PCS3100

Engenharia de Software

Fábio Levy Siqueira

Escola Politécnica da Universidade de São Paulo

Motivação

- Desenvolver software é só programação?
  - Mas como saber **o que precisa ser feito**?
    - ...no mundo real não existe enunciado de EP...
  - Como tratar do desempenho, disponibilidade, segurança, capacidade de instalação, etc.?
  - Como **organizar** a solução?

Motivação

- Um problema maior: *complexidade do software*

- Code::Blocks
  - 969.789 Linhas de código (LOC)
  - 148.306 Linhas de comentário

- Firefox
  - 20,7 **milhões** de linhas de código
  - 4,5 milhões de linhas de comentário
  - 47 linguagens de programação
    - C++: 5,7 milhões
    - JavaScript: 4,9 milhões
    - HTML: 3,0 milhões
    - C: 2,5 milhões
    - Rust: 1,7 milhões
    - Python: 1,0 milhão

(https://www.openhub.net)

Motivação

- Um problema maior: *complexidade do software*
  - Um outro exemplo: *Facebook*
    - *Frontend*
      - JavaScript
    - *Backend:*
      - PHP (Hack e XHP são baseadas em PHP)
      - Python
      - C++
      - Java
      - Erlang
      - D
      - Haskell

https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites

Motivação

- Como trabalhar em equipe?
  - É improvável que um projeto seja desenvolvido só por 1 pessoa
  - Quantos desenvolvedores estão envolvidos?
    - Windows

      - 2017: 4.000 desenvolvedores
      - 3,5 milhões de arquivos
      - https://devblogs.microsoft.com/bharry/the-largest-git-repo-on-the-planet/
  - Whatsapp

      - 2015: 50 desenvolvedores
      - https://www.wired.com/2015/09/whatsapp-serves-900-million-users-50-engineers/

Motivação

- Como diminuir a quantidade de defeitos?
  - Impacto de defeitos de **software** em 2017
    - Pelo menos **3,6 bilhões** de pessoas afetadas
    - Perdas de **US$1,7 trilhões**
      - 606 falhas de software em 314 empresas
      - (Analisaram notícias em língua inglesa)

https://www.tricentis.com/software-fail-watch/

  - Defeitos de software não causam apenas perdas financeiras!
    - Airbags da Nissan (2014)
      - 1 milhão de carros
      - Airbag não identifica adultos no banco de passageiro (e não infla)

Motivação

- Como facilitar a atividade de manutenção?
  - Manutenção tem alto custo (Grubb e Takang, 2003)
    - **49% a 75%** do custo do software é manutenção
    - 50% do custo de manutenção é entender o software!
  - Temos softwares da década de 1970 em funcionamento...
  - Importante planejar a manutenção durante o desenvolvimento

Engenharia de Software

- O que é Engenharia de Software?
  - Engenharia (IEEE 610, 1990)

> A aplicação de uma abordagem sistemática, disciplinada e quantificada para estruturas, máquinas, produtos, sistemas ou processos.

  - Software (IEEE 610, 1990)

> Programas de computador, procedimentos e possivelmente a documentação associada e os dados relativos à operação de um sistema computacional.

Engenharia de Software

- O que é Engenharia de Software?
  - Definição (IEEE 610, 1990)

    A. A aplicação de uma abordagem sistemática, disciplinada e quantificada para o desenvolvimento, a operação e a manutenção do software, isto é a **aplicação de engenharia ao software**
    B. O estudo de abordagens como definido em (A)

  - 1º uso: 1968 em uma conferência da OTAN

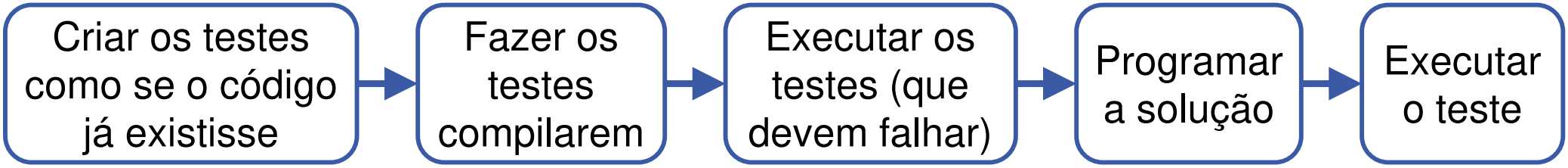
Processo de software

- Atividades clássicas de desenvolvimento

Definição e Análise de Requisitos	• Elicitar os requisitos do software • Refinar e estruturar os requisitos
Projeto (*Design*)	• Projetar como o software deve fazer o que foi requisitado
Implementação	• Criar o software
Teste	• Executar o software criado em busca de defeitos
Implantação	• Colocar o software no ambiente real

Processo de software

- As atividades não necessariamente são executadas nessa sequência!
  - Exemplo: *Test-Driven Development* (TDD)
    - **Testes de unidade** são feitos antes do código
    - Abordagem usada pelo *Extreme Programming*



Processo de software

- Cada atividade tem **diversos detalhes**
  - Definição e análise de requisitos
    - Existem diversas técnicas e representações
      - História do usuário
      - Casos de uso
      - Modelo de metas
  - Teste
    - Existem diversos níveis
      - Teste de unidade
      - Teste de integração
      - Teste funcional
      - Teste de sistema
      - Teste de aceitação

Processos

- Um projeto de software não executa *apenas* essas atividades
  - Existem diversos processos executados por uma empresa de software
  - **Referência**: ISO 12207 (2017)

Processos

Processos de acordo

- Processo de aquisição
- Processo de fornecimento

Processos organizacionais habilitadores do projeto

- Processo de gerência do modelo de ciclo de vida
- Processo de gerência de infraestrutura
- Processo de gerência de portfólio
- Processo de gerência de recursos humanos
- Processo de gerência da qualidade
- Processo de gerência do conhecimento

Processos de gerenciamento técnico

- Processo de planejamento de projeto
- Processo de avaliação e controle de projeto
- Processo de gerência de decisão
- Processo de gerência de risco
- Processo de gerência de configuração
- Processo de gerência de informação
- Processo de medição
- Processo de garantia da qualidade

Processos técnicos

- Processo de análise de negócio ou da missão
- Processo de definição das necessidades e requisitos dos *stakeholders*
- Processo de análise dos requisitos de sistema / software
- Processo de definição da arquitetura
- Processo de definição do *design*
- Processo de análise de sistema
- Processo de implementação
- Processo de integração
- Processo de verificação
- Processo de transição
- Processo de validação
- Processo de operação
- Processo de manutenção
- Processo de descarte

Alguns processos importantes

- Gerência de configuração
  - Controle de versão
    - **Como permitir que várias pessoas alterem o mesmo código**?
    - Como gerenciar a evolução dos itens?
    - Algumas ferramentas
      - Git (http://git-scm.com/)
      - Mercurial (https://www.mercurial-scm.org/)

  - Controle de mudança
    - Como analisar o impacto das mudanças e realiza-las?
    - Algumas ferramentas
      - Bugzilla (http://www.bugzilla.org/)
      - Jira (https://www.atlassian.com/br/software/jira)

Alguns processos importantes

- Gerência de configuração
  - Gerência de *release* (entrega)
    - **Como gerar uma "entrega" para os usuários?**
    - Algumas ferramentas de *build*
      - Make (https://www.gnu.org/software/make/)
      - Gradle (https://gradle.org/)
      - Apache Maven (http://maven.apache.org/)

  - Algumas ferramentas/serviços integram o controle de versão, gerência de mudança e gerência de *release*
    - GitHub (https://github.com/)
      - Foco no controle de versão
    - Jira (https://www.atlassian.com/br/software/jira)

Alguns processos importantes

- Gerência de projetos de software
  - Atividades básicas
    - Planejamento do projeto
    - Gerenciamento de riscos
    - Gerenciamento de pessoas
    - Geração de relatórios
    - Elaboração de propostas
  - Scrum
    - *Framework* de gestão de projetos bastante popular atualmente
      - https://www.scrumguides.org

Alguns processos importantes

- Verificação e validação
  - **Teste** não é a única forma de verificar e validar os produtos de software
  - Outras opções
    - Ferramentas de análise estática
      - FindBugs (http://findbugs.sourceforge.net/)
      - SonarQube (http://sonarqube.org/)
    - Modelagem e simulação
    - Revisões
    - Protótipos
    - *Pair programming*

Um pouco sobre testes

- O que é teste?

  "Teste é o processo de executar um programa com o objetivo de encontrar erros" (MYERS, 2004, p.6)

  - É improvável que o software não tenha defeitos...

- Saídas corretas em um teste não garantem que o produto é adequado
  - O teste pode não ter sido bom o suficiente!
- Entrada para a depuração
  - Processo de localizar um *suposto* erro e corrigi-lo

Um pouco sobre testes

- Então por que não testar **tudo**?
  - *Exemplo*: calcula um valor com juros

```
double calculaJuros(double valor, double taxa, int prazo):
```

    - Supondo double de 64 bits e um inteiro de 32 bits
    - Valores possíveis: $2^{64} * 2^{64} * 2^{32} = 1{,}46*10^{48}$
    - Melhor supercomputador disponível: 148.600 TFlops/s
      - Oak Ridge National Laboratory (https://www.top500.org)
      - $1{,}486*10^{17}$ operações de ponto flutuante por segundo
    - Se cada teste fosse executado em 1 operação: $9{,}84*10^{30}$s
      - ...Universo tem *somente* $4{,}4*10^{17}$s de existência...

  - Estratégias de teste ajudam a escolher conjuntos de dados que têm maior probabilidade de erros

Processos de software

- Não existe um processo ***ideal*** para desenvolver software
  - Alguns fatores que afetam o processo
    - Tamanho do projeto
    - Complexidade do software
    - Experiência da equipe
    - Uso de novas tecnologias
    - Tipo de projeto
    - Distribuição da equipe
    - *Time to market*
    - Contrato / questões legais
    - (e outras questões normais de projetos)

Processos de software

- Algumas variáveis a considerar
  - **Cultura organizacional**
  - **Processos**: RUP, XP, FDD, Iconix...
  - **Abordagens**: métodos de elicitação de requisitos, test-first, refatoração, SCRUM...
  - **Linguagens de programação**: Visual Basic .Net, Cobol, C, C++, C#, Java, Ruby...
  - **Tecnologias**: ferramentas, máquinas, bibliotecas...

Conclusão

- Desenvolvimento de software não é *somente* programação

- Importância de uma abordagem **sistemática**, **disciplinada** e **quantificada**
  - Desenvolvimento, a operação e a manutenção
  - **Engenharia de software**

- Engenharia de software não é “burocracia”
  - Depende dos processos usados e da adaptação local

Conclusão

- Algumas referências básicas
  - BOURQUE, P.; FAIRLEY, R. E (eds.). **SWEBOK: Guide to the Software Engineering Body of Knowledge**. v.3. IEEE Computer Society. 2014.
    - https://www.computer.org/education/bodies-of-knowledge/software-engineering
  - SOMMERVILLE, I. **Engenharia de Software**. 10. ed. Pearson, 2019.

Referências

- GRUBB, P.; TAKANG, A. A**. Software Maintenance: Concepts and Practice**. 2. ed. World Scientific Pub Co, 2003.
- IEEE. **IEEE Standard Computer Dictionary**. IEEE 610. IEEE, 1990.
- ISO. **Systems and software engineering - Software life cycle processes**. ISO/IEC/IEEE 12207, 2017.