UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE GRADUAÇÃO E EDUCAÇÃO PROFISSIONAL
ENGENHARIA DE COMPUTAÇÃO

ALINE FERREIRA DA SILVA

UM ESTUDO EXPERIMENTAL SOBRE O USO DO TESTE BASEADO EM MODELO NA DETECÇÃO DE DEFEITOS EM REQUISITOS DE SOFTWARE

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2015

ALINE FERREIRA DA SILVA

UM ESTUDO EXPERIMENTAL SOBRE O USO DO TESTE BASEADO EM MODELO NA DETECÇÃO DE DEFEITOS EM REQUISITOS DE SOFTWARE

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso 2, do curso de Engenharia de Computação, Departamento de Computação – DACOM, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Dr. André Takeshi Endo

CORNÉLIO PROCÓPIO

2015

Dedico este trabalho a minha família, em especial aos meus pais, que sempre incentivaram o estudo em minha vida, ao meu noivo, Allison, que deu todo o apoio necessário nesta fase e a todos meus amigos da faculdade que me acompanharam nesses cinco anos de estudo.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus por tornar real todas as grandes conquistas que tive até hoje em minha vida, como a realização deste trabalho que contribui para a minha formação profissional.

Agradeço ao meu orientador Prof. Dr. André Takeshi Endo, por repassar seu conhecimento e sabedoria, os quais ajudaram muito a fazer este trabalho, também agradeço por sua imensa dedicação, esforço e comprometimento nesta orientação.

A minha família, principalmente aos meus pais, Ednilson e Edvâna, que sempre me apoiou nos momentos mais difíceis, incentivando a continuar e buscar cada vez mais por conquistas gloriosas em minha vida.

Ao meu noivo, Allison, que deu o apoio suficiente para seguir em frente e vencer mais este desafio.

Aos meus grandes amigos da faculdade: Bruno, Marcelo, Felipe Bianca, Ana Beatriz, Marcos e Rhuan, que me acompanharam desde o início da faculdade e que também contribuíram com minha formação acadêmica com as inéditas horas de estudos. Importante ressaltar a grande ajuda de Bruno e Mariana que ajudaram a validar uma das etapas deste trabalho

Também quero registrar minha gratidão a todos os professores, coordenadores e demais funcionário da Universidade Tecnológica Federal do Paraná de Cornélio Procópio que contribuíram para minha formação e realização desta pesquisa.

RESUMO

SILVA, Aline Ferreira da. **Um Estudo Experimental sobre o Uso do Teste Baseado em Modelo na Detecção de Defeitos em Requisitos de Software.** Trabalho de Conclusão de Curso (Graduação) – Engenharia de Computação. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2015.

Teste Baseado em Modelo é uma técnica que permite automatizar casos de teste para um determinado software seguindo as exigências de um modelo de teste. Portanto este trabalho apresenta uma experiência com estudantes da área de engenharia de software para verificar se o Teste Baseado em Modelo (TBM) pode auxiliar na inspeção de requisitos durante o projeto do modelo de teste. Este experimento possui o objetivo de verificar se há a detecção de defeitos no documento de requisitos, durante a etapa de modelagem do TBM, utilizando a técnica *Event Sequence Graphs* para elaborar o modelo de teste. Foi verificado que é possível encontrar defeitos no documento de requisitos, principalmente os defeitos de digitação, omissão e inconsistência.

Palavras-chave: Teste Baseado em Modelo. Inspeção de Requisitos. Grafo de Sequência de Eventos.

ABSTRACT

SILVA, Aline Ferreira da. **An Experimental Study on the Use of Model-Based Testing for Fault Detection in Software Requeriments**. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Software. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2015.

Model-Based Testing is a technique that allows to automate test cases for a particular software following the requirements of a test model. Therefore, this work presents an experiment with software engineering students to verify whether Model-Based Testing can support the requirement inspection during test model design. This experiment has the objective of verifying the faults detection in the requirements document during the modeling stage, using the Event Sequence Graphs technique to design the test model. We found that it is possible to find faults in the requirements document, especially typing, omission and inconsistency faults.

Keywords: Model-Based Testing. Requirement Inspection. Event Sequence Graphs.

SUMÁRIO

1 INTRODUÇÃO

O uso da tecnologia para armazenar informações e satisfazer as necessidades humanas vem crescendo consideravelmente. Diante da demanda crescente, a área de tecnologia da informação emprega 1,3 milhão de profissionais no país, e segundo a Associação Brasileira de Empresas de Tecnologia da Informação e Comunicação (Brasscom), é previsto que em 2016 este número cresça 30% (LAZZAROTTO, 2013). Dessa forma, é perceptível que o desenvolvimento de software está se tornando cada vez mais uma necessidade.

O desenvolvimento de software não é algo tão simples de ser feito, pois para um software se tornar um produto final deve passar por vária etapas e aprovações que consiga obter o melhor desempenho e qualidade possível. Em suma, há uma etapa de especificação dos requisitos de software, etapa de construção, validação, manutenção e assim por diante, de forma que todas tenham sua importância e colaboração na construção do produto.

Diante disso, é importante ressaltar a importância de requisitos de software, o qual é um tema abordado na área de engenharia de software. Os requisitos de um software consistem em características operacionais, interface e outros elementos do sistema, até mesmo as restrições que o software deve satisfazer (PRESSMAN, 2006). Em outras palavras, são critérios necessários que são capazes de satisfazer as necessidades declaradas e os objetivos do software.

Uma atividade relacionada à especificação de requisitos do software é o processo de inspeção, o qual é considerado um dos processos mais eficientes para detectar defeitos em diversos subprodutos, principalmente no documento de requisitos (LANUBILE, 1996; BIFFL, 2001; WONG, 2002). Este processo também colabora para que defeitos sejam encontrados antes mesmo da construção do software, reduzindo futuros custos com manutenção e ajudando a adquirir mais qualidade e eficiência.

Outra atividade importante relacionada ao desenvolvimento de um software é o processo de teste, que é de suma importância para verificar se os requisitos do software foram implementados da maneira correta e esperada, para assim, o usuário conseguir satisfazer suas necessidades. Uma das técnicas utilizadas no processo de teste é o Teste Baseado em Modelo (TBM). O TBM é uma técnica na qual casos de teste são formados com base em um modelo criado com o objetivo de descrever o

software, este que irá ser testado (ENDO, 2010). Para construção de modelos de teste existem várias técnicas; uma delas é o *Event Sequence Graphs* (ESG), o qual é composto por gráficos de sequência de eventos que descrevem características específicas do sistema, composto por nós, que indicam os eventos possíveis do software, e arestas, as quais representam a sequência dos eventos (BELLI et al., 2014; BELLI et al.,2006b).

1.1 MOTIVAÇÃO

O desenvolvimento de um software consiste em várias fases que seguem alguns padrões, e que estes definem um conjunto de atividades, tarefas, produtos, ações necessárias para que ocorra de fato o desenvolvimento do software (PRESSMAN, 2006). Uma das fases mais importante que o software passa é a atividade de teste. A atividade de teste consiste em testar todos os pontos e características cruciais que o software deve atender e assim verificar se o desenvolvimento de software está sendo feita de maneira correta e se irá satisfazer as necessidades do usuário (ENDO, 2010).

Assim, uma das formas de testar um software é utilizando a técnica do TBM, o qual consiste em que o testador elabore um modelo de teste com base nos requisitos do software que irá ser testado, e posteriormente possibilite a automatização dos casos de testes, de forma que este processo possa obter mais qualidade e ser menos propenso a erros humanos (UTTING e LEGEARD, 2006).

Diante disso, este trabalho está focado em uma das etapas do TBM, a modelagem. A etapa de modelagem requer que o testador construa o modelo de teste e que este possua detalhes suficientes para descrever exatamente o que será testado (ENDO, 2010). Dessa forma, quanto maior for o entendimento do testador sobre as funcionalidades que o software deve possuir, maior será a chance de construir um modelo de teste e casos de teste mais eficientes e que gerem bons resultados.

A etapa de modelagem do TBM pode ir um pouco além de só compreender requisitos do software e elaborar o modelo de teste, pois ao realizar este processo de teste pode ocorrer que o testador encontre defeitos nos requisitos do software (BLACKBURN et al., 2004; UTTING e LEGEARD, 2006), principalmente em estágios iniciais da modelagem. Esta atividade de detecção pertence ao objetivo do processo de inspeção de requisitos, o qual fiscaliza o documento de requisitos do software a fim

de encontrar defeitos e poder removê-los, contribuindo com a qualidade e a redução de custos do software.

Dessa forma, é de grande valia conseguir verificar e validar se o TBM além de estar relacionada com a execução de teste de um software, também possibilita realizar a inspeção de requisitos de um software, pois segundo Pressman (2005), detectar defeitos em estágios iniciais tem sua importância em promover aumento de qualidade e redução de custos, contribuindo com o desenvolvimento de software.

1.2 OBJETIVOS

O principal objetivo deste trabalho é verificar se a seguinte hipótese é válida: construir o modelo de teste durante o TBM possibilita realizar a inspeção de requisitos do software, detectando defeitos existentes no documento de requisitos do software.

1.3 ORGANIZAÇÃO TEXTUAL

O desenvolvimento deste trabalho foi dividido em cinco capítulos. O Capítulo 2 consiste na fundamentação teórica, a qual é subdividida em revisão bibliográfica e trabalhos relacionados, sendo a revisão bibliográfica subdivida nos principais assuntos deste trabalho, inspeção de requisitos, TBM e ESG, os quais serão mais detalhados. Já o Capítulo 3 consiste em apresentar o método realizado para chegar ao objetivo deste trabalho e as ameaças à validade; o Capítulo 4 apresenta os resultados encontrados a partir do que foi realizado no Capítulo 3 e suas análises e discussões. Por fim o Capítulo 5 expõe as conclusões deste trabalho a partir dos resultados encontrados e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentado todo embasamento teórico que está relacionado com o desígnio deste trabalho, com o objetivo de auxiliar e facilitar o entendimento do leitor sobre a mesma. Além disso, como contribuição, também serão apresentados alguns trabalhos relacionados com este trabalho.

2.1 REVISÃO BIBLIOGRÁFICA

Nesta seção, estão contidos os principais enfoques deste trabalho, sendo assim dividida nas seguintes subseções: Inspeção de Requisitos, TBM e ESG.

2.1.1 Inspeção de Requisitos

A palavra inspeção está relacionada com o intuito de fiscalizar, observar ou vistoriar algo que está sendo alvo de estudos. Desta forma, quando se trata de inspeção de requisitos na área de engenharia de software, o alvo de fiscalização, observação ou vistoria é os requisitos do software. Segundo Nuseibeh (2002) o processo de inspeção de um software consiste na subdivisão de uma tarefa em várias outras, as quais estão relacionadas com a transformação das informações de entrada em informações de saída, de forma organizadas e que possam ser repetidas. É importante ressaltar que inspeção de requisitos é apenas um dos tipos de inspeção que pode ser feito em um software, pois também podem ser inspecionadas outros tipos de documento do mesmo, tais como: documentos de projeto, testes, código-fonte, entre outros.

O documento de requisitos de um software é um material de extrema importância para o desenvolvimento e entendimento de um software, pois contém todas suas características e objetivos gerais e específicos. Assim, se um documento de requisitos possui erros ou informações incorretas, isto poderá comprometer a qualidade e finalidade do produto, por isso que inspecionar o documento de requisitos pode trazer muitos benefícios, este que são apresentados por Fagan (1986) e Pressman (1995) a seguir:

- Melhoria na qualidade do produto de software;
- Aumento na produtividade no período de desenvolvimento do software;

- Detecção de defeitos que previnem problemas futuros no software;
- Verificação do cumprimento dos requisitos do software;
- Contribuição para projetos mais administráveis.

Este processo realizado em documentos de software é composto por um ciclo contendo três tarefas, conforme é mostrado na Figura 1. A primeira tarefa consiste em uma leitura individual do documento que está sendo fiscalizado pelos avaliadores de inspeção, seguida por uma reunião entre eles com o objetivo de discutirem as falhas encontradas por cada indivíduo e encontrarem mais falhas no documento. Depois da reunião, a tarefa de remover as falhas encontradas é realizada, e assim inicia-se o ciclo novamente, e esse é feito até que a qualidade desejada for adquirida (LAITENBERGER, 1995; ABIB, 1998).

Figura 1: Processo de Inspeção de Documentos de Software
Fonte: Adaptado de BERTINI (2006).

A etapa de Leitura é muito importante para que cada participante possa encontrar defeitos no documento fiscalizado. E para isso, existem algumas técnicas de leitura que podem auxiliar neste processo, tais como: Leitura *Ad-Hoc*, a qual é uma técnica muita utilizada onde os inspetores não recebem nenhuma orientação, fazendo que muitas vezes descobrir um defeito seja questão de sorte ou experiência de cada inspetor (BERTINI, 2006). Leitura Baseada em *Checklists* é uma técnica onde os inspetores recebem uma orientação, a qual é uma lista de perguntas sobre o produto que está sendo fiscalizado, e estes devem respondê-la com sim ou não de acordo com o que o documento relata (CIOLKOWSKI, 1999).

Outra técnica também é a de Leitura Baseada em Cenários que possui o objetivo de atribuir especialidades específicas para cada inspetor, assim o inspetor recebe um cenário, que pode ser composto por *checklists*, porém *checklists* especializados de uma área específica do inspetor; dessa forma, este tipo de técnica possibilita que todo inspetor possua um cenário e uma especialidade diferente, tornando assim a equipe de inspeção mais eficiente (BERTINI, 2006). Leitura Baseada em Perspectivas (LBPe), como o próprio nome já diz, leva em consideração a perspectiva de diferentes consumidores ou *stakeholders*, os quais se interessam em

diferentes fatores de qualidade, ou veem de forma diferente o mesmo fator de qualidade (CIOLKOWSKI, 1999), sendo que estes *stakeholders* podem ser o usuário do sistema, o projetista ou o responsável pelos testes do software.

Para entender de forma mais clara sobre que tipos de defeitos podem ser encontrados em um documento de requisitos de um software, Basili (1981) expõe sete categorias de defeitos: defeitos de digitação, ambiguidade, omissão de informação, inconsistência, fato incorreto, informações escritas em seções erradas e fato de implementação não fornecido.

A Figura 1, apresentada anteriormente, ilustra as etapas do processo de inspeção de documentos de software, mas o processo de inspeção pode ser refinado além dessas três etapas, como mostra a Figura 2.

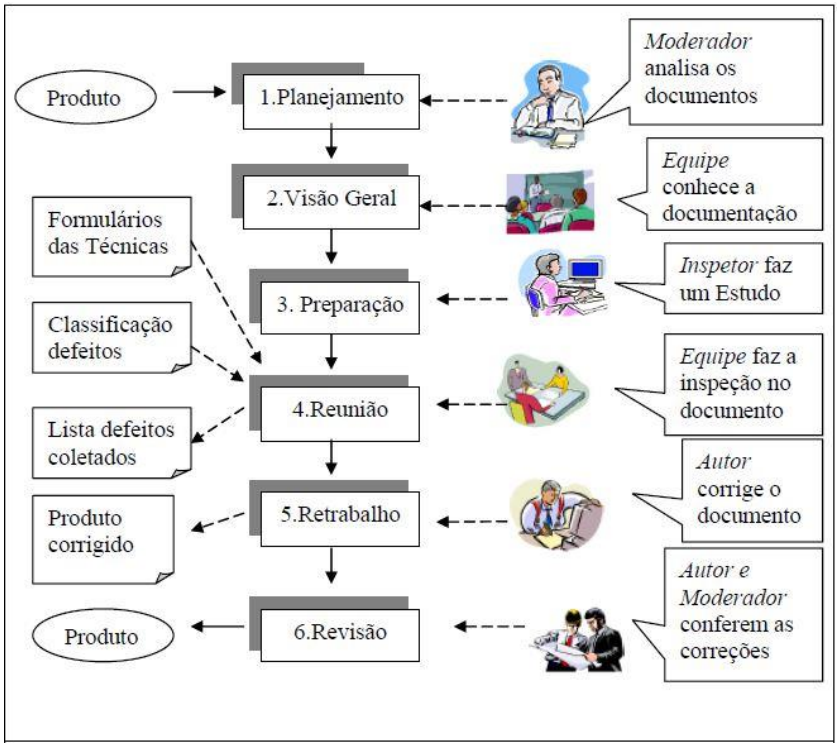


Figura 2: Fases do Processo de Inspeção
FONTE: Extraído de Bertini (2006).

De acordo com Fagan (1986), Ebenau (1994), Abib (1998), Tyran (2002) e Wong (2002) e conforme ilustrado na Figura 2, a primeira etapa do processo de inspeção é dada pelo Planejamento, o qual organiza todos os documentos necessários que serão inspecionados, bem como os avaliadores e o local que será realizado este processo. A segunda etapa, Visão Geral, consiste em apresentar e expor os materiais selecionados aos avaliadores e suas respectivas funções. Na

próxima etapa, Preparação, ocorre o treinamento dos avaliadores para executarem a inspeção. Na etapa de Reunião ou Realização da Inspeção, como o próprio nome já diz, a inspeção é realizada por cada avaliador com o objetivo destes encontrarem defeitos ou falhas nos materiais selecionados. Como penúltima etapa, o Retrabalho, todos os defeitos encontrados pelos avaliadores que foram documentados são repassados para o responsável do produto de inspeção para que este tome suas providências. E por fim, é realizada a etapa de Revisão, onde o responsável do produto e os avaliadores revisam o produto inspecionado para verificar se todas as correções foram realizadas e que nenhuma outra falha foi inserida.

Até agora é possível observar que o processo de inspeção pode ser muito importante para a obtenção de qualidade no produto e também pode reduzir custos no projeto pelo fato de possibilitar a verificação de defeitos antes mesmo do processo de desenvolvimento do produto ser iniciado. Mas para este processo conseguir bons resultados é necessário que se tome atenção em alguns aspectos, por exemplo, os participantes do processo de inspeção devem ser bem treinados (FAGAN, 1986), devem entender a técnica de inspeção que será utilizada e todos devem assumir suas respectivas funções (LANUBILE, 1996); a duração do processo de inspeção deve ser bem estudado para que não seja insuficiente e assim provocar uma má inspeção (LANUBILE, 1996), entre outras considerações.

É importante também ressaltar que a técnica utilizada para desenvolver o processo de inspeção também pode afetar nos resultados. Basili (1996) desenvolveu um experimento na Universidade de Maryland, a fim de comparar a técnica LBPe com uma técnica não-sistemática utilizada na NASA. Este tipo de experimento foi replicado por Ciolkowski (1997) na Universidade de *Kaiserslautem*, a fim de comparar a técnica LBPe com uma técnica *Ad Hoc*. Como resultado, o LBPe mostrou mais eficiência no processo de inspeção.

2.1.2 TBM

A fase de testes de um software é crucial para verificar se o mesmo cumpre com todos os seus requisitos esperados e também para garantir qualidade. Para isso existem técnicas que auxiliam na geração de casos de teste, e outros ainda que possuem o objetivo de automatizar esses casos de teste, tornando o processo de teste mais sistemático e formal (ENDO, 2010). TBM é uma técnica que possui a principal

finalidade de automatizar casos de teste de um determinado software, tendo como entrada as informações dos requisitos do software e como saída os resultados dos testes automatizados, para então assim verificar sua acurácia (ENDO, 2010). Segundo Sinha e Smidts (2006) o teste de software pode ser automatizado pela geração de casos de testes com base no modelo comportamental do sistema, o modelo de teste, tudo isso também conhecido como TBM.

Este modelo de teste é construído com base as informações do software que irá ser testado, podendo utilizar os documentos do software como fontes ou até mesmo o conhecimento específico que o profissional possui sobre o sistema. O modelo de teste além de ser essencial para o TBM, também auxilia no próprio entendimento do sistema (BARNETT et al., 2003), podendo apontar alguns defeitos encontrados nas primeiras fases do desenvolvimento do software, tendo como consequência a redução de custos futuros e aumento de qualidade do produto (ENDO 2010).

O TBM é uma técnica que ajuda a geração casos de teste ser mais rápida e menos propensa a erros humanos, já que os casos de testes serão automatizados (UTTING et al., 2006). E graças a isso, o processo de teste do software pode se tornar repetível e menos dependente de interpretação humana (SINHA e SMIDTS, 2006).

Existem algumas técnicas que auxiliam na criação do modelo de teste que podem agregar desde características simples até aquelas mais complexas. Hartman et al. (2007) apresentam três tipos de técnicas de modelagem de teste, as quais são especificadas a seguir:

- Técnicas de Modelagem de *Design* Genéricas: um exemplo muito utilizado deste tipo de técnica é a *Unified Modeling Language* (UML), que possui seu foco voltado ao *design* do software.
- Técnicas de Modelagem Específicas de Teste: possui como principal objetivo ajudar na modelagem de teste, onde o profissional da área precisa entender apenas sobre técnicas que possuem este propósito. Um tipo de técnica de modelagem específica de teste é o ESG, o qual será mais detalhado na Subseção 2.1.3.
- Técnicas de Modelagem Específicas de Domínio: são técnicas que possuem especificidade em um domínio em particular.

O TBM é formado por sete etapas, as quais são ilustradas na Figura 3 de acordo com os autores El-Far e Whittaker (2001), Pretschner e Philipps (2004), Utting e Legeard (2006) e Bouquet et al. (2006).

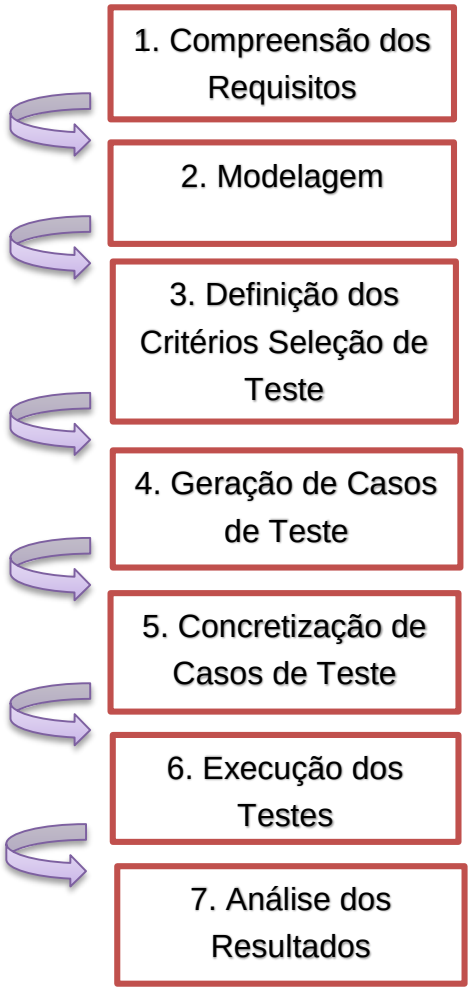


Figura 3: Fases do TBM
Fonte: Adaptado de ENDO (2010).

Na primeira fase do TBM, Compreensão dos Requisitos, o testador possui a tarefa de entender os requisitos do sistema que irá ser testado, para poder verificar os pontos cruciais que devem ser testados. A segunda fase, Modelagem, nada mais é que a criação do modelo de teste, este que não deve ser muito extenso e deve conter de forma clara e detalhada os pontos cruciais que deverão ser testados. A próxima fase, Definição de Critérios de Seleção de Teste, possui o objetivo de definir os casos de testes que serão gerados. A quarta e quinta fases, Geração e Concretização de Casos de Teste, estão relacionados a etapa de automatização dos casos de teste, de forma que estes se tornem executáveis, fazendo que o processo de teste se torne sistemático. Na penúltima fase, Execução dos Testes, como o próprio nome já diz, é onde ocorrem os testes do sistema. E por último, Análise dos Resultados, é onde os resultados da etapa anterior são analisados, e então pode ser verificado se há falhas ou não para serem corrigidas. Assim, como resultado, podem

haver três estados: passou, falhou ou inconclusivo. O primeiro indica que o teste foi realizado com sucesso, o estado de falhou indica que algo inesperado aconteceu no processo de teste, o qual não está de acordo com os objetivos do sistema, e o estado de inconclusivo indica que não ocorreu nada de inesperado, mas que também o objetivo não foi atingido.

O TBM possui suas vantagens, mas também possui suas desvantagens. Algumas das vantagens mais visíveis e destacáveis do TBM é que este possibilita a geração automatizada dos casos de teste, pode proporcionar a detecção de defeitos no sistema ou em até mesmo nos seus documentos, melhorando a qualidade do mesmo (BLACKBURN et al., 2004; UTTING e LEGEARD, 2006). Como o TBM automatiza os casos de teste, isso possibilita a redução de tempo ao testar um sistema e o custo do teste (DALAL et al., 1999). Uma de suas desvantagens, é que se o sistema for muito complexo, contendo muitas restrições e funcionalidades, torna-se difícil modelá-lo e gerar seus casos de teste, e quando um caso de teste falha, é necessário saber diferenciar se o defeito está no sistema ou até mesmo no modelo de teste, e em muitos casos a eficiência deste processo depende das habilidades do testador (EL-FAR e WHITTAKER, 2001; UTTING e LEGEARD, 2006).

2.1.3 ESG

ESG é uma das formas possíveis de modelar um subconjunto de interações de um sistema (BELLI et al., 2005). Desta forma, é possível modelar eventos de um sistema e a sequência em que eles ocorrem. Assim, os eventos são representados como nós, e a sequência válida desses é representada pelas arestas, conforme mostra a Figura 4.

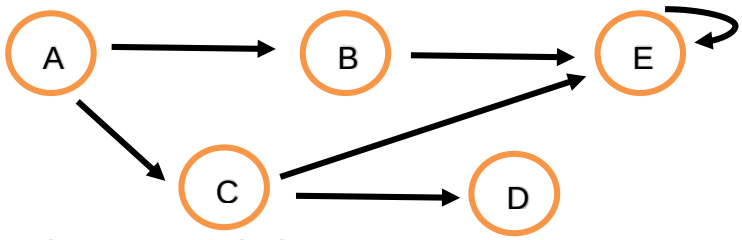


Figura 4: Exemplo de ESG
Fonte: Autoria Própria.

A Figura 5 ilustra um exemplo de modelagem da funcionalidade de copiar, recortar e colar um arquivo. Neste momento os nós representados por “[“ e “]” indicam evento de entrada e saída, respectivamente.

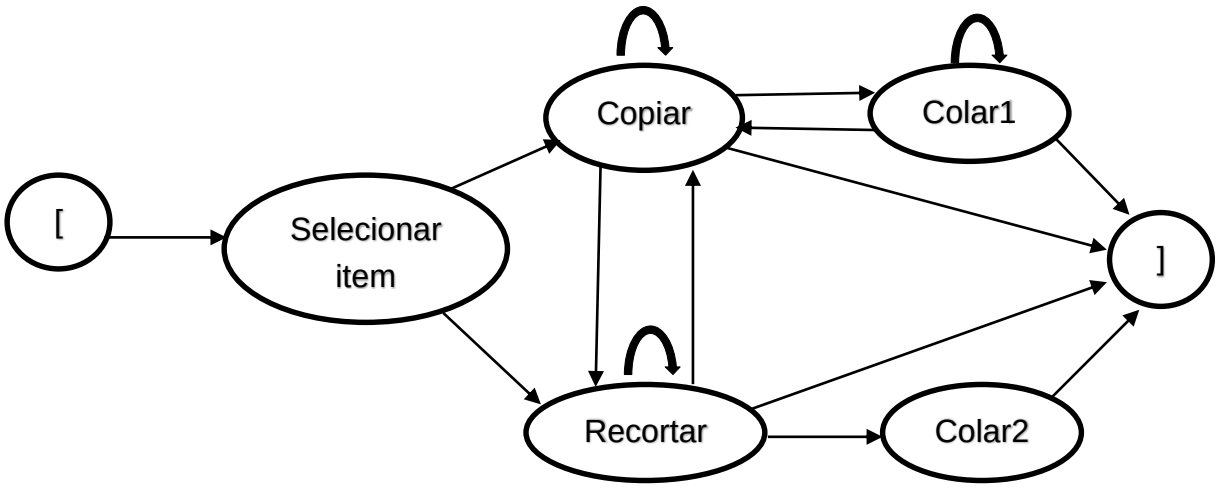


Figura 5: Modelagem da Funcionalidade de Copiar, Recortar e Colar Utilizando ESG
Fonte: Adaptado de ENDO (2013).

De acordo com a Figura 5, é possível observar que há basicamente os eventos de selecionar item, recortar, copiar e colar. Desta forma, o ciclo se inicia ao selecionar determinado item e escolher entre as opções de copiá-lo ou recortá-lo. Por fim, foram inseridos dois eventos referentes ao processo de colagem: Colar1 e Colar2, devido ao fato de que ao copiar determinado item é possível colá-lo várias vezes, enquanto que recortar é possível colar apenas uma vez.

Uma das vantagens do ESG, é que este permite que sejam modelados tantos os eventos desejáveis, quanto os indesejáveis (BELLI et al., 2014). Além disso, a modelagem utilizando ESG é possível ser aprendido em um curto período (BELLI et al., 2006a) e exige pouco trabalho manual, pois há ferramentas específicas (BELLI et al., 2006b).

2.2 TRABALHOS RELACIONADOS

Na literatura é possível encontrar alguns estudos e trabalhos relacionados ao uso da técnica do TBM e o ESG. Grieskamp et al. (2011) apresentam um estudo relacionado ao processo de garantia de qualidade para documentação dos protocolos do Windows, utilizando o TBM como técnica de teste. Foi utilizada a ferramenta *Spec Explorer* para fazer a modelagem do TBM. Nesse estudo, foi revelado que o TBM

possuiu um ganho de produtividade de 42% quando comparado com um conjunto de testes tradicionais (gerados de forma manual). Também foi possível observar que a aplicação do TBM para processos de garantia de qualidade para documentação de protocolos se limita a sistemas de médio porte.

Em Belli et al. (2014), um estudo de caso é feito para verificar se o modelo ESG4WSC, o qual permite gerar casos de teste baseado no TBM e ESG, é aplicável para o desenvolvimento de uma aplicação orientada a serviço, invocando serviços compostos de tamanhos diferentes. Assim, foi utilizado como estudo de caso o aplicativo *xTripHandling*, o qual permite consultar e reservar viagens. Para isso, foram utilizadas a ferramenta TSD, para modelar todos os recursos necessários para geração de casos de teste do ESG4WSC, e também a ferramenta ERunTE para a execução dos testes. O caso estudo apontou que a abordagem é aplicável a uma aplicação orientada a serviços web não trivial, e que numerosos defeitos foram revelados não só na aplicação, mas também na especificação, pois entre dois serviços principais da aplicação: *Travel Agent* e *Customer Service*, 33,33% dos defeitos encontrados no primeiro serviço foram relacionados a especificação, e do segundo serviço 48,65% dos defeitos encontrados também foram relacionados a especificação. Assim, a abordagem ajudou a manter a implementação e especificação sincronizada.

Utting e Legeard (2006), relatam que Pretschner et al. (2005) fizeram um estudo de caso para tentar medir os benefícios do TBM comparado com os testes manuais, utilizando como objeto de estudo um controlador de rede automotivo, e a ferramenta *AutoFocus* para modelagem. Durante a fase de modelagem do teste, foram encontrados 30 defeitos em requisitos: 3 de inconsistência, 7 de omissão e 20 de ambiguidade. E além disso, foi concluído que o TBM detecta aproximadamente a mesma quantidade de defeitos de implementação que os testes manuais, porém o TBM consegue detectar mais erros em requisitos que o os testes manuais.

Conforme os estudos de caso descritos anteriormente, é possível observar que há um estudo relacionado com o TBM, a fim de conseguir validar sua aplicabilidade e eficiência em encontrar defeitos no sistema e nas documentações do software. E nos resultados de ambos trabalhos foi relatado que esta técnica de teste realmente é viável para detectar defeitos e aumentar a qualidade do sistema. Porém, nos trabalhos mencionados, em nenhum momento a capacidade do TBM em encontrar defeitos nos requisitos durante a etapa de modelagem foi avaliada de forma sistemática.

3 CONFIGURAÇÃO DO EXPERIMENTO

Nessa seção são apresentadas as descrições das atividades que foram desenvolvidas para alcançar o principal objetivo deste trabalho, o qual é verificar se há a detecção de defeitos no documento de requisitos de software durante a etapa de modelagem do TBM, utilizando a técnica ESG para a construção do modelo de teste.

Desta forma, essa seção foi subdivida de acordo com o fluxograma da configuração do experimento e as atividades planejadas: Elaboração do Documento de Requisitos, Inserção de Defeitos no Documento de Requisitos, Elaboração dos Questionários e Material de Apoio, Planejamento e Execução do Experimento e Ameaças à Validade.

3.1 FLUXOGRAMA DA CONFIGURAÇÃO DO EXPERIMENTO

Na Figura 6 a seguir, é mostrado um fluxograma contendo as atividades desenvolvidas neste trabalho e suas relações.

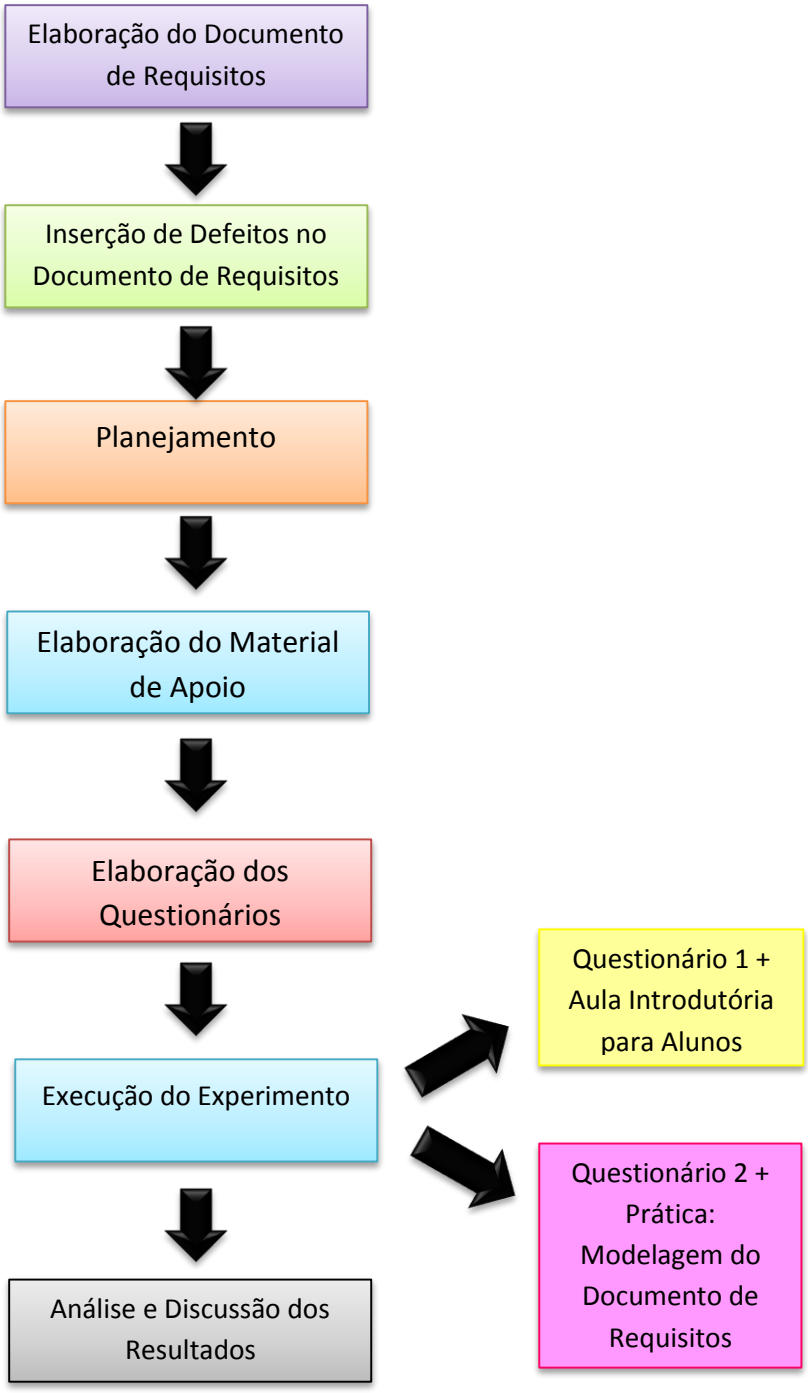


Figura 6: Fluxograma de Atividades
Fonte: Autoria Própria.

3.2 ATIVIDIDADES DE PLANEJAMENTO DO EXPERIMENTO

Essa subseção apresenta todas as atividades que foram desenvolvidas neste trabalho, para se chegar ao objetivo do mesmo, e a configuração do experimento.

3.2.1 Elaboração do Documento de Requisitos

Essa etapa de elaboração do documento de requisitos, que será utilizado na fase de modelagem do TBM no experimento, consistiu em criar um documento de requisitos de uma aplicação para dispositivos móveis. Assim, primeiramente, foi escolhida uma aplicação Android, *Enhanced SlideShow*, de Deitel et al. (2013), para elaborar o documento de requisitos. A aplicação escolhida permite criar apresentações em *slides* podendo conter imagens, músicas e vídeos, em dispositivos Android.

Essa etapa do trabalho começou com uma atividade passada pelo orientador deste trabalho, para seus alunos da turma (1/2015) de Engenharia de Software do curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR-CP), que requisitava que os alunos elaborassem um documento de requisitos da aplicação *Enhanced SlideShow*, contendo uma visão geral da mesma e suas principais funcionalidades.

A partir dessa tarefa, foram selecionados os melhores documentos de requisitos, para construir o documento deste trabalho, mas, antes disso, foram necessários realizar alguns testes na aplicação, para conhecer todas as funcionalidades que esta permite o usuário realizar e validar as informações contidas nos documentos elaborados pelos alunos.

Posteriormente, foi construído o documento de requisitos, contendo as principais informações da aplicação *Enhanced SlideShow*, e os fluxos das funcionalidades possíveis de serem feitas nesta aplicação. Depois de finalizado, o documento de requisitos passou por várias leituras *Ad-Hoc* e aprovação do professor orientador deste trabalho, de forma que houvesse menos defeitos possíveis.

Além disso, dois alunos de graduação do curso de Engenharia de Computação, que já possuíam conhecimento sobre TBM e ESG, fizeram a leitura do

documento de requisitos e elaboraram o modelo de teste utilizando a técnica ESG a partir desse documento, com o objetivo de colaborar com a validação do documento.

3.2.2 Inserção de Defeitos no Documento de Requisitos

Essa etapa compreendeu na ação de inserir alguns defeitos no documento elaborado na etapa anterior, de acordo com algumas das sete categorias de defeitos o qual Basili (1981) expõe. Assim, foram inseridos defeitos das seguintes categorias:

- Defeitos de digitação: algum erro simples de digitação no documento;
- Ambiguidade: informação no documento que possui mais de uma interpretação;
- Omissão de informação: alguma informação que foi emitida no documento;
- Inconsistência: duas informações no documento que se contrapõem;
- Fato incorreto: informação errada no documento.

Dessa forma, a partir dessas categorias de defeitos selecionadas, foram inseridos alguns defeitos de cada categoria descrita anteriormente, resultando no documento de requisitos que foi utilizado no experimento, conforme Apêndice A. Os defeitos inseridos, totalizando 14 defeitos, foram os seguintes:

- **D#1 -** Requisito R1: defeito de digitação: palavra “apricação” ao invés de aplicação;
- **D#2 -** Requisito R4: fato incorreto: “A aplicação deve permitir que o usuário visualize SOMENTE algumas apresentações das apresentações criadas, selecionadas por ele”, pois o usuário pode visualizar qualquer apresentação criada que esteja salva na galeria da aplicação;
- **D#3 -** Requisitos R5 e R7: inconsistência: no R5 é informado que o usuário pode selecionar QUAISQUER vídeos e imagens salvas na galeria ou no cartão de memória do dispositivo; e o R7 diz que o usuário só pode selecionar vídeos MP4 e imagens JPG da galeria ou cartão de memória do dispositivo;
- **D#4 -** Requisito R9: fato incorreto: o requisito informa que o usuário pode remover apenas imagens ou músicas que já foram selecionadas na apresentação, sendo que se podem ser selecionados vídeos, estes também podem ser removidos;
- **D#5 -** Requisito R10: defeitos de digitação: palavra “possilitar” ao invés de possibilitar;

- **D#6 -** Requisitos R11 e R18: inconsistência: o requisito R11 informa que o usuário pode ver cada foto ou imagem da apresentação por 5 segundos, e no requisito R18 é informado que cada imagem ou foto podem ser visualizados por um tempo determinado pelo usuário;
- **D#7 -** Requisito R13: omissão: neste requisito é omitido a unidade ao informar que a aplicação deve responder aos botões em no máximo 1, ao invés de 1 segundo;
- **D#8 -** Requisito R15: omissão: não é informado o que significaria ser “fácil de utilizar” a aplicação;
- **D#9 -** Requisitos R17 e R21: inconsistência: no requisito R17 é informado que pode ser criada uma nova apresentação apertando o botão menu do dispositivo, já o requisito R21 informa que não é só apertar o botão menu para criar uma nova apresentação, pois ao apertar este botão é exibida uma opção na tela para criar uma nova apresentação dando um nome a ela;
- **D#10 -** Requisito R19: omissão: este requisito não informa o que irá acontecer se o usuário selecionar o botão “Não” da Figura 3 do Apêndice A;
- **D#11 -** Requisito R23: ambiguidade: o termo “caso contrário” dá ambiguidade, pois não é possível entender se a mensagem de erro irá aparecer se o usuário não selecionar o botão “OK” ou se o nome não for composto por letras e números;
- **D#12 -** Requisito R28: defeito de digitação: palavra “image” ou invés de imagem;
- **D#13 -** Requisito R29: ambiguidade: o termo “caso contrário” dá ambiguidade na frase, pois não dá para entender se não será possível escolher os vídeos pelo fato de não existir o cartão de memória ou uma galeria no dispositivo, ou pelo fato de não existir vídeos salvos em um desses dois locais;
- **D#14 -** Requisito R33: fato incorreto: uma apresentação é exibida na ordem que o usuário organizou seus itens, e neste requisito diz que a apresentação será exibida sem seguir esta ordem.

Esta etapa de inserção de defeitos servirá para facilitar uma parte da validação do experimento, aquela que corresponde em observar e verificar se as pessoas que realizaram o experimento realmente encontraram defeitos coerentes e corretos no documento de requisitos. Mas, isso não impede que haja outros defeitos

não identificados previamente, pois é possível que na elaboração do documento de requisitos, alguns defeitos não tenham sido detectados.

3.2.3 Planejamento

Essa seção segue uma estrutura semelhante à estrutura de planejamento de experimentos descritas em Wohlin et al. (2012), pelo fato de possuir apenas alguns tópicos, utilizados pelo autor, para a descrição de planejamento do experimento.

3.2.3.1 Objetivo

O objetivo desse experimento é analisar se na etapa de modelagem do TBM é possível encontrar defeitos no documento de requisitos. Os participantes do experimento serão alunos de graduação que cursam a disciplina de Engenharia de Software. Assim, é necessário avaliar o desempenho desses alunos que está relacionado em encontrar ou não defeitos no documento de requisitos.

3.2.3.2 Formulação de Hipóteses

Em um experimento é necessário declarar o que será feito para se chegar ao objetivo declarado e por fim, em uma conclusão (WOHLIN et al., 2012). Para isso, são formuladas algumas considerações e hipóteses relacionadas ao objeto de estudo. Será admitido que os alunos selecionados para fazerem o experimento não possuam conhecimento específico das palavras-chave deste trabalho: TBM, ESG e inspeção de requisitos. Para isso, antes dos mesmos começarem o experimento, será dado um questionário aos alunos, a fim de conhecer seus perfis, ou seja, o grau de conhecimento específico em relação as palavras-chave deste trabalho. E posteriormente, será apresentada uma aula resumida sobre os principais conceitos deste trabalho, para que os mesmos possam realizá-lo de forma coerente e válida.

Assim, a principal hipótese deste trabalho é:

H_0: não existe a detecção de defeitos no documento de requisitos de software durante a etapa de modelagem do TBM.

H_1: existe a detecção de defeitos no documento de requisitos de software durante a etapa de modelagem do TBM.

3.2.3.3 Seleção de Variáveis

As variáveis independentes deste trabalho são a experiência e comprometimento dos alunos que fizeram o experimento. E as variáveis dependentes são a quantidade de defeitos encontrados pelos alunos, o tempo de execução do experimento e as classes de defeitos mais encontradas no documento de requisitos.

3.2.3.4 Seleção dos Participantes

Foram selecionados alunos de graduação do curso de Engenharia de Computação que estão cursando a disciplina de Engenharia de Software da Universidade Tecnológica Federal do Paraná – campus Cornélio Procópio. Foram selecionados esses alunos em específico, pelo fato da disciplina de Engenharia de Software abranger temas relacionados à inspeção de requisitos e testes de software.

3.2.4 Instrumentação

Essa subseção apresenta duas etapas do experimento que consistem em elaborar ferramentas necessárias para a execução do experimento.

3.2.4.1 Elaboração do Material de Apoio

Essa atividade consistiu em elaborar o material de apoio que foi utilizado na apresentação de uma aula breve aos participantes do experimento, com o intuito de apresentar os seguintes temas: Inspeção de Requisitos, TBM e ESG.

Assim, foi utilizado como base um material já existente feito pelo orientador deste trabalho, e foram modificados alguns conteúdos para que houvesse mais exercícios práticos para treinar o conceito de Inspeção de Requisitos, TBM e ESG. Lembrando que este material foi desenvolvido na ferramenta PowerPoint 2013.

É importante ressaltar que foram passados 4 exercícios de fixação dos conteúdos abordados na aula, todos com o objetivo de criar um modelo de teste utilizando a técnica ESG.

3.2.4.2 Elaboração dos Questionários

Essa etapa de elaboração de questionários tem o objetivo de obter algum *feedback* em relação aos perfis dos participantes e à execução do experimento. Foram elaborados dois questionários, utilizando a ferramenta Google *Docs*, a qual possibilita a criação de questionários na web, além de gerar uma boa visualização em relação aos resultados obtidos.

O primeiro questionário elaborado possui algumas perguntas pessoais com o objetivo de saber se o aluno possui algum conhecimento ou não em relação as palavras-chave deste trabalho: Inspeção de Requisitos, TBM e ESG, para depois este resultado poder ser relacionado com o desempenho geral dos participantes no experimento.

Já o segundo questionário possui algumas perguntas relacionadas à execução do experimento, para saber se o tempo necessário para fazer o experimento foi suficiente, se houve alguma dificuldade em fazê-lo, opinião sobre a técnica ESG e TBM, entre outros aspectos.

3.3 EXECUÇÃO DO EXPERIMENTO

Esse experimento foi realizado em dois dias, com duração de 1 hora e 40 minutos em cada dia de experimento, com um total de 16 alunos participantes. No primeiro dia, inicialmente, os participantes responderam o primeiro questionário elaborado, com perguntas relacionadas aos perfis e grau de conhecimento específico em relação aos principais assuntos deste trabalho. Em seguida, foi apresentado aos alunos o material de apoio elaborado sobre os assuntos: Inspeção de Requisitos, TBM e ESG.

Além das breves explicações, houve resolução de exercícios relacionados em elaborar modelo de teste a fim de fixar mais os assuntos. É importante destacar, que todos os alunos fizeram o modelo de teste, utilizando a técnica ESG na ferramenta Astah, dos exemplos e exercícios que haviam no material de apoio, também fizeram perguntas e questionamentos para sanar dúvidas, e por fim, todos os exercícios foram posteriormente corrigidos juntamente com o professor responsável pela disciplina.

O segundo dia consistiu em os alunos realizar a principal atividade deste experimento: a elaboração do ESG de acordo com o documento de requisitos da

aplicação *Enhanced SlideShow*, também utilizando a ferramenta Astah, para facilitar o desenvolvimento do ESG. Assim, foi disponibilizado no ambiente virtual de ensino da disciplina de Engenharia de Software o documento de requisitos para os alunos e foi entregue um formulário impresso para estes relatarem os defeitos, caso encontrem. Os mesmos foram orientados que se encontrassem algum defeito relacionado aos requisitos contidos no documento, era para relatá-lo no formulário, descrevendo o número do requisito e uma breve descrição do defeito.

Depois que todos os alunos terminaram a atividade de modelagem, os alunos foram guiados a responder o segundo questionário, contendo perguntas em relação a execução e objetivo do experimento realizado. Neste dia, houve três alunos que não haviam participado do treinamento dado no primeiro dia do experimento, assim, estes foram guiados a fazerem apenas o processo de inspeção de requisitos no documento, utilizando a técnica de leitura *Ad-Hoc*, sem elaborar o modelo de teste e sem preencher o questionário 2.

É importante ressaltar que todos os materiais utilizados neste experimento estão disponíveis no seguinte endereço: https://github.com/andreendo/MBTRequisitosExperimento, tais como: os dois questionários, formulário para relatar os defeitos, material de apoio e o documento de requisitos da aplicação *Enhanced SlideShow.*

3.4 AMEAÇAS À VALIDADE

A primeira ameaça é o fato do experimento ter sido realizado com alunos de graduação e não com profissionais da área de informática. Assim, os indivíduos que participaram do experimento podem não ser representativos para testadores acostumados com o TBM.

Outro aspecto é o fato de ter sido utilizado a técnica ESG para elaborar o modelo de teste, dentre outras técnicas existentes que possibilitam realizar essa mesma atividade na aplicação do TBM. Vale ressaltar que outras técnicas de modelagem poderiam direcionar o testador a revelar outros defeitos nos requisitos. Entretanto, a técnica ESG foi escolhida por permitir ser aprendida em um curto período (BELLI et al., 2006a) e por ser fácil de executar.

O documento de requisitos elaborado usou uma série de defeitos inseridos artificialmente. Porém essa ameaça tentou ser minimizada para proposta de defeitos

inseridos com base em defeitos praticados pelos alunos que elaboraram, inicialmente, o documento de requisitos (de acordo com a atividade passada pelo professor, descrita na Subseção 3.1.1).

Por fim, pode ser que o documento de requisitos, utilizado neste experimento, não seja representativo em relação a um documento real de uma aplicação. Entretanto, o documento de requisitos elaborado foi baseado em uma aplicação Android real e proposta por outros autores.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Esse capítulo é dividido em 3 subseções conforme foi executado o experimento. Dessa forma, a Subseção 4.1 revela os resultados encontrados a partir do questionário 1 aplicado; a Subseção 4.2 aponta os resultados do experimento em relação a modelagem e por fim a subseção 4.3 expõe os resultados adquiridos a partir do questionário 2.

4.1 RESULTADOS DO QUESTIONÁRIO 1

De acordo com o primeiro dia do experimento, o questionário 1, o qual possuía perguntas relacionadas aos perfis dos alunos com o intuito de conhecer o grau de conhecimento dos mesmos em relação a Inspeção de Requisitos, TBM e ESG. De acordo com a Figura 7 é possível destacar que a maioria dos alunos são do sexo masculino e possuem faixa etária de 22 a 26 anos, ou seja, são jovens ainda em fase de formação.

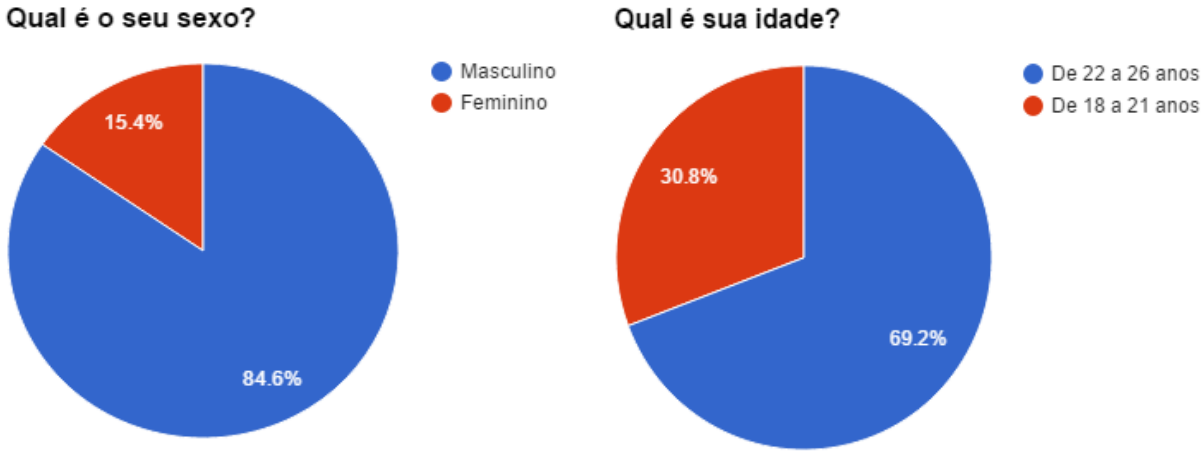


Figura 7: Respostas das Perguntas 1 e 2 – Questionário 1
Fonte: Google Docs.

A Figura 8 mostra que a maioria sabe o conceito de Engenharia de Software e apenas 46,2%, menos da metade, possui conhecimento sobre o que é o Documento de Requisitos de Software, sendo que 15,4% responderam que não ouviram falar sobre este processo. Também é importante ressaltar, que mesmo a maioria tendo respondido que sabe o que significa Engenharia de Software, 23,1% responderam que não tem certeza sobre este conceito.

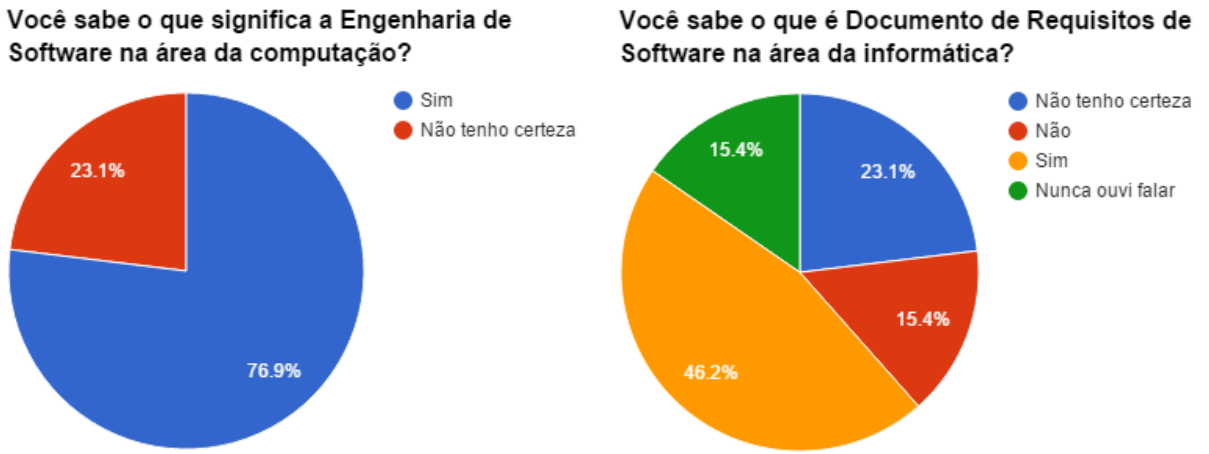


Figura 8: Respostas das Perguntas 3 e 4 – Questionário 1
Fonte: Google Docs.

Em relação a Figura 9, percebe-se que a maioria dos alunos não conhecem as classes de defeitos que podem existir em um Documento de Requisitos de Software, sendo que apenas 7,7% dos alunos conhecem essas classes. E também é possível verificar que a maioria também não conhece o processo de Inspeção de Requisitos, o qual ajuda a encontrar esses defeitos no documento, sendo que 15,4% responderam que possuem conhecimento sobre o processo.

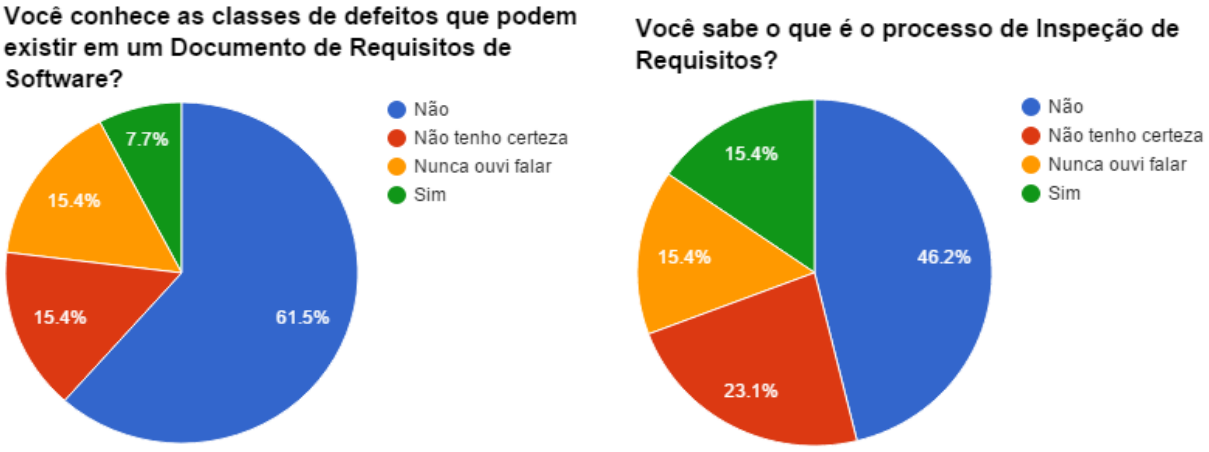


Figura 9: Respostas das Perguntas 5 e 6 – Questionário 1
Fonte: Google Docs.

E por fim, a Figura 10 destaca que a maioria não conhece ou nunca ouviu falar sobre TBM e a técnica ESG, o que já era esperado para este experimento.

Figura 10: Respostas das Perguntas 7 e 8– Questionário 1
Fonte: Google Docs.

4.2 RESULTADOS DO EXPERIMENTO

De acordo com os resultados do segundo dia do experimento, foi possível verificar o tempo que os 16 participantes levaram para realizar o experimento e a quantidade de defeitos encontrados individualmente em relação às classes de defeitos, os quais estão expostos no Quadro 1:

De acordo com o Quadro 1, é possível perceber que a maioria dos alunos encontraram menos que 50% do total de defeitos que tinham sido inseridos no documento. Apenas 2 participantes encontraram 7 defeitos, sendo 50% do total (14 defeitos). Também é interessante destacar que nenhum participante encontrou defeitos de ambiguidade. E é importante ressaltar que o participante A13, o qual não encontrou nenhum defeito, acabou encontrando outros defeitos que não foram previstos, os quais serão expostos mais adiante.

Alunos	Tempo (minutos)	Digitação (3 Defeitos)	Omissão (3 Defeitos)	Inconsistência (3 Defeitos)	Fato Incorreto (3 Defeitos)	Ambiguidade (2 Defeitos)	Total (14 Defeitos)	Total (%)	Outros Defeitos (Quadro 2)
A1	71	1	1	0	1	0	3	21,43	4
A2	66	2	1	0	0	0	3	21,43	2
A3	64	1	1	0	0	0	2	14,28	5
A4	62	2	2	1	0	0	5	35,71	1
A5	63	1	1	2	0	0	4	28,57	1
A6	42	2	1	1	0	0	4	28,57	1
A7	34	2	1	1	1	0	5	35,71	0
A8	42	2	1	2	2	0	7	50,00	1
A9	32	2	1	0	0	0	3	21,43	1
A10	45	1	2	1	3	0	7	50,00	1
A11	48	1	0	1	1	0	3	21,43	1
A12	53	1	2	2	0	0	5	35,71	1
A13	54	0	0	0	0	0	0	0,00	1
A14	56	2	1	2	1	0	6	42,86	3
A15	57	0	1	1	0	0	2	14,28	0
A16	61	1	2	1	0	0	4	28,57	1

Quadro 1: Defeitos Encontrados pelos Alunos
Fonte: Autoria Própria

No Gráfico 1, tem-se o *boxplot* em relação ao tempo que os alunos levaram para realizar o experimento, lembrando que a divisão do *boxplot* refere-se a mediana dos dados, a qual é igual a 55. Assim, é possível destacar que os valores acima da mediana (Q3), estão mais próximos, ou seja, menos dispersos que os valores a baixo (Q1) da mediana. Desta forma, pode-se concluir, que os dados são negativamente assimétricos, pelo fato da mediana ser mais próxima dos valores mais altos.

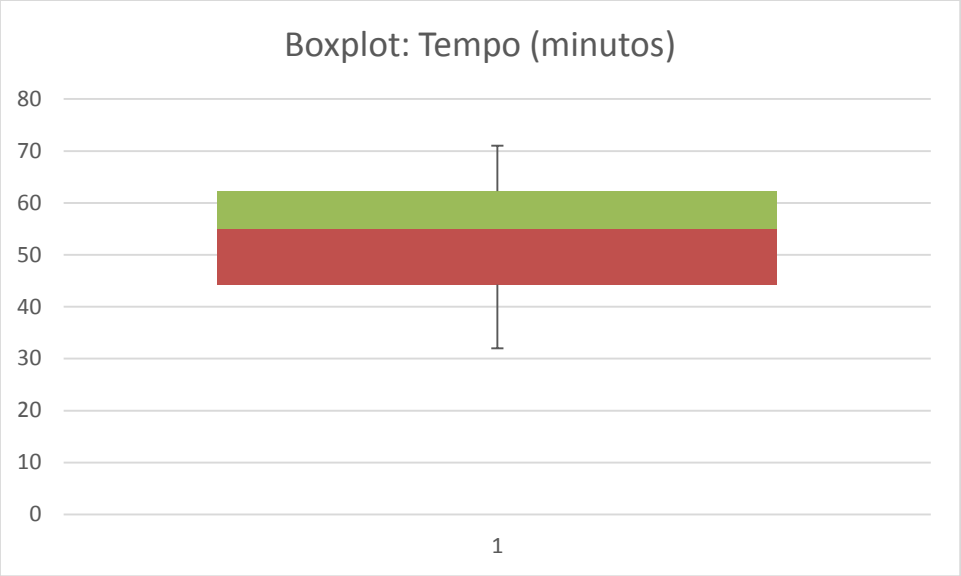


Gráfico 1: *Boxplot* do Tempo de Duração do Experimento
Fonte: Autoria Própria.

No Gráfico 2, tem-se o *boxplot* em relação a quantidade de defeitos que os alunos encontraram no documento durante o experimento. Assim, é possível destacar que os valores de Q1 e Q3 estão bem distribuídos em relação a mediana, a qual é igual a 4. Desta forma, pode-se concluir, que os valores possuem uma distribuição simétrica.

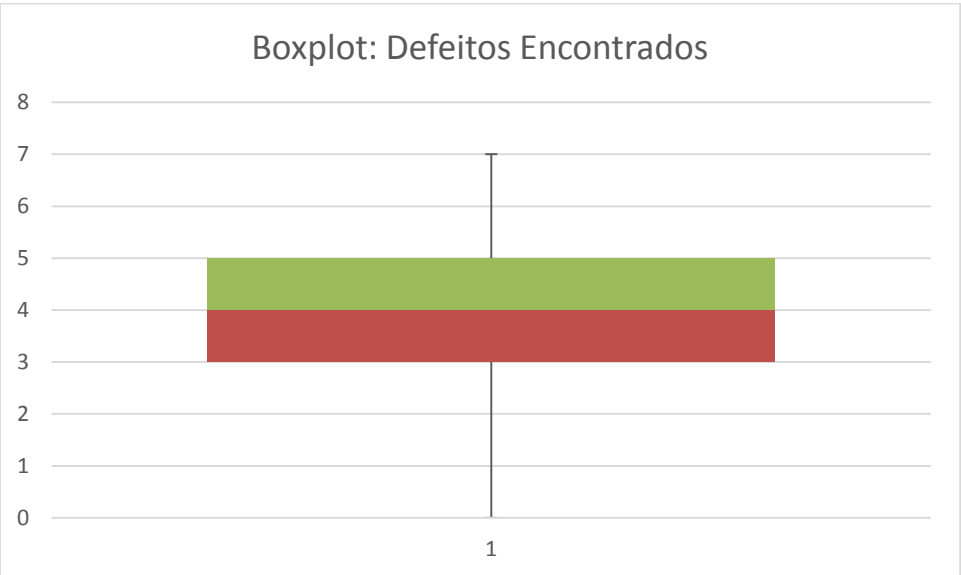


Gráfico 1: *Boxplot* da Quantidade de Defeitos Encontrados pelos Alunos
Fonte: Autoria Própria.

O Gráfico 3 mostra uma visão geral dos defeitos encontrados pelos alunos, destacando que alguns defeitos, como D#1 (defeito de digitação), D#3 (defeito de inconsistência) e D#7 (defeito de omissão de unidade de medida), foram encontrados com uma quantidade maior de vezes, pelos alunos. Entretanto, alguns defeitos, como D#9 (defeito de inconsistência), D#11 e D#13 (defeitos de ambiguidade), não foram encontrados.

Gráfico 3: Quantidade de Alunos que Encontraram Defeitos
Fonte: Autoria Própria.

Em relação aos três alunos que não participaram do treinamento, e só realizaram o processo de inspeção *Ad-Hoc* no documento, o Gráfico 4 mostra que os defeitos mais encontrados pelos alunos foram: D#2 (defeito de fato incorreto), D#7 (defeito de omissão de unidade de medida) e D#12 (defeito de digitação). Estudando os Gráfico 3 e Gráfico 4, é importante ressaltar que o defeito de omissão de unidade de medida D#7 foram verificados em ambos os casos, e o defeito da classe de erro de digitação também.

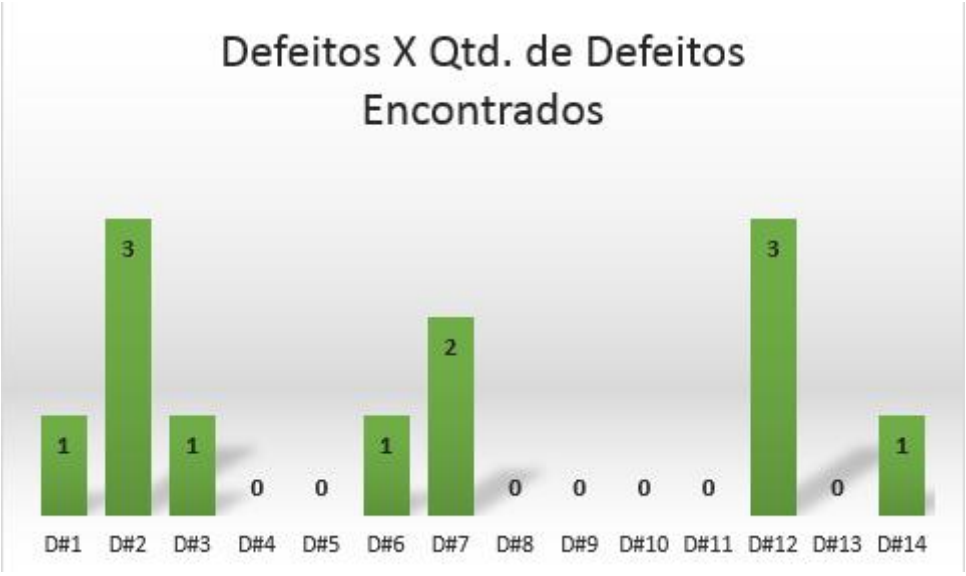


Gráfico 4: Defeitos X Defeitos Encontrados por Alunos não Treinados
Fonte: Autoria Própria.

Ainda em relação aos resultados do experimento, é possível perceber outro fato interessante: tantos os participantes treinados e os não treinados verificaram outros defeitos além dos 14 defeitos inseridos no documento de requisitos, os quais não tinham sido previstos inicialmente. No Quadro 2, são expostos alguns defeitos relatados tanto pelos 16 alunos que foram treinados tanto pelos três alunos que não tiveram treinamento.

Em suma, foi possível verificar neste experimento, que houve a detecção de defeitos no documento de requisitos durante a elaboração do modelo de teste, utilizando a técnica ESG. E as classes de defeitos mais encontradas foram: defeito de digitação, omissão e inconsistência. Destacando que os alunos treinados encontraram uma quantidade maior de defeitos que os três alunos não treinados que só executaram o processo de inspeção de requisitos, utilizando a técnica de leitura *Ad-Hoc.*

Em relação aos modelos de testes elaborados, a maioria dos alunos modelaram a aplicação como um todo, utilizando o padrão de início e fim da técnica ESG, através dos símbolos “[” e “]”, respectivamente. Já alguns alunos, criaram um modelo de teste para cada funcionalidade escolhida, tais como: editar, remover, criar uma nova apresentação, a forma que a apresentação é exibida (pausar, voltar, adiantar) e até mesmo os requisitos de interface, dentre as ações que o usuário pode realizar. O Anexo A mostra esses dois tipos de diagramas, descritos anteriormente, elaborada por dois participantes do experimento.

Requisitos	Descrição	Classe do Defeito
R6	Torna-se inconsistente junto com os requisitos R7 e R5	Inconsistência
R12	Omissão em relação a palavra qualidade	Omissão
R4	"Algumas apresentações" quantas seriam? Como visualizar uma apresentação nunca criada?	Omissão
R26	Pode escolher mais de uma imagem?	Omissão
R16	Erro de tempo verbal: deveria ser "ao se iniciada"	Digitação
R6	Não ficou claro se o áudio precisa ser salvo na galeria. Pode-se gravar um áudio ou usar uma gravação já salva?	Omissão
R21/R22	Omissão: não tem tela com a mensagem "novo *slideshow*" e nem botão para criar um novo	Omissão
R17/R21	Não ficou claro que o botão menu é o botão menu do DISPOSITIVO	Omissão
R18	O R31 não faz referência a esse requisito	Fato Incorreto
R26	O termo "designada a esse local" é ambíguo	Ambiguidade
R14	Funcionará em todas as versões do Android?	Omissão
R32	A Figura 3 exibe mensagem para remover um slide e não um item do slide	Omissão/Fato Incorreto
R31	Erro verbal: o certo seria "usuário FOR"	Digitação
VISÃO GERAL	Erro de digitação na visão geral: exibeS, deveria ser exibe	Digitação

Quadro 2: Outros Defeitos Destacados pelos Alunos
Fonte: Autoria Própria

4.3 RESULTADOS DO QUESTIONÁRIO 2

Em relação ao questionário 2, teve-se os seguintes resultados. De acordo com a Figura 11, é possível destacar que a maioria dos alunos concordaram totalmente/parcialmente que o material de apoio foi suficiente para ter uma base de conhecimento para realizar o experimento, e também que o tempo para executar o experimento foi ideal.

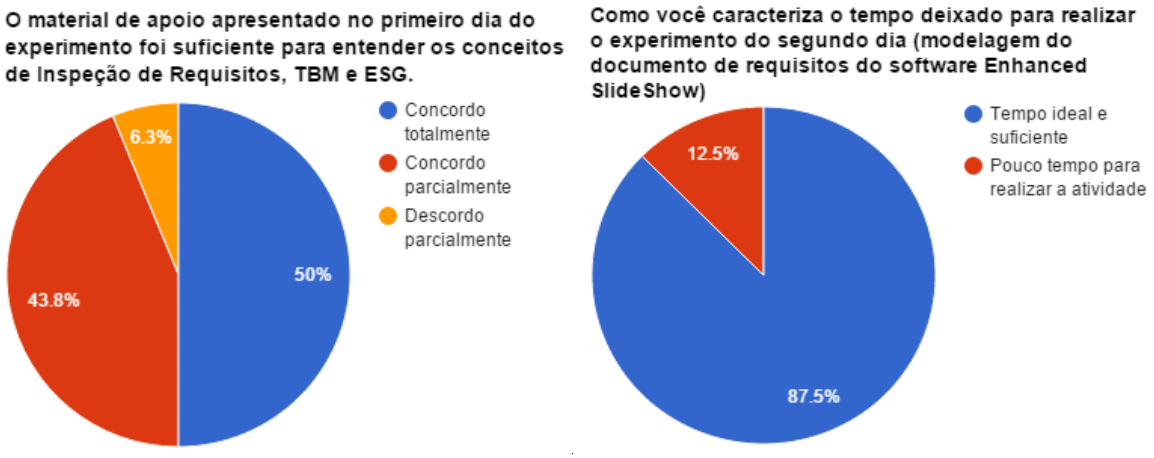


Figura 11: Respostas das Perguntas 3 e 4– Questionário 2
Fonte: Google Docs.

Na Figura 12, é destacado que a maioria concorda que a técnica ESG contribui para o processo de inspeção de requisitos e que permite encontrar defeitos de digitação, a qual estava entre a maioria de classe de defeitos encontrados de acordo com o Gráfico 1.

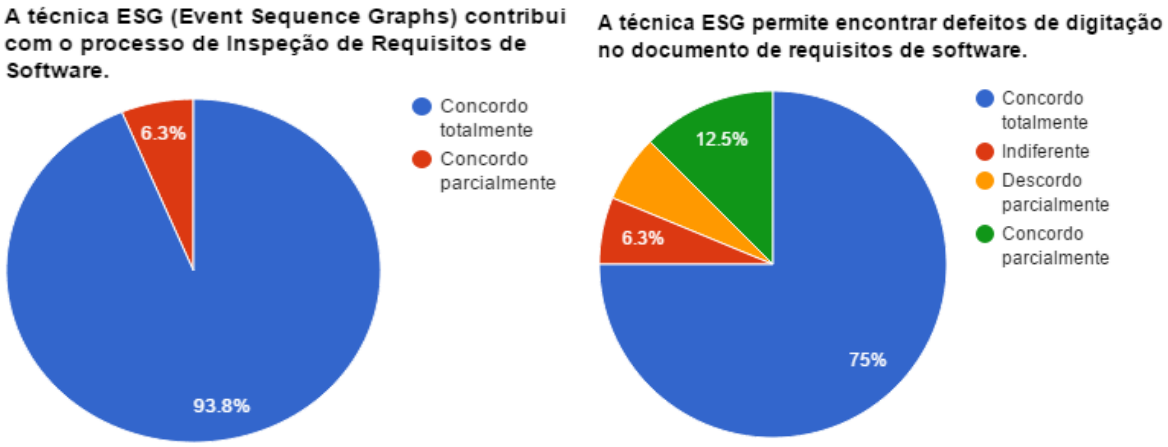


Figura 12: Respostas das Perguntas 5 e 6– Questionário 2
Fonte: Google Docs.

A Figura 13 também mostra que a maioria concorda que a técnica ESG permite encontrar defeitos de inconsistência e fato incorreto, que também foram verificados no Gráfico 1.

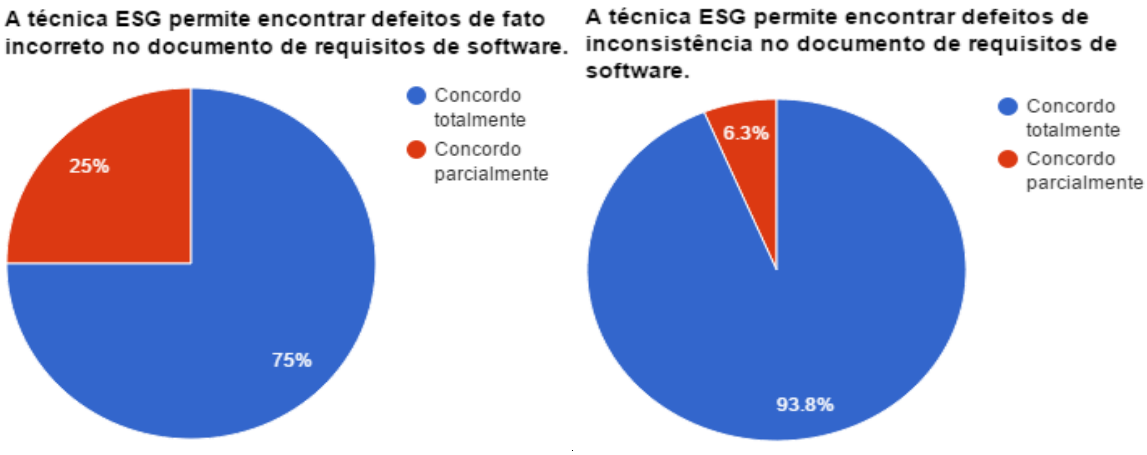


Figura 13: Respostas das Perguntas 7 e 8– Questionário 2
Fonte: Google Docs.

E por fim, a Figura 14 exibe que a maioria também concorda que a técnica ESG contribui para detectar defeitos de ambiguidade e omissão. Importante destacar, que nenhum participante havia encontrado um defeito de ambiguidade no documento de requisitos, mas mesmo assim concordaram que a técnica ESG permite encontrar essas classes de defeitos no documento.

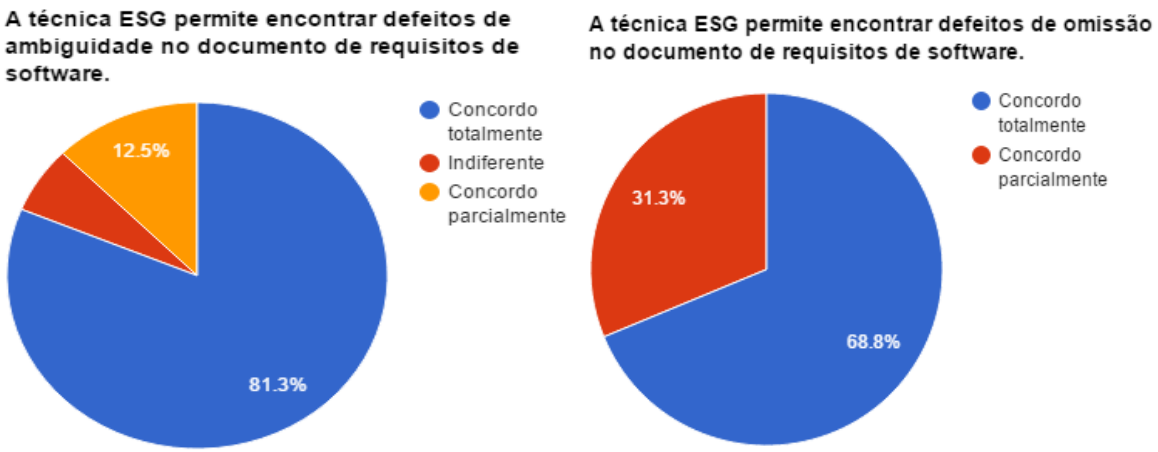


Figura 14: Respostas das Perguntas 9 e 10– Questionário 2
Fonte: Google Docs.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como principal objetivo verificar se ao realizar o processo de modelagem do TBM é possível realizar a inspeção de requisitos, de forma que sejam encontrados defeitos no documento de requisitos do software testado. Assim, o desenvolvimento e os resultados deste trabalho podem contribuir com especialistas da área de Engenharia de Software, bem como ajudar no desenvolvimento de software aumentando sua qualidade e reduzindo seus custos.

Assim, para alcançar o principal objetivo deste trabalho, foi realizado um experimento com 16 alunos do curso de Engenharia de Computação, que cursavam a disciplina de Engenharia de Software, o qual consistiu em dois dias. No primeiro dia, foi passado um questionário aos alunos, a fim de conhecer seus perfis, e posteriormente foi dado um treinamento básico sobre inspeção de requisitos, TBM e ESG através de explicações e exercícios de fixação.

O segundo dia do experimento coube os alunos elaborarem um modelo de teste com base em um documento de requisitos de uma aplicação Android, utilizando a técnica ESG, e os mesmos foram alertados que se caso encontrassem defeitos neste documento, que anotassem em um formulário. Depois, os alunos preencheram um segundo questionário, o qual daria um *feedback* em relação ao experimento.

Desta forma, foi possível verificar que a maioria dos alunos detectaram os defeitos que foram inseridos no documento de requisitos, e também conseguiram apontar outros defeitos que não tinham sido previstos inicialmente. Também foi verificado que as classes de defeitos mais encontradas foram: defeito de digitação, omissão e inconsistência.

Como trabalhos futuros, deseja-se replicar este experimento com profissionais formados da área de informática, a fim de comparar o desempenho desses profissionais com os alunos de graduação. Também deseja-se utilizar outros parâmetros para este experimento, tais como: outro documento de requisitos de uma aplicação diferente e outra técnica de modelagem diferente da ESG.

REFERÊNCIAS

ABIB, J.C. ***Abordagem Goal Question Metric (GQM) para Avaliação da Qualidade de Software***. Dissertação de Mestrado. DC/UFSCar, São Carlos, SP, 1998.

BASILI, V.R.; WEISS, D.M. **Evaluation of a Software Requirements Document by Analysis of Change Data**. *In: Proceedings 5th International Conference On Software Engineering, IEEE CS Press*, California, EUA, p. 314-323, 1981.

BASILI, V.R.; GREEN, S.; LAITENBERGER, O.; LANUBILE, F.; SHULL, F. **The Empirical Investigation of Perspective-based Reading**. *Empirical Software Engineering*, n. 2, vol. 1, p. 133-164, 1996.

BELLI F, BUDNIK CJ, LINSCHULTE M, SCHIEFERDECKER I. **Testen Web-basierter Systeme mittels strukturierter, graphischer Modelle—Vergleich anhand einer Fallstudie**. In Gi jahrestagung (2). Gesellschaft für Informatik e.V.: Bonn, Germany, 2006a.

BELLI F, BUDNIK CJ, WHITE L. **Event-based modelling, analysis and testing of user interactions: approach and case study**. Software Testing, Verification & Reliability 2006b.

BELLI, Fevzi; ENDO, André Takeshi; LINSCHULTE, Michael; SIMAO, Adenilso. **A holistic approach to model-based testing of Web service compositions**. Software – Practice and Experience, 2014.

BELLI, Fevzi; NISSANKE, Nimal; BUDNIK, Christof J.; MATHUR, Aditya. **Test Generation Using Event Sequence Graphs**. University of Paderborn, Institute for Electrical Engineering and Information Technology, 2005. Disponível em: < http://adt.ivknet.de/download/papers/BNBM05_ESG_TestGen.pdf >. Acesso em: 05 de Jan. de 2015.

BERTINI, Lilian Aparecida. **Técnicas de Inspeção Aplicadas à Avaliação de Requisitos de Sistemas de Software: Um Estudo Comparativo**. Dissertação (Pós-Gradução). UNIMEP, Piracicaba, 2006.

BIFFL, B.; BED, F.; LAITENBERGER, O. **Investigating the Cost-Effectiveness of Reinspections in Software Development**. In: International Conference on Software Engineering, IEEE Computer, Toronto, Canada, p.155-164, 2001.

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: MoDeVa'06, 3rd Int. Workshop on Model Development, Validation and Verification, Genova, Italy, 2006, p. 2–15.

CIOLKOWSKI, M.; DIFFERDING, C.; LAITENBERGER, O.; MUNCH, J. **Empirical Investigation of Perspective-based Reading: A Replicated Experiment**. *Technical Report ISERN-97-13*, University of Kaiserslautern, Alemanha, 1997.

CIOLKOWSKI, M. **Evaluating the Effectiveness of Different Inspection Techniques on Informal Requirements Documents**. Master Thesis. Department of Computer Science, University of Kaiserslautern, Alemanha, Junho, 1999.

DEITEL, Paul; DEITEL, Abbey; DEITEL, Harvey; MORGANO, Michael. **Android para programadores – uma abordagem baseada em aplicativo**. Bookmam. 2013.

EBENAU, R.G.; STRAUSS, S.H. **Software Inspection Process**. 1ª edição, McGraw Hill, 1994.

EL-FAR, I. K.; WHITTAKER, J. A. **Model-based software testing**. In: Encyclopedia on Software Engineering, Wiley, 2001, p. 825–837.

ENDO, André Takeshi. **Teste baseado em máquinas de estados finitos para aplicações orientadas a serviço.** Qualificação (Doutorado). USP – São Carlos, SP. 2010.

ENDO, André Takeshi. **Model based testing of servisse oriented applications**. Tese (Doutorado). USP – São Carlos, SP. 2013.

FAGAN, M.E. **Advances in Software Inspections.** ***IEEE Transactions on Software Engineering****,* n. 7, vol. 12, p. 744-751, 1986.

GRIESKAMP, Wolfgang; KICILLOF, Nicolas; STOBIE, Keith; BRABERMAN, Victor. **Model-based quality assurance of protocol documentation: tools and methodology**. Software Testing, Verification and Reliability. 2011.

HARTMAN, A.; KATARA, M.; OLVOVSKY, S. **Choosing a test modeling language: a survey**. In: HVC'06: Proceedings of the 2nd international Haifa verification conference on Hardware and software, verification and testing, Haifa, Israel, 2007, p. 204–218.

LAITENBERGER, O. **Perspective-based Reading: Technique, Validation and Research in Future**. *Technical Report ISERN-95-01*, University of Kaiserslautern, Alemanha, 1995.

LANUBILE, F.; VISAGGIO, G. **Assessing Defect Detection Methods for Software Requirements Inspections Through External Replication**. *Technical Report ISERN-96-01,* University of Bari, Itália, 1996.

LAZZAROTTO, Belisa. **Área de Tecnologia da Informação emprega 1,3 milhões de profissionais no Brasil**. Revista Pense Empregos. Disponível em: < http://revista.penseempregos.com.br/noticia/2013/08/area-de-tecnologia-da-informacao-emprega-1-3-milhao-de-profissionais-no-brasil-4241575.html >. Acesso em: 06 de Jan. de 2015.

NUSEIBEH, B.; EASTERBROOK, S. **Requirements Engineering: A Roadmap**. *Communications of the ACM*, n. 4, vol. 10, p. 35-47, 2002.

PRESSMAN, R.S. ***Engenharia de Software***. São Paulo: Makron Books, 1995.

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

PRESSMAN, Roger S. ***Engenharia de Software***. 6 ed. São Paulo: Makron Books, 2006.

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

PRETSCHNER, A., PRENNINGER, W., WAGNER, S., et al. **One evaluation of model-based testing and its automation**. In Proceedings of the 27th International Conference on Softwar Engineering, 392–401. ACM Press, 2005.

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

TYRAN,C. K.; GEORGE, J.F. **Improving Software Inspections with Group Process Support**. *Communications of The ACM*, n. 9, vol.45, p. 87-92, 2002.

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

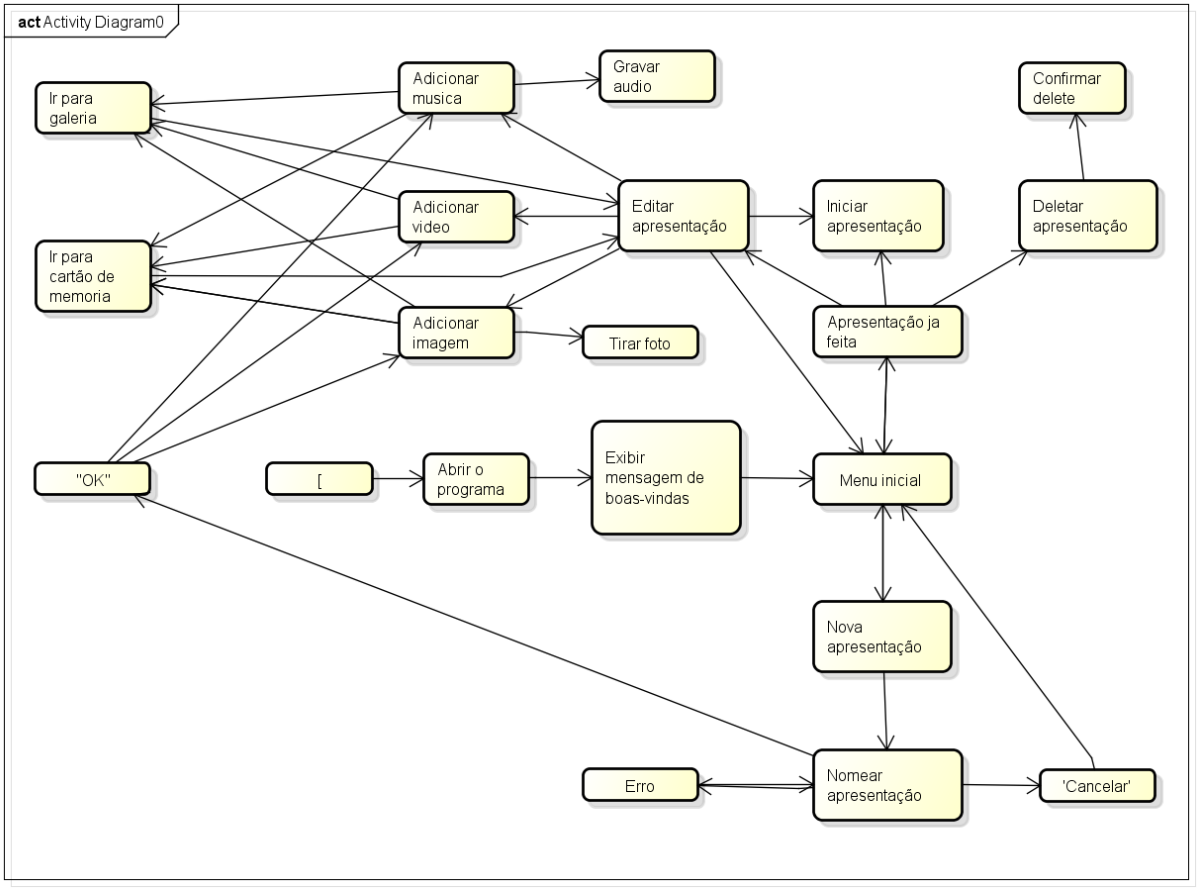
UTTING, M.; PRETSCHNER, A.; LEGEARD, B. **A taxonomy of model-based testing**. Technical Report, Hamilton, New Zealand, 2006.

WOHLIN, Claes; RUNESON, Per; HÖST, Martin; OHLSSON, Magnus C.; REGNELL, Björn; WESSLÉN, Anders. **Experimentation in Software Enginnering**. Springer: New York, 2012.

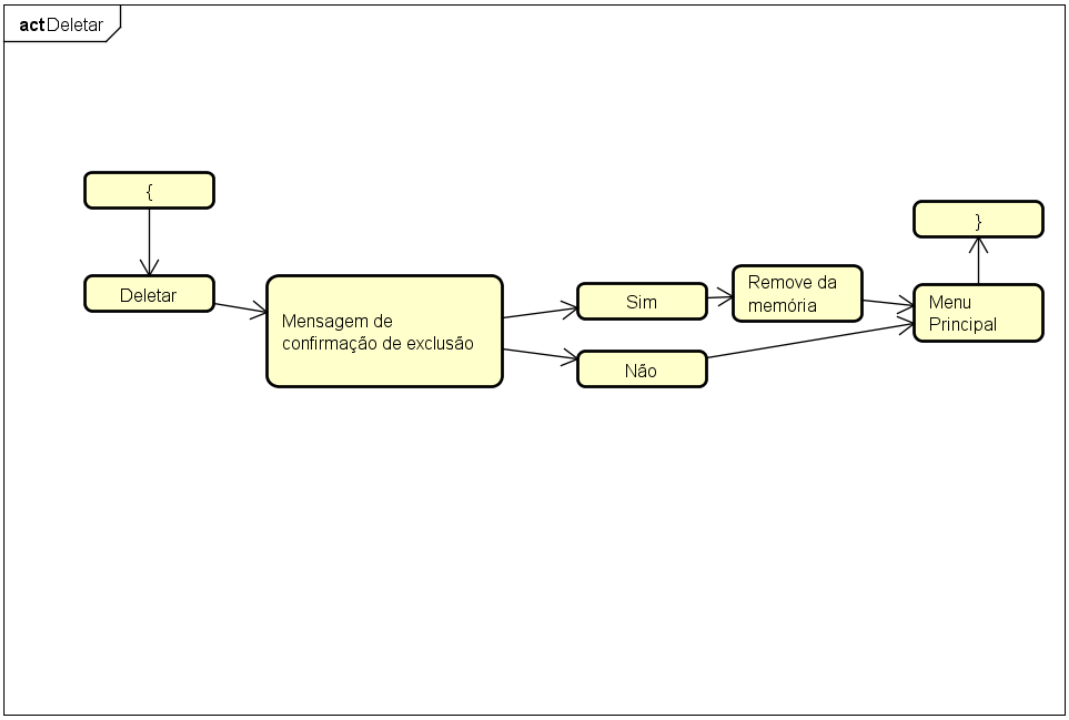
WONG, Y.K. **Use of Software Inspection Inputs in Pratice**. *Communications of the ACM*, n. 8, vol. 35, p. 725-726, 2002.

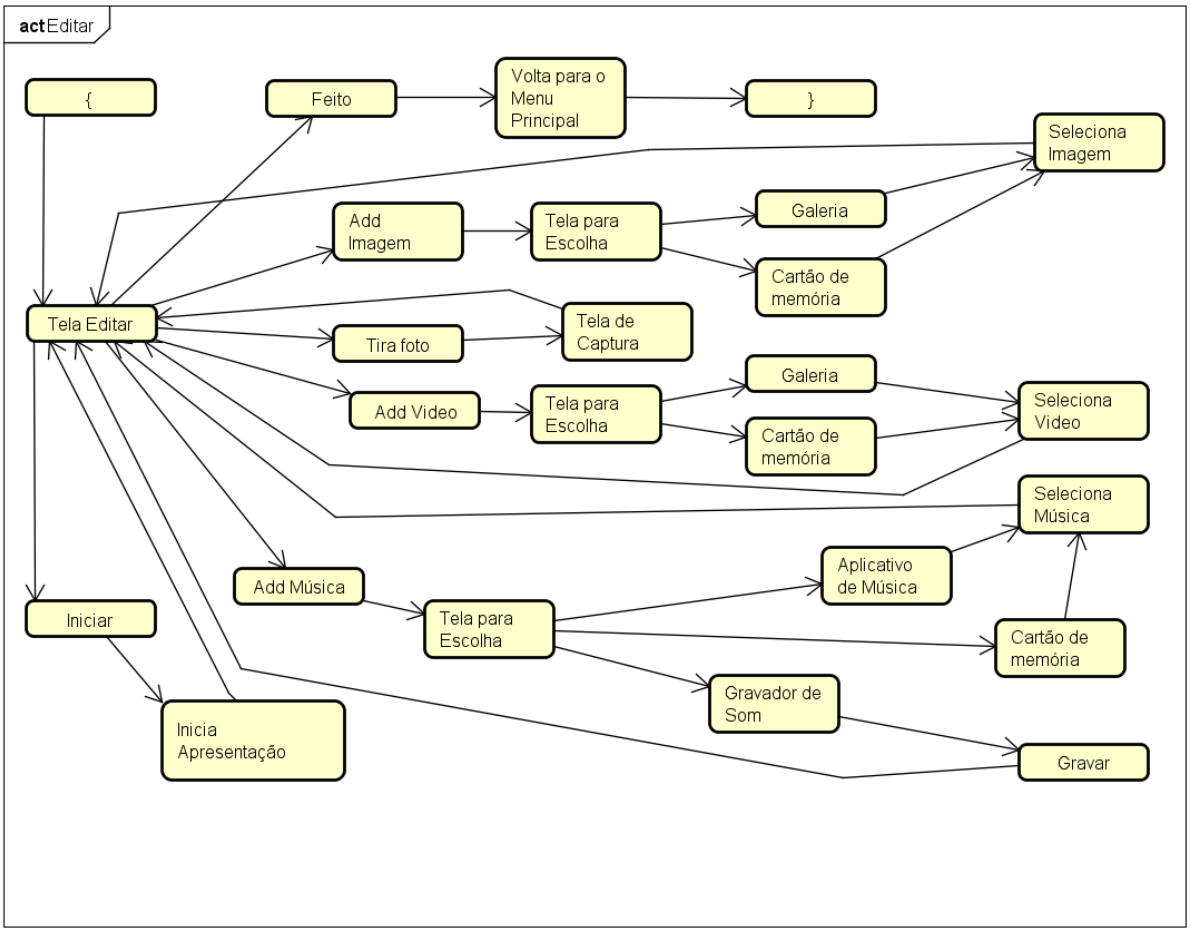
ANEXO A – MODELOS DE TESTE

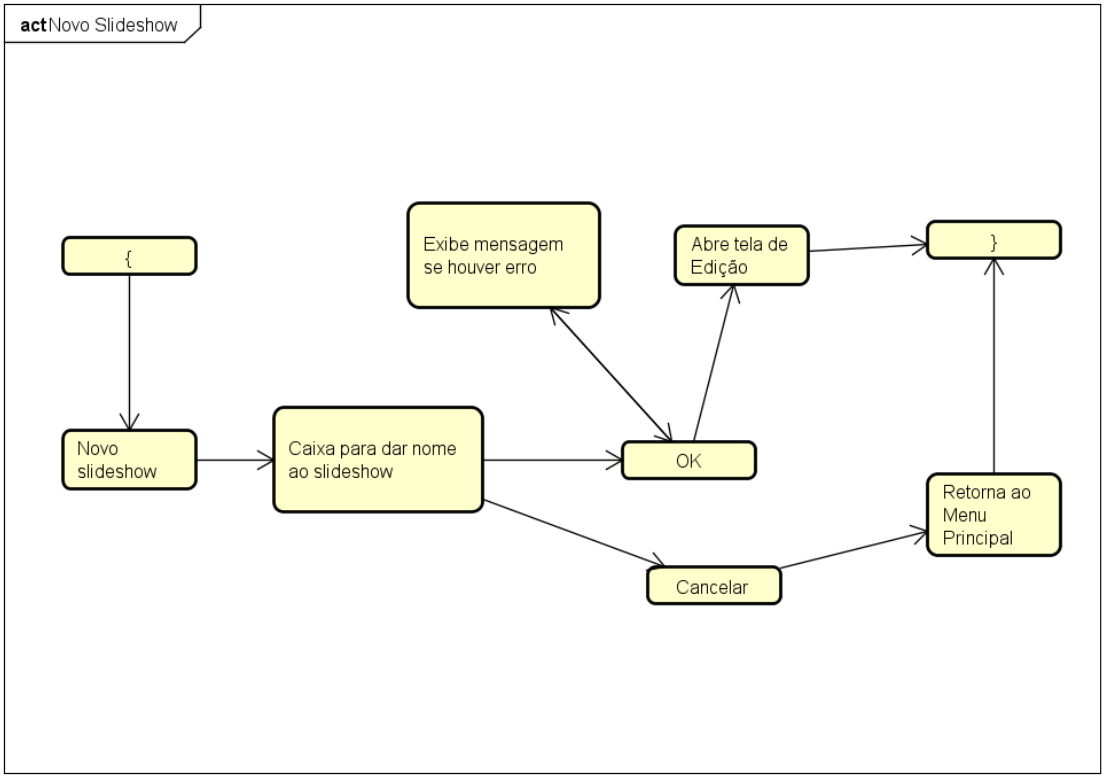
MODELO DE TESTE DA APLICAÇÃO COMO UM TODO

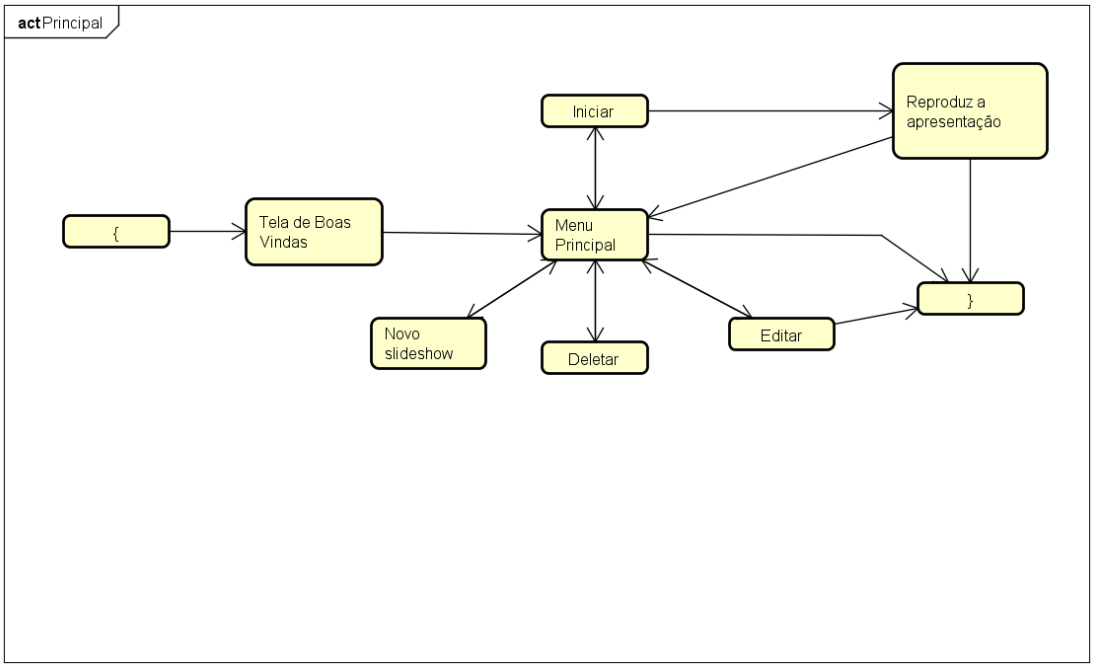


MODELO DE TESTE DE CADA FUNCIONALIDADE DA APLICAÇÃO






actNovo Slideshow
{
Exibe mensagem se houver erro
Abre tela de Edição
}
Novo slideshow
Caixa para dar nome ao slideshow
OK
Retorna ao Menu Principal
Cancelar


actPrincipal
Iniciar
Reproduz a apresentação
{
Tela de Boas Vindas
Menu Principal
}
Novo slideshow
Deletar
Editar

APÊNDICE A – DOCUMENTO DE REQUISITOS

DOCUMENTO DE REQUISITOS COM DEFEITOS DA APLICAÇÃO *ENHANCED SLIDESHOW*

VISÃO GERAL

A aplicação *Enhanced SlideShow* permite criar apresentações, podendo conter imagens, músicas e vídeos usando dispositivos *Android*, até mesmo fotos tiradas em instantes, possibilitando que sejam adicionados efeitos visuais nestas fotos. A partir dos itens selecionados, a aplicação os exibes na forma de apresentação de slides e salva o *slideshow* feito.

REQUISITOS FUNCIONAIS

R1 – A apricação deve armazenar todas as apresentações criadas pelo usuário.

R2 – A aplicação deve permitir que o usuário remova qualquer apresentação salva, selecionada por ele.

R3 – A aplicação deve permitir que o usuário edite qualquer apresentação, selecionada por ele.

R4 – A aplicação deve permitir que o usuário visualize somente algumas apresentações das apresentações criadas, selecionadas por ele.

R5 – A aplicação deve permitir que o usuário selecione quaisquer vídeos e imagens salvas na galeria ou no cartão de memória do dispositivo.

R6 – A aplicação deve permitir que o usuário selecione músicas salvas na galeria ou no cartão de memória do dispositivo, ou até mesmo, que o usuário grave um áudio no exato momento, e que este seja salvo na galeria.

R7 – A aplicação deve permitir que o usuário selecione apenas vídeos MP4 e imagens JPG salvas na galeria ou no cartão de memória do dispositivo.

R8 – A aplicação deve permitir que o usuário tire fotos com a câmera do dispositivo no exato momento, para que estas sejam salvas na galeria e utilizadas na aplicação.

R9 – A aplicação deve permitir que o usuário remova apenas imagens ou músicas já selecionadas para a apresentação.

R10 – A apresentação de um *slideshow* deve possilitar que o usuário o visualize de forma que tenha opções para adiantar, voltar e pausar a apresentação.

R11 – A apresentação de um *slideshow* deve possibilitar que o usuário visualize cada imagem ou cada foto por 5 segundos.

REQUISITOS NÃO FUNCIONAIS

R12 – A aplicação deve reproduzir todas as apresentações salvas com a mesma qualidade.

R13 – A aplicação deve responder aos botões, selecionados pelo usuário, em no máximo 1.

R14 – A aplicação deve ser executada em sistemas *Android*, independente do modelo do dispositivo.

R15 – A aplicação deve ser fácil de utilizar.

REQUISITOS DE INTERFACE

R16 – A aplicação, ao ser iniciado, deve exibir um painel de mensagem, conforme mostra a Figura 1.

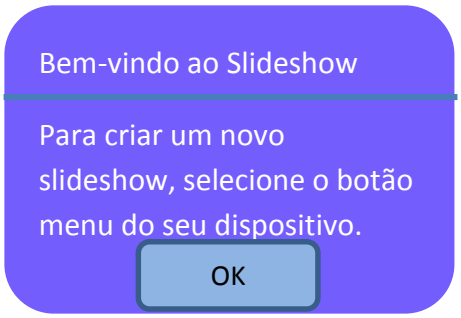


Figura 1: Tela Inicial

R17 – Ao selecionar o botão OK da tela inicial (Figura 1), a aplicação deve permitir que o usuário selecione uma apresentação salva na aplicação (Figura 2), ou que crie uma nova apresentação, ao usuário pressionar o botão menu do dispositivo.

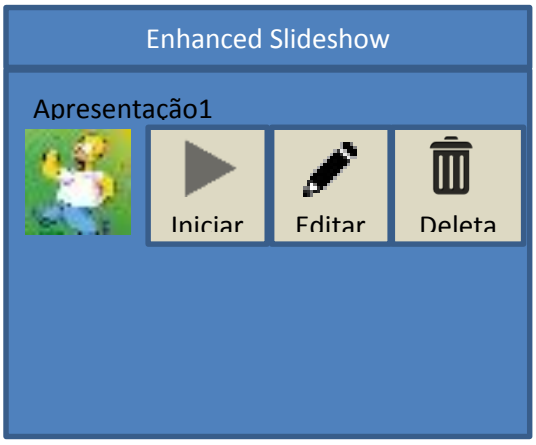


Figura 2: Tela para Selecionar *Slideshow*

R18 - Se o usuário selecionar o botão “Iniciar”, da Figura 2, a aplicação deve iniciar a apresentação do respectivo *slideshow* selecionado, passando cada imagem ou foto por um tempo determinado pelo usuário, conforme Requisitos R10, R11 e R31.

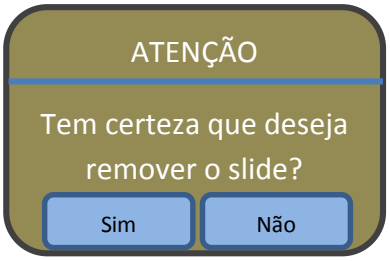


Figura 3: Tela para Deletar o *Slideshow*

R19 - Se o usuário selecionar o botão “Deletar”, da Figura 2, irá aparecer uma mensagem para o usuário confirmar a remoção, conforme mostra a Figura 3. Se o usuário selecionar o botão “Sim”, a aplicação deverá remover o *slideshow* selecionado da memória.

R20 - Se o usuário selecionar o botão “Editar”, da Figura 2, a aplicação deve abrir a tela de edição (Figura 5) para o usuário editar a apresentação do respectivo *slideshow* selecionado.

R21 – Ao usuário pressionar o botão menu do seu dispositivo, ainda na tela da Figura 2, a aplicação deve exibir uma opção para criar um novo *slideshow*, com a seguinte mensagem: “Novo *slideshow*”.

R22 – Se o usuário selecionar a opção “Novo *slideshow*”, a aplicação deve exibir a tela da Figura 4, a qual o usuário deve escrever o nome do *slideshow* que será criado.

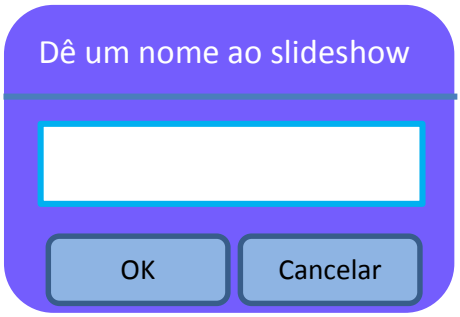


Figura 4: Tela para Dar Nome ao Slideshow

R23 – Se o usuário selecionar o botão “OK” da Figura 4, a aplicação deve seguir para a tela de edição de *slideshow* (Figura 5), se o nome dado ao *slideshow* for composto por letras e números. Caso contrário, fornece uma mensagem de erro na tela.

R24 – Se o usuário selecionar o botão “Cancelar” da Figura 4, a aplicação deve retornar a tela de selecionar um *slideshow* (Figura 2).

Figura 5: Tela para Editar *Slideshow*

R25 – Se o usuário selecionar o botão “Feito” da Figura 5, a aplicação deve salvar o *slideshow* na galeria da aplicação e voltar para a tela de selecionar um *slideshow* (Figura 2).

R26 – Se o usuário selecionar o botão “Add Imagem”, a aplicação deve exibir uma tela para que o usuário possa escolher entre uma imagem salva na galeria ou uma imagem salva no cartão de memória. Escolhido o local onde estão as imagens, a aplicação é designada a esse local e exibe as imagens existentes para que o usuário possa escolher.

R27 – Se o usuário selecionar o botão “Tirar Foto”, a aplicação deve abrir a tela correspondente à captura de imagem da câmera do dispositivo, para que o usuário possa tirar uma foto.

R28 - A aplicação deve capturar a image quando acontecer um *click* na tela, incrementando-se sequencialmente aos arquivos de exibição.

R29 – Se o usuário selecionar o botão “Add Vídeo”, a aplicação deve exibir uma tela para que o usuário possa escolher entre um vídeo salvo na galeria ou um vídeo salvo no cartão de memória. Escolhido o local onde estão os vídeos, a aplicação é designada a esse local e exibe os vídeos, se existirem, para que o usuário possa escolher. Caso contrário, não haverá opções de vídeos para escolher.

R30 – Se o usuário selecionar o botão “Add Música”, a aplicação deve exibir uma tela para que o usuário possa escolher entre uma música salva na galeria, uma música salva no cartão de memória ou a opção de gravar um som. Escolhido a primeira ou segunda opção, a aplicação é designada ao local correspondente e exibe as músicas existentes para que o usuário possa escolher. Caso o usuário escolha a terceira opção, a aplicação deve abrir a tela correspondente a gravação de som do dispositivo, para que o usuário possa gravar um som.

R31 – Conforme o usuário vai selecionando os itens (imagens, fotos, músicas ou vídeos), vão sendo exibidos, na ordem da seleção, na tela de edição (Figura 5), logo abaixo dos botões, com a opção ao lado de deletar do respectivo item, conforme mostra a Figura 6.

Figura 6: Tela para Editar *Slideshow* com Itens Selecionados

R32 – Se o usuário selecionar a opção “Deletar” de algum item do *slideshow*, a mensagem de aviso para confirmar a remoção será exibida (Figura 3).

R33 - A aplicação deve exibir a apresentação sem seguir a ordem dos itens escolhidos para compor o *slideshow*.