

O Modelo de Componentes CORBA

Frank A. Siqueira
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina

Roteiro de Apresentação

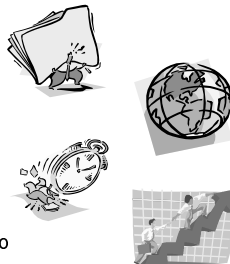
1. Visão Geral
2. O Modelo de Objetos do CORBA
3. O Modelo de Componentes CORBA
4. Interação entre Componentes
5. Especificação de Componentes
6. Desenvolvimento de Componentes
7. Instanciação de Componentes
8. Exemplo de Aplicação
9. Integração entre CCM e EJB
10. Considerações Finais

2

Visão Geral

■ Mercado de software dinâmico e competitivo

- ◆ Aplicações são cada vez mais complexas
- ◆ Necessidade de comunicação via rede
- ◆ Menos tempo para desenvolvimento
- ◆ Exigência de maior qualidade com menor custo



3

Visão Geral

■ Métodos e técnicas padronizadas são usados para desenvolver e manter software

■ Modelo de Componentes CORBA (CCM)

- ◆ É uma técnica padrão de desenvolvimento de componentes *plug & play* compatíveis com CORBA
- ◆ Componentes são introduzidos na aplicação e configurados de acordo com seus requisitos
- ◆ Componentes podem ser substituídos e atualizados independentemente do restante da aplicação

4

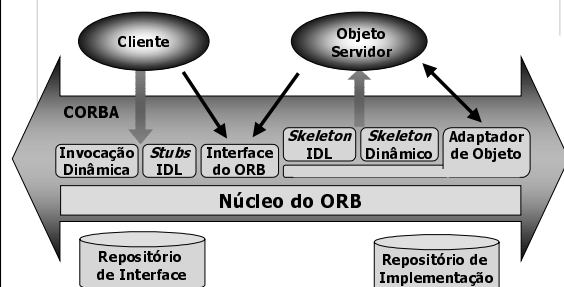
Roteiro de Apresentação

1. Visão Geral
2. O Modelo de Objetos do CORBA
3. O Modelo de Componentes CORBA
4. Interação entre Componentes
5. Especificação de Componentes
6. Desenvolvimento de Componentes
7. Instanciação de Componentes
8. Exemplo de Aplicação
9. Integração entre CCM e EJB
10. Considerações Finais

5

O Modelo de Objetos do CORBA

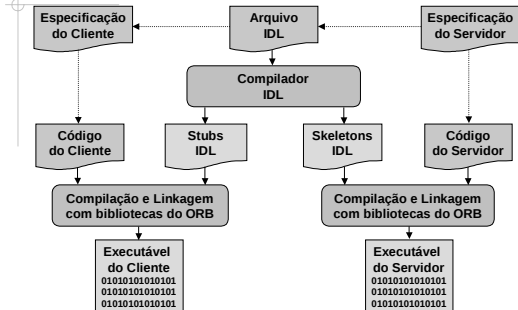
A Arquitetura CORBA



6

O Modelo de Objetos do CORBA

Desenvolvimento de Aplicações



7

O Modelo de Objetos do CORBA

Limitações do Modelo

- ◆ Pouca reutilização de objetos
- ◆ Possibilidade de estender a funcionalidade de objetos e aplicações é limitada
- ◆ Falta de um modo padrão de distribuição e instanciação de objetos
- ◆ Dificuldade de utilizar e configurar aplicações e serviços CORBA

8

Roteiro de Apresentação

1. Visão Geral
2. O Modelo de Objetos do CORBA
3. O Modelo de Componentes CORBA
4. Interação entre Componentes
5. Especificação de Componentes
6. Desenvolvimento de Componentes
7. Instanciação de Componentes
8. Exemplo de Aplicação
9. Integração entre CCM e EJB
10. Considerações Finais

9

O Modelo de Componentes CORBA

- CCM tem como objetivo suprir as deficiências do modelo de objetos do CORBA
- CCM foi aprovado pela OMG no final de 1999
 - ◆ Documentos da OMG ptc/99-10-03 a ptc/99-10-10
 - ◆ Produtos já estão disponíveis em fase experimental
- CCM é parte do CORBA 3.0, ainda a ser lançado
- Middleware CORBA é usado pelo CCM como infra-estrutura em tempo de execução

10

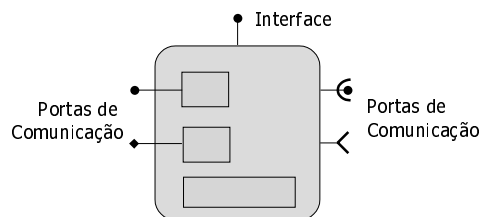
O Modelo de Componentes CORBA

- Componentes CCM são elementos básicos usados na construção de aplicações
 - ◆ Seus ciclo de desenvolvimento é padronizado
 - ◆ Podem ser estendidos e/ou combinados para formar novos componentes
 - ◆ Podem ser configurados de acordo com os requisitos da aplicação
 - ◆ São armazenados, instanciados e conectados usando uma técnica padronizada
 - ◆ Ficam disponíveis em servidores de componentes para serem reusados em novas aplicações

11

O Modelo de Componentes CORBA

- Componentes suportam uma interface IDL e possuem vários tipos de portas de comunicação



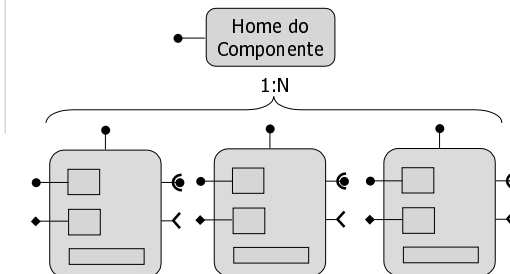
12

O Modelo de Componentes CORBA

- Componentes possuem *Homes*
- *Homes* gerenciam instâncias de um componente
 - ◆ Criam e destroem instâncias de um componente
 - ◆ Localizam instâncias do componente
 - ◆ Efetuam outras operações de gerenciamento das instâncias de um componente (operações estáticas)
 - ◆ Armazenam atributos relacionados a um tipo de componente e suas instâncias (dados estáticos)
- *Homes* são objetos CORBA especializados

13

O Modelo de Componentes CORBA



14

Roteiro de Apresentação

1. Visão Geral
2. O Modelo de Objetos do CORBA
3. O Modelo de Componentes CORBA
4. Interação entre Componentes
5. Especificação de Componentes
6. Desenvolvimento de Componentes
7. Instanciação de Componentes
8. Exemplo de Aplicação
9. Integração entre CCM e EJB
10. Considerações Finais

15

Interação entre Componentes

- Componentes interagem através de portas de comunicação
- Portas de comunicação são conectadas durante o processo de configuração do componente
- Tipos de porta de comunicação:
 - ◆ Facetas
 - ◆ Receptáculos
 - ◆ Produtores e Consumidores de Eventos
 - ◆ Atributos

16

Interação entre Componentes

- Facetas
 - ◆ Interfaces IDL fornecidas pelo componente para acesso aos seus serviços
 - ◆ Invocação síncrona nas facetas usa chamadas de operações CORBA
 - ◆ Chamadas assíncronas usam AMI (*Asynchronous Method Invocation*), introduzida no CORBA 3.0
- Receptáculos
 - ◆ Referências para outros objetos com os quais o componente interage
 - ◆ Através deles o componente pode invocar operações de outros componentes

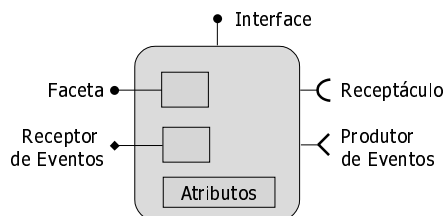
17

Interação entre Componentes

- Produtores de Eventos
 - ◆ Emitem eventos assíncronos (para um consumidor) ou os publicam (número arbitrário de consumidores) usando o serviço de notificação do CORBA
- Consumidores de Eventos
 - ◆ Recebem eventos assíncronos usando o serviço de notificação do CORBA
- Atributos
 - ◆ Parâmetros de configuração do componente
 - ◆ CCM permite que eventos gerem exceções

18

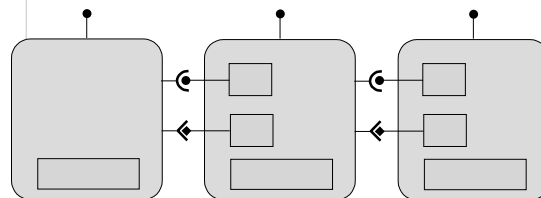
Interação entre Componentes



19

Interação entre Componentes

- Portas compatíveis podem ser conectadas
 - Facetas com Receptáculos
 - Produtores com Consumidores de eventos



20

Roteiro de Apresentação

1. Visão Geral
2. O Modelo de Objetos do CORBA
3. O Modelo de Componentes CORBA
4. Interação entre Componentes
5. Especificação de Componentes
6. Desenvolvimento de Componentes
7. Instanciação de Componentes
8. Exemplo de Aplicação
9. Integração entre CCM e EJB
10. Considerações Finais

21

Especificação de Componentes

- Componentes são descritos em IDL
- Na IDL do componente são descritas:
 - As interfaces suportadas pelo componente
 - As portas de comunicação do componente
- As interfaces suportadas pelo componente e os tipos das portas e são declarados em IDL
- Componentes suportam uma interface padrão com operações para gerenciamento, navegação, conexão de portas, etc.

22

Especificação de Componentes

Definição do Componente em IDL

```
component MyComponent supports MyInterface {
    provides      MyFacet      facet;
    uses          MyReceptacle recep;
    uses multiple MyReceptacle recep2;
    emits         MyEventSource ev_src;
    publishes     MyEventSource2 ev_src2;
    consumes      MyEventSink   ev_snk;
    attribute     MyAttribute   attr;
};
```

23

Especificação de Componentes

Atributos

- Atributos podem possuir exceções
- Exceções de atributos são de dois tipos:
 - exceções ocorridas na leitura do atributo: declaradas com **getRaises()**
 - exceções ocorridas na atribuição do valor do atributo: declaradas com **setRaises()**

```
attribute AttrType AttrIdent
    getRaises(exception)
    setRaises(exception);
```

24

Especificação de Componentes

Herança

- Componentes suportam herança simples

```
component MyComponent : MyBaseComponent {
    // ...
};
```

- Se existir uma relação de herança, as interfaces suportadas pelo componente ficam implícitas
 - São as mesmas do componente base!

25

Especificação de Componentes

Homes

- Homes* também são descritas em IDL
- São declarados na IDL de uma *Home*:
 - o tipo de componente gerenciado
 - operações de criação e localização
 - operações e atributos normais do CORBA
- Homes* derivam de interfaces padrão que contém operações de configuração, alocação e localização de componentes
- PrimaryKeys* podem ser usadas para identificar unicamente instâncias do componente

26

Especificação de Componentes

Homes

- Declaração de uma *Home* em IDL:

```
home MyHome manages MyComponent
primaryKey MyKeyType {
    factory      factory_operation(parameters)
                raises(exception);
    finder       finder_operation(parameters)
                raises(exception);
    MyRetType    mgmt_operation(parameters)
                raises(exception);
    attribute    MyAttributeType attr;
};
```

27

Roteiro de Apresentação

- Visão Geral
- O Modelo de Objetos do CORBA
- O Modelo de Componentes CORBA
- Interação entre Componentes
- Especificação de Componentes
- Desenvolvimento de Componentes
- Instanciação de Componentes
- Exemplo de Aplicação
- Integração entre CCM e EJB
- Considerações Finais

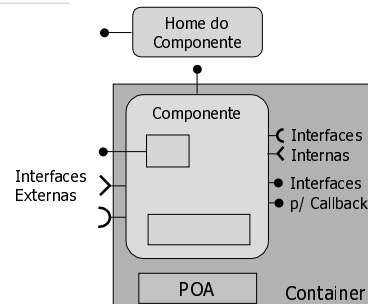
28

Desenvolvimento de Componentes

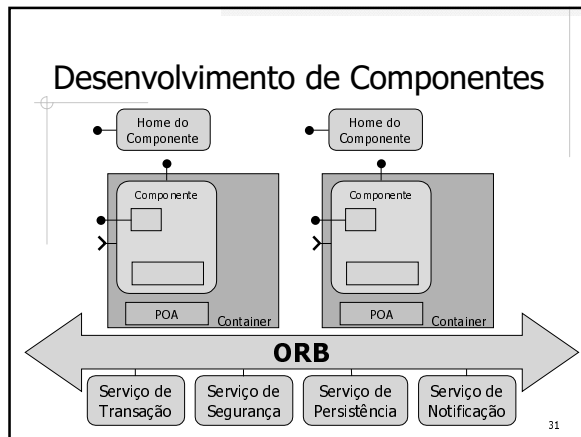
- O *Container Programming Model* simplifica o desenvolvimento e instanciação de componentes
- Funções do *Container*:
 - Ativar/desativar componentes
 - Gerenciar o POA do componente
 - Interagir com serviços do ORB para o componente
 - Serviço de Notificação (Eventos)
 - Serviço de Segurança
 - Serviço de Transação
 - Serviço de Persistência
 - Interagir com o componente através de *callbacks*

29

Desenvolvimento de Componentes



30



Desenvolvimento de Componentes

- **Component Implementation Framework (CIF)**
 - ♦ Automatiza parte do processo de desenvolvimento de componentes
- **Component Implementation Description Language (CIDL)**
 - ♦ É uma linguagem declarativa, assim como a IDL
 - ♦ Descreve composições: unidades de implementação de componentes e de suas *homes*

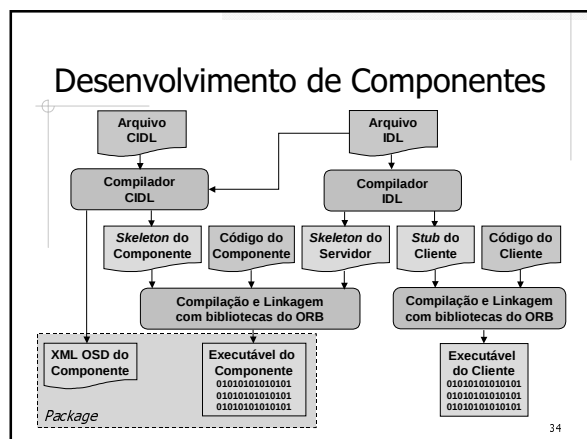
```
composition session MyComposition {
    home executor MyHomeImpl {
        implements MyHome;
        manages MyComponentImpl;
    };
};
```

32

Desenvolvimento de Componentes

- **Compilador CIDL gera:**
 - ♦ *Skeleton* do componente: código para navegação, identificação, ativação e gerenciamento de estado
 - ♦ Descritor do componente em XML OSD (*Open Software Description*)
- **Distribuição de Componentes:**
 - ♦ O código executável de um componente e seu descritor são empacotados em um *package*
 - ♦ Um *package* pode ter uma ou mais implementações de um componente
 - ♦ *Packages* podem ser disponibilizados em servidores de componentes

33



Roteiro de Apresentação

1. Visão Geral
2. O Modelo de Objetos do CORBA
3. O Modelo de Componentes CORBA
4. Interação entre Componentes
5. Especificação de Componentes
6. Desenvolvimento de Componentes
7. Instanciação de Componentes
8. Exemplo de Aplicação
9. Integração entre CCM e EJB
10. Considerações Finais

35

Instanciação de Componentes

- O CCM define procedimentos padrão para instanciação de componentes
 - ♦ O componente é instanciado carregando seu *package* em um servidor de componentes
 - ♦ Instanciado pelo servidor de componentes, o componente torna-se apto a receber requisições
- Componentes podem ser reunidos em *assemblies*
 - ♦ Um *assembly* é um agrupamento de componentes
 - ♦ Os componentes de um *assembly* e as conexões entre eles são descritos em um arquivo no formato CSD (*CORBA Software Descriptor*)

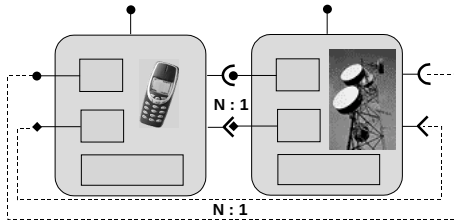
36

Exemplo de Aplicação

Telefonia Celular

Componentes:

- ◆ Telefones Celulares
- ◆ Operadora de Telefonia Celular



37

Exemplo de Aplicação

Telefonia Celular

```
// Componente Telefone Celular
component
    TelefoneCelular
supports
    InterfaceDoAparelho {
provides
    AparelhoCelular    Aparelho;
uses
    LinhaCelular      Linha;
emits
    MensagemDeTexto   Envia;
consumes
    MensagemDeTexto   Recebe;
};

// Componente Operadora Celular
component
    OperadoraCelular
supports
    AtendimentoAoCliente {
provides
    LinhaCelular          Linhas;
uses multiple
    AparelhoCelular       Aparelho;
emits
    MensagemDeTexto       Envia;
consumes
    MensagemDeTexto       Recebe;
};
```

38

Exemplo de Aplicação

Telefonia Celular

```
// Home dos Telefones Celulares
home
    PortfolioDaOperadoraCelular
manages
    TelefoneCelular
primaryKey long {
    factory  Habilita(out long num);
    finder   Localiza(in long num);
};

// Home das Operadoras
home
    AgênciaReguladora
manages
    OperadoraCelular
primaryKey short {
    factory  AutorizaOperadora(out short op);
    finder   LocalizaOperadora(in short op);
    void     MultaOperadora(in double multa);
};
```

39

Exemplo de Aplicação

Telefonia Celular

```
// Interface suportada pelo telefone celular
interface InterfaceDoAparelho {
    void Liga ();
    void Disca (in long numero);
    void Atende ();
    void Desliga ();
};

// Faceta do telefone celular
interface AparelhoCelular {
    void RecebeLigação ();
    oneway void EncerraLigação ();
};

// Definição do Tipo das Mensagens de Texto
typedef string<128> MensagemDeTexto;
```

40

Exemplo de Aplicação

Telefonia Celular

```
exception Ocupado {};
exception NumeroInvalido {};

// Interface suportada pela operadora
interface AtendimentoAoCliente {
    void AuxilioALista(in string nome,
                      out NumeroFone num)
        raises (NumeroInvalido);
    void SolicitaReparo(in NumeroFone num);
};

// Faceta da operadora
interface LinhaCelular {
    void EfetuaLigação (in NumeroFone num)
        raises (Ocupado, NumeroInvalido);
    void CompletaLigação ();
    oneway void EncerraLigação ();
};
```

41

Roteiro de Apresentação

1. Visão Geral
2. O Modelo de Objetos do CORBA
3. O Modelo de Componentes CORBA
4. Interação entre Componentes
5. Especificação de Componentes
6. Desenvolvimento de Componentes
7. Instanciação de Componentes
8. Exemplo de Aplicação
9. Integração entre CCM e EJB
10. Considerações Finais

42

Integração entre CCM e EJB

- CCM pode ser suportado em dois níveis
 - ♦ Básico
 - ♦ Estendido
- Nível Básico
 - ♦ Componentes possuem somente atributos
 - ♦ O suporte deve ser capaz de transformar objetos CORBA em componentes
 - ♦ Outras limitações existem em relação a herança e ao uso de *homes* (maiores detalhes na especificação)
- Nível Estendido
 - ♦ CCM completo

43

Integração entre CCM e EJB

- A especificação do CCM define o mapeamento entre o nível básico do CCM e o EJB



- Vantagens do CCM em relação ao EJB:
 - ♦ Permite usar outras linguagens além de Java para implementar componentes
 - ♦ Possui um modelo de comunicação mais completo
 - ♦ É um padrão da OMG

44

Roteiro de Apresentação

1. Visão Geral
2. O Modelo de Objetos do CORBA
3. O Modelo de Componentes CORBA
4. Interação entre Componentes
5. Especificação de Componentes
6. Desenvolvimento de Componentes
7. Instanciação de Componentes
8. Exemplo de Aplicação
9. Integração entre CCM e EJB
10. Considerações Finais

45

Considerações Finais

- CORBA está tendo uma aceitação cada vez maior como *middleware* para integração de tecnologias
- O CCM adiciona ao CORBA suporte ao modelo de programação baseado em componentes, cuja utilidade e eficiência foi comprovada pelo EJB
- Implementações parciais do CCM estão sendo testadas, e produtos comerciais virão em breve
- Produtos precisam ser amplamente testados para que o CCM seja utilizado em larga escala

46

Perguntas?



47

Muito Obrigado!

Frank A. Siqueira
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
Florianópolis – SC 88040-900
E-Mail: frank@lcmi.ufsc.br
Home Page: <http://www.lcmi.ufsc.br/~frank>