Gestão da qualidade de software por meio da personalização de software: Uma abordagem com o Versatile Test Automation (VTA) e Versatile Test Manager (VTM)

Fábio Alves[1], Nayane Maia[2], Carlos Santana[1], Thyago Faria Borges[1]

[1]Instituto de Pesquisas Eldorado – Brasília/DF – Brasil
[2]Instituto de Pesquisas Eldorado – Manaus/AM – Brasil

{fabio.alves, nayane.alves, carlos.santana,thyago.borges}
@eldorado.org.br

Resumo. *Este artigo discute a importância da gestão da qualidade do produto para que as organizações obtenham vantagens competitivas, com foco na customização de software no processo de testes com base em um estudo de caso na indústria. Várias limitações nas ferramentas convencionais foram identificadas através de resultados empíricos de projetos reais da indústria. Para otimizar o processo de testes, foram criadas ferramentas experimentais customizadas: VTA (Versatile Test Automation), voltado para automação de testes, e VTM (Versatile Test Manager), gerenciador de testes integrado ao VTA. Seu uso foi comparado com outras ferramentas de automação de testes em termos de facilidade de instalação, tempo de execução, facilidade de uso, facilidade de manutenção, integração CI/CD, geração de relatórios e métricas de execução e custo de licenciamento.*

1. INTRODUÇÃO

A importância da Gestão da Qualidade como uma estratégia para o sucesso empresarial é amplamente reconhecida e os seus efeitos podem ser percebidos de maneira interna ou externa à organização. No primeiro caso, ela contribui com a redução de desperdícios, com o aumento da produtividade e com a eliminação de defeitos nos produtos. Já de maneira externa, ela é significativa no processo de conquista e manutenção de clientes, uma vez que a qualidade dos produtos e serviços oferecidos é um fator decisivo para os consumidores [1].

No contexto atual, em que o mercado está em constante em expansão e cada vez mais acirrado, a tecnologia desempenha um papel crucial para fortalecer a vantagem competitiva das organizações. Investir, portanto, em softwares personalizados é uma estratégia adotada por muitas empresas para enfrentar os desafios do mercado, pois esses produtos permitem atender demandas e necessidades específicas de um setor, otimizando os processos e eliminando falhas operacionais que poderiam prejudicar a empresa [2]. É destacado que a plataforma totalmente personalizada para cada negócio engloba soluções e detalhes pensados para atender demandas específicas, permitindo adaptações futuras para acompanhar as mudanças do mercado. Essa flexibilidade garante uma vantagem competitiva, mantendo a empresa atualizada com as tendências e inovações mais recentes [3].

Este relato de experiência visa destacar a importância da gestão da qualidade na obtenção de vantagens competitivas para as organizações, com um enfoque na personalização de software no processo de teste. Diante dessa necessidade identificada em um Instituto de Pesquisas e Desenvolvimento, foi desenvolvida uma ferramenta personalizada de testes automatizados, a qual gerou uma patente, denominada VTA (*Versatile Test Automation*), integrada ao seu gerenciador de testes, chamado VTM (*Versatile Test Manager*) [4]. Foi realizado uma análise comparativa em relação as ferramentas de automação de scripts de teste convencionais do mercado *Selenium WebDriver, PlayWright, Katalon, Cypress* e *Robot Framework*, considerando os seguintes aspectos da qualidade: facilidade de instalação, o tempo de execução, a facilidade de uso, a manutenibilidade, a integração CI/CD, a obtenção de relatórios de execução e métricas e o custo de licença.

Os softwares de teste desempenham um papel fundamental na identificação de defeitos, avaliação de desempenho e validação das funcionalidades dos sistemas, sendo considerados produtos de alta qualidade. A fundamentação teórica por trás das ferramentas de teste modernas baseia-se em técnicas avançadas de desenvolvimento e abordagens de teste automatizado com inteligência artificial (IA) [5]. A construção da ferramenta VTA, adotou as seguintes técnicas: Programação por demonstração (*Programming by Demonstration* - PbD) [6]; Auto-recuperação (*self-healing*) [7]; Teste visual (*visual testing*) [8]; Estrutura de prevenção de flaky test [9]; Espera inteligente (*smart wait*) [10]; Teste orientado por palavras-chave (*Keyword-Driven Testing* - KDT) [11].

Com base nessas definições, as ferramentas foram projetadas e desenvolvidas pelos próprios analistas de testes do Instituto, os quais elicitaram os requisitos que a ferramenta deveria oferecer, dentre os quais se destacaram aquelas voltadas à Programação por Demonstrações (PbD), a teste visual e o self-healing. A ferramenta VTA foi desenvolvida, com uma estrutura robusta e resiliente para evitar flaky tests, e os algoritmos relacionados às três técnicas mencionadas (PbD, teste visual e self-healing) foram implementados. Na implementação do algoritmo de PbD, particularmente, a base utilizada foi o .NET CORE [12] e o JavaScript, devido à familiaridade dos desenvolvedores com essas tecnologias. A construção dos scripts baseou-se nos padrões de projeto de teste automatizado Keyword-Driven Testing (KDT), que representa ações ou verificações específicas que podem ser reutilizados em vários testes, e espera inteligente, que é usado para lidar com a espera de elementos da interface do usuário. Para a implementação do teste visual do software, foi utilizada a biblioteca Magick .NET [13], que é um wrapper de código aberto para a biblioteca de manipulação de imagens (ImageMagick). O Magick .NET é uma biblioteca poderosa para manipulação de imagens no ecossistema .NET. Devido à sua facilidade de uso, suporte a vários formatos de imagem e alto desempenho, é uma escolha popular para os desenvolvedores que precisam realizar operações de processamento de imagem em seus aplicativos. No painel gravador de ações do usuário, pela familiaridade técnica de desenvolvimento, foram utilizados o WPF .NET (Windows Presentation Foundation) [14] para criação do painel iterativo e o JavaScript para captura e apresentação das ações do usuário no painel. Durante a implementação da ferramenta VTM, os requisitos prioritários foram aqueles voltados à gestão de casos de teste, ao monitoramento de defeitos, ao seu uso integrado ao *Versatile Test Automation* (VTA) e à geração de

relatórios de execução de teste. Foram utilizados o *framework Express* [15], a biblioteca de *JavaScript Sequelize*® [16] e o banco de dados MySQL® [17].

2. ANÁLISE DOS RESULTADOS E CONCLUSÕES

Durante a análise comparativa entre as ferramentas disponíveis no mercado, destacou-se a ferramenta de código aberto – Playwright, por ser resiliente, rápida e moderna, possuindo, ainda, recursos avançados, como PbD e teste visual. Dessa forma, mesmo que possua menos recursos avançados em relação às ferramentas licenciadas, pode se tornar viável a depender das necessidades do projeto. Ao se comparar o Playwright ao VTA, contudo, foi verificado que o segundo possui mais recursos avançados e uma estrutura robusta e resiliente pré-montada, o que lhe garante uma maior cobertura de qualidade de software, além de ser de mais fácil manuseio e manutenção. Além disso, o VTA possui um gerenciador de teste integrado de alto desempenho, o VTM, para a geração de relatórios de execução de teste, o que permite o planejamento, a execução e o monitoramento eficiente dos testes. Também se ressalta que o processo de integração de scripts de teste automatizado é feito de uma forma muito menos complexa que as ferramentas de mercado, ou seja, qualquer testador com qualquer nível de habilidade pode estabelecer a integração dos casos de teste com os seus respectivos scripts de teste automatizado.

O VTA também é vantajoso quando comparado às ferramentas licenciadas, pois disponibiliza recursos semelhantes às versões pagas dessas ferramentas, mas sem custo. Além disso, para a empresa proprietária da ferramenta, o VTA e o VTM poderão ser atualizados para implantar os conceitos mais inovadores do mercado, garantindo-lhe vantagem competitiva.

Por outro lado, é importante ponderar que, por ser uma ferramenta personalizada, desenvolvida pelo próprio Instituto, demandou, naturalmente, a alocação de desenvolvedores para a sua construção, o que gerou impactos financeiros. Esses custos indiretos ainda serão necessários ao longo da sua utilização, devido à necessidade de alocação dos desenvolvedores para a sua manutenção contínua, correção de bugs, atualizações e fornecimento de suporte, o que seria eliminado se a empresa optasse por utilizar qualquer outra ferramenta. Ressalta-se, ainda, que o VTA possui algumas limitações, como a dependência de uma boa qualidade de imagem e a necessidade de ajustes manuais em casos mais complexos. Dessa forma, embora, no contexto geral, o seu uso tenha sido vantajoso no caso mencionado, é necessário que outras empresas realizem a análise de viabilidade para as suas situações específicas.

AGRADECIMENTOS

A publicação desse artigo foi incentivada pelo Instituto de Pesquisas Eldorado.

REFERÊNCIAS

[1] Mainardes, Emerson Fagner; Lourenço, Luis; Tontini, Gerson. "Percepções dos Conceitos de Qualidade e Gestão pela Qualidade Total: estudo de caso na universidade" pagina 282.

[2] Sommerville, Ian. "Engenharia de Software", Pearson Universidades; 10ª edição (22 abril 2019).

[3] Dyche, Jill. “The New IT: How Technology Leaders are Enabling Business Strategy in the Digital Age", McGraw Hill; 1st edition (January 26, 2015).

[4] Alves, Fabio. “MVP da ferramenta Versatile Test Automation Artificial Intelligence [Vídeo]”; You Tube; https://www.youtube.com/watch?v=iAqMFpTb3eg&list=PL_9wM-v2Sl1o3yKXyg-XftIWtBdVmmzql&pp=gAQBiAQB (November 9, 2022)

[5] Dhaya Sindhu Battina. 2019. Artificial intelligence in software test automation: A systematic literature review. International Journal of Emerging Technologies and Innovative Research (www. jetir. org| UGC and issn Approved), ISSN (2019), 1329–1332.

[6] Allen Cypher and Daniel Conrad Halbert. 1993. Watch what I do: programming by demonstration. MIT press.

[7] Debanjan Ghosh, Raj Sharman, H Raghav Rao, and Shambhu Upadhyaya. 2007. Self-healing systems—survey and synthesis. Decision support systems 42, 4 (2007).

[8] Kateryna Ivanova, Galyna V Kondratenko, Ievgen V Sidenko, and Yuriy P Kondratenko. 2020. Artificial Intelligence in Automated System for Web-Interfaces Visual Testing.. In COLINS.

[9] Alan Romano, Zihe Song, Sampath Grandhi, Wei Yang, and Weihang Wang. 2021. An empirical analysis of UI-based flaky tests. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 1585–1597.

[10] Satya Avasarala. 2014. Selenium WebDriver practical guide. PACKT publishing.

[11] Renaud Rwemalika, Marinos Kintis, Mike Papadakis, Yves Le Traon, and Pierre Lorrach. 2019. On the evolution of keyword-driven test suites. In 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST). IEEE, 335–345.

[12] .NET Core. 2023. .NET Core. https://learn.microsoft.com/pt-br/dotnet/

[13] Magick.NET. 2023. Magick.NET. https://github.com/dlemstra/Magick.NET

[14] WPF .NET. 2023. WPF .NET. https://tinyurl.com/yeyk8j99

[15] Express JS. 2023. Express JS. https://expressjs.com/pt-br/

[16] Sequelize. 2023. Sequelize. https://sequelize.org/

[17] MySQL. 2023. MySQL. https://www.mysql.com/