

O IMPACTO DA AUTOMAÇÃO DE TESTES NA QUALIDADE DE SOFTWARE

MATEUS RODRIGUES MARTINI

Resumo

Este artigo analisa a relação entre automação de testes e qualidade de software por meio de revisão de literatura e de dois estudos de caso publicados. O primeiro estudo examina testes automatizados de regressão integrados a um processo de integração contínua em uma instituição pública e apresenta resultados predominantemente qualitativos sobre viabilidade técnica, identificação de falhas e limitações de implementação. O segundo trata da automação de testes funcionais e menciona comparação de tempo entre execução manual e automatizada, mas os valores originais não estão disponíveis no material analisado. Os achados indicam que a automação pode favorecer a repetibilidade e a integração ao processo de desenvolvimento, embora seus efeitos dependam do contexto, das ferramentas, da manutenção dos *scripts* e das métricas utilizadas. Conclui-se que as evidências disponíveis permitem discutir benefícios potenciais, mas não sustentam generalizações quantitativas sobre redução de custos, aumento da confiabilidade ou melhoria global da qualidade.

Palavras-chave: automação de testes; qualidade de software; testes de regressão; testes funcionais; integração contínua.

THE IMPACT OF TEST AUTOMATION ON SOFTWARE QUALITY

Abstract

This article analyzes the relationship between test automation and software quality through a literature review and two published case studies. The first study examines automated regression testing integrated into a continuous integration process in a public institution and presents predominantly qualitative findings on technical feasibility, defect identification, and implementation limitations. The second addresses functional test automation and reports a comparison between manual and automated execution times, although the original values are not available in the analyzed material. The findings indicate that automation may improve repeatability and integration with the development process, but its effects depend on the context, tools, script maintenance, and metrics adopted. The available evidence supports a discussion of potential benefits but does not justify quantitative generalizations about cost reduction, reliability gains, or overall software quality improvement.

Keywords: test automation; software quality; regression testing; functional testing; continuous integration.

1 INTRODUÇÃO

A crescente complexidade dos sistemas de software, aliada à demanda por ciclos de entrega mais curtos, tem ampliado o interesse por práticas automatizadas no desenvolvimento.

Nesse contexto, a automação de testes pode favorecer a repetibilidade das execuções e apoiar a identificação de defeitos, desde que sua adoção seja adequada ao projeto e acompanhada por manutenção dos *scripts*. Entretanto, efeitos sobre confiabilidade, desempenho, custos e qualidade não devem ser tratados como universais, pois dependem das condições e das métricas examinadas em cada estudo (Bertolino, 2007; Myers; Sandler; Badgett, 2011).

A literatura especializada destaca que as decisões sobre quando e o que automatizar dependem do contexto do sistema, dos testes, das ferramentas e de fatores humanos e organizacionais (Garousi; Mäntylä, 2016). Esses benefícios, contudo, não são automáticos: a automação exige investimento inicial, capacitação das equipes e adaptação organizacional (Fewster; Graham, 1999). Além disso, a manutenção de *scripts* e a adaptação a mudanças frequentes nos requisitos permanecem desafios recorrentes.

Este artigo tem como objetivo analisar, por meio de pesquisa bibliográfica qualitativa e da análise de dois estudos de caso publicados, de que modo a automação de testes se relaciona com a qualidade de software. A análise considera benefícios relatados, limitações, desafios de implementação e a natureza das evidências apresentadas, distinguindo resultados quantitativos de interpretações qualitativas, sem pretensão de generalização.

2 CONCEITOS FUNDAMENTAIS DE QUALIDADE DE SOFTWARE

Duarte e Falbo (2000, p. 2) explicam que, no desenvolvimento de software, a qualidade pode ser entendida como um conjunto de características que devem ser satisfeitas em determinado grau para que o produto atenda às necessidades explícitas e implícitas dos usuários. Os autores ressaltam que a qualidade do produto não surge espontaneamente, mas é construída ao longo do desenvolvimento e depende fortemente da qualidade do processo adotado.

De acordo com Sommerville (2011), a qualidade de software não se resume à ausência de defeitos, mas envolve também a conformidade com padrões de desenvolvimento e a satisfação do cliente. Dessa forma, qualidade é um conceito tanto técnico quanto subjetivo, uma vez que depende da experiência e expectativas do usuário final.

Conforme Delamaro, Jino e Maldonado (2007), o desenvolvimento de software é um processo complexo no qual podem ocorrer problemas decorrentes, em grande parte, de erros humanos. Por isso, são necessárias práticas capazes de identificar e corrigir esses erros antes da finalização do produto, entre as quais se destacam os testes de software.

2.1 Modelos e normas de garantia da qualidade

A ISO/IEC 25010:2011 apresenta um modelo para avaliação da qualidade de produtos de software e organiza a qualidade do produto em oito características principais.

As oito características são: adequação funcional, eficiência de desempenho, compatibilidade, usabilidade, confiabilidade, segurança, manutenibilidade e portabilidade (ISO; IEC, 2011).

Além da ISO/IEC 25010, outras normas e modelos estabelecem referências para processos, avaliação e gestão da qualidade de software.

A ISO/IEC/IEEE 12207:2017 estabelece uma estrutura comum para os processos do ciclo de vida de software, abrangendo processos, atividades e tarefas aplicáveis ao desenvolvimento, à operação e à manutenção. A norma pode ser utilizada como referência para definição, controle e melhoria de processos de software (ISO; IEC; IEEE, 2017).

Complementarmente, o modelo CMMI (Capability Maturity Model Integration) desenvolvido pelo *Software Engineering Institute* (SEI) organiza práticas de melhoria de processos em níveis de maturidade. A versão CMMI-DEV 1.3 apresenta práticas voltadas ao

desenvolvimento de produtos e serviços e à melhoria sistemática dos processos organizacionais (SEI, 2010).

Outra referência histórica para avaliação de processos é a ISO/IEC 15504, conhecida como SPICE (*Software Process Improvement and Capability Determination*). A Parte 1 apresenta conceitos e vocabulário relacionados à avaliação de processos e à determinação de capacidade (ISO; IEC, 2004).

3 TESTES DE SOFTWARE

Conforme Sommerville (2011), os testes de software integram as atividades de verificação e validação e têm como finalidade revelar defeitos e verificar o atendimento aos requisitos especificados. A identificação de defeitos antes da disponibilização do software pode reduzir retrabalho e contribuir para a confiabilidade do produto, dependendo do contexto e da efetividade do processo de testes.

Segundo Sommerville (2011), os testes podem ser organizados em níveis como teste unitário, teste de integração, teste de sistema e teste de aceitação. Cada nível examina um escopo distinto do software e contribui para identificar defeitos em diferentes etapas do desenvolvimento.

Os testes unitários avaliam componentes isolados, enquanto os testes de integração verificam a interação entre esses componentes. Os testes de sistema examinam o comportamento do sistema integrado. Por sua vez, os testes de aceitação verificam se a solução atende aos critérios acordados e às necessidades dos usuários ou clientes.

Além disso, os testes podem ser classificados conforme diferentes abordagens, como os testes funcionais, estruturais, baseados em falhas, de caixa branca e de regressão.

3.1 Tipos de testes

O teste funcional, também conhecido como caixa-preta, é utilizado para verificar se o software atende aos requisitos especificados sem considerar sua estrutura interna. Nessa abordagem, os casos de teste são definidos a partir de entradas e saídas esperadas. Técnicas comuns incluem particionamento de equivalência, análise de valores-limite e tabelas de decisão (Pressman; Maxim, 2016).

Já o teste estrutural, de acordo com Sommerville (2011), concentra-se na lógica interna do código-fonte. Ele exige conhecimento técnico do sistema e utiliza critérios de cobertura de comandos, decisões, laços e caminhos de execução para avaliar a extensão dos testes. Esse tipo de teste é amplamente empregado por desenvolvedores na verificação de estruturas condicionais e da complexidade ciclomática do código.

Outra técnica importante é o teste baseado em falhas, que parte da premissa de que determinados tipos de erro podem ser antecipados e explorados durante o planejamento dos testes. Myers et al. (2004) discutem princípios de teste voltados à revelação de defeitos e à seleção de casos capazes de expor comportamentos incorretos. Práticas como *mutation testing* (teste por mutação) e *error seeding* (inserção proposital de erros) são exemplos de abordagens relacionadas à introdução ou modelagem deliberada de falhas.

O teste de caixa branca é frequentemente associado ao teste estrutural, pois considera diretamente o código e a lógica de implementação. Seu planejamento pode utilizar critérios de cobertura de instruções, decisões, condições e caminhos. Em sistemas não triviais, a execução exaustiva de todos os caminhos tende a ser inviável; por isso, critérios específicos de cobertura podem ser utilizados para avaliar a extensão dos testes estruturais.

Por fim, destaca-se o teste de regressão, utilizado para verificar se alterações realizadas no software afetaram funcionalidades anteriormente validadas. Esse tipo de teste é

especialmente relevante em contextos de manutenção evolutiva e pode se beneficiar da automação quando os mesmos cenários precisam ser reexecutados após sucessivas modificações no código.

4 TESTES AUTOMATIZADOS

Bernardo e Kon (2008) descrevem testes automatizados como *scripts* que exercitam funcionalidades e verificam resultados esperados, favorecendo a repetibilidade de cenários já definidos. Essa característica pode reduzir o esforço de reexecução e apoiar testes com grande volume de dados ou múltiplas combinações. Entretanto, efeitos sobre confiabilidade, qualidade e manutenção dependem da adequação dos casos, dos oráculos de teste, da estabilidade do ambiente e do custo de manutenção dos *scripts*. Assim, a automação deve ser apresentada como recurso de apoio à qualidade, e não como garantia automática de melhoria.

Com o aumento da complexidade dos sistemas e a adoção de metodologias ágeis, os testes automatizados passaram a ocupar papel relevante nos processos de desenvolvimento e garantia da qualidade.

De acordo com Pressman e Maxim (2016), a automação de testes pode desempenhar papel estratégico no desenvolvimento moderno de software, especialmente em ambientes que adotam Integração Contínua (CI) e Entrega Contínua (CD). A execução frequente de testes automatizados pode fornecer retorno mais rápido sobre alterações realizadas no software; contudo, seus efeitos dependem da qualidade dos casos de teste, da frequência de execução e da integração da automação ao processo de desenvolvimento.

Segundo Pressman e Maxim (2016), durante o desenvolvimento, testes unitários automatizados permitem verificar o comportamento esperado de componentes isolados com o uso de ferramentas como JUnit, NUnit e PyTest. Já nas fases de integração e validação, *frameworks* como Selenium e Robot Framework permitem simular interações de usuários e testar o comportamento da aplicação integrada.

Integradas a *pipelines* com ferramentas como Jenkins e GitLab CI/CD, essas soluções auxiliam na detecção de regressões introduzidas por novas alterações. Para que a automação seja eficaz, seu planejamento deve considerar a manutenção dos testes, a frequência de mudanças no sistema e a priorização de cenários críticos e repetitivos.

5 DESAFIOS DA AUTOMAÇÃO DE TESTES DE SOFTWARE

A automação de testes pode oferecer ganhos de eficiência e repetibilidade, mas sua implementação envolve desafios técnicos e organizacionais. Segundo Fewster e Graham (1999), um dos principais obstáculos está na seleção inadequada de ferramentas, que muitas vezes não se alinham ao contexto do projeto e podem gerar retrabalho e desperdício de recursos. Ferramentas escolhidas apenas por popularidade ou preço, sem análise das necessidades específicas, tendem a dificultar a manutenção dos testes automatizados.

Além disso, a manutenção dos *scripts* de teste é um problema recorrente, especialmente quando não se seguem boas práticas de design e modularização. Meszaros (2007) argumenta que testes mal estruturados se tornam frágeis e caros para manter, comprometendo a confiabilidade da automação em longo prazo. A ausência de padrões consistentes no desenvolvimento dos testes prejudica sua reutilização e dificulta a detecção de falhas reais.

Outro desafio crítico é a cultura organizacional. De acordo com Ahmed, Mahmood e Bano (2016), muitas equipes enfrentam resistência à adoção da automação por falta de treinamento, visão de longo prazo ou pela crença de que os testes manuais são mais flexíveis. A automação exige mudança de mentalidade, investimentos contínuos e integração com

práticas como integração contínua (CI), o que pode gerar atritos em equipes com pouca maturidade em testes.

O retorno sobre o investimento em automação depende do custo inicial de implementação, da frequência de reexecução, da estabilidade do sistema, do esforço de manutenção e do volume de cenários reutilizados. Como o artigo não apresenta dados econômicos próprios, não é possível estabelecer um prazo universal para a obtenção de benefícios. Dessa forma, a avaliação do retorno sobre o investimento requer métricas específicas de custo, esforço de manutenção e frequência de reutilização dos testes.

6 ESTUDOS DE CASO

6.1 Aplicação de testes automatizados de regressão

Alberto *et al.* (2020) realizaram um estudo de caso no Tribunal de Contas do Estado de Mato Grosso, comparando testes manuais e automatizados com foco na automação de testes de regressão integrada ao processo de integração contínua.

O processo de automação de testes no Tribunal de Contas do Estado de Mato Grosso (TCE-MT) foi implementado no contexto do sistema JusConex-e, com foco no módulo de Cadastro, Áreas e Temas. A automação seguiu o roteiro definido no PDMS (Processo de Desenvolvimento e Manutenção de Software), utilizando como base testes manuais previamente validados.

A metodologia compreendeu a seleção de testes manuais existentes, sua reescrita como *scripts* no Katalon Studio e a configuração de um fluxo de integração contínua com Jenkins, SVN e JBoss. Os testes foram executados em ambiente Linux e, conforme Alberto *et al.* (2020), em modo headless. Os critérios de seleção e a quantidade de casos de teste convertidos não foram informados na fonte consultada.

O estudo relata a identificação de falhas que não haviam sido detectadas manualmente, incluindo mensagens redundantes e comportamentos inadequados em operações de exclusão. Também atribui à padronização da execução a redução da suscetibilidade a falhas humanas e descreve a integração dos testes ao fluxo de integração contínua. As menções a aumento de cobertura e liberação de recursos humanos são resultados qualitativos, pois a fonte não apresenta métricas de cobertura, quantidade de falhas, esforço economizado ou alocação de pessoal. Portanto, esses achados devem ser interpretados como evidências descritivas do contexto analisado, e não como estimativas generalizáveis.

O estudo documenta dificuldades relacionadas ao carregamento incompleto de elementos HTML, aos identificadores dinâmicos e à execução em ambiente sem interface gráfica. Para contornar esses problemas, foram empregados mecanismos de espera, atrasos configurados manualmente, seletores XPath e comparação por classes CSS. Também são mencionados tempo de configuração e curva de aprendizagem da ferramenta. Essas limitações influenciam a manutenção e a reprodutibilidade da solução e, por isso, devem ser consideradas na interpretação dos resultados.

Com base nos dados apresentados, o estudo indica que a automação de testes de regressão integrada ao processo de integração contínua foi tecnicamente viável no contexto analisado. Contudo, não há evidência quantitativa suficiente, no material apresentado, para afirmar que a solução foi econômica ou que produziu melhoria significativa de confiabilidade. Os resultados sustentam principalmente uma interpretação qualitativa sobre padronização, repetibilidade e apoio à identificação de falhas.

6.2 Aplicação de testes automatizados funcionais

No estudo de caso de Barreto da Silva, Alves e Bruno (2011), a automação de testes foi aplicada à funcionalidade de inserção do objeto 'Linha' na aplicação denominada Teste. Os *scripts* realizaram inserções automáticas em campos numéricos, caixas de edição e áreas de texto com dados fornecidos por um datapool. Cada execução utilizou uma linha de dados, permitindo testar diferentes combinações de atributos. A extensão dessa varredura e seu efeito sobre a detecção de falhas não podem ser quantificados com as informações disponíveis.

O estudo compara a execução manual e a execução automatizada da inserção do objeto 'Linha'. Entretanto, os valores numéricos da tabela citada não estão disponíveis no material analisado. Por isso, não é possível confirmar a afirmação de que o tempo automatizado correspondeu à metade do tempo manual nem calcular a redução percentual. Com os dados disponíveis, os resultados podem ser descritos apenas como uma comparação de tempo relatada pelos autores, sem quantificação adicional.

Os testes automatizados descritos no estudo indicam potencial de redução do tempo de execução e de ampliação da repetibilidade na verificação da funcionalidade de inserção do objeto 'Linha'. Contudo, os dados disponíveis não permitem concluir que houve aumento da qualidade global do produto ou da satisfação de usuários e clientes, uma vez que essas dimensões não foram diretamente mensuradas. Tais relações podem ser discutidas apenas como possíveis implicações da automação, e não como resultados comprovados pelo estudo.

6.3 Análise comparativa dos estudos

Os estudos examinam objetos e evidências diferentes. Alberto *et al.* (2020) analisam a automação de testes de regressão integrada a um fluxo de integração contínua no sistema JusConex-e, com resultados predominantemente qualitativos sobre viabilidade técnica, identificação de falhas e dificuldades de implementação. Barreto da Silva, Alves e Bruno (2011) analisam testes funcionais na inserção do objeto 'Linha' e relatam uma comparação entre execução manual e automatizada; contudo, os valores originais dessa comparação não estão disponíveis no material analisado. Em consequência, os estudos não devem ser tratados como equivalentes: o primeiro oferece evidências contextuais sobre regressão e integração contínua, enquanto o segundo relata diferença de tempo entre as execuções manual e automatizada, mas a magnitude dessa diferença não pode ser verificada com os dados disponíveis.

7 CONSIDERAÇÕES FINAIS

A literatura e os estudos analisados indicam que a automação pode contribuir para a repetibilidade dos testes, a integração contínua e a execução de cenários previamente definidos. No primeiro estudo, os resultados disponíveis são predominantemente qualitativos e descrevem viabilidade técnica, identificação de falhas e desafios de implementação. No segundo, há referência a uma comparação de tempo entre execução manual e automatizada, mas os valores originais não constam do material analisado e, portanto, não permitem quantificação independente. Assim, as evidências examinadas permitem discutir benefícios potenciais e condições de adoção, mas não sustentam afirmações universais sobre redução de custos, aumento de confiabilidade ou melhoria global da qualidade.

A automação não constitui uma solução universal. Sua adoção depende de fatores discutidos na literatura, como capacitação técnica, clareza dos requisitos, atualização dos testes e contexto organizacional. Por isso, sua implementação requer planejamento e

acompanhamento por métricas capazes de avaliar custos, esforço de manutenção, reutilização e resultados no contexto específico de cada projeto.

Futuras pesquisas podem aprofundar a análise dos impactos econômicos da automação em longo prazo e explorar abordagens híbridas que combinem testes automatizados e manuais, especialmente em contextos de testes exploratórios e sistemas com interfaces complexas.

REFERÊNCIAS

- AHMED, F.; MAHMOOD, S.; BANO, M. Test automation challenges: a review. *International Journal of Software Engineering & Applications*, v. 7, n. 6, p. 37-50, 2016. DOI: 10.5121/ijsea.2016.7604.
- ALBERTO, J.; OLIVEIRA, J.; REIS, S.; CUNHA, V.; PELEGRINI, S.; NUNES, E. P. dos S. Automação de testes: um estudo de caso. *ISLA 2020 Proceedings*, 2020. Disponível em: <https://aisel.aisnet.org/isla2020/8>. Acesso em: 06 jun. 2025.
- BARRETO DA SILVA, P. C.; ALVES, T. S.; BRUNO, E. A. Automação de testes funcionais: testes funcionais automatizados de software. *Revista de Ciências Exatas e Tecnologia*, v. 6, n. 6, p. 113-133, 2011. Disponível em: <https://exatastechnologias.pgsscogna.com.br/rcext/article/view/2311/2210>. Acesso em: 28 ago. 2026.
- BERTOLINO, A. *Software testing research: achievements, challenges, dreams*. In: FUTURE OF SOFTWARE ENGINEERING (FOSE '07), 2007. Proceedings [...]. Minneapolis: IEEE Computer Society, 2007. p. 85-103.
- BERNARDO, P. C.; KON, F. A importância dos testes automatizados: controle ágil, rápido e confiável de qualidade. *Engenharia de Software Magazine*, São Paulo, ano 1, n. 3, p. 54-57, 2008. Disponível em: <https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf>. Acesso em: 28 ago. 2026.
- DELAMARO, M. E.; JINO, M.; MALDONADO, J. C. *Introdução ao teste de software*. São Paulo: Pearson Prentice Hall, 2007.
- DUARTE, K. C.; FALBO, R. A. Uma ontologia de qualidade de software. In: WORKSHOP DE QUALIDADE DE SOFTWARE, 2000, João Pessoa. *Anais [...]*. Vitória: UFES, 2000. Disponível em: https://nemo.inf.ufes.br/wp-content/papercite-data/pdf/uma_ontologia_de_qualidade_de_software_2000.pdf. Acesso em: 06 jun. 2025.
- FEWSTER, M.; GRAHAM, D. *Software test automation: effective use of test execution tools*. Harlow: Addison-Wesley, 1999.
- GAROUSI, V.; MÄNTYLÄ, M. V. When and what to automate in software testing? A multi-vocal literature review. *Information and Software Technology*, v. 76, p. 92-117, 2016. DOI: 10.1016/j.infsof.2016.04.015.
- ISO; IEC. ISO/IEC 15504-1:2004: *Information technology - Process assessment - Part 1: Concepts and vocabulary*. Geneva: International Organization for Standardization, 2004.

ISO; IEC. ISO/IEC 25010:2011: *Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. Geneva: International Organization for Standardization, 2011.

ISO; IEC; IEEE. ISO/IEC/IEEE 12207:2017: *Systems and software engineering - Software life cycle processes*. Geneva: International Organization for Standardization, 2017.

MESZAROS, G. *xUnit test patterns: refactoring test code*. Boston: Addison-Wesley, 2007.

MYERS, G. J.; SANDLER, C.; BADGETT, T.; THOMAS, T. M. *The art of software testing*. 2. ed. Hoboken: John Wiley & Sons, 2004.

MYERS, G. J.; SANDLER, C.; BADGETT, T. *The art of software testing*. 3. ed. Hoboken: Wiley, 2011.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. 8. ed. Porto Alegre: AMGH, 2016.

SOFTWARE ENGINEERING INSTITUTE (SEI). *CMMI for Development, Version 1.3*. Pittsburgh: Carnegie Mellon University, 2010. Technical Report CMU/SEI-2010-TR-033.

SOMMERVILLE, I. *Engenharia de software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011.