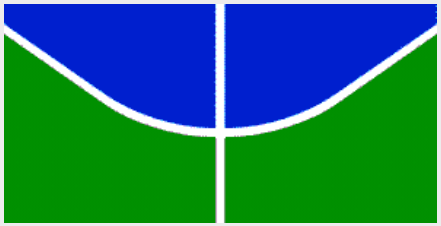


Programação Sistemática

Genaína Nunes Rodrigues

CIC/UnB

01/2015

Programação Sistemática

- Quem sou eu? Quem são vocês?
- Qual é o problema abordado no curso?
- Qual é o objetivo do curso
- Organização: aulas, avaliação: provas e trabalhos, etc...

QUEM SOU EU?

- Genaína Nunes Rodrigues (www.cic.unb/~genaina)
  - Sala PAT 044
- Graduação: BSc/CIC (Brasília)
- Mestrado: DCC/UFPE (Recife)
  - Foco em Objetos Distribuídos para Dispositivos Ubíquos
- Doutorado: CS/UCL (Londres)
  - Foco em Análise de Confiabilidade de Software Dirigida a Modelos
- Pós-doutorado: DCC/UFMG (BH)
  - Foco em estender o trabalho do doutorado para adaptar a análise de desempenho... mas descontinuada para assumir o CIC.
  - Cooperação ainda continua em TGs e Estudos de Caso.

Áreas de Atuação e Projetos Desenvolvidos

- Áreas de pesquisa
  - Atualmente no CIC:
    - Engenharia de Software: Requisitos, LPS
    - Sistemas Computacionais: Workflows, Redes Sociais, SOC
    - Métodos analíticos de Dependabilidade
  - Computação Orientada a Serviços
    - novos mecanismos de Tolerância a Faltas
  - Model Driven Development
    - MDA, Perfis UML, DSM
- Alguns projetos realizados e em andamento
  - Ubival (realizado na UCL)
  - LPS para ambientes de Vida Ambiente Assistida (com prof. Vander Alves)
  - Fase inicial do INCT para Web antes de vir para UnB
  - Universal CNPq (Cenários Implícitos)
  - Estudos de casos de portes diversos.
  - Workflows para Sistemas em Nuvens. (Bionimbus)

Por que mencionar tudo isso?

- Não se percebe a necessidade de programação modular sem experiência com projetos de média e larga escala
  - À priori, princípios desta disciplina podem não parecer importante
  - **Portanto**: a disciplina consiste de trabalhos práticos
- Sem raciocínio e projeto cuidadoso não se desenvolve software de porte razoável
  - decomposição em partes bem definidas (módulos)
  - princípios: interfaces explícitas, separação de interesses, etc...
- Um dos objetivos do curso é capacitar os alunos a usarem técnicas de modularidade adequadas ao desenvolvimento de software reutilizável, manutenível e confiável
  - teste, documentação, métricas, modelos, refatoração, etc...
  - inspeção, instrumentação, argumentação, estratégias de cobertura de teste, etc...

Quem são vocês?

- Nome?
- Período?
- Curso? Ênfase?

- Já cursou Programação OO?
- Quais linguagens de programação já tem bom domínio: C? C++? Java?

Programação Sistemática

Adaptação de material cedido pelo prof.
Alessandro Garcia
DI/PUC-Rio

O que é programação modular?

- Programação modular é a base para se desenvolver *programas de porte médio a muito grande* a partir da garantia de *qualidade de cada um dos módulos*

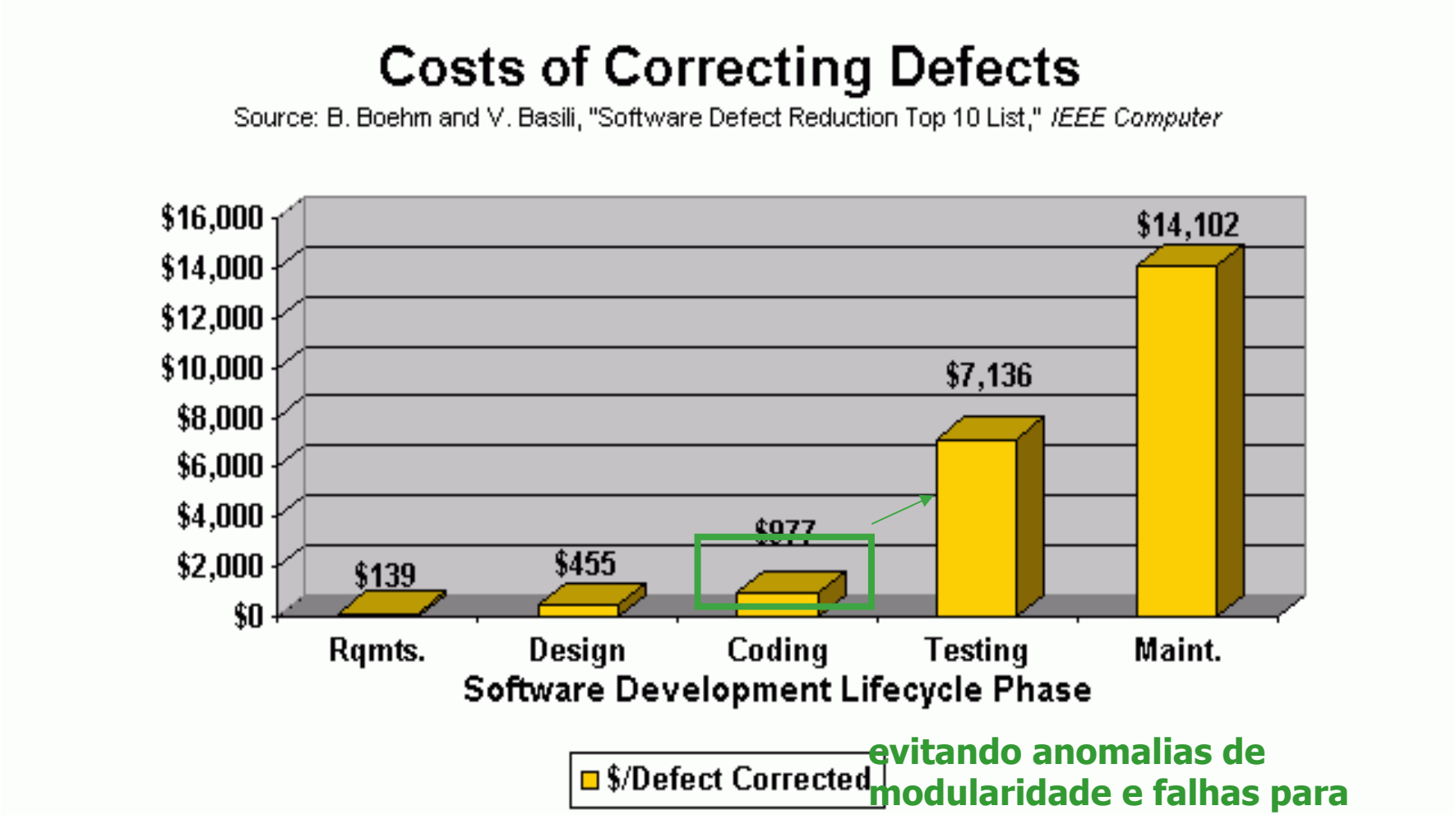
- Estratégia básica: particionamento sistemático do programa em módulos (p.e. funções) de tal forma a:
  - possibilitar o trabalho em equipe
  - facilitar a gerência do desenvolvimento
  - facilitar a manutenção de software
    - software, cada vez mais, é desenvolvido de forma incremental
  - possibilitar o reuso de módulos já desenvolvidos
  - viabilizar a criação de bibliotecas de componentes

O que é programação modular?

- Quais são os outros elementos importantes da programação modular?
  - necessidade de **atitude** visando produzir módulos isentos de defeitos
  - cada módulo deve ser desenvolvido com o devido cuidado
    - seguir convenções apropriadas de programação
      - além dos princípios de modularidade
      - ... também é importante que padrões de nomes e estruturação sejam seguidos

- Caso contrário:
  - software dificilmente atingirá nível de qualidade satisfatório
  - custo do software será mais alto

Por que programação modular é efetiva?



evitando anomalias de modularidade e falhas para reduzir custos maiores mais tarde

Problema abordado no curso de PS

- Como desenvolver um programa complexo tendo certeza que de fato funcionará e será de qualidade?
  - definir o que é esperado (**especificar**)
  - organizar a solução em temos de componentes e interfaces bem definidos (**arquitetura**)
  - **quebrar o programa** em partes bem definidas (projeto)
    - módulos são essas partes
  - **assegurar** continuamente a **corretude**
    - saber ler e criticar o código sendo lido ➔ inspeções
    - testar de forma sistemática
      - automatizar os testes

Pré-requisitos esperados

- Conhecimento da Linguagem C
- Estruturas de dados
  - básicas: ex. vetores e listas
  - árvores e grafos: não são pré-requisitos obrigatórios, mas os trabalhos podem exigir (já que a coleta de novos requisitos faz parte da ementa da disciplina)
    - entretanto: vários exemplos em sala de aula são baseados nestas estruturas
- Saber utilizar as ferramentas de desenvolvimento
  - GCC (GNU Compiler)
  - janela de linha de comando

Os deveres do aluno

- Ler o documento "Apresentação da Disciplina" ...
- **Dedicar-se** aos trabalhos práticos da disciplina
  - *não deixar para a última hora*
  - ser *autodidata* quando necessário
    - coleta de requisitos (ex. "novas" estruturas de dados)
- **Comparecer** às aulas
  - para os **ausentes**: **não serão tiradas dúvidas** sobre conceitos básicos já explicados em sala de aula
- **Fazer os exercícios** dados em sala de aula
- **Entender como trabalhar em equipe**
  - respeitar os companheiros
  - **mas lembre**: devem trabalhar com a hipótese que desentendimentos podem ocorrer e devem ser solucionados

Programação Sistemática & Outras Disciplinas

- *Estrutura de Dados*
  - PS provê princípios e técnicas para lidar de forma modular com a implementação de tais estruturas
- *Teste de software*: aprofundamento no tópico
  - PS: introdução à testes
- Programação OO:
  - PS serve de motivação para tal disciplina

As seguintes disciplinas são complementares a PS:

- Engenharia de Software
- Modelagem Orientada a Objetos
- Engenharia de Requisitos
- Projeto de Sistemas de Software

Livro texto

- Staa, A.v.; Programação Modular; Rio de Janeiro: Campus; 2000

- Texto complementar
  - algum livro que trate de programação usando a linguagem "C"

- Outras referências de interesse:
  - Fowler, M. *et al*; *Refactoring: Improving the Design of Existing Code;* Addison-Wesley, 1999.
    - Versão em Português: Fowler, M.; Refatoração: Aperfeiçoando o Projeto de Código Existente; Bookman, 2004.
  - Souza, J.N.; *Lógica para Ciência da Computação*; Rio de Janeiro: Campus; 2002

Organização

- Toda a comunicação deverá ser feita eletronicamente
  - e-mail: genaina@cic.unb.br
- Sítio da disciplina:
  - Notas de aula disponívels no moodle, após a aula.
  - Código para inscrição no moodle: **PS-A**
  - Conforme orientação no site www.inf.puc-rio.br/~inf1301.
  - Software e documentos para download
- Monitoria:
  - A ser definida

Critério de aprovação

- 2 provas, com consulta – questões teóricas, práticas e/ou relacionadas aos trabalhos
- 3 trabalhos
  - Datas e prazos a serem divulgados na próxima aula
- Cálculo da nota final
  - G1 = (P1 + MA(T1,T2)) / 2
  - G2 = (P2 + T3) / 2
  - MA = (T1 + T2)/2
  - GrauFinal (em 100) = (G1 + G2) *5
  - Observação importante:

Aprovação final requer MA(P1,P2) >= 5, mesmo que grau final seja >=5

Convivência

- Pontualidade
  - início as aulas às 10:15

- Sua participação é importante
  - façam perguntas
  - perguntas podem ser feitas sempre

Trabalhos

- O aprendizado também é adquirido e demonstrado através da realização de uma série de trabalhos ***interdependentes***
- O **objetivo** dos trabalhos **não é escrever algum programa**, mas, sim:
  - desenvolver programas *modulares de boa qualidade* e que *comprovadamente* satisfaçam massas de teste previamente estabelecidas
  - estamos pouco interessados no seu conhecimento sobre particularidades da linguagem de programação: C
- **Não é** objetivo verificar se o aluno conhece todas as sutilezas da linguagem de programação, ou dos algoritmos empregados

Trabalhos

- De maneira geral **os trabalhos são bastante trabalhosos**
  - sua realização deve ser iniciada imediatamente ao receber o enunciado
  - os enunciados deixarão margens para dúvidas
    - pressuposições e justificativas devem ser feitas nestes casos
- **Intenção**: simular o "**mundo real**" ao desenvolver programas em empresas
  - ambigüidades, inconsistências e "incompletudes" são elementos naturais em especificações de requisitos
  - desenvolvimento incremental
- **Recompensa**: dedicação aos trabalhos se refletirá naturalmente em bom desempenho também nas provas
  - cuidado! prova detecta elementos "ausentes" do trabalho em grupo

Trabalhos

- Os trabalhos serão feitos em grupos de 2 ou 3 alunos
- Os trabalhos serão corrigidos descritos na seção *Critérios de Correção de Trabalhos*
- Os programas devem ser compiláveis utilizando o compilador GCC
- Vide outras regras no documento...
- Caso tenham dúvida de como utilizar o gcc em linha de comando, acionem os monitores o quanto antes, tão logo saibamos!

Trabalhos

- **Entrega**: os trabalhos devem ser entregues via moodle.
- **Vírus**:
  - caso a mensagem contenha vírus, a nota será 0 (zero)
  - são utilizados diversos controladores de vírus
- **Atrasos**:
  - será descontado 1 ponto por dia de atraso
- **Critério:**
  - leia com atenção o folheto de critérios de avaliação dos trabalhos em anexo.

Perguntas?