UNIVERSIDADE FEDERAL DO PARANÁ

ALEJANDRO GEMIN ROSALES
JOÃO PEDRO DE MATOS DE OLIVEIRA

ANÁLISE DE SOFTWARES DE TESTE WEB A PARTIR DO GUIA DE AVALIAÇÃO DE QUALIDADE DE SOFTWARE MEDE-PROS

CURITIBA

2023

ALEJANDRO GEMIN ROSALES

JOÃO PEDRO DE MATOS DE OLIVEIRA

ANÁLISE DE SOFTWARES DE TESTE WEB A PARTIR DO GUIA DE AVALIAÇÃO DE QUALIDADE DE SOFTWARE MEDE-PROS

Artigo apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, setor de Ciências Exatas, da Universidade Federal do Paraná.

Orientadores: Profª. Dra. Leticia Mara Peres e Prof. Dr. Andrey Ricardo Pimentel

CURITIBA

2023

RESUMO

O presente Trabalho – vinculado aos preceitos da disciplina de TCC é intitulado – Análise de Softwares de Teste Web a partir do Guia de Avaliação de Qualidade de Software MEDE-PROS. Sabe-se que a Qualidade de Software encontra-se integrada com a Engenharia de Software e tem como principais objetivos garantir e validar os produtos que irão ao mercado consumidor. A avaliação da qualidade do produto de software é um processo de grande importância e que beneficia tanto o desenvolvedor do sistema quanto o usuário final. Para tanto, buscou-se aqui descrever e analisar alguns softwares de testes e suas especificidades. Para a conclusão deste breve relato, foi consultada literatura técnica que aborda o tema específico elencado e objeto deste estudo. O objetivo deste artigo é obter uma avaliação final detalhada do software através do guia MEDE-PROS e também uma análise do próprio modelo utilizado, apresentando uma proposta de melhoria tanto do guia quanto do software.

Palavras-chave: Qualidade de Software. Engenharia de Software. Softwares de Teste Web . Avaliação de Qualidade. Teste Web. MEDE-PROS.

ABSTRACT

This Work – linked to the precepts of the TCC discipline is entitled – Analysis of Web Test Software from the MEDE-PROS Software Quality assessment guide. It is known that Software Quality is integrated with Software Engineering and has as main objectives to guarantee and validate the products that will go to the consumer market. The evaluation of software product quality is a very important process that benefits both the system developer and the end user. For this purpose, we sought to describe and analyze some testing software and their specificities. For the conclusion of this brief report, technical literature was consulted that addresses the specific topic listed and object of this study. The objective of this article is to obtain a detailed final evaluation of the software through the MEDE-PROS guide and also an analysis of the model used, presenting a proposal for improvement of both the guide and the software.

Key-words: Software Quality. Software Engineering. Web Software testing. Quality evaluation. Web test. MEDE-PROS.

LISTA DE ILUSTRAÇÕES

FIGURA 1 - CONCEITOS SOBRE TESTE DE SOFTWARE. 12
FIGURA 2 - ESTRATÉGIAS DE TESTE 14
FIGURA 3 - ETAPAS DE TESTE DE SOFTWARE 15
FIGURA 4 - TESTE DE CAIXA-PRETA 18
FIGURA 5 - (A) FLUXOGRAMA E (B) GRAFO DE FLUXO 19
FIGURA 6 - CATEGORIAS DE APLICAÇÕES WEB (ADAPTADO DE KAPPEL, 2004). 20
FIGURA 7 - ESTRUTURA GERAL DO GUIA MEDE-PROS 24
FIGURA 8 - EXEMPLO DAS QUESTÕES PREENCHIDAS POR PARTE DO AVALIADOR 25
FIGURA 9 - PÁGINA UTILIZADA PARA AVALIAÇÃO DE QUALIDADE. ENCONTRADO EM: HTTPS://DEVEXPRESS.GITHUB.IO/TESTCAFE/EXAMPLE/ 44
FIGURA 10 - PÁGINA UTILIZADA PARA AVALIAÇÃO DE QUALIDADE. ENCONTRADO EM: HTTPS://WWW.SELENIUM.DEV/SELENIUM/WEB/WEB-FORM.HTML 45

LISTA DE TABELAS

TABELA 1 - OBJETIVO DO MAPEAMENTO 28
TABELA 2 - QUESTÃO DE PESQUISA 28
TABELA 3 - REPOSITÓRIOS UTILIZADOS VERSÃO FINAL 29
TABELA 4 - PALAVRAS-CHAVE 29
TABELA 5 - DISPOSITIVO UTILIZADO PARA A BUSCA 30
TABELA 6 - CRITÉRIOS DE INCLUSÃO E EXCLUSÃO 31
TABELA 7 - RESULTADO DAS BUSCAS USANDO GITHUB 31
TABELA 8 - FILTRO DE SELEÇÃO FINAL 32
TABELA 9 - PROCESSO DE SNOWBALLING 33
TABELA 10 - ITENS DE QUALIDADE 33
TABELA 11 - AVALIAÇÃO DE QUALIDADE 34
TABELA 12 - TESTCAFE 34
TABELA 13 - TESTSIGMA 35
TABELA 14 - SMART TESTAUTO FRAMEWORK 37
TABELA 15 - SELENIUM 39

SUMÁRIO

1 INTRODUÇÃO 9

2 REVISÃO BIBLIOGRÁFICA 11

2.1 INTRODUÇÃO À TESTE DE SOFTWARE 11

2.1.1 Conceitos associados ao testar Software 11

2.1.2 Defeitos no desenvolvimento do software 12

2.2 TÉCNICAS DE TESTE DE SOFTWARE 13

2.2.1 Teste Funcional (Teste de Caixa-Preta) 16

2.2.2 Teste Estrutural (Caixa-Branca) 18

2.2.3 Teste Web 20

2.3 QUALIDADE DE SOFTWARE 21

2.4 GUIA DE QUALIDADE DE SOFTWARE–MEDE-PROS 22

2.5 ISO 25010 25

3 PROTOCOLO DE MAPEAMENTO SISTEMÁTICO PARA BUSCA DE APLICATIVOS EM REPOSITÓRIOS NÃO-ACADÊMICOS 27

3.1 INTRODUÇÃO 27

3.2 ETAPA DE PLANEJAMENTO 27

3.3 ETAPA DE CONDUÇÃO 30

4 FERRAMENTAS PARA TESTE NA WEB – SELEÇÃO FINAL 41

4.1 TESTCAFE (TABELA 12) 41

4.2 SELENIUM (TABELA 15) 41

4.3 TEST SIGMA(TABELA 13) 42

4.4 SMART TESTAUTO(TABELA 14) 42

5 AVALIAÇÃO DE QUALIDADE DAS FERRAMENTAS SELECIONADAS 43

5.1 TESTCAFE 45

5.1.1 Avaliação da Instalação 45

5.1.2 Avaliação da Documentação, Interface e Software 45

5.1.3 Avaliação da Desinstalação 47

5.2 SELENIUM 47
5.2.1 Instalação 47
5.2.2 Avaliação da Documentação, Interface e Software 48
5.2.3 Desinstalação 48
5.3 TEST SIGMA 49
5.3.1 Instalação 49
5.3.2 Avaliação da Documentação, Interface e Software 49
5.3.3 Desinstalação 50
5.4 SMART TESTAUTO 51
5.4.1 Instalação 51
5.4.2 Avaliação da Documentação, Interface e Software 51
5.4.3 Desinstalação 51
6 RESULTADOS 52
6.1 AVALIAÇÃO DA UTILIZAÇÃO DO MEDE-PROS 52
6.2 PROPOSTA DE ADEQUAÇÃO DO GUIA MEDE-PROS PARA O CONTEXTO DE DESENVOLVIMENTO WEB 53
6.2.1 Instalação e Desinstalação (Portabilidade) 53
6.2.2 Completude das funcionalidades (Adequação Funcional) 54
6.2.3 Compatibilidade 54
6.2.4 Eficiência de desempenho 55
6.2.5 Outros comentários 55
7 CONCLUSÃO 56
8 REFERÊNCIAS 57
9 APÊNDICE A– AVALIAÇÃO TEST CAFE UTILIZANDO MEDE-PROS 59
10 APÊNDICE B– AVALIAÇÃO SELENIUM UTILIZANDO MEDE-PROS 61
11 APÊNDICE C– AVALIAÇÃO TEST SIGMA UTILIZANDO MEDE-PROS 63
12 APÊNDICE D– AVALIAÇÃO SMART TEST AUTO UTILIZANDO MEDE-PROS 64

1 INTRODUÇÃO

A Qualidade de Software é uma área integrada pela Engenharia de Software que tem como principais objetivos garantir o gerenciamento de requisitos, processos de desenvolvimento e manutenção de Software, segundo Vasconcelos, et al (2006). Como reguladores das medidas que definem a qualidade de um processo ou produto de Software, organizações normalizadoras como a ISO – International Organization Standardization e a IEC – International Organization Standardization, criaram um conjunto de normas a serem seguidas que padronizam esse tipo de avaliação, em conformidade com Gama e Oliveira (2011).

Utilizando o conjunto de normas ISO/IEC 25010, o Instituto Nacional de Tecnologia da Informação – ITI elaborou o Método de Avaliação da Qualidade de Produto de Software – MEDE-PROS, um guia que apoia a atividade de avaliação de qualidade de software provendo requisitos e recomendações para tal, conforme o Guia de Avaliação da Qualidade de Produto de Software (2001) e também Martinez, et al (1999).

Dentre as atividades realizadas na garantia de qualidade, técnicas de prevenção e detecção de problemas e erros no projeto são utilizadas, sendo o teste de software uma delas, segundo Vasconcelos, et al (2006). Como artifício para a atividade de teste de software, aplicações e ferramentas são desenvolvidas e utilizadas para tal fim, tornando assim necessária a avaliação de qualidade desses recursos.

Em publicações realizadas pela União Internacional de Telecomunicações [18], observou-se que o número de usuários da internet cresceu de 2 bilhões (29% da população mundial em 2010) para 5,6 bilhões em 2022 (66% da população mundial neste ano). Em 2020 estimou-se que 91,5% da população na América do Norte teria a web e 73,7% da população da América Latina teria o mesmo acesso.

O constante aumento do número de usuários na internet de aplicações web e o surgimento de aplicações híbridas (aplicações nativas - ou desktop - que

utilizam de browsers para renderizar os elementos de interação com o usuário) evidencia a necessidade da utilização de mecanismos de avaliação e garantia de qualidade por parte de usuários e desenvolvedores de software no contexto web. Um dos mecanismos que podem auxiliar na garantia de qualidade dos softwares web é a atividade de teste de software. Para isso, existem diversas ferramentas open-source na internet que permitem essa prática e a avaliação de qualidade dessas ferramentas também é necessária.

O MEDE-PROS é um documento que pode ser utilizado para a garantia dessa qualidade e regulação que as aplicações de teste na web necessitam. Porém, por ser publicado na década de 90, é possível que sua utilização não seja satisfatória quando utilizado no contexto de desenvolvimento e teste, isto porque as questões por ele abordadas podem estar desatualizadas com relação aos softwares hoje desenvolvidos.

Realizando a avaliação do MEDE-PROS e propondo melhorias para o guia, abre-se a possibilidade de sua utilização por parte de desenvolvedores que buscam avaliar e melhorar suas aplicações ou até mesmo empresas que procuram realizar a avaliação de softwares de teste para possível uso.

Desta forma, o trabalho é separado em objetivos principais e objetivos específicos. Os objetivos principais são divididos em: realizar uma análise da documentação, desempenho, interface e usabilidade de softwares de teste web em código aberto, utilizando o guia de avaliação de qualidade de software MEDE-PROS, avaliando a adequação deste guia para o contexto aplicado e propondo melhorias de acordo com modelos de qualidade. Os objetivos específicos são divididos em: pesquisar e catalogar as ferramentas de software de teste web open-source que serão analisados, implementar o modelo de qualidade do MEDE-PROS aos softwares escolhidos e propor melhorias tanto nos softwares analisados quanto no modelo aplicado.

2 REVISÃO BIBLIOGRÁFICA

2.1 INTRODUÇÃO À TESTE DE SOFTWARE

O Teste de Software consiste em colocar à prova determinado produto gerado, com o intuito de observar se ele atende as especificações e funcionalidades para o fim a que se destina, tendo por objetivo principal mostrar falhas que podem ocorrer. Assim, se permite notar e ajustar as falhas, conforme necessidade, antes de finalizar o processo e entregar o produto acabado, em conformidade com Barbosa, et al (2000).

Existem várias definições para o conceito de Teste de Software, que pode ser através de uma visão intuitiva ou de maneira formal. A seguir discorremos os principais Conceitos, Técnicas e Critérios de teste utilizados para validação de um produto.

2.1.1 Conceitos associados ao testar Software

Iniciamos esclarecendo alguns conceitos relacionados ao Teste de Software, definindo a diferença entre Defeito, Erro e Falhas, conforme terminologia padrão para Engenharia de Software IEEE [Pressman].

Defeito: caracteriza-se por algo que foi implementado de maneira errada, resultando numa inconsistência no Software.

Erro: o erro evidencia o defeito, isto é, a diferença entre o valor obtido e esperado, causado por erro humano.

Falha: a falha pode ser causada por diversos erros, porém alguns erros podem nunca causar uma falha. A falha gera um comportamento inesperado do Software.

Figura 1 - Conceitos sobre teste de software.

Fonte: Figura organizada pelos autores (2022).

2.1.2 Defeitos no desenvolvimento do software

O Teste de Software aponta os defeitos que aparecem, gerados por parte dos sujeitos que o desenvolvem. Quanto maior o número de pessoas envolvidas, dependendo da proporção do projeto, é ascendente a quantidade de defeitos que podem ser detectados. Os defeitos podem surgir durante toda a etapa de desenvolvimento do software e em diferentes níveis, dependendo da transformação que a informação manipulada sofreu no decorrer da criação e desenvolvimento.

Toda essa dinâmica aqui descrita admitirá a incidência de falhas que tiveram origem a partir da ação humana. A complexidade relativa dos testes e os erros que os testes revelam são limitados pelo escopo restrito estabelecido para o teste de unidade.

2.2 TÉCNICAS DE TESTE DE SOFTWARE

Ao iniciarmos nossos estudos pudemos observar que no rol da engenharia de software, aqueles que estão comprometidos, em todas as etapas de criação e execução, tem sua cota de responsabilidade no desenvolvimento e qualidade do produto.

O retrabalho, que é refazer etapas de trabalho que apresentaram defeitos, erros e falhas – conceitos abordados mais adiante – causam prejuízos operacionais no orçamento, bem como, atrasos na disponibilidade do produto para o cliente e mercado.

Ao se buscar a qualidade de software produzido se faz necessário, primeiramente, perceber seu significado e organizar atividades que propiciem e garantam a qualidade ao encontrar os erros e saná-los, em todas as etapas da criação e confecção do produto. Por isso, consumar atividades que propiciem controle de qualidade, usando métricas que observem o processo e a qualidade do produto finalizado, é muito importante no contexto ao se falar em qualidade de softwares.

Testagens se fazem necessárias ao considerar padrões e exigências próprias aos usos atribuídos, na percepção da qualidade, diminuindo então imperfeições operacionais. O teste também contribui no entendimento que, ao se detectar e corrigir, se distingue a categoria do produto.

Por se sentir a exigência do controle da qualidade por parte da estrutura mercadológica, testar se torna imprescindível. Mas, o que é testar?

O termo – testar – é encontrado e definido na Enciclopédia Barsa (2007) da seguinte maneira: “testificar, atestar, experimentar.” Para Pressman e Maxim (2021) “Teste é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente.” Conforme nosso entendimento, a ação sobre o produto consiste em colocar sob avaliação os passos adotados no estágio de confecção, atestando a eficiência e aplicabilidade do objeto ou sistema.

Assim, na etapa de se aplicar testes no software, alguns agentes entrarão em cena, quer sejam o gerente do projeto, os engenheiros envolvidos em cada etapa da configuração e sujeitos especializados em testagens – o analista

de teste ou testador de software. Eles trabalharão com o intuito de detecção de irregularidades, quer estejam na parte lógica de processamento ou na parte dos dados do software, abordando todos os níveis de constructo, projeto e/ou construção desde o início da tarefa.

Desse modo, passo a passo, planos e procedimentos serão pensados e encabeçados em vários níveis de desenvolvimento de testagens, para que ao término da empreitada a qualidade do produto se evidencie, que é o que interessa ao consumidor final em posse do software.

As testagens devem iniciar-se em nível individual com os testes de unidade. Segundo Pressman e Maxim (2021) “A complexidade relativa dos testes e os erros que os testes revelam são limitados pelo escopo restrito estabelecido para o teste de unidade.” Depois, passa-se para a parte de integração de unidades e adiante se analisará os testes no sistema em conformidade com o andamento da construção do software.

Existe uma enorme gama de testes que se aplicam no processo. Resumindo e esquematizando o contexto, apresenta-se um esquema sugerido por Pressman e Maxim (2021) “O teste de unidade começa no centro da espiral e se concentra em cada unidade [...] Por fim, chegamos ao *teste do sistema*, no qual o *software* e outros elementos são testados como um todo.”

A **Figura 2** esquematiza as ideias dos autores citados.

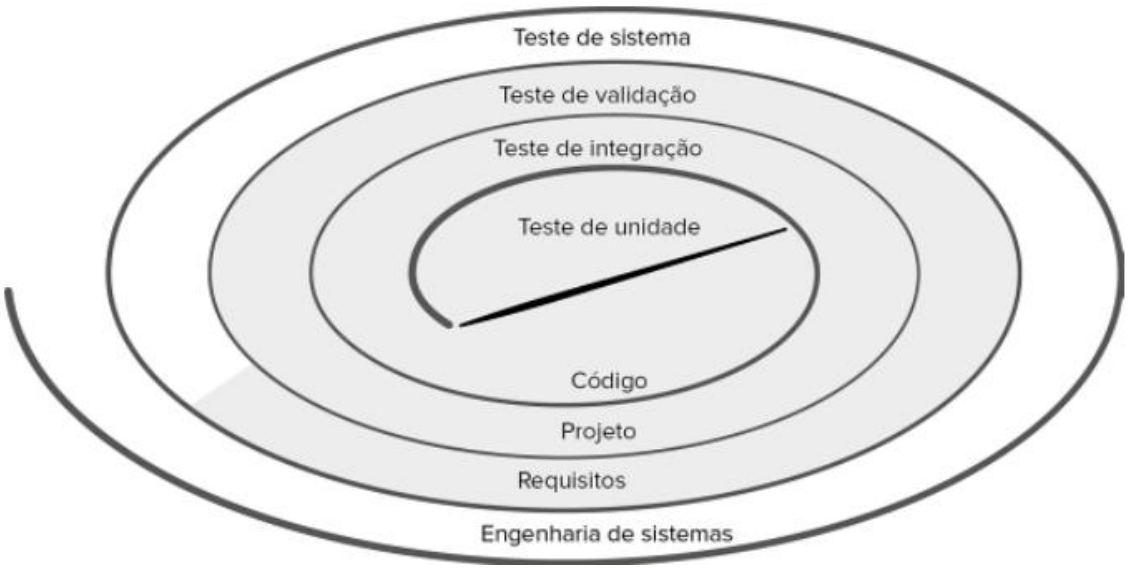


Figura 2 - Estratégias de Teste

Fonte: Pressman e Maxim - 2021.

Seguindo essa linha de raciocínio, se admite e entende que a engenharia de software fará uso de quatro estágios distintos, interligados e que estes seguem uma ordem contínua a saber: teste de unidade, teste de integração, teste de validação, teste de sistema.

No teste de unidade podem ser executados de maneira rápida e individualizada, e verificará erros de lógica em cada unidade de programação, buscando neste teste encontrar o maior número de incorreções.

No teste de integração se observará funções globais e parciais, localizando erros acomodados de interface entre os módulos do software. Neste teste, existe a necessidade de que os componentes do software possam ser executados e demonstrem atividade.

No teste de validação, neste ponto, serão analisados se os componentes do sistema atuam e funcionam em harmonia, de maneira conjunta, garantindo a funcionalidade e o desempenho almejados.

No teste de sistema possui um caráter de disposição superior. Nele o software constitui-se em elemento de um grande sistema. Este teste examina se existe combinação, operação e boa atuação das partes que compõem o sistema.

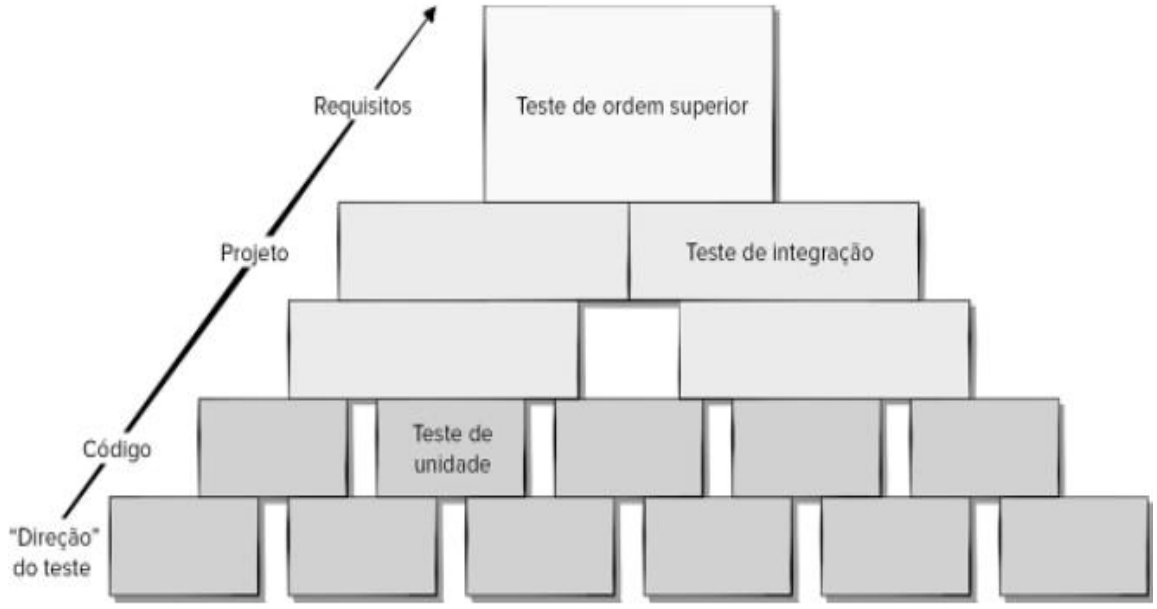


Figura 3 - Etapas de teste de Software

Fonte: Pressman e Maxim - 2021.

Desse modo, testando-se amostras e observando-se a métrica e estatísticas, obtém-se validação do software. Cabe ressaltar que considera-se o trabalho de testagens contínuo, mesmo quando o software passa para as mãos do cliente. Para Pressman e Maxim (2021) “Todas as vezes que o usuário executa o programa no computador, o programa está sendo testado. Esse fato destaca a importância de outras atividades de garantia da qualidade do software.”

Observa-se que ainda hoje existem técnicas de testagem, mais tradicionais, que foram usadas em sistemas com linguagens organizadas e que também tem por intuito localizar as falhas e, apesar de terem a forma tradicional, estão arranjadas de modo a ainda serem muito aplicáveis, contemplando as vantagens e os aspectos complementares dessas técnicas que se dividem em Funcional e Estrutural.

2.2.1 Teste Funcional (Teste de Caixa-Preta)

Também conhecido como Teste de Caixa-Preta, tem por finalidade verificar a saída dos dados utilizando entradas de diversos tipos, conforme Myers (1979). Assim, o teste de Caixa-Preta tem fundamentos a partir do referencial operacional do software e terá como base a parte externa ou as possíveis ações dos usuários finais. A partir dessa premissa, o especialista irá analisar as funções que o software deve cumprir. De conformidade com as entradas observadas no teste, poderá obter-se um vislumbre das prováveis saídas eminentes.

Ao fazer uso do Teste de Caixa-Preta, o analista de teste deve simular enganos comuns que usuários podem cometer e que não aparecem especificados no software. Se comparado a outras espécies de testes, ele pode ser caracterizado como simples e mostrar-se útil por detectar incorreções como omissões ou que funções inexatas, erros de comportamento/desempenho, de interface e de início e término.

De conformidade com Pressman e Maxim (2021) “As técnicas de teste caixa-preta permitem derivar séries de condições de entrada que utilizarão completamente todos os requisitos funcionais para um programa.”

O teste funcional tenta encontrar erros nas (1) funções incorretas ou ausentes; (2) erros de interface; (3) erros de estruturas de dados ou acesso a bases de dados externas; (4) erros de comportamento ou de desempenho e (5) erros de inicialização e término.

O teste de caixa-preta é diferente do teste de caixa-branca, o qual é executado antes no processo de teste. Por outro lado, o teste de caixa-preta inclina-se a ser aplicado durante estágios posteriores do teste.

Algumas perguntas precisam ser respondidas após os testes. São elas:

1. Como a validade funcional é testada?
2. Como o comportamento e o desempenho do sistema são testados?
3. Quais classes de entrada farão bons casos de teste?
4. O sistema é particularmente sensível a certos valores de entrada?
5. Como as fronteiras de uma classe de dados são isoladas?
6. Quais taxas e volumes de dados o sistema pode tolerar?
7. Combinações específicas de dados terão quais efeitos sobre a operação do sistema?

Esse tipo de teste, apesar de necessário, é insuficiente para identificar determinados riscos num projeto de software, pois torna-se impossível realizar testagem de todas as entradas, bem como o resultado das entradas específicas não serem as mesmas aceitas para o teste devido a ambiguidade em relação ao sistema produzido.

O ideal para o Teste da Caixa-Preta seria escolher um subconjunto de entradas que enriqueça o mesmo, agrupando subconjuntos de forma que a testagem unitária sirva para averiguar a qualidade de todo o subconjunto.

Vale lembrar que as diversas formas de testagem também possuem a sua aplicabilidade e modo de uso específicas a cada caso, sendo esta técnica de Teste Funcional aplicável a todos os níveis de teste.

Figura 4 - Teste de Caixa-Preta

2.2.2 Teste Estrutural (Caixa-Branca)

O teste estrutural, também conhecido como teste de caixa-branca ou até mesmo teste de caixa-de-vidro, é uma ideologia de projeto de casos de teste que usa a estrutura de controle descrita como parte do projeto em nível de componente para derivar casos de teste. O engenheiro de software pode criar casos de teste que garantem que todos os caminhos independentes em um módulo foram exercitados pelo menos uma vez. Exercitam todas as decisões lógicas em seu verdadeiro e falso e executam todos os ciclos dentro de seus limites e dentro de suas fronteiras (Pressman, R. S. e Maxim, B. R.). O teste possui esse nome pois o testador tem acesso à estrutura interna da aplicação, logo, seu foco é garantir que os componentes do software estejam concisos.

Proposta por Tom McCabe, o teste de caminho básico é uma técnica de teste caixa-branca. Tal teste permite ao projetista de casos de teste derivar uma medida da complexidade lógica de um projeto procedimental e utilizá-la como guia para definir um conjunto-base de caminhos de execução (Pressman, R. S. e Maxim, B. R). Essa técnica calcula a complexidade lógica do software e utiliza esta medida como base para descobrir os caminhos básicos do software, exercendo o teste de modo que todos os caminhos sejam efetuados.

Para que possa se usar a técnica de teste de caixa branca, o código fonte deve estar pronto, pois é através dele que se extrai o grafo de fluxo que representa a lógica do código fonte.

De acordo com Pressman, na construção do grafo de fluxo de controle,

existem representações simbólicas correspondentes do grafo de fluxo. Os círculos são os ramos, os quais demonstram uma ou várias linhas do código fonte, e as setas, representam as arestas, as quais mostram o caminho ou caminhos que o código pode fazer.

Pressman definiu o caminho independente como um caminho ao longo do código fonte que execute um novo comando, e no grafo de fluxo seria uma nova área que não foi exercida antes. Com tal ferramenta, é possível calcular um valor para os atributos do software e o cálculo de complexidade ciclomática (V(G)) para o grafo de fluxo. Conforme Pressman (2021) a V(G) pode ser calculada de três formas: 1. Pelos números de regiões do grafo de fluxo; 2. Pela fórmula V(G) = E - N + 2, onde V(G) é a variável de complexidade ciclomática, G é a representante do grafo de fluxo, E representa a quantidade de arestas no grafo e N a quantidade de ramos no grafo; 3. V(G) = P + 1, onde V(G) representa a variável de complexidade ciclomática; G representa o grafo de fluxo e P representa a quantidade de ramos predicativos. Através do resultado da complexidade ciclomática, temos o número de caminhos independentes do código.

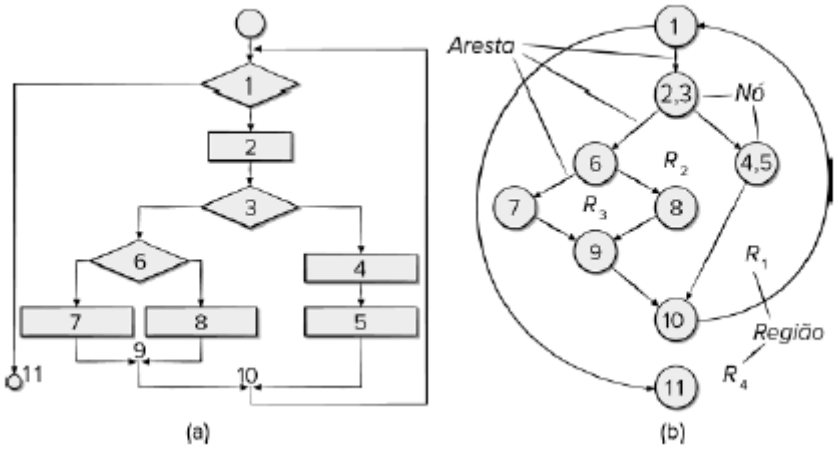


Figura 5 - (a) Fluxograma e (b) Grafo de Fluxo

Fonte: Roger-S.-Pressman_-Bruce-R.-Maxim-Engenharia-de-software

2.2.3 Teste Web

Com o passar do tempo a web (rede), vem sofrendo transformações, evoluções consecutivas e em processo de continuidade futura. Inicialmente sua aplicabilidade apresentava o formato de hipermídia fixa, se concentrando em documentos até chegar na aplicabilidade mais atual que gira em torno do conhecimento. Conforme Santa Isabel (2011), “O desenvolvimento de sistemas baseado em Web é uma atividade contínua e sem releases específicas como ocorre no desenvolvimento de software convencional.”

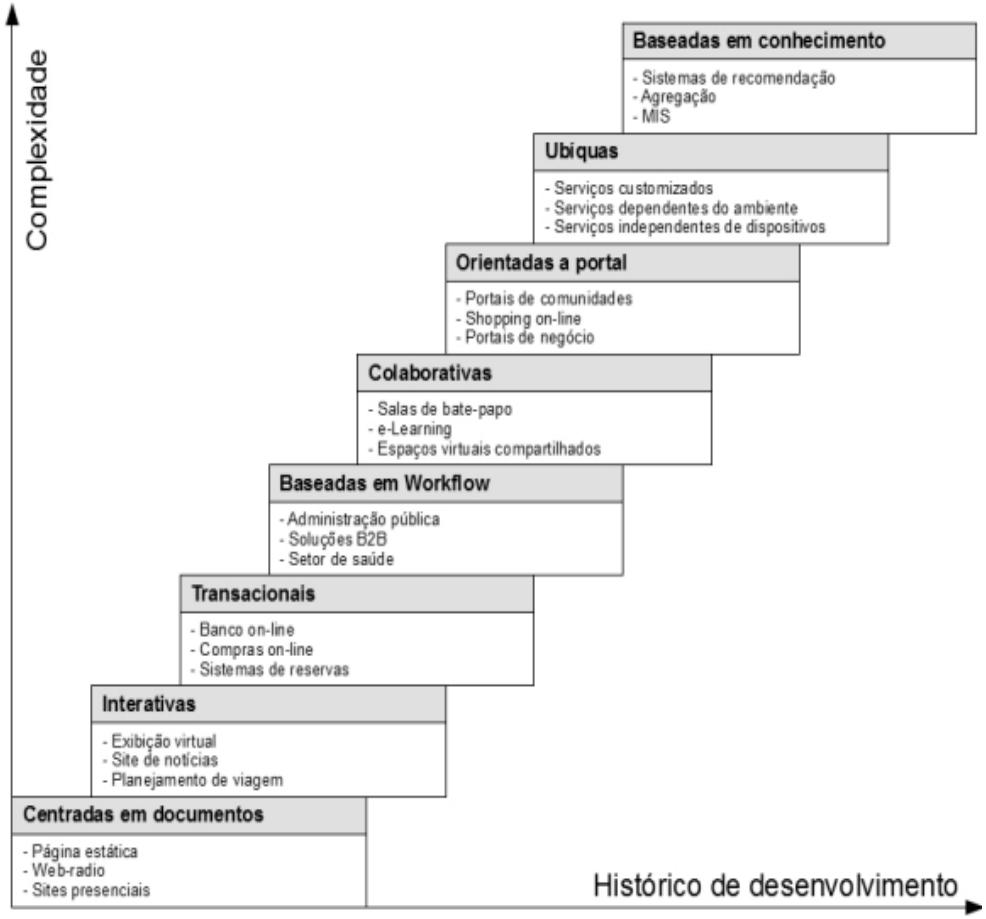


Figura 6 - Categorias de Aplicações Web (Adaptado de KAPPEL, 2004).

Fonte: Santa Isabel - 2011.

Para apurar a qualidade de softwares diante da complexidade de aplicações, aqui tratadas, para o Teste Web serão embasadas as características da funcionalidade do produto. A exemplo dos softwares convencionais, aqui

também se aplicarão o teste Funcional (Caixa-Preta) e o Estrutural (Caixa-Branca).

Na testagem, onde se utilizar a técnica funcional, poderá haver necessidade de adaptações específicas e tradicionais, devido ao comportamento dos componentes e ao montante das aplicações Web. Já no caso de uso da técnica estrutural, serão verificados as condições planejadas e acessos lógicos sobre o código fonte. Devido a complexidade no contexto serão consideradas também o fluxo de dados e controle.

De conformidade com os autores Ricca e Tonella (2002), dentro do teste estrutural se propõem mais quatro abordagens. Uma delas seria o Teste de todas as páginas (“all-pages”), onde a cada atividade realizada de teste deve-se retornar na página do aplicativo Web. Já no Teste de todos os caminhos (“all-paths”), os caminhos na Web deverão ser examinados ao menos uma vez, em um dos testes. Outro critério seria o Teste de todos os Hyperlinks (“all-links”). Nele o link da página deve ser averiguado nas testagens, no mínimo uma vez. Os autores também cogitam o uso do Teste de todos os usos (“all-use”), onde os caminhos de navegação passarão por investigação, testando prováveis variações, consolidando neste perfil uma configuração de cadeia de dados.

Admite-se também para controle da qualidade de softwares, o uso de uma combinação de testagens funcional e estrutural que geram o Teste da Caixa-Cinza. Esse teste irá concentrar-se nas dificuldades de apresentação, quantidade de informação e combinação de softwares, conforme Di Lucca e Fasolino, (2006).

Devemos levar em consideração que as páginas da Web podem ser denominadas de estáticas ou dinâmicas, e que para cada estilo exigirá um tipo de testagem conforme especificidade contida para seu controle de qualidade.

2.3 QUALIDADE DE SOFTWARE

O conceito de qualidade pode parecer algo simples e intuitivo. Todavia, após uma análise mais profunda, é perceptível a complexidade para definir tal conceito subjetivo. Qualidade é tratada no Dicionário Brasileiro da Língua Portuguesa Michaelis como: “Atributo, condição natural, propriedade pela qual

algo ou alguém se individualiza; maneira de ser, essência, natureza". De acordo com a ISO/IEC 9126-1:2003, qualidade é a totalidade de características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.

Na ocasião em que tratamos produto de software, qualidade é definida como uma área pertencente à engenharia de software, que possui como objetivo a garantia de produtos adequados a partir de processos eficientes. Portanto, existem dois aspectos que se referem a qualidade, sendo eles: qualidade do produto e a qualidade do processo (Raul Wazlawick).

Processo é um conjunto de atividades, ações e resultados realizados na criação de algum produto de software (Ian Sommerville). O produto de software, por sua vez, é objeto que compreende as etapas de documentação, os procedimentos do computador e os dados associados, os quais foram desenvolvidos para o usuário (Ian Sommerville).

De acordo com Pressman, o produto de software possui dois tipos de qualidade e são eles: a qualidade do projeto e a qualidade de conformidade. A qualidade do projeto se refere às características dadas pelos projetistas para especificar um produto, e a qualidade de conformidade possui foco no grau em que a implementação segue o projeto e que o sistema resultante atende às suas necessidades e às metas de desempenho.

2.4 GUIA DE QUALIDADE DE SOFTWARE–MEDE-PROS

Desenvolvido pelo Instituto Nacional de Tecnologia da Informação - ITI, o MEDE-PROS é um guia de qualidade de software baseado na ISO/IEC 25010. Foi criado com o objetivo de avaliar a qualidade de produto de software. O guia apoia a atividade de avaliação de qualidade de software, fornecendo ao avaliador uma série de questões que, quando respondidas, permitirão uma análise do produto do ponto de vista do usuário final. O modelo considera o conjunto de elementos que compõem o software e as características de qualidade associadas a estes elementos.

As questões fornecidas pelo guia abrangem as atividades de avaliação da instalação, documentação, interface, software, descrição do produto, embalagem e desinstalação. Cada aspecto desses é avaliado pelas características de qualidade de software:

1. Completude
2. Funcionalidade
3. Usabilidade
4. Confiabilidade
5. Portabilidade

As características de qualidade de software são, por sua vez, subdivididas em subcaracterísticas:

1. Especificações dos requisitos de hardware e software
2. Adequação
3. Identificações e indicações
4. Inteligibilidade
5. Apreensibilidade
6. Operacionalidade
7. Acurácia
8. Acesso seletivo
9. Maturidade
10. Tolerância à falhas
11. Declarações sobre usabilidade
12. Capacidade de desinstalação

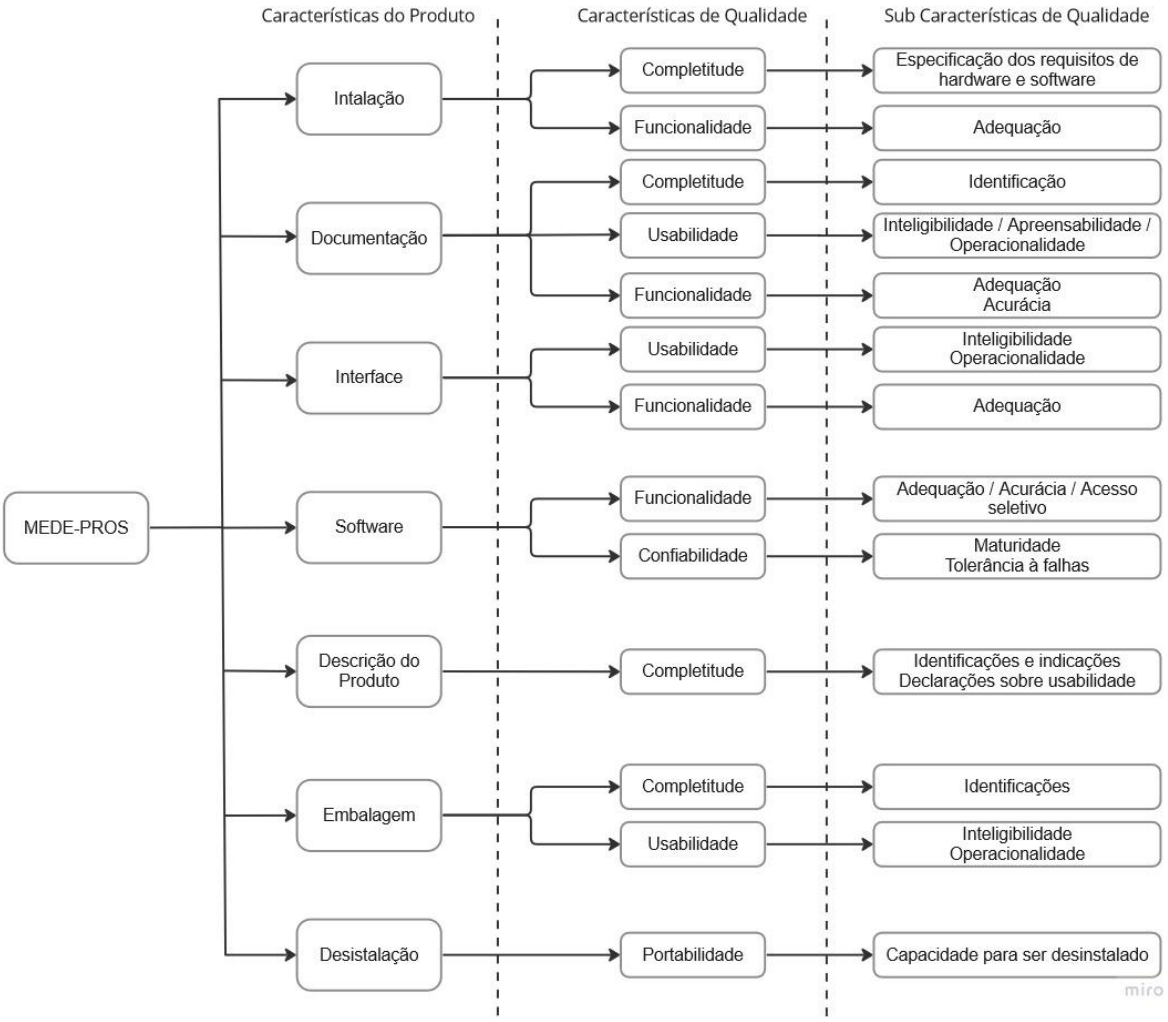
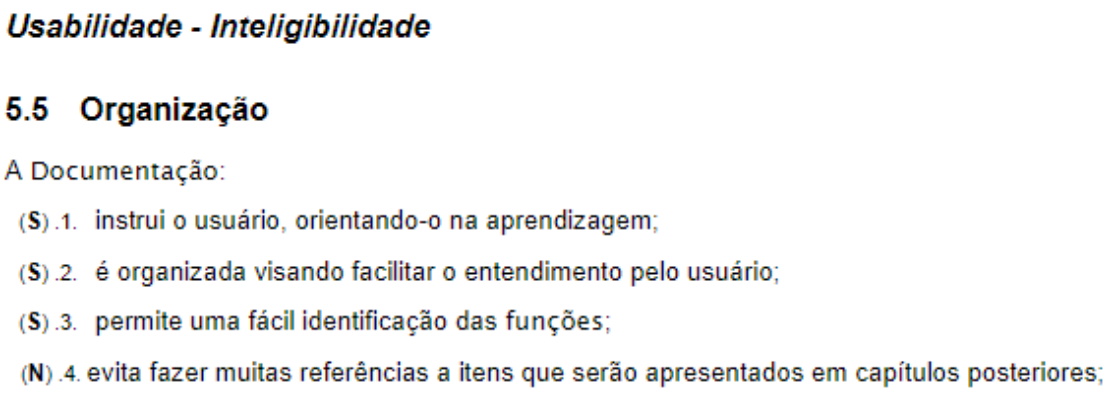


Figura 7 - Estrutura geral do guia MEDE-PROS

As questões que o MEDE-PROS aborda, são questões lógicas sobre um atributo que necessita ser verificado na avaliação e devem ser objetivas nas respostas. A avaliação segue um modelo de proposições lógicas sobre atributos, que devem ser respondidas com as possíveis respostas:

1. S (Sim) para sentenças verdadeiras.
2. N (Não) para sentenças falsas.
3. NA (Não se Aplica) para sentenças que fazem referência a aspectos que não se enquadram no produto avaliado.
4. AP (Avaliação Prejudicada) para questões que o avaliador não está em condições de avaliar, isto é, por falta de recursos, informações ou até mesmo conhecimento específico para a atividade.

Para respostas desfavoráveis ou marcadas como NA ou AP, o avaliador deve incluir uma justificativa para a resposta dada. Erros ou falhas no software também devem ser documentados e possuem uma seção própria para isso no guia.



Usabilidade - Inteligibilidade

5.5 Organização

A Documentação:

(**S**) .1. instrui o usuário, orientando-o na aprendizagem;

(**S**) .2. é organizada visando facilitar o entendimento pelo usuário;

(**S**) .3. permite uma fácil identificação das funções;

(**N**) .4. evita fazer muitas referências a itens que serão apresentados em capítulos posteriores;

Figura 8 - Exemplo das questões preenchidas por parte do avaliador

2.5 ISO 25010

Usada como base para o desenvolvimento do MEDE-PROS, a ISO 25010 é a norma que define modelos de qualidade de software, isto é, conjuntos de características e subcaracterísticas de qualidade que servem de guia e permitem a avaliação de um produto de software.

Dentre as características existentes na norma, foram utilizadas para o MEDE-PROS as seguintes:

1. Adequação Funcional

Diz respeito ao quão bem o produto de software fornece suas funcionalidades e se estas atendem a necessidade do usuário. Esta característica pode ser subdivida em completude (se o produto apresenta todas as funcionalidades especificadas), acurácia (se o produto fornece resultados corretos

e com precisão) e adequação (o quão bem as funcionalidades são capazes de realizar as tarefas e objetivos especificados).

2. Usabilidade

Se refere ao quão bem o produto pode ser usado de maneira satisfatória. Através desta característica é possível observar a facilidade com que o usuário aprende a utilizar o produto avaliado, o quão complexa é a sua utilização, se protege e notifica o usuário de erros, se é visualmente agradável e acessível à pessoas de diferentes características e capacidades.

3. Confiabilidade

A confiabilidade é uma característica que permite ao avaliador verificar se um produto atende às necessidades do usuário. Permite avaliar a disponibilidade do sistema, se é operacional e acessível e, em caso de falhas, qual o seu comportamento e tolerância para tal. Além do quão bem este sistema pode ser recuperado em caso de falha.

4. Portabilidade

Diz respeito ao quão bem o produto de software pode ser migrado para diferentes plataformas. Se este é adaptável para diferentes tipos de software e hardware. Qual a facilidade de instalá-lo ou desinstalá-lo com sucesso e se o produto pode ser utilizado no lugar de outros softwares semelhantes.

3 PROTOCOLO DE MAPEAMENTO SISTEMÁTICO PARA BUSCA DE APLICATIVOS EM REPOSITÓRIOS NÃO-ACADÊMICOS

3.1 Introdução

Os Mapeamentos Sistemáticos, ou MSL, são estudos utilizados para auxiliar pesquisas em determinado tópico. São realizados com metodologias contendo normas e passos para garantir a confiabilidade e documentação dos processos de busca de trabalhos (Lima, L. F. e Peres L. M.).

Os MSL são estruturados em três etapas: Planejamento, Condução e Publicação. Passos que envolvem a definição do objetivo do Mapeamento Sistemático e estratégia de buscas, realização e estratégia de seleção dos resultados de busca, extração e análise de dados e por fim a publicação dos resultados (Lima, L. F. e Peres L. M.).

O Protocolo de Mapeamento Sistemático utilizado para o desenvolvimento deste relatório (Lima, L. F. e Peres L. M.) é uma adaptação do protocolo de MSL de Petersen, K., et al. (2015). Suas alterações favorecem a busca de ferramentas de software no contexto de desenvolvimento, uma vez que sugere a utilização de repositórios de código-aberto e adapta as buscas e seleções para o contexto de software.

3.2 Etapa de Planejamento

Nesta etapa são definidos os objetivos da utilização do MSL, as questões de pesquisa a serem respondidas ao final do processo e quais repositórios de código-fonte serão utilizados para tal. A definição do objetivo do mapeamento sistemático realizado está documentado na **Tabela 1**. As questões de pesquisa, isto é, o que será respondido ao final do uso do Mapeamento Sistemático, está documentado na **Tabela 2**.

Tabela 1 - Objetivo do Mapeamento

Objetivo do Mapeamento	
Analisar	Ferramentas de teste de software web
Com o propósito de	Identificar, documentar e avaliar
Em relação a	Guia de qualidade de software MEDE-PROS
Do ponto de vista do	Desenvolvedor
No contexto	Acadêmico

Tabela 2 - Questão de Pesquisa

Questão de Pesquisa	
QP Principal	As ferramentas atendem as diretrizes de qualidade propostas pelo guia de qualidade de software MEDE-PROS?

Subquestões de Pesquisa	
Subquestão	**Descrição**
SQ1	Completude da descrição do Produto
SQ2	Funcionalidade - Adequação
SQ3	Avaliação da Documentação

SQ4	Usabilidade
SQ5	Funcionalidade
SQ6	Interface

Definidos os objetivos e as questões de pesquisa, a próxima etapa do mapeamento sistemático é definir as estratégias de busca. Este processo está documentado nas **Tabelas 3, 4 e 5**. O repositório de código fonte utilizado para as buscas, no caso o github, é especificado na **Tabela 3**. O dispositivo utilizado para a condução do MSL está especificado na **Tabela 5**.

Em uma primeira versão do mapeamento, observou-se que as palavras-chave escolhidas e usadas individualmente para busca, ofereciam opções de ferramentas na casa de centenas de milhares ou milhões, inviabilizando a utilização do protocolo a partir do passo de filtragem de resultados. Como alternativa e permitido pelo sistema de buscas do github, optou-se por concatenar as palavras-chave, diminuindo assim, o tamanho dos resultados de busca. Essa alternativa foi utilizada na versão final da aplicação do protocolo MSL e está especificado na **Tabela 4**.

Tabela 3 - Repositórios Utilizados Versão Final

Repositórios	
Nome	**Link**
Github	https://github.com

Tabela 4 - Palavras-Chave

Palavras-Chave	
Palavra-chave 1	Testing-tools

Palavra-chave 2	Test-automation
Palavra-chave 3	Web
Palavra-chave 4	Functional-testing
Palavra-chave 5	Security-testing
Palavra-chave 6	Web-testing

Tabela 5 - Dispositivo Utilizado para a busca

Dispositivo de Busca	
Modelo	Notebook
SO	Windows 10
Memória	16GB
Processador	Intel™ Core™ i7-9750H CPU @ 2.60GHz 2.59 GHz

3.3 Etapa de condução

Esta etapa envolve a elaboração dos critérios de inclusão e exclusão das ferramentas encontradas (**Tabela 6**). Realizam-se as buscas nas bases de dados (quantidade de resultados especificado na **Tabela 7**) escolhidas, além das seleções desses softwares e filtragem dos resultados com base nos critérios de inclusão e exclusão (**Tabela 8**). Na condução também realiza-se o processo de snowballing (**Tabela 9**), que consiste na seleção de ferramentas que inicialmente não se encontravam nos resultados de busca, porém, são referenciadas pela seleção inicial.

Tabela 6 - Critérios de Inclusão e Exclusão

Critérios de Inclusão	
CI1	Ferramentas usadas em desenvolvimento web
CI2	Ferramenta de teste
CI3	Possui documentação
Critérios de Exclusão	
CE1	Sem atualizações nos últimos 1,5 anos
CE2	Não atende aos CIs
CE3	Ferramentas que não estejam em inglês, português e espanhol

Tabela 7 - Resultado das Buscas usando Github

Resultados das Buscas – Github	
Palavra-chave	**Aplicativos retornados**
Testing-tools	25.728
Test-automation	22.062
Web	3.823.517
Functional-testing	12.449
Web-testing	39.904

web-testing functional-testing test-automation web testing-tools	22
web-testing security-testing test-automation web testing-tools	6
web-testing security-testing test-automation web	16

Tabela 8 - Filtro de Seleção Final

Aplicativos da amostra inicial	**Critério de Inclusão**	**Critério de Exclusão**	**Motivo da exclusão**
https://github.com/DevExpress/testcafe	CI1, CI2, CI3		
https://github.com/testsigmahq/testsigma	CI1, CI2, CI3		
https://github.com/mkrishna4u/smart-testauto-framework	CI1, CI2, CI3		
https://github.com/impyrosis/aos	CI1, CI2, CI3		Avaliação de Qualidade
https://github.com/blessingcharles/bountycat	CI1, CI2, CI3		
https://github.com/eaccmk/Test-Buffet	O repositório possui uma lista de ferramentas de teste que		

Aplicativos da amostra inicial	Critério de Inclusão	Critério de Exclusão	Motivo da exclusão
	podem ser utilizadas		
https://github.com/Karmaz95/crimson	CI1, CI2, CI3		

Tabela 9 - Processo de Snowballing

Processo de Snowballing		
Aplicativo 1	**Aplicativo 2**	**Critério de Inclusão**
https://github.com/mkrishna4u/smart-testauto-framework	https://www.selenium.dev/	CI1, CI2, CI3

Por fim, são definidos itens de qualidade que quando aplicados a seleção de ferramentas, também podem filtrar os resultados obtidos (**Tabela 10**). A realização dessa última filtragem é demonstrada na **Tabela 11**.

As **Tabelas 12, 13, 14 e 15** listam a filtragem final das ferramentas pesquisadas, totalizando 4 ferramentas de teste na web.

Tabela 10 - Itens de Qualidade

Itens de Qualidade	
Item 1	Existe documentação com instruções

	para compilação?
Item 2	Existe documentação com instruções para execução?

Tabela 11 - Avaliação de Qualidade

Avaliação de Qualidade	
Aplicativo da amostra final preliminar	**Item de qualidade usado para a exclusão**
https://github.com/impyrosis/aos	Item 1 e Item 2

Tabela 12 - TestCafe

Tipo do dado	**ID do dado**	**Descrição do dado**	**Dado extraído**
Dados do repositório	DR1	Nome do Aplicativo	TestCafé
	DR2	Descrição do Aplicativo	Uma aplicação em node.js para automatizar testes web end-to-end
	DR3	Desenvolvedor do Aplicativo	Developer Express Inc.
	DR4	Linguagem de implementação	Javascript TypeScript HTML
	DR5	Data de inclusão	Fevereiro de 2016
	DR6	Última versão	V1.18.6

Tipo do dado	ID do dado	Descrição do dado	Dado extraído
	DR7	Requisitos/recursos para uso	Node.js e npm instalados no computador
	DR8	Disponível integralmente?	Sim
	DR9	Instruções de instalação/compilação	https://testcafe.io/documentation/402635/getting-started#installing-testcafe
	DR10	Instruções para uso	https://testcafe.io/documentation/402635/getting-started#creating-a-test
Dados das subquestões de pesquisa	QP1	A ferramenta oferece artifícios para testes de segurança?	Não
	QP2	A ferramenta oferece artifícios para testes funcionais?	Sim

Tabela 13 - TestSigma

Tipo do dado	ID do dado	Descrição do dado	Dado extraído
Dados do repositório	DR1	Nome do Aplicativo	testsigma

Tipo do dado	ID do dado	Descrição do dado	Dado extraído
	DR2	Descrição do Aplicativo	Plataforma open-source para testes automatizados web.
	DR3	Desenvolvedor do Aplicativo	Testsigma Technologies Inc
	DR4	Linguagem de implementação	Java TypeScript HTML SCSS
	DR5	Data de inclusão	Janeiro de 2022
	DR6	Última versão	V1.4.0
	DR7	Requisitos/recursos para uso	“Docker” ou “MySQL, Nginx, Testsigma server e Testsigma agente”
	DR8	Disponível integralmente?	Sim
	DR9	Instruções de instalação/compilação	https://testsigma.com/docs/getting-started/setup/docker/ ou https://testsigma.com/docs/getting-started/setup/downloadable-package/

Tipo do dado	ID do dado	Descrição do dado	Dado extraído
	DR10	Instruções para uso	https://testsigma.com/tutorials/getting-started/automate-web-applications/
Dados das subquestões de pesquisa	QP1	A ferramenta oferece artifícios para testes de segurança?	Não
	QP2	A ferramenta oferece artifícios para testes funcionais?	Sim

Tabela 14 - Smart TestAuto Framework

Tipo do dado	ID do dado	Descrição do dado	Dado extraído
Dados do repositório	DR1	Nome do Aplicativo	Smart Software Testing Automation Framework (smart-testauto-framework)
	DR2	Descrição do Aplicativo	Um framework para automatização de testes que pode ser utilizado com qualquer tipo de aplicação
	DR3	Desenvolvedor do Aplicativo	Madhav Krishna

Tipo do dado	ID do dado	Descrição do dado	Dado extraído
	DR4	Linguagem de implementação	Java
	DR5	Data de inclusão	Dezembro de 2021
	DR6	Última versão	05 de Maio de 2022
	DR7	Requisitos/recur-sos para uso	TestNG, Cucumber, Selenium, Sikulix, Hibernate, Appium
	DR8	Disponível integralmente?	Sim
	DR9	Instruções de instalação/compila-ção	https://github.com/mkrishna4u/smart-testauto-framework/wiki/How-to-setup-%3F
	DR10	Instruções para uso	https://github.com/mkrishna4u/smart-testauto-framework/wiki/Quickstart-Guide-for-Testers
Dados das subquestões de pesquisa	QP1	A ferramenta oferece artifícios para testes de segurança?	Não
	QP2	A ferramenta oferece artifícios para testes funcionais?	Sim

Tabela 15 - Selenium

Tipo do dado	ID do dado	Descrição do dado	Dado extraído
Dados do repositório	DR1	Nome do Aplicativo	Selenium
	DR2	Descrição do Aplicativo	Ferramenta que automatiza a utilizações de web browsers
	DR3	Desenvolvedor do Aplicativo	Jason Huggins
	DR4	Linguagem de implementação	Java C# Javascript C++ HTML Ruby
	DR5	Data de inclusão	2004
	DR6	Última versão	Novembro 2021
	DR7	Requisitos/recur-sos para uso	Bazelisk Java 11 OpenJDK Python 3.7+ Ruby 3+ Visual Studio

Tipo do dado	ID do dado	Descrição do dado	Dado extraído
	DR8	Disponível integralmente?	Sim
	DR9	Instruções de instalação/compi-lação	https://www.selenium.dev/documentation/
	DR10	Instruções para uso	https://www.selenium.dev/documentation/
Dados das subquestões de pesquisa	QP1	A ferramenta oferece artifícios para testes de segurança?	Não
	QP2	A ferramenta oferece artifícios para testes funcionais?	Sim

4 FERRAMENTAS PARA TESTE NA WEB – SELEÇÃO FINAL

4.1 TestCafe (Tabela 12)

TestCafe é uma estrutura de teste end-to-end voltada a aplicativos da Web. Suportando os três principais sistemas operacionais - Linux, Windows e macOS, é possível simular cenários de usuários comuns nos principais navegadores de desktop, navegadores de nuvem e em dispositivos móveis, não precisando de plug-ins de navegador. A estrutura de teste cuida de todas as etapas: iniciar navegadores, executar testes, coletar resultados de testes e gerar relatórios. Caso o site a ser avaliado funcione mal, os relatórios gerados pelo TestCafe podem ajudar a diagnosticar e solucionar o problema.

Executado em Node.js, o TestCafe requer a última versão da plataforma, sendo assim, a melhor forma de instalar a estrutura é via npm. O repositório npm hospeda compilações prontas para uso da estrutura. TestCafe permite que o usuário possa trabalhar por meio do Node para levantar os navegadores, ou seja, não é necessário ocupar WebDrives, diferente do Selenium.

4.2 Selenium (Tabela 15)

Selenium é um framework portátil para testar aplicativos da web. Ele é um projeto para uma variedade de ferramentas e bibliotecas que permitem e suportam a automação de navegadores da Web. O Selenium fornece ao usuário uma ferramenta de reprodução para a criação de testes funcionais sem a necessidade de aprender uma linguagem de script de teste, a qual é conhecida como Selenium IDE. O framework fornece extensões para emular a interação do usuário com os navegadores, um servidor de distribuição para dimensionar a alocação do navegador e a infraestrutura para implementações da especificação WebDriver do W3C, o que permite escrever código intercambiável para todos os

principais navegadores da Web. O Ecossistema do Selenium é composto por: Selenium IDE, Selenium WebDriver e Selenium Grid.

4.3 Test Sigma (Tabela 13)

TestSigma oferece um framework com diversas funcionalidades que abstraem a necessidade do conhecimento em programação por parte do usuário. A automação dos testes é feita em uma interface sem código, na qual o usuário precisa apenas ter conhecimento das funcionalidades da aplicação a ser testada. A aplicação oferece testes da Interface do Usuário, APIs, bases de dados e outros. A partir de dados recolhidos das aplicações testadas e dos testes elaborados, o TestSigma também oferece a automação da atualização dos testes caso a aplicação a ser testada sofra alterações.

4.4 Smart TestAuto (Tabela 14)

O Smart TestAuto Studio é uma ferramenta de automação de teste de software de estrutura que pode ser usada para testar de forma inteligente qualquer tipo de software aplicativo. Ele suporta a estratégia NO CODE LOW CODE para tornar as definições de etapas mais fáceis de manter e padronizar a maneira de configurar os dados do aplicativo. Esta ferramenta suporta ambiente real para automação de testes de software.

5 AVALIAÇÃO DE QUALIDADE DAS FERRAMENTAS SELECIONADAS

Após a conclusão da leitura e análise da documentação dos softwares testados, foi realizada uma avaliação com auxílio do guia de qualidade de software MEDE-PROS, dos itens referentes à documentação geral do software.

A primeira etapa consistiu em uma leitura da documentação e do guia MEDE-PROS. Após o preenchimento da seção “Identificação da Avaliação” e do item 4 “Avaliação da Instalação”, o qual possui como objetivo principal avaliar a qualidade de instalação para se colocar o produto em uso, foi preenchido o item 5 “Avaliação da documentação, interface e software”.

No quinto item do guia, o usuário pode avaliar a parte de documentação do software, a qual detém diversos tópicos que tratam da efetividade do documento em geral, como por exemplo: a organização, clareza, coerência e consistência interna e externa. Neste item, é possível também realizar uma análise da interface do software e sua completude com relação a sua funcionalidade e atendimento aos requisitos. Para esta etapa, as funcionalidades dos softwares de teste utilizados foram verificadas em duas páginas web fornecidas na documentação do TestCafe (**Figura 9**) e Selenium (**Figura 10**). As páginas utilizadas possuem os elementos básicos que constituem as diversas páginas da web, isto é, elementos com texto, inputs de clique do mouse e entradas de teclado, botões de redirecionamento e validação de formulários. Através delas foi possível verificar se os softwares testados conseguem interagir com esses elementos, atendendo a necessidade de um desenvolvedor web. No item 8 “Avaliação da desinstalação”, o usuário se depara com itens referentes a instruções de desinstalação do produto de software.

A seguir estão relatadas as avaliações gerais de cada um dos softwares selecionados a partir da utilização do MEDE-PROS. Nos Apêndices A, B, C e D está documentada a utilização do guia de qualidade pelas características consideradas mais relevantes a avaliação.

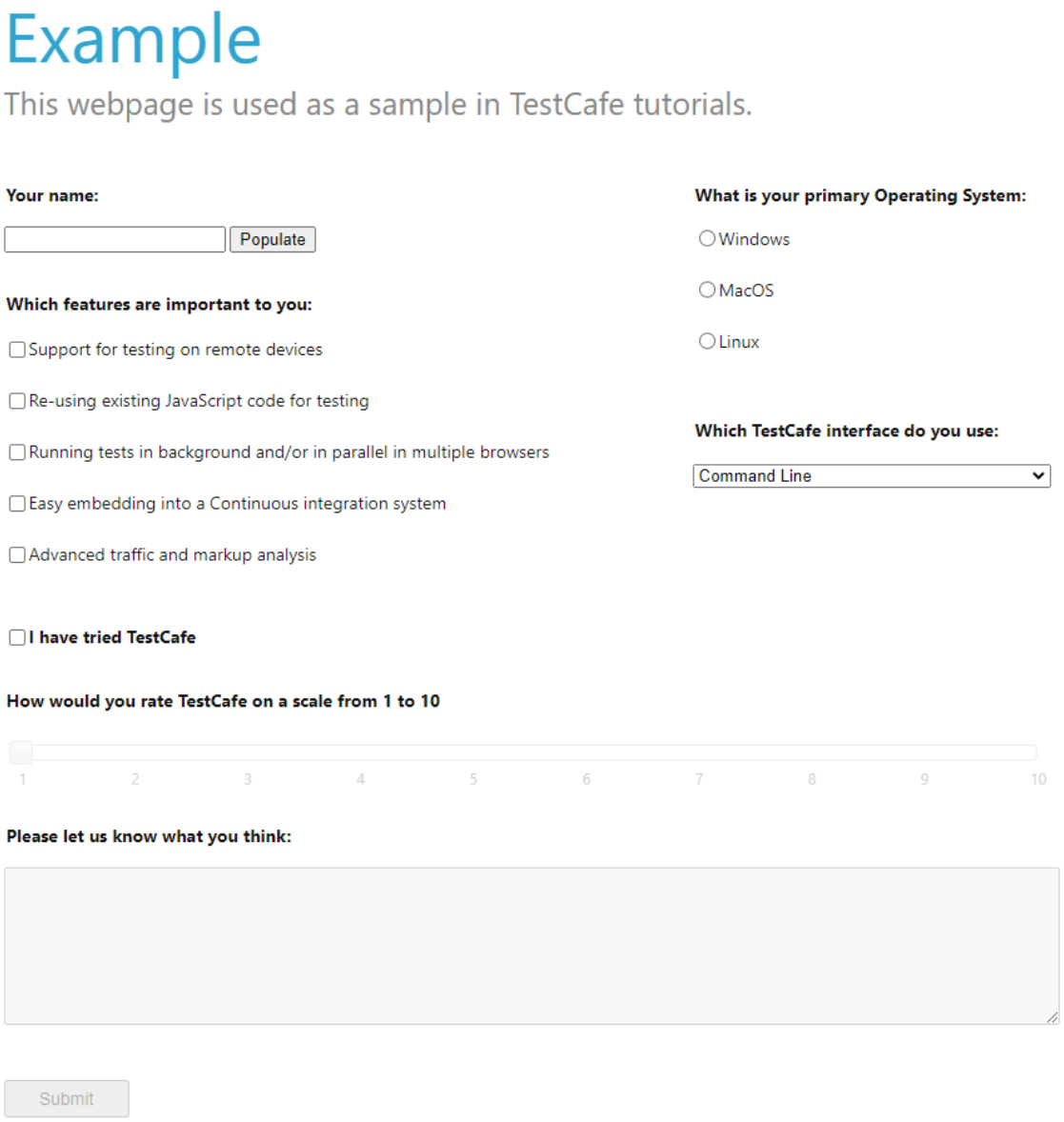


Figura 9 - Página utilizada para avaliação de qualidade. Encontrado em: *https://devexpress.github.io/testcafe/example/*

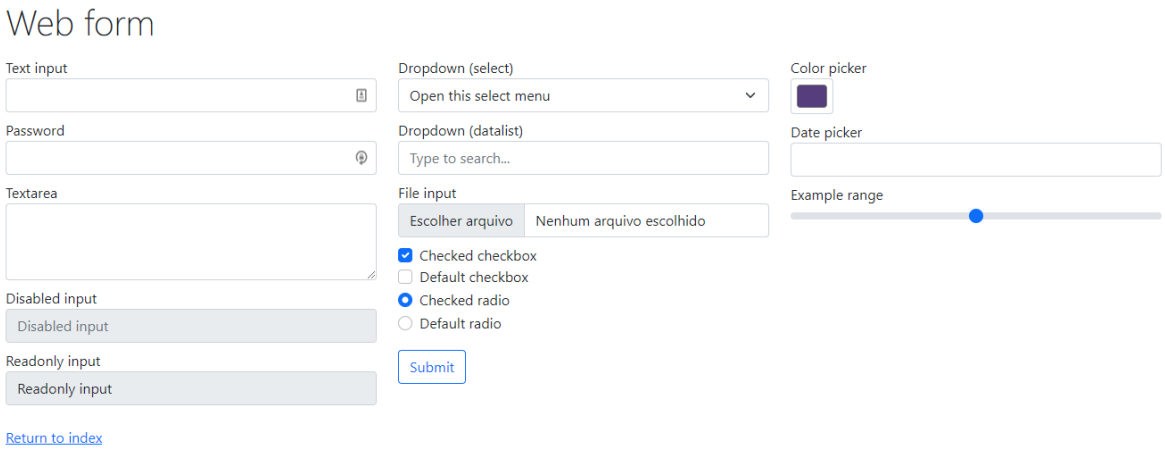


Figura 10 - Página utilizada para avaliação de qualidade. Encontrado em: *https://www.selenium.dev/selenium/web/web-form.html*

5.1 TestCafe

5.1.1 Avaliação da Instalação

De acordo com a documentação fornecida pela equipe do TestCafe, o usuário deve possuir uma versão recente de Node.js. Isto porque, para a instalação do software, utiliza-se o npm, um popular gerenciador de pacotes de Node.js.

Seguindo as instruções e as regras de pré-requisito, a instalação mostrou-se muito simples. Utilizando o comando “npm install testcafe”, o gerenciador de pacotes faz toda a configuração do software de maneira rápida e automática e em poucos segundos é possível utilizá-lo.

5.1.2 Avaliação da Documentação, Interface e Software

A documentação do produto é escrita em inglês e é dividida em 7 seções, as quais são divididas em:

1. Guias Básicos

   São apresentados guias com o passo a passo da instalação, da criação de scripts de teste e sua utilização e um resumo das funcionalidades principais do produto.
2. Boas Práticas

Aqui é documentado erros comuns dos utilizadores do produto e o que deve ser feito para mitigar esses erros. Mostra-se como a estrutura de testes deve ser feita, entre outros.

3. Guias Intermediários

   Esta seção desenvolve a funcionalidade de testes em diferentes browsers, testes de API e requisições HTTP.

4. Guias Avançados

   O TestCafe permite a criação de ações de teste customizadas (além das fornecidas nos Guias Básicos). Esta seção demonstra como tais ações podem ser criadas, além de demonstrar interações mais complexas com o navegador (como no disparo de eventos de interação nas páginas web).

5. Capacidades Experimentais

   Como é um produto em desenvolvimento, algumas funcionalidades do TestCafe são citadas e explicadas nessa seção.

6. Integração com outras plataformas

   É possível integrar o TestCafe com outras plataformas de desenvolvimento, como por exemplo o Azure DevOps e assim automatizar a execução de testes em produtos em desenvolvimento.

7. Extensões

   É possível criar extensões e plugins utilizando o TestCafe como base, e nesta seção é possível encontrar uma lista de extensões existentes ou como desenvolver a sua própria.

Foi possível utilizar o produto exatamente da maneira descrita na documentação, e o teste para verificar se a aplicação consegue interagir com páginas web foi concluído com sucesso. O TestCafe conseguiu realizar o acesso das URLs fornecidas (as mesmas utilizadas nas **Figuras 9 e 10**) e interagiu com todos os inputs da tela carregada, completou o formulário para validação de campos e verificou se o output esperado para as ações executadas foi correto.

Toda interação do usuário com o TestCafe é feita através da criação de scripts em Javascript ou Typescript e na sua execução através de linhas de comando. Por isso não foi identificada uma interface além desta descrita.

5.1.3 Avaliação da Desinstalação

Apesar da documentação não especificar como desinstalar a ferramenta, esta etapa é realizada da mesma maneira que a instalação. O TestCafe utiliza do gerenciador de pacotes npm para a sua desinstalação. O comando “npm uninstall testcafe” realiza o processo e o gerenciador de pacotes apaga todos os arquivos que compõem o TestCafe.

5.2 Selenium

5.2.1 Instalação

A página de instalação do software conta com as últimas versões dos componentes do Selenium, junto de uma documentação exclusiva do Selenium Server, que permite a execução de scripts WebDriver em máquinas remotas, roteando comandos enviados pelo cliente para instâncias remotas do navegador. Tais componentes são: o próprio Selenium Server, The Internet Explorer Driver Server, o qual é necessário caso o usuário deseje usar os mais novos recursos do Internet Explorer Driver e o Selenium IDE, o qual é um plug-in dos principais navegadores Web, tais como Chrome, Firefox e Edge. Tal IDE faz com que seja possível registrar e reproduzir as interações do usuário com o navegador, sendo possível através dela a criação de scripts simples ou auxiliar em testes exploratórios.

A parte de instalação por si não é extremamente completa e explicativa, devido ao motivo de que, para instalar o framework, é necessário seguir os passos que o seu sistema operacional solicita, ou seja, é algo que se torna intuitivo após clicar no link para download. Tal sessão oferece ao usuário uma vasta opção de linguagem WebDriver que o Selenium suporta, como por exemplo Java, C Sharp, Python, Ruby e JavaScript. Também é disponibilizado ao usuário os navegadores, junto dos principais sistemas operacionais suportados pela plataforma.

5.2.2 Avaliação da Documentação, Interface e Software

Disponibilizada em sua página web, a documentação do Selenium é bem detalhada e intuitiva, não deixando dúvidas para um usuário de primeira viagem. Possui muitas referências e direciona o usuário para explicações mais detalhadas e fornece trechos de código como exemplo. O texto explica e mostra como o software funciona de uma maneira clara e objetiva.

Para um desenvolvedor, a utilização do software pode ser feita através de scripts em diversas linguagens de programação como Java, Python, C#, JavaScript entre outros. A partir de um script em Java, foi possível avaliar todas as funcionalidades necessárias para a utilização das páginas especificadas pelas **figuras 9 e 10**. A ferramenta selecionou com sucesso os diferentes inputs da página, interagiu com estes simulando cliques de mouse, inputs de teclado e arquivos e verificou a validação de forms na web, permitindo assim um teste completo das funcionalidades essenciais de uma página.

O mesmo resultado pode ser obtido utilizando o IDE do Selenium, que abstrai e simplifica a prática de codificação. O IDE possui uma interface visual, direcionando o usuário em suas ações.

5.2.3 Desinstalação

A página do Selenium não conta com uma sessão de desinstalação do software. O usuário por si necessita descobrir como desinstalar o framework de sua máquina. Para usuários que possuem experiência com computadores, não é algo complexo, todavia, usuários inexperientes podem vivenciar algum tipo de dificuldade caso necessite remover o Selenium de sua máquina. Para os sistemas operacionais mais conhecidos no mercado como Linux e Windows, a desinstalação vem através de comandos e do painel de controle. Um exemplo utilizando a linguagem Python, o usuário digita no terminal, sudo apt-get remove --auto-remove python-selenium. Para Linux e para Windows é necessário entrar no painel de controle e desinstalar o programa.

5.3 Test Sigma

5.3.1 Instalação

A instalação da aplicação foi realizada de duas maneiras, uma seguindo a documentação de instalação por Docker e outra fazendo o download do pacote completo do software.

A instalação por Docker, que supostamente exigiria menos esforço de configuração por parte do usuário, foi feita exatamente como especificada pela documentação, porém não foi possível iniciar a ferramenta por esse meio e não foi possível identificar o problema que causou essa falha, nem por mensagem de erro, nem pela própria documentação da ferramenta. Por isso, optou-se por configurar a ferramenta fazendo o download do pacote completo do software.

Esta instalação por sua vez, também seguindo a documentação, foi completada com sucesso. Foi possível instalar o software e suas dependências e iniciá-los na máquina.

5.3.2 Avaliação da Documentação, Interface e Software

A documentação fornecida pelo TestSigma, em uma primeira impressão, parece completa, possui diversas seções que vão desde os conceitos mais básicos para se iniciar a utilização da ferramenta até funcionalidades mais complexas. Possui muitas imagens e um passo a passo para realizar estas tarefas.

A interface fornecida é inteiramente interativa, isto é, observa-se a presença de menus, botões, imagens e outros recursos visuais que visam guiar o usuário na utilização da aplicação. A quantidade de informações na tela é muito grande e a navegação no portal de testes é dificultada para um usuário que não possui familiaridade com a ferramenta.

Quando o uso da aplicação é iniciado, percebe-se que a documentação é bem escassa. Isto porque, já na seção de criação dos primeiros testes, o passo a passo fornecido não condizia com o que era observado na tela, causando a

impressão de que havia informações faltantes, inclusive a falta de documentação de erros.

Após um processo de tentativa e erro, foi possível a criação de um teste básico, de identificação de um input de texto na tela de teste (**Imagem 9**). De acordo com a documentação do Test Sigma, uma vez que os testes são criados, é possível executá-los de maneira automática ou sob demanda. Em um primeiro momento foi feita a tentativa de teste sob demanda, e quando este se iniciou, verificou-se que os recursos de processamento e memória do computador passaram a ser consumidos em grande quantidade, principalmente o uso da memória, que chegava aos quase 100% de utilização em uma máquina com 8GB. A utilização da máquina para outros recursos ficou inviabilizada e por isso o teste básico foi deixado em execução por ao menos duas horas. Após esse tempo, observou-se que o teste não havia sido concluído, não havia mensagens de sucesso ou falha, além do portal do TestSigma, no qual são cadastrados estes testes, não contabilizar a execução do teste até então realizado, já que o sistema apresenta indicação de que essas métricas são armazenadas. Para este caso, a documentação também não forneceu informações suficientes em como proceder. Outras tentativas foram feitas, porém sem sucesso.

5.3.3 Desinstalação

A desinstalação feita por Docker pode ser realizada pelo gerenciador de containeres do próprio docker e foi realizada com sucesso.

A desinstalação do programa instalado diretamente na máquina exigiu alguns passos extras, uma vez que o Test Sigma exige algumas extensões e dependências para funcionar por completo. Todas essas dependências precisaram ser desinstaladas separadamente, e após isso, a aplicação foi totalmente removida do computador.

5.4 Smart TestAuto

5.4.1 Instalação

O Smart TestAuto é uma aplicação que funciona em Java. Seguindo o guia fornecido pelo desenvolvedor, a instalação da aplicação parece ser bem sucedida por não fornecer nenhuma mensagem de erro.

Após a tentativa de inicializar a aplicação, mensagens de alerta são recebidas sobre a necessidade de dependências de Java não especificadas na documentação. Mesmo após a instalação dessas dependências, não foi possível inicializar o Smart TestAuto. Uma busca no código-fonte da ferramenta foi realizada na esperança de encontrar a origem do problema, porém também sem sucesso.

5.4.2 Avaliação da Documentação, Interface e Software

A documentação fornecida pelo desenvolvedor da ferramenta é um arquivo de texto organizado em capítulos, com explicações que vão desde a definição do software, que tipos de sistema este roda, como instalar e utilizar a aplicação, até sua configuração para utilização em aplicações mobile. Como houve problemas com a instalação, seria necessária uma seção para lidar com erros de instalação, mas isso não foi encontrado.

Não foi possível prosseguir na avaliação de interface e software, pois foram prejudicadas pela falha na instalação.

5.4.3 Desinstalação

Como a ferramenta é instalada no diretório do projeto a ser testado, a desinstalação pode ser feita removendo os arquivos do código-fonte obtido no github.

6 Resultados

Na seção 10.1 deste capítulo, apresentamos constatações da utilização do guia MEDE-PROS no contexto de teste web e o que julgou-se faltante no documento para que a avaliação de qualidade se tornasse mais completa.

Na seção 10.2 documentamos uma proposta de adequação do guia para o contexto estudado.

6.1 Avaliação da utilização do MEDE-PROS

Na etapa de utilização do guia de qualidade, respondendo ao questionário por ele proposto, verificou-se que o guia não oferece a abrangência desejada para o contexto de teste web, principalmente quando diz respeito às etapas de instalação e desinstalação, completude das funcionalidades, compatibilidade, desempenho e documentação.

Com relação às etapas de instalação e desinstalação, observou-se que o MEDE-PROS não faz avaliação de instalação por gerenciadores de pacote e da documentação fornecida para isso. O usuário é guiado a procurar informações em documentação impressa, capa e contra capa de CDs e outros artifícios físicos, como também não considerar informações contidas na documentação online. A avaliação é proposta para ferramentas que possuem instaladores nativos do próprio software. Como a seleção obtida pela aplicação do MSL é apenas de ferramentas web, a utilização do guia pode não ser clara para o usuário.

Quanto à completude das funcionalidades, o guia se atém a verificar se as funções propostas pela documentação do software estão corretamente aplicadas. Porém, para o contexto de teste na web, algumas funcionalidades são desejadas para que a atividade de teste seja satisfatória. Funcionalidades que, não estando presentes no guia, dificultam a comparação entre diferentes ferramentas. Espera-se, por exemplo, que um software de teste na web possa interagir com diferentes elementos de um website, o que não é checado no guia. O guia também não avalia a compatibilidade de uma aplicação web em diferentes browsers.

Por fim, como relatado na avaliação de qualidade do Test Sigma (Seção 9.4), a utilização do software foi impossibilitado também pelo uso excessivo de recursos de máquina. O MEDE-PROS porém, não possui uma seção destinada a análise de desempenho, e nessa etapa consideramos que a avaliação de tal característica é necessária.

6.2 Proposta de adequação do guia MEDE-PROS para o contexto de desenvolvimento web

Com base na avaliação do guia MEDE-PROS e nos modelos de qualidade de software propostos pela ISO 25010, propomos um total de 18 perguntas a serem adicionadas ao guia que o tornariam melhor adaptado para a avaliação de teste na web.

6.2.1 Instalação e Desinstalação (Portabilidade)

Para atender ao modelo de qualidade de Portabilidade e necessidade de avaliar a instalação da ferramenta por gerenciadores de pacotes, foram propostas 6 novas questões. São elas:

1. O software pode ser instalado por um instalador nativo da ferramenta?
2. O software pode ser instalado por gerenciadores de pacote? (ex: npm, Docker, etc)
3. Caso o software permita instalação por gerenciador de pacotes, a documentação da ferramenta apresenta um detalhamento de como realizar a instalação?
4. Caso o software permita instalação por gerenciador de pacotes, a documentação da ferramenta lista as dependências da ferramenta?
5. A documentação do software indica como instalar dependências da ferramenta caso necessário?
6. A instalação por gerenciador de pacotes realizou o procedimento corretamente?

6.2.2 Completude das funcionalidades (Adequação Funcional)

Atendendo a necessidade de adequar o guia de qualidade ao contexto de teste na web, foram propostas 9 perguntas com relação as funcionalidades necessárias para avaliar um software de teste:

1. O software consegue reproduzir a ação de navegação na web? (ex: ser direcionado para diferentes endereços e navegar por eles)
2. O software consegue interagir com uma página web dada uma URL?
3. O software consegue ter acesso aos diferentes elementos do DOM? (Document Object Model, ou, as diferentes estruturas que compõe uma página web)
4. O software permite reproduzir ações de clique do mouse?
5. O software permite reproduzir ações de input do teclado?
6. O software permite testar o som de uma página?
7. O software permite interação com inputs de arquivo?
8. O software permite a sequencialização de ações em uma página?
9. Permite a verificação de saídas produzidas na execução?

6.2.3 Compatibilidade

Usuários da internet navegam de diferentes maneiras, pelo computador pessoal ou telefone celular, utilizando diferentes browsers. Para isso, o software web precisa ser compatível com a maior parte destas plataformas. Com isto em mente, foram propostas 2 perguntas que avaliariam a questão da compatibilidade:

1. O software permite o teste em diferentes plataformas? (Ex: Notebook, mobile, etc)
2. O software permite o teste em diferentes navegadores? (Ex: Firefox, Chrome, Edge, Safari, etc.)

6.2.4 Eficiência de desempenho

Uma aplicação de teste na web, geralmente é uma aplicação que consome poucos recursos da máquina, rodando em segundo plano. Como a avaliação de qualidade do Test Sigma foi comprometida por este mesmo motivo, julgamos necessária a inclusão de 1 pergunta que atende a essa característica:

1. O software utiliza os recursos da máquina da maneira especificada nos requisitos de hardware?

6.2.5 Outros comentários

O preenchimento do documento é totalmente manual e, levando em consideração soluções que a própria internet oferece, a digitalização e experiência do usuário do guia poderia ser melhorada pela criação de uma aplicação.

O MEDE-PROS faz referências a textos que ainda não foram lidos. Por exemplo, para resolver os itens ***4.1 "Especificação dos Requisitos de Hardware e 4.2 "Especificação dos Requisitos de Software,*** é necessário que o usuário leia as informações da página 24, a qual está páginas à frente. Também é necessário avaliar os itens 4.1 e 4.2 após ter avaliado o atributo ***6.1 "Existência da Descrição do Produto", na página 24.*** Isso é um ponto negativo, pois o preenchimento da documentação não segue um padrão linear e fácil para o usuário. O que poderia ser implementado é uma avaliação mais linear, ou seja, em que o usuário não precise responder questões posteriores para então retornar a responder a questão em que ele está.

7 CONCLUSÃO

O objetivo deste trabalho foi realizar uma avaliação do guia de qualidade MEDE-PROS e, a partir disso, contribuir com uma proposta de mudança que se adapte a uma avaliação mais aprofundada de softwares de teste na web.

A utilização do guia com as ferramentas selecionadas no processo de Mapeamento Sistemático, e com o conhecimento obtido com relação à testes na web, permitiu uma análise levando em consideração o olhar do usuário final dessas ferramentas.

As modificações indicadas, de acordo com modelos de qualidade da ISO 25010, ajudam o utilizador do guia em situações antes não muito claras, como por exemplo na avaliação da condução da instalação do software, além de também oferecer uma lista de funcionalidades antes não presentes, que direcionam o usuário à completude necessária de uma ferramenta de teste na web.

Adaptou-se o guia focando-se principalmente nas funcionalidades que simulam a interação do usuário final com as aplicações web, isto é, funcionalidades que reproduzem inputs de mouse, teclado e interação com tela. Um aprodundamento dessa adaptação é possível, uma vez que testes na web também podem abranger testes de base de dados e de segurança, por exemplo.

Além disso,a abrangência do guia pode ser expandida ao se incluir novos modelos de qualidade da ISO 25010 à sua composição, tornando-o assim, ainda mais completo.

Como sugestão para trabalhos futuros temos:

- Proposta para migrar o guia MEDE-PROS para uma plataforma digital.
- Aprofundamento da adaptação do MEDE-PROS para testes na web direcionados a segurança.
- Aprofundamento da adaptação do MEDE-PROS para testes na web direcionados a bases de dado.
- Aprofundamento da adaptação do MEDE-PROS para outros modelos de qualidade de software segundo a ISO 25010.

8 REFERÊNCIAS

[1] Barbosa, E., et al. (2000) “Introdução ao Teste de Software. XIV SimpósioBrasileiro de Engenharia de Software”.

[2] Di Lucca, G.A. e Fasolino, A.R. (2006) “Testing Web Applications: The state of the art and future trends”, J. Inf. Softw. Technol. 48(2), p. 1172–1186.

[3] Gama, G. N. e Oliveira, S. R. B. (2011)."Spider-PQ: Uma Ferramenta de Apoio à Avaliação de Produtos de Software com base no MEDE-PROS." Disponível em: <https://www.academia.edu/27962606/Um_Modelo_Baseado_Em_Ontologia_e_Orientado_a_Riscos_Para_Certifica%C3%A7%C3%A3o_De_Qualidade_De_Produtos_De_Software>. Acesso em: 03/09/2022.

[4] Guia de Avaliação da Qualidade de Produto de Software” (2001), Instituto Nacional de Tecnologia da Informação, Laboratório de tecnologia da Avaliação da Qualidade de Produtos de Software. Disponível em: <https://www.dcce.ibilce.unesp.br/~ines/apeti/GuiaAvaliacao.pdf>. Acesso em: 03/09/2022.

[5] Lima, L. F. e Peres L. M. (2021) "Protocolo de mapeamento sistemático para busca de aplicativos de saúde em repositórios não- acadêmicos." Anais do I Workshop de Práticas de Ciência Aberta para Engenharia de Software. SBC.

[6] Martinez, M. R. M.; et al., (1999). "The software product evaluation database-supporting MEDE-PROS, " Proceedings 4th IEEE International Software Engineering Standards Symposium and Forum (ISESS'99). 'Best Software Practices for the Internet Age', pp. 182-191, doi: 10.1109/SESS.1999.766593.

[7]Myers, G. J.; (1979). The Art of Software Testing. Wiley, New York.

[8] Petersen, K., et al. (2015). Guidelines for conducting systematic mapping studies in SE: An update. Information and Software Technology.

[9] Pressman, R. S. e Maxim, B. R. (2021). Engenharia de software: uma abordagem profissional [recurso eletrônico] /; [tradução: Francisco Araújo da Costa; revisão técnica: Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade]. P. 750,751,753,755,756 e 774 – 9. ed. – Porto Alegre: AMGH. E-pub.

[10] Ricca, F. e Tonella, P. (2002), “Testing processes of web applications”, Annals of Software Engineering (14:1-4), p. 93-114.

[11] Santa Isabel, S. L.; (2011). Seleção de Abordagens de Teste para Aplicações Web/ Silvia Lopes Santa Isabel. – Rio de Janeiro: UFRJ/COPPE, 2011. Orientador: Guilherme Horta Travassos. Dissertação (Mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação. P. 25. Disponível em: < https://www.cos.ufrj.br/uploadfile/1311770421.pdf >. Acesso em: 03/09/2022.

[12] Testar. In: Enciclopédia Barsa Universal. (2007). São Paulo: Editora Planeta, v. 17, p. 5971.

[13] Vasconcelos, A. M. L., et al (2006)."Introdução à Engenharia de Software e à Qualidade de Software." Minas Gerais: Universidade Federal de Lavras.

[14]José P Miguel, David Mauricio, e Glen Rodríguez. A review of software qualitymodels for the evaluation of software products. arXiv preprint arXiv:1412.2977,2014.

[15] Raul Wazlawick. Engenharia de software: conceitos e práticas, volume 1. Elsevier Brasil, 2013.

[16] Ian Sommerville, Selma Shin Shimizu Melnikoff, Reginaldo Arakaki, e Edilson de Andrade Barbosa. Engenharia de software, volume 6. Addison Wesley São Paulo, 2003.

[17] ISO/IEC 25010:2011. Systems and software engineering - systems and softwarequality requirements and evaluation (square) - system and software qualitymodels. http://iso25000.com/index.php/en/iso-25000-standards/iso-25010.

[18] Individuals using the Internet, International Telecommunication Union. Disponível em:<https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx>

9 Apêndice A – Avaliação TestCafe utilizando MEDE-PROS

Completitude da Descrição do Produto

1. Especificação dos Requisitos de Hardware

O documento de Descrição do Produto do TestCafe específica:

1.1 Outros produtos de software fundamentais para colocar o produto em uso.

Ex: Visual Basil, Access, Delphy.

Funcionalidade - Adequação

2. Avaliação da Instalação

2.1 Execução da Instalação:

Instalação realizada por npm, por isso:

Exibe mensagens informando o andamento da tarefa.

Permite definir o subdiretório onde se quer instalar o Software.

Cria automaticamente subdiretórios e copia arquivos necessários.

Fornece função para interromper procedimento de instalação.

Exibe mensagem de sucesso, ou não, da instalação, após concluída.

Uma vez concluído, é possível executar o produto.

3. Avaliação da Documentação, Interface e Software

3.1 Avaliação da Documentação

A documentação especifica nome e versão do software.

A documentação apresenta, através de texto, imagens ou fotos que a interface com o usuário é feita através de linhas de comando.

Apresenta um texto introdutório, uma descrição geral do produto e suas funções.

Apresenta uma visão geral da documentação.

É de fácil compreensão para usuários experientes no assunto e apresenta com ideias claras seu conteúdo.

A documentação instrui o usuário, orientando-o na aprendizagem e permite uma fácil identificação das funções utilizadas por possuir uma organização lógica e evolutiva.

Apresenta símbolos e convenções ao usuário.

É clara, precisa e não apresenta erros gramaticais ou ortográficos.
Possui exemplos para auxiliar a compreensão do assunto tratado.
Apresenta um índice geral, com tópicos e links que funcionam corretamente.

3.2 Avaliação da Interface

A avaliação da interface foi majoritariamente prejudicada pelo fato da aplicação não possuir uma interface gráfica. A ferramenta é executada por linha de comando e suas funções escritas por linhas de código.

3.3 Avaliação do Software

As funções especificadas na documentação foram todas implementadas de forma completa e objetiva. Satisfazem a necessidade das tarefas a que o produto se propõe a realizar.
As funções estão implementadas corretamente e geram resultados conforme o esperado.
Não houve ocorrência de falhar durante a execução.

4. Avaliação da Desinstalação

A desinstalação também foi realizada por npm e por isso:
Realiza a tarefa de maneira automatizada.
Exibe mensagens informando o andamento da tarefa.
Exclui subdiretórios e arquivos automaticamente.
Fornece função para interromper procedimento de instalação.
Exibe mensagem de sucesso, ou não, da desinstalação, após concluída.

10 Apêndice B – Avaliação Selenium utilizando MEDE-PROS

Completitude da Descrição do Produto

1. Especificação dos Requisitos de Hardware

O documento de Descrição do Produto do Selenium especifica:

O software de sistema operacional necessário para colocar o produto em uso. Ex: Windows 10, Linux, Mac OS.

Outros produtos de software fundamentais para colocar o produto em uso. Ex: Visual Basic, Access, Delphy.

Funcionalidade - Adequação

2. Avaliação da Instalação

2.1 Execução da Instalação:

Instalação realizada por Java e npm, por isso:

Exibe mensagens informando o andamento da tarefa.

Permite definir o subdiretório onde se quer instalar o Software.

Cria automaticamente subdiretórios e copia arquivos necessários.

Fornece função para interromper procedimento de instalação.

Exibe mensagem de sucesso, ou não, da instalação, após concluída.

Uma vez concluído, é possível executar o produto.

3. Avaliação da Documentação, Interface e Software

3.1 Avaliação da Documentação

A documentação especifica nome e versão do software.

A documentação apresenta, através de texto, imagens ou fotos que a interface com o usuário é feita através de linhas de comando ou pela IDE Selenium.

Apresenta um texto introdutório, uma descrição geral do produto e suas funções.

Apresenta uma visão geral da documentação.

É de fácil compreensão para usuários experientes no assunto e apresenta com ideias claras seu conteúdo.

A documentação instrui o usuário, orientando-o na aprendizagem e permite uma fácil identificação das funções utilizadas, por possuir uma organização lógica e

evolutiva.
Apresenta símbolos e convenções ao usuário.
É clara, precisa e não apresenta erros gramaticais ou ortográficos.
Possui exemplos para auxiliar a compreensão do assunto tratado.
Apresenta um índice geral, com tópicos e links que funcionam corretamente.

3.2 Avaliação da Interface

A interface analisada foi feita a partir do IDE fornecido pelo Selenium:
A interface está organizada em grupos seguindo uma lógica compreensível.
Faz uso de identificadores que representam seu significado.
Possibilita a realização das tarefas desejadas.
Possui uma distribuição uniforme do conteúdo, de fácil leitura e identificação, apresentando mensagens com bom aspecto visual.
Exibe mensagens de orientação ao usuário que orientam de forma efetiva e eficiente, são auto explicativas e se limitam ao contexto da tarefa.
Possui funções de interface bem definidas e de forma a não deixar dúvidas sobre o que fazem.

3.3 Avaliação do Software

As funções especificadas na documentação foram todas implementadas de forma completa e objetiva. Satisfazem a necessidade das tarefas a que o produto se propõe a realizar.
As funções estão implementadas corretamente e geram resultados conforme o esperado.
Não houve ocorrência de falhar durante a execução.

4. Avaliação da Desinstalação

A desinstalação também foi realizada por Java e npm e por isso:
Realiza a tarefa de maneira automatizada.
Exibe mensagens informando o andamento da tarefa.
Exclui subdiretórios e arquivos automaticamente.
Fornece função para interromper procedimento de instalação.

11 Apêndice C– Avaliação Test Sigma utilizando MEDE-PROS

Completitude da Descrição do Produto

1. Especificação dos Requisitos de Hardware

O documento de Descrição do Produto do Test Sigma especifica:

Outros produtos de software fundamentais para colocar o produto em uso. Ex: Visual Basic, Access, Delphy.

Funcionalidade - Adequação

2. Avaliação da Instalação

2.1 Execução da Instalação:

Exibe mensagens informando o andamento da tarefa.

Cria automaticamente subdiretórios e copia arquivos necessários.

Fornece função para interromper procedimento de instalação.

Exibe mensagem de sucesso, ou não, da instalação, após concluída.

Uma vez concluído, não é possível executar o produto.

3. Avaliação da Documentação, Interface e Software

3.1 Avaliação da Documentação

A documentação apresenta, através de texto, imagens ou fotos que a interface com o usuário é feita através do portal de testes do Test Sigma.

Apresenta um texto introdutório, uma descrição geral do produto e suas funções.

Apresenta uma visão geral da documentação.

A documentação **não** é de fácil compreensão e **não** instrui o usuário de maneira eficiente.

3.2 Avaliação da Interface

A interface **não** está organizada de forma lógica e facilmente compreendida pelo usuário.

Faz uso de identificadores, botões e menus.

Não apresenta uma distribuição uniforme do conteúdo.

Não exibe mensagens de orientação ao usuário relevantes às tarefas executadas.

3.3 Avaliação do Software

A avaliação das funções implementadas foi prejudicada pelo uso excessivo de recursos da máquina de uma maneira **não** especificada na documentação.

As funções **não** realizavam as tarefas da maneira esperada e **não** apresentavam mensagens relevantes ao usuário quanto ao seu sucesso ou falha.

4. Avaliação da Desinstalação

Realiza a tarefa de maneira automatizada.

Exibe mensagens informando o andamento da tarefa.

Fornece função para interromper procedimento de instalação.

12 Apêndice D – Avaliação Smart Test Auto utilizando MEDE-PROS

Completitude da Descrição do Produto

1. Especificação dos Requisitos de Hardware

O documento de Descrição do Produto do Smart Test Auto **não** especifica:

Outros produtos de software fundamentais para colocar o produto em uso. Ex: Visual Basic, Access, Delphy.

O documento de Descrição do Produto do Smart Test Auto especifica:

O software de sistema operacional necessário para colocar o produto em uso. Ex: Windows 10, Linux, Mac OS.

Funcionalidade - Adequação

2. Avaliação da Instalação

2.1 Execução da Instalação:

Não exibe mensagens informando o andamento da tarefa.

Cria automaticamente subdiretórios e copia arquivos necessários.

Não fornece função para interromper procedimento de instalação.

Não exibe mensagem de sucesso, ou não, da instalação, após concluída.

Uma vez concluído, **não** é possível executar o produto.

3. Avaliação da Documentação, Interface e Software

3.1 Avaliação da Documentação

Apresenta um texto introdutório, uma descrição geral do produto e suas funções.
Apresenta uma visão geral da documentação.
A documentação **não** é de fácil compreensão e não instrui o usuário de maneira eficiente.

3.2 Avaliação da Interface

A avaliação da Interface foi prejudicada pela falha da execução do programa após sua instalação.

3.3 Avaliação do Software

A avaliação das funções implementadas foi prejudicada pela falha da execução do programa após sua instalação.
A documentação fornecida não atendia ao erro recebido. Foi realizada uma verificação do código fonte em busca de uma solução, porém sem sucesso.

4. Avaliação da Desinstalação

Não realiza a tarefa de maneira automatizada.
Não exibe mensagens informando o andamento da tarefa.
Não fornece função para interromper procedimento de instalação.