[Confiabilidade]:

Disponibilidade,
Tolerância a falhas,
Recuperabilidade

Equipe:
Mateus Araújo – 500924
Rayan Victor – 476676
Kellen Raizy – 548328
Levi de Castro – 511097
Juvenal Lavres – 485170
Adriano Mendes – 508201
Jânio Lima - 519325

Introdução: Norma ISO/IEC 25010 e a importância da qualidade do software

- A norma define critérios de qualidade do software (funcionalidade, confiabilidade, usabilidade, eficiência, segurança, manutenibilidade e portabilidade).
- A qualidade do software é importante porque um software de alta qualidade é mais confiável, seguro, fácil de usar e manter.
- A funcionalidade se refere à capacidade do software de fornecer as funcionalidades que foram especificadas e atender às necessidades do usuário.
- A segurança se refere à capacidade do software de proteger as informações do usuário e garantir que os dados não sejam comprometidos.
- A usabilidade se refere à facilidade de uso do software, e um software com alta usabilidade é intuitivo e fácil de usar para os usuários finais.

Característica de qualidade: Confiabilidade

- A confiabilidade é a capacidade do software de realizar suas funções corretamente em um determinado período de tempo e em condições específicas.
- A confiabilidade é importante para evitar falhas e erros que possam ter consequências graves, como perda de dados e riscos à segurança.
- A confiabilidade pode ser medida por meio de métricas como MTBF, MTTR e taxa de falhas por hora.
- Os testes de confiabilidade podem avaliar o software em condições extremas, como altas temperaturas ou cargas de trabalho pesadas.
- A confiabilidade pode ser melhorada por meio de técnicas como redundância, monitoramento contínuo e análise de falhas.

Subcaracterística: Maturidade

- A maturidade é a capacidade do software de evitar erros e falhas durante o uso normal.
- Um software imaturo pode levar a problemas como perda de dados e tempo de inatividade do sistema.
- A maturidade pode ser medida por meio de métricas como a taxa de falhas e correção de defeitos.
- Os testes de maturidade avaliam a capacidade do software de evitar erros e falhas em condições normais de uso.
- A maturidade pode ser melhorada com técnicas como testes de unidade, boas práticas de codificação e revisão de código.

Exemplos de softwares que atendem (ou não) à subcaracterística de Maturidade

- Exemplos de softwares maduros incluem o sistema operacional Windows, o navegador Google Chrome e o software de processamento de texto Microsoft Word.
- O sistema operacional Windows Vista e o navegador Internet Explorer 6 são exemplos de softwares que não atendem à subcaracterística de Maturidade.
- Um software maduro é confiável e estável, enquanto um software imaturo pode ter bugs e problemas de segurança e desempenho.
- A maturidade de um software pode ser medida por meio de métricas como a taxa de falhas e correção de defeitos.
- É importante que os desenvolvedores de software trabalhem continuamente para melhorar a maturidade de seus produtos, por meio de técnicas como testes de unidade, boas práticas de codificação e revisão de código.

Métricas de confiabilidade: como medir a maturidade do software

- Taxa de falhas: mede a frequência de falhas no software durante o uso normal.
- Tempo médio entre falhas (MTBF): mede o tempo médio que o software funciona corretamente antes de ocorrer uma falha.
- Tempo médio para reparo (MTTR): mede o tempo médio que é necessário para corrigir uma falha de software.
- Taxa de sucesso: mede a proporção de transações de software bem-sucedidas em relação ao total de transações.
- Nível de severidade das falhas: mede o impacto das falhas no software, que pode variar de levemente irritante a catastrófico.

Métodos de avaliação: inspeção, revisão, teste e auditoria para avaliar a maturidade do software

- Inspeção: revisão detalhada do código fonte e documentação do software para identificar problemas e melhorias necessárias.
- Revisão: análise crítica do software e seus componentes para identificar problemas e melhorias necessárias.
- Teste: execução do software com diferentes entradas e condições para identificar problemas e melhorias necessárias.
- Auditoria: revisão completa do software e seus processos para garantir a conformidade com padrões, regulamentações e requisitos específicos.
- Combinação de métodos: uso de uma combinação de métodos para obter uma avaliação mais abrangente e precisa da maturidade do software.

Subcaracterística: Tolerância a falhas

- Resiliência: capacidade do software de lidar com falhas e se recuperar sem interrupções significativas.
- Redundância: recursos redundantes para garantir a continuidade do sistema mesmo em caso de falha.
- Segurança: proteção de dados e sistema contra ameaças externas.
- Prevenção de falhas: detecção precoce de erros e implementação de medidas para corrigi-los antes que se tornem críticos.
- Tempo de recuperação: tempo necessário para o software se recuperar de uma falha e voltar a funcionar normalmente.

Exemplos de softwares que atendem (ou não) à subcaracterística de Tolerância a falhas

- Google Docs: altamente tolerante a falhas
- Netflix: altamente tolerante a falhas
- Twitter: historicamente não era muito tolerante a falhas, mas melhorou nos últimos anos
- Microsoft Windows: historicamente propenso a falhas, mas tem melhorado a tolerância a falhas em versões mais recentes
- Adobe Photoshop: não é particularmente tolerante a falhas, mas inclui recursos como salvamento automático para minimizar o risco de perda de trabalho.

Métricas de confiabilidade: como medir a tolerância a falhas do software

- MTTF (Mean Time To Failure): mede o tempo médio que leva para o software falhar.
- MTTR (Mean Time To Repair): mede o tempo médio que leva para reparar uma falha.
- Taxa de falhas: mede a frequência com que o software falha em um determinado período de tempo.
- Taxa de sucesso: mede a proporção de vezes que o software executa corretamente em relação ao número total de tentativas.
- Testes de carga: simula a quantidade máxima de usuários que o software pode suportar antes de falhar.

Métodos de avaliação: inspeção, revisão, teste e auditoria para avaliar a tolerância a falhas do software

- Inspeção: revisores experientes analisam o código-fonte em busca de erros, vulnerabilidades e outras falhas de design
- Revisão: profissionais de qualidade avaliam a qualidade do software, observando características como usabilidade, confiabilidade e segurança
- Teste de software: técnica de avaliação que verifica se o software está funcionando conforme o esperado, pode ser manual ou automatizada
- Auditoria: processo formal de avaliação de software que envolve revisão detalhada da documentação, código-fonte e processos de desenvolvimento
- Análise de risco: técnica que avalia os riscos associados ao uso do software e como esses riscos podem afetar o usuário final

Subcaracterística: Recuperabilidade

- Mecanismos de backup
- Detecção de falhas
- Tempo de recuperação
- Verificação de integridade
- Tolerância a falhas

Exemplos de softwares que atendem (ou não) à subcaracterística de Recuperabilidade

- Sistemas de gerenciamento de banco de dados que possuem recursos de backup e recuperação de dados.
- Sistemas operacionais que possuem mecanismos de proteção contra falhas, como reinicialização automática após uma falha.
- Um software que não faz backup de dados automaticamente e não permite a recuperação dos mesmos em caso de falha.
- Um software que não detecta falhas e erros, deixando o usuário sem informação sobre o que pode ter ocorrido.
- Um software que demora muito tempo para se recuperar após uma falha, impactando significativamente o usuário e o negócio

Métricas de confiabilidade: como medir a recuperabilidade do software

- Tempo médio de recuperação (MTTR)
- Tempo máximo de inatividade (MTDU)
- Frequência de falhas
- Taxa de sucesso de recuperação
- Testes de recuperação

Métodos de avaliação: inspeção, revisão, teste e auditoria para avaliar a recuperabilidade do software

- Inspeção: análise visual do código para identificar pontos de falha e melhorias na estrutura do sistema.
- Revisão: análise detalhada da arquitetura do software para identificar problemas na recuperação de falhas.
- Testes de recuperação: simulação de falhas para verificar a capacidade de recuperação do software.
- Auditoria: análise profunda da estrutura e funcionamento do sistema para verificar conformidade com normas e padrões.
- Esses métodos ajudam a identificar problemas na recuperação de falhas e oportunidades de melhoria.

Conclusão: Importância da confiabilidade do software e resumo dos principais pontos.

- A confiabilidade do software é uma das características mais importantes para garantir a satisfação do usuário e o sucesso do produto.
- A Norma ISO/IEC 25010 define a confiabilidade como a capacidade do software de manter seu desempenho sob condições específicas por um período de tempo determinado.
- As subcaracterísticas da confiabilidade incluem maturidade, tolerância a falhas e recuperabilidade.
- Métricas como MTBF (Mean Time Between Failures) e MTTR (Mean Time To Recover) podem ser usadas para medir a confiabilidade e a recuperabilidade do software.
- Métodos de avaliação, como inspeção, revisão, teste e auditoria, são essenciais para garantir que o software atenda aos requisitos de confiabilidade.

Referências

- ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.
- ISO/IEC 9126-1:2001 Software engineering -- Product quality -- Part 1: Quality model.
- ISO/IEC TR 9126-3:2003 Software engineering -- Product quality -- Part 3: Internal metrics.
- M. Freire, A. Amaral, and R. Abreu, “An empirical study of ISO/IEC 25010 usability criteria,” Empirical Software Engineering, vol. 23, no. 2, pp. 1129–1168, Apr. 2018.