Engenharia de Software
01 - Introdução

Márcio Daniel Puntel
marciopuntel@ulbra.edu.br

Engenharia de Software

- Nações desenvolvidas dependem de software;
- Cada vez mais sistemas são controlados por software;
  - Engenharia de Software preocupa-se com as teorias, os métodos e as ferramentas para o desenvolvimento profissional de software.

Conceito

"... é a ciência e a arte de especificar, projetar, implementar e manter atualizados e corretos, com economia, em tempo útil e de forma elegante, programas, documentação e procedimentos operacionais para sistemas computacionais de utilidade para a humanidade."

A. Brown, A. Earl e J. McDermid

Conceito

"... é o estabelecimento e o uso de um conjunto de princípios de engenharia com o objetivo de construir software confiável, eficiente e economicamente viável em máquinas reais."

Bauer

História

Antes 1969

- desenvolvimento fora de controle;
- surge termo ENGENHARIA DE SOFTWARE;

1969-1971

- desenvolvimento top-down e modularizado;
- novas linguagens de programação;
- novas técnicas de trabalho em grupo;

História

1972-1973

- programação estruturada;
- propostas de apoio ao Desenvolvimento e Gerência;

1974-1975

- conceitos de confiabilidade e controle da qualidade (teste sistemático, prova formal, tolerância a falhas);

1976-1977

- foco em requisitos, especificação e projeto;
- integração e validação das fases do ciclo de vida;

História

1978-1980

- uso de ferramentas automatizadas;
- cursos de engenharia de software;

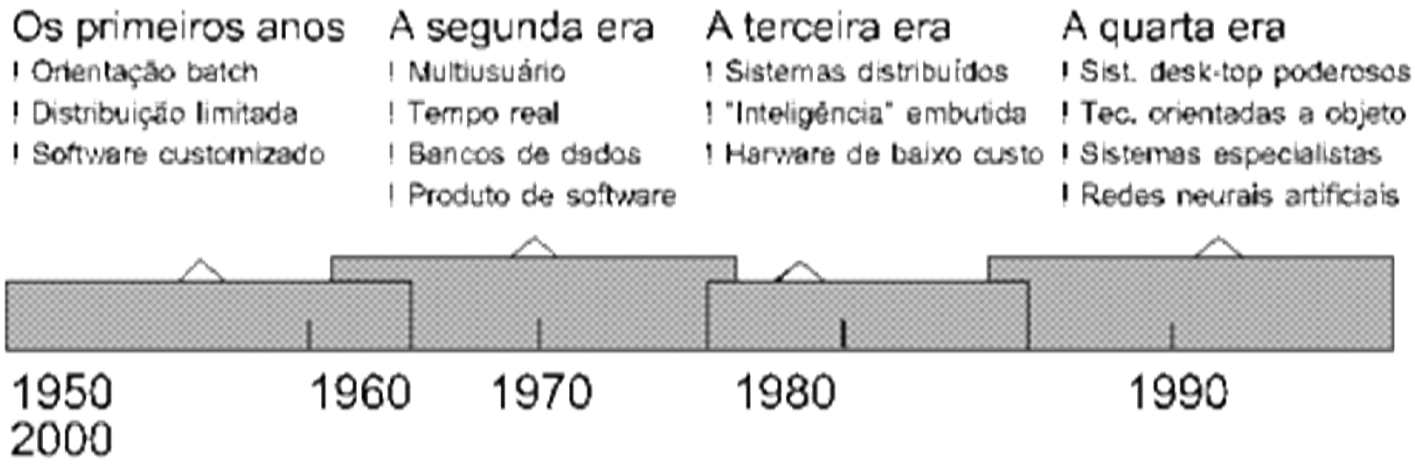
1980-1989

- era CASE;

1999 em diante

- técnicas de IA aplicas a Engenharia de Software, gerando as “ferramentas inteligentes”;
- era QUALIDADE.

Evolução do software



Fonte: http://www.batebyte.pr.gov.br

Por quê ES...

- aumentar a qualidade dos produtos;
- aumentar a produtividade do processo de desenvolvimento;
- diminuir os custos do processo de desenvolvimento;
- maior confiabilidade nos prazos estabelecidos.

Software com qualidade

- É aquele que satisfaz os requisitos solicitados.
- Deve ser fácil de manter, ter boa performance, ser confiável e fácil de usar.
- Alguns atributos:
  - **Manutenibilidade:** o software deve evoluir para atender os requisitos que mudam;
  - **Eficiência:** o software não deve desperdiçar os recursos do sistema;
  - **Usabilidade:** o software deve ser fácil de usar pelos usuários para os quais ele foi projetado.

Crise do Software

- 25% dos projetos são cancelados
- o tempo de desenvolvimento é bem maior do que o estimado;
- 75% dos sistemas não funcionam como planejado;
- a manutenção e reutilização são difíceis;
- os problemas são proporcionais a complexidade dos sistemas.

Crise: motivos

- Falta de envolvimento do usuário;
- Análise e projeto inadequados;
- Falta de flexibilidade no projeto;
- Prazos longos;
- Elevada rotatividade de pessoal;
- Má qualidade dos métodos, linguagens, ferramentas e processos;

Crise: motivos

- Velocidade da mudança tecnológica;
- Dificuldade de formalização;
- Velocidade na mudança dos mercados;
- Velocidade na obsolescência dos sistemas;
- Incapacidade das estruturas de SI para acompanhar a velocidade das mudanças.

Crise: perguntas

Por que os custos são tão elevados?

- Não há controle sobre prazos ou planejamento sobre equipes e recursos;
- O levantamento de requisitos não é realizado de forma integrada com o cliente e dentro de padrões antecipadamente projetados;

Crise: perguntas

Por que os cronogramas não são cumpridos?

- Falta de interação entre usuário e analista;
- Baixo conhecimento/descrição do problema;
- Nível de da experiência da equipe de desenvolvimento;

Crise: perguntas

Por que é difícil medir o progresso durante o desenvolvimento?

- Porque não há referências. Há falta de um plano com metas e prazos definidos;

Por que os erros não são detectados antes da liberação das versões?

- Controle de qualidade não é objetivo desde o princípio do desenvolvimento;

Crise: perguntas

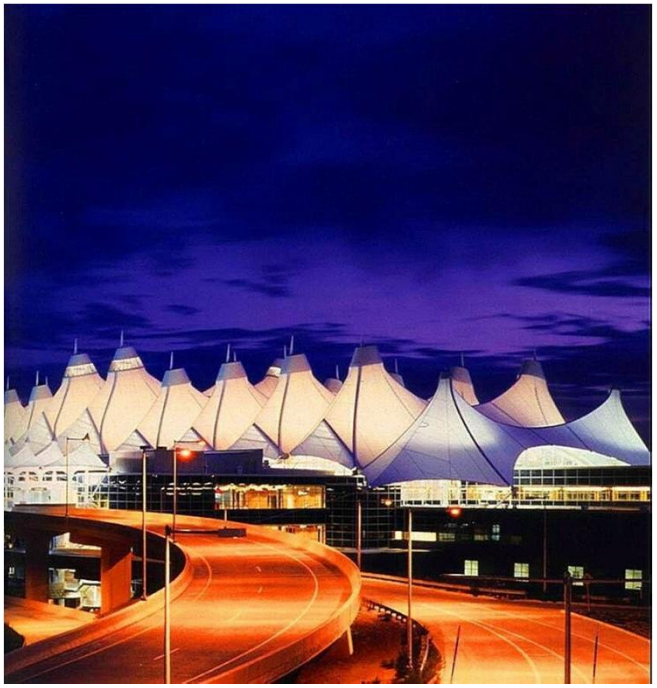
Por que a produtividade é baixa?

- Aprendizado dos usuários ao longo do projeto;
- Alterações dos requisitos provocadas por mudanças de tecnologia, legislação ou mercado;
- Implicações para a implementação somente percebidas ao tempo da implementação.

Crise do software - exemplos

Denver International Airport

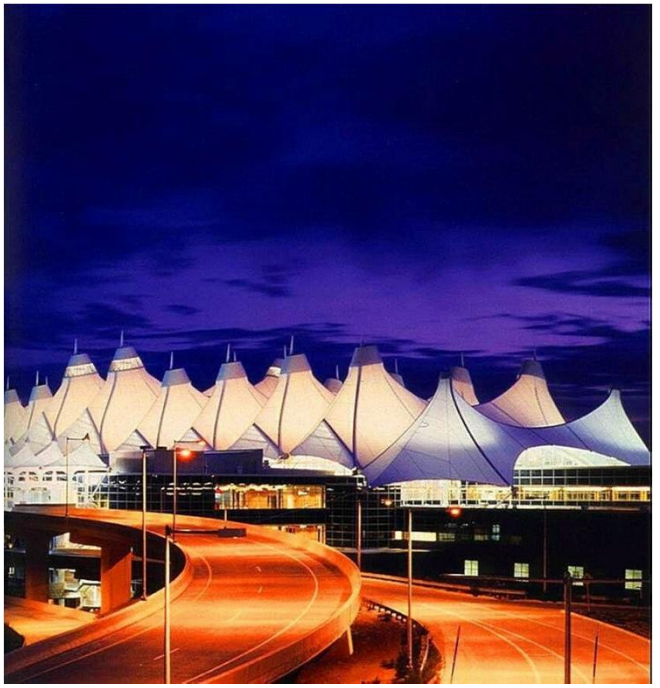
- Custo projeto U$ 4,9 bi
- 100 mil passageiros/dia
- 1.200 voos
- 94 portões (in/out)



Crise do software - exemplos

Denver International Airport

- Erro no sistema automático de bagagens;
- Atraso na abertura
  U$ 360 milhões
- U$ 86 milhões para
  consertar sistema



Crise do software - exemplos

Ariane 5

- Projeto da Agência Espacial Europeia que custou:
- 10 anos;
- U$ 8 bi

Crise do software - exemplos

Ariane 5

- Explosão 40 segundos após decolagem;
- Destruição do foguete e perda da carga avaliada em U$ 500 milhões;

Crise do software - exemplos

- Ariane 5 – o que aconteceu?
  - Fato: o veículo detonou suas cargas explosivas de autodestruição e explodiu no ar. Por que?
  - Porque ele estava se quebrando devido às forças aerodinâmicas. Mas por que?
  - O foguete tinha perdido o controle de direção (altitude). Causa disso?
  - O computador reserva e principal desligaram ao mesmo tempo;

Crise do software - exemplos

- Ariane 5 – o que aconteceu?
  - Por que o shutdown? Ocorrera um runtime error e os computadores se desligaram. De onde veio esse erro?
  - **Um programa que convertia um valor em ponto flutuante para inteiro de 16 bits recebeu como entrada um valor que estava fora da faixa permitida.**

Crise do software - exemplos

- Ariane 5 – Ironia...
  - O resultado dessa conversão não era mais necessário após a decolagem.

Elementos vs. Atividades

Elementos

- Modelos do ciclo de vida
- Linguagens
- Métodos
- Ferramentas
- Processos

Atividades

- Modelagem do negócio
- Elicitação de requisitos
- Análise e Projeto
- Implementação
- Testes
- Distribuição
- Planejamento
- Gerenciamento
- Gerência de Configuração e Mudanças
- Manutenção

Ciclo de vida

- Uma representação abstrata e simplificada do processo de desenvolvimento software, tipicamente mostrando as principais atividades e dados usados na produção e manutenção de software

Modelos de ciclo de vida

- Cascata
- Espiral
- Incremental
- Sequencial linear
- Prototipação

Linguagem

- Notação com sintaxe e semântica bem definidas;
- Com representação gráfica ou textual;
- Usada para descrever os artefatos gerados durante o desenvolvimento de software;
- Exemplos: UML, Java.

Métodos

- Descrição sistemática de como deve-se realizar uma determinada atividade ou tarefa;
- A descrição é normalmente feita através de padrões e guias;
- Exemplos: Método para descoberta das classes de análise no RUP.

Ferramentas

- Provê suporte computacional a um determinado método ou linguagem;
- Ambiente de desenvolvimento: conjunto de ferramentas integradas (CASE);
- Exemplos: Rational Rose, JBuilder, Entreprise Architect, Er-win.

Processo

Conjunto de atividades:

- bem definidas;
- com responsáveis;
- com artefatos de entrada e saída;
- com dependências entre as mesmas e ordem de execução;
- com modelo de ciclo de vida.

Processo de software

- Um conjunto de atividades cujo objetivo é o desenvolvimento ou evolução do software;
- Conjunto coerente de atividades para especificação, projeto, implementação e teste de sistemas de software;
- Atividades que são envolvidas no desenvolvimento de produtos de software.

Referências

- PRESSMAN, Roger S.; **Engenharia de Software**. São Paulo: Pearson, 2006.
- SOMMERVILLE, Ian; **Engenharia de Software**. São Paulo: Pearson, 2004.
- KOTONYA, Gerald; SOMMERVILLE, Ian; **Requeriments Engineering: Processes and Tecniques**. John Wiley & Sons, 1998.