DCC / ICEx / UFMG

Engenharia de Software baseada em Componentes (CBSE)

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

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

Fundamentos da CBSE

- CBSE apóia-se em princípios básicos de construção de software
  - Ocultamento de informação
  - Estabilidade das interfaces
  - Reuso de serviços

1. Ocultamento de informação
   - detalhes de implementação são ocultados
   - A implementação de um componente pode ser alterado sem afetar os outros

Fundamentos da CBSE

2. Estabilidade das interfaces
   - Um componente é facilmente substituível se as interfaces forem mantidas
   - Componentes novos pode ter novas funcionalidades, mantendo as anteriores
3. Reuso de serviços
   - Um componente fornece serviços que podem ser reusados em diferentes sistemas

Problema de Padronização

- Uma variedade de protocolos e normas foi desenvolvida para apoiar a CBSE
  - Exemplo: CORBA, Enterprise Java Beans, COM e .NET
- Existência de vários padrões dificultam a adoção de CBSE
  - Os componentes em diferentes padrões podem não interoperar
- Complexidade das normas e protocolos

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 devem ser de fácil integração

O que é um componente?

Componentes de Software

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

Padronizado e Independente

- Padronizado
  - A padronização facilita a integração
  - Além da interface, metadados e documentação são padronizados
- Independente
  - Devem ser auto suficientes com uma interface mínima
  - Os serviços necessários pelo componente devem ser especificados

Implantável e Bem Documentado

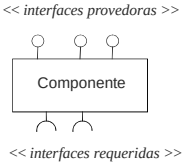
- Implantável
  - O componente não precisa ser compilado (pode ser binário) antes de implantar
  - Devem ser pensados como uma entidade autônoma
- Bem documentado
  - As interfaces e os serviços devem ser completamente especificados
  - Facilita a escolha do componente

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

Diagrama de Componentes

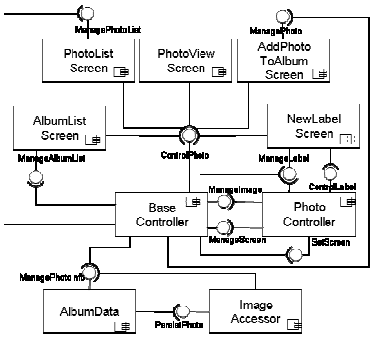
- Componentes são representados por uma caixa com o seu nome
- Definem dois tipos de interfaces (UML)
  - Interfaces provedoras
  - Interfaces requeridas



Tipos de Interfaces

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

Exemplo de Sistema



Modelos de Componentes

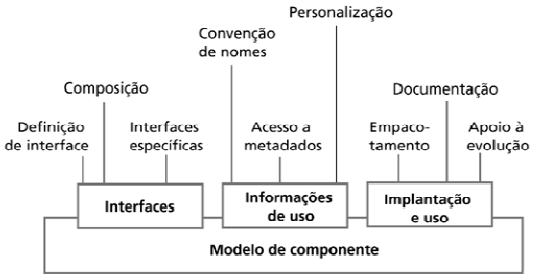
Modelo de Componentes

- 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
  - .NET da Microsoft
- Os vários padrões dificultam que componente trabalhem juntos

Elementos dos Modelos



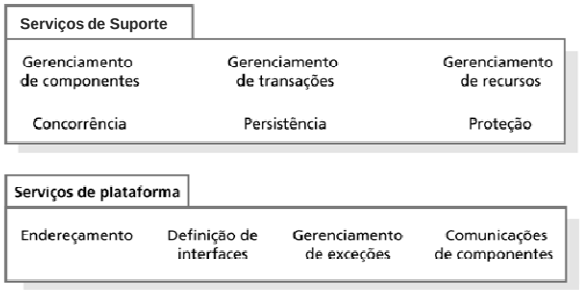
Principais Elementos

- Interface
  - Especifica como as interfaces devem ser definidas (nomes, composição, etc.)
- Informação de Uso
  - Serviços providos e requeridos
  - 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



Bibliografia

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Cap. 17 (até a Seção 17.1)
- Bibliografia adicional
  - Clemens Szyperski. **Component Software: Beyond Object-Oriented Programming**, 2nd edition. Pearson Education, 2002.