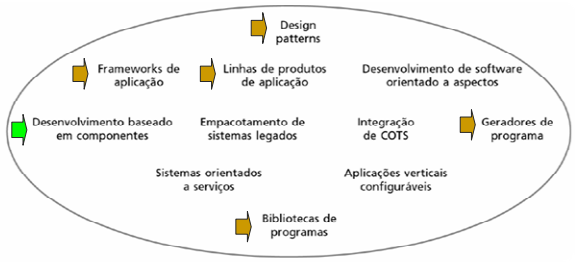
Engenharia de Software Aula 22

Engenharia de Software baseada em Componentes (CBSE)

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

23 Maio 2012

Abordagens de Reuso



Tópicos da Aula

- Componentes
- Modelos de Componentes
- O Processo CBSE
- Formas de Composição

Motivação para Reuso

- Desenvolvimento baseado em reuso está se tornando a principal forma de se produzir software
- A granularidade de reuso pode variar
  - É comum reusar objetos e funções
  - Ainda é difícil reusar componentes maiores

CBSE

- A CBSE foi proposta na década de 90
  - Foi motivado pelo limitado suporte ao reuso em desenvolvimento OO
- CBSE é um processo de definição, implementação e composição de *componentes independentes*
  - Componentes são fracamente acoplados ao sistema

Características da CBSE

- Independência
- Padronização
- Middleware
- Processo Específico

Independência e Padronização

- Componentes independentes
  - Completamente especificados por suas interfaces
- Padronização de componentes para integração
  - Se os componentes seguirem padrões, eles podem ser independentes de linguagens de programação

Middleware e Processo

- Uso de *middleware* favorece apoio para integração de componentes
  - Comunicação entre componentes
  - Alocação de recursos
  - Gerenciamento de transações
  - Proteção e controle de concorrência, etc.
- Requer um processo de desenvolvimento específico
  - Incentivo ao reuso de componentes

Principais Problemas

- Confiabilidade de componentes
- Certificação de componentes
- Comportamento após composição
- Compromisso com requisitos

Confiabilidade e Certificação

- Confiabilidade de componentes
  - Componentes são geralmente “caixas pretas”
  - Podem não atender a certos requisitos não-funcionais
- Certificação de componentes
  - A proposta seria que avaliadores independentes certifiquem componentes
  - Mas, não está claro como isso ocorreria
  - Quais as responsabilidades envolvidas?

Comportamento e Requisitos

- Difícil prever o comportamento após a composição dos componentes
  - Cada componente funciona de forma independente
  - No final, como o sistema irá comportar?
- Compromisso com os requisitos
  - É difícil equacionar os requisitos ideais do cliente com os componentes disponíveis

O que é um componente?

Componente de Software

- Um componente é uma unidade de software independente que pode ser composta a outros componentes para formar um sistema
- Características
  - Padronizado
  - Independente
  - Substituível
  - Documentado

Características de Componentes

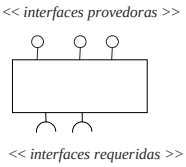
- Padronização
  - Devem seguir um padrão para facilitar integração
- Independência
  - Devem ser auto suficientes com uma interface mínima
- Substituível
  - Devem ser pensados para plugar ou remover
- Bem documentado
  - As interfaces e os serviços devem ser completamente especificados

Componente Provê Serviços

- Entidade independente e definida por suas interfaces
  - Para usá-lo, não é necessário ter acesso ao código fonte
- Serviços oferecidos são disponibilizados pelas interfaces
  - Todas as interações com os componentes são pelas interfaces

Notação de Componentes

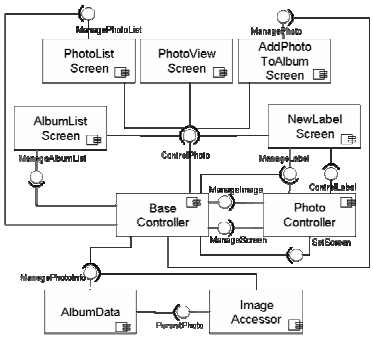
- Uma caixa com o nome do componente
- Definem dois tipos de interfaces
  - Interfaces provedoras
  - Interfaces requeridas



Tipos de Interfaces

- Interfaces provedoras
  - Definem os serviços fornecidos pelo componente
  - São representadas por círculos
- Interfaces requeridas
  - Especificam quais serviços devem ser fornecidos pelos outros componentes do sistema
  - São representadas por semicírculos

Exemplo de Sistema



Reuso de Componentes

- Componentes específicos de uma aplicação geralmente não são reusáveis
  - É preciso adaptar e ampliar o componente para criar uma versão mais genérica
- Mudanças para tornar o componente mais reusável incluem
  - Remover métodos específicos da aplicação
  - Mudar nomes para torná-los mais geral
  - Adicionar métodos, criar interfaces, etc.

Componentes x Objetos

- Componentes são geralmente implementados por uma linguagem OO
- Componentes estão prontos para serem implantados
  - Componentes não são compilados, mas instalados sobre uma plataforma de execução
- Componentes não definem tipos
  - Uma classe define um tipo, objetos são instâncias deste tipo

Componentes x Objetos

- Implementações de componentes são opacas
  - Os componentes devem ser especificados pelas interfaces
  - O código fonte pode não ser fornecido
- Componentes são independentes de linguagem
  - Objetos geralmente comunicam com outros objetos da mesma linguagem

Componentes x Objetos

- Componentes são padronizados
  - O modelo de componentes restringe a implementação
  - A padronização favorece a comunicação
- Mesmo desenvolvidos em linguagens diferentes, componentes são integráveis

Modelos de Componentes

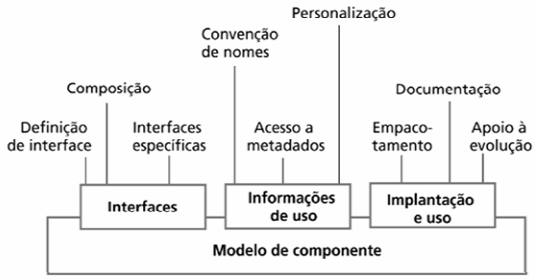
Modelo de Componentes

- Um modelo de componentes define
  - Padrões para implementação
  - Padrões de documentação
  - Padrões de implantação
- O objetivo é garantir que componentes possam interagir entre si

Exemplos de Modelos

- Existem várias implementações de modelos de componentes
  - CORBA da OMG
  - Enterprise Java Beans da Oracle
  - COM+ / .NET da Microsoft
- Os vários padrões dificultam que componente trabalhem juntos

Elementos dos Modelos



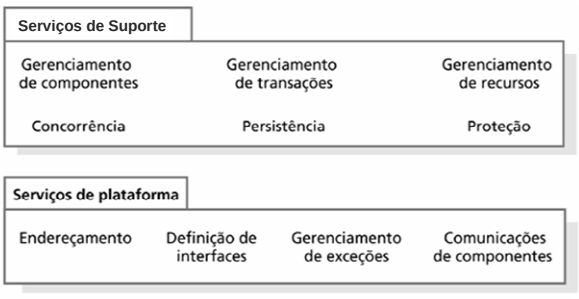
Principais Elementos

- Interface
  - Especifica como as interfaces devem ser definidas (nomes, composição, etc.)
- Informação de Uso
  - Informações para usar os componentes
  - Definições para acesso remoto
- Implantação
  - Especifica como os componentes devem ser empacotados e implantados

Tipos de Serviços Fornecidos

- Serviços de Plataforma
  - Serviços fundamentais que permitem comunicação entre componentes
- Serviços de Suporte
  - Serviços que são independentes de aplicação e são usados por muitos componentes

Serviços dos Modelos



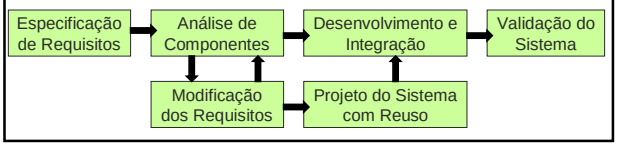
O Processo CBSE

O que é 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
- Inspirado em componentes de hardware
  - Exemplo: componentes elétricos / eletrônicos

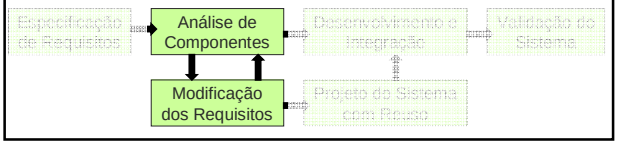
Modelo CBSE com Reuso

- 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



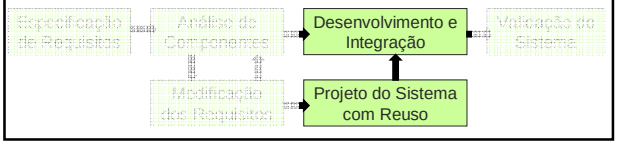
Alinhar componentes aos requisitos

- 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



Vantagens

- Reduz a quantidade de software a ser desenvolvido
- Espera-se reduzir os custos e os riscos
- Espera-se uma entrega do produto mais rápida ao cliente

Desvantagens

- 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

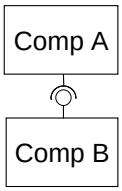
Composição de Componentes

Composição de Componentes

- É o processo de montagem de componentes para criar o sistema
- Tipos de composição
  - Composição hierárquica
  - Composição sequencial
  - Composição aditiva

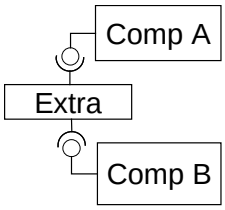
Composição Hierárquica

- Um componente chama diretamente os serviços de outro componente
- Uma interface provedora é composta diretamente a uma interface requerida



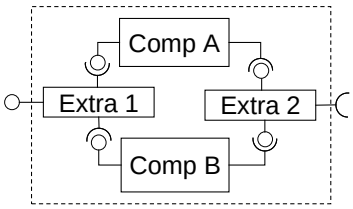
Composição Sequêncial

- Serviços dos componentes constituintes são executados em sequência
- Interfaces provedoras de dois componentes são compostas
- Algum código extra é necessário na composição



Composição Aditiva

- Interfaces de dois ou mais componentes são compostas para criar um novo componente
  - A remoção de operações duplicadas pode ser necessária



Incompatibilidades

- Durante a composição de componentes, podem ocorrer incompatibilidades
- Tipos comuns de incompatibilidades são
  - Incompatibilidade de operação
  - Incompatibilidade de parâmetros
  - Operações incompletas

Incompatibilidade de Operação

- Os nomes das operações nas interfaces provedoras e requeridas são diferentes
- Exemplo
  - Requerida:
    public void debitar(double valor);
  - Provedora:
    public void sacar(double valor);

Incompatibilidade de Parâmetros

- As operações têm o mesmo nome nas interfaces, mas os parâmetros são diferentes
- Exemplo
  - Requerida:
    public void debitar(double valor);
  - Provedora:
    public void debitar(int conta, double valor);

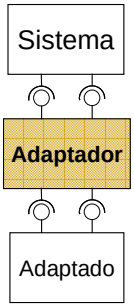
Operações incompletas

- As operações de uma interface provedora é um subconjunto das operações da interface requerida
  - Ou vice-versa
- Exemplo
  - Provedora: debitar e creditar
  - Requerida: debitar

Componente Adaptador

- Problemas de incompatibilidade são geralmente resolvidos usando um componente adaptador
- Exemplos de adaptações
  - Adicionar ou remover operações (específicas da aplicação)
  - Renomear interfaces ou elementos internos das interfaces, etc.

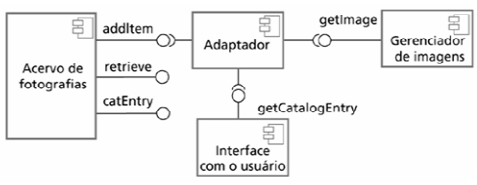
Representação do Adaptador



- Um componente adaptador converte uma interface em outra
  - A implementação é mínima
  - Tipicamente, componentes adaptadores não implementam novos serviços

Tipos de Adaptadores

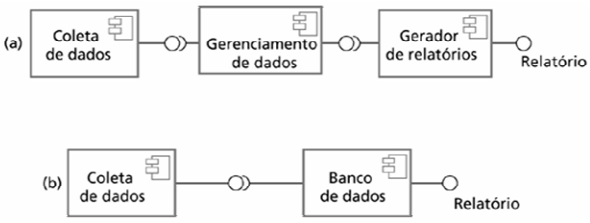
- O tipo do adaptador depende da natureza da composição
  - Hierárquica, sequencial ou aditiva



Escolha da Solução

- Em muitas situações, existem soluções variadas para o mesmo problema
- Exemplo
  - Considere um sistema de coleta de dados de fontes diferentes e geração de relatórios a partir destes dados

Possíveis Soluções CBSE



Vantagens da Solução A

- Os gerenciamentos dos dados e da geração de relatórios estão separados
  - Há maior flexibilidade para mudanças futuras
- Tanto do componente de Gerenciamento de Dados quanto o Gerador de Relatórios podem ser substituídos de forma independente

Vantagens da Solução B

- Um componente de Banco de Dados tem a geração de relatórios *built-in*
  - Semelhante ao que ocorre no MS Access
- Menos componentes são necessários
  - O sistema pode ter melhor desempenho, pois evita o overhead de comunicação
- Regras de integridade dos dados podem ser implementadas no mesmo componente
  - As regras são válidas tanto para os dados quanto para o relatório que apresenta os dados (menos replicação)

Resumo

- CBSE é uma abordagem baseada em reuso para definir e implementar componentes fracamente acoplados nos sistemas
- Um componente é uma unidade de software cuja funcionalidade e dependência são completamente definidas pelas suas interfaces

Resumo

- Um modelo de componentes define um conjunto de padrões que os componentes devem seguir
- Composição de componentes é o processo de junção de componentes para criar o sistema
- Ao compor componentes reusáveis, é normalmente necessário escrever adaptadores

Bibliografia da Aula

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Cap. 17 Engenharia de Software baseada em Componentes
- Bibliografia adicional
  - Clemens Szyperski. **Component Software: Beyond Object-Oriented Programming**, 2nd edition. Pearson Education, 2002.