UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

GUILHERME DE CLEVA FARTO

UMA CONTRIBUIÇÃO AO TESTE BASEADO EM MODELO NO CONTEXTO DE APLICAÇÕES MÓVEIS

DISSERTAÇÃO DE MESTRADO

CORNÉLIO PROCÓPIO
2016

GUILHERME DE CLEVA FARTO

UMA CONTRIBUIÇÃO AO TESTE BASEADO EM MODELO NO CONTEXTO DE APLICAÇÕES MÓVEIS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná – UTFPR como requisito parcial para a obtenção do título de “Mestre em Informática”.

Orientador: Prof. Dr. André Takeshi Endo

CORNÉLIO PROCÓPIO
2016

Dados Internacionais de Catalogação na Publicação

F247 Farto, Guilherme de Cleva

Uma contribuição ao teste baseado em modelo no contexto de aplicações móveis / Guilherme de Cleva Farto. – 2016.

154 f. : il. color. ; 30 cm

Orientador: André Takeshi Endo.

Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Informática. Cornélio Procópio, 2016. Bibliografia: p. 126-135.

1. Software de aplicação. 2. Software - Testes. 3. Modelagem de dados. 4. Informática – Dissertações. I. Endo, André Takeshi, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD (22. ed.) 004

Biblioteca da UTFPR - Câmpus Cornélio Procópio

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Programa de Pós-Graduação em Informática

Título da Dissertação Nº 16:

“Uma contribuição ao Teste Baseado em Modelo no contexto de aplicações móveis”

por

Guilherme de Cleva Farto

Orientador: **Prof. Dr. André Takeshi Endo**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM INFORMÁTICA – Área de Concentração: Computação Aplicada, pelo Programa de Pós-Graduação em Informática – PPGI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 16h00 do dia 08 de março de 2016. O trabalho foi APROVADO pela Banca Examinadora, composta pelos professores:

Prof. Dr. André Takeshi Endo
Presidente

Prof. Dr. Alexandre L’Erario
UTFPR - Cornélio Procópio/PR

Profa. Dra. Simone do Rocio Senger de Souza
ICMC/USP - São Carlos/SP

Visto da coordenação:

Prof. Dr. André Takeshi Endo
Coordenador do Programa de Pós-Graduação em Informática
UTFPR Câmpus Cornélio Procópio

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

À minha família, por toda a paciência, compreensão e incentivo incondicionais.

AGRADECIMENTOS

Sendo este trabalho o culminar de um ciclo, provavelmente o mais importante e desafiador de minha vida até o presente momento, gostaria de expressar meus sinceros reconhecimentos e agradecimentos a todos que, direta e indiretamente, apoiaram-me na jornada e nas experiências alcançadas por esta dissertação.

A Deus, por me dar oportunidade, capacidade e vontade para realizar este trabalho.

À minha família, em especial, meus pais Cecilia e Mauro, meu irmão Gabriel e minha irmã Silvia. Os ensinamentos, os gestos de carinho e os momentos dedicados a mim jamais serão esquecidos.

A Tamires, pelos momentos de inspiração, de incentivo e de carinho.

Ao Prof. Dr. André Takeshi Endo, pela exímia orientação e singular amizade. Um exemplo de profissional, de pesquisador e de pessoa que admiro. Obrigado, Endo!

Ao Prof. Dr. Marco Aurélio Graciotto Silva, Prof. Dr. Alexandre L'Erario e Profa. Dra. Simone do Rocio Senger de Souza pelas relevantes considerações e contribuições nas etapas de qualificação e defesa desta dissertação.

Aos professores da UTFPR, pela atenção e dedicação com que desenvolveram as aulas, bem como pelo compartilhar de conhecimentos e experiências que contribuíram para a minha formação.

Remember these six rules to success:
trust yourself, break some rules, don't be afraid to fail,
ignore the naysayers, work like hell, and give something back.

Arnold Alois Schwarzenegger (1947-)

RESUMO

FARTO, Guilherme de Cleva. **Uma contribuição ao Teste Baseado em Modelo no contexto de aplicações móveis**. 2016. 159 f. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

Devido ao crescente número e diversidade de usuários, novas abordagens de teste são necessárias para reduzir a ocorrência de defeitos e garantir uma melhor qualidade em aplicações móveis. As particularidades desse tipo de software exigem que as técnicas de teste tradicionais sejam revisitadas e novas abordagens propostas. A natureza direcionada a eventos e as funcionalidades de aplicações móveis demandam que os testes sejam executados de maneira automatizada. O Teste Baseado em Modelo (TBM) apresenta-se como uma abordagem válida e promissora que oportuniza o uso de um processo definido, bem como de mecanismos e técnicas formais para o teste de aplicações móveis. Esta dissertação de mestrado tem como objetivo investigar a adoção de TBM junto à técnica de modelagem *Event Sequence Graph* (ESG) no teste de aplicações móveis para a plataforma Android. Inicialmente, o TBM é avaliado com o apoio de ESG e da ferramenta *Robotium*. Com base nos resultados obtidos e desafios identificados, propõe-se uma abordagem específica que fundamenta o reúso de modelos de teste para (i) reduzir o esforço manual demandado na etapa de concretização de casos de teste e (ii) testar distintas características inerentes ao contexto de mobilidade. Uma ferramenta de apoio foi projetada e desenvolvida para automatizar a abordagem proposta. Um estudo experimental em ambiente industrial é conduzido para avaliar a abordagem e a ferramenta propostas quanto à efetividade na redução do esforço requisitado para a concretização, bem como à capacidade de detecção de defeitos em aplicações móveis desenvolvidas na plataforma Android.

Palavras-chave: aplicações móveis, teste baseado em modelo, teste de aplicações móveis, geração de casos de teste, testes automatizados.

ABSTRACT

FARTO, Guilherme de Cleva. **A contribution to the Model-Based Testing in the context of mobile applications**. 2016. 159 f. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

Due to the increasing number and diversity of users, new testing approaches are necessary to reduce the presence of faults and ensure better quality in mobile applications. The particularities of this class of software require that traditional testing techniques are revisited and new approaches proposed. The event oriented nature and functionalities of mobile applications demand tests that can be performed automatically. Model-Based Testing (MBT) is a valid and promising approach that favors the use of a defined process, as well as mechanisms and formal techniques for the testing of mobile applications. This dissertation investigates the adoption of MBT along with the modeling technique Event Sequence Graph (ESG) to test Android applications. Initially, we evaluate TBM supported by ESG and the Robotium tool. Based on the results and challenges identified, we propose a specific approach underlying the reuse of test models to (i) reduce the manual effort to the concretization of test cases and to (ii) test different and inherited characteristics of the mobility context. A supporting tool was designed and implemented to automate the proposed approach. Finally, we conducted an experimental study in an industrial environment to evaluate the proposed approach and tool regarding the effectiveness in reducing the concretization's efforts, as well as the fault detection capability in Android mobile applications.

Keywords: mobile applications, model-based testing, mobile application testing, test case generation, automated tests.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	Application Programming Interface
AVD	Android Virtual Device
CE	Critério de exclusão
CES	Complete Event Sequence
CI	Critério de inclusão
DFPN	Dynamic Feature Petri Nets
DSML	Domain-Specific Modeling Language
EMS	Estudo de Mapeamento Sistemático
ESG	Event Sequence Graph
GPS	Global Positioning System
GUI	Graphical User Interface
JME	Java Micro Edition
JVM	Java Virtual Machine
LoC	Lines of Code
LSTS	Labelled State Transition System
LTS	Labelled Transition System
MEF	Máquina de Estados Finitos
OHA	Open Handset Alliance
SDK	Software Development Kit
SPL	Software Product Line
SUT	Software Under Test
TaaS	Testing-as-a-Service
TBM	Teste Baseado em Modelo
TI	Tecnologia da Informação
UML	Unified Modeling Language
VV&T	Verificação, Validação e Teste

SUMÁRIO

1. INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO

A computação móvel é um paradigma no qual usuários manipulam dispositivos portáteis, ou seja, fáceis de serem transportados, também denominados de dispositivos móveis. Tais dispositivos acessam serviços para obter e submeter informações por meio de uma infraestrutura compartilhada que independe da localização física (JING et al., 1999). Segundo Forman e Zahorjan (1994), os desafios impostos pela computação móvel baseiam-se em três fundamentais propriedades: (i) comunicação, (ii) mobilidade e (iii) portabilidade. A comunicação em dispositivos móveis se refere à capacidade de estarem conectados a redes sem fio, aumentando a versatilidade e oportunizando a interação desses com outros equipamentos computacionais, bem como com a Internet. A mobilidade é caracterizada como a possibilidade do usuário mudar sua localização física, independente da conectividade do dispositivo móvel a uma rede de dados. A portabilidade se preocupa em manter um dispositivo pequeno, leve e sem conexões por fios, com o objetivo de minimizar o impacto na capacidade de computação (FORMAN; ZAHORJAN, 1994).

As propriedades fundamentais da computação móvel impõem restrições como, por exemplo, telas limitadas, menor poder de processamento, mecanismos instáveis de comunicação que dependem do acesso a redes e da Internet e baixa disponibilidade ou autonomia de energia (MUCCINI et al., 2012). Além disso, Wasserman (2010) identificou alguns fatores que tornam a engenharia de software para aplicações móveis mais especializada em relação à tradicional engenharia de software. Os principais pontos que a diferem são interação com outras aplicações, manipulação de sensores, variedade em tamanhos e resoluções de telas e consumo de energia.

Entretanto, mesmo com os desafios relacionados, há um rápido crescimento na popularidade de dispositivos móveis, como *smartphones*, *tablets* e *e-readers*, expandindo a variedade de aplicações que são desenvolvidas com o apoio de tecnologia de computação móvel. Apesar de as aplicações móveis e a própria

computação móvel terem sido inicialmente concebidas para a indústria de entretenimento, tecnologias baseadas nos conceitos de mobilidade têm sido utilizadas com sucesso em outros domínios, inclusive críticos, como sistemas financeiros, de saúde, educacionais e militares (MUCCINI et al., 2012). Devido ao aumento no número de usuários e a diversidade de sistemas operacionais e aplicações móveis, defeitos existentes em soluções computacionais podem implicar em prejuízos econômicos e, em casos mais graves, até mesmo humanos (HARROLD, 2000).

A dinamicidade das aplicações móveis e seu uso em ambientes críticos demandam testes precisos e repetíveis. Segundo Muccini et al. (2012), as peculiaridades impostas pelo teste de aplicações móveis devem ser solucionadas com base na definição de abordagens de teste específicas. Dentre as particularidades identificadas, afirma-se que conectividade, recursos limitados, interface com o usuário, diversidade de configurações e telas sensíveis ao toque são elementos presentes no contexto de computação móvel que interferem diretamente na atividade de teste (MUCCINI et al., 2012).

De acordo com Hierons et al. (2009), a existência de especificações formais pode contribuir com a geração de casos de teste mais eficientes e eficazes. O teste formal é caracterizado pela adoção de modelos matemáticos que apoiam a atividade de teste. Dessa forma, modelos oferecem benefícios para a automação desde que a sintaxe e semântica estejam bem definidas, possibilitando o desenvolvimento de ferramentas de apoio que manipulem as representações formais (HIERONS et al., 2009). Nesse contexto, o Teste Baseado em Modelo (TBM) é uma abordagem que deriva casos de teste a partir de modelos formais. Segundo Dalal et al. (1999) e Pretschner et al. (2005), o TBM reduz o tempo e custo de teste se comparado às técnicas de teste manual, além da melhoria na qualidade do teste, uma vez que o processo de teste é repetível e os requisitos do sistema são utilizados na criação do modelo formal (UTTING; LEGEARD, 2006).

Os estudos que fundamentam o processo de TBM sugerem sua divisão em quatro passos principais: (i) modelagem, (ii) geração de casos de teste, (iii) concretização e (iv) execução dos testes (PRETSCHNER; PHILIPPS, 2004; BOUQUET et al., 2006; EL-FAR; WHITTAKER, 2011). Na modelagem, o testador ou analista de teste elabora modelos que representam as funcionalidades do software em teste (em Inglês, *Software Under Test* – SUT). Na geração de casos de teste, os

modelos são artefatos de entrada para uma ferramenta de apoio que extraí as informações acerca das funcionalidades modeladas, resultando em um conjunto de casos de teste abstratos. Na concretização, os casos de teste abstratos gerados com base no modelo formal são transformados em casos de teste executáveis. Portanto, a etapa de concretização é responsável pela definição de como os casos de teste gerados são aplicados ao SUT (UTTING; LEGEARD, 2006). Por fim, na execução dos testes, os casos de teste gerados e concretizados serão submetidos ao SUT para que as funcionalidades modeladas sejam executadas e, posteriormente, os resultados obtidos sejam validados em relação ao que se espera como saídas corretas.

Neste trabalho, a técnica de modelagem *Event Sequence Graph* (ESG) é adotada para representar os comportamentos esperados do software a ser testado. A escolha pelo ESG como técnica de modelagem é apoiada pela fácil utilização e pelos relatos de experiência junto à TBM. Segundo Belli et al. (2006), um ESG é um grafo direcionado que pode ser utilizado para modelar eventos e funcionalidades de um software. A representação visual de um ESG é constituída por nós que representam os eventos e arestas que representam as sequências válidas de transição entre eventos (BELLI et al., 2006).

1.2. MOTIVAÇÃO

Devido à adoção de dispositivos móveis ao cotidiano que simplifica e colabora com a realização de tarefas profissionais e pessoais, empresas investem cada vez mais no desenvolvimento de hardware e software direcionados à mobilidade. Um relatório da *International Telecommunications Union* (ITU) apontava que 6,9 bilhões de assinaturas de serviços móveis seriam ativas ao final de 2014. Tal estatística equivale a aproximadamente 95% da população mundial com acesso a dispositivos móveis (ITU, 2014). O relatório divulgado em 2015 comprova a assertividade das estimativas anteriormente apresentadas e confirma que 6,9 bilhões de dispositivos móveis constavam como ativos até o início de 2015. Novas estimativas são apresentadas e se espera que a quantidade de assinaturas de

serviços que se baseiam em dispositivos móveis supere o valor de 7 bilhões em 2015 (ITU, 2015).

Em 2014, a venda de *smartphones* alcançou o total de 1,2 bilhões de unidades e, dessa forma, constata-se um aumento de 28,4% em relação ao ano anterior. A plataforma Android manteve a posição de líder como sistema operacional móvel com a aquisição de mais de um bilhão de dispositivos móveis, sendo acompanhada pelo iOS da Apple com pouco mais de 190 milhões de unidades vendidas (GARTNER, 2015a). A estabilidade da plataforma Android como líder no mercado de mobilidade também é ressaltada em relatórios da *International Data Corporation* (IDC) (IDC, 2015). Estatísticas atualizadas relatam que os índices de vendas de dispositivos móveis no terceiro trimestre de 2015 superaram o total de 352 milhões de unidades e, consequentemente, um acréscimo de 15,5% quando comparado ao mesmo período de 2014. Por fim, a somatória de vendas de dispositivos móveis para os três primeiros trimestres de 2015 supera um bilhão de unidades, sendo que 590 milhões operam sob a plataforma Android (GARTNER, 2015b; GARTNER, 2015c; GARTNER, 2015d).

A ampla aquisição de dispositivos móveis favorece e fomenta o desenvolvimento não somente de hardware, mas também o de software. Entretanto, empresas e desenvolvedores que atuam no desenvolvimento de aplicações móveis não consideram a etapa de teste ou a executam incorreta ou incompletamente (JOORABCHI et al., 2013). Distintos fatores se relacionam a dificuldades e desafios no contexto de mobilidade como, por exemplo, a execução de testes em diferentes plataformas e diversas configurações, a verificação de funcionalidades específicas como o uso de sensores e, por fim, limitações impostas à atividade de teste como melhores ferramentas para a execução e análise de testes.

No contexto de aplicações móveis, o teste de software apresenta características e particularidades que resultam em novos desafios que precisam ser superados e, para isso, pesquisas têm sido realizadas nas áreas de engenharia de software e mobilidade (WASSERMAN, 2010; MUCCINI et al., 2012; JOORABCHI et al., 2013; GAO et al., 2014; PICCO et al., 2014). Neste contexto, estudos investigam o uso de TBM para verificar aplicações móveis (RIDENE; BARBIER, 2011; TAKALA et al., 2011; JANICKI et al., 2012; LU et al., 2012; JENSEN et al., 2013; YANG et al., 2013; GRIEBE; GRUHN, 2014; LI et al., 2014; SCHWEIGHOFER; HERIČKO, 2014).

Dentre as técnicas de teste existentes, o TBM é uma técnica promissora para ser aplicada no contexto de mobilidade. Algumas vantagens são apontadas como, por exemplo, (i) geração automática de casos de teste, (ii) rastreabilidade de requisitos de sistema e critérios de teste e (iii) adequação a mudanças em requisitos (BLACKBURN et al., 2004; UTTING; LEGEARD, 2006; GRIESKAMP et al., 2011). A geração automática é sistematicamente efetuada por meio de ferramentas de apoio ao TBM. Dessa forma, casos de teste são gerados automaticamente a partir do modelo formal que descreve os requisitos do software a ser testado. O TBM pode ser aplicado para avaliar se as funcionalidades implementadas estão de acordo com os documentos e artefatos de especificação. Além disso, pode-se atualizar o modelo de teste para que acompanhe as evoluções do software quando necessário. Assim sendo, alterações no software se refletem no modelo e novos casos de teste são gerados, reduzindo o tempo e custo em relação ao teste manual. Por fim, considera-se importante investigar o reúso de modelos de teste para apoiar a verificação de distintas características e funcionalidades em aplicações móveis.

Particularmente no contexto de aplicações móveis, pesquisas têm sido conduzidas para investigar e relatar experiências com a adoção de TBM e de técnicas de modelagem que apoiam a representação, gráfica e/ou textual, de eventos (JÄÄSKELÄINEN et al., 2009a; RIDENE; BARBIER, 2011; LU et al., 2012; PÜSCHEL et al., 2012; JENSEN et al., 2013; YANG et al., 2013; AMALFITANO et al., 2014a; COSTA et al., 2014; GRIEBE; GRUHN, 2014; SCHWEIGHOFER; HERIČKO, 2014; XU et al., 2014). A escolha pela técnica baseada em ESG como recurso de modelagem formal para as funcionalidades de aplicações móveis é justificada devido à fácil utilização. Além disso, deve-se considerar que as aplicações móveis são compostas por telas e componentes de interface que disparam eventos e resultam na execução de determinadas funcionalidades a partir da interação com o usuário (JENSEN et al., 2013).

1.3. OBJETIVOS

O desenvolvimento de aplicações móveis demanda métodos de garantia de qualidade e um alto nível de confiabilidade, tendo em vista que a computação

móvel provê novos meios de comunicação, além de ser amparada pelos conceitos de mobilidade e portabilidade. O uso de dispositivos móveis na indústria também é um importante fator que orienta a necessidade de abordagens e ferramentas de apoio ao teste de aplicações móveis. Embora o processo de teste em mobilidade venha sendo explorado pela comunidade acadêmica em pesquisas recentes, mais contribuições podem ser vislumbradas e alcançadas por meio de TBM e ESG.

Nesta dissertação de mestrado, investiga-se a aplicabilidade de TBM com o apoio da técnica de modelagem ESG no contexto de aplicações móveis. Objetivos específicos são definidos para delinear as investigações propostas e contribuir com avanços no teste de software em aplicações móveis.

- ***Análise de pesquisas relevantes em teste de aplicações móveis***. Objetiva-se conduzir um mapeamento sistemático para identificar as pesquisas relevantes e classificar as principais características dos estudos selecionados que exploram o teste no contexto de aplicações móveis.

- ***Avaliação de TBM no contexto de aplicações móveis***. Objetiva-se investigar o TBM no contexto de mobilidade para avaliar a adoção a partir da definição padrão do processo, analisar os resultados e os desafios identificados e, por fim, verificar a efetividade na detecção de defeitos.

- ***Definição de abordagem fundamentada em TBM***. Objetiva-se propor uma abordagem específica e apoiada nos fundamentos de reúso de modelos de teste para contribuir com avanços no TBM em aplicações móveis. Dentre os progressos pretendidos, é importante ressaltar (i) a redução do esforço requisitado para a concretização de casos de teste, assim como (ii) a efetividade na detecção de defeitos a partir do teste de distintas características diretamente relacionadas ao contexto de mobilidade. Além disso, pretende-se realizar a concepção, a modelagem e a implementação de uma ferramenta de apoio à abordagem.

- ***Avaliação experimental de abordagem proposta***. Objetiva-se avaliar a adoção da abordagem e ferramenta propostas em um ambiente industrial de uma empresa multinacional.

1.4. ORGANIZAÇÃO

Esta dissertação de mestrado está organizada da seguinte forma:

Capítulo 2 – Revisão da literatura. Os conceitos de computação móvel e engenharia de software para aplicações móveis são apresentados, bem como a plataforma Google Android. Posteriormente, os fundamentos de teste de software são explorados. Maior ênfase é direcionada ao TBM, focando-se nas técnicas de modelagem, no processo de teste e nas vantagens e desvantagens. Trabalhos que investigam o uso de TBM na indústria são apresentados. O capítulo também descreve os conceitos de teste de aplicações móveis, relacionando-os com as dificuldades e particularidades existentes no contexto de mobilidade.

Capítulo 3 – Estudo de Mapeamento Sistemático. Um mapeamento sistemático é conduzido para investigar o teste de aplicações móveis e relatar as características de estudos relevantes. Pesquisas que adotam o TBM no contexto de mobilidade também são caracterizadas.

Capítulo 4 – TBM no contexto de aplicações móveis. Uma abordagem de TBM no contexto de aplicações móveis é avaliada. O capítulo descreve a configuração do estudo e as atividades realizadas durante a avaliação experimental, bem como os resultados obtidos e analisados. Limitações identificadas e lições aprendidas também são apresentadas.

Capítulo 5 – Abordagem de TBM para reúso de modelos. Uma abordagem para o reúso de modelos de teste no contexto de aplicações móveis é apresentada. A definição padrão da abordagem objetiva (i) reduzir o esforço requisitado na etapa de concretização do TBM, bem como (ii) testar distintas características diretamente relacionadas ao contexto de mobilidade. Para isso, duas estratégias são descritas: *Method Template* e *Edge Template*. Uma ferramenta, *Model-Based Test Suite For Mobile Applications* (MBTS4MA), foi projetada e desenvolvida para apoiar a abordagem proposta.

Capítulo 6 – Avaliação da abordagem proposta. Um estudo experimental conduzido para avaliar a abordagem e a ferramenta propostas no Capítulo 5 é

descrito. A configuração do estudo em um ambiente industrial de uma empresa multinacional de TI é apresentada e, posteriormente, realiza-se uma análise e discussão dos resultados obtidos. Limitações identificadas e lições aprendidas também são apresentadas.

Capítulo 7 – Conclusão. Por fim, contribuições da dissertação, limitações e trabalhos futuros são apresentados e delineiam possíveis tópicos de interesse para novas pesquisas.

2. REVISÃO DA LITERATURA

2.1. CONSIDERAÇÕES INICIAIS

Neste capítulo, é apresentada uma visão geral dos tópicos que fundamentam a base de pesquisa desta dissertação. A organização do capítulo se dará da seguinte forma. Na Seção 2.2, são apresentados os principais conceitos da computação móvel, destacando fatores que fazem com que essa se diferencie da engenharia de software tradicional. Na Seção 2.3, os fundamentos de teste de software são descritos. Na Seção 2.4, é caracterizado o Teste Baseado em Modelo (TBM) e seus detalhes são explorados. Na Seção 2.5, são apresentados os conceitos de teste de aplicações móveis, sumarizando as dificuldades e particularidades presentes.

2.2. COMPUTAÇÃO MÓVEL

Atualmente, existe uma infinidade de dispositivos que possibilitam aos usuários ter maior mobilidade na utilização de recursos de software. Esses dispositivos, representados principalmente pelos *smartphones*, *tablets* e *e-readers*, trouxeram uma ampla variedade de aplicações que são desenvolvidas com o apoio de tecnologias de computação móvel. Segundo Wasserman (2010), a construção de aplicações móveis tem sido simplificada, tendo em vista que os ambientes de desenvolvimento, as ferramentas de apoio e as plataformas de computação móvel são poderosas e robustas na atualidade.

A computação móvel (em Inglês, *Mobile Computing*) é um paradigma no qual usuários manipulam dispositivos portáteis com acesso a serviços de informação por meio de uma infraestrutura compartilhada, independentemente da localização física ou padrão de movimento (JING et al., 1999). No estágio atual, os dispositivos móveis fornecem, além do poder de computação, a praticidade de movimento e a interação sem fio com a Internet.

Segundo Dinh et al. (2011), a computação móvel constitui uma importante tendência que se confirma no desenvolvimento de novas tecnologias para as áreas comerciais e industriais. Satyanarayanan (1996) afirma que a computação móvel foi um importante passo para a evolução de tecnologias existentes, bem como a criação de novas demandas por desenvolvimento de software. Dentre os interesses providos pela computação móvel, destaca-se o acesso uniforme e imediato à informação e, consequentemente, a execução de tarefas de maneira ágil e independente da localização física do usuário (SATYANARAYANAN, 1996). Entretanto, fatores limitantes interferem diretamente no paradigma da computação móvel (GADDAH; KUNZ, 2003).

Gaddah e Kunz (2003) descrevem três fatores que caracterizam e limitam as tecnologias relacionadas à computação móvel: (i) variedade de dispositivos móveis, (ii) conectividade e (iii) mobilidade. Diferentemente de computadores pessoais, os dispositivos móveis atuais possuem uma grande variedade de configurações, porém o poder computacional disponibilizado por tais dispositivos é reduzido, tanto de processamento quanto de capacidade de armazenamento. A conectividade a redes de dados em contextos móveis é prejudicada pela alta taxa de erros, falta de segurança e indisponibilidade, prejudicando, dessa forma, o uso de dispositivos móveis e de aplicações. Por fim, a mobilidade deve ser transparente ao usuário, possibilitando que ocorram mudanças de ambientes e que o dispositivo móvel se conecte às redes de dados disponíveis quando o usuário se locomove de um ambiente a outra.

Além desses, a mobilidade dada aos dispositivos impõe outras restrições como telas limitadas, menor poder de processamento, mecanismos instáveis de comunicação e baixa disponibilidade de energia (MUCCINI et al., 2012). Essas restrições contribuíram para que sistemas operacionais específicos fossem desenvolvidos para *smartphones* e *tablets*, como, por exemplo, o Google Android, o Apple iOS e o Windows Phone. Dentre tais sistemas, a plataforma Android tem alcançado ampla e relevante aceitação (GARTNER, 2015b; GARTNER, 2015c; GARTNER, 2015d).

Um dispositivo móvel pode ser caracterizado levando-se em consideração particularidades e especialidades que o diferem de outros equipamentos computacionais. Na Figura 1 é ilustrada uma representação visual que caracteriza um dispositivo móvel com base em propriedades como conectividade, unidade de

persistência embarcada ou centralizada, integração a outros dispositivos computacionais, configurações de hardware, presença de sensores e atuadores, fonte de energia ou bateria e recursos de interface.

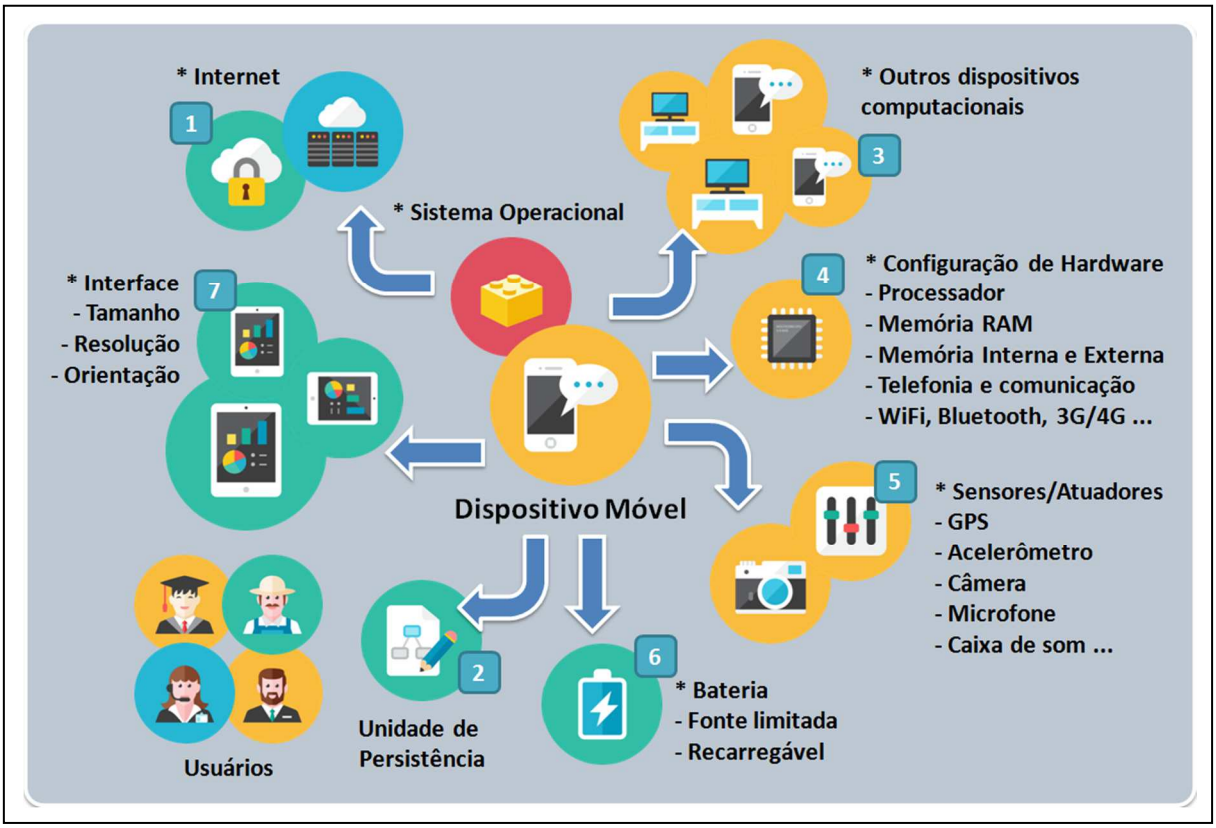

Figura 1 - Caracterização de dispositivo móvel
Fonte: Elaborada pelo autor

A conectividade (Item 1 na Figura 1) de um dispositivo móvel é fundamental à computação móvel, tendo em vista que aplicações podem consumir serviços disponibilizados na Internet. Tais serviços são funcionalidades que, quando requisitados, executam operações descentralizadas em relação ao dispositivo como, por exemplo, consultar uma base de dados central ou transferir dados a um servidor.

Para que o dispositivo móvel armazene dados, um mecanismo de persistência (Item 2 na Figura 1) deve estar embarcado junto a ele. Assim sendo, um repositório é acessado e manipulado por aplicações com a finalidade de armazenar dados que, posteriormente, serão extraídos, transferidos e analisados por outras aplicações. Entretanto, antes mesmo de as informações serem enviadas, pode ser necessário realizar uma sincronização entre o dispositivo e demais equipamentos computacionais, tais como computadores pessoais e outros dispositivos móveis

(Item 3 na Figura 1). Dessa forma, as aplicações e o próprio sistema operacional devem fornecer meios para que tal comunicação e compartilhamento aconteçam.

O funcionamento correto dos dispositivos móveis depende diretamente da configuração de hardware disponibilizada (Item 4 na Figura 1). Destacam-se o processador, memória RAM para processamento, memórias interna e externa para armazenamento, recursos de telefonia, comunicação e conectividade como rede sem fio, *Bluetooth* e redes de dados. É importante ressaltar que, apesar de as características serem parecidas a um computador pessoal, o poder de processamento e a configuração de hardware podem ser miniaturizados.

Além de possuir uma configuração de hardware projetada para dispositivos móveis, há também o uso de sensores e atuadores (Item 5 na Figura 1). Um sensor é um elemento de hardware que obtém informações referentes ao ambiente em que o dispositivo está sendo executado como, por exemplo, um *Global Positioning System* (GPS) e acelerômetro. Enquanto que um atuador é um componente que fornece um sinal ou estímulo externo como, por exemplo, o *flash* da câmera e as caixas de som (ABLESON et al., 2012; LECHETA, 2013).

A bateria e sua capacidade de duração (Item 6 na Figura 1) são, entre outras, características essenciais para o correto funcionamento de aplicações e dispositivos móveis. Diz-se que a bateria é uma fonte de energia limitada, porém recarregável. Portanto, os fabricantes devem avaliar o consumo a partir do uso de hardware e software.

Por fim, dispositivos móveis têm sido fabricados com distintas configurações de interface (Item 7 na Figura 1). O tamanho e a resolução das telas resultam na qualidade apresentada pela interface gráfica ao executar as aplicações. Outro ponto de atenção é a alternância da orientação, ou seja, a possibilidade da aplicação ser executada em modo paisagem (horizontal) e retrato (vertical).

2.2.1. Engenharia de software para aplicações móveis

O software que é desenvolvido seguindo o paradigma da computação móvel é coletivamente chamado de aplicações móveis (WASSERMAN, 2010; PICCO et al., 2014). Como quaisquer produtos de software, o desenvolvimento de

tais aplicações deve ser amparado por um processo de engenharia de software. Além disso, as aplicações têm sido utilizadas com sucesso em diversos domínios como, por exemplo, entretenimento, sistemas financeiros, educação, indústrias e sistemas críticos. Wasserman (2010) identificou alguns fatores que interferem na engenharia de software para aplicações móveis:

- **Potencial interação com outras aplicações**. As aplicações móveis podem interagir com outras aplicações instaladas no mesmo dispositivo ou disponibilizadas na Internet. Há a possibilidade de distribuir o processamento pesado para um computador externo como, por exemplo, na nuvem (WASSERMAN, 2010; PICCO et al., 2014);
- **Manipulação de sensores**. Os dispositivos móveis modernos contêm uma variedade de sensores como acelerômetro, GPS, microfone e câmeras. Os sensores podem gerar dados com ruído e precisam ser corretamente manipulados para resultar em uma adequada utilização pelas aplicações móveis (WASSERMAN, 2010; PICCO et al., 2014);
- **Aplicações nativas e híbridas**. O desenvolvimento de aplicações móveis deve considerar que a instalação e execução localmente nos dispositivos móveis. Entretanto, as funcionalidades podem consumir dados e serviços de redes de telefonia ou da Internet;
- **Famílias de plataformas de hardware e software**. O desenvolvedor deve se preocupar com quais plataformas que serão utilizadas e quais os dispositivos que serão suportados;
- **Segurança**. A conexão com redes externas e a constante instalação de novas aplicações apresentam lacunas de segurança que podem ser exploradas em possíveis ataques;
- **Telas de usuário**. As aplicações móveis possuem características específicas como limitações no tamanho da tela e formas diferenciadas de interação com a aplicação como, por exemplo, *touchscreen* ou teclado virtual; e
- **Consumo de energia**. Funcionalidades da aplicação móvel podem influenciar o nível de consumo de energia do dispositivo. Dessa forma, recursos que demandam um alto consumo de energia como, por exemplo, GPS e câmera diretamente interferem na autonomia do dispositivo móvel.

Picco et al. (2014) investigam as mudanças na engenharia de software para aplicações móveis desde 2000 e, além de avaliarem o que se mostrava como tendência naquela época, apresentam desafios e oportunidades que se relacionam ao contexto de mobilidade. Dentre os assuntos que são sugeridos como temas para pesquisas, os autores ressaltam:

- **Computação antecipatória**. As aplicações móveis constantemente analisarão informações do ambiente e do próprio usuário para predizer ou recomendar ações e atividades. Por exemplo, aplicações podem auxiliar na (i) definição de séries de exercícios a partir de registros diários de condicionamento físico do usuário ou na (ii) recomendação de um trajeto comum em relação a seus contatos que se exercitam no mesmo horário;
- **Aplicações móveis para robótica**. Lojas inteligentes e empresas de correspondência utilizarão robôs e quadricópteros para, por exemplo, realizar a entrega de produtos. Portanto, as aplicações móveis poderão se fundamentar em conceitos de inteligência computacional e artificial, apoiando atividades executadas por robôs autônomos;
- **Interfaces neurais**. Além do uso de robôs, o contexto de mobilidade permitirá que as redes neurais se integrem a aplicações móveis, possibilitando o controle e execução de tarefas com estímulos do cérebro. Os autores exemplificam a adoção de redes neurais em computação móvel para auxiliar autistas na interação com professores e outros alunos, bem como notificar o professor de que o aluno autista está desconcentrado durante a explanação da aula; e
- **Privacidade no contexto de mobilidade**. Atualmente, aplicações móveis registram e capturam um grande conjunto de informações acerca do que ocorre no cotidiano e, por isso, a privacidade deve ser considerada como uma característica essencial no contexto de mobilidade. Dessa forma, espera-se que (i) não exista compartilhamento de dados privados (ii) as aplicações móveis sejam capazes de identificar mudanças no contexto e, por exemplo, deixar de exibir informações pessoais no monitor de uma sala de reunião assim que outro funcionário adentra o espaço.

2.2.2. Plataforma Google Android

Dentre as tecnologias relacionadas à computação móvel, a plataforma Google Android é a primeira de código fonte aberto que possibilita o desenvolvimento de aplicações para dispositivos móveis (ABLESON et al., 2012). Inclui um sistema operacional baseado no *kernel* do Linux, um *middleware* que integra o hardware e software do dispositivo, aplicações básicas e ferramentas de apoio a desenvolvedores (ABLESON et al., 2012; LECHETA, 2013).

Desenvolvida inicialmente pela Android Inc. e adquirida pela Google em 2005, a plataforma Android surgiu com a pretensão de acompanhar o mercado e a evolução tecnológica no contexto de mobilidade. Na atualidade, a Google e empresas de telefonia, tecnologia da informação (TI) e fabricantes de hardware e software investem na plataforma por meio de um consórcio chamado de *Open Handset Alliance* (OHA) (OPEN HANDSET ALLIANCE, 2015). A OHA é composta, entre outras, por: HTC, LG, Motorola, Samsung, Sony Ericsson, Toshiba, Strint Nextel, China Mobile, T-Mobile, ASUS, Intel e Garmun. O objetivo principal é fomentar o desenvolvimento de uma plataforma única, aberta, moderna e flexível (OPEN HANDSET ALLIANCE, 2015). Segundo Mednieks et al. (2012) e Lecheta (2013), o fato da plataforma Android possuir código fonte aberto e não ser mantido por apenas uma empresa contribui para seu aperfeiçoamento, uma vez que desenvolvedores podem colaborar com melhorias no projeto como um todo, acrescentando novas funcionalidades ou simplesmente corrigindo defeitos.

As aplicações desenvolvidas em Android são escritas em Java e compiladas em *bytecodes*[1] que, em um processo de otimização, são interpretados por uma máquina virtual chamada Dalvik (ABLESON et al., 2012). Antes de a máquina virtual Dalvik interpretar os *bytecodes*, esses são convertidos ou traduzidos para uma representação similar, porém otimizada, resultando em arquivos *dex* (em Inglês, *dex files*) (MEDNIEKS et al. 2012). Tal conversão se deve por três fatores fundamentais: (i) otimização de desempenho das aplicações em ambientes de pouca memória ou recursos de hardware em geral, (ii) execução de aplicações na máquina virtual do Android e, dessa forma, livre das limitações de licenciamento da

[1] *Bytecode* é uma representação intermediária de aplicações Java após o processo de compilação e é interpretada por uma máquina virtual Java, tornando possível sua execução em qualquer plataforma e sistema operacional (ABLESON et al., 2012).

Oracle e, por fim, (iii) possibilidade de melhorias na máquina virtual pela comunidade (ABLESON et al., 2012; MEDNIEKS et al. 2012).

Além de o desenvolvimento na plataforma Android fazer uso da tecnologia e recursos de Java, a plataforma da Google provê componentes que podem ser utilizados em aplicações móveis (LECHETA, 2013). Segundo Ableson et al. (2012), há quatro principais tipos de componentes Android para o desenvolvimento de uma aplicação móvel: (i) *Activity*, (ii) *Service*, (iii) *Broadcast Receiver* e (iv) *Content Provider*.

O componente *Activity* representa a camada de interface e interação com o usuário. Por padrão, entende-se que, para cada tela que uma aplicação móvel em Android exibe, uma classe Java é implementada para representar uma *Activity*. Dessa forma, é por meio de uma *Activity* que eventos são acionados pelo usuário e esses são capturados pela aplicação. Diferentemente de uma *Activity*, um *Service* é utilizado quando se torna necessária a execução de uma instrução ou processo de vida longa como, por exemplo, a sincronização de dados em segundo plano. Assim como o componente *Service*, um *Broadcast Receiver* não possui interface com o usuário, porém deve ser utilizado para capturar e responder a eventos lançados pelo sistema e/ou por aplicações. A manipulação de dados e a possibilidade de expô-los a outras aplicações e serviços podem ser alcançadas pelo componente *Content Provider*. O componente define um conjunto métodos que permitem operações de leitura e/ou escrita entre distintas aplicações que acessam um mesmo repositório de dados (ABLESON et al., 2012; MEDNIEKS et al. 2012; LECHETA, 2013).

Um fato fundamental do desenvolvimento em Android é que uma aplicação conterá, ao menos, uma *Activity*, *Service*, *Broadcast Receiver* ou *Content Provider*. Os componentes devem estar declarados em um arquivo de configuração chamado de AndroidManifest (ABLESON et al., 2012). O AndroidManifest contém informações referentes aos componentes anteriormente descritos, bem como o nome e pacote da aplicação, permissões e bibliotecas necessárias e definição de versões do sistema operacional (ABLESON et al., 2012; LECHETA, 2013).

Ambiente de desenvolvimento. Assim como em aplicações tradicionais, o desenvolvimento de aplicações Android é apoiado por um ambiente de desenvolvimento integrado (em Inglês, *Integrated Development Environment* – IDE). O Android Studio (ANDROID, 2015) é uma IDE que fornece mecanismos para

projetar, codificar, depurar e testar aplicações Android. Além disso, o Android Studio possibilita executar aplicações por meio de emuladores de diversos dispositivos disponíveis no mercado.

Para que o ambiente de desenvolvimento seja corretamente configurado, deve-se possuir o Android SDK. Segundo Lecheta (2013), o Android SDK fornece bibliotecas e recursos necessários para construir e testar aplicações Android. Além das ferramentas disponibilizadas, uma documentação descreve os pacotes e classes que podem ser utilizados no desenvolvimento com a plataforma Android (ABLESON et al., 2012; ANDROID, 2015). Ableson et al. (2012), Mednieks et al. (2012) e Lecheta (2013) destacam os principais recursos providos pelo Android SDK: (i) ambiente de desenvolvimento Java, (ii) depuração no nível do código fonte, (iii) *Dalvik Debug Monitor Server*, (iv) gerenciamento de sistema de arquivos do emulador e (v) *log* das aplicações e do sistema operacional Android.

O ambiente de desenvolvimento da plataforma Android se baseia na plataforma Java (JAVA, 2015) e fornece uma *Application Programming Interface* (API) completa para o desenvolvimento orientado a objetos com recursos de computação distribuída e acesso a redes de dados além de ser uma das principais tecnologias na atualidade (MEDNIEKS et al. 2012; LECHETA, 2013).

Segundo Android (2015), o *Dalvik Debug Monitor Server* (DDMS) é uma ferramenta utilitária integrada ao ambiente de desenvolvimento que possibilita gerenciar eventos em emuladores, também chamados de *Android Virtual Device* (AVD). Por meio do DDMS, é possível capturar imagens de telas de um AVD, acessar informações de processos e memória, simular chamadas telefônicas, envio de mensagens de texto, dados de geolocalização e outras ações.

Apesar de um AVD exercer o papel de emulador de um ambiente Android, o sistema de arquivos pode ser gerenciado por meio do ambiente de desenvolvimento. Dessa forma, arquivos podem ser criados, manipulados e gravados em um cartão de memória virtual do emulador (ABLESON et al., 2012).

Outro recurso importante é a possibilidade de registrar eventos em *logs* de aplicações e do sistema operacional. O uso de *logs* é essencial no desenvolvimento de software, tornando possível o registro de informações, alertas e/ou erros durante a execução de aplicações ou serviços. Posteriormente, uma análise pode ser feita para coletar e visualizar dados como, por exemplo, quantidade

de memória utilizada, número de processos ativos ou causa de um defeito na aplicação (MEDNIEKS et al. 2012; LECHETA, 2013).

Por fim, o ambiente de desenvolvimento apoia o conceito de perspectivas, no qual o *layout* da tela possui um conjunto de janelas e ferramentas relacionadas, simplificando ainda mais o uso de recursos da plataforma Android como a criação de classes e interfaces, arquivos de tela, *scripts* de banco de dados e a execução e depuração de aplicações (ABLESON et al., 2012).

2.3. FUNDAMENTOS DE TESTE DE SOFTWARE

O processo de desenvolvimento de software compreende uma coleção de atividades e passos nos quais, independentemente das técnicas, métodos e ferramentas utilizados, defeitos ainda podem existir (MALDONADO, 1991; HARROLD, 2000). Para minimizar a ocorrência de defeitos, atividades de Verificação, Validação e Teste (VV&T) têm sido incorporadas ao processo de desenvolvimento com o objetivo de garantir a qualidade do produto de software. A atividade de teste é um processo em que um programa é executado com a intenção de que defeitos sejam detectados. Portanto, o teste de software é caracterizado como elemento crítico na revisão de especificações, modelagens e código fonte (MALDONADO, 1991; MYERS et al., 2004; PRESSMAN, 2005). Além disso, provê subsídios acerca da confiabilidade do software em conjunto a demais atividades como, por exemplo, revisões formais e técnicas de especificação e de validação (HARROLD, 2000; MALDONADO et al., 2004).

O padrão IEEE 610.12 de 1990 apresenta as definições para os seguintes termos: defeito (em Inglês, *fault*) – passo, processo ou definição de dados incorreta como, por exemplo, uma instrução ou comando incorreto; engano (em Inglês, *mistake*) – ação humana que produz um resultado incorreto como, por exemplo, uma ação incorreta realizada pelo desenvolvedor; erro (em Inglês, *error*) – a diferença entre o valor obtido e o esperado, ou seja, qualquer valor intermediário incorreto ou resultado inesperado na execução; e falha (em Inglês, *failure*) – produção de uma saída incorreta em relação à especificação (IEEE, 1990).

O padrão ainda define caso de teste (em Inglês, *test case*) como um conjunto de dados de entrada, condições de execução e resultados esperados elaborado para alcançar um objetivo específico como, por exemplo, executar um fluxo no software ou verificar sua conformidade em relação a um determinado requisito. Dessa forma, um caso de teste também pode ser definido como uma sequência de eventos a ser aplicada e/ou analisada. Assim, o termo caso de teste pode ser referenciado como sequência de teste (em Inglês, *test sequence*). Uma coleção de casos de teste é denominada conjunto de teste (em Inglês, *test suite*) (IEEE, 1990).

A atividade de teste pode ser dividida em quatro etapas e devem ser desenvolvidas ao longo do processo de desenvolvimento de software (BEIZER, 1990; IEEE, 1990; PRESSMAN, 2005):

- **Planejamento de testes**. Um plano de teste é criado e integra objetivos e critérios de finalização que julgam quando uma fase de teste fora completada, cronogramas, distribuição de responsabilidades, padrões para casos de teste, identificações das ferramentas, configuração de hardware, estratégia de integração, rastreamento do progresso, procedimentos para depuração e teste de regressão (MALDONADO, 1991);
- **Projeto de casos de teste**. As técnicas de testes são utilizadas para a elaboração de casos de teste efetivos na detecção de defeitos como, por exemplo, teste funcional e teste estrutural (MYERS et al., 2004);
- **Execução de testes**. Os procedimentos especificados no plano de teste são aplicados;
- **Coleta e avaliação dos resultados de testes**. Os resultados obtidos são registrados, organizados e analisados a partir da execução de testes.

Em geral, a atividade de teste deve ser aplicada em três fases distintas classificadas como teste de unidade, teste de integração e teste de sistema (MALDONADO et al., 2004; PRESSMAN, 2005):

- **Teste de unidade**. A verificação e validação são realizadas na menor unidade do projeto de software, podendo ser chamada de componente, módulo do software, método, classe ou até mesmo função no paradigma procedimental;

- **Teste de integração**. Os testes são executados paralelamente à etapa de integração das partes unitárias do software com a finalidade de revelar defeitos associados às interfaces de módulos integrados do produto de software (AMMANN; OFFUTT, 2008);
- **Teste de sistema**. Os testes são aplicados após a etapa de integração e objetiva identificar defeitos no sistema como um todo por meio de vários tipos de testes como de recuperação, segurança, estresse e desempenho.

2.3.1. Técnicas de teste

Para que a atividade de teste alcance o objetivo de revelar defeitos, é necessário verificar se a qualidade do teste realmente é eficaz e de baixo custo (MYERS et al., 2004). As técnicas e critérios de teste definem uma abordagem sistemática e fundamentada que constituem um mecanismo que pode auxiliar na avaliação da qualidade e adequação da atividade de teste (ZHU, 1997). Os critérios de teste podem ser classificados a partir de técnicas em (i) teste funcional, (ii) teste estrutural e (iii) teste baseado em defeitos.

Técnica de Teste Funcional. O teste funcional é uma técnica de teste em que o testador considera o software como uma caixa-preta, ou seja, não possui conhecimento acerca do comportamento interno e da estrutura do programa (MYERS et al., 2004). Dessa forma, a geração dos casos de teste é realizada com base nas funcionalidades da aplicação e dos documentos que a especificaram.

Técnica de Teste Estrutural. O teste estrutural é uma técnica de teste em que o testador possui conhecimento ou acesso à parte lógica do software para apoiar a derivação dos casos de teste. Por esse motivo, o teste estrutural é considerado como caixa-branca, pois não apenas elementos que especificam as funcionalidades podem ser utilizados na geração dos casos de teste, como artefatos criados durante o desenvolvimento do software a ser testado (MYERS et al., 2004). Por exemplo, código fonte, arquivos de interface e *scripts* de banco de dados são considerados como artefatos para o teste estrutural.

Técnica de Teste Baseado em Defeitos. O teste baseado em defeitos utiliza o conhecimento acerca da existência de defeitos comuns no processo de desenvolvimento de software. A partir da experiência adquirida, os testadores derivam ou criam os requisitos de teste (MYERS et al., 2004). A técnica de teste baseado em defeitos é, entre outros casos, incorporada à Semeadura de Erros (em Inglês, *Error Seeding*) (BUDD, 1981) e Teste de Mutação (em Inglês, *Mutation Testing*) (DEMILLO, 1987).

2.4. TESTE BASEADO EM MODELO

A área de teste de software é abundante em técnicas que podem ser aplicadas durante o processo de desenvolvimento com o propósito de revelar defeitos no software (BEIZER, 1990; MYERS et al., 2004; AMMANN; OFFUTT, 2008). Em meio a essas técnicas, abordagens são definidas para a geração automática de casos de teste com base em um modelo comportamental ou estrutural, também chamado de modelo de teste, do software a ser testado.

Segundo Sinha e Smidts (2006), tal abordagem é conhecida como Teste Baseado em Modelo (TBM). A partir do modelo que representa o comportamento esperado do software ou da combinação de dois ou mais modelos, casos de teste podem ser automaticamente gerados e executados no software.

Utting e Legeard (2006) definem o TBM como uma abordagem para a automação do projeto de testes funcionais, apesar de que há casos em que detalhes da implementação do SUT devem ser utilizados para a derivação do modelo. Portanto, diferentemente do processo manual da técnica de teste funcional que se baseia no projeto de testes a partir de documentos e especificações de requisitos, a abordagem de TBM objetiva a elaboração de modelos que definem o comportamento esperado do SUT. Posteriormente, ferramentas de apoio são utilizadas para gerar testes de maneira automatizada a partir de modelos de teste. O processo de TBM pode ser aplicado nas fases de teste de unidade, de integração e de sistema (UTTING; LEGEARD, 2006).

2.4.1. Técnicas de modelagem

Para a construção do modelo de teste, algumas técnicas de modelagem podem ser utilizadas para expressar os requisitos e funcionalidades do software a ser testado. Espera-se que a técnica de modelagem adotada para o TBM seja formal, ou seja, bem definida sintática e semanticamente, resultando em testes mais eficientes e efetivos (HIERONS et al., 2009). Segundo Utting e Legeard (2006), um modelo é considerado formal desde que seja preciso e não ambíguo para representar o comportamento de uma maneira que possa ser facilmente compreendida e manipulada por ferramentas.

O TBM pode ser utilizado em conjunto a uma extensa série de técnicas de modelagem, incluindo (i) modelos baseados em cenários como *Message Sequence Charts* ou Diagrama de Casos de Uso, (ii) modelos orientados a estados e (iii) modelos orientados a processos (ORSO; ROTHERMEL, 2014). Além disso, Orso e Rothermel (2014) afirmam que os modelos podem corresponder ou derivar de modelos da *Unified Modeling Language* (UML), e até mesmo de modelos de interface, como Grafos de Fluxo (em Inglês, *Flow Graph*). Dentre as técnicas de modelagem utilizadas em TBM, pesquisas destacam as Máquinas de Estados Finitos (MEFs) (LEE; YANNAKAKIS, 1996), as Máquinas de Estados Finitos Estendidas (MEFEs) (WANG; LIU, 1993; PETRENKO et al., 2004), o *Labelled Transition System* (LTS) (TRETMANS, 1996), o *Symbolic Transition System* (STS) (FRANTZEN et al., 2006), a AsmL (BARNETT et al., 2003) e a UML (BOUQUET et al., 2007; WANG et al., 2013). Além das técnicas listadas, os requisitos podem ser modelados por meio de *Event Sequence Graphs* (ESGs).

Event Sequence Graph **(ESG)**. Um ESG é uma técnica utilizada para modelar interações de eventos do software a ser testado e é constituído por nós que representam os eventos enquanto que as arestas representam sequências válidas entre esses eventos (BELLI; LINSCHULTE, 2008; YUAN et al., 2011). Na Figura 2 é ilustrado um modelo ESG que define os eventos e sequências de interação para copiar, recortar e colar arquivos. Existem dois nós especiais, “[”e “]”, que representam os nós de entrada e de saída, respectivamente. Por fim, duas suposições devem ser verdadeiras acerca do grafo: (i) todo nó deve ser alcançável,

por meio de uma sequência, do nó de entrada, e (ii) o nó de saída deve ser alcançável, por meio de uma sequência, de qualquer evento (BELLI et al., 2006).

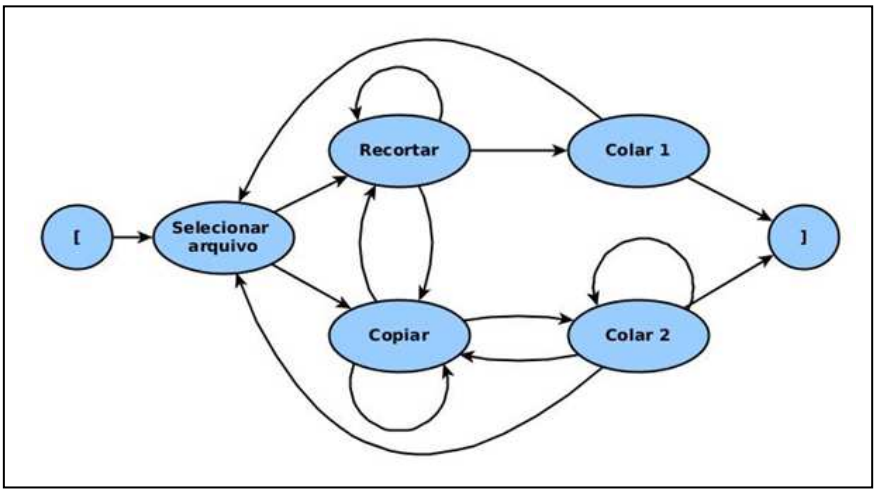

Figura 2 - Exemplo de um modelo ESG
Fonte: Adaptada de Endo (2013)

Utilizando-se um modelo de teste, como na Figura 2, algoritmos propostos podem ser aplicados para a geração de *Complete Event Sequences* (CESs). Uma CES é uma sequência linear de eventos, que começa em "[" e terminada em "]", gerada para cobrir todos os arcos do modelo de teste e auxiliar os desenvolvedores e testadores na obtenção dos fluxos que a aplicação pode assumir (LINSCHULTE, 2013). Um exemplo de duas CESs geradas a partir do modelo elaborado na Figura 2 é apresentado a seguir.

```
Recortar: [, Selecionar arquivo, Recortar, Copiar, Recortar, Recortar,
Colar 1, Selecionar arquivo, Recortar, Colar 1, ]

Copiar: [, Selecionar arquivo, Copiar, Recortar, Copiar, Copiar, Colar 2,
Copiar, Colar 2, Colar 2, Selecionar arquivo, Copiar, Colar 2, ]
```

2.4.2. Processo de Teste Baseado em Modelo

Pesquisas na área de TBM sugerem uma divisão em quatro passos principais, categorizando-os em (i) modelagem, (ii) geração de testes, (iii)

concretização e (iv) execução dos testes (PRETSCHNER; PHILIPPS, 2004; BOUQUET et al., 2006; EL-FAR; WHITTAKER, 2011; ENDO, 2013). Na Figura 3 é ilustrado o processo de TBM, apresentando os passos e artefatos envolvidos.

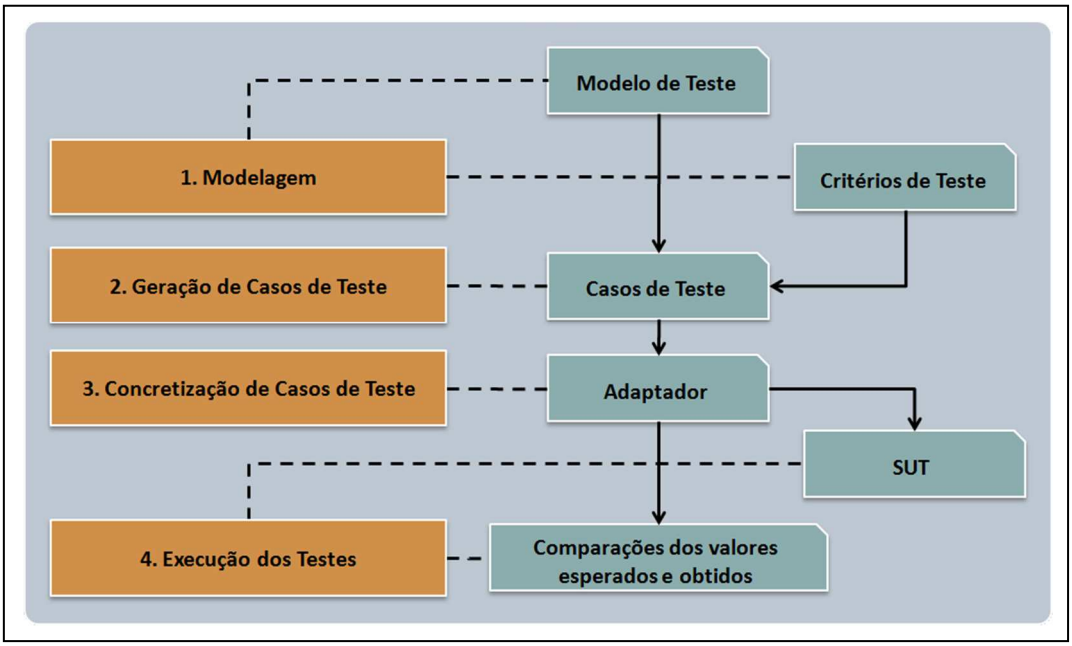

Figura 3 - Processo de Teste Baseado em Modelo
Fonte: Adaptada de Utting et al. (2006)

Na modelagem, o testador utiliza o entendimento acerca do software para criar um modelo de teste. Os requisitos são fontes de informações que tornam possível uma melhor compreensão das funcionalidades do software a ser testado. Além disso, o produto de software está contido em um ambiente que apresenta diversos fatores, como sistemas operacionais, outras soluções computacionais, diferentes tipos de bibliotecas e outras características. Dessa forma, o testador necessita compreender tanto do software a ser testado quanto do ambiente de execução do teste. Nesse passo, recomenda-se a criação de modelos de teste com base nos requisitos com o objetivo de maximizar a independência entre o modelo e o software a ser testado (UTTING et al., 2006).

A geração do teste depende da técnica de modelagem adotada para representar o modelo de teste. Segundo El-Far e Whittaker (2011), as técnicas de modelagem devem possuir propriedades que tornam a geração do teste menos custosa, além de promover a automação de tal atividade. Nesse passo, uma ferramenta é essencial para apoiar a geração automática dos casos de teste. O modelo de teste é submetido como entrada, bem como o critério de seleção de

teste, e a ferramenta gera, como artefato resultante, um conjunto de casos de teste. Nesse momento, os casos de teste gerados ainda são abstratos e não executáveis por estarem em um nível de abstração diferente do software a ser testado.

O terceiro passo, responsável pela concretização, envolve a transformação dos casos de teste abstratos em casos de teste executáveis no software a ser testado. Portanto, os casos de teste abstratos e derivados do modelo de teste necessitam ser concretizados por meio de adaptadores (PRETSCHNER; PHILIPPS, 2004; UTTING; LEGEARD, 2006). Um adaptador em TBM é um componente de software que transforma as entradas e saídas entre os níveis de abstração de modelo de teste e do software a ser testado, tornando possível a execução de um caso de teste abstrato.

A execução do teste, último passo do TBM, é a aplicação do caso de teste concretizado no software a ser testado. Após a execução do teste, os resultados são analisados e ações corretivas tomadas. Uma verificação automática com ferramentas de apoio pode ser realizada para os modelos de teste que especificam valores de entrada e de saída.

2.4.3. Vantagens e desvantagens

Segundo Blackburn et al. (2004), Utting e Legeard (2006) e Grieskamp et al. (2011), há uma série de benefícios que podem ser obtidos quando o TBM é corretamente aplicado durante o processo de desenvolvimento:

- **Geração automática de casos de teste.** A adoção de modelos de teste e o uso de ferramentas de apoio tornam possível a geração automática e sistemática de casos de teste;
- **Detecção de defeitos.** Os estudos conduzidos pela indústria e centros acadêmicos têm avaliado, positivamente, a efetividade na detecção de defeitos quando o TBM é comparado ao teste tradicional (UTTING; LEGEARD, 2006);
- **Redução do tempo e custo de teste.** As pesquisas que comparam o uso de TBM e testes manuais apresentam resultados favoráveis à primeira abordagem

quando tempo e custo de teste são avaliados (DALAL et al., 1999; FARCHI et al. 2002; PRETSCHNER et al., 2005);

- **Melhoria da qualidade de teste.** Quando realizado manualmente, a qualidade do teste é dependente da habilidade do testador, o processo não é repetível, a rastreabilidade entre os casos de teste e os requisitos não são facilmente alcançados;
- **Detecção de defeitos em requisitos.** A modelagem de testes nas fases iniciais de um processo de desenvolvimento de software pode contribuir com o refinamento de requisitos que possuem problemas de especificação. Dessa forma, o TBM também pode ser utilizado na verificação dos requisitos de software;
- **Evolução dos requisitos de software.** Durante o processo de desenvolvimento, é comum que mudanças sejam necessárias para que o software evolua. Com TBM, tais mudanças são mais facilmente acomodadas por alterações no modelo de teste e regeração dos testes; e
- **Rastreabilidade.** Segundo Utting e Legeard (2006), a rastreabilidade é a capacidade de relacionar casos de teste com o modelo elaborado, o critério de seleção ou os requisitos definidos do sistema. Portanto, o TBM possibilita explicar a razão para um determinado caso de teste, selecionar um subconjunto de testes para uma mudança no modelo e, por fim, identificar quais requisitos foram testados ou não.

Embora a abordagem de TBM acrescente inúmeros benefícios ao processo de desenvolvimento e verificação de software, desvantagens envolvidas devem ser listadas (UTTING; LEGEARD, 2006; EL-FAR; WHITTAKER, 2011):

- **Uso inapropriado.** Partes de um software podem ser difíceis de modelar e, em contextos específicos, o teste manual pode ser mais efetivo. Entretanto, o testador deve possuir a experiência necessária para tomar essa decisão;
- **Tempo necessário para analisar a falha em um caso de teste.** Quando um teste falha, o defeito pode estar no software, no modelo de teste ou até mesmo no adaptador. Além disso, uma sequência de teste pode ser complexa, extensa e até menos intuitiva, dificultando ainda mais a descoberta do defeito;

- **Métricas sem sentido.** TBM pode resultar em uma grande quantidade de casos de teste. Portanto, métricas baseadas no número de casos de teste não são úteis, sendo necessário selecionar métricas como cobertura de código, de requisitos e do modelo (UTTING; LEGEARD, 2006);
- **Habilidade do testador.** O testador deve possuir habilidades em modelagem e no uso da técnica de modelagem de teste adotada. Com isso, geram-se custos com treinamentos além de ser comum uma curva de aprendizagem inicial mais acentuada; e
- **Explosão de estados.** O uso de modelos para representar cenários de estados pode requerer um detalhamento acerca dos eventos definidos, resultando em modelos complexos e difíceis de serem mantidos.

Com a finalidade de contribuir com a resolução dos problemas identificados, podem-se utilizar ferramentas de apoio, estabelecer um processo bem definido e aplicar um treinamento adequado para a técnica de modelagem adotada.

2.4.4. Teste Baseado em Modelo na indústria

Samih et al. (2014) investigam a aplicação de TBM em um processo de desenvolvimento de software baseado em linha de produtos de software (em Inglês, *Software Product Line* – SPL) para gerar modelos formais e garantir a qualidade de funcionalidades que podem ser incorporadas ou removidas de um software em produção. A abordagem foi avaliada em uma indústria no domínio aeroespacial. A avaliação experimental foi realizada na *Airbus Defence & Space*, uma empresa internacional que desenvolve sistemas de aviação para treinamento de pilotos. A partir das funcionalidades e características selecionadas na SPL, informações podem ser obtidas para apoiar a geração dos modelos formais utilizados no processo de TBM. Como resultado, os autores identificaram que a adoção de TBM contribuiu positivamente na geração de casos de teste em um cenário de SPL.

Dias-Neto e Travassos (2014) apresentam um estudo quanto às variações de TBM, num total de 219 identificadas, descrevem suas características e

finalidades e propõem uma técnica que auxilie na seleção e combinação entre elas. Para comprovar a efetividade, os autores validam a abordagem proposta em projetos industriais de uma organização internacional, combinando variações de TBM quando mais de uma técnica é necessária ou aplicável. Além disso, os autores conduziram uma comparação da abordagem proposta com outras existentes. Por fim, o mecanismo de seleção e combinação de variações de TBM se confirmou como uma alternativa recomendada, tendo em vista que a avaliação experimental resultou no correto uso dos processos de TBM em conjunto.

Entin et al. (2012) afirmam que abordagens têm sido propostas para a geração automática de casos de teste quanto à validação de interface gráfica, destacando a técnica de TBM. A pesquisa avalia TBM como parte de um processo de desenvolvimento de software em um cenário industrial que adota *Scrum*, acrescentando a verificação do software por modelos a cada *sprint*. A avaliação experimental foi realizada em uma empresa internacional chamada OMICRON que atua na indústria de geração de energia elétrica. O estudo experimental conduzido resultou na identificação de desafios que indicam a necessidade de um processo rigoroso para a modelagem, bem como a reutilização de modelos quando o TBM é aplicado em um contexto de metodologia ágil. Outras considerações ressaltadas foram a dificuldade encontrada ao incorporar uma abordagem acadêmica em um ambiente industrial e a escolha de uma plataforma de técnicas de TBM para uso diário pelas equipes de *Scrum*.

Marijan (2012) reporta dois experimentos realizados com o uso de TBM na indústria e discute as perspectivas para adoção em ambientes reais. Na pesquisa, aplica-se TBM como técnica de teste para dois contextos industriais: (i) um sistema de vídeo-conferência e (ii) um ambiente para detecção de movimentos sísmicos. Após a realização de dois estudos de caso, o autor afirma que a adoção de TBM na indústria envolve tanto a parte técnica quanto a organizacional da empresa. As dificuldades encontradas são discutidas e relacionadas ao uso de ferramentas que apoiam o processo de teste e os investimentos necessários para treinamento dos envolvidos.

Ali et al. (2011) investigam o uso de linguagens e padrões como UML e *Object Constraint Language* (OCL) para formalizar, de maneira bem definida, os modelos de teste elaborados junto ao TBM. A pesquisa propõe um conjunto de heurísticas baseadas em restrições OCL para auxiliar na geração de dados e

automatizar o uso de TBM em ambientes industriais. A abordagem foi avaliada em um sistema real de vídeo-conferência chamado *Saturn*, desenvolvido pela *Cisco Systems*. Os autores afirmam que a abordagem proposta demonstrou efetividade e eficiência na geração de dados de teste a partir de restrições e que, mesmo em situações complexas, a geração de dados foi completada em 27 minutos.

Grieskamp et al. (2011) realizam uma avaliação experimental da adoção de TBM no ambiente industrial da Microsoft, aplicando a técnica de teste no processo de verificação da qualidade em documentos de especificação de protocolos de servidores. Ao final, os autores destacam um ganho de 42% de produtividade em testes com um processo de TBM quando comparado a técnicas tradicionais de teste.

2.5. TESTE DE APLICAÇÕES MÓVEIS

Apesar dos avanços na pesquisa em teste de software, o surgimento de tecnologias e plataformas contribui na ampliação ou criação de novos campos de estudo. Dentre os temas recentes, destaca-se o exponencial uso de dispositivos móveis em atividades de âmbito profissional e corporativo, bem como em domínios críticos como, por exemplo, nas áreas de saúde, financeiro e industrial (MUCCINI et al., 2012). Nesse contexto, o teste de aplicações móveis é uma importante área de pesquisa e tem recebido uma atenção especial da comunidade científica (WASSERMAN, 2010; MUCCINI et al., 2012; GAO et al., 2014).

Devido ao amplo número e diversidade de usuários e à utilização em sistemas críticos, defeitos na operação dessas aplicações móveis podem acarretar em sérias consequências humanas e econômicas (MUCCINI et al., 2012). A evolução constante de dispositivos e tecnologias móveis ao longo da última década motivou mudanças na atividade de teste (PICCO et al., 2014).

Segundo Picco et al. (2014), a adoção da computação móvel em ambientes críticos colabora para que novas demandas de abordagens de teste específicas sejam consideradas e propostas. Além disso, Wasserman (2010) relata o desafio de testar uma aplicação em diferentes tipos de dispositivos com diferentes versões de sistemas operacionais e operando em redes de comunicação diversas

(WASSERMAN, 2010; MUCCINI et al., 2012). Como alternativa, Picco et al. (2014) sugerem que o processo de teste em mobilidade deva ser repetível, ou seja, que uma vez executado para um dispositivo específico, a mesma execução de casos de teste se adapte a diferentes configurações.

Muccini et al. (2012) identificam características que tornam as aplicações móveis diferentes em relação a outros tipos de software e, em seguida, discutem as implicações causadas por tais particularidades:

- **Conectividade.** Descrita como uma das características mais críticas no contexto de mobilidade, a conectividade está relacionada à velocidade e segurança. Aplicações móveis podem necessitar de conexão a uma rede de dados segura e com bom desempenho para que as funcionalidades sejam corretamente executadas e, por isso, os testes devem considerar diferentes situações como instabilidade e/ou falha em protocolos de segurança;
- **Recursos limitados.** Apesar da evolução no software, as configurações de hardware de dispositivos móveis ainda são limitadas se comparadas às encontradas em computadores pessoais. Portanto, os testes devem considerar requisitos de desempenho como o tempo de resposta para uma determinada funcionalidade;
- **Autonomia.** Outra característica relacionada a recursos limitados é a autonomia dos dispositivos móveis que varia de acordo com a configuração de hardware, bem como com as funcionalidades habilitadas durante o uso como, por exemplo, GPS, rede dados e câmera. Simulações e monitoramento devem ser utilizados junto aos testes para garantir a correta execução da aplicação, além de identificar defeitos relacionados à autonomia do dispositivo móvel;
- **Interface com o usuário.** O desenvolvimento móvel implica no uso de padrões para a construção de interfaces, atendendo diferentes dimensões e resoluções encontradas em telas dos dispositivos móveis. Dessa forma, o teste de interface deve considerar que as aplicações móveis podem se comportar de maneira diferente a partir da característica de interface;
- **Sensibilidade ao contexto.** Aplicações móveis podem depender de informações providas pelo ambiente ou contexto em que são executadas, sendo obtidas por meio de sensores como GPS e acelerômetro. Portanto,

testes específicos devem auxiliam na detecção de defeitos ocasionados pela mudança de contexto;

- **Novas linguagens de programação.** Com a evolução da computação móvel, novas linguagens de programação também devem ser desenvolvidas para, entre outros resultados, (i) simplificar o desenvolvimento e (ii) fornecer novos elementos de interface gráfica. Assim sendo, as técnicas de teste funcional e estrutural devem ser revistas para melhor aderência ao teste de aplicações móveis;
- **Novos sistemas operacionais.** A evolução constante e a criação de novos sistemas operacionais para o contexto móvel podem resultar em incompatibilidades com versões anteriores, bem com defeitos específicos relacionados à plataforma;
- **Diversidade de dispositivos móveis.** A existência de diferentes dispositivos móveis produzidos por diferentes fabricantes com diferentes componentes de hardware e software resulta na necessidade de processos e/ou técnicas que, com o menor esforço necessário, são capazes de maximizar a cobertura dos casos de teste projetados ou gerados; e
- **Telas sensíveis ao toque.** No contexto de mobilidade, o toque na tela constitui o principal recurso de interação com as aplicações móveis. Portanto, técnicas de testes têm sido propostas para avaliar o desempenho e o tempo de resposta a partir do toque em telas em diferentes circunstâncias como, por exemplo, elevado uso de processador ou memória e com pouco espaço de armazenamento no dispositivo.

2.5.1. Teste automatizado em aplicações Android

A implementação de casos de teste executáveis em aplicações Android é apoiada por ferramentas e plataformas que fornecem APIs e funcionalidades para a simulação de interações e comportamentos de um usuário real. Nativamente, a plataforma Android dispõe de ferramentas e plataformas que permitem a automação de teste como, por exemplo, *Monkey* e *MonkeyRunner*. Entretanto, também se pode

optar por ferramentas e plataformas externas, destacando-se *Robolectric*, *Espresso* e *Robotium*.

Monkey. A ferramenta *Monkey* ou *UI/Application Exerciser Monkey* possibilita a geração de pseudoeventos em uma aplicação móvel executada em dispositivos reais ou emuladores (MONKEY, 2015). Pode ser utilizada para simular comportamentos aleatórios de um usuário. Por exemplo, abrir e fechar a mesma tela diversas vezes, forçar o preenchimento de campos de entrada de dados ou encerrar a aplicação e a reiniciar. Apesar de ser uma estratégia simples, a ferramenta *Monkey* colabora com testes de estresse e apoia a detecção de defeitos em situações que o testador dificilmente consideraria ao testar manualmente. Após a execução dos testes, os eventos aleatoriamente gerados e os defeitos detectados na aplicação móvel são exibidos em um relatório que sumariza os resultados (FARTO; ENDO, 2015a).

MonkeyRunner. A plataforma *MonkeyRunner* provê uma biblioteca para a escrita de *scripts* em Python que testam as aplicações Android a partir da simulação de interações de um usuário (MONKEYRUNNER, 2015). Dentre as funcionalidades, tais *scripts* são utilizados para (i) instalar uma aplicação, (ii) verificar a existência de um pacote específico e/ou executar a aplicação que se pretende testar, (iii) enviar comandos como pressionar, selecionar ou digitar, (iv) verificar o conteúdo de componentes de entrada de dados e (v) capturar imagens de telas da aplicação em teste. Os *scripts* de teste desenvolvidos na plataforma *MonkeyRunner* podem ser executados em dispositivos reais ou emuladores. O foco de *MonkeyRunner* é a execução de testes funcionais, estruturais e de regressão (FARTO; ENDO, 2015a).

Robolectric. A plataforma *Robolectric* possibilita a automação de testes unitários por meio da simulação de eventos que representam ações de um usuário na aplicação móvel em teste (ROBOLECTRIC, 2015). A arquitetura de classes e métodos é adaptada do Android SDK e, portanto, os métodos de teste implementados com *Robolectric* são executados na Máquina Virtual Java (em Inglês, *Java Virtual Machine* – JVM). A execução de testes na JVM aperfeiçoa a verificação de aplicações móveis, pois não se torna necessário possuir um dispositivo real ou emulado. Além disso, o tempo de execução de um teste é inferior quando comparado a outras ferramentas e plataformas, justamente por utilizar a JVM ao

invés de um dispositivo Android. Os testes implementados podem ser executados em um ambiente local ou até mesmo em um ambiente de integração contínua (FARTO; ENDO, 2015b).

Espresso. A plataforma *Espresso* é um projeto *open source* para automação de testes estruturais e de interface que possibilita o desenvolvimento de casos de teste com uma sintaxe concisa (ESPRESSO, 2015). A API é composta por quatro classes: *Espresso*, *ViewMatchers*, *ViewActions* e *ViewAssertions*. A primeira classe, *Espresso*, contém métodos para executar ações no dispositivo móvel como, por exemplo, pressionar o botão “Menu”. A segunda classe, *ViewMatchers*, disponibiliza métodos para recuperar instâncias de componentes como botões e campos de texto. A terceira classe, *ViewActions*, fornece métodos para interagir com os componentes recuperados, fazendo-se uso de eventos como toques, toques longos e preenchimento de campos. A quarta e última classe, *ViewAssertions*, provê métodos para a verificação de valores e estados da aplicação em teste. Dessa forma, testes complexos são desenvolvidos com uma sintaxe compreensível e objetiva que assegura a manutenibilidade e evolução dos casos de teste são alcançados (FARTO; ENDO, 2015b).

Robotium. A plataforma *Robotium* também é um projeto *open source* para automação de testes em Android e fornece uma API para verificar aplicações nativas ou híbridas[2] (ROBOTIUM, 2015). As classes e métodos tornam possível a execução de testes funcionais e estruturais. Assim como a plataforma *Espresso*, os conceitos de JUnit são incorporados ao *Robotium* e permitem agrupar e melhor organizar os casos de teste. O *Robotium* centraliza a maioria das funcionalidades no pacote “`com.robotium.solo`” e sua principal classe é a `Solo`. Pode-se (i) iniciar uma *Activity* específica, (ii) modificar a orientação da tela, (iii) capturar imagens de telas e (iv) simular eventos de interação como toques, preenchimentos de campos e uso de menus. Na atualidade, *Robotium* é a plataforma mais utilizada para testes automatizados em Android com uma API extensa, além de se destacar por ser intuitiva e bem documentada (FARTO; ENDO, 2015c).

[2] Uma aplicação híbrida, além de fazer uso de recursos nativos, também agrega conceitos de *design* responsivo e, comumente, é implementada com HTML5, CSS e JavaScript.

2.6. CONSIDERAÇÕES FINAIS

Neste capítulo, foram apresentados os tópicos essenciais para fundamentar os demais capítulos desta dissertação. Os conceitos de computação móvel foram abordados, bem como os principais fatores da engenharia de software para aplicações móveis e a plataforma Android. Posteriormente, uma visão geral acerca do teste de software foi descrita. Direcionou-se um foco maior aos fundamentos de TBM, pois é a principal abordagem adotada nesta dissertação.

No próximo capítulo, apresenta-se um mapeamento sistemático conduzido para caracterizar o estado da arte com base nas pesquisas realizadas na área de teste de aplicações móveis e identificar estudos relevantes. Inicialmente, 1836 pesquisas foram recuperadas e, após análise, 108 estudos foram selecionados. Além do planejamento e condução, características e aspectos quanto às pesquisas identificadas são discutidos.

3. ESTUDO DE MAPEAMENTO SISTEMÁTICO

3.1. CONSIDERAÇÕES INICIAIS

Devido ao aumento da quantidade de usuários, dispositivos e aplicações móveis, o teste de software no contexto de mobilidade tem sido cada vez mais explorado por pesquisadores e empresas. Na Figura 4 é ilustrado um gráfico com os valores de indexação na plataforma de buscas da Google para os termos "*mobile application testing*" e "*mobile app testing*" até o mês de novembro de 2015. Constata-se o crescente interesse pela área de teste de aplicações móveis, porém uma investigação formal se torna necessária para avaliar as pesquisas relevantes.

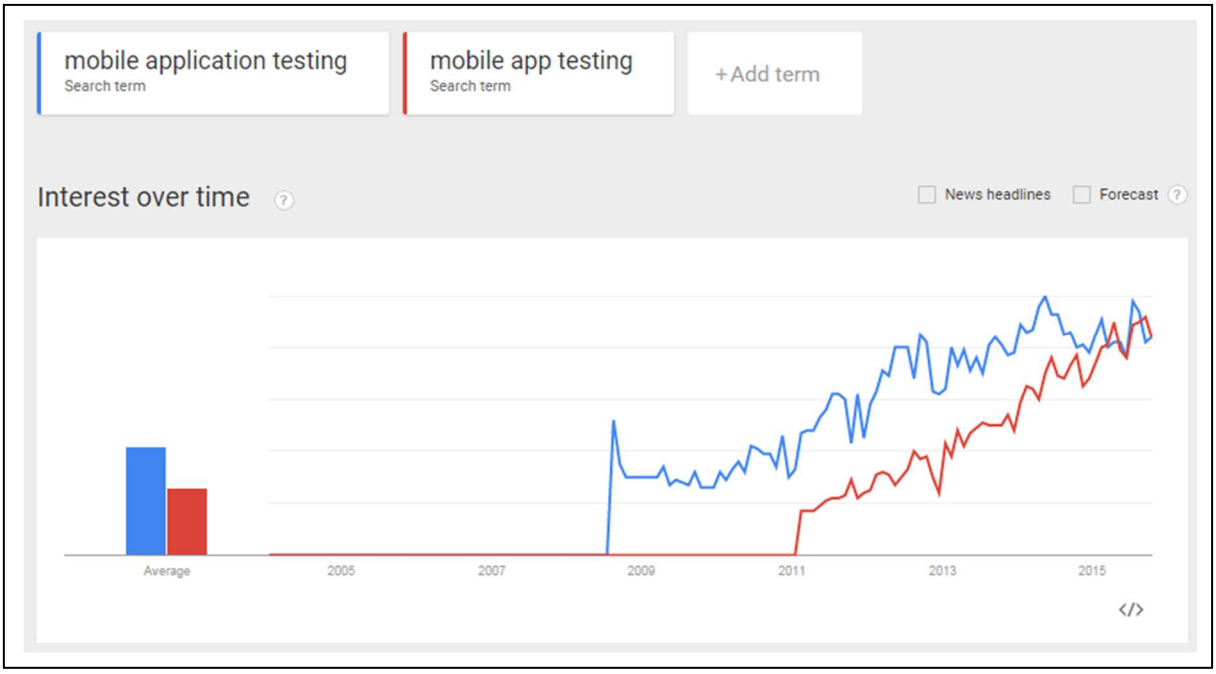

Figura 4 - Indexação de busca para "mobile application testing" e "mobile app testing"
Fonte: Google Trends (2015)

Um Estudo de Mapeamento Sistemático (EMS), do Inglês *Systematic Mapping Study*, é um método que pode ser utilizado para evidenciar tópicos específicos de interesse (KITCHENHAM et al., 2002; PETERSEN et al., 2008). O EMS provê uma visão ampla de uma área de pesquisa e é adotado quando se deseja identificar a quantidade e características de estudos, resultando no mapeamento dos resultados conhecidos.

Neste capítulo, é apresentado um mapeamento sistemático que descreve a visão geral de teste de software no contexto de aplicações móveis. Na Seção 3.2, o planejamento do EMS é descrito. Na Seção 3.3, são apresentados os objetivos e as questões de pesquisa. Nas Seções 3.4 e 3.5, são definidos o processo de busca e os critérios de inclusão e de exclusão, respectivamente. Na Seção 3.6, a condução do mapeamento sistemático é relatada. Na Seção 3.7, são apresentadas a análise e discussão dos resultados. Na Seção 3.8, os trabalhos relacionados são descritos.

3.2. PLANEJAMENTO

Para alcançar os resultados esperados, deve-se seguir um processo definido que auxilie na condução do EMS. Para isso, o processo proposto por Petersen et al. (2008) foi escolhido por prover um modelo que apoia a busca, triagem, avaliação e análise dos estudos.

3.3. OBJETIVOS E QUESTÕES DE PESQUISA

A condução do EMS objetivou fornecer uma visão ampla do que tem sido pesquisado na área de teste de aplicações móveis. Dentre as contribuições esperadas, destacam-se a identificação da quantidade de pesquisas publicadas e a análise dos estudos definidos como relevantes. As pesquisas selecionadas foram classificadas a partir de características específicas como, por exemplo, (i) avaliação experimental em ambiente industrial, (ii) adoção de abordagem fundamentada em TBM, (iii) abordagem para modelagem ou execução do teste, se aplicável, e (iv) plataforma de desenvolvimento móvel.

Para auxiliar na definição do escopo do estudo, as seguintes questões de pesquisa foram elencadas:

***Q1**. Quantas pesquisas em teste de aplicações móveis foram publicadas?*

***Q2**. Quais os principais tópicos ou termos utilizados?*

***Q3**. Quantas pesquisas em teste de aplicações móveis são avaliadas na indústria ou em ambientes reais?*

***Q4**. Quantas pesquisas em teste de aplicações móveis se baseiam em TBM?*

***Q5**. Quais as abordagens para modelagem ou execução do teste?*

***Q6**. Quais as principais estratégias de teste?*

***Q7**. Quais as plataformas de desenvolvimento móvel adotadas?*

3.4. PROCESSO DE BUSCA

Para a coleta de pesquisas, selecionaram-se bases de dados que (i) possuem um mecanismo de busca baseado na Web, (ii) permitem a busca por comandos e palavras-chave e (iii) centralizam pesquisas de ciência da computação e áreas relacionadas. As bases de dados utilizadas foram Scopus, IEEExplore e ACM *Digital Library*[3]. A *string* de busca foi construída para representar a combinação de expressões-chave com base em "aplicação móvel" e "teste", definidas como áreas de interesse para a realização do EMS. Utiliza-se o operador booleano "*OR*" para considerar palavras e sinônimos alternativos, enquanto que o operador "*AND*" relaciona as áreas investigadas. A *string* é apresentada a seguir.

```
( ("mobile application" OR "mobile applications" OR
  "mobile software" OR "mobile softwares" OR "mobile app" OR
  "mobile apps" OR "mob application" OR "mob software" OR
  "mob app" OR "smartphone application")
AND (test OR testing OR verification) )
```

3.5. CRITÉRIOS DE INCLUSÃO E EXCLUSÃO

Mesmo com a construção de uma *string* de busca bem elaborada e delimitada, uma importante atividade do planejamento de um EMS é a definição de

[3] http://scopus.com , http://ieeexplore.ieee.org e http://dl.acm.org/

critérios de inclusão (CI) e de exclusão (CE). Respectivamente, os critérios de inclusão e exclusão apoiam a seleção apropriada de estudos e a redução do número de pesquisas obtidas pelos mecanismos de busca. A correta utilização de CIs e CEs é fundamentada pelas premissas de que uma dada pesquisa (i) será válida somente se atender a todos os CIs (conjunção lógica *AND*), porém (ii) não será válida quando um ou mais CEs forem atendidos (disjunção lógica *OR*). Os CIs e CEs formulados são apresentados a seguir.

- **CI1.** Artigos disponíveis na Web;
- **CI2.** Artigos em Inglês ou Português;
- **CI3.** Artigos que possuem, como foco principal, o teste de aplicações móveis;
- **CI4.** Artigos que, mesmo não adotando TBM, utilizam técnicas de modelagem.

- **CE1.** Artigos que não sejam completos, ou seja, resumos ou *short papers*;
- **CE2**. Artigos que foram estendidos ou que apresentam menor contribuição se comparados às versões atualizadas;
- **CE3.** Artigos que descrevem o teste de aplicações móveis, porém a incorporam como uma etapa adicional a outro assunto principal discutido na pesquisa.

3.6. CONDUÇÃO

Após a realização de buscas, um total de 1836 artigos foi obtido. Inicialmente, efetuou-se uma leitura do título e do resumo das pesquisas recuperadas e estas foram catalogadas separadamente por fonte de pesquisa, possibilitando uma medição do percentual de relevância para Scopus, IEEExplore e ACM *Digital Library*. Posteriormente, os CIs e CEs foram aplicados e, em casos específicos, uma leitura mais abrangente do artigo foi necessária para identificar as contribuições descritas na pesquisa, reduzindo a quantidade de publicações selecionadas para 150. Após eliminação dos estudos repetidos, 108 pesquisas foram catalogadas como relevantes ao contexto de teste de aplicações móveis e representam 5,88% dos 1836 artigos inicialmente recuperados.

Na Tabela 1 são apresentadas a quantidade de pesquisas retornadas com base nas buscas e a quantidade das classificadas como relevantes. É importante ressaltar que as buscas não limitam um período inicial, porém se restringem até o mês de outubro de 2015.

Tabela 1 - Resultado da condução das buscas e avaliação dos artigos

Base de dados	**Pesquisas identificadas**	**Pesquisas selecionadas [1]**	**% [2]**	**Pesquisas unificadas [3]**
Scopus	955	83	55,33	**108**
IEEExplore	703	45	30,00	
ACM *Digital Library*	178	22	14,67	
Totais	**1836**	**150**	**100,0**	**5,88% de 1836**

[1] Resultados obtidos após leitura de título e resumo e, casualmente, do artigo, bem como a validação de CIs e CEs

[2] Percentual de pesquisas relevantes por bases de dados em relação ao total de pesquisas classificadas como relevantes

[3] Resultados unificados após a remoção de 42 pesquisas repetidas

3.7. ANÁLISE DOS RESULTADOS E DISCUSSÃO

Uma tabulação dos resultados foi realizada para classificar e sumarizar as pesquisas com base nas principais características dos artigos selecionados. Um formulário de extração de dados do EMS foi elaborado conforme apresentado no Apêndice A. Dessa forma, as questões de pesquisa elaboradas para o EMS são discutidas e as características obtidas relatadas.

Q1. *Quantas pesquisas em teste de aplicações móveis foram publicadas?*

Na Figura 5 é ilustrado um gráfico que representa a evolução de estudos acadêmicos conduzidos e publicados. Verifica-se um aumento na quantidade de pesquisas que discutem e avaliam o teste de aplicações móveis, evidenciando um

possível engajamento da comunidade acadêmica em explorar e propor contribuições. Constata-se que, desde 2004, há um crescimento das pesquisas relevantes que investigam o teste de aplicações móveis. Como mencionado, o EMS se limita até outubro de 2015, totalizando 19 pesquisas no mesmo ano.

Figura 5 - Quantidade de publicações por ano
Fonte: Elaborada pelo autor

Analisando o indicador de tipo de evento ou material, as pesquisas podem ser classificadas em contribuições de (i) conferência, (ii) *workshop*, (iii) periódico, (iv) livro e (v) capítulo de livro. Na Figura 6 é ilustrado um gráfico que apresenta a quantidade de pesquisa relevante por tipo de publicação.

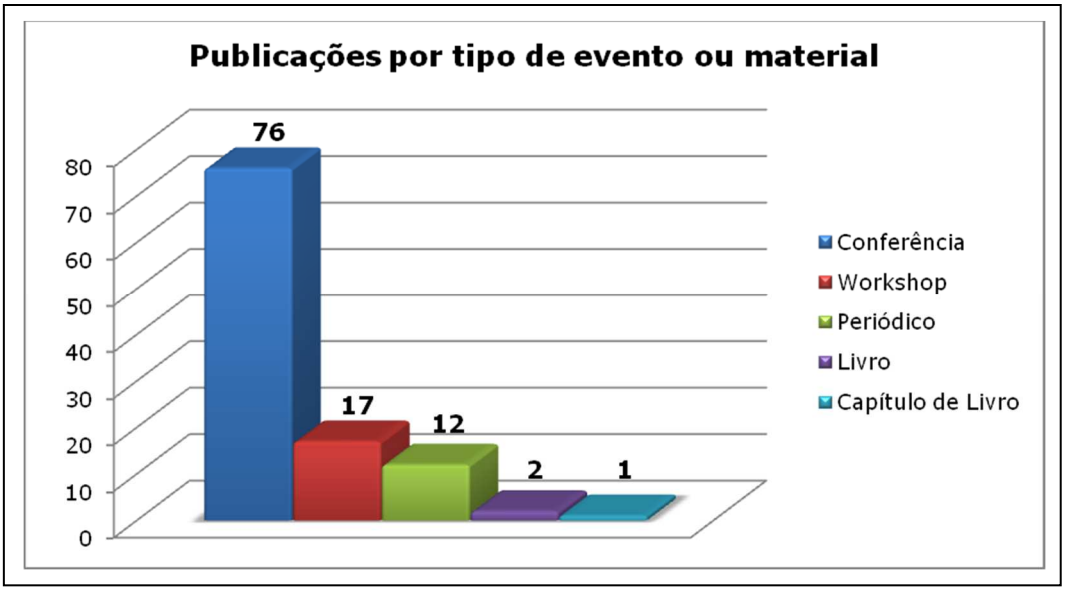

Figura 6 - Quantidade de publicações por tipo de evento ou material
Fonte: Elaborada pelo autor

Q2. *Quais os principais tópicos ou termos utilizados?*

Assim como considerado nas buscas realizadas na condução do EMS, assume-se que o título e o resumo contribuem na formulação de elementos úteis a um possível leitor, fornecendo um entendimento inicial acerca do estudo relatado. Após a extração de todos os títulos e resumos das 108 pesquisas selecionadas, os conteúdos textuais foram utilizados na geração de uma representação visual em nuvem de palavras (em Inglês, *Word Clouds*). Dessa forma, verificam-se os principais tópicos e termos adotados no título e na descrição resumida dos estudos que investigam o teste de aplicações móveis.

Os termos são valorados a partir da quantidade de repetições que ocorrem no conjunto total de títulos e resumos selecionados. Para destacar os tópicos mais relevantes, eliminaram-se termos e suas variações que notadamente são utilizadas como (i) "*mobile*", (ii) "*application*" e (iii) "*test*", bem como expressões com pouca significância como (iv) "*however*", (v) "*also*" e (vi) "*due*". Como resultado, obteve-se um extenso conjunto de palavras que, de acordo com a proposta da nuvem de palavras, representa os termos mais utilizados em relação a outros.

Na Figura 7 é ilustrada uma nuvem de palavras confeccionada com base nos termos utilizados nos títulos e resumos dos estudos. Apesar de não ser uma estratégia que garante a corretude de uma real análise acerca da relevância e do direcionamento de pesquisas, o uso de nuvem de palavras simplifica a busca por palavras mais adotadas.

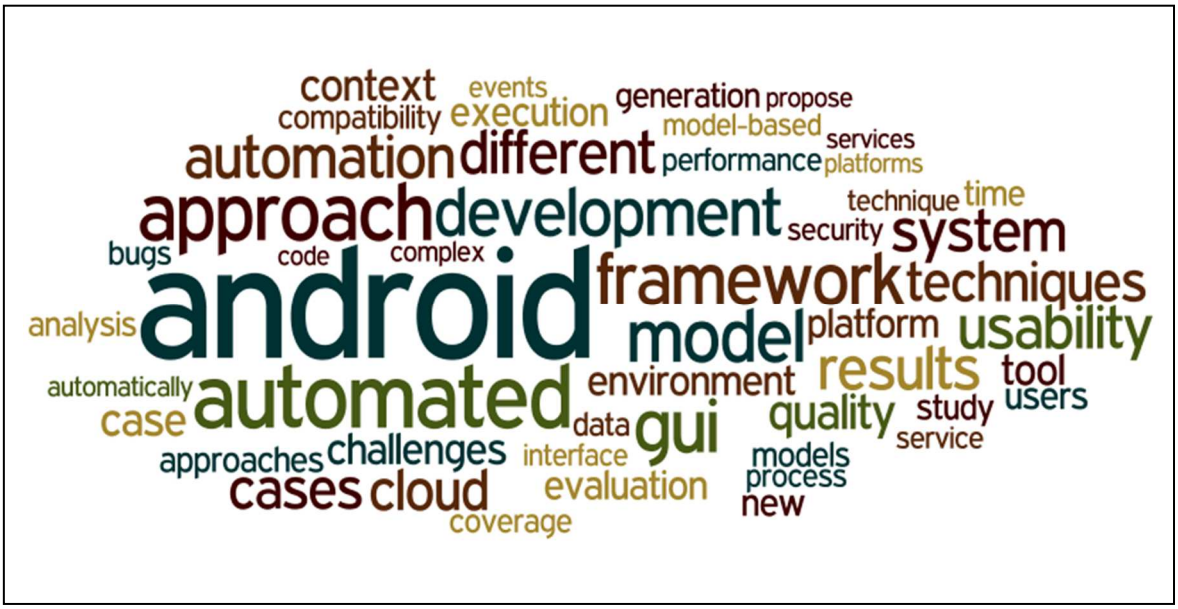

Figura 7 - Nuvem de palavras elaborada com base nos títulos e resumos
Fonte: Elaborada pelo autor

Ao considerar as 50 palavras mais bem valoradas a partir dos títulos e resumos das pesquisas, os termos “*android*” (131 repetições), “*model*” (86 repetições), “*automated*” (70 repetições), “*gui*” (65 repetições) e “*approach*” (63 repetições) se destacam. Logo em seguida, verifica-se um maior uso das palavras “*framework*” (57 repetições), “*development*” (53 repetições), “*automation*” (48 repetições), “*usability*” (48 repetições) e “*cases*” (45 repetições). Com menor valor, porém ainda no conjunto de palavras destacadas, ressaltam-se “*techniques*” (44 repetições), “*quality*” (42 repetições) e “*challenges*” (34 repetições).

Uma vez que títulos e resumos estejam bem elaborados e condizentes com o escopo, objetivo, metodologia, resultados e demais elementos descritivos de uma pesquisa científica, a nuvem de palavras apoiará na identificação de pontos de interesse comuns entre estudos. Apenas considerando a nuvem de palavras apresentada, observações podem ser suscitadas. Por exemplo, (i) ênfase na plataforma Android, (ii) uso de modelos de teste, (ii) adoção de mecanismos de teste que se baseiam em processos automatizados, (iv) análise de testes quanto a aspectos de interface gráfica (em Inglês, “*Graphical User Interface*” – GUI) e (v) proposição e/ou uso de abordagens. Além disso, observa-se que os estudos investigam e/ou incorporam *frameworks* de teste em mobilidade, desenvolvem estratégias e/ou ferramentas de teste, definem e/ou avaliam técnicas de teste e relatam os desafios identificados ou relacionados ao teste de aplicações móveis.

Q3. *Quantas pesquisas em teste de aplicações móveis são avaliadas na indústria ou em ambientes reais?*

Independentemente dos fundamentos e das vantagens que tornam o TBM uma das promissoras técnicas quanto aos aspectos de modelagem e de execução de casos de teste, verifica-se que, no contexto de aplicações móveis, pouco do que se propõe ou desenvolve tem sido avaliado em ambientes industriais. A condução do EMS resultou na classificação das 108 pesquisas em grupos de estudos nomeados como (i) exploratório, (ii) experimental e (iii) industrial.

Pesquisas de escopo exploratório. Objetivam, primariamente, explorar, fundamentar e documentar informações de determinado(s) assunto(s) ou tema(s)

que pouco se investigam. Portanto, têm como objetivo proporcionar maior familiaridade com a área investigada (GIL, 2010).

Dantas et al. (2009) propõem e descrevem requisitos de teste para aplicações móveis, explorando aspectos e características de teste funcional, de interface e de usabilidade. Amalfitano et al. (2013) sumarizam os desafios e dificuldades para o teste na plataforma Android e apresentam princípios, técnicas e diretrizes que podem contribuir com a execução. Kirubakaran e Karthikeyani (2013) descrevem as particularidades e desafios similarmente identificados por Gao et al. (2014) e Muccini et al. (2012) e afirmam que estratégias específicas são necessárias para a realização de teste de aplicações móveis. Méndez-Porras et al. (2015) apresentam uma revisão sistemática das abordagens, técnicas e desafios do teste automatizado em aplicações móveis. Diferentemente do EMS conduzido nesta dissertação, os autores limitam as buscas em pesquisas que adotam o teste automatizado.

Pesquisas de escopo experimental. Descrevem a proposta e/ou uso de técnicas, abordagens, ferramentas e outras estratégias na área investigada, porém sob aspectos de experimentação. Dessa forma, objetivam avaliar contribuições em um ambiente controlado ou simulado com o propósito de verificar, por exemplo, benefícios, dificuldades, limitações e lições aprendidas a partir da definição de objeto(s) de estudo, variáveis e formas de controle e de observação (GIL, 2010).

Kronbauer et al. (2012) propõem uma infraestrutura para testes de usabilidade, bem como um modelo automático para monitoria e coleta de dados e métricas. A abordagem proposta foi avaliada por 21 usuários em três aplicações móveis durante seis meses. Ravindranath et al. (2014) apresentam uma ferramenta que integra teste funcional e estrutural para projetos desenvolvidos em Windows Phone. Em um ambiente experimental, 3000 aplicações foram verificadas e a abordagem proposta detectou 2969 defeitos, incluindo 1227 que não haviam sido reportadas anteriormente. Deng et al. (2015) propõem uma abordagem específica para o teste de mutação com base nas características e particularidades da plataforma Android. Os autores reportam resultados preliminares acerca da possibilidade do uso de teste de mutação em Android. Ao final, relatam desafios identificados para que abordagem seja mais efetiva.

Pesquisas de escopo industrial. Além de explorar e, na maioria dos casos, propor novas abordagens ou ferramentas, delineiam acerca da adoção em ambientes reais. Por meio das pesquisas industriais, avalia-se efetivamente como se dá o comportamento do que está sendo proposto em cenários de empresas de tecnologia com profissionais e equipes relacionados ao assunto ou contexto explorado. Mesmo em número reduzido, algumas pesquisas são conduzidas na realidade industrial. As contribuições de Ridene e Barbier (2011), Janicki et al. (2012) e Li et al. (2014) são apresentadas como trabalhos relacionados.

Na Figura 8 é ilustrado um gráfico que apresenta a quantidade de estudos conduzidos de natureza exploratória, experimental e industrial. Confirma-se que uma parcela reduzida de apenas 5% das pesquisas identificadas para o teste de aplicações móveis transpõe os cenários exploratórios e experimentais, resultando em uma avaliação real na indústria e em ambientes reais. Em seguida, 25 pesquisas constituem 22% dos estudos relevantes que atuam no âmbito exploratório, relatando o cenário de teste de aplicações móveis. Por fim, 82 estudos que representam a grande maioria de 73% das pesquisas selecionadas direcionam esforços para avaliações experimentais. É importante ressaltar que há pesquisas que realizam dois tipos de investigações em um mesmo estudo.

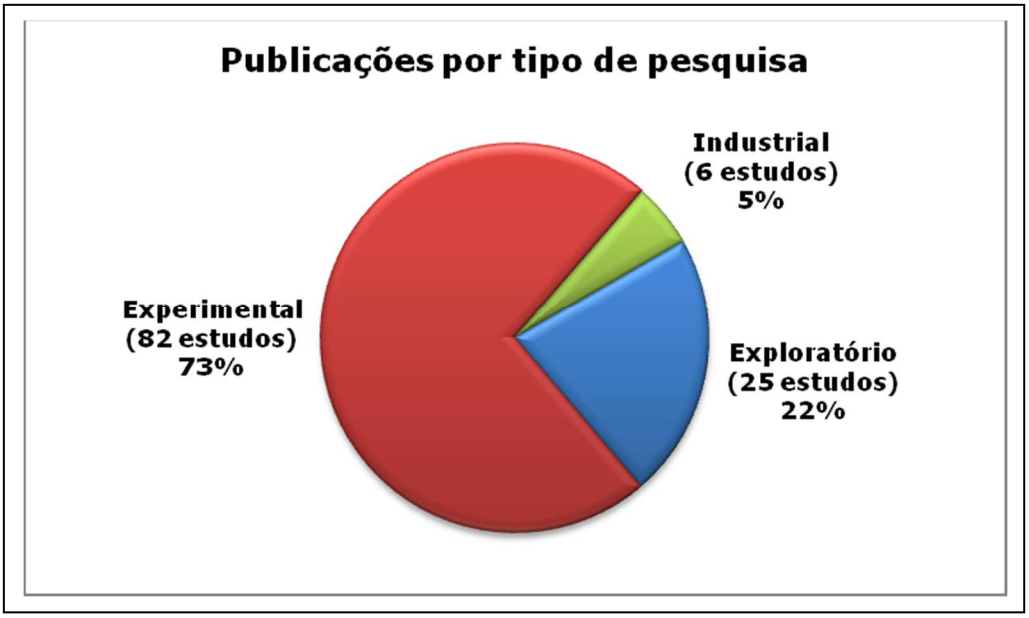

Figura 8 - Quantidade de publicações por tipo de pesquisa
Fonte: Elaborada pelo autor

Q4. *Quantas pesquisas em teste de aplicações móveis se baseiam em TBM?*

O processo de TBM tem sido explorado em diferentes contextos e, mais recentemente, distintas pesquisas propõem ferramentas comerciais, acadêmicas e *open source* para apoiar e melhor explorar os benefícios do teste automatizado (RODRIGUES, 2013). Para o teste de aplicações móveis, abordagens fundamentadas em TBM ocupam 22% das 108 pesquisas relevantes. Estudos que consideram TBM no contexto de mobilidade são descritos na Seção 3.8.

Na Figura 9 é ilustrado um gráfico que apresenta a quantidade de pesquisas que se baseiam em um processo de TBM como estratégia para o teste automatizado de aplicações móveis.

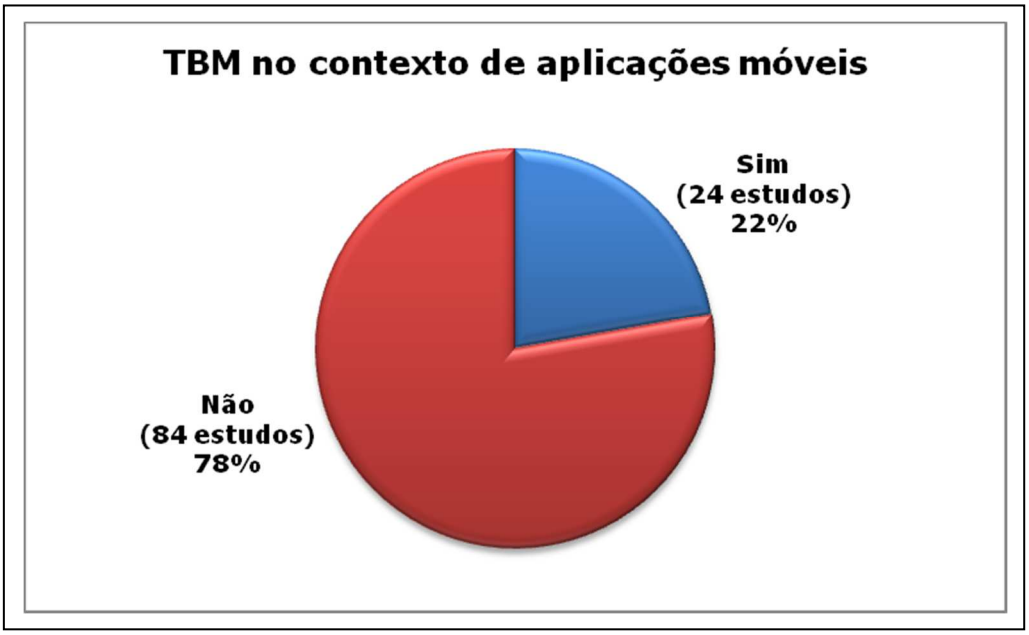

Figura 9 - Quantidade de publicações que adotam TBM
Fonte: Elaborada pelo autor

Q5. *Quais as abordagens para modelagem ou execução do teste?*

Apesar do fato de que aproximadamente um quarto das pesquisas adota alguma estratégia apoiada por TBM, vários outros estudos selecionados incorporam abordagens para modelagem ou execução do teste. Tais indicadores auxiliam no reconhecimento das técnicas utilizadas para elaboração de modelos de teste,

representação de comportamentos e funcionalidades e, por fim, estratégias específicas que têm sido descritas e avaliadas.

Na Figura 10 é apresentado um gráfico que relaciona as técnicas identificadas com o EMS para modelagem ou execução do teste de aplicações móveis. Em destaque na cor vermelha, nota-se que 72 pesquisas não aplicam ou descrevem o uso de recursos para a modelagem ou execução do teste. Em seguida, os itens destacados em verde apontam as seis principais abordagens mencionadas nos estudos: *State Machine*, *Domain-Specific Language*, *Labelled Transition System*, *Test Script*, UML e *Event Sequence Graph*. Duas ou mais técnicas para modelagem ou execução do teste são adotadas em um mesmo estudo.

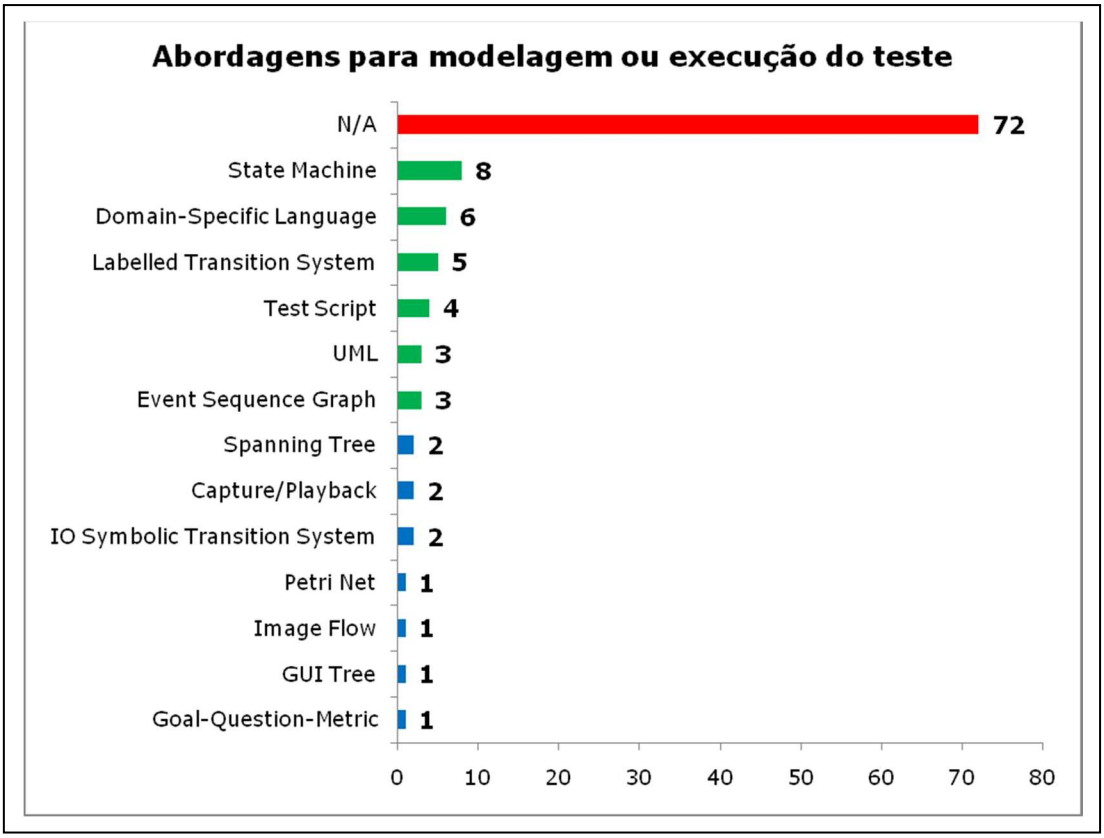

Figura 10 - Abordagens para modelagem ou execução do teste
Fonte: Elaborada pelo autor

Q6. *Quais as principais estratégias de teste?*

Além das abordagens identificadas, destaca-se a importância em verificar as estratégias de testes comumente exploradas no contexto de mobilidade, constatando, dessa forma, os aspectos de teste mais pesquisados. Na Figura 11 é

ilustrado um gráfico que apresenta as diferentes estratégias de teste identificadas. Para colaborar com a visualização dos resultados obtidos, as estratégias de teste menos adotadas foram agrupadas: estrutural, baseado em defeitos (teste de mutação), combinatório, acessibilidade, conformidade e integração. Há pesquisas que investigam duas ou mais estratégias de teste em um mesmo estudo.

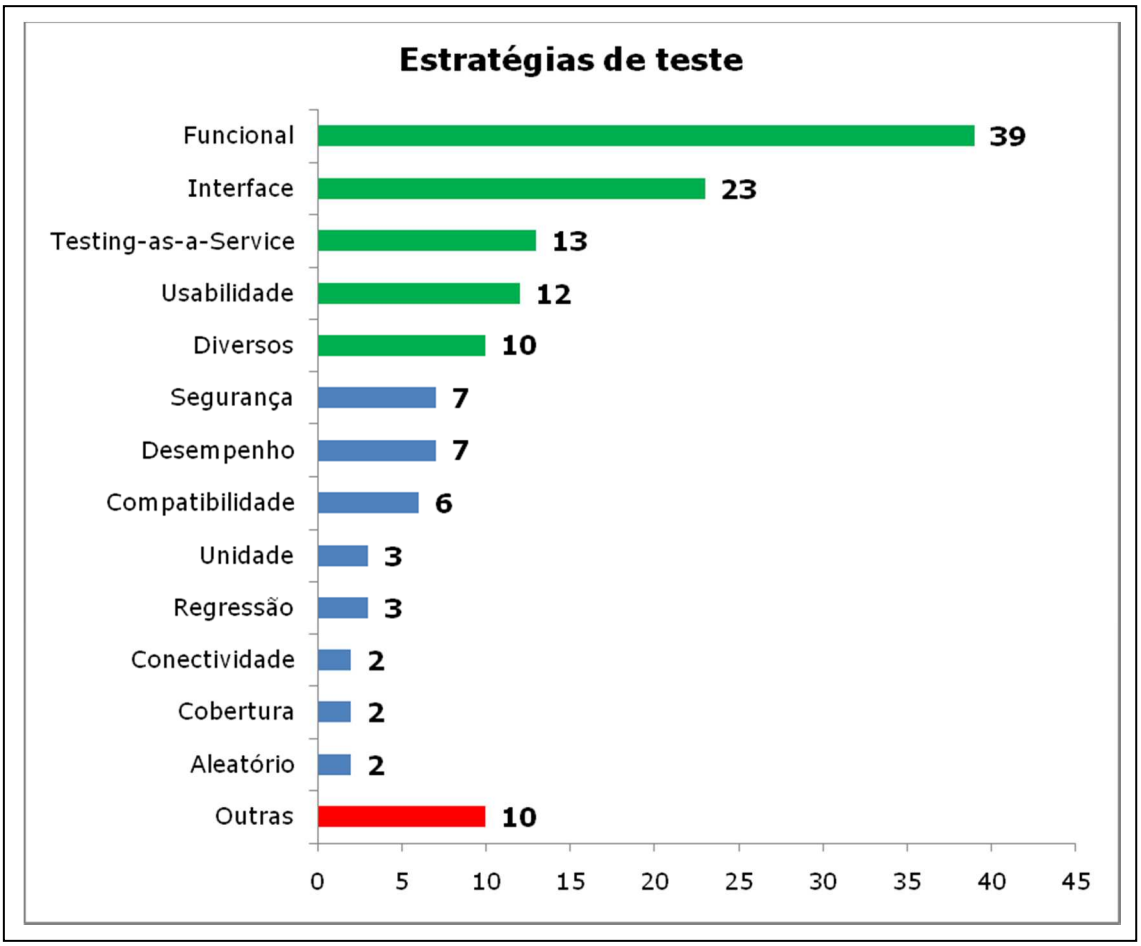

Figura 11 - Estratégias de teste
Fonte: Elaborada pelo autor

Q7. *Quais as plataformas de desenvolvimento móvel adotadas?*

Por fim, outro ponto de interesse analisado se refere à identificação de plataformas de desenvolvimento e sistemas operacionais considerados nas pesquisas selecionadas. Na Figura 12 é ilustrado um gráfico de bolhas que exibe as tecnologias móveis no eixo Y de análise e uma listagem entre 2006 e Outubro de 2015 no eixo X de análise. A largura das bolhas é definida com base na quantidade de publicações em que tais plataformas foram investigadas nas pesquisas.

Observa-se uma transição ao longo do tempo no uso de plataformas móveis legadas, como JME e Symbian, para plataformas modernas como Google Android, Apple iOS e Microsoft Windows Phone. Verifica-se que a plataforma Android se destaca em relação às demais, além de sua adoção em pesquisas estar em crescimento desde 2011.

É importante ressaltar que as bolhas preenchidas em vermelho, ou sem rótulo de dados, identificam pesquisas isoladas que utilizaram determinada plataforma naquele ano. Os anos de 2004 e 2005 foram omitidos, pois os estudos não mencionam plataformas ou sistemas operacionais específicos. Nos anos subsequentes, incluem-se pesquisas que (i) não descrevem nenhuma plataforma ou sistema operacional, (ii) descrevem uma plataforma ou sistema operacional e (iii) descrevem duas ou mais plataformas ou sistemas operacionais.

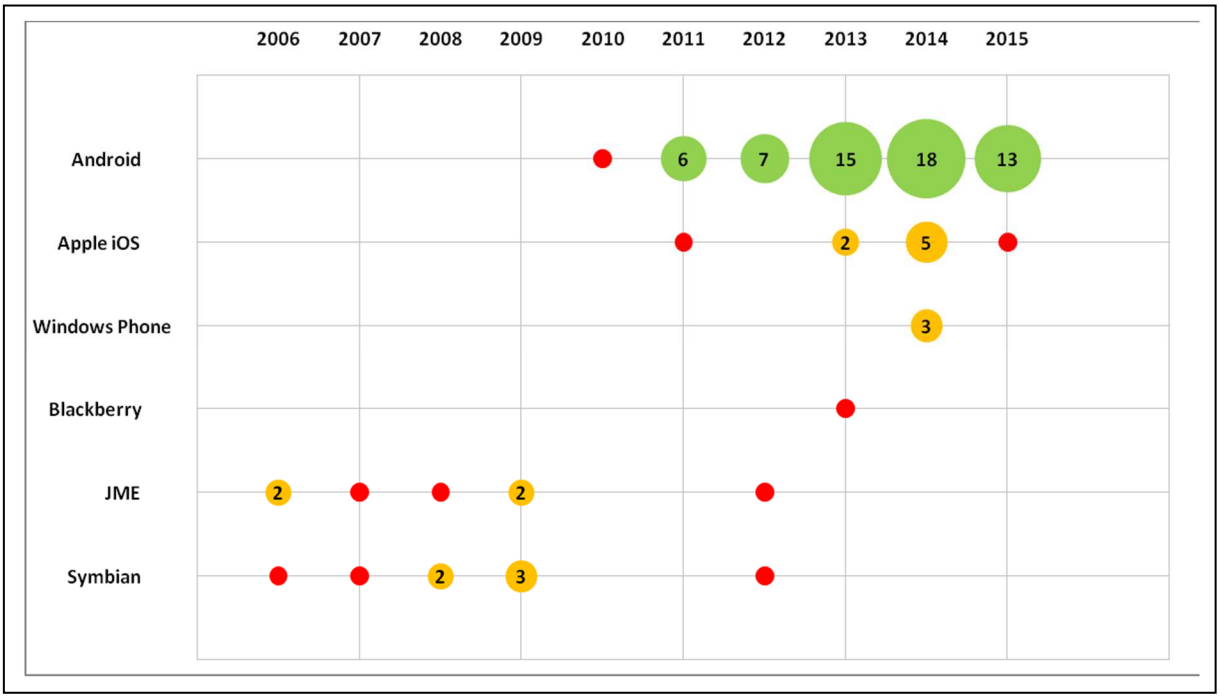

Figura 12 - Gráfico de bolhas das plataformas móveis utilizadas em pesquisas
Fonte: Elaborada pelo autor

3.8. TRABALHOS RELACIONADOS

A condução do EMS auxiliou na identificação de 24 pesquisas, incluindo o estudo de Farto e Endo (2014a), que exploram, investigam, propõem e avaliam

abordagens, técnicas, estratégias e ferramentas de apoio à adoção de TBM no contexto de aplicações móveis. Tais contribuições e avanços das pesquisas são descritos a seguir.

Jääskeläinen et al. (2008a) propõem e implementam um conceito de serviço Web e LSTS para apoiar a elaboração de modelos de teste extensos quando o TBM é direcionado a aplicações móveis. Jääskeläinen et al. (2008b) descrevem uma biblioteca de modelos de teste específica para a verificação de aplicações móveis que se baseiam no sistema operacional Symbian. Jääskeläinen et al. (2009a) relatam os resultados de estudos realizados para avaliar a efetividade na geração de testes automatizados a partir de GUI de aplicações móveis. A abordagem se torna praticável a partir de uma ferramenta de apoio que auxilia na modelagem, geração, concretização e execução de casos de teste. A avaliação experimental resulta na detecção de 20 defeitos das aplicações móveis verificadas. Jääskeläinen et al. (2009b) descrevem uma metodologia para sintetizar modelos de teste a partir de casos de teste previamente definidos. Dessa forma, a abordagem proposta contribui com a otimização da adoção de TBM em aplicações móveis, pois casos de teste existentes são reusados. Dois estudos de caso foram conduzidos em ambientes industriais para avaliar a abordagem proposta, resultando na geração de modelos de teste executáveis com 12523 e 2327 estados.

Ridene e Barbier (2011) descrevem uma abordagem e ferramenta para o uso de *Domain-Specific Modeling Language* (DSML) em um ambiente industrial. Os fundamentos apresentados se baseiam em SPL. Uma ferramenta chamada *Mobile Applications Testing Language* (MATeL) foi implementada para elaborar modelos de teste que definem os requisitos da aplicação móvel, bem como marcadores de variabilidade das características e funcionalidades do SUT e/ou do dispositivo móvel utilizado na execução dos testes. Uma avaliação em ambiente industrial é conduzida para investigar a abordagem de TBM e SPL.

Takala et al. (2011) relatam experiências com a abordagem de TBM como alternativa para automatizar o teste de interface em aplicações Android. A pesquisa tem como foco a avaliação de um conjunto de ferramentas baseadas em TBM chamado de TEMA ou TEMA *Toolset*. A representação dos eventos da aplicação em teste é elaborada com MEFs e LSTS e uma avaliação experimental foi conduzida para aplicar as ferramentas propostas no teste de uma aplicação Android. As funcionalidades básicas da aplicação em teste foram modeladas no ambiente TEMA

e, após a execução dos testes, 14 defeitos de interface foram identificados, dos quais oito foram detectados na modelagem e outros seis na execução dos testes.

Dev et al. (2012) relatam experiências de diferentes abordagens amparadas por TBM e avaliam benefícios e dificuldades. Os autores enfatizam a adoção de TBM na execução de testes de interface. Um estudo de caso foi conduzido para avaliar uma ferramenta de TBM na execução paralela de testes em múltiplos dispositivos móveis.

Janicki et al. (2012) descrevem um *survey* que investiga e relata obstáculos e oportunidades ao aplicar TBM em aplicações móveis em contextos industriais. Os autores afirmam que pesquisas devem ser realizadas para simplificar a elaboração e a evolução de modelos de teste, contribuindo com a adoção de TBM em ambientes reais. Além disso, métricas devem ser desenvolvidas para relatar resultados obtidos e possibilitar a comparação com outras estratégias de teste. Trabalhos futuros são apresentados para direcionar esforços em pesquisas de TBM na verificação de aplicações móveis.

Lu et al. (2012) propõem uma abordagem para o teste funcional que se baseia na extração de um modelo gráfico que representa uma árvore de eventos. Tal modelo é dinamicamente obtido por meio da verificação de classes *Activity* e das transições de eventos. Posteriormente, utiliza-se o modelo para a geração e concretização de casos de teste. Um estudo experimental foi conduzido para avaliar a abordagem proposta.

Püschel et al. (2012) apresentam uma abordagem que explora os conceitos de Redes de *Petri* como técnica de modelagem junto ao processo de TBM. Os autores propõem uma estratégia chamada *Dynamic Feature Petri Nets* (DFPN) para a elaboração de modelos de teste que, posteriormente, são derivados para contemplar distintas configurações como, por exemplo, uso de sensores e redes de dados. Dessa forma, os modelos que se baseiam em DFPN definem os comportamentos do SUT e consideram características adicionais ao teste de aplicações móveis. Uma ferramenta chamada *Mobile Application Test Environment* (MATE) foi implementada para integrar os fundamentos propostos de DFPN.

Jensen el al. (2013) propõem uma abordagem que integra o processo de TBM e modelagem de eventos a uma estratégia de teste dividida em duas etapas. Inicialmente, uma execução simbólica (KING, 1976; CADAR; SEN, 2013) é realizada para identificar os eventos e transições de eventos que a aplicação pode assumir.

Posteriormente, o caminho reverso de determinado componente ou trecho de código fonte é percorrido até que uma entrada inicial de interação pela camada de interface seja identificada. Uma ferramenta chamada *Colider* apoia a estratégia de geração de sequências de eventos apresentada. Como resultados, obtêm-se uma sequência de eventos que pode ser aplicada na construção de casos de teste para percorrer, na maior parte ou em totalidade, eventos lançados pela aplicação móveis. Estudos experimentais avaliam a abordagem proposta em cinco aplicações Android.

Salva et al. (2013) exploram o uso de TBM no teste de aplicações móveis e enfatizam a verificação de aspectos de segurança e de vulnerabilidades. Os autores implementam uma ferramenta chamada *Android aPplications SEcurity Testing* (APSET) para apoiar a estratégia de teste proposta. Um estudo de caso foi conduzido para avaliar a estratégia e ferramenta APSET propostas no teste de requisitos de segurança em dez aplicações Android. Como resultado, defeitos relacionados (i) a integridade de dados e (ii) a código fonte malicioso foram detectados. Salva e Zafimiharisoa (2013) propõem uma abordagem para o teste de segurança baseado em modelo para apoiar a detecção de defeitos relacionados a vulnerabilidades em aplicações Android. A estratégia de teste gera automaticamente casos de teste que verificam aspectos de segurança no SUT. A ferramenta de apoio APSET é avaliada em oito aplicações desenvolvidas na plataforma Android, resultando na detecção de defeitos.

Yang et al. (2013) descrevem uma abordagem de TBM para a extração de informações e elaboração de um modelo de eventos de maneira automatizada. Para isso, a camada de interface gráfica é analisada e os componentes visuais são identificados, tornando possível a coleta de informações acerca de eventos que podem ser lançados. Posteriormente, as informações extraídas são utilizadas para modelar os eventos identificados com base em uma MEF. Uma avaliação experimental foi conduzida em oito aplicações móveis Android que, segundo afirmação dos autores, resultou na capacidade de extrair modelos compreensíveis, porém ainda reduzidos em relação à complexidade das aplicações.

Amalfitano et al. (2014a) propõem e implementam uma ferramenta de teste chamada MobiGUITAR que apoia o TBM em aplicações móveis desenvolvidas em Android. A ferramenta MobiGUITAR utiliza MEF e incorpora uma estratégia específica para a geração de casos de teste. Uma avaliação experimental foi conduzida em quatro aplicações móveis. Os autores relatam (i) as experiências

positivas de MobiGUITAR na verificação de aplicações Android e (ii) a efetividade de detecção de defeitos de MobiGUITAR quando comparada às ferramentas Monkey e Dynodroid. Após a geração e execução de 7711 casos de teste, dez defeitos foram detectados. Amalfitano et al. (2014b) propõem uma estratégia para aumentar a cobertura de código em testes com base em padrões identificados em modelos extraídos da aplicação em teste. Após a análise dos padrões, casos de teste são adicionalmente gerados para executar testes de interface.

Costa et al. (2014) apresentam e avaliam uma abordagem chamada *Pattern Based GUI Testing* (PBGT) que abstrai os modelos de teste e reduz os esforços das etapas de modelagem e execução. Para isso, os autores propõem a inclusão de padrões e/ou comportamentos utilizados em testes de interface nos modelos elaborados. Uma avaliação experimental foi conduzida em uma aplicação Android e 41 defeitos foram propositalmente gerados por mutação. A execução de testes de interface com a abordagem PBGT resultou na detecção de 85,4% dos defeitos inseridos.

Griebe e Gruhn (2014) investigam o teste de aplicações móveis sensíveis ao contexto. O conceito de sensibilidade ao contexto exige que as aplicações móveis considerem parâmetros do ambiente como, por exemplo, sensores de acelerômetro e localização por GPS. Os autores propõem uma abordagem estendida de TBM com UML para a geração de casos de teste. A pesquisa utiliza *Calabash* (CALABASH, 2015), uma plataforma de teste de aplicações móveis desenvolvidas em Android. Por meio da abordagem proposta, novas funcionalidades foram acrescidas à *Calabash* para testar aspectos de sensibilidade ao contexto. Um estudo de caso foi conduzido para avaliar a abordagem quanto à geração e execução dos casos de teste a partir dos modelos elaborados com UML.

Li et al. (2014) apresentam um framework de teste de interface implementado para apoiar a modelagem de funcionalidades por meio da UML. Os autores avaliam a plataforma ADAutomation em duas aplicações industriais e relatam a redução no tempo de teste e a melhoria na elaboração de modelos e geração de casos de teste.

Schweighofer e Heričko (2014) exploram abordagens de MEFs e UML junto ao processo de TBM como estratégias a serem utilizadas na elaboração de modelos de teste. Uma análise exploratória e detalhada apresenta vantagens, porém também desvantagens e problemas na definição de diagramas complexos.

Tao e Gao (2014) propõem uma abordagem chamada *Mobile Test Environment Semantic Tree* (MTEst) que auxilia na definição de ambientes configuráveis para a execução de testes em mobilidade. O modelo MTEst se fundamenta na necessidade de testar aplicações móveis em distintos cenários como, por exemplo, diferentes versões de sistemas operacionais e de configurações de dispositivos móveis. Um estudo de caso foi realizado para aplicar e avaliar a abordagem MTEst em duas aplicações móveis.

Xu et al. (2014) propõem uma técnica de modelagem chamada *Coarse-Grained GUI Model* (CGGM) que se baseia em MEFs. A técnica proposta objetiva a elaboração de modelos que representam funcionalidades sensíveis ao contexto com base na extração de metadados da aplicação em teste. Dois estudos de caso foram conduzidos para exemplificar o uso da técnica de modelagem proposta.

Zaeem et al. (2014) apresentam uma abordagem baseada em MEFs para a geração automatizada de casos de teste e a inclusão de oráculos de teste na verificação de aplicações móveis. Uma ferramenta chamada QUANTUM apoia a estratégia e contribui com o teste exaustivo de aplicações móveis a partir de uma biblioteca extensível que contempla eventos de interação entre usuários e dispositivos móvel. Uma avaliação experimental foi realizada em seis aplicações Android e relata a efetividade da abordagem proposta e da ferramenta QUANTUM na detecção de defeitos críticos.

Na Tabela 2 são apresentadas as pesquisas que investigam o TBM no contexto de aplicações móveis e características são relacionadas aos estudos. Destaca-se que todas as pesquisas direcionam esforços aos aspectos de modelagem (23 estudos). A maioria propõe e/ou adota abordagens específicas que objetivam promover avanços ao teste de aplicações móveis (22 estudos). Pesquisas que definem e/ou implementam ferramentas de apoio (18 estudos) e, principalmente, estudos orientados à geração de casos de teste (17 estudos) fornecem avanços ao TBM em aplicações móveis. Entretanto, nota-se a escassez de avaliações em ambientes industriais (quatro estudos). Por fim, também se observa que a etapa de concretização (oito estudos), que demanda um considerável esforço manual, e a de execução de casos de teste (14 estudos) são secundariamente investigadas.

Tabela 2 - Pesquisas de TBM em aplicações móveis

Pesquisas	**Modelagem**	**Geração de testes**	**Concretização**	**Execução dos testes**	**Abordagem específica**	**Ferramenta de apoio**	**Avaliação na indústria**
	23	**17**	**8**	**14**	**22**	**18**	**4**
(JÄÄSKELÄINEN et al., 2008a)	✓	✓		✓	✓	✓	
(JÄÄSKELÄINEN et al., 2008b)	✓				✓		
(JÄÄSKELÄINEN et al., 2009a)	✓	✓	✓	✓	✓	✓	
(JÄÄSKELÄINEN et al., 2009b)	✓				✓		✓
(RIDENE; BARBIER, 2011)	✓				✓	✓	✓
(TAKALA et al., 2011)	✓	✓	✓	✓	✓	✓	
(DEV et al., 2012)	✓	✓	✓	✓		✓	
(JANICKI et al., 2012)	✓	✓	✓	✓	✓	✓	✓
(LU et al., 2012)	✓	✓	✓		✓		
(PÜSCHEL et al., 2012)	✓	✓			✓	✓	
(JENSEN et al., 2013)	✓			✓	✓	✓	
(SALVA et al., 2013)	✓	✓		✓	✓	✓	
(SALVA; ZAFIMIHARISOA, 2013)	✓	✓		✓	✓	✓	
(YANG et al., 2013)	✓	✓		✓	✓	✓	
(AMALFITANO et al., 2014a)	✓	✓		✓	✓	✓	
(AMALFITANO et al., 2014b)	✓	✓			✓	✓	
(COSTA et al., 2014)	✓	✓		✓	✓	✓	
(GRIEBE; GRUHN, 2014)	✓	✓	✓	✓	✓		
(LI et al., 2014)	✓	✓	✓	✓	✓	✓	✓
(SCHWEIGHOFER; HERIČKO, 2014)	✓	✓			✓		
(TAO; GAO, 2014)	✓				✓	✓	
(XU et al., 2014)	✓				✓	✓	
(ZAEEM et al., 2014)	✓	✓	✓	✓	✓	✓	

3.9. CONSIDERAÇÕES FINAIS

Neste capítulo, foram apresentados os resultados de um mapeamento sistemático na área de teste de aplicações móveis. A condução do estudo contribuiu com a identificação e análise de 108 pesquisas relevantes desenvolvidas ao longo dos últimos anos pela comunidade acadêmica. Tais pesquisas foram classificadas quanto as suas características. Além de avaliar a quantidade de publicações e a adoção de TBM no contexto de mobilidade, buscou-se verificar os principais tópicos ou termos utilizados em títulos e resumos, a quantidade de pesquisas com escopo exploratório, experimental e industrial, as principais abordagens para modelagem ou execução do teste, os principais tipos e/ou técnicas para teste e, ao final, as plataformas de desenvolvimento e sistemas operacionais adotados. Posteriormente, estudos que fornecem avanços ao TBM em aplicações móveis foram destacados.

No próximo capítulo, um estudo experimental é conduzido para avaliar uma abordagem de TBM e ESG na verificação de aplicações móveis desenvolvidas em Android. A avaliação objetiva evidenciar a aplicabilidade, os resultados e os desafios, bem como a capacidade de detecção de defeitos.

4. TBM NO CONTEXTO DE APLICAÇÕES MÓVEIS

4.1. CONSIDERAÇÕES INICIAIS

O mapeamento sistemático apresentado no Capítulo 3 fornece evidências acerca do crescente interesse por pesquisas na área de teste de aplicações móveis. Estudos relevantes que adotam o processo de TBM no contexto de aplicações móveis foram descritos.

Neste capítulo, é apresentado um estudo experimental realizado com o objetivo de avaliar o uso de TBM e ESG no contexto de aplicações móveis. Na Seção 4.2, a configuração do estudo é descrita. Na Seção 4.3, a análise dos resultados identificados é apresentada. Nas Seções 4.4 e 4.5, são evidenciadas as limitações e as lições aprendidas, respectivamente. Na Seção 4.6, é relatada a discussão dos resultados.

O estudo experimental descrito neste capítulo também é relatado no *paper* "*Evaluating the Model-Based Testing Approach in the Context of Mobile Applications*", *Farto, G. C., Endo, A. T.*, publicado e apresentado na XL Latin American Computing Conference (CLEI), Montevidéu, Uruguai. Posteriormente convidado para publicação em uma edição especial da *Electronic Notes in Theoretical Computer Science* (ENTCS), DOI: 10.1016/j.entcs.2015.05.001.

4.2. CONFIGURAÇÃO DO ESTUDO

Devido à ampla expansão no número e diversidade de usuários e dispositivos móveis, abordagens, técnicas e ferramentas de apoio ao teste automatizado são essenciais para melhor elaborar, conduzir e analisar a ocorrência de defeitos em aplicações móveis. Junto a essa distinta configuração de ambiente de teste, constatam-se as particularidades e dificuldades mencionadas por pesquisas como explorado por Muccini et al. (2012) e Wasserman (2010). Além

disso, devem-se considerar pontos de interesse para a indústria como, por exemplo, (i) baixo custo e (ii) tempo reduzido para configuração e execução de testes.

Pretende-se, por meio de um estudo experimental, avaliar o uso de TBM e ESG no contexto de aplicações móveis e, dessa forma, contribuir com evidências para as seguintes questões de pesquisa:

- ***Q1***. “Os conceitos de TBM podem ser utilizados, em seu estado atual, para verificar requisitos funcionais em aplicações móveis?”;
- ***Q2***. “Quais os resultados e desafios identificados a partir de testes automatizados com TBM em aplicações móveis?”; e
- ***Q3***. “Quão efetivos, em relação à detecção de defeitos, foram os modelos e casos de teste gerados, concretizados e executados na aplicação móvel avaliada?”.

O estudo experimental realizado envolveu um grupo de 15 pessoas composto por cinco profissionais que atuam no desenvolvimento de aplicações móveis em Android e dez alunos de um curso de graduação em Ciência da Computação. Os profissionais integram uma célula de mobilidade de uma empresa multinacional de TI e possuem de três a cinco anos de experiência com desenvolvimento de aplicações móveis. Os alunos colaboraram com o experimento durante uma aula em Tópicos Avançados que é ministrada no último ano do curso de Ciência da Computação, inclusive por abranger aulas teóricas e práticas em Android.

Aplicação utilizada: AddressBook App. Inicialmente, uma aplicação móvel desenvolvida em Android e apresentada como projeto didático em Deitel et al. (2012) foi selecionada. A aplicação móvel chamada “*AddressBook App*” é implementada no Capítulo 10 de Deitel et al. (2012) e contempla o escopo de uma agenda de contatos. O desenvolvimento de *AddressBook App* utiliza diferentes conceitos e recursos tecnológicos da plataforma Android que são incorporados a inúmeras aplicações existentes. Portanto, a escolha por *AddressBook App* objetiva contemplar elementos básicos, porém indispensáveis para uma aplicação Android.

Ao executar a aplicação, uma tela listagem dos contatos cadastrados é exibida por meio de uma classe *ListActivity* que manipula componentes visuais para exibição de itens em uma lista. Além da listagem de contatos, outra *Activity* é

utilizada para as operações de inserção, quando um novo contato é preenchido, e atualização, quando se deseja modificar alguma informação previamente cadastrada. Para a atualização do contato, informações são compartilhadas entre a tela de listagem e a de alteração, permitindo identificar qual o contato a ser modificado. A operação a ser realizada, inclusão ou alteração de contatos, é selecionada a partir de um menu de opções que é criado e manipulado por intermédio de uma classe *MenuInflater*. Os registros de contatos são persistidos e manipulados por um banco de dados embarcado nativo à plataforma Android chamado de *SQLite Database* (SQLITE, 2015). Por fim, a aplicação utiliza janelas de notificação por *Toast* para lançar mensagens de aviso. Na Figura 13 são ilustradas telas da aplicação *AddressBook App*, proposta por Deitel et al. (2012).

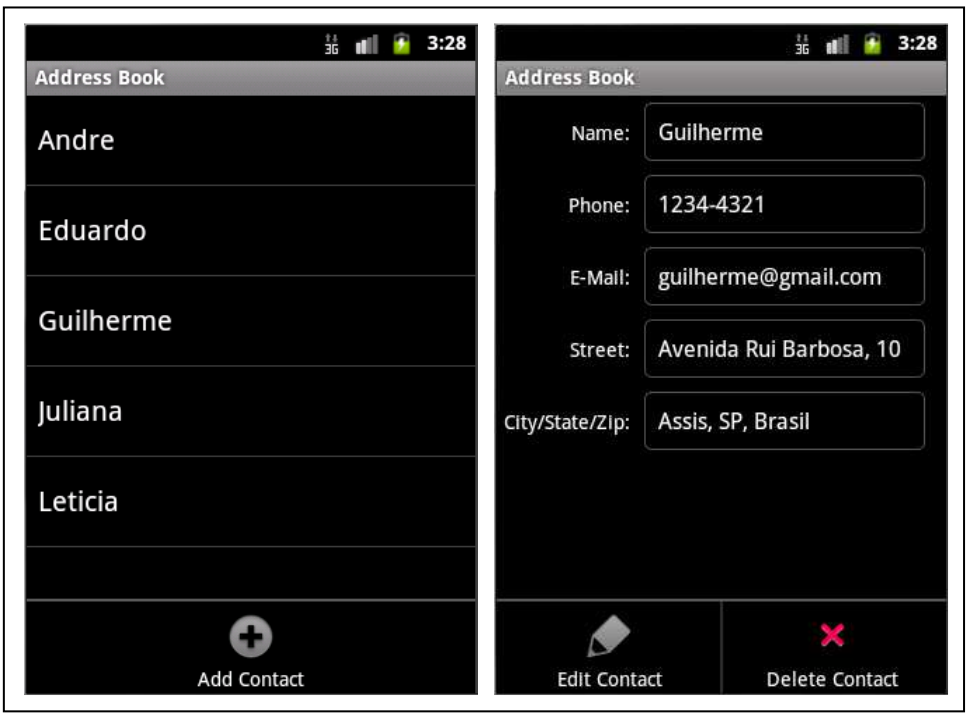

Figura 13 - Telas da aplicação *AddressBook App*
Fonte: Adaptada de Deitel et al. (2012)

Após uma breve explanação do propósito do estudo experimental, os envolvidos foram divididos em três grupos, dos quais dois se compuseram por cinco alunos cada (Grupos 1 e 2), enquanto o último era composto pelos desenvolvedores (Grupo 3). Posteriormente, cada grupo recebeu um documento contendo instruções descritas em Deitel et al. (2012). Os grupos atuaram no desenvolvimento de sua própria versão da aplicação a ser avaliada. Dessa forma, os participantes do

experimento puderam compreender e identificar os detalhes da aplicação *AddressBook App*, auxiliando-os na etapa de modelagem com ESG.

A atividade de preparação do ambiente de desenvolvimento em Android, implementação do projeto selecionado e testes manuais da aplicação móvel em um emulador foi concluída em 50 minutos.

Sessão de treinamento. A implementação de *AddressBook App* possibilitou um maior entendimento acerca da arquitetura e elementos tecnológicos que compõem a aplicação móvel utilizada no experimento. Entretanto, os envolvidos não possuíam conhecimentos ou experiências quanto aos conceitos do processo de TBM e da técnica de modelagem ESG.

Para isso, uma sessão de treinamento foi realizada para explicar os principais fundamentos, apresentar um breve histórico das pesquisas relacionadas e elencar benefícios e avanços na atividade de teste de software quando se utilizam processos automatizados. O treinamento foi concluído com uma demonstração prática que exemplificou a modelagem de requisitos funcionais de aplicações móveis em ESG. A ferramenta yEd *Graph Editor*[4] foi adotada na etapa de modelagem.

A sessão de treinamento consumiu 30 minutos, incluindo o necessário para esclarecer dúvidas e comentários expostos pelos envolvidos. Pôde-se notar que o objetivo de fundamentar e treinar os participantes do experimento em TBM e ESG foi alcançado ao finalizar o exemplo de modelagem proposto.

Uso de Robotium. Estudos exploratórios e implementações práticas foram desenvolvidos com *Monkey*, *MonkeyRunner*, *Robolectric*, *Espresso* e *Robotium* (FARTO; ENDO, 2015a; FARTO; ENDO, 2015b; FARTO; ENDO, 2015c). Dentre as plataformas de teste apresentadas, optou-se pela adoção de *Robotium* por se destacar pelos aspectos descritos a seguir.

(i) Possibilitar a construção de casos de teste com pouco conhecimento da aplicação a ser testada, pois permite testes funcionais e estruturais, além de ser possível testar em dispositivos reais e/ou emuladores;

(ii) Contribuir com uma aprendizagem rápida, tendo em vista que a API se baseia em Java e JUnit, dispondo de uma documentação com exemplos que demonstram sua aplicabilidade; e

[4] http://www.yworks.com/products/yed/

(iii) Incorporar recursos para a manipulação e verificação de diferentes componentes do Android.

Resumidamente, as principais classes e métodos da plataforma de teste *Robotium* são descritos no Apêndice B.

4.3. ANÁLISE DE RESULTADOS

Para relatá-los a partir das etapas do processo de TBM, os resultados são analisados sob as perspectivas de (i) modelagem, (ii) geração de testes e concretização e (iii) execução dos testes.

Modelagem. Amparados pelos conhecimentos de TBM e ESG explorados e aplicados durante a sessão de treinamento, foi requisitado aos grupos que elaborassem um modelo de teste em ESG para representar os requisitos funcionais da aplicação móvel *AddressBook App*. A atividade de modelagem foi conduzida em sequência ao treinamento e finalizada em aproximadamente 40 minutos. Nas Figuras 14, 15 e 16 são ilustrados, respectivamente, os modelos ESG obtidos a partir do entendimento dos envolvidos dos Grupos 1, 2 e 3.

O modelo ESG elaborado pelo Grupo 1, apresentando na Figura 14, contempla os requisitos funcionais da aplicação móvel, porém dois nós ausentes o torna incompleto: (i) um evento para exibir uma mensagem de notificação quando um contato é selecionado na listagem e (ii) um evento para exibir uma janela de diálogo de confirmação quando o usuário deseja remover um contato selecionado. Verifica-se, também, a falta dos nós especiais de início e término do modelo ESG.

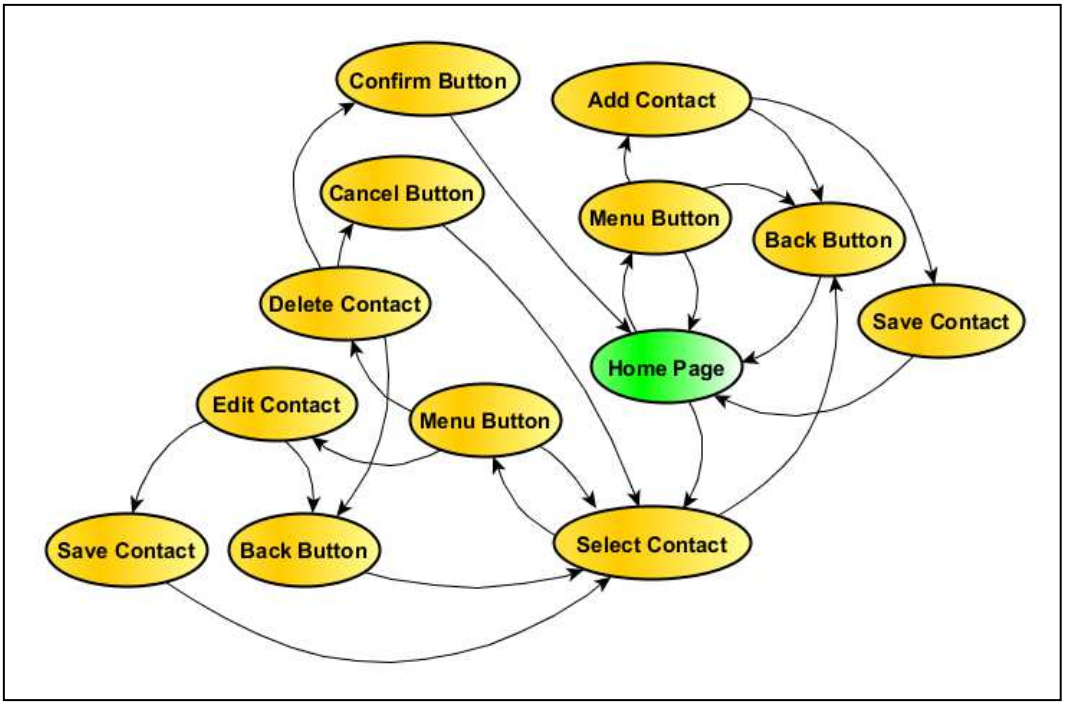

Figura 14 - Modelo ESG elaborado pelo Grupo 1 (cinco alunos)

O modelo ESG elaborado pelo Grupo 2, apresentado na Figura 15, atende os requisitos funcionais da aplicação móvel e, inclusive, possui um nó para exibir uma mensagem de notificação ao selecionar um contato na listagem. Entretanto, assim como o modelo ESG do Grupo 1, não considera a exibição de uma janela de confirmação ao optar pela remoção de um registro. Não constam os nós especiais de início e término do modelo ESG.

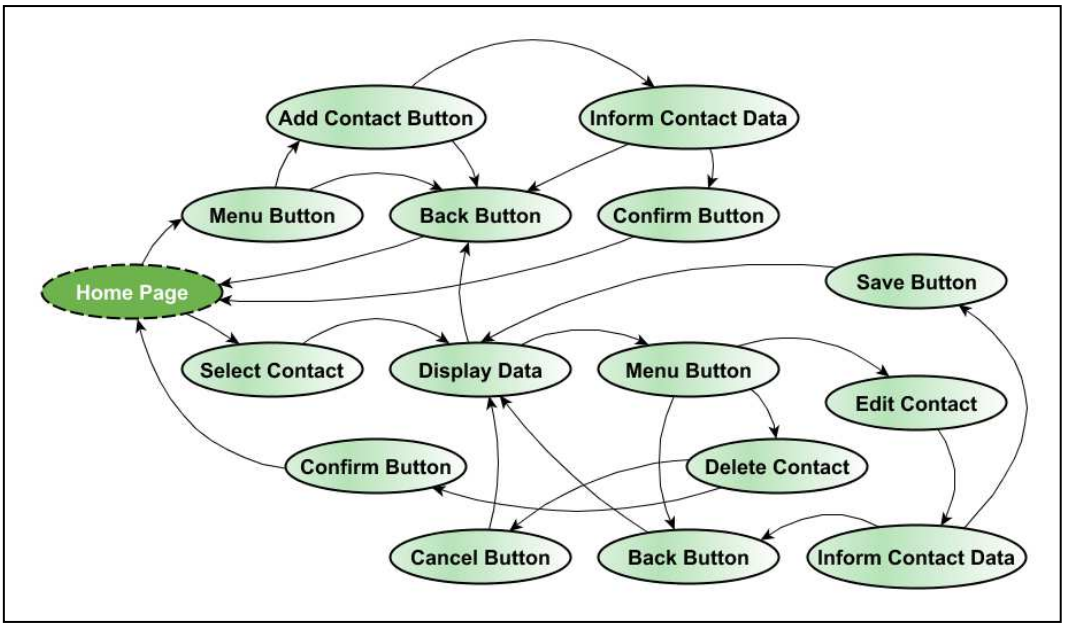

Figura 15 - Modelo ESG elaborado pelo Grupo 2 (cinco alunos)

O modelo ESG elaborado pelo Grupo 3, apresentado na Figura 16, é válido quanto à representação de requisitos funcionais da aplicação móvel e, quando comparado aos demais modelos do estudo experimental, contempla a mensagem de seleção e a de confirmação ao remover o contato selecionado. Porém, consta apenas um nó para o evento de pressionar o botão “Voltar”. Dessa forma, independentemente se o estado atual da aplicação é inserir um contato ou alterar um contato existente, o botão “Voltar”, ao ser pressionado, fará com que a aplicação móvel *AddressBook App* retorne à tela principal. Ao pressionar o botão “Voltar”, os comportamentos corretos são: (i) retornar à tela principal quando em modo de inserção e (ii) retornar à tela de visualização de dados do contato quando em modo de alteração. Mais uma vez, inexistem os nós especiais que delimitam o início e término do ESG.

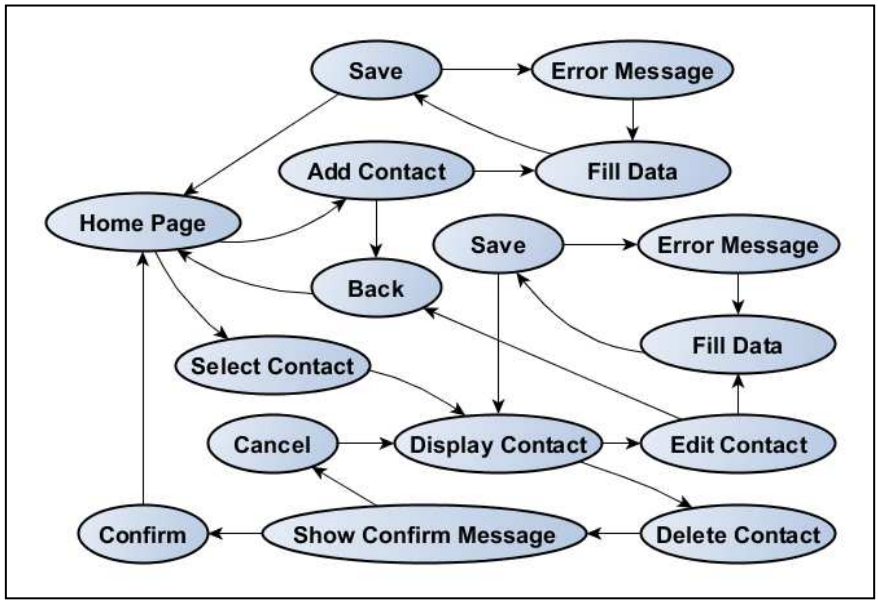

Figura 16 - Modelo ESG elaborado pelo Grupo 3 (cinco profissionais)

Ao analisar os modelos ESG, constatou-se que os três artefatos de modelagem elaborados pelos grupos poderiam ser utilizados na geração e concretização dos casos de teste, porém não seriam suficientes para uma verificação completa de requisitos funcionais da aplicação *AddressBook App*. Além disso, os eventos dos modelos não foram rotulados de maneira padronizada. Por exemplo, para representar a ação de pressionar o botão “Voltar”, os grupos utilizaram as expressões “*Back Button*” e “*Back*”. Para um entendimento mais claro e

para simplificar as etapas de geração e concretização dos casos de teste, o mesmo requisito funcional pode ser descrito como “*Press Back*”.

Uma nova modelagem com base nas versões apresentadas pelos grupos foi proposta pelos pesquisadores. Na Figura 17 é ilustrado um modelo ESG que representa os requisitos funcionais identificados na aplicação móvel de modo completo quanto às funcionalidades da aplicação e padronizada nos aspectos de nós e rótulos. É importante ressaltar que a validação do modelo ESG é de fundamental importância, pois a qualidade e efetividade da abordagem de TBM estão diretamente relacionadas à qualidade do modelo elaborado.

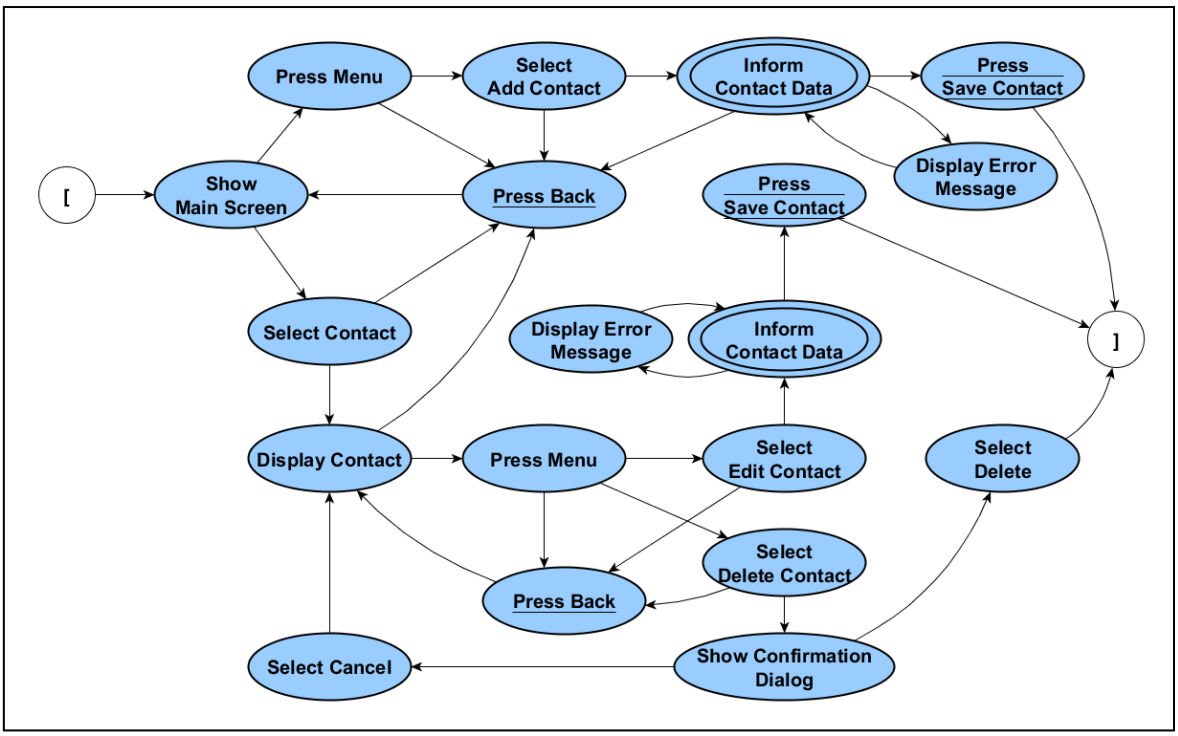

Figura 17 - Modelo ESG proposto pelos pesquisadores

Geração de testes e concretização. A geração de casos de teste abstratos e a concretização em casos de teste executáveis foram conduzidas pelos pesquisadores. A ferramenta *Test Suite Designer* (TSD) foi adotada, pois possibilita extrair CESs por meio de algoritmos que identificam sequências válidas de eventos a partir de um modelo ESG (BELLI et al., 2013). Utilizando-se o modelo apresentado na Figura 17 e, ao aplicar um algoritmo implementado na ferramenta TSD que se baseia no *Chinese Postman Problem* (AHO et al., 1995), obtêm-se uma ou mais

CESs que representam o caminho mais curto entre os nós de início e término do grafo, além de percorrer todas as arestas definidas no modelo ao menos uma vez.

Os espaços existentes nos rótulos dos nós foram substituídos por *underscore* ("_") e o evento "*Inform Contact Data*", representado duas vezes para distintos comportamentos, receberam um sufixo de "R1" e "R2". Essa padronização nos rótulos pode ser realizada no modelo ESG antes do processamento da ferramenta TSD, pois objetiva definir identificadores únicos para os eventos, bem como os considerar como sugestões para nomes de métodos de teste na etapa de concretização dos casos de teste. A ferramenta TSD gerou três CESs com 7, 20 e 22 eventos, respectivamente. Tais CESs são apresentadas a seguir.

```
    [, Show_Main_Screen, Press_Menu, Select_Add_Contact,
Inform_Contact_Data_R1, Display_Error_Message, Inform_Contact_Data_R1,
Press_Save_Contact, ]
```

```
    [, Show_Main_Screen, Select_Contact, Display_Contact, Press_Menu,
Select_Delete_Contact, Press_Back, Display_Contact, Press_Menu,
Select_Edit_Contact, Press_Back, Display_Contact, Press_Menu,
Select_Delete_Contact, Show_Confirmation_Dialog, Select_Cancel,
Display_Contact, Press_Menu, Select_Delete_Contact,
Show_Confirmation_Dialog, Select_Delete, ]
```

```
    [, Show_Main_Screen, Press_Menu, Press_Back, Show_Main_Screen,
Select_Add_Contact, Inform_Contact_Data_R1, Press_Save_Contact,
Show_Main_Screen, Select_Contact, Press_Back, Show_Main_Screen,
Select_Contact, Display_Contact, Press_Menu, Press_Back, Display_Contact,
Press_Menu, Select_Edit_Contact, Inform_Contact_Data_R2,
Display_Error_Message, Inform_Contact_Data_R2, Press_Save_Contact, ]
```

As CESs devem ser concretizadas para que cada evento mapeado no modelo ESG se torne executável na aplicação em teste. Por exemplo, "pressionar um determinado botão ou tecla", "preencher um campo de texto" ou "verificar alguma mensagem de notificação foi exibida".

Um fragmento da classe `AddressBookTestCase` que demonstra a concretização de casos de teste com a plataforma *Robotium* é apresentado no Apêndice C. Todos os casos de teste abstratos do estudo experimental foram

concretizados da mesma maneira. A classe `TestCaseUtil`, utilizada na concretização, disponibiliza um método genérico responsável pela execução das CESs geradas pela ferramenta TSD. A aplicação *AddressBook App* e *AddressBookTest*, contendo os testes com *Robotium*, foram compartilhados em:

https://github.com/guilhermefarto/AddressBookTest

Uma versão simplificada do fragmento da classe concretizada com *Robotium* é apresentada a seguir. O método `Press_Menu()` simula a ação de um usuário ao pressionar o botão “Menu”. A instância de `solo` é utilizada e o método `sendKey()` da API de *Robotium* é invocado com o valor de parâmetro igual a `KEYCODE_MENU`. Outro exemplo de concretização é visualizado no método `Select_Add_Contact()`, responsável pela ação de pressionar um item de menu. Para isso, o método `clickOnMenuItem()` da instância de `solo` é invocado e o item de menu é definido pela constante `R.string.menuitem_add_contact`.

```
// The full implementation have been omitted
public class AddressBookTestCase extends ActivityInstrumentationTestCase2 {

    public void testCesOne() throws Exception {
        String cesOne = "[, Show_Main_Screen, Press_Menu, Press_Back,
            Show_Main_Screen, Select_Add_Contact, Inform_Contact_Data_R1,
            Press_Save_Contact, Show_Main_Screen, Select_Contact,
            Press_Back, Show_Main_Screen, Select_Contact, Display_Contact,
            Press_Menu, Press_Back, Display_Contact, Press_Menu,
            Select_Edit_Contact, Inform_Contact_Data_R2,
            Display_Error_Message, Inform_Contact_Data_R2,
            Press_Save_Contact, ]";

        TestCaseUtil.executeTestCase(this, cesOne);
    }

    public void Press_Menu() throws Exception {
        this.solo.sendKey(KeyEvent.KEYCODE_MENU);
    }

    public void Select_Add_Contact() throws Exception {
        this.solo.clickOnMenuItem(
            this.solo.getString(R.string.menuitem_add_contact));
    }
}
```

Execução dos testes. Um AVD foi configurado, simulando um dispositivo real de uso da aplicação *AddressBook App*. A configuração do AVD inclui a versão 2.3.3 (API 10) da plataforma Android, uma tela de 3.2” com qualidade média (*mdpi*), 512

MB de memória RAM e 50 MB de cartão de memória. A aplicação foi instalada e executada no emulador. Ao ser iniciada, os casos de teste foram executados conforme a sequência definida pelos CESs, sendo possível acompanhar cada um dos requisitos funcionais. Dados referentes à etapa de execução dos testes foram coletados, focando-se no (i) tempo de execução e em (ii) defeitos identificados.

A execução dos casos de teste concretizados com *Robotium* consumiu 56,8 segundos para a CES com 7 eventos, 79,3 segundos para a CES com 20 eventos e, por fim, 163,7 segundos para a CES com 22 eventos. Dessa forma, os casos de teste foram executados em 299,8 segundos ou aproximadamente 5 minutos. Apesar da baixa complexidade da aplicação móvel e dos casos de teste, constatou-se que o tempo total foi elevado. Justifica-se essa demora pelo fato de que a plataforma *Robotium* demonstra visualmente a execução dos casos de teste. Tal aspecto pode ser uma vantagem, pois permite que os testadores acompanhem a execução e comportamento da aplicação móvel em teste. Entretanto, a espera pela conclusão dos testes pode se tornar crítica em aplicações móveis complexas que necessitam de casos de teste mais extensos ou quando se deseja reexecutar o teste em distintas configurações de dispositivos móveis.

Na Tabela 3 são apresentadas as linhas de código (em Inglês, *Lines of Code* – LoC) e a complexidade ciclomática de McCabe (MCCABE, 1976) para os projetos *AddressBook App* e *AddressBookTest*. Os valores para média, desvio padrão e máxima complexidade ciclomática se mantiveram próximos entre as aplicações. A proximidade de valores se deve pela implementação do método genérico `executeTestCase()` da classe `TestCaseUtil`, responsável pela execução dinâmica de CESs e que pode ser utilizada em testes mais complexos.

Tabela 3 - Dados sumarizados para *AddressBook App* e *AddressBookTest*

Aplicação móvel	**LoC**	**Complexidade Ciclomática de McCabe**		
		Média	**Desvio Padrão**	**Máxima**
AddressBook App	416	1,22	0,497	3
AddressBookTest	124	1,23	0,516	3

A execução dos casos de teste concretizados com *Robotium* resultou na identificação de quatro defeitos na aplicação *AddressBook App* apresentados a seguir.

- ***(1) Inclusão e alteração de contatos com valor apenas no atributo "nome"***. A aplicação *AddressBook App* possibilita a inclusão e alteração de contatos que possuem apenas o campo "nome do contato" preenchido. A detecção do defeito foi possível, pois o caso de teste verifica se os campos de telefone e endereço eletrônico foram preenchidos antes de persistir o contato.
- ***(2) Inclusão e alteração de contatos com valores numéricos no atributo "nome"***. A aplicação *AddressBook App* possibilita a inclusão e alteração de contatos que possuem apenas o campo "nome do contato" preenchido com valores numéricos como, por exemplo, "1234". A detecção do defeito foi possível, pois o caso de teste verifica se os campos de telefone e endereço eletrônico foram preenchidos antes de persistir o contato, assim como no Defeito 1. Além disso, um teste é conduzido para verificar a existência apenas de caracteres alfabéticos.
- ***(3) Inclusão de contatos com valores repetidos***. A aplicação *AddressBook App* possibilita a inclusão de contatos com valores repetidos. Dessa forma, contatos com nomes idênticos podem ser armazenados. A detecção do defeito foi possível, pois o caso de teste verifica se dois ou mais contatos são persistidos com os mesmos valores para o campo nome.
- ***(4) Falta de mensagem de notificação***. A aplicação *AddressBook App* não exibe mensagem de notificação quando uma ou mais situações mencionadas anteriormente são identificadas.

4.4. LIMITAÇÕES

O estudo apresentado foi conduzido para avaliar a aplicabilidade do TBM e ESG no contexto de aplicações móveis. Os resultados obtidos e analisados fornecem evidências para as questões de pesquisa exploradas e motivam trabalhos futuros. Entretanto, o experimento apresenta limitações que impossibilitam a

generalização dos resultados para outros cenários de testes automatizados em mobilidade.

Como mencionado, os pesquisadores atuaram na geração e concretização dos casos de teste por meio da ferramenta TSD e da plataforma *Robotium*. Optou-se por esse modelo nas etapas citadas, pois o objetivo principal do estudo era verificar a adoção de uma abordagem de TBM e ESG e não o uso das ferramentas de apoio. Outro tópico importante é a seleção de apenas uma aplicação móvel, mesmo que apresente uma arquitetura semelhante a outras aplicações.

A escolha por abordagens e ferramentas para TBM se torna uma possível limitação. Não há consenso na literatura acerca de uma única técnica e/ou ferramenta que se caracteriza como uma boa ou recomendada abordagem para TBM. Dessa forma, determinados recursos utilizados junto ao processo de TBM podem não ser representativos ou adequados para outras situações.

4.5. LIÇÕES APRENDIDAS

Orientações e lições aprendidas foram identificadas quanto à avaliação experimental e são apresentadas a seguir.

Modelagem e validação de modelos ESG. As dificuldades detectadas contribuíram com a definição de orientações que podem melhorar a modelagem e validação de modelos ESG que definem requisitos funcionais em aplicações móveis.

Os nós especiais, “[”e “]”, que representam o início e término do modelo ESG foram substituídos por eventos com rótulos. A ausência desses elementos na modelagem dificulta a identificação dos nós de entrada e de saída do grafo. Além disso, as ferramentas necessitam de recursos de notação ou marcação para que o modelo ESG possa ser manipulado automaticamente. Dessa forma, o uso dos nós especiais é uma orientação relevante e deve ser seguida na modelagem com ESG.

Outro ponto de atenção é a falta de padrão ao descrever os rótulos dos eventos. É importante definir rótulos concisos que definam o requisito funcional. Como boa prática, recomenda-se seguir uma norma como, por exemplo, “Pressionar XYZ”, “Selecionar XYZ”, “Exibir XYZ” e “Preencher XYZ”.

Alguns eventos são descritos com rótulos idênticos, porém representam distintos comportamentos na aplicação móvel e dependem do contexto da aplicação. Por exemplo, o botão “Salvar” desempenha a inclusão ou alteração de um registro de acordo do estado da aplicação. Dependendo do contexto e da aplicação móvel, pode ser que uma única concretização seja necessária. Caso contrário, cada evento deverá ser unicamente identificado na modelagem ou na concretização. Portanto, deve-se atentar à possibilidade de diferentes comportamentos de requisitos funcionais quando mapeados com rótulos iguais.

4.6. DISCUSSÃO DOS RESULTADOS

A execução e coleta dos resultados do estudo experimental fornecem subsídios que contribuem na resolução da questão de pesquisa Q1 (“*Os conceitos de TBM podem ser utilizados, em seu estado atual, para verificar requisitos funcionais em aplicações móveis?*”). As evidências identificadas possibilitam afirmar que o processo de TBM e a técnica de modelagem ESG podem ser incorporados como uma abordagem válida para automação de testes em aplicações Android.

Com o intuito de complementar a assertiva dada como resposta à Q1, percebe-se que o uso de *Robotium* é uma interessante alternativa dentre demais ferramentas e plataformas de teste disponíveis em Android para as etapas de concretização e execução dos casos de teste. Os recursos providos pela plataforma *Robotium* proporcionaram uma rápida concretização dos casos de teste, além de viabilizar o acompanhamento visual dos requisitos funcionais testados no AVD. Por meio de uma API extensa e documentada, eventos e interações podem ser simulados para interagir de diferentes maneiras com a aplicação móvel.

Os resultados constatados podem ser relacionados à questão de pesquisa Q2 (“*Quais os resultados e desafios identificados a partir de testes automatizados com TBM em aplicações móveis?*”), destacando:

- ***Geração automática de casos de teste***. Particularmente no contexto de aplicações móveis, a elaboração de modelos é uma importante característica do processo de teste, pois auxilia na definição de artefatos formais que descrevem as

funcionalidades e/ou cenários a serem verificados. Como apresentado, a geração de casos de teste com base nos modelos ESG resulta em CESs que contemplam os requisitos funcionais mapeados da aplicação *AddressBook App*. Portanto, todos os eventos definidos em modelos de teste podem ser executados após a concretização, garantindo uma verificação completa e formal da aplicação móvel em teste.

- ***Efetividade na detecção de defeitos***. A execução dos casos de teste concretizados com *Robotium* revelou defeitos na aplicação *AddressBook App*. Dessa forma, confirma-se a capacidade de detecção de defeitos para a abordagem selecionada no estudo experimental, contribuindo positivamente com a questão de pesquisa Q3.

- ***Melhoria na qualidade do teste***. Testes automatizados reduzem a interferência humana no teste de software e colaboram com um processo formal. Uma vez elaborados, os modelos de teste podem ser adaptados e evoluídos para atender a novas funcionalidades implementadas. Além disso, casos de teste podem ser acrescidos a outros já concretizados, expandindo as situações e eventos verificados em testes anteriores. Ainda que não realizado no estudo experimental, instruções podem ser incorporadas aos casos de teste para (i) capturar imagens de telas durante o teste e (ii) armazenar informações extras do teste em arquivos de *logs* como, por exemplo, valores de campos e parâmetros da aplicação. Com isso, a rastreabilidade pode simplificar a análise de uma ou mais funcionalidades que resultaram em defeitos durante o teste automatizado.

- ***Redução no tempo e custo do teste***. No contexto de aplicações móveis, a possibilidade de retestar os mesmos requisitos funcionais otimiza a verificação formal de funcionalidades em distintas configurações de dispositivos reais e/ou emuladores. Não somente por permitir o teste em novas versões de uma aplicação móvel testada, mas principalmente por conduzir exatamente o mesmo teste com diferentes versões do sistema operacional, diversas configurações de tamanhos e resoluções de telas e outras características de dispositivos móveis.

- ***Evolução dos modelos de teste***. A evolução de modelos de teste em ESG pode ser necessária para contemplar novas funcionalidades da aplicação móvel. Por meio da abordagem proposta, a inclusão de novos requisitos funcionais pode ser alcançada sem grandes esforços ou dificuldades. Para isso, o modelo ESG deve ser atualizado quanto aos novos eventos e reusado na geração de casos de teste. Em

seguida, os casos de teste existentes podem ser mantidos e, portanto, apenas os recém mapeados necessitam ser concretizados. A facilidade em concretizar os casos de teste com *Robotium* também é um fator importante identificado no estudo experimental. Dessa forma, a abordagem pode ser explorada por desenvolvedores com experiência, mas também por profissionais de equipes de qualidade de software com pouco ou nenhum conhecimento na plataforma Android.

O estudo experimental também auxiliou no reconhecimento de desafios quanto ao uso de TBM em aplicações móveis. As dificuldades observadas na modelagem e concretização de casos de teste apoiaram a definição de lições aprendidas. Os desafios constatados são apresentados a seguir e complementam os resultados discutidos que se relacionam à questão de pesquisa Q2.

- ***Dificuldades na elaboração de modelos de teste***. A elaboração de modelos ESG pode ser uma atividade complexa por exigir conhecimentos específicos da aplicação móvel em teste. Entretanto, a própria tarefa de modelar pode ocasionar certa dificuldade ao testador, prejudicando a construção de modelos de teste concisos e de fácil manipulação por ferramentas. Alguns conceitos explicados na sessão de treinamento foram desconsiderados como o uso de nós especiais e a definição de rótulos padronizados. Além disso, há a possibilidade de dois ou mais eventos serem descritos com o mesmo rótulo, porém representam distintos requisitos funcionais ou comportamentos. Por exemplo, o rótulo "*Inform Contact Data*" é modelado duas vezes, pois especifica a ação do usuário de preencher dados do contato para a inclusão e, com outros valores, para a alteração. Entretanto, o rótulo "*Press Save Contact*", que também é duplicado pelo mesmo motivo, pode ser concretizado de uma única forma, pois representa a ação de salvar e, nesse contexto, não é preciso identificar se é uma inclusão ou alteração.

- ***Particularidades na concretização de casos de teste***. Particularidades do contexto de aplicações móveis devem ser consideradas na concretização de casos de teste como a existência de botões físicos dos dispositivos como "Voltar" e "Menu", além de comportamentos que não precisam ser modelados, pois são executados por outros eventos, como ocorre na transição de *Activities*. Como mencionado, requisitos funcionais podem ser identicamente rotulados para representar diferentes comportamentos e estados da aplicação móvel. Para o estudo, a alternativa aplicada

ao evento “*Inform Contact Data*” foi a de acrescentar o sufixo “R#” para identificar unicamente cada comportamento antes da geração de casos de teste. Enquanto que para o evento “*Press Save Contact*” uma única concretização foi necessária, representando o evento de pressionar o botão “Salvar”.

A detecção de quatro defeitos na aplicação *AddressBook App* contribui com evidências que positivamente validam a questão de pesquisa Q3 (“*Quão efetivos, em relação à detecção de defeitos, foram os modelos e casos de teste gerados, concretizados e executados na aplicação móvel avaliada?*”). Ressalta-se que os defeitos mencionados podem ter sido detectados na aplicação móvel *AddressBook App* por ser um projeto simples e de cunho educacional. Entretanto, foi possível observar a capacidade de revelar defeitos em aplicações móveis por meio da abordagem formal de TBM e da técnica de modelagem ESG. Por fim, as etapas de modelagem com ESG e de concretização com a plataforma *Robotium* se mostraram estratégias promissoras ao TBM em aplicações móveis.

4.7. CONSIDERAÇÕES FINAIS

Neste capítulo, foi apresentado um estudo experimental para investigar a aplicabilidade do TBM em aplicações móveis. Buscou-se avaliar a adoção de TBM e ESG em uma aplicação móvel selecionada sem a proposição de elementos ou estratégias adicionais aos conceitos e etapas padrões.

Os resultados descritos neste capítulo provêem evidências de que o processo de TBM e a técnica de modelagem ESG podem ser incorporados na verificação de requisitos funcionais de aplicações móveis em seu estado atual. A plataforma de teste *Robotium* também foi apresentada e analisada como uma alternativa válida para concretizar casos de teste abstratos em Android. Os resultados identificados, (i) a geração automática de casos de teste e (ii) a capacidade de detecção de defeitos, fornecem subsídios para considerar o TBM uma abordagem aplicável e promissora ao contexto de mobilidade. Desafios também foram descritos, destacando-se (i) as dificuldades na modelagem de requisitos funcionais e (ii) as particularidades na concretização de casos de teste.

Ao considerar a aplicabilidade e os desafios apresentados por este capítulo, novas pesquisas podem ser conduzidas para explorar o teste de aplicações móveis junto ao processo de TBM. Portanto, as características de estudos relacionados e apresentados na Seção 3.8, bem como as lições aprendidas descritas na Seção 4.5 e os resultados discutidos na Seção 4.6, motivam a concepção, a implementação e a avaliação de abordagens fundamentadas em TBM para o contexto de aplicações móveis. Dentre os avanços, destaca-se a importância (i) de estratégias para reduzir o esforço demandado na concretização de casos de teste, (ii) de estratégias de testes que verificam distintas características de aplicações móveis e (iii) de ferramentas de apoio na automação de abordagens apoiadas por TBM.

Para tais questões, uma abordagem específica de TBM e uma ferramenta de apoio são apresentadas no próximo capítulo para contribuir com o teste de software no contexto de aplicações móveis.

5. ABORDAGEM DE TBM PARA REÚSO DE MODELOS

5.1. CONSIDERAÇÕES INICIAIS

Apesar do uso do TBM ter sido avaliado de acordo com os passos comumente apresentados na literatura, estratégias podem apoiar e contribuir com melhorias para o teste no contexto de aplicações móveis. Dessa forma, novas abordagens e ferramentas podem ser incorporadas ao TBM, aperfeiçoando e especializando o processo avaliado no Capítulo 4.

Neste capítulo, é apresentada uma abordagem que possibilita o reúso de artefatos de modelagem, mais precisamente os modelos de teste em ESG. Pretende-se, por meio da abordagem proposta, contribuir com (i) a redução do esforço demandado para a concretização de casos de teste, assim como (ii) o teste de distintas características diretamente relacionadas ao contexto de mobilidade. Na Seção 5.2, é apresentado um exemplo motivacional para elucidar tópicos de interesse e perspectivas de avanços no TBM em aplicações móveis. Na Seção 5.3, a abordagem proposta e sua respectiva definição formal, bem como os fundamentos propostos para o reúso de modelos de teste são descritos. Na Seção 5.4, uma ferramenta, *Model-Based Test Suite For Mobile Applications* (MBTS4MA), é projetada e desenvolvida para apoiar a automação da abordagem proposta.

5.2. EXEMPLO MOTIVACIONAL

Nesta seção, é apresentado um exemplo para revisitar a adoção de TBM em uma aplicação Android e motivar a proposta de uma abordagem orientada ao reúso de modelos de teste que promova avanços ao TBM em aplicações móveis. Para isso, foi selecionada a aplicação móvel “*RouteTracker*”, apresentada no Capítulo 11 de Deitel et al. (2012). A aplicação *RouteTracker* contempla recursos como o sensor de GPS e a API para manipulação do Google Maps.

A aplicação selecionada objetiva o registro de rotas a partir da coleta de posições geográficas de latitude e longitude. Os dados providos pelo sensor de GPS são armazenados em memória e exibidos como rotas em mapas da API da Google. Ao interagir com a aplicação, o usuário pode iniciar e parar a coleta de dados das rotas percorridas por meio de um botão. A aplicação também possibilita a mudança da orientação do dispositivo móvel entre paisagem e retrato, além de se adaptar a diferentes tamanhos de tela. Na Figura 18 são ilustradas telas da aplicação *RouteTracker*, apresentada por Deitel et al. (2012).

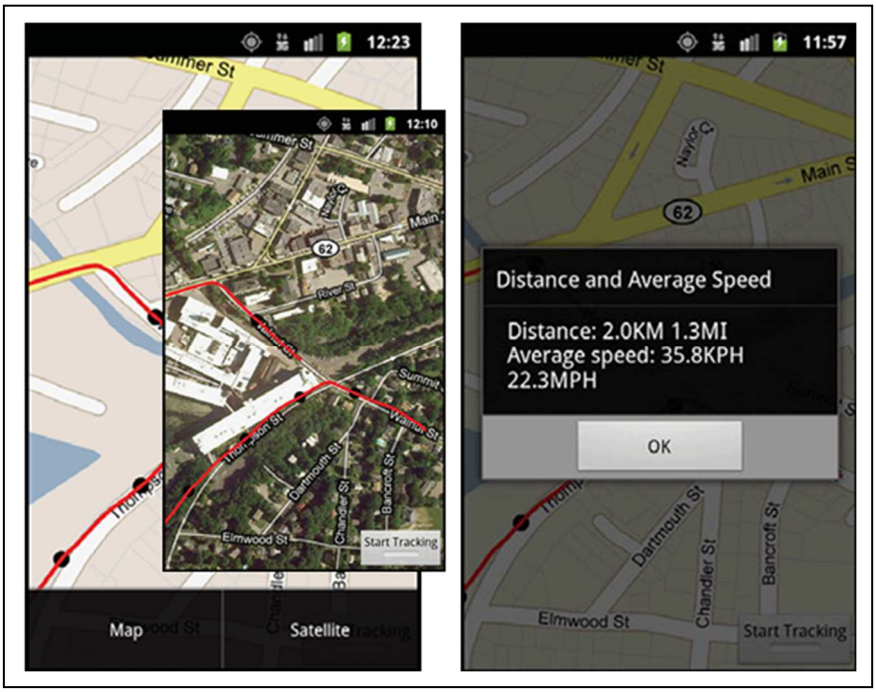

Figura 18 - Telas da aplicação *RouteTracker*
Fonte: Deitel et al. (2012)

Na Figura 19 é ilustrado um modelo ESG que define os eventos e suas sequências válidas da aplicação *RouteTracker*. Após a manipulação do modelo ESG por ferramentas e algoritmos de apoio ao TBM, CESs são geradas. Uma das possíveis CESs extraídas é apresentada a seguir.

```
     [, Display_tracking_screen, Press_Menu, Display_options, Press_Back,
Display_tracking_screen, Press_start_tracking, Listen_to_GPS,
Read_updated_location, Update_screen_map, Listen_to_GPS,
Press_stop_tracking, Show_distance_and_avg_speed, Press_OK,
Display_tracking_screen, Press_Back, ]
```

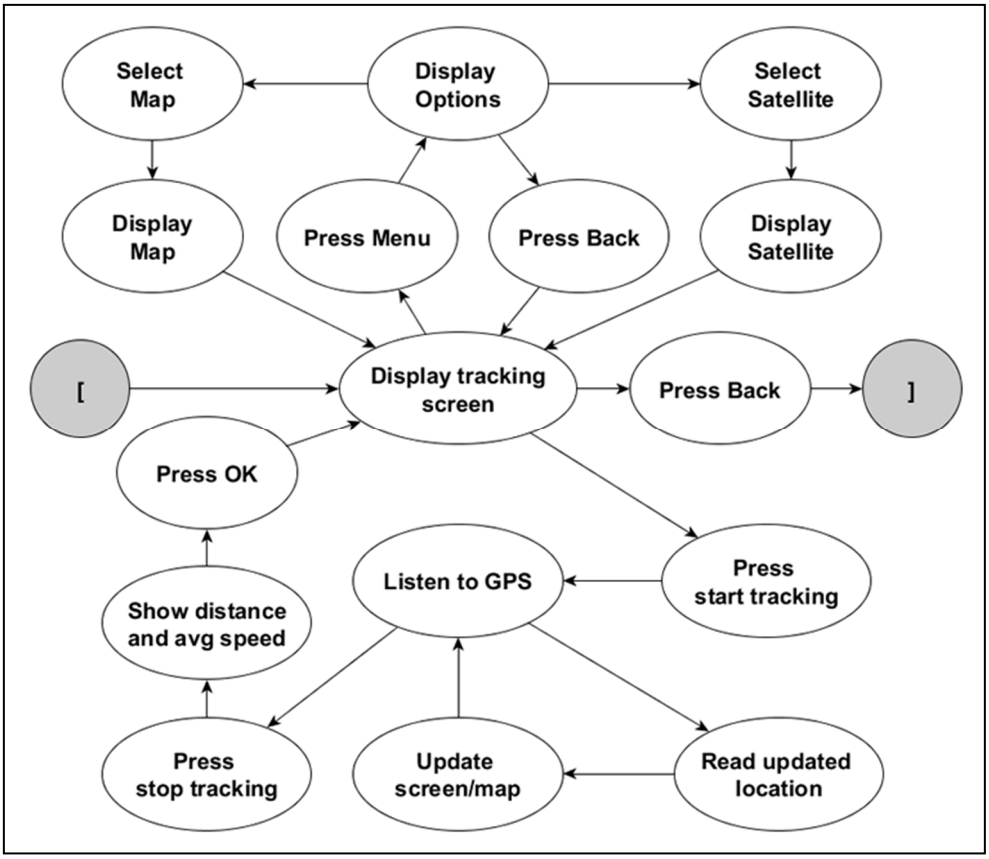

Figura 19 - Modelo ESG para *RouteTracker*
Fonte: Elaborada pelo autor

Na etapa de concretização, ferramentas e plataformas de teste tornam os casos de teste, até então abstratos, em implementações executáveis na aplicação em teste. Especificamente para a plataforma Android, o teste automatizado pode ser alcançado por meio de *Monkey*, *MonkeyRunner*, *Robolectric*, *Espresso* e *Robotium* (FARTO; ENDO, 2015a; FARTO; ENDO, 2015b; FARTO; ENDO, 2015c). A escolha da ferramenta e/ou plataforma de teste deve ser justificada por aspectos como, por exemplo, (i) facilidade de utilização e rápida aprendizagem e (ii) disponibilidade de recursos para a manipulação e verificação de componentes do SUT.

A aplicação móvel *RouteTracker* utiliza o sensor de GPS para capturar e consumir as informações geográficas. Portanto, a escolha da plataforma de teste deve considerar tal característica. Para demonstrar a concretização de casos de teste com *Robotium*, um fragmento da classe `RouteTrackerTestCase` é apresentado a seguir. Os eventos "*Press start tracking*" e "*Read updated location*" são, respectivamente, concretizados pelos métodos `Press_start_tracking()` e `Read_updated_location()` e representam a ação de pressionar o botão "*Start tracking*" e a captura de informações do GPS. A classe `TestLocationProvider`

representa um pseudosensor de GPS que simula posições geográficas de latitude e longitude. Da mesma maneira, classes utilitárias podem ser implementadas para verificar, por exemplo, (i) o envio e recebimento de mensagens, (ii) a interação com chamadas telefônicas e (iii) a manipulação de outros sensores. Como resultado, espera-se que a execução dos testes resulte na detecção dos defeitos.

```
public class RouteTrackerTestCase extends ActivityInstrumentationTestCase2 {

    // The full implementation of RouteTrackerTestCase have been omitted

    public void Press_start_tracking() {
        boolean actualTest = solo.searchToggleButton(solo.getString(R.string.button_start));

        assertEquals("Toggle Button not found", true, actualTest);

        solo.clickOnToggleButton(solo.getString(R.string.button_start));
    }

    public void Read_updated_location() {
        TestLocationProvider.sendLocation(solo, new double[] { 43.723180, 44.697660 },
                                                new double[] { 10.396677, 10.631054 });
    }

    // ...
}
```

A abordagem apresentada no exemplo motivacional é idêntica à investigada no estudo experimental descrito no Capítulo 4. Como relatado, a estratégia proposta foi avaliada e definida como válida para o TBM em aplicações Android. Entretanto, dificuldades e desafios inerentes ao contexto de mobilidade persistem perante a adoção de TBM e ESG. Além disso, abordagens e ferramentas de apoio específicas promovem contribuições ao teste de aplicações móveis. Dentre tais tópicos de interesse, destacam-se os apresentados a seguir.

Elaboração de modelos de teste válidos e padronizados. Os modelos de teste elaborados na etapa de modelagem do processo de TBM devem ser válidos e padronizados, independente da técnica de modelagem adotada. Particularmente para a técnica ESG, a definição de nós especiais, “[”e “]”, e de rótulos concisos e padronizados, bem como a validação do grafo quanto à alcançabilidade de nós são funcionalidades relevantes para uma abordagem e ferramenta de apoio ao TBM.

Redução do esforço demandado para a concretização. A concretização de casos de teste demanda esforços de tempo e de conhecimento acerca da aplicação em teste, bem como da própria plataforma de teste utilizada. Modelos de teste extensos e/ou complexos resultam em maior quantidade de CESs geradas. Mesmo que os

casos de teste abstratos sejam facilmente concretizados, há de se considerar que a concretização requisita tanto quanto, senão mais, esforços manuais quando comparada à, também manual, etapa de modelagem. Abordagens e estratégias que automatizam e/ou reduzem o esforço demandado para a concretização são necessárias para o TBM em aplicações móveis.

Teste de distintas características de aplicações móveis. O contexto de mobilidade promove particularidades ao teste de software (WASSERMAN, 2010; MUCCINI et al., 2012; GAO et al., 2014). A considerar que o modelo de teste elaborado com TBM é, por padrão, uma representação formal dos eventos do SUT, distintas características do contexto de aplicações móveis podem ser acrescidas ao modelo original para apoiar o teste sob diferentes perspectivas. Dessa forma, para um mesmo modelo de teste, eventos adicionais oportunizam a verificação não somente dos requisitos funcionais mapeados pelo testador, bem como o uso da aplicação móvel em diferentes cenários. Por exemplo, (i) falhas e/ou mudanças de redes e conectividade, (ii) simulações de eventos de sensores como GPS e acelerômetro e (iii) manipulação de outras aplicações e/ou funcionalidades do dispositivo móvel durante a execução dos testes.

5.3. ABORDAGEM PROPOSTA

O objetivo da abordagem é reusar os modelos de teste elaborados com ESG. Pretende-se, por meio do reúso de modelos ESG, otimizar as atividades e passos estabelecidos pela abordagem padrão de TBM, enfatizando a etapa de concretização, bem como tornar o teste de aplicações móveis mais adequado às distintas características e possíveis cenários de aplicações e dispositivos móveis.

Para fundamentar a abordagem específica de TBM, duas estratégias foram definidas:

Method Template. Nesta estratégia, os eventos mapeados no modelo de teste são relacionados a um marcador especial, definido como estereótipo. Portanto, metadados são atribuídos aos nós do grafo, tornando-os eventos manipuláveis por

uma ferramenta de apoio para gerar casos de teste completa ou parcialmente concretizados.

Edge Template. Nesta estratégia, os estereótipos são atribuídos às arestas de eventos. Posteriormente, uma ferramenta de apoio acrescenta novos eventos ao modelo de teste por meio da manipulação dos metadados anteriormente definidos nas arestas.

Define-se, como estereótipo, um ou mais metadados atribuídos ao modelo de teste elaborado, complementando-o com informações e valores que podem ser utilizados em diferentes etapas do processo de TBM. Para a abordagem proposta, tais estereótipos são manipulados nos passos de geração e concretização de casos de teste. Portanto, o modelo ESG, até então um artefato formal, porém estático de modelagem para uma abordagem padrão de TBM e ESG, introduz novos elementos à representação visual e conceitual de modelos de teste.

Ambas as estratégias incorporam um conceito comum definido como estereótipo. Os estereótipos são atribuídos aos nós (eventos) para a estratégia *Method Template* e às arestas (transição entre eventos) para a estratégia *Edge Template*. A lista de estereótipos projetada para as estratégias *Method Template* e *Edge Template* é apresentada no Apêndice D. É importante destacar que, além dos metadados atribuídos aos eventos do modelo ESG na forma de estereótipos, a ferramenta que apoiará a automação da abordagem proposta pode ser projetada para manipular outros tipos de metadados, inclusive da aplicação móvel em teste. Dessa forma, o testador utiliza metadados do SUT na elaboração de artefatos de modelagem. Particularmente, os metadados extraídos da aplicação em teste como, por exemplo, valores de rótulos, nomes de telas e configurações gerais podem ser associados aos estereótipos atribuídos.

De maneira mais técnica, cada estereótipo disponibilizado pela abordagem específica de TBM está relacionado a um determinado *snippet*, um fragmento de código fonte desenvolvido com as funcionalidades e os recursos providos pela API da ferramenta e/ou plataforma de teste utilizada.

Junto à abordagem, delimita-se um conjunto de estereótipos para a estratégia *Method Template* como, por exemplo, (i) pressionar botões físicos ou virtuais, (ii) selecionar itens de menu, (iii) preencher campos de entrada de dados,

(iv) manipular componentes de interface e (v) capturar imagens de telas. Na Figura 20 é ilustrado um protótipo de um modelo ESG reusado a partir da estratégia *Method Template* em um modelo original (Figura 19). Conceitualmente, os nós destacados na cor verde representam eventos estereotipados. Para os eventos "*Press Menu*" e "*Press Back*", os estereótipos de "pressionar botão Voltar" e de "pressionar botões físicos ou virtuais" foram utilizados, respectivamente. Para eventos "*Select Map*" e "*Select Satellite*", foi atribuído um estereótipo de "selecionar itens de menu".

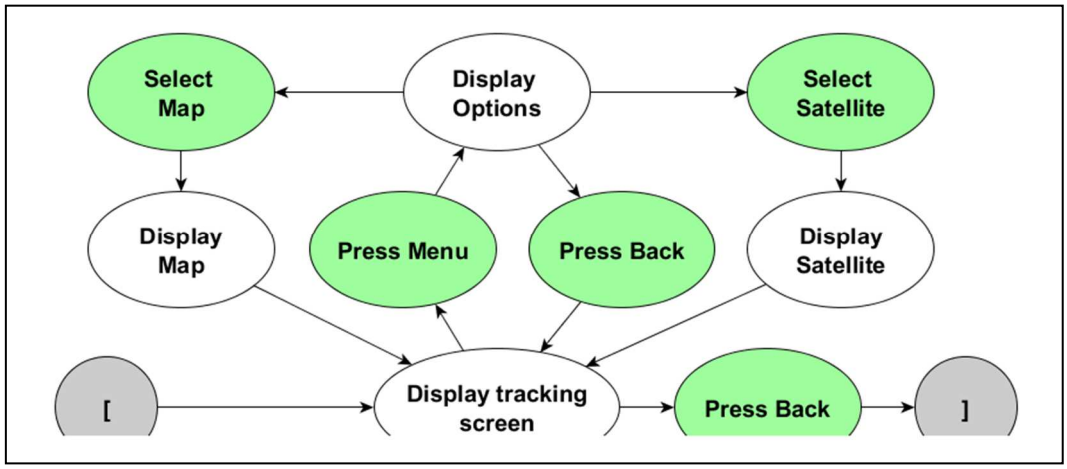

Figura 20 - Modelo ESG para *RouteTracker* com *Method Template*
Fonte: Elaborada pelo autor

Posteriormente, na etapa de geração de casos de teste, uma ferramenta de apoio pode manipular o modelo ESG e interpretar os eventos estereotipados. Dessa forma, os eventos em destaque não demandam esforço para a concretização manual, tendo em vista que a geração dos casos de teste utiliza os *snippets* previamente desenvolvidos. Caso, ainda assim, seja necessária alguma intervenção do testador, o esforço total será inferior ao de uma concretização totalmente manual. Para exemplificar, os *snippets* para os estereótipos de "pressionar botão Voltar" e de "pressionar botões físicos ou virtuais" foram implementados na plataforma *Robotium* e são apresentados a seguir.

```
// Snippet para estereótipo de "pressionar Botão Voltar"
solo.goBack();
```

```
// Snippet para estereótipo de "pressionar botões físicos ou virtuais"
solo.clickOnButton(solo.getString( {{idbutton}} ));
```

Para a estratégia *Edge Template*, também se define um conjunto de estereótipos. As distintas características, acrescidas ao modelo de teste quando se adota a estratégia *Edge Template*, são categorizadas em (i) eventos específicos de dispositivos, (ii) interação não previsível de usuários, (iii) eventos de telefonia para GSM/SMS e (iv) eventos de sensores e componentes de hardware.

Eventos específicos de dispositivos. O dispositivo móvel e sistema operacional executam processos em *background* que implicam na geração de eventos nativos. A plataforma Android dispõe de um gerenciador de bateria que verifica se o nível está muito reduzido e pode comprometer o correto funcionamento das aplicações móveis. Outro recurso é o controle nativo acerca da perda de conexão *wireless* que pode resultar no acionamento automático de uma rede 3G.

Interação não previsível de usuários. A interação de um usuário real com a aplicação móvel é conduzida de uma maneira diferente do que normalmente se avalia nos testes. Espera-se que as ações de um testador humano sejam consideradas como, por exemplo, (i) bloquear e desbloquear o dispositivo móvel, (ii) ir para a tela principal e retornar à aplicação móvel, (iii) permanecer inativo e sem interações com o dispositivo móvel por um longo período de tempo e (iv) avançar para uma determinada tela e retornar à anterior por algumas vezes. Portanto, o teste deve explorar diversas ações e comportamentos inesperados que um usuário pode realizar e que são desconsideradas pelos testadores.

Eventos de telefonia para GSM/SMS. Distintos eventos de telefonia como, por exemplo, (i) chamadas telefônicas e (ii) mensagens de texto podem interferir no correto funcionamento das aplicações móveis. A prioridade de processos no sistema operacional do dispositivo móvel é suscetível a mudanças e, portanto, a aplicação móvel em teste pode ser definida como de importância secundária. Por esse motivo, é necessário investigar e verificar os requisitos funcionais da aplicação móvel quando tal situação ocorre.

Eventos de sensores e componentes de hardware. Progressivamente, os dispositivos móveis incorporam novas funcionalidades e recursos encapsulados em sensores e componentes de hardware. Na atualidade, um dispositivo móvel é dotado por GPS, acelerômetro, giroscópio, medidor de temperatura, barômetro e outros sensores. Os componentes de hardware fornecem informações do ambiente físico e

aplicações móveis específicas manipulam determinados sensores. Dessa forma, o teste no contexto de mobilidade deve considerar a execução da aplicação móvel quando sensores e componentes de hardware estão desabilitados ou indisponíveis.

Nas Figuras 21 e 22 são ilustrados, respectivamente, protótipos de atribuições de estereótipos em arestas para a estratégia *Edge Template* e, após manipulação do modelo de teste por uma ferramenta de apoio, eventos são acrescidos ao modelo de teste conforme os estereótipos definidos. Conceitualmente, as arestas destacadas na cor azul na Figura 21 representam as transições de eventos estereotipadas, enquanto que os nós na cor amarela na Figura 22 representam eventos gerados pela estratégia *Edge Template*.

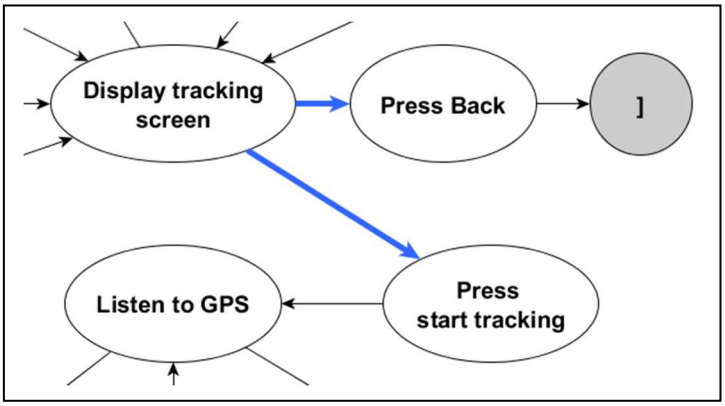

Figura 21 - Modelo ESG para *RouteTracker* com *Edge Template*
Fonte: Elaborada pelo autor

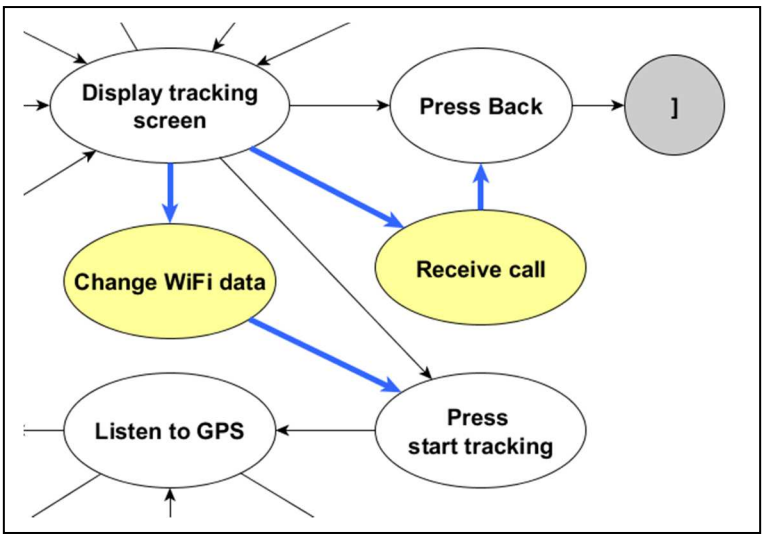

Figura 22 - Modelo ESG para *RouteTracker* com *Edge Template* (eventos)
Fonte: Elaborada pelo autor

A diversidade e a evolução dos requisitos funcionais das aplicações móveis, bem como da própria plataforma de desenvolvimento evidenciam que a abordagem deva ser flexível e escalável. Dessa forma, a abordagem e ferramenta de apoio podem ser estendidas para que novos estereótipos sejam incorporados.

Na Figura 23 é ilustrado um protótipo conceitual de uma possível versão do modelo ESG reusado a partir das estratégias *Method Template* e *Edge Template* quando utilizadas em um modelo ESG original (Figura 19).

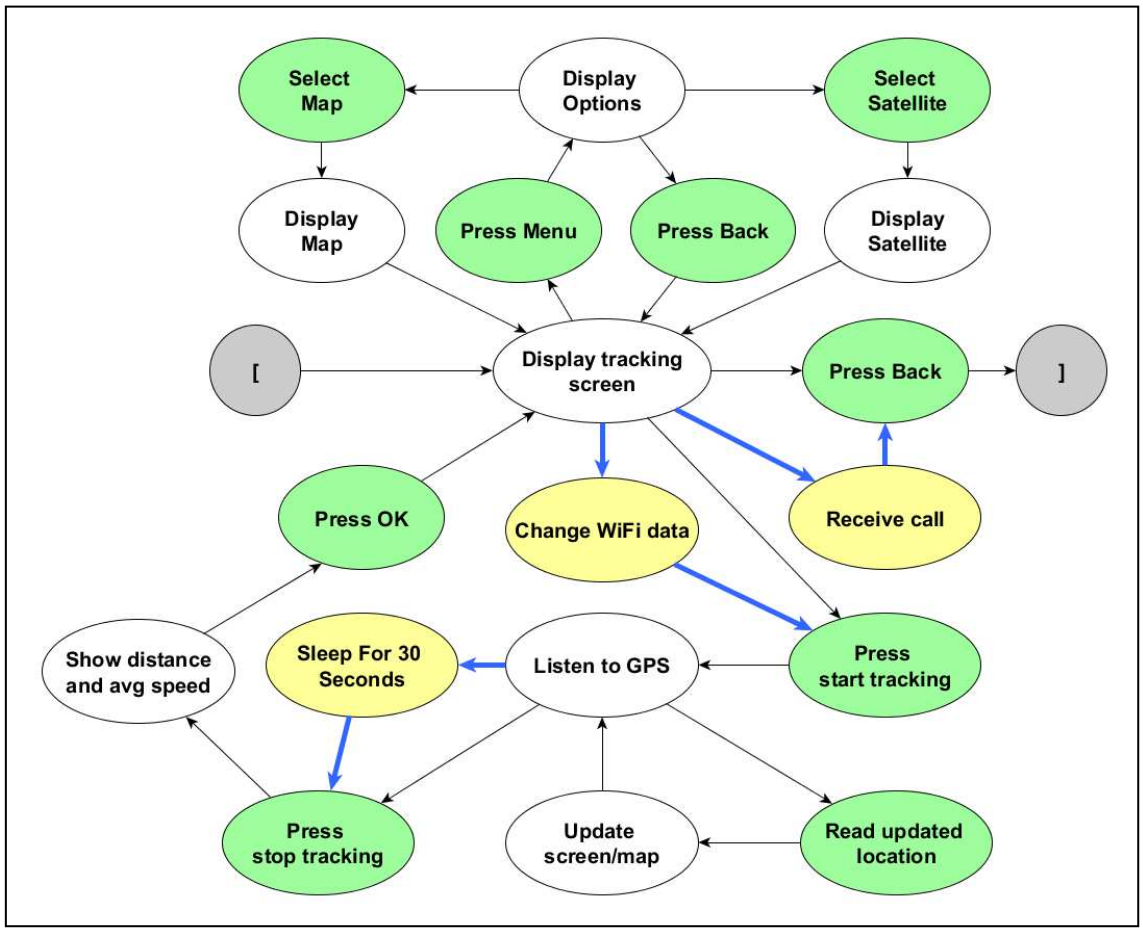

Figura 23 - Modelo ESG para *RouteTracker* com as estratégias propostas
Fonte: Elaborada pelo autor

Definição da abordagem de TBM para reúso de modelos.

Formalmente, a abordagem assume a existência de um modelo ESG definido como M = (V, E, Ξ, Γ), onde:

- *V* é o conjunto finito não vazio de vértices (ou nós) que representa os eventos;
- $E \subseteq V \times V$ é o conjunto finito de arestas que representa sequências válidas de transição entre eventos;

- $\Xi \subseteq V$ é o conjunto finito de vértices distintos com $\xi \in \Xi$ definido como “nós de entrada”, em que, para cada $\upsilon \in V \setminus \Xi$, há pelo menos uma sequência de vértices $\langle \xi , \upsilon_0 , ... , \upsilon_k \rangle$ de $\xi \in \Xi$ para $\upsilon_k = \upsilon$, para i = 0, ..., k - 1 e $(\xi , \upsilon_0) \in E$; e
- $\Gamma \subseteq V$ é o conjunto finito de vértices distintos com $\gamma \in \Gamma$ definido como “nós de saída”, em que, para cada $\upsilon \in V \setminus \Gamma$, há pelo menos uma sequência de vértices $\langle \upsilon_0 , ... , \upsilon_k , \gamma \rangle$ de $\upsilon_0 = \upsilon$ para $\gamma \in \Gamma$ com $(\upsilon_i , \upsilon_{i+1}) \in E$, para i = 0, ..., k - 1 e $(\upsilon_k, \gamma) \in E$.

Os nós de entrada e de saída são representados no modelo gráfico pela conexão com os nós especiais “[”e “]”. Para ambas as estratégias de reúso de modelos de teste, um novo modelo M_{reused} = (V, E, Ξ, Γ, Me_τ, Ed_τ, f_{mt}, f_{et}) é elaborado com o auxílio do testador, onde:

- V, E, Ξ e Γ são os mesmos conjuntos descritos para M;
- Me_τ é o conjunto finito de estereótipos para a estratégia *Method Template*;
- Ed_τ é o conjunto finito de estereótipos para a estratégia *Edge Template*;
- $f_{mt} : V \rightarrow p(Me_\tau)$ é uma função que define os nós anotados com estereótipo da estratégia *Method Template*, ou $\varnothing$ caso não tenha nenhum estereótipo associado; e
- $f_{et} : E \rightarrow p(Ed_\tau)$ é uma função que define as arestas anotadas com estereótipo da estratégia *Edge Template*, ou $\varnothing$ caso não tenha nenhum estereótipo associado.

5.4. MBTS4MA: FERRAMENTA DE APOIO

A ferramenta que apoia a automação da abordagem proposta foi implementada com a tecnologia Java, tendo como alvo aplicações móveis desenvolvidas na plataforma Android. Para a concretização de casos de teste gerados, assim como o desenvolvimento dos *snippets* propostos para as estratégias *Method Template* e *Edge Template*, a plataforma de teste *Robotium* foi escolhida.

Inicialmente, a ferramenta *Model-Based Test Suite For Mobile Applications* – (MBTS4MA) apoia o processo de TBM e a técnica de modelagem ESG como aplicado no estudo experimental relatado no Capítulo 4. Dessa forma, a ferramenta MBTS4MA fornece as funcionalidades necessárias para adotar uma abordagem padrão de TBM e ESG no contexto de aplicações móveis. A elaboração de modelos de teste é realizada por meio de uma interface gráfica que permite a criação e edição de modelos ESG.

As estratégias *Method Template* e *Edge Template* são disponibilizadas como recursos intuitivos no ambiente de MBTS4MA. Assim como realizado para a modelagem, a elaboração e reúso de modelos são acessíveis por menus e botões de ação. Uma arquitetura componentizada assegura que as estratégias descritas pela abordagem de reúso de modelos de teste possam evoluir, garantindo que a inclusão de novos estereótipos e *snippets* seja alcançada sem dificuldades e/ou impedimentos tecnológicos. Na Figura 24 é ilustrado um diagrama arquitetural da ferramenta de apoio MBTS4MA e os principais módulos são apresentados.

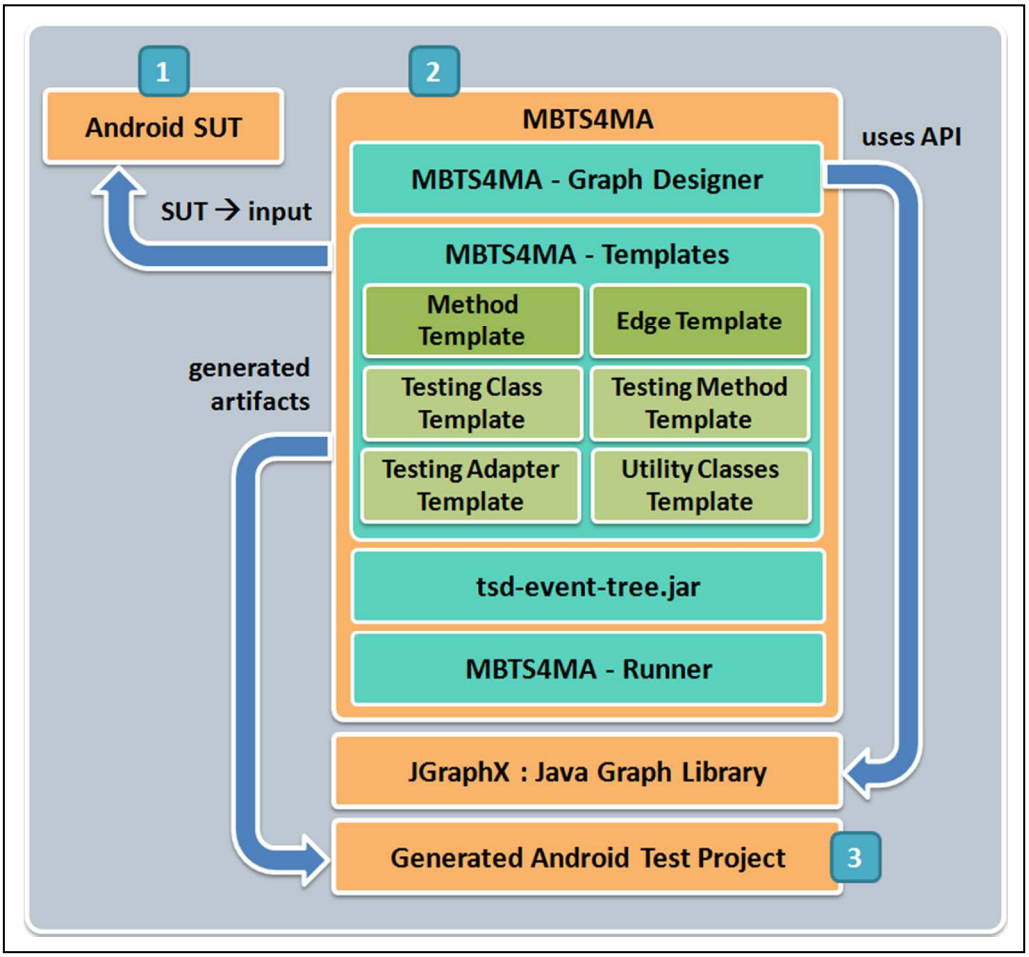

Figura 24 - Diagrama arquitetural da ferramenta de apoio MBTS4MA
Fonte: Elaborada pelo autor

Ao criar um novo projeto na ferramenta MBTS4MA, o testador pode relacioná-lo à aplicação móvel em teste (Item 1 na Figura 24). Dessa forma, o SUT se torna acessível pela ferramenta de apoio e metadados são extraídos para apoiar a elaboração de modelos de teste com informações recuperadas da aplicação móvel em teste. A ferramenta MBTS4MA (Item 2 na Figura 24) se baseia em uma arquitetura componentizada que integra distintas tecnologias e recursos de apoio à abordagem proposta.

MBTS4MA - Graph Designer. Componente que estende a API de *JGraphX*[5] para prover uma interface de modelagem baseada em grafos. Particularmente, a interface desenvolvida na ferramenta MBTS4MA oportuniza a elaboração de modelos ESG e a integração destes com a aplicação móvel em teste. Assim, metadados são extraídos do SUT e valores de rótulos, nomes de telas e configurações gerais podem ser utilizados como parâmetros para os estereótipos disponibilizados pelas estratégias *Method Template* e *Edge Template*;

A interface de modelagem permite que eventos representados por nós do grafo sejam criados, renomeados, dimensionados, reposicionados e relacionados a outros nós. Além disso, estereótipos podem ser facilmente adicionados a nós de eventos por meio de um menu de seleção rápida. Na Figura 25 é ilustrada a interface principal da ferramenta MBTS4MA que exibe um modelo ESG elaborado a partir dos eventos da aplicação *RouteTracker*.

MBTS4MA - Templates. Conjunto de componentes, classes e métodos, utilizados como *templates* na abordagem. Pretende-se, a partir dos *templates*, otimizar e viabilizar a geração de projetos de teste após a elaboração e/ou reúso de modelos de teste. O conjunto de componentes de *templates* é apresentado a seguir.

- ***Method Template***. Componente responsável pela definição de estereótipos e *snippets* relacionados à estratégia *Method Template*. A inclusão de novos estereótipos e *snippets* a este componente é dinamicamente identificada pela ferramenta de apoio e, de maneira automatizada, tais recursos são disponibilizados ao testador. Na Figura 26 são ilustrados (i) o procedimento de atribuição do estereótipo "*Press Back*" ao respectivo evento e (ii) a representação estereotipada no modelo ESG para a estratégia *Method Template*.

[5] Biblioteca Java que disponibiliza objetos e algoritmos para elaboração e manipulação visual de diferentes estruturas de dados baseadas em grafos. A API pode ser consultada em https://github.com/jgraph/jgraphx.

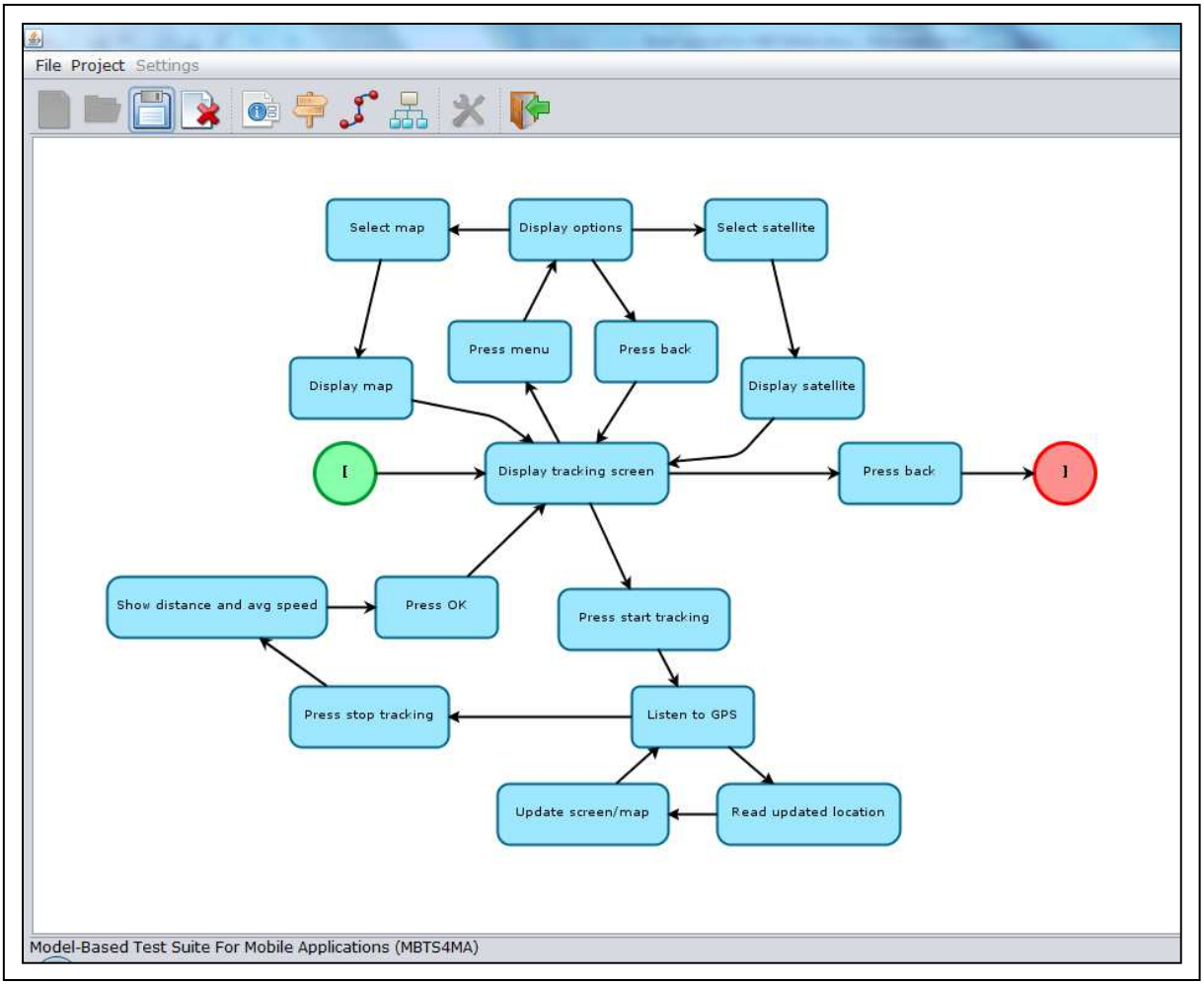

Figura 25 - Tela principal da ferramenta de apoio MBTS4MA
Fonte: Elaborada pelo autor

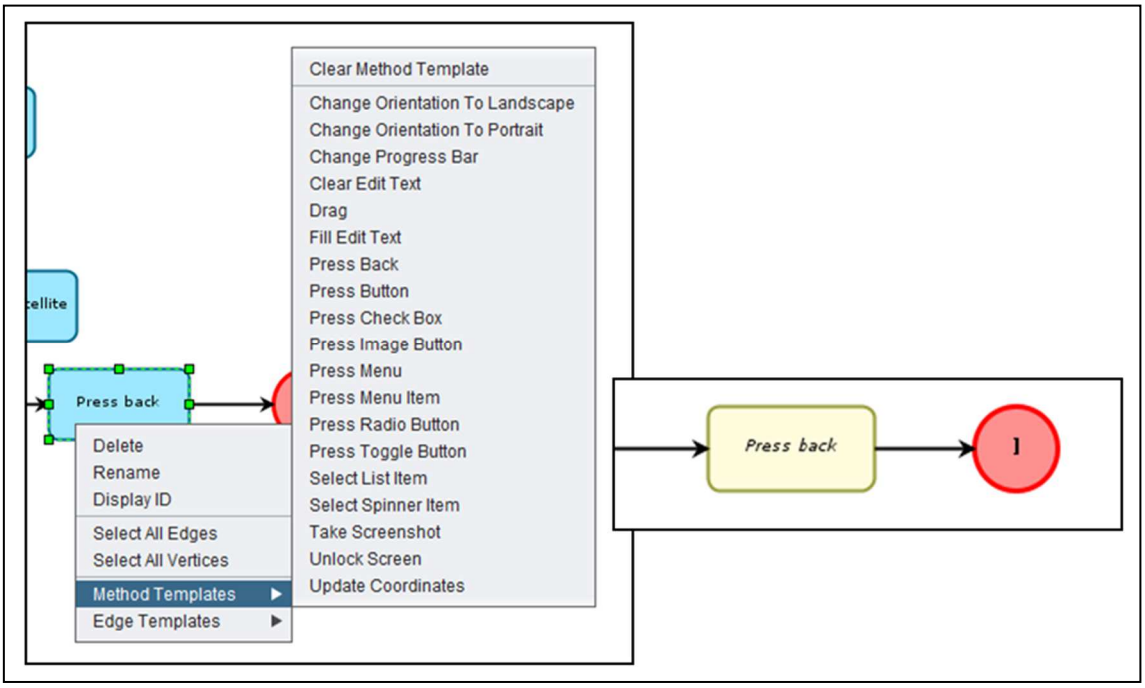

Figura 26 - Atribuição de estereótipo com *Method Template*
Fonte: Elaborada pelo autor

- ***Edge Template***. Componente responsável pela definição de estereótipos e *snippets* relacionados à estratégia *Edge Template*. A inclusão de novos estereótipos e *snippets* a este componente é dinamicamente identificada pela ferramenta de apoio e, de maneira automatizada, tais recursos são disponibilizados ao testador.

Na Figura 27 são ilustrados (i) o procedimento de atribuição de estereótipo "*Change Wi Fi Data*" para a geração de evento e (ii) a representação estereotipada no modelo ESG. O estereótipo selecionado que objetiva desabilitar o acesso a redes de dados sem fio. Em seguida, o testador utiliza um botão de ação na ferramenta MBTS4MA para manipular o modelo ESG, verificar os estereótipos de reúso e, efetivamente, gerar os eventos que acrescentam distintas características ao modelo de teste.

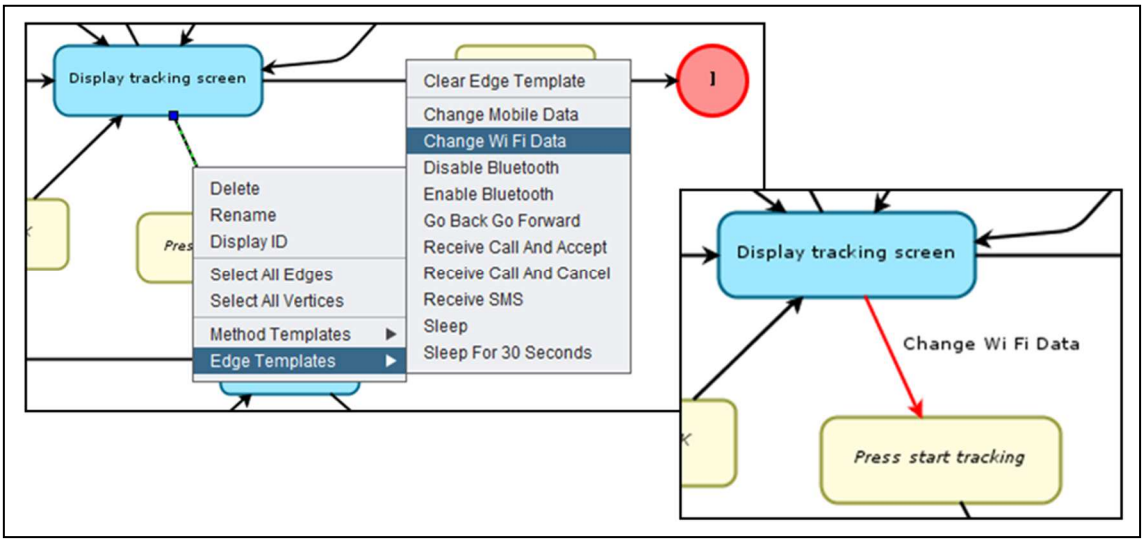

Figura 27 - Atribuição de estereótipo com *Edge Template*
Fonte: Elaborada pelo autor

- ***Testing Class Template***. Classe de *template*, baseada em JUnit, que representa o conjunto de teste gerado a partir das CESs obtidas do modelo ESG. Dessa forma, todas as CESs geradas de um mesmo modelo de teste serão elencadas em uma única implementação da classe de *template*. A versão desenvolvida da classe de *template* incorpora as funcionalidades providas pela plataforma *Robotium*;

- ***Testing Method Template***. Fragmento de classe de *template* que representa a estrutura e a sintaxe padrões a ser utilizada nos métodos de teste. Dessa forma, uma replicação deste fragmento de classe será realizada para cada CES obtida do

modelo ESG. Os métodos de teste gerados a partir do *template* serão inseridos na classe de *template* (item anterior, *Testing Class Template*);

- ***Testing Adapter Template***. Classe de *template* para os adaptadores dos casos de teste abstratos. Para isso, os métodos de adaptadores serão anotados por meio do componente *MBTS4MA - Runner*; e
- ***Utility Classes Template***. Classes utilitárias que apoiam a abordagem de TBM e as estratégias de reúso de modelos de teste.

tsd-event-tree.jar. Biblioteca de classes e artefatos em Java, inicialmente desenvolvidos por Belli et al. (2013), que compõem uma estrutura de dados fundamentada em grafos direcionados. Uma implementação adicional foi realizada para incorporar um algoritmo baseado em *Spanning Tree* (ENDO, 2013) e, dessa forma, apoiar a geração de CESs. O critério de teste todos-arcos foi adotado para permitir que cada aresta do grafo seja, ao menos uma vez, exercitada no teste (MALDONADO et al., 2004);

MBTS4MA - Runner. Classes e anotações em Java (em Inglês, *Java Annotations*) que colaboram com uma estrutura abstrata para o relacionamento entre os casos de teste gerados, ainda não executáveis no SUT, e os métodos de teste concretizados. Uma vez concretizados, os métodos de teste são anotados e podem ser executados na aplicação em teste. Portanto, tal componente está diretamente relacionado ao *Testing Adapter Template*. Nos Apêndices E e F são apresentadas (i) a classe `RouteTrackerTest`, responsável pela execução dos testes gerados com base nas CESs, e (ii) a classe `RouteTrackerAdapter`, responsável pela implementação dos adaptadores de teste.

Como resultados do uso da ferramenta de apoio, projetos de teste são gerados (Item 3 na Figura 24) como artefatos de saída da abordagem padrão de TBM ou da abordagem de reúso de modelos de teste. Em seguida, os projetos de teste resultantes da ferramenta MBTS4MA podem ser executados em AVDs e/ou dispositivos reais.

Diferentemente de outras ferramentas de apoio ao TBM que contemplam apenas a etapa de modelagem, a ferramenta MBTS4MA mantém uma relação direta com a aplicação móvel a ser testada. Dessa forma, a leitura e manipulação de

metadados da aplicação móvel em teste é factível e pode ser realizada pelo testador a qualquer momento.

A extração e uso de características do SUT simplifica a elaboração do modelo ESG, pois se torna possível relacionar eventos do modelo de teste a constantes e rótulos utilizados nas telas e demais componentes da aplicação em teste. Por exemplo, um evento de "*Press Save*" encapsula um metadado relacionado à constante que representa o rótulo "*Save*". A mesma funcionalidade é aplicável a diferentes componentes visuais como botões, caixas de seleção, itens de menu, janelas de diálogo, mensagens de notificação, entre outros. Os metadados atribuídos a eventos no modelo ESG são processados nas etapas de geração e concretização de casos de teste e resultam em casos de teste completa ou parcialmente concretizados. Na Figura 28 é ilustrada uma nova versão do modelo ESG reusado com eventos estereotipados e gerados pelas estratégias *Method Template* e *Edge Template*.

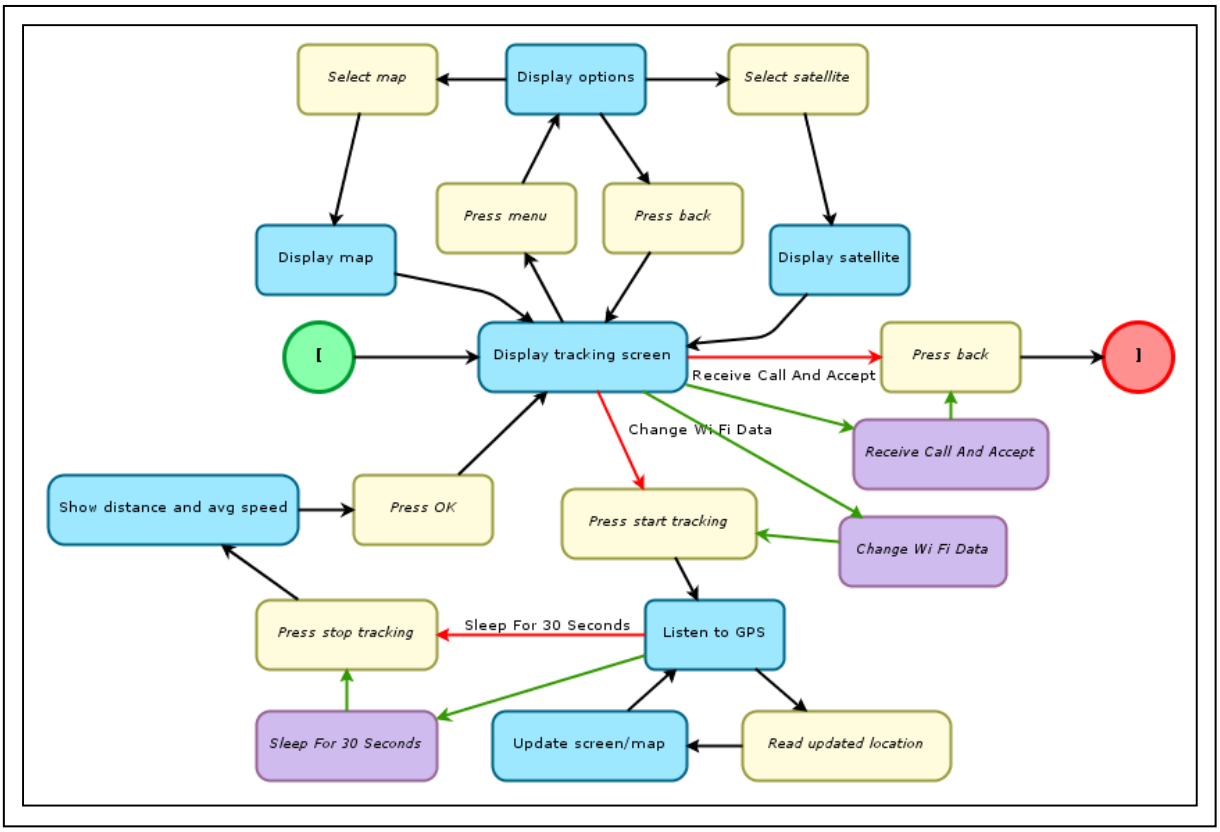

Figura 28 - Modelo ESG para *RouteTracker* elaborado na ferramenta MBTS4MA
Fonte: Elaborada pelo autor

A ferramenta de apoio à abordagem de TBM para o reúso de modelo de teste, foi compartilhada como um projeto *open source* em:

https://github.com/guilhermefarto/MBTS4MA

5.5. CONSIDERAÇÕES FINAIS

Neste capítulo, foi apresentada uma abordagem específica de TBM para o reúso de modelos de teste no contexto de aplicações móveis. Um exemplo motivacional foi descrito para revisitar a adoção de uma abordagem padrão de TBM e ESG e, posteriormente, tópicos de interesse no teste de aplicações móveis foram relatados. A abordagem proposta se fundamenta em duas principais perspectivas de avanços no TBM em aplicações móveis: (i) a redução do esforço demandado para a concretização e (ii) o teste de distintas características de aplicações móveis. Para isso, duas estratégias foram definidas: *Method Template* e *Edge Template*, respectivamente. As características foram categorizadas em (i) eventos específicos de dispositivos, (ii) interação não previsível de usuários, (iii) eventos de telefonia para GSM/SMS e (iv) eventos de sensores e componentes de hardware.

A abordagem proposta é apoiada pela ferramenta MBTS4MA, utilizada para a elaboração de modelos de teste em ESG, bem como na adoção das estratégias *Method Template* e *Edge Template*. Portanto, após a modelagem de requisitos funcionais, os recursos disponibilizados na ferramenta MBTS4MA possibilitam a definição de estereótipos e o acréscimo de distintas características aos modelos de teste. Dinamicamente, a ferramenta de apoio extrai metadados do SUT que podem ser incorporados ao modelo ESG para reduzir o esforço manual demandado na concretização de casos de teste.

No próximo capítulo, um estudo experimental é conduzido para avaliar a abordagem e ferramenta propostas em um ambiente industrial. A avaliação objetiva investigar os esforços demandados na modelagem e concretização, bem como verificar a capacidade de detecção de defeitos a partir do reúso de modelos ESG.

6. AVALIAÇÃO DA ABORDAGEM PROPOSTA

6.1. CONSIDERAÇÕES INICIAIS

Abordagens, metodologias, ferramentas e estratégias de apoio ao teste automatizado têm sido propostas para considerar particularidades e desafios apresentados pelo contexto de aplicações móveis. Como identificado a partir dos resultados do EMS apresentado no Capítulo 3, as contribuições dadas por pesquisas relevantes ainda se mantêm em cenários exploratórios e experimentais. Dessa forma, apesar dos avanços providos por TBM e, particularmente, os relacionados às aplicações móveis, poucos estudos transpõem avaliações empíricas acadêmica e cientificamente instituídas (JÄÄSKELÄINEN et al., 2009b; RIDENE; BARBIER, 2011; JANICKI et al., 2012; LI et al., 2014). Entretanto, é importante investigar e discutir os resultados e desafios originados pela adoção de abordagens e de ferramentas propostas na indústria e em ambientes reais (UTTING; LEGEARD, 2006; ULRICH, 2007; PELESKA, 2013; BINDER et al., 2014).

Neste capítulo, é apresentado um estudo experimental conduzido em ambiente industrial para avaliar a abordagem e a ferramenta MBTS4MA propostas no Capítulo 5. Na Seção 6.2, a configuração do estudo é descrita. Na Seção 6.3, é apresentada a análise dos resultados obtidos. Nas Seções 6.4 e 6.5, são evidenciadas as limitações e as lições aprendidas, respectivamente. Na Seção 6.6, a discussão dos resultados é apresentada.

6.2. CONFIGURAÇÃO DO ESTUDO

Pretende-se, por meio de um estudo experimental conduzido na indústria, avaliar a abordagem de TBM e a ferramenta de apoio propostas para o reúso de modelos de teste na verificação de aplicações móveis. Este estudo objetiva responder às seguintes questões de pesquisa:

- ***Q1***. “Quais os esforços demandados na modelagem e concretização de casos de teste a partir das estratégias *Method Template* e *Edge Template*?”; e
- ***Q2***. “Em relação às características e fundamentos propostos, a abordagem demonstrou ser promissora quanto à capacidade de detectar defeitos?”.

A avaliação experimental foi conduzida em colaboração a uma empresa multinacional de TI. Seis participantes da empresa foram selecionados, no qual três atuam como desenvolvedores de aplicações móveis com expertise nas tecnologias Java e Google Android. Outros três participantes exercem a função de analistas de teste e detêm conhecimentos acerca dos conceitos de negócio das aplicações móveis desenvolvidas. A empresa proveu acesso às aplicações móveis e a seus respectivos códigos fontes, bem como a documentos de especificação. Além disso, forneceu o suporte tecnológico necessário para a configuração de ambientes utilizados no estudo. Os dados sumarizados e discutidos foram produzidos pelos participantes na avaliação experimental e coletados pelos pesquisadores.

Formulário de entrevista inicial. Inicialmente, um formulário de entrevista foi aplicado para auxiliar na identificação dos perfis dos participantes e dos processos atuais de teste. O modelo de questionário é exemplificado no Apêndice G. Os objetivos almejados ao aplicar os questionamentos são descritos a seguir.

(i) Identificar os conhecimentos e as experiências profissionais dos participantes;

(ii) Compreender os processos e as atividades de teste adotadas pela empresa; e

(iii) Relatar o cenário vigente e as perspectivas esperadas quanto aos testes de aplicações móveis.

Sessão de treinamento. Posteriormente à coleta de dados do formulário de entrevista inicial, uma sessão de treinamento foi realizada para apresentar e conceituar (i) os fundamentos de TBM e ESG e (ii) a abordagem e ferramenta de apoio MBTS4MA. As lições identificadas na Seção 4.5 foram apresentadas para acentuar as orientações de uma correta e válida modelagem. Por fim, uma breve introdução à plataforma de teste *Robotium* também foi realizada. Para exemplificar o

processo de TBM em sua totalidade, os artefatos elaborados na avaliação experimental relatada no Capítulo 4 foram apresentados como um exemplo prático e motivacional. A sessão de treinamento consumiu 60 minutos.

Estudo de caso: Aplicações industriais. Posteriormente à sessão de treinamento, os seis participantes da empresa foram divididos em três duplas, cada qual composta por um desenvolvedor e um analista de teste. Dessa forma, buscou-se um equilíbrio entre conhecimentos técnicos e de negócio. Cada dupla selecionou uma aplicação móvel desenvolvida em Android com aspectos que a classifique como de alta relevância e criticidade para a empresa. Além disso, os pesquisadores recomendaram a seleção de aplicações efetivamente utilizadas pelos clientes.

As versões das aplicações móveis foram obtidas de uma base histórica de compilação de seis meses anteriores em relação ao período em que a avaliação experimental foi realizada. A disponibilidade de uma versão consolidada é respaldada pelo interesse da empresa em testar aplicações móveis já verificadas pela equipe de qualidade e disponibilizadas em ambiente de produção.

Na Tabela 4 são apresentadas as aplicações móveis selecionadas e suas funcionalidades e estatísticas. Os nomes dos projetos foram omitidos para evitar a divulgação de informações sensíveis da empresa.

Tabela 4 - Aplicações móveis investigadas na avaliação experimental

App ID	**Funcionalidades**						**Estatísticas**		
	Acelerômetro	**Áudio**	***Bluetooth***	**Câmera**	**Geolocalização**	**Mapa *Offline***	**LoC**	***Activities***	**Média de Complexidade Ciclomática**
MobApp1		✓		✓	✓	✓	46.602	133	2,14
MobApp2			✓				24.032	44	2,43
MobApp3		✓					143.970	123	2,10

Uso de Robotium. A plataforma de teste *Robotium* foi novamente adotada na etapa de concretização. Como mencionado, os *snippets* expostos pela ferramenta MBTS4MA também são implementados com *Robotium*. Dessa forma, a concretização de casos de teste gerados pela ferramenta pode ser alcançada manualmente ou, parcialmente, de maneira automatizada por meio de *snippets*. Para este estudo experimental não foram avaliados tópicos como (i) a possibilidade de inclusão e/ou adaptação dos *snippets* e (ii) o esforço demandado para propor e implementar novos estereótipos e *snippets*.

Dispositivos móveis utilizados nos testes. Os projetos de teste gerados no estudo experimental foram executados em três dispositivos móveis. Para isso, um AVD e dois dispositivos móveis reais foram utilizados. Apesar de ser um conjunto reduzido, a execução de testes em dispositivos móveis com distintas especificações de hardware e software promove a característica de multiconfigurações de dispositivos móveis e viabiliza a diversidade de telas:

- ***(D1) Emulador (AVD)***. Versão 2.3.3 ou API 10 da plataforma Android, uma tela de 3.2” com qualidade média (mdpi), 512 MB de memória RAM e 50 MB de cartão de memória.
- ***(D2) Dispositivo real 1***. Versão 3.2 ou API 13 da plataforma Android, uma tela de 7” com qualidade alta (hdpi), 1 GB de memória RAM, 1.2GHz DualCore e 2 GB de cartão de memória.
- ***(D3) Dispositivo real 2***. Versão 4.0.3 ou API 15 da plataforma Android, uma tela de 10.1” com média alta (mdpi), 1 GB de memória RAM, 1GHz DualCore e 2 GB de cartão de memória.

Os AVDs não dispõem de recursos necessários para manipulação de hardware *Bluetooth*. Portanto, os testes devem ser executados em dispositivos reais para as aplicações móveis que apresentam funcionalidades que se baseiam no envio e recebimento de arquivos e/ou dados por *Bluetooth*. Além disso, é necessário que outro dispositivo móvel, que também possua *Bluetooth*, esteja próximo para simular transações de dados por *Bluetooth*.

Formulário de entrevista final. Após as principais atividades da avaliação experimental, um novo formulário de entrevista foi aplicado para relatar e pontuar os aspectos desejados por profissionais da indústria no teste de aplicações móveis. O modelo de questionário é exemplificado no Apêndice H. Os objetivos pretendidos ao solicitar o preenchimento do formulário são descritos a seguir.

(i) Mensurar a complexidade avaliada pelos participantes acerca dos conceitos de TBM e ESG e, principalmente, da abordagem e ferramenta propostas para o reúso de modelos de teste;

(ii) Avaliar a criticidade e/ou relevância de tópicos sugeridos para o teste de aplicações móveis como, por exemplo, “facilidade na elaboração dos modelos de teste” e “rastreabilidade dos casos de teste”; e

(iii) Oportunizar a contribuição dos participantes a partir das experiências adquiridas quanto a possíveis melhorias (i) na abordagem de reúso de modelos de teste e (ii) na ferramenta de apoio MBTS4MA.

6.3. ANÁLISE DE RESULTADOS

Os resultados coletados com base nos formulários de entrevista e, principalmente, na própria avaliação experimental são relatados a seguir.

Perfis dos participantes e processo atual de teste. As respostas obtidas no preenchimento do formulário apresentado no Apêndice G auxiliaram na formulação dos perfis dos participantes e na compreensão do atual processo de teste de aplicações móveis realizado pela empresa. Os três participantes que atuam como desenvolvedores declaram possuir experiência nas tecnologias Java e Google Android e classificam seus níveis de conhecimento, unanimemente, como alto. Os três analistas de teste afirmam deter conhecimentos de nível baixo nas tecnologias Java e Google Android. Os fundamentos de teste de software são avaliados como de nível médio, para os desenvolvedores, e de nível alto, para os analistas de teste. Uniformemente, o processo de TBM é conceituado como baixo, pois nenhum dos participantes possuía conhecimentos ou experiências além da sessão de

treinamento realizada. Na Tabela 5 são apresentados os conhecimentos avaliados pelos participantes.

Tabela 5 - Perfis dos participantes na avaliação experimental

Atividade ou cargo exercido	Participante	Java	Google Android	Teste de Software	TBM
Desenvolvedor	**1**	Alto	Alto	Médio	Baixo
	2	Alto	Alto	Médio	Baixo
	3	Alto	Alto	Médio	Baixo
Analista de teste	**4**	Baixo	Baixo	Alto	Baixo
	5	Baixo	Baixo	Alto	Baixo
	6	Baixo	Baixo	Alto	Baixo

Por meio do formulário de entrevista, também se identificou que os desenvolvedores realizam testes manuais durante o desenvolvimento e/ou manutenção das aplicações móveis. Posteriormente, um novo teste manual é realizado pelos analistas de teste e os passos executados compõem um documento de evidências de teste. Entretanto, o artefato de documentação gerado a partir dos testes não incorpora detalhes que o torne rastreável, repetível e/ou escalável, conforme reportado pelos participantes. Além disso, não são armazenadas informações acerca das configurações e demais detalhes dos dispositivos móveis utilizados em testes.

Os participantes declaram que não se estabelece um processo formal e padronizado para o teste de aplicações móveis. De acordo com as respostas apresentadas, os requisitos funcionais a serem testados são compreendidos com base em documentos de especificação e na própria experiência do analista de teste. Os participantes consideram que os testes manualmente realizados são trabalhosos e ineficientes. Além disso, consideram que muito tempo é consumido devido aos testes manuais e a constante necessidade de retestar as aplicações quando novas versões são disponibilizadas. Como resultados, os analistas de teste reportam a dificuldade e a sobrecarga de atividades durante um projeto.

Modelagem e concretização de testes. Os pesquisadores solicitaram a elaboração de um modelo de teste para alguma funcionalidade de cada aplicação selecionada pelas duplas. Para isso, a ferramenta MBTS4MA foi utilizada e, num primeiro momento, as estratégias *Method Template* e *Edge Template* foram desconsideradas. Dessa forma, a primeira versão do modelo ESG não se baseia na abordagem proposta, mas sim na definição padrão de TBM e da técnica de modelagem ESG, também viabilizada pela ferramenta MBTS4MA.

Na Tabela 6 são apresentados os dados sumarizados para as etapas de modelagem e concretização dos casos de teste para os modelos elaborados sem o apoio das estratégias *Method Template* e *Edge Template*.

Tabela 6 - Dados sumarizados para o modelo elaborado

App ID	Nós	Arestas	Tempo (em minutos)		LoC	*Activities* verificadas
			Modelagem	Concretização		
MobApp1	48	60	40	50	452	6
MobApp2	56	62	50	50	563	8
MobApp3	36	48	40	40	427	5

Posteriormente à modelagem inicial, uma segunda versão foi requisitada, porém com a adoção da abordagem de reúso de modelos de teste a partir da estratégia *Method Template*. Os modelos de teste elaborados na primeira modelagem foram, então, evoluídos.

Na Tabela 7 são apresentados os dados atualizados para os modelos reusados junto à estratégia *Method Template*. É importante ressaltar que as colunas “Modelagem”, “Concretização” e “LoC” exibem os valores modificados pelo uso de *Method Template* e, entre parênteses, são apresentados os valores positiva ou negativamente comparados ao modelo de teste da Tabela 6. Por exemplo, verifica-se que, para a etapa de modelagem da aplicação MobApp1, a adoção de *Method Template* estendeu o tempo total para 52 minutos. Dessa forma, em comparação ao modelo puramente definido com TBM e ESG, a estratégia *Method Template* foi completada em 12 minutos. Em contrapartida, o esforço direcionado à etapa de concretização resultou na redução do tempo, perfazendo 31 minutos, uma

otimização de 19 minutos quando comparada à mesma atividade anteriormente executada com TBM e ESG. Além disso, também se nota uma redução em LoC desenvolvidas, eventualmente motivada pela padronização dos *snippets* providos pela estratégia *Method Template*.

Tabela 7 - Dados sumarizados para o modelo reusado com *Method Template*

App ID [1]	**Nós**	**Nós Ester.** [2]	**Arestas**	**Tempo (em minutos)**		**LoC**	***Activities* verificadas**
				Modelagem	**Concretização**		
MobApp1	48	22	60	52 (↑ 12)	31 (↓ 19)	431 (↓ 21)	6
MobApp2	56	24	62	65 (↑ 15)	26 (↓ 24)	533 (↓ 30)	8
MobApp3	36	15	48	50 (↑ 10)	22 (↓ 18)	414 (↓ 13)	5

[1] Dados comparados aos valores apresentados na Tabela 6

[2] Nós estereotipados

Por fim, os pesquisadores solicitaram uma terceira e última versão que foi alcançada por uma nova evolução aos modelos anteriormente reusados com *Method Template*. Para isso, características relacionadas à estratégia *Edge Template* foram adicionadas.

Na Tabela 8 são apresentados os dados atualizados para os modelos reusados com o uso da estratégia *Edge Template*. Quando comparada à Tabela 7, os dados apresentados na Tabela 8 confirmam um aumento na quantidade de eventos (nós) e na quantidade de transições entre eventos (arestas), bem como um acréscimo no tempo necessário para a modelagem. A inclusão de novos eventos interfere na etapa de concretização e, como consequência, a quantidade de LoC também é aumentada. Entretanto, os tempos destinados à concretização permanecem próximos à duração apresentada na Tabela 7.

Ao final da etapa de modelagem, nove modelos ESG foram elaborados: três que representam apenas os requisitos funcionais das aplicações móveis em teste e dois grupos de três modelos de teste que, respectivamente, contemplam os requisitos funcionais e as estratégias *Method Template* e *Edge Template*.

Tabela 8 - Dados sumarizados para o modelo reusado com *Edge Template*

App ID [1]	Nós	Nós Ester. [2]	Arestas	Tempo (em minutos)		LoC	*Activities* verificadas
				Modelagem	Concretização		
MobApp1	56 (↑ 8)	30 (↑ 8)	76 (↑ 16)	62 (↑ 10)	30 (↓ 1)	473 (↑ 42)	6
MobApp2	63 (↑ 7)	31 (↑ 7)	76 (↑ 14)	74 (↑ 9)	29 (↑ 3)	568 (↑ 35)	8
MobApp3	41 (↑ 5)	20 (↑ 5)	58 (↑ 10)	56 (↑ 6)	20 (↓ 2)	438 (↑ 24)	5

[1] Dados comparados aos valores apresentados na Tabela 7

[2] Nós estereotipados

Execução dos testes. Para fins de investigação da abordagem proposta, apenas os modelos de teste que adotam as estratégias *Method Template* e *Edge Template* foram considerados. A execução dos testes foi conduzida em dois estágios para cada uma das aplicações móveis selecionadas. Os testes das aplicações móveis *MobApp1* e *MobApp3* foram realizados no AVD (*D1*) e nos dispositivos reais (*D2* e *D3*). Entretanto, os testes da aplicação móvel *MobApp2* foram executados apenas nos dispositivos reais (*D2* e *D3*). Tal particularidade é justificada devido à limitação do hardware *Bluetooth* em emuladores.

Estágio 1. A execução dos testes no Estágio 1 contempla os casos de teste gerados e concretizados a partir dos modelos que integram a estratégia *Method Template*. A execução dos casos de teste resultou na identificação de dois defeitos apresentados a seguir.

- ***(1) MobApp2 - Botão físico de "Voltar" habilitado***. As interfaces das aplicações móveis selecionadas incorporam um botão virtual "Cancelar" que substitui o botão físico "Voltar". Por esse motivo, o botão físico deve ser programaticamente desabilitado, impossibilitando que a ação nativa seja executada. Durante a execução dos testes, verificou-se que o bloqueio do botão físico não foi corretamente implementado em uma *Activity* específica e, por isso, campos de um

formulário específico permaneceram preenchidos após encerrar e retornar a mesma tela. O defeito foi revelado na execução de testes nos dispositivos *D2* e *D3*.

- ***(2) MobApp3 - Preenchimento incompleto e/ou incorreto de formulário de dados***. O não preenchimento completo e correto de um formulário de dados da aplicação móvel em teste gerou um defeito de `NullPointerException`. Campos obrigatórios não haviam sido informados na etapa de concretização. Em um ambiente real, o desconhecimento ou falta de atenção por parte de um usuário real também pode provocar a mesma situação. O defeito foi revelado na execução de testes nos dispositivos *D1*, *D2* e *D3*.

De acordo com os participantes na avaliação experimental, os defeitos identificados preliminarmente também haviam sido detectados em testes manuais em versões posteriores e manutenções foram realizadas para corrigi-los.

Estágio 2. A execução dos testes no Estágio 2 abrange os casos de teste gerados e concretizados de modelos que se baseiam na estratégia *Edge Template*, além de incorporar também a modelagem realizada com *Method Template*. A execução dos casos de teste motivou a identificação de três defeitos não relevados anteriormente.

- ***(3) MobApp1 - Recebimento de chamada telefônica quando exibindo uma Activity de mapa offline***. A aplicação móvel em teste incorpora uma API que possibilita o uso de mapas *offline* para exibição de dados geográficos. A verificação de tal funcionalidade quando uma chamada telefônica é recebida, atendida e cancelada resultou no encerramento indevido da aplicação móvel em teste. O defeito foi revelado na execução de testes nos dispositivos *D1*, *D2* e *D3*.
- ***(4) MobApp1 - Envio de coordenadas geográficas com precisão reduzida (poucas casas decimais)***. A classe utilitária `TestLocationProvider` é responsável por enviar posições geográficas de latitude e longitude com uma configuração de até seis casas decimais. A simulação de coordenadas geográficas evidenciou um defeito de `IndexOutOfBoundsException`, pois a aplicação manipula dados de longitude e latitude com dez casas decimais de precisão. O defeito foi revelado na execução de testes nos dispositivos *D1*, *D2* e *D3*.

- ***(5) MobApp2 - Envio de dados por Bluetooth quando o hardware Bluetooth não está disponível***. Aplicações móveis utilizam recursos de *Bluetooth* para o envio e recebimento de dados e arquivos. Assim que a conexão *Bluetooth* foi estabelecida, um novo evento desabilitou o respectivo componente de hardware. Dessa forma, um defeito de `NullPointerException` é lançado, pois os objetos de `InputStream` e `OutputStream` assumem valores nulos quando o *Bluetooth* não está disponível para uso. O defeito foi revelado na execução de testes nos dispositivos *D2* e *D3*.

Na tabela 9 são apresentados os dados sumarizados para a execução dos testes nos dispositivos móveis. Os Estágios 1 e 2 contemplam, respectivamente, os modelos de teste reusados com (i) a estratégia *Method Template* e com (ii) as estratégias *Method Template* e *Edge Template*. Dois defeitos foram detectados no Estágio 1, enquanto que outros três foram revelados no Estágio 2. Os tempos, em minutos, são definidos com base na média calculada para as execuções nos dispositivos móveis especificados na coluna “Dispositivos móveis”. O tempo total para as execuções dos testes foi de 52m21s, para o Estágio 1, e 67m59s, para o Estágio 2. Destaca-se que o Estágio 2 (*Method Template* e *Edge Template*) foi concluído com um tempo superior a 15m38s em relação ao Estágio 1 (somente *Method Template*). A plataforma *Robotium* demonstra, passo a passo, as interações realizadas nas aplicações móveis em teste. Assim sendo, os casos de teste são lentamente executados nos dispositivos móveis utilizados.

Tabela 9 - Dados sumarizados para a execução dos testes

App ID	Dispositivos móveis	Estágio 1 [1]		Estágio 2 [2]		Comparação de tempos
		Tempo	Defeitos	Tempo	Defeitos	
MobApp1	D1, D2 e D3	16m20s	0	21m17s	2	↑ 4m57s (30,3%)
MobApp2	D2 e D3	18m43s	1	25m33s	1	↑ 6m50s (36,5%)
MobApp3	D1, D2 e D3	17m18s	1	21m09s	0	↑ 3m51s (22,2%)
Total		52m21s	2	67m59s	3	↑ 15m38s (29,9%)

[1] *Method Template* e [2] *Method Template* e *Edge Template*

Complexidade avaliada pelos participantes. Na Figura 29 é ilustrado um gráfico que apresenta a complexidade avaliada pelos participantes. Os tópicos avaliados foram (i) TBM e ESG, (ii) abordagem proposta e (iii) ferramenta MBTS4MA. A classificação das respostas se limitou aos valores (i) nenhuma dificuldade e (ii) pouca dificuldade. Verifica-se que, para os conhecimentos de TBM e ESG, apenas um dos participantes classificou o entendimento como pouca dificuldade, enquanto que os demais não relatam dificuldades. Por unanimidade, os tópicos de abordagem proposta e ferramenta MBTS4MA foram destacados com a classificação de nenhuma dificuldade na avaliação experimental.

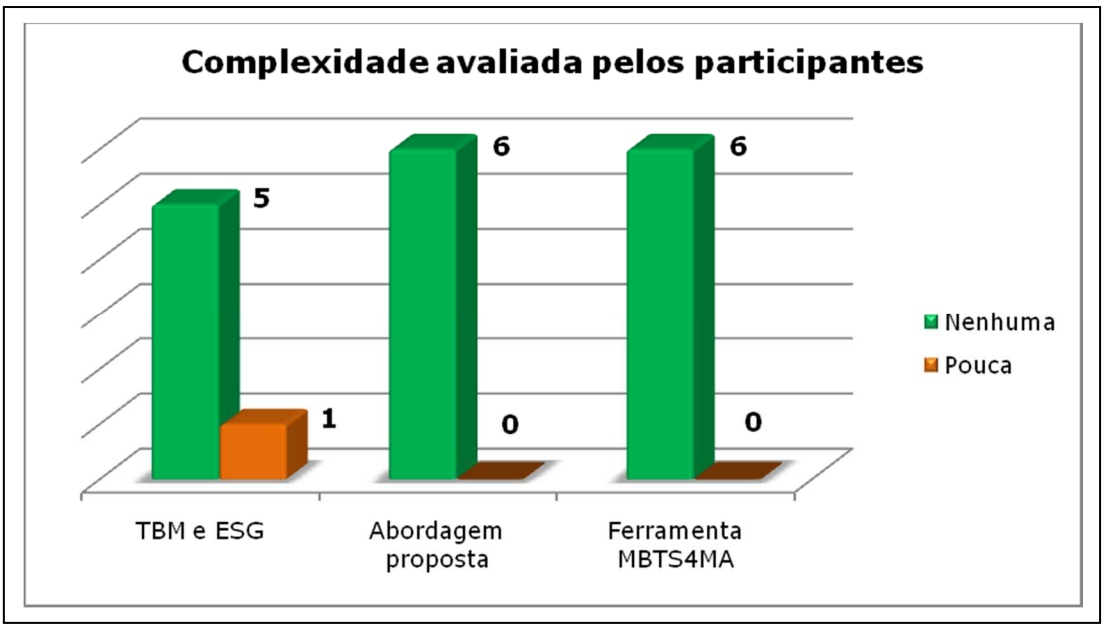

Figura 29 - Complexidade avaliada pelos participantes
Fonte: Elaborada pelo autor

Avaliação de criticidade e relevância apontadas. Na Figura 30 é ilustrado um gráfico que apresenta a classificação avaliada pelos participantes, por relevância, de tópicos que se relacionam ao teste de aplicações móveis. Destacam-se, como tópicos mais bem valorados, (i) repetibilidade (reexecução) dos testes e (ii) manutenibilidade dos casos de teste. Posteriormente, os tópicos (iii) facilidade na criação de casos de teste e (iv) rastreabilidade dos casos de teste são ressaltados. Em sequência, (v) uso de modelos e submodelos de teste e (vi) uso de informações da aplicação móvel em modelos de teste. Por último, (vii) relatório de informações pós-teste. Para facilitar a construção do gráfico, os tópicos sugeridos pelos pesquisadores foram enumerados de 1 a 7.

- ***Tópico 1***. Facilidade na criação de casos de teste;
- ***Tópico 2***. Uso de modelos e submodelos de teste;
- ***Tópico 3***. Uso de informações da aplicação móvel em modelos de teste;
- ***Tópico 4***. Repetibilidade (reexecução) dos testes;
- ***Tópico 5***. Rastreabilidade dos casos de teste;
- ***Tópico 6***. Manutenibilidade dos casos de teste;
- ***Tópico 7***. Relatório de informações pós-teste.

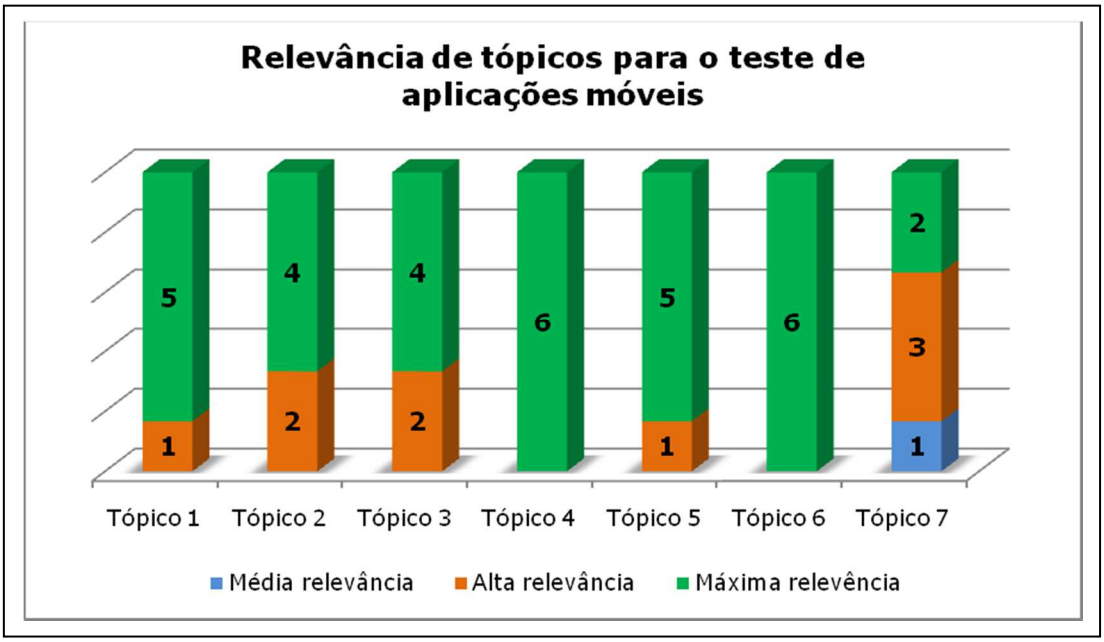

Figura 30 - Tópicos relevantes para o teste de aplicações móveis
Fonte: Elaborada pelo autor

Sugestão de melhorias na abordagem proposta. Os participantes descreveram que uma estratégia apoiada pela geração dinâmica de eventos simplificaria a inclusão de características ao modelo de teste com nenhuma ou pouca decisão a ser tomada por testadores. Tal fundamento poderia, inclusive, estender a estratégia *Edge Template* e apoiar a geração de distintas versões recomendadas de modelos ESG com base em diferentes perfis parametrizados de modelos de teste. Por exemplo, (i) avaliar aspectos de interface e usabilidade, (ii) simular propriedades de ambientes críticos como indisponibilidade de redes de dados e (iii) estimular ações e comportamentos de usuários em diferentes situações como estresse ou falta de atenção ao utilizar a aplicação móvel.

6.4. LIMITAÇÕES

A ferramenta MBTS4MA se baseia na estratégia para a geração orientada de eventos. Dessa forma, impõe-se que o testador revisite os eventos do modelo de teste e defina quais novos eventos serão acrescidos ao modelo ESG com base na estratégia *Edge Template*. Presume-se que aplicações móveis mais complexas e o uso extenso de distintas características podem dificultar e delongar a inclusão manual de eventos no modelo de teste, aumentando, consequentemente, o esforço na modelagem.

O reúso de modelos de teste pode ser conduzido para investigar a mesma aplicação móvel como consequência da inclusão de diversas características. Portanto, modelos de teste são propostos para verificar uma única aplicação móvel e, não obstante, até o mesmo conjunto de requisitos funcionais. A ferramenta MBTS4MA não dispõe de um módulo que administre diferentes versões de um mesmo modelo ESG. Portanto, o controle de versões é uma atividade manual gerenciada pelo testador e erros podem ocasionar a perda ou falta de integridade dos modelos de teste original e reusados com *Method Template* e *Edge Template*.

As características de (i) eventos específicos de dispositivos e (ii) interação não previsível de usuários foram incorporadas aos modelos de teste, porém os defeitos detectados não se relacionam a tais classes de eventos. Além disso, a execução dos testes em três dispositivos móveis com multiconfigurações e diversidade de telas não demonstrou, neste estudo, motivar ou propiciar a geração dos defeitos constatados. Futuras avaliações experimentais podem avaliar se as referidas características são efetivas e válidas em outras aplicações móveis e estudos industriais. Novos fundamentos e classes de características também podem ser idealizados para apoiar a abordagem proposta e estratégias investigadas.

6.5. LIÇÕES APRENDIDAS

Considerações e lições aprendidas foram identificadas quanto à experimentação em ambiente real e são apresentadas a seguir.

Modelos de teste integrados a metadados. O uso de estereótipos e *snippets* colaboram com uma melhor execução das etapas do processo de TBM, pois agilizam e simplificam a modelagem, geração e concretização, mesmo que parcial, de casos de teste. A ferramenta MBTS4MA disponibiliza as funcionalidades para que o modelo de teste seja elaborado de maneira concisa, padronizada e integrada a metadados extraídos da aplicação móvel em teste. Apesar de não terem sido necessários, novos estereótipos podem ser incorporados à ferramenta de apoio.

Padronização e estereotipação do modelo de teste. Visualmente, os modelos de teste elaborados na ferramenta MBTS4MA podem ser considerados mais facilmente compreendidos do que os obtidos no estudo relatado no Capítulo 4. Dentre os detalhes, cores e elementos gráficos diferenciados colaboram com a identificação de estereótipos e características do modelo ESG. Além disso, problemas anteriormente identificados não se repetiram como, por exemplo, (i) ausência dos nós especiais, "[“e “]" e (ii) falta de padrão na definição de rótulos dos eventos mapeados.

Integração da ferramenta de apoio com o SUT. A integração entre abordagem proposta, ferramenta de apoio e a aplicação móvel em teste contribuiu com a centralização de atividades a serem conduzidas pelos participantes no experimento. Afirma-se que a existência de um ambiente comum para a modelagem, geração e concretização de testes proporciona um maior foco e atenção dos testadores quanto às etapas do processo de teste de software.

6.6. DISCUSSÃO DOS RESULTADOS

O estudo experimental relatado avaliou a abordagem e ferramenta de apoio propostas para o reúso de modelos de teste em um ambiente industrial. Os resultados obtidos fornecem evidências acerca da viabilidade e aplicabilidade da abordagem e ferramenta apresentadas.

Os participantes que atuaram na avaliação experimental não possuíam conhecimentos e/ou experiências com o processo de TBM e a técnica de modelagem ESG. Por meio de uma breve sessão de treinamento, os conceitos de TBM e as orientações para uma correta modelagem com ESG foram apresentados.

A abordagem proposta para o reúso de modelos de teste por meio das estratégias *Method Template* e *Edge Template* foi explorada, assim como a ferramenta MBTS4MA e a plataforma de teste *Robotium*. Os resultados alcançados com êxito na condução do estudo experimental evidenciam que a sessão de treinamento foi satisfatória, promovendo uma específica, porém concisa e válida aprendizagem dos fundamentais conceitos.

A adoção da abordagem e ferramenta MBTS4MA em um ambiente industrial possibilitou analisar o reúso de modelos de teste na verificação de aplicações móveis desenvolvidas em Android. Os resultados identificados comprovam que a abordagem fundamentada em TBM e apoiada pelas estratégias *Method Template* e *Edge Template* são instrumentos que promovem avanços ao teste no contexto de mobilidade. Investigou-se (i) a redução do esforço demandado para a concretização de casos de teste e (ii) o teste de distintas características relacionadas ao contexto de aplicações móveis.

A resolução da questão de pesquisa Q1 (“*Quais os esforços demandados na modelagem e concretização de casos de teste a partir das estratégias Method Template e Edge Template?*”) deve ser formulada sob os aspectos de (i) modelagem e (ii) concretização. As assertivas e discussões referidas à Q1 são embasadas na análise dos resultados apresentados nas Tabelas 6, 7 e 8.

Esforço para modelagem. Constata-se que a estratégia *Method Template* interfere diretamente na etapa de modelagem, pois se observa que o tempo demandado no reúso do modelo de teste para incorporar tal estratégia é aumentado. Portanto, a definição de estereótipos em conjunto à estratégia *Method Template* resulta em um esforço manual superior quando comparado a uma abordagem padrão de TBM. Mais precisamente, o esforço avaliado no estudo é confirmado por um aumento de 25% a 30% no tempo de modelagem em relação à mesma atividade sem a estratégia *Method Template*.

A adoção da estratégia *Edge Template* também resulta no aumento do tempo consumido para o reúso do modelo de teste. O esforço requisitado é inferior ao relatado para a estratégia *Method Template*. Para o estudo experimental, verificou-se um acréscimo de 12% a 19% no tempo de modelagem a partir da estratégia *Edge Template*.

Esforço para concretização. A estratégia *Method Template* contribui com a redução do esforço orientado à concretização de testes. Os resultados obtidos indicam que o tempo demandado para concretizar os casos de teste abstratos diminuiu entre 38% e 48% quando comparado à concretização realizada em uma abordagem padrão de TBM. A diferença no tempo é expressiva e tal avanço se deve ao uso de estereótipos em eventos do modelo de teste. A geração de casos de teste completa ou até mesmo parcialmente concretizados otimiza a etapa de concretização, pois menos eventos devem ser manualmente concretizados e, por consequência, o esforço também é reduzido.

Para a avaliação experimental, a estratégia *Edge Template* produziu modificações quase que nulas, negativa e positivamente, quanto ao esforço direcionado à concretização. Apesar dos modelos de teste reusados com *Edge Template* incorporarem novos eventos, os estereótipos utilizados no estudo não influenciaram no esforço já relatado para a concretização quando se adota a estratégia *Method Template*. Portanto, a estratégia *Edge Template* oportuniza o teste de distintas características inerentes ao contexto de aplicações móveis sem que o tempo e o esforço para a concretização manual sejam aumentados. Há de se considerar que, se a estratégia *Edge Template* for utilizada independentemente da estratégia *Method Template*, o testador assumirá a responsabilidade de concretizar os eventos previamente mapeados. Dessa forma, o tempo e o esforço para a concretização serão equivalentes na adoção de uma abordagem padrão.

É importante ressaltar que os esforços tanto na modelagem quanto na concretização com as estratégias *Method Template* e *Edge Template* estão diretamente relacionados à quantidade de eventos, bem como à extensão e complexidade do modelo ESG. Conclui-se que o reúso de modelos de teste é uma abordagem válida para apoiar o TBM em aplicações móveis e evidências demonstram a redução do esforço manual demandado na etapa de concretização de casos de teste.

A detecção de cinco defeitos nas aplicações móveis selecionadas colaboram com a resolução da questão de pesquisa Q2 (“*Em relação às características e fundamentos propostos, a abordagem demonstrou ser promissora quanto à capacidade de detectar defeitos?*”).

De cinco defeitos revelados pela abordagem de reúso de modelos de teste, dois foram identificados a partir de modelos reusados com a estratégia *Method*

Template. Apesar de não terem sido executados, presume-se que os testes gerados pela definição padrão de TBM e ESG também são adequados para detectar tais defeitos, tendo em vista que a estratégia *Method Template* não modifica a estrutura do modelo de teste, mas auxilia na redução do esforço para a concretização de casos de teste. Três defeitos foram constatados com a execução dos testes relacionados aos modelos que adotaram a estratégia *Edge Template*. Os defeitos detectados estão relacionados às distintas características de (i) eventos de telefonia para GSM/SMS e (ii) eventos de sensores e componentes de hardware.

Os testes foram executados em três diferentes configurações de dispositivos móveis e telas. A verificação das aplicações em multiconfigurações de dispositivos não contribuiu com a detecção de novos defeitos. Além disso, a variedade em tamanhos e resoluções de telas igualmente não revelou defeitos.

A simulação de (iii) eventos específicos de dispositivos também não resultou na detecção de defeitos. Mudanças no nível e estado da bateria do dispositivo móvel, bem como perda de conexão e reconexões em redes de dados não interferiram nas funcionalidades verificadas.

A característica de (iv) interação não previsível de usuários possibilitou verificar o funcionamento das aplicações móveis como se um típico usuário real a estivesse manualmente testando. Defeitos não foram detectados por tal motivo, porém se presume que empresas podem, involuntariamente, enfatizar o teste manual apenas nos requisitos funcionais e omitir ou desconsiderar a verificação de diferentes comportamentos assumidos por usuários.

Os resultados comparativos relatados na Tabela 9 apresentam uma diferença aceitável quanto aos tempos de execução dos testes nos Estágios 1 e 2. Portanto, considera-se que a adoção da abordagem proposta e, em notável, da estratégia *Edge Template* aumentam o tempo total de execução dos casos de teste concretizados, porém se estendem as expectativas e oportunidades de detecção de defeitos.

A capacidade de detecção de defeitos por meio da abordagem proposta evidencia que o reúso de modelos de teste e as estratégias *Method Template* e *Edge Template* são mecanismos válidos para a automação de testes em aplicações móveis. Além disso, a abordagem promove avanços quanto (i) à redução do esforço demandado na concretização e (ii) à detecção de defeitos com base em distintas características do contexto de mobilidade. Futuras pesquisas devem investigar,

propor e avaliar abordagens específicas e/ou estratégias para apoiar a etapa de modelagem, otimizando-a para resultar em esforços menores e/ou com pouca dependência de processos manuais.

6.7. CONSIDERAÇÕES FINAIS

Neste capítulo, foi apresentado um estudo experimental para avaliar a abordagem de reúso de modelos de teste, bem como as estratégias *Method Template* e *Edge Template*. A avaliação foi conduzida em colaboração a uma empresa multinacional de TI que atua no desenvolvimento de aplicações móveis para a plataforma Android.

Durante o estudo experimental, uma sessão de treinamento forneceu os conceitos essenciais acerca de TBM e das definições e fundamentos relatados no Capítulo 5. A abordagem proposta apresentou facilidade na compreensão e aplicabilidade em testes reais, pois se verificou a utilização de maneira correta e sem dificuldades. Os participantes demonstraram significante interesse em adotar a abordagem e ferramentas propostas como parte do processo de teste de aplicações móveis. Além disso, sugestões de melhorias e a recomendação de novas funcionalidades à ferramenta MBTS4MA foram apontadas como (i) uma estratégia para a geração dinâmica de eventos e (ii) um módulo na ferramenta MBTS4MA para o controle de versões de modelos ESG.

Os resultados identificados e discutidos fornecem evidências que comprovam a redução do esforço demandado na etapa de concretização de casos de teste em TBM e ESG quando se adota o reúso de modelos de teste a partir das estratégias *Method Template* e *Edge Template*. Os subsídios constatados na avaliação experimental na indústria demonstram que a abordagem e os fundamentos propostos são mecanismos válidos e promissores para o TBM no contexto de mobilidade. A abordagem e ferramenta de apoio MBTS4MA contribuíram com a modelagem, geração e concretização de casos de teste, otimizando e especializando os passos de TBM para o reúso de modelos de teste. Como resultados da execução dos testes, cinco defeitos foram detectados nas aplicações móveis verificadas.

No próximo capítulo, conclui-se esta dissertação e, objetivamente, as principais contribuições e limitações são revisitadas. Trabalhos futuros são apresentados, esboçando possíveis tópicos de interesse para novas pesquisas.

7. CONCLUSÃO

O teste no contexto de mobilidade tem sido cada vez mais investigado devido a sua importância no processo de desenvolvimento de aplicações móveis (WASSERMAN, 2010; MUCCINI et al., 2012; JOORABCHI et al., 2013; GAO et al., 2014; PICCO et al., 2014). Particularmente, a engenharia de software para aplicações móveis se difere da tradicional por ser mais especializada e por estabelecer particularidades como, por exemplo, diversidade de configurações e telas, uso de sensores e componentes de hardware e mecanismos de conectividade e comunicação. Portanto, abordagens e estratégias específicas, bem como ferramentas de apoio vêm sendo propostas para considerar as particularidades e desafios impostos pelo teste no contexto de aplicações móveis.

Nesta dissertação, contribuições são apresentadas com o objetivo de investigar a adoção de TBM e da técnica de modelagem ESG no contexto de aplicações móveis. Os resultados discutidos fornecem evidências positivas acerca da aplicabilidade de abordagens que se fundamentam em TBM e ESG. Além disso, destacam-se avanços quanto à abordagem de TBM orientada ao reúso de modelos de teste. Dentre os progressos avaliados, os resultados evidenciam (i) a redução do esforço demandado para a concretização de casos de teste e (ii) a capacidade de detecção de defeitos a partir do teste de distintas características relacionadas ao contexto de mobilidade.

Na Seção 7.1, são revisitadas as contribuições desta dissertação de mestrado. Na Seção 7.2, são apresentados as limitações e trabalhos futuros que estimulam avanços no teste de aplicações móveis. Por fim, na Seção 7.3, são listadas as publicações que se relacionam a esta dissertação.

7.1. CONTRIBUIÇÕES

Nesta seção, são revisitadas as principais contribuições identificadas na condução da dissertação.

Análise de pesquisas relevantes na área de testes de aplicações móveis. No Capítulo 3 é apresentado um mapeamento sistemático conduzido que auxiliou na identificação de 108 pesquisas consideradas como relevante na área de teste de aplicações móveis. Os resultados confirmam o crescimento em publicações de pesquisas ao longo dos últimos anos. Pesquisas experimentais e estudos exploratórios são evidenciados. Entretanto, avaliações industriais são exploradas em proporção reduzida. Outro detalhe é o uso de alguma estratégia apoiada por TBM em aproximadamente 22% das pesquisas selecionadas. A plataforma Android é amplamente utilizada dentre às pesquisas e mantém uma posição ascendente a cada ano. Por fim, estudos motivam a concepção, a modelagem e a implementação de abordagens e ferramentas específicas para o teste de aplicações móveis.

Avaliação experimental do TBM no contexto de aplicações móveis. No Capítulo 4 é descrito um estudo que avalia a adoção de TBM no contexto de aplicações móveis. A avaliação experimental fornece subsídios para a aplicabilidade da abordagem de TBM e ESG na verificação de requisitos funcionais em aplicações móveis. A etapa de concretização foi conduzida por meio da API provida pela plataforma de teste *Robotium*. Dentre os resultados identificados, (i) a geração automática de casos de teste e (ii) a capacidade de detecção de defeitos em aplicações Android se evidenciam como contribuições à adoção de TBM em aplicações móveis.

Proposta de abordagem específica de TBM para aplicações móveis. No Capítulo 5 é apresentada uma abordagem de TBM para apoiar a verificação de aplicações Android, incorporando duas estratégias: *Method Template* e *Edge Template*. Respectivamente, as estratégias propostas objetivam (i) reduzir o esforço manual demandado na concretização de casos de teste e (ii) testar características típicas de aplicações móveis que são adicionadas aos modelos ESG. As características adicionadas aos modelos de teste são categorizadas em (i) eventos específicos de dispositivos móveis, (ii) interação não previsível de usuários, (iii) eventos de telefonia para GSM/SMS e (iv) eventos de sensores e componentes de hardware.

Ferramenta de apoio MBTS4MA. Na Seção 5.4, são apresentados os detalhes arquiteturais e as funcionalidades da ferramenta *Model-Based Test Suite For Mobile Applications* (MBTS4MA). A ferramenta simplifica as etapas de modelagem, geração

e concretização, mesmo que parcial, dos casos de teste. Além disso, possibilita a extração e manipulação de metadados do SUT que são utilizados na estereotipação de eventos do modelo ESG. A versão implementada nesta dissertação se integra à plataforma *Robotium* para a concretização dos casos de teste em Android.

Avaliação da abordagem e ferramenta MBTS4MA em um ambiente industrial. No Capítulo 6 é relatado um estudo conduzido em ambiente industrial para analisar a abordagem e a ferramenta propostas. O estudo foi realizado com seis profissionais que atuam em uma empresa multinacional de TI que não adota testes automatizados. Os resultados obtidos fornecem evidências acerca da redução do esforço demandado na concretização de casos de teste, bem como da capacidade de detecção de defeitos. No total, cinco defeitos foram detectados na execução dos testes gerados a partir de modelos reusados com as estratégias *Method Template* e *Edge Template*. Dessa forma, as evidências apresentadas demonstram que o reúso de modelos de teste é um mecanismo válido e promissor para o teste no contexto de aplicações móveis.

7.2. LIMITAÇÕES E TRABALHOS FUTUROS

As limitações e trabalhos futuros identificados são apresentados a seguir.

Novos estudos experimentais. Apesar da abordagem e ferramenta de apoio MBTS4MA terem sido avaliadas em um ambiente industrial, novos estudos devem investigar e relatar as vantagens e desafios em distintos cenários empíricos e/ou reais. Por exemplo, a evolução e manutenção em modelos de teste podem ser investigadas para avaliar os benefícios e dificuldades em incorporar o reúso de TBM em uma fábrica de software ou em uma *Software Product Line*. *Benchmarks* que se baseiam em aplicações móveis *open source* também podem contribuir com a avaliação dos fundamentos mencionados e com a identificação de novas características que apoiam a abordagem proposta. Métricas de custo e eficácia devem ser mensuradas e correlacionadas para avaliar aspectos quantitativos e qualitativos da abordagem proposta (MALDONADO et al., 2004). Por fim, espera-se

comparar a abordagem e a ferramenta propostas em empresas que já utilizam o teste automatizado de aplicações móveis.

Geração parcial de modelo ESG a partir de metadados do SUT. A modelagem é uma etapa manual e extensa quando muitos requisitos funcionais devem ser verificados. Mecanismos automatizados podem ser implementados para extrair metadados referentes aos fluxos que a aplicação pode assumir. Dessa forma, modelos ESG podem ser parcialmente sugeridos com base em eventos de componentes, transições entre telas e outros artefatos de uma aplicação móvel.

Refatoração de ferramenta de apoio MBTS4MA. A ferramenta MBTS4MA pode ser estendida para incorporar novos fundamentos e características. Entretanto, trabalhos futuros podem conduzir à refatoração de componentes tecnológicos para possibilitar a integração de distintas plataformas de teste, como *Robolectric* e *Espresso*, e até de plataforma de desenvolvimento, como Apple iOS e Microsoft Windows Phone. Por fim, também se deve considerar o interesse de testes em aplicações móveis para *smartwatches* e *wearables*.

7.3. DIVULGAÇÃO DOS RESULTADOS

Durante o progresso desta dissertação de mestrado, obtiveram-se as publicações apresentadas a seguir.

- FARTO, G. C.; ENDO, A. T. **Mechanisms to support automated testing of mobile applications**. In: XII Workshop de Teses e Dissertações em Qualidade de Software (WTDQS), Blumenau, Santa Catarina, p. 1-6, 2014.

- FARTO, G. C.; ENDO, A. T. **Evaluating the Model-Based Testing Approach in the Context of Mobile Applications**. In: Proc. of the XL Latin American Computing Conference (CLEI), Montevidéu, Uruguai, p. 1-12, 2014.
  - Convidado para publicação em uma edição especial da *Electronic Notes in Theoretical Computer Science* (ENTCS), DOI: 10.1016/j.entcs.2015.05.001.

- FARTO, G. C.; ENDO, A. T. Automação de testes para mobile - Trabalhe com as ferramentas Monkey e MonkeyRunner. **Engenharia de Software Magazine**, n. 76, p. 44-59, Junho de 2015. Disponível em <http://www.devmedia.com.br/revista-engenharia-de-software-magazine-76/32701>.

- FARTO, G. C.; ENDO, A. T. Automação de testes para mobile - Uso das ferramentas Robolectric e Espresso. **Engenharia de Software Magazine**, n. 77, p. 53-63, Julho de 2015. Disponível em <http://www.devmedia.com.br/revista-engenharia-de-software-magazine-77/32941>.

- FARTO, G. C.; ENDO, A. T. Automação de testes para mobile - Plataforma de testes Robotium. **Engenharia de Software Magazine**, n. 78, p. 48-57, Agosto de 2015. Disponível em <http://www.devmedia.com.br/revista-engenharia-de-software-magazine-78/33219>.

A seguir, os seguintes artigos encontram-se em fase de elaboração:

- FARTO, G. C.; ENDO, A. T. **An approach to reuse model-based testing in mobile applications**. A ser submetido em revista na área de engenharia de software.

- FARTO, G. C.; ENDO, A. T. **A systematic mapping study on mobile application testing**. A ser submetido em revista na área de engenharia de software.

REFERÊNCIAS

ABLESON, W. F.; SEN, R.; KING, C.; ORTIZ, E. **Android in action**. 3rd ed., New York: Elsevier/Manning Publications Co., 2012.

AHO, A. V.; DAHBURA, A. T.; LEE, D.; UYAR, M. U. **An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours**. Los Alamitos, CA, USA: IEEE Computer Society Press, p. 427-438, 1995.

ALI, S.; IQBAL, M. Z.; ARCURI, A.; BRIAND, L. **A Search-Based OCL constraint solver for Model-Based Test data generation**. In: Proc. of the 11th International Conference on Quality Software (QSIC), p. 41-50, 2011.

AMALFITANO, D.; AMATUCCI, N.; FASOLINO, A. R.; GENTILE, U.; MELE, G.; NARDONE, R.; VITTORINI, V.; MARRONE, S. **Improving code coverage in Android apps testing by exploiting patterns and automatic test case generation**. In: Proc. of the International Workshop on Long-term Industrial Collaboration on Software Engineering, p. 29-34, 2014b.

AMALFITANO, D.; FASOLINO, A. R.; TRAMONTANA, P.; ROBBINS, B. **Testing Android mobile applications: Challenges, strategies, and approaches**. Advances in Computers, v. 89, p. 1-52, 2013.

AMALFITANO, D.; FASOLINO, A. R.; TRAMONTANA, P.; TA, B. D.; MEMON, A. M. **MobiGUITAR: Automated Model-Based Testing of mobile apps**. IEEE Software, v. 32, p. 53-59, 2014a.

AMMANN, P.; OFFUTT, J. **Introduction to software testing**. Cambridge University Press: New York, NY, USA, 2008.

ANDROID. Android Developers. Disponível em: <http://developer.android.com>. Acesso em: Setembro de 2015.

BARNETT, M.; GRIESKAMP, W.; NACHMANSON, L.; SCHULTE, W.; TILMMANN, N.; VEANES, M. **Model-based testing with AsmL.NET**. In: Proc. of the 1st European Conference on Model-Driven Software Engineering, p. 12-19, 2003.

BEIZER, B.; **Software testing techniques**. 2nd ed., New York: Van Nostrand Eeinhold, 1990.

BELLI, F.; BUDNIK, C. J.; WHITE, L. **Event-based modelling, analysis and testing of user interactions: Approach and case study**. Software Testing, Verification and Reliability, v. 16, n. 1, p. 3-32, 2006.

BELLI, F.; ENDO, A. T.; LINSCHULTE, M.; SIMAO, A. **A holistic approach to model-based testing of Web service compositions**. Software: Practice and Experience, v. 44, n. 2, p. 201-234, 2013.

BELLI, F.; LINSCHULTE, M. **Event-driver modeling and testing of web services**. In: Proc. of the 32nd IEEE Internacional Computer Software and Applications Conference (COMPSAC), p. 1168-1173, 2008.

BINDER, R. V.; KRAMER, A.; LEGEARD, B. **2014 Model-based testing user survey: Results**. Relatório Técnico, ETSI 2nd User Conference Advanced Automated Testing (UCAAT), Munique, Alemanha, 2014. Disponível em: <http://model-based-testing.info/2014/12/09/2014-mbt-user-survey-results/>. Acesso em: Novembro de 2015.

BLACKBURN, M.; BUSSER, R.; NAUMAN, A. **Why model-based test automation is different and what you should know to get started**. Relatório Técnico, Software Productivity Consortium, 2004.

BOUQUET, F.; DEBRICON, S.; LEGEARD, B.; NICOLET, J. B. **Extending the unified process with model-based testing**. In: Proc. of the 3rd International Workshop on Model Development, Validation and Verification (MoDeVa), Genova, Italy, p. 2-15, 2006.

BOUQUET, F.; GRANDPIERRE, C.; LEGEARD, B.; PEUREUX, F.; VACELET, N.; UTTING, M. **A subset of precise UML for Model-Based Testing**. In: Proc. of the 3rd International Workshop on Advances in Model-Based Testing (A-MOST), New York, NY, USA: ACM, p. 95-104, 2007.

BUDD, T. A. **Mutation analysis: Ideas, example, problems, and prospects**. Computer Program Testing, p. 129-148, 1981.

CALABASH. Automated acceptance testing for mobile apps. Disponível em: <http://calaba.sh/>. Acesso em: Novembro de 2015.

CADAR, C.; SEN, K. **Symbolic execution for software testing: Three decades later**. Communications of the ACM, v. 56, n. 2, p. 82-90, 2013.

COSTA, P.; PAIVA, A. C. R.; NABUCO, M. **Pattern Based GUI Testing for mobile applications**. In: Proc. of the 9th International Conference on the Quality of Information and Communications Technology (QUATIC), p. 66-74, 2014.

DALAL, S. R.; JAIN A.; KARUNANITHI N.; LEATON J. M.; LOTT C. M.; PATTON G. C.; HOROWITZ B. M. **Model-based testing in practice.** In: Proc. of the 21st International Conference on Software Engineering (ICSE) (Los Angeles, USA: ACM), p. 285-294, 1999.

DANTAS, V.L.L.; MARINHO, F.G.; DA COSTA, A.L.; ANDRADE, R.M.C. **Testing requirements for mobile applications.** In: Proc. of the 24th International Symposium on Computer and Information Sciences (ISCIS), p. 555-560, 2009.

DEITEL, P. J.; DEITEL, H. M.; DEITEL, A.; MORGANO, M. **Android for programmers: An app-driven approach**. Upper Saddle River, NJ: Prentice Hall, 2012.

DENG, L.; MIRZAEI, N.; AMMANN, P.; OFFUTT, J. **Towards mutation analysis of Android apps**. In: Proc. of the 8th International Conference on Software Testing, Verification and Validation Workshops (ICSTW), p. 1-10, 2015.

DEMILLO, R. A. **Software testing and evaluation**. The Benjamin/Commings Publishing Company, Inc, 1987.

DEV, R.; JÄÄSKELÄINEN, A.; KATARA, M. **Model-Based GUI Testing: Case smartphone camera and messaging development**. Advances in Computers, v. 85, p. 65-122, 2012.

DIAS-NETO, A. C.; TRAVASSOS, G. H. **Supporting the combined selection of Model-Based Testing techniques**. In: Proc. of the IEEE Transactions on Software Engineering (TSE), v. 40, n. 10, p. 1025-1041, 2014.

DINH, H. T.; LEE, C.; NIYATO, D.; WANG, P. **A survey of mobile cloud computing: Architecture, applications, and approaches**. Wireless Communications and Mobile Computing, v. 13, n. 18, p. 1587-1611, 2011.

EL-FAR, I. K.; WHITTAKER, J. A. **Model-based software testing**. In: Encyclopedia on Software Engineering, Wiley, p. 825-837, 2011.

ENDO, A. T. **Model-based testing of service oriented applications**. Tese de doutoramento, ICMC/USP, São Carlos, SP, 2013. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-20062013-140259>. Acesso em: Setembro de 2015.

ENTIN, V.; WINDER, M.; ZHANG, B.; CHRISTMANN, S. **Introducing model-based testing in an industrial scrum project**. In: Proc. of the 7th International Workshop on Automation of Software Test (AST), p. 43-49, 2012.

ESPRESSO. Espresso. Disponível em: <https://google.github.io/android-testing-support-library/docs/espresso/index.html>. Acesso em: Novembro de 2015.

FARCHI, E.; HARTMAN, A.; PINTER, S. S. **Using a model-based test generator to test for standard conformance**. In: IBM Systems Journal 41, p. 89-110, 2002.

FARTO, G. C.; ENDO, A. T. Automação de testes para mobile - Plataforma de testes Robotium. **Engenharia de Software Magazine**, n. 78, p. 48-57, Agosto de 2015, 2015c. Disponível em <http://www.devmedia.com.br/revista-engenharia-de-software-magazine-78/33219>. Acesso em: Outubro de 2015.

FARTO, G. C.; ENDO, A. T. Automação de testes para mobile - Trabalhe com as ferramentas Monkey e MonkeyRunner. **Engenharia de Software Magazine**, n. 76, p. 44-59, Junho de 2015, 2015a. Disponível em <http://www.devmedia.com.br/revista-engenharia-de-software-magazine-76/32701>. Acesso em: Outubro de 2015.

FARTO, G. C.; ENDO, A. T. Automação de testes para mobile - Uso das ferramentas Robolectric e Espresso. **Engenharia de Software Magazine**, n. 77, p. 53-63, Julho

de 2015, 2015b. Disponível em <http://www.devmedia.com.br/revista-engenharia-de-software-magazine-77/32941>. Acesso em: Outubro de 2015.

FARTO, G. C.; ENDO, A. T. **Evaluating the Model-Based Testing approach in the context of mobile applications**. In: Proc. of the XL Latin American Computing Conference (CLEI), Montevidéu, Uruguai, p. 1-12, 2014a.

FARTO, G. C.; ENDO, A. T. **Mechanisms to support automated testing of mobile applications**. In: XIII Simpósio Brasileiro de Qualidade de Software (SBQS) - XII Workshop de Teses e Dissertações em Qualidade de Software (WTDQS), Blumenau, Santa Catarina, p. 1-6, 2014b.

FORMAN, G. H.; ZAHORJAN, J. **The challenges of mobile computing**. Computer, v. 47, n. 4, p. 38-47, 1994.

FRANTZEN, L.; TRETMANS, J.; WILLEMSE, T. **A symbolic framework for Model-Based Testing**. In: Formal Approaches to Software Testing and Runtime Verification (FATES/RV). Lecture Notes in Computer Science, n. 4262. Springer, p. 40-54, 2006.

GADDAH, A.; KUNZ, T. **A survey of middleware paradigms for mobile computing**. Relatório Técnico, Carleton University Systems and Computing Engineering, 2003.

GAO, J.; XIAOYING B.; WEI-TEK T.; UEHARA, T. **Mobile application testing: A tutorial**. Computer, v. 47, n. 2, p. 46-55, 2014.

GARTNER, Inc. Gartner says emerging markets drove worldwide smartphone sales to 15.5 percent growth in third quarter of 2015, Novembro de 2015 (2015b). Disponível em: <http://www.gartner.com/newsroom/id/3169417>. Acesso em: Novembro de 2015.

GARTNER, Inc. Gartner says emerging markets drove worldwide smartphone sales to 19 percent growth in first quarter of 2015, Maio de 2015 (2015c). Disponível em: <http://www.gartner.com/newsroom/id/3061917>. Acesso em: Novembro de 2015.

GARTNER, Inc. Gartner says smartphone sales surpassed one billion units in 2014, Março de 2015 (2015a). Disponível em: <http://www.gartner.com/newsroom/id/2996817>. Acesso em: Novembro de 2015.

GARTNER, Inc. Gartner says worldwide smartphone sales recorded slowest growth rate since 2013, Agosto de 2015 (2015d). Disponível em: <http://www.gartner.com/newsroom/id/3115517>. Acesso em: Novembro de 2015.

GIL, A. C. **Como elaborar projetos de pesquisa**. 5th ed., São Paulo: Atlas, 2010.

GOOGLE TRENDS. Indexação de busca para “mobile application testing” e “mobile app testing”. Disponível em: <https://www.google.com/trends/explore#q=mobile+application+testing%2C+mobile+app+testing>. Acesso em: Novembro de 2015.

GRIEBE, T.; GRUHN, V. **A model-based approach to test automation for context-aware mobile applications**. In: Proc. of the 29th Annual ACM Symposium on Applied Computing (SAC), p. 420-427, 2014.

GRIESKAMP, W.; KICILLOF, N.; STOBIE, K.; BRABERMAN, V. A. **Model-based quality assurance of protocol documentation: Tools and methodology**. Software Testing, Verification and Reliability, v. 21, n. 1, p. 55-71, 2011.

HARROLD, M. J. **Testing: A roadmap**. In: Proc. of the 22th International Conference on Software Engineering (ICSE), p. 61-72, 2000.

HIERONS, R. M.; BOGDANOV, K.; BOWEN, J. P.; CLEAVELAND, R.; DERRICK, J.; DICK, J.; GHEORGHE, M.; HARMAN, M.; KAPOOR, K.; KRAUSE, P.; LÜTTGEN, G.; SIMONS, A. J. H.; VILKOMIR, S.; WOODWARD, M. R.; ZEDAN, H. **Using formal specifications to support testing**. ACM Computing Surveys (CSUR), v. 41, n. 2, p. 1-76, 2009.

IDC. Worldwide smartphone growth expected to slow to 10.4% in 2015, down from 27.5% growth in 2014, according to IDC, Agosto de 2015. Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS25860315>. Acesso em: Novembro de 2015.

IEEE Standard Glossary of Software Engineering Terminology. **Padrão 620.12**, IEEE, 1990.

ITU. International Telecommunications Union (ITU) - ICT Facts and Figures - The world in 2015, Maio de 2015. Disponível em: <http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf>. Acesso em: Novembro de 2015.

ITU. International Telecommunications Union (ITU) - The world in 2014 - ICT Facts and Figures, Abril de 2014. Disponível em: <http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2014-e.pdf>. Acesso em: Novembro de 2015.

JÄÄSKELÄINEN, A.; KATARA, M.; KERVINEN, A.; HEISKANEN, H.; MAUNUMAA, M.; PÄÄKKÖNEN, T. **Model-Based Testing service on the Web**. In: Proc. of the 21st International Conference on Testing of Communicating Systems (IFIP). Lecture Notes in Computer Science, v. 5047. Springer, p. 38-53, 2008a.

JÄÄSKELÄINEN, A.; KATARA, M.; KERVINEN, A.; MAUNUMAA, M.; PÄÄKKÖNEN, T.; TAKALA, T.; VIRTANEN, H. **Automatic GUI test generation for smartphone applications: An evaluation**. In: Proc. of the 31st International Conference on Software Engineering (ICSE), p. 112-122, 2009a.

JÄÄSKELÄINEN, A.; KERVINEN, A.; KATARA, M. **Creating a test model library for GUI testing of smartphone applications**. In: Proc. of the 8th International Conference on Quality Software (QSIC), p. 276-282, 2008b.

JÄÄSKELÄINEN, A.; KERVINEN, A.; KATARA, M.; VALMARI, A.; VIRTANEN, H. **Synthesizing test models from test cases**. In: Proc. of the 4th International Haifa

Verification Conference on Hardware and Software: Verification and Testing. Lecture Notes in Computer Science, v. 5394. Springer, p. 179-193, 2009b.

JANICKI, M.; KATARA, M.; PÄÄKKÖNEN, T. **Obstacles and opportunities in deploying model-based GUI testing of mobile software: A survey**. Software Testing, Verification and Reliability, v. 22, n. 5, p. 313-341, 2012.

JAVA. Java.com. Disponível em: <https://www.java.com/en/>. Acesso em: Outubro de 2015.

JENSEN, C. S.; PRASAD, M. R.; MOLLER, A. **Automated testing with targeted event sequence generation**. In: Proc. of the 22nd International Symposium on Software Testing and Analysis (ISSTA), p. 67-77, 2013.

JING, J.; HELAL, A. S.; ELMAGARMID, A. **Client-server computing in mobile environments**. ACM Computing Surveys (CSUR), v. 31, n. 2, p. 117-157, 1999.

JOORABCHI, M. E.; MESBAH, A.; KRUCHTEN, P. **Real challenges in mobile app development**. In: Proc. of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), p. 15-24, 2013.

KING, J. C. **Symbolic execution and program testing**. Communications of the ACM, v. 19, n. 7, p. 385-394, 1976.

KIRUBAKARAN, B.; KARTHIKEYANI, V. **Mobile application testing: Challenges and solution approach through automation**. In: Proc. of the International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), p. 79-84, 2013.

KITCHENHAM, B.; PFLEEGER, S. L.; PICKARD, L. M.; JONES, P. W.; HOAGLIN, D. C.; EMAM, K. E.; ROSENBERG, J. **Preliminary guidelines for empirical research in software engineering**. In: IEEE Transactions on Software Engineering (TSE), v. 28, p. 721-734, 2002.

KRONBAUER, A. H.; SANTOS, C. A. S.; VIEIRA, V. **Smartphone applications usability evaluation: A hybrid model and its implementation**. In: Proc. of the 4th International Conference on Human-Centered Software Engineering (HCSE), p. 146-163, 2012.

LECHETA, R. **Google Android: Aprenda a criar aplicações com dispositivos móveis com o Android SDK**. 3rd ed., São Paulo: Novatec, 2013.

LEE, D.; YANNAKAKIS, M. **Principles and methods of testing finite state machines: A survey**. Proc. of the IEEE, v. 84, n. 8, p. 1090-1123, 1996.

LI, A.; QIN, Z.; CHEN, M.; LIU, J. **ADAutomation: An activity diagram based automated GUI testing framework for smartphone applications**. In: Proc. of the 8th International Conference on Software Security and Reliability (SERE), p. 68-77, 2014.

LINSCHULTE, M. **On the role of test sequence length, model refinement, and test coverage for reliability**. Tese de doutoramento, Universität Paderborn, 2013. Disponível em: <http://d-nb.info/1037776089/34>. Acesso em: Julho de 2015.

LU, L.; HONG, Y.; HUANG, Y.; SU, K.; YAN, Y. **Activity page based functional test automation for Android application**. In: Proc. of the 3rd World Congress on Software Engineering (WCSE), p. 1-4, 2012.

MALDONADO, J. C. **Critérios potenciais usos: Uma contribuição ao teste estrutural de software**. Tese de doutoramento, DCA/FEE/UNICAMP, Campinas, SP, 1991.

MALDONADO, J. C.; AURI, E. F. B.; VINCENZI, M. R.; DELAMARO, M. E.; SOUZA, S. R. S.; JINO, M. **Introdução ao teste de software**. Instituto de Ciências Matemáticas e de Computação - ICMC/USP, Nota Didática, n. 65, 2004.

MARIJAN, D. **A review of two experiences from applying Model Based Testing in practice**. In: Proc. of the 23rd IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), p. 231-236, 2012.

MCCABE, T. J. **A complexity measure**. In: Proc. of the 2nd International Conference on Software Engineering (ICSE). Los Alamitos, CA, USA: IEEE Computer Society Press, p. 407, 1976.

MEDNIEKS, Z.; DORNIN, L.; MEIKE, G. B.; NAKAMURA, M. **Programming Android: Java programming for the new generation of mobile devices**. 2nd ed., O'Reilly Media, 2012.

MÉNDEZ-PORRAS, A.; QUESADA-LÓPEZ, C.; JENKINS, M. **Automated testing of mobile applications: A systematic map and review**. In: Proc. of the 18th Ibero-American Conference on Software Engineering (CIBSE), p. 195-208, 2015.

MONKEY. Monkey - UI/Application Exerciser Monkey. Disponível em: <http://developer.android.com/tools/help/monkey.html>. Acesso em: Novembro de 2015.

MONKEYRUNNER. MonkeyRunner. Disponível em: <http://developer.android.com/tools/help/monkeyrunner_concepts.html>. Acesso em: Novembro de 2015.

MUCCINI, H.; DI FRANCESCO, A.; ESPOSITO, P. **Software testing of mobile applications: Challenges and future research directions**. In: Proc. of the 7th International Workshop on Automation of Software Test (AST), p. 29-35, 2012.

MYERS, G. J.; SANDLER, C.; BADGETT, T.; THOMAS, T. M. **The art of software testing**. John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.

OPEN HANDSET ALLIANCE. Open Handset Alliance Overview. Disponível em: <http://www.openhandsetalliance.com/oha_overview.html>. Acesso em: Agosto de 2015.

ORSO, A.; ROTHERMEL, G. **Software testing: A research travelogue (2000-2014)**. In: Proc. of the 36th International Conference on Software Engineering (ICSE), Hyderabad, India, p. 117-132, 2014.

PELESKA, J. **Industrial-strength Model-Based Testing; State of the art and current challenges**. In: Proc. of the 8th Workshop on Model-Based Testing. Electronic Proceedings in Theoretical Computer Science, v. 111. Open Publishing Association, Roma, Itália, p. 3-28, 2013.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. **Systematic mapping studies in software engineering**. In: Proc. of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), vol. 17, p. 1-10, 2008.

PETRENKO, A.; BORODAY, S.; GROZ, R. **Confirming configurations in EFSM testing**. In: IEEE Transactions on Software Engineering (TSE), v. 30, n. 1, p. 29-42, 2004.

PICCO, G. P.; JULIEN, C.; MURPHY, A. L.; MUSOLESI, M.; ROMAN, G. C. **Software engineering for mobility: Reflecting on the past, peering into the future**. In: Proc. of the 34th ACM/IEEE International Conference on Software Engineering (ICSE) (New York, NY, USA: ACM), p. 13-28, 2014.

PRESSMAN, R. S. **Software engineering: A practitioner's approach**. 6th ed., McGraw-Hill, 2005.

PRETSCHNER, A.; PHILIPPS, J. **Methodological issues in model-based testing**. In: Model-Based Testing of Reactive Systems. Lecture Notes in Computer Science, p. 281-291, 2004.

PRETSCHNER, A.; PRENNINGER W.; WAGNER S.; KÜHNEL C.; BAUMGARTNER M.; SOSTAWA B.; ZÖLCH R.; STA T. **One evaluation of model-based testing and its automation**. In: Proc. of the 27th International Conference on Software Engineering (ICSE) (New York, NY, USA: ACM), p. 392-401, 2005.

PÜSCHEL, G.; SEIGER, R.; SCHLEGEL, T. **Test modeling for context-aware ubiquitous applications with feature Petri nets**. In: Proc. of the 2nd International Workshop on Model-based Interactive Ubiquitous Systems, p. 37-40, 2012.

RAVINDRANATH, L.; NATH, S.; PADHYE, J.; BALAKRISHNAN, H. **Automatic and scalable fault detection for mobile applications**. In: Proc. of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), p. 190-203, 2014.

RIDENE, Y.; BARBIER, F. **A model-driven approach for automating mobile applications testing**. In: Proc. of the 5th European Conference on Software Architecture (ECSA), p. 1-7, 2011.

ROBOLECTRIC. Robolectric - Test-drive your Android code. Disponível em: <http://robolectric.org/>. Acesso em: Novembro de 2015.

ROBOTIUM. Robotium - The world's leading Android test automation framework. Disponível em: <http://code.google.com/p/robotium>. Acesso em: Novembro de 2015.

RODRIGUES, E. M. **Plets: A product line of model-based testing tools**. Tese de doutoramento, PUCRS, Porto Alegre, RS, 2013. Disponível em: <http://repositorio.pucrs.br/dspace/handle/10923/5577>. Acesso em: Setembro de 2015.

SALVA, S.; ZAFIMIHARISOA, S. R. **Data vulnerability detection by security testing for Android applications**. Information Security for South Africa, p. 1-8, 2013.

SALVA, S.; ZAFIMIHARISOA, S. R.; LAURENCOT, P. **Intent security testing: An approach to testing the intent-based vulnerability of Android components**. In: Proc. of the 2013 International Conference on Security and Cryptography (SECRYPT), p. 355-362, 2013.

SAMIH, H.; LE GUEN, H.; BOGUSCH, R.; ACHER, M.; BAUDRY, B. **Deriving usage model variants for Model-Based Testing: An industrial case study**. In: Proc. of the 19th International Conference on Engineering of Complex Computer Systems (ICECCS), p. 77-80, 2014.

SATYANARAYANAN, M. **Fundamental challenges in mobile computing**. In: Proc. of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC), p. 1-7, 1996.

SCHWEIGHOFER T.; HERIČKO, M. **Approaches for test case generation from UML diagrams**. In: Proc. of the 3rd Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications (SQAMIA), Croatia, p.91-98, 2014.

SINHA, A.; SMIDTS, C. **HOTTest: A model-based test design technique for enhanced testing of domain-specific applications**. In: ACM Transactions on Software Engineering and Methodology (TOSEM), v. 15, n. 3, p. 242-278, 2006.

SQLITE. SQLite Home Page. Disponível em: <https://www.sqlite.org/>. Acesso em: Novembro de 2015.

TAKALA T.; KATARA M.; HARTY J. **Experiences of system-level model-based GUI testing of an Android application**. In: Proc. of the 4th IEEE International Conference on Software Testing, Verification, and Validation (ICST), p. 377-386, 2011.

TAO, C.; GAO, J. **Modeling mobile application test platform and environment: Testing criteria and complexity analysis**. In: Proc. of the Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing (JAIMACA), p. 28-33, 2014.

TRETMANS, G. J. **Conformance testing with labelled transition systems: Implementation relations and test generation**. Computer Networks and ISDN Systems, v. 29, n. 1, p. 49-79, 1996.

ULRICH, A. **Introducing Model-Based Testing techniques in industrial projects**. In: GI-Edition. Lecture Notes in Informatics (LNI), Proc. Bd. 106, p. 29-34, 2007.

UTTING, M.; LEGEARD, B. **Practical model-based testing: A tools approach**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006.

UTTING, M.; PRETSCHNER, A.; LEGEARD, B. **A taxonomy of model-based testing**. Relatório Técnico, Hamilton, New Zealand, 2006.

WANG, C. J.; LIU, M. **Generating test cases for EFSM with given fault models**. In: Proc. of 12th Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future (INFOCOM), vol. 2, p. 774-781, 1993.

WANG, S.; ALI, S.; YUE, T.; LIAAEN, M. **Using feature model to tupport Model-Based Testing of product lines: An industrial case study**. In: Proc. of the 13th International Conference of Software Quality (QSIC), p. 75-84, 2013.

WASSERMAN, A. I. **Software engineering issues for mobile application development**. In: Proc. of the FSE/SDP Workshop on Future of Software Engineering Research (FoSER) (New York, NY, USA: ACM), p. 397-400, 2010.

XU, J.; DING, X.; CHEN, G.; DRURY, J.; WANG, L.; LI, X. **A new method for automated GUI modeling of mobile applications**. In: Proc. of the 10th International Conference on Mobile and Ubiquitous Systems. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, v. 131. Springer, p. 688-693, 2014.

YANG, W.; PRASAD, M. R.; XIE, T. **A grey-box approach for automated GUI-model generation of mobile applications**. In: Proc. of the 16th International Conference on Fundamental Approaches to Software Engineering (FASE), p. 250-265, 2013.

YUAN, X.; COHEN, M. B.; MEMON, A. M. **GUI interaction testing: Incorporating event context**. In: Proc. of the IEEE Transactions on Software Engineering (TSE), v. 37, n. 4, p. 559-574, 2011.

ZAEEM, R. N.; PRASAD, M. R.; KHURSHID, S. **Automated generation of oracles for testing user-interaction features of mobile apps**. In: Proc. of the 7th IEEE International Conference on Software Testing, Verification and Validation (ICST), p. 183-192, 2014.

ZHU, H.; HALL, P. A. V.; MAY, J. H. R. **Software unit test coverage and adequacy**. ACM Computing Surveys (CSUR), v. 29, n. 4, p. 366-427, 1997.

APÊNDICE A - FORMULÁRIO DE EXTRAÇÃO DOS DADOS DO EMS

ID	1
Ano	2015
Nome do evento	Electronic Notes in Theoretical Computer Science
Tipo do evento [1]	Artigo de periódico
Título do trabalho	Evaluating the Model-Based Testing Approach in the Context of Mobile Applications
Autor(es)	FARTO, G. C.; ENDO, A. T.
Uso de TBM	Sim
Técnica de modelagem	ESG
Tipo de teste [2]	Funcional
Tipo de estudo [3]	Experimental
Plataforma de desenvolvimento móvel [4]	Android
Resumo original	The popularity of portable devices has grown rapidly in recent years. Due to the high number and diversity of users, new testing approaches are necessary to reduce the occurrence of faults and ensure better quality in mobile applications. The major objective of this paper is to evaluate the use of Model-Based Testing (MBT) in the construction and implementation of automated tests to verify and validate mobility solutions developed in the Google Android platform. The research proposal is guided by three questions: (Q1) - “Can the concepts of MBT be used in its current state to verify and validate functional requirements in mobile applications?”; (Q2) - “What are the results and challenges identified from adoption of MBT in mobile applications?”; and (Q3) - “How effective were the models and test cases generated, implemented and executed in the mobile application evaluated?”. The results obtained from an experimental evaluation are discussed and related to questions of this research.
Comentários	Farto e Endo (2015) avaliam uma abordagem de Teste Baseado em Modelo (TBM) com modelos elaborados em ESG para descrever requisitos funcionais de aplicações móveis desenvolvidas em Android. Um estudo experimental é realizado junto a alunos e desenvolvedores de uma empresa multinacional de TI. Resultados identificados evidenciam que TBM e ESG podem ser aplicados ao contexto de mobilidade, apoiando a geração automática de casos de teste e a concretização por meio do Robotium. A abordagem se mostrou efetiva na detecção de defeitos ao revelar quatro defeitos em uma aplicação Android selecionada. Desafios relatados direcionam esforços para trabalhos futuros, sugerindo pesquisas que explorem a geração parcial ou completa de casos de teste a partir das informações contidas no próprio projeto a ser testado.

[1] Conferência, *workshop*, periódico, livro e capítulo de livro;

[2] Acessibilidade, aleatório, baseado em defeitos, cobertura, combinatório, compatibilidade, conectividade, conformidade, desempenho, estrutural, funcional, integração, interface, regressão, segurança, *Testing-as-a-Service* (TaaS), unidade, usabilidade (e outros);

[3] Exploratório, experimental e industrial;

[4] Apple iOS, Blackberry, Google Android, JME, Microsoft Windows Phone e Symbian.

As características de pesquisas selecionadas na condução do mapeamento sistemático podem ser categorizadas em:

Identificador (ID). Definição de número sequencial que possibilita identificar unicamente cada pesquisa recuperada e selecionada;

Ano. Identificação do ano em que a pesquisa foi realizada ou divulgada na comunidade acadêmica;

Nome do evento e tipo do evento. Identificação do nome do evento e do tipo de pesquisa ou modalidade de divulgação do trabalho, classificando-a em (i) conferência, (ii) *workshop*, (iii) periódico, (iv) livro e (v) capítulo de livro;

Título do trabalho e autor(es). Identificação do título do trabalho e relação do(s) autor(es) que contribuíram com a realização;

Uso de TBM. Verificação do uso de um processo tradicional e/ou adaptado de TBM no contexto de aplicações móveis;

Abordagem para modelagem ou execução do teste. Classificação das abordagens propostas ou utilizadas para modelar eventos e comportamentos do software, fluxos de dados e representações da interação com o usuário das aplicações móveis em teste (quando aplicável);

Tipo e/ou técnica de teste. Classificação dos tipos e técnicas de teste propostos ou utilizados para verificar as aplicações móveis como, por exemplo, estrutural, funcional, interface, usabilidade, regressão e outros (quando aplicável);

Tipo de estudo. Identificação do tipo de estudo conduzindo, classificando-o em (i) exploratório, (ii) experimental e (iii) industrial;

Plataforma de desenvolvimento móvel. Identificação das plataformas de desenvolvimento móvel, classificando-as em (i) *Android*, (ii) *Apple iOS*, (iii) *Microsoft Windows Phone*, (iv) *Blackberry*, (v) *Java Micro Edition* (JME) e (vi) *Symbian*.

APÊNDICE B - CLASSES E MÉTODOS DA PLATAFORMA ROBOTIUM

Classes e métodos da plataforma *Robotium*

Instrução	**Descrição**
ActivityInstrumentationTestCase2	Classe que provê funcionalidades nativas para testes em Android
com.robotium.solo.Solo	Classe que provê os métodos utilizados na interação e interface entre projeto de teste e projeto a ser testado
solo.sendKey(int)	Método que envia ou simula um evento de tecla
solo.goBack()	Método que simula o evento do botão “Voltar”
solo.clickInList(int)	Método que simula o toque em um componente de lista (ListView)
solo.getString(int)	Método que retorna uma String a partir da constante que a representa
solo.clickOnMenuItem(String)	Método que simula a seleção de um item de menu
solo.clickOnButton(String)	Método que simula o toque em um botão
solo.clearEditText(int)	Método que limpa o conteúdo de um campo de texto
solo.enterText(String)	Método que atribui uma String a um campo de texto
solo.finishOpenedActivities()	Método que finaliza as telas em exibição

APÊNDICE C - FRAGMENTO DE CLASSE COM CASOS DE TESTE CONCRETIZADOS COM ROBOTIUM

```
public class AddressBookTestCase extends ActivityInstrumentationTestCase2 {

    private Solo solo;

    @SuppressWarnings("unchecked")
    public AddressBookTestCase() {
        super(AddressBook.class);
    }

    @Override
    public void setUp() throws Exception {
        this.solo = new Solo(getInstrumentation(), getActivity());
    }

    public void testCesOne() throws Exception {
        String cesOne = "[, Show_Main_Screen, Press_Menu, Press_Back, Show_Main_Screen,
          Select_Add_Contact, Inform_Contact_Data_R1, Press_Save_Contact, Show_Main_Screen,
          Select_Contact, Press_Back, Show_Main_Screen, Select_Contact, Display_Contact,
          Press_Menu, Press_Back, Display_Contact, Press_Menu, Select_Edit_Contact,
          Inform_Contact_Data_R2, Display_Error_Message, Inform_Contact_Data_R2,
          Press_Save_Contact, ]";

        TestCaseUtil.executeTestCase(this, cesOne);
    }

    public void testCesTwo() throws Exception { // The full CES have been omitted
        String cesTwo = "[, Show_Main_Screen, Press_Menu, ... , ]"

        TestCaseUtil.executeTestCase(this, cesTwo);
    }

    public void testCesThree() throws Exception { // The full CES have been omitted
        String cesThree = "[, Show_Main_Screen, Select_Contact, Display_Contact, ... , ]"

        TestCaseUtil.executeTestCase(this, cesThree);
    }

    public void Press_Menu() throws Exception {
        this.solo.sendKey(KeyEvent.KEYCODE_MENU);
    }

    public void Press_Back() throws Exception {
        this.solo.goBack();
    }

    public void Select_Contact() throws Exception {
        this.solo.clickInList(0);
    }

    public void Select_Add_Contact() throws Exception {
        this.solo.clickOnMenuItem(this.solo.getString(R.string.menuitem_add_contact));
    }

    public void Select_Edit_Contact() throws Exception {
        this.solo.clickOnMenuItem(this.solo.getString(R.string.menuitem_edit_contact));
    }

    public void Press_Save_Contact() throws Exception {
        this.solo.clickOnButton(this.solo.getString(R.string.button_save_contact));
    }

    public void Inform_Contact_Data_R1() throws Exception {
        for (int i = 0; i < 5; i++) { this.solo.clearEditText(i); }

        this.solo.enterText(0, "Guilherme");
        this.solo.enterText(1, "4321-1234");
        this.solo.enterText(2, "guilherme@gmail.com");
        this.solo.enterText(3, "Av. Faria Lima, 100");
        this.solo.enterText(4, "São Paulo, SP, Brasil");
    }

    @Override
    public void tearDown() throws Exception {
        this.solo.finishOpenedActivities();
    }
}
```

APÊNDICE D - ESTEREÓTIPOS PARA AS ESTRATÉGIAS METHOD TEMPLATE E EDGE TEMPLATE

Estereótipos para as estratégias *Method Template* e *Edge Template*

Estereótipo ID	**Estratégias de reúso de modelos de teste**				
	Method Template	***Edge Template***			
		Eventos específicos de dispositivos	**Interação não previsível de usuários**	**Eventos de telefonia para GSM/SMS**	**Eventos de sensores e componentes de hardware**
ChangeProgressBar	✓				
ClearEditText	✓				
Drag	✓				
FillEditText	✓				
PressBack	✓				
PressButton	✓				
PressCheckBox	✓				
PressImageButton	✓				
PressMenu	✓				
PressMenuItem	✓				
PressRadioButton	✓				
PressToggleButton	✓				
SelectListItem	✓				
SelectSpinnerItem	✓				
TakeScreenshot	✓				
UnlockScreen	✓				
ChangeOrientationToLandscape		✓			
ChangeOrientationToPortrait		✓			
ChangeMobileData		✓			
ChangeWiFiData		✓			

ChangeAccelerationData					✓
ChangeGPSData					✓
DisableBluetooth					✓
EnableBluetooth					✓
GoBackGoForward			✓		
ReceiveCallAndAccept				✓	
ReceiveCallAndCancel				✓	
ReceiveSMS				✓	
Sleep			✓		
SleepFor30Seconds			✓		
UpdateCoordinates					✓

APÊNDICE E - CLASSE COM MÉTODOS DE TESTE GERADOS A PARTIR DAS CESs OBTIDAS POR MBTS4MA

```
package com.deitel.routetracker.test;

public class RouteTrackerTest extends ActivityInstrumentationTestCase2 {

    private Solo solo;

    private boolean actualTest = false;

    @SuppressWarnings("unchecked")
    public RouteTrackerTest() {
        super(RouteTracker.class);
    }

    @Override
    public void setUp() throws Exception {
        this.solo = new Solo(getInstrumentation(), getActivity());
    }

    public void testCES1() {
        String ces = "displayTrackingScreen, pressMenu, displayOptions, pressBack,
                      displayTrackingScreen, pressStartTracking, listenToGps,
                      readUpdatedLocation, updateScreenMap, listenToGps, pressStopTracking,
                      showDistanceAndAvgSpeed, pressOk, displayTrackingScreen,
                      receiveCallAndAccept, pressBack";

        new EventRunner().executeCompleteEventSequence(
            new RouteTrackerAdapter(this.solo), ces);
    }

    public void testCES2() {
        String ces = "displayTrackingScreen, changeWiFiData, pressStartTracking,
                      listenToGps, pressStopTracking, showDistanceAndAvgSpeed, pressOk,
                      displayTrackingScreen, pressBack";

        new EventRunner().executeCompleteEventSequence(
            new RouteTrackerAdapter(this.solo), ces);
    }

    public void testCES3() {
        String ces = "displayTrackingScreen, pressMenu, displayOptions, selectMap,
                      displayMap, displayTrackingScreen, pressBack";

        new EventRunner().executeCompleteEventSequence(
            new RouteTrackerAdapter(this.solo), ces);
    }

    public void testCES4() {
        String ces = "displayTrackingScreen, pressMenu, displayOptions, selectSatellite,
                      displaySatellite, displayTrackingScreen, pressBack";

        new EventRunner().executeCompleteEventSequence(
            new RouteTrackerAdapter(this.solo), ces);
    }

    public void testCES5() {
        String ces = "displayTrackingScreen, pressMenu, displayOptions, pressBack,
                      displayTrackingScreen, pressStartTracking, listenToGps,
                      sleepFor30Seconds, pressStopTracking, showDistanceAndAvgSpeed,
                      pressOk, displayTrackingScreen, pressBack";

        new EventRunner().executeCompleteEventSequence(
            new RouteTrackerAdapter(this.solo), ces);
    }

    @Override
    public void tearDown() throws Exception {
        this.solo.finishOpenedActivities();
    }
}
```

APÊNDICE F - CLASSE COM ADAPTADORES PARCIALMENTE CONCRETIZADOS POR MBTS4MA

```
package com.deitel.routetracker.test.adapters;

import java.awt.event.KeyEvent;

public class RouteTrackerAdapter {

    private Solo solo = null;

    public RouteTrackerAdapter(Solo solo) {
        this.solo = solo;
    }

    @Event(label = "displayTrackingScreen")
    public boolean displayTrackingScreen() {

    }

    @Event(label = "pressMenu")
    public boolean pressMenu() {
        new Thread(new Runnable() {
            public void run() {
                getInstrumentation().sendKeyDownUpSync(KeyEvent.KEYCODE_MENU);
            }
        }).start();
    }

    @Event(label = "displayOptions")
    public boolean displayOptions() {

    }

    @Event(label = "pressBack")
    public boolean pressBack() {
        this.solo.goBack();
    }

    @Event(label = "pressStartTracking")
    public boolean pressStartTracking() {
        this.solo.clickOnButton(this.solo.getString(R.string.button_start_tracking));
    }

    @Event(label = "listenToGps")
    public boolean listenToGps() {

    }

    @Event(label = "readUpdatedLocation")
    public boolean readUpdatedLocation() {
        TestLocationProvider.sendLocation(this.solo,
            new double[] { 43.723180, 44.697660 }, new double[] { 10.396677, 10.631054 });
    }

    @Event(label = "updateScreenMap")
    public boolean updateScreenMap() {

    }

    @Event(label = "pressStopTracking")
    public boolean pressStopTracking() {
        this.solo.clickOnButton(this.solo.getString(R.string.button_stop_tracking));
    }

    @Event(label = "showDistanceAndAvgSpeed")
    public boolean showDistanceAndAvgSpeed() {

    }

    @Event(label = "pressOk")
    public boolean pressOk() {
        this.solo.clickOnButton(this.solo.getString(R.string.button_ok));
    }
```

```
    @Event(label = "receiveCallAndAccept")
    public boolean receiveCallAndAccept() {
        TestPhoneCallEvent.phoneCallAction(PhoneCallActions.call, "12349876");

        this.solo.sleep(2000);

        TestPhoneCallEvent.phoneCallAction(PhoneCallActions.accept, "12349876");

        this.solo.sleep(1000);

        TestPhoneCallEvent.phoneCallAction(PhoneCallActions.cancel, "12349876");

        this.solo.sleep(5000);

        TestPhoneCallEvent.goBack();
    }

    @Event(label = "pressBack")
    public boolean pressBack() {
        this.solo.goBack();
    }

    @Event(label = "changeWiFiData")
    public boolean changeWiFiData() {
        this.solo.setWiFiData(false);
    }

    @Event(label = "selectMap")
    public boolean selectMap() {
        this.solo.clickOnMenuItem(this.solo.getString(R.string.menuitem_map));
    }

    @Event(label = "displayMap")
    public boolean displayMap() {

    }

    @Event(label = "selectSatellite")
    public boolean selectSatellite() {
        this.solo.clickOnMenuItem(this.solo.getString(R.string.menuitem_satellite));
    }

    @Event(label = "displaySatellite")
    public boolean displaySatellite() {

    }

    @Event(label = "sleepFor30Seconds")
    public boolean sleepFor30Seconds() {
        this.solo.sleep(30000);
    }
}
```

APÊNDICE G - FORMULÁRIO DE ENTREVISTA 1

Formulário de Entrevista 1

Nome (opcional): _____

Empresa: _____ Equipe/Célula: _____

Atua na empresa há: [ ] < 2 anos

[ ] >= 2 anos e < 5 anos [ ] >= 5 anos e < 10 anos

[ ] >= 10 anos

1. Avalie seus conhecimentos e marque com um X o valor que considera como mais apropriado:

Área	Muito baixo	Baixo	Médio	Alto	Muito alto
Java					
Google Android					
Teste de Software					
Teste Baseado em Modelo (TBM)					

2. Como você executa os testes em aplicações móveis (marque com um X):

Atividade de teste	Nenhuma vez	Não muitas vezes	Algumas vezes	Na maioria das vezes	Sempre
Teste manuais					
Testes unitários					
Processos automatizados					

3. Os desenvolvimentos passam por atividades de teste formais? Se sim, descreva os processos, passos e artefatos.

[ ] Sim [ ] Não

Descreva: _____ ...

4. É possível obter um histórico acerca das atividades de testes de cada aplicação móvel testada? Se sim, descreva as informações que podem ser coletadas pelo histórico.

[ ] Sim [ ] Não

Descreva: _____ ...

5. Você considera a metodologia de testes atual como:

[ ] Eficiente (as atividades são feitas corretamente)

[ ] Eficaz (as atividades feitas são corretas)

[ ] Rápido/Ágil [ ] Repetível [ ] Rastreável [ ] Confiável [ ] Escalável

Justifique: _____ ...

6. Além de testar as funcionalidades implementadas e/ou modificadas, outros fatores ou aspectos são verificados?

[ ] Sim [ ] Não

Descreva: _____ ...

7. Dê uma nota para o controle e gestão de testes em aplicações móveis:

[ ] 0-2 [ ] 2-4 [ ] 4-6 [ ] 6-8 [ ] 8-10

APÊNDICE H - FORMULÁRIO DE ENTREVISTA 2

Formulário de Entrevista 2

Nome (opcional): _____

Empresa: _____ Equipe/Célula: _____

Atua na empresa há: [ ] < 2 anos

[ ] >= 2 anos e < 5 anos [ ] >= 5 anos e < 10 anos

[ ] >= 10 anos

1. Descreva as informações da modelagem realizada:

Nome da aplicação móvel/nome do módulo: _____ ...

Funcionalidade modelada: _____ ...

2. Avalie a complexidade identificada ao realizar o experimento:

Área	Nenhuma dificuldade	Pouca dificuldade	Dificuldade moderada	Alta dificuldade
Conceitos de TBM e ESG				
Conceitos da abordagem de reúso de TBM				
Uso da ferramenta de apoio MBTS4MA				

3. Avalie o que você julga importante para o teste de aplicações móveis:

Tópico	Nota 0 - pouca importância 10 - máxima importância				
	0-2	2-4	4-6	6-8	8-10
Facilidade na criação de casos de teste					
Uso de modelos e submodelos de teste					
Uso de informações da aplicação móvel em modelos de teste					
Repetibilidade (reexecução) dos testes					
Rastreabilidade dos casos de teste					
Manutenibilidade dos casos de teste					
Relatório de informações pós-teste					

4. O que pode ser melhorado na proposta de modelagem com MBT e ESG?

Descreva: _____ ...

5. O que pode ser melhorado na ferramenta de apoio MBTSM4?

Descreva: _____ ...