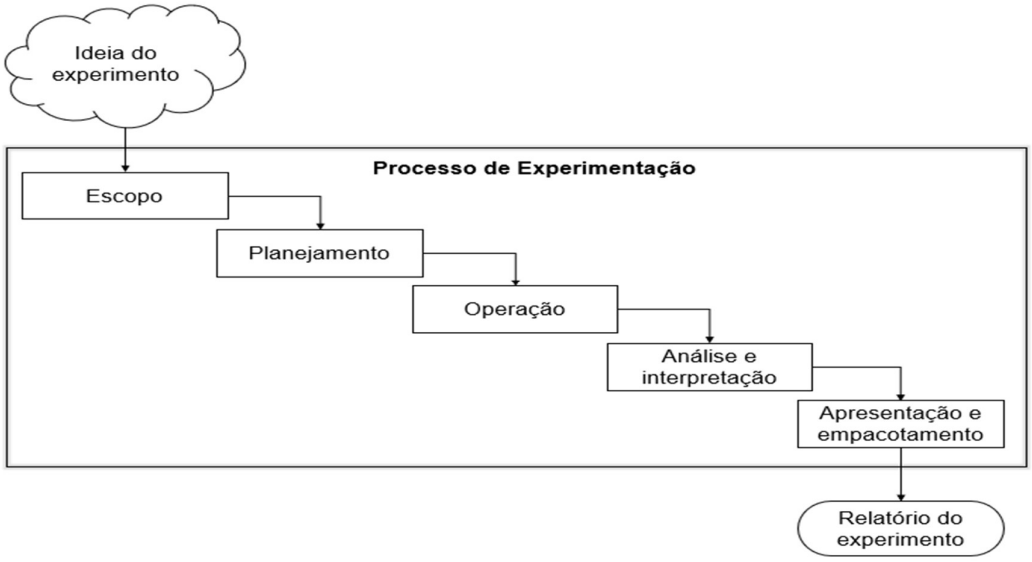


Figura 2. Processo de Experimentação (Wohlin et al., 2012).

As atividades que compõem um experimento são definidas como (WOHLIN *et al.*, 2012; DELAMARO *et al.*, 2017):

- **Escopo:** onde será definido a hipótese e o objetivo do experimento a partir do problema a ser resolvido.
- **Planejamento:** onde será definido o planejamento do experimento. Essa atividade consiste em sete etapas:
    - Seleção do contexto: determinar o contexto no qual será realizado o experimento;

    - Formulação da hipótese: hipótese nula e alternativas devem ser formalizadas para que sejam realizadas análises estatísticas;
    - Seleção das variáveis: definir as variáveis dependentes (efeito) e independentes (causa);
    - Seleção dos sujeitos: também chamada de amostra de uma população;
    - Tipo de design: descreve como os testes são organizados e executados;
    - Instrumentação: prover meios para a realização do experimento e seu monitoramento;
    - Avaliação de validade: verificar influências que podem limitar a interpretação dos resultados do experimento.
- **Operação:** essa atividade consiste em três etapas:
    - Preparação: etapa que os sujeitos são selecionados;
    - Execução: onde os sujeitos realizam suas tarefas;
    - Validação dos dados: etapa para verificar se os dados coletados são válidos.
- **Análise e interpretação:** os dados coletados durante a etapa de operação são analisados e interpretados através de análises estatísticas.
- **Apresentação e empacotamento:** etapa de documentação dos resultados.

2.3 Experimentos em Teste de Aplicações Móveis

Com o propósito de obter conhecimento sobre os experimentos em teste de aplicações móveis descritos na literatura técnica, foi planejado um Mapeamento Sistemático. Estudos de Mapeamento Sistemático são projetados para fornecer uma visão geral de uma área de investigação para determinar se existe evidência de pesquisa sobre um tema e fornecer uma indicação da quantidade de provas.

O principal objetivo desse estudo era entender como eram realizados os experimentos que avaliam técnicas de teste de aplicações móveis. Para abordar o objetivo apresentado, quatro questões foram definidas:

- **QP_1**. Quais os critérios de seleção utilizados para a seleção das aplicações utilizadas nos experimentos?
- **QP_2**. Quais eram as origens das aplicações?
- **QP_3**. Quantas aplicações eram utilizadas nos experimentos?
- **QP_4**. Quais as métricas de avaliação utilizadas?

Foi identificado o trabalho de Zein *et al.* (2016), que apresenta um mapeamento da literatura técnica fornecendo uma ampla visão geral dos estudos experimentais na área de teste de aplicações móveis. Dos 79 estudos apresentados por Zein *et al.* (2016), foram selecionados 21, pois apresentavam experimentos sobre teste funcional e utilizavam aplicações móveis como sujeitos do experimento. Para os estudos selecionados foi investigado e extraído as informações relevantes

para o contexto dessa dissertação. Os estudos selecionados, juntamente com a nova extração, encontram-se na a ***Tabela 1*** e no ***APÊNDICE A*** com as referências.

Tabela 1. Experimentos sobre teste funcional utilizando aplicações móveis retiradas de (Zein et al., 2016).

ID	Critério de escolha das aplicações	Origem das aplicações	# Aplicações	Métricas de avaliação
[1]	Aplicações sensíveis ao contexto	Aplicações retiradas de outros estudos	1	Tempo de execução e cobertura de código
[2]	Não descreve	Não descreve	Não descreve	Número de casos de teste gerados, número de casos de teste modificados e tempo de execução
[3]	Aplicações *open-source*	Não descreve	4	Número de falhas
[4]	Aplicações reais	Aplicações nativas e aplicações de terceiros	6	Número de casos de teste necessários para encontrar a primeira falha e tempo para encontrar a primeira falha
[5]	Aplicações desenvolvidas para o experimento	Aplicação desenvolvida para o estudo	1	Número de falhas
[6]	Aplicações desenvolvidas para o experimento	Aplicação desenvolvida para o estudo	1	Detecção de falhas
[7]	Aplicações reais com falha	Wiki Web	1	Número de falhas e cobertura de código
[8]	Aplicações desenvolvidas para o experimento	Aplicação desenvolvida para o estudo	1	Tempo para criar e executar os casos de teste
[9]	Aplicações reais	Google Play	5	Cobertura de linha e cobertura de método
[10]	Aplicações selecionadas do F-Droid	F-Droid	10	Cobertura de aresta
[11]	Aplicações sensíveis ao contexto	Não descreve	1	Capacidade de detecção de falhas, tamanho do caso de teste e tempo para gerar caso de teste
[12]	Aplicações populares	Google Play	64	Número de falhas e cobertura
[13]	Aplicações *open-source*	Google Play	4	Número de falhas

[14]	Aplicações populares	Google Play	5	Tempo necessário para executar cada caso de teste
[15]	Aplicações reais	Google Play	4	Tempo médio do teste
[16]	Aplicações retiradas de outros estudos	Aplicações retiradas de outros estudos	91	Número de aplicações instrumentadas e tempo necessário para instrumentação
[17]	Aplicações utilizando localização	Google Play	1	Número de casos de teste e número de falhas
[18]	Variedade de aplicações	Não descreve	3	Realização de casos de teste
[19]	Aplicações populares	Google Play	50	Tempo de gravação e tempo de execução de casos de teste
[20]	Aplicações *open-source*	Não descreve	4	Número de falhas, tempo de execução e número de testes
[21]	Aplicações *open-source*	Não descreve	5	Tempo de execução, casos de teste e cobertura

2.3.1 Critério de Escolha das Aplicações

Alguns estudos apresentam critérios de escolha das aplicações móveis para os experimentos, tais como: aplicações *open-source* (ID 3, 13, 20, 21), aplicações populares (ID 12, 14, 19), aplicações desenvolvidas para o experimento (ID 5, 6, 8), aplicações reais (ID 4, 9, 15), aplicações sensíveis ao contexto (ID 1, 11), aplicações reais com falha (ID 7), aplicações selecionadas do F-Droid[2] aleatoriamente (ID 10), aplicações utilizando localização (ID 17), aplicações selecionadas de outros estudos (ID 16), variedade de aplicações (ID 18), e os estudos que não descrevem os critérios (ID 2).

Pode-se observar na ***Figura 3*** que a maioria dos estudos utiliza critérios como: aplicações *open-source* (4 estudos), aplicações desenvolvidas para o experimento (3 estudos) e aplicações populares (3 estudos).

[2] https://f-droid.org/

Figura 3. Quantidade de trabalhos de acordo com os critérios de escolha das aplicações.

2.3.2 Origem das Aplicações

Os estudos apresentam as origens das aplicações móveis para os experimentos, tais como: aplicações retiradas de outro estudo (ID 1, 16), aplicações desenvolvidas para o experimento (ID 5, 6, 8), e aplicações retiradas de algum repositório (ID 4, 7, 9, 10, 12, 13, 15, 14, 17, 19). Há também alguns estudos que não descrevem de onde tiraram as aplicações (ID 2, 3, 11, 18, 20, 21). Pode ser observado na ***Figura 4*** que a maioria dos estudos retirou aplicações de algum repositório (10 estudos), sendo que seis deles retiraram da loja Google Play[3].

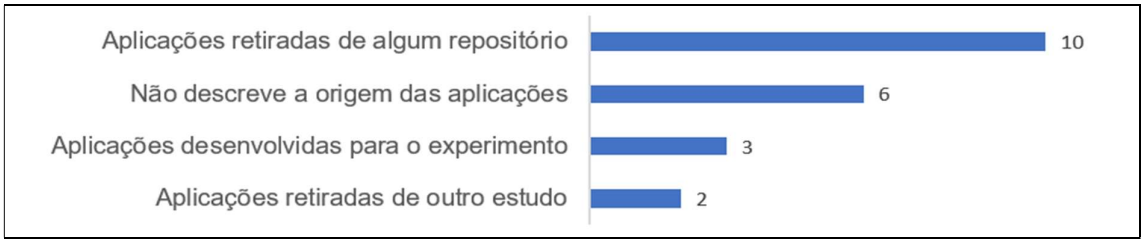


Figura 4. Quantidade de trabalhos de acordo com as origens das aplicações.

2.3.3 Quantidade de Aplicações

Um ponto importante para a realização dos experimentos é decidir o conjunto de aplicações móveis que tenha uma boa representatividade. Na ***Figura 5*** é

[3] https://play.google.com/store

possível visualizar que muitos utilizam apenas uma aplicação para a validação do experimento (ID 1, 5, 6, 7,8 11, 17), alguns utilizaram entre 3 e 10 aplicações (ID 3, 4, 9, 10, 14, 15, 18, 20, 21). Apenas um estudo utiliza 50 (ID 19), outro 64 (ID12) e a maior quantidade de aplicações utilizadas foi 91 (ID 16). Apenas um estudo não descreveu a quantidade de aplicações utilizadas (ID 2).

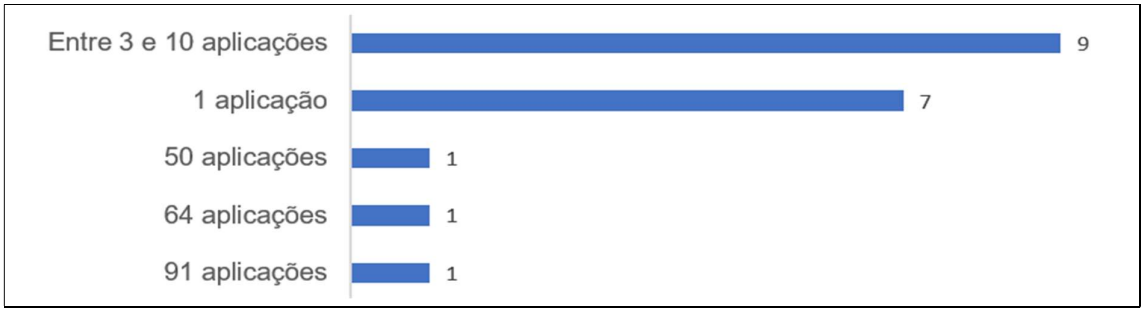


Figura 5. Quantidade de trabalhos de acordo com as quantidades de aplicações utilizadas.

2.3.4 Métricas de Avaliação

A métrica é uma propriedade ou característica mensurável de um produto ou processo. Nos estudos pode haver uma ou mais métricas utilizadas no experimento. Na ***Figura 6*** é possível visualizar que as métricas mais utilizadas nos estudos são: tempo (ID 1, 2, 4, 8, 11, 14, 15, 16, 19, 20, 21), quantidade de casos de teste (ID 2, 4, 8, 11, 14, 17, 18, 19, 20, 21), quantidade/detecção de falhas (ID 3, 5, 6, 7, 11, 12, 13, 17, 20), cobertura (ID 1, 7, 9, 10, 12, 21). Apenas um estudo (ID 16) se preocupou com a criação de *scripts* automatizados para cada aplicação.

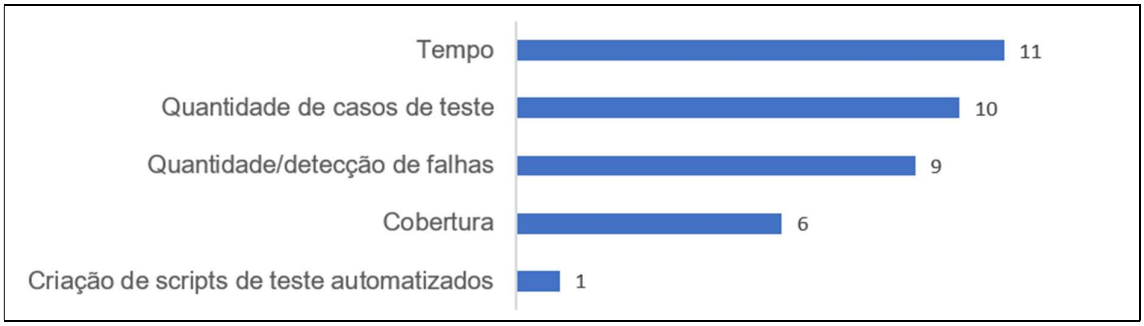


Figura 6. Quantidade de trabalhos de acordo com as métricas de avaliação utilizadas no experimento.

2.3.5 Considerações Sobre Experimentos em Teste de Aplicações Móveis

Os trabalhos apresentados na ***Seção 2.3*** abordam experimentação de teste funcional e comportamental de aplicações móveis. O objetivo foi prover uma visão geral da literatura técnica existente sobre experimentos de teste funcional e comportamental de aplicações móveis.

No trabalho de Zein *et al.* (2016) foram analisados 79 estudos do quais 21 estudos foram analisados neste capítulo devido apresentarem experimentos sobre teste funcional e utilizarem aplicações móveis como sujeitos do experimento. Os resultados mostram que os principais critérios para a escolha das aplicações são: aplicações *open-source*, aplicações desenvolvidas para o experimento e aplicações populares. Um ponto positivo observado é que na maioria dos estudos, as aplicações usadas são retiradas de algum repositório, ou seja, são aplicações usadas no mundo real e é possível utilizar as mesmas aplicações.

Quando se analisa a quantidade de aplicações que são usadas nos experimentos, percebe-se que a maioria utiliza menos de 10 aplicações, o que pode caracterizar que os experimentos não utilizaram uma amostra significativa e representativa das aplicações. Foi observado também as métricas de avaliação utilizadas nos estudos que são o tempo, cobertura, quantidade de casos de teste, quantidade/detecção de falhas e apenas um estudo se preocupou com a criação de *scripts* automatizados para cada aplicação. Nas próximas seções serão apresentados os conceitos sobre *benchmark* e os trabalhos relacionados a *benchmark* de aplicações móveis.

2.4 Visão Geral Sobre *Benchmark*

Um *benchmark* pode ser definido como um teste ou conjunto de testes usados para comparar o desempenho de ferramentas ou técnicas (SIM *et al.*, 2003). O uso de *benchmarks* para comparar sistemas computacionais é um método estabelecido no campo de avaliação de desempenho. Uma suíte de *benchmark* é uma coleção de aplicações que são usadas para exercitar algum elemento de uma plataforma ou sistema com o objetivo de simular o uso de um ou mais recursos. Utilizando os dados obtidos a partir desses exercícios é possível comparar os diferentes sistemas. O processo de utilizar essas aplicações é chamado de *benchmarking* (FERNANDES, 2015).

Segundo Brown *et al.* (2016), um bom *benchmark* requer aplicações do mundo real que representem vários recursos utilizados e podem ser executados no ambiente de execução em dispositivos, emuladores ou simuladores e os *benchmarks* devem refletir o ambiente real (BROWN *et al.*, 2016).

2.5 *Benchmark* com Aplicações Móveis

No contexto de aplicações móveis, as lojas oficiais oferecem aplicações que realizam *benchmarking* e permitem ao usuário comparar diferentes dispositivos. Muitos *benchmarks* de dispositivos possuem, de modo geral, o objetivo de avaliar, gerar pontuação e, posteriormente, comparar essa pontuação com outros dispositivos no mercado (FERNANDES, 2015).

As aplicações de *benchmark* não oferecem informações relevantes dos testes realizados em relação às aplicações para os testadores de aplicações móveis, como quantidade de casos de teste, relatórios de teste, dentre outras informações. Os trabalhos apresentados a seguir, sumarizados na ***Tabela 2*** através dos seus objetivos e quantidades de aplicações utilizadas e no ***APÊNDICE B*** com as referências, apresentam soluções desenvolvidas no meio acadêmico e que envolvem o uso ou a definição de *benchmarks* para o ambiente de aplicações móveis.

Tabela 2. Benchmarks de aplicações móveis.

Estudo	Objetivo	Quantidade
MOONSAMY *et al.* (2014)	Avaliaram o desempenho da técnica de mineração de padrões para identificar os padrões de permissão declaradas pelos desenvolvedores.	1.227
KRUTZ *et al.* (2015)	Verificaram ferramentas de análise estática para entender como as aplicações Android são desenvolvidas e mantidas.	1.179
CHOUDHARY *et al.* (2015)	Realizaram um estudo comparativo com as principais ferramentas de geração de entrada de teste. Avaliam as ferramentas discutindo os pontos fortes e fracos das diferentes técnicas destacando potenciais pesquisas futuras.	60
MISHRA *et al.* (2016)	Verificaram a cobertura e a solidez da análise para chamadas assíncronas e propriedades de ciclo de vida de aplicações.	19
BROWN *et al.* (2016)	Propuseram um conjunto de aplicações completas que representam atividades normalmente executadas.	19
ALLIX *et al.* (2016)	Apresentaram AndroZoo[4] que tem o objetivo de coletar a maior quantidade de apks possíveis buscando contribuir com os esforços de pesquisa em andamento, bem como para permitir novos tópicos de pesquisa em potencial em aplicações Android.	3.182.590
SILVA *et al*. (2018)	O principal objetivo foi identificar e quantificar as características específicas das aplicações para dispositivos móveis. Investigam a presença de testes automatizados, *frameworks* adotados, conectividade externa, elementos da interface gráfica do usuário (GUI), sensores e diferentes configurações do sistema.	663

[4] https://androzoo.uni.lu/

Em Silva (2017) são apresentados alguns estudos que investigam conjuntos de aplicações *open-source*. O trabalho relata que pesquisadores têm estudado conjunto de aplicações móveis e como as comunidades de desenvolvimento tem trabalhado nesse cenário. Na ***Tabela 3*** é sumarizado os trabalhos investigados por Silva (2017) mostrando o objetivo de cada trabalho e o número de aplicações utilizadas.

Dos trabalhos apresentados nas ***Tabela 2*** e ***3*** apenas 5 estão voltados para pesquisas sobre teste de software (FRASER e ARCURI (2012); KOCHHAR *et al*. (2015); LINARES-VÁSQUEZ *et al*. (2015); CHOUDHARY *et al*. (2015); SILVA *et al*. (2018)). Todos os 5 trabalhos utilizaram aplicações *open-source*. Apesar dos trabalhos estarem focados em teste de aplicações móveis, 'todos tem um objetivo diferente como é possível verificar nas ***Tabela 2*** e ***3***. Foi analisado nesses trabalhos quais as informações eram obtidas das aplicações.

Tabela 3. Trabalhos apresentados por Silva (2017) que investigam conjuntos de aplicações open-source.

Estudo	Objetivo	Quantidade
OSSHER *et al*. (2010)	Apresentam uma técnica para resolver, automaticamente, dependências de projetos *open-source*.	13.241
BUTLER *et al*. (2011)	Investigam a composição léxica e sistemática de identificadores de classe Java.	60
FRASER e ARCURI (2012)	Avaliam o teste de software baseado na ferramenta EvoSuite quando aplicado para testar a geração de dados para projetos *open-source*.	100
BHATTACHARYA *et al*. (2013)	Realizaram um estudo empírico com aplicações Android *open-source* com o objetivo de entender o processo de correção de defeitos.	25
KOCHHAR *et al*. (2015)	Realizaram estudos para entender o estado atual do teste de aplicações Android, as ferramentas que são utilizadas e os problemas enfrentados.	627
SYER *et al*. (2013)	Realizaram um estudo exploratório, comparando aplicações móveis. com utilitários unix de pequeno porte e aplicações grandes que são desenvolvidas para *desktops* e servidores.	20
LIU *et al*. (2014)	Fizeram um estudo experimental para analisar defeitos de desempenho em aplicações Android *open-source*.	29
LINARES-VÁSQUEZ *et al*. (2015)	Propõem uma abordagem que registra e minera os *logs* de dados produzidos a partir do uso de aplicações Android e extrai modelos baseados em GUI úteis para geração de casos de teste.	5

No trabalho de Fraser e Arcuri (2012) todas as aplicações foram retiradas do repositório *SourceForge*[5]. Para a seleção das aplicações fora utilizadas 3 critérios: diversidade (diversas categorias para reduzir o viés da seleção), popularidade (aplicações com alto número de download e com mais de mil avaliações), relatório de erros (aplicações com pelo menos 200 relatórios de defeitos). As informações disponíveis sobre as aplicações são as quantidades de classe e *branches*.

As aplicações do trabalho de Kochhar *et al*. (2015) foram retiradas do repositório F-Droid e selecionadas apenas as que estavam hospedadas no GitHub[6]. As informações obtidas sobre as aplicações são a quantidade linhas de código, número de desenvolvedores, se possuíam casos de teste e a quantidade de suítes de teste.

Linares-Vásquez *et al*. (2015) selecionaram aplicações como diversos conjuntos de eventos (click, duplo click e rolagem). As informações disponíveis das aplicações foram a versão, a quantidade de linhas de código, quantidade de *activities*, quantidades de métodos e quantidade de componentes GUI.

Choudhary *et al*. (2015) retiraram as aplicações de outros trabalhos ((AMALFITANO *et al. (*2012); ANAND *et al.* (2012); CHOI *et al.* (2013); MACHIRY *et al.* (2013)) e as únicas informações sobre as aplicações eram a versão e categoria.

O trabalho de Silva *et al*. (2018) também utilizou aplicações retiradas de outros trabalhos (KOCHHAR *et al.* (2015); BHATTACHARYA *et al.* (2013); SYER *et al.* (2013)). Foi o trabalho que disponibilizou o maior número de informações sobre as aplicações: categoria, rating, quantidade de linhas de código, instalações (quantidade mínima e máxima), linguagem, ferramenta de testes utilizadas, elementos de layout, eventos e interação, quantidade de sensores, múltiplas configurações (informações referentes a compatibilidade de dispositivos), conectividade, quantidade de diretórios, classe, método e pacote.

O diferencial desta dissertação é que além de todas as informações apontadas nos 5 trabalhos acima, será apresentado apenas aplicações que contenham *scripts* de teste e que estejam executando. A execução dos *scripts* de teste resultará em relatórios informando quantos testes passaram e a cobertura de teste.

Pode-se notar que todos os trabalhos apresentados realizam experimentos com aplicações Android. Outro ponto observado é a falta de informações mais detalhadas para realização de teste das aplicações. Poucos trabalhos estão voltados para o teste de software. O único que se aproxima do objetivo desta dissertação é o de Silva *et al*. (2018), que busca identificar e quantificar as características específicas das aplicações para dispositivos móveis no contexto de

[5] https://sourceforge.net
[6] https://github.com/

teste de software. Verificando a presença de testes automatizados, *frameworks* adotados e características das aplicações relevantes para o teste. Como diferencial, será executado todas as aplicações que indicam presença de teste automatizado GUI, resultando assim em relatórios de teste e cobertura de teste.

2.6 Considerações Finais

Neste capítulo foram apresentados os fundamentos necessários para o desenvolvimento deste trabalho. Os trabalhos apresentados na ***Seção 2.3*** abordam experimentação de teste funcional e comportamental de aplicações móveis. O objetivo foi prover uma visão geral da literatura técnica existente sobre experimentos de teste funcional e comportamental de aplicações móveis.

Também foi realizado uma investigação na literatura técnica sobre *benchmark* de aplicações móveis. Os trabalhos foram apresentados na ***Seção 2.5*** e abordam diferentes formas de *benchmark* para aplicações móveis. O objetivo desta investigação foi identificar quais informações sobre as aplicações móveis eram disponibilizadas nos trabalhos.

No próximo capítulo será apresentado o *benchmark* descrevendo o processo de seleção das aplicações móveis para compor o benchmark e a classificação das aplicações.

3. *BENCHMARK* DE APLICAÇÕES MÓVEIS PARA EXPERIMENTAÇÃO DE TÉCNICAS DE TESTE

Neste capítulo será apresentado o benchmark proposto para auxiliar na experimentação de técnicas de teste de aplicações móveis e o processo da sua concepção.

3.1 Componentes do *Benchmark*

O *benchmark* servirá de auxílio para pesquisadores na fase de planejamento, na etapa de seleção do conjunto de aplicações adequadas para avaliar técnicas de teste para aplicações móveis. Na ***Figura 7*** é apresentada a estrutura do *benchmark* e seus componentes.

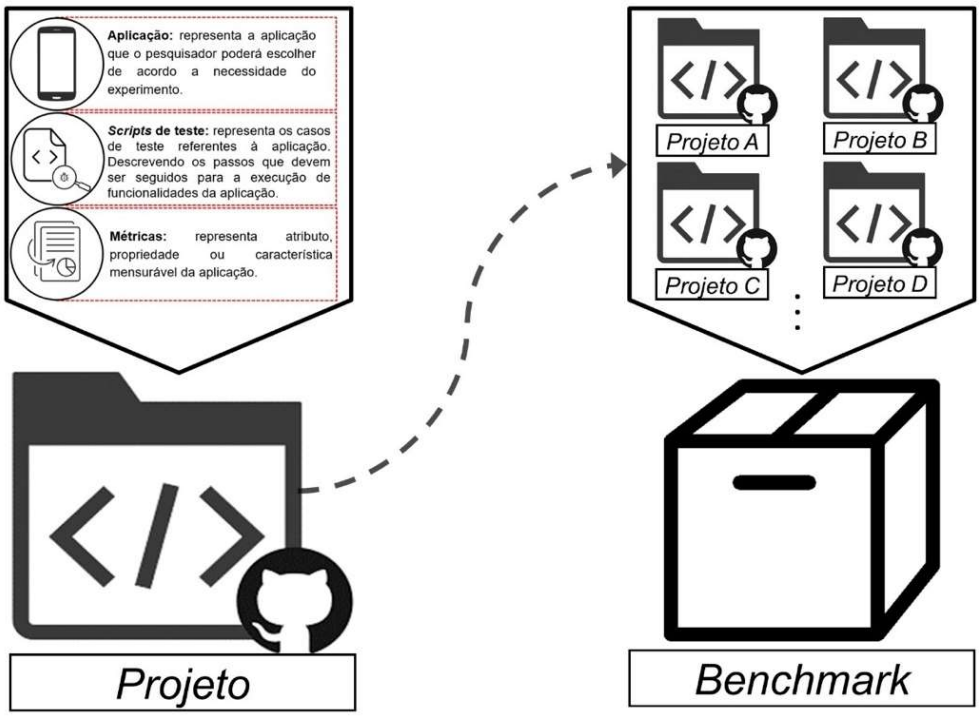


Figura 7. Benchmark.

- **Aplicação:** representa a aplicação que o pesquisador poderá escolher de acordo a necessidade para o experimento.
- ***Scripts* de teste:** representa os casos de teste referentes à aplicação. Os casos de teste descrevem passos que devem ser seguidos para a execução

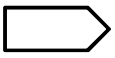
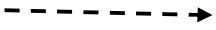
de funcionalidades da aplicação e podem ser descritos em forma de *scripts* de teste.

- **Métricas:** representa atributo, propriedade ou característica mensurável da aplicação. Como por exemplo, a quantidade de testes presentes em cada aplicação e o seu repositório no GitHub.
- **Projeto:** representa a composição da aplicação, *scripts* de teste e métricas.

O *benchmark* é formado por diversos projetos obtidos de diferentes fontes. Cada projeto é composto pelo código fonte da aplicação, *scripts* de teste e métricas. As métricas dos projetos foram retiradas da loja de aplicações móveis *Google Play e* utilizando uma ferramenta chamada *Profiling Mobile Apps* (Prof.MApp)[7] que busca investigar e caracterizar aplicações móveis *open-source* do ponto de vista de teste de software analisando o código fonte da aplicação.

3.2 Seleção das Aplicações para Compor o *Benchmark*

Para a obtenção dos componentes para compor o *benchmark*, foi seguido um processo de seleção das aplicações. A notação utilizada para a modelagem do processo seguido nessa pesquisa foi baseada na notação utilizada por Fontão (2016):

Entidade	Forma de representação	Descrição
Atividade		São as tarefas ou trabalhos a serem realizados pelo processo.
Documento/Artefato (um ou mais)		Produto de software gerado ou consumido por atividades do processo durante a sua realização e que podem existir mais de um do mesmo tipo.
Fluxo de Entrada/Saída		Indica o fluxo de entrada/saída. Se a seta apontar para um artefato, indica que é um produto, se apontar para a atividade, indica que é um insumo.
Dependência entre atividades		Representa a dependência entre as atividades do processo.
Estado inicial		Indica onde é iniciado o fluxo de atividades que definem o processo.
Estado final		Indica onde é encerrado o fluxo de atividades que definem o processo.

Os artefatos do processo seguido para seleção das aplicações móveis que compõem o benchmark são:

[7] https://github.com/davibernardos/ProfMapp

Artefato	Descrição
Fontes Base	São repositórios de aplicações móveis identificados na literatura
Aplicações	Conjunto de aplicações disponíveis para serem baixadas
Aplicações + *Scripts* de teste	Conjunto de aplicações disponíveis para serem baixadas que contém *scripts* de teste
Aplicações + *Scripts* de teste executados	Conjunto de aplicações disponíveis para serem baixadas que contém *scripts* de teste e foram executadas com sucesso

Neste processo, as atividades têm como objetivo principal gerar a base de aplicações e testes para compor o *benchmark* utilizando filtros nos repositórios de aplicações móveis descritos na literatura técnica. Esse processo é ilustrado na ***Figura 8***.

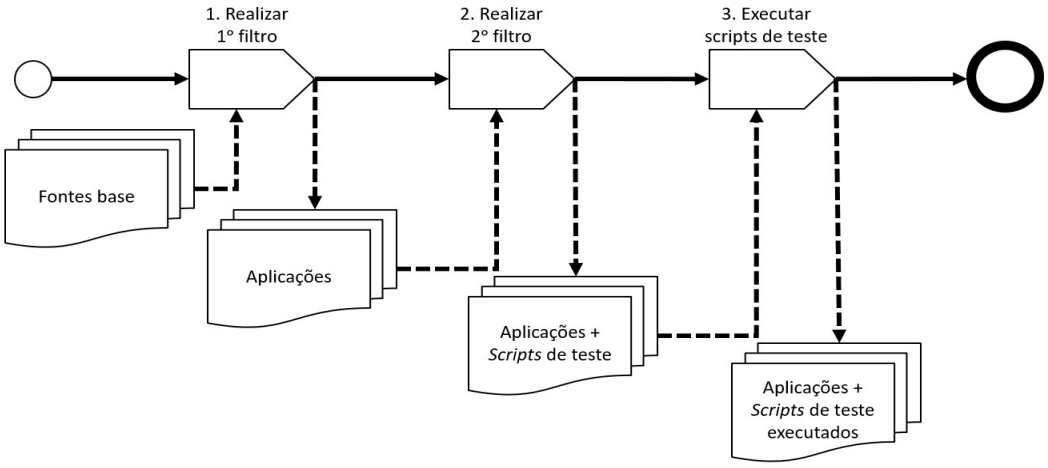


Figura 8. Processo de seleção das aplicações para compor o benchmark.

As atividades do processo de seleção são:

1. **Realizar 1º filtro:** nesta atividade, devem ser utilizados todos os critérios de exclusão a seguir:
   - Aplicações não estarem disponíveis para download no GitHub
   - Aplicações ausentes da loja de aplicações Google Play

Os artefatos usados nesta atividade são as fontes base e é obtido como resultado um subconjunto de aplicações que estão disponíveis no GitHub e Google Play.

2. **Realizar 2º filtro:** nesta atividade, deve ser realizada a análise dos projetos utilizando o Prof.MApp e após essa etapa aplicar o critério de exclusão a seguir:

- Aplicações que não tenham indícios de *script* de teste GUI

O conjunto de aplicações que estão disponíveis no GitHub e Google Play é o artefato utilizado nesta atividade e como resultado é obtido um subconjunto com as aplicações que contêm indícios de *script* de teste GUI.

3. **Executar *scripts* de teste:** nesta atividade, devem ser executados os *scripts* de teste para verificar e validar a funcionalidade dos *scripts*.

O artefato utilizado nesta atividade é o conjunto de aplicações que contêm indícios de *script* de teste GUI e ao final do processo é obtido um subconjunto com as aplicações que contêm indícios de *script* de teste que foram executados com sucesso.

3.2.1 Realização dos Filtros para a Seleção das Aplicações

Para que uma aplicação pertença ao *benchmark*, foi definido que esta seja uma aplicação móvel da plataforma Android, que possua o código-fonte público e que esteja presente na loja Google Play. Para construir o conjunto de aplicações foram utilizadas como fontes base dois trabalhos da literatura técnica: Silva *et al.* (2018) e Tramontana *et al.* (2019).

No trabalho de Silva *et al.* (2018), foi realizada uma análise de 663 projetos. De acordo com esta análise, apenas 231 dos 663 projetos analisados apresentam alguma evidência de testes automatizados.

Para identificar a evidência de testes automatizados nos projetos, uma das informações disponibilizadas pelo Prof.MApp é o conjunto de *frameworks* empregados. Os *frameworks* identificados fornecem evidências sobre qual abordagem de teste foi adotada para testar a aplicação. Os *frameworks* identificados para testes unitários são representados por Android.Test, JUnit, Robolectric, Assertj, Hamcrest e Fest. *Framework* Mockup representados por Android.Test.Mock, EasyMock, Mockito, MockWebServer e PowerMock. Testes GUI, representados pelo Appium, Espresso, MonkeyRunner, Robotium e UIAutomator (Silva *et al.* 2018).

Para o foco deste trabalho, foram analisados apenas os projetos que apresentam indícios de teste GUI automatizado. Porque, em teoria, teste de unidade é igual para qualquer outro tipo de programa Java e como o foco deste trabalho é em aplicações Android que possuem suas características especificas só foram considerados os testes de interface.

Dentre os 231 projetos que apresentam alguma evidência de testes automatizados, apenas 33 utilizam algum *framework* para teste GUI. Na análise feita por Silva *et al*. (2018) foi verificado que 27 dos 33 projetos estavam disponíveis na loja *Google Play*, o que equivale a 4% da amostra apresentada.

No trabalho de Tramontana *et al*. (2019), foi realizado um mapeamento sistemático de teste funcional automatizado de aplicações móveis. Uma das informações apresentadas são as entradas do processo de teste. Para o foco deste, foram analisados os 23 trabalhos existentes que estavam na categoria casos de teste, e resultaram em 106 aplicações. Destas, apenas 17 estavam presentes no GitHub e *Google Play*. Após o *download* dos projetos e utilização da ferramenta Prof.MApp foram identificados 10 com alguma evidência de testes automatizados utilizando *frameworks* para teste GUI, o que equivale a 9,4% da amostra apresentada. É possível visualizar a identificação dos 37 projetos na ***Figura 9***.

Para identificar as aplicações na loja da Google Play, foi por meio do link https://play.google.com/store/Apps/details?id=ID, onde ID é o identificador único de cada aplicação disponível na ***Tabela 4***, na coluna "Google Play" e para *download* do projeto foi por meio do link https://github.com/ID, onde o ID está disponível na coluna GitHub. Por exemplo, o link para acessar a aplicação AlarmClock na loja é https://play.google.com/store/Apps/details?id=com.better.alarm e para acessar o projeto no GitHub no link https://github.com/yuriykulikov/AlarmClock.

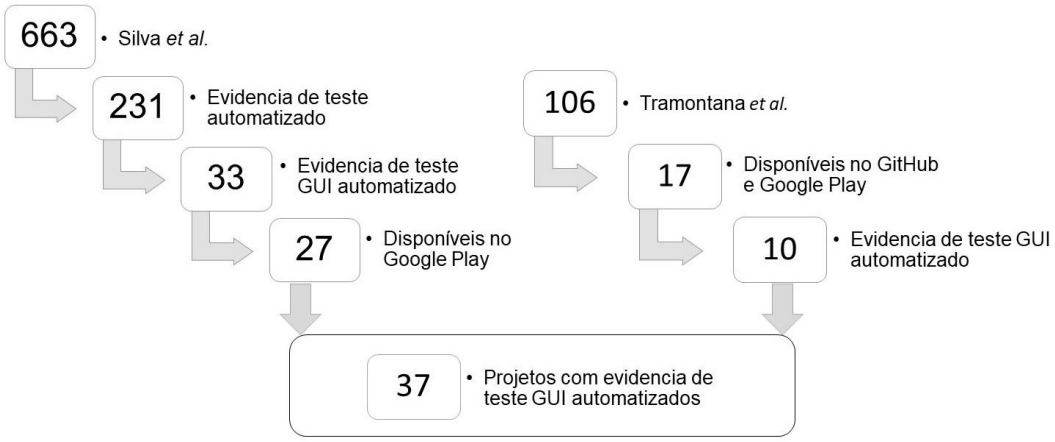


Figura 9. Filtro dos projetos que apresentam alguma evidencia de testes automatizados.

Após as atividades de filtro do processo de seleção do conjunto de aplicações é realizado a atividade de execução dos *scripts* de teste que visa verificar e validar a funcionalidade dos *scripts*. Na próxima seção será descrito a atividade de execução dos *scripts* e geração de relatórios de teste.

3.2.2 Execução dos *Scripts* de Teste

Para que os projetos fossem executados mesmo com falha foi adicionada a tarefa no *build.gradle* de cada projeto e esses foram executados utilizado o comando *./gradlew connectedAndroidTest* que pode ser usado para executar testes instrumentados:

Listagem 1: *Script* da tarefa adicionada ao *gradle* para continuar a execução dos testes mesmo com falha

```
1. project.gradle.taskGraph.whenReady {
2.     -> project.tasks.findAll { it.name =~ /connected.+AndroidTest/ }.each {
3.         it.ignoreFailures = true
4.     }
5. }
```

Na ***Tabela 4*** é possível visualizar todos os 37 projetos e suas informações. Em validação manual, observou-se que não foi possível executar 18 projetos devido a dependências de bibliotecas e erros de projeto. Foi disponibilizado o tempo máximo de 1 hora para tentar resolver os problemas de cada projeto. Se mesmo com esse tempo não fosse possível encontrar o erro, o projeto teria o status de "não executado". Na coluna "Execução" o x representa as aplicações que não foram possíveis serem executadas.

Tabela 4. Projetos que apresentam alguma evidência de testes automatizados.

	ID	Google Play	GitHub	Execução
Silva *et al.* (2018)	P01	com.better.alarm	yuriykulikov/AlarmClock	✓
	P02	li.klass.fhem	klassm/andFHEM	✗
	P03	com.owncloud.android	owncloud/android	✗
	P04	org.flyve.inventory.agent	flyve-mdm/android-inventory-agent	✓
	P05	com.uploadedlobster.PwdHash	phw/Android-PwdHash	✓
	P06	io.gresse.hugo.anecdote	HugoGresse/Anecdote	✗
	P07	de.danoeh.antennapod	danieloeh/AntennaPod	✓
	P08	fr.free.nrw.commons	commons-app/apps-android-commons	✗
	P09	com.eleybourn.bookcatalogue	eleybourn/Book-Catalogue	✗
	P10	es.usc.citius.servando.calendula	citiususc/calêndula	✓
	P11	com.jadn.cc	bherrmann7/Car-Cast	✗
	P12	com.veniosg.dir	veniosg/Dir	✓
	P13	com.llamacorp.equate	EvanRespaut/Equate	✓
	P14	org.gnucash.android	codinguser/gnucash-android	✓
	P15	com.aragaer.jtt	aragaer/jtt_android	✗
	P16	fr.neamar.kiss	Neamar/KISS	✓
	P17	org.xbmc.kore	xbmc/Kore	✓
	P18	org.totschnig.myexpenses	mtotschnig/MyExpenses	✗
	P19	com.nextcloud.client	nextcloud/android	✗
	P20	com.gunshippenguin.openflood	GunshipPenguin/open_flood	✓
	P21	org.sufficientlysecure.keychain	open-keychain/open-keychain	✗
	P22	com.health.openscale	oliexdev/openScale	✗
	P23	monakhv.android.samlib	monakhv/samlib-Info	✓
	P24	org.openintents.shopping	openintents/shoppinglist	✓
	P25	souch.smp	souch/SMP	✓
	P26	org.wheelmap.android.online	sozialhelden/wheelmap-android	✗
	P27	com.vrem.wifianalyzer	VREMSoftwareDevelopment/WifiAnalyzer	✓
Tramontana *et al.* (2019)	P28	org.catrobat.catroid	Catrobat/Catroid	✗
	P29	com.jadn.cc	bherrmann7/Car-Cast	✗
	P30	org.liberty.android.fantastischmemo	helloworld1/AnyMemo	✓
	P31	org.catrobat.paintroid	Catrobat/Paintroid	✗
	P32	org.connectbot	connectbot/connectbot	✗
	P33	org.andstatus.app	andstatus/andstatus	✓
	P34	org.wikipedia	wikimedia/apps-android-wikipedia	✗
	P35	org.openintents.filemanager	openintents/filemanager	✓
	P36	org.openintents.safe	openintents/safe	✗
	P37	org.connectbot	connectbot/connectbot	✗

Utilizando o comando *./gradlew connectedAndroidTest*, é gerado um relatório no formato *HTML* informando a quantidade de testes e quantos deles falharam. Para que esse relatório seja gerado, a execução deve resultar em "BUILD SUCCESSFUL". Caso algum teste falhe, os testes param de executar e resultam em "BUILD FAILED". Na ***Figura 10*** é possível visualizar o relatório de teste gerado do projeto "*com.llamacorp.equate*".

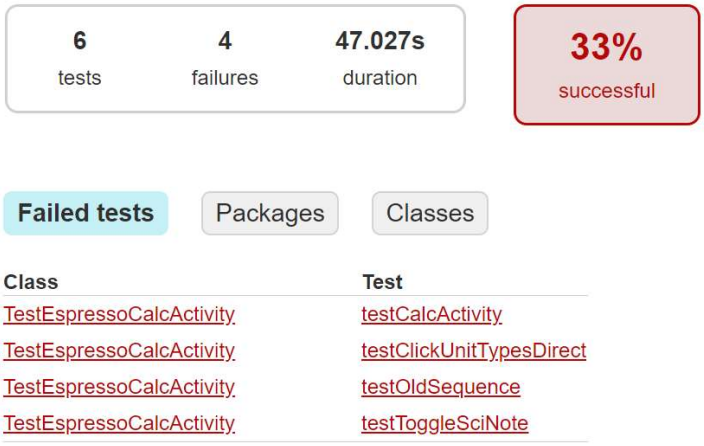


Figura 10. Visualização do relatório de teste gerado.

Na ***Tabela 5*** é possível visualizar a quantidade de testes de cada projeto que foi executado, juntamente com a quantidade de testes que falharam, a porcentagem de testes que passaram e o *framework* que utilizaram.

Tabela 5. Quantidade de testes e falhas de cada projeto informado pelo relatório.

Projeto (ID)	#Teste	#Falha	%	Framework
P01	5	1	80%	Espresso
P04	8	2	75%	Espresso
P05	18	0	100%	Espresso
P07	127	23	81%	Robotium
P10	13	0	100%	Espresso
P12	54	51	5%	Espresso
P13	6	4	33%	Espresso
P14	8	1	87%	Espresso
P16	16	3	81%	Espresso
P17	115	3	97%	Espresso
P20	3	0	100%	Espresso
P23	4	2	50%	Espresso
P24	7	2	71%	Espresso
P25	30	16	46%	Robotium
P27	3	1	66%	Espresso
P30	150	2	98%	Espresso
P33	1	1	0%	Espresso
P35	7	4	42%	Espresso/ Robotium

Outra análise realizada foi a verificação da cobertura de código. Foi utilizado a biblioteca de cobertura de código livre JaCoCo[8]. Para ativar essa opção é preciso adicionar uma propriedade à variante de compilação de *debug*. Para ativar a cobertura deve incluir a propriedade *testCoverageEnabled* no *build.gradle*:

Listagem 2: *Script* da tarefa adicionada ao gradle para gerar relatório de cobertura de código

```
1. android {
2.         …
3.         buildTypes {
4.             debug {
5.                 testCoverageEnabled true
6.             }
7.             …
8.         }
9. }
```

Ao final da execução, é gerado um relatório informando cobertura de instruções perdidas, cobertura de ramificações perdidas, complexidade ciclomática, linhas, métodos e classes. Algumas aplicações não geraram o relatório de cobertura (6 aplicações). Apenas 12 projetos geraram os relatórios de cobertura de código. Na ***Figura 11*** é possível visualizar o relatório de cobertura gerado do projeto "*com.llamacorp.equate*" e na ***Tabela 6*** é possível ver as informações de cobertura geradas de cada projeto.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.llamacorp.equate.view		69%		59%	193	458	422	1,327	60	233	8	57
com.llamacorp.equate		57%		41%	205	394	355	889	34	176	1	12
com.llamacorp.equate.unit		87%		51%	65	170	115	654	26	110	2	10
com.llamacorp.equate.unit.updater		61%		44%	66	115	114	303	16	44	4	10
com.viewpagerindicator		58%		39%	51	87	78	203	6	32	0	5
com.llamacorp.equate.view.IdlingResource		100%		75%	1	7	0	10	0	5	0	1
Total	5,040 of 16,689	70%	632 of 1,223	48%	581	1,231	1,084	3,386	142	600	15	95

Figura 11. Visualização do relatório de cobertura gerado.

Ao final da execução é gerado um relatório informando cobertura de instruções perdidas, cobertura de ramificações perdidas, complexidade ciclomática e quantidade de linhas de código. Na ***Tabela 7*** é possível visualizar quais relatórios foram gerados de cada projeto. No ***APÊNDICE C*** é possível visualizar todas as informações obtidas de cada aplicação.

[8] https://www.eclemma.org/jacoco/

Tabela 6. Informações do relatório de cobertura de cada projeto.

Projeto (ID)	Cobertura de instruções perdidas (%)	Cobertura de ramificações perdidas (%)	Perdido	# Complexidade ciclomática	Perdido	# Linhas	Perdido	# Métodos	Perdido	# Classes
P05	94%	80%	20	97	34	259	5	53	1	12
P10	27%	15%	4.383	5.583	10.884	14.998	2.232	3.277	424	700
P12	3%	1%	1.699	1.738	4.204	4.336	953	986	165	179
P13	70%	48%	581	1.231	1.084	3.386	142	600	15	95
P16	34%	24%	1.393	1.893	3.214	4.932	505	864	72	168
P17	23%	14%	4.703	5.602	12.065	15.773	2.525	3.266	618	827
P20	73%	63%	67	175	155	521	31	109	8	30
P24	34%	26%	1.192	1.537	3.213	4.945	420	670	85	144
P25	5%	3%	814	843	2.087	2.180	336	353	43	50
P27	73%	47%	604	1.502	926	3.393	297	1.092	20	237
P30	28%	22%	3.333	4.389	8.958	12.678	1.945	2.808	396	610
P35	23%	13%	1.176	1.378	2.547	3.353	553	726	84	140

Tabela 7. Relatórios gerados de cada projeto.

ID	Projeto	Relatório de teste	Relatório de cobertura
P01	com.better.alarm	✓	
P04	org.flyve.inventory.agent	✓	
P05	com.uploadedlobster.PwdHash	✓	✓
P07	de.danoeh.antennapod	✓	
P10	es.usc.citius.servando.calendula	✓	✓
P12	com.veniosg.dir	✓	✓
P13	com.llamacorp.equate	✓	✓
P14	org.gnucash.android	✓	
P16	fr.neamar.kiss	✓	✓
P17	org.xbmc.kore	✓	✓
P20	com.gunshippenguin.openflood	✓	✓
P23	monakhv.android.samlib	✓	
P24	org.openintents.shopping	✓	✓
P25	souch.smp	✓	✓
P27	com.vrem.wifianalyzer	✓	✓
P30	org.liberty.android.fantastischmemo	✓	✓
P33	org.andstatus.app	✓	
P35	org.openintents.filemanager	✓	✓

3.3 *Benchmark* Composto

Após o processo seguido para concepção da base de aplicações do *benchmark* na ***Figura 8***, é possível visualizar na ***Figura 12*** a obtenção das 18 aplicações que compõem o benchmark. Com a realização do processo, também são obtidos todos os componentes descritos na ***Figura 7***.

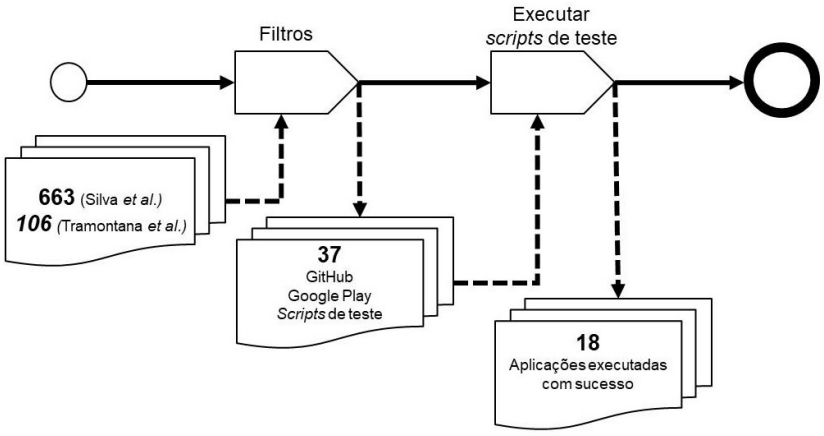


Figura 12. Resultado da utilização do processo de seleção das aplicações para compor o benchmark.

As aplicações foram obtidas dos trabalhos de Silva *et al.* (2018) e Tramontana *et al.* (2019). Após os filtros foram extraídas apenas aplicações que tinham indícios de *scripts* de teste GUI de aplicações móveis. Após a análise foi verificado que apenas dois *frameworks* para teste GUI foram utilizados: Espresso e Robotium.

Todas as métricas obtidas das aplicações são classificadas em 5 categorias: informações gerais, conectividade, GUIs, sensores e múltiplas configurações. As informações gerais são obtidas por meio da loja *Google Play*, dos relatórios gerados (relatório de teste e relatório de cobertura descritos na ***Seção 3.2.2***) por meio da execução das aplicações e da extração utilizando o Prof.MApp. As outras categorias (conectividade, GUIs, sensores e múltiplas configurações) foram informações obtidas por meio da extração do Prof.MApp, retiradas do trabalho de Silva *et al.* (2018). Essa classificação foi feita a partir da identificação das diferentes características que representam desafios no teste de dispositivos móveis.

Como foi visualizado na ***Figura 7***, cada projeto será composto pela aplicação móvel, scripts de teste e métricas. Na ***Tabela 8*** contém todas as métricas disponíveis no *benchmark* e na ***Tabela 9*** é possível visualizar o exemplo com o projeto "*de.danoeh.antennapod*" para demostrar as métricas disponíveis.

Tabela 8. Métricas que compõem o benchmark.

Informações Gerais	**Múltiplas Configurações**	**GUIs**
GitHub	**# Configurações de tela**	**# Elementos de layout**
Google Play	small	framelayout
Categoria	normal	linearlayout
Rating	large	tablelayout
Instalação	xlarge	gridlayout
minSDKVersion	ldpi	relativelayout
targetSDKVersion	mdpi	**# Elementos de widgest**
LOC	hdpi	textview
LOC test	xhdpi	button
Classe src	xxhdpi	radiobutton
Classe test	xxxhdpi	checkbox
Metodo src	nodpi	switch
Metodo test	**# Suporte a diferentes telas**	togglebutton
# Teste	resizeable	imageview
% Teste	smallscreens	progressbar
Cobertura	normalscreens	seekbar
Framework	largescreens	ratingbar
Conectividade	xlargescreens	spinner
access_network_state	anydensity	webview
access_wifi_state	requiressmallestwidthdp	text
bluetooth	compatiblewidthlimitdp	edittext
bluetooth_admin	largestwidthlimitdp	textcapwords
internet		textpassword
change_network_state		phone
chage_wifi_state		textmultiline
Sensores		number
accelerometer	rotation_vector	radiogroup
ambient_temperature	temperature	listview
gravity	gps_location	gridview
gyroscope	camera	**# Elementos de eventos de interface**
light	wifi	Onswipetouchlistener
linear_acceleration	bluetooth	Ondoubletaplistener
magnetic_field	internet	Onscalegesturelistener
orientation	network	Shakelistener
pressure	location	Ondraglistener
proximity	gps	Onscrolllistener
relative_humidity	microfone	

Tabela 9. Informações do projeto "de.danoeh.antennapod".

Informações Gerais	GitHub	danieloeh/AntennaPod
	Google Play	de.danoeh.antennapod
	Categoria	Reproduzir e editar vídeos
	Rating	4.6
	Instalação	100.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	38.183
	LOC test	6.549
	Classe src	296
	Classe test	38
	Metodo src	116.634
	Metodo test	19.026
	# Teste	127
	% Teste	81%
	Cobertura	-
	Framework	Robotium
Conectividade		4
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	15
	# Elementos de eventos de interface	1
Sensores		6
Múltiplas Configurações	**# Configurações de tela**	6
	# Suporte a diferentes telas	5

3.4 Classificação das Aplicações

Ao realizar a identificação das métricas de cada aplicação utilizando o Prof.MApp, relatórios gerados e informação da loja *Google Play*, foi realizada a classificação das aplicações de acordo com a quantidade de testes existentes e cobertura de código. Na ***Tabela 10*** pode ser visualizado o *ranking* de acordo com a quantidade de testes existentes na aplicação representado pela coluna "# Teste" seguido pelas colunas "% Teste", que informa o percentual de teste que passou, "Cobertura" com o percentual de cobertura de código que os testes conseguiram cobrir, "LOC" com a quantidade de linhas de código e "LOC test" com a quantidade de linhas de teste.

Tabela 10. Ranking de acordo com a quantidade de testes existentes na aplicação.

ID	Informações Gerais				
	# Teste	% Teste	Cobertura	LOC	LOC test
P30	150	98%	28%	20.504	2.953
P07	127	81%	-	38.183	6.549
P17	115	97%	23%	28.636	6.148
P12	54	5%	3%	8.839	1.778
P25	30	46%	5%	3.468	455
P05	18	100%	94%	541	250
P16	16	81%	35%	7.740	186
P10	13	100%	27%	25.495	753
P04	8	75%	-	3.964	201
P14	8	87%	-	22.979	4.818
P24	7	71%	34%	10.647	314
P35	7	42%	23%	6.780	1.391
P13	6	33%	70%	5.564	1.045
P01	5	80%	-	3.883	1.219
P23	4	50%	-	18.361	720
P20	3	100%	73%	948	175
P27	3	66%	73%	7.588	9.659
P33	1	0%	-	49.346	9.276

Na ***Tabela 11*** pode ser visualizado o *ranking* de acordo com o percentual de cobertura de código no campo “Cobertura”. Os campos que não estão preenchidos são os projetos que não geraram o relatório de cobertura como descritos na ***Tabela 7***.

Tabela 11. Ranking de acordo com o percentual de cobertura de código.

ID	Informações Gerais				
	Cobertura	# Teste	% Teste	Linha src	Linha test
P05	94%	18	100%	541	250
P20	73%	3	100%	948	175
P27	73%	3	66%	7.588	9.659
P13	70%	6	33%	5.564	1.045
P16	35%	16	81%	7.740	186
P24	34%	7	71%	10.647	314
P30	28%	150	98%	20.504	2.953
P10	27%	13	100%	25.495	753
P17	23%	115	97%	28.636	6.148
P35	23%	7	42%	6.780	1.391
P25	5%	30	46%	3.468	455
P12	3%	54	5%	8.839	1.778
P07	-	127	81%	38.183	6.549
P04	-	8	75%	3.964	201
P14	-	8	87%	22.979	4.818
P01	-	5	80%	3.883	1.219
P23	-	4	50%	18.361	720
P33	-	1	0%	49.346	9.276

Para fins de validação, todos os dados extraídos estão disponíveis online em https://goo.gl/PFDgw2 na forma de uma planilha Google.

3.5 Considerações Finais

Com base nos resultados obtidos na análise feita em 769 aplicações móveis, devem ser consideradas as decisões subjetivas que podem ter ocorrido durante a análise. Algumas aplicações relevantes podem não ter sido selecionadas como aplicações para compor o benchmark. Para minimizar essa ameaça, seguiu-se o processo descrito na ***Figura 8***, guiado pelos critérios de exclusão.

Os resultados mostram que apenas 4,8% dos projetos apresentam evidências de teste automatizado GUI. Percebe-se que dentre os projetos em que foi identificada a presença de teste (37 projetos), apenas metade (18 projetos) foram executadas com sucesso. Na ***Figura 12*** é possível visualizar o processo de seleção das 18 aplicações.

Outra análise realizada foi a verificação dos *frameworks* utilizados. Segundo os resultados obtidos, para teste GUI, o Espresso tem sido o *framework* mais utilizado aparecendo em 16 dos 18 projetos analisados. Em relação à quantidade de testes, pode ser observado na ***Tabela 10*** que apenas 4 projetos apresentam mais de 50 testes. Analisando de forma geral, não é proporcional a quantidade de testes com a quantidade de linhas de teste. Isso se dá por duas razões: o tamanho do teste e/ou pela quantidade de testes de outro nível (por exemplo, unidade) presente no projeto.

Pode ser observado na ***Tabela 11*** que dentre os projetos foi identificado que a porcentagem de cobertura de código é baixa. Apenas 4 projetos têm cobertura maior que 50%. Tais resultados levam ao entendimento de que os projetos utilizados nos trabalhos não possuem um conjunto de testes adequados para a cobertura.

4. ESTUDOS DE AVALIAÇÃO DO *BENCHMARK* DE APLICAÇÕES MÓVEIS

Neste capítulo serão apresentados os estudos realizados para investigar e avaliar o benchmark. Serão respondidas duas questões de pesquisa como forma de verificar como o benchmark se comporta dentro de um cenário de experimentação real*.*

4.1 Avaliação do *Benchmark* de Aplicações Móveis

Foram elaborados dois estudos para investigar e avaliar o *benchmark* no contexto de técnicas de teste de aplicações móveis. Com base no objetivo geral deste trabalho foram elaboradas questões de pesquisa para conduzir esta avaliação. Tais questões de pesquisa são:

- **Q_1**. O *benchmark* auxilia os pesquisadores no planejamento do experimento na etapa de seleção dos sujeitos?
- **Q_2**. O *benchmark* auxilia na execução de experimentos para avaliação de técnicas de teste de aplicações móveis?

O conjunto de aplicações utilizado no estudo foi o selecionado no ***Capítulo 3***. Para responder a **Q_1**, foi necessário identificar um trabalho que descrevesse detalhes do planejamento da experimentação de uma técnica de teste para aplicações móveis. Utilizou-se o trabalho de Usman *et al.* (2018) como prova de conceito, descrevendo os benefícios que resultariam no uso do *benchmark* para auxiliar nos experimentos utilizados.

Para responder a **Q_2**, foi necessário identificar um trabalho que descrevesse passos da execução da experimentação de uma técnica de teste de aplicações móveis. O trabalho utilizado foi o de Adamsen *et al.* (2015), que investigou os benefícios que resultaria o uso do *benchmark* na execução dos experimentos.

4.1.1 Auxílio no Planejamento do Experimento

Q_1. O *benchmark* auxilia os pesquisadores no planejamento do experimento na etapa de seleção dos sujeitos?

Para responder a **Q_1**, foi utilizado o trabalho de Usman *et al.* (2018) que propõe uma abordagem para testar aplicações móveis considerando conjuntos de eventos extraídos da lista de permissões declaradas.

A lista de permissões consiste no nome da permissão e nos recursos relacionados. Para gerar vários cenários do contexto de execução de uma aplicação, foi aplicado o método de combinação nas condições do recurso. Essas

informações são usadas para gerar casos de teste capazes de testar os eventos de contexto. Com isso, Usman *et al.* (2018) geraram uma lista de permissões e os recursos relacionados às permissões usando a lista obtida do arquivo de manifesto, conforme mostrado na ***Tabela 12***.

Tabela 12. Lista de permissão com sua descrição e fontes relacionadas retirada de Usman et al. (2018).

Permissão	Descrição da Permissão	Recursos	Estado
Access_Fine_Location	Acessar localização precisa	Wifi, GPS, Radio	On/Off
Access_Network_State	Autorização para acessar redes	Wifi, GPS, Radio	On/Off
Access_Course_Location	Acessar localização aproximada	Wifi, GPS, Radio	On/Off
Write_External_ Storage	Escrita de armazenamento externo	Cartão SD	Free/Full
Use_Finger_Print	Usar hardware de impressão digital	LCD, Câmera	On/Off
Internet	Abrir soquetes de rede	Wifi, Radio	On/Off
Camera	Acesso ao dispositivo da câmera	Lanterna, Cartão SD	On/Off, Free/Full
Vibrate	Acesso ao vibrador	Vibrador	On/Off
Read_External_Storage	Leitura de armazenamento externo	Cartão SD	Free/Full
Receive Boot Complete	Ouvir /Reverter a ação BOOT_COMPLETED após a inicialização do sistema	Cartão SD	Free/Full

A avaliação experimental do trabalho de Usman *et al.* (2018) inclui 3 etapas: selecionar os sujeitos, analisar e projetar dados de teste, executar testes e ler os resultados. Utilizando o *benchmark* proposto neste trabalho, Usman *et al.* (2018) teriam auxílio na etapa de selecionar os sujeitos da avaliação experimental.

Para a realização da avaliação Usman *et al.* (2018) utilizaram apenas duas aplicações Android *open-source* e as características utilizadas sobre as informações foram: linha de código, classes, métodos, categoria, classificação e quantidade de instalações feitas pela loja. As características estão detalhadas na ***Tabela 13***. Para o contexto desta dissertação, todos os dados foram atualizados utilizando o Prof.MApp e retirados da Google Play.

Tabela 13. Características das aplicações móveis testadas no trabalho de Usman et al. (2018) com os dados atualizados utilizando o Prof.MApp e retirados da Google Play.

Projeto	Beem	Subsonic
LOC	12.962	10.934
Classe	134	106
Método	46.116	37.506
Categoria	Comunicação	Música e áudio
Classificação	3.6	4.3
Comentário	823	6.514
Instalação	50.000 – 100.000	500.000 – 1.000.000

Beem é uma aplicação móvel gratuita e *open-source* que é usado para fornecer um cliente Jabber (XMPP) completo e fácil de usar no Android. O streamer de música Subsonic é um reprodutor de mídia gratuito para música e vídeo que suporta praticamente todos os formatos de áudio. Também pode ser usado como um controle remoto para música em um servidor (Usman *et al.* 2018).

Os recursos de conectividade e sensores declarados nos projetos utilizados no trabalho de Usman *et al.* (2018) foram extraídos utilizando o Prof.MApp. A ***Tabela 14*** mostra detalhadamente quais recursos de conectividade foram declarados nos projetos.

Tabela 14. Recursos de conectividade dos projetos utilizados no trabalho de Usman et al. (2018).

Informações Gerais	**Projeto**	Beem	Subsonic
Conectividade	# Recursos de conectividade	3	4
# Recursos de conectividade	access_network_state	1	1
	access_wifi_state	1	1
	bluetooth	0	1
	bluetooth_admin	0	0
	internet	1	1
	change_network_state	0	0
	chage_wifi_state	0	0

Na ***Tabela 15*** é possível ver detalhadamente quais sensores foram declarados em cada projeto. Com essas informações, o trabalho de Usman *et al.* (2018) teria informações dos recursos utilizados já na fase de planejamento do experimento.

Tabela 15. Sensores dos projetos utilizados no trabalho de Usman et al. (2018).

Informações Gerais	**Projeto**	Beem	Subsonic
Sensores	# Sensores	3	4
# Sensores	accelerometer	0	0
	ambient_temperature	0	0
	gravity	0	0
	gyroscope	0	0
	light	0	0
	linear_acceleration	0	0
	magnetic_field	0	0
	orientation	0	0
	pressure	0	0
	proximity	0	0
	relative_humidity	0	0
	rotation_vector	0	0
	temperature	0	0
	gps_location	0	0
	camera	1	0
	wifi	0	0
	bluetooth	0	1
	internet	1	1
	network	1	1
	location	0	0
	gps	0	0
	microfone	0	1

O trabalho Usman *et al.* (2018) poderia utilizar o *benchmark* na fase de selecionar os sujeitos. Das 18 aplicações apresentadas no *benchmark*, apenas 12 poderiam ser utilizadas no trabalho de Usman *et al.* (2018), pois 6 aplicações não apresentam recursos de conectividade e sensores. Essas informações podem ser visualizadas na ***Tabela 16***. Na coluna "Utilização", o X representa as aplicações que não poderiam ser utilizadas no experimento. As informações como recursos de conectividade e sensores são necessárias para a aplicação conseguir realizar o experimento. Todos os recursos de conectividade e sensores detalhados de cada projeto podem ser visualizados em https://goo.gl/PFDgw2.

Tabela 16. Recursos de conectividade e sensores dos projetos presentes no benchmark.

Projeto	# Recursos de conectividade	# Sensores	Utilização
com.better.alarm	0	0	✗
org.flyve.inventory.agent	5	6	
com.uploadedlobster.PwdHash	0	0	✗
de.danoeh.antennapod	4	6	
es.usc.citius.servando.calendula	3	2	
com.veniosg.dir	0	0	✗
com.llamacorp.equate	3	2	
org.gnucash.android	3	2	
fr.neamar.kiss	0	0	✗
org.xbmc.kore	3	3	
com.gunshippenguin.openflood	0	0	✗
monakhv.android.samlib	3	3	
org.openintents.shopping	1	2	
souch.smp	0	1	
com.vrem.wifianalyzer	1	2	
org.liberty.android.fantastischmemo	1	2	
org.andstatus.app	3	2	
org.openintents.filemanager	0	0	✗

O *benchmark* não conseguiu suprir todas as informações necessárias utilizadas no trabalho de Usman *et al.* (2018). A lista de permissões é uma informação importante para o trabalho, mas não está disponível nas métricas do *benchmark*. Em compensação, ao utilizar o *benchmark* poderia ser utilizado um número maior de aplicações com quantidade diferentes de recursos. Teria um conjunto de aplicações com diversidade de recursos.

Como resultado da Q_1, pode ser observado nas informações acima que o *benchmark* poderia auxiliar na fase do planejamento do experimento na etapa de seleção dos sujeitos. Com o *benchmark*, é possível obter informações relevantes como quantidade de recursos disponíveis da aplicação ainda na fase de planejamento. Além dessa informação, também é possível utilizar um conjunto maior de aplicações no experimento, juntamente com a diversidade de recursos utilizados, podendo assim ter uma melhor generalização dos resultados. Por fim, os testes presentes no *benchmark* podem ser comparados com os testes gerados pela abordagem de proposta por Usman *et al.* (2018).

4.1.2 Auxílio na Execução do Experimento

Q_2. O *benchmark* auxilia na execução de experimentos para avaliação de técnicas de teste de aplicações móveis?

Para responder a **Q_2**, foi utilizado o trabalho de Adamsen *et al.* (2015) que implementou a ferramenta Thor[9], a qual propõe uma nova metodologia aproveitando suítes de testes existentes de modo que cada caso de teste seja exposto a condições em que certos eventos inesperados possam interferir na execução.

Para a realização da avaliação do trabalho de Adamsen *et al.* (2015), foram utilizadas apenas quatro aplicações Android *open-source*, todas com suítes de testes existentes e com características como versão, classificação, quantidade de instalações feitas pela loja, linhas de código, linhas de teste, quantidade de testes[10] estáveis e instáveis, cobertura de código e tempo de execução dos testes estáveis. As características estão detalhadas na ***Tabela 17*** e todos os dados foram retirados de Adamsen *et al.* (2015).

Tabela 17. Características das aplicações móveis testadas no trabalho de Adamsen et al. (2015).

Projeto		Pocket Code	Pocket Paint	Car Cast	AnyMemo
Versão		Agosto/2014	Setembro/2014	Julho/2014	Abril/2014
Classificação		3.6	3.7	4.1	4.5
Instalação		50.000 – 100.000	10.000 – 50.000	100.000 – 500.000	100.000 – 500.000
LOC	**SRC**	34.000	6.600	6.000	20.100
	Teste	19.400	3.800	500	1.200
Teste	**Estável**	340	121	17	29
	Instável	142	11	1	19
Cobertura		65%	76%	48%	31%
Tempo		2h 21m 17s	45m 25s	12m 47s	12m 17s

A característica mais importante para a aplicação fazer parte do conjunto de aplicações utilizadas no trabalho de Adamsen *et al.* (2015) foi a existência de *scripts* de teste, pois a ferramenta Thor aproveita suítes de testes existentes na sua execução.

Como forma de mostrar que o *benchmark* auxilia na execução de experimentos para avaliação de técnicas de teste de aplicações móveis foi

[9] https://github.com/cs-au-dk/thor

[10] A linha Teste mostra o número de testes que foram estáveis vs. instáveis (isto é, falhas consistentes ou aleatórias) no ambiente de teste.

executado o exemplo[11] apresentado por Adamsen *et al.* (2015) utilizando as aplicações presentes no *benchmark*. Nesse exemplo, Thor é usado para execuções de testes comuns, ou seja, sem condições adversas. Na ***Figura 13*** é possível verificar que o algoritmo utilizado no exemplo foi o "No" que significa que nenhuma condição adversa foi acionada durante a execução do teste.

Algumas etapas[12] são necessárias para usar novas aplicações na ferramenta Thor. Todas as etapas foram realizadas nas 18 aplicações presentes no *benchmark*.

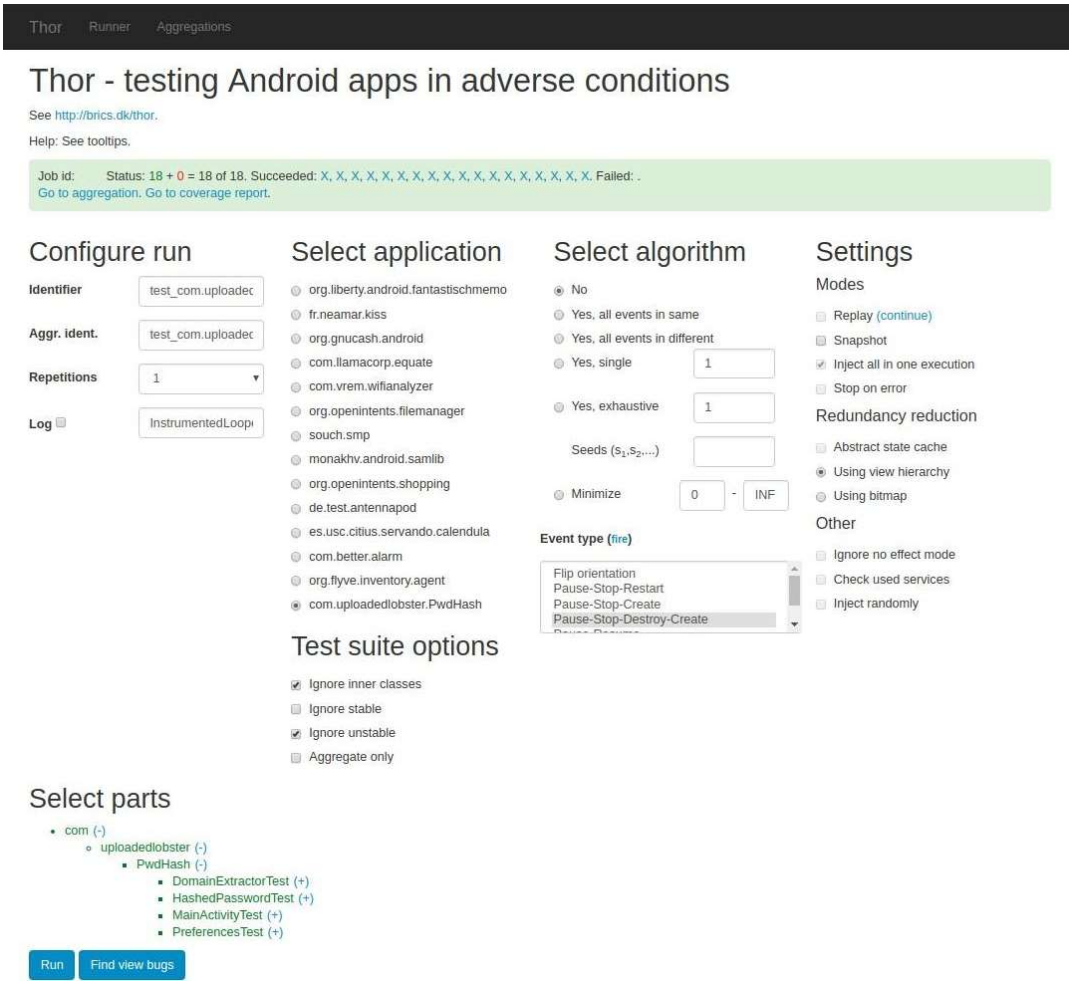


Figura 13. Interface da ferramenta Thor com as aplicações do benchmark inseridas.

Das 18 aplicações apresentadas no *benchmark* dessa dissertação, apenas 12 poderiam ser utilizadas no trabalho de trabalho Adamsen *et al.* (2015), pois 6

[11] https://github.com/cs-au-dk/thor/wiki/Example-1:-Ordinary-Execution
[12] https://github.com/cs-au-dk/thor/wiki/Running:-Adding-new-apps

aplicações não conseguiram ser executadas na ferramenta devido a configurações especificas dos projetos. As informações detalhadas da execução das aplicações na ferramenta podem ser visualizadas na ***Tabela 18***. Na coluna “Execução” o X representa as aplicações que não foram possíveis serem executadas.

Tabela 18. Execução da ferramenta Thor com as aplicações do benchmark.

Projeto	Testes			Execução
	Passaram	Falhas	# Total	
com.better.alarm				✗
org.flyve.inventory.agent	1	0	1	
com.uploadedlobster.PwdHash	18	0	18	
de.danoeh.antennapod	0	170	170	
es.usc.citius.servando.calendula	13	0	13	
com.veniosg.dir				✗
com.llamacorp.equate	1	5	6	
org.gnucash.android	31	0	31	
fr.neamar.kiss	15	1	16	
org.xbmc.kore				✗
com.gunshippenguin.openflood				✗
monakhv.android.samlib	0	2	2	
org.openintents.shopping	0	7	7	
souch.smp	10	16	26	
com.vrem.wifianalyzer	1	2	3	
org.liberty.android.fantastischmemo	139	0	139	
org.andstatus.app				✗
org.openintents.filemanager				✗

Como resultado da Q_2, pode ser observado nas informações descritas acima que o *benchmark* poderia auxiliar na execução dos experimentos para avaliação de técnicas de teste de aplicações móveis. Com o *benchmark*, é possível obter informações relevantes, como os *scripts* de teste existentes necessários para a execução da ferramenta.

Com a ferramenta, cada caso de teste é exposto a condições em que certos eventos inesperados possam interferir na execução, resultando assim em mais falhas. Com isso, outra informação importante que o *benchmark* disponibiliza é a quantidade de testes que falharam com os *scripts* de teste existentes. Portanto, é possível comparar o resultado com a quantidade de testes que a ferramenta expos resultando na falha. Analisando os resultados das aplicações presentes no

benchmark, é possível verificar que a maioria das aplicações aumentou a quantidade de falhas utilizando a ferramenta.

4.2 Ameaças à Validade

Uma das principais ameaças à validade do trabalho é que foram analisadas apenas aplicações *open-source* disponibilizadas no GitHub e com *scripts* de teste presentes no projeto. Portanto, os resultados desta avaliação poderiam ser diferentes se fossem levados em consideração aplicações desenvolvidas por empresas privadas ou aplicações disponibilizadas fora da plataforma GitHub. Uma forma de minimizar a ameaça foi a utilização de aplicações que estavam presentes na Google Play.

Uma vez que foram analisadas apenas aplicações com *scripts* de teste, os resultados da avaliação não podem ser considerados para aplicações que não apresentem *scripts* de teste em seu projeto. O conjunto de aplicações móveis utilizado compromete a generalização dos resultados para a população em geral.

Foi proposto selecionar aplicações que já foram utilizadas em estudos anteriores de técnicas de teste de aplicações móveis. Isto foi importante para assumir que eles são representativos para a população de aplicações móveis utilizadas em experimentos de técnicas de testes de aplicações móveis.

As aplicações que compõem o *benchmark* passaram por um processo de seleção, apresentado na ***Seção 3.2***. Após isso foram realizados os estudos que visam a aplicabilidade das aplicações com relação a experimentos de técnicas de teste de aplicações móveis.

4.3 Considerações Finais

Neste capítulo, foi conduzida um estudo utilizando o trabalho de Usman *et al.* (2018) como prova de conceitos. O objetivo principal era verificar se o *benchmark* proposto por essa dissertação auxilia os pesquisadores no planejamento do experimento na etapa de seleção dos sujeitos. Segundo os resultados obtidos utilizando o trabalho Usman *et al.* (2018) verifica-se que poderia ser utilizado o *benchmark* na fase de selecionar os sujeitos. Das 18 aplicações apresentadas no *benchmark*, poderiam ser utilizadas 12 no trabalho de trabalho Usman *et al.* (2018) pois 6 aplicações não apresentam recursos de conectividade e sensores que são recursos necessários para a aplicação conseguir realizar o experimento.

Foi conduzida uma segunda avaliação utilizando o trabalho de Adamsen *et al.* (2015) que implementou a ferramenta Thor. O objetivo principal era verificar se o benchmark auxilia na execução de experimentos para avaliação de técnicas de teste de aplicações móveis. A característica mais importante para a aplicação fazer parte do conjunto de aplicações utilizados no trabalho de Adamsen *et al.* (2015) foi a existência de scripts de teste. Como resultado, observou-se que das 18 aplicações apresentadas no benchmark, poderiam ser utilizadas 12 no trabalho de trabalho

Adamsen *et al.* (2015) pois 6 aplicações não conseguiram ser executadas na ferramenta. Considerando as aplicações que não funcionaram no primeiro e segundo estudo, 4 (*com.gunshippenguin.openflood; org.openintents.filemanager; com.better.alarm; com.veniosg.dir*) delas não funcionam para os dois estudos.

No próximo capítulo serão apresentadas considerações finais e contribuições deste trabalho, assim como as perspectivas de trabalhos futuros para continuação desta pesquisa.

5. CONCLUSÕES

Neste capítulo serão apresentadas as conclusões deste trabalho de pesquisa juntamente com suas limitações e oportunidade de investigação e melhorias em experimentos de técnicas de teste de aplicações móveis.

5.1 Considerações Finais

A pesquisa em teste de aplicações móveis precisa de diretrizes para melhorar os processos de validação, por meio de experimentos, das técnicas propostas como resultados de pesquisas científicas. Tramontana *et al.* (2019) relatam que é muito difícil estabelecer um conjunto de dados usados para comparar o desempenho de ferramentas ou técnicas. Nesse contexto, o principal objetivo dessa dissertação é construir um *benchmark* composto por um conjunto de aplicações móveis, *scripts* de teste e métricas, a fim de auxiliar na fase de planejamento, na etapa de seleção dos sujeitos de um experimento para avaliar técnicas de teste no contexto de aplicações móveis.

Tendo em vista o cenário apresentado, esse trabalho de pesquisa seguiu um processo de seleção das aplicações para compor o *benchmark*, juntamente com as métricas específicas de cada aplicação utilizando o Prof.MApp. Os trabalhos de Silva *et al.* (2018) e Tramontana *et al.* (2019) foram utilizados como fonte base para a obtenção do conjunto de aplicações que compuseram o *benchmark*. Após a obtenção do conjunto inicial do *benchmark* foram realizadas duas avaliações. A primeira com o objetivo de verificar se o *benchmark* auxilia os pesquisadores no planejamento do experimento na etapa de seleção dos sujeitos e a segunda visando verificar se o *benchmark* auxilia na execução de experimentos para avaliação de técnicas de teste de aplicações móveis.

No decorrer desse trabalho de pesquisa também foram encontradas algumas dificuldades técnicas, como na etapa 3 do processo de seleção, pois nem todas as aplicações conseguiram ser executadas com sucesso por problemas com dependência.

Em geral, tais avaliações mostraram que o *benchmark* composto por aplicações, *scripts* de teste e métricas auxiliam os pesquisadores em experimentos de técnicas de teste de aplicações móveis.

5.2 Contribuições

As principais contribuições identificadas na condução dessa dissertação oferecidas à comunidade acadêmica são:

- Definição do processo de seleção das aplicações para compor o *benchmark*;
- Obtenção do conjunto de aplicações para compor o *benchmark*;
- Extração das métricas de cada aplicação que compõem o *benchmark* utilizando a ferramenta Prof.MApp;
- Categorização do conjunto de aplicações móveis, composta por *scripts* de teste e métricas de avaliação disponíveis no link https://goo.gl/PFDgw2;
- Apresentação dos resultados de duas investigações visando avaliar a utilização do *benchmark*:
  - Prova de conceitos que investigou o auxílio no planejamento do experimento na etapa de seleção dos sujeitos;
  - Estudo de caso visando verificar se o *benchmark* auxilia na execução de experimentos para avaliação de técnicas de teste de aplicações móveis

5.3 Limitações

As limitações identificadas desta pesquisa são apresentadas a seguir:

- Foram consideradas apenas aplicações Android *open-source* disponíveis na loja Google Play e nos repositórios do GitHub.
- O *benchmark* gerado tem a limitação de conter uma pequena amostra com 18 aplicações.
- O *benchmark* é composto apenas de aplicações com *scripts* de teste GUI. Os resultados da avaliação não podem ser considerados para aplicações que não apresentam *scripts* de teste GUI em seu projeto.

5.4 Trabalhos Futuros

As oportunidades de pesquisa identificadas para trabalhos futuros são apresentadas a seguir:

- Realizar estudos com um conjunto maior de aplicações;
- O estudo pode ser estendido para aplicações que não apresentam *scripts* de teste GUI;
- Estender o estudo para aplicações que não estão disponíveis na loja Google Play;
- Estender o estudo para aplicações que estejam disponíveis em outros repositórios fora do GitHub;
- Investigar e catalogar as falhas encontradas através dos *scripts* de teste disponibilizados;
- Conseguir realizar as etapas do processo descrito na ***Figura 8*** de forma automática;
- Realizar o processo descrito na ***Figura 8*** para contextos de aplicações iOS.

REFERÊNCIAS

Adamsen, C. Q., Mezzetti, G., & Møller, A. (2015). Systematic execution of android test suites in adverse conditions. *In Proceedings of the 2015 International Symposium on Software Testing and Analysis* (pp. 83-93).

Allix, K., Bissyandé, T. F., Klein, J., & Le Traon, Y. (2016). Androzoo: Collecting millions of android apps for the research community. *Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on*, (pp. 468-471). Austin.

Amalfitano, D., Fasolino, A. R., Tramontana, P., De Carmine, S., & Memon, A. M. (2012). Using GUI ripping for automated testing of Android applications. *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, (pp. 258-261).

Anand, S., Naik, M., Harrold, M. J., & Yang, H. (2012). Automated concolic testing of smartphone apps. *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, (p. 59).

Bhattacharya, P., Ulanova, L., Neamtiu, I., & Koduru, S. C. (2013). An empirical analysis of bug reports and bug fixing in open source android apps. *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, (pp. 133-143).

Brown, M. K., Yannes, Z., Lustig, M., Sanati, M., McKee, S. A., Tyson, G. S., & Reinhardt, S. K. (2016). Agave: A benchmark suite for exploring the complexities of the Android software stack. *Performance Analysis of Systems and Software (ISPASS), 2016 IEEE International Symposium on*, (pp. 157-158).

Butler, S.; Wermelinger, M.; Yu, Y.; Sharp, H. (2011). Mining java class naming conventions. In: Software Maintenance (ICSM), 2011 27th IEEE International Conference on. Williamsburg, VI: IEEE, 2011. p. 93–102. ISSN 1063-6773

Choi, W., Necula, G., & Sen, K. (2013). Guided gui testing of android apps with minimal restart and approximate learning. *Acm Sigplan Notices*, *48*, pp. 623-640.

Choudhary, S. R., Gorla, A., & Orso, A. (2015). Automated test input generation for android: Are we there yet?(e). *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on, (pp. 429-440).*

Dallmeier, V., Burger, M., Orth, T., & Zeller, A. (2013). WebMate: Generating test cases for web 2.0. *In International Conference on Software Quality* (pp. 55-69).

Delamaro, M., Jino, M., & Maldonado, J. (2017). *Introdução ao teste de software.* Elsevier Brasil.

Fernandes, T. S. (2015). Framework para estimar requisitos não funcionais em aplicações móveis. 115 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal do Rio Grande do Sul.

Fontão, A. L. (2016). MSECO-CERT: uma abordagem baseada em processo para apoiar a certificação de apps em ecossistema de software móvel. 185 f. Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

Fraser, G.; Arcuri, A. Sound empirical evidence in software testing. (2012). In: Proceedings of the 34th International Conference on Software Engineering. Piscataway, NJ, USA: IEEE Press. (ICSE '12), p. 178–188. ISBN 978-1-4673-1067-3.

Freire, M. A. (2015). Formalização de experimentos controlados em engenharia de software. 216 f. Tese (Doutorado em Sistemas e Computação) – Universidade Federal do Rio Grande do Norte.

Gao, J., Bai, X., Tsai, W.-T., & Uehara, T. (2014). Mobile application testing: a tutorial. *Computer, 47*, 46-55.

ISTQB, F. L. (2011). Foundation Level Syllabus Version 2011. *International Software Testing Qualifications Board*.

Kamran, M., Rashid, J., & Nisar, M. W. (2016). Android Fragmentation Classification, Causes, Problems and Solutions. *International Journal of Computer Science and Information Security, 14*, 992.

Kochhar, P. S., Thung, F., Nagappan, N., Zimmermann, T., & Lo, D. (2015). Understanding the test automation culture of app developers. *Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on*, (pp. 1-10).

Krutz, D. E., Mirakhorli, M., Malachowsky, S. A., Ruiz, A., Peterson, J., Filipski, A., & Smith, J. (2015). A dataset of open-source Android applications. *Proceedings of the 12th Working Conference on Mining Software Repositories*, (pp. 522-525).

Linares-Vásquez, M.; White, M.; Bernal-Cárdenas, C.; Moran, K.; Poshyvanyk, D. (2015). Mining android app usages for generating actionable gui-based execution scenarios. In: Proceedings of the 12th Working Conference on

Mining Software Repositories. Piscataway, NJ, USA: IEEE Press, 2015. (MSR ’15), p. 111–122.

Liu, Y.; Xu, C.; Cheung, S.-C. (2014). Characterizing and detecting performance bugs for smartphone applications. *In: Proceedings of the 36th International Conference on Software Engineering*. New York, NY, USA: ACM. (ICSE 2014), p. 1013–1024. ISBN 978-1-4503-2756-5

Machiry, A., Tahiliani, R., & Naik, M. (2013). Dynodroid: An input generation system for android apps. *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, (pp. 224-234).

Mahmood, R., Mirzaei, N., & Malek, S. (2014). Evodroid: Segmented evolutionary testing of android apps*. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 599-609).

Meireles, S. R. (2015). Evolução da ferramenta web guitar para geração automática de casos de teste de interface para aplicações web. 118 f. Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas

Mishra, A., Kanade, A., & Srikant, Y. N. (2016). Asynchrony-aware static analysis of Android applications. *Formal Methods and Models for System Design (MEMOCODE), 2016 ACM/IEEE International Conference on*, (pp. 163-172).

Moonsamy, V., Rong, J., & Liu, S. (2014). Mining permission patterns for contrasting clean and malicious android applications. *Future Generation Computer Systems, 36*, 122-132.

Myers, G.J. (2004) “The Art of Software Testing”. 2 ed., John Wiley & Sons, Nova Jersey.

Ossher, J.; Bajracharya, S.; Lopes, C. (2010). Automated dependency resolution for open source software. In: Mining Software Repositories (MSR), 2010 7th IEEEWorking Conference on. Cape, ZA: IEEE, 2010. p. 130–140.

Reis, R. d. (2016). MTCONTEXT: apoio à geração de casos de teste para aplicações móveis sensíveis ao contexto. 102 f. Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

Samuel, T., & Pfahl, D. (2016). Problems and solutions in mobile application testing. *International Conference on Product-Focused Software Process Improvement*, (pp. 249-267).

Santos, J. S. (2016). Insumos para a utilização do critério de teste baseado em erros para aplicações móveis. 112 f. Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

Silva, D. B. (2017). Caracterização de aplicações móveis do ponto de vista de Teste de Software. 110 f. Dissertação (Mestrado em Informática) - Universidade Federal do Paraná.

Silva, D. B., Eler, M. M., Durelli, V. H., & Endo, A. T. (2018). Characterizing mobile apps from a source and test code viewpoint. *Information and Software Technology, 101*, 32-50.

Sim, S. E., Easterbrook, S., & Holt, R. C. (2003). Using benchmarking to advance research: A challenge to software engineering. *Software Engineering, 2003. Proceedings. 25th International Conference on*, (pp. 74-83).

Souza, I. E., Inocêncio, A. C., Oliveira, P. H., Júnior, P. A., & Junior, E. L. (2015). TESE--Um Sistema de Informação para Gerenciamento de Projetos Experimentais em Engenharia de Software. *CEP, 75801*, 61.

Spínola, R. O., Dias-Neto, A. C., & Travassos, G. H. (2008). Abordagem para Desenvolver Tecnologia de Software com Apoio de Estudos Secundários e Primários. *Experimental Software Engineering Latin American Workshop (ESELAW).*

Syer, M. D., Nagappan, M., Hassan, A. E., & Adams, B. (2013). Revisiting prior empirical findings for mobile apps: An empirical case study on the 15 most popular open-source Android apps. *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, (pp. 283-297).

Tramontana, P., Amalfitano, D., Amatucci, N., & Fasolino, A. R. (2019). Automated functional testing of mobile applications: a systematic mapping study. *Software Quality Journal*, 27(1), 149-201.

Usman, A., Ibrahim, N., & Salihu, I. A. (2018). Test Case Generation from Android Mobile Applications Focusing on Context Events. *In Proceedings of the 2018 7th International Conference on Software and Computer Applications* (pp. 25-30).

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering.* Springer Science & Business Media.

Zein, S., Salleh, N., & Grundy, J. (2016). A systematic mapping study of mobile application testing techniques. *117*, 334-356.

Zhou, Y., & Jiang, X. (2012). Dissecting android malware: Characterization and evolution. *Security and Privacy (SP), 2012 IEEE Symposium on*, (pp. 95-109).

APÊNDICE A - EXPERIMENTOS SOBRE TESTE FUNCIONAL UTILIZANDO APLICAÇÕES MÓVEIS RETIRADOS DE ZEIN *ET AL.* (2016)

[1] Zhimin, W., Elbaum, S., Rosenblum, D.S., 2007. Automated Generation of Context-Aware Tests, Software Engineering, 2007. ICSE 2007. 29th International Conference on, pp. 406-415.

[2] Jiang, B., Long, X., Gao, X., 2007. MobileTest: A Tool Supporting Automatic Black Box Test for Software on Smart Mobile Devices, Automation of Software Test, 2007. AST '07. Second International Workshop on, pp. 8-8.

[3] She, S., Sivapalan, S., Warren, I., 2009. Hermes: A Tool for Testing Mobile Device Applications, Software Engineering Conference, 2009. ASWEC '09. Australian, pp. 121-130.

[4] Zhifang, L., Xiaopeng, G., Xiang, L., 2010a. Adaptive random testing of mobile application, Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, pp. V2-297-V292-301.

[5] Zhifang, L., Bin, L., Xiaopeng, G., 2010b. Test automation on mobile device, Proceedings of the 5th Workshop on Automation of Software Test. ACM, Cape Town, South Africa, pp. 1-7.

[6] Amalfitano, D., Fasolino, A.R., Tramontana, P., 2011. A GUI Crawling-Based Technique for Android Mobile Application Testing, Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on, pp. 252-261.

[7] Amalfitano, D., Fasolino, A.R., Tramontana, P., De Carmine, S., Memon, A.M., 2012. Using GUI ripping for automated testing of Android applications, Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on, pp. 258-261.

[8] Nagowah, L., Sowamber, G., 2012. A novel approach of automation testing on mobile devices, Computer & Information Science (ICCIS), 2012 International Conference on, pp. 924-930.

[9] Amalfitano, D., Fasolino, A.R., Tramontana, P., Amatucci, N., 2013. Considering Context Events in Event-Based Testing of Mobile Applications, Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on, pp. 126-133.

[10] Choi, W., Necula, G., Sen, K., 2013. Guided gui testing of android apps with minimal restart and approximate learning, ACM SIGPLAN Notices. ACM, pp. 623-640.

[11] Yu, L., Tsai, W.T., Jiang, Y., Gao, J., 2014. Generating test cases for context-aware applications using bigraphs, Software Security and Reliability (SERE), 2014 Eighth International Conference on. IEEE, pp. 137-146.

[12] Hu, G., Yuan, X., Tang, Y., Yang, J., 2014. Efficiently, effectively detecting mobile app bugs with AppDoctor, Proceedings of the Ninth European Conference on Computer Systems. ACM, p. 18.

[13] Amalfitano, D., Fasolino, A.R., Tramontana, P., Ta, B.D., Memon, A.M., 2015b. MobiGUITAR: Automated Model-Based Testing of Mobile Apps. IEEE Software 32, 53-59.

[14] Villanes, I.K., Bezerra Costa, E.A., Dias-Neto, A.C., 2015. Automated Mobile Testing as a Service (AM-TaaS), Services (SERVICES), 2015 IEEE World Congress on. IEEE, pp. 79-86.

[15] Wen, H.-L., Lin, C.-H., Hsieh, T.-H., Yang, C.-Z., 2015. PATS: A Parallel GUI Testing Framework for Android Applications, Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual. IEEE, pp. 210-215.

[16] Zhauniarovich, Y., Philippov, A., Gadyatskaya, O., Crispo, B., Massacci, F., 2015. Towards Black Box Testing of Android Apps, Availability, Reliability and Security (ARES), 2015 10th International Conference on. IEEE, pp. 501-510.

[17] Aktouf, O.E.K., Tao, Z., Gao, J., Uehara, T., 2015. Testing Location-Based Function Services for Mobile Applications, Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on, pp. 308-314.

[18] Vieira, V., Holl, K., Hassel, M., 2015. A context simulator as testing support for mobile apps, Proceedings of the 30th Annual ACM Symposium on Applied Computing. ACM, pp. 535-541.

[19] Hu, Y., Azim, T., Neamtiu, I., 2015. Versatile yet lightweight record-and-replay for Android, Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications. ACM, pp. 349-366.

[20] Adamsen, C.Q., Mezzetti, G., Møller, A., 2015. Systematic execution of Android test suites in adverse conditions, Proceedings of the 2015 International Symposium on Software Testing and Analysis. ACM, pp. 83-93.

[21] Kumar, A., Lee, S., Lee, W.J., 2015. Modeling and test case generation of inter-component communication in android, Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM International Conference on. IEEE, pp. 113-116.

APÊNDICE B - BENCHMARKS UTILIZANDO APLICAÇÕES MÓVEIS

Allix, K.; Bissyandé, T. F.; Klein, J.; Le Traon, Y., 2016. Androzoo: Collecting millions of android apps for the research community. IEEE. In: Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on. IEEE, p. 468-471.

Bhattacharya, P., Ulanova, L., Neamtiu, I., & Koduru, S. C. (2013). An empirical analysis of bug reports and bug fixing in open source android apps. *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, (pp. 133-143).

Brown, M. K., Yannes, Z.; Lustig, M., Sanati, M., Mckee, S. A., Tyson, G. S., Reinhardt, S. K., 2016. Agave: A benchmark suite for exploring the complexities of the Android software stack. In: Performance Analysis of Systems and Software (ISPASS), 2016 IEEE International Symposium on. IEEE, p. 157-158.

Butler, S.; Wermelinger, M.; Yu, Y.; Sharp, H. (2011). Mining java class naming conventions. In: Software Maintenance (ICSM), 2011 27th IEEE International Conference on. Williamsburg, VI: IEEE, 2011. p. 93–102. ISSN 1063-6773.

Choudhary, S. R., Gorla, A., Orso, A., 2015. Automated test input generation for android: Are we there yet?(e). In: Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on. IEEE, p. 429-440.

Fraser, G.; Arcuri, A. Sound empirical evidence in software testing. (2012). *In: Proceedings of the 34th International Conference on Software Engineering. Piscataway,* NJ, USA: IEEE Press. (ICSE '12), p. 178–188. ISBN 978-1-4673-1067-3.

Kochhar, P. S., Thung, F., Nagappan, N., Zimmermann, T., Lo, D., 2015. Understanding the test automation culture of app developers. In: Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on. IEEE, p. 1-10.

Krutz, D. E., Mirakhorli, M., Malachowsky, S. A., Ruiz, A., Peterson, J., Filipski, A., Smith, J., 2015. A dataset of open-source Android applications. In: Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on. IEEE, p. 522-525.

Linares-Vásquez, M.; White, M.; Bernal-Cárdenas, C.; Moran, K.; Poshyvanyk, D. (2015). Mining android app usages for generating actionable gui-based execution scenarios. In: Proceedings of the 12th Working Conference on

Mining Software Repositories. Piscataway, NJ, USA: IEEE Press, 2015. (MSR ’15), p. 111–122.

Liu, Y.; Xu, C.; Cheung, S.-C. (2014). Characterizing and detecting performance bugs for smartphone applications. *In: Proceedings of the 36th International Conference on Software Engineering*. New York, NY, USA: ACM. (ICSE 2014), p. 1013–1024. ISBN 978-1-4503-2756-5.

Mishra, A., Kanade, A., Srikant, Y. N., 2016. Asynchrony-aware static analysis of Android applications. In: Formal Methods and Models for System Design (MEMOCODE), 2016 ACM/IEEE International Conference on. IEEE, p. 163-172.

Moonsamy, V., Rong, J., Liu, S., 2014. Mining permission patterns for contrasting clean and malicious android applications. Future Generation Computer Systems, 36, 122-132.

Ossher, J.; Bajracharya, S.; Lopes, C. (2010). Automated dependency resolution for open source software. In: Mining Software Repositories (MSR), 2010 7th IEEEWorking Conference on. Cape, ZA: IEEE, 2010. p. 130–140.

Silva, D. B., Eler, M. M., Durelli, V. H., & Endo, A. T. (2018). Characterizing mobile apps from a source and test code viewpoint. Information and Software Technology, 101, 32-50.

Syer, M. D., Nagappan, M., Hassan, A. E., & Adams, B. (2013). Revisiting prior empirical findings for mobile apps: An empirical case study on the 15 most popular open-source Android apps. *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, (pp. 283-297).

APÊNDICE C – INFORMAÇÕES OBTIDAS DE CADA APLICAÇÃO

1. Aplicação: Simple Alarm Clock Free No Ads

- GitHub: https://github.com/yuriykulikov/AlarmClock
- Google Play: https://play.google.com/store/Apps/details?id=com.better.alarm

Informações Gerais	GitHub	yuriykulikov/AlarmClock
	Google Play	com.better.alarm
	Categoria	Produtividade
	Rating	4.2
	Instalação	1.000.000+
	minSDKVersion	15
	targetSDKVersion	21
	LOC	3.883
	LOC test	1.219
	Classe src	55
	Classe test	10
	Metodo src	15.246
	Metodo test	3.612
	# Teste	5
	% Teste	80%
	Cobertura	-
	Framework	Espresso
Conectividade		0
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	8
	# Elementos de eventos de interface	0
Sensores		0
Múltiplas Configurações	**# Configurações de tela**	4
	# Suporte a diferentes telas	4

2. Aplicação: Inventory Agent

- GitHub: https://github.com/flyve-mdm/android-inventory-agent
- Google Play: https://play.google.com/store/Apps/details?id=org.flyve.inventory.agent

Informações Gerais	GitHub	org.flyve.inventory.agent
	Google Play	flyve-mdm/android-inventory-agent
	Categoria	Ferramentas
	Rating	3.3
	Instalação	1.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	3.964
	LOC test	201
	Classe src	73
	Classe test	5
	Metodo src	14.868
	Metodo test	462
	# Teste	8
	% Teste	75%
	Cobertura	-
	Framework	Espresso
Conectividade		5
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	11
	# Elementos de eventos de interface	0
Sensores		6
Múltiplas Configurações	**# Configurações de tela**	1
	# Suporte a diferentes telas	4

3. Aplicação: Password Hash

- GitHub: https://github.com/phw/Android-PwdHash
- Google Play: https://play.google.com/store/Apps/details?id=com.uploadedlobster.PwdHash

Informações Gerais	GitHub	phw/Android-PwdHash
	Google Play	com.uploadedlobster.PwdHash
	Categoria	Ferramentas
	Rating	4.2
	Instalação	10.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	541
	LOC test	250
	Classe src	8
	Classe test	5
	Metodo src	1.386
	Metodo test	966
	# Teste	18
	% Teste	100%
	Cobertura	94%
	Framework	Espresso
Conectividade		0
GUIs	**# Elementos de layout**	1
	# Elementos de widgest	4
	# Elementos de eventos de interface	0
Sensores		0
Múltiplas Configurações	**# Configurações de tela**	6
	# Suporte a diferentes telas	4

4. Aplicação: AntennaPod

- GitHub: https://github.com/danieloeh/AntennaPod
- Google Play: https://play.google.com/store/Apps/details?id=de.danoeh.antennapod

Informações Gerais	GitHub	danieloeh/AntennaPod
	Google Play	de.danoeh.antennapod
	Categoria	Reproduzir e editar vídeos
	Rating	4.6
	Instalação	100.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	38.183
	LOC test	6.549
	Classe src	296
	Classe test	38
	Metodo src	116.634
	Metodo test	19.026
	# Teste	127
	% Teste	81%
	Cobertura	-
	Framework	Robotium
Conectividade		4
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	15
	# Elementos de eventos de interface	1
Sensores		6
Múltiplas Configurações	**# Configurações de tela**	6
	# Suporte a diferentes telas	5

5. Aplicação: Calendula

- GitHub: https://github.com/citiususc/calendula
- GooglePlay: https://play.google.com/store/Apps/details?id=es.usc.citius.servando.calendula

Informações Gerais	GitHub	citiususc/calendula
	Google Play	es.usc.citius.servando.calendula
	Categoria	Saúde e fitness
	Rating	4.4
	Instalação	1.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	25.495
	LOC test	753
	Classe src	214
	Classe test	12
	Metodo src	80.472
	Metodo test	1.428
	# Teste	13
	% Teste	100%
	Cobertura	27%
	Framework	Espresso
Conectividade		3
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	13
	# Elementos de eventos de interface	1
Sensores		2
Múltiplas Configurações	**# Configurações de tela**	5
	# Suporte a diferentes telas	5

6. Aplicação: Dir - Gerenciador de Arquivos

- GitHub: https://github.com/veniosg/Dir
- Google Play: https://play.google.com/store/Apps/details?id=com.veniosg.dir

Informações Gerais	GitHub	veniosg/Dir
	Google Play	com.veniosg.dir
	Categoria	Ferramentas
	Rating	4.3
	Instalação	10.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	8.839
	LOC test	1.778
	Classe src	98
	Classe test	24
	Metodo src	31.920
	Metodo test	7.686
	# Teste	54
	% Teste	5%
	Cobertura	3%
	Framework	Espresso
Conectividade		0
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	5
	# Elementos de eventos de interface	0
Sensores		0
Múltiplas Configurações	**# Configurações de tela**	2
	# Suporte a diferentes telas	4

7. Aplicação: Equate

- GitHub: https://github.com/EvanRespaut/Equate
- Google Play: https://play.google.com/store/Apps/details?id=com.llamacorp.equate

Informações Gerais	GitHub	EvanRespaut/Equate
	Google Play	com.llamacorp.equate
	Categoria	Ferramentas
	Rating	4.8
	Instalação	10.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	5.564
	LOC test	1.045
	Classe src	40
	Classe test	7
	Metodo src	18.228
	Metodo test	2.688
	# Teste	6
	% Teste	33%
	Cobertura	70%
	Framework	Espresso
Conectividade		3
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	3
	# Elementos de eventos de interface	0
Sensores		2
Múltiplas Configurações	**# Configurações de tela**	5
	# Suporte a diferentes telas	4

8. Aplicação: GnuCash

- GitHub: https://github.com/codinguser/gnucash-android
- Google Play: https://play.google.com/store/Apps/details?id=org.gnucash.android

Informações Gerais	GitHub	codinguser/gnucash-android
	Google Play	org.gnucash.android
	Categoria	Finanças
	Rating	4.4
	Instalação	100.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	22.979
	LOC test	4.818
	Classe src	149
	Classe test	39
	Metodo src	60.984
	Metodo test	11.844
	# Teste	8
	% Teste	87%
	Cobertura	-
	Framework	Espresso
Conectividade		3
GUIs	**# Elementos de layout**	4
	# Elementos de widgest	12
	# Elementos de eventos de interface	0
Sensores		2
Múltiplas Configurações	**# Configurações de tela**	7
	# Suporte a diferentes telas	4

9. Aplicação: KISS Launcher

- GitHub: https://github.com/Neamar/KISS
- Google Play: https://play.google.com/store/Apps/details?id=fr.neamar.kiss

Informações Gerais	GitHub	Neamar/KISS
	Google Play	fr.neamar.kiss
	Categoria	Personalização
	Rating	4.3
	Instalação	100.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	7.740
	LOC test	186
	Classe src	97
	Classe test	3
	Metodo src	22.722
	Metodo test	756
	# Teste	16
	% Teste	81%
	Cobertura	35%
	Framework	Espresso
Conectividade		0
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	4
	# Elementos de eventos de interface	1
Sensores		0
Múltiplas Configurações	**# Configurações de tela**	4
	# Suporte a diferentes telas	4

10. Aplicação: Kore, Official Remote for Kodi

- GitHub: https://github.com/xbmc/Kore
- Google Play: https://play.google.com/store/Apps/details?id=org.xbmc.kore

Informações Gerais	GitHub	xbmc/Kore
	Google Play	org.xbmc.kore
	Categoria	Reproduzir e editar vídeos
	Rating	4.4
	Instalação	1.000.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	28.636
	LOC test	6.148
	Classe src	173
	Classe test	54
	Metodo src	72.660
	Metodo test	19.152
	# Teste	115
	% Teste	97%
	Cobertura	23%
	Framework	Espresso
Conectividade		3
GUIs	**# Elementos de layout**	4
	# Elementos de widgest	12
	# Elementos de eventos de interface	1
Sensores		3
Múltiplas Configurações	**# Configurações de tela**	2
	# Suporte a diferentes telas	4

11. Aplicação: Open Flood

- GitHub: https://github.com/GunshipPenguin/open_flood
- Google Play: https://play.google.com/store/Apps/details?id=com.gunshippenguin.openflood

Informações Gerais	GitHub	GunshipPenguin/open_flood
	Google Play	com.gunshippenguin.openflood
	Categoria	Quebra-cabeça
	Rating	4.6
	Instalação	10.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	948
	LOC test	175
	Classe src	10
	Classe test	3
	Metodo src	2.604
	Metodo test	210
	# Teste	3
	% Teste	100%
	Cobertura	73%
	Framework	Espresso
Conectividade		0
GUIs	**# Elementos de layout**	1
	# Elementos de widgest	6
	# Elementos de eventos de interface	0
Sensores		0
Múltiplas Configurações	**# Configurações de tela**	0
	# Suporte a diferentes telas	4

12. Aplicação: SamLib Инфо

- GitHub: https://github.com/monakhv/samlib-Info
- Google Play: https://play.google.com/store/Apps/details?id=monakhv.android.samlib

Informações Gerais	GitHub	monakhv/samlib-Info
	Google Play	monakhv.android.samlib
	Categoria	Notícias e revistas
	Rating	4.5
	Instalação	10.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	18.361
	LOC test	720
	Classe src	172
	Classe test	11
	Metodo src	66.192
	Metodo test	2.058
	# Teste	4
	% Teste	50%
	Cobertura	-
	Framework	Espresso
Conectividade		3
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	9
	# Elementos de eventos de interface	0
Sensores		3
Múltiplas Configurações	**# Configurações de tela**	5
	# Suporte a diferentes telas	4

13. Aplicação: OI Shopping list

- GitHub: https://github.com/openintents/shoppinglist
- Google Play: https://play.google.com/store/Apps/details?id=org.openintents.shopping

Informações Gerais	GitHub	openintents/shoppinglist
	Google Play	org.openintents.shopping
	Categoria	Compras
	Rating	4.3
	Instalação	1.000.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	10.647
	LOC test	314
	Classe src	70
	Classe test	3
	Metodo src	22.470
	Metodo test	882
	# Teste	7
	% Teste	71%
	Cobertura	34%
	Framework	Espresso
Conectividade		1
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	11
	# Elementos de eventos de interface	0
Sensores		2
Múltiplas Configurações	**# Configurações de tela**	5
	# Suporte a diferentes telas	4

14. Aplicação: SicMu Player (tree view based music player)

- GitHub: https://github.com/souch/SMP
- Google Play: https://play.google.com/store/Apps/details?id=souch.smp

Informações Gerais	GitHub	souch/SMP
	Google Play	souch.smp
	Categoria	Música e áudio
	Rating	4.4
	Instalação	1.000+
	minSDKVersion	9
	targetSDKVersion	19
	LOC	3.468
	LOC test	455
	Classe src	20
	Classe test	5
	Metodo src	12.516
	Metodo test	3.024
	# Teste	30
	% Teste	46%
	Cobertura	5%
	Framework	Robotium
Conectividade		0
GUIs	**# Elementos de layout**	2
	# Elementos de widgest	4
	# Elementos de eventos de interface	1
Sensores		1
Múltiplas Configurações	**# Configurações de tela**	4
	# Suporte a diferentes telas	4

15. Aplicação: WiFiAnalyzer (open-source)

- GitHub: https://github.com/VREMSoftwareDevelopment/WifiAnalyzer
- Google Play: https://play.google.com/store/Apps/details?id=com.vrem.wifianalyzer

Informações Gerais	GitHub	VREMSoftwareDevelopment/WifiAnalyzer
	Google Play	com.vrem.wifianalyzer
	Categoria	Ferramentas
	Rating	4.3
	Instalação	1.000.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	7.588
	LOC test	9.659
	Classe src	145
	Classe test	130
	Metodo src	34.104
	Metodo test	39.354
	# Teste	3
	% Teste	66%
	Cobertura	73%
	Framework	Robotium
Conectividade		1
GUIs	**# Elementos de layout**	2
	# Elementos de widgest	8
	# Elementos de eventos de interface	0
Sensores		2
Múltiplas Configurações	**# Configurações de tela**	5
	# Suporte a diferentes telas	4

16. Aplicação: AnyMemo: Flash Card Study

- GitHub: https://github.com/helloworld1/AnyMemo
- Google Play: https://play.google.com/store/Apps/details?id=org.liberty.android.fantastischmemo

Informações Gerais	GitHub	helloworld1/AnyMemo
	Google Play	org.liberty.android.fantastischmemo
	Categoria	Educação
	Rating	4.5
	Instalação	100.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	20.504
	LOC test	2.953
	Classe src	198
	Classe test	38
	Metodo src	63.630
	Metodo test	8.736
	# Teste	150
	% Teste	98%
	Cobertura	28%
	Framework	Espresso
Conectividade		1
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	14
	# Elementos de eventos de interface	0
Sensores		2
Múltiplas Configurações	**# Configurações de tela**	5
	# Suporte a diferentes telas	1

17. Aplicação: AndStatus

- GitHub: https://github.com/andstatus/andstatus
- Google Play: https://play.google.com/store/Apps/details?id=org.andstatus.app

Informações Gerais	GitHub	andstatus/andstatus
	Google Play	org.andstatus.app
	Categoria	Social
	Rating	4.1
	Instalação	5.000+
	minSDKVersion	-
	targetSDKVersion	-
	LOC	49.346
	LOC test	9.276
	Classe src	400
	Classe test	99
	Metodo src	172.914
	Metodo test	24.486
	# Teste	1
	% Teste	0%
	Cobertura	-
	Framework	Espresso
Conectividade		3
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	9
	# Elementos de eventos de interface	1
Sensores		2
Múltiplas Configurações	**# Configurações de tela**	4
	# Suporte a diferentes telas	5

18. Aplicação: OI File Manager

- GitHub: https://github.com/openintents/filemanager
- Google Play: https://play.google.com/store/Apps/details?id=org.openintents.filemanager

Informações Gerais	GitHub	openintents/filemanager
	Google Play	org.openintents.filemanager
	Categoria	Produtividade
	Rating	4.2
	Instalação	5.000.000+
	minSDKVersion	2
	targetSDKVersion	19
	LOC	6.780
	LOC test	1.391
	Classe src	65
	Classe test	12
	Metodo src	22.512
	Metodo test	3.948
	# Teste	7
	% Teste	42%
	Cobertura	23%
	Framework	Espresso/Robotium
Conectividade		0
GUIs	**# Elementos de layout**	3
	# Elementos de widgest	7
	# Elementos de eventos de interface	1
Sensores		0
Múltiplas Configurações	**# Configurações de tela**	5
	# Suporte a diferentes telas	4