Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Informática
Curso de Bacharelado em Sistemas de Informação

Fernando Sencioles
Giulliano Menezes dos Santos

Revisão Sistemática de Teste em Aplicações Móveis: Análise Preliminar

Trabalho de Conclusão de Curso

Curitiba
2017

Fernando Sencioles
Giulliano Menezes dos Santos

Revisão Sistemática de Teste em Aplicações Móveis: Análise Preliminar

Proposta apresentada à disciplina de Trabalho de Conclusão de Curso de Bacharelado em Sistemas de Informação, da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientadora: Profa. Dra. Maria Cláudia Figueiredo Pereira Emer

Curitiba
2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Coordenação do Curso de Bacharelado em Sistemas de Informação

TERMO DE APROVAÇÃO

"REVISÃO SISTEMÁTICA DE TESTE EM APLICAÇÕES MÓVEIS: ANÁLISE PRELIMINAR"

por

"Fernando Sencioles e

Giulliano Menezes dos Santos"

Este Trabalho de Conclusão de Curso foi apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Curitiba. Os alunos foram arguidos pelos membros da Banca de Avaliação abaixo assinados. Após deliberação a Banca de Avaliação considerou o trabalho _____.

_____ **Profa. Maria Cláudia Figueiredo Pereira Emer** (Presidente - UTFPR/Curitiba)	_____ **Profa. Marilia Abrahão Amaral** (Avaliador 1 - UTFPR/Curitiba)
_____ **Prof. Paulo Cezar Stadzisz** (Avaliador 2 - UTFPR/Curitiba)	_____ **Profa. Leyza Baldo Dorini** (Professor Responsável pelo TCC – UTFPR/Curitiba)
_____ **Prof. Leonelo Dell Anhol Almeida** (Coordenador do curso de Bacharelado em Sistemas de Informação – UTFPR/Curitiba)	

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso."

AGRADECIMENTOS

A Deus, por nossas vidas, famílias e amigos.

À Universidade Tecnológica Federal do Paraná, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbramos um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

À nossa orientadora Profa. Dra. Maria Cláudia Figueiredo Pereira Emer, pelo suporte no pouco tempo que lhe coube, pelas suas correções, incentivos e, principalmente, paciência.

À nossa amiga Marília Costa Pessanha Lara, por ter nos ajudado nas correções e revisões desse trabalho.

Aos nossos pais, pelo amor, incentivo e apoio incondicional.

A todos que direta ou indiretamente fizeram parte da nossa formação, o nosso muito obrigado.

Há mais pessoas que desistem
do que pessoas que fracassam.

(Henry Ford)

Resumo

SENCIOLES, Fernando. DOS SANTOS, Giulliano Menezes. Revisão Sistemática de Teste em Aplicações Móveis: Análise Preliminar. 85 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Aplicações móveis em *smartphones* e *tablets* vem sendo cada vez mais utilizadas tanto para acesso à internet quanto para atividades mais específicas e complexas. Considerando os prejuízos que podem acontecer se essas aplicações contivessem defeitos, a execução de testes com o propósito de detectar defeitos e aumentar o nível de confiança dessas aplicações é muito importante. Assim sendo, este trabalho pretende contribuir para o mapeamento de pesquisas da literatura que abordem o teste em aplicações móveis. Os resultados de uma análise preliminar da revisão sistemática da literatura sobre teste em aplicações voltadas a dispositivos móveis são apresentados neste trabalho. A partir de premissas definidas no protocolo de revisão, deu-se início à revisão sistemática que permitiu a enumeração dos principais desafios, lacunas de pesquisa, tipos de teste, abordagens propostas e outros fatores relacionados ao teste em aplicações móveis. Por fim, os resultados da revisão sistemática e as conclusões deste trabalho apresentam o cenário atual da pesquisa em teste de *software* no desenvolvimento de aplicações móveis.

Palavras-chave: Revisão Sistemática. Dispositivos Móveis. Teste de *Software*. Qualidade de *Software*.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ABREVIAÇÕES

ACM *Association for Computing Machinery*

GPS *Global Positioning System*

IEEE *Institute of Electrical and Electronics Engineers*

ISO *International Organization for Standardization*

NFC *Near-field Communication*

PDA *Personal Digital Assistant*

SciELO *Scientific Electronic Library Online*

SMS *Short Message Service*

UML *Unified Modeling Language*

SUMÁRIO

1 Introdução

Praticamente todos os países dependem de sistemas robustos baseados em computadores (SOMMERVILLE, 2007). Infraestruturas, serviços governamentais, eletroeletrônicos, manufatura, distribuição industrial, sistemas financeiros – todos possuem algum tipo de automatização e, por trás dessa automatização, há um *software* para controlá-la. A engenharia de *software* é um ramo da engenharia cujo foco é o desenvolvimento, dentro de custos adequados, de sistemas de *software* de alta qualidade (SOMMERVILLE, 2007).

Não existem limitações físicas no potencial de *software*; entretanto, a falta de restrições naturais significa que o *software* pode facilmente se tornar extremamente complexo (SOMMERVILLE, 2007).

A engenharia de *software* evoluiu substancialmente nas últimas décadas em busca de técnicas, métodos, critérios e instrumentos para a produção de novos *softwares*. Isso é facilmente observável pelo fato de que programas computadorizados são cada vez mais comuns e frequentes em todas as áreas do nosso cotidiano. Dada a sua relevância, o *software* produzido precisa ter qualidade tanto em seu processo de desenvolvimento quanto em relação ao produto final entregue (BARBOSA et al, 2006).

Pode-se afirmar que a engenharia de *software* é uma disciplina que utiliza os conceitos fundamentais das engenharias para desenvolver um programa de alta qualidade e baixo custo (PRESSMAN, 1997). Sem *softwares* de grande escala, não seria explorado o espaço, não existiriam a Internet e as modernas telecomunicações e todos os meios de viagem seriam mais perigosos e caros (SOMMERVILLE, 2007).

Ao comparar a elaboração de um *software* com outro processo de produção, a manufatura, pode-se perceber que nesta há uma ligação nítida entre processo e qualidade do produto final. O processo é relativamente fácil de padronizar e monitorar. Logo, basta calibrar os sistemas para que se tenham produtos produzidos de maneira uniforme. Por outro lado, o *software*

não é manufaturado, e sim projetado (PRESSMAN, 1997). O desenvolvimento de um *software* é menos mecânico do que o de um produto manufaturado, mas o planejamento é crucial para que os processos não influenciem de forma negativa a qualidade do produto. O gerenciamento da qualidade de processo e o aprimoramento do produto podem certamente conduzir a poucos defeitos no *software* entregue (SOMMERVILLE, 2007). Uma das atividades do processo de desenvolvimento de *software* que auxilia para a obtenção de um produto de *software* de alta qualidade é o teste.

O teste é um processo voltado a atingir a confiabilidade do *software*. O objetivo principal do estágio de testes é de, por meio do teste de componentes individuais do programa, descobrir defeitos. De acordo com Miller (2001), o foco do teste é assegurar a qualidade de *softwares* colocando-os sistematicamente em prática sob circunstâncias cuidadosamente controladas.

SOMMERVILLE (2007) afirma que o processo de teste de *software* tem duas metas distintas:

- Mostrar ao desenvolvedor e ao cliente que o *software* atende aos requisitos;
- Descobrir falhas ou defeitos no *software* que apresenta comportamento incorreto, não desejável ou em não conformidade com sua especificação.

No caso da primeira meta, espera-se que o sistema seja executado corretamente em um dado conjunto de casos de teste que refletem o uso esperado do sistema; dessa forma, realiza-se o teste de validação. O teste bem-sucedido é aquele em que o sistema funciona corretamente.

Já para a segunda meta, a ideia é remover anomalias como travamentos ou cálculos incorretos. Para isso, aplica-se o teste de defeitos, no qual são projetados casos de teste absurdos, não necessariamente de acordo com a utilização correta do sistema, a fim de expor seus defeitos. Ao expor um defeito que causa o funcionamento incorreto do sistema, o teste de defeitos é considerado um teste bem-sucedido.

Apesar das melhorias em métodos e técnicas de validações formais, um sistema ainda precisa ser testado antes de ser usado. Testes continuam sendo o meio verdadeiramente eficaz para assegurar a qualidade de um *software* de complexidade não-trivial (MILLER, 2001).

Considerando a alta demanda atual por *softwares*, a área de teste é uma área que atrai atenção e investimento de desenvolvedores, principalmente para serviços em que a tolerância a falhas é mínima, se não zero. A aplicação de técnicas de testes gradualmente evoluiu, a partir de um único programador para pequenas equipes de desenvolvimento, para, enfim, se consolidar numa sistemática e controlada área dentro da engenharia de *software* (SOMMERVILLE, 2007).

Segundo Miller (2001), apesar das inúmeras pesquisas envolvendo teste de *software* (cujos resultados são, em sua maioria, fundamentados), a prática de teste, muitas vezes, continua sendo considerada anormal, uma perda de tempo e de recursos; não são todas as ferramentas de teste que podem ser diretamente aplicadas a projetos sem que estes sejam amplamente modificados ou refeitos; planos de teste ainda são escritos em papel, ambientes de teste permanecem simples e a fase de teste sempre acaba sendo uma atividade corrida uma vez que, a essa altura, o projeto já está com orçamento e prazo estourados. Portanto, essa inconsistência de pesquisa e prática de teste, além de outras questões e problemas envolvidos, motiva futuras pesquisas para buscar a solução desses problemas.

A maturidade de técnicas de testes tem apresentado resultados positivos, porém, ainda não são apropriados. A pressão para produzir um *software* de alta qualidade e de baixo custo tem aumentado. Nesse contexto, o teste de *software* em aplicações móveis também vem sendo investigado em trabalhos da literatura, devido à necessidade de produzir *software* confiável, de qualidade, com esforço e custo reduzidos também para dispositivos móveis. Portanto, pesquisas são fundamentais para enfrentar esse tipo de problema e, assim, buscar por melhorias significativas nas maneiras de se testar um *software* (SOMMERVILLE, 2007).

O difundido uso de *softwares* e o aumento do custo de suas validações motivam a elaboração de parcerias entre indústria e pesquisadores para o desenvolvimento de novas técnicas e a facilitação de colocá-las em prática. Uma forma de auxiliar os pesquisadores é providenciar uma base consolidada de informações e estudos: uma revisão sistemática (MILLER, 2001).

De acordo com Linde e Willich (2003), a revisão sistemática é uma forma de pesquisa que utiliza como fonte de dados a literatura sobre determinado tema. Ela disponibiliza uma síntese das evidências relacionadas a uma estratégia de intervenção específica, mediante a aplicação de métodos explícitos e sistematizados de busca, apreciação crítica e síntese da informação selecionada.

As revisões sistemáticas são particularmente úteis para integrar as informações de um conjunto de estudos realizados separadamente, que podem apresentar resultados conflitantes e/ou coincidentes, bem como identificar temas que necessitam de evidência, auxiliando na orientação para investigações futuras (LINDE; WILLICH, 2003).

1.1 Motivação e justificativa

Atualmente, a dependência da sociedade pelos eletrônicos é impressionante. É possível ver computadores embutidos em bens das mais variadas complexidades, de relógios a aviões, de geladeiras a edifícios. Não é difícil afirmar que, basicamente, qualquer indústria é dependente de computadores para funcionar, seja ela do ramo da química, da medicina ou de telecomunicações, por exemplo.

Nas últimas duas décadas, a complexidade e o uso de sistemas de computadores expandiu. Com essa tendência em alta (e que certamente não deve mudar nos próximos anos), aumenta a preocupação em relação à qualidade e à confiabilidade desses sistemas (LYU, 1995).

Controladores de voo, por exemplo, são sistemas que não podem conter divergência alguma; são serviços que não podem ser interrompidos. Por menor que seja, um simples erro de cálculo de rota pode resultar em fatalidades. É dito que o avião é o meio de transporte mais seguro que existe, mas isso depende diretamente dos controladores e dos sistemas nele embarcados (LYU, 1995).

Softwares foram, e ainda são, a causa de diversas interrupções de serviços. O impacto que defeitos, erros de especificação ou falhas de processamento podem causar são dos mais diversos, abrangendo danos à economia (interrupções de sistemas bancários), adiamento de lançamentos de foguetes e/ou viagens espaciais (problemas na interação entre *hardware* e *software*) e até perda de vidas (mau funcionamento em *softwares* de equipamentos médicos) (LYU, 1995).

O aquecimento global e o aumento do nível do mar talvez não fossem problemas hoje se o maior buraco da camada de ozônio, situado sobre a região da Antártida, não tivesse sido classificado como irrelevante por um sistema de análise de dados, por estar fora de alcance (LYU, 1995).

Diante do cenário apresentado, pode-se observar que diversas técnicas abordadas em qualidade de *software* e em testes, por muitas vezes, são apresentadas de diferentes formas e com diferentes linguagens para aqueles que as utilizam. A demanda por *hardwares* e *softwares* complexos evoluiu mais rapidamente do que a habilidade para os projetar, implementar, testar e manter (LYU, 1995). O problema se coloca quando as poucas ferramentas existentes são subutilizadas ou mesmo ignoradas pelos desenvolvedores que, pressionados por um mercado exigente quanto a prazo e qualidade, entregam um resultado final nem sempre conforme o esperado ou otimizado.

Chernonozhkin (2001) afirma que o teste de *software* corresponde a cerca de metade do custo total do desenvolvimento do sistema. Viana (2006) diz que o custo da detecção de um erro após a entrega do produto é, no mínimo, o dobro do que se o mesmo tivesse sido detectado em tempo de desenvolvimento.

Em se tratando de aplicações móveis, atualmente, dispositivos móveis como *smartphones* e *tablets* vêm sendo cada vez mais utilizados, seja para acesso à Internet em geral, para compra de produtos ou até para atividades mais específicas como controle de atendimentos de um estabelecimento. Considerando as perdas que podem vir a ocorrer, os erros que ocorrem nestes aplicativos podem ter consequências variando de insignificantes a desastrosas, causando enormes prejuízos, dependendo da complexidade da aplicação (MULLER et al, 2007).

Esses fatores levaram à elaboração dessa proposta, considerando que o desenvolvimento de ferramentas e técnicas eficientes de testes que irão auxiliar na criação de *softwares* de alta qualidade é e deve ser vista como uma área de pesquisa de suma importância. Assim sendo, com a realização deste trabalho de pesquisa, espera-se contribuir com uma documentação sólida e embasada em pesquisas da literatura relevantes sobre as áreas de qualidade de *software,* teste de *software* e desenvolvimento de aplicações móveis.

1.2 Objetivos

O objetivo geral deste trabalho de conclusão de curso é mapear estudos da literatura relacionados ao teste de aplicações móveis.

Para que o objetivo geral seja alcançado, os objetivos específicos a seguir são propostos:

- Pesquisar conceitos básicos sobre teste de *software* e qualidade de *software*;
- Verificar quais são os tipos de teste empregados em aplicações móveis;
- Investigar abordagens de teste aplicadas para o teste de aplicações móveis, na literatura;

- Identificar lacunas de pesquisa sobre o teste em aplicações móveis.

1.3 Organização do Documento

O desenvolvimento dessa proposta foi organizado da seguinte maneira: primeiramente, o referencial teórico sobre o tema da proposta é exposto. Depois, a metodologia de realização do trabalho é apresentada. Em seguida, mostra-se o desenvolvimento do trabalho. Por fim, os resultados do trabalho são apresentados, antecedendo a conclusão e as referências.

2 Referencial Teórico

Este capítulo aborda o referencial de base para o desenvolvimento dessa proposta. Ele levanta os três principais pontos de relevância para a pesquisa, mostrando os principais conceitos sobre os temas que serão apresentados. O capítulo está distribuído da seguinte maneira: 2.1 Aplicações Móveis; 2.2 Conceitos de Qualidade; 2.3 Teste de *Software*, incluindo o teste em aplicações móveis.

2.1 Aplicações Móveis

A computação móvel teve início na década de 1980, quando a fabricação de computadores leves e pessoais permitiu os usuários levarem o equipamento de forma rápida e fácil para qualquer lugar (COULOURIS et al, 2013). A partir disso, dispositivos cada vez menores e leves foram sendo desenvolvidos. Com o surgimento de tecnologias como conexão sem fio, Wi-Fi e 3G/4G, tornou-se possível a mobilidade dos usuários, deixando-os conectados aos seus sistemas computacionais (LECHETA, 2013).

No cotidiano, é cada dia mais evidente que usuários estejam procurando celulares com funcionalidades melhores, tais como câmeras, reprodutores de músicas, conexão Bluetooth, jogos, GPS, acesso à Internet e TV digital (LECHETA, 2013). Assim, o mercado corporativo se aquece e busca incorporar aplicações móveis a seu dia-a-dia para apressurar seus negócios e integrar as aplicações móveis com seus sistemas de *back-end*. Empresas têm sempre como foco principal a busca por mais lucros e os celulares e *smartphones* ocupam um espaço no mundo em que a mobilidade é cada vez mais necessária (LECHETA, 2013).

O ambiente móvel traz novos e interessantes desafios. Até meados dos anos 2000, a mobilidade só era possível graças a dispositivos como *notebooks* e PDAs. Atualmente, celulares oferecem recursos iguais aos de um computador convencional, como sistema operacional, grande variedade de aplicativos, conexão à Internet, tudo isso junto com a possibilidade de se comunicar, ligando de forma convencional e enviando SMS (OLIVER, 2009).

Nos últimos anos, o mercado de telefonia móvel tem crescido cada vez mais. Estudos apontam que, no mundo, existem mais de 7 bilhões de aparelhos celulares, uma média de um aparelho por habitante (LECHETA, 2013).

Com o crescimento de vendas de celulares e *smartphones,* cria-se a inevitável necessidade de teste para essas aplicações. Pode-se observar que existem poucas ferramentas de injeção de falhas criadas exclusivamente para aplicações móveis (ACKER, 2010).

Segundo Lecheta (2013), a computação móvel possui três características principais:

1. Comunicação móvel: aborda os problemas de comunicação em rede de infraestrutura e características de comunicação, protocolos, formatos e tecnologias de dados concretos;

2. *Hardware* móvel: relacionado ao *hardware*, como dispositivos móveis ou os componentes destes dispositivos;

3. *Software* móvel: refere-se às características e requisitos de aplicações móveis, que são diferentes em muitos aspectos das questões das aplicações para computadores convencionais.

Kirschner (2013) nos mostra que as tecnologias envolvidas nos sistemas móveis são resultantes da evolução das áreas de comunicação sem fio e tecnologia da informação. A comunicação sem fio trata do suporte para a comunicação entre servidores e dispositivos móveis (LIMA, 2000).

Smartphones são dispositivos móveis com aplicabilidades avançadas que podem ser estendidas por meio de programas executados em seu sistema operacional. Grande parte dos sistemas operacionais desses dispositivos é *open-source* (codificação aberta), o que incentiva

desenvolvedores independentes a desenvolver aplicativos para esses sistemas operacionais (LACHETA, 2013). As principais plataformas de computação móvel atualmente são: Android (da Google), iOS (Apple), Windows (Microsoft) e BlackBerry (RIM).

2.2 Conceitos de Qualidade

"*A qualidade é relativa. O que é qualidade para uma pessoa pode ser a falta de qualidade para outra*" (WEINBERG, 1971). Essa citação de um dos maiores pesquisadores na área de qualidade de *software* incentiva a busca do entendimento de o que é qualidade e, mais importante para essa pesquisa, identificar o que é qualidade de *software*.

Koscianski e Soares (2007) apresentam um breve histórico sobre o tema de qualidade. Apesar de o controle de qualidade e a utilização de padrões como ISO serem assuntos contemporâneos e que atraíram muita atenção, com um estudo histórico pode-se observar que são na realidade desdobramentos de uma discussão muito mais antiga.

Segundo Juran e Gryna (1988), relatos antigos mostram que os egípcios montaram um sistema que gerenciava a qualidade de suas construções e, assim, criaram um padrão para medida de comprimento: o cúbito. Essa medida era referente ao comprimento do braço do faraó que estava no poder, assim, quando havia a troca de faraó, o tamanho do cúbito também era alterado e atualizado.

As construções desse tempo deviam seguir a unidade de medida do cúbito corrente. Todos os empregados dessas construções recebiam bastões referentes ao tamanho do cúbito e, a cada lua cheia, o responsável pela construção fazia a validação e comprovação de que os padrões estabelecidos estavam corretos. Caso não fossem atendidos conforme os padrões, o responsável poderia ser punido com a morte (JURAN; GRYNA, 1988). Após tantos anos, é possível observar que essa preocupação com a

qualidade das construções fez com que as pirâmides tenham obtido precisões da ordem de 0,05% (KOSCIANSKI; SOARES, 2007).

Historicamente, a qualidade evolui de inúmeras formas e com muitos exemplos fabulosos: em grandes templos construídos pelos hebreus, gregos e romanos; as navegações da Europa para a Ásia e Américas; construção de grandes catedrais nos tempos medievais – isso tudo sem recursos sofisticados ou tecnológicos. Na França, por exemplo, os construtores de catedrais utilizavam simples compassos e cordas para desenhar esses edifícios (HAGENS, 2004).

Koscianski e Soares (2007) sugerem que, com maior acesso a recursos e/ou tecnologias robustas, a precisão e a qualidade do produto final serão maiores. Tomando como exemplo o caso dos egípcios, como seriam as pirâmides se os trabalhadores tivessem medidores a *laser*?

Cada vez mais o *software* tem se tornado presente no dia-a-dia das pessoas. Ele está em dispositivos móveis como *smartphones,* dispositivos de áudio, micro-ondas, carros, máquinas fotográficas, filmadoras, entre outros (RINCON, 2009). De acordo com Reed (2000), caso alguns sistemas de uso global simplesmente deixassem de funcionar, cerca de 40% da população mundial seria afetada com isso.

À vista desses tópicos, a necessidade por maior qualidade de *software* começou a ser mais evidente quando os *softwares* passaram a se tornar cada vez mais integrados com as atividades do cotidiano das pessoas. Pressman (2011) relata que, na década de 1990, as principais empresas globais reconheciam que desperdiçavam bilhões de dólares por ano em *softwares* que não apresentavam as características e funcionalidades prometidas. E o fato se complicava tanto, que tanto os governos quanto as empresas ficavam preocupados, pois muitos desses *softwares* poderiam inutilizar importantes infraestruturas, agravando assim em dezenas de bilhões os prejuízos.

Por melhores que sejam as intenções, uma codificação malfeita continua sendo o grande problema do mercado de *software*, sendo responsável por até 45% do tempo de ociosidade e parada de sistemas

computacionais e com custo às empresas norte-americanas de US$ 100 bilhões no ano de 2010, de acordo com o Standish Group, uma empresa de pesquisas de mercado. Isso não afeta apenas as empresas que perdem bilhões de dólares corrigindo erros, afeta também clientes insatisfeitos que acabam cancelando o produto ou serviço, gerando ainda mais prejuízos às empresas (PRESSMAN, 2011).

Mas afinal, qual o prejuízo causado pela baixa qualidade de um *software*? Pressman (2011) diz que existem muitas vertentes, mas especialistas em desenvolvimento de aplicativos e novos programas afirmam que, a cada mil linhas de código, bastam três ou quatro linhas defeituosas para que seu programa não execute da maneira para qual estava programado. Além disso, acrescente que a maioria dos programadores comuns costumam inserir em média um erro a cada dez linhas de código. Multiplicando isso por milhões de linhas de códigos de inúmeros produtos, pode-se perceber que quase metade do orçamento da criação de um *software* é para a correção de erros apresentados.

2.2.1 Qualidade de Software

Desenvolvedores mais experientes concordam que qualidade de *software* é um objetivo muito importante. Mas como pode ser definida a qualidade de *software*? Bessin (2004) define da seguinte maneira: *"uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam*".

Pressman (2011) incorpora os conceitos de Bessin e define três pontos para a qualidade de *software*:

- Uma gestão de qualidade efetiva estabelece uma base sólida que dá suporte a qualquer tentativa de desenvolver um programa de alta qualidade;

- Um produto útil deve fornecer ao seu cliente todas as funcionalidades e, além disso, e não menos importante, deve fornecer confiabilidade e isenção de erros;
- Ao agregar valor tanto para o desenvolvedor quanto para o usuário, um *software* de alta qualidade gera benefícios tanto para a comunidade atendida por ele quanto para a empresa desenvolvedora.

2.2.2 Dimensões de Qualidade de Garvin

Garvin (1987) propõe que a qualidade deve ser analisada partindo da avaliação de conformidade e terminando com a visão estética do mesmo. Apesar das oito dimensões de qualidade de Garvin não terem sido originalmente criadas para *software*, elas podem ser aplicadas neste contexto. São elas:

1. Qualidade de desempenho: Inclui acessibilidade ao produto, disponibilidade e oportunidade, facilidade de uso e contato, além de interatividade e customização;
2. Qualidade dos recursos: Essa etapa consiste no algo a mais na entrega do produto;
3. Confiabilidade: Esta dimensão está ligada a probabilidade de falhas ou mal funcionamento dos produtos;
4. Conformidade: O grau de conformidade está diretamente ligado a qualidade do produto, pois todo produto e serviço têm padrões estabelecidos e quanto mais próximos a empresa produzir o modelo, menor será a chance de falhas de produto;
5. Durabilidade: É uma medida da duração da vida ou da quantidade de uso possível de um produto. É a quantidade de vezes que seu produto será utilizado antes que se desgaste;

6. Facilidade de Manutenção: Etapa que traz a qualidade e padrão de excelência no produto entregue;

7. Estética: Etapa referente aos aspectos, visuais, sonoros, degustativos ou olfativo;

8. Percepção: É a etapa onde o cliente/usuário tem a percepção sobre o produto entregue.

Pressman (2011) diz que as dimensões de Garvin fornecem uma percepção “complacente” de qualidade de *software*; muitas dessas dimensões podem ser consideradas subjetivas. Por isso, é necessário identificar características de qualidade que possam ser colocadas em dois grupos: fatores que podem ser medidos diretamente (defeitos apresentados em testes, por exemplo) e fatores que podem ser medidos apenas indiretamente (usabilidade, por exemplo).

2.2.3 Dimensões de Qualidade de McCall

McCall (1977) elaborara uma categorização dos fatores que afetavam a qualidade de *software.* Essa distribuição considera três aspectos importantes do produto de qualidade de um *software*: revisão, transição e operação do produto (Figura 1).

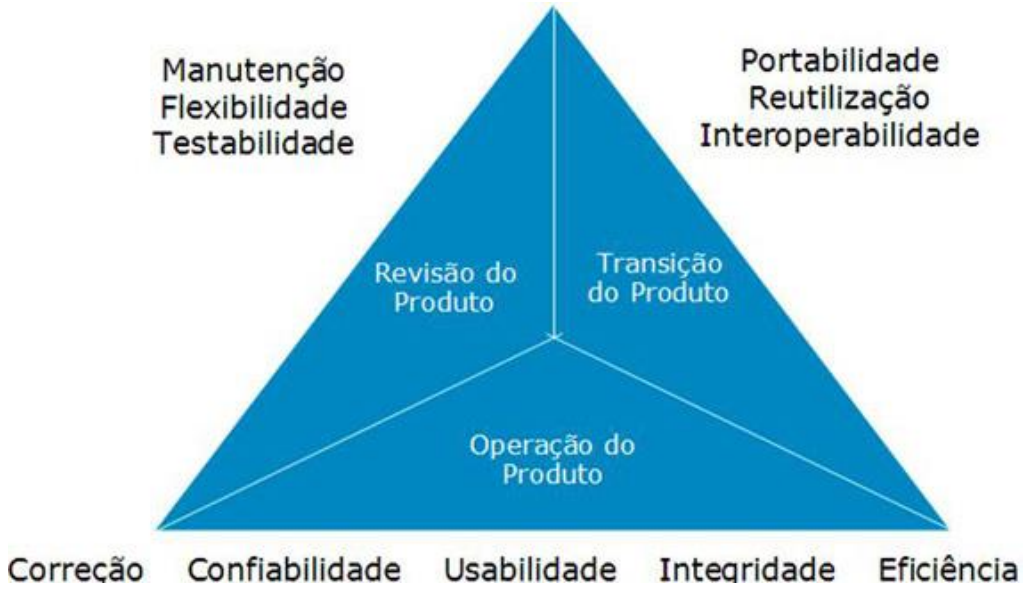


Figura 1. Fatores de qualidade de McCall. Fonte: SOMMERVILLE, 2007

A operação do produto se ramifica em:

- Correção: o quanto um programa satisfaz as especificações que o cliente solicitou;
- Confiabilidade: o quanto se pode esperar que um programa realize com precisão o que lhe foi pedido;
- Usabilidade: esforço necessário para aprender a utilizar o programa;
- Integridade: o quanto o acesso do *software* é dado a pessoas não autorizadas a controla-lo;
- Eficiência: a quantidade de recursos computacionais exigidos para que o *software* funcione.

A revisão do produto se ramifica em:

- Facilidade de manutenção: esforço necessário para localizar e corrigir um possível erro;
- Flexibilidade: esforço necessário para modificar uma funcionalidade em execução;
- Testabilidade: esforço necessário para testar um programa para que ele faça o que lhe foi proposto.

A transição do produto se ramifica em:

- Portabilidade: esforço necessário para transferir o *software* de um ambiente de *hardware* e/ou *software* para outro;
- Reutilização: o quanto o programa (ou partes do programa) pode ser reutilizado em outras aplicações;
- Interoperabilidade: o quanto de esforço é necessário para se acoplar um programa a outro.

2.2.4 Fatores de Qualidade da ISO 9126

De acordo com Sommerville (2007), o padrão da ISO 9126 foi desenvolvido para tentar levantar quais eram os principais atributos de qualidade de *software* (Quadro 1). O padrão mostrou seis atributos básicos de qualidade:

1. Funcionalidade: capacidade de um *software* prover funcionalidades que satisfaçam o usuário em suas necessidades declaradas e implícitas, dentro de um determinado contexto de uso. Sub-características são: adequação, interoperabilidade, conformidade;

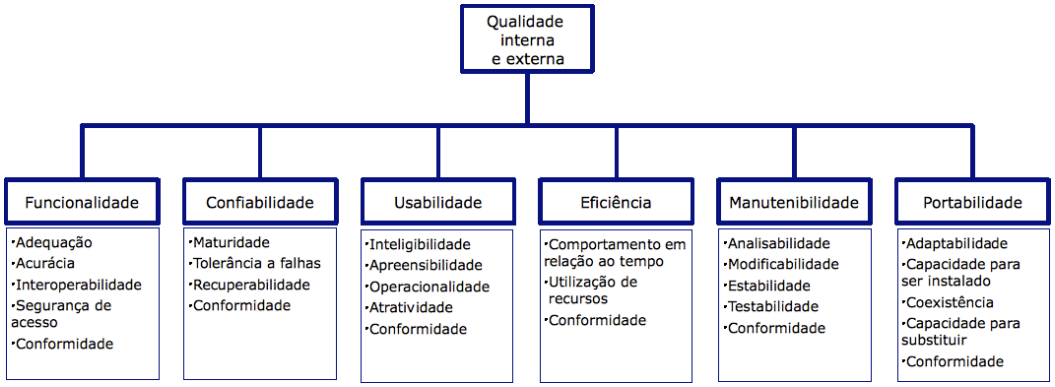
2. Confiabilidade: O produto se mantém no nível de desempenho nas condições estabelecidas. Sub-características são: maturidade, tolerância a falhas, conformidade;

3. Usabilidade: A capacidade do produto de *software* ser compreendido, seu funcionamento aprendido, ser operado e ser atraente ao usuário. Sub-características são: inteligibilidade, apreensibilidade, operacionalidade, atratividade;

4. Eficiência: O tempo de execução e os recursos envolvidos são compatíveis com o nível de desempenho do *software*. Sub-características são: comportamento em relação ao tempo, utilização de recursos;

5. Facilidade de manutenção: A capacidade (ou facilidade) do produto de *software* ser modificado, incluindo tanto as melhorias ou extensões de funcionalidade quanto as correções de defeitos, falhas ou erros. Sub-características são: analisabilidade, modificabilidade, testabilidade;

6. Portabilidade: A capacidade do sistema ser transferido de um ambiente para outro. Sub-características são: adaptabilidade, capacidade para ser instalado, coexistência.



Quadro 1. Modelo da ISO 9126. Fonte: SOMMERVILLE, 2007

Assim como em outros fatores de qualidade de *software*, o ISO 9126 não leva diretamente e nem necessariamente à medição direta (PRESSMAN, 2011), mas oferece um modelo de base bem razoável para medidas internas e uma lista de verificação para avaliar a qualidade de um sistema.

2.3 Teste de Software

Teste de *software* é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado (NETO, 2008). Delamaro (2007) define teste de *software* de uma maneira semelhante: “executar um programa ou um

modelo utilizando algumas entradas em particular e verificar se seu comportamento está de acordo com o esperado".

De acordo com Pressman (2002), a atividade de teste de *software* trata-se de um elemento crítico da garantia de qualidade de *software* e representa a revisão final da especificação, projeto e geração de código. Uma estratégia de teste de *software* deve ser flexível o bastante para promover uma interpelação de teste particularizada; deve também ser suficientemente rígida a ponto de estimular um planejamento básico e o acompanhamento à medida que o projeto avança.

Inicialmente, o teste de *software* foi definido como uma atividade à parte dos processos de desenvolvimento, que só era realizada ao final do desenvolvimento dos sistemas. Esse entendimento do processo de teste tem se mostrado bastante ineficiente, devido aos altos custos associados com a correção de erros encontrados na manutenção do *software*. A definição de métodos e técnicas sistemáticas de teste compreende um conjunto de tarefas e pode ser aplicado ao longo do processo de desenvolvimento (McGregor; Sykes, 2001).

O objetivo do teste de *software* é encontrar erros, sendo que um bom teste é aquele que tem uma alta probabilidade de achar um erro (PRESSMAN, 2011). Assim, um engenheiro de *software* deve projetar e desenvolver um programa pensando sempre na sua "testabilidade".

Sobre a testabilidade, Bach (1994) apresenta os seguintes atributos que levam um *software* a ser testável:

- Operabilidade: "*quanto melhor funcionar, mais eficientemente pode ser testado*";
- Observabilidade: "*o que você vê é o que você testa*";
- Controlabilidade: "*quanto melhor puder ser controlado o software, mais o teste pode ser automatizado e otimizado*";
- Decomponibilidade: "*controlado o escopo do teste, pode-se isolar problemas mais rapidamente e executar um reteste mais racionalmente*";

- Simplicidade: “*quanto menos tiver que testar, mais rapidamente pode-se testá-lo*”;
- Estabilidade: “*quanto menos alterações, menos interrupções no teste*”;
- Compreensibilidade: “*quanto mais informações tivermos, mais inteligente será o teste*”.

2.3.1 Conceitos Básicos de Teste de Software

Teste é um conjunto de atividades que podem ser arquitetadas com precedência e depois realizadas sistematicamente. Assim, deve-se escolher e definir um modelo (*template*) de teste que será seguido por meio do desenvolvimento do seu programa; um conjunto de passos no qual pode-se colocar técnicas específicas de projeto de caso de teste e métodos de teste (PRESSMAN, 2011).

Para se entender ainda mais sobre teste de *software*, deve se entender e distinguir a diferença entre defeitos, erros e falhas (NETO, 2008); a IEEE 610 (1990) define da seguinte maneira:

Defeito. É um ato inconsistente realizado por um personagem ao tentar entender uma certa informação, resolver um problema ou operar um método ou uma ferramenta. Como exemplo, um comando ou instrução incorreta;

Erro. É uma demonstração correta de um defeito em uma parte do *software*. É a discrepância entre o valor solicitado e o valor obtido, ou seja, qualquer resultado diferente do solicitado durante a execução de um programa;

Falha. É o comportamento operacional do *software* diferente do esperado pelo usuário. Uma falha pode ser originada por um erro e alguns erros podem nunca causar uma falha.

A Figura 2 apresenta a diferença entre os conceitos abordados pela IEEE 610. Defeitos fazem parte do universo físico (a própria aplicação, por assim dizer) e são ocasionados pelas pessoas. Os defeitos podem ocasionar o aparecimento de erros em um produto, ou seja, desenvolver algo diferente do que foi especificado no início do projeto (universo da informação). Os erros geram falhas, comportamentos inesperados de um *software* que afetam diretamente o usuário final do programa (universo do usuário) e podem fazer com que seu programa seja inviável (NETO, 2008).

Figura 2. Defeito x Erro x Falha. Fonte: NETO, 2008

A atividade de teste é composta por alguns fundamentos essenciais que assessoram a formalização dessa atividade (NETO, 2008). São os seguintes:

- Caso de Teste: descreve uma condição particular que irá ser testada e é composto por valores de entrada, restrições para a sua execução e um resultado ou comportamento esperado (CRAIG; JASKIEL, 2002);
- Procedimento de Teste: é uma especificação dos passos que são necessários para executar um caso (ou grupo de casos) de teste (CRAIG; JASKIEL, 2002);
- Critério de Teste: serve para escolher e avaliar casos de teste para que aumente as possibilidades de acontecer falhas ou, quando isso não acontecer, estabelecer um nível de confiança na correção do *software* (ROCHA et al, 2001). Os critérios de teste podem ser utilizados como:

- Critério de Cobertura dos Testes: determina o percentual de elementos necessários por um determinado critério de teste que foram feitos pelo conjunto de casos de teste (RAPPS; WEYURKER, 1982);
- Critério de Adequação de Casos de Teste: avalia se os casos de teste definidos são suficientes ou não para a avaliação (ROCHA et al, 2001);
- Critério de Geração de Casos de Teste: define regras e diretrizes para geração dos casos de teste de um produto que esteja de acordo com o critério de adequação definido anteriormente (ROCHA et al, 2001).

2.3.2 Defeitos no Desenvolvimento de Software

Durante o processo de desenvolvimento de um programa, todos os defeitos que ocorrem são humanos. Por mais que se utilizem de várias técnicas, métodos e melhores práticas de desenvolvimento, ferramentas ou profissionais, erros continuam presentes nos *softwares*, o que torna a atividade de teste fundamental durante o processo de desenvolvimento do mesmo (NETO, 2008).

O número de pessoas envolvidas e tamanho do projeto são dois fatores que elevam a dificuldade dessa tarefa, aumentando a probabilidade de erros. Neto (2008) apresenta alguns motivos que podem gerar defeitos nos programas:

- A especificação pode estar incompleta ou errada;
- A especificação pode conter requisitos que não podem ser realizados por limitações de *software* e/ou *hardware*;
- O modo que a base de dados está organizada pode impedir a distinção dos tipos de usuário;

- O algoritmo de controle dos usuários pode conter um defeito.

Os defeitos geralmente são inseridos durante as várias etapas de transformação das informações do cliente no produto final que será desenvolvido. Partindo do exemplo de um ciclo simples de desenvolvimento de um *software*, os requisitos pedidos pelo cliente são colocados de forma textual e simples em um documento de especificação de requisitos. Esse documento é transformado em casos de uso e a sua arquitetura é descrita utilizando diagramas de classes em UML. Esses modelos de projetos são posteriormente utilizados para a construção e o desenvolvimento do *software*. Durante todo esse ciclo, defeitos podem ter sido introduzidos (NETO, 2008). A Figura 3 apresenta a metáfora ostentada nesse parágrafo.

Figura 3. Diferentes interpretações ao longo do ciclo de desenvolvimento de um *software*. Fonte: NETO, 2008

Essas transformações fizeram com que a realização de testes em diferentes níveis fosse necessária, buscando assim avaliar o *software* de diferentes visões e perspectivas de acordo com o produto gerado em cada fase do ciclo de vida de desenvolvimento dele. Isso será apresentado na Seção 2.3.3.

2.3.3 Níveis de Teste de Software

O planejamento dos testes deve ocorrer em diferentes níveis e em paralelo durante o desenvolvimento do *software* (Figura 4). Assim, Neto (2008), embasado no livro “Qualidade de *Software* – Teoria e Prática” (ROCHA et al, 2001), define que os principais níveis de teste de *software* são:

- Teste de Unidade: conhecido também como testes unitários, tem como objetivo explorar a menor unidade do projeto, buscando explicitar falhas por defeitos de lógica ou de implementação em cada parte, separadamente;
- Teste de Integração: busca ocasionar falhas associadas às interfaces entre os módulos quando esses são integrados, para construir a estrutura do *software* que foi estabelecida na fase de projeto;
- Teste de Sistema: realiza uma avaliação do *software* em busca de falhas por meio de sua utilização, como se esse fosse o próprio usuário final. Assim, os testes são feitos nos mesmos ambientes, com as mesmas condições e dados de entrada que o usuário utilizaria cotidianamente; logo, vê-se se o programa atende os requisitos designados;
- Teste de Aceitação: realizados por um grupo de usuários que serão os usuários finais do sistema. Simulam operações de rotina do sistema com o objetivo de verificar o seu desempenho e comportamento, observando se está realizando o que lhe foi solicitado de acordo com suas especificações;
- Teste de Regressão: consiste em se aplicar, a cada nova versão do *software*, todos os testes que já foram aplicados nas versões anteriores do sistema.

Neto (2008) indica, por meio da Figura 4, que o planejamento e projeto dos testes devem ocorrer de cima para baixo, isto é:

1. Primeiramente, é planejado o teste de aceitação com base no documento de requisitos do projeto;

2. Em seguida, há o planejamento do teste de sistema a partir do projeto de alto nível do *software*;

3. Logo após, ocorre o planejamento dos testes de integração a partir do projeto em nível menor;

4. Finamente, o planejamento dos testes a partir do código-fonte.

Já a execução ocorre no sentido inverso ao descrito acima, como esquematizado na Figura 4.

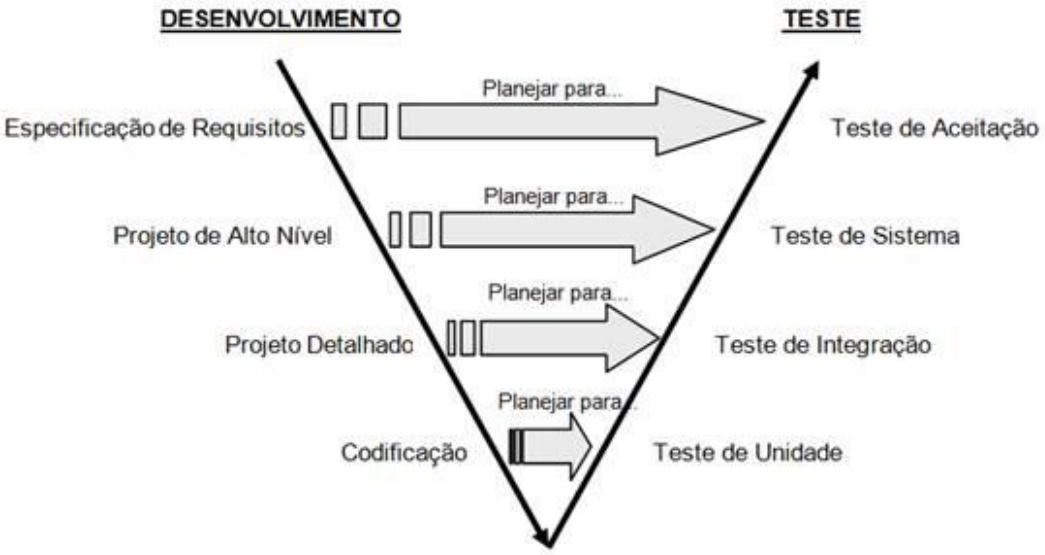


Figura 4. Modelo V, descrevendo o paralelismo entre as atividades de desenvolvimento e teste de *software*. Fonte: CRAIG; JASKIE, 2002

2.3.4 Teste de Aplicações Móveis

De acordo com uma pesquisa realizada pela SmartBear Software (2014), empresa responsável pelo desenvolvimento de *softwares* com foco em automação de testes, os principais desafios para os profissionais da área de teste de aplicações móveis são:

1. Fragmentação de dispositivos: diversos fabricantes colocam à venda diversos dispositivos contendo diversas modificações, criando variações no sistema operacional, na visualização e na experiência do usuário;

2. Consumo de dados: evitar que a aplicação envie e/ou receba um grande fluxo de dados, o que pode decepcionar usuários que não possuem um plano de dados eficiente com sua operadora;

3. Poder de processamento e bateria: além de jogos, gerenciamento de memória e renderização de imagens são os principais culpados pelo alto processamento e, consequentemente, baixo tempo de vida da bateria;

4. Sensores: GPS, acelerômetro, giroscópio e outros podem continuar ativados se não houver esse tipo de validação no código-fonte, o que pode resultar em menor tempo de vida da bateria;

5. *Touchscreen* e interface: aplicativos podem não ser intuitivos para todos; gestos simples como um toque com dois dedos na tela podem não funcionar corretamente.

É seguro afirmar que, quando se trata de aplicativos móveis, os usuários têm pouca (ou quase nenhuma) paciência para desempenho lento; de acordo com Rapoza (2012), 25% dos usuários abandonam um aplicativo móvel depois de três segundos de atraso.

Testar aplicações móveis é mais complexo e consome mais tempo em relação a aplicações tradicionais de *desktop* e *web*. Diferentemente dos computadores de mesa, os dispositivos móveis não possuem uma plataforma “predominante” como o Microsoft Windows (PRADHAN, 2011).

Além do grande problema já citado anteriormente da fragmentação de dispositivos móveis, há também a fragmentação de sistemas operacionais e suas versões. Com múltiplas versões, independentemente destas conterem grandes ou pequenas atualizações, equipes de teste são continuamente acionadas para testar novas funcionalidades ou para certificar de que a versão

atual da aplicação funciona adequadamente com a versão nova do sistema operacional (PRADHAN, 2011).

De acordo com Pradhan (2011), há oito dimensões de teste de aplicações móveis, cada uma contendo um tipo específico de teste, que varia de acordo com vários fatores, como o tipo da aplicação, o público-alvo e o canal de distribuição. São elas:

1. Funcionalidade: testes de interação com usuário e de transação;
2. Desempenho: tempo de resposta da interface, tempo de execução da transação, picos de atividades, longevidade;
3. Rede: tipo de rede (Wi-Fi, 2G, 3G, 4G), impacto caso houver problemas de conectividade;
4. Segurança: retenção de dados no dispositivo, transmissão de dados;
5. Compatibilidade: entre plataformas (mesmo sistema operacional, versões diferentes), entre dispositivos (mesma marca, modelos diferentes), retrocompatibilidade (com versão anterior do aplicativo);
6. Conformidade: seguir os códigos de conduta dos distribuidores (políticas) ou da empresa (conteúdo proibido);
7. Usabilidade: experiência do usuário;
8. Instalação e aprovisionamento: processos de instalação/desinstalação e de aprovisionamento/desaprovisionamento.

De acordo com Gao et al (2013), há quatro abordagens de teste para aplicações para dispositivos móveis:

1. Teste baseado em emulador: uma máquina virtual do dispositivo móvel no computador é usada para representar o dispositivo em questão; tendo em vista que não há necessidade de um laboratório ou dispositivos físicos para testes, é uma abordagem sem custos expressivos, porém, possui limitações como validar gestos e funções específicas de um dispositivo;
2. Teste baseado em dispositivo: utiliza-se de um laboratório e dispositivos físicos, sendo capaz de validar inclusive a rede utilizada pelos dispositivos para conectividade, além de funções mais específicas;

3. Teste na nuvem: a ideia é construir uma nuvem de dispositivos móveis capaz de apoiar serviços de teste em larga escala, como por exemplo o modelo de serviços pré-pagos;

4. Teste baseado em público: envolve trabalho *freelance* ou uma comunidade de usuários para relatarem falhas e controlarem seus testes, sem necessidade de investir num laboratório ou comprar ou alugar dispositivos móveis, porém, à mercê de testes de baixa qualidade e de um cronograma incerto.

3 Metodologia

Para essa pesquisa, a metodologia de pesquisa científica se apropriou das seguintes classificações de tipo de pesquisa: natureza, forma de abordagem, objetivos e procedimentos (GIL, 2002; LAKATOS; MARCONI, 1991).

Quanto à natureza da pesquisa, essa foi uma pesquisa aplicada. A pesquisa aplicada objetiva gerar conhecimentos para aplicações práticas dirigidas à solução de problemas específicos (GIL, 1994); neste contexto, problemas específicos no âmbito da engenharia de *software*.

Do ponto de vista da forma de abordagem ao problema, foi qualitativa e, em relação aos objetivos, foi exploratória, pois proporcionou maior familiaridade com o problema, envolveu levantamento bibliográfico e, de maneira geral, assumiu a forma de uma pesquisa bibliográfica.

Quanto aos procedimentos técnicos, esta pesquisa envolveu: (i) a pesquisa bibliográfica, elaborada a partir de materiais já publicados, como periódicos e artigos (revisão sistemática), e (ii) a pesquisa *ex-post-facto* (literalmente, “a partir do fato passado”), sendo o experimento realizado depois dos fatos apresentados (consolidação da pesquisa realizada com a proposição de uma estratégia de teste para aplicações móveis).

Em 1968, numa palestra para o Prêmio Turing (concedido por contribuições duradouras e fundamentais no campo computacional), Richard Hamming, matemático premiado por seu trabalho em métodos numéricos, sistemas de códigos automáticos e detectores e corretores de erros de código, disse:

> Uma das maiores reclamações sobre a área da computação é que, enquanto Newton diria ‘se consegui ver mais longe foi por estar de pé sobre ombros de gigantes’, sou forçado a dizer que, hoje, estamos de pé sobre os pés dos outros. O principal problema que encontramos em toda a ciência da computação talvez seja como chegamos onde chegamos atravessando o trabalho dos outros, em vez de refazer isso tudo trivialmente de maneira diferente. A ciência

> deveria ser cumulativa, e não repetições quase infinitas de coisas iguais.

Em todas as disciplinas, a revisão sistemática permite ficar de “pé sobre ombros de gigantes”; na computação, nos permite sair de cima dos pés dos outros (KITCHENHAM; CHARTERS, 2007).

A revisão sistemática da literatura é uma forma de avaliar e interpretar as pesquisas disponíveis e relevantes de uma específica questão, área ou um fenômeno de interesse (KITCHENHAM; CHARTERS, 2007). Revisões sistemáticas apresentam conjectura de um tópico de pesquisa utilizando uma metodologia confiável, rigorosa e passível de validação.

A necessidade de se realizar uma revisão sistemática surge a partir da exigência dos pesquisadores de sintetizar todas as informações existentes de um determinado fenômeno, de maneira rigorosa e imparcial. Consequentemente, isso pode gerar novas conclusões gerais do que seria possível com estudos isolados ou pode ser usado como prelúdio para pesquisas futuras.

Existem diversas diretrizes e protocolos para se fundamentar uma revisão sistemática, como o PRISMA-P: *Preferred Reporting Items for Systematic review and Meta-Analysis Protocols* (lit. “Itens Reportáveis Preferíveis para Protocolos de Revisão Sistemática e Metanálise”), ou o QUOROM: *Quality of Reporting of Meta-Analyses* (lit. “Qualidade de Elaboração de Relatórios de Metanálise”). Ainda pode-se citar a recomendação da Colaboração Cochrane, que sugere que a revisão sistemática seja efetuada em sete passos (HIGGINS; GREEN, 2011). São eles:

1. Formulação da pergunta: objetivos, questões de pesquisa que a revisão sistemática deverá responder;

2. Localização dos estudos: desenvolvimento da estratégia de busca, pesquisa por artigos ou estudos nas bases selecionadas;

3. Avaliação crítica dos estudos: quantificação e qualificação (seleção) dos artigos selecionados;

4. Coleta de dados: definição das características do estudo, cruzamento de estudos selecionados contra critérios de inclusão/exclusão para a análise de dados;

5. Análise e apresentação dos dados: planejamento da análise, sensibilidade e problemas da pesquisa;

6. Interpretação dos dados: força da evidência buscada e documentada, aplicabilidade da revisão;

7. Aprimoramento e atualização da revisão: emprego de novas abordagens ou técnicas, atualização de dados baseada em novas buscas (caso julgar-se necessária a mudança do protocolo de revisão).

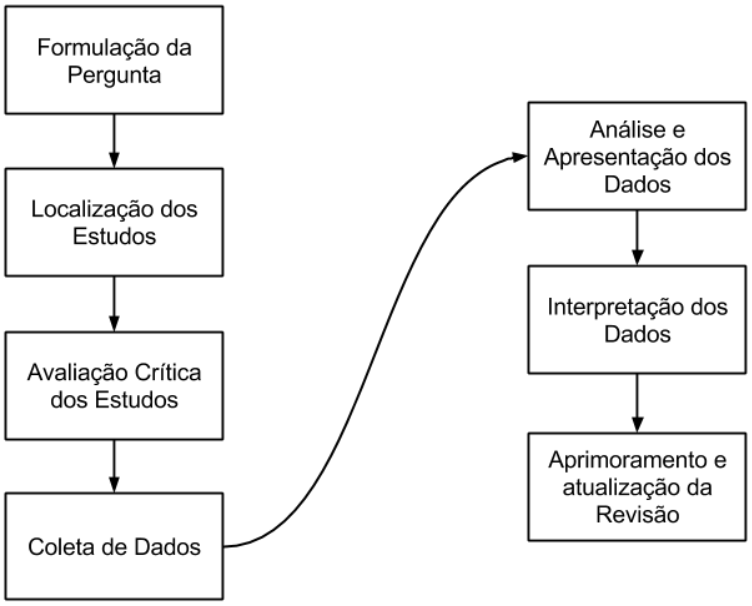


Figura 5. Sete passos da revisão sistemática. Fonte: Adaptado de Higgins e Green (2011).

Na Figura 6, são apresentados os passos executados para a realização deste trabalho de pesquisa e posteriormente estes passos são brevemente descritos.

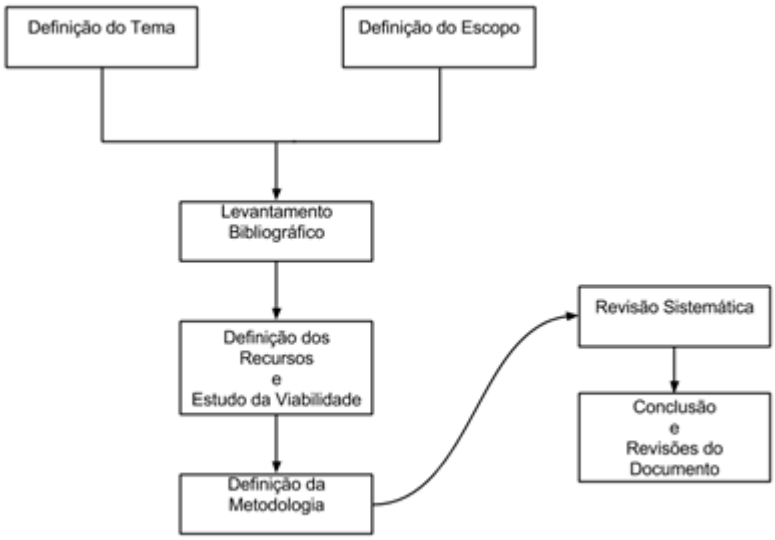


Figura 6. Etapas de desenvolvimento do trabalho. Fonte: Autoria Própria

Definição do Tema e Definição do Escopo. O projeto teve seu início com a definição do tema de pesquisa e a definição do seu escopo, tentando entender e qualificar os objetivos que seriam alcançados. A definição do escopo propôs, além da definição do tema e objetivos, a definição do orientador e a busca por materiais de apoio para embasar a proposta e comprovar a viabilidade do escopo.

Levantamento Bibliográfico. O levantamento bibliográfico abrange a pesquisa da literatura, elaborada a partir de materiais publicados. Neste contexto, foram utilizadas palavras-chave como teste de *software*, teste de aplicações móveis, revisão sistemática, entre outras; elas foram utilizadas em mecanismos de busca para que documentos e artigos relevantes para o trabalho fossem encontrados de maneira mais assertiva.

Definição de Recursos e Estudo de Viabilidade. A definição de recursos levantará o que será necessário com relação a *software, hardware* e recursos humanos para o desenvolvimento dessa proposta de pesquisa. O estudo de viabilidade analisa os recursos que serão utilizados para a execução, o tempo requerido e a capacidade técnica dos recursos que serão utilizados, mostrando se é possível ou não realizar a pesquisa.

Definição da Metodologia. Na fase de definição da metodologia, é analisada a intenção da pesquisa, as formas de abordagem e o modelo seguido.

Revisão Sistemática. A revisão sistemática é descrita de maneira mais específica no Apêndice A – Conceitos de Revisão Sistemática.

A conclusão e as revisões finais do documento compõem o bloco de fases finais do desenvolvimento deste trabalho, realizadas após o desenvolvimento da revisão sistemática.

Neste trabalho, dada a exigência de controle e registro de atividades do processo de revisão sistemática, um planejamento para condução e distribuição de tarefas entre os autores foi criado; dessa forma, estes puderam seguir um roteiro de pesquisa em comum, garantindo imparcialidade e criteriosidade. A consolidação da revisão sistemática foi planejada pelos autores nas três etapas seguintes.

A primeira etapa fundamenta a premissa ou protocolo de revisão dos autores. Além de definir critérios de inclusão e exclusão e listar as bases de pesquisa, os autores definem as perguntas ou questões de pesquisa.

A segunda etapa engloba a condução da pesquisa, categorização e registro de estudos que potencialmente contêm informações que podem referenciar e evidenciar o tópico abordado. Trata-se da etapa mais extensa da revisão sistemática, mostrando-se necessária organização e alinhamento entre os autores para que não haja resultados incoerentes ou tendenciosos.

A terceira etapa envolve análise dos resultados obtidos após a aplicação dos critérios de pesquisa e o registro de evidências da etapa anterior (KITCHENHAM; CHARTERS, 2007). É a etapa responsável por representar e apresentar as primeiras conclusões da revisão sistemática (e da pesquisa em si).

Conclusão e Revisões do Documento. A conclusão e a revisão final do documento compõem o bloco de fases finais do desenvolvimento deste trabalho.

4 Desenvolvimento da Revisão Sistemática

Neste capítulo são apresentados o protocolo que será seguido para a realização da revisão e os resultados obtidos com a condução da revisão.

4.1 Protocolo de Revisão e Estratégia de Busca

O protocolo (ou planejamento) de uma revisão sistemática especifica as principais perguntas de pesquisa e metodologia aplicada para a condução da revisão (BIOLCHINI et al, 2005).

A estratégia de busca para seleção de estudos desta revisão sistemática inclui a seleção de fontes ou bases de pesquisa, o emprego de filtros de pesquisa e consolidação da lista de palavras-chave para buscas nas fontes de pesquisa.

4.1.1 Objetivos

O principal objetivo dessa revisão sistemática é mapear os estudos relacionados a testes de aplicações móveis para identificar a presença (ou ausência) de tipos ou metodologias de testes de *software* empregadas durante o desenvolvimento dessas aplicações.

4.1.2 Questões de Pesquisa

A partir da seleção de trabalhos ou estudos, as seguintes questões de pesquisa devem ser respondidas:

QP1. Quais são os desafios para a realização de teste em aplicações móveis?

QP2. Quais são os tipos de teste utilizados em aplicações móveis?

QP3. Existem lacunas de pesquisa na área de teste em aplicações móveis?

QP4. Quais são os grupos de pesquisa e/ou autores que trabalham com pesquisa em teste de aplicações móveis?

QP5. Abordagens de teste para aplicações móveis têm sido propostas?

4.1.3 Palavras-chave

Para que trabalhos relacionados ao tema e às perguntas de pesquisa desta revisão sistemática fossem encontrados pelas bases de pesquisa, mostrou-se necessária a construção de uma lista de palavras-chave, sinônimas ou relativas aos principais termos da pesquisa, como “teste”, “programa” e “dispositivo móvel”.

Da mesma forma, foram incluídas palavras-chave que remetem à garantia de qualidade do *software*, como “erro” e “defeito”.

Como a revisão sistemática inclui trabalhos da língua inglesa, também foi necessária a tradução (e adaptação) das palavras-chave, previamente construídas em português.

Palavra-chave (Português)	Palavra-chave (Inglês)
Android	Android
Celular	Smartphone
Defeito	Defect
Desenvolvimento	Development
Dispositivo Móvel	Mobile
Erro	Error
Falha	Flaw
iOS	iOS
Programa	Software
Qualidade	Quality
Tablet	Tablet
Teste	Test
Windows Phone	Windows Phone

Tabela 1. Lista de palavras-chave em Português e Inglês.

4.1.4 Strings de Busca

A partir das duas listas, a construção dos termos de pesquisa foi baseada no cruzamento de palavras-chave relacionadas a um determinado propósito ou escopo de busca.

Alguns desses termos têm como propósito buscar estudos de forma mais abrangente para que sejam consideradas estratégias de testes, propostas de qualidade, identificação de erros ou defeitos de *software* e outros critérios gerais que também possam ser aplicáveis em dispositivos móveis.

Esses termos incluem máscaras (*) para incluir variações (de gênero e de quantidade) e operadores lógicos para restringir (*AND*) ou ampliar (*OR*) o escopo.

Alguns mecanismos de busca consideram o espaço entre palavras-chave como um operador lógico. Para palavras compostas como “dispositivo móvel”, o uso de aspas duplas (") foi necessário para evitar que os mecanismos não as interpretasse de forma errônea.

Termo (Português)	Termo (Inglês)	Propósito
Test* AND Programa* AND Celular*	Test* AND Software* AND Smartphone*	Testes de programas de celulares (geral)
Test* AND Programa* AND "Dispositivo* Move*"	Test* AND Software* AND Mobile	Testes de programas de dispositivos móveis (geral)
Test* AND Programa* AND Android	Test* AND Software* AND Android	Testes de programas de plataformas Android (Google)
Test* AND Programa* AND iOS	Test* AND Software* AND iOS	Testes de programas de plataformas iOS (Apple)
Test* AND Programa* AND Windows	Test* AND Software* AND Windows	Testes de programas de plataformas Windows (Microsoft)
Test* AND Qualidade AND Programa*	Test* AND Quality AND Software*	Testes e qualidade de programas (geral)
Qualidade AND Programa* AND Falh*	Quality AND Software* AND Flaw*	Qualidade e falha(s) de programas (geral)
Qualidade AND Programa* AND Err*	Quality AND Software* AND Error*	Qualidade e erro(s) de programas (geral)
Qualidade AND Programa* AND Defeit*	Quality AND Software* AND Defect*	Qualidade e defeito(s) de programas (geral)

Tabela 2. Lista de termos de pesquisa em Português e Inglês.

4.1.5 Método de Busca de Fontes

Os estudos foram pesquisados a partir de fontes de dados eletrônicas indexadas, que oferecem detalhes como citações, resumos e até o conteúdo completo de pesquisas, artigos e periódicos científicos publicados.

4.1.6 Bases de Pesquisa

Para o desenvolvimento desta revisão sistemática, foram selecionadas as seguintes bases:

- ACM: https://dl.acm.org
- IEEE: http://ieeexplore.ieee.org

- SciELO: http://www.scielo.org
- ScienceDirect: http://www.sciencedirect.com

4.1.7 Idioma dos Artigos

Os artigos considerados devem estar escritos em inglês ou português.

4.1.8 Critérios de Inclusão

Para esta revisão sistemática, foram considerados trabalhos que atendem aos seguintes critérios:

- Atendem aos objetivos específicos deste trabalho;
- Estar escrito em inglês ou português;
- Respondem às questões de pesquisa;
- Publicados dentro do período de pesquisa (2010 a 2016).

4.1.9 Critérios de Exclusão

A fim de evitar a seleção de estudos irrelevantes para a revisão sistemática, foram empregados os seguintes critérios de exclusão:

- Artigos duplicados;
- Artigos considerados irrelevantes para a pesquisa pela leitura do título;

- Artigos considerados irrelevantes para a pesquisa pela leitura do resumo;
- Artigos considerados irrelevantes para a pesquisa após a leitura completa;
- Artigos que não são acessíveis gratuitamente.

4.2 Condução da Revisão Sistemática

Com os termos de busca consolidados, deu-se início às buscas propriamente ditas. Os resultados foram registrados de forma distinta, isto é, separados por base de pesquisa.

Como o mecanismo de busca de cada base funciona de maneira diferente, alguns termos foram adaptados de acordo com o funcionamento de cada uma das bibliotecas para que estas retornassem resultados relevantes e esperados nas buscas.

Inevitavelmente, ocorreram replicações de trabalhos recuperados por diferentes termos de pesquisa em diferentes bases. Essas repetições foram descartadas durante cada consolidação de resultados das buscas.

Os trabalhos foram selecionados pela leitura do título, das palavras-chave e do resumo. Os autores realizaram as pesquisas em conjunto para decidirem, sob unanimidade, quais trabalhos seriam enfim selecionados para análise. Todos os acessos, pesquisas e registros de buscas foram realizados entre os meses de setembro e outubro de 2017.

Nas próximas seções, são apresentados em mais detalhes os resultados de cada base de pesquisa, bem como a lista de termos de pesquisa aplicada.

4.2.1 ACM

O motor de buscas da base da ACM não suporta o uso de operadores lógicos (AND e OR), portanto, todos os termos de pesquisa precisaram ser ajustados para essa base.

O termo “Test* AND Programa* AND "Dispositivo* Move*"” precisou ser especificamente modificado para abranger a palavra-chave “Dispositivo Móvel” e seu plural, visto que a base não suporta máscaras (asterisco).

O único filtro disponível e relevante à redução do escopo de busca desta revisão sistemática é o de período de publicação dos trabalhos.

Um total de 14 trabalhos únicos foram selecionados a partir das buscas, cujos detalhes estão apresentados na Tabela 3.

Os trabalhos não selecionados se enquadram em pelo menos um dos seguintes critérios:

- Não possuem relação com abordagens de testes;
- Não possuem relação com plataformas ou dispositivos móveis.

Palavras-chave	Idioma	Resultados (sem filtros)	Resultados (com filtros)	Selecionados
+(Test* +(Programa* +(Celular*)))	Português	10	9	0
+(Test* +(Programa* +("Dispositivo Movel" "Dispositivos Moveis")))	Português	15	10	2
+(Test* +(Programa* +(Android)))	Português	2.570	2.528	4
+(Test* +(Programa* +(iOS)))	Português	4.197	2.248	0
+(Test* +(Programa* +(Windows)))	Português	4.958	2.164	0
+(Test* +(Qualidade +(Programa*)))	Português	92	56	0
+(Qualidade +(Programa* +(Falh*)))	Português	3	3	0
+(Qualidade +(Programa* +(Err*)))	Português	20.944	10.696	0
+(Qualidade +(Programa* +(Defeit*)))	Português	0	0	0
+(Test* +(Software* +(Smartphone*)))	Inglês	191	190	0
+(Test* +(Software* +(Mobile)))	Inglês	41.666	27.034	3
+(Test* +(Software* +(Android)))	Inglês	2.570	2.528	0
+(Test* +(Software* +(iOS)))	Inglês	4.197	2.248	0
+(Test* +(Software* +(Windows)))	Inglês	4.958	2.164	0
+(Test* +(Quality +(Software*)))	Inglês	20.122	10.904	3
+(Quality +(Software* +(Flaw*)))	Inglês	968	493	1
+(Quality +(Software* +(Error*)))	Inglês	20.080	10.260	0
+(Quality +(Software* +(Defect*)))	Inglês	7.195	3.476	1
Total		**134.736**	**77.011**	**14**

Tabela 3. Resultados das buscas por termos de pesquisa na base ACM.

4.2.2 IEEE

Na base do IEEE, não houve necessidade em modificar os termos de pesquisa antes das buscas e do registro de resultados.

Como a base não possui filtros de categoria ou de idioma disponíveis, o único filtro aplicado para as pesquisas resumiu-se ao período de publicação dos trabalhos. A exclusão de trabalhos estrangeiros (que não em Inglês ou Português) foi realizada de forma manual.

Foram selecionados 8 trabalhos únicos a partir das buscas; os resultados estão detalhados na Tabela 4.

Os trabalhos não selecionados se enquadram em pelo menos um dos seguintes critérios:

- Trabalhos publicados em Espanhol;
- Não possuem relação com abordagens de testes;
- Não possuem relação com plataformas ou dispositivos móveis.

Palavras-chave	Idioma	Resultados (sem filtros)	Resultados (com filtros)	Selecionados
Test* AND Programa* AND Celular*	Português	36	31	0
Test* AND Programa* AND "Dispositivo* Move*"	Português	13	13	0
Test* AND Programa* AND Android	Português	102	101	0
Test* AND Programa* AND iOS	Português	64	54	0
Test* AND Programa* AND Windows	Português	490	351	0
Test* AND Qualidade AND Programa*	Português	235	213	0
Qualidade AND Programa* AND Falh*	Português	1	1	0
Qualidade AND Programa* AND Err*	Português	37	35	0
Qualidade AND Programa* AND Defeit*	Português	0	0	0
Test* AND Software* AND Smartphone*	Inglês	323	299	1
Test* AND Software* AND Mobile	Inglês	4.635	2.658	4
Test* AND Software* AND Android	Inglês	608	603	3
Test* AND Software* AND iOS	Inglês	58	48	0
Test* AND Software* AND Windows	Inglês	1.123	463	0
Test* AND Quality AND Software*	Inglês	10.438	4.882	0
Quality AND Software* AND Flaw*	Inglês	173	83	0
Quality AND Software* AND Error*	Inglês	2.746	1.263	0
Quality AND Software* AND Defect*	Inglês	1.459	735	0
Total		**22.541**	**11.833**	**8**

Tabela 4. Resultados das buscas por termos de pesquisa na base IEEE.

4.2.3 SciELO

Na SciELO, foi necessário modificar um dos termos de pesquisa antes do registro de resultados.

O mecanismo de busca identificava o asterisco como parte do texto (e não como máscara), logo, não encontrava resultados. A remoção das aspas do termo "*Test* AND Programa* AND "Dispositivo* Move*"*" foi o suficiente para que a base de pesquisa retornasse resultados desejados.

Em seguida, para facilitar e agilizar o registro de buscas com filtro de data de publicação (a partir de 2010), foi adicionada a seguinte condição a cada um dos termos de pesquisa: "*AND year_cluster:("2017" OR "2016" OR "2015" OR "2014" OR "2013" OR "2012" OR "2011" OR "2010")*".

Finalmente, foi aplicado o filtro de idioma, com o qual foi possível focar na seleção de trabalhos das línguas portuguesa e inglesa.

Para esta base de pesquisa, não foi aplicado filtro de categoria da publicação. Alguns trabalhos relacionados ao desenvolvimento de *software* estavam categorizados somente para área da saúde, por exemplo, o que automaticamente os removia do escopo de seleção. Assim, optou-se por não levar em consideração a categoria dos trabalhos.

Um total de 3 trabalhos únicos foram selecionados por meio das buscas na base SciELO, como detalha a Tabela 5.

Os trabalhos não selecionados se enquadram em pelo menos um dos seguintes critérios:

- Não possuem relação com a área da computação;
- Não possuem relação com plataformas ou dispositivos móveis;
- Trabalhos publicados apenas em Espanhol.

Palavra-chave	Idioma	Resultado (sem filtros)	Resultado (com filtros)	Selecionados
Test* AND Programa* AND Celular*	Português	65	22	0
Test* AND Programa* AND Dispositivo* Move*	Português	8	5	1
Test* AND Programa* AND Android	Português	4	4	1
Test* AND Programa* AND iOS	Português	0	0	0
Test* AND Programa* AND Windows	Português	90	33	0
Test* AND Qualidade AND Programa*	Português	584	340	0
Qualidade AND Programa* AND Falh*	Português	41	23	0
Qualidade AND Programa* AND Err*	Português	69	37	0
Qualidade AND Programa* AND Defeit*	Português	15	10	0
Test* AND Software* AND Smartphone*	Inglês	2	2	0
Test* AND Software* AND Mobile	Inglês	29	12	1
Test* AND Software* AND Android	Inglês	4	3	0
Test* AND Software* AND iOS	Inglês	1	1	0
Test* AND Software* AND Windows	Inglês	65	24	0
Test* AND Quality AND Software*	Inglês	250	119	0
Quality AND Software* AND Flaw*	Inglês	3	3	0
Quality AND Software* AND Error*	Inglês	55	22	0
Quality AND Software* AND Defect*	Inglês	14	7	0
Total		**1.299**	**667**	**3**

Tabela 5. Resultados das buscas por termos de pesquisa na base SciELO.

4.2.4 ScienceDirect

Após a leitura da documentação do mecanismo de pesquisa da base ScienceDirect, concluiu-se que não seria necessário modificar, nem adaptar os termos de pesquisa.

O mecanismo de busca retornou muitos falsos-positivos a partir da busca no conteúdo de cada trabalho (principalmente nas referências). Além disso, a base contém muitos trabalhos da área médica.

A fim de concentrar os estudos em trabalhos relevantes à revisão sistemática e, ao mesmo tempo, reduzir o escopo de avaliação da pesquisa, foram aplicados os seguintes filtros além do filtro de data de publicação do artigo:

- Busca pelos termos de pesquisa no resumo, título ou palavras-chave (*Abstract, Title, Keywords*) dos artigos;
- Busca por trabalhos dentro da categoria de ciência da computação (*Computer Science*).

Muito embora algumas buscas retornarem trabalhos que ainda não haviam sido publicados (inclusive datados de 2018), os artigos pesquisados e selecionados estavam em sua versão final, sendo, portanto, citáveis e utilizáveis como parte da revisão sistemática.

Ao todo, foram selecionados 14 trabalhos únicos através dos mecanismos de busca da base ScienceDirect, como detalha a Tabela 6.

Os trabalhos não selecionados se enquadram em pelo menos um dos seguintes critérios:

- Não possuem relação com abordagens de testes;
- Não possuem relação com plataformas ou dispositivos móveis;
- Trabalhos publicados em outras línguas: Espanhol e Francês.

Palavra-chave	Idioma	Resultado (sem filtros)	Resultado (com filtros)	Selecionados
Test* AND Programa* AND Celular*	Português	8	0	0
Test* AND Programa* AND "Dispositivo* Move*"	Português	0	0	0
Test* AND Programa* AND Android	Português	0	0	0
Test* AND Programa* AND iOS	Português	10	2	0
Test* AND Programa* AND Windows	Português	40	6	0
Test* AND Qualidade AND Programa*	Português	34	0	0
Qualidade AND Programa* AND Falh*	Português	4	0	0
Qualidade AND Programa* AND Err*	Português	4	0	0
Qualidade AND Programa* AND Defeit*	Português	0	0	0
Test* AND Software* AND Smartphone*	Inglês	74	31	3
Test* AND Software* AND Mobile	Inglês	333	118	9
Test* AND Software* AND Android	Inglês	29	18	0
Test* AND Software* AND iOS	Inglês	25	9	0
Test* AND Software* AND Windows	Inglês	523	125	0
Test* AND Quality AND Software*	Inglês	2.333	487	2
Quality AND Software* AND Flaw*	Inglês	41	11	0
Quality AND Software* AND Error*	Inglês	833	162	0
Quality AND Software* AND Defect*	Inglês	291	99	0
Total		**4.582**	**1.068**	**14**

Tabela 6. Resultados das buscas por termos de pesquisa na base ScienceDirect.

4.2.5 Consolidação de Resultados

Aplicados os critérios de inclusão e exclusão, a leitura do título e dos resumos dos trabalhos possibilitou a identificação de 39 trabalhos possivelmente relevantes para os objetivos desta revisão.

O resultado dividido por bases pode ser conferido na Tabela 7 e no Gráfico 1. A lista contendo todos os trabalhos selecionados está detalhada no Apêndice B – Lista de Trabalhos Selecionados a Partir das Buscas nas Bases de Pesquisa.

Base de pesquisa	Endereço de acesso	Resultado total (sem filtros)	Resultado total (com filtros)	Total de trabalhos selecionados
ACM	https://goo.gl/hkdRKV	134.736	77.011	14
IEEE	https://goo.gl/bJhtTs	22.541	11.833	8
SciELO	https://goo.gl/pUJJYN	1.299	667	3
Science Direct	https://goo.gl/oSxGtj	4.582	1.068	14
Total		**163.158**	**90.579**	**39**

Tabela 7. Consolidação de resultados das buscas em cada base de pesquisa.

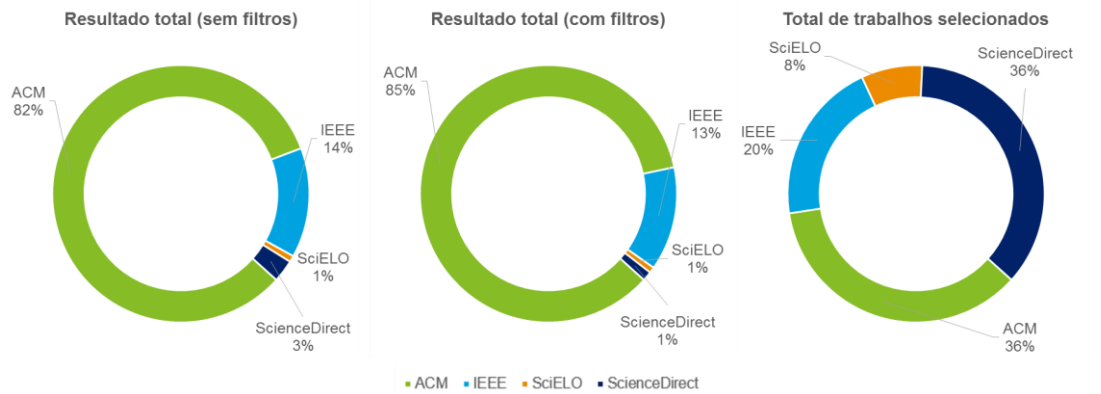


Gráfico 1. Representação gráfica dos resultados das buscas. Fonte: Autoria Própria

5 Resultados da Revisão Sistemática da Literatura

Neste capítulo, as respostas às perguntas de pesquisa e a análise dos resultados da revisão sistemática são apresentadas.

5.1 Respostas às Questões de Pesquisa

Com a leitura dos 39 trabalhos selecionados, foi possível responder às perguntas de pesquisa, relacionar e entender dificuldades, desafios, lacunas e diferentes abordagens no âmbito de testes em aplicações móveis. Os trabalhos selecionados são referenciados no Apêndice B.

QP1. Quais são os desafios para a realização de teste em aplicações móveis?

Abordando especificamente a plataforma Android, diversos trabalhos apontaram como principal ponto de dificuldade a heterogeneidade de recursos físicos (*hardware*) dos dispositivos, como telas de diferentes tamanhos e resoluções ou a presença/ausência de sensores (infravermelho, Bluetooth, etc).

Além disso, a plataforma é conhecida por possuir uma grande fragmentação de sistema operacional entre os dispositivos. Isso significa que nem sempre os dispositivos terão a mesma versão de sistema, o que pode acarretar em diferentes resultados em diferentes testes.

De acordo com Rupesh, Jääskeläinen e Katara (2012), testes baseados em interface e/ou interação com usuário podem não funcionar apropriadamente por conta de alterações de interface, podendo ser

decorrentes de telas com divergentes resoluções, ou simples atualizações do sistema operacional.

Shauvik (2014) encontrou inconsistências ao testar diferentes aplicações e até acessar a mesma página *web* em diferentes dispositivos, como falha de estruturação da página e diferente disposição de botões na aplicação.

Domenico et al (2013) afirmam que a prática de testar suas aplicações é uma atividade desafiadora, com vários questionamentos e problemas específicos; a imaturidade do desenvolvimento de aplicações Android gera mais necessidade de testes precisos e controle de qualidade rigoroso.

Para Samer, Salleh e Grundy (2016), desafios de análise de usabilidade e avaliação de aplicações móveis envolvem restrições dos dispositivos e falta de ferramentas de apoio. Joorabchi et al. (2013) compartilham do mesmo ponto de vista, afirmando que há falta de ferramentas robustas de monitoramento, testes e análise, e complementam dizendo que os emuladores disponíveis em 2013 (ano de publicação do trabalho) são lentos e não possuem boa parte dos recursos dos dispositivos móveis.

Deng et al (2016) realçam o fato do baixo desempenho das ferramentas deixarem os testes e suas avaliações mais demoradas, sugerindo que testes em paralelo em um novo *framework* possivelmente resolveria esse problema.

Outro obstáculo observado na pesquisa é o alto custo, seja de tempo e/ou de recursos., de Dev, Jääskeläinen e Katara (2012), percebe-se que para criar, manipular e trabalhar com testes automatizados, é necessário conhecimento, mão de obra especializada, treinamento e atualização de novas técnicas. Montar um ambiente ou ferramenta de testes apropriadamente poderia levar meses ou até anos.

Os desenvolvedores entrevistados pelos autores Joorabchi, et al. (2012) afirmam que são responsáveis pelos testes das aplicações que eles mesmos desenvolveram, o que evidencia ainda mais a falta de investimento (ou interesse) na área de testes.

Para testes de usabilidade, Geisen e Bergstrom (2017) apresentam a necessidade de se levar em consideração o grupo de pessoas selecionadas para avaliação, o ambiente, as atividades a serem realizadas, os equipamentos necessários para acompanhamento, entre outros fatores; e tudo isso envolve custos.

Ainda sobre testes de usabilidade, Biel et al.(2010) mostram que fatores externos podem influenciar o desenvolvimento de aplicações. Limitações físicas do usuário e usuários que possuem necessidades especiais podem ter experiência de uso (e de testes) diferente de um usuário comum. Da mesma forma, distrações no ambiente podem comprometer a avaliação de resultados de testes.

QP2. Quais são os tipos de teste utilizados em aplicações móveis?

Dentre o observado durante a pesquisa, o tipo de teste que mais se mostrou recorrente foi o teste de usabilidade, sendo relacionada por e/ou aplicada em 11 trabalhos selecionados. O teste de integração também foi um tipo de teste copioso, estando presente em 9 trabalhos lidos pelos autores.

Um total de 10 tipos de teste diferentes foram registradas pelos autores desta revisão sistemática. A Tabela 8 e o Gráfico 2 apresentam a relação de tipos de testes e a quantidade de trabalhos selecionados em que eles foram abordados.

Tipos de teste	Quantidade
Teste de usabilidade	11
Teste de integração	9
Teste de automação	5
Teste de interação com interface gráfica	2
Teste de segurança	2
Teste baseado em modelo	1
Teste baseado em eventos	1
Teste de mutação	1
Teste adaptável	1
Teste de componentes	1

Tabela 8. Relação de tipos de testes observados nos trabalhos selecionados.

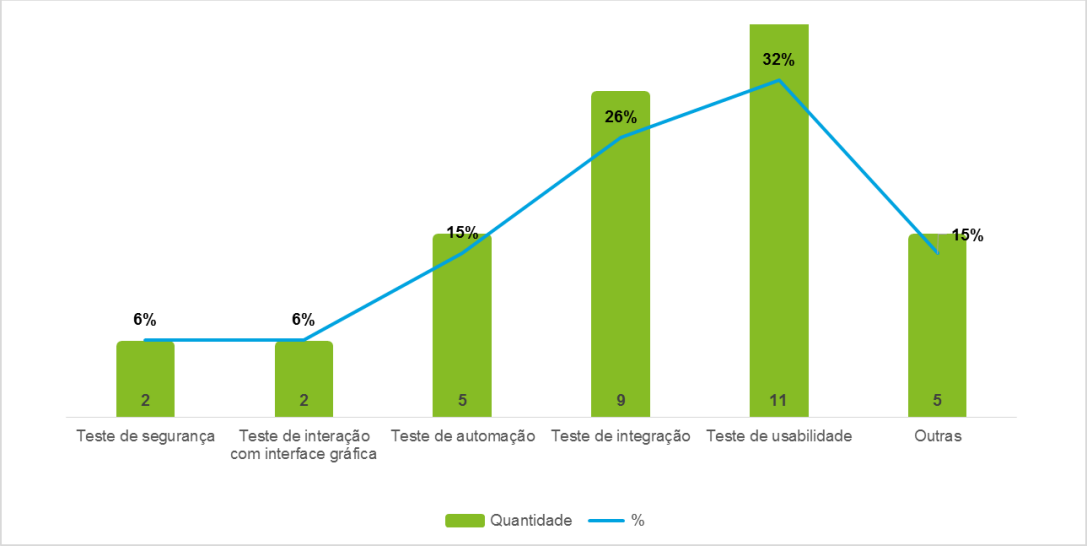


Gráfico 2. Representação gráfica dos tipos de teste. Fonte: Autoria Própria

QP3. Existem lacunas de pesquisa na área de teste em aplicações móveis?

De uma forma geral, Zein, Salleh e John Grund (2016) apontam poucos estudos baseados em ambientes reais de desenvolvimento; poucos utilizam como recurso principal a engenharia de requisitos.

Amalfitano, Domenico, et al. (2012) complementam essa linha de raciocínio; os autores afirmam que, apesar de aplicações móveis necessitarem de ênfase em determinadas atividades de testes, os testes

baseiam-se em princípios e técnicas generalistas. É possível constatar que não se trata de uma abordagem válida devido aos resultados inconsistentes apresentados, por conta da fragmentação dos dispositivos e outros pontos citados na QP1.

De acordo com Holl, Vieira e Faria (2016), os autores do, grande parte das ferramentas e de estratégias de teste não levam em consideração as variações de conectividade, como oscilação de sinal de rede ou a ativação e desativação de recursos como Wi-Fi ou NFC. Além disso, os mesmos autores explicam que o uso de registros de atividades (ou *logs*) é imprescindível para reproduzir erros e falhas de aplicações. Nem sempre esse recurso está disponível, seja na execução da aplicação ou durante a sua fase de testes.

Deng et al (2016) explicam que novas técnicas ou abordagens de testes talvez possam gerar novas classificações, categorizações e definições não aplicáveis a elas; isso é, os resultados e as análises das abordagens atuais podem não ser válidas para estratégias propostas.

QP4. Quais são os grupos de pesquisa e/ou autores que trabalham com pesquisa em teste de aplicações móveis?

A partir dos artigos selecionados não foi possível observar a existência de grupos de pesquisa ou de autores que apresentem trabalhos recorrentes em teste de aplicações móveis.

QP5. Abordagens de teste para aplicações móveis têm sido propostas?

A pesquisa de Holl et al. (2016) se trata de uma proposta de avaliação e melhoria de processos de teste, visando melhorar a eficácia desses processos em plataformas móveis.

O trabalho de Biel et al. (2010), por sua vez, apresenta uma proposta de inspeção e avaliação de usabilidade para apontar problemas em diferentes metodologias.

O trabalho de Deng et al. (2016) utiliza teste de mutação em aplicações Android; as modificações (ou mutações) no código-fonte dos aplicativos gera novos aplicativos; desta forma, é possível analisar a responsividade deles em relação ao original.

No trabalho de Geisen e Bergstrom (2017), os autores e desenvolvedores da aplicação de reporte de emergências inicialmente pensaram em utilizar uma espécie de questionário tanto na execução da aplicação quanto na abordagem de testes. Os desenvolvedores então julgaram que um questionário não seria a forma ideal de se reportar uma emergência, visto que muitas pessoas não estariam interessadas em responder perguntas pré-definidas durante um evento de grande porte, por exemplo. Alternativamente, utilizaram um tipo de *checklist* de acompanhamento cognitivo para avaliar a usabilidade das aplicações e responsividade dos usuários para com elas.

O trabalho de Dos Santos, Thassyane, et al. (2017) sugere o uso de um sistema para gerenciamento de registro de testes de usabilidade, algo similar a um ambiente de controle de falhas de desenvolvimento, conhecido como *bug tracker*. Além disso, os mesmos autores afirmam que o uso de ferramentas de rastreamento ocular (*eye-tracking*) auxilia o acompanhamento cognitivo dos testes de usabilidade. Entretanto, exige maior investimento financeiro do que ferramentas mais simples.

Os autores Amalfitano, Domenico, et al. (2016) propõe a utilização de análise combinatória para cobrir todas as características e recursos possíveis de uma gama de dispositivos móveis.

Vilkomir (2014), apresenta uma proposta de teste automatizado que explora a interface gráfica de uma aplicação Android para detectar falhas de codificação e estruturação.

Wetzlinger et al (2014) mostram uma proposta de abordagem de testes de aplicações Android para explorar e otimizar automaticamente sequências de testes e maximizar a descoberta de falhas.

O trabalho de Vilkomir, Sergiy, et al. (2015) avalia técnicas para seleção de modelos de dispositivos para revelar falhas específicas em cada

modelo, o que aumenta a eficiência e reduz os custos de testes de aplicações móveis.

Wetzlinger, Werner, et al. (2016) realizaram entrevistas orais com os avaliados, levantando tópicos relevantes como contexto de uso, métricas de usabilidade, métodos de avaliação e apresentação dos dados.

5.2 Análise dos Resultados

Alguns artigos tratam de desenvolvimento de aplicações móveis e suas respectivas abordagens de teste; outros, propõem diferentes abordagens ou estratégias de teste.

Da pesquisa, foi possível observar que há interesse significativo em buscar novas e melhores abordagens de testes; entretanto, ainda faltam técnicas e tipos de testes eficientes e úteis para avaliação de estratégias de teste.

Ficou ainda mais evidente como a questão da fragmentação de diferentes dispositivos e versões dos sistemas operacionais impactam no desenvolvimento e nos testes das aplicações.

Com a constante mudança e necessidade por novos recursos de *hardware* e atualizações de *software*, padronizar metodologias de testes se torna uma tarefa árdua; de fato, a própria atividade de testar se torna um contratempo aos desenvolvedores.

Desta forma, além de precisarem acompanhar atualizações de sistema e inovações em novos modelos, os desenvolvedores precisam também prestar suporte às falhas de suas próprias aplicações.

O *feedback* do usuário, sem dúvida, ajuda uma aplicação a atender à demanda do consumidor, ao mesmo tempo que representa um risco para a reputação dos desenvolvedores, caso a aplicação contenha falhas.

Klaus (2013), afirma que várias aplicações analisadas não atendem a critérios básicos de qualidade, como ter uma versão atualizada mais estável

do que a anterior. Ele complementa dizendo que, apesar de entender a demanda do usuário ser a prioridade, a prática de testes de aplicações móveis ainda precisa de muitas melhorias.

Também foi possível identificar que uma grande parte dos trabalhos aqui estudados acabavam utilizando de metodologias para os seus testes similares às utilizadas em computadores de mesa, não tendo assim uma técnica ou algo específico para dispositivos móveis.

Da mesma maneira, foi evidente a dificuldade dos desenvolvedores em aplicar a mesma técnica para os sistemas operacionais mais utilizados (Android e iOS). Por diversos motivos isso aconteceu, entre eles: problemas relacionados a codificação, arquitetura e distribuição dos aplicativos. Assim, muitos dos trabalhos analisados utilizavam a sua análise para apenas um dos sistemas operacionais.

Entre os tipos de testes expostos, quase 30% deles utilizavam o teste de usabilidade exclusivamente ou juntamente com outro tipo de teste. Isso ocorreu pela similaridade em realizar esse tipo de teste de uma maneira semelhante às plataformas *desktop*.

Por outro lado, para 81% dos trabalhos que utilizavam o teste de usabilidade, esse método não foi exclusivo para o trabalho. Juntamente com o teste de usabilidade, foram utilizados testes de integração, de automação, de segurança, além de outros recursos e materiais como questionários, entrevistas e tentativas de melhoria nos processos dos próprios testes.

Os autores Holzmann, Clemens e Hutflesz (2014), por exemplo, utilizaram em conjunto o teste de usabilidade e também o chamado Teste A/B, que vem importado da área de *design*, no qual são colocadas duas “telas” com pequenas alterações para que o usuário escolha entre suas opções. Os autores comentam que essa é uma técnica muito utilizada para páginas *web* em navegadores *desktop*, porém pouco aplicada para dispositivos móveis. Eles propõem um modo de utilizar a mesma lógica e metodologia dentro dos celulares e *tablets* para testar aplicações e a própria navegação da aplicação.

Outro viés que foi muito apresentado nos trabalhos foi a necessidade de automação de muitos processos de testes. Atualmente,

muitas falhas e problemas acontecem durante o teste dos usuários, podendo ser anteriormente reveladas caso houvesse uma etapa automatizada anterior ao teste do usuário. Assim, caberia ao usuário outros tipos de testes e sugestões de melhorias. Shauvik (2014) propõe novas técnicas de automatização para melhorar as etapas de testes na manutenção dos programas desenvolvidos.

6 Conclusão

Durante a condução das buscas desta revisão sistemática, os autores encontraram diversas dificuldades e limitações, principalmente em relação às bases de pesquisa.

Cada uma das bases possui um motor de busca que trabalha de maneira diferente dos outros. Consequentemente, os operadores lógicos não apresentam um mesmo padrão de funcionamento, bem como o agrupamento das palavras-chave de cada termo.

Além disso, as bases possuem recursos limitados para extração de dados; a exclusão de trabalhos a partir da leitura do título mostrou-se mais árdua do que o esperado.

A página de resultados de busca de todas as bases de pesquisa recorrentemente apresentava erro de navegação; na maioria dos casos, a busca precisava ser refeita. A exportação da lista de resultados para arquivo de texto era comumente limitada aos 200 primeiros resultados, prejudicando o desempenho dos autores.

Inicialmente, 55 trabalhos seriam selecionados para a análise de resultado; entretanto, 16 deles não foram publicados gratuitamente na Internet. Desta forma, foi necessário alterar o protocolo de revisão para incluir um último critério de exclusão: a remoção de artigos que não são acessíveis gratuitamente.

Realizada a revisão sistemática, os autores puderam perceber como é comum pesquisadores evidenciarem e enaltecerem como testes são importantes para qualquer tipo de aplicação móvel. Por outro lado, por ser uma atividade que demanda tempo e dinheiro, acaba sendo subjugada.

Para a plataforma Android, por exemplo, que possui ferramentas gratuitas de desenvolvimento e publicação de aplicações móveis, não existe critério nem controle de qualidade dessas aplicações. A forma mais comum de se realizar melhorias nas aplicações para melhor atender às necessidades dos usuários ocorre pelo *feedback* deles.

No trabalho de Klaus (2013), ao analisar diversas avaliações de usuários, o autor descobriu que várias aplicações não encontram critérios básicos de qualidade, como a aplicação apresentar erro ao ser instalada ou executada, ou uma aplicação atualizada apresentar mais instabilidade do que sua versão anterior.

As plataformas e os próprios dispositivos móveis sofrem atualizações constantes, o que implica em adaptações de ferramentas e abordagens de testes para acompanhar esse ritmo de mudanças. Além disso, ainda é frequente o uso de técnicas e critérios de teste de outros dispositivos em aplicações móveis. Pelo estudo e a análise dos resultados da revisão sistemática, foi perceptível aos autores a indiferença que uma técnica ou estratégia genérica de teste faz para dispositivos móveis.

A falta de uma centralização do como a aplicação é desenvolvida por vezes atrapalha a etapa de seus testes, uma vez que se é gasto mais tempo de desenvolvimento fazendo com que a aplicação seja executável em mais de um sistema operacional causa encurtamento da etapa de testes. O fato de muitas vezes o aplicativo precisar funcionar e ser lançado simultaneamente para mais de um sistema operacional acarreta esse cenário.

Traçar um paralelo entre expectativa e qualidade do produto entregue muitas vezes é necessário para entender até que ponto se quer chegar com o produto que está sendo desenvolvido.

Para melhor entendimento do ciclo de vida do desenvolvimento de aplicações móveis, mais especificamente no que se diz respeito às atividades de testes, seria necessário responder a várias outras perguntas de pesquisa, por exemplo:

- Quais foram os critérios aplicados para a seleção do tipo, da técnica ou da estratégia de teste aplicada?
- Existe investimento de tempo e recursos para as atividades de testes em aplicações móveis?
- Como entregar uma aplicação de qualidade, quando cada vez mais existe a necessidade de se criar mais em pouco tempo?

- De que maneira pode-se diminuir a quantidade de erros que chegam aos usuários em versões finais do programa?

Em relação às principais ameaças ao resultado e a validade desta revisão sistemática, pode-se incluir:

- a alteração do critério de exclusão, pois alguns artigos não estavam disponíveis gratuitamente;
- a relação de palavras-chave e *strings* de busca, que poderiam ter sido melhor calibradas, de modo que podem ter levado a resultados equivocados, ou ainda, que não contemplem com totalidade o escopo deste trabalho;
- o erro humano na classificação e/ou seleção de trabalhos relevantes para a revisão sistemática.

Para futuros trabalhos, os autores entendem a necessidade do estudo aprofundado, com questões de pesquisa mais específicas, para que possam ser analisadas outras possíveis técnicas e/ou abordagens de teste em aplicações móveis que não foram exploradas nesse trabalho.

A partir desta revisão sistemática, foi possível apontar e evidenciar o valor da etapa de testes não só de aplicações móveis, mas na engenharia de software como um todo. Considerando a ascensão ininterrupta do cenário de dispositivos móveis, a relação entre a necessidade de solução e a necessidade da entrega (de prazos cada vez mais curtos) deve ser aliada à necessidade de garantia de qualidade.

7 Referências

ACKER, E. V.; WEBER, T. S.; CECHIN, S. L. “Injeção de falhas para validar aplicações em ambientes móveis”. Workshop de Testes e Tolerância a Falhas, 2010.

BACH, J. “The Immaturity of the CMM”. American Programmer, v. 7, p. 13-13, 1994.

BARBOSA, E. F.; MALDONADO, J. C.; VINCENZI, A. M. R. “Introdução ao Teste de *Software*”. USP, 2006.

BESSIN, G. “The business value of *software* quality”. The Rational Edge, p. 1-8, 2004.

BIOLCHINI, J. et al. “*Systematic review in software engineering*”. Tech. Report RT-ES 679/05, 2005.

CHERNONOZHKIN, S. K. “Automated test generation and static analysis”. Programmingand Computer *Software*, v. 27, n. 2, p. 86–94, mar 2001.

COULOURIS, G. et al. “Sistemas Distribuídos: Conceitos e Projeto”. Bookman Editora, 2013.

CRAIG, R. D.; JASKIEL, S. P. “Systematic *Software* Testing”. Artech House Publishers. Boston, 2002.

GAO, J. et al. “Mobile Application Testing – Research, Practice, Issues and Needs Testing, Requirements and Features, 4(1).” 2013.

GARVIN, D. A. “Competing on the 8 dimensions of quality”. Harvard business review, v. 65, n. 6, p. 101-109, 1987.

GIL, A. C. “Métodos e técnicas de pesquisa social. 4ª ed.” Atlas. São Paulo, 1994.

GIL, A. C. “Como elaborar projetos de pesquisa”. São Paulo, 2002.

HAGENS, V. E. et al. “Effect of rate or rhythm control on quality of life in persistent atrial fibrillation: Results from the Rate Control Versus Electrical Cardioversion (RACE) study”. Journal of the American College of Cardiology, v. 43, n. 2, p. 241-247, 2004.

HIGGINS, J. P. T.; GREEN, S. “Cochrane Handbook for Systematic Reviews of Interventions”. The Cochrane Colaboration, 2011. – Disponível em http://handbook.cochrane.org/, acessado em 23 de junho de 2015.

IEEE Standards Board. “IEEE Standard Glossary of *Software* Engineering Terminology”. IEEE Press, 1990.

JURAN, J. M.; GRYNA, F. M. “Juran’s quality control handbook”. NY: McGraw-Hill, 1988.

KIRSCHNER, S. F. “Um sistema de auxílio à coleta de dados na área de agricultura de precisão baseado em aplicações móveis”. 2013.

KITCHENHAM, B.; CHARTERS, S. “Guidelines for performing Systematic Literature Reviews in *Software* Engineering”. Keele University and Durham University Joint Report, 2007. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=17B9360155C581F8A64CB8031389646C?doi=10.1.1.117.471&rep=rep1&type=pdf, acessado em 19 de maio de 2015.

KOSCIANSKI, A; SOARES, M. S. “Qualidade de *software*”. 2007.

LAKATOS, E. M.; MARCONI, M. de A. “Metodologia científica”, São Paulo: Atlas, 1991.

LECHETA, R. R. “Google Android - 3ª Edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK”. Novatec Editora, 2013.

LIMA, L. A. T. “Computação Móvel”. 2000. Disponível em http://www.dimap.ufrn.br/~gold/CMovel.html, acessado em 24 de junho de 2015.

LINDE, K.; WILLICH, S. N. “How objective are systematic reviews? Differences between reviews on complementary medicine”. J R Soc Med, 2003. Disponível em http://www.scielo.br/pdf/rbfis/v11n1/12.pdf, acessado em 20 de maio de 2015.

LYU, M. R. “Handbook of *software* reliability engineering”. IEEE Computer Society Press; McGraw-Hill Book Company, 1995. – Disponível em http://www.cse.cuhk.edu.hk/~lyu/book/reliability/, acessado em 18 de maio de 2015.

MCCALL, J. A.; RICHARDS, P. K.; WALTERS, G. F. “Factors in *software* quality. Volume I. Concepts and definitions of *software* quality”. GENERAL ELECTRIC CO SUNNYVALE CA, 1977.

MILLER, E. F. “Introduction to *Software* Testing Technology” Tutorial: *Software* Testing & Validation Techniques, Second Edition, IEEE Catalog No. EHO 180-0, pp. 4-16, 2001.

MULLER, T. et al. “Certified Tester Foundation Level Syllabus”. International *Software* Testing Qualifications Board, 2007.

NETO, A. C. D. “Introdução a Teste de *Software*”. Engenharia de *Software* Magazine, v. 1, 2008. – Disponível em http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035, acessado em 29 de abril de 2015.

OLIVER, E. “A survey of platforms for mobile networks research”. ACM SIGMOBILE Mobile Computing and Communications Review, v. 12, n. 4, p. 56-63, 2009.

PRADHAN, T. “Mobile Application Testing”. White Paper, 2011. – Disponível em http://www.tcs.com/SiteCollectionDocuments/White%20Papers/Mobility_Whit

epaper_Mobile-Application-Testing_1012-1.pdf, acessado em 20 de junho de 2015.

PRESSMAN, R. S. “Engenharia de *software*”. McGraw Hill Brasil, 2011.

PRESSMAN, R. S. “*Software* Engineering – A Practitioner’s Approach”. McGraw-Hill, 4th edition, 1997.

RAPOZA, J. “First Class Mobile Application Performance Management”. The Aberdeen Group, 2012. – Disponível em http://info.soasta.com/rs/soasta/images/aberdeen-first-class-mobile-application-performance-management.pdf, acessado em 18 de junho de 2015.

RAPPS, S.; WEYUKER, E. J. “Data Flow analysis techniques for test data selection”. International Conference on *Software* Engineering, p. 272-278, Tokyo, Sep. 1982.

REED, K. “*Software* engineering – a new millennium?” IEEE *Software*, jul.-ago. 2000.

RINCON, A. M. “Qualidade de *Software*”. Instituto de Informática: Universidade Federal de Goiás, v. 8, 2009 – Disponível em http://www3.ulbra-to.br/eventos/encoinfo/2009/anais/Qualidade_de_Software.pdf, acessado em 30 de maio de 2015.

ROCHA, A. R. C. et al. “Qualidade de *software* – Teoria e prática”, Prentice Hall, São Paulo, 2001.

SmartBear Software. “What is mobile testing?”. 2014. – Disponível em http://smartbear.com/all-resources/articles/what-is-mobile-testing/, acessado em 19 de junho de 2015.

SOMMERVILLE, I. “Engenharia de *Software*”. 8ª edição. Pearson Addison-Wesley, São Paulo, 2007.

VIANA, V. “Um Método para Seleção de Testes de Regressão para Automação”. 2006.

WEINBERG, G. M. “The psychology of computer programming”. New York: Van Nostrand Reinhold, 1971.

APÊNDICE A – Conceitos de Revisão Sistemática

A revisão sistemática da literatura, ou simplesmente revisão sistemática, é uma forma de avaliar e interpretar as pesquisas disponíveis e relevantes de uma específica questão, área ou um fenômeno de interesse (KITCHENHAM; CHARTERS, 2007). Revisões sistemáticas apresentam uma boa avaliação de um tópico de pesquisa utilizando uma metodologia confiável, rigorosa e passível de validação.

A necessidade de se realizar uma revisão sistemática surge a partir da exigência dos pesquisadores de sintetizar todas as informações existentes de um determinado fenômeno, de maneira rigorosa e imparcial. Consequentemente, isso pode gerar novas conclusões gerais do que seria possível com estudos isolados, ou pode ser usado como prelúdio para pesquisas futuras.

Estudos que contribuem para revisões sistemáticas são chamados de estudos primários. Revisão sistemática é uma forma de estudo secundário, que usa uma metodologia bem-definida para identificar, analisar e interpretar as evidências disponíveis, relacionadas a um assunto de pesquisa específico, de forma que seja imparcial e, a certo ponto, reproduzível (KITCHENHAM; CHARTERS, 2007).

Há muitos motivos para se realizar uma revisão sistemática. De acordo com Kitchenham e Charters (2007), as mais comuns são:

- Sintetizar as evidências existentes a um tratamento ou uma tecnologia; por exemplo, sintetizar evidências empíricas dos benefícios e limitações de uma certa metodologia ágil;
- Identificar obstáculos em pesquisas, a fim de sugerir áreas para investigações futuras;
- Providenciar estrutura ou histórico para posicionar apropriadamente novas pesquisas.

Revisões sistemáticas podem também ser realizadas para analisar em que ponto uma evidência empírica apoia ou contradiz hipóteses, ou até para auxiliar na criação de novas hipóteses (KITCHENHAM; CHARTERS, 2007).

A maioria das pesquisas se iniciam com algum tipo de revisão de literatura. Entretanto, a não ser que uma revisão de literatura seja robusta e de qualidade, ela tem pouco valor científico (KITCHENHAM; CHARTERS, 2007). Essa é a principal base lógica para proceder com revisões sistemáticas.

A revisão sistemática sintetiza trabalhos existentes, de forma que seja e demonstre ter boa qualidade (KITCHENHAM; CHARTERS, 2007). Por exemplo, revisões sistemáticas devem ser realizadas de acordo com uma estratégia de pesquisa predefinida. A estratégia de pesquisa deve permitir que a integridade da pesquisa seja avaliada. Especificamente, pesquisadores realizando uma revisão sistemática devem se esforçar ao máximo para identificar e relatar pesquisas que não apoiam sua hipótese de pesquisa escolhida, assim como identificar e relatar pesquisas que a apoiam.

De acordo com Kitchenham e Charters (2007), algumas vantagens da revisão sistemática são:

- Dada a metodologia bem-definida, é improvável que os resultados da literatura sejam tendenciosos, apesar de não impedir publicações parciais de estudos primários;
- Informar sobre efeitos de alguns fenômenos por meio de uma ampla gama de definições e métodos empíricos. Se estudos dão resultados consistentes, revisões sistemáticas entregam evidências de que o fenômeno é robusto e transferível; se os resultados forem inconsistentes, as fontes das variações podem ser estudadas;
- Em estudos quantitativos, é possível combinar informações usando metodologias de metanálise. Isso aumenta a probabilidade de detectar efeitos que estudos menores são incapazes de detectar.

A grande desvantagem da revisão sistemática é que ela exige um esforço consideravelmente maior do que as tradicionais revisões de literatura. Além disso, maior força em metanálise pode também se tornar uma desvantagem, já que é possível detectar pequenas vieses, bem como efeitos verdadeiros (KITCHENHAM; CHARTERS, 2007).

APÊNDICE B – Lista de Trabalhos Selecionados a Partir das Buscas nas Bases de Pesquisa

#	Base de pesquisa	Palavra-chave resultante	Referência do trabalho	Página de referência
1	ScienceDirect	Test* AND Software* AND Mobile	Zein, Samer, Norsaremah Salleh, and John Grundy. "A systematic mapping study of mobile application testing techniques." Journal of Systems and Software 117 (2016): 334-356.	http://www.sciencedirect.com/science/article/pii/S0164121216300140
2	ScienceDirect	Test* AND Software* AND Mobile	Amalfitano, Domenico, et al. "Testing Android Mobile Applications: Challenges, Strategies, and Approaches." Advances in Computers 89.6 (2013): 1-52.	http://www.sciencedirect.com/science/article/pii/B9780124080942000011
3	ScienceDirect	Test* AND Software* AND Mobile	Holl, Konstantin, Vaninha Vieira, and Igor Faria. "An Approach for Evaluating and Improving the Test Processes of Mobile Application Developments." Procedia Computer Science 94 (2016): 33-40.	http://www.sciencedirect.com/science/article/pii/S1877050916317495
4	ScienceDirect	Test* AND Software* AND Mobile	Braun, Susanne, Frank Elberzhager, and Konstantin Holl. "Automation Support for Mobile App Quality Assurance–A Tool Landscape." Procedia Computer Science 110 (2017): 117-124.	http://www.sciencedirect.com/science/article/pii/S187705091731308X
5	ScienceDirect	Test* AND Quality AND Software*	Biel, Bettina, Thomas Grill, and Volker Gruhn. "Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application." *Journal of Systems and Software* 83.11 (2010): 2031-2044.	http://www.sciencedirect.com/science/article/pii/S0164121210001421
6	ScienceDirect	Test* AND Software* AND Mobile	Karahoca, Adem, et al. "Information system design for a hospital emergency department: A usability analysis of software prototypes." *Journal of biomedical informatics* 43.2 (2010): 224-232.	http://www.sciencedirect.com/science/article/pii/S1532046409001191
7	ScienceDirect	Test* AND Software* AND Smartphone*	Dev, Rupesh, Antti Jääskeläinen, and Mika Katara. "Model-based GUI testing: Case smartphone camera and messaging development." *Advances in Computers* 85 (2012): 65-122.	http://www.sciencedirect.com/science/article/pii/B9780123965264000002
8	ScienceDirect	Test* AND Software* AND Mobile	Deng, Lin, et al. "Mutation operators for testing Android apps." *Information and Software Technology* 81 (2017): 154-168.	http://www.sciencedirect.com/science/article/pii/S0950584916300684

9	ScienceDirect	Test* AND Software* AND Mobile	Moldovan, Liviu. "New evaluation model by means of mobile technology." *Procedia Technology* 19 (2015): 1094-1101.	http://www.sciencedirect.com/science/article/pii/S221201731500157
10	ScienceDirect	Test* AND Software* AND Smartphone*	Liu, Bo, and A. Bulent Koc. "SafeDriving: A mobile application for tractor rollover detection and emergency reporting." *Computers and electronics in agriculture* 98 (2013): 117-120.	http://www.sciencedirect.com/science/article/pii/S016816991300170
11	ScienceDirect	Test* AND Software* AND Mobile	Cugola, Gianpaolo, et al. "Selfmotion: A declarative approach for adaptive service-oriented mobile applications." *Journal of Systems and Software* 92 (2014): 32-44.	http://www.sciencedirect.com/science/article/pii/S016412121300265
12	ScienceDirect	Test* AND Quality AND Software*	de Sousa Santos, Ismayle, et al. "Test Case Design for Context-Aware Applications: Are We There Yet?." *Information and Software Technology* (2017).	http://www.sciencedirect.com/science/article/pii/S095058491730251
13	ScienceDirect	Test* AND Software* AND Smartphone*	Holl, Konstantin, et al. "Towards a lightweight approach for on-site interaction evaluation of safety-critical mobile systems." *Procedia computer science* 94 (2016): 41-48.	http://www.sciencedirect.com/science/article/pii/S187705091631750
14	ScienceDirect	Test* AND Software* AND Mobile	Usability Testing for Survey Research Author(s): Emily Geisen and Jennifer Romano Bergstrom ISBN: 978-0-12-803656-3	http://www.sciencedirect.com/science/article/pii/B978012803656300004
15	SciELO	Test* AND Programa* AND Dispositivo* Move*	Silva dos Santos, Thassyane, et al. "Desenvolvimento de aplicativo para dispositivos móveis voltado para identificação do fenótipo de fragilidade em idosos." *Revista Brasileira de Geriatria e Gerontologia* 20.1 (2017).	http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1809-98232017000100067&lng=en&nrm=iso&tlng=p
16	SciELO	Test* AND Programa* AND Android	de Oliveira, Renata Marques, et al. "Desenvolvimento do aplicativo TabacoQuest para informatização de coleta de dados sobre tabagismo na enfermagem psiquiátrica." *Revista Latino-Americana de Enfermagem* 24 (2016): 1-10.	http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-11692016000100399&lng=en&nrm=iso&tlng=p
17	SciELO	Test* AND Software* AND Mobile	Ermoshina, Ksenia. "Is There an App for Everything? Potentials and Limits of Civic Hacking." *Observatorio (OBS*)*10.ESPECIAL (2016): 116lpage-140.	http://www.scielo.mec.pt/scielo.php?script=sci_arttext&pid=S1646-59542016000300007&lang=p
18	IEEE	Test* AND Software* AND Smartphone*	Joorabchi, Mona Erfani, Ali Mesbah, and Philippe Kruchten. "Real challenges in mobile app development." *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*. IEEE, 2013.	http://ieeexplore.ieee.org/document/6681334

19	IEEE	Test* AND Software* AND Android	Silva, Davi Bernardo, et al. "An analysis of automated tests for mobile Android applications." *Computing Conference (CLEI), 2016 XLII Latin American*. IEEE, 2016.	http://ieeexplore.ieee.org/document/7833334
20	IEEE	Test* AND Software* AND Android	Ahmed, Maryam, Rosziati Ibrahim, and Noraini Ibrahim. "Adaptation model for testing android application." *Computing Technology and Information Management (ICCTIM), 2015 Second International Conference on*. IEEE, 2015.	http://ieeexplore.ieee.org/document/7224606
21	IEEE	Test* AND Software* AND Android	AMALFITANO, Domenico et al. MobiGUITAR: Automated model-based testing of mobile apps. IEEE Software, v. 32, n. 5, p. 53-59, 2015.	http://ieeexplore.ieee.org/document/6786194
22	IEEE	Test* AND Software* AND Mobile	CHUNYE, Du; WEI, Song; JIANHUA, Wu. Based on the analysis of mobile terminal application software performance test. In: Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2017 18th IEEE/ACIS International Conference on. IEEE, 2017. p. 391-395.	http://ieeexplore.ieee.org/document/8022751
23	IEEE	Test* AND Software* AND Mobile	SANTOS, Andreia; CORREIA, Igor. Mobile testing in software industry using agile: challenges and opportunities. In: Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on. IEEE, 2015. p. 1-2.	http://ieeexplore.ieee.org/document/7102625
24	IEEE	Test* AND Software* AND Mobile	MUCCINI, Henry; DI FRANCESCO, Antonio; ESPOSITO, Patrizio. Software testing of mobile applications: Challenges and future research directions. In: Proceedings of the 7th International Workshop on Automation of Software Test. IEEE Press, 2012. p. 29-35.	http://ieeexplore.ieee.org/document/6228987
25	IEEE	Test* AND Software* AND Mobile	VILKOMIR, Sergiy; AMSTUTZ, Brandi. Using combinatorial approaches for testing mobile applications. In: Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on. IEEE, 2014. p. 78-83.	http://ieeexplore.ieee.org/document/6825641
26	ACM	+(Test* +(Programa* +("Dispositivo Movel" "Dispositivos Moveis")))	BRITTO, Talita CP et al. Técnicas de Prototipação para Smartphones no Apoio à Avaliação de Interfaces com o Usuário. In: Proceedings of the Companion Proceedings of the 10th Brazilian Symposium on Human Factors in Computing Systems	https://dl.acm.org/citation.cfm?id=2254535&CFID=820331379&CFTOKEN=8720287

			and the 5th Latin American Conference on Human-Computer Interaction. Brazilian Computer Society, 2011. p. 39-42.	
27	ACM	+(Test* +(Programa* +("Dispositivo Movel" "Dispositivos Moveis")))	BONIFÁCIO, Bruno; OLIVEIRA, Horácio; CONTE, Tayana. Avaliação de usabilidade de aplicações em dispositivos móveis. In: Proceedings of the IX Symposium on Human Factors in Computing Systems. Brazilian Computer Society, 2010. p. 269-270.	https://dl.acm.org/citation.cfm?id=1999642&CFID=820331379&CFTOKEN=8720287
28	ACM	+(Quality +(Software* +(Flaw*)))	Lettner, Florian, and Clemens Holzmann. "Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications." *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*. ACM, 2012.	https://dl.acm.org/citation.cfm?id=2428983&CFID=820331379&CFTOKEN=8720287
29	ACM	+(Test* +(Programa* +(Android)))	Amalfitano, Domenico, et al. "Using GUI ripping for automated testing of Android applications." Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. ACM, 2012.	https://dl.acm.org/citation.cfm?id=2351717&CFID=820331379&CFTOKEN=8720287
30	ACM	+(Test* +(Programa* +(Android)))	Mahmood, Riyadh, Nariman Mirzaei, and Sam Malek. "Evodroid: Segmented evolutionary testing of android apps." Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2014.	https://dl.acm.org/citation.cfm?id=2635896&CFID=820331379&CFTOKEN=8720287
31	ACM	+(Test* +(Programa* +(Android)))	Tiwari, Nitish. "Compatibility test suite--test your Android implementation." Linux Journal 2013.230 (2013): 2.	https://dl.acm.org/citation.cfm?id=2502866&CFID=820331379&CFTOKEN=8720287
32	ACM	+(Test* +(Programa* +(Android)))	MAO, Ke; HARMAN, Mark; JIA, Yue. Sapienz: Multi-objective automated testing for android applications. In: Proceedings of the 25th International Symposium on Software Testing and Analysis. ACM, 2016. p. 94-105.	https://dl.acm.org/citation.cfm?id=2931054&CFID=820331379&CFTOKEN=8720287
33	ACM	+(Quality +(Software* +(Defect*)))	Haller, Klaus. "Mobile testing." ACM SIGSOFT Software Engineering Notes 38.6 (2013): 1-8.	https://dl.acm.org/citation.cfm?id=2532813&CFID=820331379&CFTOKEN=8720287
34	ACM	+(Test* +(Quality +(Software*)))	Roy Choudhary, Shauvik. "Cross-platform testing and maintenance of web and mobile applications." *Companion*	https://dl.acm.org/citation.cfm?id=2591097&CFID=820331379&CFTOKEN=8720287

			Proceedings of the 36th International Conference on Software Engineering. ACM, 2014.	
35	ACM	+(Test* +(Quality +(Software*)))	Vilkomir, Sergiy, et al. "Effectiveness of multi-device testing mobile applications." *Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM International Conference on*. IEEE, 2015.	https://dl.acm.org/citation.cfm?id=2825047&CFID=820331379&CFTOKEN=8720287
36	ACM	+(Test* +(Quality +(Software*)))	ZIPT: Zero-Integration Performance Testing of Mobile App Designs Biplab Deka, Zifeng Huang, Chad Franzen, Jeffrey Nichols, Yang Li, and Ranjitha Kumar Proceedings of UIST '17	https://dl.acm.org/citation.cfm?id=3126647&CFID=820331379&CFTOKEN=8720287
37	ACM	+(Test* +(Software* +(Mobile)))	Holzmann, Clemens, and Patrick Hutflesz. "Multivariate Testing of Native Mobile Applications." *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*. ACM, 2014.	https://dl.acm.org/citation.cfm?id=2684119&CFID=820331379&CFTOKEN=8720287
38	ACM	+(Test* +(Software* +(Mobile)))	Wetzlinger, Werner, et al. "Mobile usability testing requirements and their implementation in the automation engineering industry." *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*. ACM, 2014.	https://dl.acm.org/citation.cfm?id=2684120&CFID=820331379&CFTOKEN=8720287
39	ACM	+(Test* +(Software* +(Mobile)))	Harman, Mark, et al. "Mobile app and app store analysis, testing and optimisation." *Proceedings of the International Workshop on Mobile Software Engineering and Systems*. ACM, 2016.	https://dl.acm.org/citation.cfm?id=2897076&CFID=820331379&CFTOKEN=8720287