UNIVERSIDADE SÃO JUDAS TADEU

BACHARELADO EM SISTEMA DE INFORMAÇÃO

Geovane Augusto Costa dos Santos
Natan Fernandes Araujo Ibiapina
Henryk Bagdanovicius Roza
Olivia Frankiw
João Luiz Santana Borean

CONCEITO DE GARANTIA DE QUALIDADE DE SOFTWARE

São Paulo

2025

Sumario

Introdução à Garantia de Qualidade de Software (SQA) 3

Pilares da SQA 3

Técnicas de SQA 4

SQA em Metodologias de Desenvolvimento 4

Desafios na Implementação 5

Estudos de Caso 5

Tendências Futuras 5

Conclusão 5

Referências 6

1. Introdução à Garantia de Qualidade de Software (SQA)

A Garantia de Qualidade de Software (SQA) é um conjunto sistemático de atividades para prevenir defeitos, garantir conformidade com padrões e assegurar que o software atenda às expectativas dos usuários. Segundo Pressman (2016, p. 381), a SQA envolve "processos, métricas e ferramentas para monitorar e melhorar a qualidade durante todo o ciclo de vida do software". Surgiu na década de 1970, impulsionada por falhas críticas como a do software do Mariner 1 da NASA, que custou US$ 80 milhões (Sommerville, 2015, p. 655). Seu objetivo principal é evitar erros, não apenas corrigi-los, reduzindo custos com retrabalho e garantindo confiabilidade.

2. Pilares da SQA

2.1 Processos Padronizados

Modelos de Maturidade**:**

CMMI: Avalia a capacidade da equipe em níveis (caótico a otimizado) (Sommerville, 2015, p. 662).

ISO 9001: Foca na gestão contínua da qualidade (Artigo: Pereira et al., 2018).

Exemplo: Revisões de código e gestão de requisitos (Pressman, 2016, p. 385).

2.2 Métricas de Qualidade

Processo: Taxa de defeitos (ex: 2 erros/1000 linhas), tempo de ciclo.

Produto: Complexidade ciclomática, cobertura de testes (Pressman, 2016, p. 389).

Dado: Corrigir um defeito pós-lançamento custa 100x mais (Boehm, 1984).

2.3 Ferramentas de Automação

Testes: Selenium (web), JUnit (unitários), LoadRunner (performance).

Análise Estática: SonarQube para identificar code smells.

Integração Contínua (CI): Jenkins e GitHub Actions (Sommerville, 2015, p. 670).

2.4 Cultura Organizacional

Envolvimento de toda a equipe (desenvolvedores, testadores, gestores).

Empresas com cultura de qualidade têm 40% menos defeitos (Runeson et al., 2012).

3. Técnicas de SQA

3.1 Revisões Formais

Inspeção de Fagan: Identifica até 75% dos erros antes dos testes (Pressman, 2016, p. 388).

Walkthroughs: Discussões informais para validar requisitos (Sommerville, 2015, p. 657).

3.2 Testes Especializados

Segurança: OWASP Top 10 para vulnerabilidades.

Usabilidade: Heurísticas de Nielsen (Nielsen, 1994).

Baseado em Modelos (MBT): Geração de casos de teste via UML (Ammann & Offutt, 2016).

3.3 Gestão de Configuração

Controle de versões (Git) para evitar conflitos entre versões (Sommerville, 2015, p. 670).

4. SQA em Metodologias de Desenvolvimento

4.1 Cascata

Revisões formais em cada fase (ex: design antes da codificação) (Sommerville, 2015, p. 52).

4.2 Ágil (Scrum)

Test-Driven Development (TDD): Testes escritos antes do código (Beck, 2003).

Definition of Done (DoD): Critérios claros para aceitação (Pressman, 2016, p. 402).

4.3 DevOps

Pipelines CI/CD automatizados (ex: GitHub Actions) com testes contínuos (Netflix Tech Blog, 2020).

Exemplo: Netflix usa Chaos Engineering para testar resiliência.

5. Desafios na Implementação

Custos Iniciais: Ferramentas e treinamento exigem investimento (Pressman, 2016, p. 395).

Resistência Cultural: Equipes rejeitam processos burocráticos (Šmite et al., 2019).

Complexidade em Projetos Grandes: Coordenação em equipes globais (Sommerville, 2015, p. 672).

6. Estudos de Caso

6.1 NASA (Mars Rover)

Revisões diárias de código e simulações de falhas em ambientes extremos (Sommerville, 2015, p. 664).

6.2 Spotify

Squads autônomos com testes automatizados reduzem tempo de entrega em 30% (Spotify Labs, 2017).

7. Tendências Futuras

IA na Qualidade: Machine Learning prevê defeitos (ex: Google Tricorder) (McConnell, 2021).

Shift-Left Testing: Testes iniciados na fase de requisitos (Pressman, 2016, p. 392).

8. Conclusão

A SQA é essencial para entregar software confiável, combinando processos, automação e cultura. Desafios como custos e resistência exigem abordagens estratégicas, enquanto tendências como IA prometem revolucionar a área. Como afirmou Henry Ford: "Qualidade é fazer certo quando ninguém está olhando".

Referências:

PRESSMAN, R. S. Engenharia de Software: Uma Abordagem Profissional. 8. ed. AMGH, 2016.

SOMMERVILLE, I. Engenharia de Software. 10. ed. Pearson, 2015.

AMMANN, P.; OFFUTT, J. Introduction to Software Testing. 2. ed. Cambridge, 2016.

NIELSEN, J. Usability Engineering. Academic Press, 1994.

SPOTIFY LABS. "Scaling Agile @ Spotify". 2017.

NETFLIX TECH BLOG. "Chaos Engineering". 2020.