Práticas de Garantia de Qualidade aplicadas ao CACTO: Desafios e Benefícios para Evolução do Software de Fiscalização do Trânsito de Mercadorias

Eliane Santos Silva, Ana Carolina Rodrigues Lima de Aguiar, Yuska Paola Costa Aguiar, Marcelo Iury Oliveira

Centro de Informática (CI) – Universidade Federal da Paraíba (UFPB)
Avenida dos Escoteiros, s/nº, Mangabeira, João Pessoa, Paraíba, Brasil

elianesantossilva1998@gmail.com, acrla@academico.ufpb.br, yuska@ci.ufpb.br, marceloiury@gmail.com

Abstract. *This paper presents the impacts of introducing Quality Assurance (QA) practices in the CACTO system, highlighting the significant improvement in delivery quality. It shows that the QA team's efforts promoted the standardization of testing processes, the creation/improvement of documentation, the implementation of automated tests, the development of test cases based on registered issues, and the validation of fixes, resulting in greater system reliability and stability. Additionally, future work perspectives are discussed, such as improving documentation, mitigating technical debt, expanding test automation, and validating new interfaces, aiming for continuous evolution.*

Resumo. *Este relato de experiência apresenta os impactos da introdução de práticas de Garantia de Qualidade (QA) no sistema CACTO, destacando a melhoria significativa na qualidade das entregas. A atuação da equipe de QA promoveu a padronização dos processos de testes, a criação/melhoria de documentações, a automatização de teste, a criação de casos de testes baseados nas issues cadastradas e a validação das correções, resultando em maior confiabilidade e estabilidade do sistema. Além disso, são discutidas perspectivas de trabalhos futuros, como a melhoria de documentações, a mitigação da dívida técnica, a ampliação da automação de testes e a validação das novas interfaces, visando a evolução contínua.*

1. Controle e Acompanhamento de Mercadorias em Trânsito *Online*

O ICMS é o Imposto sobre Operações relativas à Circulação de Mercadorias e sobre Prestações de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação [Lei nº 5.122]. Para o Direito Tributário, as transações comerciais sujeitas ao ICMS consideram operações ligadas à compra e venda de produtos que resultem na transferência da propriedade desses bens, ou seja, na concretização de um negócio jurídico. Portanto, se caracteriza como uma das principais captações de recursos no país. No Estado da Paraíba, conforme informações da SEFAZ-PB (Secretaria do Estado da Fazenda da PB), o ICMS é o principal tributo do Estado, representando 91,7% das receitas dos impostos do exercício de 2022 (ICMS, IPVA, e ITCD). No contexto da SEFAZ-PB, a Gerência de Fiscalização de Mercadoria em Trânsito (GOFMT) trabalha, de forma integrada, com os auditores das subgerências de mercadoria em trânsito na monitoração, na fiscalização do trânsito de mercadorias dentro e fora do Estado. Em particular, GOFMT atua na inspeção de documentos e na verificação da regularidade fiscal dos produtos, além de combater práticas de sonegação e contrabando, garantindo a

conformidade com a legislação tributária. Ainda, essa gerência promove a arrecadação de tributos, oferece orientação aos contribuintes e controla a documentação fiscal, contribuindo para a justiça fiscal e a concorrência leal no mercado.

Por décadas, o modelo de fiscalização de mercadorias em trânsito que foi aplicado no território brasileiro era baseado em trabalho manual de vários auditores. Uma ação frequente era a parada de veículos de carga e análise dos documentos fiscais, bem como das mercadorias sendo transportadas. Essa abordagem exigia a utilização maciça de mão-de-obra, o que por sua vez reduzia a assertividade do modelo de fiscalização. Assim, a SEFAZ-PB e outros órgãos de fiscalização estaduais buscaram a criação e a implementação de mecanismos mais modernos de gestão tributária que auxiliem no combate à sonegação. O Sistema CACTO (Controle e Acompanhamento de Mercadorias em Trânsito *Online*) é uma destas iniciativas focado no uso de tecnologias de comunicação e processamento de dados aliado ao uso de inteligência artificial, cujo objetivo maior é o aumento da eficiência fiscal. Ele vem sendo desenvolvido e utilizado com objetivo de dar suporte computacional e em tempo real ao controle, acompanhamento e monitoramento de mercadorias em trânsito no Estado da Paraíba. Para tanto, são realizados registros de passagem de veículos por sensores instalados em rodovias e cidades (câmeras de leitura de placas, LPR), que geram alertas com base em documentos fiscais eletrônicos (Manifesto Eletrônico de Documentos Fiscais (MDF-e), Nota Fiscal Eletrônica (NF-e) e Conhecimento de Transporte Eletrônico (CT-e), em tempo real. Tais alertas dispõem das informações sobre a localização atual do veículo, o que favorece a realização de uma abordagem mais assertiva na fiscalização. Pois, a partir do cruzamento de bases de documentos fiscais e imagens, com uso de inteligência artificial, é possível a análise e identificação de possíveis ilícitos tributários.

Considerando a complexidade do CACTO e seu impacto para a sociedade, este artigo apresenta o relato da experiência da atuação de uma equipe de Garantia de Qualidade de Software (*Quality Assurance;* QA) no desenvolvimento e evolução do sistema. Com esse objetivo, se deu a implementação de processos de verificação e validação de software, de forma integrada às ações das equipes de desenvolvimento e DevOps. Portanto, se fez necessário preparar o processo e ambiente de teste, elaborar os artefatos de teste e trabalhar na conscientização sobre a importância da melhoria contínua de qualidade de software. Adicionalmente, considerando a necessidade de redução da dívida técnica em relação aos testes do sistema, além da documentação sobre os fluxos de transição de estado das entidades presentes no sistema, foram definidas as suítes de casos de teste de aceitação, mapeamento entre *issues* e casos de teste, assim como automatização a partir de *scripts* de teste. Com as ações realizadas, tem-se a promoção de maior confiabilidade ao software, reduzindo a incidência de falhas em nível de produção, assim como na manutenção e evolução do sistema. Este relato finda com a indicação dos principais ganhos percebidos até então com a equipe de QA no CACTO, assim como, com as lições aprendidas e possibilidade de trabalhos futuros.

2. Qualidade de Software: Visão Transversal na Engenharia de Software

A engenharia de software engloba uma ampla gama de atividades que abrangem todo o ciclo de produção de software, incluindo a especificação, desenvolvimento, verificação, validação e evolução do software [Sommerville, 2011]. A verificação e validação do

software é uma etapa de extrema importância para a qualidade do software, muitas vezes subestimada ou abordada tardiamente no processo de desenvolvimento. É comum uma compreensão equivocada sobre esta atividade no contexto do desenvolvimento de software. Sendo, muitas vezes resumida ao termo teste de software, no que diz respeito a, apenas, executar o software e verificar os resultados, se correspondem ou não ao esperado (testes dinâmicos), ou direcionando o foco para, exclusivamente, verificar a corretude e completude dos requisitos do sistema. No entanto, esse processo é transversal ao desenvolvimento de software e deve ter como objetivo a qualidade de software. Ao considerar os atributos de qualidade de software [ISO/IEC 25000: 2014], o teste vai além da sua funcionalidade (corretude, completude e adequação) e perpassa por requisitos não funcionais de desempenho, usabilidade, portabilidade, segurança, compatibilidade e confiabilidade. Ainda, tem impacto crucial na manutenção do sistema de software. Ou seja, diante da evolução do sistema, modificação para melhorá-lo, corrigi-lo ou adaptá-lo, a manutenibilidade representa o grau de eficácia e eficiência com o qual um sistema pode ser alterado, considerando sua modularidade, capacidade de reuso, analisabilidade, modificabilidade e testabilidade.

Neste sentido, os testes auxiliam na evolução do sistema, favorecendo a realização de mudanças com segurança, para garantir que as alterações não impactem na regressão do sistema. Portanto, os testes incluem atividades de planejamento, análise, modelagem e implementação dos testes, relatórios de progresso e resultados de testes e avaliação da qualidade dos objetos de teste [Stapp, Roman & Pilaeten, 2024]. Nesse contexto, são contemplados tanto testes dinâmicos (com execução do sistema), quanto testes estáticos, como a revisão de produtos de trabalho (requisitos, histórias de usuários e código-fonte). Além disso, a abordagem de verificação dos requisitos é ampliada para a validação do sistema em busca de conhecer se o mesmo atende ou não as necessidades das pessoas usuárias e demais *stakeholders*, em diferentes ambientes de operação. Ainda, os testes de software devem acontecer de forma prematura, sistematizada e alinhada com o processo de desenvolvimento de software. Caso contrário, defeitos de software podem ser propagados ao longo do ciclo de vida do projeto, trazendo prejuízos e retrabalho [Delamaro, Maldonado, & Jino, 2007]. Em outras palavras, a má compreensão de um requisito, por exemplo, leva à inserção de um defeito em um artefato de software (modelo de classe), sendo este implementado em código. Uma vez o código executado, a falha acontece quando o comportamento difere do esperado.

Portanto, desde a Especificação dos Requisitos é possível definir Testes de Aceitação, ou realizar testes estáticos de revisão do documento de requisitos para encontrar ambiguidades, inconsistências, informações ausentes ou obsoletas, e a partir de então melhorar a qualidade do artefato alvo. Vale ressaltar a importância dos testes e seus diferentes níveis [Contan, Dehelean & Miclea, 2018] a depender da fase de desenvolvimento. Os níveis de teste referem-se ao tamanho dos componentes do software que são testados, a saber: testes de unidades de código atômico individuais, o teste de integração sobre a integração de múltiplas unidades, e o teste de sistema contempla o uso do sistema (ou funcionalidade/serviço), próximo do que um usuário final faria [Fulcini *et al.,* 2023]. Cada nível tem especificidades em termos de objetos de teste, tipos de falhas encontradas, defeitos a serem corrigidos e técnicas de teste. Ainda,

os testes podem ser caracterizados, conforme a necessidade ou não de acesso ao código (como caixa-preta, caixa-branca ou caixa-cinza), e sobre o tipo de execução.

Na eventualidade de os testes de software não serem realizados como parte transversal no processo de desenvolvimento, pode-se indicar a existência de uma Dívida Técnica (DT) [Rios *et al.* 2018] que, se não gerenciada e mitigada, tem como principal impacto a dificuldade de evolução do sistema de forma segura [Alves *et al.* 2018] com maior custo de reparação. Logo, destaca-se a importância das ações integradas entre as equipes de QA e de desenvolvimento, na compreensão de um trabalho coletivo com objetivo de garantir qualidade de software para produto, processo, projeto e pessoa.

3. Implantação e Adoção de Práticas de Qualidade no Sistema CACTO

O Sistema CACTO vem sendo desenvolvido nos últimos três anos, estando em produção e uso pela Gerência de Fiscalização de Mercadoria em Trânsito (GOFMT) do Estado da Paraíba. Em princípio, não foi possível integrar uma equipe dedicada à Garantia de Qualidade de Software (QA) à equipe de desenvolvimento (DevOps). No período em questão, as atividades de verificação e validação das funcionalidades implementadas eram realizadas de forma restrita, não sistematizada e exploratória. Como consequência, falhas eram encontradas tardiamente, em alguns casos pelo cliente em ambiente de produção, impactando na confiabilidade e gerando um ciclo contínuo de retrabalho para a equipe de DevOps (*debug* e correção em diferentes módulos).

Este cenário favoreceu a presença de uma Dívida Técnica relativa aos artefatos de teste. A amortização desta tem sido alvo na atual fase do projeto que visa, além da evolução do CACTO com a inserção de novas funcionalidades, mudanças na interface e na interação. Como estratégia, foi priorizada a integração de uma equipe de Garantia da Qualidade para alinhar o processo de teste ao de desenvolvimento, tendo como principais responsabilidades preparar o processo e ambiente de teste, documentar os fluxos de transição de estado das entidades presentes no sistema, especificar suítes de casos de teste, mapear *issues* em casos de teste e automatizar teste.

A equipe de QA é composta por três integrantes, com experiência em teste e automatização. Inicialmente foi realizado um treinamento sobre o CACTO com a exploração livre do sistema, apoiada pela interação e troca de conhecimento com os demais integrantes da equipe, sendo o *Tech Lead* essencial neste processo. As reuniões semanais acontecem, tanto para a equipe de QA, quanto para a equipe de projeto, com todas as pessoas envolvidas. As ações implantadas pela equipe de QA estão sendo realizadas e refinadas gradativamente, desde Abril de 2024.

As práticas adotadas são: **[Documentação] Fluxos dos estados dos alertas e da ação fiscal** tiveram a documentação formalizada para serem utilizados como referência na especificação de casos de testes, para guiar o desenvolvimento de novas funcionalidades, promovendo um melhor entendimento do comportamento do sistema; **[Documentação] Modelo padronizado para registro das *issues*** foi definido, tornando a comunicação clara e objetiva entre os membros das equipes de QA e desenvolvimento, compreendendo: Título, Navegador, Telas, Descrição, Procedimento de teste, Comportamento esperado, *Branch* e Anexo (imagem e/ou vídeo); **[Documentação] Severidade e Tipo de *Issue*** são definidas na criação das *issues*, a partir do uso de

labels. Para a severidade, considera-se o impacto da *issue* no sistema com a classificação de Baixa, Média e Alta (*Low*, *Medium*, *High*), pela equipe de QA diante da avaliação do impacto na experiência do uso e o comprometimento de funcionalidades críticas. Para o tipo de *issue* têm-se *bug* ou *melhoria*, útil na análise de criticidade e priorização de *issues*; **[Gestão de Projetos e *Issues*] GitHub** tem sido utilizado como ferramenta de gestão colaborativa entre a equipe de QA, desenvolvimento e DevOps, permitindo o registro e rastreamento das *issues* identificadas, controle de versão de código, gestão de *branches* e revisão de *pull requests*, apoiando o ciclo de entrega contínua. As falhas são reportadas seguindo um modelo padrão e organizadas no *backlog*, facilitando o planejamento e priorização das correções; **[Processo] Fluxo de Processo Reportar, Corrigir e Validar** estabelece a sequência de ações a serem realizadas indicando a equipe responsável (QA e/ou de desenvolvimento) contribuindo para uma comunicação clara entre as pessoas envolvidas em suas respectivas atribuições; **[Documentação] Especificação da Suíte de Testes** tem como ponto de partida a análise da documentação dos Fluxos dos estados dos alertas e da ação fiscal, juntamente com as telas do sistema, para mapeamento entre requisitos e funcionalidades implementadas. De forma complementar, casos de testes também foram especificados considerando as *issues* reportadas; e, **[Instrumentalização de Testes] Automatização dos casos de teste** foi realizada com base na Suíte de Testes para funcionalidades transversais, contemplando diferentes telas do sistema e níveis de teste, utilizando a ferramenta *Selenium* e a linguagem *Python*.

As ações implementadas pela equipe de QA e a integração destas com a equipe de desenvolvimento tem como reflexo a redução da Dívida Técnica no contexto de Teste de Software, seus processos e artefatos. A documentação dos Fluxos dos estados dos alertas e da ação fiscal permite uma melhor apropriação da equipe do projeto ao conhecimento sobre o domínio da aplicação, passando a ser uma base única para consulta sobre o funcionamento da aplicação. O mapeamento entre os requisitos do sistema e funcionalidades implementadas que são passíveis de teste (manual ou automático, com base na suíte de teste especificada) permite quantificar a evolução do sistema e a segurança em alterá-lo. Adicionalmente, o Fluxo de Processo Reportar, Corrigir e Validar, aliado à documentação das *issues* de forma padronizada e com informações sobre severidade e tipo, favorece uma sinergia entre as equipes. Findando com a automação dos testes, tais estratégias promovem uma melhoria progressiva na qualidade do software a partir da adoção de uma abordagem mais estruturada para a evolução do Sistema CACTO, com consequente melhoria no nível de confiabilidade das entregas em ambiente de produção.

4. Resultados das Práticas de Qualidade no Sistema CACTO

Considerando a atuação da equipe de QA no projeto CACTO durante o período de abril de 2024 a março de 2025, é possível observar que a sistematização no processo de identificação, registro, correção e verificação de *issues* permitiu um melhor acompanhamento da evolução dos sistema, sobretudo no tocante à identificação e correção de problemas. O quantitativo de *issues* cadastradas mensalmente no projeto (Gráfico 1) revela um expressivo número de problemas identificados nos primeiros meses de atuação da equipe de QA (154 *issues* em 5 meses), havendo uma redução

deste quantitativo nos meses subsequentes (94 *issues* em 7 meses). Este comportamento pode ser associado a uma maior estabilidade do sistema. A distribuição de *issues* por telas (Gráfico 2) permite a identificação dos módulos com maior concentração de problemas, em conformidade ao princípio de testes de que problemas se agrupam, permitindo uma melhor alocação de esforços da equipe do projeto mitigar recorrências de problemas nos módulos mais complexos do CACTO.

A comparação entre *issues* abertas (identificadas e registradas) e concluídas (corrigidas e validadas) (Gráfico 3), evidencia um ritmo contínuo de evolução da equipe na correção de problemas, reduzindo o *backlog* ao longo do tempo. No tocante à distribuição das *issues* de acordo com a severidade (Gráfico 4), percebe-se a predominância de *issues* de severidade Média (125; 50,6%), seguidas por Baixa (85; 34,4%) e Alta (37, 14,9%). Essa configuração indica que a maior parte das ocorrências apresentava impacto moderado no sistema, exigindo correções prioritárias, não urgentes.

5. Lições Aprendidas e Perspectivas Futuras

Do exposto, três dimensões podem ser mencionadas como ganhos das práticas da equipe de QA no CACTO: (i) Qualidade do Produto com o aumento da estabilidade e previsibilidade das entregas, redução do número de *bugs* encontrados em produção e com a especificação prévia dos casos de teste contribuindo para evitar falhas em funcionalidades críticas; (ii) Qualidade do Processo a partir da implementação de fluxos formais de rastreabilidade de *issues*, documentação de testes e uso estruturado de ferramentas como GitHub para gestão e integração das atividades; e Qualidade do Projeto com a redução do retrabalho na equipe de desenvolvimento, permitindo que o *tech lead* e os desenvolvedores pudessem focar em melhorias e novas implementações, em vez de correções emergenciais.

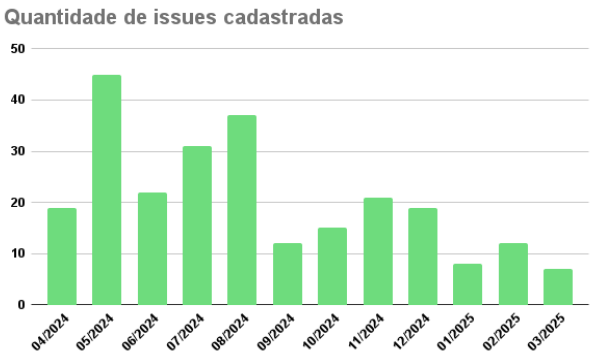

Gráfico 1. *Issues* por mês

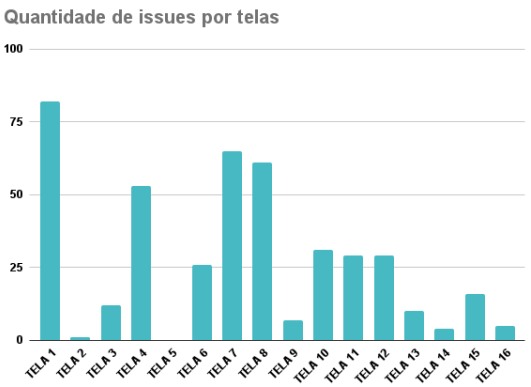

Gráfico 2. *Issues* por telas

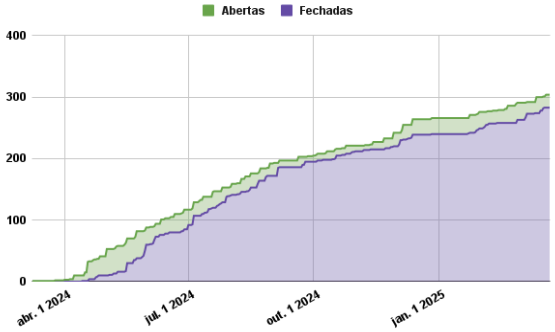

Gráfico 3. *Issues* abertas e fechadas

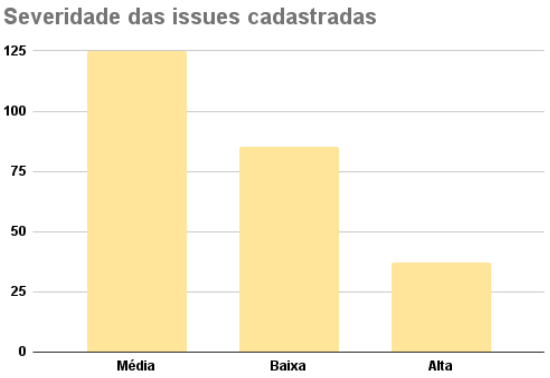

Gráfico 4. Severidade das *issues*

As métricas quantitativas reforçam o impacto positivo da atuação da equipe de QA, pois o número de *issues* identificadas e corrigidas tende a uma estabilidade ao longo do tempo, com a manutenção de ritmo consistente de identificação, correção e verificação, reduzindo a incidência de problemas reincidentes. Assim como pela redução de *bugs* críticos identificados em ambiente de produção, pois com a implementação de testes automatizados e a formalização, houve um declínio perceptível na quantidade de falhas reportadas por usuários finais. Ao considerar as métricas subjetivas, coletadas a partir do *feedback* da equipe do projeto (incluindo *stakeholders*), indicam uma melhora na percepção da confiabilidade do sistema e no alinhamento entre as equipes de desenvolvimento e QA.

A maior parte das respostas foi fornecida por membros da equipe de desenvolvimento (77,8%) e por integrantes do CACTO com mais de um ano de atuação no projeto (77,8%), o que demonstra maturidade sobre o sistema. A percepção sobre o impacto do time de QA é bem positiva: 88,9% avaliaram o impacto como “muito significativo” na melhoria da qualidade e 77,8% enxergam a contribuição como “muito alta” nas entregas. Entre os principais avanços apontados, destacam-se a redução de bugs em produção, a melhoria na documentação de testes e funcionalidades, e a identificação mais rápida de falhas.

Além disso, o feedback revela que o papel da equipe de QA no projeto é visto majoritariamente como “apoiar por meio de testes e validações constantes” e aponta a responsabilidade de “garantir entregas sem falhas”. Essa percepção, combinada aos avanços observados, mostra que a atuação do time de QA no CACTO contribuiu para ampliar a confiabilidade do sistema, o alinhamento entre as equipes e a cultura de qualidade no projeto.

Adicionalmente, o uso de uma ferramenta única de gestão para documentar e acompanhar o histórico de correções e melhorias, com registro de cada *issue*, associando-a a uma ramificação (*branch*) específica e anexando evidências de testes (imagens e/ou vídeos), torna o processo mais simples e eficiente. Consequentemente, a atuação da equipe de QA agrega valor à qualidade técnica, mas também na confiabilidade do projeto por completo.

Visando a evolução contínua do CACTO, e a possibilidade de seu uso ampliado para outros Estados do Brasil, é imprescindível que as práticas de Garantia da Qualidade continuem sendo aplicadas e aperfeiçoadas. Dentre as estratégias, têm-se como alvo a alocação de esforço para mitigar a dívida técnica considerando os itens de teste necessários para tal, desde a documentação de requisitos, especificação dos casos de testes, inclusive para todas as *issues* identificadas e findando na automatização dos casos de teste. Adicionalmente, considerando o *redesign* para a interface do sistema, se faz necessário conduzir um processo abrangente de validação de todos os comportamentos a partir de testes regressivos, de modo a assegurar a consistência funcional. Sob a perspectiva de usabilidade, acessibilidade e qualidade da experiência de uso, estão previstas avaliações heurística de usabilidade [Nielsen, 1995], inspeção à norma de Ergonomia da interação humano-sistema [NBR ISO 9241-171], assim como, testes com participação de usuários e o Sistema CACTO para coleta de métricas de uso e sondagem da satisfação subjetiva.

Agradecimentos

Agradecemos à Secretaria da Fazenda (SEFAZ-PB), à coordenação do CACTO e às equipes de desenvolvimento e de DevOps no empenho pela evolução do sistema.

Referências

Alves, M., Nunes, Gava, V., and Luiz (2018). Uma proposta para identificar, medir e gerenciar a dívida técnica em requisitos de software. International Conference on Information Systems and Technology Management.

Andrei Contan, Catalin Dehelean, and Liviu Miclea. (2018). Test automation pyramid from theory to practice. In Proceedings of the IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR). IEEE, 1–5.

Delamaro, M. E., Maldonado, J. C., Jino, M. (2007) Introdução ao teste de software. Elsevier

Fulcini, T., Coppola, R., Ardito, L., & Torchiano, M. (2023). A review on tools, mechanics, benefits, and challenges of gamified software testing. ACM Computing Surveys, 55(14s), 1-37.

Ian Sommerville. (2011). Engenharia de Software (9ª ed.). Pearson Prentice Hall. p.5.

Nielsen, J. (1995). How to conduct a heuristic evaluation. Nielsen Norman Group, 1(1), 8.

ISO: ISO/IEC 25000: 2014, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE

ISO 9241–171:2008 - Ergonomics of human-system interaction - Part 171: Guidance on software accessibility

Rios, N., de Mendonça Neto, M. G., and Sp´ınola, R. O. (2018). A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Information and Software Technology, 102:117–145

Stapp, L., Roman, A., & Pilaeten, M. (2024). ISTQB® Certified Tester Foundation Level. Springer Nature Switzerland. https://doi. org/10.1007/978-3-031-42767-1.