FACULDADE DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO

MILENA CEZAR DA SILVA

MAIARA CEZAR DA SILVA

ESTUDO PARA ANALISAR E COMPARAR FERRAMENTAS DE TESTE DE SOFTWARES

RIO DE JANEIRO

2024

MILENA CEZAR DA SILVA

MAIARA CEZAR DA SILVA

ESTUDO PARA ANALISAR E COMPARAR FERRAMENTAS DE TESTE DE SOFTWARES

Trabalho apresentado à FAETERJ-Faculdade de Educação Tecnológica do Estado do Rio de Janeiro ao curso de Tecnologia em Sistemas de Informação, como requisito parcial para obtenção de nota final.
Orientadora: Prof. Tulio Queto

BANCA EXAMINADORA

Túlio Queto de Souza Pinto
Título Nome

Adilson Ricardo da Silva
Título Nome

Rubens Saviano
Título Nome

RIO DE JANEIRO

2024

AGRADECIMENTOS

Primeiramente, gostaria de expressar minha imensa gratidão a Deus, cuja presença inabalável foi a luz que guiou cada passo desta jornada acadêmica. Sua orientação divina, força inspiradora e misericórdia infinita foram o alicerce sobre o qual este trabalho foi construído. Sem Sua graça, nada seria possível.

Em seguida, dedico um profundo agradecimento aos meus queridos avós, cujo amor e cuidado foram verdadeiros pilares em minha vida. Que Deus os abençoe abundantemente por todo o carinho, dedicação e sacrifício que sempre ofereceram generosamente. Este trabalho é também uma homenagem ao amor incondicional e ao suporte que vocês compartilharam ao longo dos anos. Muito obrigado por tudo.

Por fim, deixo meu sincero agradecimento ao meu orientador, cuja orientação, paciência e conhecimentos foram fundamentais para a realização deste trabalho. Suas valiosas contribuições não apenas enriqueceram este projeto, mas também inspiraram meu crescimento pessoal e acadêmico. Muito obrigado por todo o suporte durante essa jornada.

RESUMO

Este artigo apresenta uma análise comparativa de ferramentas de teste de software, destacando seu papel essencial na garantia da qualidade e confiabilidade no desenvolvimento de sistemas. A análise baseia-se na revisão de três artigos selecionados por meio de um processo de revisão sistemática, utilizando critérios de inclusão como relevância ao tema e rigor metodológico. Os resultados ressaltam a importância da automação na otimização de processos de teste repetitivos e na garantia de resultados consistentes. Além disso, o estudo enfatiza a relevância dos testes de unidade e de integração, ao mesmo tempo que aborda desafios como a cobertura insuficiente de testes não funcionais, especialmente em ambientes móveis e distribuídos. A conclusão reforça que a escolha das ferramentas de teste deve priorizar eficiência, flexibilidade e adaptabilidade às necessidades específicas do projeto, com a automação desempenhando um papel fundamental na manutenção da qualidade e estabilidade dos sistemas.

Palavras-chave: Testes de software, automação, ferramentas de teste, qualidade de software.

ABSTRACT

This paper aims to conduct a comparative analysis of software testing tools, highlighting their importance in ensuring quality and reliability in system development. Three articles were reviewed, selected from a systematic review based on inclusion criteria such as relevance to the topic and methodological quality. The critical analysis points to automation as an essential approach to optimizing repetitive tests and ensuring consistent results. Furthermore, it highlights the importance of unit and integration testing while identifying challenges such as insufficient coverage of non-functional tests, especially in mobile and distributed environments. It is concluded that the choice of tools should be based on efficiency, flexibility, and adaptability to the specific project context, with automation being a key factor in maintaining the quality and stability of developed systems.

Keywords: Software testing, automation, testing tools, software quality.

SUMÁRIO

1 INTRODUÇÃO

Os testes desempenham um papel crucial no desenvolvimento de software, garantindo a qualidade e a confiabilidade dos produtos finais. Em um cenário de crescente competitividade e demanda por software de alta qualidade, os testes são essenciais para identificar e corrigir falhas, assegurando que o software atenda às expectativas dos usuários e às especificações do projeto. Eles permitem detectar problemas precocemente, evitando impactos negativos na experiência do usuário e prejuízos significativos. No desenvolvimento ágil, onde mudanças são frequentes e o tempo de entrega é reduzido, os testes garantem que as alterações não causem regressões ou quebras no sistema, mantendo a qualidade e a estabilidade do software.(Sommerville,2011)

De acordo com Ian Sommerville (2011, p. 63, Cap.8), "o objetivo dos testes de software é revelar a presença de defeitos. A citação é atribuída a Ian Sommerville em seu livro "Engenharia de Software". Nessa seção, Sommerville enfatiza que os testes são projetados para identificar a presença de defeitos, não para confirmar sua ausência. Não se destina a provar que não existem defeitos".Eles são responsáveis por identificar e corrigir falhas, garantindo que o software atenda às expectativas dos usuários e às especificações do projeto (DE SOUSA, 2019). Os testes são fundamentais para assegurar que o software funcione conforme o esperado e atenda aos requisitos de qualidade estabelecidos. Eles ajudam a identificar problemas precocemente, permitindo que sejam corrigidos antes que impactem negativamente a experiência do usuário ou causem prejuízos significativos.

Além disso, os testes proporcionam uma maior confiança no software, tanto para os desenvolvedores quanto para os usuários finais (BARNUM, 2020). Em um ambiente ágil, onde as mudanças são frequentes e o tempo de entrega é reduzido, os testes garantem que as alterações introduzidas não causem regressões ou quebras no sistema. Eles permitem uma interação rápida e contínua do software, mantendo a qualidade e a estabilidade ao longo do processo de desenvolvimento (DE SOUSA, 2019).

Adicionalmente, os testes são uma parte integrante das práticas de DevOps,

que é a combinação de *devlopment* (desenvolvimento) e *operations* (operações), visando a integração contínua, entrega contínua e operação contínua de software. Eles fazem parte de um ciclo de *feedback* contínuo, onde as alterações no código são testadas e validadas automaticamente, garantindo uma entrega rápida e confiável do software (SPIRLANDELI e ROLAND, 2019).

Apesar da existência de diversas ferramentas de teste, muitas exigências permanecem não atendidas. A falta de ferramentas apropriadas para administrar as atividades de teste é uma ocorrência frequente. Optar por ferramentas inapropriadas pode resultar em perda de tempo e afetar adversamente a eficácia do produto, principalmente se não puderem detectar os defeitos específicos (WEBER,2001).

Neste contexto, é essencial realizar uma análise comparativa de ferramentas de teste que possam efetivamente contribuir para o processo de desenvolvimento, proporcionando economia de tempo, melhoria de desempenho e detecção mais eficaz de erros.

Assim, este trabalho busca realizar uma revisão sistemática da literatura sobre o tema "comparação de ferramentas de apoio aos testes", com o objetivo de encontrar modelos de avaliação que possam auxiliar na escolha das ferramentas mais adequadas para uma organização. No entanto, diversas questões, como o tamanho da empresa, os modelos de processos de desenvolvimento e a cultura organizacional, não serão consideradas, pois o contexto em que a ferramenta é operacionalizada é complexo e difícil de ser analisado como um todo.

1.1 Objetivo

O objetivo deste trabalho é realizar uma revisão sistemática da literatura sobre o tema "comparação de ferramentas de apoio aos testes".

1.2 Motivação

A motivação para este objetivo é a necessidade de garantir a qualidade e eficiência dos sistemas de informação desenvolvidos.Contudo a escolha da

ferramenta adequada pode ser desafiadora, especialmente ao buscar soluções gratuitas que podem afetar a qualidade dos testes de software.

Estrutura do trabalho

Este trabalho, além da introdução já apresentada, está organizado nas seguintes seções: Capítulo 2, Conceitos Iniciais: fundamentação sobre teste de software. Capítulo 3, Trabalhos Relacionados: este capítulo apresenta uma pesquisa documental no contexto da comparação de ferramentas de teste de software, por meio de uma revisão da literatura acadêmica. Capítulo 4, Análise Crítica: nesta seção foram analisados os pontos fortes e fracos dos trabalhos apresentados no capítulo anterior. Capítulo 5, Conclusão: São expostos os aspectos considerados para a proposta de solução e os trabalhos em andamento.

1 CONCEITOS INICIAIS

No contexto da engenharia de software, o teste de software é um processo vital que visa garantir a qualidade e a confiabilidade dos produtos desenvolvidos. Segundo Roger Pressman (2005) o teste de software envolve a execução de um programa com a intenção específica de encontrar erros, verificar a conformidade com os requisitos e assegurar que o software atende às expectativas dos usuários.

Os principais objetivos dos testes de software são verificar se o software cumpre os requisitos especificados e identificar situações em que o comportamento do software é incorreto, indesejável ou não conforme as especificações. Para atingir esses objetivos, os testes são realizados em diferentes níveis, como testes de unidade, testes de integração, testes de sistema e testes de aceitação, como demonstrado na figura abaixo.

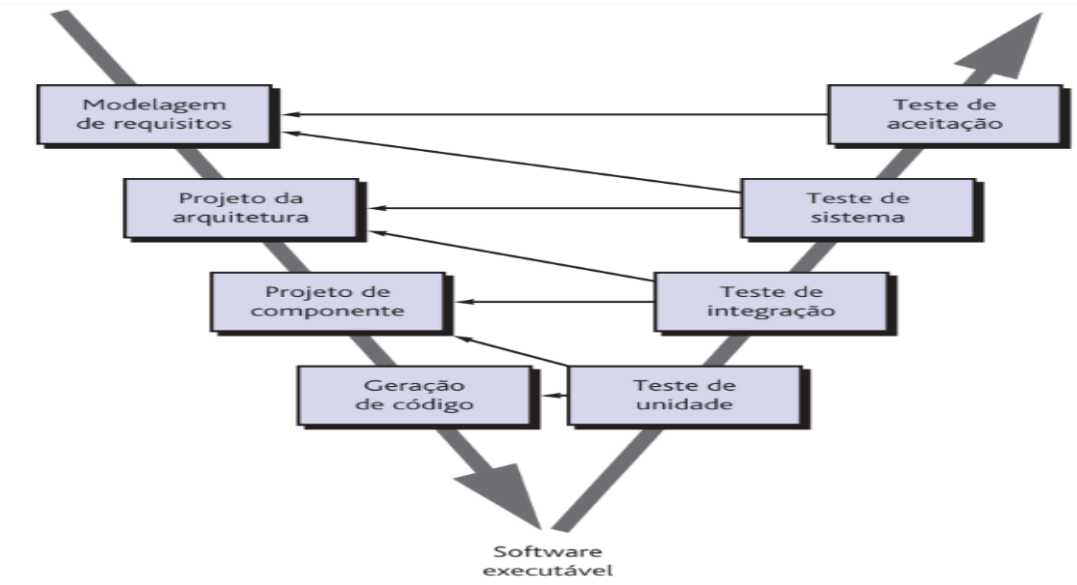

Figura 1 - Modelo V.

Fonte: Pressman, 2011.

Os testes de unidade focam na validação de componentes individuais do código, enquanto os testes de integração avaliam a interação entre diferentes módulos ou unidades. Os testes de sistema verificam o software como um todo em relação aos requisitos funcionais e não funcionais, e os testes de aceitação garantem que o sistema está pronto para ser usado pelos usuários finais.

Quadro 1 - Tipos de testes de software

Tipo de Teste	Tipo de Teste	Foco	Nível
Teste de Unidade	Validar componentes individuais do código	Cada pedaço isolado do código (funções, métodos)	Micro (Detalhado)
Teste de Integração	Avaliar a interação entre diferentes módulos ou unidades	Como os módulos se comunicam entre si	Médio

Teste de Sistema	Verificar o software como um todo	Todo o sistema, como um único bloco	Macro
Teste de Aceitação	Garantir que o sistema está pronto para o uso pelos usuários finais	Requisitos funcionais e não funcionais	Final (Pronto para uso)

Fonte: Elaboração própria.

Existem diferentes tipos de testes que podem ser realizados, incluindo testes funcionais e não-funcionais. Os testes funcionais avaliam se o software realiza as funções esperadas conforme as especificações, enquanto os testes não-funcionais analisam aspectos como desempenho, usabilidade, confiabilidade e segurança.

Roger Pressman (2005), também destaca a importância das estratégias de testes, como os testes de caixa-branca e de caixa-preta. Os testes de caixa-branca focam na lógica interna do código, testando todos os caminhos possíveis através do código. Já os testes de caixa-preta são baseados nas especificações do software, sem considerar a estrutura interna do código.

O processo de teste envolve várias etapas, incluindo o planejamento, o projeto, a execução, a análise de resultados e a documentação dos testes. O planejamento de teste define o escopo, a abordagem, os recursos e o cronograma das atividades de teste. O projeto de teste cria casos de teste detalhados e especificações de teste. A execução de teste realiza os testes e registra os resultados. A análise de resultados compara os resultados esperados com os reais e identifica defeitos. Por fim, os relatórios de teste documentam os resultados e as métricas de qualidade. A automação de testes é outra área importante abordada por Pressman (2005).

A automação de testes utiliza ferramentas de software para executar testes automaticamente, melhorando a eficiência e reduzindo o esforço manual. Entre as vantagens da automação de testes estão a maior cobertura

de testes, a reprodutibilidade e a possibilidade de realizar testes frequentes e consistentes.

Pressman (2005) também apresenta várias técnicas de teste, como o particionamento de equivalência, a análise de valor limite, o teste de caminho básico e o teste de caso de uso. O particionamento de equivalência divide os dados de entrada em classes de equivalência para reduzir o número de casos de teste necessários. A análise de valor limite foca em valores nos limites das classes de equivalência para identificar defeitos. O teste de caminho básico testa todos os caminhos de execução possíveis através do código, enquanto o teste de caso de uso é baseado em cenários que representam interações reais dos usuários com o sistema.

Os testes de software são fundamentais para garantir a qualidade, a confiabilidade e a usabilidade do software, atendendo às expectativas dos usuários e minimizando riscos de falhas na produção. As abordagens e conceitos de Roger Pressman (2005) oferecem uma base sólida para a implementação de práticas de teste de software eficazes e eficientes, contribuindo para o sucesso dos projetos de desenvolvimento de software.

A citação "os testes ajudam a construir confiança no software tanto para os desenvolvedores quanto para os usuários finais" é atribuída a Ian Sommerville em seu livro "Engenharia de Software". Embora a página exata possa variar entre edições e traduções, essa ideia é discutida no Capítulo 8, que trata de Testes de Software. Nessa seção, Sommerville enfatiza que os testes são projetados para identificar a presença de defeitos, não para confirmar sua ausência.

METODOLOGIA DE SELEÇÃO DE ARTIGOS

Durante a pesquisa documental realizada no "Google Acadêmico", foram identificados 359 artigos utilizando as palavras-chave "avaliação de ferramentas", "comparação de ferramentas de teste", "ferramentas de apoio aos testes" e "ferramentas de testes". Onde 213 tratavam de ferramentas ou contextos que não estavam diretamente relacionados ao foco do estudo, 90

estavam defasados, para garantir que as conclusões fossem relevantes e aplicáveis ao cenário atual, foram selecionados apenas artigos publicados nos últimos 5 anos ou que discutem ferramentas amplamente utilizadas e modernas, restaram 53 onde alguns artigos analisavam ferramentas de forma muito geral, sem fornecer insights significativos sobre o desempenho ou aplicabilidade em contextos reais. Outros eram extremamente específicos, restringindo-se a casos de uso muito particulares, não atendendo à intenção de uma análise comparativa abrangente, desses foram considerados apenas 3. Considerando as limitações de tempo e os recursos disponíveis para análise, optou-se por uma amostragem representativa, mas reduzida, que fosse suficiente para atender aos objetivos da pesquisa.

Esses artigos representaram a base inicial para a seleção de materiais relevantes ao tema abordado neste trabalho, que visa comparar e avaliar ferramentas de teste de software. O grande volume de resultados exigiu uma triagem criteriosa, seguindo parâmetros específicos, de modo a garantir que apenas os artigos mais relevantes fossem utilizados na análise.

Os artigos foram selecionados de acordo com critérios de inclusão claros e bem definidos. O primeiro critério foi a relevância ao tema central do estudo. Somente foram incluídos os artigos que tratavam diretamente da comparação de ferramentas de teste de software, com foco em automação de testes ou ferramentas de suporte à qualidade. O segundo critério foi a qualidade metodológica dos artigos. Foram priorizados aqueles que apresentavam uma metodologia rigorosa, com uma análise comparativa detalhada entre diferentes ferramentas. O terceiro critério de inclusão foi a atualidade. Para assegurar a relevância tecnológica das ferramentas analisadas, apenas artigos publicados nos últimos dez anos foram considerados. A aplicação desses critérios resultou na seleção de três artigos essenciais para a discussão e conclusão deste trabalho.

Por outro lado, também foram aplicados critérios de exclusão para garantir a consistência do estudo. Artigos cujo foco se desviava significativamente do tema central foram eliminados. Por exemplo, muitos artigos abordavam ferramentas específicas de testes em hardware ou outras

áreas fora do desenvolvimento de software, o que não se encaixava nos objetivos deste trabalho. Além disso, artigos que não ofereciam uma avaliação detalhada ou comparações práticas das ferramentas de teste também foram excluídos. Dessa forma, garantimos que o conteúdo selecionado fosse diretamente aplicável à análise comparativa proposta. Apesar da utilização de apenas três artigos, o processo de seleção foi suficientemente rigoroso para garantir que essas fontes fossem as mais adequadas para embasar a pesquisa e fornecer as informações necessárias para a análise crítica.

2 TRABALHOS RELACIONADOS

Esta seção apresenta a pesquisa documental que sustentou este trabalho. Utilizamos o buscador “Google Acadêmico” para buscar materiais com as palavras-chave: “avaliação de ferramentas”, "comparação de ferramentas de teste”, “comparação de ferramentas de apoio aos testes" e “ferramentas de testes”.

Após analisar os resultados, descartamos os artigos que não estavam alinhados com os objetivos da pesquisa. Dessa análise, selecionamos três trabalhos relevantes, que serão apresentados a seguir.

2.1 Avaliação de ferramentas de automação para engenheiros de testes

Segundo os autores deste estudo, escolher a ferramenta adequada para automação de processos de testes de software é uma questão complexa e custosa, exigindo tempo e elevado conhecimento dos profissionais envolvidos. Todo esse grau de dificuldade é agravado pela insuficiente descrição dos aspectos das funcionalidades das ferramentas. O estudo de DÓREA, A. D. de O.; CARVALHO, F. de S.; SANTOS, M. T.; NETO, M. C. M.; MOISES, D., 2008 teve o objetivo de avaliar, de forma descritiva e comparativa, algumas ferramentas para automação de testes.

Este projeto contou com 8 etapas: definição dos critérios, elaboração, aplicação, avaliação das respostas do questionário, validação dos critérios,

seleção de ferramentas, testes e validação das ferramentas e conclusão.

4.1.1 Definição dos critérios

Inicialmente os autores deste estudo criaram um questionário a partir de critérios específicos: portabilidade, eficiência, *performance*, flexibilidade, legibilidade e interoperabilidade.

A portabilidade refere-se à capacidade de um sistema ser transferido e executado em diferentes ambientes de hardware ou software com pouca ou nenhuma modificação. Eficiência é a habilidade do sistema de resolver tarefas ou atingir objetivos específicos de maneira eficaz. A *performance* está relacionada à capacidade do sistema de realizar essas tarefas ou atingir os objetivos com qualidade. Flexibilidade é a exigência do sistema para se adaptar a diferentes ambientes e condições. Legibilidade se refere à clareza e facilidade de compreensão das tarefas ou do sistema. Interoperabilidade, por sua vez, é a capacidade de um sistema, processo ou atividade ser interrompido e retomado sem perda significativa de sua eficiência.

4.1.2 Elaboração, aplicação e avaliação das respostas do questionário

Com base nos primeiros critérios definidos foi criado um questionário com a intenção de validar os critérios pré-definidos. Com o objetivo de coletar e validar quais os critérios utilizados para a escolha de ferramentas, outro ponto importante foi a intenção de saber o grau de conhecimentos que as empresas possuem quando se trata de ferramentas automatizadas para testes de software.

Sua aplicação contou com a participação de 13 empresas desenvolvedoras de softwares de Salvador, sendo que somente 7 dessas utilizam softwares para automatização de testes. A pesquisa foi dividida em 8 etapas: ferramenta adotada, técnicas de teste, estratégias de teste, métricas, escopo do sistema, tecnologia, preço da ferramenta e critérios para escolha da

ferramenta.

Tentamos entrar em contato com os autores por meio de e-mail, mas infelizmente não obtivemos resposta. Realizamos diversas tentativas de comunicação em diferentes horários e dias, mas não tivemos sucesso em estabelecer um diálogo. Reconhecemos a importância de obter o questionário e as contribuições dos autores, e continuaremos buscando formas alternativas de comunicação para garantir que todas as partes envolvidas sejam devidamente ouvidas.

Os resultados obtidos revelam como as empresas se comportam no processo de desenvolvimento de testes. Observou-se que o JUnit é a ferramenta de automação de testes mais amplamente adotada pelas organizações avaliadas. Em relação às técnicas de teste, verificou-se que as empresas geralmente utilizam o teste de caixa preta ou uma combinação de testes de caixa preta e caixa branca. Os testes de unidade e de sistema foram as duas estratégias mais frequentemente empregadas na hora de testar e validar os sistemas, corroborando a popularidade do JUnit, que é uma ferramenta voltada para testes de unidade.

Além das estratégias, técnicas e tecnologias utilizadas na realização dos testes, outros critérios como o tamanho do sistema, o custo, a *performance* e a eficiência também são fatores importantes que influenciam a escolha de uma ferramenta de teste pelas empresas.

4.1.3 Validação dos critérios

A fase de validação começou com a análise das respostas do questionário, que visava identificar a importância dos critérios na escolha de ferramentas automáticas de teste. Devido à falta de um padrão para esses critérios, essa etapa foi essencial para o projeto. Constatou-se a necessidade de avaliar as ferramentas com base em critérios técnicos como performance, legibilidade e eficiência, além de permitir que as empresas adicionassem outros

critérios. A análise revelou que as organizações priorizam ferramentas com alta performance (31%) e eficiência (26%), enquanto 71% consideram o preço e 57% o tamanho do sistema como fatores decisivos.

4.1.4 Seleção das ferramentas

Essa etapa foi responsável por selecionar ferramentas automatizadas com base nos critérios gerais, esse projeto limitou o universo do processo de seleção, forçando apenas em ferramentas automáticas gratuitas que utilizam a tecnologia Java para aplicações desktop.

Quadro 2 - Descrição das Ferramentas (Avaliação de Ferramentas de automação para engenheiros de Testes)

Nome	Versão	Tipo de Teste
FINDBugs	1.2	Cobertura de código
Emma	2.1	Cobertura de còdigo para aplicações e para teste de Unidade
TestNG	5.5	Unidade
AgitarOne	4.0	Unidade
JUnit	4.3.1	Unidade
TeamCity	2.0	Integração
EasyAccept	1.0.2	Validação

Fonte: Elaboração própria.

4.1.5 Escala de avaliação

A escala de avaliação detalhou as etapas adotadas para tornar a avaliação das ferramentas mais consistente, fundamentando-se no referencial teórico e na análise dos questionários aplicados.

Tabela 3 - Escala de avaliação (Avaliação de Ferramentas de automação para engenheiros de Testes)

Critérios	**Escala**			
	Características	Ótima	Boa	regular
Performance	● Completude com sucesso dos scripts de Teste sem falhas e dentro do tempo esperado ● Requerido para múltiplos usuários ● Requerido(por transação) para um usuário único	Atende as Três características	Atende pelo menos duas características	Apenas uma
Legibilidade	● Estruturas que identifiquem o início e o final do código ● Palavras reservadas ● Facilidade na leitura	Atende as Três características	Atende pelo menos duas características	Apenas uma
Flexibilidade	● Código aberto ● Permite	Atende as duas caracter´sticas	Atende pelo menos uma característica	Não Atende

	extensões			
Portabilidade	• Plataformas diferentes	Três ou mais	Pelo menos duas	Apenas Uma
Interoperabilidade	• Padrão aberto a fim de oferecer uma comunicação transparente entre os sistemas.	Atende	–	Não atende
Eficiência	• Atende os critérios de flexibilidade, performance, legibilidade e interoperabilidade	Atende aos critérios descritos nas características	Atende á pelo menos dois critérios descritos nas características	Atende á apenas um critérios descritos nas características

Fontes: (Alexsandro D. de Oliveira Dórea,Fernanda de S.Carvalho,Monique T. Santos,Manoel Carvalho Marques Neto,David Moises).

4.1.6 Avaliação de ferramentas

Nesta seção, os autores analisaram diversas ferramentas para realizar testes de unidade, integração, validação e sistema, aplicando casos específicos para cada tipo de teste. Esses casos ilustram tanto testes bem-sucedidos quanto aqueles que falharam, além dos resultados esperados para cada um.

Nos testes de unidade, foram utilizados dois casos distintos. O primeiro, denominado "Pessoa Física", testou métodos como inserção, atualização, exclusão, recuperação de registros e obtenção da chave primária. Esperava-se o retorno do número inteiro 1 para testes bem-sucedidos e 0 para falhas. No segundo caso, chamado "Operações", foram testados métodos de soma, subtração, divisão, multiplicação e cálculo fatorial, com resultados corretos esperados para sucessos e incorretos para falhas.

Nos testes de integração, não foi necessário utilizar casos específicos, pois a ferramenta analisada utiliza frameworks como JUnit, NUnit e TestNG para realizar testes de unidade de maneira integrada.

No teste de validação, foi usado o caso "MonopolyGame". Para executar esses testes, foi criada a classe "MonopolyFacade.java" e um arquivo com scripts diversos, cobrindo situações de sucesso e falha, para testar as regras de negócio do jogo.

4.1.7 Análise dos resultados

Nesta seção, são apresentados os resultados da análise das ferramentas para testes de unidade, integração, validação e sistema, utilizando os critérios previamente validados e uma escala de avaliação estabelecida.

Quadro 4. Avaliação das ferramentas segundo os autores (Avaliação de Ferramentas de automação para engenheiros de Testes).

Ferramenta de teste	Tipo de Teste	Observação
JUnit	Teste de Unidade	Se destacou entre as ferramentas avaliadas para testes de unidade. É um framework que facilita a criação automática de testes de unidade com apresentação dos resultados, exibindo uma possível falha. Utilizado tanto para a execução de baterias de testes como para extensão. Possui ótima performance e ótima legibilidade, tem ótima flexibilidade, portabilidade e interoperatividade, sendo bastante eficiente para seu domínio. Permite acoplamento a ferramentas de desenvolvimento como Eclipse.
AgitarOne	Teste de Unidade	Apresenta todas as características do JUnit com a nova funcionalidade de geração automatizada das classes de teste. Utilizando o Caso de Teste Operações, mostrou-se muito eficiente. Apresentou ótima performance, portabilidade e interoperabilidade , boa eficiência, legibilidade e flexibilidade.
TestNG	Teste de Unidade	Possui características semelhantes ao JUnit, mas com

		funcionalidades adicionais que a tornam mais fácil de usar. Apresentou ótima performance, portabilidade e interoperabilidade, boa eficiência e ótima flexibilidade, permitindo criação de dependências entre métodos de testes.
EMMA	Teste de Unidade	Ferramenta Java de código aberto que analisa a cobertura de código. Avaliada com EclEmma, apresentou ótima performance, boa legibilidade, ótima flexibilidade, portabilidade e interoperabilidade, e boa eficiência.
FINDBugs	Teste de Unidade	Ferramenta de código aberto para encontrar erros no código Java a partir da inspeção dos bytecodes. Apresentou boa performance, legibilidade,eficiência, ótima flexibilidade, portabilidade e interoperabilidade.
TeamCity	Teste de Integração	Funciona integrado com ferramentas de gerenciamento de configuração. Executa testes no momento da atualização no repositório, exibindo relatórios de erros . Apresentou boa performance e legibilidade, flexibilidade e portabilidade regulares, eficiência e interoperabilidade regulares.
Easy Accept	Teste de Validação	Auxilia na criação, execução e automatização de testes de validação. Utilizou o caso de teste MonopolyGame. Apresentou boa performance e legibilidade, ótima flexibilidade, portabilidade e interoperabilidade, e boa eficiência.

Fonte: Autoria própria.

Avaliar ferramentas de testes automatizados é uma tarefa essencial para equipes de desenvolvimento de software, e um documento de análise bem estruturado pode oferecer uma orientação valiosa. Um ponto positivo importante é a clareza e a estrutura do documento, segue uma metodologia bem definida, como o uso do questionário para validar critérios de avaliação, o que reforça a credibilidade dos resultados. A seleção de critérios relevantes, como performance, legibilidade, flexibilidade, portabilidade e eficiência, é outro aspecto positivo, pois são fatores cruciais que impactam diretamente a eficácia e a adequação das ferramentas para diferentes cenários de teste. Além disso, a análise de uma variedade de ferramentas, como JUnit, AgitarOne, TestNG, Emma e FINDBugs, permite uma comparação rica e abrangente,

proporcionando ao leitor uma visão mais completa.

A validação prática das ferramentas com casos de teste específicos é especialmente valiosa, oferecendo uma base empírica que complementa a análise teórica. No entanto, alguns pontos negativos podem ser destacados. O artigo avaliou um número limitado de empresas e equipes de desenvolvimento, isso pode restringir a generalização dos resultados, o que é uma limitação significativa. A ênfase em ferramentas específicas, especialmente se focada apenas em ferramentas Java para desktop, pode deixar de lado outras tecnologias e linguagens relevantes, limitando a aplicabilidade das conclusões. Além disso, a dependência do questionário pode introduzir fatores subjetivos, e a análise pode ser superficial se não explorar detalhes técnicos ou casos de uso específicos de maneira aprofundada,a não disponibilidade deste questionário também é um ponto negativo ao artigo, colocando em dúvida sua veracidade.

Um documento de análise de ferramentas de testes automatizados pode ser extremamente útil, mas sua eficácia depende da abrangência e da profundidade da análise, bem como da representatividade dos dados coletados. Uma abordagem mais diversificada e detalhada poderia fortalecer ainda mais suas conclusões e fornecer uma base sólida para a tomada de decisões informadas.

4.2 Trabalho sobre “um estudo sobre ferramentas para o teste de aplicações android no contexto do laboratório leds.” Martins ferreira, (2016)

O principal objetivo deste trabalho foi avaliar as ferramentas disponíveis para teste de aplicativos móveis, usando o método Software Quality Function Deployment (SQFD) para identificar e priorizar as necessidades do usuário e o método de avaliação de ferramentas proposto por Veloso et al. (2010).

O trabalho teve foco na 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. 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.

A pesquisa inicial sobre ferramentas de suporte a testes foi conduzida online e na literatura, identificando 56 ferramentas com características específicas, como ambiente de execução, idioma, tipo de licença, versão, link e outras observações relevantes no Quadro 1.

Quadro 5 – Trecho extraído da tabela completa de seleção de ferramentas

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	-	01/01/2024	https://code.google.com/p/robotium/wiki/Downloas?tm=2	Testa apenas Aplicativos Android
MonkeyRunner	Android Studio/Eclipse	Java	Open-Source		-	https://github.com/lvc/abicompliance-checker	Testa apenas Aplicativos Android
Ranorex	IOS,Android e Windows 8 e web	Java	Open-Source, Free Trial/ Pago		5.03 24/03/20215	http://www.ranorex.com/	Testa Android, IOS, Web
Appium	Windows, Linux, IOS, Android	Java,Python, ruby…	Open-Source	0.18x	1.3.7 beta	http://appium.io	Testa Android e IOS - Testes OK
UI Automator	Android	Java Library	-			http://developer.android.com/tools/testing-support-library/index.htm	
MonkeyTalk	IOS and Android	Java	Free		22/12/14 MonkeyTalk Profissional	http://www.cloudmonkeymobile.co	Testes OK

					Edition 2.0.10 beta	m./monkeytalk	
Testing With frank	IOS	Java	Open-Source			http://www.testingwithfrank.com/	

Fonte: Trecho extraído da tabela completa de seleção de ferramentas (Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS).

Após conduzir pesquisas e testes em ferramentas de teste identificadas, ferramentas de código aberto foram selecionadas com documentação abrangente para uso e instalação, usando sensores de posição e movimento GPS para aplicações LEDS, e estudos de caso detalhados e aplicação do método SQFD serão discutidos nas seções a seguir.

4.2.1 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 das capacidades das ferramentas para o tratamento de dados de sensores.

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 partir 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.

4.2.2 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.

Quadro 6 - 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
		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 reincidências 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 testes
		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 implantação 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 implementação	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.

Fonte: Tabela de desdobramento das Qualidade Exigidas (Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS).

4.2.3 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, refere-se a um trecho da tabela de desdobramento dos elementos da qualidade, apresentados em grupos e níveis.

Quadro 7 - Tabela de desdobramento dos elementos da qualidade

	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 sistemas
4	Gerador manual de testes	Mecanismo de captura-reprodução
		Uso de linguagens de alto nível
		Acesso as funções de 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
		Manuel 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

Fonte: Trecho extraído da Tabela completa de Desdobramento dos Elementos da Qualidade (Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS).

4.2.4 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:

Quadro 8 - Total da Matriz da Qualidade

		Funções da ferramenta										
			Nível 1	Gerador de plano de teste			Gerador de dados	Gerador de testes funcionais				
		Nível 2	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

Fonte: Trecho extraído da Matriz de Qualidade (Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS).

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.

4.2.5 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.

Tabela 9 - Grau de importância.

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 testes	1
3	Apoio para identificação da complexidade das partes do sistemas	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	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

Fonte: Tabela completa de requisitos priorizados (Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS).

4.2.6 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.

Quadro 10 - Priorização das especificações técnicas.

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 utilizado critérios	5.1%
6	Oráculo para gerar as saídas esperadas	3.5%
7	Extrator de dados de modelos descrevendo o sistemas	2.8%
8	Mecanismo de captura-reprodução	1.0%
9	Uso de linguagens de alto nível	1.2%
10	Acesso às 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%

Fonte: Trecho da Tabela completa de aspectos técnicos priorizados (Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS).

4.2.7 Avaliação das ferramentas

O estudo utilizou o Método de Avaliação de Veloso et al. para avaliar ferramentas de suporte para testes de software. O método envolveu um questionário com aspectos técnicos identificados para avaliar o nível de suporte técnico da ferramenta. Critérios foram atribuídos a cada ferramenta e o nível de suporte técnico da ferramenta. As três ferramentas foram testadas, instaladas e

configuradas, com foco nas necessidades do cliente. A ferramenta foi testada em aplicativos Android gratuitos usando sensores durante a interação do usuário. Uma mediana foi atribuída para cada pergunta, com uma resposta completa na Tabela 11.

Quadro 11 – Testes.

				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 linguagem 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 funções 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	SIM(1)		0		0		0
			PARCIALMENTE (0.5)						
			NÃO (0)	X		X		X	

		arquivos) ?							

Fonte: Trecho extraído do questionário completo sobre as ferramentas (Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS).

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, apresentado na coluna Atendido. Os resultados apresentam em porcentagem o quão aquele aspecto técnico atende às necessidades do cliente.

Quadro 12 – Aspecto técnico.

				Ferramentas					
				MonkeyTalk		**Sikuli**		**Selendroid**	
	Grupo	**Aspectos Técnicos**	**Pesos**	**Pontuação**	**Atendido**	**Pontuação**	**Atendido**	**Pontuação**	**Atendido**
1	**Gerador de plano de teste**	**Gestão de dados de 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	**2.8**	**0**		**0**		**0**	

		descrevendo o sistema							
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 funções do SO	2.1	1		0		0	
		Acesso ao mecanismo de persistência	2.1	0		0		0	

Fontes: Trecho extraído da tabela completa de comparação entre as ferramentas de testes para Aplicações Android: MonkeyTalk, Sikuli e Selendroid. (Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS).

Foi realizado um estudo para analisar o suporte fornecido por sensores em aplicativos, com foco nos sensores de movimento Acelerômetro, Giroscópio e sensor de posição GPS. Os resultados mostraram que o Sikuli, apesar de testado para aplicativos Android, não suporta acesso por sensor. O Selendroid e o MonkeyTalk oferecem métodos para controlar a orientação do dispositivo por meio de comandos, enquanto o MonkeyTalk permite aplicativos externos como o FakeGPS para interação GPS na direção. O nível de satisfação de cada ferramenta foi determinado usando o SQFD e o Método de Avaliação proposto por Veloso (2010).

Quadro 13 – Nível de satisfação de cada ferramenta.

				Ferramentas					
				MonkeyTalk		Sikuli		Selendroid	
	Grupo	Aspectos Técnicos	Pesos	Pontuação	Atendido	Pontuação	Atendido	Pontuação	Atendido
1	Gerador de plano de teste	Gestão de dados de plano de teste	3.3	0	0		0		0
		Registro de tarefas	1.6	0					
		Integração com ferramentas de gerenciamento de projetos	4.2	0					
2	Gerador de	Gerador de objetos	4.7	0	0		0		0

	dados								
3	Gerador de testes funcionais	Gerador de entradas utilizando critérios	5.1	0	0		0		0
		Oráculo para gerar as saídas esperadas	3.5	0					
		Extrator de dados de modelos descrevendo o sistema	2.8	0					
4	Gerador manual de testes	Mecanismo de captura-reprodução	1	1	4.3		1.7		1.7
		Uso de linguagens de alto nível	1.2	1					
		Acesso as funções do SO	2.1	1					
		Acesso ao mecanismo de persistência	2.1	0					
5	Integrador	Integração com ferramentas de acompanhamento de falhas	5.1	0	0		0		0
		Integração com ferramentas de gestão de configuração	5.4	0					
6	Avaliador de testes	Avaliador de cobertura	3.5	1	3,5		0		0
7	Gerador de relatórios	Gerador de relatório com formato definido pelo usuário	5.6	0	0		0		0
8	Suporte da ferramenta	Cadastro de usuários	3.3	0	0		0		0
		Cadastro de grupos	3.3	0					
		Cadastro de projeto	3.5	0					
		Cadastro de equipe	3.5	0					
9	Arquitetura da ferramenta	Uso de software livres	2.6	1	6.1		3.5		4.35
		Seguir um guia de estilo	3.5	1					
		Utilizar terminologia adequada ao contexto	4.9	0					
		Help om-line	2.1						

10	Abertura da ferramenta	Manual de usuário	2.8	1	4.9		4.9		4.9
		Sítio de apoio com exemplos de uso	2.1	1					
11	Executor de teste	Agrupador e escalonador de testes	2.3	0.5	6.02		4.2		4.2
		Gerador de log de execução de testes	5.6	0.87					
		Comparador de arquivos ignorando padrões configuráveis	1.2	0					
		Povoador de dados	7.2	0					
		Analisador de Falhas	1	0					
			Satisfação do Cliente		24.82%		14.30%		15.15%

Fonte: Comparação entre as ferramentas de testes MonkeyTalk, Sikuli e Selendroid. (Um Estudo sobre Ferramentas para o Teste de Aplicações Android no Contexto do Laboratório LEDS).

O estudo descobriu que o MonkeyTalk Ferramenta é o mais apropriado para as necessidades do cliente, fornecendo 24,82% do atendimento total ao cliente, 61% a mais do que a segunda ferramenta mais popular. Aspectos técnicos, como o grupo "Auxílio da Ferramenta", representaram 5% do suporte total ao cliente. O MonkeyTalk também teve um impacto significativo no grupo "Executor do Teste", que foi responsável por quase 100% do log de testes. O estudo destaca a necessidade de melhores ferramentas de teste para atender não apenas às necessidades do cliente, mas também a todos os desenvolvedores.

2.2 Trabalho sobre “ferramentas free para teste de software: um estudo comparativo”. Wanessa mariana silva, angélica toff ano seidel calazans.

O objetivo geral desta pesquisa foi identificar ferramentas free para teste

de software, analisar e contextualizar sua atuação no processo de teste. Foram objetivos específicos deste trabalho Conceituar o processo de teste identificando suas fases e atividades; Identificar a importância do processo de teste e da automação desse processo; Identificar as ferramentas free de teste existentes e sua atuação nesse processo; e Identificar objetivos e funcionalidades de cada ferramenta.

4.3.1 Metodologia

O método adotado foi o estudo de caso, e os instrumentos de coleta de dados a serem aplicados são pesquisa documental e questionários semiestruturados para obter a percepção dos técnicos da área de testes em TI. Para análise dos dados coletados dos questionários, foi utilizada a técnica de análise de conteúdo.

É interessante ressaltar que o Estudo de Caso é uma investigação empírica que pesquisa fenômenos dentro do seu contexto real e no qual o pesquisador não tem controle sobre eventos e variáveis, buscando descrever, compreender e interpretar a complexidade de um caso concreto (KOLHBACHER, 2006).

O questionário foi aplicado aos técnicos da área de TI da Embrapa, RSI e Banco do Brasil. O critério para seleção dos participantes foi definido, principalmente, com relação ao conhecimento e utilização das ferramentas de teste no ambiente organizacional, e ao tempo de trabalho na área de TI superior a 11 anos. Foi considerada também a escolaridade do respondente igual ou superior a lato sensu. A escolha dessas três variáveis visa à obtenção da percepção técnica dessa solução no contexto organizacional. Com relação à idade, todos são maiores de 18 anos. Responderam ao questionário três técnicos que trabalham com testes nessas instituições.

4.3.2 Aplicação da pesquisa e resultados

Com base nos objetivos gerais e específicos da pesquisa e para selecionar as ferramentas de teste a serem analisadas, foi definido o seguinte string de pesquisa que foi implementado no google acadêmico: ferramenta teste software free, ferramenta teste sistema free, ferramenta teste software open source ou ferramenta teste sistema open source. Foram encontrados 359 resultados na pesquisa realizada no período de 01 ano 2011/2012. A Figura 2 apresenta o quantitativo de resultados obtidos na pesquisa efetuada. O material foi analisado e foram descartados artigos que não se relacionam aos objetivos da pesquisa.

Para selecionar o conjunto inicial de ferramentas para o estudo de todos os artigos classificados como pertinentes, foram obtidos e lidos títulos, palavras-chave e resumos, visando identificar as ferramentas citadas, funcionalidades e a sua utilização. A consulta feita em livros (citados nas Referências) foi realizada observando-se os mesmos termos. A Figura 3 apresenta o quantitativo de resultados obtidos na pesquisa realizada na literatura pesquisada sobre o teste.

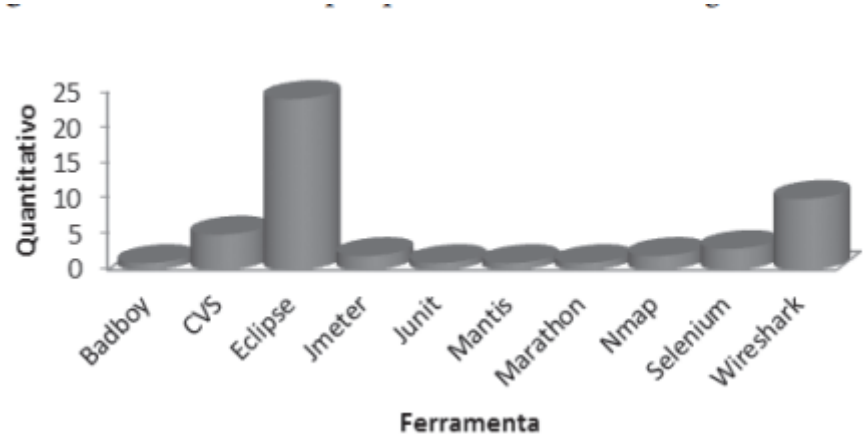

Figura 2 - Resultado da pesquisa utilizando o string adotado.

Fonte: Autoria própria.

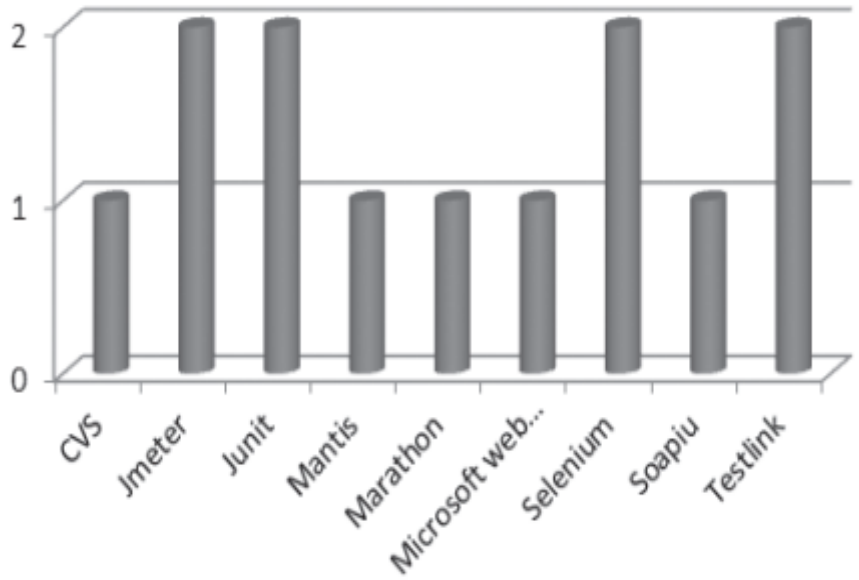

Figura 3 - Resultado da pesquisa utilizando o string adotado na literatura consultada.

Fonte: Autoria própria.

As ferramentas identificadas em ambas as coletas, foram classificadas em dois grupos: ferramentas de gestão do processo de teste e ferramentas vinculadas ao tipo de teste a ser realizado. Essas ferramentas são descritas a seguir. É interessante ressaltar que, por serem ferramentas livres, foram consultados também os sites das respectivas ferramentas para obter as suas características e aspectos positivos e negativos.

4.3.3 Ferramentas de automação de teste

4.3.3.1 Ferramentas de gestão do processo de teste

4.3.3.1.1 Testlink

Segundo Molinari (2010), Testlink é uma ferramenta que apoia “projetos”, sendo sua estrutura básica construída a partir da definição de um projeto; tudo debaixo dele será informado (requisitos de testes, especificações

gerais, planos, testes, execuções etc.). O TestLink é uma aplicação Open Source que possibilita associar um conjunto de Test Cases a um testador e acompanhar os resultados da execução dos testes, assim como, gerar relatórios com diversas métricas para o acompanhamento da execução dos testes. Além disso, o TestLink oferece um recurso para registrar e organizar os requisitos do projeto, assim como, associar os Test Cases aos requisitos.

Dessa forma, pode-se garantir o rastreamento entre os requisitos e os Test Cases por meio de uma matriz de rastreabilidade (Caetano, 2007, p. 28). Dentre suas principais funcionalidades pode-se destacar: Pode ser executado em qualquer plataforma que suportar PHP/Apache/Mysql (Windows, Linux, Mac, Solaris, AS400/i5 etc.). Traduzido em várias línguas diferentes (incluindo “Brazil Portuguese”). Controle de acesso e níveis de permissões por papéis (Líder, Testador, etc.). Criação ilimitada de projetos e casos de testes. Integração com ferramentas de gestão de defeitos (Bugzilla, Mantis).

4.3.3.1.2 CVS

Basicamente, o CVS armazena em seu repositório as modificações realizadas num arquivo ao longo do tempo; cada modificação é identificada por um número chamado Revisão. É especialmente útil para se controlar versões de um software durante seu desenvolvimento, ou para composição colaborativa de um documento, pois toda Revisão no CVS armazena as modificações realizadas, quem realizou as modificações e quando foram realizadas, entre outras informações .

Além disso, o CVS conta com um mecanismo capaz de controlar os acessos simultâneos e as modificações paralelas, garantindo a integridade das modificações e a atomicidade das operações. São algumas vantagens do CVS, segundo Caetano (2007), a possibilidade de comparar diferentes versões de um arquivo, pedir um histórico completo das alterações, ou baixar uma determinada versão do projeto, ou de uma data específica, não necessariamente a versão mais atual.

Caetano (2007) cita algumas desvantagens da ferramenta, tais como: os arquivos em um repositório CVS não podem ser renomeados a partir do cliente, eles devem ser explicitamente removidos e adicionados. Não é permitido também que os diretórios sejam movidos ou renomeados. Cada subdiretório em questão deve ser individualmente removido e adicionado. Além disso, não há checkout reservado (permite que dois usuários alterem o mesmo arquivo ao mesmo tempo), podendo ser mais custoso resolver o conflito do que evitar que ele ocorra.

4.3.3.1.3 MANTIS

O Mantis é uma ferramenta gratuita, escrita em PHP e de código aberto, permitindo uma grande variedade de customizações. Um diferencial é a possibilidade de utilização em diversos idiomas, incluindo o português (Caetano, 2007).

São 6 níveis de usuários (Visualizador, Relator, Atualizador, Desenvolvedor, Gerente e Administrador), cada um com diferentes atribuições e permissões; por exemplo, um visualizador pode apenas ver os registros de erros, enquanto o administrador tem poderes completos. Os casos reportados são organizados por projetos e subprojetos e designados a um usuário que recebe uma notificação por e-mail. Após solucionar o erro, o responsável deve responder ao remetente, que avaliará se o caso pode ser encerrado. Todas as ações são registradas em logs, gerando relatórios consistentes sobre o andamento dos casos (Mantis, 2012).

Com o Mantis, um gerente de projetos pode ter controle total dos erros encontrados pelos testadores, o tempo de resposta dos desenvolvedores e quais desenvolvedores estão com quais erros. Há também uma opção que permite a um usuário acompanhar um caso e ser avisado sobre qualquer alteração no seu registro (Mantis, 2012).

4.3.3.1.4 BUGZILLA

A ferramenta Bugzilla é um “Sistema de Acompanhamento de defeito” ou “bug tracking system”. A ferramenta possibilita que indivíduos ou grupos de desenvolvedores mantenham o controle de bugs em seus produtos. Entre os muitos benefícios desse programa é que ele é muito escalável, mantendo os dados confidenciais e seguros, além de ter recursos avançados de pesquisa, incluindo a capacidade de armazenar as pesquisas anteriores (BUGZILLA, 2012).

O Bugzilla permite registrar um ciclo de vida para cada alteração de código efetuada no projeto, desde a sua solicitação até a sua conclusão. Com a ferramenta, é possível consultar em tempo real a quantidade de alterações do sistema, quais bugs estão sendo corrigidos, quem está corrigindo determinado bug etc. Também é considerada uma ferramenta para testes unitários. Além disso, o Bugzilla oferece outras funcionalidades, tais como: gerenciamento da garantia de qualidade (QA), gerenciamento de bugs, fácil comunicação com os participantes do processo, segurança e confidencialidade, consultas avançadas e sistemas de permissões com perfis editáveis (BUGZILLA, 2012).

4.3.3.2 Ferramentas classificadas por tipos de testes

3.3.3.2.1 BADBOY

BadBoy (BADBOY, 2012) é uma ferramenta desenvolvida em C++, não estando disponível para Linux, que grava todas as ações em uma página web (java, php, ruby, etc.). A ferramenta é capaz de gravar como um macro tudo que é feito na página web como requisitos, parâmetros, alertas, respostas etc. Com essa ferramenta, pode-se também alterar parâmetros das páginas que estão sendo testadas, efetuar asserções por texto (simples ou html), cor, javascript , etc.

Algumas características da ferramenta são (BADBOY, 2012): Modo de gravação em Internet Explorer e Firefox; É uma ferramenta de teste funcional e teste de carga; e Tem capacidade de exportar para Jmeter.

4.3.3.2.2 JUNIT

O JUnit é uma ferramenta de testes unitários que fornece uma série de classes que permitem chamar outras classes (que são as que devem ser testadas) “[...] é uma ferramenta de utilização simples, além de ser Open source o que permite que o código seja baixado e modificado”. (MOLINARI, 2010, p 65). Em termos de funcionamento, o JUnit é um programa usado para executar testes unitários em virtualmente qualquer aplicação. Os testes com JUnit são efetuados escrevendo “casos de teste” em Java, compilando-os e executando as classes resultantes com o “JUnit Test Runner”.

A ferramenta permite a criação rápida de código de teste enquanto possibilita um aumento na qualidade do sistema desenvolvido e testado. Além disso, é uma ferramenta amplamente utilizada pelos desenvolvedores da comunidade código-aberto e consequentemente disponibiliza um grande número de exemplos. É interessante ressaltar que os testes, uma vez escritos, são executados rapidamente sem que, para isso, seja interrompido o processo de desenvolvimento (JUNIT, 2012). Além disso, o JUnit possibilita checar os resultados dos testes, fornecendo uma resposta imediata. Possibilita também criar uma hierarquia de testes que permitirá testar apenas uma parte do sistema ou todo ele. Ou seja, escrever testes com JUnit permite ao programador a otimização do tempo gasto na depuração do código.

4.3.3.2.3 SELENIUM

Conforme Molinari (2010), o Selenium é uma ferramenta de automação de testes funcionais web, sendo que ela compõe um pacote maior denominado “Open QA Selenium”, sendo uma grande concorrente com outras ferramentas pagas que dominam o mercado de testes. O Selenium é uma ferramenta Open

Source usada para a criação de testes de regressão automatizados para aplicações WEB. O Selenium foi escrito utilizando Java Script e DHTML. Em função disso, os testes rodam diretamente a partir do navegador. Na realidade, em virtude dessa característica do Selenium, os testes podem rodar virtualmente em qualquer navegador que suporte JavaScript (Internet Explorer, Firefox, Opera, Safari, Konqueror etc).

É dividido em três modos diferentes (SELENIUM, 2012): Selenium IDE: é uma ferramenta criada com um plug-in do navegador Mozilla Firefox, permite gravar o que é feito nesse browser em um script cujo conteúdo tem forma de tabela e assim repetir, se necessário, as ações gravadas no browser; Selenium Remote Control (RC): permite salvar o que está gravado em outras linguagens, tais como Ruby e Java. Nele você pode alterar o que foi gravado acrescentando loops funções e bibliotecas que venham a criar; Selenium Grid: permite que você distribua seus testes em diversos computadores e os execute de forma coordenada.

4.3.3.2.4 MARATHON

O Marathon é uma ferramenta Open Source de testes de regressão automatizados, para aplicações Java desenvolvidas com o toolkit gráfico Swing. Com essa ferramenta, podem ser criados testes automatizados por meio da captura das ações (cliques do mouse, digitação, etc.). Essas ações são convertidas em scripts Jython (implementação Java da linguagem de script Python) para que se possa executar posteriormente o teste (MARATHON, 2012). Dentre as suas principais características, podemos destacar as seguintes (CAETANO, 2007, p.83): Ambiente de desenvolvimento integrado onde se pode capturar depurar e executar os testes automatizados; Módulos reutilizáveis para facilitar a manutenção; Fixtures (scripts em Python cuja principal função é criar as pré-condições de execução dos testes, assim como, a posterior limpeza dos recursos criados); Component Resolver e Custom Component Resolver (recurso utilizado para detectar todos os componentes do Swing (botões, campos de edição, menus, etc.), assim como os seus métodos

e propriedades).

4.3.3.2.5 SOAPUI

Segundo Caetano (2007), o SoapUI é uma ferramenta Open Source multiplataforma escrita em Java cuja principal função é testar WEB Services. WEB Service é uma tecnologia baseada em XML e HTTP cuja principal função é disponibilizar serviços interativos na WEB que podem ser acessados (ou consumidos) por qualquer outra aplicação independente da linguagem ou plataforma em que a aplicação foi construída. O soapUI permite rapidamente criar e executar de forma automatizada testes de regressão funcional, conformidade e testes de carga.

Em um ambiente de teste único, soapUI fornece cobertura de teste completo e suporta todos os protocolos e tecnologias. O SoapUI facilita todo o processo de criação e depuração dos testes por meio de uma interface gráfica intuitiva. Dentre as suas principais características, podemos destacar as seguintes (SOAPUI, 2012): Importação e geração automática das requisições descritas no WSDL; Capacidade de gerenciar um número ilimitado de requisições para cada operação; Gerenciamento de múltiplos endpoints para cada WEB Service; Validação das requisições e respostas contra as suas definições no WSDL; Testes funcionais, desempenho e stress; Execução de diversos testes em paralelo; Editores com syntax highlight e formatação automática; Suporta expressões XPATH; e Suporta criação de testes complexos utilizando scripts Groovy.

4.3.3.2.6 JMETER

Apache JMeter pode ser usado para testar o desempenho tanto em recursos estáticos e dinâmicos (arquivos, Servlets, scripts Perl, objetos Java, Bases de Dados e Consultas, servidores FTP e muito mais). Ele pode ser usado para simular uma carga pesada em um servidor de rede, ou objeto para testar a resistência ou para analisar o desempenho global de diferentes tipos

de carga. Pode-se realizar também uma análise gráfica de desempenho ou comportamento do objeto sob carga pesada concorrente (JMETER,2012). É uma ferramenta open source mantida como parte do projeto Jakarta da Apache Software Foundation (JMETER, 2012), que fornece suporte organizacional, legal e financeiro para uma ampla gama de projetos de software open source (MOLINARI, 2010).

Simplificando, o Jmeter permite fazer testes de cargas e performance em sistemas web,conexões FTP (File Transfer Protocol/ Protocolo de Transferência de Arquivos), web services, objetos Java e no próprio JUnit testes, ou seja, testes de carga em testes de unidade que utilizam o framework do JUnit. Principais recursos oferecidos pelo JMeter, segundo Caetano (2007, p. 110): Pode ser executado em qualquer plataforma que suporte a máquina virtual do Java 1.4 ou superior (Solaris, Linux, Windows, OpenVMS Alpha 7.3+); Suporta a criação de testes de performance para os protocolos (HTTP, JDBC, FTP, JMS, LDAP, SOAP, entre outros.); Possibilidade de executar os testes em computadores distribuídos; Criação de asserções para validar os requisitos de performance e funcionalidade; Possibilidade de monitorar a performance de um servidor Apache Tomcat; Permite a utilização de pré-processadores e pós-processadores para modificar o comportamento das requisições; e Suporta diversos tipos de monitores para avaliar a performance da aplicação em teste.

4.3.3.2.7 MICROSOFT WEB APPLICATION STRESS

Segundo afirma Caetano (2007), Microsoft WEB Application Stress (WAS) é uma ferramenta gratuita para o Windows. Por meio do WAS, você poderá realizar testes de performance, volume e estresse nas suas aplicações WEB. O WAS é uma ferramenta que possui menos recursos que o JMeter, no entanto, ela possibilita testes rápidos. Os testes podem ser escritos manualmente, por meio da criação manual das requisições HTTP (GET, POST, etc.), ou gravados automaticamente enquanto você navega na aplicação simulando um usuário real por meio de um Proxy Server (CAETANO, 2007).

Entre os diversos recursos oferecidos pelo WAS, deve-se destacar a possibilidade de executar os testes em computadores distribuídos, a configuração do comportamento do teste de performance e a possibilidade de monitorar a utilização dos recursos dos servidores Windows (apenas no Windows 2000 e NT).

4.3.3.2.8 WIRESHARK

O Wireshark, anteriormente conhecido como Ethereal, é um programa que analisa o tráfego de rede e o organiza por protocolos. O primeiro passo na avaliação de vulnerabilidades é ter uma imagem clara do que está acontecendo na rede. O Wireshark funciona em modo promíscuo para capturar todo o tráfego de um domínio de broadcast TCP. (WIRESHARK, 2012). Filtros personalizados podem ser ajustados para interceptar o tráfego específico, por exemplo, para capturar a comunicação entre dois endereços IP, ou capturar consultas DNS baseados na rede. Os dados de tráfego podem ser despejados em um arquivo de captura, que pode ser revisto mais tarde. Os filtros adicionais também podem ser definidos durante a revisão. Normalmente, o testador está à procura de endereços IP vadios, pacotes com endereços forjados, pacotes desnecessários e geração de pacotes suspeitos a partir de um único endereço IP. O Wireshark dá uma visão ampla e clara do que está acontecendo na sua rede.

4.3.3.2.9 NMAP

Segundo Nmap (2012), a ferramenta Nmap (Network Mapper) é considerada código aberto para exploração de rede e auditoria de segurança. Ela foi desenhada para escanear rapidamente redes amplas, embora também funcione bem em hosts individuais. O Nmap utiliza pacotes IP em estado bruto para determinar quais hosts estão disponíveis na rede, quais serviços (nome da aplicação e versão) os hosts oferecem, quais sistemas operacionais (e versões) eles estão executando, que tipos de filtro de pacotes/firewalls estão

em uso, e dezenas de outras características.

Embora o Nmap seja normalmente utilizado para auditorias de segurança, é útil para tarefas rotineiras como inventário de rede, gerenciamento de serviços de atualização agendados e monitoramento de host ou disponibilidade de serviço. A saída do Nmap é uma lista de alvos escaneados, com informações adicionais de cada um dependendo das opções utilizadas. Uma informação chave é a "tabela de portas interessantes", que lista o número da porta e o protocolo, o nome do serviço e o estado (NMAP, 2012).

Os estados das portas podem ser abertos (open), filtrados (filtered), fechados (closed) ou não-filtrados (unfiltered). "Aberto" significa que uma aplicação na máquina-alvo está escutando as conexões/pacotes naquela porta. "Filtrado" significa que o firewall ou outro obstáculo de rede está bloqueando a porta de forma que o Nmap não consegue dizer se ela está aberta ou fechada. Portas fechadas não possuem uma aplicação escutando nelas, embora possam abrir a qualquer instante. Portas são classificadas como não-filtradas quando respondem às sondagens do Nmap, mas o Nmap não consegue determinar se as portas estão abertas ou fechadas.

O Nmap reporta as combinações aberta/filtrada e fechada/filtrada quando não consegue determinar qual dos dois estados descreve melhor a porta. A tabela de portas também pode incluir detalhes de versão de software quando a detecção de versão for solicitada. Quando um scan do protocolo IP é solicitado (-sO), o Nmap fornece informações dos protocolos IP suportados ao invés de portas que estejam abertas.

A ferramenta é gratuita e encontrada nas versões Linux, Windows (95, 98, NT, Me, 2K e XP), Mac OS, Solaris, FreeBSD e OpenBSD. As versões para Windows e Linux contam com uma interface gráfica que facilita a vida dos usuários. O Nmap detecta dispositivos remotos e, na maioria dos casos, identifica firewalls e roteadores, além de sua marca e modelo. Pode-se usar o Nmap para verificar quais portas estão abertas e se essas portas podem ser exploradas ainda mais em ataques simulados.

4.3.3.2.10 ECLIPSE TPTP

O Teste de Eclipse (ECLIPSE, 2012) e Performance Tools Platform Project (TPTP), fornece uma plataforma aberta e serviços que permitem aos desenvolvedores de software construírem teste único e ferramentas de desempenho, tanto open source e comercial, que podem ser integrados com plataformas e com outras ferramentas. O Eclipse não é uma IDE propriamente dita; é um arcabouço para desenvolvimento de ferramentas, sendo extensível, aberto e portável. Por meio de plugins, diversas ferramentas podem ser combinadas, criando um ambiente de desenvolvimento integrado.

A ferramenta possibilita assim, uma mesma plataforma ser utilizada para vários papéis de desenvolvimento: programador de componentes, integrador, responsável por testes, web designer. A ferramenta engloba todo o teste e ciclo de vida, desempenho de testes iniciais para monitoramento de aplicações de produção, incluindo a edição de teste e execução, monitoramento, rastreamento e perfis, e capacidades de análise de log. A plataforma suporta um amplo espectro de sistemas de computação de alto desempenho, múltiplos ambientes Java, substituição dinâmica de código, refatoração, etc. O Eclipse possui, ainda, modelos de ajuda prontos (Wizards) para efetuar várias tarefas. Criar testes usando JUnit é uma delas.

4.3.3.3 Correlação entre modelos ou tipos de teste e ferramentas

A Tabela 1 apresenta as ferramentas de teste analisadas e classificadas em dois grandes grupos: Ferramentas de gestão do processo de teste e Ferramentas vinculadas ao tipo de teste. Além disso, essa tabela identifica, a partir dos modelos citados, os tipos de testes e as ferramentas de testes que podem ser utilizadas para a automatização e gerenciamento desse processo. Por exemplo, a identificação da abordagem de teste a ser utilizada (RUP) e a especificação do projeto de teste (segundo padrão IEEE 829) poderiam englobar a utilização das ferramentas Testlink, CVS e Mantis.

Quadro 14 – Correlação modelos vs processo e ferramentas.

<table>
<tr><th rowspan="2">Assunto</th><th colspan="5">Abordagem</th></tr>
<tr><th>Processo RUP</th><th>Padrão IEEE 829 para documentação de teste</th><th>Tipo de teste (Pressman, 1995)</th><th>Classificação de ferramentas (Caetano, 2007)</th><th>Ferramentas</th></tr>
<tr><td rowspan="5"></td><td rowspan="5">-Definição da missão
-Verificação da abordagem</td><td rowspan="5">Plano de teste

Especificação do projeto de teste</td><td rowspan="5">Não se aplica</td><td>Gestão de testes</td><td>Test link</td></tr>
<tr><td>Controle de versão</td><td>CVS</td></tr>
<tr><td rowspan="3">Gestão de defeitos</td><td>Mantis</td></tr>
<tr><td>Bugzilla</td></tr>
<tr><td rowspan="11"></td><td rowspan="11">-Definição da missão
-Verificação da abordagem</td><td rowspan="8">Plano de teste

Especificação do projeto de teste</td><td rowspan="4">-Teste de validação
- Teste de sistema</td><td rowspan="4">Automação de testes funcionais e de aceitação</td><td>Selenium</td></tr>
<tr><td>Marathon</td></tr>
<tr><td>Soapui</td></tr>
<tr><td>Badboy</td></tr>
<tr><td rowspan="3">-Teste de desempenho
- Teste de stress</td><td rowspan="3">Automação de teste de performance</td><td>Jmeter</td></tr>
<tr><td>Eclipse TPTP</td></tr>
<tr><td>Microsoft Web application

Stress</td></tr>
<tr><td rowspan="2">-Segurança</td><td rowspan="2">Não se aplica</td><td>Wireshark</td></tr>
<tr><td>Nmap</td></tr>
<tr><td rowspan="3">Especificação dos casos de testes</td><td rowspan="3">-Teste de unidade
-Teste integração</td><td rowspan="3">Gerencia de casos de teste</td><td>Badboy</td></tr>
<tr><td>JUnit</td></tr>
<tr><td>Bugzilla</td></tr>
</table>

Fonte: Autoria própria.

4.3.3.4 Resultados da aplicação dos questionários

As respostas obtidas pela aplicação do questionário demonstraram, nas empresas Embrapa, RSI e Banco do Brasil, que: A utilização da ferramenta Mantis, em duas das organizações pesquisadas, foi avaliada satisfatoriamente por seus objetivos. Uma das organizações pesquisadas utiliza apenas teste de sistema, uma das organizações trabalha somente com testes de unidade e desempenho.

Uma das organizações utiliza todos os tipos de teste que são: teste de unidade, teste de integração, validação, sistema, desempenho, stress e segurança. Testlink, CVS, Bad boy foram avaliadas por um dos participantes como atendendo satisfatoriamente os objetivos. Jmeter foi avaliada como atendendo parcialmente os objetivos. Cvs e Selenium foram avaliadas por um dos participantes como atendendo satisfatoriamente os objetivos. O objetivo geral deste trabalho foi identificar ferramentas gratuitas para teste de software, analisando e contextualizando sua atuação no processo de teste. Para isso, foram descritos os conceitos de teste e processo de teste, além de identificar suas fases e atividades.

O estudo mostra que a automação é essencial para o desenvolvimento e a manutenção de software de qualidade. Algumas das ferramentas gratuitas de teste, já referenciadas academicamente e em livros, foram selecionadas e sua atuação no processo foi analisada. Identificaram-se vários recursos e funcionalidades dessas ferramentas, que estão disponíveis para qualquer usuário baixá-las a qualquer momento. Além disso, as ferramentas de teste foram relacionadas às fases do processo de teste. Foram pesquisadas na literatura e na Web as vantagens e desvantagens dessas ferramentas.

Para complementar o trabalho, foi realizado um estudo de caso em três instituições para obter a percepção dos técnicos da área de testes de TI sobre a utilização dessas ferramentas. Identificou-se que a ferramenta Mantis é a mais utilizada nas instituições pesquisadas e tem uma avaliação satisfatória.

5 ANÁLISE CRÍTICA

A análise dos três artigos selecionados demonstrou a importância fundamental dos testes de software para assegurar a qualidade e a confiabilidade dos sistemas desenvolvidos. Um ponto comum entre os artigos é a ênfase na automação de testes como uma estratégia eficaz para aumentar a eficiência e reduzir o tempo necessário para a execução de testes repetitivos. Além disso, a automação permite que os testes sejam realizados de maneira mais consistente, garantindo a reprodutibilidade dos resultados e a identificação precoce de falhas.

Os artigos também reforçam a importância dos testes unitários e de integração, destacando que essas práticas são essenciais para garantir o correto funcionamento de cada componente individual e a interação entre eles. Testes de caixa preta e caixa branca foram mencionados como abordagens complementares, fornecendo uma análise completa do comportamento interno e externo do software.

Os artigos analisados sugerem abordagens eficazes para a realização de testes. Entre elas, destaca-se a automação, apontada como a forma mais eficiente de conduzir testes repetitivos e obter resultados confiáveis. Ferramentas como JUnit e TestNG são amplamente recomendadas para a execução de testes unitários, enquanto o Selenium é uma das ferramentas preferidas para testes de integração e automação de testes funcionais.

Além disso, os testes de caixa preta e caixa branca desempenham um papel crucial na avaliação da lógica interna do software e no comportamento do sistema como um todo. Essas ferramentas, quando usadas de forma integrada, proporcionam uma cobertura de testes robusta e eficiente, garantindo a qualidade do software durante todas as fases do desenvolvimento.

No entanto, apesar dos avanços nas abordagens e nas ferramentas de teste de software, ainda existem desafios significativos que precisam ser superados. Um dos principais problemas apontados pelos artigos é a cobertura insuficiente de testes não-funcionais, como testes de desempenho, usabilidade e segurança. Essas áreas exigem ferramentas mais especializadas, capazes de avaliar aspectos críticos que não são cobertos pelos testes tradicionais de

funcionalidade. Para atender a essa demanda, a solução envolve a adoção de ferramentas especializadas para cada tipo de teste não-funcional, além da integração dessas práticas em um ciclo contínuo de desenvolvimento. A automação de testes de desempenho, segurança e confiabilidade, combinada com métodos que envolvem interação humana, como nos testes de usabilidade, proporciona uma cobertura mais robusta. Investir nessas abordagens garante não apenas a qualidade funcional do software, mas também sua capacidade de oferecer uma experiência de usuário eficiente e segura, além de assegurar a estabilidade em ambientes de alta demanda.

Além disso, a falta de suporte adequado para ambientes específicos, como aplicativos móveis e sistemas distribuídos, foi uma preocupação recorrente. Isso mostra que, embora as ferramentas de teste atuais sejam eficazes para uma ampla gama de cenários, há espaço para melhorias, especialmente em termos de adaptação a novos contextos tecnológicos.

6 CONCLUSÃO

A análise dos três artigos selecionados ressalta a importância dos testes de software no desenvolvimento de sistemas robustos e confiáveis. Em todos os casos, ficou evidente que a automação é a abordagem mais recomendada, especialmente em projetos que exigem execução repetitiva e confiabilidade. A automação de testes não só otimiza o tempo e reduz o esforço manual, mas também garante a consistência nos resultados e a detecção precoce de falhas.

Os artigos sugerem algumas das melhores práticas e ferramentas para garantir testes eficazes. Entre elas, destacam-se o uso de ferramentas como JUnit e TestNG para testes unitários e Selenium para a automação de testes funcionais e de integração. Essas ferramentas foram amplamente recomendadas, pois oferecem uma cobertura robusta, eficiência e flexibilidade no processo de testes, essenciais para assegurar que o software atenda às expectativas de qualidade.

Entretanto, existem pontos que ainda necessitam de maior observação. A cobertura de testes não-funcionais, como testes de desempenho, usabilidade

e segurança, é uma área que apresenta desafios. Muitos dos artigos revelam que as ferramentas atuais não oferecem suporte suficiente para esses tipos de teste, especialmente em contextos específicos, como aplicativos móveis e sistemas distribuídos. Isso ressalta a necessidade de desenvolver soluções mais especializadas e adaptáveis, capazes de lidar com esses cenários mais complexos.

O estudo comparativo demonstrou que, embora existam ferramentas e abordagens eficazes para testes de software, a escolha da ferramenta mais adequada deve levar em conta o tipo de projeto e os requisitos específicos da equipe de desenvolvimento. Ferramentas que oferecem desempenho, flexibilidade e suporte à automação são fundamentais para garantir a qualidade do sistema. Ainda assim, há um espaço significativo para melhorias no suporte a testes não-funcionais e adaptação a novos contextos tecnológicos. Dessa forma, com uma escolha criteriosa de ferramentas e uma abordagem de testes bem estruturada, é possível garantir que os sistemas desenvolvidos atendam aos padrões de qualidade esperados e funcionem conforme as necessidades dos usuários.

REFERÊNCIAS

BARNUM, Carol. ***Noções básicas de teste de usabilidade: Preparar, apontar...Teste!*****. 2ª ed. Morgan Kaufmann, 2020.**

BUGZILLA. ***Bugzilla: um sistema de envio de defeitos.*** **2012.**

CAETANO, ARS. ***Ferramentas de teste de software: uma visão geral.*** **2007.**

CALDEIRA, AMF. ***Avaliação de ferramentas de teste de software: um estudo de caso.*** **2014.**

DE SOUSA, C. ***Ferramentas de Testes de Software.*** **São Paulo: Atlas, 2019.**

-+DÓREA, AD de O.; CARVALHO, F. de S.; SANTOS, MT; NETO, MCM; MOISES, D. ***Avaliação de ferramentas de automação para engenharia de testes.*** **2008.**

FERREIRA, M. ***Um estudo sobre ferramentas para o teste de aplicações Android no contexto do laboratório LEDS.*** **2016.**

IAN, S. ***Engenharia de Software: uma abordagem profissional.*** **2011.**

JMETAR. ***JMeter: uma ferramenta de teste de desempenho.*** **2012.**

KOLHBACHER, F. ***Estudo de caso: uma abordagem qualitativa.*** **2006.**

MOLINARI, A. ***TestLink: uma ferramenta de gestão de testes.*** **2010.**

NMAP. ***Nmap: uma ferramenta de exploração de rede e auditoria de segurança.*** **2012.**

PRESSMAN, Roger S. ***Engenharia de Software: uma abordagem profissional.*** **7ª ed. Porto Alegre: AMGH, 2011.**

ROGER, P. ***Engenharia de Software: uma abordagem prática.*** **2005.**

SELÊNIO. ***Selenium: uma ferramenta de automação de testes funcionais na web.*** **2012.**

SPIRLANDELI, AC; ROLAND, MT. ***DevOps: A integração contínua sem desenvolvimento de software.*** **Rio de Janeiro: Brasport, 2019.**

VELOSO, A. et al. ***Método de avaliação de ferramentas de teste de software.*** **2010.**

WEBER, R. ***A importância da automação de testes.*** **2001.**

WIRESHARK. ***Wireshark: uma ferramenta de análise de tráfego de rede.*** **2012.**

SOMMERVILLE, Ian. ***Engenharia de software*****, 9ª. São Paulo, SP, Brasil, p. 63, 2011.**