UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM

INSTITUTO DE COMPUTAÇÃO- ICOMP

PROGRAMA PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

DeSeCT: uma estratégia heurística usando atributos de aplicações móveis na seleção de dispositivos para testes de compatibilidade

Isabel Karina Villanes Rojas

Manaus - AM

Novembro, 2022

Isabel Karina Villanes Rojas

DeSeCT: uma estratégia heurística usando atributos de aplicações móveis na seleção de dispositivos para testes de compatibilidade

Tese de doutorado apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas, como requisito parcial para a obtenção do título de Doutor em Informática.

Orientador

Profa. Dra. Rosiane de Freitas Rodrigues

Coorientador Prof. Dr. Arilo Claudio Dias Neto

Universidade Federal do Amazonas - UFAM

Instituto de Computação- IComp

Manaus - AM

Novembro, 2022

Ficha Catalográfica

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

Villanes Rojas, Isabel Karina
V716d DeSeCT : uma estratégia heurística usando atributos de aplicações móveis na seleção de dispositivos para testes de compatibilidade / Isabel Karina Villanes Rojas . 2022
142 f.: il. color; 31 cm.

Orientadora: Rosiane de Freitas Rodrigues
Coorientador: Arilo Claudio Dias Neto
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. Teste de aplicações móveis. 2. Teste de compatibilidade. 3. Características de aplicações móveis. 4. Seleção de dispositivos. 5. Algoritmo genético. I. Rodrigues, Rosiane de Freitas. II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Usando atributos das app móveis para o Teste de Compatibilidade"

ISABEL KARINA VILLANES ROJA

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos Professores:

PROF. ARILO CLAÚDIO DIAS NETO - MEMBRO INTERNO - Presidente

PROF. JUAN GABRIEL COLONNA - MEMBRO INTERNO

PROF. EDUARDO JAMES PEREIRA SOUTO - MEMBRO INTERNO

Dr. ANDRÉ TAKESHI ENDO - MEMBRO EXTERNO

Dr. EDUARDO NORONHA DE ANDRADE FREITAS - MEMBRO EXTERNO

Documento assinado digitalmente
gov.br
EDUARDO NORONHA DE ANDRADE FREITAS
Data: 12/12/2022 11:42:55-0300
Verifique em https://verificador.iti.br

Manaus, 09 de novembro de 2022

A Deus, meus pais e toda minha família que sempre acreditaram em mim.

Agradecimentos

A Deus por ser a minha luz o tempo todo, por cuidar de minha saúde e ser o meu suporte, sem Ele nada teria sido possível.

A meu noivo César, que foi meu suporte desde o início desta caminhada acadêmica, por sua compreensão em todos os momentos que tive que estar ausente e por ter me ajudado sempre que precisei.

A minha mãe, Yolanda Salome, que sempre me incentivou a colocar os estudos em primeiro lugar, mesmo estando distante, sempre foi um exemplo de força e coragem, por meio de sua grande devoção e suas orações.

Ao meu pai, José Luis por ter sido vigilante de minha educação desde pequena, incentivando-me a enfrentar as adversidades e aprender com elas, e também aproveitar as oportunidades que a vida proporciona.

As minhas irmãs, Erika, Roxana e Cinthia, por serem amigas, companheiras, confidentes e suporte em todo momento, ainda na distância, sempre estiverem presentes e por sempre me incentivarem a continuar. Erika é a pessoa que está aí para te ouvir sempre. Roxana é clara e direta, obrigada irmã por cuidar de nossos pais. Cinthia ela está sempre que alguém precisa de ajuda. Minhã admiração e muito obrigada a cada uma de vocês.

Ao meu irmão Alan, por ser amigo e sempre direto e incentivando a alcançar nossos objetivos, por ser um excelente exemplo de superação e persistência, obrigada por cuidar de toda a família.

Ao professor Arilo Claudio por ter me guiado nesta caminhada, fazendo-me pensar fora da caixa na busca de soluções a minha pesquisa, e por ter me apresentado

ao teste de software, uma área, que era desconhecida para mim, desde que iniciei o mestrado e continuo com o doutorado. Muito obrigada por ajudar-me a ver uma variedade caminhos nessa área que ainda tem muito a ser explorada.

Ao professor André Endo pelas inúmeras reuniões e sugestões para o desenvolvimento desta pesquisa, meu imenso agradecimento por todas as horas de reuniões sempre com ideias, sugestões e pela colaboração na escrita dos artigos.

A professora Rosiane Freitas, pela sua colaboração na defesa desta pesquisa.

Aos meus amigos e colegas Awdren, Silvia e Josias, pela amizade e parceria em todas as revisões e colaboração, sou muito feliz com a amizade deles, pois os considero parte da minha família brasileira.

A todos os membros que passaram pelo grupo de pesquisa ExperTS, que contribuíram com revisões, ideias, discussões, participação de estudos, etc. A cada um deles minha imensa gratitude.

A todos os colegas dos outros grupos de pesquisa do IComp, que sempre se disponibilizaram para ajudar quando tinha alguma dúvida em temas que não dominava.

A todos os participantes que fizeram parte dos estudos experimentais, obrigada pelo tempo disponibilizado e contribuição.

Aos professores e técnicos administrativos do Instituto de Computação pelo suporte nos laboratórios.

A CAPES pelo apoio financeiro ao longo do doutorado e todos que colaboraram com essa pesquisa.

Finalmente, e não menos importante, a Milo, meu pet, foi meu suporte emocional nos tempos de pandemia e companheiro nos dias que precisei ficar ate tarde ele ficava junto.

DeSeCT: uma estratégia heurística usando atributos de aplicações móveis na seleção de dispositivos para testes de compatibilidade

Autor: Isabel Karina Villanes Rojas
Orientador: Profa. Dra. Rosiane de Freitas Rodrigues
Coorientador: Prof. Dr. Arilo Claudio Dias Neto

Resumo

Garantir a qualidade das aplicações móveis e atingir uma maior cobertura de mercado de dispositivos está cada vez mais necessário, uma vez que existe uma diversidade de dispositivos móveis, marcas e versões de sistema operacional. A execução dos testes de compatibilidade é precedida pela seleção dos dispositivos onde o aplicativo será testado. Selecionar os dispositivos nos quais serão executados os testes é uma tarefa importante, porém decidir quantos e quais serão usados para a execução dos testes, diante de milhares de modelos de dispositivos, não é uma tarefa trivial. As abordagens existentes estão baseadas principalmente nas características dos dispositivos, assim são desconsideradas as características do aplicativo que está sendo testado. Diante disso, neste trabalho é proposto o DeSeCT (do inglês, Devices Selection for Compatibility Testing) Seleção de dispositivos para testes de compatibilidade. O DeSeCT está baseado no algoritmo NSGA-II e usa informações dos dispositivos e atributos do aplicativo que está sendo testado visando minimizar o número e custo dos dispositivos selecionados e maximizar a cobertura de características como o tamanho de tela, resolução, rede, versão de api e marketshare; também foi construído um dataset com 4039 dispositivos móveis de marcas a nível mundial. Os resultados do primeiro estudo com 30 aplicativos

permitiram identificar a importância no uso dos atributos do aplicativo, o que permitiu melhorar o espaço de busca e ter soluções com menos dispositivos e alta cobertura de características de dispositivos. Para identificar o estado da prática, foi realizado um Survey, que permitiu identificar os três critérios mais usados pelos profissionais da indústria no processo de seleção de dispositivos, sendo: marketshare, tamanho de tela, e versão do SO. Outros fatores que impactam na decisão são a localização, e marcas conhecidas. DeSeCT apresenta um conjunto de soluções com a relação entre o custo da solução e sua cobertura de características de dispositivos, visando auxiliar na tomada de decisão no processo de seleção de dispositivos móveis.

Palavras-chave: teste de aplicações móveis; teste de compatibilidade; fragmentação Android; características de aplicações móveis; seleção de dispositivos; algoritmo genético.

DeSeCT: uma estratégia heurística usando atributos de aplicações móveis na seleção de dispositivos para testes de compatibilidade

Autor: Isabel Karina Villanes Rojas

Orientador: Profa. Dra. Rosiane de Freitas Rodrigues

Coorientador: Prof. Dr. Arilo Claudio Dias Neto

Abstract

Ensuring the quality of mobile applications and achieving greater device market coverage is increasingly necessary, as there is a diversity of mobile devices, brands and operating system versions. The execution of compatibility tests is preceded by the selection of devices on which the application will be tested. Selecting devices on which the tests will be performed is an important task, but deciding how many and which ones will be used for the test execution, in the face of thousands of device models, is not a trivial task. Existing approaches are mainly based on device characteristics, thus disregarding the characteristics of the application being tested. Therefore, in this work, DeSeCT (Devices Selection for Compatibility Testing) is proposed. This proposal is based on the NSGA-II algorithm and uses device information and attributes of the application being tested in order to minimize the number and cost of selected devices and maximize the coverage of characteristics such as screen size, resolution, network, version api and marketshare. A device dataset was also built with 4039 mobile devices from brands worldwide. The results of the first study with 30 applications made it possible to identify the importance of using the attributes of the application, which allowed us to improve the search space and have solutions with fewer devices and high

coverage of device characteristics. To identify the state-of-practice, a Survey was carried out, which allowed us to identify the three criteria most used by industry professionals in the device selection process, which are: marketshare, screen size, and OS version. Other factors that impact the decision are location and known brands. DeSeCT presents a set of solutions with the relationship between the cost of the solution and its coverage of device features, aiming to assist in decision-making in the mobile device selection process.

Keywords: mobile testing; compatibility testing; Android fragmentation; mobile App characteristics; select devices; genetic algorithm.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1

INTRODUÇÃO

Neste capítulo será apresentado o contexto do trabalho e o que motivou esta pesquisa. São também apresentados os seus objetivos, a hipótese, a metodologia de pesquisa adotada e a organização deste texto.

1.1 Contextualização e motivação: descrição do problema

O alto uso de dispositivos móveis observado nos últimos anos resulta na demanda de novas aplicações móveis (*apps*) que ajudem ou facilitem nas tarefas diárias dos usuários. Pesquisas divulgadas por Statista (2021a) indicam que, mundialmente, o número de usuários de *apps* deverá chegar até 7,49 bilhões até o fim de 2025. Em 2021, os usuários fizeram *download* de 144 bilhões de *apps* em diferentes dispositivos (ex. *smartphones, tablets e wearables*) nas lojas *Google Play Store* e *Apple Store*. Estima-se que em 2025 esse número deva crescer para 173 bilhões de *downloads* de *apps* (Statista, 2022a).

É inegável que a complexidade das tarefas realizadas pelos *apps* está aumentando, e a procura por *apps* que consigam resolver um problema real da vida do usuário, no contexto em que o usuário se encontra, é outro fator que influencia no seu uso e *download*. As informações usadas nas *apps* possuem um alto impacto em relação à segurança dos dados pessoais dos usuários. Por exemplo, *apps* que realizam compras *online* usando os dados de crédito do usuário usam informações confidenciais que devem ser tratadas com o maior cuidado possível. Para garantir essa qualidade da aplicação, é necessário que este passe por um conjunto de testes visando oferecer

uma melhor qualidade do produto e possibilitar a garantia de segurança aos usuários durante a sua utilização. No entanto, testes em *apps* são mais complexos de serem executados do que em *software* convencional, pois se espera que as *apps* sejam testados em diferentes dispositivos com diferentes plataformas, versões de sistemas operacionais (SO) e diferentes recursos, como: tamanhos de tela, resolução de câmera e conexões de rede. Isso é conhecido pelos desenvolvedores como o problema da fragmentação (JOORABCHI; MESBAH; KRUCHTEN, 2013; KIRUBAKARAN; KARTHIKEYANI, 2013; HAM; PARK, 2011; KHALID et al., 2014).

Segundo o dicionário[1], fragmentação é o processo ou estado de quebra. No contexto de dispositivo móvel, entende-se como a diversidade de versões do sistema operacional oferecidas por diferentes marcas de dispositivos móveis. Este problema torna-se mais perceptível na plataforma *Android*, devido à diversidade de empresas (ex. *Samsung, Google, LG, HTC, Motorola e Blackberry*) que usam esta plataforma como SO de seus dispositivos móveis. *Android* é a plataforma de dispositivos móveis mais usada no mundo. Até agosto de 2022, o *Android* manteve a sua posição de SO móvel líder mundial, controlando o mercado com 71,47% (Statista, 2022c).

Dado esse cenário, testar as *apps* em um grande número de dispositivos é caro e ineficiente (KHALID et al., 2014; NAITH; CIRAVEGNA, 2018b), visto que isso requer mais tempo, recursos humanos e financeiros, o que aumenta o custo do teste (HAM; PARK, 2011; VILKOMIR; AMSTUTZ, 2014). Para garantir a qualidade das *apps*, são realizados diversos tipos de teste, seja teste funcional ou não-funcional. Enquanto os testes funcionais procuram testar "o que" um *app* deve fazer, o teste não-funcional olha o "quão bem" um *app* funciona. Um dos tipos de testes não-funcionais é o teste de compatibilidade (em inglês, *Mobile Compatibility Testing*), que visa realizar testes em diferentes ambientes (ex. diversidade de dispositivos ou versões de sistema operacional). Os pesquisadores sugerem que o teste de compatibilidade é uma forma de auxiliar a contornar o problema da fragmentação (CHENG et al., 2015).

Antes da execução dos testes de compatibilidade (ou outros testes) é necessário decidir quais dispositivos móveis serão selecionados para o teste (VILKOMIR et al., 2015). Além disso, decidir quais e quantos dispositivos serão necessários para os testes

[1] https://dicionario.priberam.org/fragmentação

é uma tarefa importante (VILKOMIR, 2018), pois uma seleção abaixo do ideal pode causar desperdício de dinheiro, esforço humano e prejudicar a qualidade do produto final, deixando de encontrar erros (*bugs*) ou até perder receita (LU et al., 2016).

A seleção de dispositivos móveis para testes de compatibilidade não é uma tarefa trivial, considerando que somente em 2015 havia mais de 24 mil modelos de dispositivos diferentes (OPENSIGNAL, 2015) e que até 2019 havia 2,5[2] bilhões de dispositivos *Android* ativos no mundo. Este problema é bem compreendido inclusive na indústria. Por exemplo, a *Salesforce*, uma empresa de gerenciamento de relacionamento com o cliente, anunciou em 2016, que não oferecerá mais suporte para todos os dispositivos móveis, devido ao problema da fragmentação; desde então, *Salesforce*[3] têm recomendações de dispositivos onde são realizados os testes.

Como é percebido, os desenvolvedores ou uma organização que espera alcançar um grande número de usuários para suas *apps*, devem ter como alvo o funcionamento adequado de sua *app* em uma grande variedade de dispositivos, obtendo uma boa cobertura deles, seja pela marca, tamanhos de tela, recursos, etc. Essas características são definidas de acordo com o objetivo que os desenvolvedores ou a organização querem alcançar. Portanto, a seleção de dispositivos é sim uma atividade importante, principalmente para as *apps* que buscam um maior número de usuários. Pois, tendo um número maior de usuários, é muito provável ter uma maior variedade de dispositivos.

No geral, a seleção de dispositivos móveis para a realização de testes de compatibilidade pode ocorrer da seguinte forma: (1) usar os dispositivos que estejam no alcance das mãos ou baseado no seu ambiente de teste (VILKOMIR et al., 2015); (2) pagar um serviço na nuvem e ter disponibilidade de dispositivos para testar; ou (3) adquirir os dispositivos necessários para fazer os testes. Em qualquer opção, os testadores precisam decidir quais dispositivos serão os selecionados. Na maioria dos casos, os dispositivos selecionados são os mais populares do momento (VILKOMIR; AMSTUTZ, 2014), porém não se tem evidência empírica sobre qual seria a melhor opção.

A seleção de dispositivos não somente gera impacto na execução dos testes, mas também impacta na qualidade dos casos de teste que são automatizados. Uma atividade

[2] https://twitter.com/Android/status/1125822326183014401
[3] https://help.salesforce.com/articleView?id=salesforce_app_requirements.htm

a ser realizada após a seleção de dispositivos é a criação dos casos de teste e os *scripts* de execução. Nessas fases, deverão ser consideradas as informações ou características de dispositivos (ex. diversidade de SO, resoluções, tamanhos de tela, etc.) para a criação dos testes, principalmente nos casos de teste que são automatizados, pois os testes (ou *scripts*) serão executados nesses dispositivos. É preciso ter ciência de que é possível que a *app* não funcione em um dispositivo não selecionado para os testes; consequentemente, não se perde um só usuário e sim uma base de usuários que possuam aquele modelo de dispositivo. Com base nessas necessidades e tendências, torna oportuna a proposta de uma abordagem de apoio para a tomada de decisões na escolha de dispositivos móveis para teste de compatibilidade.

1.2 Hipótese

A partir do contexto e o problema apresentado na seção anterior e a carência de abordagens na área de teste de compatibilidade para *apps* móveis, especificamente no processo de seleção de dispositivos para teste, foi definida a seguinte hipótese a ser investigada:

"A seleção de dispositivos para teste de compatibilidade impacta positivamente na qualidade dos testes, em relação à cobertura de mercado e a identificação de falhas em dispositivos específicos"

Para esta pesquisa, a qualidade dos testes para *apps* móveis está relacionada à cobertura de mercado em relação às características da *app* e dos dispositivos selecionados para os testes.

1.3 Questão de pesquisa

A partir do que foi apresentado na seção anterior, esta pesquisa tem como objetivo investigar e responder à seguinte questão:

"O uso das informações/atributos das apps móveis impacta positivamente na seleção de dispositivos para teste de compatibilidade?"

Para entender e responder à questão de pesquisa principal, outras subquestões precisam ser respondidas no decorrer dos estudos realizados nesta tese:

1. O que diz a literatura técnica com respeito ao teste de compatibilidade em *apps* móveis?

2. O que diz a literatura técnica com respeito à seleção de dispositivos móveis para teste?

3. Qual é a abordagem e os critérios usados para a seleção de dispositivos de teste na indústria?

As subquestões de pesquisa 1 e 2 focam em entender na literatura técnica quais são as abordagens para teste de compatibilidade e sua aplicação no processo de seleção de dispositivos para teste de *apps* móveis. A subquestão 3 pretende entender na indústria como é realizado este processo e quais são os critérios usados.

1.4 Objetivos

Desenvolver e avaliar uma abordagem para apoiar a tomada de decisão na seleção de dispositivos móveis, por meio do uso de heurísticas que utilizem características de dispositivos e aplicações móveis, com a finalidade de auxiliar o teste de compatibilidade de *apps*, contribuindo assim com a qualidade dos testes em relação à cobertura de mercado e com a identificação de possíveis falhas em dispositivos específicos (*do inglês, device-faults*). Para atingir o objetivo geral desta pesquisa, pretende-se alcançar os seguintes objetivos secundários:

- Identificação, comparação e avaliação das heurísticas usadas para a seleção de dispositivos móveis dentro do teste de compatibilidade;

- Identificação de taxonomias para identificar características relevantes dos dispositivos móveis e das *apps*, que podem contribuir com a abordagem proposta.

- Elaboração e disponibilização de uma abordagem de apoio a tomada de decisões para a seleção dos dispositivos móveis para teste de compatibilidade de *apps*.

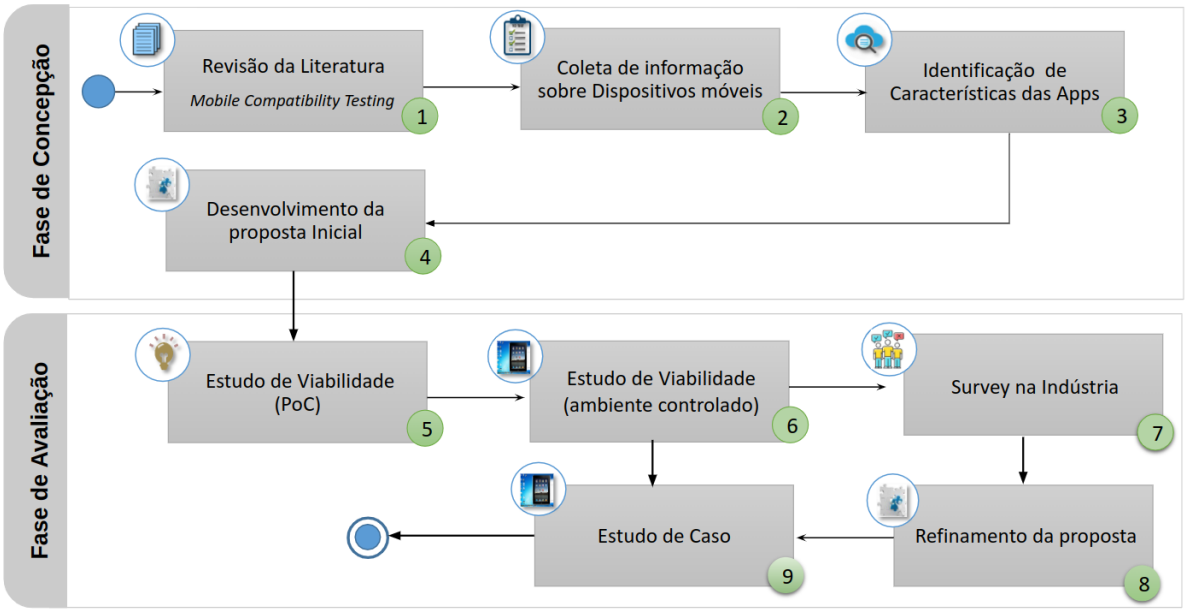


Figura 1 – Metodologia de pesquisa.

- Elaboração e disponibilização de experimentos para a avaliação da abordagem desenvolvida, visando o amadurecimento da abordagem.

1.5 Metodologia de Pesquisa

A metodologia de pesquisa utilizada neste trabalho é composta por duas fases: concepção, onde o foco é a definição e construção da abordagem, e; avaliação, focada em avaliar a viabilidade da abordagem. Essas fases são apresentadas na Figura 1.

1.5.1 Fase de Concepção

- (1) Revisão da literatura: será executado em duas fases por meio do mapeamento de Literatura Multivocal *(do inglês Multivocal Literature Mapping MLM)*. A primeira fase está relacionada ao Teste de Compatibilidade, seguindo as guias de Wohlin (2014) para revisão da literatura técnica usando o método *Snowballing*; e a segunda fase será a revisão da literatura cinza. O protocolo definido, os procedimentos de execução e a análise dos resultados podem ser vistos no Capítulo 3;

- (2) Coleta de informação sobre dispositivos móveis: será abordado um estudo de

como obter e identificar as características mais atuais dos dispositivos que serão usados na abordagem; a implementação e uso são descritos no Capítulo 4.

- (3) Identificação de características das *apps*: consistirá num estudo sobre as características das *apps*, definindo uma ferramenta para ajudar na extração dessas informações; a implementação e uso estão descritos no Capítulo 4;

- (4) Desenvolvimento da abordagem: Uma visão geral da abordagem, assim como os detalhes da implementação e evolução são apresentados na Seção 4.1;

1.5.2 Fase de Avaliação

Para a avaliação da abordagem foram necessários diversos estudos experimentais que permitiram evoluí-la, visando caracterizar a abordagem proposta em relação ao objetivo desta pesquisa. Seguindo a metodologia definida por Shull, Carver e Travassos (2001), os estudos que foram usados para essa pesquisa são os seguintes:

- (5) Estudo de Viabilidade: uma prova de conceito foi planejada e executada com o objetivo de avaliar a viabilidade inicial da abordagem em um escopo menor (uma aplicação);

- (6) Estudo de Viabilidade: um estudo em escala maior num ambiente controlado foi necessário com o objetivo de confirmar a aplicabilidade do estudo anterior em maior escala (30 aplicações móveis);

- (7) *Survey* na indústria: foi necessário obter informações com profissionais da indústria sobre os critérios de seleção de dispositivos usados nos testes. O objetivo foi verificar se a abordagem era adequada para as necessidades da indústria. Os resultados permitiram evoluir a abordagem;

- (8) Refinamento da abordagem: os resultados de cada estudo permitiu a evolução da abordagem; o objetivo desta etapa foi concentrada na evolução da abordagem;

- (9) Estudo de caso: A abordagem na sua versão final foi avaliada em um conjunto de *apps*. O objetivo desta etapa foi verificar o comportamento da abordagem em diversas *apps* com todos os refinamentos implantados na etapa anterior.

1.6 Estrutura do Documento

Este trabalho está organizado em sete capítulos, sendo este o primeiro capítulo de introdução, que apresentou a motivação, problema, objetivos e metodologia e o contexto da pesquisa. A organização do texto deste trabalho segue a estrutura abaixo:

Capítulo 2 - "Referencial Teórico": descreve os principais conceitos relacionados à pesquisa: aplicação móvel, teste de *apps* móveis, tipos de teste, teste de compatibilidade, Plataforma *Android*, fragmentação, engenharia de software baseada em busca, e algoritmos genéticos.

Capítulo 3 - "Mapeamento sistemático sobre teste de compatibilidade para *apps* móveis": Apresenta o corpo de conhecimento sobre as diferentes abordagens usadas na literatura relacionadas ao teste de compatibilidade por meio de um Mapeamento de Literatura Multivocal.

Capítulo 4 - "DeSeCT: Seleção de dispositivos para Testes de Compatibilidade em aplicações móveis": Descreve a visão geral da abordagem e os detalhes dos módulos que compõem a *DeSeCT*.

Capítulo 5 - "Estudo de caso e refinamento da abordagem DeSeCT": Apresenta os resultados de um estudo em maior escala para confirmar a viabilidade da proposta e os resultados com uma organização na indústria em relação ao processo de seleção de dispositivos. Com os resultados obtidos descreve-se a evolução da abordagem.

Capítulo 6 - "*Survey* e experimento com Profissionais na indústria": Descreve os resultados de um *survey* aplicados em profissionais da indústria e os resultados de um experimento para identificar os critérios que são usados pelos profissionais na seleção de dispositivos móveis. Finalmente, apresenta-se os detalhes da evolução da abordagem por meio de um estudo de caso.

Capítulo 7 - "Considerações Finais": Neste capítulo são apresentados as conclu-

sões gerais da tese, os resultados obtidos e as contribuições já geradas, as planejadas e os trabalhos futuros.

2

REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os conceitos relacionados ao desenvolvimento da pesquisa, a fundamentação teórica que embasa este trabalho e as pesquisas correlatas encontradas na literatura.

2.1 Aplicação Móvel

A aplicação móvel, ou *app*, é definida como um software independente projetado para um dispositivo móvel que executa tarefas específicas para usuários (AMALFITANO et al., 2013)). Ela pode ser instalada e executada em uma diversidade de dispositivos móveis (MUCCINI; FRANCESCO; ESPOSITO, 2012). A evolução das *apps* em relação à complexidade tem sido cada vez mais notória, indo desde *apps* para uso recreativo ou jogos até os mais críticos negócios.

As aplicações podem estar pré-instaladas nos dispositivos móveis ou os usuários podem baixar das lojas de aplicações. Existem três tipos de aplicações: nativa, web e híbrida.

- *(1) App nativa* são específicos para um sistema operacional, são estruturadas para aproveitar ao máximo os recursos do dispositivo como a câmera, lista de contatos, GPS, dentre outros.

- *(2) App web* é considerado uma página web responsiva que é acessada pelo navegador de um dispositivo móvel. A *app* web não precisa ser instalada no dispositivo,

porém precisa de acesso à internet.

- *(3) App Híbrida* é parcialmente nativa e parcialmente *web app*, este tipo de *app* também precisa ser baixada das lojas de *apps* (Google Play ou Apple Store); pela sua natureza esses *apps* podem ser baseadas em HTML5 e exibidas através de um navegador embutido na aplicação.

Esta pesquisa pode ser aplicada a qualquer tipo de aplicação, devido ser usada a *app* no seu estado final (pacote *apk* no caso da plataforma Android) e as informações podem ser obtidas na loja de *apps* ou no pacote *apk*.

2.2 Teste de aplicações móveis

O termo teste de aplicações móveis (do inglês *Mobile App Testing*) refere-se à atividade de teste em aplicações nativas, web ou híbridas em dispositivos móveis usando métodos e ferramentas de teste de software bem definidos para garantir a qualidade da *app* quanto à sua funcionalidade, usabilidade e consistência. (CHENG et al., 2015).

Os testes são necessários para garantir o funcionamento desejado em relação à qualidade do serviço (QoS), mobilidade, usabilidade, interoperabilidade, conectividade móvel, segurança e privacidade (AKOUR; AL-ZYOUD, 2016). As *apps* estão ficando cada vez mais complexas. Quando surgiram, eram usadas apenas para atividades de entretenimento, e com o tempo passaram a domínios mais críticos, como saúde e finanças, tornando assim uma necessidade testar sua estabilidade e robustez.

No mercado de dispositivos móveis amplamente diverso e um ciclo de desenvolvimento mais curto, conhecer e aplicar os tipos e técnicas de teste se faz essencial para garantir a qualidade da *app*.

2.2.1 Tipos de teste

No teste de software, é necessário responder a três perguntas: Quando testar?; O que testar? e Como testar? como mostrado na Figura 2. Os tipos de teste responde a segunda

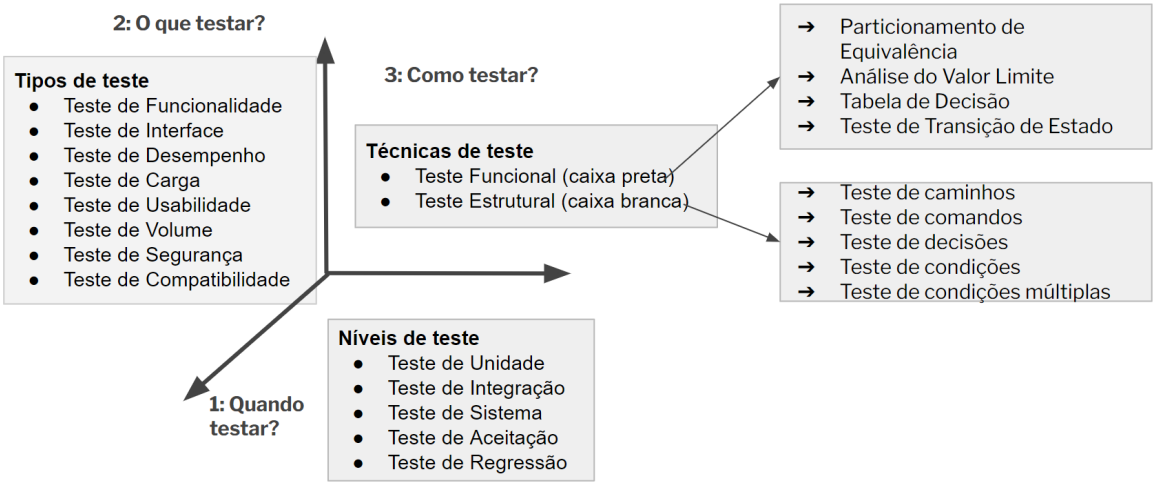


Figura 2 – Tipos, níveis e técnicas de teste.

pergunta *"O que testar?"* Dentre os diversos tipos de teste existentes, para esta pesquisa foram divididos em dois tipos de teste: teste funcional e teste não-funcional.

O teste funcional pode ser executado dentro de vários níveis de teste (unitário, integração, sistema, etc); este tipo de teste avalia as funções que o sistema deve executar, pergunta-se o **QUE** faz o sistema?. Enquanto que o teste não-funcional avalia as características de um sistema como usabilidade, eficiência, compatibilidade, etc; neste caso pergunta-se **COMO** faz o sistema? Este tipo de teste é aplicado geralmente no nível de teste de sistema e aceitação, porém nada impede ser executado em outros níveis de teste dependendo do contexto da *app*.

No caso das *apps*, as diversas características das *apps* motivam uma diversidade de testes a serem recomendados. Entender as características e funcionalidades dessas *apps* é importante para todo o processo de testes, garantindo assim a sua qualidade. Baseadas nas características das *apps*, na Tabela 1 é apresentado os diferentes tipos de teste (funcionais e não-funcionais) que são sugeridos para as *apps* (KIRUBAKARAN; KARTHIKEYANI, 2013). Dentre as características das *apps*, é mencionado o suporte a uma diversidade de dispositivos que pode ser auxiliado por meio do teste de compatibilidade (AMALFITANO et al., 2013; M. Amen; M. Mahmood; LU, 2015) que é um dos tipos de teste não-funcionais.

Tabela 1 – Tipos de teste de *apps* baseados em suas características.

Tipos de teste	**Característica da app**
Teste funcional, desempenho, segurança e confiabilidade	Conectividade em diversas redes
Teste de interface (GUI)	Diversidade de GUI
Teste de compatibilidade	Diversidade de dispositivos físicos e SOs
Teste de usabilidade e desempenho	Telas sensíveis ao toque
Teste de caixa branca e preta	Novas linguagens de programação
Teste de funcional e monitoramento de desempenho	Restrições de recursos
Teste funcional dependente do contexto	Sensibilidade ao contexto

2.2.2 Teste de compatibilidade

No padrão ISO-25010 são contemplados oito características de qualidade de um sistema: adequação funcional, confiabilidade, eficiência de desempenho, usabilidade, segurança, compatibilidade, manutenibilidade e portabilidade. Compatibilidade refere-se a quão bem um sistema pode executar suas funções necessárias enquanto compartilha o mesmo ambiente de hardware ou software.

No contexto de *apps* móveis, o teste de compatibilidade se concentra em determinar se o hardware e o software exibem e permitem que uma aplicação funcione corretamente. Visa descobrir falhas da *app* devido à variedade de configurações de hardware e falhas relacionadas a diversidade de dispositivos (AMALFITANO et al., 2013). Este tipo de teste verifica e valida se uma *app* se comporta conforme o esperado numa combinação de dispositivos móveis. Caso a compatibilidade não seja verificada e testada, a *app* pode sofrer lentidão, perda de dados ou travamento.

Segundo Cheng et al. (2015), o teste de compatibilidade refere-se à condução do teste de *apps* para avaliar sua compatibilidade em diferentes ambientes móveis que podem ser a combinação de diferentes plataformas móveis, dispositivos e APIs. Essa diversidade existente de dispositivos, plataformas e outras características móveis é denominada fragmentação móvel.

Algumas dificuldades e desafios no teste de compatibilidade com dispositivos móveis são: custos mais altos; mudanças frequentes e atualizações dos dispositivos móveis e plataformas; interfaces de usuário complexas e falta de modelos e métodos sistemáticos para testes de compatibilidade.

2.3 Plataforma Android

O *Android* é um sistema operacional desenvolvido pelo *Google* para dispositivos móveis. Uma das características da plataforma *Android* foi disponibilizar publicamente seu código-fonte de forma que qualquer pessoa pudesse desenvolver aplicações que fossem executadas em dispositivos *Android* (HAM; PARK, 2011). Essa proposta teve grande aceitação e envolvimento de diversos fabricantes de dispositivos *Android*, de tal forma que a popularização da plataforma foi uma questão de tempo. Atualmente, o número de dispositivos *Android* ativos passa de 3 bilhões[1]. Esses dispositivos pertencem a mais de 100 fabricantes.

Cada fabricante customiza o código-fonte para prover uma maior usabilidade e permitir que seus clientes escolham o dispositivo e software que melhor atenda às suas necessidades (HAN et al., 2012). Por isso, eles adaptam seu desenvolvimento para alcançar uma ampla gama de dispositivos com diferentes resoluções, tamanho de tela, câmera, CPU, processador, armazenamento de memória, compatibilidade de rede e versões de API e SO. Atualmente, o *Android* é uma das plataformas mais usadas para o desenvolvimento de aplicações móveis.

2.3.1 Fragmentação móvel

Devido à diversidade de características e combinações de hardware, e também de versões de APIs do *Android*, ocorre o problema conhecido na literatura técnica como **Fragmentação móvel**. A fragmentação do *Android* refere-se ao comportamento diferenciado de *apps* em cada dispositivo *Android* (HAM; PARK, 2011). Por exemplo, certas *apps* podem funcionar normalmente em diversos dispositivos, mas apresentar falhas no *layout* e *display* em dispositivos com determinado tamanho e/ou resolução de tela (CHENG et al., 2015). O grande número de combinações de diferentes modelos de dispositivos e versões do sistema operacional impossibilitam que os desenvolvedores de *apps Android* testem exaustivamente suas aplicações e, assim, surgem vários problemas de compatibilidade (WEI et al., 2018).

[1] https://io.google/2022/products/android/

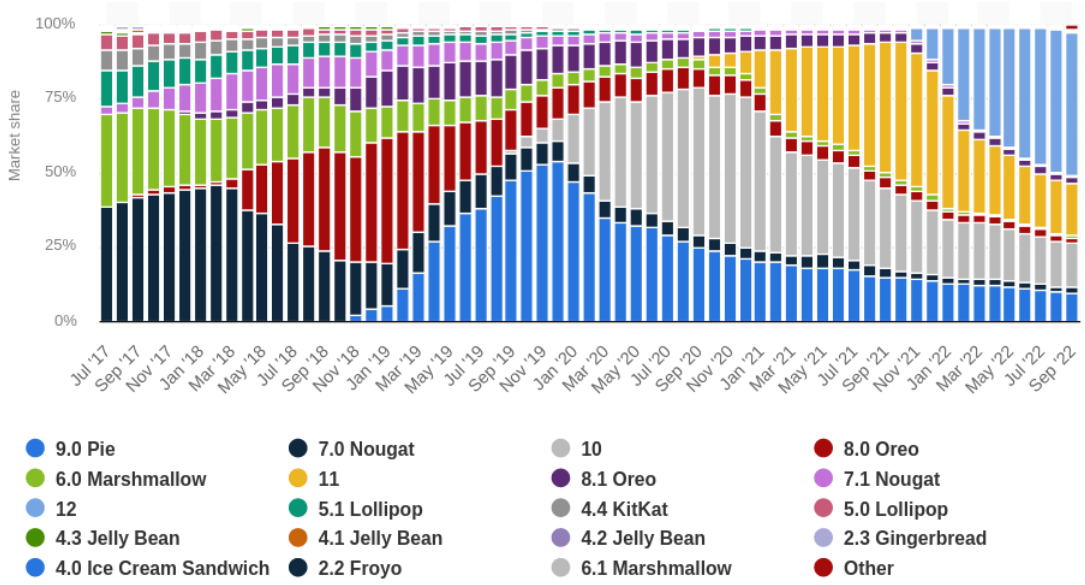


Figura 3 – Android Fragmentação (EEUU) - 2017 a 2022

Como mostra a Figura 3 , o ambiente do sistema operacional do *Android* ainda está fragmentado. Em setembro de 2022, a versão 12 do sistema operacional *Android* (SO) ultrapassou a versão 11 do sistema operacional e detinha uma participação de quase 48% entre todos os dispositivos móveis *Android* nos Estados Unidos (Statista, 2022b). Segundo Jiang, Long e Gao (2007), uma desvantagem da fragmentação é que a diversidade de dispositivos e plataformas móveis está reduzindo a capacidade de reutilização e manutenção dos casos de teste. O problema da fragmentação pode ser tratado como um problema de otimização dentro da Engenharia de Software Baseada em Busca.

2.4 Engenharia de Software Baseada em Busca

A Engenharia de Software Baseada em Busca *(do inglês, Search Based Software Engineering SBSE)* consiste em aplicar a otimização para problemas de Engenharia de Software (SE). No SBSE, o termo *"search"* é usado para se referir às técnicas meta-heurísticas de Otimização Baseada em Busca (SBO) que são usadas. O SBSE procura reformular os problemas de SE como problemas SBO (HARMAN; MANSOURI; ZHANG, 2012).

A otimização consiste em encontrar um ou um conjunto de soluções válidas para um determinado problema. Existe a otimização mono-objetiva que busca soluções

baseado num só objetivo e a otimização multi-objetivas que busca soluções baseado em dois ou mais objetivos. Na SBSE, a representação da solução ou conjunto de soluções para um problema define o espaço de busca onde a pesquisa ocorre; este processo é guiado por uma função objetivo que determina qual é a melhor solução entre duas soluções candidatas (HARMAN; MANSOURI, 2010).

Uma ampla gama de diferentes técnicas de otimização e busca tem sido usada na literatura. Os mais utilizados são: busca local, *Simulated Annealing (SA)*, *Hill Climbing (HC)*, Algoritmos Genéticos (GAs), e Programação Genética (GP).

2.5 Algoritmo genético

Dentre os algoritmos mais usados no SBSE temos o Algoritmo Genético (AG). O conceito de Algoritmo Genético foi desenvolvido por Holland (2012). O AG é baseado no mecanismo de seleção natural proposto por Charles Darwin, onde os indivíduos mais adaptados têm maior probabilidade de sobreviver e se reproduzir, gerando descendentes.

Segundo Golbarg e Luna (2000), as principais características de um AG são: (1) Operam em um conjunto de pontos (denominado população) e não a partir de pontos isolados. (2) Operam em um espaço de soluções codificadas e não diretamente no espaço de busca. (3) Necessitam como informação somente o valor de uma função objetivo (denominada função de adaptabilidade ou *fitness*). (4) Usam transições probabilísticas e não regras determinísticas. A Figura 4 apresenta a representação dos principais componentes de um AG. Os componentes principais são os seguintes:

- **Espaço de busca:** é o espaço onde são consideradas todas as possibilidades de solução de um determinado problema.
- **População:** conjunto de indivíduos que representa um conjunto de soluções do problema.
- **Cromossomo:** representa um indivíduo na população (uma configuração ou solução). Definido normalmente como um vetor de componentes.

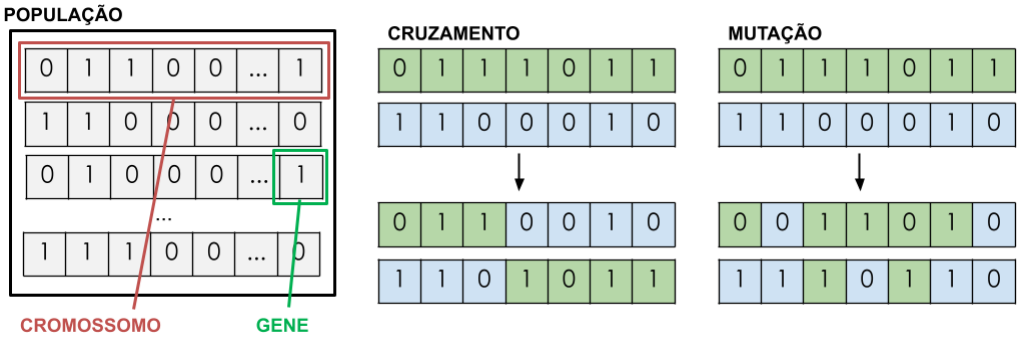


Figura 4 – Representação dos componentes do algoritmo genético.

- **Função objetivo (*Fitness function*):** tem por objetivo fornecer uma medida de aptidão de um indivíduo por meio de uma expressão matemática que mede o quanto uma solução está próxima da solução desejada. Todos os indivíduos estão dentro do espaço de busca.

- **Gene:** é o componente básico do cromossomo, define uma característica específica. Cada cromossomo pode conter vários genes.

- **Operadores genéticos:** são as regras que permitem a manipulação dos cromossomos as quais são:

  - **Cruzamento (*crossover*):** operador que permite a obtenção de indivíduos filhos a partir da combinação ou cruzamento dos cromossomos dos pais.

  - **Mutação:** operador que permite a produção de um novo indivíduo por alterações diretas no cromossomo pai.

O AG tradicional, apresentado na Figura 5 possui as seguintes etapas: *(1) Inicialização:* nessa etapa é gerada uma população inicial com soluções candidatas a partir do espaço de busca, em geral, é de forma aleatória. *(2) Avaliação:* a população inicial é avaliada baseada numa função *fitness*, logo é gerada uma nova população; *(3) Seleção:* são selecionadas as soluções com os melhores valores de *fitness* para serem usados nos próximos passos; *(4) Cruzamento:* duas soluções são cruzadas para obter uma nova solução; *(5) Mutação:* a mutação introduz mudanças aleatórias nas características do indivíduo, promovendo a diversidade genética na população; *(6) Nova Avaliação:* atualização das novas populações geradas que substituem as populações pai; *(7) Convergir:* a

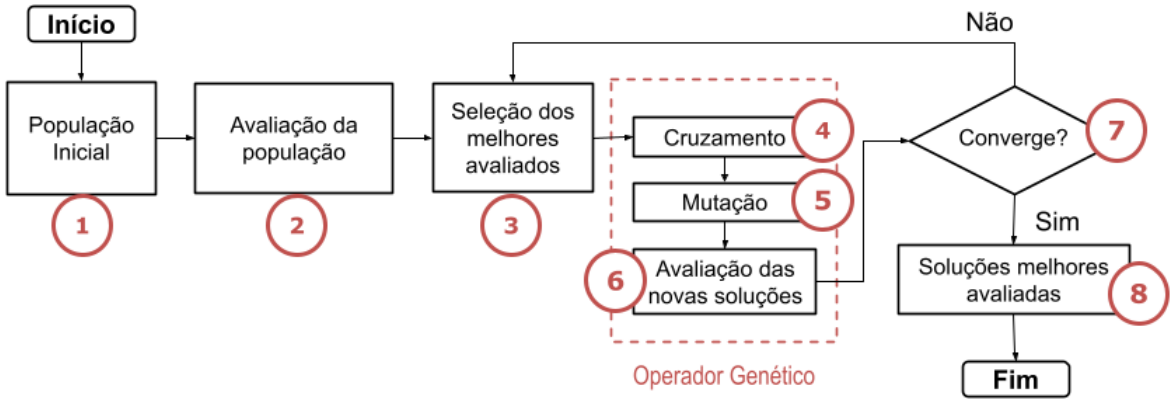


Figura 5 – Fluxo do algoritmo genético.

forma de convergência mais usada é o número de gerações, a qual determina a parada do AG. *(8) Resultado:* apresentação do conjunto de soluções melhores avaliadas

2.6 Considerações finais

Neste capítulo foram considerados os principais conceitos sobre aplicação móvel, teste de aplicações móveis, tipos de teste, teste de compatibilidade, assim como seus desafios. Em seguida, foi detalhado a Plataforma *Android* e suas características, bem como o grande desafio em relação à fragmentação móvel. Finalmente foram apresentados o SBSE e o AG que são abordagens usadas no problema de fragmentação. No próximo capítulo é apresentado o planejamento e os resultados do mapeamento sistemático para entender os problemas e desafios do teste de compatibilidade em *apps*.

3

MAPEAMENTO SISTEMÁTICO SOBRE TESTE DE COMPATIBILIDADE EM APLICAÇÕES MÓVEIS

Neste capítulo é apresentado o corpo de conhecimento sobre as diferentes abordagens que foram usadas para o problema de teste de compatibilidade em aplicações móveis. Para isto, foi planejado e executado um Mapeamento de Literatura Multivocal (*Multivocal Literature Mapping - MLM*). Este capítulo foi baseado nos resultados do journal publicado no *International Journal of Computer Applications in Technology (IJCAT), Vol. 69, No. 2, 2022* - DOI: 10.1504/IJCAT.2022.10051895.

3.1 Motivação Necessidade e Revisões Anteriores

As revisões da literatura técnica existentes sobre teste de apps exploram o corpo de conhecimento principalmente sobre automação de teste, técnicas de geração de teste e os desafios nessa área. Os estudos estão organizados de forma cronológica para observar o avanço das necessidades de pesquisa.

Sahinoglu, Incki e Aktas (2015) analisaram 123 artigos para identificar quais os tipos e níveis de teste são mais frequentemente usados em testes de aplicações móveis. Os resultados identificam maior tendência no teste funcional e usabilidade. Por outro

lado, os resultados indicaram que o teste de performance é uma alternativa de pesquisa devido aos poucos estudos encontrados. Os estudos encontrados relacionados a teste de compatibilidade são ainda menores do que performance, dificultando assim uma melhor análise e entendimento sobre os desafios do teste de compatibilidade.

Holl e Elberzhager (2016) identificaram 230 trabalhos que investigaram a garantia de qualidade em apps. Os autores descrevem as abordagens baseadas no nível de teste além dos desafios de pesquisa. Os resultados apontam que as características funcionais são mais estudadas do que as não-funcionais. Em relação ao teste de compatibilidade, somente 4% dos estudos estão relacionados a este tema. No entanto, os autores ressaltam o desafio, ainda existente, sobre a cobertura de testes em dispositivos específicos (*device-specific*) e a análise de falhas para teste de compatibilidade, especialmente no que diz respeito às relações entre recursos do dispositivo móvel e as falhas de compatibilidade.

Zein, Salleh e Grundy (2016) analisaram 79 artigos em um estudo para categorizar e estruturar as evidências de pesquisa que foram publicadas na área de técnicas de teste de apps e os desafios que eles relatam. Diversas lacunas de pesquisa foram identificadas, como a elicitação de requisitos de teste no início do processo de desenvolvimento, assim como estudos comparativos para teste de segurança e usabilidade. O estudo não aborda teste de compatibilidade. Kong et al. (2018) analisaram 103 artigos para avaliar o estado da arte relacionado ao teste de apps Android. O objetivo foi identificar quais os tipos, níveis e técnicas de teste são mais abordados nos estudos. Os autores propuseram uma taxonomia categorizando os estudos pelos objetivos, níveis e técnicas aplicados nos testes de apps. Dentro dos objetivos do teste mais citados, são apresentados Concorrência, Segurança, Performance, Energia, Compatibilidade e Erros (*Bugs/Defect*). Este último é o que apresenta o maior número de estudos, enquanto que Compatibilidade é o menos estudado. Como consequência, pouco se sabe sobre os desafios específicos do teste de compatibilidade.

Tramontana et al. (2018) analisaram 131 artigos, onde o foco principal do estudo foi a classificação dos trabalhos relacionados à automação do teste funcional em apps. Os artigos foram classificados com base nas atividades de teste suportadas, as características das técnicas e as ferramentas apresentadas. O estudo não considerou os trabalhos

relacionados ao teste de compatibilidade. Porém, o estudo cita que a automação de teste funcional pode ser reusada no contexto do teste de compatibilidade.

Considerando os estudos citados acima, não há indícios de estudos sistemáticos que investiguem sobre os desafios e as problemáticas relacionadas ao teste de compatibilidade. No entanto, há uma necessidade de entender melhor como é realizado o teste de compatibilidade em apps e quais são os seus desafios. Para isso, foi realizada uma revisão da literatura baseada na técnica do *snowballing* e análise na literatura cinza. Nas próximas seções serão apresentados o planejamento, execução e resultados deste estudo.

3.2 Metodologia de Pesquisa

A metodologia de pesquisa para o Mapeamento de Literatura Multivocal - MLM foi dividida em duas fases principais: a primeira fase foi baseada na abordagem *Snowballing* para estudos na literatura técnica, e a segunda fase foi baseada na revisão da literatura cinza.

Primeira fase: O *Snowballing* é usado como uma abordagem de busca para estudos da literatura técnica. Badampudi, Wohlin e Petersen (2015) referem que a eficiência do *snowballing* é comparável aos mapeamentos sistemáticos que usam as bibliotecas digitais. Esta técnica utiliza a lista de referências de um artigo ou das citações do artigo para identificar documentos adicionais (WOHLIN, 2014). A lista de referências de um artigo é a lista que se encontra no seu final, enquanto que as citações são os artigos que citam o artigo que está sendo analisado. Nesse processo, o uso das referências é conhecido como *Backward Snowballing*, enquanto que o uso das citações é referido como *Forward Snowballing*. Nesta tese, usaremos as siglas BSB para *Backward Snowballing* e FSB para *Forward Snowballing*. Segunda fase: Conforme sugerido pela técnica de MLM, incluímos a literatura cinza (do inglês, *Grey Literature* - GL) para saber como é tratada na indústria o teste de compatibilidade em apps. Assim, para a revisão da GL o planejamento foi baseado nas diretrizes apresentadas por Garousi, Felderer e Mäntylä (2019).

3.2.1 Planejamento

O planejamento deste estudo seguiu as diretrizes e passos apresentados por Wohlin (2014). O objetivo principal foi estruturado usando a abordagem GQM (*Goal-Question-Metric*) proposta em Basili, Caldiera e Rombach (1994). Assim, o objetivo do estudo é analisar o estado da arte de teste de compatibilidade no contexto de teste de software em apps móveis, com o propósito de caracterizar os desafios e problemas do ponto de vista do pesquisador. Este estudo foi realizado entre dezembro de 2018 e janeiro de 2019; logo uma atualização foi realizada em junho de 2021 e uma segunda atualização em outubro de 2022.

3.2.2 Questão de pesquisa

A necessidade de identificar e caracterizar os desafios que enfrentam os desenvolvedores e testadores na execução dos testes de compatibilidade direcionou a realização de uma revisão da literatura técnica para responder à seguinte questão de pesquisa: **"O que a literatura técnica diz sobre os desafios do teste de compatibilidade em apps?"**. Para abordar o objetivo apresentado e responder à principal questão de pesquisa, foram definidas três subquestões de pesquisa:

- QP1. Quais são as áreas onde é explorado o Teste de Compatibilidade em apps?
- QP2. Quais são as oportunidades de pesquisa nas áreas que exploram o Teste de Compatibilidade em apps?
- QP3. Como os profissionais na indústria selecionam dispositivos móveis para testes de compatibilidade?

Para responder QP1 e QP2, usamos a abordagem *Snowballing* e conforme sugerido por Kitchenham e Charters (2007), os critérios de inclusão foram definidos com base nas questões de pesquisa. Os estudos que não atendem algum dos critérios foram excluídos. Critérios de Inclusão: (1) Estudos relacionados a teste de compatibilidade em apps (*Mobile Compatibility Testing*); (2) Serão incluídos os artigos disponíveis na web ou por meio de contato com os autores no idioma inglês; (3) Serão incluídos os artigos que

apresentarem textos completos dos estudos em formato eletrônico; (4) Serão incluídos os artigos que sejam publicados a partir do ano 2008. O motivo desse critério foi baseado no surgimento dos termos "*Compatibility e Fragmentation*", como pode ser observado na Figura 6 que mostra o número de pesquisas no *Google* por um termo específico em um período de tempo. Além disso, as plataformas *iOS* e *Android* foram introduzidas em 2007 e 2008, respectivamente.

Quanto à QP3, definimos uma questão de pesquisa específica para o contexto da indústria relacionada à "seleção de dispositivos para teste". Já que essa atividade é muito comum entre desenvolvedores e testadores e está intimamente relacionada à diversidade de dispositivos móveis e o teste de compatibilidade em apps. Quase todos os desenvolvedores e testadores precisam selecionar um conjunto de dispositivos ou emuladores para executar os testes.

Como exemplo, *SALESFORCE* anunciou que não oferecerá mais suporte para todos os dispositivos móveis; a empresa divulgou algumas recomendações sobre dispositivos compatíveis com sua aplicação. Atualmente, eles tornaram público o conjunto de dispositivos onde sua aplicação móvel é testada (SALESFORCE, 2021). Assim, conhecer o estado da prática sobre este tema é relevante devido à constante atualização e lançamento de novos dispositivos móveis no mercado.

Portanto, conforme sugerido pela técnica de MLM, foi utilizada a literatura cinza (GL) para responder à QP3. Revisamos sistematicamente a GL para encontrar fontes relacionadas à "seleção de dispositivos para teste". Assim, algumas dessas diretrizes seguidas, apresentadas por Garousi, Felderer e Mäntylä (2019), são as seguintes: (1) A decisão de incluir a GL em um estudo de mapeamento (ao invés de um SLR convencional) deve ser feita de forma sistemática; usando um bem- conjunto definido de critérios/questões; (2) Com base em seu objetivo de pesquisa e público-alvo, defina as questões de pesquisa (QPs); (3) Identifique os tipos de GL relevantes e/ou produtores de GL (fontes de dados) para seu estudo de revisão desde o início; (4) Mecanismos gerais de pesquisa na web, bancos de dados e sites especializados, e contato direto com indivíduos são formas de pesquisar literatura cinza.

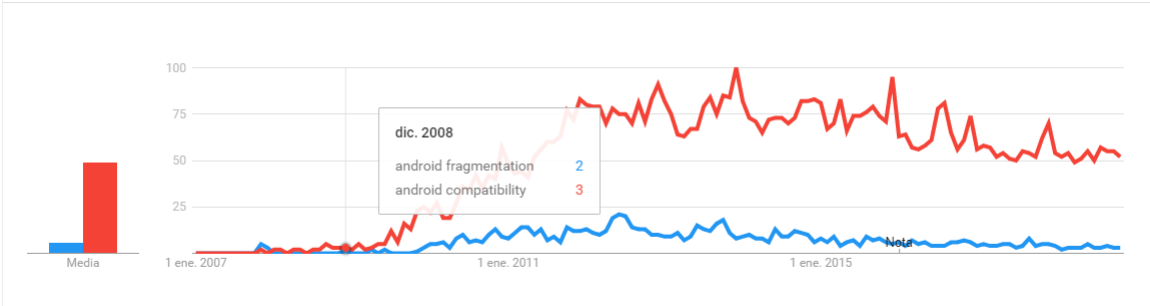


Figura 6 – Ano de surgimento dos termos "compatibility"e "fragmentation".

Tabela 2 – *String* de busca para o conjunto inicial de artigos.

Strings de busca
“mobile app compatibility”,“mobile app fragmentation”, “mobile app cross-platform", “mobile app cross-device”, “mobile app device-specific fault”

3.3 Execução do *Snowballing*

3.3.1 Definição do conjunto inicial de artigos:

Um passo muito importante na execução da técnica de *Snowballing* é a definição do conjunto inicial de artigos. Wohlin (2014) recomenda definir um conjunto de *strings* de busca e logo fazer uma pesquisa no *Google Scholar* para evitar viés em favor de alguma revista específica. Foi feita a busca na ferramenta *Google Scholar* com o intuito de cumprir as características levantadas pelo autor. As *strings* de busca usadas para definir o conjunto inicial de artigos são listadas na Tabela 2. Para definir as palavras (*strings*) principais, iniciou-se com “*compatibility*” e depois foram adicionando-se os sinônimos que eram usados pelos autores. Foi usada também a palavra “*fragmentation*” devido ela ser muito usada junto com “*compatibility*”, além dos termos para se referir às apps e teste.

Foram aplicadas as *strings* definidas no *Google Scholar* e os artigos foram selecionados com base nos critérios de inclusão definidos na Seção 3.2.2; Parou-se na página 5 depois que a página não encontrou novos resultados. No término da busca foram selecionados 6 artigos base que cumprem com as seguintes características descritas por Wohlin (2014):

(1) Os artigos são de diferentes autores e não são citados entre eles;

(2) O ano de publicação dos artigos são diferentes e diversificados (considerando

os mais antigos e os mais atualizados).

Os seis artigos base SS (*do inglês Start Set*) foram identificados como SS1 até SS6. A lista desses artigos está na Tabela 3. Após a definição dos artigos base, iniciou-se as iterações, que são descritas na próxima subseção.

Tabela 3 – Conjunto inicial de *papers* (*Start Set - SS#*).

ID	Autor	Ano	Titulo	Local
SS1	Ham & Park	2011	Mobile application compatibility test system design for Android fragmentation.	International Conference on Advanced Software Engineering and Its Applications (ASEA)
SS2	Khalid et al.	2014	Prioritizing the Devices to Test Your App on: A Case Study of Android Game Apps.	International Symposium on Foundations of Software Engineering (FSE)
SS3	Cheng et al.	2015	Mobile Compatibility Testing Using Multi-objective Genetic Algorithm.	IEEE Symposium on Service-Oriented System Engineering (SOSE)
SS4	Zhang et al.	2015	Compatibility Testing Service for Mobile Applications.	IEEE Symposium on Service-Oriented System Engineering (SOSE)
SS5	Lu et al.	2016	Prada Prioritizing Android Devices for Apps by Mining Large-Scale Usage Data.	IEEE/ACM International Conference on Software Engineering (ICSE)
SS6	Vilkomir, S.	2018	Multi-device coverage testing of mobile applications.	Software Quality Journal

3.3.2 Iterações:

O processo das iterações foi dividido em duas etapas: *Backward Snowballing* (BSB) e *Forward Snowballing* (FSB). Cada etapa foi inicializada com o conjunto inicial de artigos (SS1 - SS6). Nesse processo foram aplicados os critérios de inclusão definidos na etapa do planejamento.

Backward SnowBalling (BSB): Nessa etapa, para cada artigo foi extraída a lista de referências, como possíveis candidatos a serem incluídos durante o processo de *Snowballing*. Para cada item da lista de referências de um artigo, foram examinados o título e o resumo por meio da aplicação dos critérios para inclusão de estudos. No total, os seis artigos base (SS1 - SS6) ajudaram na identificação de 575 artigos candidatos a serem incluídos. Destes, 40 artigos foram selecionados, dos quais 18 eram duplicados. Na Tabela 4 mostra-se o total de artigos candidatos e selecionados para cada artigo do conjunto base (SS1 - SS6) em cada iteração.

Forward SnowBalling (FSB): nesta etapa, foram usadas as seguintes bibliotecas digitais: Google Scholar, IEEE Xplore e Scopus para identificar as citações por cada

Tabela 4 – Número de artigos candidatos e selecionados do Conjunto inicial - BSB.

Start Set Papers	**Candidatos**				**Dupl.**	**Selecionados**			
	1 Iter	**2 Iter**	**3 Iter**	**4 Iter**		**1 Iter**	**2 Iter**	**3 Iter**	**4 Iter**
SS1: Ham2011	9	-	-	-	18	0	-	-	-
SS2: Khalid2014	35	69	44	-		4	2	0	-
SS3: Cheng2015	29	39	13	-		4	3	0	-
SS4: Zhang2015	24	66	88	-		4	7	4	-
SS5: Lu2016	41	25	43	10		4	1	1	0
SS6: Vilkomir2018	23	17	-	-		5	1	-	-
Total	**575**				**18**	**22**			

Tabela 5 – Número de artigos candidatos e selecionados do Conjunto inicial - FSB.

Start Set Papers	**Candidatos**				**Dupl.**	**Selecionados**			
	1 Iter	**2 Iter**	**3 Iter**	**4 Iter**		**1 Iter**	**2 Iter**	**3 Iter**	**4 Iter**
SS1: Ham2011	22	76	83	5	23	5	11	15	2
SS2: Khalid2014	38	0	-	-		6	0	-	-
SS3: Cheng2015	3	-	-	-		0	-	-	-
SS4: Zhang2015	14	7	-	-		4	1	-	-
SS5: Lu2016*	0	-	-	-		-	-	-	-
SS6: Vilkomir2018*	0	-	-	-		-	-	-	-
Total	**248**				**23**	**21**			

artigo base. Novamente foi extraída a lista de artigos como possíveis candidatos a serem incluídos em cada iteração. Para cada artigo que citava o artigo base foram examinados o título e o resumo por meio da aplicação dos critérios para inclusão dos artigos. No total, os seis artigos base (SS1 - SS6) ajudaram na identificação de 248 artigos candidatos a serem incluídos. Destes, 23 foram duplicados e 21 foram selecionados. Na Tabela 5 mostra-se o total de artigos candidatos e selecionados para cada artigo do conjunto base (SS1 - SS6) em cada iteração. Os artigos com a marcação * possuem zero na contagem devido a eles terem sido executados em iterações de outros artigos. No caso de SS5:Lu2016 foi executado na segunda iteração de SS1:Ham2011 e no caso de SS6:Vilkomir2018 foi executado na segunda iteração de SS4:Zhang2015.

Nas duas etapas (BSB e FSB), após a quarta iteração, não encontrou-se nenhum artigo novo. Muitos deles já tinham sido identificados em iterações anteriores. Os resultados das duas etapas foram unificados, resultando em 823 artigos candidatos, dos quais foram selecionados 43 artigos.

Para atualizar a seleção dos estudos, uma nova iteração do FSB foi realizada em junho de 2021 e outra em outubro de 2022. Foram utilizados os 21 artigos previamente

Tabela 6 – Artigos selecionados na atualização do *Forward Snowballing*.

Atualização	**Conjunto inicial**	**Candidatos**	**Duplicados**	**Selecionados**
2021	21 Papers	361	76	14
2022	35 Papers	297	69	7

selecionados como conjunto inicial, e os principais resultados desta atualização são mostrados na Tabela 6. Na primeira atualização de 2021, inicialmente, foram identificados 361 artigos candidatos. Após a primeira etapa de filtragem, foram obtidos 111 estudos primários, dos quais 76 eram duplicatas. Por fim, após sua leitura na íntegra, 14 artigos foram selecionados e agregados ao conjunto de 21 artigos. Assim, 35 artigos foram o total de artigos selecionados. Na segunda atualização de 2022, foram identificados 297 artigos candidatos. Após a primeira etapa de filtragem, foram obtidos 93 estudos primários, dos quais 69 foram duplicados. Por fim, após sua leitura na íntegra, 7 artigos foram selecionados e agregados ao conjunto de 35 artigos. No final, foram 42 artigos utilizados para responder às questões de pesquisa.

A Figura 7 apresenta o resultado final da execução do *snowballing*. O gráfico possui cinco iterações; as três primeiras iterações, resultou no processo de unificação dos resultados das duas etapas (BSB e FSB), os artigos duplicados foram eliminados ficando somente o artigo com a iteração mais recente; e as duas últimas iterações representam as atualizações do snowballing.

A lista dos artigos selecionados está no Apêndice A; para identificar os artigos, a etapa e iteração em que foi selecionado criou-se a seguinte estrutura "[A / B / C / D / E] [B / F] [#]", sendo que na primeira posição a letra "A" representa a primeira iteração, "B" a segunda iteração, "C" a terceira iteração, "D" a quarta iteração (primeira atualização) e "E" a quinta iteração (segunda atualização); na segunda posição, a letra "B" representa *backward* e "F" *forward* e finalmente o número "#" da terceira posição representa a numeração. Assim, o código **[BB9]** indica que o artigo foi selecionado na segunda iteração **[B]**, na etapa do *backward* **[B]** e é o item número **9**.

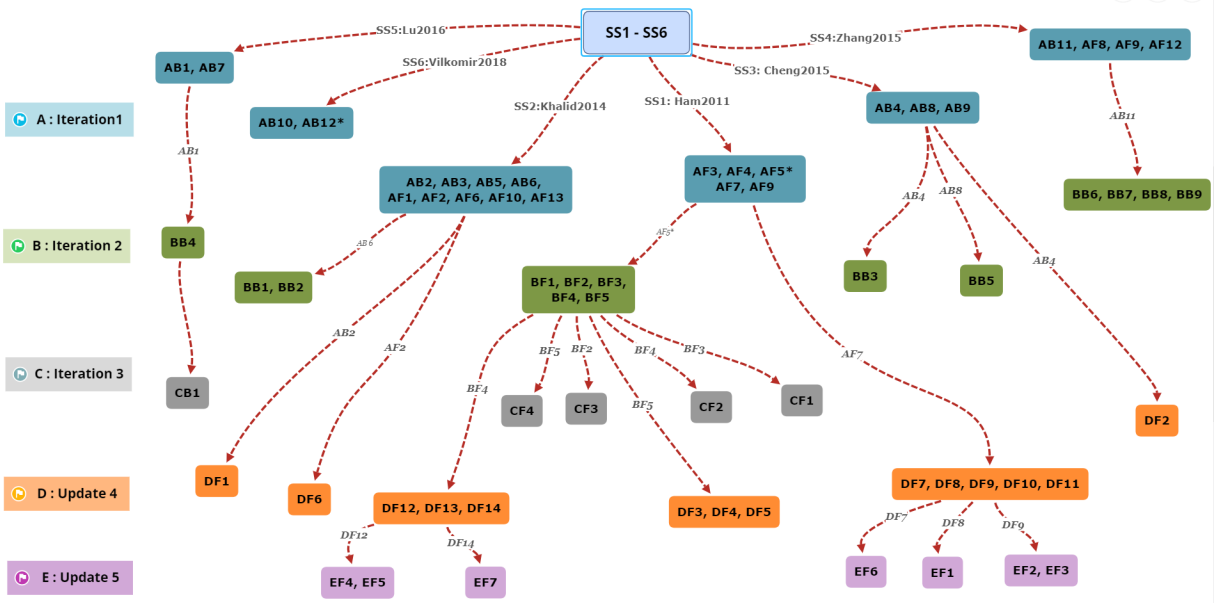


Figura 7 – Resultado final do processo de *Snowballing*.

3.4 Análise da Literatura cinza

Depois da primeira fase com a abordagem *Snowballing*, a segunda fase foi iniciada com a identificação da estratégia de busca da Literatura Cinza (GL). *"A estratégia de busca para GL é obviamente diferente, pois as bases de dados acadêmicas não indexam GL - (Garousi et al., 2019)"*. Assim, conforme sugerido por Garousi, Felderer e Mäntylä (2019), foi utilizada a busca do *Google* e alguns sites especializados, como o site do *International Software Testing Qualifications Board* (ISTQB[1]) e a pesquisa anual do *World Quality Report* (WQR[2]) que poderiam fornecer indícios para a questão definida.

Baseados na QP3 e nas diretrizes de Garousi, Felderer e Mäntylä (2019), foram executadas as seguintes etapas:

- Definir as seguintes *strings* de busca "*mobile devices*", "*selection*" e "*compatibility testing*"; para cada um foram usados seus sinônimos;
- Definir os seguintes critérios de inclusão (baseados na QP3):
  - Fontes relacionadas a técnicas ou métodos de seleção de dispositivos móveis para teste;

[1] www.istqb.org
[2] www.capgemini.com/service/world-quality-report-2018-19/

Tabela 7 – Artigos selecionados na Literatura Cinza.

ID	Site	Year	Fonte
GL1	Qatestlab	2014	qatestlab (2014)
GL2	99tests	2016	Medium@99tests (2016)
GL3	Coveros	2016	Coveros (2016)
GL4	SauceLabs	2016	SauceLabs (2016)
GL5	Avantica	2017	Avantica (2017)
GL6	Plusqa	2018	plusqa (2018)
GL7	Test48	2019*	Test48 (2019)
GL8	BrowserStack	2020	BrowserStack (2019)
GL9	EasyQA	2019*	EasyQA (2019)
GL10	Medium	2017*	Medium (2017)
GL11	Medium	2018	Chaitanya (2018)
GL12	Qatestlab	2019	qatestlab (2019)
GL13	Plusqa	2019	plusqa (2019)
GL14	Avantica	2020	Avantica (2020)
GL15	Plusqa	2021	plusqa (2021)

  - As fontes precisam estar disponíveis online;
  - Descartar textos relacionados à compatibilidade no *browser*.

- Usar as *strings* de busca nas fontes de dados definidas e colete resultados.
- Atualizar os resultados, uma nova busca foi feita em junho de 2021, para atualização dos resultados.

A Tabela 7 mostra o resultado com os quinze artigos selecionados; os resultados marcados com (*) indicam que a fonte não indicava o ano de publicação. Nesses casos, foi considerado o ano em curso. Algumas fontes (GL12, GL13, GL14, GL15) publicaram informações em dois anos diferentes. As pesquisas em ISTQB e WQR não retornaram resultados relacionados à seleção de dispositivos para teste.

Para avaliar a qualidade das fontes encontradas, a lista de verificação sugerida por Garousi, Felderer e Mäntylä (2019) foi aplicada. A Tabela 8 apresenta a avaliação a cada resultado com questões relacionadas aos autores da fonte e sua experiência no assunto, bem como a metodologia, objetividade e apresentação dos dados. Para cada questão, se a resposta for afirmativa, atribuímos um ponto à fonte, caso contrário zero. Na maioria dos casos, o valor de avaliação foi superior a 0,5. Todas as publicações se concentram na seleção de dispositivos ou dispositivos para teste.

Tabela 8 – Avaliação da qualidade das publicações na Literatura Cinza.

Critério	Questões	GL1	GL2	GL3	GL4	GL5	GL6	GL7	GL8	GL9	GL10	GL11	GL12	GL13	GL14	GL15
Autoridade do produtor/autor	- A organização da publicação é respeitável?	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1
	- O autor individual está associado a uma organização respeitável?	0	0	1	1	0	0	0	0	0	1	1	1	0	1	0
	- O autor publicou outro trabalho na área?	1	0	1	1	1	1	0	0	0	0	1	1	1	1	1
	- O autor possui expertise na área?	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1
Metodologia	- A fonte tem um objetivo claramente definido?	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
	- A fonte tem uma metodologia declarada?	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
	- A fonte é apoiada por referências oficiais e documentadas?	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0
	- Existem limites claramente definidos?	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Objectividade	- O trabalho cobre uma questão específica?	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1
	- O trabalho se refere a uma determinada população ou caso?	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	- A declaração nas fontes é a mais objetiva possível?	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1
	- As conclusões são apoiadas pelos dados?	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1
	- O item tem uma data claramente indicada?	1	1	1	1	1	0	0	1	0	1	1	1	0	1	0
Resultados	**Soma**	**7**	**6**	**10**	**10**	**8**	**8**	**6**	**8**	**4**	**3**	**10**	**9**	**8**	**9**	**8**
	Normalizado (0-1)	**0.54**	**0.46**	**0.77**	**0.77**	**0.62**	**0.62**	**0.46**	**0.62**	**0.31**	**0.23**	**0.77**	**0.69**	**0.62**	**0.69**	**0.62**

Algumas informações utilizadas para a avaliação das fontes foram as seguintes: Para a reputação da organização, foi verificado se a fonte pertence a alguma empresa ou organização que oferece serviços ou pública tópicos relacionados a testes em dispositivos móveis. No caso de publicações publicadas em outras plataformas, como sites pessoais ou blogs, foi verificada a experiência do autor em relação ao seu trabalho ou outras publicações na mesma área. Apenas duas fontes (GL10, GL11) são postagens de blog pessoais; e os outros treze estão relacionados a empresas de tecnologia. Dez fontes são blogs que pertencem a uma empresa relacionada a serviços de teste (GL1, GL2, GL4, GL6, GL7, GL8, GL9, GL12, GL13, GL14), três são postagens de empresas relacionadas a serviços de engenharia de software (GL3, GL5, GL14).

Segundo informações dos autores, cinco pertencem à área de engenharia de software (GL3, GL4, GL11, GL12, GL14), uma delas é da área de *marketing* (GL10) e as demais fontes não tiveram autor identificado. Nenhum dos autores cita o uso de qualquer ferramenta para seleção de dispositivos. Apenas dois autores (GL7, GL8) citam suas referências.

3.5 Ameaças à validade

Nesta seção, são discutidas as principais ameaças à validade do mapeamento sistemático, que foram organizadas em quatro categorias: validade, interna, constructo e externa.

Validade: No Mapeamento Sistemático, existe a ameaça de perder algum estudo relevante. Para mitigar essa ameaça, foi adotada uma estratégia de pesquisa com o uso de todas as *strings* de pesquisa relevantes e seus sinônimos, considerando o ano de início do uso das *strings* de pesquisa.

Interna: As ameaças à validade interna estão relacionadas à execução do mapeamento sistemático. É importante destacar que foram seguidas rigorosamente as diretrizes sugeridas por Wohlin (2014) e Garousi, Felderer e Mäntylä (2019).

Construto: Está relacionado à falta de estudos relevantes. Para minimizar essa ameaça, foi seguido um protocolo com alguns critérios para definir o conjunto inicial de papéis. Foram executados dois tipos de *Snowballing*, *Backward* e *Forward*. Além disso, foram usadas fontes de busca para extração de citações, como *Google Scholar, IEEE Xplore e Scopus* para não perder nenhum artigo importante.

Externa: Ameaças à validade externa são condições que limitam a capacidade de generalizar nossos resultados. Com o objetivo de classificar os artigos, foi criada uma nova taxonomia baseada em outras previamente definidas por Ham e Park (2011) e Wei et al. (2018).

3.6 Análise de resultados

3.6.1 Resultados e categorização

Neste Mapeamento da Literatura Multivocal são apresentados dois resultados. O primeiro, relacionado ao processo de *Snowballing*, e o segundo, relacionado à revisão da Literatura Cinza (GL). Para o processo de *Snowballing*, foram identificados 42 artigos a serem analisados e classificados. A Tabela 9 mostra o ID do *Snowballing* para identificar cada estudo, o ano, a citação do estudo e os locais onde foram publicados. Dos 42 artigos, seis foram publicados em periódicos e trinta e seis foram publicados em congressos. Também mostra que a pesquisa de compatibilidade de apps móveis aparece em diferentes conferências, incluindo *workshops* e simpósios. A maioria dos trabalhos foi publicada em 2018 e 2020, mas uma pesquisa inicial foi publicada em 2011, três anos após o surgimento dos termos *"compatibilidade e fragmentação"* no contexto de aplicações

móveis.

Categorização: Para classificar os estudos, foram utilizadas como base as taxonomias propostas por Wei et al. (2018) e Ham e Park (2011). Existem dois ambientes principais onde a compatibilidade móvel é evidente: *Hardware* e *Software*. O contexto de *hardware* está relacionado a diferentes modelos e marcas de dispositivos móveis, enquanto o contexto de *software* está relacionado às versões da API do *Android* e às versões do sistema operacional *Android*. No entanto, esses dois contextos estão intimamente relacionados, uma vez que os dispositivos móveis (*hardware*) possuem diferentes níveis de API e versões de SO (*software*). Então, para classificar os estudos, foram executadas as seguintes etapas: (1) Identificar o foco principal do estudo. Como isso foi feito de forma interativa, foram identificadas semelhança entre os estudos; (2) Foi extraído o problema de compatibilidade para cada artigo e definido uma palavra-chave temporária para identificar cada subcategoria; (3) Em seguida, verificou-se se o estudo pode estar relacionado a alguma atividade de teste móvel; (4) Uma palavra-chave final foi definida para cada subcategoria e cada subgrupo foi classificado nas duas principais categorias: *hardware* e *software*. (5) Essas atividades foram conduzidas pelo primeiro autor e avaliadas pelos demais autores.

A Figura 8 mostra o total de publicações por ano e as principais categorias: *hardware* e *software*. Os primeiros estudos publicados estão mais relacionados ao contexto de *hardware*, ou seja, aos dispositivos e seus diferentes modelos e marcas. Sendo assim, a fragmentação é um dos primeiros problemas detectados na indústria (OPENSIGNAL, 2015). Os estudos publicados após 2017 são mais orientados para o contexto de *software*. Podemos associar este fato, entre outros eventos, ao “projeto Treble" lançado pelo *Android* em 2018, este projeto é um esforço ambicioso do *Google* para re-arquitetar o *Android* para dividir a estrutura principal do sistema operacional e o código de baixo nível específico do dispositivo criado por fornecedores de silício. Este projeto visa tornar as atualizações do sistema operacional *Android* mais rápidas e fáceis para desenvolvedores e usuários.

A Figura 9 apresenta o número de estudos para cada subcategoria. A categoria *Hardware* representa 48% do total, com 20 estudos. Ele consiste nas seguintes subcatego-

Tabela 9 – Artigos selecionados e local de publicação.

ID	Ano	Fonte	Local de publicação
AB2	2011	(HAM; PARK, 2011)	International Conference on Advanced Software Engineering and Its Applications (ASEA)
AB3	2012	(HAN et al., 2012)	Working Conference on Reverse Engineering (WCRE)
BB3	2012	(HUANG; GONG, 2012)	IEEE International Conference on Cloud Computing Technology and Science (CloudCom)
AB4	2014	(HUANG, 2014)	IEEE International Conference on Mobile Cloud Computing Services and Engineer (MOBILECLOUD)
AF5	2014	(KHALID et al., 2014)	International Symposium on Foundations of Software Engineering (FSE)
AB9	2014	(VILKOMIR; AMSTUTZ, 2014)	IEEE International Conference on Software Testing Verification and Validation (ICST)
AB1	2015	(HALPERN et al., 2015)	IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)
AF1	2015	(CHENG et al., 2015)	IEEE Symposium on Service-Oriented System Engineering (SOSE)
AB12	2015	(ZHANG et al., 2015)	IEEE Symposium on Service-Oriented System Engineering (SOSE)
AF10	2016	(LU et al., 2016)	IEEE/ACM International Conference on Software Engineering (ICSE)
BF5	2016	(WEI; LIU; CHEUNG, 2016)	International Conference on Automated Software Engineering (ASE)
AF2	2017	(FAZZINI; ORSO, 2017)	International Conference on Automated Software Engineering (ASE)
AF11	2018	(NAITH; CIRAVEGNA, 2018b)	International Conference on Crowd Science and Engineering (ICCSE)
CF3	2018	(NAITH; CIRAVEGNA, 2018a)	International Journal of Crowd Science
AF12	2018	(VILKOMIR, 2018)	Software Quality Journal
AF3	2018	(HE et al., 2018)	International Conference on Automated Software Engineering (ASE)
AF4	2018	(HUANG et al., 2018)	International Conference on Automated Software Engineering (ASE)
AF7	2018	(LI et al., 2018b)	International Symposium on Software Testing and Analysis (ISSTA)
CF2	2018	(LI et al., 2018a)	Asia-Pacific Software Engineering Conference (APSEC)
BF4	2018	(WEI et al., 2018)	IEEE Transactions on Software Engineering (TSE)
AF6	2018	(KOWALCZYK; COHEN; MEMON, 2018)	International Workshop on Advances in Mobile App Analysis (A-Mobile)
DF1	2019	(FARIA et al., 2019)	Future Generation Computer Systems
DF2	2019	(LIU, 2019)	Multimedia Tools and Applications
DF3	2019	(KI et al., 2019)	IEEE/ACM International Conference on Software Engineering (ICSE)
DF8	2019	(SCALABRINO et al., 2019)	IEEE/ACM International Conference on Mining Software Repositories (MSR)
DF9	2019	(CAI et al., 2019)	International Symposium on Software Testing and Analysis (ISSTA)
DF12	2019	(WEI; LIU; CHEUNG, 2019)	IEEE/ACM International Conference on Software Engineering (ICSE)
DF5	2020	(MENEGASSI; ENDO, 2020)	IET Software
DF6	2020	(MOBILIO et al., 2020)	International Conference on Automated Software Engineering (ASE)
DF7	2020	(XIA et al., 2020)	IEEE/ACM International Conference on Software Engineering (ICSE)
DF10	2020	(MUKHERJEE; RUHE, 2020)	International Workshop on Artificial Intelligence and Requirements Engineering (AIRE)
DF11	2020	(SCALABRINO et al., 2020)	IEEE/ACM International Conference on Mining Software Repositories (MSR)
DF14	2020	(VILLANES; ENDO; DIAS-NETO, 2020)	Symposium on Systematic and Automated Software Testing (SAST)
DF4	2021	(HARYONO et al., 2021)	IEEE/ACM International Conference on Software Engineering (ICSE)
DF13	2021	(NIELEBOCK et al., 2021)	IEEE/ACM International Conference on Mining Software Repositories (MSR)
EF1	2022	(SILVA et al., 2022)	IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)
EF2	2022	(ZHAO et al., 2022)	IEEE/ACM International Conference on Software Engineering (ICSE)
EF3	2022	(SUN et al., 2022)	IEEE/ACM International Conference on Automated Software Engineering (ASE)
EF4	2021	(MAHMUD; CHE; YANG, 2021)	IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)
EF5	2022	(MAHMUD; CHE; YANG, 2022a)	IEEE/ACM International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)
EF6	2022	(MAHMUD; CHE; YANG, 2022b)	ACM / IEEE International Symposium on Empirical Software Engineering and Measurement
EF7	2022	(REN et al., 2022)	IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)

rias: serviço de teste, seleção de dispositivo, teste baseado em GUI e fragmentação. A categoria *Software* representa 52%, com 22 estudos, e consiste nas seguintes subcategorias: problema de detecção de API, suporte à API, evolução da API e Teste unitário de API.

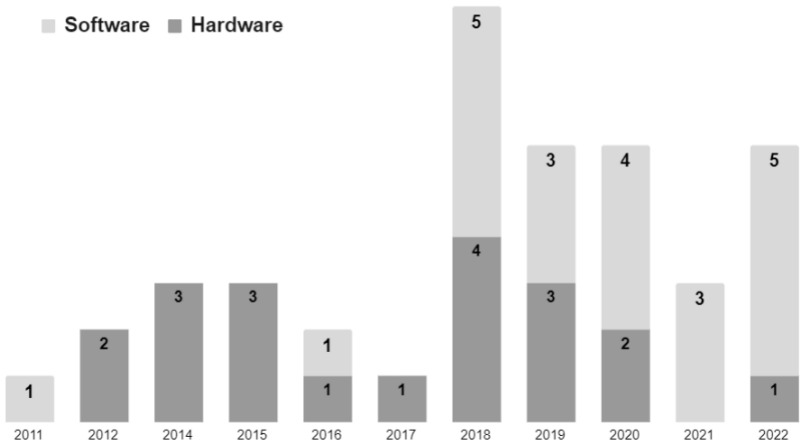


Figura 8 – Publicações por ano e categoria: Hardware e Software.

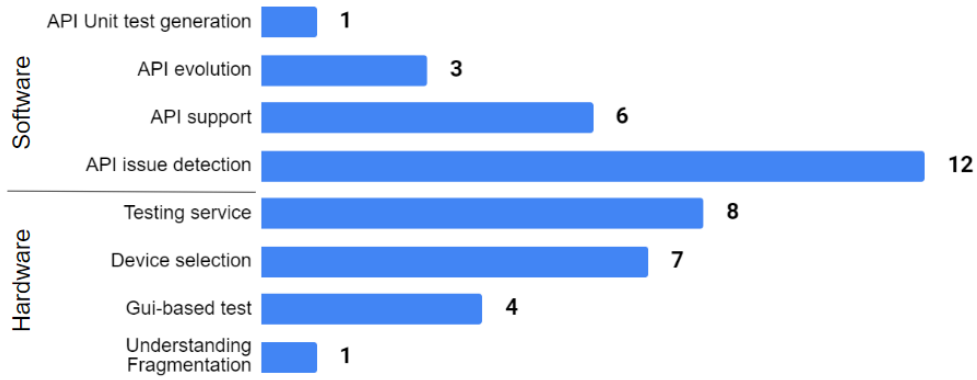


Figura 9 – Artigos selecionados: Categorias & subcategorias.

3.6.2 QP1: Quais são as áreas onde é explorado o Teste de Compatibilidade?

Os estudos incluídos para responder a esta pergunta de pesquisa foram categorizados de acordo com o esquema de classificação descrito na Seção 3.6.1 e com base em artigos selecionados (Ver Tabela 9). Alguns artigos incluídos para análise não estão diretamente relacionados ao teste. No entanto, eles são focados em "Compatibilidade" ou "Fragmentação", e esses tópicos são relevantes quando se estuda do ponto de vista de testes de compatibilidade móvel.

A Tabela 10 mostra os estudos incluídos por cada categoria e subcategoria. Até janeiro de 2021 a maioria dos estudos estava no contexto de *hardware* que está intimamente relacionado à diversidade móvel; porém este ano de 2022 os estudos relacionados a software cresceram e representam 52% dos artigos. Durante o processo de categorização, para cada artigo, extraímos o problema tratado e sua correlação com os testes de compatibilidade. Por fim, foram encontrados suporte em diferentes atividades de teste, como execução de teste (AB4, AB1, AF11, CF3, BB3, DF1, DF2, DF3), geração de script de teste (AF6, DF5), seleção de dispositivos para teste (AF5, AB9, AF1 , AB12, AF12, AF10, DF14), teste de interface de usuário (AF2,EF7) e teste unitário de API (EF3).

3.6.2.1 Pesquisas voltadas ao Hardware

Os estudos nesta categoria são focados em modelos de dispositivos e notamos que a maioria dos artigos deste grupo é orientada a testes e um deles está orientado ao

Tabela 10 – Categorização dos artigos selecionados.

Hardware		**Software**	
Device selection		**API evolution**	
[AF5]	Khalid2014	[AF3]	He2018
[AB9]	Vilkomir2014	[DF13]	Nielebock2021
[AF1]	Cheng2015	[EF6]	Mahmud2022a
[AB12]	Zhang2015	**Detect API issue**	
[AF12]	Vilkomir2018	[AF7]	Li2018
[AF10]	Lu2016	[BF5]	Wei2016
[DF14]	Villanes2020	[AF4]	Huang2018
Testing service		[CF2]	Li2018a
[AB4]	Huang2014	[BF4]	Wei2018
[AB1]	Halpern2015	[AB2]	Ham2011
[AF11]	Naith2018	[DF7]	Xia2020
[CF3]	Naith2018a	[DF9]	Cai2019
[BB3]	Huang2012	[DF10]	Mukherjee2020
[DF1]	Faria2019	[EF1]	Silva2022
[DF2]	Liu2019	[EF4]	Mahmud2021
[DF3]	Ki2019	[EF5]	Mahmud2022
Gui-based test		**API support**	
[AF6]	Kowalczyk2018	[DF4]	Haryono2021
[DF5]	Menegassi2020	[DF6]	Mobilio2020
[AF2]	Fazzini2017	[DF8]	Scalabrino2019
[EF7]	Ren2022	[DF11]	Scalabrino2020
Understanding Frag.		[DF12]	Wei2019
[AB3]	Han2012	[EF2]	Zhao2022
		API Unit test generation	
		[EF3]	Sun2022

entendimento do problema da fragmentação. Assim, os estudos foram divididos em quatro categorias: Entendendo a Fragmentação (*Understanding Fragmentation*), Seleção de dispositivos (*Device selection*), Serviço de teste (*Testing service*) e Teste baseado em Interface (*GUI-based test*).

Na categoria ***Entendendo a Fragmentação (Understanding Fragmentation)***, o estudo de Han et al. (2012) [AB3] extrai relatórios de *bugs* do repositório *Android* para as marcas HTC e Motorola. Esses relatórios foram etiquetados manualmente por cada *bug* usando os algoritmos LDA e "Labeled LDA". O objetivo da proposta foi encontrar quais seriam os tópicos mais relevantes relacionados à Fragmentação. No resultado o "Labeled LDA"produziu mais tópicos relevantes, porém o tempo de execução toma

aproximadamente 60 vezes mais de esforço. Os autores afirmam que a fragmentação baseada no *hardware* é mais evidente devido aos diversos tópicos de cada dispositivo móvel. Um dos tópicos que tem revelado maior relevância é os teclados de *software*.

Esse artigo foi um dos primeiros trabalhos publicados relacionados ao entendimento do problema de fragmentação. Os autores afirmam que a fragmentação baseada em *hardware* é mais evidente devido aos diversos tópicos de cada dispositivo móvel. Após nove anos, a diversidade de dispositivos continua aumentando e pesquisas ainda são necessárias para entender essas novas mudanças. Para o restante dos estudos, eles foram agrupados em três subcategorias: seleção de dispositivo, serviço de teste e teste baseado em GUI.

Seleção de Dispositivos (Device Selection): Diferentes abordagens foram propostas para o desafio de seleção de dispositivos; Vilkomir e Amstutz (2014) [AB9] propõem uma abordagem que está baseada em métodos combinatórios para cobertura de cada característica do dispositivo. Os métodos sugeridos foram "Each Choice (EC)"e "Pair-wise". O experimento avaliou o método *Each Choice*, que requer que cada valor de cada característica seja usado pelo menos uma vez. Para avaliar a eficácia da abordagem proposta, foi comparado o resultado de testar a mesma app usando dois conjuntos de dispositivos diferentes; a primeira usando o método *Each Choice* e outro conjunto de dispositivos selecionados aleatoriamente. Os resultados indicam que o uso do *Each Choice* consegue encontrar mais falhas. Para o experimento, o número e as falhas já tinham sido identificados, o que permitiu a comparação dos métodos.

Três estudos ([AB12] [AF5] [AF10]) usaram mineração de dados para selecionar dispositivos. Zhang et al. (2015) [AB12] usaram a mineração de dados junto ao algoritmo *K-means* para agrupar dispositivos móveis com recursos de compatibilidade semelhantes; Todos os *clusters* de dispositivos foram classificados por participação de mercado. Foram usadas informações de 2500 dispositivos. Com os dados das características dos dispositivos, foi gerado um modelo de árvore de característica (*Feature Tree Model*). Para a avaliação da proposta, foram comparados os 20 dispositivos mais populares dos *clusters* com 20 dispositivos selecionados por engenheiros de teste, que escolheram os dispositivos baseados na sua experiência. O conjunto gerado pela proposta conseguiu

alcançar uma maior cobertura de características e participação no mercado.

Khalid et al. (2014) [AF5] mineraram as revisões dos usuários da Loja de apps *open-source Google PlayStore*, com o objetivo de priorizar os dispositivos móveis para serem testados. A metodologia comparou as revisões ruins, médias e boas fornecidas por diferentes dispositivos para identificar se determinados dispositivos apresentam classificações diferentes (ou piores) do que outros dispositivos. As revisões consideradas foram somente aquelas que estavam associadas a um dispositivo. Os resultados mostram que os usuários com o *Motorola Droid X2* atribuem mais avaliações ruins aos apps do que usuários de outros dispositivos. Essas avaliações ruins podem ser problemas específicos do fabricante. Por outro lado, foi evidenciado que uma pequena porcentagem de dispositivos é responsável pela maioria das revisões dadas as apps. Esses resultados confirmam a necessidade de uma abordagem que procure identificar e selecionar quais dispositivos seriam os mais problemáticos.

Lu et al. (2016) [AF10] apresentaram PRADA, uma abordagem para priorizar modelos de dispositivos para testar apps Android baseados na mineração de dados de uso em grande escala. Os autores coletaram três meses de dados de uso da app "Wandouja" (Uma app chinesa que é similar a *App Google Play*, que serve para gerenciar as apps do usuário). PRADA inclui uma técnica de filtragem colaborativa para prever o uso de um aplicativo em diferentes modelos de dispositivos, mesmo que o aplicativo seja novo (sem uso no mercado). Para a extração dos dados, foram usadas 200 apps que cobriram 3,86 milhões de usuários e 14,71 mil modelos de dispositivos. Porém, devido à app ser somente usada na China, os dados são representativos para os dispositivos que são usados nesse país. Os três estudos utilizam mineração de dados, porém, atualmente a loja Google Play não disponibiliza os modelos dos dispositivos nos comentários dos aplicativos; apenas o proprietário do aplicativo tem acesso aos modelos de dispositivos dos usuários. Assim, a mineração de dados só pode ser realizada em casos limitados.

Determinar o número de dispositivos a serem usados no teste também é um problema, pois depende dos recursos e o objetivo do teste. Vilkomir (2018) [AF12] avaliou o número de dispositivos que seriam necessários para encontrar 100% das falhas identificadas em passos anteriores. No total, quinze (15) apps foram testadas em trinta

(30) dispositivos móveis. Para identificar essas 24 falhas, foram selecionados conjuntos de dispositivos por meio randômico, baseados nos tipos de sistema operacional, e no método *each-choice*. O método de seleção randômica alcançou 85% de efetividade e alcançou 100% de efetividade com 13 dispositivos, enquanto que os métodos por cobertura de SO e *each-choice* alcançou 90% com 5 dispositivos.

Cheng et al. (2015) [AF1] propõem um método de teste de compatibilidade móvel sistemático e econômico usando o algoritmo genético multi-objetivo. Foram usadas informações de 4000 dispositivos extraídos do sítio da empresa *Amazon*. Para medir a cobertura de dispositivos, foram usadas as características: tamanho e resolução de telas, tipos de conectividade e quota de mercado. Os resultados indicam que o número de dispositivos para teste a serem escolhidos deve ser entre 36 e 40. Porém, somente o número de dispositivos sem as características dos dispositivos que devem ser escolhidos, não contribui para o teste.

Serviço de teste (Testing service): Para abordar a grande diversidade de dispositivos e o problema da fragmentação, algumas soluções foram propostas com o objetivo de executar testes de compatibilidade. Fazendo uso do sistema *Mobile Testing as a Service (MTaaS)* para o problema da fragmentação, esses serviços oferecem a disponibilidade de uma variedade de dispositivos para que os usuários possam escolher e pagar pelo uso. Huang e Gong (2012) [BB3] propuseram o Sistema de Teste Móvel Remoto (RTMS), hospedada na nuvem, alojando um conjunto de dispositivos móveis para testar aplicativos móveis. No RTMS, os usuários podem acessar remotamente um pool de dispositivos móveis que existem em um laboratório local e realizar testes de upload, inicialização e teste de aplicativos, clicando e deslizando as ações. Em outra versão mais atualizada da pesquisa, Huang (2014) [AB4] apresenta um serviço na nuvem AppACTS (Mobile App Compatibility Testing Service). Essa nova proposta consiste de um conjunto de dispositivos móveis que poderiam estar geograficamente distribuídos e conectados a um servidor.

Outro estudo de Liu (2019) [DF2] apresentou uma *Cloud Testing Platform (CTP)*, para suportar testes de compatibilidade para aplicativos multimídia Android. Em particular, o CTP permite que aplicativos multimídia Android sejam testados automati-

camente em um número escalável de dispositivos físicos em paralelo. Essa proposta suporta teste de interface (GUI), teste de estresse e *crash*. Além disso, Ki et al. (2019) [DF3] descrevem um novo sistema de teste de compatibilidade de interface do usuário para aplicativos Android chamado Mimic. Um dispositivo executa uma sequência de ações de interface do usuário e todos os outros dispositivos repetem essa sequência. Esse modelo é útil para testes de compatibilidade de interface do usuário em diferentes versões do Android, tipos de dispositivos e versões de aplicativos.

A técnica de *record & replay* para execução de testes usada por Halpern et al. (2015) [AB1] apresenta Mosaic, um serviço na nuvem *cross-platfom* para a execução de testes em apps Android. Para superar os problemas da fragmentação relacionados ao tamanho de tela e orientação, os autores usaram a técnica da virtualização. As entradas do usuário são capturadas num dispositivo, logo os dados são virtualizados em uma representação intermediária que é independente da plataforma, que pode ser redirecionada para novos dispositivos.

O *Crowd testing* envolve um grande número de testadores distribuídos em diferentes locais. Naith e Ciravegna (2018b), Naith e Ciravegna (2018a) [CF3][AF11] apresentam uma plataforma de teste *AskCrowd2Test*. A plataforma é um método de *crowdsourcing* híbrido, que usa *crowd testers* experientes e usuários sem muita experiência em teste; o objetivo é usar o poder público dos *crowd testers* para usar seus próprios dispositivos e nessa forma ter uma maior diversidade de dispositivos e diversidade nos tipos de teste, de um lado os testadores experientes e do outro lado os usuários comuns. Um serviço semelhante, porém orientado ao aluguel de dispositivos, é apresentado por Faria et al. (2019) [DF1]; os autores propõem uma plataforma chamada *Distributed Bug Buster - DBB* que executa casos de teste baseados em Espresso em vários dispositivos reais. Esta proposta é considerada como um novo mercado para os proprietários de dispositivos alugarem seus dispositivos móveis e alcançarem uma maior diversidade.

Teste baseado em interface (GUI-based testing): Os estudos nesta subcategoria apresentam soluções baseadas tanto na configuração do dispositivo, interface GUI e geração de *script* de teste.

Kowalczyk, Cohen e Memon (2018) [AF6] fizeram um estudo sobre a importân-

cia das configurações dos dispositivos, se ignorados eles podem levar a diferenças nas saídas do teste e na cobertura do código. Para o estudo foram usados 18 apps com suas respectivas suítes de teste. Foi elaborada uma matriz de teste com 88 configurações usando quatro (4) parâmetros (*Device, Android API level, Locale, Device orientation*). Os resultados do estudo mostraram que somente uma das 18 apps não apresentou variação. O restante das apps teve variação seja na cobertura de código ou nos resultados dos testes. Isso evidencia que as variações nas configurações dos dispositivos podem levar a erros de compatibilidade dentro do mesmo dispositivo. Menegassi e Endo (2020) [DF5] apresentam uma abordagem chamada x-PATESCO para gerar *scripts* para testar aplicativos móveis multiplataforma em várias configurações. A abordagem identifica elementos de interface do usuário em dois dispositivos de plataformas diferentes (*Android e iOS*) e registra *scripts* robustos que podem ser executados em outros dispositivos.

No contexto GUI também foram identificadas as possíveis *issues* geradas pela fragmentação. Fazzini e Orso (2017) [AF2] apresentaram DIFFDROID, uma ferramenta baseada na ferramenta Monkey, UiAutomation e o Teste diferencial; a ferramenta estendeu o framework Espresso para a geração e execução dos casos de teste. Essa ferramenta compara o modelo gerado em um dispositivo de referência com os modelos gerados em um conjunto maior de dispositivos com diferentes plataformas. O objetivo é identificar alguma possível inconsistência e relatar as diferenças da interface.

Os autores usaram o serviço do "AWS Device farm"para a execução do experimento, usando 147 dispositivos, os quais estavam agrupados pela resolução e a versão do sistema operacional. Os resultados do experimento encontraram 96 inconsistências devido a diferenças na versão do SO usado ou na configuração da tela. As inconsistências mais repetidas foram cosméticas, seguido de funcional, versão de SO e estrutural.

No mesmo contexto, Ren et al. (2022) [EF7] apresentaram um método baseado em imagem chamado de Detector de Diferenças entre Dispositivos (do inglês, *Cross-Device Difference Detector - CdDiff*). Esta ferramenta usa um par de capturas de tela de dois dispositivos como entradas, emprega um módulo para extrair os principais elementos da tela de cada captura de tela e, em seguida, utiliza um módulo de posicionamento de discrepância para detectar e visualizar os elementos de diferença entre as duas capturas

de tela em uma forma facilmente reconhecível.

Discussão: As categorias apresentadas nesta seção refletem algumas etapas dentro do planejamento do teste que estão relacionadas ao dispositivo (hardware). Uma das etapas é identificar quais dispositivos serão usados para teste e definir a infraestrutura onde o teste será realizado, seja num laboratório local ou usando dispositivos na nuvem. Independentemente de qual escolha é feita, essa tarefa é fundamental para a próxima etapa da execução dos testes.

As pesquisas nesse sentido são relevantes porque ajudam os desenvolvedores a tomar decisões que permitam evidenciar possíveis erros de compatibilidade nesses dispositivos selecionados. Porém alguns trabalhos (KHALID et al., 2014; CHENG et al., 2015; LU et al., 2016) não são mais replicáveis devido a que algumas informações de dispositivos não estão disponíveis na Internet o que dificulta o uso dessas propostas, gerando a necessidade de mais pesquisas na seleção de dispositivos. Existem diversas opções para executar testes móveis (serviços locais ou em nuvem), porém a grande diversidade exige uma melhor seleção de dispositivos, pois é impossível testar em todos os dispositivos.

3.6.2.2 Pesquisas voltadas ao *Software*

As pesquisas nessa área se concentram em encontrar as principais causas de erros de compatibilidade analisando o código dos aplicativos móveis. O objetivo é encontrar padrões de uso; código relacionado à evolução das APIs e como isso afeta o funcionamento correto dos aplicativos móveis. Além disso, suporte para corrigir problemas de compatibilidade; recentemente um novo estudo foi adicionado relacionado ao teste unitário de API. Organizamos essas pesquisas em quatro tópicos (subcategorias): evolução da API (*API evolution*), detecção de problemas da API (*API issue detection*), suporte à API (*API Support*) e Teste unitário de API (*API Unit test generation*).

Evolução da API (API evolution): A evolução de software é uma das importantes áreas de pesquisa em engenharia de software. Foi reconhecida como a atividade mais difícil, cara e trabalhosa no ciclo de vida de desenvolvimento de software (XIA et

al., 2020). No mesmo contexto, a API do Android evolui constantemente; e esse processo gera problemas de compatibilidade nos aplicativos.

O estudo apresentado por He et al. (2018) conduziu um estudo empírico para entender a evolução das API do Android e como elas poderiam induzir a problemas de compatibilidade. Os autores propuseram a ferramenta *IctApiFinder*, que detecta problemas no uso de APIs incompatíveis nas aplicações. Para a análise da evolução das APIs foram usadas 11 versões diferentes de Android (API 16 até API 27); os autores fizeram a suposição de que o uso da variável "SDK_INT" pode ajudar a resolver os problemas de compatibilidade induzidos pela evolução. De 4936 apps foram selecionadas 10 apps com o maior uso da variável "SDK_INT" as quais foram manualmente inspecionadas para entender como os desenvolvedores enfrentam os problemas de compatibilidade induzidos pela evolução das APIs. Para avaliar a ferramenta *IctApiFinder* foram usadas 1425 apps extraídas do repositório *AndroZoo*; das quais foram encontrados 361 (25.33%) usos incompatíveis de API do total das 1425 apps. *Google* faz grandes esforços para solucionar os problemas de compatibilidade, porém os resultados do estudo revelaram que, por exemplo, a biblioteca de suporte de Android provê suporte para menos do 23% das novas APIs em cada lançamento e a maioria das apps precisa abordar ou confrontar com os problemas de compatibilidade, induzidos pela evolução das APIs, no desenvolvimento dessas apps.

O estudo de Nielebock et al. (2021) [DF13] concentra-se em informações históricas e evolutivas de verificações de compatibilidade no código-fonte de aplicativos *Android*. Os autores apresentam o *dataset* "AndroidCompass"que inclui alterações nas verificações de compatibilidade que os desenvolvedores usam para aplicar soluções alternativas para versões específicas do Android em seus aplicativos. Novos estudos são necessários sobre detecção de padrões, reparo automatizado de programas e benchmarking.

Finalmente, o estudo de Mahmud, Che e Yang (2022b) [EF6] investigou empiricamente como os campos da API *Android* evoluem e como essa evolução afeta a compatibilidade de aplicativos *Android*. Foi realizado um estudo baseado em dados de histórico de desenvolvimento de aplicativos do mundo real envolvendo 11.098 *tags* de

105 aplicativos *Android* de código aberto. Os resultados identificaram que existem dois problemas de compatibilidade de API por aplicativo e que em média, leva mais de três meses para corrigir estes problemas, isso leva a necessidade de continuar investigando em ferramentas de suporte para detecção e correção de problemas de compatibilidade.

Detecção de problemas de API (API issue detection): Uma das primeiras iniciativas para compreender o problema da fragmentação foi entender se o código das aplicações era o causador dos problemas. Os primeiros trabalhos que falam sobre fragmentação apresentam propostas de solução, porém, sem nenhuma implementação. Assim Ham e Park (2011) [AB2] propuseram dois métodos para confrontar a fragmentação. Um método a nível de código, que busca encontrar inconsistências no código que poderiam levar a um erro de compatibilidade, e outro a nível de API. Nas duas propostas é apresentado um exemplo de código que poderia induzir ao erro.

Com o objetivo de entender e conhecer quais seriam as causas (*root cause*), os sintomas e quais são as estratégias de solução aos problemas da fragmentação Wei, Liu e Cheung (2016) [BF5] fizeram uma busca manual nos repositórios de *issues* do *GitHub* em 27 apps e como resultado obtiveram 191 *issues* chamadas de problemas de Compatibilidade induzidas por fragmentação (*Fragmentation-Induced Compatibility - FIC*). Os autores propuseram uma ferramenta *FicFinder*, baseada no modelo de pares (*pair model API-context*) para detectar automaticamente problemas FIC. Os resultados do *FicFinder* demonstram que de 51 *issues* foram reportadas 46 como verdadeiro positivo (uma precisão de 90.2%).

Em uma nova versão da ferramenta *FicFinder*, Wei et al. (2018) [BF4] adicionaram o número de *issues* relacionadas a compatibilidade para 220. Uma nova avaliação da ferramenta foi executada com 53 *open-source Androi*d apps. Os resultados foram categorizados em dois tipos: "*device-specific*"(120 *issues*) e "*non-device-specific*"(100 *issues*). Os problemas "*device-specific*"acontecem na mesma versão de API, porém em diferentes modelos de dispositivos. Enquanto que os problemas "*non-device-specific*"foram detectados por causa da evolução das APIs e pelos *bugs* existentes na plataforma Android. Esses resultados podem nos guiar para novas oportunidades de pesquisa, como *bugs* relacionados a dispositivos específicos e quais modelos de dispositivos específicos podem ser

considerados para teste.

Uma nova extensão ao trabalho de, Wei et al. (2018) [BF4] foi proposta por Li et al. (2018b) [CF2]. Os autores estenderam o trabalho e apresentaram a ferramenta *ELEGANT* que usa a base de *API-Context pairs* do *FincFinder* e incluem 41 novos pares de *API-Context*. A principal diferença do ELEGANT é que a ferramenta isola as bibliotecas de terceiros, devido os autores afirmarem que muitos dos problemas são gerados nessas bibliotecas e isso está fora do controle dos desenvolvedores, então evitar essas bibliotecas reduz os falsos positivos.

Huang et al. (2018) [AF4] propuseram um modelo baseado em gráficos (*PI-Graph*) para capturar as alterações dos protocolos de invocação da API de retorno de chamada do *Android* (*Android callback*). Foi proposta a ferramenta *CIDER* baseada na análise estática que detecta dois tipos de problemas de compatibilidade: alteração de acessibilidade e modificação comportamental da API. Os resultados da avaliação do *CIDER* detectaram 13 verdadeiros positivos e 1 falso positivo, do total de 14 problemas, alcançando uma precisão de 92.9% que supera significativamente o *Lint* que é uma ferramenta dentro do *Android Studio*.

Xia et al. (2020) [DF7] realizaram um estudo em larga escala sobre a prática de lidar com problemas de compatibilidade induzidos pela evolução da API. Além disso, os autores apresentam uma ferramenta automatizada chamada *RAPID* para determinar se um problema de compatibilidade foi resolvido ou não, incorporando técnicas de análise estática e aprendizado de máquina. Em outro estudo, Cai et al. (2019) [DF9] abordaram o problema de compatibilidade com um estudo longitudinal em larga escala com 62894 apps benignos para entender os sintomas e as causas desses problemas.

Mukherjee e Ruhe (2020) [DF10] realizaram um estudo em 40 aplicativos de código aberto selecionados aleatoriamente da *Google Play Store* e analisou 258.056 *commits* (extraídos de seu sistema de controle de versão) para identificar problemas de incompatibilidade nos aplicativos. Os autores também estudaram 205.847 revisões para identificar e categorizar os requisitos de compatibilidade das revisões dos usuários. Os resultados mostraram que 3,2% do esforço do desenvolvedor é dedicado a problemas de compatibilidade e 4,3% das avaliações de usuários relatam problemas de

compatibilidade em aplicativos.

Li et al. (2018b) [AF7] propuseram uma abordagem automatizada chamada *CiD* para modelar sistematicamente o ciclo de vida das APIs do Android e analisar o *bytecode* da app para sinalizar os usos que podem levar a possíveis problemas de compatibilidade. *CiD* realiza análise estática em três etapas: Na primeira etapa é analisada o código dos 24 *releases* do Android, logo é gerada um modelo do ciclo de vida dos métodos das API; na segunda etapa todos os métodos de API usados pela app são identificados; e finalmente na terceira etapa os dados da primeira e segunda etapa são usados para identificar os possíveis problemas de compatibilidade.

Silva et al. (2022) [EF1] propuseram uma abordagem *SAINTdroid* de análise de código para permitir a identificação eficiente e escalável de vários tipos de problemas de compatibilidade do Android. A diferença dentre outras ferramentas nesta linha *SAINTdroid* suporta a identificação de erros de compatibilidade até a versão do API 29. Em relação a velocidade, os experimentos mostraram que é até quatro vezes em média mais rápido do que outras técnicas como *CiD*.

Finalmente, Mahmud, Che e Yang (2021), Mahmud, Che e Yang (2022a) [EF5] [EF6] propuseram *ACID* que é outra ferramenta que utiliza diferenças de API e análise estática de aplicativos Android para detectar problemas de compatibilidade de invocação de API e problemas de compatibilidade de retorno de chamada de API. os autores avaliaram em 20 aplicativos de referência mostrando o *ACID* mais preciso e rápido na detecção de problemas de compatibilidade do que *CiD, Cider e Lint*.

Suporte ao API (API support): Após detectar problemas de compatibilidade da API, o próximo passo é corrigi-los e garantir o funcionamento correto do aplicativo. Haryono et al. (2021) [DF4] propõem o *AndroEvolve*, uma ferramenta automatizada para atualizar o uso de APIs Android obsoletas, que aborda as limitações da ferramenta *CocciEvolve*. Esta nova versão apresenta melhorias relacionadas ao número de atualizações corretas e demonstra sua maior precisão e legibilidade. Outro estudo do Mobilio et al. (2020) [DF6] propõe o *FILO*, uma ferramenta que auxilia aos desenvolvedores na identificação dos métodos que devem ser modificados para tornar um aplicativo, com falhas de compatibilidade, para uma versão mais recente do Android. A ferramenta é

capaz de rastrear a interação entre um aplicativo e seu framework em diferentes versões da API, então as diferenças entre elas são consideradas como blocos de invocações suspeitas.

Scalabrino et al. (2019), Scalabrino et al. (2020) [DF8][DF11] propõem uma solução orientada a dados, *ACRYL*, que aprende com as mudanças implementadas em outros aplicativos em resposta a mudanças que ocasionaram quebras de API. Isso permite que a *ACRYL* (i) recomende como corrigir o problema detectado e (ii) identifique usos de API que não são ótimas, além de problemas de compatibilidade de API. Outro estudo de Wei, Liu e Cheung (2019) [DF12] propôs *PIVOT* uma abordagem automatizada de aprendizado de correlação de dispositivos de API *(API-device correlation)* para facilitar a detecção de problemas. Para identificar efetivamente correlações válidas de API, o *PIVOT* extrai e prioriza correlações de uma determinada seção do aplicativo Android. Os autores focam em questões específicas do dispositivo devido ao espaço de busca ser grande e dinâmico; esta ferramenta oferece suporte aos desenvolvedores para identificar problemas específicos do dispositivo.

Zhao et al. (2022) [EF2] propuseram *RepairDroid*, esta ferramenta fornece uma linguagem genérica de descrição de *patch* para que os usuários criem modelos de correção para problemas de compatibilidade. Os modelos criados logo são aproveitados pelo *RepairDroid* para corrigir automaticamente o problema correspondente no nível de *bytecode*. Os resultados experimentais mostraram que *RepairDroid* é capaz de descrever corretamente 42 dos 44 problemas, e o módulo de reparo é eficaz, sendo capaz de alcançar um total de 85,34% de sucesso taxa na reparação de 1.000 aplicativos *Android* do *Google Play* selecionados aleatoriamente.

Teste Unitário para API (API Unit test generation): Além de corrigir os problemas de compatibilidade, uma atividade importante é a criação de testes unitários. Sun et al. (2022) [EF3] apresentaram o *JUnitTestGen*; esta ferramenta explora o uso existente da API do Android nos aplicativos para gerar casos de teste unitários. o *JUnitTestGen* executa a análise de fluxo de dados reverso entre procedimentos para gerar um trecho de código executável mínimo que depois será o caso de teste. Os experimentos na ferramenta atingiram um 80% de sucesso na criação de casos de teste válidos.

Discussão: A maioria dos artigos usou análise estática para identificar problemas de compatibilidade em aplicativos móveis. No entanto, com o passar do tempo, o grande conjunto de dispositivos no mercado é atualizado e outros estão sendo descartados ou eliminados do mercado, assim como as versões de SO e APIs. Esse fenômeno mantém o processo de pesquisa ativo. Um exemplo claro é observado na evolução contínua da API; a cada ano, novas versões de API são lançadas e os desenvolvedores precisam descobrir como isso afeta seus aplicativos e quais ações devem ser tomadas para garantir ainda mais a qualidade de seu software. Algumas propostas utilizam revisão manual de código, porém a grande diversidade de aplicativos móveis faz com que cada pesquisa amplie o número de aplicativos avaliados para ter uma melhor representação.

A maioria dos artigos se concentra na detecção de problemas de API usando aprendizado de máquina, estudos empíricos, análise de *bytecode* e, portanto, continua sendo um desafio. Algumas abordagens propõem métodos de correção, como atualizar APIs obsoletas, recomendar como corrigir problemas e detectar problemas específicos do dispositivo, ajudando os desenvolvedores no desenvolvimento. Tanto a detecção de problemas de compatibilidade quanto os métodos de correção precisam de esforços humanos para validar se um código específico representa um problema de compatibilidade ou se a correção proposta está correta. Portanto, ainda são necessárias novas pesquisas para automatizar a detecção e correção de problemas devido à atualização contínua de versões de APIs e novos dispositivos móveis.

3.6.3 QP2: Quais são as oportunidades de pesquisa na área de teste de Compatibilidade?

As oportunidades de pesquisa foram alinhadas de acordo à taxonomia criada na subseção anterior. Assim, as oportunidades de pesquisa foram divididas nas duas áreas maiores: hardware e software.

Software, nessa área, as oportunidades de pesquisa mais exploradas são as que estão relacionadas ao código e identificação de padrões e a evolução das APIs. No caso das APIs, a sua constante evolução é um problema aberto, devido principalmente a dois

aspectos; (1) o surgimento de novas funcionalidades e (2) a retirada ou eliminação de funcionalidades que ficaram obsoletas do suporte oficial, causando diversos problemas para os desenvolvedores que deverão atualizar seu código. Um claro exemplo disso é a desativação da *API Google+* que deixou de funcionar em março 7 de 2019. Essa decisão leva a que os desenvolvedores que usam essa API procurem outras alternativas para manter sua app funcionando. Então, continua em aberto pesquisar uma maneira de auxiliar os desenvolvedores na atualização do código das suas apps com a evolução contínua das APIs.

Por outro lado, relacionado ao código da app, existe ainda o desafio de auxiliar os desenvolvedores no uso adequado das boas práticas no desenvolvimento de apps. Por exemplo, um problema muito comum está relacionado a forma como os desenvolvedores indicam qual é a versão mínima e máxima que sua app suporta, a boa prática de indicar esses atributos não é muito bem estabelecida na comunidade Android (LI et al., 2018b). Em um estudo feito por Wu et al. (2017) encontrou-se que alguns apps (lançadas em 2015) declararam a variável "*targetSdkVersion*" como 24, 25, 26 ou maior, no entanto, essas versões do SDK ainda não foram lançadas naquele ano. Ainda mais surpreendente, uma aplicação móvel define o valor "*targetSdkVersion*" como "10000", causando assim erros de compatibilidade.

Em relação ao ***Hardware***, a principal oportunidade de pesquisa está relacionada à seleção de dispositivos para teste. Segundo Lu et al. (2016), uma seleção abaixo do ideal pode desperdiçar dinheiro e esforço humano, deixar de encontrar erros potenciais ou até mesmo perder receita.

As opções de dispositivos na nuvem para teste são cada vez maiores e na indústria temos diversas opções (Ex. *AWS Device farm , Firebase Test Lab , Xamarin test Cloud*) porém, ainda tendo muitos dispositivos disponíveis para o uso na nuvem, existe o desafio de escolher quais seriam os ideais para a execução de testes. Escolher os dispositivos, na nuvem ou para compra, ainda é um desafio, pois envolve o uso de recursos financeiros.

Em geral, as pesquisas focam em duas opções: (1) seleção de dispositivos e (2) indicar o número de dispositivos que devem ser usados para teste. As técnicas

Tabela 11 – Técnicas e dados usados para a seleção de dispositivos móveis.

Autor	Objetivo	Técnica	Descrição
[AF5] Khalid2014	Seleção de dispositivos	Mineração de dados	Comentários dos usuários (Google Play Store)
[AB9] Vilkomir2014	Seleção de dispositivos	Método Each-choice	5 Características de dispositivos (Device type, brand, OS version, resolution, RAM) Informação de 4000 disp. (Amazon.com)
[AF1] Cheng2015	Número de dispositivos	Algoritmo Genético	5 Características de dispositivos (platform, screen size, resolution, connection, market) Informação de 2500 disp. (site chines)
[AB12] Zhang2015	Seleção de dispositivos	Algoritmo K-means	4 Características de dispositivos (OS version, camera, screen size, resolution)
[AF12] Vilkomir2018	Número de dispositivos	Método Randômico e Cobertura de características	5 Características de dispositivos (OS version, brand, screen size,resolution, RAM)
[AF10] Lu2016	Seleção de dispositivos	Mineração de dados	Uso dos dados da App Indica os disp. que deve usar uma determinada App.

usadas para propor uma solução são diversas, tais como, a mineração de dados, uso de algoritmo genético, métodos combinatórios e randômico. A Tabela 11 apresenta um breve resumo de cada uma delas.

No detalhe de cada trabalho podemos observar um aspecto em comum, todas as pesquisas usam informações relacionadas à os dispositivos ou dados coletados dos usuários, no entanto, o objetivo por trás disso é usar os dispositivos para testar uma app e auxiliar na compatibilidade da app em diversos dispositivos. Porém, as informações da app não são consideradas em nenhum dos trabalhos. Rubinov e Baresi (2018) sugerem entender a estrutura interna das apps para uma melhor seleção dos dispositivos para o teste.

3.6.4 QP3: Seleção de dispositivos móveis na prática

O objetivo principal desta questão foi identificar o estado-da-arte na indústria em relação a seleção de dispositivos móveis. Assim, para identificar os critérios mais frequentes utilizados pelos profissionais na seleção de dispositivos móveis, foi revisado como tal prática é relatada no GL.

O processo de seleção de dispositivos para teste é uma atividade muito comum, pois é uma atividade essencial antes da execução de qualquer teste. Normalmente os desenvolvedores usam seus dispositivos, porém em algumas empresas maiores um conjunto de dispositivos é usado para testes.

A informação sobre como os dispositivos foram selecionados dentro de uma organização não é acessível publicamente. No entanto, os resultados encontrados na literatura cinzenta apresentam um conjunto de questões que podem ajudar a obter uma

resposta para este processo. As fontes selecionadas (Tabela 7) para análise tiveram os seguintes tópicos, a lista contém o título original:

- GL1 - Qatestlab: *How to Choose Devices for Mobile Testing?*
- GL2 - 99tests: *Top 5 Parameters For Choosing Devices For Mobile App Testing*
- GL3 - Coveros: *How Do I Choose Mobile Devices for Testing?*
- GL4 - SauceLabs: *How to Choose Mobile Devices for Testing?*
- GL5 - Avantica: *Selecting Mobile Devices for Testing*
- GL6 - Plusqa: *Top Devices for Android Testing in 2018*
- GL7 - Test48: *5 Simple Guidelines For Picking The Right Mobile Testing Devices*
- GL8 - BrowserStack: *Test on the right mobile devices*
- GL9 - EasyQA: *How to test mobile application*
- GL10 - Medium: *Test on the Right Devices*
- GL11 - Medium: *Choose your Test Devices Wisely*
- GL12 - Qatestlab: *Top Android Devices to choose for Testing Apps in 2019*
- GL13 - Plusqa: *The Top Devices for Android App Testing in 2019*
- GL14 - Avantica: *Testing: Choosing Mobile Devices*
- GL15 - Plusqa: *Top Devices for Android App Testing in 2021*

Esses tópicos permitiram identificar quais características ou critérios são utilizados pelos profissionais. Foram identificados 16 características apresentadas na Tabela 12 que são usadas para a seleção de dispositivos móveis. Os resultados foram divididos em duas categorias, da mesma forma que nos resultados do Snowballing: *hardware* e *software*, e foi adicionada uma nova categoria chamada *estatística*.

A nova característica é muito comum entre os profissionais no contexto da indústria, no entanto não está presente no contexto da literatura acadêmica. Essa é uma

lacuna que precisa de mais pesquisas; como exemplo, a característica "Popularidade" é dinâmica, pois um dispositivo móvel é popular por um certo tempo e depois se torna um dispositivo antigo, obsoleto ou fora do mercado.

Em relação ao *Sotfware*, as características mais citadas é a versão do sistema operacional SO. A principal diferença entre “SO” e “SO mais usado” é que o SO cobre pelo menos um dispositivo com um SO diferente (Ex. *Android, iOS*) e o "SO mais usado"seleciona apenas um SO dentre os OS disponíveis (Ex. Android); entanto que a "versão do SO mais usada"escolhe dentre as versões de cada SO (Ex. Android 11, iOS 13). Para a categoria de *Hardware*, o tamanho da tela e os dispositivos novos e próximos do lançamento são os mais citados pelos profissionais. O tamanho da tela ainda é uma característica da fragmentação e está intimamente relacionado aos novos dispositivos. Por fim, para a categoria *estatística*, a popularidade e a participação de mercado foram as mais citadas; a popularidade representa o 56% do total.

Um ponto interessante é que apenas 4 fontes GL3, GL5, GL6, e GL11 indicam características relacionadas ao aplicativo móvel (público-alvo e dispositivos direcionados ao objetivo do aplicativo). Portanto, observa-se que, mesmo na indústria, poucas fontes consideram as informações do aplicativo para a seleção dos dispositivos.

3.7 Discussão

De acordo com as informações obtidas na seção anterior, é importante focar em dois aspectos principais: As informações sobre as características dos dispositivos e as características das apps.

Características dos dispositivos: Selecionar dispositivos para teste implica identificar os critérios com os quais serão avaliados para um dispositivo ser selecionado. Todos os trabalhos apresentados na Tabela 11 usam informações relacionadas ao dispositivo. As técnicas usadas pelos autores são diversas, porém algumas delas já não são aplicáveis devido a sua natureza ou a sua disponibilidade.

A proposta de Khalid et al. (2014), que minera os comentários e os dispositivos dos usuários, não é viável devido ao fato de que as informações sobre os dispositivos não

Tabela 12 – Características mais usadas para a seleção de dispositivos

Características	**Fonte**	**Contagem**
	Software	
OS version	GL8; GL7; GL4; GL2; GL9; GL11	6
OS	GL8; GL2; GL11	3
Most used OS	GL4	1
Most used OS version	GL4	1
	Hardware	
Screen size	GL8; GL6; GL4; GL1; GL11; GL12; GL13; GL15	8
New/Upcoming devices	GL8; GL6; GL1; GL12; GL13; GL15	6
Manufacturers	GL8; GL7; GL2; GL4; GL11	5
Hardware	GL2; GL5; GL3; GL11; GL14	5
Old devices	GL6; GL1; GL12; GL13; GL15	5
Network	GL7; GL2; GL1; GL12	4
Resolution	GL8; GL7; GL11	3
	Statistics	
Popularity	GL8; GL7; GL9; GL5; GL6; GL1; GL13; GL15; GL14	9
Target audience	GL3; GL5; GL6; GL11; GL13; GL15; GL14	7
Market share	GL8; GL7; GL2; GL9; GL1; GL11	6
Devices for App goal	GL3; GL5; GL14	3
Devices cause problems	GL1; GL12	2

estão mais disponíveis. É possível identificar o modelo do dispositivo usado somente se o usuário descreve dentro do comentário qual foi o modelo de seu dispositivo; porém nem todos os usuários descrevem essa informação.

A proposta de Cheng et al. (2015) coletou as informações de 4000 dispositivos do site Amazon.com. O trabalho não indica se as informações coletadas eram atuais ou se havia algum tipo de filtro. No mercado móvel os dispositivos são atualizados e desfasados constantemente e é importante considerar a utilidade e atualização dos dados que são usados para a seleção de dispositivos. Os autores usaram o algoritmo genético para a identificação do número de dispositivos a serem usados. Portanto, faz-se necessária a recopilação de dispositivos atualizados para uma melhor tomada de decisão.

No caso da proposta de Lu et al. (2016), os autores mineraram dados de uso de uma app, o qual só é possível quando a app já foi publicada e usada pelos usuários.

Uma nova app não poderia minerar seus dados de uso sem ser usada antes. O objetivo desta proposta é atuar antes da publicação da app.

As características dos dispositivos que mais foram usados nos estudos foram: Sistema operacional (SO), Tamanho e Resolução de Tela. Outras características que também foram consideradas foram: Marca, RAM, Tipo de conexão e mercado (VILKOMIR; AMSTUTZ, 2014; CHENG et al., 2015; ZHANG et al., 2015; VILKOMIR, 2018).

Características das apps: Dentro dos trabalhos sobre seleção de dispositivos, nenhum trabalho usa as características da app (*AUT – App Under Test*). Porém, Rubinov e Baresi (2018) sugerem entender melhor a estrutura interna das apps para uma melhor seleção dos dispositivos para o teste. No trabalho de Silva et al. (2018) as características da app foram usadas para ter uma visão do código de teste. O trabalho de Kaur e Kaur (2019) analisa as características da app para estimar o esforço do teste; as cinco características que são mais abordadas no estudo são as Interfaces de entrada (*voice, touch, keypad*); Memória limitada, Bateria, Conexão e Portabilidade em diferentes dispositivos.

Silva et al. (2018) identificaram algumas principais características que são relevantes para o teste. Dentre elas, estão Conectividade, Interface de Usuário Gráfico (GUI), Sensores e as Múltiplas configurações que um dispositivo pode ter. A proposta também disponibiliza a estrutura da app.

Por outro lado, os resultados na indústria (ver Tabela 12) mostram que apenas algumas fontes recomendam usar as informações do aplicativo, como o público-alvo do aplicativo e o objetivo do aplicativo. Uma possível direção é considerar quais recursos e sensores do dispositivo o aplicativo usa ao selecionar os dispositivos para testar. Devido a que as informações das app móveis ainda não estão sendo usadas para a seleção do dispositivo, há uma demanda por uma taxonomia dos recursos das apps que poderia contribuir para a seleção dos dispositivos. Acreditamos que os resultados aqui apresentados podem ser usados para apoiar esta tarefa.

Portanto, considerando que o uso das características das apps ainda não está sendo usada para a seleção de dispositivos, faz-se necessária a criação de uma taxonomia das características das apps que podem contribuir na seleção de dispositivos.

Para esta etapa inicial da proposta, o *baseline* será a implementação de um algoritmo genético usando as informações dos dispositivos e apps que foram as mais usadas pelos estudos citados antes. No próximo capítulo será apresentada a abordagem proposta para esta pesquisa.

3.8 Conclusões do capítulo

Percebe-se que a partir da análise das questões de pesquisa e as oportunidades de pesquisa é necessário estudar e entender as características das apps, identificar quais dos métodos apresentam melhor resultado para o problema de seleção de dispositivos, devido a sua importância na identificação de falhas nas apps que estão relacionados a dispositivos específicos. Um dos maiores desafios na execução do teste de compatibilidade está relacionado às falhas específicas em dispositivos móveis, ou seja, as apps falham em modelos específicos de dispositivos algumas vezes relacionado a uma característica de hardware específica do dispositivo ou a uma modificação no software modificado (Sistema operacional modificado).

O propósito dessa revisão foi identificar quais eram as áreas de pesquisa sobre teste de compatibilidade. Além de identificar as oportunidades de pesquisa dentro desta área. No capítulo seguinte será apresentada a proposta para seleção de dispositivos móveis para teste de compatibilidade.

4

DeSeCT: Seleção de dispositivos para Testes de Compatibilidade em aplicações móveis

Neste capítulo é apresentada uma abordagem para apoiar a seleção de dispositivos móveis para a realização de testes de compatibilidade, assim como resultados parciais dos experimentos realizados. Este capítulo foi baseado nos resultados do artigo publicado no V Simpósio Brasileiro de Teste de Software Sistemático e Automatizado (SAST20) - DOI: 10.1145/3425174.3425215.

4.1 Visão geral da abordagem

A seleção de dispositivos para teste impacta em várias etapas do planejamento dos testes. Na etapa de configuração do ambiente, é definido quais serão os dispositivos que serão usados para os testes e essa informação é usada na modelagem e implementação do teste. Para apoiar a seleção de dispositivos, é necessário conhecer quais são as características desses dispositivos e quais são os mais relevantes para serem usados nesta tarefa.

Neste contexto, apresentamos a abordagem **DeSeCT** *(do inglês, **De**vices **Se**lection for **C**ompatibility **T**esting)* - Seleção de dispositivos para testes de compatibilidade. O objetivo da abordagem *DeSeCT* é apoiar na seleção de dispositivos móveis para a

realização de testes de compatibilidade que contribuam com a qualidade dos testes e a identificação de possíveis falhas em dispositivos específicos.

DeSeCT usa as informações das características dos dispositivos e as informações da app a ser testada (AUT - *App Under Testing*), pois essas informações podem contribuir a uma melhor tomada de decisão sobre a seleção de quais dispositivos devem ser usados nos testes da app.

A versão final da abordagem **DeSeCT** é apresentada na Figura 10 contendo cinco etapas que são: (1) *Dataset* - contem as informações de diversos dispositivos móveis que representam o espaço de busca para as possíveis soluções; (2) Identificação de atributos e características da app e dos dispositivos - extração das informações necessárias dos dispositivos e da app; (3) Filtro do *dataset* - nesta etapa é usado as informações obtidas na etapa anterior; (4) Informação adicional - informações relacionadas ao objetivo da organização ou usuário são usadas nesta etapa e finalmente; (5) Processo de seleção - aplicação do algoritmo NSGA-II para a salação de um conjunto de soluções que serão apresentadas ao usuário.

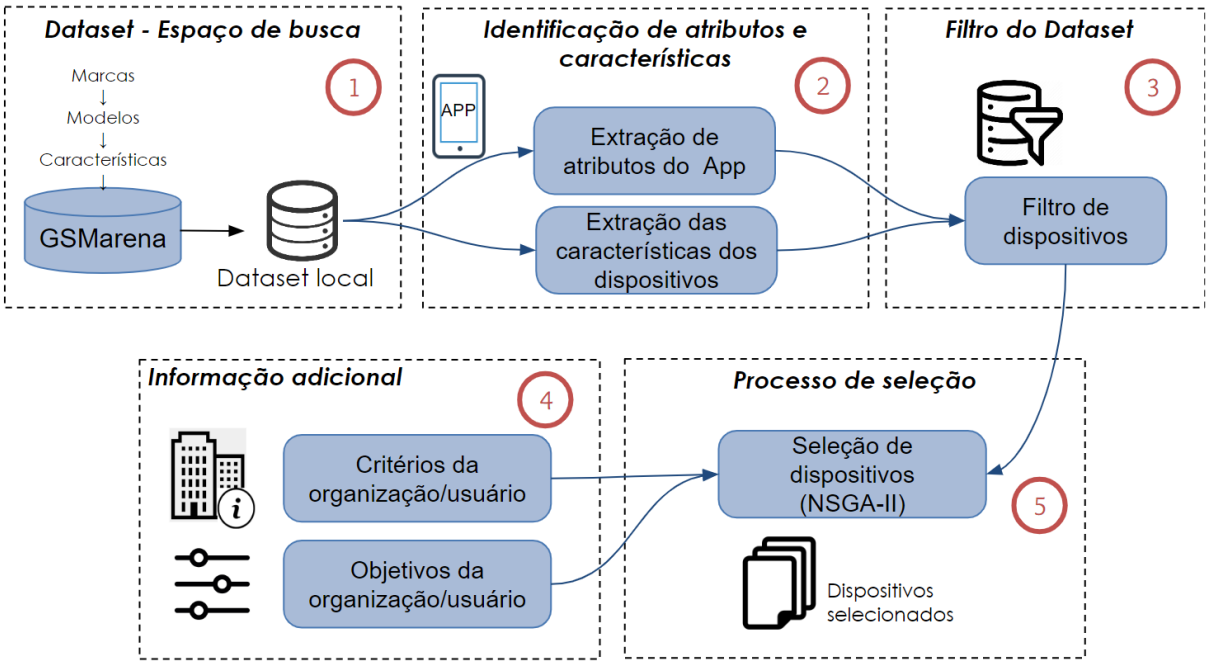


Figura 10 – DeSeCT - Seleção de dispositivos para testes de compatibilidade.

Para chegar nesta versão foram realizados diversos estudos que serão detalhados a seguir. A visão geral da primeira versão da abordagem **DeSeCT** está composta por três etapas principais, apresentadas na Figura 11. Inicialmente, na Etapa 1, o conjunto de

dados com as informações dos dispositivos móveis foi coletado da Internet por meio de um *crawler* construído. Como resultado tivemos uma *dataset* local que será usado como espaço de busca de possíveis soluções nas etapas seguintes. Na Etapa 2, a abordagem *DeSeCT* usa as informações coletadas no passo anterior para identificar quais serão as características dos dispositivos que serão usadas. Logo, *DeSeCT* usa o arquivo binário da app para extrair os atributos que caracterizam a AUT. Finalmente, na Etapa 3, o *dataset* é filtrado usando os dados coletados da Etapa 2 e é executado um algoritmo genético desenvolvido nesta pesquisa para a seleção de dispositivos móveis para o teste. Nas próximas seções serão explicadas cada etapa que compõe a abordagem $DeSeCT$ com mais detalhe.

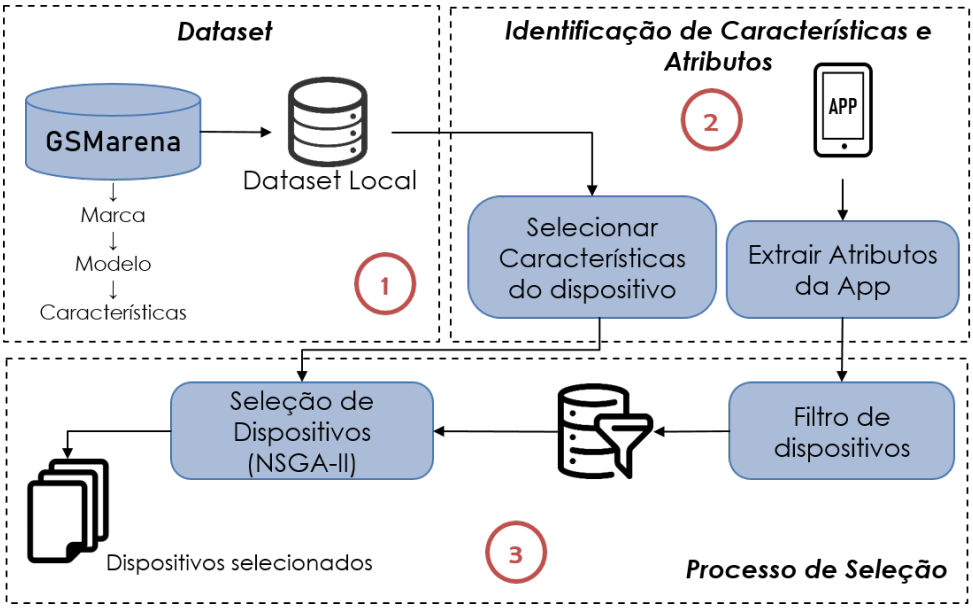


Figura 11 – Visão geral da abordagem DeSeCTv1.

4.1.1 Etapa 1: Coleta do *Dataset*

Esta etapa envolve a coleta de informações atualizadas dos dispositivos móveis da Web, representada na Figura 11 (Etapa 1). Para isso, foi implementado um rastreador (*crawler*) na web que extrai dados, como modelo do dispositivo, sistema operacional, tamanho da tela e muitas outras informações. Foi incluída como primeira etapa devido aos dispositivos móveis serem constantemente atualizados no mercado. O conjunto de dados foi rastreado do site da *GSMArena*. Este *website* foi adotado em estudos anteriores

(NOEI et al., 2018; HALPERN et al., 2015), e fornece informações sobre dispositivos móveis antigos/novos, bem como mais de cem marcas vendidas em todo o mundo. A abordagem nos permite atualizar o conjunto de dados a qualquer momento. A Figura 12 mostra o processo de coleta de informações de dispositivos móveis.

Primeiro, foi usado o rastreador para coletar informações de 8200 dispositivos. Foram aplicados dois filtros; no primeiro filtro, todos os dispositivos com *status* "descontinuado" e "cancelado" foram descartados; Dispositivos cancelados são aqueles que não chegaram ao mercado e seu lançamento foi cancelado, enquanto os dispositivos descontinuados são aqueles que não possuem novas atualizações e podem ficar fora do mercado posteriormente. Portanto, apenas os dispositivos com o *status* "disponível"permaneceram. O segundo filtro está relacionado à plataforma; para este estudo, o foco foi Android, a plataforma com mais fragmentação no mercado. O conjunto de dados final é composto por 2.559 dispositivos móveis.

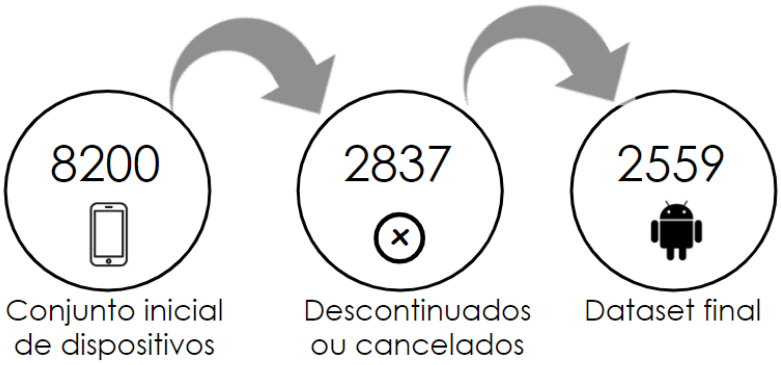


Figura 12 – Etapas do pré-processamento do *dataset*.

4.1.2 Etapa 2: Identificação de características e atributos

Essa etapa contém duas atividades, na Figura 11 (Etapa 2): selecionar as características dos dispositivos e extrair os atributos da app.

Para a **extração das características** dos dispositivos, foram listadas as características que foram mais citadas nos trabalhos correlatos (ZHANG et al., 2015; VILKOMIR; AMSTUTZ, 2014; CHENG et al., 2015; VILKOMIR, 2018) e as que foram mais citadas pela indústria.

- **SO:** para identificar qual versão e plataforma usa o dispositivo móvel. As versões da plataforma Android [1] estão entre 1.5 e 13;
- **API:** versão do API ou nível da API é um valor inteiro que identifica exclusivamente a revisão da API da estrutura oferecida por uma versão da plataforma Android;
- **Display:** resolução e tamanho da tela são as principais características que o usuário leva em consideração ao comprar um celular. Para nossa abordagem, usamos ambos os parâmetros, a resolução e o tamanho da tela em polegadas;
- **Tecnologia da rede:** as redes que estão sendo consideradas são 2G, 3G, 4G, 5G;
- **Marketshare do tamanho de tela:** a plataforma Android mostra informações sobre o número relativo de dispositivos que possuem uma configuração de tela específica, definida por uma combinação do tamanho de tela e densidade [2]. Para nossa abordagem, usamos esses dois valores do nosso *dataset* para identificar a porcentagem de mercado por cada dispositivo.

Com o objetivo de confirmar quais características seriam usadas, foi feita uma análise de correlação (COHEN, 1988) entre as características selecionadas. Para a análise de correlação, foram extraídas de forma aleatória as informações de um conjunto de 800 dispositivos do *dataset*, que representam 95% de confidencialidade com uma margem de erro de +-2.87%. Os resultados mostraram uma alta correlação positiva (0,918) entre as características "API"e "SO". Devido a esse resultado, foi decidido não usar a característica "SO", pois as apps móveis de forma geral costumam usar os valores da "API".

Para a **extração de atributos** da App, foi usado o arquivo binário da aplicação (APK) para extrair informações da App *androidmanifest*[3]. Este arquivo contém os atributos da App usados para cada recurso. Esta informação permite pré-filtrar os dispositivos, evitando a seleção de dispositivos não-usáveis ou incompatíveis para a execução dos testes.

[1] https://source.android.com/setup/start/build-numbers
[2] https://developer.android.com/about/dashboards
[3] *The manifest file describes essential information about the app to the Android build tools; it is required to declare the permissions, and hardware/software features the app requires.*

Alguns atributos usados são *usesMinSdkVersion, usesTargetSdkVersion, e usesMaxSdkVersion,* que estão relacionados à **API**; *supportsScreensSmall, supportsScreensNormal, supportsScreensLarge, e supportsScreensXlarge* que estão relacionados a ***Display*** e ***Marketshare do tamanho de tela***. Para esse processo, foi usada a ferramenta *ApkAnalyzer*[4] para extrair essas informações; essas informações logo são usadas para fazer um filtro no *dataset*. Assim, estão sendo usadas as informações dos atributos da app e as características dos dispositivos.

4.1.3 Etapa 3: Processo de Seleção

Esta etapa inclui duas fases, o filtro de dispositivos e a seleção de dispositivos, apresentados na etapa 3 da Figura 11. Para o **"Filtro dos dispositivos"**, são usadas as informações dos atributos que foram extraídas do arquivo *manifest* da app descrito na seção anterior. Assim, é gerada uma nova versão do conjunto de dados (*dataset*) compatível com a app sob teste. Este novo *dataset* é usado para a próxima fase, **"Seleção de dispositivos"**.

A fase da **"Seleção de dispositivos"** usa o algoritmo NSGA-II (DEB et al., 2002), que foi selecionado por ser o algoritmo que é o estado-da-arte para otimização evolutiva multiobjetivo (CHANDRASEKARAN et al., 2019; MAO; HARMAN; JIA, 2016). Ele é eficiente para resolver problemas de otimização multiobjetivo (SHAO et al., 2016) e também na área de engenharia de software e testes (Di Nucci et al., 2018; Wei et al., 2017; GRANDE; RODRIGUES; DIAS-NETO, 2014). A abordagem foi implementada em cima do *DEAP framework* (FORTIN et al., 2012). A escolha no uso deste *framework* foi devido a estar disponibilizado de livre acesso e estar em constante atualização pela comunidade, além de já ter o NSGA-II embutido.

De acordo com Harman e Jones (HARMAN, 2007; HARMAN; JONES, 2001), existem duas etapas-chave para problemas multiobjetivos: (1) a representação do problema e (2) a definição da função de aptidão *(fitness function)*. Assim, esses componentes principais são detalhados nas seções a seguir.

[4] https://github.com/MartinStyk/ApkAnalyzer

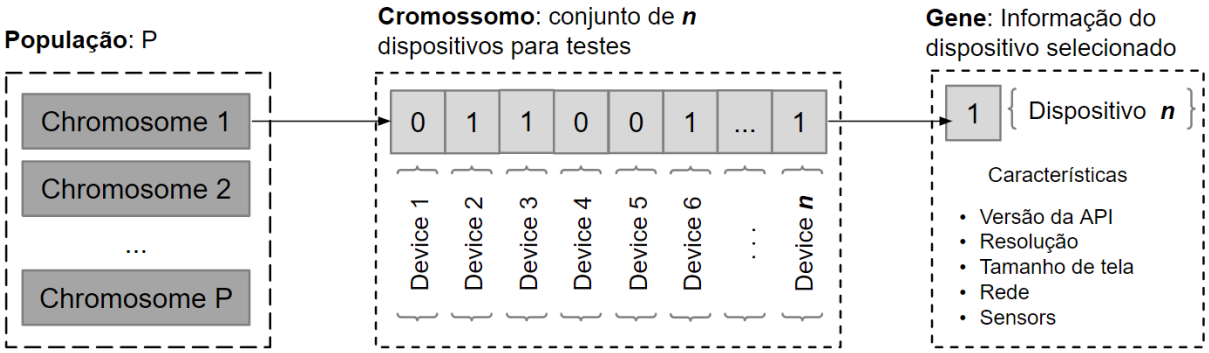


Figura 13 – Representação genética do indivíduo no contexto DeSeCT.

4.1.3.1 Problema de representação

Na fase de seleção de dispositivos para teste, um grande conjunto de dispositivos deve ser levado em consideração (nosso *dataset*) a partir do qual um subconjunto deve ser selecionado e deve atender dois objetivos: (1) baixo custo (isto é, número de dispositivos selecionados), e (2) alta cobertura (ou seja, características diferentes dos dispositivos). Como foi usado um algoritmo genético, os dispositivos candidatos precisam ser codificados. Assim, procura-se maximizar a cobertura minimizando o esforço e a quantidade de dispositivos selecionados. A representação do problema foi baseada em Cheng et al. (2015).

A Figura 13 mostra a representação que gera uma população P de cromossomos com o mesmo comprimento; um cromossomo é um vetor de números binários que representa um conjunto de dispositivos; o comprimento n de um cromossomo corresponde ao tamanho do *dataset*; cada cromossomo contém vários genes. Finalmente, um gene representa um dispositivo de teste que consiste em valores binários (0 e 1). Esses valores são projetados para identificar se um dispositivo de teste foi selecionado (valor 1) ou não (valor 0). Além disso, cada gene tem uma lista de recursos do dispositivo.

4.1.3.2 Função de aptidão *(Fitness Function)*

Esta seção foi baseada no trabalho de Cheng et al. (2015). A função de aptidão mede a adequação de um cromossomo para atingir o objetivo. O operador *crossover* troca genes de dois cromossomos e cria dois novos cromossomos. O operador de mutação altera um gene em um cromossomo e cria um novo cromossomo. Este processo é repetido

até um critério de parada. A abordagem *DeSeCT* usa o número de gerações como um critério de parada. Como o problema é multiobjetivo, foram definidos dois objetivos: Um objetivo de minimização e um de maximização.

(i) Objetivo de minimização visa reduzir o custo de teste com a menor quantidade de dispositivos de teste possível. Sua função de aptidão é representada pela equação (4.1).

$$MinFit = \sum_{i=1}^{n}(D_i) \tag{4.1}$$

Onde i significa o índice de cada dispositivo móvel. D_i pode ser "1"se o dispositivo foi selecionado ou "0" se não foi selecionado. Logo, n representa o tamanho do cromossomo. Esta função realiza a soma de todos os dispositivos selecionados com valor 1. O resultado representa o número de dispositivos selecionados para o teste. Quanto menor o resultado, melhor, pois se tem um conjunto menor de dispositivos para testar.

E (2) **objetivo de maximização**, cujo objetivo principal é obter uma alta cobertura de recursos do dispositivo; no primeiro estágio foram definidos cinco características, descritas na Seção 4.1.2 (API, Resolução, Tamanho de tela, tecnologia da rede, *Marketshare* do tamanho de tela), para medir a cobertura.

A equação (4.2) representa o objetivo de maximização do objectivo; onde i representa o índice de cada dispositivo móvel e n representa o comprimento do cromossomo. D_i, representa o dispositivo selecionado; f, representa o número (cinco) de recursos a serem avaliados; $CovFeat_j$, a cobertura total para as cinco características; e W, representa o peso associado a cada característica; também a soma de todos os pesos é igual a 1 $(W_1 + W_2 + \ldots + W_f)$; os pesos podem ser vistos na Tabela 13.

A cobertura total ($CovFeat$) é calculada baseada na equação (4.3), que representa a cobertura para cada recurso (API, resolução, tamanho de tela, tecnologia da rede, *marketshare* do tamanho de tela). Para calcular o valor do dividendo, foram identificados os dispositivos selecionados D_i, logo é calculada a cobertura por recurso $DevicesCoveredByFeature$. Para calcular o valor de divisor, a abordagem obtém o valor da quantidade total (únicos) por recurso.

$$MaxFit = \sum_{i=1}^{n} (D_i * CovFeat(D_i)) \tag{4.2}$$

$$CovFeat(D_i) = \sum_{j=1}^{f} \left(\frac{DevicesCoveredByFeature_j(D_i)}{TotalValuesByFeature_j} * W_j \right) \tag{4.3}$$

4.2 Prova de conceito

Um primeiro estudo foi executado como prova de conceito da abordagem apresentada neste trabalho. Para avaliar a abordagem, foram definidas as seguintes questões de pesquisa (QP):

- **QP1:** Qual é o impacto que tem o uso dos atributos das Apps na seleção de dispositivos para teste usando o $DeSeCT$?
- **QP2:** Podem ser definidas configurações diferentes para ajustar o desempenho do $DeSeCT$?

O principal objetivo da **QP1** é identificar se existem diferenças entre "usar"e "não usar" as informações da App ao selecionar dispositivos para teste. Assim, é comparada $DeSeCT$ com a abordagem do Cheng et al. (2015), ao qual será chamada de $Baseline$. De forma específica, é analisado se filtrar o *dataset* usando os atributos da App que está sob teste possui um impacto positivo ou não. O mesmo *dataset* foi usado para as duas abordagens, o primeiro sem as informações da app ($Baseline$) e o segundo, com as informações da app ($DeSeCT$).

Nos dois casos, a seguinte configuração foi usada: Tipo de algoritmo genético: *MultiObjetivo NSGA-II*; Codificação: *Binary*; Cruzamento: *Two point*; Mutação: *Flip Bit Based*; Processo de seleção: *Elitismo*; a probabilidade usada foi de 80% de taxa de cruzamento e 1% de probabilidade de mutação; o tamanho da população inicial foi 200 e o número máximo da geração foi definido como 1000, 5000 e 10000. Os outros parâmetros foram a configuração padrão do *framework DEAP*. Além disso, em ambas as abordagens, não foi limitado o número máximo de dispositivos selecionados. Como o algoritmo

fornece um conjunto de soluções a partir das quais é possível escolher qual será usada; foi usada a ferramenta *"the hall of fame"* do *framework DEAP*, que contém o melhor indivíduo que já viveu na população durante a evolução do algoritmo. Para o estudo foi usado a App EasyTaxi[5]. A Figura 14 mostra as informações que foram usadas na abordagem $DeSeCT$.

Para responder a **QP1**, foram coletadas as seguintes métricas: a cobertura da característica, o número de dispositivos selecionados e o número de dispositivos considerados utilizáveis e inutilizáveis. Para o contexto da proposta, um dispositivo *"utilizável"* é um dispositivo compatível com a app que está sob teste, enquanto que um dispositivo *"inutilizável"* não é compatível com a app sob teste.


```
[{
  "fileName": "easy-taxi-9-9-0-b26.apk",
  "fileSize": 9574920,
  "dexSize": 5990764,
  "arscSize": 1247488,
  "androidManifest": {
    "packageName": "br.com.easytaxi",
    "usesMinSdkVersion": "15",
    "usesTargetSdkVersion": "23",
    "usesMaxSdkVersion": "",
    "supportsScreensSmall": true,
    "supportsScreensNormal": true,
    "supportsScreensLarge": true,
    "supportsScreensXlarge": true
  }
}]
```


Figura 14 – Informações da app Easy Taxi.

Em relação a **QP2**, segundo Arcuri e Fraser (2011), o ajuste dos parâmetros tem impacto no desempenho dos algoritmos de busca. Assim, foram testadas diferentes configurações no algoritmo NSGA-II do $DeSeCT$ e os resultados foram analisados.

Para otimizar os primeiros resultados, foi modificada a forma de geração dos valores dos cromossomos, na QP1 os valores dos cromossomos foram criados aleatoriamente. Considerando o fato de que este problema de modelagem usa valores binários ("0"e "1"), há uma chance de 50:50 de que, durante a criação da população inicial, o número de dispositivos selecionados seja de mil dispositivos. Isso ocorre porque o conjunto de dados tem mais de dois mil dispositivos. Assim, se a população inicial é muito grande, existe a possibilidade de que a função de minimização não seja alcançada tão

[5] https://apkpure.com/easy-taxi/com.easytaxi.easyuser

facilmente. Portanto, não é boa para o desempenho do $DeSeCT$. Este comportamento foi verificado executando o algoritmo várias vezes.

Para a **QP2**, foi definido um número máximo de dispositivos selecionado na primeira etapa de seleção do algoritmo (cromossomos com valor "1"), como mostrado na Figura 15. Assim, para a execução do algoritmo foram definidos 4 valores iniciais (20, 50, 100 e 250) como o número máximo de dispositivos selecionados no início do algoritmo. Fora dessa mudança, as outras configurações foram as mesmas do primeiro estudo.

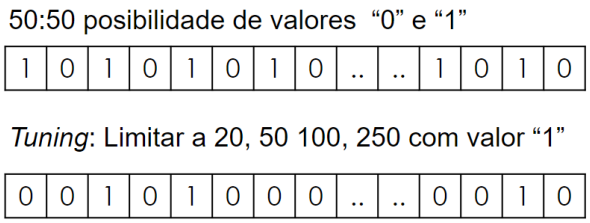


Figura 15 – Ajuste da população inicial dos cromossomos.

4.2.1 Implementação

A abordagem $DeSeCT$ foi implementada sobre o *Framework DEAP* (FORTIN et al., 2012) para a seleção de dispositivos de teste. Este Framework é amplamente utilizado na literatura técnica (Vandana; Singh, 2017; MAO; HARMAN; JIA, 2016). Para essa proposta, a abordagem $DeSeCT$ foi comparada com a abordagem de Cheng et al. (2015). Pelo fato de que o *dataset* da outra abordagem não estava disponível, usamos o *dataset* provido nesta pesquisa e implementamos o $Baseline$ seguindo a descrição no artigo.

A abordagem de Cheng et al. (2015) usa a característica "*plataforma*"; entretanto, como o *dataset* nesta pesquisa contém dados unicamente da plataforma *Android*, foi decidido não usar essa característica. Para esse estudo, foram definidos os objetivos de cobertura e pesos por características que são apresentadas na Tabela 13, esse valores foram baseados no $Baseline$. Além disso, esta tabela mostra a quantidade única de cada característica com base no *dataset*.

O pacote experimental contendo o dataset, implementação e dados brutos cole-

Tabela 13 – Cobertura de objetivos e pesos.

Item	**Quantidade**	**Cobertura**	**Peso**
Resolução	80	70%	0.2
Tamanho de tela	129	70%	0.2
API	16	70%	0.3
Rede (e.g. 4G)	4	70%	0.1
Android Marketshare	-	60%	0.2

tados está disponível online[6].

4.3 Análise dos resultados

4.3.1 QP1: Impacto no uso dos atributos da App

Para responder ao **QP1**, $DeSeCT$ foi comparada com o $Baseline$. Foi analisado o número de dispositivos selecionados e os dispositivos inutilizáveis para cada abordagem; também, comparou-se a cobertura de cada recurso. Os resultados de ambas as abordagens são apresentados na Tabela 14; Foi realizado com diferentes tamanhos de gerações (1000, 5000, 10000). Para o objetivo de minimização, $DeSeCT$ selecionou menos dispositivos em todas as três gerações, provendo uma redução de 20,5%, 23,2% e 22,4% para 1000, 5000 e 10000 gerações respectivamente, em comparação com o $Baseline$, que tem dispositivos inutilizáveis para o aplicativo usado neste estudo.

Em relação à cobertura de características (objetivo de maximização), os objetivos definidos na Tabela 13 foram alcançados para Resolução e Tamanho de recursos e o $DeSeCT$ atingiu 100% de cobertura. Para o recurso API, o $Baseline$ não atingiu o objetivo e a cada geração foi observado o mesmo percentual; isso se deve ao fato de haver apenas 16 tipos de APIs diferentes e em cada geração apenas 9 foram cobertos. Para a participação de mercado do tamanho da Rede e da Tela, tanto o $Baseline$ quanto $DeSeCT$ têm os mesmos resultados. Esses resultados mostraram que, usando informações de dispositivos móveis, pode-se obter uma melhor cobertura com menos dispositivos para teste.

Visando analisar o impacto de não usar as informações da app, uma análise extra

[6] https://github.com/MobileCompatibilityTesting/DeSeCT

Tabela 14 – Resultados para a questão QP1.

	Baseline			**DeSeCT**		
Gerações	1,000	5,000	10,000	1,000	5,000	10,000
Dispositivos selecionados	728	650	578	703	603	530
Dispositivos inutilizáveis	156	135	105	0	0	0
Características (Max)						
Resolução	80%	82.5%	80%	100%	100%	100%
Tamanho de tela	76.7%	79.8%	79.1%	99%	100%	100%
API	56.2%	56.2%	56.2%	100%	100%	100%
Redes	100%	100%	100%	100%	100%	100%
Android Marketshare	97.2%	97.2%	97.2%	97.2%	97.2%	97.2%

foi realizada para esses resultados. A Tabela 14 apresenta o número de dispositivos inutilizáveis para 1000, 5000 e 10000 gerações. O cálculo dos dispositivos inutilizáveis (não compatíveis com a app) foi baseado na equação 4.4; enquanto ao cálculo da porcentagem das coberturas por características foi baseado na equação 4.5.

$$DispositivosInutilizaveis = TotalDispSelecionados - DispCompativeis \tag{4.4}$$

$$CoberturaCaracterstica = \frac{NumeroCaracteristica(DispSelecionados)}{TotalCaracteristicaDataset} * 100 \tag{4.5}$$

Os dados dos dispositivos inutilizáveis representam 18%, 17% e 15%, respectivamente para cada geração. Estes resultados mostram a importância de usar as informações da app para selecionar dispositivos para teste de compatibilidade. Outro benefício é o *tempo*, os dispositivos inutilizáveis consomem tempo que poderiam ser usados em outras atividades de teste. A cobertura das características da **DeSeCT** foi maior em três das cinco características e em duas (redes e *Android Marketshare*) foram iguais ao *Baseline*. Assim respondendo a QP1 concluímos com o seguinte resumo:

"O uso das informações da app permite aos testadores descartar dispositivos que não são suportados pela app, reduzindo assim o número de dispositivos e obtendo uma maior cobertura de características."

4.3.2 RQ2: Modificando as configurações do *DeSeCT*

Como acontece com qualquer algoritmo genético, é possível testar diferentes configurações para encontrar melhores resultados. Para este estudo específico, apenas a geração de cromossomos foi ajustada. Assim, foi definida uma nova forma aleatória de geração dos valores dos cromossomos, limitando os valores máximos iniciais a 20, 50, 100 e 250 dos dispositivos selecionados (cromossomos com valor "1"). Assim, este segundo estudo foi executado para cada valor.

A Tabela 15, mostra os resultados com diferentes números de dispositivos selecionados no início da população (20, 50, 100 e 250) para cada geração. Resultados com 1000 gerações têm melhores resultados com 250 dispositivos selecionados inicialmente; Dessa forma, o objetivo para os quatro recursos definidos na Tabela 13 foi alcançado, exceto para o recurso *tamanho de tela*.

Além disso, os dispositivos selecionados (objetivo de minimização) foram melhores que o primeiro estudo, com uma diferença de 610 dispositivos. Os resultados para 5000 e 10000 gerações e 250 dispositivos selecionados são menores que os resultados para 1000 gerações. Utilizando o algoritmo com uma população inicial de 250 dispositivos selecionados, $DeSeCT$ obteve melhores resultados de cobertura, de acordo com os objetivos (Tabela 13). O resultado do objetivo de minimização também foi melhor do que o primeiro estudo.

Dentre os cinco recursos avaliados, "*Tamanho da tela*" é aquele com o menor percentual de cobertura obtido. Analisamos os dados desta característica no *Dataset* e foi identificado que tem-se 129 tipos de tamanho de tela diferentes, dos quais 21 tipos representam 86,3% e 108 tipos representam 13,7% do total. Essa grande diversidade de tamanhos de tela pode ter impactado na baixa cobertura desta característica. Assim, uma nova análise poderia considerar apenas os 21 tipos de tela e considerar os outros tipos como *diferente*. Essa e outras mudanças serão observadas nos próximos estudos. Para responder a QP2 concluímos com o seguinte resumo:

"Definir a população inicial do algoritmo NSGA-II pode ajudar a obter melhores resultados."

Tabela 15 – Resultados de cobertura após modificações no $DeSeCT$ (QP2).

# Gerações	1,000				5,000				10,000			
# Dispositivos												
População inicial	20	50	100	250	20	50	100	250	20	50	100	250
Disp. Selecionados	28	45	56	93	31	43	50	80	23	38	52	79
Cobertura Caract.												
Resolução	34.8%	54.5%	66.7%	78.8%	37.9%	45.5%	60.6%	77.3%	30.3%	50%	66.7%	75.8%
Tamanho de tela	24.8%	40%	43.8%	68.6%	26.7%	39%	43.8%	67.6%	18.1%	31.4%	45.7%	67.6%
API	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Redes	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Marketshare	92.0%	96.7%	96.7%	96.7%	95.1%	96.8%	96.7%	96.7%	94.2%	96%	97.1%	96.8%

4.4 Conclusões e Contribuições

Os modelos de dispositivos móveis são cada vez mais diversificados, pois pertencem a fornecedores diferentes; realizar testes de compatibilidade torna-se mais difícil frente ao tamanho da diversidade. Este estudo explorou o potencial do uso de atributos de aplicativos móveis para selecionar dispositivos. Esta proposta tem como objetivo construir uma abordagem baseada em heurísticas de otimização para dar o suporte na escolha de dispositivos para teste de compatibilidade em apps.

Foi proposta uma abordagem $DeSeCT$ que usa um algoritmo genético multiobjetivo e inclui informações do aplicativo para selecionar dispositivos para testes de compatibilidade. Os resultados mostram que $DeSeCT$ pode efetivamente identificar dispositivos com mais cobertura de recursos em menos número de dispositivos e superar a abordagem $Baseline$ usada para comparação. Além disso, os resultados mostraram uma melhor eficiência ao usar informações de aplicativos móveis; evitando assim a seleção de cerca de 15% a 18% de "dispositivos inutilizáveis", por não atenderem às características da app. Nas próximas seções serão apresentados outros estudos empíricos e estudos na indústria visando validar e evoluir a abordagem proposta.

5

ESTUDO DE CASO E REFINAMENTO DA ABORDAGEM DESECT

Neste capítulo apresenta-se o planejamento, execução e análise dos resultados de um estudo de caso com duas partes: a primeira um estudo em maior escala para confirmar a prova de conceito apresentada no capítulo anterior e a segunda parte um estudo com a indústria.

5.1 Objetivo do estudo

Seguindo a metodologia de pesquisa adotada para o desenvolvimento da proposta $DeSeCT$, descrita na Seção 1.5, o seguinte passo é o estudo de viabilidade em um ambiente controlado. Este estudo tem duas partes, a primeira refere-se à avaliação do $DeSeCT$ num ambiente controlado e em maior escala, enquanto que a segunda parte é a avaliação do $DeSeCT$ frente à seleção de dispositivos móveis em uma organização de software na indústria.

Assim, a Tabela 16 apresenta o objetivo deste estudo, que foi baseado na abordagem GQM *(Goal-Question-Metric)* (BASILI; CALDIERA; ROMBACH, 1994).

Tabela 16 – Objetivo do estudo - GQM.

Analisar	a proposta DeSeCT
Com o propósito de	caracterizar
Em respeito	a relevância dos atributos da aplicação
Do ponto de vista do	pesquisador
No contexto de	teste de compatibilidade de apps móveis.

5.2 Planejamento do estudo

O planejamento deste estudo foi dividido em duas partes. Na primeira parte, com o resultado obtido na prova de conceito (Seção 4.2), foi necessário a realização de um estudo maior devido ao fato de que na prova de conceito foi usada apenas uma app para teste. Neste novo estudo, a proposta foi avaliada com um conjunto maior de apps e um *dataset* atualizado. Na segunda parte, foi feita a avaliação na indústria, foi seguido o processo real de seleção de dispositivos móveis de uma organização da indústria. Assim, foram geradas duas questões de pesquisa relacionadas ao estudo em maior escala e uma questão de pesquisa sobre a seleção de dispositivos na indústria. As questões de pesquisa foram as seguintes:

- **QP1: Qual é a importância do uso dos atributos da aplicação antes da seleção de dispositivos móveis para teste?** Trabalhos relacionados mostram que não há uso de informações de apps móveis para o processo de seleção de dispositivos para teste. No entanto, os resultados da prova de conceito mostraram a importância no seu uso. Portanto, esta QP1 irá analisar se o filtro do *dataset* usando as informações das apps é válido para uma diversidade maior de dispositivos.

- **QP2: Em diversas apps, o uso de diferentes configurações melhora a performance do $DeSeCT$?** Baseado no melhor resultado da prova de conceito, está QP2 irá analisar os resultados em diversas apps aplicando as mudanças de configuração na abordagem proposta com o objetivo de identificar se persiste uma melhoria na performance do $DeSeCT$.

- **QP3: Quais informações de uma organização de software na indústria podem ajudar no desempenho do $DeSeCT$?** O objetivo nesta questão de pesquisa é conhecer e analisar o processo de seleção de dispositivos para testes de compa-

tibilidade em uma organização de software na indústria do ponto de vista da pesquisa e identificar quais informações da organização de forma genérica podem ser incorporadas na abordagem proposta.

O número de dispositivos selecionados será usado como uma métrica para as três questões de pesquisa. Finalmente, todas as QPs foram baseadas no paradigma GQM.

5.2.1 Seleção dos dados

Para o estudo, foi usado um conjunto de apps (formato apk) obtido do repositório AndroZoo (ALLIX et al., 2016). AndroZoo é um repositório atualizado de apps para a comunidade de pesquisa e é frequentemente usado em diferentes pesquisas (HE et al., 2018; LI et al., 2018b). De forma aleatória, foram escolhidas 100 apps adotando alguns filtros. Foram selecionadas apenas as apps que estivessem presentes no Google Play, considerando assim somente apps que estivessem na loja disponível aos usuários. Logo, para garantir que seja uma app muito usada, foram selecionadas apenas as apps que tenham mais de 10.000 (dez mil) downloads. No final, restaram 30 apps que cumpriam os dois requisitos citados. As apps selecionadas e suas características são apresentadas na Tabela 17. As informações das apps apresentadas são: categoria, número de estrelas, número de *downloads*, versão mínima e compilada da API, suporte para telas de tamanho *small, normal, large, xlarge*; segundo a classificação da plataforma Android, sendo *"1"* possui suporte e *"0"* não possui suporte.

Além das apps, foi utilizado um conjunto de dados (*Dataset*) de dispositivos móveis. Este conjunto de dados foi coletado do site GSMArena[1]. A primeira versão do conjunto de dados é composta por 2559 dispositivos móveis coletada em julho/2019; uma nova versão atualizada até agosto/2020 deste conjunto de dados foi usada para QP2, com 5144 dispositivos; a atualização foi devido à importância na representatividade de um *dataset* atual.

[1] https://www.gsmarena.com

Tabela 17 – Androzoo - aplicações usadas.

id	Nome do pacote	Categoria	Estrelas	downloads	minsdk	targetsdk	small	normal	large	xlarge
app1	appenate.appenate	Productivity	4	10000	19	27	0	0	0	0
app2	com.skbose11.Train	travel	4	100000	21	28	0	0	0	0
app3	com.robot.car.golden.gangaster.crime.city.game.sam	sports	4	50000	16	29	1	1	1	1
app4	com.barclays.barclaysnow	business	3	10000	23	28	0	0	0	0
app5	com.pop.star.popstar.funny	games	4	500000	17	28	0	0	0	0
app6	kaydev.recordaudio.voiceapp	music	4	100000	16	28	1	1	1	1
app7	ragas.spb.melody.songs	music	4	10000	16	28	1	1	1	1
app8	air.com.sgn.juicejam.gp	games	4	10000000	16	28	0	0	0	0
app9	com.hawsoft.mobile.speechtrans	travel	4	5000000	16	29	0	0	0	0
app10	br.com.vitreo.app_vitreo	finance	5	10000	23	28	1	1	1	1
app11	com.mandg.funny.firescreen	entertainment	4	1000000	21	26	0	0	0	0
app12	com.casio.gshockconnected	lifestyle	5	500000	21	27	0	0	0	0
app13	com.orange.contultauorange	tools	4	1000000	16	28	1	1	1	1
app14	com.forqan.tech.OwisPuzzles.free	games	4	1000000	14	27	0	0	0	0
app15	pl.linksoft.camtronome	music	4	10000	16	26	0	0	0	0
app16	ch.transn.app.android	travel	4	10000	21	29	0	0	0	0
app17	app.bwy.v420	sports	5	10000	16	28	0	0	0	0
app18	com.dubberss.koo.dub2	education	5	50000	19	28	0	0	0	0
app19	de.WDR.DerElefant	entertainment	4	100000	19	28	1	1	1	1
app20	amanita_design.samorost3.GP.demo	games	4	1000000	21	28	1	1	1	1
app21	com.broadsoft.ucone.android	communication	4	10000	19	27	0	0	0	0
app22	com.luglobal.android	finance	4,5	10000	16	27	1	1	1	1
app23	kh.com.onetv	entertainment	4,5	500000	21	28	1	1	1	0
app24	com.buildtownn.skimlyblock	games	3	100000	16	29	0	0	0	0
app25	app.habitaclia2	house	4,5	1000000	19	28	0	0	0	0
app26	com.gib.meembah	finance	4,5	10000	21	28	1	1	1	1
app27	air.com.smileygamer.christmassweeper2	games	4,5	100000	19	28	1	1	1	1
app28	com.ironbear.everrich	games	4,5	100000	16	28	1	1	1	1
app29	com.tapNNFun.justiceWarriorStickDragonSuperFight	games	4,5	10000	19	28	0	0	0	0
app30	com.levelex.gastroex	medical	4	10000	19	26	1	1	1	1

Para a QP3, foi selecionada uma organização que realizaria o processo de seleção de novos dispositivos móveis para teste de sua app. Para este estudo, foi utilizada a entrevista como técnica de coleta de dados do processo de seleção de dispositivos móveis. Foi necessário duas entrevistas com a responsável por esse processo. A primeira entrevista antes do processo de seleção e a segunda após o processo. Neste estudo, foram consideradas as diretrizes de Runeson e Höst (2009). Assim, foi definido como um *estudo de caso embutido* onde várias unidades de análise são estudadas dentro de um contexto. O contexto definido para este estudo foi *o processo de seleção do dispositivo móveis para teste de uma app* e as duas unidades de análise foram o *Processo de seleção da organização* e o *Processo de seleção do* $DeSeCT$.

Foram apresentados para a responsável os objetivos da entrevista e do estudo de caso, além da explicação de como os dados da entrevista seriam usados. Para ambas as entrevistas buscou-se o consentimento da entrevistada para fazer a gravação em áudio da entrevista para análise posterior. As perguntas da primeira entrevista foram relacionadas a: tipos de testes realizados; porcentagem de testes manuais e automatizados; dispositivos por plataforma; e emuladores usados para teste. As perguntas da segunda entrevista foram relacionadas a: o processo seguido para selecionar uma lista final de dispositivos móveis; equipe que participa da tomada de decisão; base de dispositivo

móvel; os critérios usados para a seleção; e as possíveis restrições.

5.3 Execução do estudo

A execução do estudo foi dividida em três fases, cada fase relacionada a uma questão de pesquisa. Nas fases 1 e 2, relacionadas às QP1 e QP2, foi utilizada a seguinte configuração de DeSeCT: Tipo de algoritmo genético: MultiObjective NSGA-II; Codificação: Binária; Crossover: Dois pontos; Mutação: Flip Bit Based; Processo de seleção: Elitismo; a probabilidade usada foi de 80% de taxa de cruzamento e 1% de probabilidade de mutação; e o tamanho inicial da população foi 200.

QP1: Para identificar a importância e impacto do uso dos atributos da aplicação antes da seleção de dispositivos móveis para teste, a abordagem $DeSeCT$ foi avaliada em 30 aplicações Android e comparada com a abordagem $Baseline$, a mesma usada na prova de conceito (Seção 4.2). Como a abordagem usa algoritmo genético, definimos como critério de parada três números de gerações (1000, 5000 e 10000) para a execução de cada aplicação. Assim, cada aplicação foi executada dez vezes para cada geração. A média das dez execuções por cada app foi utilizada para a análise dos resultados.

No caso do $Baseline$, que não usa informações do app, a abordagem foi executada dez vezes para cada geração. Logo, para ser possível uma comparação entre $Baseline$ e $DeSeCT$, foram usadas as informações de cada app e identificou-se os "dispositivos compatíveis"(usáveis) e os "não compatíveis"(não-usáveis) para cada app. Um dispositivo usável foi definido como aquele que cumpre com as características da app e o não compatível como aquele que não cumpre com as características de uma app. Por exemplo, se uma app X possui como versão mínima a API 23 e o dispositivo Y tem a sua versão mínima de API 16, então o dispositivo Y é classificado como "não compatível"para a app X. No caso do $DeSeCT$, as informações do app foram usadas como um filtro para o Dataset, em seguida, executou-se dez vezes para cada app e para cada geração.

QP2: Os algoritmos genéticos permitem testar diferentes configurações para encontrar melhores resultados. Assim, o primeiro ajuste foi nos valores aleatórios para

a geração de cromossomos. Como visto nos resultados preliminares, para este estudo, definiu-se o número máximo de dispositivos selecionados no início do algoritmo em 250 e como critério de parada foi definido até 1000 gerações. Além disso, cada aplicação foi executada dez vezes e usou-se a média desses resultados para a análise.

QP3: A organização que fez parte deste estudo possui uma app com mais de 10.000.000 milhões de downloads e 4,7 estrelas de avaliação. A pessoa responsável pelo processo de seleção de dispositivos tem 17 anos de experiência na área de testes e é gerente da equipe de QA dentro da organização. Conforme o planejamento, foi definida a execução de duas entrevistas semiestruturadas. As questões das duas entrevistas foram baseadas na questão de pesquisa definida acima.

5.4 Análise dos Resultados

5.4.1 QP1: Qual é a importância do uso dos atributos da aplicação antes da seleção de dispositivos móveis para teste?

O principal objetivo da questão de pesquisa QP1 foi identificar se o uso ou não de informações de dispositivos móveis tem impacto na seleção de dispositivos para testes de compatibilidade. Como pode ser observado na Figura 16, os resultados do $DeSeCT$ mostram menos dispositivos selecionados em três gerações diferentes (1000, 5000 e 10000). Para as três gerações, o número médio mínimo de dispositivos selecionados para $DeSeCT$ foi de *216, 154 e 133* dispositivos para o app10; o número médio máximo de dispositivos selecionados foi *701, 578, e 528* dispositivos nas app15, app09 e app14, respectivamente. Por outro lado, para a $Baseline$, nas três gerações, o número mínimo de dispositivos selecionados foi *670, 547 e 500* dispositivos; o número médio máximo de dispositivos selecionados foi *725, 613 e 553* dispositivos. Os resultados são iguais para todas as apps devido o $Baseline$ não usar as informações da app. Assim, o resultado pode servir para todas as apps.

Foi realizada uma análise extra para identificar a importância do uso das informações da app. Com os resultados do $Baseline$, para cada app foram identificados todos os dispositivos que seriam descartados por não atender às características da app.

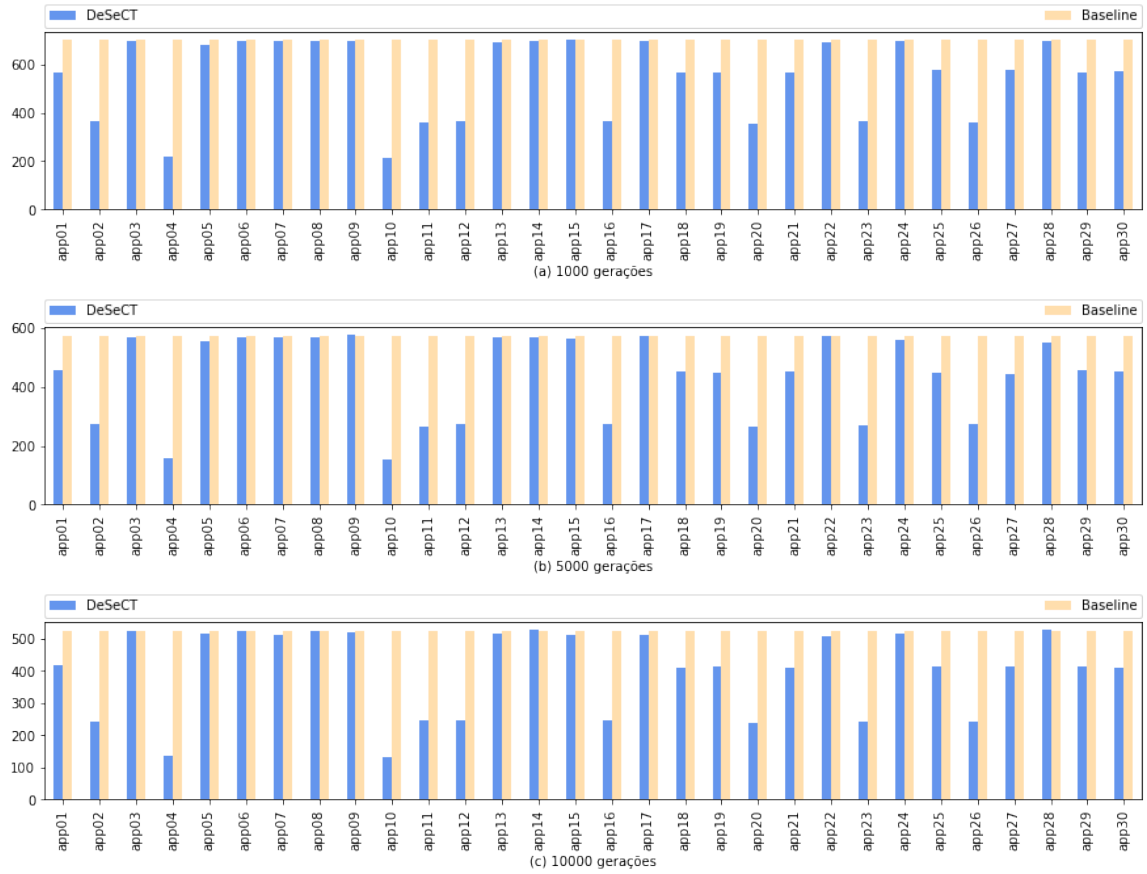


Figura 16 – Dispositivos selecionados pelas duas abordagens: *DeSeCT e Baseline*.

Assim, na Figura 17, identificou-se os "dispositivos não compatíveis"que não poderiam ser usados para teste e teriam que ser descartados. O maior número médio de dispositivos descartados foi de *404, 331, e 296* para 1000, 5000 e 10000 gerações respectivamente na app10. Por outro lado, 7 apps (app02, app11, app12, app16, app20, app23, app26) têm mais de 38% de dispositivos não compatíveis, 2 apps (app04, app10) têm mais de 57% dispositivos não compatíveis. Estes dispositivos descartados poderiam representar um desperdício de dinheiro e tempo dos testadores caso fossem selecionados de alguma forma dentro do processo de seleção de dispositivos móveis para teste. Uma análise adicional foi realizada, usando apenas os dispositivos compatíveis para $DeSeCT$ e $Baseline$, na Figura 18 percebe-se que $DeSeCT$ continua a ter melhor resultado em comparação ao $Baseline$.

A Tabela 18 mostra o total de dispositivos selecionados pelo $Baseline$ e pela abordagem $DeSeCT$ e a redução percentual de dispositivos descartados para cada app. A média de redução percentual de dispositivos selecionados para cada geração

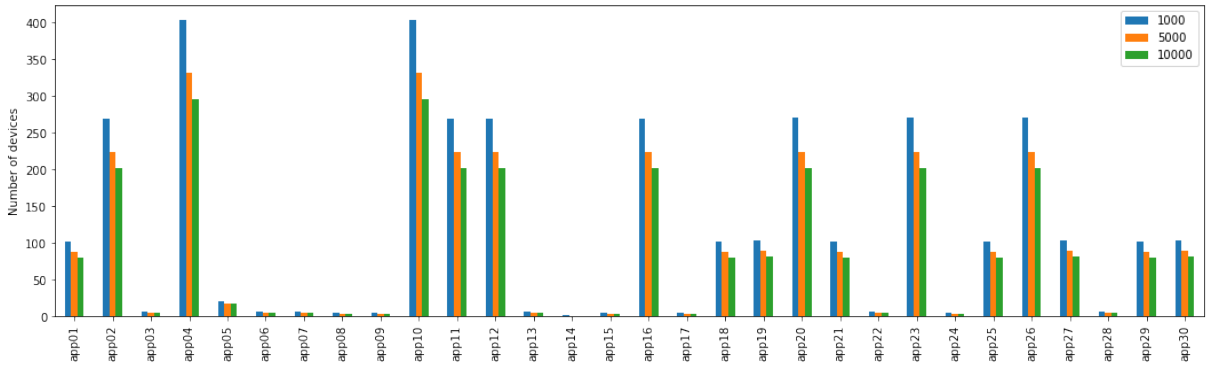


Figura 17 – *Baseline* - dispositivos não compatíveis por geração.

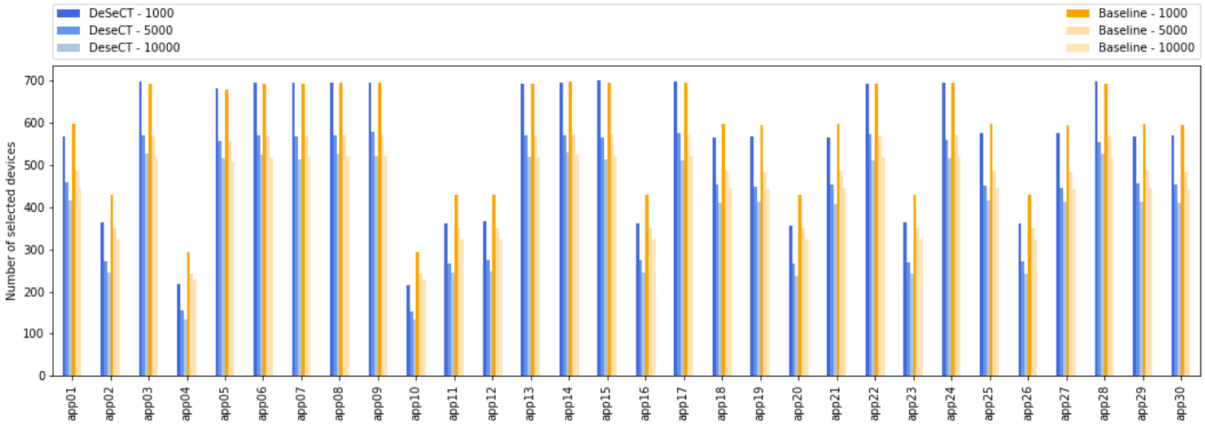


Figura 18 – *DeSeCT vs Baseline* - dispositivos compatíveis .

(1000, 5000, 10000) foi de -21,1%, -23,3% e -23,7% respectivamente. Para a geração 10000, das 30 aplicações, 17 deles conseguiram ter uma redução percentual entre -20,7% e -74,7% de dispositivos descartados em relação ao número total de dispositivos selecionados pela abordagem $Baseline$. Apenas 1 app (app15) de 30 na geração 1000 e 3 apps (app03, app15, app28) de 30 na geração 10000 na abordagem $DeSeCT$ selecionou mais dispositivos do que a abordagem $Baseline$. Pelo menos dez aplicações têm mais de 200 dispositivos inutilizáveis.

Por outro lado, a Figura 19 mostra a análise de cobertura; $Baseline$ apresenta menos cobertura em quatro atributos (resolução, tamanho de tela, API e participação de mercado); Para a análise de cobertura do $Baseline$, apenas os dispositivos compatíveis com a app foram considerados; os dispositivos inutilizáveis foram descartados. Na cobertura de rede, ambas as abordagens têm 100% de cobertura, devido a que temos apenas 4 tipos de rede (2G, 3G, 4G, e 5G) que é de fácil cobertura com uma quantidade mínima de dispositivos. Não usar as informações do app pode ter um grande impacto,

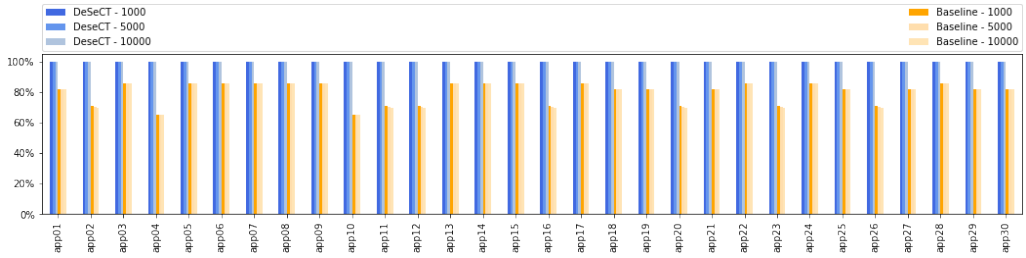
Tabela 18 – *Baseline e DeSeCT:* Dispositivos selecionados e porcentagem de descartados.

Gen	**1000**			**5000**			**10000**		
app	**Baseline**	**DeSeCT**	**Redução**	**Baseline**	**DeSeCT**	**Redução**	**Baseline**	**DeSeCT**	**Redução**
app01	698	566	**-18,9%**	573	458	**-20,2%**	525	416	**-20,7%**
app02	698	363	**-47,9%**	573	273	**-52,4%**	525	244	**-53,4%**
app03	698	697	**-0,2%**	573	571	**-0,5%**	525	525	0,1%
app04	698	219	**-68,7%**	573	157	**-72,7%**	525	135	**-74,3%**
app05	698	680	**-2,5%**	573	556	**-3,0%**	525	515	**-1,8%**
app06	698	693	**-0,7%**	573	571	**-0,5%**	525	524	**-0,1%**
app07	698	694	**-0,5%**	573	568	**-1,0%**	525	512	**-2,4%**
app08	698	693	**-0,7%**	573	570	**-0,7%**	525	526	0,2%
app09	698	693	**-0,7%**	573	578	0,7%	525	521	**-0,7%**
app10	698	216	**-69,1%**	573	154	**-73,2%**	525	133	**-74,7%**
app11	698	361	**-48,3%**	573	267	**-53,5%**	525	245	**-53,3%**
app12	698	367	**-47,4%**	573	275	**-52,1%**	525	246	**-53,1%**
app13	698	692	**-0,9%**	573	570	**-0,7%**	525	517	**-1,4%**
app14	698	693	**-0,6%**	573	570	**-0,6%**	525	528	0,6%
app15	698	701	0,4%	573	565	**-1,4%**	525	512	**-2,5%**
app16	698	362	**-48,1%**	573	274	**-52,1%**	525	245	**-53,3%**
app17	698	696	**-0,2%**	573	574	0,1%	525	510	**-2,7%**
app18	698	564	**-19,2%**	573	452	**-21,1%**	525	411	**-21,7%**
app19	698	567	**-18,7%**	573	446	**-22,1%**	525	413	**-21,2%**
app20	698	357	**-48,9%**	573	266	**-53,5%**	525	238	**-54,7%**
app21	698	564	**-19,2%**	573	453	**-20,9%**	525	408	**-22,2%**
app22	698	692	**-0,9%**	573	571	**-0,4%**	525	509	**-3,0%**
app23	698	365	**-47,7%**	573	268	**-53,3%**	525	243	**-53,7%**
app24	698	694	**-0,6%**	573	559	**-2,6%**	525	516	**-1,7%**
app25	698	574	**-17,7%**	573	450	**-21,6%**	525	415	**-20,9%**
app26	698	360	**-48,4%**	573	273	**-52,5%**	525	243	**-53,7%**
app27	698	574	**-17,8%**	573	444	**-22,6%**	525	414	**-21,2%**
app28	698	697	**-0,1%**	573	553	**-3,6%**	525	527	0,4%
app29	698	567	**-18,8%**	573	455	**-20,7%**	525	414	**-21,1%**
app30	698	570	**-18,3%**	573	453	**-21,1%**	525	411	**-21,7%**
Média	698	572	**-21,1%**	573	454	**-23,3%**	525	411	**-23,7%**
Des. Padrão	0	160	**0,23**	0	141	**0,25**	0	131	**0,25**
Min	698	216	**-69,1%**	573	154	**-73,2%**	525	133	**-74,7%**
Max	698	701	**0,4%**	573	578	**0,7%**	525	528	**0,6%**

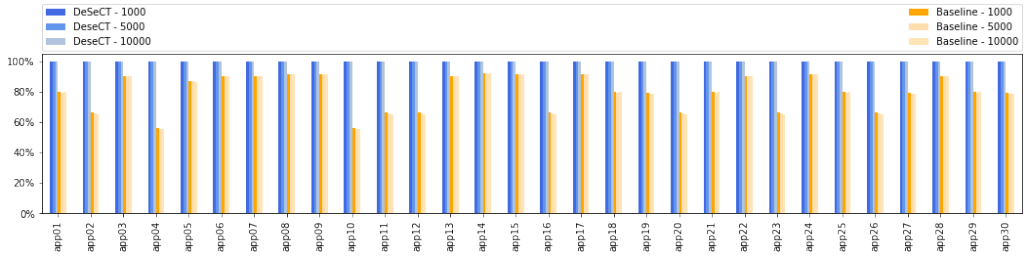
como nos apps app10 e app04 que possuem um número muito maior de dispositivos descartados em comparação com o número de dispositivos selecionados pela abordagem $DeSeCT$. Outras sete aplicações descartaram dispositivos próximos ao número de dispositivos que seriam selecionados pela abordagem $DeSeCT$.

5.4.2 QP2: Podem ser definidas configurações diferentes para ajustar o desempenho do DeSeCT?

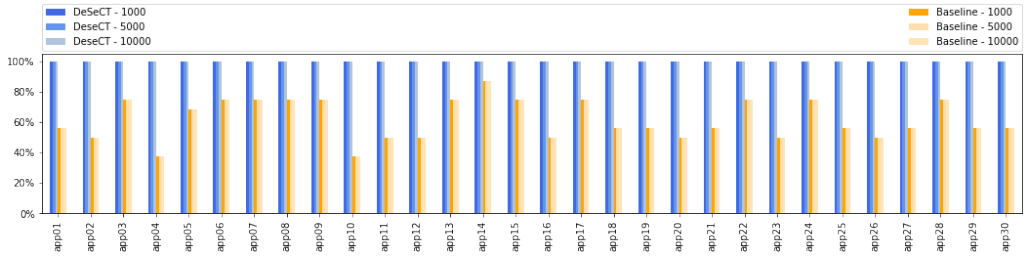
A primeira configuração testada foi a definição de valores máximos para o número de dispositivos selecionados no processo de geração de cromossomos, considerando os resultados da prova de conceito no capítulo anterior. Foi definido como 250 o número inicial de dispositivos selecionados.



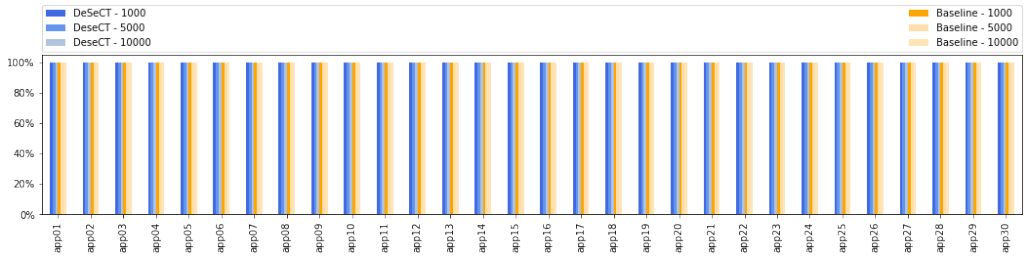
(a) Cobertura de resolução.



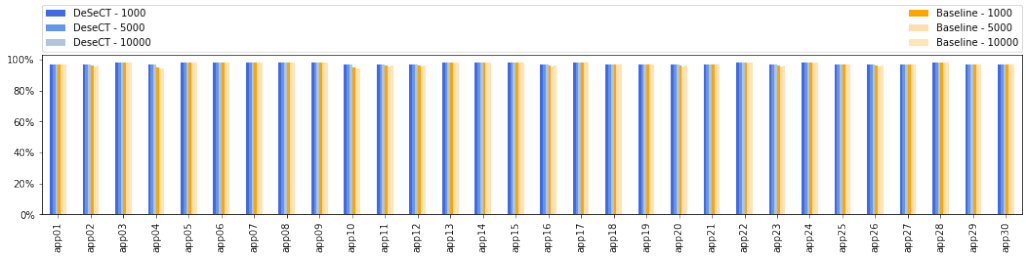
(b) Cobertura de tamanho de tela.



(c) Cobertura de API.



(d) Cobertura de rede.



(e) Cobertura do *Android Marketshare*.

Figura 19 – Cobertura de características - *DeSeCT e Baseline*.

A Figura 20, mostra os resultados com a nova configuração. As apps, pela sua diversidade, selecionaram diversas quantidades de dispositivos selecionados. O número mínimo de dispositivos selecionados foi 59 para a app07, enquanto o número máximo de

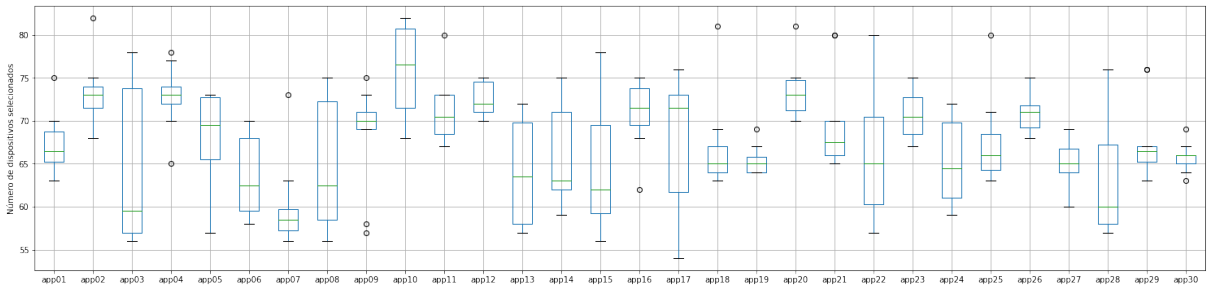


Figura 20 – Dispositivos selecionados - limite inicial de 250 dispositivos.

dispositivos selecionados foi de 76 dispositivos para a app10. Pode-se observar que esta nova configuração melhora o objetivo de minimização dos dispositivos selecionados. O número de dispositivos selecionados é quase três vezes menor do que os obtidos na QP1; tendo uma redução entre 74% e 91% a menos de dispositivos selecionados.

Em relação ao percentual de cobertura dos recursos, a Figura 21, mostra o percentual de cobertura que foi alcançado por cada recurso e app. A cobertura média de todas as apps para as características foram as seguintes: resolução 59,7%, tamanho de tela 51,2%, API 100%, rede 100%, e participação de mercado 97,5%. Os resultados são inferiores a QP1, porém, os resultados da QP2 são mais viáveis do que QP1; isso porque é muito mais rentável financeiramente 59 dispositivos (app07-QP2) do que 691 dispositivos (app07-QP1).

Assim, a escolha de um número inicial de dispositivos permitiu evoluir a proposta, porém a quantidade de dispositivos selecionados entre 59 e 76 dispositivos ainda é uma quantidade muito alta para se pensar na aquisição deles e foge da realidade dos desenvolvedores e testadores da indústria. Ainda são necessárias outras mudanças que serão analisadas nos próximos capítulos.

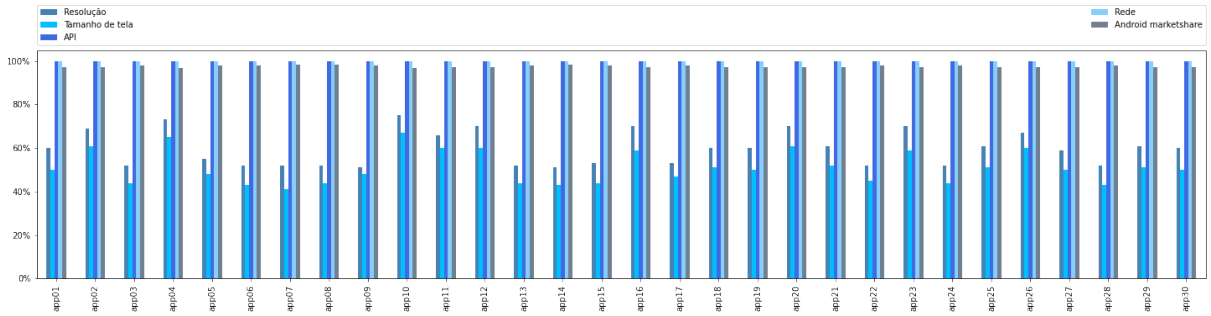


Figura 21 – *DeSeCT*: cobertura de características - limite inicial de 250 dispositivos.

5.4.3 QP3: Como o DeSeCT se compara ao estado da prática?

O processo de seleção de dispositivos na indústria é uma atividade que é executada a cada certo tempo que podem ser meses ou anos. Na organização que aceitou participar deste estudo, a equipe de QA estava prestes a executar a atualização dos dispositivos de testes. Esse processo foi acompanhado e depois os resultados do $DeSeCT$ foi apresentado e com isso foram obtidos *feedbacks* por parte deles.

Equipe de testes e dispositivos: os testes realizados pela equipe de QA são funcionais e de desempenho; a porcentagem de teste manual é de 90% e 10% é automatizado. A equipe de teste possui em média 10 dispositivos de teste entre as marcas Samsung, Motorola, Xiaomi e Apple. Esses dispositivos foram selecionados a partir de uma lista de modelos de dispositivos mais usados por usuários de apps (*marketshare*).

A atualização desses dispositivos surge devido à necessidade ou desgaste físico do próprio dispositivo, ou ao identificar que um dispositivo quebrou e não está funcionando normalmente. Por outro lado, a atualização do sistema operacional ocorre conforme a necessidade do teste realizado. Dessa forma, os dispositivos são atualizados para as versões mais recentes do sistema operacional. O uso de emuladores é voltado para a plataforma iOS.

Processo de seleção da organização: O novo processo de seleção do dispositivo se deve a dois fatores; (1) os dispositivos utilizados atualmente já estão desatualizados em relação ao mercado e (2) o número de dispositivos está abaixo da necessidade de teste. Com relação ao teste de compatibilidade, a cobertura é baseada principalmente nos dispositivos usados pelos usuários da aplicação.

A organização utiliza uma ferramenta para a análise estatística da app. A ferramenta mostra quais modelos de dispositivos são mais usados pelos usuários da aplicação. Esses dados foram usados como conjunto de dados de dispositivos. Com base nessas informações, foram selecionadas as seguintes marcas: *IPhone, Samsung, Motorola, Xiaomi, LG*. De cada marca, foi selecionado o modelo mais utilizado pelos usuários. Em seguida, foi feita uma comparação com os dispositivos atuais para verificar se havia algum dispositivo com resolução e tamanho de tela semelhantes. Dois dispositivos foram trocados porque o mesmo modelo foi usado por uma pessoa da

equipe de QA e poderia ser usado para testes. Após a primeira fase de seleção, a lista foi compartilhada com mais duas pessoas da equipe de QA para dar sua opinião. Um critério que é tomado em consideração é considerar os modelos de dispositivos pessoais da equipe de controle de qualidade, pois eles também são usados para teste e esses modelos não são mais considerados para compra. Por exemplo, a equipe possuía um aparelho Iphone X que foi substituído pelo Iphone 11.

Os critérios de seleção utilizados foram: Os aparelhos mais utilizados, as versões mais atualizadas dos aparelhos e diversidade de tamanho e resolução de tela. Esses critérios têm a mesma importância. Segundo a responsável pelo processo de seleção, "*o mais trabalhoso é chegar a uma lista final que tenha diversidade e maior abrangência entre todas as opções que temos*". Ante a extensa diversidade de dispositivos, essa tarefa pode se tornar complicada.

Processo de seleção DeSeCT: Para o processo $DeSeCT$, foi usado o *dataset* atualizado com 5144 dispositivos. Foi feita uma análise do conjunto de dados e identificou-se que alguns atributos existiam em apenas um dispositivo. Essa característica não seria adequada para a organização, pois não é objetivo testar um dispositivo único e sim um conjunto que seja representativo. Assim, foi definido considerar apenas os atributos que estejam presentes em mais de 20 dispositivos. Os atributos aos quais esse critério foi aplicado foram "tamanho da tela" e "resolução". Além disso, a abordagem $DeSeCT$ usa os atributos da aplicação para realizar um filtro. Assim, o conjunto de dados final tinha 1855 dispositivos. Os critérios utilizados para este estudo foram: tamanho de tela (polegadas), resolução, API, rede e *marketshare*. Os atributos da aplicação foram extraídos para serem usados com a abordagem $DeSeCT$.

Após a execução da abordagem $DeSeCT$, foi obtido como resultado um conjunto de 25 dispositivos cujas características principais podem ser observadas na Tabela 19. Esses dispositivos obtiveram 100% de cobertura nos critérios de resolução, tamanho, API e rede, apenas para os critérios de *marketshare* a cobertura foi de 94,7%.

Apresentação de resultados: A organização apresentou a lista de dispositivos móveis que tinha inicialmente antes de realizar o processo de seleção e uma lista final com os dispositivos móveis após o seu processo de seleção. Todas as características

Tabela 19 – DeSeCT - Dispositivos selecionados para a organização

Primeira execução				Segunda execução			
Marca e Modelo	**API**	**Tamanho**	**Resolução**	**Marca e Modelo**	**API**	**Tamanho**	**Resolução**
Alcatel Idol 3 (4.7)	21	4.7	720 x 1280	LG G Pad 10.1 LTE	21	10.1	1280 x 800
Asus Zenpad 10 Z300C	21	10.1	800 x 1280	LG G Pad II 10.1	22	10.1	1920 x 1200
Celkon Millennia Hero	22	4	480 x 800	LG G Pad III 8.0 FHD	23	8	1200 x 1920
Coolpad Conjr	23	5	720 x 1280	LG K10 (2017)	24	5.3	720 x 1280
Coolpad Note 6	25	5.5	1080 x 1920	LG K20 (2019)	28	5.45	480 x 960
Honor Note 8	23	6.6	1440 x 2560	LG K50	28	6.26	720 x 1520
Honor Play 4T	29	6.39	720 x 1560	LG K7	22	5	480 x 854
HTC Wildfire R70	28	6.53	720 x 1560	LG Wine Smart	22	3.2	320 x 480
Huawei Enjoy Z 5G	29	6.5	1080 x 2400	Motorola Moto E (2nd gen)	21	4.5	540 x 960
Infinix Hot 7	26	6.2	720 x 1440	Motorola Moto E5 Plus	26	6	720 x 1440
Infinix Hot 8 Lite	27	6.6	720 x 1600	Motorola Moto G8	29	6.4	720 x 1560
Lenovo Tab3 7	23	7	600 x 1024	Motorola Moto X Style	22	5.7	1440 x 2560
Lenovo Yoga Smart Tab	28	10.1	1200 x 1920	Motorola Moto Z2 Play	25	5.5	1080 x 1920
LG G Pad III 8.0 FHD	23	8	1200 x 1920	Motorola One Macro	28	6.2	720 x 1520
Micromax Bharat 5 Infinity	27	5.45	720 x 1440	Samsung Galaxy A51 5G	29	6.5	1080 x 2400
Motorola Moto E Dual SIM	21	4.5	540 x 960	Samsung Galaxy Folder	22	3.8	480 x 800
Motorola Moto G Fast	29	6.4	720 x 1560	Samsung Galaxy J1 mini prime	22	4	480 x 800
Realme U1	27	6.3	1080 x 2340	Samsung Galaxy J5 (2017)	24	5.2	720 x 1280
Samsung Galaxy J Max	22	7	800 x 1280	Samsung Galaxy Note10 5G	28	6.3	1080 x 2280
Samsung Galaxy J7 Max	24	5.7	1080 x 1920	Samsung Galaxy S7 active	23	5.1	1440 x 2560
TECNO Pop 2 Plus	27	6	480 x 960	Samsung Galaxy Tab A 8.0 (2015)	21	8	768 x 1024
Verykool s5200 Orion	24	5.2	480 x 854	Samsung Galaxy Tab Advanced2	26	10.1	1200 x 1920
Vivo Z1	27	6.26	1080 x 2280	Samsung Galaxy Tab J	22	7	800 x 1280
Xiaomi Redmi 8A	28	6.22	720 x 1520	Xiaomi Mi CC9	28	6.39	1080 x 2340
Xiaomi Redmi Note 5 Pro	25	5.99	1080 x 2160	Xiaomi Redmi 2A	21	4.7	720 x 1280
				Xiaomi Redmi 6A	27	5.45	720 x 1440
				Xiaomi Redmi 8A	28	6.22	720 x 1520
				Xiaomi Redmi 9C	29	6.53	720 x 1600
				Xiaomi Redmi Note 5 Pro	25	5.99	1080 x 2160

dos dispositivos das duas listas foram coletadas para depois calcular a cobertura. O resultado do $DeSeCT$, considerando as informações do app da organização, foi apresentado à responsável. Um *feedback* foi fornecido, de que devido à organização participante deste estudo estar no Brasil, existem algumas marcas na Tabela 19-(primeira execução) que não são comercializadas no Brasil. Portanto, não seria de interesse deles testar nesses dispositivos.

Assim, foi realizada uma segunda execução considerando apenas as marcas mais utilizadas pelos usuários da organização. A Tabela 19 apresenta os resultados (segunda execução) com o novo conjunto de 29 dispositivos selecionados. Esses dispositivos são mais representativos para a organização e poderiam ser considerados para teste.

Por fim, foi comparada a cobertura percentual entre os dispositivos selecionados na etapa inicial e final da organização com a abordagem $DeSeCT$. Para efeitos de análise, foi considerado o resultado da segunda execução do $DeSeCT$ e o uso do mesmo *dataset*. Os resultados são mostrados na Tabela 20; $DeSeCT$ tem mais cobertura porque possui mais dispositivos. Porém, se a organização considerar o uso ou inclusão de alguns

Tabela 20 – Dispositivos selecionados: comparação de cobertura

Tamaho	**Dispositivos**	**Resolução**	**Tamanho**	**API**	**Rede**	**Android markeshare**
Org. Lista Inicial	6	29,4%	19%	33,3%	75%	83,1 %
Org. Lista final	10	41,2%	33,3%	44,4%	75%	83,1 %
DeSeCT	29	100%	100%	100%	100%	95%

Tabela 21 – Dispositivos selecionados - *Android marketshare*

Tamaho	**mdpi**	**hdpi**	**xhdpi**	**xxhdpi**	**Cobertura**
small		LG Wine Smart	Samsung Galaxy Folder		**0,1 %**
normal		LG K20 (2019)	Motorola Moto G8	Samsung Galaxy A51 5G	**83,1 %**
large	LG G Pad 10.1 LTE	Samsung Galaxy Tab J	LG G Pad III 8.0 FHD		**5,7 %**
xlarge	Samsung Galaxy Tab A8.0(2015)	Samsung Galaxy Tab Advanced2			**6,1 %**
Total					**95 %**

dos dispositivos que o $DeSeCT$ sugere, poderia aumentar a porcentagem no *Android marketshare* e as outras características.

Ao analisar o *Android marketshare* na Tabela 21, foi identificado que considerando apenas um dispositivo, do resultado do $DeSeCT$, por cada tamanho de tela, é possível alcançar o 95% de cobertura de *marketshare* com 10 dispositivos.

As duas implicações mais importantes que foram observadas neste estudo é: (1) Além das informações da app, é necessário considerar os objetivos ou informações da organização de forma que os resultados sejam próximos da realidade deles e os resultados do $DeSeCT$ possam ser considerados como suporte para a seleção de dispositivos. (2) O uso do $DeSeCT$ pode diminuir o tempo gasto no processo de seleção de dispositivos. Isso é um problema que a organização tem atualmente. Com isso, a abordagem auxiliaria a dar maior visibilidade sobre a representatividade do conjunto de dispositivos selecionados.

5.5 Ameaças à Validade

Os resultados do estudo não podem ser generalizados para todas as aplicações móveis. O foco é a seleção de dispositivos móveis para testes de compatibilidade. Assim, algumas ameaças à validade estão relacionadas a:

Validade de constructo: As duas partes do estudo foram realizadas seguindo as diretrizes do GQM e diretrizes para a execução de uma entrevista semi-estruturada.

Foram realizados os planejamentos para cada seção.

Validade interna: o conjunto de apps selecionados foi obtido do repositório *Androzoo*, que é muito usado para pesquisas. Além disso, para evitar viés, foram escolhidos aleatoriamente e depois foram aplicados filtros para incluir as apps que estão na loja de apps.

Validade externa: Uma seção deste estudo foi feita com a participação de uma organização da indústria. Devido ao fato de ter sido feito estudo com apenas uma organização, o resultado pode não ser representativo, porém a organização é uma organização nacional que está presente em todo o país e possui mais de 25 milhões de contas ativas. Os materiais usados no estudo podem ser considerados representativos e atuais. No caso dos dados coletados para o *dataset* foram coletados com informações atuais.

5.6 Evolução da abordagem

Os resultados deste estudo permitiram identificar novas melhorias que poderiam ser implantadas no $DeSeCT$. Nesta seção detalharemos as mudanças que serão implementadas na abordagem e sua evolução.

Detalhes do $DeSeCTv2$*:* A primeira parte do estudo, em maior escala, permitiu confirmar que o filtro do *dataset* usando as informações da app permitiram um melhor desempenho em relação a não ter dispositivos não-usáveis. Assim, o filtro do *dataset* passou a ter mais destaque pela sua importância. A segunda parte do estudo, com os resultados da organização da indústria, mostrou-se a importância de considerar os dados ou critérios relacionados ao objetivo deles, com o objetivo de apresentar soluções que reflitam a realidade da organização.

Portanto, a Figura 22 apresenta a evolução da proposta. Nessa nova versão, a partir dos resultados dos estudos executados, foram implementados as seguintes mudanças:

- Nova etapa separada "Filtro do Dataset - *Dataset filter*"que recebe informações da app e do usuário que pode ser uma organização para executar os filtros necessá-

rios.

- Nova etapa adicionada "Informações adicionais - *Additional information*"que informa critérios a serem aplicados no filtro do *dataset* e também informa os objetivos em relação a cobertura de características ao processo principal de seleção de dispositivos.

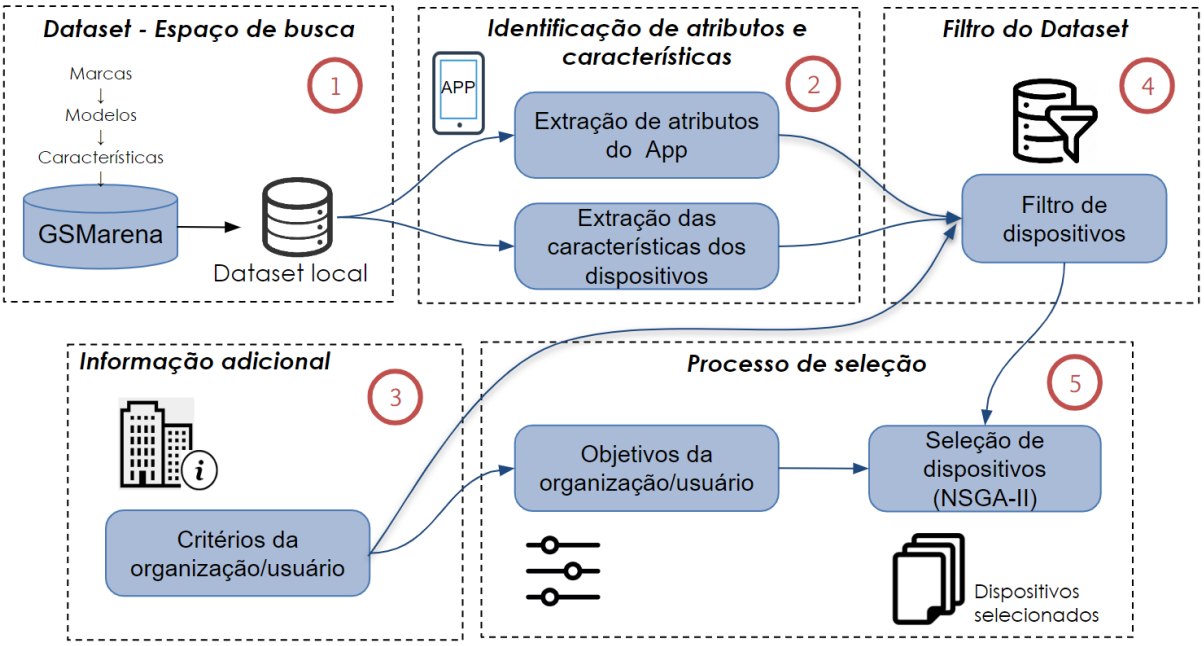


Figura 22 – *DeSeCTv2*: Filtro no dataset e informações de uma organização externa.

5.7 Considerações do Capítulo

Foi possível avançar no amadurecimento da proposta e confirmar a viabilidade. Com o estudo executado em maior escala foi possível identificar a importância do filtro do *dataset* e das informações provenientes da app. O estudo de caso na indústria permitiu identificar a necessidade de incluir informações referentes ao usuário ou organização que pretenda usar o $DeSeCT$ e obter resultados mais próximos da realidade.

Os resultados mostram que $DeSeCT$ pode efetivamente identificar dispositivos com maior cobertura de recursos em menor número de dispositivos e superar a abordagem $Baseline$ usada para comparação. Além disso, os resultados mostraram uma melhor eficiência ao usar as informações da aplicação móvel; evitando assim a

seleção de aproximadamente 15% a 18% de *dispositivos inutilizáveis*, pois não atendem às características da app.

O estudo de caso na indústria, neste capítulo, foi apenas com uma organização, devido ser muito difícil encontrar outras organizações estejam prestes a executar esse processo, um novo estudo será feito com mais profissionais da indústria. Assim, no próximo capítulo será apresentado um estudo com profissionais da indústria e novamente a otimização de parâmetros do Algoritmo Genético.

6

SURVEY E EXPERIMENTO COM PROFISSIONAIS NA INDÚSTRIA

Neste capítulo apresenta-se o planejamento, execução e análise dos resultados do *survey* e experimento com profissionais da indústria.

6.1 Objetivo do estudo

No capítulo anterior foi apresentado o caso de uma organização de *software* que realiza o processo de seleção de dispositivos. No entanto, se faz necessário estudar outros casos na indústria e ter uma maior representatividade. Assim, foi seguida a metodologia de pesquisa adotada para o desenvolvimento da proposta $DeSeCT$, descrita na Seção 1.5. O seguinte passo é a execução de um *survey* e experimento na indústria. O objetivo principal deste estudo foi estruturado usando a abordagem GQM (BASILI; CALDIERA; ROMBACH, 1994), apresentada na Tabela 22.

Tabela 22 – Objetivo do estudo - GQM.

Analisar	A seleção de dispositivos móveis para teste de uma app
Com o propósito de	Caracterizar e compreender
Em respeito a	Critérios usados na tomada de decisão para o processo de seleção de dispositivos
Do ponto de vista do	Profissional da indústria
No contexto de	Teste de compatibilidade de apps móveis.

6.2 Planejamento

O estudo teve duas etapas: (1) um *survey* e (2) uma atividade de seleção de dispositivos para teste de uma aplicação móvel real publicada na *Google play store*. Os participantes tiveram que selecionar um conjunto de dispositivos nos quais seriam executados os testes da app apresentada. Por razões de tempo e disponibilidade de dispositivos móveis para a execução de testes, foi decidido que a atividade a ser realizada seria restrita à seleção de dispositivos móveis que seriam utilizados na execução dos testes. O propósito deste estudo foi responder às seguintes questões:

- QP1: Quais são os critérios usados pelos profissionais da indústria na seleção de dispositivos móveis para teste?
- QP2: Quais são as características dos dispositivos móveis selecionados pelos profissionais da indústria e qual é a sua cobertura em relação às características da app a ser testada?

6.2.1 Instrumentação

Foi apresentado aos participantes um questionário com o objetivo de caracterizar a sua experiência em relação à atividade de teste de software (web e app), as plataformas e ambientes usados nos testes, além do Termo de Consentimento Livre e Esclarecido – TCLE.

6.2.1.1 Dataset e escolha da app

Dataset: O *dataset* inicial de dispositivos foi coletado do site GSMArena, que possui 5144 dispositivos. Para este estudo, foram selecionados de forma aleatória 50 dispositivos móveis com marcas conhecidas a nível mundial.

Escolha da app: Na *Google Play Store*, as duas categorias mais populares em 2021 (Statista, 2021b) foram "jogos" e "educação", a categoria educação foi selecionada para este estudo. De forma aleatória, foram escolhidas 100 apps, e para garantir que

Figura 23 – Características da app 2DUB

seja uma app muito usada, foram selecionadas apenas as apps que tinham mais de 10000 *downloads*. Dentre as 30 aplicações, havia 13 categorias de apps; a categoria jogos representa o 27% das apps, as categorias negócios, comunicação, educação, saúde, produtividade e ferramentas representam 1% cada uma. Na categoria educação, tinha apenas uma app $2DUB$, a qual foi selecionada.

$2DUB$ é uma aplicação móvel na categoria de Educação da plataforma *Android*, indicada para quem pretende melhorar a pronúncia e dublagem no idioma inglês e coreano. Ela possui mais de 1500 videoclipes que abrangem diversos gêneros. A app possui mais de 10000 *downloads* e possui uma avaliação de 4 estrelas.

As informações da app $2DUB$ e um exemplo do catálogo de dispositivos são apresentados na Figura 23 e Figura 24, respectivamente. Os artefatos podem ser encontrados no link bit.ly/selecaodispositivos.

6.2.1.2 Seleção de Participantes

Os participantes para este estudo são profissionais da indústria. Eles foram convidados para responder o *survey* e realizar uma atividade de seleção de dispositivos móveis para realizar testes na app em questão. Foi feita uma pesquisa no *Linkedin* por pessoas que estivessem trabalhando na área de teste de software. Algumas delas eram líderes de

Figura 24 – Exemplo de catálogo de dispositivos móveis

times, assim eles também estenderam o convite de participação para outras pessoas. Foram enviados 40 e-mails, dos quais 28 profissionais responderam o *survey* e 9 deles executaram a atividade. Para a execução da atividade, foram disponibilizados os seguintes artefatos:

- Um arquivo em formato PDF contendo as informações e características da app $2DUB$, que seria testada, e um catálogo dos 50 dispositivos móveis disponíveis para seleção.
- Um arquivo em formato XLS contendo uma lista com os 50 dispositivos móveis e suas características, a ser preenchido pelo participante indicando quais dispositivos selecionaria. Este arquivo deveria ser devolvido via e-mail.

6.2.1.3 Metodologia do estudo

Para responder às questões de pesquisa QP1 e QP2, foi seguida a seguinte metodologia: para QP1 foram analisados os resultados do *survey* e da atividade de seleção de dispositivos móveis em conjunto; enquanto que para a QP2 foram analisados apenas os resultados da atividade de seleção de dispositivos móveis.

QP1: para responder a essa QP foram consideradas as duas etapas: O *survey* e a *atividade de seleção*. No ***survey***, uma das questões principais do *survey* estava relacio-

Tabela 23 – Categorias e critérios de seleção de dispositivos de teste

Categoria	**Critérios**
Hardware	Tamanho de Tela Resolução Modelo Memória Processador Rede
Software	SO API Recursos e permissões
Externos	Marketshare Disponibilidade

nada a identificar se os participantes usam algum critério de seleção de dispositivos móveis para a execução dos testes. No formulário, apresentou-se duas opções: "Sim"e "Não". Para aqueles que responderam "Sim", tinha um campo livre no formulário, onde descreveram os critérios que eles consideravam na hora de selecionar dispositivos móveis. Na ***atividade de seleção***, os participantes preencheram na planilha XLS que foi disponibilizada os critérios de seleção que eles usaram na atividade. Assim, uma lista de critérios de seleção de dispositivos móveis foi obtida das duas etapas, que logo foram agrupadas pela similaridade de termos e por ter relação a uma mesma característica do dispositivo móvel.

Foram identificados 33 critérios das respostas dos participantes. No entanto, foi necessário realizar um agrupamento, resultando em 11 critérios para seleção de dispositivos. O agrupamento por similaridade foi revisado e validado duas vezes por outros 2 pesquisadores. Os 11 critérios também foram agrupados em 3 categorias maiores: critérios relacionados ao hardware, software e critérios externos, apresentadas na Tabela 23. A categoria *hardware* e *software* contempla critérios tangíveis e intangíveis, respectivamente; e a *categoria externo*, são critérios que não estão relacionados ao dispositivo e sim a aspectos externos que podem impactar na decisão de selecionar um dispositivo, como o *marketshare* e a disponibilidade de dispositivos. Esses 11 critérios serão usados para a análise dos resultados.

QP2: Para responder essa QP, foi realizada a atividade de selecionar dispositivos

de teste para a app $2DUB$. Os participantes enviaram suas respostas numa planilha com a lista dos 50 dispositivos móveis. Nesse arquivo, os participantes selecionaram quais dispositivos escolheram e qual foi o motivo da escolha.

As configurações consideradas para a análise desta QP foram as seguintes:

- Para a análise de cobertura de características foi necessário identificar o número de dispositivos móveis compatíveis com a app $2DUB$. Dos 50 dispositivos apresentados, 11 deles eram compatíveis com as informações da app $2DUB$; ou seja, a app $2DUB$ poderia ser instalada e testada nesses 11 dispositivos. Esses 11 dispositivos móveis foram considerados para fazer o cálculo de cobertura de características.
- As características consideradas para análise de cobertura foram: tamanho de tela, resolução, versão de *API*, *networking*, e *Android marketshare*.

Baseados nas métricas existentes para o teste de software, foi calculada a eficiência da seleção baseado na equação 6.1, onde DS_p representa os dispositivos móveis selecionados pelo participante e o DNU_p representa o número de dispositivos móveis que não serão usados (não-usáveis); assim, a E_p representa a eficiência da seleção do participante.

$$Eficiencia_p() = \frac{DispositivosSelecionados(DS_p) - DispositivosNoUsaveis(DNU_p)}{DispositivosSelecionados(DS_p)} \tag{6.1}$$

6.3 Execução do estudo

Para participar deste estudo, foram convidados 40 profissionais da indústria via e-mail; foi disponibilizado um formulário contendo questões relacionadas a sua experiência em teste de software e no final um convite para executar uma atividade de seleção de dispositivos. A execução do estudo esteve dividido em duas etapas:

Na primeira etapa, relacionada ao *survey*, dos 40 profissionais convidados, 28 deles responderam às questões no formulário (*survey*), obtendo uma taxa de resposta de 70%. Este tamanho de amostra leva a 90% de nível de confiança conforme a fórmula de

Hamburg (1980). Para os 28 profissionais, foi questionado se os participantes consideram algum método ou critério para a seleção de dispositivos móveis para teste. As opções de resposta eram "Sim" ou "Não". Caso a resposta fosse "Sim", eles teriam que descrever de forma aberta quais critérios eles usam.

Na segunda etapa, dos 28 participantes que responderam o *survey*, 9 deles realizaram a atividade de seleção de dispositivos. Para esta parte da atividade, o tamanho da amostra leva a 73% de nível de confiança conforme a fórmula de Hamburg (1980). Todos os participantes receberam dois artefatos (arquivos em formato pdf e xls), mencionados na seção 6.2.1.2. Os passos seguidos por cada participante foram os seguintes:

1. *Download* dos artefatos (arquivos nos formatos pdf e xls).
2. Ler as informações da app $2DUB$ a ser testada, descritas no arquivo pdf.
3. Identificar os dispositivos a serem selecionados para o teste desta app.
4. Editar o arquivo xls e marcar os dispositivos selecionados.
5. Descrever no arquivo xls quais critérios foram usados para a seleção.
6. Enviar via e-mail o arquivo xls.

Todos os arquivos recebidos foram incluídos para a análise dos resultados da segunda etapa.

A caracterização dos participantes não será analisada individualmente e estes não terão seus dados apresentados por questões de privacidade. Uma análise geral será conduzida para a análise dos resultados. Todos os participantes, caracterizados na Figura 25, trabalharam ou trabalham nos seguintes perfis: Engenheiro de Qualidade, Testador, Desenvolvedor, Gerente, Líder Técnico, Pesquisador e Gerente de produto (PM - *Product Manager*). Do total de participantes, 67,9% pertencem à área de desenvolvimento e qualidade de software e o 32,1% pertence à área de gerência e pesquisa.

Em relação à experiência, dos 28 profissionais, 12 deles possuem mais de 5 anos de atuação em testes de software no geral. Enquanto que 10 profissionais possuem experiência de 3 a 5 anos em testes de aplicações móveis, como mostrados na Figura 26.

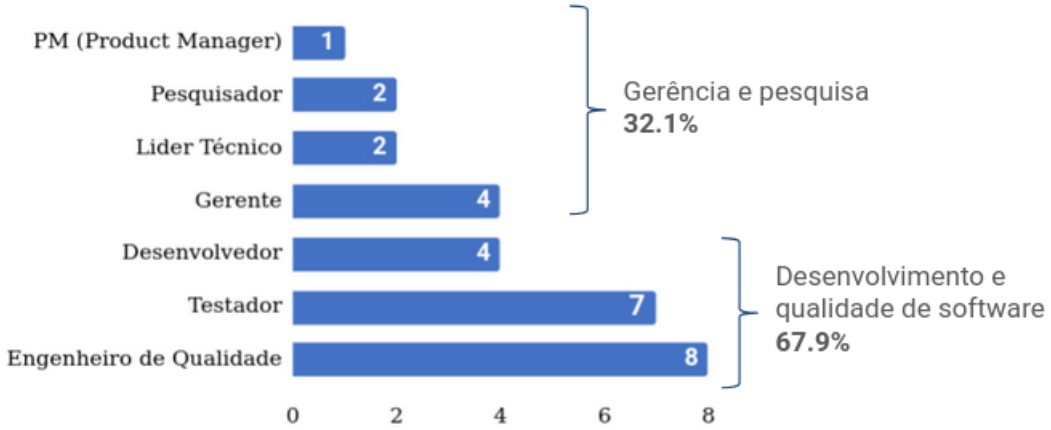


Figura 25 – Perfis dos participantes

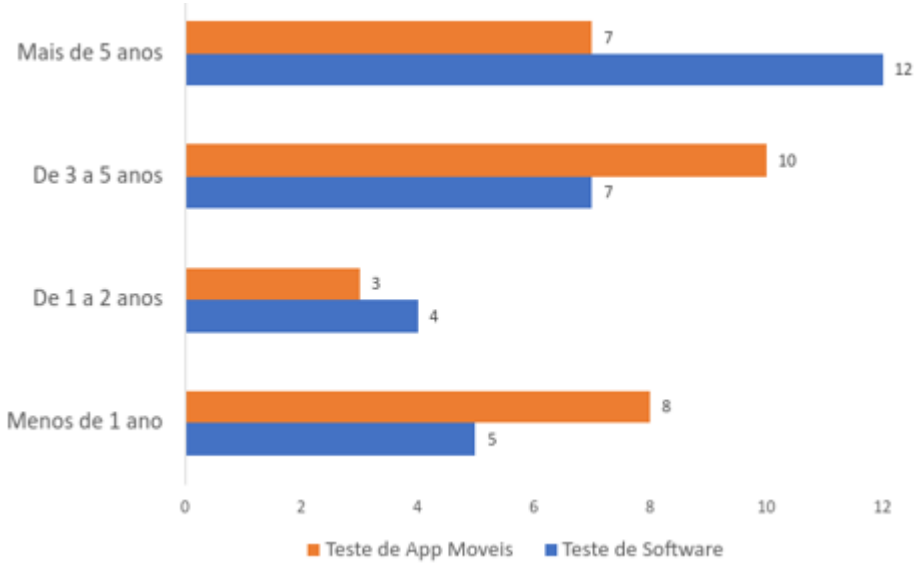


Figura 26 – Experiência dos participantes

Todos os participantes relataram ter experiência na plataforma *Android*, sendo que 13 deles também possuem experiência na plataforma *iOS*.

6.4 Análise dos resultados

6.4.1 QP1: Quais são os critérios usados pelos profissionais na seleção de dispositivos para teste?

Para responder essa questão, foi considerado o resultado das duas etapas, conforme descrito na Seção 6.2.1.3. A primeira provém dos resultados do *survey*, e a segunda, da atividade de seleção de dispositivos móveis para a app $2DUB$. Os dois resultados estão

relacionados aos critérios para seleção de dispositivos móveis.

A Tabela 24 apresenta os resultados para as duas etapas e de forma geral, com os 11 critérios identificados (tela, resolução, modelo, memória, processador, versão de SO, versão de API, recursos e permissões, *marketshare*, e disponibilidade) e as 3 categorias: hardware, software e externo.

Survey: Dos 28 participantes, 15 (56%) deles responderam que "Sim", usam algum critério na seleção de dispositivos móveis, cinco dos que responderam "Sim"usaram até quatro critérios para a seleção de dispositivos. Doze participantes (44%) responderam que "Não" usam nenhum critério de seleção de dispositivos.

Para os participantes do *Survey*, dos onze critérios, às mais citadas por cada categoria foram "Tela (15%)" em Hardware, "versão do SO (13%)" em Software e "*Marketshare* (24%)" em Externos. A categoria "Versão da API" obteve pouca percentagem no uso. Esse resultado é interessante, pois a versão da API do dispositivo móvel é uma característica que define se o mesmo é compatível ou não com uma app. Essa informação está disponível nas informações de uma app na *Google play store*, porém não foi muito considerada pelos participantes.

Atividade: Nove participantes completaram a atividade de seleção de dispositivos móveis. Assim como no *survey*, nesta atividade foi questionado quais critérios foram usados para selecionar os dispositivos móveis que seriam usados no teste do app $2DUB$.

Para os participantes desta *Atividade*, as categorias que foram mais citadas estão relacionados à Tela (18%) em Hardware, Versão de SO e API (21%) em Software e *Marketshare* (18%) em Externos. Diferente do *Survey*, na atividade, a versão da API teve um uso maior, pois essa informação foi apresentada nas características da app. A quantidade máxima de critérios usados foi sete e a mínima foi um critério. Três participantes usaram até quatro critérios de seleção.

Geral: De forma geral, as categorias mais citadas estão relacionadas a *Marketshare* (21%), tamanho de tela (16%), e versão de SO (16%).

O *Marketshare* é um critério que é de fácil acesso, pois está relacionado à popularidade dos dispositivos, as versões de SO, as marcas mais conhecidas ou ao dispositivo

Tabela 24 – Categorias e critérios de seleção de dispositivos, *survey* e atividade

Categoria	**Criterios**	**Survey**		**Atividade**		**Geral**	
		#	**%**	**#**	**%**	**#**	**%**
Hardware	Tamanho de Tela	7	15%	6	18%	13	16%
	Resolução	3	7%	3	9%	6	8%
	Modelo	5	11%	1	3%	6	8%
	Memória	5	11%	1	3%	6	8%
	Rede	1	2%	2	6%	3	4%
	Processador	1	2%	-	-	1	1%
Software	SO	6	13%	7	21%	13	16%
	API	2	4%	7	21%	9	11%
	Recursos e permissões	1	2%	-	-	1	1%
Externos	Marketshare	11	24%	6	18%	17	21%
	Disponibilidade	4	9%	1	3%	5	6%

mais usado num determinado período; por isso, é uma das que é mais usada pelos participantes. Por exemplo, o dispositivo mais usado no Brasil em 2021 difere do mais usado nos Estados Unidos em 2021, e isso pode variar de ano em ano.

No caso do critério *"tamanho de tela"*, é uma característica de fácil percepção e tangível. Por outro lado, a categoria "versão da API" aparece apenas com 11% no seu uso, sendo que é uma característica importante para definir se um dispositivo será compatível ou não com uma app.

Uma análise das características dos dispositivos móveis selecionados mostrou que os participantes escolheram modelos de marcas mais conhecidas no Brasil. O *dataset* usado neste estudo continha 50 dispositivos de marcas conhecidas a nível mundial e alguns deles não eram comuns ou populares no Brasil. Assim, a localização dos participantes pode influenciar a não selecionar dispositivos móveis com marcas desconhecidas para eles. Esta informação permitiu evoluir a proposta desta tese.

Respondendo QP1: Os três critérios mais usados pelos profissionais na seleção de dispositivos são: Marketshare, tamanho de tela e versão de SO.

6.4.2 RQ2: Quais são as características dos dispositivos móveis selecionados pelos profissionais e qual é a sua cobertura?

Para cada planilha de resposta foram extraídos os dispositivos móveis selecionados por cada participante. Em seguida, foram identificados os dispositivos compatíveis (usáveis) e os não compatíveis (não-usáveis). Um dispositivo usável é aquele que cumpre com as características da app $2DUB$ indicadas na Figura 23. Do outro lado, um dispositivo não-usável é aquele onde não poderia ser instalado nem testado a app $2DUB$.

Na Tabela 25 pode-se observar os resultados de cada participante, a quantidade de dispositivos móveis selecionados, os dispositivos usáveis, e não-usáveis, assim como os valores médios e o desvio padrão. Pode ser observado que quatro participantes (P03, P05, P01, P04) selecionaram dispositivos móveis não usáveis. No caso do participante P04, dentro dos 13 dispositivos selecionados, apenas 6 poderiam ser usados para os testes e 7 dispositivos teriam que ser descartados, porque os mesmos não são compatíveis. O participante P03 selecionou 2 dispositivos que não são compatíveis com a app $2DUB$. Assim, ele não poderia testar a app com esses dispositivos. No caso deste participante, ele usou apenas um critério que foi "Marca conhecida", sendo desconsiderada a informação da versão da API suportada pela app, o que levou a escolher dispositivos que não seriam usáveis para o teste.

Assim, a principal causa para selecionar dispositivos não compatíveis (não-usáveis) é não considerar as informações da app que será testada. Os resultados da ***QP2*** mostra que selecionar dispositivos móveis que não poderiam ser usados para teste (não-usáveis) pode gerar prejuízos em relação ao custo e tempo dedicado no dispositivo que seria descartado. Assim, é importante que durante o processo da escolha de dispositivos para teste sejam consideradas as informações da app a ser testada.

Em uma análise adicional, na Tabela 26, foi calculado o nível de cobertura das cinco características (resolução, tamanho de tela, versão da API, *networking* e *Android marketshare*) alcançado pelos dispositivos selecionados pelos participantes. Estes resultados foram baseados usando apenas os dispositivos usáveis e compatíveis com a app. Três participantes (P01, P07, P08) selecionaram 11 dispositivos, com os quais conseguiram 100% de cobertura para as características resolução, tamanho de tela, e

Tabela 25 – Dispositivos selecionados, usáveis e não-usáveis.

IDP	Selecionados	Usável	Não-Usável
P02	2	2	0
P03	2	0	2
P09	4	4	0
P05	6	4	2
P06	6	6	0
P07	11	11	0
P08	11	11	0
P01	12	11	1
P04	13	6	7
Média	7,4	6,1	1,3
Desvio Padrão	4,4	4,1	2,3

Tabela 26 – Cobertura de características dos dispositivos "usáveis" selecionados.

IDP	Usável	Resolução	Tamanho	API	Netwk	MarketShare	Eficiência
P03	0	0	0	0	0	0	0%
P02	2	28,6	25,0	33,3	75,0	24,2	100,0%
P05	4	42,9	50,0	50,0	75,0	61,3	66,7%
P09	4	57,1	37,5	66,7	75,0	61,3	100,0%
P04	6	71,4	62,5	83,3	75,0	65,0	46,2%
P06	6	71,4	75,0	100	75,0	64,4	100,0%
P01	11	100	100	100	75,0	65,0	91,7%
P07	11	100	100	100	75,0	65,0	100,0%
P08	11	100	100	100	75,0	65,0	100,0%
Min/Max	**0/11**	**0/100**	**0/100**	**0/100**	**0/75**	**0/65**	**0/100%**
Média	**6,1**	**63,5**	**61,1**	**70,4**	**66,7**	**52,4**	**78,3%**
Desvio padrão	**4,1**	**35,1**	**36,1**	**36,1**	**25,0**	**23,7**	**35,1%**

versão da API. Por outro lado, o participante P03 não alcançou cobertura em nenhuma das características, pois os dispositivos que ele selecionou não seriam compatíveis com a app.

O resultado da eficiência da seleção de dispositivos por cada participante, usando a equação 6.1, é apresentado na última coluna da Tabela 26. Para o participante P03, a porcentagem de eficiência chega a zero. Como mencionado antes, este resultado é devido ao fato de que não foi considerada a informação da app que seria testada. A média obtida foi de 78.3%.

Respondendo o QP2: dentre os dispositivos selecionados pelos participantes, 18% foram modelos não-usáveis. O modelo que foi mais selecionado foi o Moto G6, o que

pode indicar uma forte inclinação a modelos conhecidos na localização da pessoa.

As principais conclusões deste estudo revelam que, ainda que as informações de uma app sejam apresentadas, alguns participantes não consideram tais informações na tomada de decisão para selecionar dispositivos, o que pode gerar prejuízo em relação ao tempo e custo. Da mesma forma, os dois principais critérios na hora de selecionar dispositivos estão relacionados ao dispositivo e ao mercado.

Por outro lado, esse estudo permitiu observar serem necessárias mais características dos dispositivos que permitam calcular o melhor custo-benefício na seleção de dispositivos, por exemplo, o preço dos dispositivos. Assim, foi feita uma atualização do *dataset* e algumas melhorias na abordagem *DeSeCT* que serão explicadas na seguinte seção.

6.5 Ameaças à Validade

Os resultados do *survey* podem ser generalizados para outros profissionais no mesmo contexto, porém não será generalizado para outras localidades. O foco deste estudo foi identificar os critérios que são usados pelos profissionais na seleção de dispositivos para teste. Assim, algumas ameaças à validade estão relacionadas a:

Validade de constructo: As duas partes do estudo foram realizadas seguindo as diretrizes do GQM e diretrizes para a execução de uma entrevista semi-estruturada. Foram realizados o planejamento para cada seção.

Validade externa: Os participantes selecionados foram pessoas da indústria que possuem experiência na área e aceitaram participar do estudo, além disso, cada participante é de organizações diferentes. Isso ajudou a mitigar o viés entre os participantes. A app usada no estudo pode ser considerada representativa e atual. No caso dos dados coletados para o *dataset* foram coletados com informações atuais.

6.6 Evolução da proposta

Os resultados do estudo permitiram confirmar os seguintes pontos: (1) As informações do dispositivo e o *marketshare* ainda são os mais considerados pelos profissionais da indústria na seleção de dispositivos; enquanto as informações da app nem tanto. (2) Alguns dos participantes foram influenciados pela sua localização na escolha da marca do dispositivo, evitando selecionar aqueles que não eram comercializados no país.

Da mesma forma que o lugar influencia na seleção, os resultados evidenciaram que o preço poderia ser um fator influenciador, um dos critérios que os profissionais usam é a disponibilidade do dispositivo, a mais dispositivos, maior será o recurso financeiro necessário. Assim, o preço é um fator importante a ser considerado, pois cada organização determina recursos de formas diferentes para os dispositivos. Considerando a variável preço, se faz necessário apresentar várias soluções onde o usuário possa ter diversas opções conforme os seus recursos. Logo, essas duas novas características foram adicionadas à proposta.

6.6.1 Detalhes da evolução do $DeSeCTv3$:

A versão $DeSeCTv2$ apresentava como resultado apenas um conjunto de dispositivos; porém, nessa nova versão será incluída a variável preço tanto no *dataset* como na proposta. Essas mudanças são apresentadas na Figura 27 nas etapas 1 e 5.

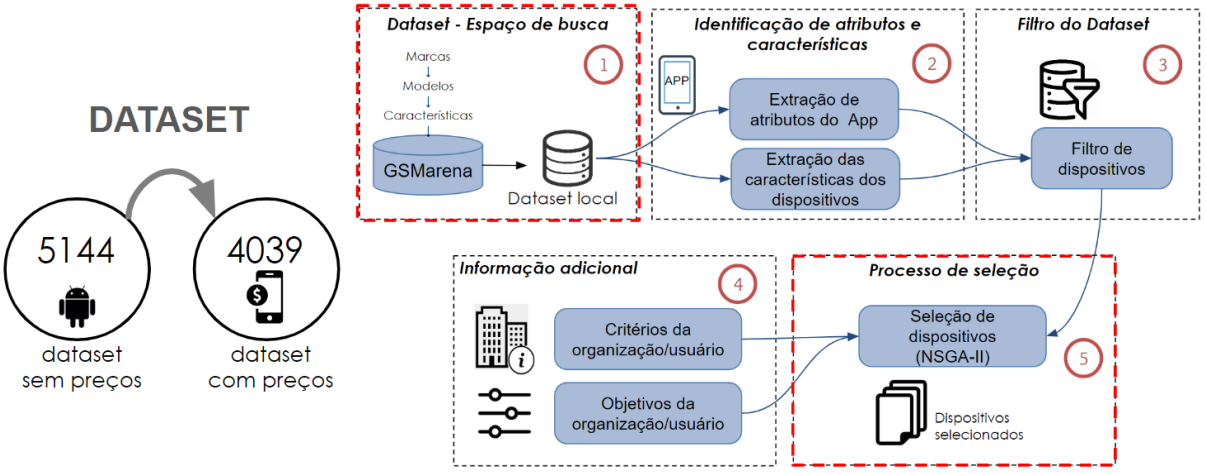


Figura 27 – Evolução do $DeSeCTv3$

Dataset: No *dataset* foi necessário obter o preço de todos os dispositivos móveis e realizar um filtro apenas com os dispositivos que tinham esses dados. Os dispositivos que não tinham preço foram descartados. Foi necessário fazer uma normalização em relação à moeda, para todos seguirem o mesmo padrão.

Metodologia: Nesta seção, foi necessário modificar a função objetivo de minimização; o preço do dispositivo foi incorporado na função. Assim, esta função visa reduzir o custo com a menor quantidade de dispositivos móveis e o menor preço possível. A equação atualizada com o preço é apresentada na equação (6.2).

$$MinFit = \sum_{i=1}^{n}(P_i * D_i) \tag{6.2}$$

Onde i significa o índice de cada dispositivo móvel. P_i representa o preço do dispositivo; D_i pode ser "1" se o dispositivo foi selecionado ou "0" se não foi selecionado. Logo, n representa o tamanho do cromossomo. Esta função realiza a soma do preço de todos os dispositivos selecionados com valor 1. O resultado representa o preço total dos dispositivos que foram selecionados para o teste. Quanto menor o resultado, melhor, pois se tem um conjunto menor e menos custoso de dispositivos para testar. O objetivo de maximização se mantém da mesma forma que no $DeSeCTv2$.

Resultado: O resultado esperado para uma app é a apresentação de um conjunto de soluções, as quais representam um conjunto de dispositivos e o preço total por cada solução.

6.6.2 Estudo com 30 apps no *DeSeCTv3*

O objetivo deste estudo foi verificar o comportamento do $DeSeCTv3$ com as novas configurações em diversas apps. Para este estudo foram consideradas 30 apps (Tabela 17) sendo definidos os seguintes parâmetros de *input*:

- (1) O AG seguiu as seguintes configurações: tipo de algoritmo genético: *Multi-Objective NSGA-II*; codificação: *Binária*; cruzamento: *Dois pontos*; mutação: *Flip Bit Based*; processo de seleção: *Elitismo*; a probabilidade usada foi 80% de taxa de cruzamento e 1% de probabilidade de mutação;

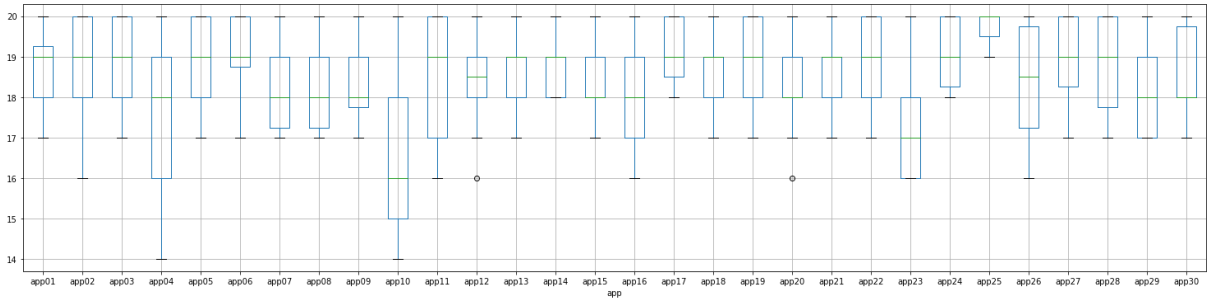


Figura 28 – Resultado das 30 Apps - quantidade mínima e máxima das soluções

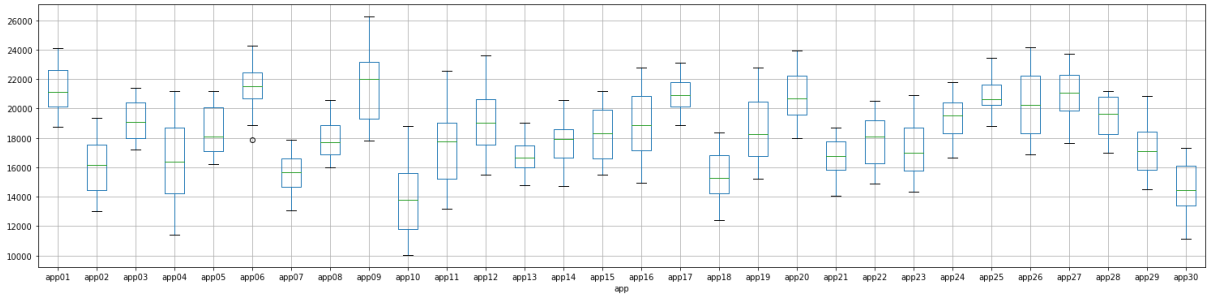


Figura 29 – Resultado das 30 Apps - preço mínimo e máximo das soluções

- (2) A população Inicial de dispositivos selecionados foi 20;
- (3) O número máximo de dispositivos por solução foi 20;
- (4) Foi apresentado todas as soluções dentro do limite máximo de dispositivos.

Para cada app foi extraída as suas informações, filtro do *dataset*, aplicar as configurações e logo executado o $DeSeCT$. Assim, a Tabela 27 apresenta os resultados das 30 apps. A coluna "Soluções" representa o número de soluções disponíveis para acada app dos quais o usuário poderá escolher. Estas soluções foram obtidas considerando os parâmetros definidos anteriormente. A coluna "Dispositivos" representa o número mínimo, máximo, e médio de dispositivos selecionados dentre as soluções encontradas. A coluna "Custo" representa o custo total mínimo, máximo, e médio dentre as soluções encontradas.

De forma gráfica, a Figura 28 indica os valores mínimo, máximo, mediana e *outlier* (se tiver) para cada app. Observa-se que para todas as apps o número máximo de dispositivos foi 20, isso se deve à regra número (3) definido nos parâmetros de *input*, este valor pode ser ajustável conforme as necessidades do usuário. A Figura 29 apresenta o preço total mínimo, máximo e mediana de todas as soluções para cada app.

Tabela 27 – Resultado das 30 Apps com o $DeSeCTv3$

APP	Soluções	Dispositivos			Custo (R$)		
		min	max	med	min	max	med
app01	8	17	20	19	18776	24108	21132
app02	25	16	20	19	13034	19358	16134
app03	9	17	20	19	17228	21400	19106
app04	31	14	20	18	11430	21164	16390
app05	9	17	20	19	16188	21178	18110
app06	12	17	20	19	17882	24255	21534
app07	26	17	20	18	13046	17893	15663
app08	10	17	20	18	15984	20556	17720
app09	16	17	20	18	17811	26287	22029
app10	34	14	20	16	10026	18803	13806
app11	21	16	20	19	13193	22542	17781
app12	18	16	20	19	15512	23598	19041
app13	9	17	20	19	14778	19056	16638
app14	25	18	20	19	14719	20552	17914
app15	10	17	20	18	15473	21161	18317
app16	14	16	20	18	14939	22800	18870
app17	11	18	20	19	18884	23100	20930
app18	24	17	20	19	12399	18391	15291
app19	23	17	20	19	15197	22761	18265
app20	13	16	20	18	18006	23948	20662
app21	13	17	20	19	14054	18702	16780
app22	14	17	20	19	14893	20508	18103
app23	9	16	20	17	14342	20887	16964
app24	10	18	20	19	16680	21762	19532
app25	7	19	20	20	18790	23447	20657
app26	18	16	20	19	16866	24182	20245
app27	14	17	20	19	17644	23707	21089
app28	8	17	20	19	16978	21169	19644
app29	22	17	20	18	14524	20848	17097
app30	18	17	20	18	11109	17309	14466

Assim, a Figura 31 apresenta o conjunto de soluções encontrado para cada app. Nesta figura é possível observar a relação entre o custo e a cobertura de cada solução. O eixo horizontal representa o custo da solução e o eixo vertical representa a cobertura em relação à função *fitness* do AG. Dessa forma, os usuários podem escolher, dentre as soluções, baseados nos seus objetivos e recursos financeiros.

Como exemplo, na Figura 30, para a *APP01*, foram encontradas 8 soluções, o número mínimo de dispositivos selecionados foi 17, com um custo de R$ 18776 e o número máximo foi 20 dispositivos com um custo de R$ 24108.

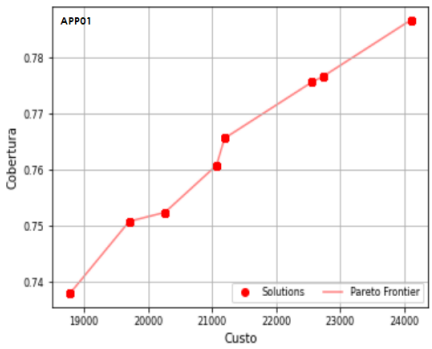


Figura 30 – Relação Custo - Cobertura para as soluções da App01

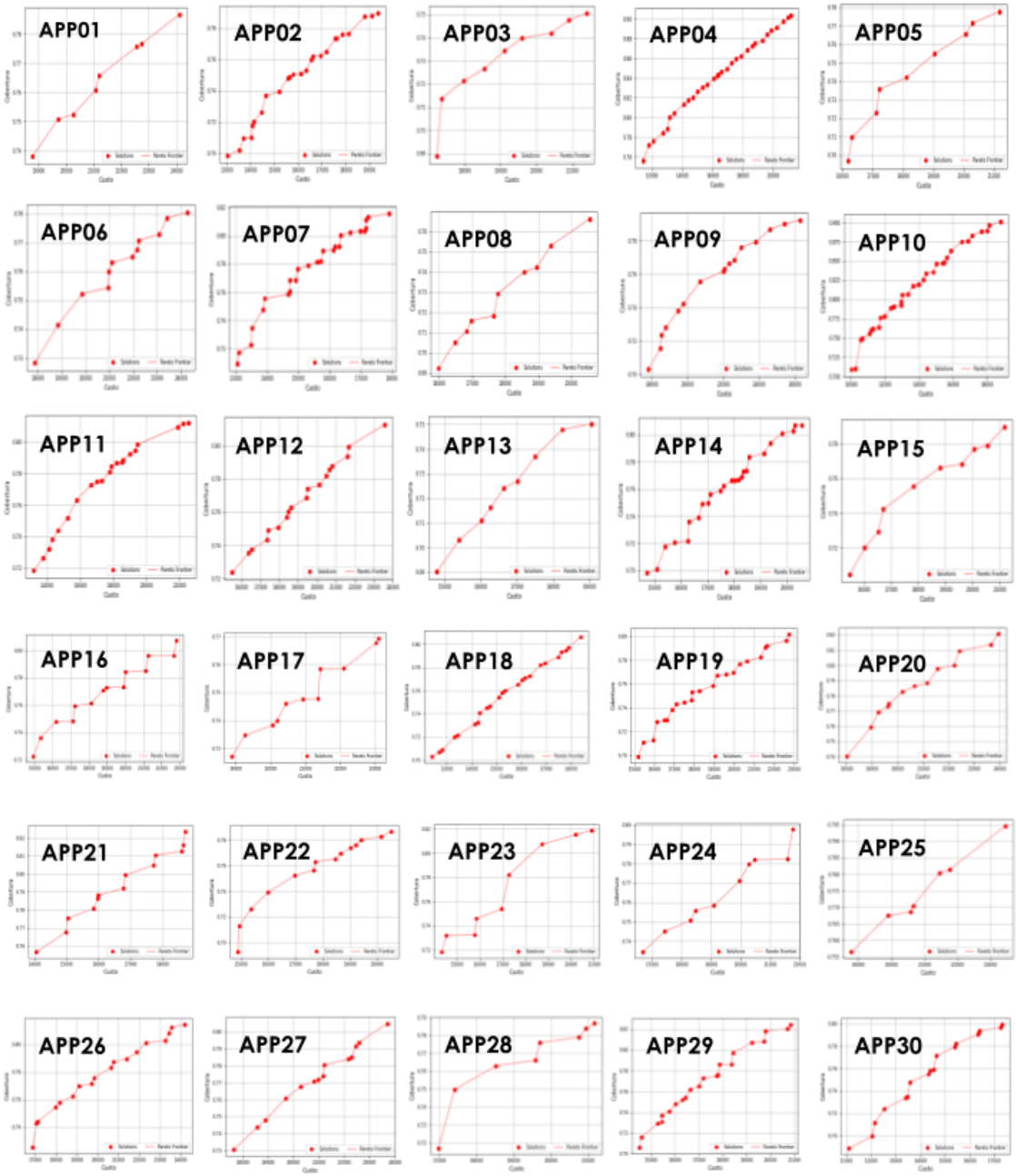


Figura 31 – Resultado das soluções para 30 Apps - $DeSeCTv3$

6.7 Considerações do Capítulo

O estudo realizado com profissionais da indústria permitiu constatar que não usar as informações da app pode levar a uma escolha de dispositivos não-utilizáveis para teste. Um ponto muito importante foi identificar a necessidade de adequar os parâmetros do AG conforme o contexto do usuário e a inclusão da característica preço, pois é uma característica altamente influenciada pelo *marketshare*, a qual é um dos critérios mais usados pelos profissionais. Assim, as contribuições deste estudo envolvem um conjunto de critérios de seleção de dispositivos pelos profissionais da indústria. Além disso, permitiu identificar alguns fatores que podem influenciar a seleção, como a localização do participante.

A inclusão do preço na proposta permitiu que os resultados do $DeSeCT$ tenham informações mais relevantes para ajudar na escolha e permitir que o usuário possa escolher dentre várias soluções conforme seu contexto e recursos.

7

CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as considerações finais em relação à contribuição desta tese. Como perspectivas futuras, além de estudos que podem ser realizados para refinamento do $DeSeCT$.

7.1 Conclusões

A diversidade dos dispositivos móveis continua crescendo a cada ano. É inegável o fato de que muitos dispositivos irão desaparecer do mercado e outros novos irão aparecer. Essa atualização constante também impacta no desenvolvimento das app e consequentemente no teste dessas apps. O teste de compatibilidade é o tipo de teste que está relacionado com esta diversidade de dispositivos no mercado.

Assim, um primeiro passo desta tese envolveu a identificação do corpo de conhecimento, por meio de um mapeamento da literatura. Foi executado o Mapeamento de Literatura Multivocal - MLM, a qual teve duas fases principais; a primeira fase foi baseada na abordagem *Snowballing* para estudos na literatura técnica, e a segunda fase foi baseada na revisão da literatura cinza. A literatura cinza foi considerada devido à necessidade de conhecer como é feito ou abordado a seleção de dispositivos na indústria.

Dentro dos resultados obtidos nos estudos realizados nesta tese, foi identificado que o processo de seleção de dispositivos móveis é considerado uma tarefa que demanda muito esforço e tempo, o que pode ser a causa de que um dos critérios que são usados para a seleção é a disponibilidade do dispositivo ou dentro do contexto de localização

da pessoa, ou organização. Isso torna o processo de seleção de dispositivos uma tarefa não primordial, dentro do processo de qualidade de uma app, afetando diretamente com a qualidade da app, quando são encontrados erros pelos usuários em dispositivos que não foram considerados para teste.

Neste sentido, esta tese definiu uma abordagem chamada de $DeSeCT$ (*Devices Selection for Compatibility Testing*) desenvolvida a partir de uma metodologia científica baseada na condução de estudos experimentais que permitem a evolução da abordagem. O $DeSeCT$ inicia desde a identificação de características da app e dos dispositivos móveis e apresenta um conjunto de soluções que auxilie na tomada de decisão no processo de seleção de dispositivos móveis.

7.2 Resultados obtidos

Analisando as sub-questões de pesquisa da Seção 1.3 deste documento, as seguintes resultados foram obtidos:

O que diz a literatura com respeito a teste de compatibilidade em apps móveis?

A literatura não tinha mapeamentos relacionados especificamente a teste de compatibilidade. Assim, a primeira contribuição é o corpo de conhecimento relacionado ao teste de compatibilidade em apps móveis. O resultado foi dividido em duas grandes áreas: *Hardware* e *Software*. Na área de hardware, os principais estudos estão relacionados à seleção de dispositivos, serviços de teste e teste baseado em interface. Um fator em comum dentre os estudos encontrados está relacionado à diversidade de dispositivos no mercado. Devido a sua constante atualização, os estudos precisam ir se adequando a novas tecnologias e tipos de serviço de teste. Um claro exemplo é a evolução das telas de dispositivos móveis. Alguns dos modelos atuais possuem telas dobráveis e o teste baseado em interface deve ser estudado nesse novo contexto.

Enquanto na área de software, os principais estudos estão relacionados à evolução da API, detecção de problemas de API e suporte ao API. Estes estudos permitiram observar a constante atualização do SO dos dispositivos móveis. No caso do Android,

a evolução da API é um fator que afeta os testes de compatibilidade. O mapeamento permitiu verificar que o problema da fragmentação está presente tanto no hardware quanto no software.

O que diz a literatura a respeito das técnicas empregadas na seleção de dispositivos móveis para teste?

Dentro do corpo de conhecimento gerado, dentre os resultados obtidos foram as técnicas mais utilizadas para seleção de dispositivos; dentre elas temos: mineração de dados, Método each-choice, algoritmo genético, algoritmo K-means, e método Randômico. No entanto, algumas dessas abordagens não são mais possíveis de reproduzir, o que motivou este trabalho a ser pesquisado melhoras nas abordagens usadas atualmente.

Qual é a abordagem e os critérios usados para a seleção de dispositivos de teste na indústria?

Os estudos realizados com a indústria permitiram conhecer e verificar de perto como é conduzido este processo. No primeiro estudo, com a participação de uma organização, foi obtido conhecimento em relação ao esforço e tempo requerido para selecionar dispositivos. Este processo não é executado com certa frequência. Foi observado que este processo se faz aproximadamente a cada um ou dois anos. Em algumas organizações menores pode ser até maior o tempo.

O segundo estudo, com a participação de mais profissionais de diversas organizações, permitiu identificar os critérios usados para a seleção de dispositivos. Os três critérios mais usados são: *marketshare*, tamanho de tela e sistema operacional. O *marketshare* nesta pesquisa está relacionado às tendências, em relação aos dispositivos mais usados ou os mais populares conforme a localização. Por ser uma informação de fácil acesso, ela é a mais usada dentre os profissionais.

7.3 Principais contribuições da pesquisa

Como resultado deste trabalho, as principais contribuições desta pesquisa foram as seguintes:

Corpo de conhecimento sobre Teste de Compatibilidade e Seleção de dispo-

sitivos móveis: O corpo de conhecimento sobre teste de compatibilidade foi construído por meio da execução do Mapeamento de Literatura Multivocal - MLM. Esta metodologia contempla os trabalhos publicados na literatura técnica, quanto publicações na literatura cinza. Permitindo ter uma visão do estado-da-arte e o estado-da-prática, desta forma o entendimento dos desafios de teste de compatibilidade em apss ficou mais claro. Os resultados deste mapeamento foram publicados no *International Journal of Computer Applications in Technology (IJCAT), Vol. 69, No. 2, 2022* - DOI: 10.1504/IJCAT.2022.10051895.

Modelagem do problema de seleção de dispositivos móveis: A seleção de dispositivos móveis é um problema multiobjetivo que busca maximizar a diversidade de características e minimizar o custo e quantidade de dispositivos selecionados. Assim, tendo objetivos conflitantes, foi possível modelar o problema como um problema de otimização, o que permitiu definir objetivos que possam variar conforme a necessidade da app.

***Dataset* de dispositivos móveis:** Para a procura de soluções num conjunto de dispositivos que atendam os objetivos definidos na modelagem foi necessário a construção de um *dataset*. Este *dataset* foi construído com informações reais, sendo a principal fonte para a busca de soluções no processo de seleção de dispositivos móveis para teste de compatibilidade. Devido ao constante atualização dos dispositivos móveis, este *dataset* precisa ser atualizado sempre.

DeSeCT - Device Selection for Compatibility Testing: A abordagem $DeSeCT$ -Seleção de Dispositivos para testes de compatibilidade- na sua versão final é o principal meio de suporte para o processo de seleção de dispositivos móveis para testes, permitindo minimizar o esforço necessário nesta atividade e otimizando a escolha conforme os requerimentos e características da app. Foram executados diversos estudos visando evoluir a abordagem para uma versão que apresente soluções vaiáveis para os usuários. A abordagem e as atualizações estão disponíveis online [1].

[1] https://github.com/MobileCompatibilityTesting/DeSeCT

7.4 Limitações

As limitações desta pesquisa envolvem principalmente os seguintes itens:

- *Amostra de participantes:* como citado antes, o processo de seleção de dispositivos móveis para teste não é um processo repetitivo em uma organização ou incluso para um desenvolvedor. Após a primeira seleção a próxima será realizada depois de um período longo de tempo. Nesse contexto, encontrar organizações que estejam prestes a executar esse processo não é fácil.
- *Avaliação da proposta:* Os estudos que foram parte deste trabalho permitiram a evolução do $DeSeCT$. As avaliações finais foram feitas com diversas apps disponíveis nos repositórios.
- *Infraestrutura:* a execução do $DeSeCT$ é via linha de comando. Assim, a disponibilização da proposta ainda precisa de uma interface que possa ser usada por qualquer desenvolvedor.
- *Dataset:* O *dataset* usado na proposta é baseado nas informações de dispositivos disponibilizados pelo site do *GSMArena*, a qual tem atualizações constantes; nesse contexto, o *dataset* ainda precisa ser atualizado constantemente para refletir as informações atualizadas dos novos modelos de dispositivos no mercado.

7.5 Perspectivas futuras

Há a intenção de continuar a pesquisa deste trabalho seguindo os seguintes direcionamentos:

- Disponibilizar o $DeSeCT$ via web é a continuação mais próxima desta pesquisa, pois seria de ajuda para todos os usuários que queiram validar a diversidade de dispositivos que precisam testar conforme o contexto deles.
- Outra área de pesquisa, com potencial de investigação, é a execução de casos de teste, usando as soluções apresentadas pelo $DeSeCT$, para investigar a descoberta de erros nos diversos dispositivos sugeridos para teste.

- Em relação ao *dataset* usado, é necessário pesquisar outras formas de manter atualizado a base de dispositivos móveis e o preço que é um dado usado no $DeSeCT$.

- A plataforma usada para esta pesquisa foi Android, porém é necessário um estudo para incorporar a plataforma iOS.

REFERÊNCIAS

AKOUR, M.; AL-ZYOUD, A. A. Mobile Software Testing : Thoughts , Strategies , Challenges , and Experimental Study. *International Journal of Advanced Computer Science and Applications*, v. 7, n. 6, 2016.

ALLIX, K. et al. Androzoo: Collecting millions of android apps for the research community. In: *IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. [S.l.: s.n.], 2016. p. 468–471.

AMALFITANO, D. et al. *Testing Android Mobile Applications: Challenges, Strategies, and Approaches*. [S.l.]: Elsevier Inc., 2013. v. 89. 1–52 p.

ARCURI, A.; FRASER, G. On parameter tuning in search based software engineering. In: *Proceedings of the Third International Conference on Search Based Software Engineering*. [S.l.]: Springer-Verlag, 2011. p. 33–47.

AVANTICA. *Selecting Mobile Devices for Testing*. 2017. Acesso em Fevereiro de 2020. Disponível em: <https://www.avantica.net/blog/selecting-mobile-devices-for-testing>.

AVANTICA. *Testing: Choosing Mobile Devices*. 2020. Acesso em Agosto de 2021. Disponível em: <https://www.avantica.com/blog/testing-choosing-mobile-devices>.

BADAMPUDI, D.; WOHLIN, C.; PETERSEN, K. Experiences from using snowballing and database searches in systematic literature studies. In: *EASE '15 Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.: s.n.], 2015. p. 1–10.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The Goal Question Metric Approach. v. 2, p. 1–10, 1994.

BROWSERSTACK. *Test on the right mobile devices*. 2019. Acesso em Agosto de 2020. Disponível em: <https://www.browserstack.com/test-on-the-right-mobile-devices>.

CAI, H. et al. A large-scale study of application incompatibilities in android. In: *ACM SIGSOFT International Symposium on Software Testing and Analysis*. [S.l.: s.n.], 2019. p. 352–362.

CHAITANYA, K. *Choose your Test Devices Wisely*. 2018. Acesso em Agosto de 2021. Disponível em: <https://medium.com/@connecttokc/choose-your-test-devices-wisely-94e51c7870ac>.

CHANDRASEKARAN, G. et al. Multiobjective optimisation of bevel gear pair design using nsga-ii. *Materials Today: Proceedings*, v. 16, p. 351 – 360, 2019.

CHENG, J. et al. Mobile compatibility testing using multi-objective genetic algorithm. In: *2015 IEEE Symposium on Service-Oriented System Engineering*. [S.l.]: IEEE Computer Society, 2015. p. 302–307.

COHEN, J. *Statistical Power Analysis for the Behavioral Sciences*. [S.l.]: Lawrence Erlbaum Associates, 1988.

COVEROS. *How Do I Choose Mobile Devices for Testing?* 2016. Acesso em Fevereiro de 2020. Disponível em: <https://www.coveros.com/how-do-i-choose-mobile-devices-for-testing/>.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, p. 182–197, 2002.

Di Nucci, D. et al. A test case prioritization genetic algorithm guided by the hypervolume indicator. *IEEE Transactions on Software Engineering*, p. 1–1, 2018.

EASYQA. *How to test mobile application*. 2019. Acesso em Fevereiro de 2020. Disponível em: <https://geteasyqa.com/qa/mobile-apps-testing/>.

FARIA, K. A. C. et al. On using collaborative economy for test cost reduction in high fragmented environments. *Future Generation Computer Systems*, Elsevier B.V., v. 95, p. 502–510, 2019.

FAZZINI, M.; ORSO, A. Automated cross-platform inconsistency detection for mobile apps. *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, p. 308–318, 2017.

FORTIN, F.-A. et al. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, v. 13, p. 2171–2175, 2012.

GAROUSI, V.; FELDERER, M.; MÄNTYLÄ, M. V. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, Elsevier B.V., v. 106, n. May 2018, p. 101–121, 2019.

GOLBARG, M. C.; LUNA, H. P. L. *Otimização combinatória e programação linear*. [S.l.]: Editora CAMPUS, Rio de Janeiro, 2000.

GRANDE, A. D. S.; RODRIGUES, R. D. F.; DIAS-NETO, A. C. A framework to support the selection of software technologies by search-based strategy. In: *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*. [S.l.: s.n.], 2014. p. 979–983.

HALPERN, M. et al. Mosaic: Cross-platform user-interaction record and replay for the fragmented android ecosystem. *IEEE International Symposium on Performance Analysis of Systems and Software*, p. 215–224, 2015.

HAM, H. K.; PARK, Y. B. Mobile application compatibility test system design for android fragmentation. In: *International Conference on Advanced Software Engineering and Its Applications*. [S.l.: s.n.], 2011. p. 314–320.

HAMBURG, M. *Basic statistics: A modern approach.* 2. ed. [S.l.]: Harcourt Brace Jovanovich, 1980.

HAN, D. et al. Understanding android fragmentation with topic analysis of vendor-specific bugs. *Proceedings - Working Conference on Reverse Engineering, WCRE*, p. 83–92, 2012.

HARMAN, M. The current state and future of search based software engineering. In: *Workshop on the Future of Software Engineering, FOSE*. [S.l.: s.n.], 2007. p. 342–357.

HARMAN, M.; JONES, B. F. Search-based software engineering. *Inf. Softw. Technol.*, v. 43, n. 14, p. 833–839, 2001.

HARMAN, M.; MANSOURI, A. Search based software engineering: Introduction to the special issue of the ieee transactions on software engineering. *IEEE Transactions on Software Engineering*, IEEE Computer Society, v. 36, n. 6, p. 737, 2010.

HARMAN, M.; MANSOURI, S. A.; ZHANG, Y. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, v. 45, n. 1, 2012.

HARYONO, S. A. et al. Androevolve: Automated update for android deprecated-api usages. In: *IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. [S.l.: s.n.], 2021. p. 1–4.

HE, D. et al. Understanding and detecting evolution-induced compatibility issues in Android apps. *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE 2018*, p. 167–177, 2018.

HOLL, K.; ELBERZHAGER, F. Quality Assurance of Mobile Applications : A Systematic Mapping Study. *15th International Conference on Mobile and Ubiquitous Multimedia ACM*, p. 101–113, 2016.

HOLLAND, J. H. Genetic algorithms. *Scholarpedia*, v. 7, p. 1482, 2012.

HUANG, H. et al. Understanding and detecting callback compatibility issues for Android applications. *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE*, p. 532–542, 2018.

HUANG, J.-f. AppACTS: Mobile App Automated Compatibility Testing Service. In: *2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. [S.l.]: IEEE, 2014. p. 85–90.

HUANG, J.-F.; GONG, Y.-Z. Remote mobile test system: A mobile phone cloud for application testing. *CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science*, p. 587–590, 2012.

JIANG, B.; LONG, X.; GAO, X. Mobiletest: A tool supporting automatic black box test for software on smart mobile devices. In: . [S.l.: s.n.], 2007. p. 0–6.

JOORABCHI, M. E.; MESBAH, A.; KRUCHTEN, P. Real Challenges in Mobile App Development. In: *IEEE International Symposium on Empirical Software Engineering and Measurement*. [S.l.: s.n.], 2013. p. 15–24.

KAUR, A.; KAUR, K. Investigation on test effort estimation of mobile applications: Systematic literature review and survey. *Information and Software Technology*, Elsevier B.V., v. 110, p. 56–77, 2019.

KHALID, H. et al. Prioritizing the Devices to Test Your App on: A Case Study of Android Game Apps. *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, p. 610–620, 2014.

KI, T. et al. Mimic: UI Compatibility testing system for android apps. *International Conference on Software Engineering*, IEEE, v. 2019-May, p. 246–256, 2019.

KIRUBAKARAN, B.; KARTHIKEYANI, V. Mobile application testing — Challenges and solution approach through automation. In: *International Conference on Pattern Recognition, Informatics and Mobile Engineering*. [S.l.: s.n.], 2013. p. 79–84.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. [S.l.], 2007.

KONG, P. et al. Automated Testing of Android Apps: A Systematic Literature Review. *IEEE Transactions on Reliability*, p. 1–22, 2018. ISSN 00189529.

KOWALCZYK, E.; COHEN, M. B.; MEMON, A. M. Configurations in Android testing: they matter. *Proceedings of the 1st International Workshop on Advances in Mobile App Analysis - A-Mobile 2018*, p. 1–6, 2018.

LI, C. et al. ELEGANT : Towards effective location of fragmentation-induced compatibility issues for android apps. In: *25th Asia-Pacific Software Engineering Conference*. [S.l.: s.n.], 2018. p. 278–287.

LI, L. et al. CiD: Automating the Detection of API-Related Compatibility Issues in Android Apps. In: *International Symposium on Software Testing and Analysis ISSTA*. [S.l.: s.n.], 2018.

LIU, C. H. A compatibility testing platform for android multimedia applications. *Multimedia Tools and Applications*, v. 78, n. 4, p. 4885–4904, 2019.

LU, X. et al. Prada Prioritizing Android Devices for Apps by Mining Large-Scale Usage Data. *Proceedings of the 38th International Conference on Software Engineering ICSE*, p. 3–13, 2016.

M. Amen, B.; M. Mahmood, S.; LU, J. Mobile Application Testing Matrix and Challenges. In: *Computer Science & Information Technology (CS & IT)*. [S.l.: s.n.], 2015. p. 27–40.

MAHMUD, T.; CHE, M.; YANG, G. Android compatibility issue detection using api differences. In: *IEEE International Conference on Software Analysis, Evolution and Reengineering SANER*. [S.l.: s.n.], 2021. p. 480–490.

MAHMUD, T.; CHE, M.; YANG, G. Acid: An api compatibility issue detector for android apps. In: *IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings ICSE-Companion*. [S.l.: s.n.], 2022. p. 1–5.

MAHMUD, T.; CHE, M.; YANG, G. Android api field evolution and its induced compatibility issues. In: *Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. [S.l.]: Association for Computing Machinery, 2022. p. 34–44.

MAO, K.; HARMAN, M.; JIA, Y. Sapienz: multi-objective automated testing for Android applications. *Proceedings of the 25th International Symposium on Software Testing and Analysis - ISSTA*, p. 94–105, 2016.

MEDIUM. *Test on the Right Devices*. 2017. Acesso em Fevereiro de 2020. Disponível em: <https://medium.com/mozilla-tech/test-on-the-right-devices-90e15a3c0254>.

MEDIUM@99TESTS. *Top 5 Parameters For Choosing Devices For Mobile App Testing*. 2016. Acesso em Fevereiro de 2020. Disponível em: <https://medium.com/@99tests/top-5-parameters-for-\choosing-devices-for-mobile-app-testing-bb4db448a723>.

MENEGASSI, A. A.; ENDO, A. T. Automated tests for cross-platform mobile apps in multiple configurations. *IET Software*, v. 14, n. 1, p. 27–38, 2020.

MOBILIO, M. et al. FILO: FIx-LOcus localization for backward incompatibilities caused by Android framework upgrades. In: *IEEE/ACM International Conference on Automated Software Engineering*. [S.l.: s.n.], 2020. p. 1292–1296.

MUCCINI, H.; FRANCESCO, A. D.; ESPOSITO, P. Software testing of mobile applications: Challenges and future research directions. In: *Automation of Software Test, Second International Workshop on*. [S.l.]: IEEE Computer Society, 2012. p. 29–35.

MUKHERJEE, D.; RUHE, G. Analysis of compatibility in open source Android mobile apps. In: *International Workshop on Artificial Intelligence and Requirements Engineering, AIRE*. [S.l.: s.n.], 2020. p. 70–78.

NAITH, Q.; CIRAVEGNA, F. Hybrid Crowd-powered Approach for Compatibility Testing of Mobile Devices and Applications. *Proceedings of the 3rd International Conference on Crowd Science and Engineering ICCSE*, p. 1–8, 2018.

NAITH, Q.; CIRAVEGNA, F. Mobile devices compatibility testing strategy via crowdsourcing. *International Journal of Crowd Science*, v. 2, n. 3, p. 225–246, 2018.

NIELEBOCK, S. et al. AndroidCompass : A Dataset of Android Compatibility Checks in Code Repositories. In: *IEEE/ACM 18th International Conference on Mining Software Repositories MSR*. [S.l.: s.n.], 2021. p. 5.

NOEI, E. et al. A study of the relation of mobile device attributes with the user-perceived quality of Android apps. *25th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER*, Empirical Software Engineering, p. 469, 2018.

OPENSIGNAL. *Android Fragmentation Visualized (August 2015)*. 2015. Acesso em Janeiro de 2017. Disponível em: <https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2015-08/2015_08_fragmentation_report.pdf>.

PLUSQA. *Top Devices for Android Testing in 2018*. 2018. Acesso em Fevereiro de 2020. Disponível em: <https://plusqa.com/2018/01/29/top-devices-for-android-testing-in-2018/>.

PLUSQA. *The Top Devices for Android App Testing in 2019*. 2019. Acesso em Fevereiro de 2020. Disponível em: <https://plusqa.com/2019/01/17/android-app-testing/>.

PLUSQA. *Top Devices for Android App Testing in 2021*. 2021. Acesso em Agosto de 2020. Disponível em: <https://plusqa.com/2018/01/29/top-devices-for-android-testing-in-2018/>.

QATESTLAB. *How to Choose Devices for Mobile Testing?* 2014. Acesso em Fevereiro de 2020. Disponível em: <https://blog.qatestlab.com/2014/04/14/how-to-choose-devices-for-mobile-testing/>.

QATESTLAB. *Top Android Devices to choose for Testing Apps in 2019*. 2019. Acesso em Fevereiro de 2020. Disponível em: <https://blog.qatestlab.com/2019/08/15/android-device-for-testing/>.

REN, Y. et al. Cross-device difference detector for mobile application gui compatibility testing. In: *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops ICSTW*. [S.l.: s.n.], 2022. p. 253–260.

RUBINOV, K.; BARESI, L. What Are We Missing When Testing Our Android Apps? *Computer*, v. 51, n. 4, p. 60–68, 2018.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, v. 14, n. 2, p. 131–164, 2009.

SAHINOGLU, M.; INCKI, K.; AKTAS, M. S. Mobile Application Verification: A Systematic Mapping Study. In: *International Conference on Computational Science and Its Applications ICCSA*. [S.l.: s.n.], 2015. p. 147–163.

SALESFORCE. *Technical Requirements for Phones - Mobile Devices Used for Testing*. 2021. Acesso em Agosto de 2021. Disponível em: <https://help.salesforce.com/s/articleView?id=sf.technical_requirements_phone.htm&type=5>.

SAUCELABS. *How to Choose Mobile Devices for Testing?* 2016. Acesso em Fevereiro de 2020. Disponível em: <https://saucelabs.com/blog/how-to-choose-mobile-devices-for-testing>.

SCALABRINO, S. et al. API compatibility issues in Android: Causes and effectiveness of data-driven detection techniques. *Empirical Software Engineering*, v. 25, n. 6, p. 5006–5046, 2020.

SCALABRINO, S. et al. Data-driven solutions to detect api compatibility issues in android: An empirical study. In: *IEEE/ACM International Conference on Mining Software Repositories (MSR)*. [S.l.: s.n.], 2019. p. 288–298.

SHAO, X. et al. NSGA-II-Based Multi-objective Mission Planning Method for Satellite Formation System. *Journal of Aerospace Technology and Management*, v. 8, p. 451 – 458, 2016.

SHULL, F.; CARVER, J.; TRAVASSOS, G. H. An Empirical Methodology for Introducing Software Processes. In: *Proceedings of European Software Engineering Conference*. [S.l.: s.n.], 2001. p. 288 – 296.

SILVA, B. et al. Saintdroid: Scalable, automated incompatibility detection for android. In: *52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks DSN*. [S.l.: s.n.], 2022. p. 567–579.

SILVA, D. B. et al. Characterizing mobile apps from a source and test code viewpoint. *Information and Software Technology*, Elsevier, v. 101, p. 32–50, 2018.

Statista. *Forecast number of mobile users worldwide from 2020 to 2025*. 2021. Acesso em Maio de 2021. Disponível em: <https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010/>.

Statista. *Most popular Google Play app categories as of 1st quarter 2021, by share of available apps*. 2021. Acesso em Julho de 2021. Disponível em: <https://www.statista.com/statistics/279286/google-play-android-app-categories/>.

Statista. *MMobile app downloads worldwide from 2021 to 2026, by store*. 2022. Acesso em Agosto de 2022. Disponível em: <https://www.statista.com/statistics/1010716/apple-app-store-google-play-app-downloads-forecast/>.

Statista. *Mobile Android operating system market share by version in the United States from July 2017 to September 2022*. 2022. Acesso em Abril de 2022. Disponível em: <https://www.statista.com/statistics/865837/mobile-android-version-share-in-the-us/>.

Statista. *Mobile operating systems' market share worldwide from January 2012 to August 2022*. 2022. Acesso em Agosto de 2020. Disponível em: <https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/>.

SUN, X. et al. Mining android api usage to generate unit test cases for pinpointing compatibility issues. *The 37th IEEE/ACM International Conference on Automated Software Engineering (ASE'22)*, 2022.

TEST48. *5 Simple Guidelines For Picking The Right Mobile Testing Devices*. 2019. Acesso em Fevereiro de 2020. Disponível em: <https://www.test48.com/blog/5-simple-guidelines-for-selecting-the-right-mobile-app\-testing-devices/>.

TRAMONTANA, P. et al. Automated functional testing of mobile applications: a systematic mapping study. *Software Quality Journal*, p. 1–53, 2018.

Vandana; Singh, A. Multi-objective test case minimization using evolutionary algorithms: A review. In: *International conference of Electronics, Communication and Aerospace Technology ICECA*. [S.l.: s.n.], 2017. p. 329–334.

VILKOMIR, S. Multi-device coverage testing of mobile applications. *Software Quality Journal*, Software Quality Journal, v. 26, n. 2, p. 197–215, 2018.

VILKOMIR, S.; AMSTUTZ, B. Using Combinatorial Approaches for Testing Mobile Applications. *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, p. 78–83, 2014.

VILKOMIR, S. et al. Effectiveness of Multi-device Testing Mobile Applications. *ACM International Conference on Mobile Software Engineering and Systems Effectiveness*, v. 400, p. 13–16, 2015.

VILLANES, I. K.; ENDO, A. T.; DIAS-NETO, A. C. Using app attributes to improve mobile device selection for compatibility testing. In: *5th Brazilian Symposium on Systematic and Automated Software Testing SAST*. [S.l.: s.n.], 2020. p. 31–39.

WEI, L. et al. Understanding and detecting fragmentation-induced compatibility issues for Android apps. *IEEE Transactions on Software Engineering*, IEEE, v. 46, n. 11, p. 1176–1199, 2018.

WEI, L.; LIU, Y.; CHEUNG, S.-C. Taming Android fragmentation: characterizing and detecting compatibility issues for Android apps. *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering - ASE*, p. 226–237, 2016.

WEI, L.; LIU, Y.; CHEUNG, S.-C. PIVOT: Learning API-Device correlations to facilitate Android compatibility issue detection. In: *International Conference on Software Engineering - ICSE*. [S.l.]: IEEE, 2019. p. 878–888.

Wei, Z. et al. Test suite minimization with mutation testing-based many-objective evolutionary optimization. In: *2017 International Conference on Software Analysis, Testing and Evolution (SATE)*. [S.l.: s.n.], 2017. p. 30–36.

WOHLIN, C. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering EASE*. [S.l.: s.n.], 2014. p. 10.

WU, D. et al. Measuring the declared SDK versions and their consistency with API calls in android apps. In: *International Conference on Wireless Algorithms, Systems, and Applications WASA*. [S.l.: s.n.], 2017. v. 10251 LNCS, p. 678–690.

XIA, H. et al. How android developers handle evolution-induced api compatibility issues: A large-scale study. In: *International Conference on Software Engineering*. [S.l.: s.n.], 2020. p. 886–898.

ZEIN, S.; SALLEH, N.; GRUNDY, J. A systematic mapping study of mobile application testing techniques. *Journal of Systems and Software*, Elsevier Inc., v. 117, p. 334–356, 2016.

ZHANG, T. et al. Compatibility Testing Service for Mobile Applications. *2015 IEEE Symposium on Service-Oriented System Engineering*, p. 179–186, 2015.

ZHAO, Y. et al. Towards automatically repairing compatibility issues in published android apps. In: *Proceedings of the 44th International Conference on Software Engineering*. [S.l.]: Association for Computing Machinery, 2022. p. 2142–2153.

A

Artigos selecionados para o mapeamento

Lista de artigos usados para o mapeamento:

AB2 - Ham e Park (2011) Mobile application compatibility test system design for Android fragmentation. International Conference on Advanced Software Engineering and Its Applications (ASEA)

AB3 - Han et al. (2012) Understanding Android fragmentation with topic analysis of vendor-specific bugs. Working Conference on Reverse Engineering (WCRE)

BB3 - Huang e Gong (2012) Remote Mobile Test System: A Mobile Phone Cloud for Application Testing. IEEE International Conference on Cloud Computing Technology and Science (CloudCom)

AB4 - Huang (2014) AppACTS: Mobile app automated compatibility testing service. IEEE International Conference on Mobile Cloud Computing Services and Engineer (MOBILECLOUD)

AF5 - Khalid et al. (2014) Prioritizing the Devices to Test Your App on: A Case Study of Android Game Apps. International Symposium on Foundations of Software Engineering (FSE)

AB9 - Vilkomir e Amstutz (2014) Using combinatorial approaches for testing mobile applications. IEEE International Conference on Software Testing Verification and Validation (ICST)

AB1 - Halpern et al. (2015) Mosaic: cross-platform user-interaction record and replay for the fragmented Android ecosystem. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)

AF1 - Cheng et al. (2015) Mobile Compatibility Testing Using Multi-objective Genetic Algorithm. IEEE Symposium on Service-Oriented System Engineering (SOSE)

AB12 - Zhang et al. (2015) Compatibility Testing Service for Mobile Applications. IEEE Symposium on Service-Oriented System Engineering (SOSE)

AF10 - Lu et al. (2016) Prada Prioritizing Android Devices for Apps by Mining Large-Scale Usage Data. IEEE/ACM International Conference on Software Engineering (ICSE)

BF5 - Wei, Liu e Cheung (2016) Taming android fragmentation: Characterizing and detecting compatibility issues for android apps. International Conference on Automated Software Engineering (ASE)

AF2 - Fazzini e Orso (2017) Automated cross-platform inconsistency detection for mobile apps. International Conference on Automated Software Engineering (ASE)

AF11 - Naith e Ciravegna (2018b) Hybrid Crowd-powered Approach for Compatibility Testing of Mobile Devices and Applications. International Conference on Crowd Science and Engineering (ICCSE)

CF3 - Naith e Ciravegna (2018a) Mobile devices compatibility testing strategy via crowdsourcing. International Journal of Crowd Science

AF12 - Vilkomir (2018) Multi-device coverage testing of mobile applications. Software Quality Journal

AF3 - He et al. (2018) Understanding and detecting evolution-induced compatibility issues in Android apps. International Conference on Automated Software Engineering (ASE)

AF4 - Huang et al. (2018) Understanding and detecting callback compatibility issues for Android applications. International Conference on Automated Software Engineering (ASE)

AF7 - Li et al. (2018b) Cid: Automating the detection of api-related compatibility issues in android apps. International Symposium on Software Testing and Analysis (ISSTA)

CF2 - Li et al. (2018a) ELEGANT: Towards Effective Location of Fragmentation-Induced Compatibility Issues for Android Apps. Asia-Pacific Software Engineering Conference (APSEC)

BF4 - Wei et al. (2018) Understanding and detecting fragmentation-induced compatibility issues for android apps. IEEE Transactions on Software Engineering (TSE)

AF6 - Kowalczyk, Cohen e Memon (2018) Configurations in android testing: They matter. International Workshop on Advances in Mobile App Analysis (A-Mobile)

DF1 - Faria et al. (2019) On using collaborative economy for test cost reduction in high fragmented environments. Future Generation Computer Systems

DF2 - Liu (2019) A compatibility testing platform for android multimedia applications. Multimedia Tools and Applications

DF3 - Ki et al. (2019) Mimic: UI compatibility testing system for Android apps. IEEE/ACM International Conference on Software Engineering (ICSE)

DF8 - Scalabrino et al. (2019) Data-driven solutions to detect api compatibility issues in android: an empirical study. IEEE/ACM International Conference on Mining Software Repositories (MSR)

DF9 - Cai et al. (2019) A large-scale study of application incompatibilities in android. International Symposium on Software Testing and Analysis (ISSTA)

DF12 - Wei, Liu e Cheung (2019) Pivot: learning api-device correlations to facilitate android compatibility issue detection. IEEE/ACM International Conference on Software Engineering (ICSE)

DF5 - Menegassi e Endo (2020) Automated tests for cross-platform mobile apps in multiple configurations. IET Software

DF6 - Mobilio et al. (2020) FILO: FIx-LOcus localization for backward incompatibilities caused by Android framework upgrades. International Conference on Automated Software Engineering (ASE)

DF7 - Xia et al. (2020) How Android developers handle evolution-induced API compatibility issues: a large-scale study. IEEE/ACM International Conference on Software Engineering (ICSE)

DF10 - Mukherjee e Ruhe (2020) Analysis of Compatibility in Open Source Android Mobile Apps. International Workshop on Artificial Intelligence and Requirements Engineering (AIRE)

DF11 - Scalabrino et al. (2020) API compatibility issues in Android: Causes and effectiveness of data-driven detection techniques. IEEE/ACM International Conference on Mining Software Repositories (MSR)

DF14 - Villanes, Endo e Dias-Neto (2020) Using App Attributes to Improve Mobile Device Selection for Compatibility Testing. Symposium on Systematic and Automated Software Testing (SAST)

DF4 - Haryono et al. (2021) AndroEvolve: Automated Update for Android Deprecated-API Usages. IEEE/ACM International Conference on Software Engineering (ICSE)

DF13 - Nielebock et al. (2021) AndroidCompass: A Dataset of Android Compatibility Checks in Code Repositories. IEEE/ACM International Conference on Mining Software Repositories (MSR)

EF1 - Silva et al. (2022) SAINTDroid: Scalable, Automated Incompatibility Detection for Android. IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)

EF2 - Zhao et al. (2022) Towards Automatically Repairing Compatibility Issues in Published Android Apps. IEEE/ACM International Conference on Software Engineering (ICSE)

EF3 - Sun et al. (2022) Mining Android API Usage to Generate Unit Test Cases for Pinpointing Compatibility Issues. IEEE/ACM International Conference on Automated Software Engineering (ASE)

EF4 - Mahmud, Che e Yang (2021) Android compatibility issue detection using api differences. IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)

EF5 - Mahmud, Che e Yang (2022a) ACID: An API Compatibility Issue Detector for Android Apps. IEEE/ACM International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)

EF6 - Mahmud, Che e Yang (2022b) Android API Field Evolution and Its Induced Compatibility Issues. ACM / IEEE International Symposium on Empirical Software Engineering and Measurement

EF7 - Ren et al. (2022) Cross-Device Difference Detector for Mobile Application GUI Compatibility Testing. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)