Engenharia de Software: Conceitos Fundamentais

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo
dcc603@gmail.com

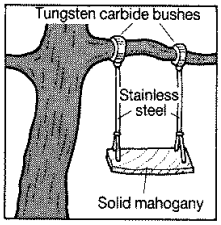
07 Março 2012

Tópicos da Aula

- Motivação e Conceitos Fundamentais
- Engenharia de Software
  - Visão em Camadas
- Atividades do desenvolvimento de software
- Evolução de Software
  - Leis de Lehman
- Revisão

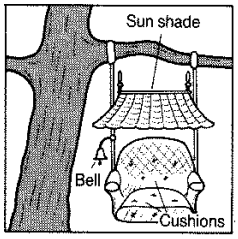
Falha de Comunicação no Desenvolvimento de Software

O que é anunciado



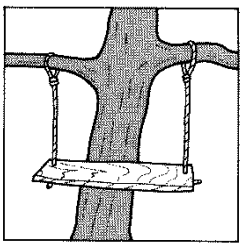
What Product Marketing specified

O que o vendedor promete



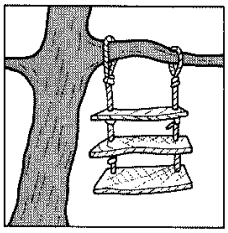
What the salesman promised

O projeto inicial



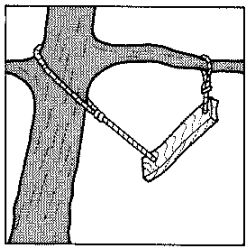
Design group's initial design

O projeto revisado do arquiteto



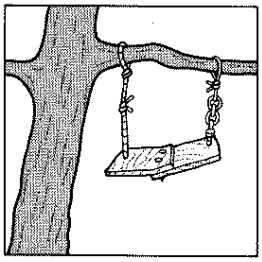
Corp. Product Architecture's modified design

A primeira versão



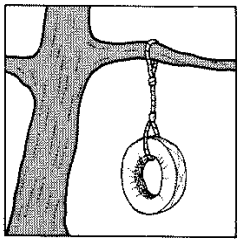
Pre-release version

A versão entregue ao cliente



General release version

O que o cliente queria



What the customer actually wanted

Motivação e Conceitos Fundamentais

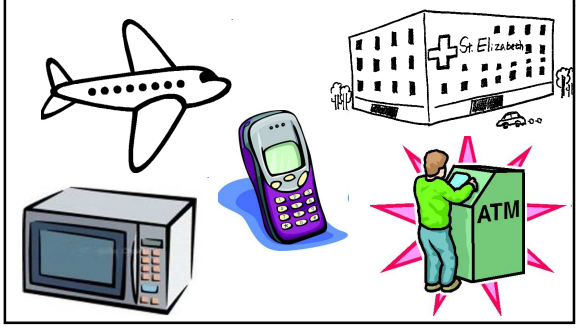
O que é software?

- Programa de computador + Documentação
- Classificação fundamental
  - Produtos genéricos (ex. MS Office)
  - Produtos encomendados (ex. Software de Controle da Locadora do Zé)

Crise do Software (1968)

- Custos de hardware caindo
  Custos do software subindo
- Avanços em hardware
  - Permitem desenvolvimento de sistemas cada vez mais complexos
- Resultado (software)
  - Custos altos, projetos atrasados, sistemas não confiáveis, desempenho insatisfatório, etc...

Software está em todo lugar



Desafios de Produzir Software

Confiabilidade

- Exemplo: Vôo Air France Rio - Paris
  1. Dados conflitantes (falha nos sensores)
  2. Sistema assume o controle (piloto automático)
  3. Piloto tenta reiniciar o sistema (boot)
  4. Em 4 minutos o avião mergulha no oceano

The Last Four Minutes of Air France Flight 447.
http://www.spiegel.de/international/world/0,1518,679980,00.html

Desafios de Produzir Software

Preço e desempenho

Celular

- Pouco espaço na memória
- Grande variação em características de aparelhos

Desafios de Produzir Software

Sistemas Críticos

- Equipamentos médicos
  - Extremamente críticos
  - Lidam com vidas

- Caixas eletrônicos
  - Prejuízos financeiros

Em Resumo...

- O desenvolvimento informal de software não é suficiente
  - Técnicas e métodos são necessários
- Algumas dificuldades
  - Heterogeneidade
  - Confiabilidade
  - Prazo de entrega
  - Mudança contínua

O que é Engenharia de Software?

A Engenharia de Software é uma disciplina de engenharia relacionada a todos os aspectos de produção de software. Ian Sommerville

- Foco no desenvolvimento de software de **alta qualidade** dentro de **custos adequados**.
  - Atender necessidades do cliente

E as outras engenharias?

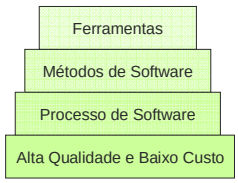
- O que difere Engenharia de Software de outras engenharias?
  - Software é desenvolvido, não fabricado
  - Software não se desgasta
  - Software é geralmente produzido para um cliente específico

Engenharia de Software: Visão Geral

Roger Pressman. Engenharia de Software, 6ª Edição. McGraw-Hill, 2006.

Eng. de Software em Camadas

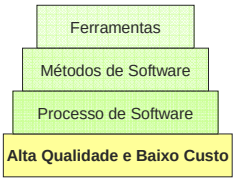
- A engenharia de software inclui
  - Processo
  - Métodos
  - Ferramentas



- Pode ser organizada em camadas

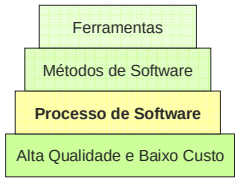
Qualidade de Software

- Atributos de um bom software
  - Facilidade de manutenção
  - Confiança
  - Eficiência
  - Usabilidade, etc.



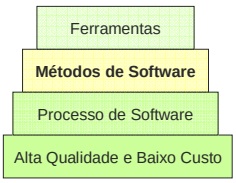
Processo de Software

- Conjunto de atividades (e seus resultados) cujo objetivo é o desenvolvimento de software
- O processo oferece estabilidade, controle e organização no ciclo de desenvolvimento
- Atividades principais
  - Especificação
  - Desenvolvimento
  - Validação
  - Evolução



Métodos de Software

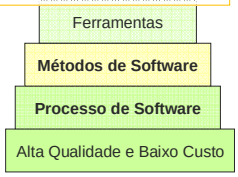
- Abordagens estruturadas para desenvolvimento de software
- Os métodos fornecem a técnica de como fazer
- Métodos incluem
  - Modelos
  - Notações
  - Regras, etc.



Métodos de Software

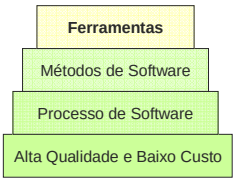
Veremos mais detalhes sobre métodos e processos de software nas próximas aulas

- Métodos incluem
  - Modelos
  - Notações
  - Regras, etc.



Ferramentas

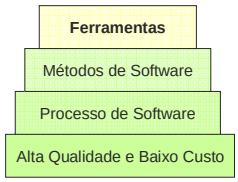
- Fornecem apoio automatizado (ou semi-automatizado) para o processo e para os métodos
- Exemplo: ferramentas de modelagem do processo
  - Permitem definir ações, tarefas, produtos, etc. de um modelo de processo



Ferramentas

Veremos algumas ferramentas nas aulas em laboratório

- Exemplo: ferramentas de modelagem do processo
  - Permitem definir ações, tarefas, produtos, etc. de um modelo de processo



Desenvolvimento de Software

Atividades Comuns

1. Especificação de requisitos
2. Projeto de Software
3. Implementação
4. Validação do software
5. Evolução de software

Atividades de Desenvolvimento

1. **Especificação de requisitos**
2. Projeto de Software
3. Implementação
4. Validação do software
5. Evolução de software

Especificação de Requisitos

- Um sistema de software deve satisfazer as necessidades de seus usuários
  - Tais necessidades são expressas na forma de **requisitos**
- **Requisito** = ação que deve ser executada pelo sistema
  - *Ex: registrar as notas dos alunos, calcular a média final, verificar aprovação, etc.*

Especificação de Requisitos

- Inclui quatro fases principais
  - Estudo de viabilidade
  - Elicitação (ou análise) de requisitos
  - Especificação de requisitos
  - Validação dos requisitos

Atividades de Desenvolvimento

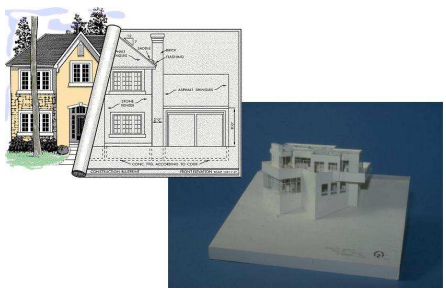
1. Especificação de requisitos
2. **Projeto de Software**
3. Implementação
4. Validação do software
5. Evolução de software

Projeto de Software

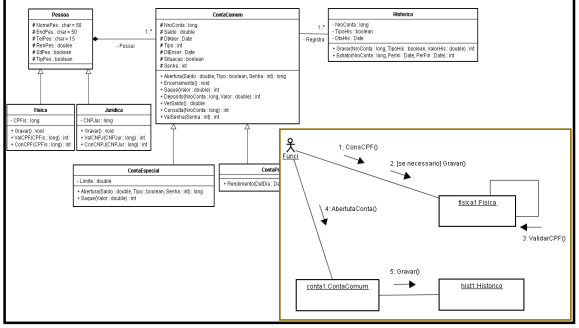
Dividido em duas etapas

- **Projeto Preliminar** define a estrutura modular do software, as interfaces e as estruturas de dados utilizadas
  - Modelo de Arquitetura
- **Projeto Detalhado** descreve detalhadamente cada módulo definido do projeto preliminar
  - Modelo de Projeto

Projeto de uma Casa



Projeto de Software



Atividades de Desenvolvimento

1. Especificação de requisitos
2. Projeto de Software
3. **Implementação**
4. Validação do software
5. Evolução de software

Implementação

- A implementação segue as definições da fase anterior
  - Transcreve as decisões de projeto arquitetural e detalhado para uma linguagem de programação
- Abordado em outras disciplinas
  - AEDS 1 e 2, POO, Programação de Computadores, Programação Modular, etc.

Atividades de Desenvolvimento

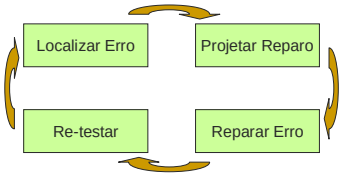
1. Especificação de requisitos
2. Projeto de Software
3. Implementação
4. **Validação do software**
5. Evolução de software

Teste e Validação

- Tem por objetivo garantir que o sistema satisfaça os requisitos
- Consiste da realização de alguns tipos de testes para encontrar erros
- A inexistência de erros não representa a *adequação operacional* do sistema
  - Deve ser feita a **validação** com o cliente

Depuração de Software

- Localizar e corrigir erros



Tipos de Testes

- Teste de Componente (unitário)
  - Garantir que um componente funciona
- Teste de Sistema (integração)
  - Garantir que dois ou mais componentes funcionam juntos
- Teste de Aceitação (validação)
  - Garantir que o sistema faz o que o cliente deseja

Atividades de Desenvolvimento

1. Especificação de requisitos
2. Projeto de Software
3. Implementação
4. Validação do software
5. **Evolução de software**

Manutenção e Evolução

- O custo de manutenção é geralmente muito maior que o custo de desenvolvimento
- Cada vez menos sistemas são desenvolvidos “do zero”
  - Sistemas são desenvolvidos/adaptados a partir de outros sistemas
- Faz sentido considerar desenvolvimento e manutenção como atividades contínuas

Leis de Lehman (Evolução)

M. M. Lehman. **Rules and Tools for Software Evolution Planning and Management.** Annals of Software Engineering, 2001.

Evolução de Software

- Objetivo de Manny Lehman
  - Definir uma teoria unificada para evolução de software
- Resultados
  - Um conjunto de oito leis que “governam” a evolução de sistemas

Lei da Modificação Contínua

- Sistemas devem ser continuamente adaptados ou eles se tornam progressivamente menos satisfatórios

Lei da Complexidade Crescente

- A medida que um sistema evolui, sua complexidade aumenta, a menos que seja realizado esforço para mantê-la ou diminuí-la

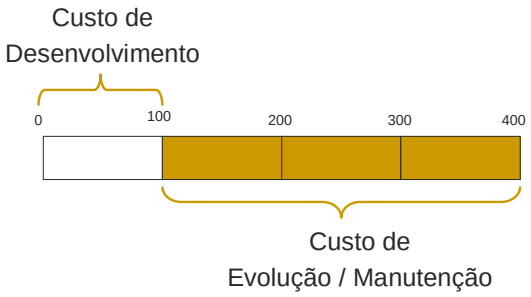
Lei do Crescimento Contínuo

- O conteúdo funcional de sistemas devem ser continuamente aumentado para manter a satisfação do usuário

Lei da Qualidade Declinante

- A qualidade de sistemas parecerá estar declinando a menos que eles sejam mantidos e adaptados às modificações do ambiente

Resultado das Leis = Custo



Revisão da Aula

Resumo

- Definições básicas de software e engenharia de software
- Alguns desafios da engenharia de software
- Porque a engenharia de software é importante
- Algumas atividades comuns ao desenvolvimento de software
  - Especificação, projeto, implementação, validação e evolução

Bibliografia Principal

- Ian Sommerville. Engenharia de Software, 9ª Edição. Pearson Education, 2011.
  - Capítulo 1, Seção 1.1 (páginas 2 a 9)
- Ian Sommerville. Engenharia de Software, 8ª Edição. Pearson Education, 2007.
  - Capítulo 1, Seção 1.1 (páginas 3 a 10)
  - Capítulo 2, Seção 2.2 (páginas 17 a 23)

Exercício (entregar nesta aula)

Responder duas das perguntas abaixo:

1. Por que leva tanto tempo para concluir um sistema de software?
2. Por que os custos são tão altos?
3. Por que não achamos todos os erros antes de entregar o software ao cliente?
4. Por que gasta-se tantos recursos para manter sistemas existentes (antigos)?