MÉTODOS PARA GARANTIR A QUALIDADE DO SOFTWARE E MELHORAR A COBERTURA E EFICIÊNCIA DOS TESTES AUTOMATIZADOS.

FLÁVIA ISHIZAKI BROCANELLA WITTER[1]
JULIA CAROLINA DA SILVA SOUSA[2]
MARIÂNGELA FERREIRA FUENTES MOLINA[3]

RESUMO

Este artigo aborda métodos para garantir a qualidade do software com foco na melhoria da cobertura e eficiência dos testes automatizados. Com sistemas cada vez mais complexos, a garantia da sua qualidade a partir da implementação de testes robustos se torna algo essencial para garantir sua confiabilidade. O objetivo principal é apresentar e analisar diferentes métodos que facilitam a implementação de testes automatizados e buscar demonstrar como essas práticas contribuem para a detecção precoce de falhas, aumentando a credibilidade do sistema. Por fim, o artigo conclui que a adoção dessas abordagens, em conjunto com um ciclo contínuo de adaptação e aprendizado, é crucial para tornar a melhoria da qualidade do software uma prática natural no processo de desenvolvimento, maximizando os benefícios da automação de testes.

Palavras-chave: Automação de testes; Cobertura de testes; Qualidade de software.

ABSTRACT

This article discusses methods to ensure software quality with a focus on improving the coverage and efficiency of automated testing. As systems become increasingly complex, ensuring their quality through the implementation of robust tests becomes essential for guaranteeing reliability. The main objective is to present and analyze different methods that facilitate the implementation of automated tests and to demonstrate how these practices contribute to the early detection of failures, thereby increasing the system's credibility. Finally, the article concludes that adopting these approaches, along with a continuous cycle of adaptation and learning, is crucial for making software quality improvement a natural practice in the development process, maximizing the benefits of test automation.

Key words: Automated testing; Software quality; Test coverage.

[1]Graduanda, Tecnologia em Análise e Desenvolvimento de Sistemas pela Faculdade de Tecnologia de Mogi das Cruzes – FATEC-MC. – Mogi das Cruzes-SP.
[2]Graduanda, Tecnologia em Análise e Desenvolvimento de Sistemas pela Faculdade de Tecnologia de Mogi das Cruzes – FATEC-MC. – Mogi das Cruzes-SP. E-mail: julia.sousa8@fatec.sp.gov.br
[3]Docente, Faculdade de Tecnologia de Mogi das Cruzes – FATEC-MC. – Mogi das Cruzes-SP.

INTRODUÇÃO

Nos últimos anos, a automação de testes cresceu como uma prática indispensável para garantir a qualidade do software e melhorar a eficiência dos processos de desenvolvimento. Collins et al (2010) diz que a adoção de metodologias ágeis e a crescente complexidade dos sistemas tornaram a automação uma escolha estratégica para reduzir riscos, otimizar tempo e garantir maior cobertura nos testes. Segundo Pressman (2011), essa preocupação se intensificou com o uso de abordagens iterativas, onde os testes de unidade, integração e sistema desempenham um papel essencial na garantia da qualidade do software.

Um dos principais desafios na automação de testes é a escolha das ferramentas e abordagens mais adequadas. Conforme Rapoza (2003), enquanto as grandes corporações podem investir em ferramentas próprias, desenvolvedores independentes ou pequenas empresas têm encontrado alternativas viáveis em soluções de código aberto. Essas ferramentas, embora acessíveis e com interfaces intuitivas, podem não oferecer a cobertura ideal e apresentar limitações que afetam a eficácia dos testes automatizados. Além disso, a estratégia de priorização dos testes para maximizar a eficiência ainda é um desafio para muitas equipes.

Além disso, Shihab et al (2010) sugerem que, para maximizar a eficiência dos testes, é essencial isolar funções e priorizar aquelas com maior risco, maior frequência de modificações e incidência de bugs. A partir dessa estratégia, o esforço inicial de automação pode ser otimizado, evitando o investimento em áreas do sistema com menor impacto no resultado.

O objetivo deste artigo é analisar e apresentar métodos eficazes para garantir a qualidade do software por meio da automação de testes e os aspectos que devem ser levados em consideração antes de automatizar. Será abordado como melhorar a cobertura e a eficiência dos testes automatizados, explorando diferentes tipos de testes, ferramentas disponíveis e melhores práticas para iniciar e escalar processos

de automação. O estudo será conduzido com base em literatura atual e análise de casos práticos, visando fornecer orientações úteis para desenvolvedores e equipes de software que buscam aprimorar suas estratégias de teste automatizado.

MATERIAL E MÉTODOS

Neste estudo, adotou-se uma metodologia de pesquisa bibliográfica com enfoque qualitativo. Foram coletados, analisados e interpretados artigos científicos publicados em bases de dados reconhecidas, como IEEE e JSTOR.

A escolha dos materiais levou em conta a relevância e o impacto da pesquisa no contexto da qualidade de software, priorizando publicações que refletissem as tendências atuais do setor. Essa abordagem permitiu identificar pontos cruciais relacionados à automação de testes, incluindo as principais dificuldades enfrentadas e suas consequências em empresas de diversos tamanhos.

O estudo foi estruturado em etapas que partiram da definição de cada um dos conceitos centrais envolvidos nos métodos de garantia de qualidade na eficiência dos testes automatizados. A coleta abrangente de dados facilitou a discussão de como a automatização afeta a dinâmica no desenvolvimento de software e como esta se aplica na realidade das empresas.

REFERENCIAL TEÓRICO

Conceitos de Qualidade de Software

Segundo a Norma ISO 8402 (International Organization for Standardization), qualidade é "a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas dos clientes" (ABNT,2000).

Em um contexto de desenvolvimento de software, podemos considerar as necessidades explícitas como as condições e objetivos propostos pelos

desenvolvedores e as necessidades implícitas seriam os fatores externos, compostos pelas necessidades do usuário.

O primeiro modelo sistemático para a gestão da qualidade surgiu no Japão na década de 1950, o Total Quality Control (TQC) que, segundo Camargo (2011), valoriza a qualidade em todos os aspectos de uma organização de forma generalizada, dando ênfase às atividades usuais de uma empresa. Na década de 80, nasceu o Total Quality Management (TQM), método americano que busca a melhoria contínua da qualidade, envolvendo todos os colaboradores e processos da organização. Mas foi a Organização Internacional de Padronização (ISO) que definiu normas para a garantia da qualidade em produção, serviços e gerenciamento.

No desenvolvimento de software, a qualidade está diretamente relacionada ao gerenciamento de requisitos e processos de desenvolvimento bem definidos. As métricas de qualidade são utilizadas para oferecer informações que contribuam no gerenciamento de projetos e na melhoria dos processos de engenharia de software. Estes indicadores devem ser escolhidos sempre com o objetivo de auxiliar na tomada de decisão e antecipação de problemas, caso eles não sirvam esses propósitos, eles não fazem sentido para o projeto em questão.

Essas métricas podem ser em nível organizacional, como índice de satisfação do cliente e margem de lucro, nível de projeto, como tempo de resposta e taxa de falhas, e nível de atividade, como número de erros por linha de código.

Métodos para Garantir a Qualidade do Software

Pressman (2011) diz que o uso de metodologias ágeis, que visam a entrega do produto de forma iterativa e incremental, evidenciou a preocupação com a automação de testes para garantir a qualidade do software que está sendo desenvolvido. Essas metodologias promovem o feedback rápido, possibilitando a adaptação contínua, assim tornando necessária que as falhas sejam detectadas e corrigidas rapidamente.

Uma prática essencial para a garantia da qualidade de software é a revisão de código, ou do inglês code review (CR). O CR consiste na inspeção do código por um desenvolvedor diferente de quem o escreveu, permitindo a identificação de erros, além da troca de conhecimento entre os membros da equipe, podendo ser realizada com o auxílio de ferramentas de revisão e definições do padrão esperado na codificação, garantindo que os aspectos críticos sejam revisados.

Existem diversos frameworks que podem auxiliar na automação dos testes, como o JUnit, Selenium e Cypress, e são frequentemente integrados com pipelines de integração contínua e entrega contínua (CI/CD).

Segundo Duvall et al (2007) a integração contínua permite que os programadores eliminem a necessidade de esperar que o seu trabalho seja concluído para compartilhá-lo com o restante do time. Assim a CI/CD ajuda a detectar problemas de forma mais rápida, mantendo o software sempre em um estado utilizável, permitindo que novas funcionalidades sejam testadas e implementadas de forma contínua, melhorando a adaptabilidade às necessidades do cliente, mesmo que novos requisitos apareçam no meio do ciclo de desenvolvimento.

Testes Automatizados

Os testes automatizados podem ser definidos como a simulação de usuários através da utilização de ferramentas, não necessitando de procedimentos manuais em sua execução, segundo Bartié (2013). Essa prática pode trazer uma maior cobertura de testes, além de auxiliar na detecção precoce de erros, economizando tempo e recursos.

Existem diferentes tipos de teste, Almeida (2019) descreve alguns deles:

- Testes unitários: tem o objetivo de testar as unidades do código fonte, verificando o funcionamento de componentes individuais, como métodos ou classes de um objeto. Esse tipo de teste é essencial para a

identificação de falhas em fases iniciais, evitando comprometer a qualidade do código fonte.

- Testes de integração: verifica o funcionamento da integração entre diferentes componentes do software, como problemas na transmissão de dados. Sua importância é para garantir a integridade do sistema como um todo.
- Testes funcionais: validam se o software está atendendo as especificações do cliente para cada uma das funcionalidades do sistema.
- Testes não-funcionais: estes são responsáveis por verificar aspectos como a performance, carga e confiabilidade do sistema, conforme citado por Vasconcelos et al (2006).

Dentre as principais técnicas de automação de testes podemos citar: record & playback, lógica de negócios, data-driven e keyword-driven

A técnica record & playback consiste em gravar as ações que seriam executadas por um usuário na interface gráfica da aplicação e converter essas ações em scripts de teste para serem executados quantas vezes necessário. Segundo Fantinato et al.(2004) a maior desvantagem do uso desta técnica é que para que os scripts sejam criados, as ferramentas utilizam objetos específicos ao projeto em questão, assim ela possui um alto custo e dificuldade de manutenção por necessitar ser refeita toda vez que haja uma mudança na interface do programa, ficando preso e não conseguindo se adaptar a mudanças no código.

Outra maneira é o uso da lógica de negócio para o desenvolvimento dos scripts, observando as funcionalidades da aplicação ao invés dos objetos do programa, conforme explica Boni (2024). Essa opção permite que testes com cálculos complexos e que necessitam de um grande número de repetições sejam realizados mais facilmente, porém é necessário um profissional com conhecimento em código e programação para a criação destes scripts. Existem diversas

ferramentas que suportam essa abordagem, como a JUnit para unidades de código Java, o MbUnit para testes em códigos .NET e o Nester para códigos em C#.

A técnica data-driven (orientada a dados) extrai dos scripts de teste os dados específicos para cada teste e os armazena em arquivos separados aos scripts, assim os dados são separados da lógica do teste, permitindo que diferentes entradas sejam testadas de forma sistemática.

E o teste keyword-driven separa a lógica dos testes da sua implementação, utilizando palavras-chave para representar ações e funções a serem executadas, facilitando a criação e manutenção dos scripts de teste.

O framework AutoTest utiliza uma técnica chamada keyword-data-driven, que é a união das duas técnicas apresentadas anteriormente, ele é formado por um conjunto de scripts e templates de planilha e procedimentos que permitem sua utilização sem a necessidade de adaptação para novos projetos. O AutoTest necessita apenas que seja elaborada a planilha de testes com os dados e procedimentos a serem executados e a única manutenção é a inclusão de novos scripts quando necessário.

Outros dois frameworks utilizados para a criação e gerenciamento de testes são o SilkCentral e o SilkTest, conforme explica Lima et al., (2012).

O SilkCenter permite o agendamento da execução dos testes, escolhendo o que será testado e a frequência em que ocorrerão. Após o agendamento, os testes ocorrem de forma completamente automática e podem ser acompanhados por relatórios que são gerados pela própria ferramenta.

Já o SilkTest é um ambiente de desenvolvimento integrado (IDE) que fornece a gravação, execução, debugging e edição dos scripts de teste. A ferramenta grava as ações do usuário e as transforma automaticamente em um script para que o teste possa ser realizado posteriormente, ou seja, ela realiza a tarefa do record & play sem a necessidade da interferência de um desenvolvedor, podendo ser utilizado em diversas linguagens de programação.

Ferramentas e Tecnologias

O uso de ferramentas e tecnologias nos testes de software representam avanços significativos na garantia da qualidade. A seguir podemos observar o ranking das ferramentas mais utilizadas, segundo o site OSS Insight, no período de 16 de setembro a 14 de outubro de 2024.

Tabela 1. Ranking de Ferramentas de Teste

POSIÇÃO	FERRAMENTA	ESTRELAS
1	keploy/keply	1009
2	microsoft/playwright	623
3	grafana/k6	563
4	SeleniumHQ/selenium	365
5	puppeteer/puppeteer	301
6	cypress-io/cypress	185
7	appium/appium	119
8	metersphere/metersphere	88
9	robotframework/robotframework	85
10	apache/jmeter	69

Fonte: Adaptado de OSS Insight, (2024).

Além dessas ferramentas, a tendência à integração de inteligência artificial e machine learning nos testes é cada vez mais explorada no mercado, possibilitando testes adaptativos a partir da identificação de padrões, geração de testes e análises de código automatizadas e a detecção de anomalias.

RESULTADOS E DISCUSSÃO

Alguns pontos que devem ser considerados na decisão de automatizar os testes em um projeto de desenvolvimento de software são:

1. a manutenção dos scripts dos testes: conforme o software vai evoluindo, os testes devem sofrer alterações para acompanhar essas mudanças e isso pode incluir a implementação de boas práticas na codificação e da revisão periódica dos scripts definidos

2. a cobertura dos testes: o uso de diferentes tipos de testes é essencial para garantir uma melhor análise do código, evitando que possíveis falhas passem despercebidas

3. o custo: automatizar os testes traz um custo de investimento inicial, tanto em ferramentas quanto com o desenvolvimento de scripts, porém, a longo prazo, a redução de erros e o aumento na eficiência dos testes e do software contrapõe esse peso inicial.

No intuito de aumentar a eficiência dos testes automatizados é preciso analisar os testes que estão sendo realizados para identificarmos os pontos mais críticos e com maior possibilidade de falhas, dando prioridades a essas áreas e assim aumentando a cobertura efetiva dos testes.

Com relação ao uso de ferramentas como as apresentadas nesse artigo, o uso de frameworks pode auxiliar na otimização dos testes, permitindo a realização de testes em diferentes ambientes paralelamente, reduzindo o tempo total necessário.

O uso de serviços em nuvem também pode ajudar na redução de custos com infraestrutura e permitindo a execução de testes em diversas configurações, o que pode ser uma alternativa para empresas com recursos menores para investimento em garantia da qualidade de software.

Embora não sejam conceitos novos, o uso da inteligência artificial e machine learning de forma integrada a automação de testes pode facilitar a adaptabilidade dos frameworks e scripts de teste com relação às mudanças nos softwares em

desenvolvimento. Uma forma disso ocorrer seria o uso de testes baseado em modelos, onde essas tecnologias auxiliam no desenvolvimento de casos de teste para prever possíveis falhas futuras usando como base os padrões de comportamento conhecidos.

CONCLUSÃO

A busca por métodos capazes de garantir a qualidade do software e aprimorar a cobertura e eficiência dos testes automatizados é fundamental no ambiente de desenvolvimento tecnológico e está em constante evolução. Os diferentes métodos discutidos, como a automação de testes com o uso ou não de frameworks e a integração contínua, demonstram o impacto que a identificação precoce de possíveis falhas e bugs na redução de custos a longo prazo.

O uso de metodologias ágeis alinhadas com ferramentas de teste, não apenas otimiza o processo de desenvolvimento, mas também traz mais qualidade para o produto, o tornando mais confiável. Embora a automação de testes traga um investimento inicial que muitas vezes não se encaixa na realidade de empresas menores e desenvolvedores autônomos, fica claro que este investimento é rapidamente compensado com os benefícios atrelados ao uso dessas tecnologias.

Assim, é essencial que os profissionais da área de desenvolvimento de software se mantenham atualizados e dispostos a se adaptar a novas estratégias de teste, uma vez que a possibilidade de integração da inteligência artificial e machine learning abrem um novo leque no qual a qualidade do software deixa de ser apenas um objetivo, mas uma prática contínua e integrada ao ciclo de desenvolvimento.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, D. E. T. de. **Avaliação de uso de automação de testes**. 2019. 60 f. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Federal Fluminense, Niteroi, 2019.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 8402: Gestão da qualidade – Vocabulário.** Rio de Janeiro, 2000.

BARTIÉ, A. **Garantia Da Qualidade De Software: Adquirindo Maturidade Organizacional**. São Paulo: Elsevier Brasil, 2013.

BONI, N. **Inteligência artificial para testes de software**. 2024. 39 f. Curso de Ciência da Computação, Universidade Federal de São Carlos – Ufscar, São Carlos, 2024.

CAMARGO, W. **Controle de Qualidade Total**. 2011. 150 f. Curso de Ciência e Tecnologia, Instituto Federal de Educação, Ciência e Tecnologia, Curitiba, 2011.

COLLINS, E. F.; LOBÃO, L. M. de A. **Experiência em automação do processo de testes em ambiente ágil com SCRUM e ferramentas Open Source**. 2010. Curso de Tecnologia em Sistemas de Informação, Instituto Nokia de Tecnologia (Indt), Universidade do Estado do Amazonas (UEA), Manaus, 2010.

DUVALL, P. M.; MATYAS, S.; GLOVER, A. **Continuous Integration: Improving Software Quality and Reducing Risk**. Boston: Addison-Wesley Professional, 2007. 366 p.

FANTINATO, M.; CUNHA, A. C. R. da; DIAS, S. V.; MIZUNO, S. A.; CUNHA, C. A. Q. **AutoTest – Um framework reutilizável para a automação de teste funcional de software**. 2004. Curso de Ciência da Computação, Unicamp – Universidade Estadual de Campinas, Campinas, 2004.

LIMA, T. L. L.; DANTAS, A.; VASCONCELOS, L. M. R. S. NETO, A. N. **Usando o SilkTest para automatizar testes: um relato de experiência**. 2012. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Federal da Paraíba (UFPB), João Pessoa, 2012.

OSS INSIGHT. **Testing tools**. 2024. Disponível em: https://ossinsight.io/collections/testing-tools/. Acesso em: 14 out. 2024.

PRESSMAN, R. S. **Engenharia De Software**. 7. ed. Porto Alegre: AMGH, 2011. 780 p.

RAPOZA, J. **“Open-source Testers offer low-cost alternatives” in eWeek.com**. 2003. Disponível em: http://www.opensourcetesting.org/articles/2003_Aug 11_Eweek.pdf. Acesso em: 03 out. 2024.

VASCONCELOS, A. M. L. de; ROUILLER, A. C.; MACHADO, C. Â. F.; MEDEIROS, T. M. M. de. **Introdução à engenharia de software e à qualidade de software**. 2006. 163 f. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Federal de Lavras, Lavras, 2006.