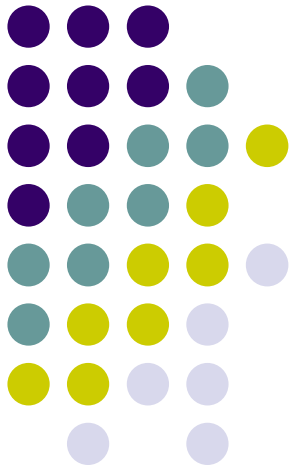
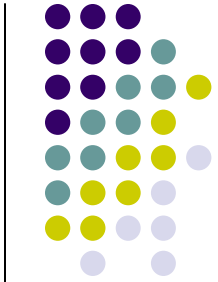
Princípios da Refatoração

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



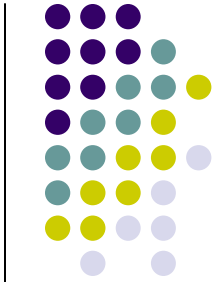
O modelo em cascata



a mentira – Edward Yourdon

- Primeiro, levante os requisitos
- Então, analise-os
- Então, projete o sistema
- Então, codifique
- Então, teste
- Então, você terminou (exceto para manutenção)
- O fato:
  - Software nunca é “finalizado”

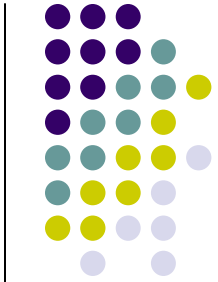
A realidade



"Build one to throw away." - Fred Brooks

- Nunca fará tudo certo de primeira. Pode ser que não entendamos:
  - O domínio do problema
  - Os requisitos do usuário
  - Como o sistema irá mudar
- Resultado
  - Projeto original é inadequado
  - Sistema se torna intrincado e frágil
  - Mudanças se tornam mais freqüentes e mais difíceis

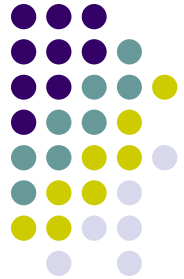
Desenvolvimento Evolutivo



"Grow, don't build software." - Fred Brooks

- Prototipagem
  - Solidifica os requisitos do usuário
  - Esboço do projeto do sistema
- Expansão
  - adicione funcionalidade
- Consolidação
  - corrija defeitos do projeto
  - Introduza novas abstrações

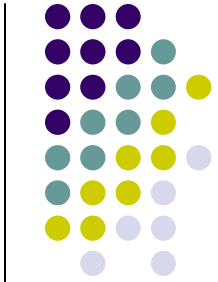
Imagine um Mundo Onde ...



- Suas tarefas começam do “zero” – não há preocupações com compatibilidade com sistemas pré-existentes
- Você entende o domínio
- Sua empresa pagará até que você esteja satisfeito com os resultados
- Condições ótimas para aplicar técnicas de orientação a objetos

Um lindo sonho, né?

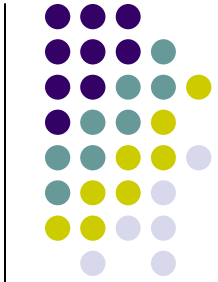
Um Cenário Mais Realista



- Você é requisitado para estender ou modificar uma peça de software existente
- Você não tem um entendimento completo do que você está fazendo
- Você está pressionado por *deadlines*

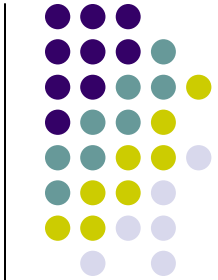
Como enfrentar mudanças?

Uma abordagem para estender software



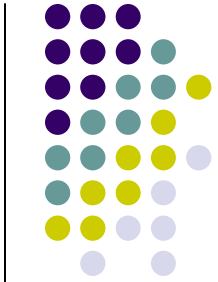
- Re-escreva o programa!
  - Aplica a experiência no projeto
  - Corrija os erros do passado
  - Criativo e divertido!
- Porém...
  - O programa fará tudo o que já fazia? Corretamente?
  - Quem pagará a conta?

Outra Abordagem



- Copie e Modifique!
  - Conveniente
  - Demonstra reuso (sem realmente entender o que se está reusando)!
- Porém...
  - Erros são propagados
  - Programas ficam inchados
  - O Projeto é corrompido
  - Custo incremental de mudanças escala

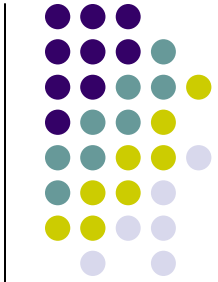
Uma Abordagem Meio Termo



- Re-estruture (refatore) o software existente:
  - Comece com o software existente como base
  - Aplique *insights* do projeto; extraia abstrações e componentes reusáveis
  - Clarifique a arquitetura do software
  - Prepare o programa para que as adições sejam mais fáceis.
- Então, adicione as novas características!

  Algumas vantagens:
  - Valoriza investimentos passados
  - Reduz duplicação
  - Simplifica o programa

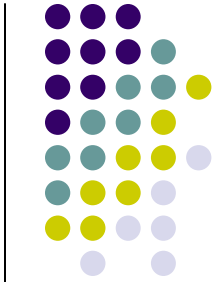
Refatoração



- “Processo de alteração de um sistema de software de modo que o comportamento externo do código não mude, mas que sua estrutura interna seja melhorada” [Fowler]
- “É uma maneira disciplinada de aperfeiçoar o código que minimiza a chance de introdução de falhas” [Fowler]
- “Melhora o projeto de código, após este ter sido escrito” [Fowler]

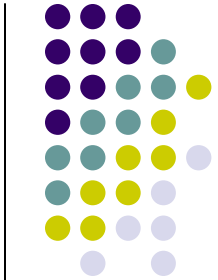
Mudanças substanciais ao software podem ser caracterizadas com refatorações mais adições

Refatoração



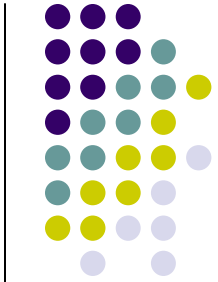
- Problemas comuns de projeto
  - Código duplicado
  - Código obscuro
  - Código complicado

Entraves para Refatorar



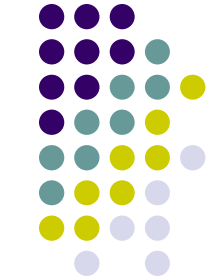
- Complexidade
  - Entender o projeto é difícil
  - Mudar o projeto de um sistema existente pode ser difícil
  - Introdução de erros frustram o propósito
- Calendários
  - Todo projeto está sobre pressão de tempo
  - Somos pagos para adicionar novas funcionalidades
  - Se não está quebrado, não conserte
  - Refatoração pode tomar muito tempo

Por que devo Refatorar



- Refatoração melhora o Projeto
- O software fica mais fácil de entender
- Ajuda a encontrar *bugs*
- Ajuda a programar mais rápido

Quando devo Refatorar



- A regra de três
  - Refatore quando for adicionar uma função
  - Quando precisa consertar um bug
  - Durante revisões de código – quando está querendo entender o código

- Existem também forças para quando não refatorar porém haverá uma dívida a ser paga

Conseqüências de não refatorar

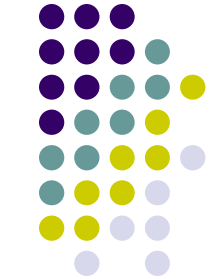
- Mudanças são feitas sobre o código original
- O projeto tende a se corromper
- Código fica cada vez mais obscuro
- Mudanças cada vez mais caras e freqüentes
- Big Ball of Mud (Brian Foote & Joseph Yoder)
  - Sistema que parece não ter uma arquitetura definida

Pré-requisitos para Refatorar

- Como você está mudando o código base, é IMPORTANTE validar com testes
  - Testar depois de Refatorar

- Também existe o tempo para refatorar e o tempo para esperar

Táticas para Refatorar



- Pequenos passos
  - Decomponha em pequenas Refatorações
- Teste antes e depois de cada pequeno passo
  - Use ferramentas (ex. JUnit e outras)
- Adote ferramentas que automatizam refatoração (ex. Eclipse)