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

Avaliação de Ferramentas para o Teste Funcional de Aplicações Android

Maxmiler Silva Torres

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Prof. Me. Euler Horta Marinho

Março, 2016

João Monlevade/MG

Maxmiler Silva Torres

Avaliação de Ferramentas para o Teste Funcional de Aplicações Android

Orientador: Prof. Me. Euler Horta Marinho

Monografia apresentada ao Curso de Sistemas de Informação do Departamento de Computação e Sistemas, como requisito parcial para aprovação na Disciplina Trabalho de Conclusão de Curso II.

Universidade Federal de Ouro Preto

João Monlevade

Março de 2016

Curso de Sistemas de Informação

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Avaliação de Ferramentas para o Teste Funcional de Aplicações Android

Maxmiler Silva Torres

Monografia apresentada ao Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CEA499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação e aprovada pela Banca Examinadora abaixo assinada:

Prof. Me. Euler Horta Marinho
Mestre pelo Instituto de Computação/Universidade Federal Fluminense – RJ, Brasil
Orientador
Departamento de Computação e Sistemas - UFOP

Prof. Dr. Diego Zuquim Guimarães Garcia
Doutor pelo Instituto de Computação/Universidade Estadual de Campinas – SP, Brasil
Examinador
Departamento de Computação e Sistemas - UFOP

Prof. Me. Igor Muzetti Pereira
Mestre pelo Departamento de Computação/Universidade Federal de Ouro Preto – MG, Brasil
Examinador
Departamento de Computação e Sistemas - UFOP

João Monlevade, 17 de Março de 2016

ATA DE DEFESA

Aos dezessete dias do mês de março de 2016, às 17 horas, na sala C203, Laboratório de Engenharia de Software do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pelo aluno **Maxmiler Silva Torres,** sendo a Comissão Examinadora constituída pelos professores: Prof. Me. Euler Horta Marinho, Prof. Dr. Diego Zuquim Guimarães Garcia e Prof. Me. Igor Muzetti Pereira.

O candidato apresentou a monografia intitulada: *"Avaliação de Ferramentas para o Teste Funcional de Aplicações Android"*. 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, 17 de Março de 2016.

Prof. Me. Euler Horta Marinho
Professor Orientador/Presidente

Prof. Dr. Diego Zuquim Guimarães Garcia
Professor Convidado

Prof. Me. Igor Muzetti Pereira
Professor Convidado

Maxmiler Silva Torres
Graduando

Resumo

Existe uma grande quantidade de ferramentas disponíveis para o desenvolvimento de Aplicações Móveis. Diante da possibilidade de acesso à informações em qualquer lugar e a qualquer momento, pode-se imaginar um grande número de oportunidades para novas aplicações e serviços, ressaltando a importância do teste de software na busca por uma melhor qualidade. Porém, existem várias ferramentas de testes para aplicações móveis, dificultando a escolha da ferramenta mais adequada. No presente trabalho, nós realizamos uma avaliação de ferramentas para o teste funcional de Aplicações Android, considerando o contexto do Laboratório de Engenharia e Desenvolvimento de Sistemas (LEDS) da Universidade Federal de Ouro Preto. Para avaliação das ferramentas, utilizamos o SQFD e um método de avaliação, que permitiram determinar o nível de adequação das ferramentas em relação aos requisitos identificados. Como resultados dessa avaliação, identificamos uma ferramenta que melhor atende às necessidades do cliente. Em seguida, fazemos uma discussão sobre a usabilidade dessas ferramentas.

Palavras-chave: Aplicações Android, Ferramentas para Teste, SQFD, Avaliação de Usabilidade, Avaliação de Qualidade.

Abstract

There is a large amount of available tools for mobile applications development. Given the possibility of access to information anywhere and at any time, one can imagine a large number of opportunities for new applications and services, emphasizing the role of the software testing in the search for better quality. However, there are several testing tools for mobile applications, making difficult the choice of the most appropriate tool. In this work, we performed an evaluation of functional testing tools for Android applications, taking into account the context of Laboratório de Engenharia e Desenvolvimento de Sistemas (LEDS) of the Federal University of Ouro Preto. In order to assess the tools, we use the SQFD together with an evaluation method, which allowed the identification of the adequacy of the tools in relation to the identified requirements. As a result of this evaluation, we identified a tool that best suits the customer's needs. Then, we make a discussion about the usability of these tools.

Keywords: Android Applications, Testing Tools, SQFD, Usability Evaluation, Quality Evaluation.

Agradecimentos

Agradeço a Deus primeiramente que me permitiu que tudo isso acontecesse, por me dar força e fé em todos os momentos que passei durante o período acadêmico.

Ao professor Me. Euler Horta Marinho por todo apoio na realização deste trabalho, por sua dedicação, orientação, empenho e suporte.

Aos meus amados pais, Eliezer Martins Torres e Maria das Graças da Silva Torres, por toda confiança, dedicação e apoio, por terem feito o possível e o impossível nessa minha caminhada.

Aos amigos de república, foi um aprendizado excepcional morar com vocês.

A minha querida turma 10.2, vocês foram as melhores pessoas que poderiam estar juntos comigo.

A galera da litragem por sempre me fazerem esquecer a faculdade e viver a vida. Pé de cobra, cerveja e farra nunca faltaram. Borá beber cambada..

E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Diante da evolução tecnológica e alta concorrência no mercado de software, as empresas precisam desenvolver seus produtos com uma preocupação crescente com a qualidade, visando garantir que o software atenda às demandas dos clientes. O desenvolvimento de software envolve diversas dificuldades, como a complexidade dos produtos, questões burocráticas, humanas, etc. O teste se apresenta como uma atividade para a busca da qualidade do produto final.

Celulares e smartphones passaram a ter um importante papel onde a palavra “mobilidade” está cada vez mais inserida na sociedade (Lecheta (2009)). Diante da possiblidade de acessar informações em qualquer lugar e a qualquer momento, pode-se imaginar um grande número de oportunidades para novas aplicações e serviços. Existem muitos ambientes e ferramentas para o desenvolvimento de aplicações e serviços para a computação móvel, necessitando da atividade de teste a fim de garantir uma melhor qualidade de seus produtos e serviços. (Figueiredo et al. (2003)).

De acordo com Mazlan (2006), o teste é uma atividade obrigatória que tem como objetivo certificar que os dispositivos móveis são confiáveis, robustos e “livres de erros”. O ambiente móvel envolve diversas limitações tais como pouca memória, tamanho da tela, baixa capacidade de processamento, pouca duração da bateria, dentre outros. Portanto os testes de Aplicações Móveis devem ser planejados levando em consideração o usuário, o contexto móvel e a aplicação móvel. Além disso, vale ressaltar que o processo de teste de software requer tempo e recursos, o que o torna uma etapa onerosa do desenvolvimento de software (Shamsoddin-Motlagh (2012)).

As ferramentas utilizadas em apoio ao teste tem o objetivo de reduzir os custos dos mesmos. Existem várias ferramentas de testes para sistemas de informação móveis disponíveis no mercado (GitHub (2015)). Contudo, a escolha da ferramenta adequada depende dos requisitos e necessidades dos desenvolvedores. A escolha de ferramentas de testes inadequadas pode trazer desperdício de tempo, custos, além de poder influenciar negativamente no sucesso do produto (Veloso et al. (2010)). Utilizar-se de uma metodologia para analisar as ferramentas mais adequadas para o teste de Aplicações Móveis seria útil no quesito produtividade e qualidade.

No presente trabalho, nós realizamos uma avaliação de ferramentas para o teste funcional de Aplicações Android e fazemos uma discussão sobre a usabilidade dessas ferramentas. Nosso estudo de caso leva em consideração o contexto do Laboratório de Engenharia e Desenvolvimento de Sistemas (LEDS) da Universidade Federal de Ouro Preto, aqui determinado como *cliente*, com o propósito de encontrar as ferramentas que melhor atendam suas necessidades. O LEDS é constituído por professores e alunos de graduação da UFOP e conta com a colaboração de estudantes, professores e pesquisadores de outras instituições.

Nesse cenário, utilizamos o SQFD (*Software Quality Function Deployment*) como metodologia, para auxiliar na escolha das ferramentas de teste de Aplicações Android mais adequadas durante a especificação dos requisitos, identificados e priorizados de acordo com as necessidades do cliente. Juntamente com o SQFD usamos um Método de Avaliação apresentado por Veloso et al. (2010) para avaliar as ferramentas de teste. As escolhas destes métodos foi decorrente da nossa experiência anterior com os mesmos.

1.1 Objetivo Geral

Como objetivo geral, procuramos realizar a avaliação das ferramentas de teste para determinar seu nível de adequação em relação aos requisitos identificados. Dessa maneira, buscamos auxiliar os desenvolvedores na escolha da ferramenta mais adequada para o teste funcional de Aplicações Android. Posteriormente realizamos uma discussão sobre a usabilidade das ferramentas.

1.2 Objetivo Específico

Como objetivo específico realizamos um estudo de caso considerando a análise das ferramentas para o teste de Aplicações Android, utilizando o SQFD e o Método de Avaliação proposto por Veloso et al. (2010), de acordo com as necessidades e requisitos do LEDS. Além disso, discutimos aspectos da usabilidade das ferramentas consideradas. O laboratório é constituído por professores e estudantes da Universidade Federal de Ouro Preto.

1.3 Organização do Trabalho

O trabalho foi dividido em 5 capítulos, organizados da seguinte maneira:

O capítulo 2 apresenta definições de alguns conceitos sobre o teste de software, teste de sistema, teste de Aplicações Móveis, Aplicações Android, apresentação do Método SQFD e conceitos de usabilidade de software.

O capítulo 3 apresenta o desenvolvimento do trabalho relatando à avaliação de ferramentas para o teste de Aplicações Móveis e a aplicação do SQFD.

O capítulo 4 apresenta os resultados obtidos do SQFD e a discussão sobre usabilidade.

O capítulo 5 apresenta as considerações finais e trabalhos futuros.

2 Revisão Bibliográfica

2.1 Teste de Software

A Engenharia de Software envolve uma grande variedade de técnicas e métodos que contribuem para a realização das inúmeras atividades envolvidas no desenvolvimento de um produto de software (Pressman (2014)). Contudo, em meio a tantas técnicas e métodos, os erros ainda podem ser introduzidos no software. Portanto, para garantir um nível aceitável de qualidade, algumas atividades podem ser realizadas durante todo o processo de desenvolvimento do software, VV&T (Verificação, Validação e Teste de Software) (Maldonado (1991)). A verificação tem por objetivo assegurar que o software esteja sendo construído de maneira correta. A validação visa garantir que o software atenda às suas especificações.

O teste de software é uma atividade dinâmica. Seu propósito é executar o programa utilizando algumas entradas em particular e verificar se seu comportamento está de acordo com o esperado. Pode ser definido também como o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. Segundo Myers (2011), é o processo de executar um programa com o objetivo de encontrar erros. É designada uma boa atividade de teste aquela que consiga definir casos de teste em que o programa falhe.

Um processo de teste tem por objetivo definir responsabilidades, papéis, estruturar as etapas de modo que a organização consiga controlar todo o ciclo do teste, minimizando riscos para que o produto final saia com qualidade e reduzindo o número de erros. Em geral, é realizado pelo testador de software no qual envolve

ações que vão desde levantamentos dos requisitos até a execução do teste propriamente dito.

O teste completo de um software é uma atividade impraticável por seu alto custo computacional e financeiro (Myers (2011)). Por isso, uma tarefa de grande importância é procurar maneiras de escolher um conjunto reduzido de casos de teste para execução de um programa, que consiga alta probabilidade de evidenciar a presença de erros (Myers (2011)).

Segundo Delamaro et al. (2007), a atividade de teste é dividida em fases com objetivos distintos. Portanto, de maneira geral podemos estabelecer como fases: teste de unidade, teste de integração e teste de sistemas.

O teste de unidade tem por foco as menores unidades de um programa, ou seja, seu objetivo é identificar erros de lógica e de implementação em cada unidade do software isoladamente, buscando garantir que essa funcione apropriadamente (Myers (2011)). Segundo o padrão IEEE 24765 (IEEE 2010), uma unidade é um componente de software indivisível. Podendo ser um método, procedimento, classe, um grupo de classe (*cluster*) ou função, onde se espera encontrar a presença de falhas que podem estar relacionados a interfaces (parâmetros de entrada e saída), estruturas de dados incorretas, algoritmos incorretos e até mesmo erros cometidos pelo desenvolvedor. Sabendo-se que cada unidade é testada isoladamente, o teste de unidade pode ocorrer na medida em que a implementação das unidades acontecem, não precisando do programa estar totalmente finalizado.

O teste de integração deve ser realizado após as unidades terem sido testadas isoladamente, visando descobrir defeitos relacionados à interação e compatibilidade entre as unidades, podendo ser por exemplo chamadas incorretas, de interface. Segundo Delamaro et al. (2007) o teste de integração deve ser executado pela própria equipe de desenvolvimento por esta possuir maior conhecimento da estrutura interna do programa.

O teste de sistema é realizado após a integração do sistema, visando à identificação de erros de funções e características de desempenho que não estejam de acordo com as especificações do projeto, e abrange tanto os requisitos funcionais, como os requisitos não-funcionais. Segundo Rocha et al. (2001), no teste de sistema ocorre a avaliação do software em busca de falhas por meio de sua utilização, como se fosse mesmo um usuário final, verificando se um produto satisfaz

seus requisitos. Os requisitos não-funcionais não se preocupam especificamente com a funcionalidade do sistema, eles definem todas as qualidades, atributos e restrições de um sistema. Já os requisitos funcionais procuram escrever explicitamente as funcionalidades e serviços do sistema. Quando o foco do teste de sistema são os requisitos funcionais, o teste de sistema é conhecido como teste funcional.

Além das três fases apresentadas, uma atividade chamada teste de regressão pode ser usada. Segundo Delamaro et al. (2007) ao contrário das outras atividades, o teste de regressão não é utilizado na fase de desenvolvimento do software, mas sim durante sua manutenção. Novos defeitos podem ser encontrados quando há modificações no software. Portanto, testes que mostrem se as modificações realizadas foram corretamente implementadas são necessários (Delamaro et al. (2007)). Segundo Myers (2011) e Pressman (2014) os testes são executados em quatro etapas bem estabelecidas, independente da fase de teste, sendo elas: planejamento de testes, projeto dos casos de teste, execução do teste e coleta e avaliação dos resultados.

2.1.1 Teste Funcional

De acordo com Myers (2011), o teste funcional é conhecido como teste caixa preta, por tratar o software como uma caixa de conteúdo desconhecido visualizando-se somente o lado externo da caixa, ou seja, dados de entrada são fornecidos e respostas produzidas como saída, verificando se estão em conformidades com os objetivos definidos. Nesta técnica não se preocupa com detalhes de implementação, são verificadas somente as funções do sistema.

O teste funcional envolve duas atividades principais: (1) identificar as funções que o software deve realizar; (2) criar casos de teste capazes de checar se essas funções estão sendo realizadas pelo software (Pressman (2014)). Na especificação do software serão identificadas as funções que o mesmo deve possuir. Portanto, uma especificação bem elaborada e seguindo as especificações do usuário será essencial para esse tipo de teste (Demillo et al. (1987)).

No teste funcional os requisitos de teste são derivados a partir da especificação. Portanto, sua realização depende de uma boa especificação de requisitos. Especificações descritivas e não formais, bem como requisitos imprecisos

e informais dificultam a realização das atividades de teste (Fabbri et al. (2007)). Como consequência, tem-se dificuldade em quantificar e automatizar sua aplicação (Fabbri et al. (2007)). Outra limitação existente no teste funcional é o fato de que não se pode garantir que partes críticas do software sejam executadas durante sua prática, em vista que detalhes de implementação não são considerados pelo teste (Fabbri et al. (2007)).

2.1.2 Teste de Aplicações Móveis

De acordo com Zhang et al. (2005), Aplicações Móveis referem-se a sistemas de software que operam em dispositivos móveis. E dispositivos móveis podem ser considerados um computador portátil no qual se comunica através de uma tecnologia sem fio. Dispositivos móveis são utilizados para acessar serviços de informação por meio de uma estrutura descentralizada, que deve estar disponível independentemente da localização física do usuário e a qualquer momento. A computação móvel está sempre em constantes mudanças e a introdução de novas funcionalidades e recursos tais como: câmera, televisão, áudio, voz, texto, sensores dentre outros, tem acarretado novas demandas por Aplicações Móveis.

Dispositivos móveis apresentam diversas limitações tais como tamanho da tela, teclado, bateria, capacidade de armazenamento, diferentes tipos de conectividade (Wi-Fi, GPRS, 3G). Algumas dessas limitações são descritas a seguir:

- **Tela:** Existem restrições relacionados ao tamanho e à baixa resolução da tela. O pequeno tamanho da tela pode fazer com que páginas Web visualizadas nos dispositivos fiquem não amigáveis, ilegíveis e até mesmo não navegável. Já as baixas resoluções podem danificar a qualidade da informação exibida na tela do dispositivo. As duas situações podem afetar a usabilidade da aplicação móvel (Zhang et al. (2005)).
- **Bateria:** O tempo de vida da bateria dos dispositivos móveis é limitado e seu consumo pode variar de acordo com a tecnologia utilizada para o desenvolvimento da aplicação (Ballard (2007)). Um exemplo é que mensagens de texto usam pouca bateria quando se faz um uso limitado de interações. Já mensagens multimídia utilizam muito mais bateria por envolverem um grande tempo de carregamento (*download)*.

- **Largura de banda:** Dependendo da localização do usuário, a velocidade de transferência dos dados e a força do sinal podem variar na rede sem fio. Para lidar com as várias condições da rede sem fio, deve ser considerado o estudo de usabilidade da aplicação móvel (Zhang et al. (2005)).
- **Capacidade de armazenamento:** A capacidade de armazenamento irá variar de dispositivo para dispositivo. Alguns dispositivos possibilitam a expansão da memória interna através de cartões de memória proporcionando um aumento na quantidade de recursos usados em cada aplicação. De acordo com Ballard (2007), o armazenamento é ainda um fator limitante porque os usuários requerem armazenamento persistente, sendo localmente ou em um servidor, variando de acordo com a natureza dos dados e requisitos da aplicação.

Testes de Aplicações Móveis devem ser planejados para lidar com os aspectos de limitações dos dispositivos e as características de uso da aplicação. Para isso, testes podem ser executados em emuladores e, em dispositivos físicos.

2.2 Aplicações Android

O Android é uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para dispositivos móveis, com um sistema operacional incluso, *middleware*, aplicativos e uma interface para o usuário (Pereira (2009)). Essa plataforma foi concebida a partir de um projeto de código aberto (*open-source*), cujo desenvolvimento é coordenado pela *Open Handset Alliance* (OHA), um grupo liderado por empresas como Google, Motorola, LG, Samsung, Ericsson dentre outras (Lecheta (2009)).

O Android SDK é o *kit* de desenvolvimento que disponibiliza as ferramentas e APIs necessárias para desenvolver aplicações para a Plataforma Android, utilizando a linguagem Java. Como exemplo de ferramentas de apoio ao teste disponíveis no Android SDK podemos citar: *Espresso, UI Automator, AndroidJUnitRunner, Monkey* e *Monkeyrunner* (Google (2015)).

A Plataforma Android é organizada em camadas. De acordo com Pereira (2009), na camada **framework** (Application Framework), encontramos todas as APIs

e recursos utilizados pelos aplicativos, como classes visuais, que incluem listas, grades, caixas de texto, botões e até um navegador web embutido, View System (componentes utilizados na construção de aplicativos), provedor de conteúdo (Content Provider), que possibilita que uma aplicação possa acessar informações de outra aplicação, ou até mesmo compartilharem as suas informações, possibilitando a troca de informações entre aplicativos e gerenciadores de recursos (GPS e Cell HD), gerenciador de notificação (fornece informações sobre eventos que ocorre no dispositivo), de pacotes e de atividade, que controla todo o ciclo de vida da aplicação e o acesso à navegação entre as aplicações.

Podemos citar os principais elementos dessa camada:

- **Activity Manager:** Gerencia todo o ciclo de vida de todas as activities, quando iniciar e quanto terminá-las, possibilitando o deslocamento de uma activity para outra e assim por diante.
- **Package Manager:** É utilizada pelo activity manager para ler as informações dos APK's (Pacotes de arquivos do Android). A package manager se comunica com o resto do sistema e diz quais os pacotes estão sendo utilizados no dispositivo, quais são as capacidades desses pacotes.
- **Window Manager:** Basicamente, gerencia as apresentações de janela, qual a janela estará ativa e assim por diante.
- **Content Providers:** Possibilita o compartilhamento de dados entre os aparelhos, possibilitando a troca de informações entre os aplicativos.
- **View System:** Disponibiliza todo o tratamento gráfico para a aplicação, como botões, layouts e frames.

Encontra-se também disponível nessa camada outros elementos, como Location Service, Bluetooth Service, Wi-Fi Service, USB Service e Sensor Service. A Figura 1 adaptada de Pereira (2009) ilustra a Arquitetura do Android.

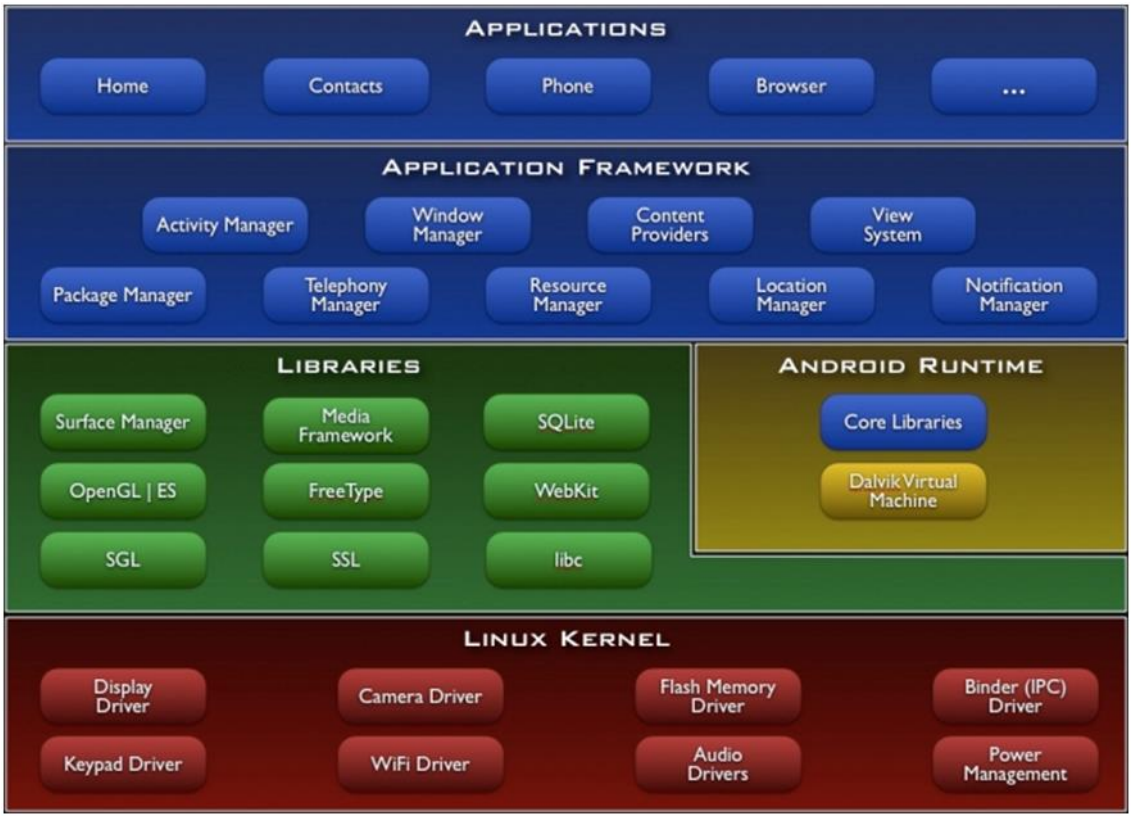

Figura 1 - Arquitetura do Android adaptada de Pereira (2009)

2.3 Métodos de Avaliação

Existem várias formas de comparar e avaliar ferramentas de teste. Através do QFD (*Quality Function Deployment* – Desdobramento da Função Qualidade) é possível identificar e priorizar os requisitos a fim de buscar uma melhor qualidade do produto. O QFD é uma técnica que pode ser usada para traduzir as necessidades dos clientes em requisitos técnicos de software (Akao (1996)). Uma adaptação do QFD para o desenvolvimento de software é denominada SQFD (Software Quality Function Deployment – Desdobramento da Função Qualidade de Software). Por meio deste método, nós realizamos a comparação das ferramentas de teste funcional de Aplicações Android. Segundo Krogstie (1999), o SQFD tem como objetivo melhorar o desenvolvimento de software aplicando técnicas de melhoria de qualidade, principalmente durante as especificações dos requisitos, confrontando as necessidades dos clientes com as restrições do próprio projeto, de modo a concentrar esforços nos aspectos de maior importância. Na seção seguinte, os passos básicos do SQFD serão detalhados.

2.3.1 SQFD

Para garantir a qualidade do software, o SQFD tem como alvo a fase de análise de requisitos. A matriz de qualidade usada no QFD tradicional é adaptada para o SQFD, sendo aplicada na tradução da voz do cliente (o que fazer), em requisitos técnicos (como fazer), e prioriza essas últimas de acordo com sua importância.

A maneira de coletar os dados, usados na definição dos requisitos, pode ser realizada através de entrevistas, observações, levantamento de dados e exame de dados históricos (caso exista). Após serem coletados, os dados são convertidos em tabelas de requisitos, a qual é revisada pelo cliente (Alves (2000)).

Segundo Haag et al. (1996), com a utilização do SQFD nas empresas, obtém-se diversos benefícios, destacando a redução de custos, definição mais rápida das características do produto, quantificação qualitativa dos requisitos do cliente e do registro de informações para a equipe de desenvolvimento e para a manutenção. E como desvantagem, Haag cita que no uso do QFD o tamanho da matriz atinge proporções enormes e isso também acontece no SQFD, além do fato de que sua implementação é difícil sem uma política de gerenciamento.

Os cinco passos do SQFD ilustrados na Figura 2, são detalhados a seguir com base no trabalho de Haag et al. (1996).

- **Passo 1 (Requisitos do cliente):**

  Realização do levantamento de requisitos (qualidades exigidas) do cliente. A partir de entrevistas, enquetes ou até mesmo técnicas de levantamento de ideias como o *Brainstorm*, as necessidades dos clientes devem ser identificadas. Os requisitos são declarações normalmente pequenas, especificamente registradas na terminologia dos clientes (isto é, “fácil de aprender”, “que tipo de ferramenta de teste você gostaria que existisse”) e são acompanhadas por uma definição detalhada. Após a identificação dos requisitos, eles são organizados em grupos hierárquicos (níveis), agrupando os que possuem conteúdo semelhante e, para cada grupo, deve ser definida uma expressão que traduza o conteúdo de todos os seus elementos. Os requisitos dos clientes são pesquisados e registrados no lado esquerdo do eixo y. Os clientes podem compreender os usuários

finais, gerentes, pessoal de desenvolvimento de sistema e pessoas que se beneficiariam do uso do produto de software proposto.

- **Passo 2 (Especificações técnicas do produto):**

  Em cooperação com os usuários do produto (clientes), os requisitos são convertidos para especificações técnicas e mensuráveis referentes ao produto de software e registrados no eixo x superior. Essas especificações técnicas correspondem às funções que as ferramentas de testes devem conter para atender aos requisitos levantados. Por exemplo, “fácil de aprender” pode ser convertido para “tempo para completar o tutorial”, “número de ícones” e “número de recursos de ajuda online”. É importante notar que alguns requisitos do cliente podem ser convertidos para múltiplas especificações técnicas do produto, tornando necessário um amplo envolvimento do cliente no processo. Adicionalmente, as especificações técnicas devem ser mensuráveis de alguma forma. As métricas predominantemente usadas são numéricas, podendo ser também booleanas. Por exemplo, o requisito do cliente "fornece vários formatos de impressão” pode ser convertido para “número de formatos de impressão” (métrica numérica) e “imprime em formato paisagem” (métrica booleana, sim/não).

- **Passo 3 (Matriz de correlação):**

  A partir de perguntas feitas ao cliente, é criada uma matriz de correlação identificando os pesos relacionados entre os vários requisitos do cliente e as especificações técnicas do produto. O grau de correlação pode ser “possível” (com valor 1), “fraco” (com valor 3), “forte” (com valor 9) ou pode não existir (em branco). Por exemplo, “fácil de aprender” está altamente correlacionado com “tempo para completar tutorial”, sendo que a alta correlação (forte) pode receber uma pontuação de 9 na matriz de correlação, mas não com “imprime em formato paisagem”, o que iria receber uma pontuação de 0 (inexistente) na matriz de correlação. Existindo uma possibilidade de correlação a pontuação poderia ser 1 (possível) ou caso exista uma baixa correlação a pontuação poderia ser 3

(fraca). Quando há muitos clientes envolvidos neste processo, é importante estabelecer um consenso quanto à intensidade dos relacionamentos.

- **Passo 4 (Requisitos prioritários do cliente):**

  Com base nos dados levantados pelo cliente e com sua ajuda, deve ser feita a priorização dos requisitos, listando-os no lado direito do eixo y. Uma maneira de se fazer isso é solicitar para cada cliente que participou da identificação das necessidades, a atribuição de um peso (uma nota) para cada requisito, gerando assim seu grau de importância. Podendo-se utilizar como valores um número entre um (1) a três (3), sendo três o mais importante. Depois a média dos valores atribuídos por cada participante é usada como o grau de importância de cada requisito em questão. No presente trabalho, tivemos apenas um cliente, no caso o LEDS, não necessitando de se calcular média, apenas atribuímos os pesos juntamente ao cliente.

- **Passo 5 (Especificações técnicas prioritárias):**

  O último passo consiste no desenvolvimento das especificações técnicas prioritárias do produto, sendo registradas na parte inferior do eixo x, onde soma-se os resultados obtidos da multiplicação dos requisitos prioritários do cliente pelos valores de correlação gerados entre os requisitos do cliente e as especificações técnicas do produto. Esse cálculo pode ser feito através de três etapas: (1) para cada aspecto técnico, verifica-se quais são os requisitos com os quais ele possui alguma correlação. (2) multiplica-se o grau de importância do requisito pelo grau de correlação com o aspecto técnico em questão. (3) soma-se os valores obtidos, resultando no peso do aspecto técnico. Estes pesos brutos de prioridades para as especificações técnicas do produto são convertidos para uma porcentagem total dos pesos brutos de prioridade. A equipe de desenvolvimento pode adicionar dados que estejam ligados às medidas das especificações técnicas do produto, simultaneamente com estimativas de custo, dificuldade e viabilidade do cronograma. O resultado final do

processo do SQFD deverá conter, no mínimo, as especificações técnicas mensuráveis dos produtos, seus percentuais de importância e as medidas principais. Esta informação será a entrada para o ciclo de desenvolvimento do software, ou no caso deste trabalho, para o método de avaliação de ferramentas para o teste funcional de Aplicações Android.

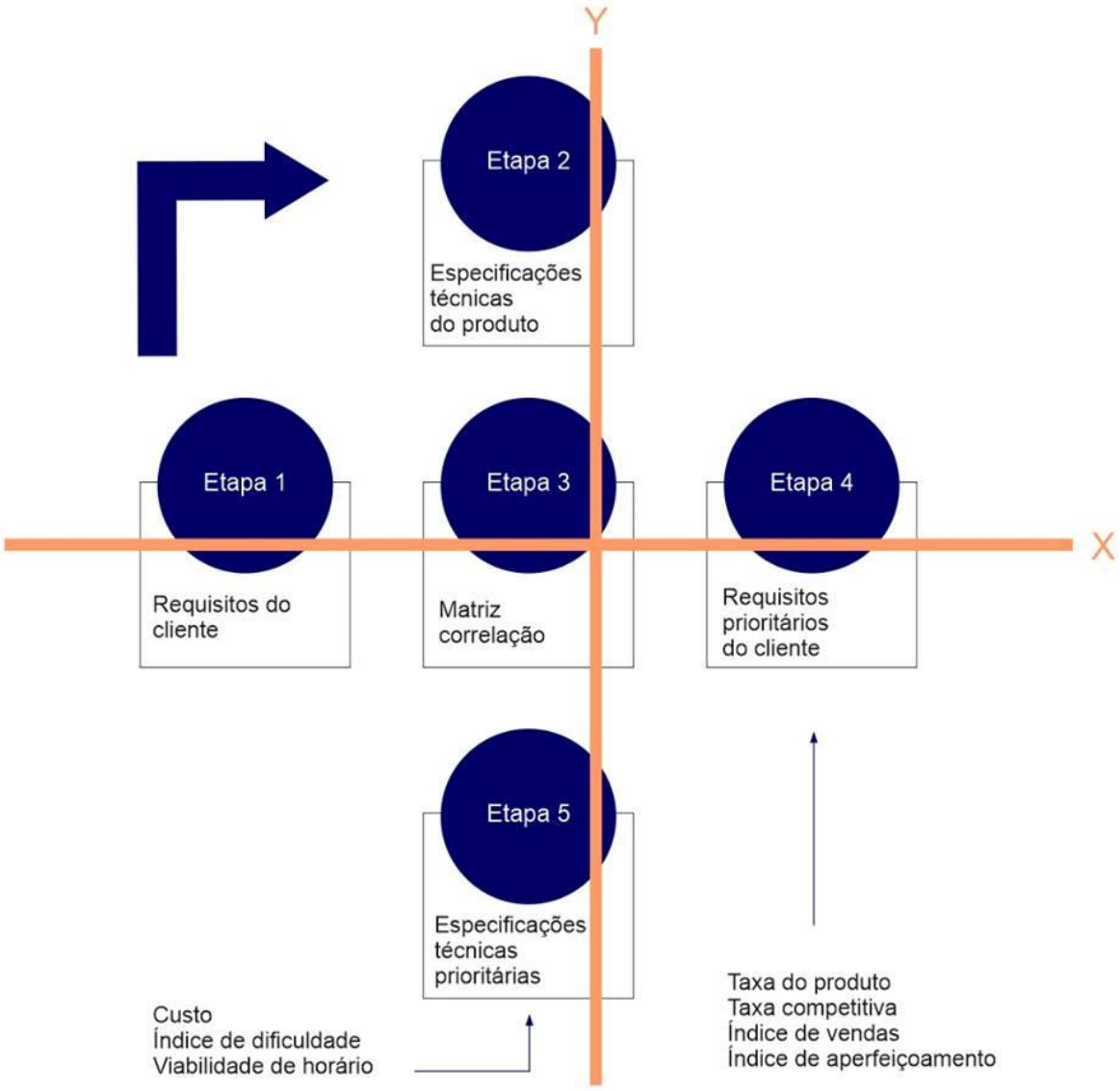

Figura 2 - SQFD - Modelo básico adaptado de Haag et al. (1996)

2.3.2 Método de Avaliação das Ferramentas de Apoio ao Teste

Utilizamos o Método de Avaliação proposto por Veloso et al. (2010). Esse método se diferencia por utilizar como base informações vindas dos usuários finais das ferramentas. Para isso, requisitos foram levantados e priorizados de acordo com

os passos do SQFD, descritos anteriormente. Esse método é aplicado em três passos e foi utilizado neste trabalho em conjunto com o SQFD. Abaixo uma visão geral dos passos:

- O primeiro passo consiste na definição de métricas de avaliação. Considerando a ferramenta analisada as métricas criadas em forma de questionários foram respondidas.
- O segundo passo define os níveis de pontuação. O nível de pontuação é avaliado levando em consideração as respostas dos questionários.
- O terceiro passo define a forma de julgamento. Relacionando os pesos das especificações técnicas encontradas pelo SQFD e a pontuação obtida com a aplicação do questionário, é possível comparar as ferramentas analisadas e com isso se obter um bom parâmetro para o cliente tomar sua decisão.

2.4 Conceitos de Usabilidade de Software

2.4.1 Interação Humano-Computador

O comportamento humano não é sempre igual, mas cheio de surpresas, o que dificulta o estabelecimento de “simples verdades” sobre o que esperar das pessoas em determinadas situações (Lindgaard (1994)).

Na mesma proporção em que os sistemas computacionais avançam, a quantidade de desafios para os profissionais desenvolvedores na área também aumenta. Um dos principais desafios está relacionado à área da Interação Humano-Computador (Nielsen (1993)).

A comunicação entre o programa e ser humano se dá através da interface de um sistema. Quando fatores humanos são considerados, este diálogo acontece em harmonia. Portanto, a interface é considerada a “embalagem do software de computador”. Dessa maneira, se ela for fácil de aprender, simples de usar, direta e amigável, o usuário estará propenso a fazer bom uso da mesma (Pressman (2014)). Quando essas características são omitidas, problemas de usabilidade possivelmente irão ocorrer.

Quando um usuário ou grupo de usuários encontram dificuldades em realizar uma tarefa com uma interface, certamente problemas de usabilidade estarão

relacionados a essas dificuldades. Podendo essas dificuldades ter origens variadas e ocasionar perdas de dados, diminuição da produtividade, ocasionando até a rejeição total do software por parte dos usuários.

Em termos gerais, a área de Interação Humano-Computador (IHC) analisa o "projeto (design), avaliação e implementação de sistemas computacionais interativos para uso humano, juntamente com os fenômenos associados a este uso" (Hewett et al. (1992)).

Segundo Moraes (2012), Interação Humano-Computador é um campo de estudo interdisciplinar que tem como objetivo geral entender como e porque as pessoas utilizam (ou não) a tecnologia da informação. Por isso, o processo de design de interação é fundamental para todas as disciplinas, campos e abordagens que se preocupam com a pesquisa e projeto de sistemas baseados em computador (Preece et al. (2013)).

Barbosa et al. (2010) destacam a Usabilidade, a Acessibilidade e a Comunicabilidade como os três principais requisitos não-funcionais que a interação e a interface devem conter para ser consideradas adequadas ao usuário final.

Os "sistemas computacionais" antes eram compostos, na sua grande maioria, por CPU, monitor, teclado, mouse e telas diversas. Equipamentos portáteis, que utilizam tecnologias de comunicação sem fio, vem mudando a forma como as pessoas interagem com informações e serviços que antes eram acessados apenas por meio de computadores fixos, em casa ou no local de trabalho. Novas aplicações, equipamentos e serviços estão surgindo para satisfazer as necessidades do usuário móvel (Cybis (2010)).

Os objetivos de IHC estão relacionados a desenvolver e melhorar sistemas computacionais para que os usuários possam executar suas tarefas com segurança, eficiência e satisfação. Esses aspectos são conhecidos em geral como usabilidade (Moraes (2012)).

2.4.2 Usabilidade

A norma ISO/IEC 25010, de 2010, sobre qualidade de software, foi a primeira norma a definir um conceito sobre usabilidade, na qual considera usabilidade como "um conjunto de atributos de software relacionado ao esforço necessário para seu

uso e para a avaliação individual de tal uso por determinado grupo de usuários" (Dias (2007)).

A usabilidade é a capacidade de um produto ou sistema, em termos funcionais, de ser usado com facilidade e eficácia por um segmento específico de usuários, concedendo-lhes treinamento e suporte específico, visando a execução de um conjunto específico de tarefas, em um contexto de utilização igualmente específico (Moraes (2012)).

De acordo com Nielsen (1993), a usabilidade é apresentada como um dos componentes da aceitabilidade de um sistema, e se refere à questão de saber se o sistema é bom o suficiente para satisfazer todas as necessidades e exigências dos usuários e *stakeholders*, por exemplo clientes e gerentes. Nielsen afirma que a usabilidade pode ser caracterizada pelas seguintes metas:

- **Facilidade no aprendizado:** Tempo e esforço gastos pelo usuário em aprender e utilizar o sistema com determinada compreensão em suas primeiras interações. O usuário interage rapidamente com o sistema.
- **Eficiência:** Uma vez utilizado e aprendido sobre o sistema, refere-se ao tempo necessário para conclusão de uma tarefa. O usuário consegue localizar a informação desejada.
- **Facilidade de memorização:** Os usuários têm clareza de como realizar as tarefas do sistema, uma vez que poucas vezes o utilizaram. Refere-se ao esforço cognitivo de memorização na realização das tarefas do sistema.
- **Minimização de erros:** Sempre que os usuários cometerem ações que os levem a erros, estes devem ser mínimos e o sistema deve ser capaz de desfazer os erros cometidos.
- **Satisfação:** Baseada na avaliação subjetiva do usuário sobre o sistema cumprir ou não o que realmente foi solicitado.

2.4.3 Métodos de Avaliação de Usabilidade

De acordo com Dias (2007) e Rosa (2013), a avaliação de usabilidade de um sistema interativo deve verificar o desempenho (eficácia e eficiência) da interação homem-computador, obtendo sinais do nível de satisfação do usuário, identificando

problemas de usabilidade durante a realização das tarefas específicas em seu contexto de uso. Um problema de usabilidade é definido como uma característica, em determinada situação, que possa retardar, prejudicar ou impossibilitar a realização de uma tarefa, desagradando, limitando ou traumatizando o usuário.

Ainda segundo Dias (2007) e Rosa (2013), os métodos de avaliação de usabilidade podem ser classificados em três grupos, de acordo com sua natureza: métodos de teste com usuários, métodos baseados em modelos e métodos de inspeção. O presente trabalho apresenta de forma sucinta os métodos de teste com usuários e métodos baseados em modelos, para depois focar no método de inspeção, destacando, dentre eles, a avaliação heurística, a qual foi usada para avaliar as ferramentas para o teste funcional de Aplicações Android.

- **Métodos de teste com usuários:** Especificado pela participação direta dos usuários do sistema na avaliação, os métodos de testes com usuários podem ser prospectivos, usando questionários e entrevistas para coletar as opiniões e experiências dos usuários ou podem ser empíricos, neste caso, adotam-se técnicas de observação do uso dos sistemas em situações reais (Dias (2007), Rosa (2013)).
- **Métodos baseados em modelos:** Procura analisar o grau de usabilidade de um sistema, a partir de modelos de sua interface e/ou de seus usuários, ou seja, de representações que capturam e modelam a forma através da qual os usuários interagem com o sistema (Dias (2007), Rosa (2013)).
- **Métodos de inspeção (Avaliação Heurística):** Baseiam-se na avaliação de usabilidade realizada por especialistas. Os avaliadores segundo Nielsen (1993) podem ser especialistas em usabilidade, consultores de desenvolvimento de software, usuários finais com conhecimento especializado ou outros profissionais, e a qualidade da avaliação ocorre no seu julgamento (Rosa (2013)).

2.4.4 Avaliação Heurística

Segundo Nielsen (1993), a avaliação heurística é um método analítico que visa identificar problemas de usabilidade de acordo com um conjunto de heurísticas

ou normas. Nielsen recomenda que a avaliação seja aplicada por vários indivíduos de maneira isolada, a fim de garantir a independência das diferentes avaliações, evitando assim a ocorrência de tendências, decorrente da interação entre os avaliadores. Mesmo que esse tipo de avaliação possa ser feita por um único indivíduo, sua eficiência aumenta com o número de avaliadores.

Segundo Rosa (2013), os resultados das avaliações podem ser elaborados como relatórios escritos, tendo como vantagem se ter um registro formal do processo, mas exigem a participação de um coordenador para reunir as avaliações dos diferentes especialistas. Diálogos entre os avaliadores, durante a interação com o sistema, podem ser analisados por um observador. Entretanto, Nielsen ressalta que o observador, ao contrário do que ocorre nos testes com usuários, não deve interpretar os comentários dos avaliadores, mas deve limitar-se a registrá-los. Com liberdade do formato escolhido, o resultado gerado pela avaliação heurística sempre será uma lista dos problemas ocorridos de usabilidade na interface, relativos a princípios de usabilidade que, de acordo com o julgamento do avaliador violam o design atual (Nielsen (1993)).

Segundo Nielsen (1993), uma avaliação heurística típica consome normalmente entre uma e duas horas. Ao se avaliar interfaces de alta complexidade, como no caso de ferramentas para o teste funcional de Aplicações Android, pode ser necessária uma maior duração da avaliação, podendo ser dividida em diferentes sessões. O avaliador pode decidir a forma em que irá proceder na avaliação, mas a recomendação seguida no presente estudo é interagir pelo menos duas vezes com o sistema. No primeiro contato, o fluxo geral da navegação e o escopo macro da interface são analisados. Já no segundo contato, elementos específicos do sistema são ressaltados. A base da metodologia heurística é a comparação do sistema avaliado com as heurísticas pré-determinadas, mas o avaliador tem a liberdade, durante o método, para considerar outros princípios de usabilidade que julgue relevantes para o sistema estudado (Rosa (2013)).

Segundo Dias (2007) e Rosa (2013), os problemas de usabilidade podem ser classificados de acordo com o grau de impacto sobre a interação do usuário com o sistema em:

- **Barreiras:** Quando o usuário não consegue solucionar o problema.

- **Obstáculos:** Quando o usuário pode aprender a solucionar o problema, mas enquanto isso seu desempenho em relação ao sistema fica comprometido.
- **Ruídos:** Problemas de menor impacto, por causarem uma diminuição do desempenho, mas não significativa.

2.4.5 Heurísticas de Usabilidade

Originado dos estudos da História, o conceito de *heurística* se refere a uma pesquisa crítica de documentos para a descoberta de fatos (Souza (2004)).

No início da década de 90, Jackob Nielsen e Rolf Molich abordaram esse conceito e aplicaram um método em que um pequeno grupo de avaliadores examinavam uma interface e iam em busca de problemas que não se enquadravam como princípios gerais para um bom projeto, esses princípios passaram a ser chamados de princípios heurísticos (Santos (2000)).

Posteriormente, Nielsen (1993) realizou uma análise de problemas encontrados sobre usabilidade e criou um conjunto de princípios para a avaliação heurística. Esses princípios foram sintetizados em dez heurísticas de usabilidade:

- **Visibilidade do estado do sistema;**

  O sistema deve manter o usuário informado continuamente e apropriadamente do que está acontecendo a cada momento, por meio de feedback, executando-o em um tempo razoável.

- **Mapeamento entre o sistema e o mundo real;**

  O sistema deverá falar a linguagem do usuário, a terminologia e elementos da interface devem ser familiares, ao invés de falar uma linguagem utilizando termos técnicos e de difícil entendimento do usuário.

- **Liberdade e controle ao usuário;**

  Deverá ser possibilitado ao usuário, opções para desfazer e refazer ações que foram executadas por engano, abortar uma tarefa e retornar ao estado anterior. O usuário controla o sistema.

- **Consistência e padrões;**

  Os usuários não deverão parar para pensar se determinadas palavras, situações ou ações significam, de fato, a mesma coisa no sistema. Devem-se utilizar ícones e palavras consistentes, padronizadas e de fácil entendimento.

- **Prevenção de erros;**

  Ter boas mensagens de erros e facilitar a utilização do sistema prevenindo a ocorrência de problemas. Antes do usuário executar determinadas ações, o sistema deve apresentar opções de confirmação ou eliminar as condições que possam levar as falhas.

- **Reconhecer em vez de relembrar;**

  O usuário não deverá ter que lembrar sobre o funcionamento de um comando especifico, o sistema deverá minimizar a quantidade de informação que o usuário precisará lembrar para usá-lo. Sempre que necessário, as informações deverão estar visíveis ou facilmente recuperáveis e de fácil entendimento e acessibilidade.

- **Flexibilidade e eficiência de uso;**

  Adequar o sistema para qualquer tipo de usuário, ou seja, tanto para os usuários experientes, como também para os inexperientes. Usuários devem ter a opção de personalizar ações frequentes.

- **Design estético e minimalista;**

  Todas as informações extras não deverão aparecer com as informações relevantes, pois neste caso poderão diminuir a sua visibilidade. Apenas diálogos importantes devem ser exibidos. Deve-se apresentar informações que o usuário precisa naquele instante, nem mais nem menos.

- **Suporte para o usuário reconhecer, diagnosticar e recuperar erros;**

As mensagens de erros deverão ser exibidas em linguagem compreensível (sem códigos), indicando o problema e informando ao usuário uma solução de forma construtiva.

- **Ajuda e documentação;**

  É preferível que o sistema possa ser usado sem documentação, caso contrário, o usuário deverá encontrar fácil a documentação no sistema, deverão ser exibidos passos concretos e focados na tarefa, e estes não devem ser muitos extensos.

Podendo ser aplicada em qualquer dos ciclos de desenvolvimento da interface, a avaliação heurística se destaca por ser um meio bastante eficaz para a usabilidade de um projeto (Souza (2004); Santos (2000)). Neste trabalho, utilizamos as dez heurísticas definidas por Nielsen para a avaliação de usabilidade das ferramentas para o teste funcional de Aplicações Android.

3 Método

Inicialmente, realizamos o levantamento de ferramentas para o teste de Aplicações Android, independente se as ferramentas possibilitassem também o teste de aplicações de outras plataformas (Ex: iOS, Windows Phone, BlackBerry). Através de buscas na Internet, leituras em artigos que citavam ferramentas para teste, foram selecionadas 49 ferramentas. Para cada ferramenta selecionada, levantamos algumas informações como: ambiente de execução, linguagem em teste, licença (comercial, gratuita ou open-source), data da primeira e última versão, link para download e observações, caso necessário.

A Tabela 1 apresenta um trecho extraído da tabela completa com todas as ferramentas e suas informações. Vale destacar que os espaços em branco na tabela indicam que não foi possível identificar uma determinada informação a respeito da ferramenta.

Em seguida, dentre as 49 ferramentas levantadas, foram selecionadas três ferramentas de código aberto para serem avaliadas, de acordo com a opção do cliente. Essa escolha contou com a ajuda de um profissional com conhecimentos mais abrangentes em relação a ferramentas de teste, levando em consideração a

qualidade da documentação das ferramentas. As ferramentas escolhidas foram: *Robotium*, *Maveryx* e *Calabash*. A seguir apresentamos uma breve descrição sobre essas ferramentas.

Robotium: É um framework de código aberto de testes funcionais automatizados para Plataforma Android. Tem seu foco em testes de caixa-preta automatizados. Os testes são escritos em Java e são instalados em um dispositivo ou emulador e executado como um aplicativo separado. Simula a interação entre usuário e o software.

Maveryx: É uma ferramenta de testes funcionais e de regressão para Aplicações Android capaz de rodar em Windows, Linux e Mac. Os testes são escritos em Java. Esta ferramenta conta com uma interface capaz de detectar objetos nas interfaces das aplicações que testa, podendo gerar os testes automaticamente a partir, por exemplo, de botões e formulários disponíveis. Possui integração com o IDE Eclipse, a ferramenta é capaz de detectar modificações e defeitos na interface da aplicação.

Calabash: É uma ferramenta de automação de teste para Android e iOS. Calabash é open-source desenvolvido e mantido pela empresa Xamarin. O desenvolvimento com Calabash é feito na linguagem Ruby.

A próxima seção apresenta o estudo de caso envolvendo a aplicação do SQFD. Na seção posterior, apresentamos uma discussão sobre a usabilidade das ferramentas selecionadas.

Tabela 1 - Trecho extraído da tabela completa de ferramentas para o teste de Aplicações Android

Nome da Ferramenta	Ambiente de Execução	Tipo da Aplicação em teste	Licença	Data da Primeira Versão	Data da Última Versão	Link para Download	Observações
Android Ripper	Windows	Android	Open Source	01/06/2012	01/07/2013	https://github.com/reverse-unina/AndroidRipper/wiki/First-AndroidRipper-Execution	

Appium	Windows, Mac	Android, iOS	Open Source	03/06/2013	08/12/2015	http://appium.io/	
Calabash	Windows, Mac	Android, iOS	Open Source	27/04/2012	03/02/2016	http://calaba.sh/	
Convertigo	Windows, Linux, Mac	Android, iOS, Windows Phone, BlackBerry	Open Source	2009	2012	http://www.convertigo.com/download/	
Remote TestKit	Windows,Mac	Android,iOS	Open Source/Comercial			https://appkitbox.com/en/testkit/download	
Robolectric	Windows	Android	Open Source	14/05/2013	07/07/2015	http://mvnrepository.com/artifact/org.robolectric/robolectric	
Scirocco	Windows	Android	Open Source	21/03/2011	05/07/2011	https://code.google.com/p/scirocco/downloads/detail?name=scirocco_v2.0.jar&can=2&q=	Outra fonte para download disponível: https://github.com/sonixlabs/scirocco-webdriver/blob/master/QuickStart.markdown
Ubertesters Platform	Windows,Mac	Android,iOS	Comercial	2012	12/01/2015	http://ubertesters.com/download-sdk-android/	

3.1 Estudo de Caso

Os requisitos inicialmente identificados por Caldeira (2014) para ferramentas de apoio ao Teste Funcional foram validados junto ao cliente para confirmar a sua adequação ao contexto deste estudo, ou seja, ferramentas para o Teste Funcional de Aplicações Android. Para essa finalidade, foram realizadas reuniões com o cliente. Neste contexto, apresentamos a aplicação dos passos do SQFD.

3.1.1 Passo 1 do SQFD (Requisitos do Cliente)

Trinta requisitos identificados foram organizados em grupos hierárquicos (níveis) e agrupados por semelhança de conteúdo semelhante. Para cada grupo, foi definida uma expressão que traduza o conteúdo de todos os seus elementos. O agrupamento faz parte da construção da Tabela de Desdobramento das Qualidades Exigidas, apresentada na Tabela 2.

Tabela 2 - Tabela de Desdobramento das Qualidades Exigidas

	Nível 1 - Grupo	Nível 2 - Requisito
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
		Gestão 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 reincidência 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 testes 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

3.1.2 Passo 2 do SQFD (Especificações Técnicas do Produto)

Os requisitos levantados no passo anterior foram convertidos para especificações técnicas do produto de software. Essas especificações técnicas são as funções que as ferramentas deverão conter para atender aos requisitos levantados anteriormente. Cada requisito pode estar associado a mais de uma especificação técnica e uma especificação técnica pode atender mais de um requisito. Selecionamos 30 especificações técnicas que serão mostradas na Tabela 3 que é a Tabela de Desdobramento dos Elementos da Qualidade organizados em grupos e níveis.

Tabela 3 - Tabela de Desdobramento dos Elementos da Qualidade

	Nível 1 - Grupo	**Nível 2 – Aspecto Técnico**
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 funções 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 softwares 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

3.1.3 Passo 3 do SQFD (Matriz de Correlação)

Neste passo foi construída a Matriz de Correlação, também chamada de Matriz da Qualidade ou Casa da Qualidade. Essa matriz é formada identificando os pesos relacionados entre os vários requisitos do cliente e as especificações técnicas do produto. Cada requisito com cada especificação técnica. O grau de correlação

pode ser “possível” (com valor 1), “fraco” (com valor 3), “forte” (com valor 9) ou pode não existir (em branco). A Tabela 4 apresenta um excerto da Tabela Matriz de Qualidade.

Tabela 4 - Trecho extraído da Matriz de Qualidade

			Nível 1	Avaliador de testes	Suporte da ferramenta				Auxílio da ferramenta		
			Nível 2 (Especificação Técnica)	Avaliador de cobertura	Cadastro de usuários	Cadastro de grupos	Cadastro de projeto	Cadastro de equipe	Help on-line	Manual de usuário	Sítio de apoio com exemplos de uso
	Nível 1	**Nível 2 (Requisitos)**									
1	Planejamento da atividade de teste	Apoio para a criação de um Plano de testes			9	9	9	9			
		Apoio para estimativa de prazos para as atividades de teste					3	3			
		Apoio para identificação da complexidade das partes do sistema									
		Integração com ferramentas de gerenciamento de projeto.			3	3	3	3			
2	Avaliação de resultados	Avaliação da cobertura e qualidade dos testes		9							
3	Documentação de testes	Documentação dos casos de testes e procedimentos de teste									
		Documentação gerencial dos testes		9	9	9	9	9			
		Documentação de execução dos testes		9							
		Fácil visualização da documentação								9	

3.1.4 Passo 4 do SQFD (Requisitos Prioritários do Cliente)

No quarto passo, com base nos requisitos levantados pelo cliente (Passo 1) foi feita a priorização desses requisitos, atribuindo um peso (uma nota) para cada

requisito, gerando assim o grau de importância que o requisito tem em relação a característica que se deseja que a ferramenta tenha. Utilizamos como valores um numero de um (1) a três (3), sendo três o mais importante. A Tabela 5 mostra a tabela completa de requisitos priorizados.

Tabela 5 - Tabela completa de requisitos priorizados

Nº	Requisito	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	**1**
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	**1**
8	Geração de um povoador a partir de um banco de dados já povoado	**1**
9	Gestão 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**
16	Identificação da produtividade dos testadores	**1**
17	Verificação da reincidência de falhas	**1**
18	Avaliação da cobertura/qualidade dos testes	**1**
19	Documentação dos casos de testes e procedimentos de teste	**3**
20	Documentação de execução dos testes	**3**
21	Documentação gerencial dos testes	**1**
22	Fácil visualização da documentação	**1**
23	Rastreabilidade entre os teste e os artefatos relacionados (código, ERSw, Desenho)	**1**
24	Identificação de testes afetados por mudanças em artefatos relacionados	**1**

25	Facilidade para criação de testes adicionais aos testes gerados automaticamente	**1**
26	Importação de testes criados em outras ferramentas de teste	**2**
27	Ser gratuita ou baixo custo	**3**
28	Possuir boa documentação	**3**
29	Funcionamento em múltiplas plataformas: Windows, Linux.	**2**
30	Boa usabilidade, favorecendo a produtividade dos seus usuários	**3**

3.1.5 Passo 5 do SQFD (Especificações Técnicas Prioritárias)

O último passo envolveu o desenvolvimento das especificações técnicas prioritárias do produto, não necessitando mais do envolvimento do cliente. A priorização se refere a um peso que cada aspecto técnico recebe e foi feita somando-se os resultados obtidos da multiplicação do grau de importância dos requisitos prioritários do cliente pelo grau de correlação do aspecto técnico em questão. Na Figura 3 é exibido um excerto adaptado da Matriz de Qualidade completa, onde pode-se notar o aspecto técnico “Seguir um guia de estilo”. O aspecto técnico possui correlação com os seguintes requisitos:

- Fácil visualização da documentação – Grau de correlação 9 e grau de importância 1;
- Possuir boa documentação – Grau de correlação 3 e grau de importância 3;
- Boa usabilidade, favorecendo a produtividade dos seus usuários – Grau de correlação 9 e grau de importância 3;

Assim, fazendo a multiplicação dos graus de correlação pelos graus de importância e realizando a soma dos produtos, obtém-se o valor 45 (9 x 1 + 3 x 3 + 9 x 3 = 45). Logo, 45 é o peso (priorização) do aspecto técnico “Seguir um guia de estilo”. Os pesos são convertidos em valores percentuais, obtidos dividindo-se o peso encontrado pela soma dos pesos de todos os aspectos técnicos.

				Funções da ferramenta								
			Nível 1	Suporte da ferramenta				Arquitetura da ferramenta				
			Nível 2	Cadastro de usuários	Cadastro de grupos	Cadastro de projeto	Cadastro de equipe	Uso de software livres	Seguir um guia de estilo	Utilizar terminologia adequada ao contexto		Grau de Importância
	Nível 1	Nível 2										
7	Documentação de testes	Documentação dos casos de testes e procedimentos de teste	19								48	3
		Documentação gerencial dos testes	20	9	9	9	9				66	1
		Documentação de execução dos testes	21								48	3
		Fácil visualização da documentação	22						9	9	36	1
11	Aquisição e implantação	Ser gratuita ou baixo custo	27					9			9	3
		Possuir boa documentação	28						3	9	39	3
12	Funcionamento	Funcionamento em múltiplas plataformas: windows, linux.	29					3			3	2
13	Usabilidade e desempenho	Boa usabilidade, favorecendo a produtividade dos seus usuários	30						9	9	18	3
				21	21	24	24	12	21	27		
Peso Absoluto									45,0			
Peso Relativo									3,5%			

Figura 3 – Trecho adaptado extraído da Matriz de Qualidade completa

A Tabela 6 mostra todos os aspectos técnicos priorizados.

Tabela 6 - Tabela completa de aspectos técnicos priorizados

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 funções 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%
16	Cadastro de usuários	3,3%
17	Cadastro de grupos	3,3%
18	Cadastro de projeto	3,5%
19	Cadastro de equipe	3,5%
20	Uso de softwares livres	2,6%
21	Seguir um guia de estilo	3,5%
22	Utilizar terminologia adequada ao contexto	4,9%
23	Help on-line	2,1%
24	Manual de usuário	2,8%
25	Sítio de apoio com exemplos de uso	2,1%
26	Agrupador e escalonador de testes	2,3%
27	Gerador de log de execução de testes	5,6%
28	Comparador de arquivos ignorando padrões configuráveis	1,2%

29	Povoador de dados	7,2%
30	Analisador de falhas	1,0%

Conforme Santos et al. (2010), é possível avaliar as ferramentas de testes com os requisitos e aspectos técnicos priorizados, e também elaborar comparações entre as ferramentas utilizando essas informações.

3.1.6 Método de Avaliação das Ferramentas

Juntamente com o SQFD, utilizamos o método de avaliação de ferramentas de teste utilizado inicialmente por Veloso et al. (2010) e posteriormente por Caldeira (2014). Esse método é composto por três passos: Definição de Métricas de Avaliação; Definição dos Níveis de Pontuação; Definição da Forma de Julgamento.

3.1.6.1 Definição de Métricas de Avaliação

Para fazer a avaliação das ferramentas, definimos que a métrica de avaliação será em forma de questionários que serão respondidos de acordo com a ferramenta analisada. Para cada aspecto técnico, são elaboradas perguntas (podendo haver mais de uma pergunta para determinado aspecto técnico) que são utilizadas para avaliar o atendimento do aspecto técnico em relação às ferramentas em teste.

3.1.6.2 Definição dos Níveis de Pontuação

Os níveis de pontuação são importantes, pois demostram o quanto a ferramenta analisada satisfaz os aspectos técnicos. Três níveis de pontuação foram utilizados:

- 0 (zero): Significa que a ferramenta NÃO atende o aspecto técnico.
- 0,5 (meio): Significa que a ferramenta atende PARCIALMENTE o aspecto técnico.
- 1 (um): Significa que a ferramenta atende ao aspecto técnico (SIM).

3.1.6.3 Definição da Forma de Julgamento

A forma de julgamento das ferramentas adotada no nosso trabalho foi:

- Primeiramente, todas as ferramentas de testes escolhidas foram instaladas, configuradas e exploradas para conseguirmos fazer uma análise de qual atenderia melhor o cliente. Para utilização das ferramentas, utilizamos aplicações de código aberto disponíveis em repositórios na Internet.
- Após as ferramentas terem sido exploradas, é aplicado o questionário definido pelo método de avaliação das ferramentas. Para cada pergunta do questionário é necessário escolher uma resposta na qual se tem uma pontuação correspondente (Sim – 1; Parcialmente – 0,5; Não – 0). Para cada aspecto técnico, deve existir apenas uma pontuação. Portanto, se existir mais de uma pergunta para um mesmo aspecto técnico, é preciso fazer a média das pontuações das perguntas para se obter um único valor resultante. A Tabela 7 mostra um excerto extraído da aplicação do questionário completo sobre as ferramentas avaliadas. Utilizando-se como exemplo da Tabela o aspecto técnico “Utilizar terminologia adequada ao contexto”, do grupo “Arquitetura da Ferramenta”. Notamos que ele possui duas perguntas. Levando em consideração a ferramenta *Robotium*, a primeira pergunta (“A ferramenta usa termos conhecidos, apoiados por padrões da indústria como IEEE, ISO e ABNT?”) foi respondida como SIM (pontuação 1), e a segunda pergunta (“A ferramenta possui uma linguagem de fácil entendimento?”) foi respondida também como SIM (pontuação 1). Contudo, para se obter um resultado coerente foi preciso fazer a média das pontuações (1 + 1) / 2 = 1. Portanto, 1 é a pontuação do aspecto técnico “Utilizar terminologia adequada ao contexto” considerando a análise feita para a ferramenta de teste *Robotium*.

Tabela 7 - Trecho extraído da aplicação do questionário completo sobre as ferramentas

				Ferramentas			
				Robotium		Maveryx	
Grupo	**Aspecto Técnico**	**Pergunta**	**Opções**	**Resposta**	**Pontuação**	**Resposta**	**Pontuação**
Arquitetura da Ferramenta	Uso de software livres	A ferramenta se integra com outras ferramentas gratuitas?	SIM (1)	x	1	x	1
			PARCIALMENTE (0.5)				
			NÃO (0)				
	Seguir um guia de estilo	A ferramenta utiliza um padrão para o desenho da interface com o usuário ?	SIM (1)	x	1	x	1
			PARCIALMENTE (0.5)				
			NÃO (0)				
	Utilizar terminologia adequada ao contexto	A ferramenta usa termos conhecidos, apoiados por padrões da indústria como IEEE, ISO e ABNT?	SIM (1)	x	1	x	1
			PARCIALMENTE (0.5)				
			NÃO (0)				
		A ferramenta possui uma linguagem de fácil entendimento?	SIM (1)	x		x	
			PARCIALMENTE (0.5)				
			NÃO (0)				

- Finalizando o julgamento das ferramentas foi construída uma tabela que mostra a comparação das três ferramentas analisadas, relacionando o grau de importância de cada aspecto técnico (peso) com a pontuação obtida no questionário.

A Tabela 8 mostra um trecho extraído da tabela completa de comparação das ferramentas de testes selecionadas. Essa tabela apresenta as colunas definidas a seguir:

- **Grupos:** Grupos de aspectos técnicos identificados.
- **Aspectos Técnicos:** Os aspectos técnicos relacionados ao seu grupo.

- **Pesos:** O peso em porcentagem de cada aspecto técnico obtido na Matriz de Correlação.
- **Pontuação:** Pontuação obtida no questionário para cada aspecto técnico em relação à ferramenta analisada.
- **Atendido:** Somatório da multiplicação de cada valor da coluna “Pesos” por cada valor da mesma linha da coluna “Pontuação”. Os valores encontrados em porcentagem representam o quanto os aspectos técnicos atendem a necessidade de um cliente. Para cada ferramenta avaliada existe uma coluna “Atendido”.

Tabela 8 - Trecho extraído da tabela completa de comparação entre as ferramentas de teste funcional de Aplicações Android: *Robotium*, *Maveryx* e *Calabash*

			FERRAMENTAS					
			Robotium		Maveryx		Calabash	
Grupo	Aspectos Técnicos	Pesos	Pontuação	Atendido	Pontuação	Atendido	Pontuação	Atendido
Auxílio da ferramenta	Help on-line	2,1	0,25	2,625	1	7	0	2,1
	Manual de usuário	2,8	0		1		0	
	Sítio de apoio com exemplos de uso	2,1	1		1		1	

Somando-se as linhas da coluna “Pesos” do grupo “Auxílio da Ferramenta” da Tabela 8, notamos que os aspectos técnicos deste grupo representam 7% da necessidade total do cliente (2,1 + 2,8 + 2,1 = 7). Considerando a ferramenta *Calabash* e aplicando os cálculos explicados anteriormente em “Atendido” sobre o grupo “Auxílio da Ferramenta”, constatamos que essa ferramenta atende a 2,1% (2,1 x 0 + 2,8 x 0 + 2,1 x 1 = 2,1) da necessidade do cliente.

A escolha da ferramenta de testes que melhor atende as necessidades do cliente é calculada através da soma de todas as linhas da coluna “Atendido” de cada ferramenta, sendo que o resultado em porcentagem representa o quanto a ferramenta analisada atende as necessidades totais. Portanto, a ferramenta que obtiver o maior resultado é a mais adequada para o cliente em relação as suas necessidades.

4 Resultados e discussão sobre usabilidade

4.1 Resultados do SQFD

Todas as etapas desse trabalho foram feitas com o intuito de identificar a ferramenta que melhor atende às necessidades do cliente. Portanto, como resultados obtidos da aplicação do SQFD e o Método de Avaliação para comparação das ferramentas para o teste funcional de Aplicações Android: *Robotium, Maveryx* e *Calabash,* temos:

- **Robotium:** Atende a 27,58% das necessidades do cliente.
- **Maveryx:** Atende a 35,28% das necessidades do cliente.
- **Calabash:** Atende a 25,85% das necessidades do cliente.

A Tabela 9 apresenta o resultado da avaliação realizada. Nota-se que na última linha da tabela, no campo “Satisfação do Cliente”, temos os resultados em porcentagem das três ferramentas, ressaltando que esses resultados são obtidos através do somatório de todas as linhas da coluna “Atendido” de cada ferramenta.

Tabela 9 - Tabela completa de comparação entre as ferramentas de teste funcional de Aplicações Android: *Robotium*, *Maveryx* e *Calabash*

				FERRAMENTAS					
				Robotium		Maveryx		Calabash	
	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	0,5	2,55	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	0,5	5,9	0	5,4	0	5,4
		Uso de linguagens de alto nível	1,2	1		1		1	
		Acesso as funcoes do SO	2,1	1		1		1	
		Acesso ao mecanismo de persistência	2,1	1		1		1	
5	Integrador	Integração com ferramentas de acompanhamento de falhas	5,1	0	2,7	0,25	3,975	0	2,7
		Integração com ferramentas de gestão de configuração	5,4	0,5		0,5		0,5	
6	Avaliador de testes	Avaliador de cobertura	3,5	0	0	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	11	1	11	1	11
		Seguir um guia de estilo	3,5	1		1		1	
		Utilizar terminologia adequada ao contexto	4,9	1		1		1	
10	Auxílio da ferramenta	Help on-line	2,1	0,25	2,625	1	7	0	2,1
		Manual de usuário	2,8	0		1		0	
		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	5,35	0,5	5,35	0,5	4,65
		Gerador de log de execução de testes	5,6	0,75		0,75		0,625	
		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		**27,58%**		**35,28%**		**25,85%**

De acordo com a Tabela 9, podemos ver que os resultados obtidos mostram que a *Maveryx* foi a ferramenta que mais satisfaz as necessidades do LEDS, seguida pela ferramenta *Robotium* e depois a ferramenta *Calabash*.

Podemos notar que no grupo “Arquitetura da Ferramenta” todas as três ferramentas obtiveram a mesma pontuação, demostrando que as ferramentas possuem particularidades em comum. Já no grupo “Gerador de testes funcionais” com o aspecto técnico “Gerador de entradas utilizando critérios” somente a ferramenta *Maveryx* atendeu ao aspecto técnico analisado. Analisando o grupo “Gerador manual de testes”, a ferramenta *Robotium* se sobressaiu no atendimento em relação às demais ferramentas, obtendo uma maior pontuação no aspecto técnico “Mecanismo de captura-reprodução” desse grupo, demostrando que mesmo ela não sendo a ferramenta que mais satisfaz as necessidades do cliente, ela possui particularidades que atendem melhor a determinado aspecto técnico em relação à ferramenta que mais satisfaz às necessidades do cliente, no nosso caso, a ferramenta *Maveryx*.

Percebemos então que algumas necessidades do cliente são atendidas por determinada ferramenta e outras necessidades, por outra ferramenta, e que dificilmente uma ferramenta atenderá a todos os aspectos técnicos requisitados.

Nota-se também que alguns grupos de aspectos técnicos de importância para o cliente não foram atendidos por nenhuma das ferramentas. Por exemplo, os grupos “Gerador de planos de teste”, “Gerador de dados”, “Avaliador de testes”, “Gerador de relatórios” e “Suporte da ferramenta”. Demonstrando que as ferramentas avaliadas precisam melhorar suas funcionalidades para atenderem necessidades consideradas importantes para a área de teste.

4.2 Discussão sobre a usabilidade das ferramentas

Nesta seção realizamos uma discussão sobre a usabilidade das três ferramentas selecionadas. Para isso, realizamos uma avaliação heurística visando identificar problemas relativos à usabilidade das ferramentas analisadas em questão, tornando por base as dez heurísticas de usabilidade definidas por Nielsen (1993). Com base em cada heurística, um ou mais problemas de usabilidade foram identificados como Barreira, Obstáculo e Ruído de acordo com a Seção 2.4.4. A Tabela 10 apresenta os problemas de usabilidade identificados para cada ferramenta em relação às heurísticas. Espaços em branco nessa tabela significa que não foi encontrado um problema de usabilidade.

Tabela 10 - Tabela completa da discussão sobre usabilidade

Heurísticas	*Ferramentas*	Problemas encontrados
1. Visibilidade do estado do sistema	***Robotium***	**Violação 1.1 (Barreira) –** Quando um usuário aborta um teste não fica claro o que está acontecendo com o sistema, essa função não funciona e os testes continuam.
	Maveryx	**Violação 1.2 (Ruído) –** Nenhum feedback de quanto tempo demorará um teste é apresentado, os testes simplesmente rodam.
	Calabash	**Violação 1.3 (Obstáculo) –** Não se sabe o que está acontecendo quando se executa um teste, é mostrado uma velocidade em KB/s não ficando claro o que significa, depois os casos de teste e o feedback só acontece realmente quando o teste acaba e é de difícil entendimento.
2. Mapeamento entre o sistema e o mundo real	***Robotium***	**Violação 2.1 (Barreira) –** A ferramenta é de difícil configuração e os testes são construídos em linguagem Java, restringindo a ferramenta a um público específico.
		Violação 2.2 (Obstáculo) – Os testes são executados com a JUnit, portanto deve-se ter uma familiarização com o framework ou necessitar treinamento.
	Maveryx	**Violação 2.1 (Obstáculo) –** O Console apresenta mensagens em linguagem de maquina ficando difícil o entendimento.
		Violação 2.2 (Obstáculo) – Os testes são executados com a JUnit, portanto deve-se ter uma familiarização com o framework ou necessitar treinamento.
	Calabash	**Violação 2.1 (Obstáculo) –** A instalação da ferramenta e sua configuração é toda feita na linha de comando do Prompt, o que dificulta muito.

		Violação 2.2 (Obstáculo) – A terminologia e os elementos da interface são todos exibidos em linha de comando exigindo uma familiarização com os mesmos ou necessitando de treinamento.
3. Liberdade e controle ao usuário	***Robotium***	**Violação 3.1 (Barreira) –** Depois que um teste começa, o usuário perde o controle da ferramenta, podendo somente cancelá-lo, mas está função não funciona. Ele também não consegue pausar um teste, ou executar mais de um simultâneo.
	Maveryx	**Violação 3.1 (Barreira) –** Os resultados dos testes são exibidos obrigatoriamente na tela.
	Calabash	**Violação 3.1 (Barreira) –** Após a inicialização de um teste, o usuário perde o controle da ferramenta até o teste acabar, podendo somente esperar ou fechar o Prompt de Comando.
		Violação 3.2 (Barreira) – Não se pode executar testes de aplicações diferentes simultaneamente na mesma janela de Prompt de Comando.
4. Consistência e padrões	***Robotium***	
	Maveryx	
	Calabash	**Violação 4.1 (Obstáculo) –** Os comandos da ferramenta são de difícil entendimento e a forma como é apresentado os resultados também, deixando inconsistente a ferramenta e necessitando de muita pesquisa ou treinamento para entender o funcionamento da mesma.
5. Prevenção de erros	***Robotium***	
	Maveryx	
	Calabash	
6. Reconhecer em vez de relembrar	***Robotium***	**Violação 6.1 (Obstáculo) –** A maioria das ações sobre a ferramenta o usuário terá que relembrar de um comando específico. Os botões são de difícil entendimento.
	Maveryx	**Violação 6.1 (Obstáculo) –** A maioria das ações sobre a ferramenta o usuário terá que relembrar de um comando específico. Os botões são de difícil entendimento.
	Calabash	**Violação 6.1 (Obstáculo) –** A maioria das ações sobre a ferramenta o usuário terá que relembrar de um comando específico, pois todas as ações estão em linha de comando.
7. Flexibilidade e eficiência de uso	***Robotium***	**Violação 7.1 (Barreira) –** O usuário não consegue personalizar a ferramenta para seu uso. A ferramenta foi feita para usuários experientes, todos os botões e mensagens somente um usuário "habituado" com a interface irá entender.
	Maveryx	**Violação 7.1 (Barreira) –** O usuário não consegue personalizar a ferramenta para seu uso. A ferramenta foi feita para usuários experientes, todos os botões e mensagens somente um usuário "habituado" com a interface irá entender.
	Calabash	**Violação 7.1 (Barreira) –** Como a ferramenta é executada em linha de comando o usuário perde muito tempo na execução dos comandos, e não consegue personalizar nada na ferramenta.
8. Design estético e minimalista	***Robotium***	
	Maveryx	
	Calabash	**Violação 8.1 (Obstáculo) –** São apresentadas várias informações durante a execução dos testes, misturando as mensagens dos casos de testes com mensagens em linguagem de maquina, deixando confuso o usuário.
9. Suporte para o usuário reconhecer, diagnosticar e	***Robotium***	**Violação 9.1 (Obstáculo) –** Todos os erros são apresentados em forma de código, até porque os testes são feitos em linguagem Java, porém não se propõe nenhuma solução para os erros.

recuperar erros	***Maveryx***	**Violação 9.1 (Obstáculo) –** Todos os erros são apresentados em forma de código, até porque os testes são feitos em linguagem Java, porém não se propõe nenhuma solução para os erros.
	Calabash	**Violação 9.1 (Obstáculo) –** Após os testes as mensagens de erros da ferramenta somente apontam para os erros, não informando uma possível solução.
10. Ajuda e documentação	***Robotium***	**Violação 10.1 (Ruído) –** A ajuda online da ferramenta é encontrada em seu site através de um e-mail do próprio fabricante. No GitHub, Google Groups e no site Stack Overflow é possível encontrar tópicos relacionados à ferramenta. Isso causa perca de tempo na ajuda que deveria ser em tempo real. A documentação da ferramenta não é disponibilizada.
	Maveryx	
	Calabash	**Violação 10.1 (Ruído) –** A ferramenta não possui ajuda online nela mesma, somente em blogs e no Google Groups. A documentação encontra-se no site do GitHub.

Com exceção da heurística “Prevenção de erros”, em que nenhuma ferramenta apresentou problemas de usabilidade, todas as outras heurísticas foram “violadas” por uma das ferramentas. Grande parte dos problemas encontrados foram do tipo “Barreira” (quando o usuário não consegue solucionar o problema) e “Obstáculo” (quando o usuário pode aprender a solucionar o problema, mas enquanto isso seu desempenho fica comprometido em relação a interface). Isso pode diminuir o desempenho e aceitação dos usuários em relação à ferramenta.

5 Considerações Finais

Neste trabalho, realizamos uma avaliação de ferramentas para o teste funcional de Aplicações Android e uma discussão sobre a usabilidade das ferramentas escolhidas, levando em consideração o contexto do Laboratório de Engenharia e Desenvolvimento de Sistemas (LEDS), da Universidade Federal de Ouro Preto.

O estudo de caso envolveu a avaliação de ferramentas para o teste funcional de Aplicações Android, por meio da aplicação do método SQFD e o método de avaliação proposto por Veloso et al. (2010). As ferramentas escolhidas para avaliação foram: *Robotium, Maveryx* e *Calabash*, ferramentas gratuitas, de acordo com a restrição do cliente. Como resultado dessa avaliação constatamos que a ferramenta *Maveryx* foi a mais adequada diante das necessidades apresentadas pelo nosso cliente. Em seguida, fizemos uma discussão sobre usabilidade das ferramentas, identificando problemas relativos a usabilidade da interface das

mesmas, problemas esses que podem mitigar o nível de satisfação do usuário em relação às ferramentas ou até mesmo causar sua rejeição.

Diante do exposto, cabe ao cliente a decisão final na escolha da ferramenta mais apropriada para o apoio às atividades de teste. Essa decisão pode ser influenciada pelo contexto do projeto, características da aplicação sob teste e diversos outros fatores.

Devido às restrições do LEDS e o prazo para conclusão deste trabalho, não foi possível realizar uma avaliação de usabilidade mais usual. Como trabalho futuro, pode ser investigado mais profundamente a usabilidade das ferramentas selecionadas, por exemplo, através de testes com os usuários ou através de uma avaliação heurística envolvendo mais especialistas.

6 Referências

AKAO, Y. **Introdução ao Desdobramento da Função da Qualidade**. Tradução de Zelinda Tomie Fujikawa e Seiichiro Takahashi. Belo Horizonte - MG: Fundação Christiano Ottoni, Escola de Engenharia da UFMG, 1996. 187 p.

ALVES, N. R. **QFD - Desdobramento da Função Qualidade Aplicado ao Desenvolvimento de Software**. Universidade Federal de Minas Gerais. Belo Horizonte, p. 10. 2000.

BALLARD, B. **Designing the Mobile User Experience.** USA: Wiley, 2007. 260 p.

BARBOSA, S. D. J.; SILVA, B. S. **Interação Humano-Computador**. São Paulo, SP: Elsevier Editora Ltda, 2010. ISBN 978-85-352-3418-3.

CALDEIRA, L. S. **Avaliação de Ferramentas para o Teste Funcional**. Universidade Federal de Ouro Preto. João Monlevade - MG. 2014.

CYBIS, W.; BETIOL, A. H.; FAUST, R. **Ergonomia e Usabilidade:** Conhecimentos, Métodos e Aplicações. 2ª. ed. São Paulo: Novatec, 2010. 352 p.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. **Introdução ao teste de software**. 1ª. ed. Rio de Janeiro - RJ: Editora Campus, v. 1, 2007.

DEMILLO, R. A. et al. **Software Testing and Evaluation**. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc. 1987.

DIAS, C. **Usabilidade na web:** criando portais mais acessíveis. Rio de Janeiro: Alta Books, 2007.

FABBRI, S. C. P. F.; VINCENZI, A. M. R.; MALDONADO, J. C. Teste Funcional. In: DELAMARO, M. E.; JINO, M. **Introdução ao Teste de Software**. 1ª. ed. Rio de Janeiro: Elsevier Editora Ltda, v. 1, 2007. p. 9-24.

FIGUEIREDO, C. M. S.; NAKAMURA, E. **Computação Móvel: Novas Oportunidades e Novos Desafios**. UFMG. Belo Horizonte, p. 28. 2003.

GITHUB. **Welcome to Calabash for Android**, 2015. Disponivel em: <https://github.com/calabash/calabash-android>. Acesso em: 20 Outubro 2015.

GITHUB. **Robotium:** User scenario testing for Android, 2015. Disponivel em: <https://github.com/robotiumtech/robotium>. Acesso em: 17 Setembro 2015.

GOOGLE. **Android Testing Tools**, 2015. Disponivel em: <http://developer.android.com/tools/testing/testing-tools.html>. Acesso em: 26 nov. 2015.

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.

HEWETT, T. T. et al. **ACM SIGCHI Curricula for Human-Computer Interaction**. NY: ACM, 1992. ISBN 0-89791-474-0.

ISO, IEC. **IEEE, Systems and Software Engineering -- Vocabulary**. ISO/IEC/IEEE 24765: 2010 (E)). Piscataway, NJ: IEEE computer society. 2010.

ISO, ISO. IEC25010: 2010 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. **International Organization for Standardization**, 2010.

KROGSTIE, J. **Using Quality Function Deployment in Software Requirements**. Anais do REFSQ'99. Heidelberg, Alemanha. 1999.

LECHETA, R. R. **Google Android:** Aprenda a Criar Aplicações para Dispositivos Móveis com Android SDK. 1ª. ed. São Paulo, SP: Novatec, 2009. ISBN 85-7522-186-0.

LINDGAARD, G. **Usability Testing and System Evaluation:** A Guide for Designing Useful Computer Systems. New York: Champman & Hall, 1994. 1-5 p. ISBN 978-0-412-46100-2.

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

MAZLAN, M. A. Stress test on J2ME compatible mobile device. **Innovations in Information Technology**, IEEE, p. 1-5, 2006.

MORAES, A. D.; ROSA, J. G. S. **Avaliação e Projeto no Design de Interfaces**. Rio de Janeiro, RJ: Editora 2AB, 2012.

MYERS, G. J.; SANDLER, C.; BADGETT, T. **The Art of Software Testing**. 3ª. ed. Hoboken, New Jersey: John Wiley & Sons, Inc., 2011. ISBN 978-1-118-03196-4.

NIELSEN, J. **Usability Engineering**. Boston - USA: Academic Press, 1993.

PEREIRA, L. C. O.; SILVA, M. L. **Android para Desenvolvedores**. Rio de Janeiro, RJ: Brasport, 2009.

PREECE, J.; ROGERS, Y.; SHARP, H. **Design de Interação:** Além da Interação Homem-Computador. 3ª. ed. Porto Alegre, RS: Bookman, 2013. 600 p.

PRESSMAN, R. S. **Software Engineering:** A Practitioner's Approach. 8ª. ed. New York, NY: McGraw-Hill, 2014.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. **Qualidade de Software:** Teoria e Prática. 1ª. ed. São Paulo: Prentice Hall, v. 1, 2001.

ROSA, J. M.; VERAS, M. **Avaliacao heurística de usabilidade em jornais online: estudo de caso em dois sites**. Perspectivas em Ciencia da Informacao. Belo Horizonte, p. 138-157. 2013.

SANTOS, I. S. et al. Requisitos e Aspectos Técnicos desejados em ferramentas de testes de software: um estudo a partir do uso do SQFD. **Aceito para publicação na revista RESI**, Novembro 2010.

SANTOS, R. L. G.; MORAES, A. **Ergonomização da interação homem-computador: abordagem heurística para avaliação da usabilidade de interfaces**. Dissertação de Mestrado - Departamento de Artes & Design. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, p. 184. 2000.

SHAMSODDIN-MOTLAGH, E. A Review of Automatic Test Cases Generation. **International Journal of Computer Applications**, v. 57, n. 13, p. 25-29, 2012.

SOUZA, A. C. **Proposta de um processo de avaliação da usabilidade de interfaces gráficas de sistemas interativos computacionais, através da integração das técnicas prospectiva, analítica e empírica**. Tese (Doutorado em Engenharia de Produção) - Universidade Federal de Santa Catarina. Florianópolis, p. 263. 2004.

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.

ZHANG, D.; ADIPAT, B. Challenges, Methodologies, and Issues in the Usability Testing of Mobile Applications. **In: Proceedings of the International Journal of Human Computer Interaction (IJHCI)**, v. 18, n. 3, p. 293-308, 2005.