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

O DILEMA DA QUALIDADE DE SOFTWARE

São Paulo

2025

Sumario

Software "Bom o Suficiente"

Um software "bom suficiente" é aquele que atende aos requisitos mínimos necessários para resolver um problema ou atender a uma necessidade específica, sem buscar perfeccionismo ou funcionalidades excessivas. Essa abordagem prioriza eficiência prática, equilibrando custos, prazos e complexidade, sem comprometer a funcionalidade básica ou a satisfação do usuário.

Características Principais (Baseadas em Sommerville):

1. **Foco em Requisitos Essenciais**:
    - O software resolve o problema central do usuário, mesmo que não inclua funcionalidades avançadas (Sommerville, Cap. 4, p. 98).
2. **Custo-Benefício Equilibrado**:
    - Não busca otimização extrema, mas entrega valor dentro de restrições orçamentárias e de tempo.
3. **Manutenção Simplificada:**
    - Código organizado o suficiente para permitir ajustes futuros, sem complexidade desnecessária.

Visão de Pressman:

Pressman reforça que um software "bom suficiente" deve:

1. Evitar "over-engineering": Não adicionar complexidade técnica além do necessário (Pressman, Cap. 1, p. 20).
2. Atender a Escopo Definido: Cumprir requisitos acordados com o cliente, mesmo que limitados.
3. Ser Testável: Garantir que as funcionalidades básicas funcionem sem falhas críticas, mesmo com testes pragmáticos (Pressman, Cap. 14, p. 381).

Quando é Aplicável?

- Projetos com Restrições Rígidas: Prazos curtos, orçamentos limitados ou recursos escassos.
- Prototipagem ou MVP (Produto Mínimo Viável): Versões iniciais para validar ideias antes de investir em melhorias.

Custo da Qualidade

Custo da qualidade

Tanto a qualidade quanto a falta dela tem um custo e é necessário saber qual priorizar, o custo de qualidade abrange todos os gastos necessários para alcançar a qualidade ou para realizar atividades ligadas à qualidade, bem como os custos gerados pela sua ausência. Para entender esses custos, uma organização precisa coletar métricas que sirvam como base para determinar o custo atual da qualidade, identificar oportunidades de redução desses gastos e estabelecer um padrão de

comparação. O custo da qualidade pode ser categorizado em custos relacionados à prevenção, avaliação e falhas.

- **Custo de prevenção**

Englobam gastos com gestão para planejar e coordenar atividades de controle garantia da qualidade, desenvolvimento técnico de modelos detalhados de requisitos e projetos, planejamento de testes e treinamento. Esses investimentos são fundamentais para evitar falhas futuras e assegurar a qualidade.

- **Custo de avaliação**

É o custo para se avaliar ou inspecionar a qualidade do que está sendo entregue para que se cumpra os requisitos do projeto após passar por cada processo.

- **Custo de falhas**

São subdivididos em: falhas internas e falhas externas. As internas são observadas quando se é detectado o erro antes da entrega e envolvem o custo de reparar, reformular e avaliar a falha, já as externas ocorrem após a entrega ao cliente e podem dizer respeito por exemplo ao custo de solucionar o problema, reformular o produto, custo de mão de obra, perda de reputação etc.

Riscos

O dilema da qualidade de software envolve equilibrar a entrega de produtos funcionais e eficientes com a mitigação de riscos que podem comprometer sua integridade e segurança.

1. **Vulnerabilidades de Segurança**

Falhas ou bugs no software podem ser explorados por malware para contornar defesas ou obter privilégios não autorizados. Por exemplo, vulnerabilidades de estouro de buffer permitem que invasores injetem código malicioso, comprometendo o sistema.

2. **Privilégios Excessivos**

Usuários ou programas com mais privilégios do que o necessário podem ser alvos de malware, que tira vantagem desses privilégios para subverter o sistema. Por exemplo, aplicativos Android que solicitam mais permissões do que o necessário podem representar riscos significativos.

3. **Homogeneidade de Sistemas Operacionais**

Ambientes onde todos os computadores executam o mesmo sistema operacional são mais vulneráveis a ataques em massa, já que uma única exploração pode afetar múltiplos sistemas simultaneamente.

4. **Falta de Testes Adequados**

A ausência de testes de software abrangentes pode resultar em falhas não detectadas, afetando a funcionalidade e a confiabilidade do produto final. Falhas podem ser originadas por diversos motivos, incluindo especificações incompletas ou erros de implementação.

5. **Gerenciamento Ineficiente de Riscos**

A falta de um processo sistemático de gerenciamento de riscos pode levar a falhas de software, resultando em atrasos e custos adicionais. O gerenciamento de riscos visa

mitigar ou minimizar os efeitos de fatores de risco, produzindo um produto de software com qualidade, que atenda às necessidades do cliente, dentro do prazo e custos estimados.

6. **Ausência de Modelagem de Ameaças**

Não identificar e documentar ameaças potenciais durante o ciclo de vida do desenvolvimento de software pode resultar em vulnerabilidades não mitigadas, comprometendo a segurança do sistema. A modelagem de ameaças é um processo que consiste no levantamento de contramedidas de segurança que resolvem as ameaças ao software.

Para mitigar os riscos associados à qualidade de software, é fundamental adotar boas práticas em todas as fases do desenvolvimento. Para lidar com vulnerabilidades de segurança, é essencial realizar auditorias de código e testes contínuos, como testes de penetração e análise estática. O problema de privilégios excessivos pode ser resolvido aplicando o princípio do menor privilégio, garantindo que usuários e sistemas tenham apenas as permissões estritamente necessária

Responsabilidade e Negligência

O que é a responsabilidade por software?
É a responsabilidade legal das empresas que fabricam softwares em questões relacionadas ao seu produto.
A empresa deve se responsabilizar por danos causados por seu software, o que geralmente depende dos termos específicos descritos nos contratos, das leis da região e de como o software é utilizado.

Os 6 tipos de passivos de software
1.Negligência nos padrões de qualidade: quando o software não atende ao que foi prometido.
2.Danos por software defeituoso: quando o software contém falhas que afetam sistemas físicos ou a segurança.
3.Incumprimento de normas regulatórias: quando não cumpre normas, podendo resultar em penalidades legais.
4.Violação de propriedade intelectual: quando há uso indevido de códigos ou ideias protegidas.
5.Violação de contrato: quando o suporte prometido não é fornecido.
6.Deturpação ou fraude: quando o software é promovido com falsas promessas ou omite falhas.

Garantir a qualidade e a conformidade do software é essencial para evitar riscos legais e prejuízos. Empresas devem adotar boas práticas de desenvolvimento, segurança e suporte para reduzir passivos e proteger clientes e negócios.

Segurança do Software

A segurança do software visa protegê-lo contra ataques maliciosos que podem comprometer sua integridade e confidencialidade. Para que seja aplicada de forma eficaz, é necessário considerar tanto aspectos técnicos quanto humanos, adotando boas práticas no planejamento do software. Alguns exemplos incluem:

Criptografia: Utilizar métodos como AES ou RSA para cifrar dados sensíveis que transitam ou são armazenados pelo software.
Autenticação e Autorização: Implementar protocolos como OAuth ou JWT para verificar a identidade e os privilégios dos usuários que acessam o sistema.
Validação e Sanitização de Dados: Aplicar expressões regulares ou filtros HTML para evitar ataques de injeção de código ou cross-site scripting (XSS).
Detecção e Prevenção de Intrusões: Utilizar ferramentas como firewalls ou IDS/IPS para bloquear ou alertar sobre tentativas de acesso não autorizado.
Seguir boas práticas em todas as etapas do desenvolvimento é essencial para garantir a qualidade, a eficiência e a segurança do software.

O Impacto das Ações Administrativas

Controle de permissões evita acessos não autorizados.
Políticas de segurança (como autenticação multifator) protegem dados sensíveis.
Manutenção e Atualizações
Atualizações regulares corrigem vulnerabilidades e melhoram a performance.
Backups garantem recuperação de dados em caso de falhas.
Otimização de Recursos
Monitoramento de servidores e bancos de dados melhora a eficiência.
Ajuste de configurações evita desperdício de recursos.
Conformidade e Auditoria
Registro de logs permite rastrear alterações e identificar falhas.
Conformidade com normas (LGPD, GDPR) evita penalidades legais.
Suporte e Atendimento
Resolução rápida de incidentes minimiza impactos nos usuários.
Implementação de melhorias contínuas mantém a satisfação do usuário.

Referências:

auditeste.com.br
Aspectos negativos gerados pela falta de qualidade de software
1 de maio de 2024 — Exploração dos custos ocultos da baixa qualidade de software: retrabalho, suporte técnico e insatisfação do cliente. · Retrabalho: a necessidade ...
Codurance
Por que o gerenciamento de riscos de software é importante?
5 de abril de 2023 — Os riscos comumente associados podem ser conformidade com regulamentos, certificações, problemas de produção, e proteção de dados, os quais ...
https://d3.works/boas-praticas-de-desenvolvimento-de-software-e-seguranca/

https://www.splunk.com/en_us/blog/learn/software-liability.html

Pressman Bruce R. Maxim. Engenharia de Software. 9ª edição

Sommerville, Ian (2015). Engenharia de Software (10ª edição).
- Capítulo 4: "Requisitos de Software" (p. 98-120): Discussão sobre priorização de requisitos essenciais.
- Capítulo 24: "Gerenciamento de Qualidade" (p. 654-670): Conceito de equilíbrio entre qualidade e restrições práticas.

Pressman, Roger S. (2016). Engenharia de Software: Uma Abordagem Profissional (8ª edição).
- Capítulo 1: "A Natureza do Software" (p. 20-22): Crítica ao over-engineering.
- Capítulo 14: "Garantia de Qualidade de Software" (p. 381-400): Importância de testes focados no essencial.