Engenharia de *Software*

Aula Prática 4

Profa. Maristela Weinfurter Teixeira

Organização da Aula

Como utilizo tudo que vi?

5. Implementação
6. Testes
7. Implantação

Implementando

Pensando no *Software*

Implementação

- A definição da arquitetura de *software* traz consigo a plataforma de desenvolvimento

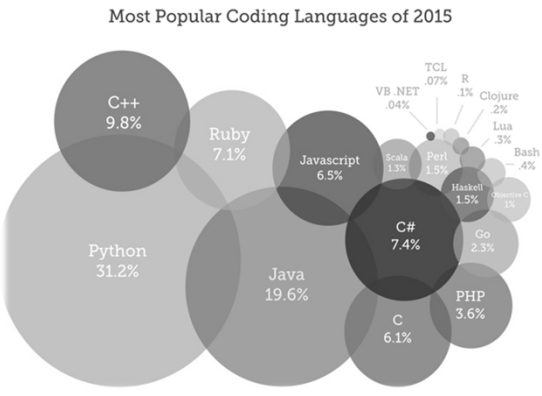
Boas Práticas de Programação

- Um bom código não é só funcional, mas também com baixo custo de manutenção
- Código mal escrito é difícil de ser mantido. Há quem prefira reescrevê-lo

- Comentários
- Nomes de variáveis significativas
- Declaração e inicialização separadas da lógica
- Programe para os outros não para você

- Comandos claros
- Comandos conhecidos
- Identação
- Não economize inicialização e finalização de blocos

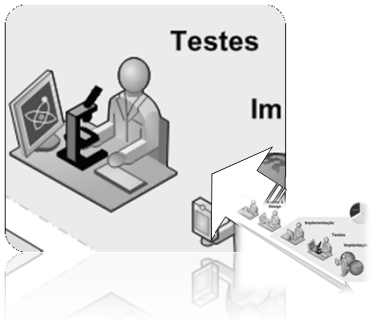
- Programe dentro do paradigma: estruturado, orientado a objetos etc.
- Modularização e decomposição
- Tipagem forte logicamente



- Para aprender linguagens de programação *on-line*:

Testando

Pensando no *Software*



Testes

IEEE 610.12, 1990

- **Erro**: defeito cometido por um indivíduo ao tentar entender uma determinada informação

- **Defeito ou falta**: manifestação concreta de um erro num artefato de *software*. Um erro pode ser resultado de vários defeitos
- **Falha**: comportamento operacional do *software* diferente do esperado pelo usuário

IEEE 830, 1998

- Recomenda práticas para especificação de requisitos de *software*. A falta de atributos dos requisitos constitui-se num tipo de defeito

- Omissão
- Ambiguidade
- Inconsistência
- Fato incorreto
- Informação estranha

Estratégias

- Baseadas em implementação
- Baseadas em especificação
- Baseadas em modelos
- Caso de teste
- Procedimento de teste (roteiro)

Inspeção de *software*

- Melhoria da qualidade de artefatos de *software* por meio da análise, detectando e removendo defeitos antes ele seja passado para o desenvolvimento

- *Checklists* sobre diferentes níveis de formalidade e de configuração das equipes de inspeções, revisões e validações

Testes de unidade

- Validação de classes
  - Exemplo: *framework* JUnit

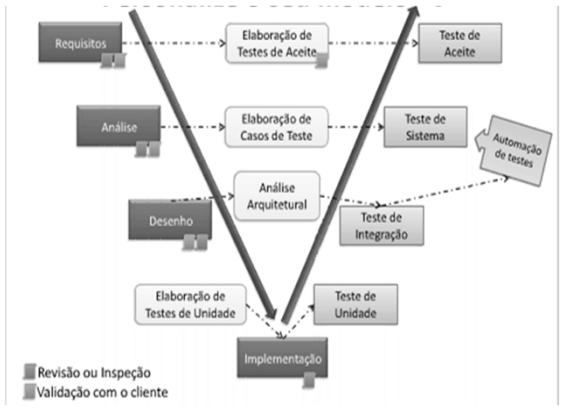
Casos de teste

- Testes eficazes sobre o que foi planejado e especificado por meio de ferramentas de geração de casos de testes
- Fluxos e regras

Testes de sistema

- Testes de aceite
- Abstração de detalhes do *software*
- Aceite de artefatos
- ISO 9126 — testes de carga, estresse e maturidade

- **Mas quais tipos de testes utilizar**?



Outros testes

- Desempenho
  - Carga
  - Estresse
  - Maturidade
- Segurança
  - XSS
  - SQL Injection

- Usabilidade
  - Acessibilidade
  - Facilidade de uso
- Caixa branca
  - Cobertura de comandos
  - Cobertura de decisão

- Caixa preta
  - Transição de estado
  - Tabela de decisão
  - Baseado em histórias do usuário

Implantando

Pensando no *Software*

Implantação

- Finalmente chegamos no momento da:
  - distribuição e entrega
  - instalação e configuração
  - utilização e manutenção

- Sejam em projetos mais tradicionais ou segundo métodos ágeis, a ideia é que haja uma finalização tranquila do projeto, utilizando-se os métodos adequados

- Essa fase requer que os *stakeholders* recebam um bom treinamento
- Instalação e configuração devem receber a devida atenção

- O importante, além do aceite oficial dos *stakeholders,* é que o *software* atinja os requisitos de satisfação na entrega do produto final

Referências de Apoio

- PAGE-JONES, M. **Fundamentos do desenho orientado a objeto com UML**. São Paulo: Pearson, 2001.
- PFLEEGER, S. L. **Engenharia de software**: teoria e prática. 2. ed. São Paulo: Prentice Hall, 2004.
- PRESMAN, R. **Engenharia de software**. 7. ed. Porto Alegre: Bookman, 2011.
- SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson, 2011.