Crise do Software e Características de Qualidade

Prof. Me. Lucas Bruzzone

Aula 02

O que foi a Crise do Software?

Contexto Histórico - Década de 1960-1970

Período em que o desenvolvimento de software enfrentou problemas sistemáticos de prazo, orçamento e qualidade

- **Origem do termo**: NATO Software Engineering Conference (1968)
- **Transição**: Hardware barato $\rightarrow$ Software caro
- **Complexidade**: Sistemas maiores e mais complexos
- **Demanda**: Crescimento exponencial por software

Problemas Identificados

Gestão de Projetos

- Atrasos constantes
- Estouros de orçamento
- Projetos cancelados
- Falta de planejamento

Qualidade do Software

- Software não confiável
- Muitos bugs
- Dificuldade de manutenção
- Não atendia requisitos

Casos Famosos de Falhas

- **IBM System/360 OS**: Atrasos de anos, custos 10x maiores
- **Ariane 5 (1996)**: Explosão por erro de conversão de dados
- **Mars Climate Orbiter (1999)**: Perdido por erro de unidades
- **Therac-25**: Overdoses de radiação por bugs de software

Consequências: Mortes, perdas financeiras bilionárias, perda de confiança

Resposta à Crise

- **Engenharia de Software**: Disciplina para desenvolvimento sistemático
- **Metodologias**: Estruturação do processo de desenvolvimento
- **Ferramentas CASE**: Computer-Aided Software Engineering
- **Padrões**: IEEE, ISO, CMM
- **Métricas**: Medição e controle de qualidade
- **Gerenciamento**: Técnicas de gestão de projetos

O que é Qualidade de Software?

Definição

Grau em que um produto de software satisfaz requisitos especificados e necessidades do usuário

Perspectivas de Qualidade:

- **Usuário**: Software faz o que preciso?
- **Desenvolvedor**: Código é manutenível?
- **Gerente**: Projeto no prazo e orçamento?
- **Cliente**: Vale o investimento?

Características de Qualidade - ISO/IEC 25010

Qualidade em Uso

- Eficácia
- Eficiência
- Satisfação
- Liberdade de risco
- Cobertura de contexto

Qualidade do Produto

- Adequação funcional
- Confiabilidade
- Usabilidade
- Eficiência de desempenho
- Manutenibilidade
- Portabilidade

Funcionalidade

- **Adequação funcional**: Funções facilitam objetivos
- **Correção**: Resultados precisos
- **Interoperabilidade**: Interage com outros sistemas
- **Segurança**: Proteção de dados e sistema
- **Conformidade**: Aderência a padrões

"O software faz o que deveria fazer?"

Confiabilidade

- **Maturidade**: Baixa frequência de falhas
- **Tolerância a falhas**: Funciona mesmo com problemas
- **Recuperabilidade**: Recupera dados após falha
- **Disponibilidade**: Sistema operacional quando necessário

"O software funciona bem por quanto tempo?"

Usabilidade

- **Inteligibilidade**: Fácil de entender
- **Apreensibilidade**: Fácil de aprender
- **Operacionalidade**: Fácil de operar
- **Atratividade**: Interface agradável
- **Acessibilidade**: Uso por pessoas com deficiências

"O software é fácil de usar?"

Eficiência e Manutenibilidade

Eficiência

- Comportamento temporal
- Utilização de recursos
- Capacidade

Manutenibilidade

- Analisabilidade
- Modificabilidade
- Estabilidade
- Testabilidade

Eficiência: "Rápido e econômico?"
Manutenibilidade: "Fácil de modificar?"

Portabilidade

- **Adaptabilidade**: Adaptação a diferentes ambientes
- **Capacidade de instalação**: Fácil instalação
- **Coexistência**: Compartilha recursos com outros
- **Capacidade de substituição**: Substitui outro software

"O software funciona em diferentes ambientes?"

Trade-offs em Qualidade

- **Performance vs. Manutenibilidade**: Código otimizado pode ser complexo
- **Funcionalidade vs. Usabilidade**: Muitas funções podem confundir
- **Segurança vs. Usabilidade**: Segurança pode reduzir facilidade de uso
- **Portabilidade vs. Performance**: Código genérico pode ser mais lento

Não existe software perfeito em todas as características!

Exercício - Análise de Trade-offs

Situação: Desenvolvimento de um aplicativo bancário móvel

Conflitos identificados:

- Segurança máxima vs. Facilidade de uso
- Performance vs. Funcionalidades avançadas
- Portabilidade vs. Interface nativa otimizada
- Confiabilidade vs. Velocidade de desenvolvimento
- Interface simples vs. Funcionalidades completas

Tarefa: Para cada conflito, justifique qual característica você priorizaria e por quê. Proponha soluções de compromisso.

Exemplo de Resposta

Segurança vs. Facilidade de uso

Prioridade escolhida: Segurança

Justificativa: Aplicativo bancário precisa proteger dados financeiros dos usuários.

Solução de compromisso: Usar biometria (digital/face) para tornar a segurança mais rápida e fácil para o usuário.

Próxima Aula

Modelos de Processo de Software

Estudaremos os modelos cascata e incremental para estruturar o desenvolvimento