Engenharia de Software Aula 21

Reuso de Software

Eduardo Figueiredo
http://www.dcc.ufmg.br/~figueiredo
dcc603@gmail.com

21 Maio 2012

Agenda do Curso

Aula	Data	Assunto
23	28/05	Desenv. Orientado a Aspectos
24	30/05	Laboratório
25	**04/06**	**Apresentações do TP (1)**
26	**06/06**	**Apresentações do TP (2)**
27	??/06	Revisão para Prova 2
28	??/06	Prova 2 (P2)

Agenda da Aula

- Introdução a Reuso de Software
- Abordagens de Reuso
  - Bibliotecas
  - Padrões de Projeto
  - Frameworks
  - Reuso de Modelos
  - Reuso de Componentes
  - Linhas de Produtos de Software

Reuso de Software

- Abordagem de desenvolvimento com o objetivo de maximizar o uso de software pré-existente
- Nos últimos 20 anos, muitas técnicas foram desenvolvidas para apoiar o reuso
  - Bibliotecas (API)
  - Classes de objetos
  - Componentes, etc.

Tipos de Reuso

- Reuso de Objetos e Funções
  - Tipo mais comum de reuso
  - Ocorre nos últimos 40 anos
- Reuso de Componentes
  - Reuso de média granularidade. Exemplo, componente arquitetural ou sub-sistema
- Reuso de Sistema
  - Um sistema pode ser reusado por incorporação à outro sistema
  - Pode ser necessário customização

Vantagens de Reuso

- Redução de tempo e custos
  - O sistema pode ser entregue em menor prazo, que reduz os custos
- Maior confiabilidade do produto
  - O software reusado já foi testado antes
- Menor risco no processo
  - O custo de software existente já é conhecido
- Adequação aos padrões
  - Alguns padrões (exemplo, GUI) podem ser implementados como componentes reusáveis

Potenciais Problemas

- Custo de Manutenção
  - Componentes reusados podem se tornar incompatíveis em versões futuras
- Falta de Apoio de Ferramenta
  - Ambientes de desenvolvimento podem não estar preparados para reuso
- É caro manter um biblioteca para reuso
  - É difícil encontrar e entender o software que se pretende reusar

Planejamento para Reuso

- O reuso não ocorre por acaso
  - Ele deve ser planejado e incentivado em toda a organização
- Muitas empresas desenvolvem sistemas de um mesmo domínio
  - Surgem situações potenciais para o reuso

Fatores do Planejamento

- Alguns fatores a considerar no planejamento de reuso
  - Cronograma de desenvolvimento
  - Ciclo de vida do software
  - Conhecimento e experiência da equipe
  - Domínio da aplicação

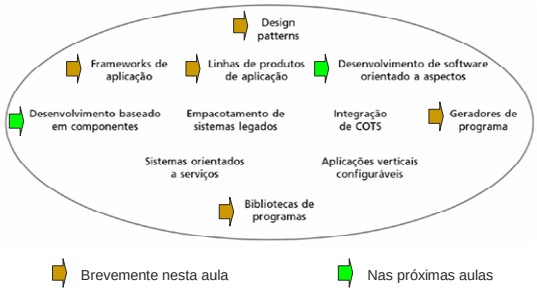
Cronograma e Ciclo de Vida

- Cronograma de Desenvolvimento
  - Se o cronograma de entrega é apertado, reusar pode agilizar a entrega do produto
- Ciclo de Vida do Software
  - Se o sistema terá vida longa e deve passar por muitas manutenções, reuso deve ser evitado
  - Componentes de terceiros podem ser de difícil manutenção (ou de código proprietário)

Equipe e Domínio

- Conhecimento e experiência da equipe na abordagem de reuso
  - Muitas abordagens de reuso são difíceis de serem usadas e requerem experiência
- Domínio da aplicação
  - Em alguns domínios, é fácil encontrar componentes e bibliotecas para reuso
  - Em outros domínios, é mais complicado

Abordagens de Reuso



Biblioteca e *Framework*

Bibliotecas de Software

- Bibliotecas implementam serviços que podem ser usados por programas
  - É um forma bem comum de reuso
- Disponibiliza funcionalidades comuns a diferentes tipos de sistemas
  - Converter informação entre formatos conhecidos (e.g., string para inteiro)
  - Acesso a recursos, arquivos, BD, etc.
  - Tipos abstratos de dados: fila, pilha, lista...

Uso de Biblioteca em Java

```
import java.util.Vector;

public class Customer {

  String name;
  Vector phoneNumbers = new Vector();

  void removePhoneNumber(String c){
    phoneNumbers.removeElement(c);
  }

  void addPhoneNumber(String c){
    phoneNumbers.addElement(c);
  }

  ...
}
```

Framework

- Frameworks são aplicações incompletas
  - São formados por interfaces, classes abstratas e classes concretas (OO)
- O conjunto de classes e interfaces formam uma estrutura genérica
- Um sistema é implementado pela adição de componentes para preencher partes do projeto
  - Por exemplo, pela implementação das classes abstratas do framework

Tipos de Frameworks

- Frameworks de infra-estrutura
  - Apóiam a criação de infra-estruturas de sistemas, tais como comunicações, interfaces de usuário e compiladores
- Frameworks de integração
  - Apóiam a comunicação e a troca de informações de componentes
- Frameworks de aplicações
  - Apóiam o desenvolvimento de um tipo de aplicações (e.g., aplicações Web)

Extensão de Frameworks

- Frameworks são entidades grandes que devem ser estendidas para reuso
- Exemplos de extensão
  - Adição de classes concretas que implementam métodos abstratos
  - Adição (sobrescrita) de métodos que implementam comportamento padrão
  - Adição de arquivos de configuração (XML)

Principal Problema

- Framework são uma entidade complexa
  - Leva um longo tempo para entendê-lo e usá-lo efetivamente
  - O desenvolvedor pode querer apenas uma funcionalidade simples

Padrão de Projeto e Padrão de Arquitetura

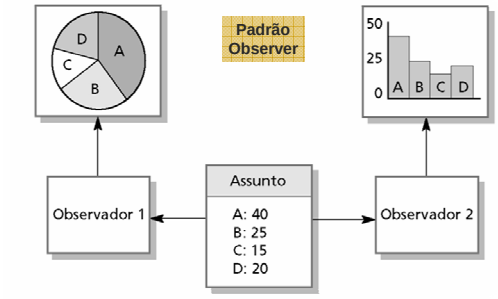
Padrões de Projeto

- Um padrão é uma descrição do problema e a essência da sua solução
- Documenta boas soluções para problemas recorrentes
  - Permite o reuso de conhecimento anterior documentados em boas práticas
- Deve ser suficientemente abstrato para ser reusado em aplicações diferentes

Elementos de um Padrão

- Nome
  - Um identificador significativo para o padrão
- Descrição do problema
- Descrição da solução
  - Um *template* de solução que pode ser instanciado em maneiras diferentes
- Consequências
  - Os resultados e compromissos de aplicação do padrão

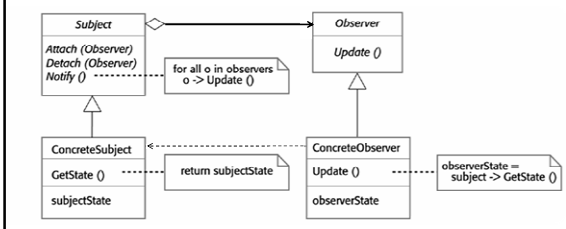
Exemplo de Problema



Padrão Obsever

- Nome
  - Observer
- Descrição do problema
  - Separa o objeto de sua forma de apresentação
- Descrição da solução (próximo slide)
- Consequências
  - Otimizações para melhorar a atualização da apresentação

Solução do Observer



Padrões Arquiteturais

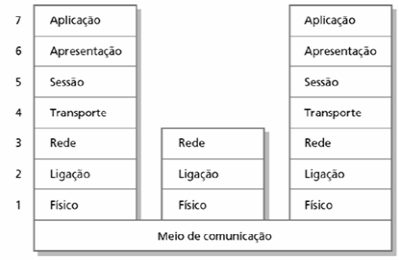
- Padrões arquiteturais expressam formas de organizar a estrutura fundamental do sistema
  - Permitem a construção de uma arquitetura aderente a certas propriedades
- O conhecimento de padrões arquiteturais pode ajudar na definição da estrutura macro do sistema

Arquitetura em Camadas

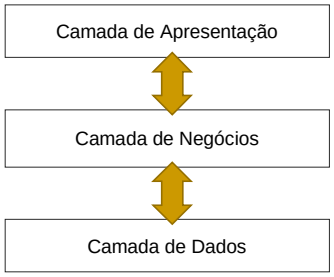
- Organiza o sistema em um conjunto de camadas
  - Cada camada oferece um conjunto de serviços
- Uma camada somente
  - Solicita serviços da camada inferior
  - Fornece serviços para a camada superior

Exemplo 1: Protocolos OSI

Modelo de camadas para sistemas de comunicação



Exemplo 2: Três Camadas



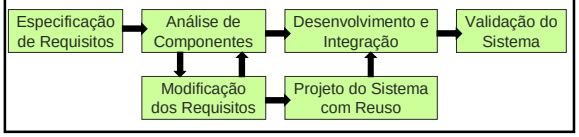
Reuso de Componentes

CBSE

- Engenharia de Software Baseada em Componentes
  - Reusar e integrar componentes pouco acoplados
  - Padrões devem ser seguidos para facilitar a integração
  - Componentes devem ser completamente especificados pelas suas interfaces

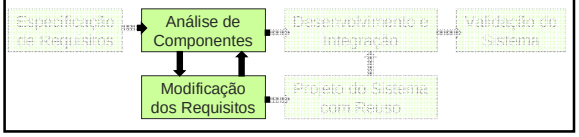
O Processo CBSE

- Modelo de processo orientado ao reuso
  - Baseia-se na existência de um número significativo de componentes reusáveis
- O processo se concentra na integração dos componentes



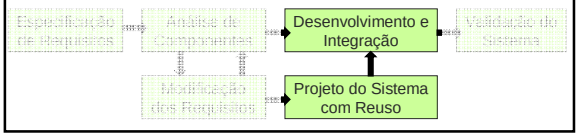
Alinhar componentes aos requisitos

- Análise de Componentes
  - Dada uma especificação, encontrar componentes que a atendam
- Modificação dos Requisitos
  - Se possível, os requisitos são adaptados aos componentes existentes



Integração dos Componentes

- Projeto do Sistema com Reuso
  - Se necessário, projeta-se novos componentes reusáveis
- Desenvolvimento e Integração
  - Desenvolvimento de novos componentes
  - Integração de **todos** os componentes



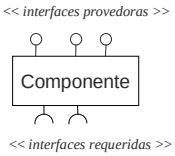
Características de Componentes

- Padronização
  - Devem seguir um padrão para facilitar integração
- Independência
  - Devem ser auto suficientes com uma interface mínima
- Substituível
  - Devem ser pensados para plugar ou remover
- Bem documentado
  - Para permitir reuso efetivo, devem ter boa documentação

Componentes x Objetos

- Componentes são geralmente implementados por uma linguagem OO
  - Mas, componentes e objetos não são a mesma coisa
- Componentes estão prontos para serem implantados
- Componentes não definem tipos
- Mesmo desenvolvidos em linguagens diferentes, componentes são integráveis
- Componentes são padronizados para integração

Notação de Componentes



- Uma caixa com o nome do componente
- Definem dois tipos de interfaces
  - Interfaces provedoras
  - Interfaces requeridas
- Existem várias implementações para os modelos de componentes
  - CORBA da OMG, Enterprise Java Beans da Oracle, COM+ da Microsoft, etc.

Vantagens

- Redução da quantidade de software a ser desenvolvido
- Redução dos custos e dos riscos
- Entrega mais rápida do produto ao cliente

Potenciais Problemas

- Pode-se desenvolver um produto que não atenda aos requisitos do cliente
- Pode ser mais difícil evoluir os sistemas
  - Componentes de terceiros
- A gerência de versões dos componentes pode ser complexa

Reuso de Modelos e Geração de Código

Motivação

- O reuso no nível de código é geralmente difícil
  - Envolve vários detalhes específicos da solução ou tecnologia adotada

Solução (MDD)

- Elevar o nível de abstração
  - O reuso passa ao nível de modelos
  - Reuso de código -> Reuso de modelos
- Pelo uso de geradores, o programa (código) é automaticamente gerado

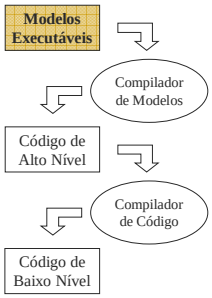
Modelos x Código

- Modelos têm vida útil mais longa
- Modelos facilitam a comunicação entre desenvolvedores (e clientes)
- Modelos são geralmente produzidos, mesmo que não se use um abordagem de geração de código

Desenvolvimento Dirigido por Modelos

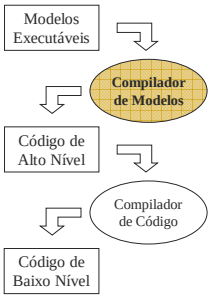
- Propõe que o desenvolvimento, manutenção, evolução e reuso sejam feitos no nível de projeto
- Reuso de modelos ainda é uma técnica pouco adotada
  - Se limitam a alguns domínios específicos ou em centros de pesquisa

Abordagem MDD



- Os modelos são independentes de software
  - Assim como, código de alto nível é independente de hardware

Abordagem MDD



- Modelos podem ser compilados para várias linguagens de programação
  - Modelos podem ser parcialmente ou totalmente reusados em diferentes contextos

O Processo MDD

- Selecionar um modelo existente
- "Recortar" as partes do modelo que interessam
  - Pode ser necessário projetar novos modelos ou adaptar os modelos existentes
- Integrar as partes selecionadas ao modelo do sistema
- Selecionar uma tecnologia de implementação
- Descrever (ou reusar) o mapeamento dos modelos para a implementação
- Gerar o sistema

Potenciais Problemas

- Imaturidade do desenvolvimento dirigido por modelos
- Falta de suporte de ferramentas e ambientes de desenvolvimento
- Modelos são vistos como extras
  - Código seria o principal
- Desenvolvedores são resistentes
  - Não gostam de "brincar" com figuras
  - Temem por seus empregos como programadores

Linhas de Produtos de Software

Linha de Produtos

- Conjunto de aplicações definidas sobre uma arquitetura comum e que compartilham componentes
  - Criada a partir de várias aplicações desenvolvidas sobre o mesmo domínio
- Uma forma efetiva de reuso em larga escala
- Usa outras técnicas de reuso
  - Frameworks, componentes, etc.

Extração da Linha

- A extração de uma linha de produtos geralmente ocorre em etapas
  1. Cria uma aplicação
  2. Cria outra aplicação no mesmo domínio
  3. Reusa algo da primeira aplicação na segunda
  4. Cria uma terceira aplicação e reusa... e assim por diante

Processo de Desenvolvimento

1. Levantar os requisitos dos usuários
2. Escolher elementos da linha de produtos que atendem de forma aproximada os requisitos
3. Negociar os requisitos com o cliente para minimizar alterações
4. Adaptar o sistema para atender todos os requisitos
5. Integrar o novo membro a “família”
   - Cada nova aplicação é um novo membro da linha de produtos

O Conceito de Característica

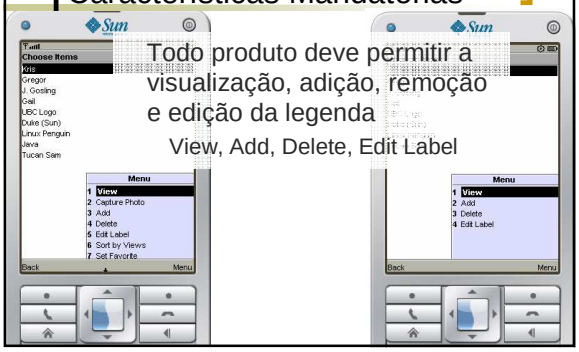
- Característica é um propriedade importante e observável do sistema
- Uma característica tem um nome conciso
  - Facilita a comunicação entre desenvolvedores
  - Ex.: Visualizar fotos, ordenar fotos, Tamanho da tela, *Logging*, etc.

Tipos de Características

- Características mandatórias
  - Aquelas que são encontradas em todos as aplicações da linha de produto
- Características variáveis
  - **Opcionais**: uma aplicação pode ou não contê-las
  - **Alternativas**: uma aplicação deve conter uma das alternativas

Exemplo de Linha de Produtos

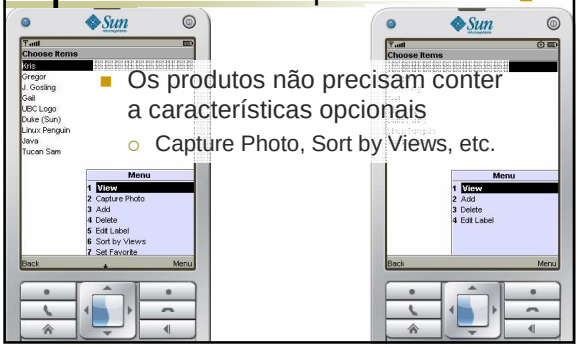
Características Mandatórias



Todo produto deve permitir a visualização, adição, remoção e edição da legenda

View, Add, Delete, Edit Label

Características Opcionais



- Os produtos não precisam conter a características opcionais
  - Capture Photo, Sort by Views, etc.

Características Alternativas

- Os produtos devem considerar algum tamanho para a tela

Bibliografia da Aula

- Ian Sommerville. Engenharia de Software, 9ª Edição. Pearson Education, 2011.
  - Cap. 16 Reuso de Software

Disciplina (Prevista para 2013-1)

- Reuso de Software
  - Reuso de funções e bibliotecas
  - Frameworks de aplicação
  - Linhas de produtos de software
  - Padrões de projeto e de arquitetura
  - Desenvolvimento orientado a aspectos
  - Programação orientada a características
  - Desenvolvimento baseado em componentes
  - Desenvolvimento dirigido por modelos