Características de Qualidade de Software - ISO/IEC 25010:2023

Introdução

Este documento apresenta as características de qualidade de software conforme definidas na norma ISO/IEC 25010:2023, seguindo rigorosamente seus tópicos e subcategorias. A norma atualiza o modelo de qualidade de software da versão 2011, incorporando características e subcaracterísticas que atendem às demandas de tecnologias modernas, como computação em nuvem, sistemas distribuídos e aplicações críticas. Para cada característica e subcaracterística, são fornecidas definições oficiais da norma, explicações detalhadas, exemplos práticos, recomendações para implementação e sugestões de imagens ilustrativas, com o objetivo de tornar a normativa mais clara e acessível para desenvolvedores, gerentes de projeto e avaliadores de qualidade. O texto mantém um tom acadêmico, mas adota uma abordagem didática para facilitar a compreensão e aplicação prática.

As nove características principais de qualidade do produto, conforme a ISO/IEC 25010:2023, são:

1. Adequação funcional (*Functional suitability*)
2. Eficiência de desempenho (*Performance efficiency*)
3. Compatibilidade (*Compatibility*)
4. Capacidade de interação (*Interaction capability*)
5. Confiabilidade (*Reliability*)
6. Segurança (*Security*)
7. Manutenibilidade (*Maintainability*)
8. Flexibilidade (*Flexibility*)
9. Segurança operacional (*Safety*)

A seguir, cada característica é detalhada com base na norma, incluindo suas subcaracterísticas e orientações práticas.

1. Adequação Funcional (*Functional Suitability*)

Definição Geral

A adequação funcional é o grau em que um produto ou sistema fornece funções que atendem às necessidades declaradas e implícitas sob condições especificadas. Em resumo, é a capacidade do software de cumprir os requisitos funcionais, garantindo que as funções sejam completas, precisas e úteis.

Explicação Complementar

Essa característica é essencial para determinar se o software alcança seu propósito principal. Um sistema com funcionalidades ausentes, imprecisas ou pouco práticas compromete sua qualidade.

Por exemplo, um aplicativo de reservas de voos deve oferecer busca, reserva e cancelamento de passagens, com cálculos corretos e uma experiência intuitiva.

Subcaracterísticas

- **Completude funcional (*Functional completeness*)**: Grau em que o conjunto de funções cobre todas as tarefas e objetivos especificados.
    - **Exemplo**: Um sistema de reservas de hotel que permite reservar, cancelar e modificar reservas demonstra completude funcional. A ausência de uma dessas funções seria uma lacuna crítica.
- **Corretude funcional (*Functional correctness*)**: Grau em que o sistema fornece resultados corretos com a precisão necessária.
    - **Exemplo**: Em um software financeiro, o cálculo exato de juros compostos garante a corretude funcional.
- **Adequação das funções (*Functional appropriateness*)**: Grau em que as funções facilitam a realização de tarefas e objetivos.
    - **Exemplo**: Um editor de texto com botões intuitivos para negrito ou itálico melhora a experiência do usuário.

Recomendações Práticas

- Realize revisões colaborativas com *stakeholders* para identificar todas as funções necessárias.
- Aplique testes de aceitação e funcionais para validar completude e corretude.
- Use *user stories* ou diagramas de caso de uso para documentar requisitos.

2. Eficiência de Desempenho (*Performance Efficiency*)

Definição Geral

A eficiência de desempenho é o grau de desempenho de um sistema em relação aos recursos utilizados sob condições especificadas. Em termos simples, avalia a rapidez, eficiência e escalabilidade do software, minimizando o uso de memória, processamento e largura de banda.

Explicação Complementar

Um sistema eficiente melhora a experiência do usuário e reduz custos operacionais. Por exemplo, um aplicativo de streaming deve carregar vídeos rapidamente e suportar múltiplos usuários sem consumir recursos excessivos.

Subcaracterísticas

- **Comportamento temporal (*Time behaviour*)**: Grau em que os tempos de resposta e processamento atendem aos requisitos.
    - **Exemplo**: Um site de *e-commerce* que carrega em menos de 2 segundos.

- **Utilização de recursos (*Resource utilization*)**: Grau em que os recursos são usados eficientemente.
    - **Exemplo**: Um aplicativo móvel que consome pouca bateria e memória.
- **Capacidade (*Capacity*)**: Grau em que o sistema suporta cargas máximas, como número de usuários ou volume de dados.
    - **Exemplo**: Uma plataforma de streaming que mantém qualidade com milhares de usuários simultâneos.

Recomendações Práticas

- Use ferramentas de *profiling* (ex.: New Relic) para identificar gargalos.
- Implemente otimizações como *caching* e indexação de bancos de dados.
- Realize testes de carga e estresse para avaliar capacidade.

3. Compatibilidade (*Compatibility*)

Definição Geral

A compatibilidade é o grau em que um sistema pode trocar informações com outros sistemas ou operar no mesmo ambiente de hardware/software sem conflitos.

Explicação Complementar

Fundamental em arquiteturas como microserviços ou sistemas em nuvem, a compatibilidade evita interrupções e facilita integrações. Por exemplo, um sistema de gestão deve se conectar a APIs externas sem problemas.

Subcaracterísticas

- **Coexistência (*Co-existence*)**: Capacidade de operar em um ambiente compartilhado sem prejudicar outros sistemas.
    - **Exemplo**: Um software de monitoramento que não sobrecarrega um servidor compartilhado.
- **Interoperabilidade (*Interoperability*)**: Capacidade de trocar informações com outros sistemas de forma transparente.
    - **Exemplo**: Um sistema de saúde que integra dados via APIs padronizadas.

Recomendações Práticas

- Adote padrões abertos (ex.: REST, JSON).
- Teste a coexistência em ambientes simulados.
- Use *middleware* para gerenciar integrações.

4. Capacidade de Interação (*Interaction Capability*)

Definição Geral

A capacidade de interação é o grau em que um sistema pode ser usado com eficácia, eficiência, segurança e satisfação por usuários em contextos específicos.

Explicação Complementar

Evoluindo o conceito de usabilidade, essa característica foca em inclusão e engajamento. Um aplicativo de aprendizado, por exemplo, deve ser intuitivo, acessível e envolvente.

Subcaracterísticas

- **Reconhecibilidade da adequação (*Appropriateness recognizability*)**: Grau em que os usuários identificam a utilidade do sistema.
  - **Exemplo**: Um site de compras com categorias claras na página inicial.
- **Facilidade de aprendizado (*Learnability*)**: Grau em que os usuários aprendem a usar o sistema rapidamente.
  - **Exemplo**: Um aplicativo com interface intuitiva.
- **Operabilidade (*Operability*)**: Grau em que o sistema é fácil de operar.
  - **Exemplo**: Um botão "desfazer" visível.
- **Proteção contra erro do usuário (*User error protection*)**: Grau em que o sistema minimiza erros.
  - **Exemplo**: Confirmação antes de excluir arquivos.
- **Engajamento do usuário (*User engagement*)**: Grau em que o sistema mantém o interesse.
  - **Exemplo**: Gamificação em um aplicativo de aprendizado.
- **Inclusividade (*Inclusivity*)**: Grau em que o sistema é acessível a todos.
  - **Exemplo**: Suporte a leitores de tela.
- **Assistência ao usuário (*User assistance*)**: Grau em que o sistema oferece suporte.
  - **Exemplo**: Tutoriais integrados.
- **Autoexplicatividade (*Self-descriptiveness*)**: Grau em que o sistema se explica sozinho.
  - **Exemplo**: Ícones intuitivos.

Recomendações Práticas

- Teste usabilidade com grupos diversos.
- Siga diretrizes do WCAG.
- Forneça feedback visual/auditivo.

5. Confiabilidade (*Reliability*)

Definição Geral

A confiabilidade é o grau em que um sistema executa funções especificadas por um período determinado sob condições definidas.

Explicação Complementar

Crucial para sistemas críticos (ex.: bancários, hospitalares), a confiabilidade garante operação contínua e confiança do usuário.

Subcaracterísticas

- **Ausência de falhas (*Faultlessness*)**: Grau em que o sistema opera sem erros.
  - **Exemplo**: Um servidor funcionando por semanas sem travar.
- **Disponibilidade (*Availability*)**: Grau em que o sistema está operacional quando necessário.
  - **Exemplo**: Um site com 99,9% de *uptime*.
- **Tolerância a falhas (*Fault tolerance*)**: Grau em que o sistema resiste a falhas parciais.
  - **Exemplo**: Servidores redundantes.
- **Recuperabilidade (*Recoverability*)**: Grau em que o sistema se restaura após falhas.
  - **Exemplo**: Banco de dados que recupera transações após quedas.

Recomendações Práticas

- Monitore com ferramentas como Prometheus.
- Use arquiteturas redundantes.
- Realize backups regulares.

6. Segurança (*Security*)

Definição Geral

A segurança é o grau em que um sistema protege dados e funções contra acessos não autorizados.

Explicação Complementar

Vital para sistemas com dados sensíveis, a segurança protege privacidade e integridade em um cenário de ameaças cibernéticas.

Subcaracterísticas

- **Confidencialidade (*Confidentiality*)**: Proteção contra acesso não autorizado.
  - **Exemplo**: Dados criptografados em bancos.
- **Integridade (*Integrity*)**: Garantia de dados inalterados.
  - **Exemplo**: Transações financeiras imutáveis.
- **Não repúdio (*Non-repudiation*)**: Atribuição inequívoca de ações.

    - **Exemplo**: Assinaturas digitais.
- **Responsabilidade (*Accountability*)**: Registro de ações.
    - **Exemplo**: Logs de auditoria.
- **Autenticidade (*Authenticity*)**: Verificação de identidade.
    - **Exemplo**: Autenticação multifator.
- **Resistência (*Resistance*)**: Proteção contra ataques.
    - **Exemplo**: Bloqueio de SQL injection.

Recomendações Práticas

- Use criptografia AES-256.
- Implemente MFA.
- Realize testes de penetração.

7. Manutenibilidade (*Maintainability*)

Definição Geral

A manutenibilidade é o grau de facilidade com que um sistema pode ser modificado.

Explicação Complementar

Impacta a longevidade e os custos, permitindo adaptações rápidas a novos requisitos.

Subcaracterísticas

- **Modularidade (*Modularity*)**: Componentes independentes.
    - **Exemplo**: Frontend e backend separados.
- **Reusabilidade (*Reusability*)**: Uso em outros contextos.
    - **Exemplo**: Componente de autenticação reutilizável.
- **Analisabilidade (*Analysability*)**: Facilidade de diagnóstico.
    - **Exemplo**: Código documentado.
- **Modificabilidade (*Modifiability*)**: Alterações sem defeitos.
    - **Exemplo**: Adição de campos em formulários.
- **Testabilidade (*Testability*)**: Facilidade de testes.
    - **Exemplo**: Testes unitários automatizados.

Recomendações Práticas

- Use princípios SOLID.
- Adote controle de versão (ex.: Git).
- Implemente CI/CD.

8. Flexibilidade (*Flexibility*)

Definição Geral

A flexibilidade é o grau em que um sistema se adapta a contextos além dos inicialmente especificados.

Explicação Complementar

Essencial em ambientes dinâmicos, como sistemas em nuvem, suporta mudanças com mínimo esforço.

Subcaracterísticas

- **Adaptabilidade (*Adaptability*)**: Ajuste a diferentes contextos.
  - **Exemplo**: ERP configurável.
- **Escalabilidade (*Scalability*)**: Ajuste de capacidade.
  - **Exemplo**: Suporte a mais usuários com novos servidores.
- **Instalabilidade (*Installability*)**: Facilidade de instalação.
  - **Exemplo**: Assistente gráfico de instalação.
- **Substituibilidade (*Replaceability*)**: Troca de componentes.
  - **Exemplo**: Substituição por OAuth2.

Recomendações Práticas

- Use contêineres (ex.: Docker).
- Adote infraestrutura como código.
- Defina interfaces claras.

9. Segurança Operacional (*Safety*)

Definição Geral

A segurança operacional é o grau em que um sistema mitiga riscos a pessoas, propriedades ou ao meio ambiente.

Explicação Complementar

Nova na versão 2023, é crítica para sistemas físicos (ex.: médicos, automotivos), prevenindo danos.

Subcaracterísticas

- **Restrição operacional (*Operational constraint*)**: Respeito a limites seguros.
  - **Exemplo**: Robô que para se exceder força.
- **Identificação de riscos (*Risk identification*)**: Reconhecimento de riscos.
  - **Exemplo**: Drone que detecta obstáculos.

- **Falha segura (*Fail safe*)**: Estado seguro em falhas.
    - **Exemplo**: Sistema médico que para em erros.
- **Sinalização de perigos (*Hazard warning*)**: Alertas claros.
    - **Exemplo**: Painel com alertas sonoros.
- **Integração segura (*Safe integration*)**: Integração sem riscos.
    - **Exemplo**: Freios ABS integrados.

Recomendações Práticas

- Realize análises de risco.
- Implemente paradas de emergência.
- Siga normas como ISO 26262.

Conclusão

Este documento oferece uma análise detalhada da ISO/IEC 25010:2023, combinando rigor técnico com clareza didática. Serve como recurso para documentação, treinamento e avaliação de qualidade em projetos de software.

Citações

- ISO/IEC 25010:2023 - *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model*. Disponível em: https://www.iso.org/standard/78176.html.
- Informações complementares baseadas em fontes públicas sobre a norma.