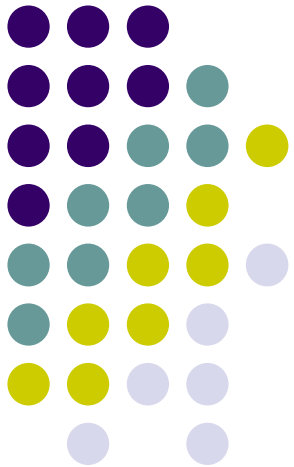
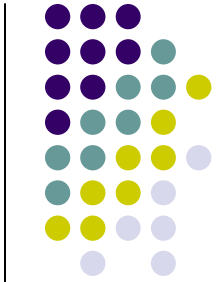
Conceitos Sobre Qualidade

Prof. Alberto Costa Neto
DComp/UFS
alberto@ufs.br

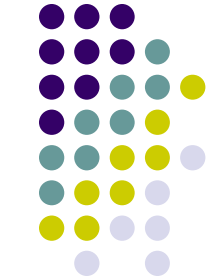


Resumo



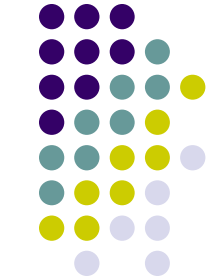
- Complexidade do Software
- Fatores de Qualidade
- Princípios fundamentais para alcançar os fatores de Qualidade
- Modularização
- Reuso
- Padrões

Complexidade inerente ao Desenvolvimento de Software



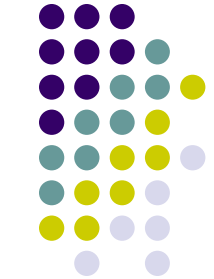
- Complexidade do domínio do problema
  - requisitos contraditórios e em constante mudança
  - comunicação entre usuários e desenvolvedores
- Dificuldade de gerenciar o processo de desenvolvimento
  - sistemas muito grandes (milhões de linhas)
  - times de desenvolvedores

Complexidade inerente ao Desenvolvimento de Software



- Falta de standards na industria do software
- Problema de caracterizar o comportamento de sistemas discretos
  - Muitas variáveis. Vários threads de controle
- Requisitos não funcionais (fatores internos/externos de qualidade) tornam o desenvolvimento mais complexo
  - Reuso, extensão, adaptabilidade, escalabilidade, ...

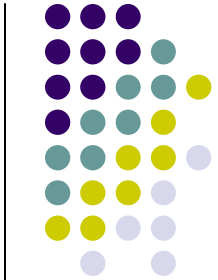
Fatores Externos e Internos de Qualidade (propriedades funcionais e não funcionais)



- Correção
  - confrontar com uma especificação
- Robustez
  - reação apropriada a condições anormais
- Extensibilidade
  - facilidade de adaptação a mudanças na especificação
- Reusabilidade

Modularidade

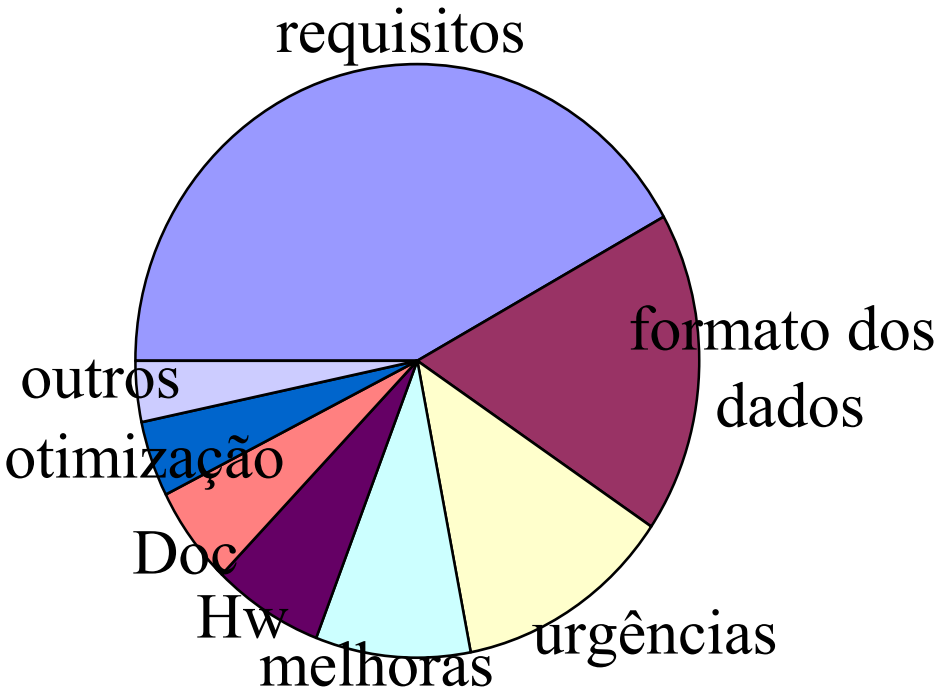
Outros fatores



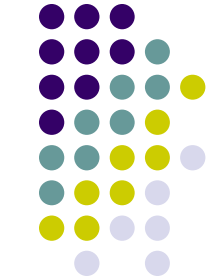
- Facilidade para escalar
- Eficiência
- Portabilidade
- Facilidade de uso
- Facilidade de Verificação e testes
- Facilidade de Reparação
- Economia e pontualidade
- Compatibilidade
- Integridade

Manutenção

- 70% do custo de software corresponde a manutenção
- Essência: modificação de requisitos



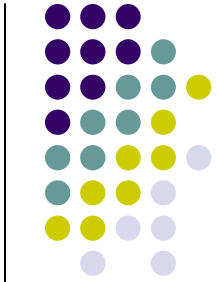
Princípios Fundamentais



Construção de software baseia-se em princípios fundamentais (*Enabling Techniques*)

- Abstração
- Encapsular
- Ocultar informação
- Modularização
- Separação de assuntos (*concerns*)
- Acoplamento e Coesão

Princípios Fundamentais

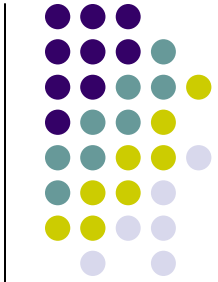


- Dividir para conquistar
- Suficiência, Completude e fácil implementação (primitiveness)
- Separação de Política e Implementação
- Separação de Interface e Implementação
- Ponto único de referência

Modularização

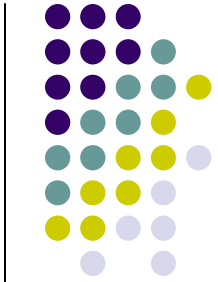
- Extensibilidade e Reuso
  - Arquiteturas flexíveis feitas de componentes autônomos
- Critérios
  - Facilidade para decompor um problema
  - Facilidade para compor componentes
  - Entendimento modular
  - Continuidade modular
  - Proteção modular

Regras para modularização



- Poucas interfaces
- Interfaces pequenas
- Esconder informação
- Interfaces explícitas
- Mapeamento direto

Metas do Reuso



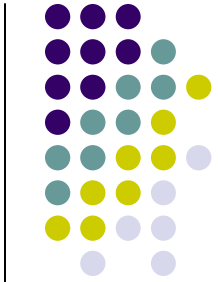
- Desenvolvimento mais rápido
- Menos manutenção
- Confiabilidade
- Eficiência
- Consistência (influenciar o estilo de prog.)
- Investimento (preservar o know-how)

Produtores ou consumidores?

- Desenvolvimento de **Software reutilizável** é uma **tarefa árdua**

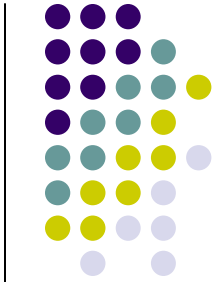
- Somos **mais consumidores** de reuso do **que produtores**

O que devemos reusar?



- Pessoal
- Projetos e especificações
- Padrões de Projeto
- Código fonte
- Componentes
  - módulos abstratos, ressaltando as propriedades relevantes para os usuários

Padrões de Software



- Princípios por si não são suficientes
- Experiência é importante
  - Não resolver a partir de princípios
  - Reusar boas soluções
- Padrões de software são registros de experiências bem sucedidas em software
  - Não são inventados – são descobertos.
  - Reuso de projetos e arquiteturas