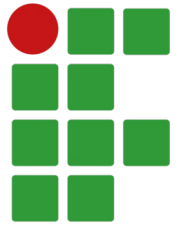
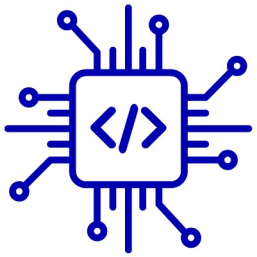


INSTITUTO FEDERAL
Rio Grande do Sul

Campus Erechim



Tecnologia em
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Engenharia de Software I

Prof. Dário Lissandro Beutler

E-mail: dario.beutler@erechim.ifrs.edu.br

Objetivo da aula

- Entender o passado, presente e futuro do desenvolvimento de software. Discutir os paradigmas de ES.

Programa da aula

- Histórico do desenvolvimento de software
- A crise do software
- Mitos em torno do desenvolvimento de software
- Soluções para os problemas de desenvolvimento

HISTÓRICO –ATÉ A DÉCADA DE 60

- Programação batch(maioria dos programas)
  - Entrada de dados
  - Processamento dos dados inseridos
  - Saída de dados
- Produzido sob encomenda
- Distribuição limitada (acesso limitado)
- Faz, usa, verifica falha, conserta(comum)

HISTÓRICO –ATÉ A DÉCADA DE 60

- Programas relativamente simples

- Limitações do hardware
  - memória
  - velocidade de processamento
  - Interfaces “simples”

FINAL DE 60 ATÉ INÍCIO DE 80

- Multiusuário
- Tempo real (ex: reserva de passagens)
- Sistemas de gerenciamento de base de dados
- Programas mais complexos – evolução do hardware

A PARTIR DA DÉCADA DE 80

- Aumento da complexidade do software
- Sistemas distribuídos
- Sistemas Especialistas
- Interfaces amigáveis (e complexas)
- Aplicações para a internet

MAIS RECENTEMENTE

- Inteligência Artificial
- Segurança da Informação
- Computação em Nuvem
- DevOps
- Big data e ciência de dados
- UI/UX

A CRISE DO SOFTWARE

- Qual o impacto do aumento da complexidade do software ?

  A crise do software (hoje “aflição crônica”)

- Verificada no final da década de 60

- Problemas referentes ao desenvolvimento e à manutenção de software

- Problemas ainda existem

A CRISE DO SOFTWARE - PROBLEMAS

- Fracasso nas estimativas de custo e prazo (desenvolvimento e manutenção)

  - falta de parâmetros (históricos) para estimar prazos e custos

- Insatisfação com o produto final

  - visão vaga dos requisitos

- Qualidade suspeita (do software produzido)

  - critérios de qualidade imaturos

A CRISE DO SOFTWARE - PROBLEMAS

- Dificuldade de manutenção, no caso de software existente

  - manutenibilidade não priorizada

  - dificuldade para compreender software existente

  - jogar fora ????

A CRISE DO SOFTWARE - CAUSAS

- Cultura de desenvolvimento

  - desenvolver software X escrever código

  - desconhecimento ou não aplicação de técnicas de desenvolvimento

  - desenvolvimento anárquico

  - criatividade

  - improviso, imediatismo

A CRISE DO SOFTWARE - CAUSAS

- **Problemas de comunicação**

  - informações não fornecidas

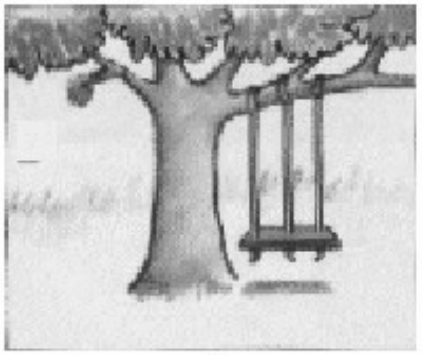
  - interpretação das comunicações

  - como comunicar “idéias” ???

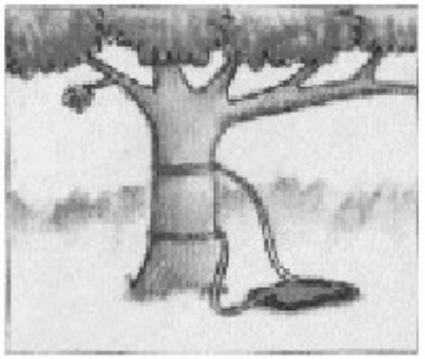
A CRISE DO SOFTWARE - CAUSAS

- **Problemas de comunicação**

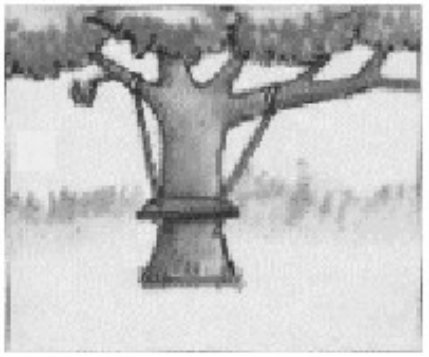
A descrição de requisitos do usuário



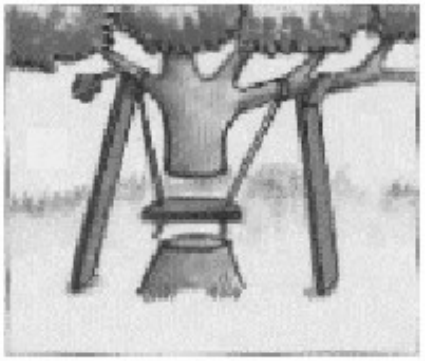
A especificação de análise



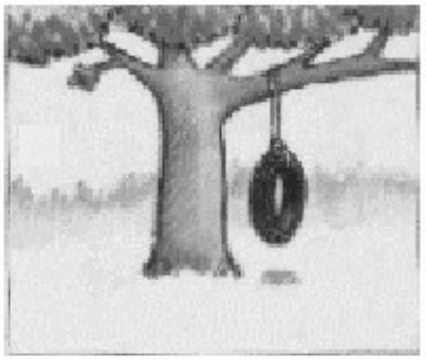
A especificação de projeto



O resultado da implementação



A implementação após os testes



O que o usuário realmente queria

MITOS EM TORNO DO DESENVOLVIMENTO DE SOFTWARE

- **Mito:** descrição geral dos objetivos é suficiente para começar a **escrever** programas – que podem ser completados posteriormente

- **Realidade:** visão incompleta dos requisitos é uma das principais causas de fracasso do esforço de desenvolvimento de software

MITOS EM TORNO DO DESENVOLVIMENTO DE SOFTWARE

- **Mito:** uma vez que um programa foi escrito e posto em funcionamento, o trabalho está encerrado

- **Realidade:** ciclo de vida

  - planejamento

  -desenvolvimento (especificar, programar)

  - manutenção (modificação)

MITOS EM TORNO DO DESENVOLVIMENTO DE SOFTWARE

- **Mito:** requisitos de projeto mudam continuamente, o que é fácil de contornar porque software é flexível

- **Realidade:** o impacto de uma mudança depende do instante do ciclo de vida em que ocorre

MITOS EM TORNO DO DESENVOLVIMENTO DE SOFTWARE

- **Mito:** revisões (ao longo do desenvolvimento) são supérfluas e consomem muito tempo

- **Realidade:** quanto mais cedo os problemas de desenvolvimento forem localizados, menores serão suas conseqüências

MITOS EM TORNO DO DESENVOLVIMENTO DE SOFTWARE

- **Mito:** se o cronograma atrasar, basta aumentar a equipe de desenvolvimento

- **Realidade:** pode atrasar ainda mais, devido às necessidades de
  - aprendizado
  - comunicação

MITOS EM TORNO DO DESENVOLVIMENTO DE SOFTWARE

- **Mito:** a meta do desenvolvimento é um programa que funcione

- **Realidade:** além disto, documentação, que é a base do desenvolvimento bem sucedido e serve de suporte a futuros esforços de modificação

MITOS EM TORNO DO DESENVOLVIMENTO DE SOFTWARE

- **Mito:** uma vez que o programa está funcionando, manutenção é simples

- **Realidade:** manutenção consome cerca de 70% do custo/esforço

SOLUÇÕES PARA OS PROBLEMAS

- **Questão:** qual a solução para o desenvolvimento de software, considerando os mitos e a cultura de desenvolvimento descritos ?

- Realidade atual:

  PARADOXO DA INDÚSTRIA DE SOFTWARE

  - programas cada vez mais complexos
  - menor tempo de desenvolvimento
  - exigências mais precisas de qualidade

SOLUÇÕES PARA OS PROBLEMAS

- **3 grandes soluções:**

<u>Solução 1</u> - Abordagem metodológica de desenvolvimento

- definição do caminho de desenvolvimento, ao invés de "reinventar a roda"
- repetição boas experiências
- o oposto de tentativa e erro
- menor dependência de
  - criatividade(para tudo)
  - inspiração (…não é arte)
  - experiência (expert indisponível…)
  - improviso

SOLUÇÕES PARA OS PROBLEMAS

- **Solução 1 - Abordagem metodológica**
  - desenvolvimento, manutenção
  - gerenciamento do processo de desenvolvimento
  - estimativa de custos
  - coordenação do trabalho em equipe

- Registro das atividades de desenvolvimento
  - base para atividades futuras
  - possibilidade de avaliação dos resultados
  - manutenção

SOLUÇÕES PARA OS PROBLEMAS

- **Solução 2 – Uso de ferramentas automatizáveis**

  - menos tarefa humana
  - menos erros
  - execução em menos tempo, com menos esforço
  - possibilidade de avaliação dos resultados
  - manutenção

SOLUÇÕES PARA OS PROBLEMAS

- **<u>Solução 3</u> – Reuso de software**

  - necessidade de desenvolver apenas parte do produto final

  - menos desenvolvimento – menos esforço – menos tempo