Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Colegiado de Sistemas de Informação

Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS

Arthur Perdigão Martins Ferreira

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Euler Horta Marinho

Março, 2016
João Monlevade/MG

Arthur Perdigão Martins Ferreira

Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS

Orientador: Euler Horta Marinho

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação

Universidade Federal de Ouro Preto

João Monlevade

Março de 2016

Arthur Perdigão Martins Ferreira

Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS/ Arthur Perdigão Martins Ferreira. – João Monlevade, 21 de março de 2016-

44 p.: il. (algumas color.) ; 30 cm.

Orientador: Euler Horta Marinho

Monografia (graduação) – Universidade Federal de Ouro Preto, 21 de março de 2016.

1. Teste de Software. 2. Aplicações Android. I. Euler Horta Marinho. II. Universidade Federal de Ouro Preto. III. Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS

CDU 02:141:005.7

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS

Arthur Perdigão Martins Ferreira

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação aprovada pela Banca Examinadora abaixo assinada:

Euler Horta Marinho
Mestre em Computação
Orientador
DECSI - UFOP

Eduardo da Silva Ribeiro
Mestre em Ciência da Computação
Examinador
DECSI - UFOP

Igor Mazetti Pereira
Mestre em Ciência da Computação
Examinador
DECSI - UFOP

João Monlevade, 21 de março de 2016

ATA DE DEFESA

Aos vinte e um dias do mês de março de 2016, às 18 horas, na sala C203, Laboratório de Engenharia de Software do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pelo aluno **Arthur Perdigão Martins Ferreira,** sendo a Comissão Examinadora constituída pelos professores: Prof. Me. Euler Horta Marinho, Prof. Me. Eduardo da Silva Ribeiro e Prof. Me. Igor Muzetti Pereira.

O candidato apresentou a monografia intitulada: *"Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS"*. A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final.

Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo graduando.

João Monlevade, 21 de Março de 2016.

Prof. Me. Euler Horta Marinho
Professor Orientador/Presidente

Prof. Me. Eduardo da Silva Ribeiro
Professor Convidado

Prof. Me. Igor Muzetti Pereira
Professor Convidado

Arthur Perdigão Martins Ferreira
Graduando

Agradecimentos

Agradeço primeiramente a Deus por guiar os meus passos e ser base e estrutura de tudo em minha vida, por me dar forças nos momentos de angústia, conforto nos momentos de desespero, pela fé, sabedoria, inteligência e persistência que me trouxeram até este momento.

Aos professores que foram base para a minha formação como pessoa e profissional, que me agregaram ainda mais conhecimento e valores, em especial ao orientador Euler Horta Marinho pela confiança, orientação, sabedoria, amizade e competência, e pela presença durante todo o desenvolvimento do trabalho, pelo apoio e força durante a reta final do curso.

Agradeço aos meus pais Antônio Alves Ferreira e Angelina Rita Perdigão Martins Ferreira e meus familiares pela paciência, compreensão, apoio e dedicação durante todo o meu curso, pois sem eles não chegaria onde cheguei.

A Minha namorada Bárbara Monteiro, pelo companheirismo, compreensão e carinho durante grande parte do meu curso, por todo apoio e ajuda oferecida.

A Universidade Federal de Ouro Preto(UFOP), por ser fonte de conhecimento e especialização.

"Deus disse através de Cristo:,
Se tens fé, cumpre saberes de que tudo é possível àquele que crê.
"Disse-lhe Jesus: Se podes alguma coisa!..Tudo é possível ao que crê."
(Bíblia Sagrada, Marcos 9:23)

Resumo

Ferramentas de apoio ao teste de software são muito importantes. Essas ferramentas auxiliam a automação de atividades de testes, a fim de diminuir os custos e esforços. O presente trabalho apresenta um estudo de ferramentas para o teste de Aplicações Android, no contexto do Laboratório LEDS da Universidade Federal de Ouro Preto. As principais necessidades do cliente são levantadas, a partir do método SQFD (HAAG; RAJA; SCHKADE, 1996) e do método proposto por Veloso et al. (2010), que cruza os requisitos necessários ao cliente com os requisitos do produto. A análise leva em consideração também os requisitos básicos propostos pelo LEDS, em busca da melhor ferramenta entre as 3 ferramentas selecionadas *(MonkeyTalk, Selendroid e Sikuli)* de código aberto e de licença gratuita, que foram selecionadas a partir da coleta e ensaios de teste que ajudaram a determinar a ferramenta mais adequada ao estudo de caso. Além disso, é realizada uma discussão envolvendo alguns tipos de sensores utilizados por Aplicações Android e como são tratadas nas ferramentas selecionadas.

Palavras-chaves: Teste de Software; Ferramentas de Teste; Aplicações Android; Sensores.

Abstract

Software test tools are very important. These tools help the automation of software test activities, in order to decrease the costs and efforts. This work presents a study of testing tools for Android Applications, in the context of LEDS laboratory in Federal University of Ouro Preto. The main needs of the client are identified, from SQFD method (HAAG; RAJA; SCHKADE, 1996) and the method proposed by Veloso et al. (2010), which crosses the customer necessary requirements with the product requirements. The analysis also takes into consideration the basic requirements proposed by the LEDS in search of the best tool among the three selected tools (MonkeyTalk, Selendroid and Sikuli) open source and license free, which were selected from the collection and testing trials that helped determine the most appropriate tool to the case study. In addition , it held a discussion involving some types of sensors used by Android applications and how they are treated in the selected tools.

Key-words: Software Test; Test tools; Android Applications; Sensors.

Lista de ilustrações

Lista de tabelas

Sumário

Introdução

A construção de software não é uma tarefa simples. Pelo contrário, pode se tornar bastante complexa, dependendo das características e dimensões do sistema a ser criado. Por isso, ela está sujeita a diversos tipos de problemas que acabam resultando na obtenção de um produto diferente do que se esperava(DELAMARO; MALDONADO; JINO, 2007). Para evitar a propagação de falhas no desenvolvimento de software, uma das principais atividades consiste no teste do software, que não se restringe somente ao produto final, mas pode ser aplicado durante todo o desenvolvimento. Dessa maneira, a atividade de teste contribui para a melhoria da qualidade do software.

Segundo (PRESSMAN, 2005), "O Teste de Software é um elemento crítico da garantia de qualidade de software e representa a revisão final da especificação, projeto e geração de código". Vale ressaltar que o processo de teste de software exige tempo e recursos humanos e computacionais (SHAMSODDIN-MOTLAGH, 2012). Com o objetivo de reduzir os custos envolvidos no teste, surgiram diversas ferramentas com o intuito de apoiar a automação de uma ou mais etapas do processo de teste. É importante destacar que a escolha de ferramentas de testes inadequadas pode trazer desperdício de tempo, além de poder influenciar negativamente no sucesso do produto (VELOSO et al., 2010). Portanto, são necessários estudos que apóiem a avaliação das ferramentas disponíveis, auxiliando a tomada de decisão dos desenvolvedores de software.

Mobilidade se refere ao uso de dispositivos móveis portáteis que tem como capacidade fornecer serviços para que usuários se conectem, obtenham dados e os forneçam a outros usuários e sistemas (LEE; SCHNEIDER; SCHELL, 2005). Com a popularização dos dispositivos móveis, a demanda pelo desenvolvimento de aplicações e serviços móveis aumentou (BUTLER, 2011). O desenvolvimento dessas aplicações pode ser orientado para diferentes classes de dispositivos, tais como, smartphones, tablets, consoles portáteis etc. Apesar dos dispositivos possuírem funcionalidades capazes de fornecer serviços em qualquer hora e lugar, eles ainda possuem restrições, quando comparados com os *desktops*, no que diz respeito às particularidades de desenvolvimento devido às limitações de processamento, tamanho e resolução de tela, uso de dados de sensores etc (DANTAS, 2009).

Existem várias plataformas e ferramentas que podem ser utilizadas para a criação de aplicações ricas em recursos para dispositivos móveis, oferecendo uma melhor experiência e satisfação das necessidades do usuário final. Algumas plataformas de desenvolvimento para smartphones têm atraído a atenção de consumidores e desenvolvedores como o Android da Google, o iOS da Apple e o Windows Phone da Microsoft. Esses sistemas vêm se transformando, atualmente, nos principais SO (Sistemas Operacionais) para telefones

móveis sendo grandes concorrentes entre si, não somente em sua arquitetura, mas também em suas funcionalidades (COSTA; FILHO; DUARTE, 2012). Atualmente, mais de 3 bilhões de pessoas possuem algum tipo de aparelho celular e já utilizam os aparelhos como principal forma de acesso à Internet, e usuários em todo mundo têm até substituído os computadores em diversos casos, inclusive no mercado coorporativo. Os usuários estão em busca de aplicações móveis que permitem o acesso a vários recursos, tais como, câmera, GPS, acesso à WWW, e-mail, sistemas bancários, jogos etc (LECHETA, 2013).

Neste contexto, o principal objetivo do trabalho é realizar uma avaliação das ferramentas disponíveis para o teste de Aplicações Móveis, tendo como cliente o LEDS (Laboratório de Engenharia e Desenvolvimento de Sistemas) da UFOP. A avaliação das ferramentas será realizada com base no Método SQFD (*Software Quality Function Deployment*) (HAAG; RAJA; SCHKADE, 1996), que auxilia na identificação e priorização das necessidades do cliente e no método de avaliação de ferramenta proposto por Veloso et al. (2010) . Também será discutido de que maneira as ferramentas de testes tratam as particularidades das Aplicações Móveis, com ênfase nos aspectos de tratamento de dados de sensores, que contribuem para a etapa de teste, durante o desenvolvimento de Aplicações Android, utilizando ferramentas que permitem automatizar estes testes e torná-los mais precisos.

1 Objetivos do Trabalho

1.1 Objetivo Geral

O objetivo geral deste trabalho é a análise das ferramentas de apoio ao teste de Aplicações Móveis, com foco ao suporte dado pelas mesmas aos dados de sensores, acelerômetros, GPS, auxiliando o cliente na escolha da ferramenta mais apta.

1.2 Objetivo Específico

O objetivo específico deste trabalho é a realização de um estudo envolvendo a análise de ferramentas de apoio ao teste para Aplicações Android, utilizando o SQFD e o Método de Avaliação proposto por Veloso et al. (2010), no contexto do desenvolvimento do LEDS.

1.3 Organização do Trabalho

O presente trabalho é apresentado em 3 partes, com um total de 7 capítulos, em que o conteúdo é organizado da seguinte forma:

A parte I apresenta os capítulos de 1 a 5, que abordam os conceitos sobre Teste de Software, enfatizando a parte de Desenvolvimento Android, a apresentação do Método SQFD e os trabalhos relacionados com o tema.

A parte II dispõe dos capítulos 6 e 7, relatando o desenvolvimento do trabalho e Aplicação do Método SQFD no estudo de caso do LEDS, e o método de avaliação das ferramentas de apoio ao teste.

A parte III, apresenta a análise e discussão dos resultados obtidos, as considerações finais e os trabalhos futuros.

Parte I

Revisão Bibliografica

O teste é uma das etapas do desenvolvimento de software, geralmente realizada pelo testador, que consiste na execução de um programa utilizando algumas entradas previamente estabelecidas e na verificação do comportamento apresentado pelo software. Segundo (MYERS, 2004), o teste é o processo de execução de um programa com a intenção de encontrar erros. Assim, o teste bem sucedido é aquele que consegue determinar casos de teste para os quais o programa em teste falhe.

(MYERS, 2004) refere-se a erros como sendo problemas na especificação de requisitos do software, uma tradução incorreta ou omissão de um requisito. O Erro significa aquilo que não é correto. Para identificar a ocorrência de erros, existem os testes de usuário, como alfa, beta, de aceitação, testes de carga, stress, etc. E temos os que são conhecidos como testes de desenvolvimento em que a atividade de teste é dividida em três etapas distintas, tais como o teste de unidade, integração e de sistemas (DELAMARO; MALDONADO; JINO, 2007).

O teste de unidade tem como foco a menor parte testável de um software conhecida como unidade. Essa pode ser interpretada como sendo uma classe, um método ou uma função. Esse tipo de teste permite a detecção de erros em estruturas de dados e pode ser realizado durante o desenvolvimento pelo próprio programador.

Após as unidades terem sido testadas isoladamente, é realizado o teste de integração que tem o objetivo de verificar a interação entre as unidades do software. O teste de integração é executado pela equipe de desenvolvimento, pois para seu planejamento é necessário o conhecimento da arquitetura do sistema.

O teste de sistema, por sua vez, acontece após a completa integração do sistema, verificando se as funcionalidades estão implementadas de acordo com os requisitos do cliente.

2 Teste de Sistema

Conforme dito anteriormente, o teste de sistema é realizado após a integração do sistema, e visa identificar erros nas funcionalidades e na avaliação das características gerais de qualidade. Tem-se na literatura, várias abordagens relativas ao teste de sistema, segundo (PRESSMAN, 1997). Dessa forma, o teste de sistema abrange tanto os requisitos funcionais quanto os não-funcionais.

Os testes de requisitos não-funcionais verificam características do software que não se relacionam diretamente com a funcionalidade. Essas características são discutidas na norma ISO/IEC 25010(ISO, 2010). Essa norma define um modelo de qualidade de produto de software, composto de características e subcaracterísticas que se manifestam externamente quando o software é utilizado e quando envolvem aspectos estáticos que podem ser obtidos por meio de medidas internas do software.

O modelo de qualidade de produto de software, caracteriza a qualidade do produto em oito características, cada qual é decomposta em subcaracterísticas, conforme apresentado na Figura 1.

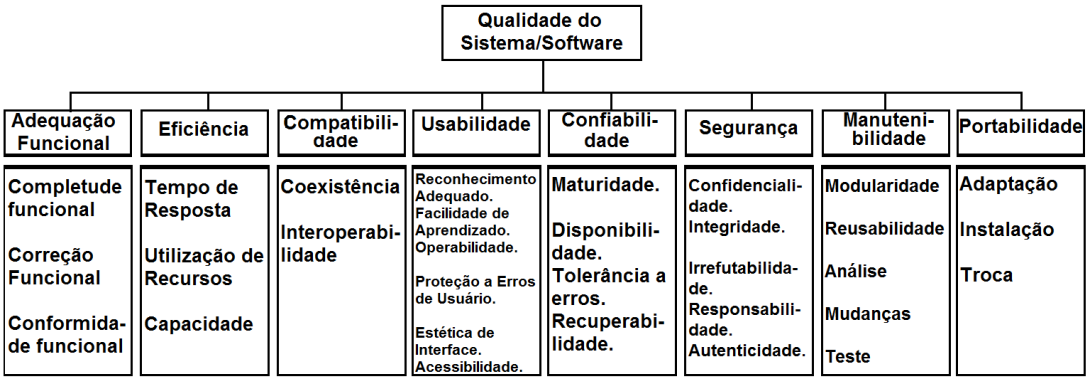

Figura 1 – Propriedades do modelo de qualidade de produto de software

De acordo com (DELAMARO; MALDONADO; JINO, 2007), o teste funcional é uma técnica que visa projetar casos de teste nos quais o programa ou sistema é considerado uma caixa preta. Nessa técnica, os detalhes de implementação não são considerados e o software é avaliado segundo o ponto de vista do usuário, a partir das funcionalidades definidas. Para a execução dos testes, são fornecidas entradas e saídas esperadas estabelecidas a partir da especificação do software. As saídas esperadas são confrontadas com as saídas obtidas e um veredito para o teste é emitido (passou ou falhou).

A fim de que o teste funcional flua da maneira correta, é necessário que a especificação esteja de acordo com os requisitos do usuário. Entretanto, como é inviável

considerar todas as entradas possíveis (teste exaustivo), critérios devem ser utilizados para o planejamento dos testes. Tais critérios dividem o domínio de entrada do programa em classes de dados, das quais os casos de teste são derivados(PRESSMAN, 2002). A divisão é feita a partir de classes de equivalência válidas e inválidas, em busca de produzir casos de teste que aumentem a chance de descoberta de erros e que contribuam para a redução do total de casos necessários.

O particionamento do domínio é feito para cada classe de equivalência, em que um dos elementos consiga representar toda a classe em que a ferramenta está sendo testada, independente do elemento escolhido na classe. Se um dos casos da classe de equivalência apresentar algum erro ao se testar, conclui-se que qualquer outro caso de teste nesta classe está sujeito a apresentar um erro semelhante.

Segundo (PRESSMAN, 1997), a definição das classes de equivalência é um processo heurístico e o critério exige que seja derivado um conjunto mínimo de casos de teste para aplicar as classes de equivalência válidas e um caso de teste para cada classe inválida. Os casos específicos por classe de equivalência inválida são de extrema importância, pois determinadas condições de entrada podem “mascarar“ a verificação de determinados erros.

O critério de Análise de Valor-limite complementa o Particionamento de Classes de Equivalência por meio de casos de teste que procuram aumentar a chance de que erros sejam encontrados. Valores-limite referem-se a situações que ocorrem dentro de uma classe de equivalência, acima ou abaixo dela. Nesse critério, o domínio de saída do programa também é particionado e auxilia na derivação de outros casos de teste, que devem ser verificados em cada um dos limites das classes.

3 Desenvolvimento de Aplicações Android

Android é uma plataforma que permite o desenvolvimento de aplicativos para dispositivos móveis, tais como smartphones e tablets. O Android foi desenvolvido pela Google e, posteriormente, seu desenvolvimento tem sido administrado pela OHA *(Open Handset Alliance)*, uma organização que une várias empresas com o objetivo de criar padrões abertos para dispositivos móveis. A disponibilidade de ferramentas em um SDK *(Software Development Kit)* gratuito, com versões para os principais sistemas operacionais (OSX, Windows e Linux) atrai muitos desenvolvedores para essa plataforma.

Aplicações Android, geralmente, são desenvolvidas na Linguagem Java. O Android SDK inclui diversas bibliotecas e ferramentas (GOOGLE; ALLIANCE, 2016a), que podem ser utilizadas em várias fases do desenvolvimento. Dentre essas, o SDK disponibiliza emuladores que permitem a execução e o teste de aplicações em diversas versões do Android e a configuração de inúmeras particularidades físicas e de hardware (tamanho de tela, processador etc). O Android Studio, baseado na *IntelliJ IDEA*, é a IDE atualmente suportada pela Google. Anteriormente, o Eclipse era estendido com o ADT *(Android Development Tools) Plugin* e o Android SDK (K19, 2012).

Os Dispositivos Móveis são mais que simples dispositivos de comunicação com conexão à Internet. Eles fornecem uma ampla gama de recursos envolvendo microfones, câmeras e sensores como acelerômetros, bússolas e medidores de temperatura e luminosidade. Um sensor é uma designação comum de dispositivos elétricos, eletrônicos, mecânicos ou biológicos, capaz de responder a estímulos da natureza física (temperatura, pressão, umidade, velocidade, aceleração, luminosidade, etc.). Desta forma, os sensores proporcionam uma série de novas possibilidades de interação com dispositivos, tais como realidade aumentada e entrada baseada em movimentos (DEVMEDIA, 2016).

O Android fornece suporte a uma série de sensores, na documentação oficial do Android para desenvolvedores(GOOGLE; ALLIANCE, 2016c) em que categorizam os sensores presentes na plataforma Android em 3 categorias:

- Sensores de ambientação: Captam dados em relação ao ambiente externo, tais como a temperatura do ar, a pressão atmosférica, iluminação do local e umidade. A categoria inclui barômetros, fotômetros e termômetros.

- Sensores de posicionamento: Detectam o posicionamento físico do dispositivo móvel, envolve os sensores de orientação, GPS e magnetômetros.

- Sensores de movimentação: estes sensores medem as forças de aceleração e as forças

rotacionais através de 3 eixos. Esta categoria inclui os acelerômetros, os sensores de gravidade, o giroscópio e os sensores de vetor rotacional.

Atualmente, estes sensores são bastante explorados nas aplicações. Principalmente os sensores de movimentação, em que o estudo se baseia por estarem presentes na maioria das Aplicações Móveis, os sensores testados e analisados nesta categoria são: O Acelerômetro, que é usado pelo próprio sistema operacional para alterar o modo de visualização da tela, de **portrait**(retrato) para **landscape**(paisagem) e vice-versa, e O Giroscópio, que opera em conjunto, e através da força da aceleração em m/s aplicada ao dispositivo em três eixos físicos (x, y e z), incluindo a força da gravidade, como demonstrado pela Figura 2, pode-se ver como as interações do usuário trabalham em conjunto com o sensor em seu funcionamento.

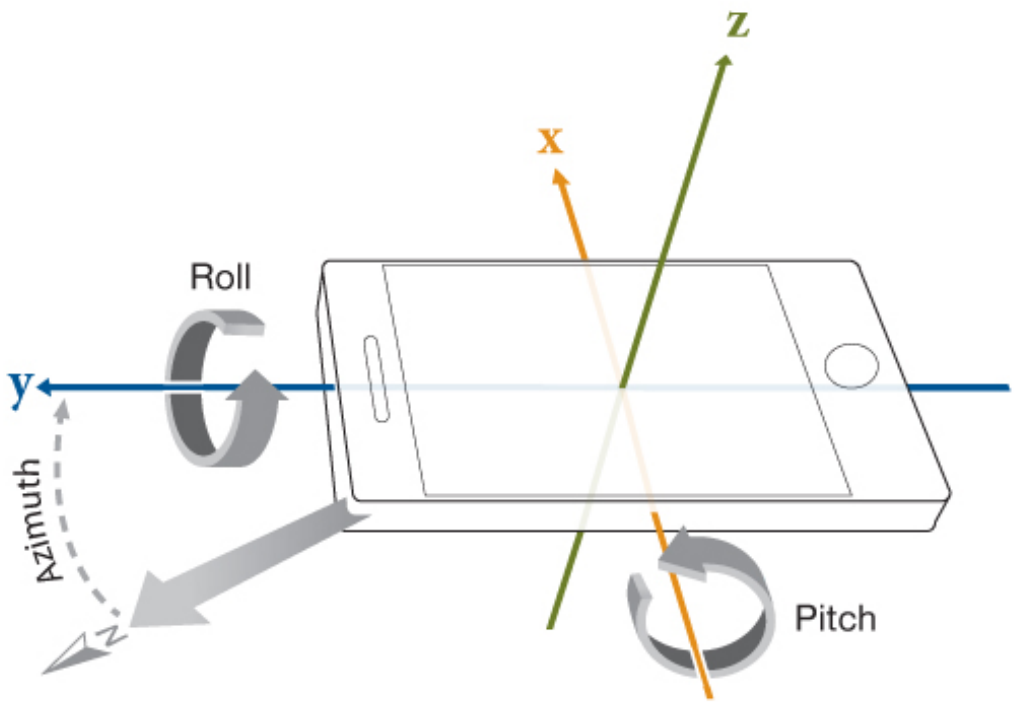

Figura 2 – Exemplo do Sensor de Movimentação (TEAM, 2014)

O Giroscópio também desempenha seu papel em torno de cada um dos 3 eixos, analisando a rotação e as interações do usuário em rad/s, detectando as mudanças de movimentação no Dispositivo Móvel.

No contexto de jogos, os sensores de movimentação também desempenham um papel muito importante, podendo ser responsáveis por controlar os movimentos dos personagens. Isto se torna particularmente útil, tendo em vista que muitos smartphones e tablets não possuem um teclado físico. Deste modo, o acelerômetro é um tipo de sensor que torna possível a interação com o usuário. Outro exemplo é o uso do sensor de proximidade. Ele

é fundamental para a redução do consumo de bateria do aparelho durante as ligações, pois desabilita a tela do dispositivo quando o usuário o aproxima do rosto (DEVMEDIA, 2016).

4 Método SQFD

O Método SQFD é uma adaptação do QFD *(Quality Function Deployment)*. O QFD é um método utilizado para traduzir as necessidades dos clientes em requisitos técnicos de software (AKAO, 1996). A adaptação do QFD para o desenvolvimento de software é o SQFD*(Software Quality Function Deployment)*, que busca melhorar o desenvolvimento de software, aplicando técnicas de melhoria de qualidade durante a especificação e análise de requisitos Haag, Raja e Schkade (1996). Assim, para avaliação das ferramentas de testes, este trabalho utiliza o SQFD, composto por cinco fases que serão descritas na próxima subseção.

4.1 Fases do SQFD

A primeira fase consiste na realização do levantamento dos requisitos, ou seja, as qualidades exigidas pelo cliente para determinada aplicação. Pode ser feita através de entrevistas, enquetes ou até mesmo técnicas de levantamento de ideias como o *Brainstorm*, levantando junto aos clientes as principais necessidades que devem ser identificadas, através de questionamentos de como determinada aplicação poderia funcionar corretamente de acordo com as dificuldades enfrentadas pelo cliente. Os requisitos identificados são organizados em grupos hierárquicos, de acordo com a similaridade de conteúdos.

A Fase 2 envolve a cooperação com os usuários do produto, neste caso os testadores, nos quais os requisitos são convertidos em especificações técnicas mensuráveis, que são registradas em uma tabela denominada Tabela de Desdobramento dos Elementos da Qualidade. Essas correspondem às funções que as ferramentas de testes para aplicações móveis devem conter para atender aos requisitos levantados que estão descritas em anexo para atender aos requisitos de geração automática de testes.

Na fase 3, através de perguntas direcionadas ao cliente, é criada a matriz de correlação que identifica e atribui pesos aos relacionamentos entre os requisitos do cliente e as especificações técnicas do produto. Assim, a equipe de teste deve se reunir para criar a casa da qualidade, na qual os pesos são atribuídos em um grau de correlação, podendo ser “Possível”(1), “Fraco”(3) ou “Forte”(9) “Inexistente”(0). Quando há muitos clientes envolvidos neste processo, é importante estabelecer um consenso quanto à intensidade dos relacionamentos.

Na fase 4, é realizada a priorização dos requisitos levantados. Com a participação do cliente, que participou da identificação de necessidades, pesos são atribuídos para cada requisito, em uma escala de um (1) a três (3), sendo o peso máximo o mais importante.

Em seguida, obtém-se a média dos valores atribuídos, usada para atribuir o grau de importância ao requisito em questão.

E por fim, na fase 5, as especificações técnicas são priorizadas com base no somatório das multiplicações dos pesos dos requisitos pelo grau de intensidade da correlação entre as fases levantadas dos requisitos e especificações técnicas e somados os resultados obtidos da multiplicação dos requisitos prioritários do cliente pelos valores de correlação gerados entre estes e as especificações técnicas do produto. Estes pesos brutos de prioridades para as especificações técnicas do produto são, geralmente, convertidos em porcentagem total dos pesos brutos de prioridade.

5 Trabalhos Relacionados

No domínio de Aplicações Móveis, o trabalho de Santos et al. (2013) aborda a avaliação de ferramentas de teste usando o Método DMADV derivado da metodologia Seis Sigma, para avaliação de ferramentas de testes para Sistemas de Informação Móvel, a partir do método de critérios proposto por Veloso.

Em Dórea et al. (2008), sete ferramentas de testes foram avaliadas empiricamente com respeito a critérios como desempenho, portabilidade, interoperabilidade e flexibilidade. Além disso, foi realizado um levantamento por meio de questionário com algumas empresas onde foi constatado que as organizações requerem ferramentas que apresentem alto grau de desempenho e eficiência.

No trabalho de Wang e Offutt (WANG; OFFUTT, 2009), os autores compararam três ferramentas para teste de unidade. Esse trabalho consistia em injetar falhas dentro das classes na linguagem Java através de uma ferramenta de mutação. Em seguida, um conjunto de dados de testes de unidade foi gerado manualmente e outros três foram gerados automaticamente pelas ferramentas. Logo, o objetivo era verificar quais conjuntos de dados de testes geravam melhores dados para encontrar as falhas injetadas nas classes. Os resultados mostraram que as ferramentas para teste de unidades ainda apresentam deficiência para a detecção de falhas.

Em (VELOSO et al., 2010), por sua vez, os autores apresentaram uma abordagem baseada em SQFD (HAAG; RAJA; SCHKADE, 1996) para analisar e comparar ferramentas de teste. O processo de avaliação se iniciou com o levantamento de requisitos e aspectos técnicos que seriam necessários em uma ferramenta de teste. Nesse trabalho, foram avaliadas três ferramentas gratuitas voltadas ao teste funcional e três direcionadas ao teste de desempenho/estresse. Dentre as ferramentas de teste funcional, a que se mostrou mais adequada foi a Selenium e, dentre as ferramentas de teste de desempenho, a que obteve melhor resultado foi a *Webload*.

Borjesson e Feldt (BORJESSON; FELDT, 2012) comparam duas ferramentas (uma comercial e outra *open source*) que automatizam testes de GUI (*Graphical User Interface*). Os testes com essas ferramentas foram conduzidos em um ambiente industrial, em um sistema real ou crítico. Foi avaliada, principalmente a capacidade de automatizar os testes, antes conduzidos manualmente. Após executar vários casos de testes com ambas as ferramentas, os autores obtiveram êxito no uso das duas ferramentas com um ganho de tempo de 78%.

Parte II

Método

Inicialmente foram realizadas pesquisas na Internet e na literatura abordando ferramentas de apoio ao teste, para identificação de ferramentas. A partir desse levantamento, foram identificadas 56 ferramentas. Para cada ferramenta, foram levantadas características tais como: o ambiente de execução da ferramenta, a linguagem utilizada para a ferramenta, o tipo de licença (comercial, gratuita ou open-source), data da primeira e última versão, o link para obtenção da ferramenta, dentre outras observações relevantes exibidas no trecho da Tabela 1.

Nome da Ferramenta	Ambiente de Execução	Linguagem	Licença/Obtenção	Data Primeira Versão	Data Última Versão	Link para Download	Observações
Robotium	Android Studio/Eclipse	Java	Open-Source	-	01/01/2014	https://code.google.com/p/robotium/wiki/Downloads?tm=2	Testa apenas Aplicativos Android
MonkeyRunner	Android Studio/Eclipse	Java	Open-Source		-	https://github.com/lvc/abi-compliance-checker	Testa apenas Aplicativos Android
Ranorex	iOS, Android e Windows 8 e web	Java	Closed-Source / Free TRIAL / Pago		5.03 24/03/2015	http://www.ranorex.com/	Testa Android, iOS, web
Appium	Windows, Linux, los, Android	Java, pythun, ruby....	Open-Source	0.18x	1.3.7beta	http://appium.io	Testa Android e los - Testes OK
UI Automator	Android	Java Library				http://developer.android.com/tools/testing-support-library/index.html	
MonkeyTalk	iOS and Android	Java	Free		22/12/14 MonkeyTalk Professional Edition 2.0.10.bet	http://www.cloudmonkeymobile.com/monkeytalk	Testes OK
Testing with frank	iOS	Java	Open-Source			http://www.testingwithfrank.com/	

Tabela 1 – Trecho extraído da tabela completa de seleção de ferramentas

Após a pesquisa e vários ensaios executados nas ferramentas de teste identificadas, foram seleciondas ferramentas que atendiam às restrições do cliente, ou seja, ferramentas *open-source*(código-aberto), licença gratuita, que possuem documentação abrangente para uso e instalação, e que tratam o uso de sensores da categoria de sensor de posicionamento: GPS, e os sensores da categoria de movimentação: Acelerômetro e Giroscópio que contribuem para as Aplicações que são desenvolvidas pelo LEDS. O estudo de caso detalhado e a aplicação do Método SQFD está aprofundado nas próximas seções.

6 Aplicação do SQFD ao Estudo de Caso LEDS

(VELOSO et al., 2010) utilizam 67 critérios para avaliação de ferramentas os quais também foram utilizados por (CALDEIRA, 2014). Entretanto, esses critérios não são específicos para ferramentas de apoio ao testes de Aplicações Móveis. Assim, foi realizada a validação desses critérios junto ao cliente. Esse trabalho também apresenta uma discussão da capacidades das ferramentas para o tratamento de dados de sensores.

7 Seleção das ferramentas

Após a análise embasada nas restrições descritas pelo cliente, três ferramentas (Figura 3), foram selecionadas e testadas.

Figura 3 – Ferramentas Escolhidas

A partir desta seleção, o estudo procurou identificar a ferramenta que mais atende aos requisitos do cliente. A seguir apresenta-se uma breve descrição de cada uma das ferramentas selecionadas:

- ***MonkeyTalk***: É uma ferramenta *Open-Source* para testes automatizados que possui linguagem e IDE próprios. A ferramenta articula-se com qualquer dispositivo conectado ao computador ou com o emulador de dispositivos. A ferramenta trabalha com linguagem de alto nível e possibilita a geração de scripts através de um mecanismo de captura.

- ***Selendroid***: É uma ferramenta de teste automatizado para múltiplos tipos de Aplicações Android: Nativas e Híbridas. É uma ferramenta derivada da Selenium que testa aplicações desktop e da Web. O teste é executado na plataforma do Selendroid através do navegador, através de dispositivos Móveis conectados ao computador ou através de emuladores.

- ***Sikuli***: A Sikuli é uma ferramenta para automatizar e testar interfaces gráficas de usuário (GUI) usando imagens obtidas a patir de captura de tela *(screenshots)*. A *Sikuli* inclui o *Sikuli Script*, que é o API de script para *Jython(Java + Python)* e o *Sikuli IDE*, um ambiente de desenvolvimento específico. O *Sikuli Script* automatiza qualquer tela apresentada, simulando a interação do usuário e testando qualquer aplicação localmente ou em emuladores.

A seguir são apresentados os passos da aplicação do SQFD.

7.1 Passo 1 - SQFD

Através de reuniões e de discussões com o cliente foi realizado o passo 1 do SQFD. Foram então identificados 30 requisitos, organizados e divididos em níveis e agrupados por tópicos, com a descrição de cada requisito. A Tabela 2 apresenta o desdobramento das qualidades exigidas, organizadas em grupos e níveis.

	Nível 1	Nível 2
1	Planejamento da atividade de teste	Apoio para a criação de um Plano de testes
		Apoio para estimativa de prazos para as atividades de teste
		Apoio para identificação da complexidade das partes do sistema
		Integração com ferramentas de gerenciamento de projeto.
2	Geração automática de testes	Geração automática de testes funcionais
		Geração automática de testes para bugs cadastrados por fora da ferramenta, através da descrição do bug
3	Apoio aos testes	Apoio para a geração de dados para execução de um teste
		Geração de um povoador a partir de um banco de dados já povoado
		Configuração de ambiente para execução dos testes
		Gestao de configuração dos artefatos de teste
		Acompanhamento de falhas
4	Execução automática	Execução automática
		Agrupamento de testes
		Agendamento da execução
		Facilidade para interrupção e retomada dos testes
5	Acompanhamento dos testes	Identificação da produtividade dos testadores
		Verificação da reincidencias de falhas
6	Avaliação de resultados	Avaliação da cobertura e qualidade dos testes
7	Documentação de testes	Documentação dos casos de testes e procedimentos de teste
		Documentação gerencial dos testes
		Documentação de execução dos testes
		Fácil visualização da documentação
8	Rastreabilidade dos testes	Rastreabilidade entre os teste e os artefatos relacionados (código, ERSw, Desenho).
		Identificação de testes afetados por mudanças em artefatos relacionados
9	Apoio à implementação de testes	Facilidade para criação de testes de forma não automática
10	Integração	Importação de testes criados em outras ferramentas de teste
11	Aquisição e implantação	Ser gratuita ou baixo custo
		Possuir boa documentação
12	Funcionamento	Funcionamento em múltiplas plataformas: windows, linux.
13	Usabilidade e desempenho	Boa usabilidade, favorecendo a produtividade dos seus usuários

Tabela 2 – Tabela de Desdobramento das Qualidade Exigidas

7.2 Passo 2 - SQFD

A partir do passo anterior, identificou-se quais funções as ferramentas deveriam ter para atender aos requisitos do cliente, ou seja, as especificações técnicas do produto, reunidas em 30 requisitos, a Tabela 3 refere-se a um trecho da tabela de desdobramento dos elementos da qualidade, apresentados em grupos e níveis.

	Nível 1	Nível 2
1	Gerador de plano de teste	Gestão de dados do plano de teste
		Registro de tarefas
		Integração com ferramentas de gerenciamento de projetos
2	Gerador de dados	Gerador de objetos
3	Gerador de testes funcionais	Gerador de entradas utilizando critérios
		Oráculo para gerar as saídas esperadas
		Extrator de dados de modelos descrevendo o sistema
4	Gerador manual de testes	Mecanismo de captura-reprodução
		Uso de linguagens de alto nível
		Acesso as funcoes do SO
		Acesso ao mecanismo de persistência
5	Integrador	Integração com ferramentas de acompanhamento de falhas
		Integração com ferramentas de gestão de configuração
6	Avaliador de testes	Avaliador de cobertura
7	Gerador de Relatórios	Gerador de relatório com formato definido pelo usuário
8	Suporte da ferramenta	Cadastro de usuários
		Cadastro de grupos
		Cadastro de projeto
		Cadastro de equipe
9	Arquitetura da ferramenta	Uso de software livres
		Seguir um guia de estilo
		Utilizar terminologia adequada ao contexto
10	Auxílio da ferramenta	Help on-line
		Manual de usuário
		Sítio de apoio com exemplos de uso
11	Executor de teste	Agrupador e escalonador de testes
		Gerador de log de execução de testes
		Comparador de arquivos ignorando padrões configuráveis
		Povoador de dados
		Analisador de Falhas

Tabela 3 – Trecho extraído da Tabela completa de Desdobramento dos Elementos da Qualidade

7.3 Passo 3 - SQFD

Neste passo, foi construída a matriz da qualidade, ou casa da qualidade, combinando as saídas dos passos anteriores, ou seja foi realizada a correlação dos requisitos do cliente com as especificações técnicas. Foi estabelecido o grau de correlação, estabelecido precisamente conforme explicado anteriormente, exibido a partir do trecho extraído da tabela total da Matriz da Qualidade, como ilustrado abaixo, na Tabela 4.

				Funções da ferramenta								
			Nível 1	Gerador de plano de teste			Gerador de dados	Gerador de testes funcionais				
		Nível 2		Gestão de dados do plano de teste	Registro de tarefas	Integração com ferramentas de gerenciamento de projetos	Gerador de objetos	Gerador de entradas utilizando critérios	Oráculo para gerar as saídas esperadas	Extrator de dados de modelos descrevendo o sistema		Grau de Importância
	Nível 1	Nível 2										
1	Planejamento da atividade de teste	Apoio para a criação de um Plano de testes	1	9	3	9	0	0	0	0	39	3
		Apoio para estimativa de prazos para as atividades de teste	2	0	3	0	0	0	0	0	9	1
		Apoio para identificação da complexidade das partes do sistema	3	0	3	0	0	0	0	0	1	1
		Integração com ferramentas de gerenciamento de projeto.	4	3	3	9	0	0	0	0	12	2
2	Geração automática de testes	Geração automática de testes funcionais	5	0	0	0	9	9	9	9	71	1
		Geração automática de testes para bugs cadastrados por fora da ferramenta, através da descrição do bug	6	0	0	0	9	9	9	9	70	1
3	Apoio aos testes	Apoio para a geração de dados para execução de um teste	7	0	0	0	9	9	0	9	27	1
		Geração de um povoador a partir de um banco de dados já povoado	8	0	0	0	0	0	0	9	9	1
		Configuração de ambiente para execução dos testes	9	0	0	0	0	0	0	0	12	1
		Gestao de configuração dos artefatos de teste	10	0	0	0	0	0	0	0	9	3
		Acompanhamento de Bugs	11	0	0	0	0	0	0	0	9	2
4	Execução automática	Execução automática	12	0	0	0	0	0	0	0	18	3
		Agrupamento de testes	13	0	0	0	0	0	0	0	9	3
		Agendamento da execução	14	0	0	0	0	0	0	0	3	1
		Facilidade para interrupção e retomada dos testes	15	0	0	0	3	0	0	0	3	1
				21	12	21	42	48	27	36		
		Peso Absoluto		42.0	21.0	54.0	60.0	66.0	45.0	36.0		1285.00
		Peso Relativo		3.3%	1.6%	4.2%	4.7%	5.1%	3.5%	2.8%		1.00

Tabela 4 – Trecho extraído da Matriz de Qualidade

Nota-se na tabela que alguns requisitos não possuem nenhuma correlação e outros

têm uma ligação forte, devido ao fato de serem aspectos determinantes na correlação para a Matriz.

7.4 Passo 4- SQFD

Neste passo, pesos são atribuídos as características necessárias que as ferramentas devem conter, atribuindo o grau de importância de 1 a 3 para cada item, sendo 3 o peso máximo. Segue abaixo a Tabela 5 completa, junto com os pesos atribuídos.

Nº	Requisitos	Grau de Importância para o Cliente
1	Apoio para a criação de um Plano de testes	3
2	Apoio para estimativa de prazos para as atividades de teste	1
3	Apoio para identificação da complexidade das partes do sistema	3
4	Integração com ferramentas de gerenciamento de projeto.	2
5	Geração automática de testes funcionais	3
6	Geração automática de testes para bugs cadastrados por fora da ferramenta, através da descrição do bug	1
7	Apoio para a geração de dados para execução de um teste	2
8	Geração de um povoador a partir de um banco de dados já povoado	1
9	Gestao de configuração de artefatos de teste	3
10	Configuração de ambiente para execução dos testes	1
11	Acompanhamento de Falhas	2
12	Execução automática	3
13	Agrupamento de testes	3
14	Agendamento da execução	1
15	Facilidade para interrupção e retomada dos testes	1

Tabela 5 – Tabela completa de requisitos priorizados

7.5 Passo 5 - SQFD

O passo 5 demonstrará para o cliente os resultados alcançados, aplicando o método SQFD, em que atribui-se a priorização das especificações técnicas. A priorização é derivada do somatório da multiplicação do grau de importância pelos pesos que cada aspecto técnico recebeu. Como mostrado na Figura do passo 3, em que se obtém a Tabela 6 completa de priorizacão:

Nº	Aspecto Técnico	Peso
1	Gestão de dados do plano de teste	3.3%
2	Registro de tarefas	1.6%
3	Integração com ferramentas de gerenciamento de projetos	4.2%
4	Gerador de objetos	4.7%
5	Gerador de entradas utilizando critérios	5.1%
6	Oráculo para gerar as saídas esperadas	3.5%
7	Extrator de dados de modelos descrevendo o sistema	2.8%
8	Mecanismo de captura-reprodução	1.0%
9	Uso de linguagens de alto nível	1.2%
10	Acesso as funcoes do SO	2.1%
11	Acesso ao mecanismo de persistência	2.1%
12	Integração com ferramentas de acompanhamento de falhas	5.1%
13	Integração com ferramentas de gestão de configuração	5.4%
14	Avaliador de cobertura	3.5%
15	Gerador de relatório com formato definido pelo usuário	5.6%

Tabela 6 – Trecho da Tabela completa de aspectos técnicos priorizados

7.6 Avaliação das Ferramentas

Para definição de resultados em conjunto com o SQFD, foi utilizado o Método de Avaliação proposto por Veloso et al. (2010), usado em seu trabalho para avaliar as ferramentas de apoio ao teste de software. No presente trabalho, buscando avaliar as ferramentas selecionadas para Aplicações Android, o método envolve um questionário que compõe a definição de Métricas de Avaliação. Neste questionário foram elaboradas perguntas que traduzem os aspectos técnicos identificados anteriormente para avaliar em porcentagem o quanto a ferramenta atende em relação ao aspecto técnico.

Para definir as Métricas de avaliação, foram atribuídos critérios que definem o nível de pontuação para cada ferramenta e o quanto a ferramenta analisada satisfaz os aspectos técnicos identificados, sendo definido da seguinte forma:

- Atributo 0 (zero): Quando não atende o aspecto técnico.
- Atributo 0,5 (meio): Quando atende parcialmente o aspecto técnico.
- Atributo 1 (um): Quanto atende totalmente o aspecto técnico.

Fortalecendo a escolha e definindo a métrica de julgamento da ferramenta que mais se enquadra às necessidades levantadas pelo cliente, as 3 ferramentas foram submetidas a testes, instaladas e configuradas, a partir da documentação da ferramenta e respeitando as restrições do cliente, a fim de explorar e analisar o apoio que as mesmas fornecem às particularidades das Aplicações Android. Nestas foram testadas Aplicações gratuitas que utilizam os sensores aqui abordados durante a interação com o usuário. E junto a execução e teste destas ferramentas foi atribuída uma pontuação para cada pergunta, sendo que, quando houve mais de uma pergunta para um determinado aspecto técnico, foi atribuída uma média entre elas. A Tabela 7 apresenta um trecho retirado do questionário completo, demonstrando como a pontuação foi atribuída.

				Ferramentas					
				MonkeyTalk		Sikuli		Selendroid	
Gerador Manual de Testes	Mecanismo de captura-reprodução	A ferramenta gera testes a partir da gravação de ações realizadas pelo usuário ou integração com essas ferramentas?	SIM (1)	x	1		0	x	0
			PARCIALMENTE (0.5)			x			
			NÃO (0)						
	Uso de linguagens de alto nível	A ferramenta possibilita o uso de linguagem de alto nível na criação dos testes?	SIM (1)	x	1	x	1		0
			PARCIALMENTE (0.5)						
			NÃO (0)					x	
		A ferramenta exporta o código de teste para linguagens de alto nível?	SIM (1)	x		x			
			PARCIALMENTE (0.5)						
			NÃO (0)					x	
	Acesso as funcoes do SO	A ferramenta possibilita o acesso a informações (configurações) do S.O.?	SIM (1)	x	1		0		0
			PARCIALMENTE (0.5)						
			NÃO (0)			x		x	
	Acesso ao mecanismo de persistência	A ferramenta permite acessar diretamente um mecanismo de persistência (como banco de dados e arquivos)?	SIM (1)		0		0		0
			PARCIALMENTE (0.5)						
			NÃO (0)	x		x		x	

Tabela 7 – Trecho extraído do questionário completo sobre as ferramentas

Por fim, elaborou-se a Tabela 8 que apresenta a comparação das três ferramentas analisadas, relacionando o grau de importância de cada aspecto técnico com o resultado obtido pelo questionário em que se tem como conteúdo as categorias:

- **Grupos:** Contém os grupos de aspectos técnicos identificados.

- **Aspectos Técnicos:** São todos os aspectos técnicos relacionados ao seu grupo.

- **Pesos:** É o peso em porcentagem de cada aspecto técnico obtido na Matriz de Qualidade.

- **Pontuação:** São as pontuações obtidas no questionário para cada aspecto técnico em relação à ferramenta analisada.

- **Atendido:** É o cálculo da multiplicação de cada valor da coluna Peso pelos valores da mesma linha da coluna Pontuação. Então tem-se o somatório dos produtos do mesmo grupo de aspectos técnicos, apresentador na coluna Atendido. Os resultados apresentam em porcentagem o quão aquele aspecto técnico atende às necessidades do cliente.

				FERRAMENTAS					
				Monkey Talk		Sikuli		Selendroid	
	Grupo	Aspectos Técnicos	Pesos	Pontuação	Atendido	Pontuação	Atendido	Pontuação	Atendido
1	Gerador de plano de teste	Gestão de dados do plano de teste	3.3	0	0	0	0	0	0
		Registro de tarefas	1.6	0		0		0	
		Integração com ferramentas de gerenciamento de projetos	4.2	0		0		0	
2	Gerador de dados	Gerador de objetos	4.7	0	0	0	0	0	0
3	Gerador de testes funcionais	Gerador de entradas utilizando critérios	5.1	0	0	0	0	0	0
		Oráculo para gerar as saídas esperadas	3.5	0		0		0	
		Extrator de dados de modelos descrevendo o sistema	2.8	0		0		0	
4	Gerador manual de testes	Mecanismo de captura-reprodução	1	1	4.3	0.5	1.7	0.5	1.7
		Uso de linguagens de alto nível	1.2	1		1		1	
		Acesso as funcoes do SO	2.1	1		0		0	
		Acesso ao mecanismo de persistência	2.1	0		0		0	

Tabela 8 – Trecho extraído da tabela completa de comparação entre as ferramentas de testes para Aplicações Android: MonkeyTalk, Sikuli e Selendroid.

Parte III

Resultados e Discussão

Para análise do suporte dado pelas ferramentas ao uso de dados de sensores, foi realizado um levantamento a partir da documentação existente, bem como foram realizados ensaios envolvendo o teste de aplicações que faziam uso de sensores para seu funcionamento. Devido ao grande número de sensores e suas características, a análise foi restrita aos sensores de movimentação Acelerômetro e Giroscópio e ao sensor de posicionamento GPS. Esses sensores são utilizados por um grande número de aplicações (EDITORASABER, 2016).

A partir da análise realizada, foi constatado que a ferramenta Sikuli, apesar de realizar testes gerais para Aplicações Android, não apresenta suporte para acesso a sensores. A Selendroid oferece um método que permite o controle da orientação do dispositivo através do uso de comandos. O comando *"driver().getOrientation();"* possibilita a obtenção da orientação atual do dispositivo (retrato ou paisagem), a qual envolve o uso de Acelerômetro e Giroscópio. Já o comando *"driver().rotate(ScreenOrientation .LANDSCAPE);"* permite a configuração da orientação do dispositivo.

A ferramenta MonkeyTalk possui comandos para ajustar a orientação do dispositivo que podem ser inseridos no script de teste. O comando *"Device * Rotate Landscape;"* opera da mesma forma do Selendroid, interferindo na ação dos sensores de movimentação.

Em relação ao teste envolvendo dados de GPS, nenhuma das ferramentas perimitem o acesso direto ao GPS. A ferramenta MonkeyTalk permite a chamada de aplicações externas, tais como a FakeGPS (GOOGLE; ALLIANCE, 2016b), o que possibilita a interação indireta com o GPS.

Com relação às necessidades do cliente e as restrições abordadas, a partir da aplicação do SQFD e do Método de Avaliação proposto por Veloso et al. (2010), foi determinado o grau de satisfação de cada ferramenta, representado em porcentagem, obtida a partir da somatória das linhas da coluna "Atendido" de cada ferramenta, como mostrado na Tabela 9.

				FERRAMENTAS					
				Monkey Talk		Sikuli		Selendroid	
	Grupo	Aspectos Técnicos	Pesos	Pontuação	Atendido	Pontuação	Atendido	Pontuação	Atendido
1	Gerador de plano de teste	Gestão de dados do plano de teste	3.3	0	0	0	0	0	0
		Registro de tarefas	1.6	0		0		0	
		Integração com ferramentas de gerenciamento de projetos	4.2	0		0		0	
2	Gerador de dados	Gerador de objetos	4.7	0	0	0	0	0	0
3	Gerador de testes funcionais	Gerador de entradas utilizando critérios	5.1	0	0	0	0	0	0
		Oráculo para gerar as saídas esperadas	3.5	0		0		0	
		Extrator de dados de modelos descrevendo o sistema	2.8	0		0		0	
4	Gerador manual de testes	Mecanismo de captura-reprodução	1	1	4.3	0.5	1.7	0.5	1.7
		Uso de linguagens de alto nível	1.2	1		1		1	
		Acesso as funcoes do SO	2.1	1		0		0	
		Acesso ao mecanismo de persistência	2.1	0		0		0	
5	Integrador	Integração com ferramentas de acompanhamento de falhas	5.1	0	0	0	0	0	0
		Integração com ferramentas de gestão de configuração	5.4	0		0		0	
6	Avaliador de testes	Avaliador de cobertura	3.5	1	3.5	0	0	0	0
7	Gerador de relatórios	Gerador de relatório com formato definido pelo usuário	5.6	0	0	0	0	0	0
8	Suporte da ferramenta	Cadastro de usuários	3.3	0	0	0	0	0	0
		Cadastro de grupos	3.3	0		0		0	
		Cadastro de projeto	3.5	0		0		0	
		Cadastro de equipe	3.5	0		0		0	
9	Arquitetura da ferramenta	Uso de software livres	2.6	1	6.1	0	3.5	1	4.35
		Seguir um guia de estilo	3.5	1		1		0.5	
		Utilizar terminologia adequada ao contexto	4.9	0		0		0	
10	Auxílio da ferramenta	Help on-line	2.1	0	4.9	0	4.9	0	4.9
		Manual de usuário	2.8	1		1		1	
		Sítio de apoio com exemplos de uso	2.1	1		1		1	
11	Executor de teste	Agrupador e escalonador de testes	2.3	0.5	6.02	0	4.2	0	4.2
		Gerador de log de execução de testes	5.6	0.87		0.75		0.75	
		Comparador de arquivos ignorando padrões configuráveis	1.2	0		0		0	
		Povoador de dados	7.2	0		0		0	
		Analisador de Falhas	1	0		0		0	
				Satisfação do Cliente	24.82%		14.30%		15.15%

Tabela 9 – Comparação entre as ferramentas de testes MonkeyTalk, Sikuli e Selendroid.

Os resultados apontam a Ferramenta MonkeyTalk como a mais adequada as necessidades do cliente em nosso estudo, com 24,82% de atendimento relativo as necessidades totais do cliente, 61% a mais que a ferramenta com a segunda maior pontuação. Vale ressaltar que este percentual não quer dizer que uma ferramenta é melhor do que a outra, e sim que atende mais as necessidades e restrições do cliente.

É importante destacar alguns aspectos técnicos que tiveram uma pontuação expressiva em relação as expectativas do cliente, o grupo "Auxílio da Ferramenta", que estão ligados ao suporte oferecido pela ferramenta aos seus usuários, que representam em todas as ferramentas quase 5% da pontuação de cada ferramenta, sendo um ponto muito positivo da análise. A ferramenta com maior porcentagem de atendimento, a MonkeyTalk apresentou uma nota considerada em nossa pesquisa no grupo "Executor de Teste", que apesar de não atender a maioria dos requisitos deste grupo, cobre quase 100% de um dos itens de maior importância, que é o gerador de log de execução de teste, que mostra o passo a passo feito pela ferramenta, o que ajuda a rastrear possíveis erros. Já no grupo "Arquitetura da Ferramenta" sobressaiu em relação as outras ferramentas comparadas no que complementa a ferramenta e seu funcionamento.

Com o estudo podemos perceber que dificilmente todas as ferramentas vão atender por igual um determinado Grupo, mas ao contrário disso podemos perceber que

alguns grupos não foram atendidos em nenhuma delas em nenhum requisito, como os grupos “Gerador de Plano de Teste”, “Gerador de Dados” e “Gerador de Testes Funcionais”, que influenciaram na baixa pontuação de todas elas. O que levanta ainda mais a atenção para melhoria de ferramentas de teste para se adequar, não só às necessidades do cliente considerado, mas para atender de forma ampla todos os desenvolvedores.

Considerações Finais

O trabalho realizado propôs identificar ferramentas adequadas as atividades de teste, no contexto de desenvolvimento do LEDS, visando atender aos requisitos propostos pelo cliente, inclusive considerando testes envolvendo as particularidade de Aplicações Android, neste caso, o uso de dados de sensores.

Por meio do estudo feito através do método SQFD, e do método proposto por Veloso et al. (2010) três ferramentas gratuitas foram avaliadas: MonkeyTalk, Sikuli e Selendroid. Dentre essas, a que mais atendeu aos requisitos do cliente foi a ferramenta MonkeyTalk.

Porém a ferramenta com maior grau de satisfação do cliente, pode não ser a que será utilizada, em última instância pelo LEDS. Cabe à equipe identificar qual se aplica melhor a esta ou aquela aplicação em questão, o que gera maior agilidade na etapa de teste, dentro do desenvolvimento, agregando maior qualidade ao produto a partir de testes automatizados.

Como trabalho futuro, é possível aprofundar o estudo das funcionalidades das ferramentas com o intuito de sugerir e implementar modificações, de modo que, as ferramentas possam se aproximar ainda mais da satisfação de um maior número dos requisitos do cliente. Em relação aos sensores, seria um caminho promissor o teste envolvendo Jogos, que fazem uso extensivo desse tipo de recurso.

Referências

AKAO, Y. *Introdução ao Desdobramento da Qualidade*. Belo Horizonte: Fundacao Cristiano Ottoni - Escola de Engenharia da UFMG, 1996. Citado na página 23.

BORJESSON, E.; FELDT, R. Automated system testing using visual gui testing tools: A comparative study in industry. In: VERIFICATION; WORKSHOPS, V. (Ed.). *IEEE Fifth International Conference on Software Testing*. IEEE, 2012. p. 350–359. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6200127&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6200127>. Acesso em: 22 fev. 2016. Citado na página 25.

BUTLER, M. Android: Changing the mobile landscape. *IEEE Pervasive Computing*, v. 10, n. 1, p. 4–7, 2011. Citado na página 13.

CALDEIRA, L. S. *Avaliação de Ferramentas para o Teste Funcional*. 52 p. Monografia (Graduação) — Universidade Federal de Ouro Preto, João Monlevade - MG, 2014. Citado na página 29.

COSTA, N. P. O.; FILHO, N. F. D.; DUARTE, A. F. Comparative evaluation of operating systems for devices mobile: Focus on functionality. In: SCHULZ, H. (Ed.). *Congresso Internacional de Gestão de Tecnologia e Sistemas de Informação (RF-196)*. 9 Contecsi, 2012. p. 3011–3025. Disponível em: <http://www.contecsi.fea.usp.br/envio/index.php/contecsi/9contecsi/paper/viewFile/3447/1949>. Acesso em: 2 jan. 2016. Citado na página 14.

DANTAS, V. *Requisitos para Testes de Aplicções Móveis*. Dissertação (Dissertação de Mestrado em Ciência da Computação) — Universidade Federal do Ceará, 2009. Citado na página 13.

DELAMARO, M. G.; MALDONADO, J. C.; JINO, M. *Introdução ao teste de Software*. Rio Janeiro: Elsevier Editora Ltda, 2007. Citado 3 vezes nas páginas 13, 17 e 18.

DEVMEDIA. *Sensores no Android - Revista Java Magazine*. 2016. Http://www.devmedia.com.br/sensores-no-android-revista-java-magazine-94/21936ixzz41CzFn4fd. Acesso em: 18 fev. 2016. Citado 2 vezes nas páginas 20 e 22.

DóREA, A. D. O. et al. Avaliação de ferramentas de automação para engenheiros de testes. *Anais do IV Simpósio Brasileiro de Sistemas de Informação*, v. 1, n. 1, p. 23–34, 2008. Citado na página 25.

EDITORASABER. *Comparativo dos acelerômetros*. 2016. Http://www.sabereletronica.com.br/artigos/1240-comparativo-entre-acelermetros. Acesso em: 3 fev. 2016. Citado na página 39.

GOOGLE, I.; ALLIANCE, O. H. *Android Developers*. 2016. Http://developer.android.com/sdk/index.html. Acesso em: 2 nov. 2016. Citado na página 20.

GOOGLE, I.; ALLIANCE, O. H. *Fake GPS Location*. 2016. Https://play.google.com/store/apps/details?id=com.lexa.fakegpshl=. Acesso em: 3 fev. 2016. Citado na página 39.

GOOGLE, I.; ALLIANCE, O. H. *Sensors Overview*. 2016. Http://developer.android.com/guide/topics/sensors/sensors_overview.html. Acesso em: 19 fev. 2016. Citado na página 20.

HAAG, S.; RAJA, M. K.; SCHKADE, L. L. Quality function deployment usage in software development. *Communications of the ACM*, v. 39, n. 1, p. 41–49, 1996. Citado 5 vezes nas páginas 8, 9, 14, 23 e 25.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *IEC25010*: 2010 systems and software engineering system ad software quality requirements and evaluation square system and software quality models. [S.l.], 2010. Citado na página 18.

K19. *Desenvolvimento Mobile com Android*. 1. ed. São Paulo, SP, 2012. 152 p. Citado na página 20.

LECHETA, R. R. *Google Android: Aprenda a Criar Aplicaçõespara Dispositivos Móveis com o Android SDK*. [S.l.]: Novatec Editora Ltda, 2013. Citado na página 14.

LEE, V.; SCHNEIDER, H.; SCHELL, R. *Aplicações Móveis - Arquitetura, projetos e desenvolvimento*. [S.l.]: Pearson Education do Brasil, 2005. Citado na página 13.

MYERS, G. *The Art of Software Testing*. [S.l.]: John Wiley e Sons, 2004. Citado na página 17.

PRESSMAN, R. S. *Software Engineering: A Practitioners Approach*. [S.l.]: McGraw-Hill, 4th ed, Nova York, NY, 1997. Citado 2 vezes nas páginas 18 e 19.

PRESSMAN, R. S. *Engenharia de Software*. [S.l.]: McGraw-Hill, 5th ed, Rio de Janeiro, Brazil, 2002. Citado na página 19.

PRESSMAN, R. S. *Software Engineering: A Practitioners Approach*. [S.l.]: McGraw-Hill, 6th ed, Nova York, NY, 2005. Citado na página 13.

SANTOS, I. S. et al. Uma avaliação de ferramentas para testes em sistemas de informação móveis baseada no método dmadv. In: *Anais do IX Simpósio Brasileiro de Sistemas de Informação*. [S.l.: s.n.], 2013. p. 553–564. Citado na página 25.

SHAMSODDIN-MOTLAGH, E. A review of automatic test cases generation. *International Journal of Computer Applications*, v. 57, n. 13, p. 25–29, 2012. Citado na página 13.

TEAM, M. M. S. C. *MATLAB Support Package for Android Sensors*. 2014. Http://www.mathworks.com/matlabcentral/fileexchange/47618-matlab-support-package-for-android-sensors. Acesso em: 15 jun. 2015. Citado 2 vezes nas páginas 10 e 21.

VELOSO, J. et al. Avaliação de ferramentas de apoio ao teste de sistemas de informação. *iSys-Revista Brasileira de Sistemas de Informação*, v. 3, p. 1–17, 2010. Citado 10 vezes nas páginas 8, 9, 13, 14, 15, 25, 29, 35, 39 e 42.

WANG, S.; OFFUTT, J. Comparison of unit-level automated test generation tools. In: VERIFICATION; WORKSHOPS, V. (Ed.). *International Conference on Software Testing*. IEEE, 2009. p. 210–219. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=4976389>. Acesso em: 2 mar. 2016. Citado na página 25.