Uma Introdução ao CORBA

Eduardo Link
Everton Batista Petró Alexandre
Joe Luiz Wolf
Marcelo da Silva Strzykalski

Generalidades

- Especificação aberta criada pela OMG em 1991
- Motivação: necessidade de compartilhamento de informações entre as empresas e de integração de plataformas de hardware e de software heterogêneas
- ”Framework” que permite o desenvolvimento de aplicações distribuídas baseadas na tecnologia de objetos
- Encapsulamento e herança
- Permite o reuso de software

Generalidades (continuação)

- Interoperabilidade entre objetos distribuídos - invocações remotas de métodos
- Esconde a complexidade inerente ao desenvolvimento de aplicações distribuídas
- Modelo cliente-servidor
- Semântica de invocação remota ”at most once”
- Neutra em relação aos protocolos de rede empregados

Generalidades (continuação)

- Versão 1.0
  - ORB e IDL
- Versão 2.0
  - Mapeamentos de IDL para C, C++, Java, Smalltalk
  - Interoperabilidade entre ORBs remotos via IIOP
  - Segurança: IIOP + SSL
- Versão 3.0
  - QOS: controle da qualidade de serviço + mensagens assíncronas
  - CORBA mínimo, Tolerante a Falhas e em Tempo Real

Conceitos gerais

- Object Request Broker (ORB)
  - Abstração do “barramento de objetos”
  - Mascara a heterogeneidade do ambiente
  - Passagem e recepção de parâmetros (marshaling/unmarshaling)
  - Localiza a implementação do objeto remoto
  - Envia a requisição ao objeto remoto referenciado
  - Transfere o resultado do objeto remoto ao cliente
  - Invocação remota de método: interface de invocação dinâmica (síncrona ou assíncrona) ou via stub gerado em IDL (síncrona)
  - Operações possam ser invocadas por programas clientes localizados em qualquer lugar
  - Interoperabilidade: GIOP e IIOP

Conceitos gerais (continuação)

- Interface Definition Language (IDL)
  - Define os tipos de objetos CORBA por meio da especificação das suas interfaces (serviços que o objeto fornece)
  - Independe da linguagem de programação e dos protocolos de comunicação
  - Não fornece qualquer detalhe sobre implementação
  - Cada objeto CORBA pode implementar mais de uma interface
  - Transparência de localização dos objetos remotos
  - Tipos de dados: 15 primitivos (short, long, double, char, boolean etc..), Object, sequence, string, vetor, record, enumerated e union

Conceitos gerais (continuação)

- Object Management Architecture (OMA)
  - Arquitetura geral de referência (1993)
  - Define de forma abstrata as várias funcionalidades necessárias a um ambiente de computação distribuída
  - Classificação dos objetos: Objetos de Aplicação, Serviços de Objetos e Ferramentas Comuns
  - Objetos de Aplicação: aplicações específicas do usuário
  - Serviços de Objetos: funcionalidades básicas do ambiente distribuído (ciclo de vida dos objetos, consulta, transações, segurança, concorrência, nomes, notificação)
  - Ferramentas Comuns: funcionalidades de propósito geral

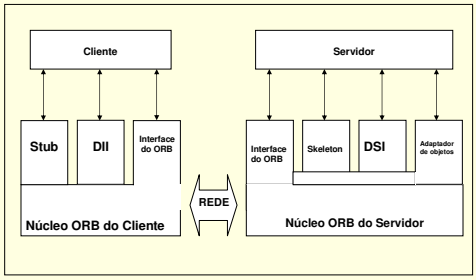
Arquitetura CORBA

- Objetivos
  - Permitir que clientes invoquem serviços de objetos remotos (objetos CORBA);
  - Linguagem do cliente independente da linguagem do objeto CORBA;
  - A linguagem de implementação do cliente não precisa ser necessariamente orientada a objetos;

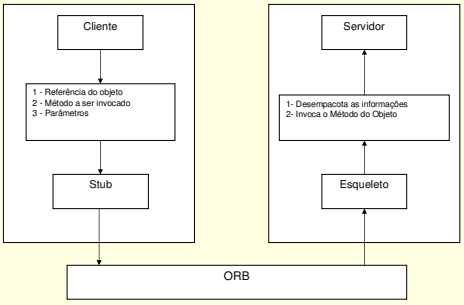
Arquitetura CORBA

- Invocações:
  - Estáticas
    - Interfaces remotas do objeto localizado no servidor é conhecido em tempo de compilação.
  - Dinâmicas
    - Interfaces remotas do objeto localizado no servidor é conhecido em tempo de execução.

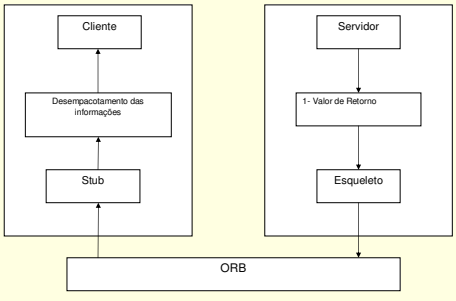
Arquitetura CORBA



Invocação Estática - Requisição



Invocação Estática - Resposta



Invocação Dinâmica

- Interface de Invocação Dinâmica
  - Localizada junto ao cliente
  - Permite que o cliente invoque métodos de objetos sem conhecer sua interface em tempo de compilação;
  - Obtenção das informações por meio do Repositório de Interfaces;

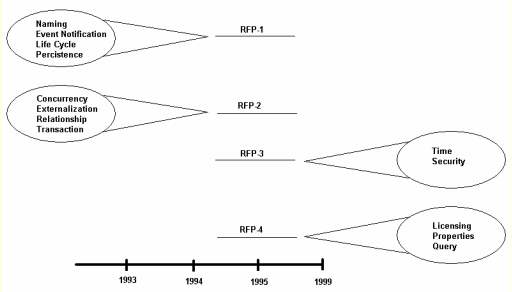
Invocação Dinâmica – 4 Passos

- Identificar o objeto que queremos invocar;
- Recuperar sua interface (buscá-la no Repositório de Interfaces);
- Construir a invocação;
- Invocar o método, através do DII e receber os resultados.

CORBAServices Introdução

- Além do ORB o Kernel da estrutura CORBA, para que seja possível a interoperabilidade entre os objetos, são necessários serviços de controle por parte do servidor.
- A OMG especificou 16 serviços desde 1993 até agora, eles são publicados sob forma de RFP's (*Request for Proposal*).

CORBAServices



CORBAServices

- **Life Cycle Service (Ciclo de Vida):** define serviços e convenções para criar, mover, deletar e copiar objetos.
- **Relationship Service (Relacionamento):** proporciona um modo para criar associações (links) dinâmicas entre componentes.
- **Naming Service (Nome):** permite que os objetos possam referenciar (e localizar) outros objetos pelo nome.
- **Object Transaction Service (Transação):** para transações on-line, ou ela é completada ou abortada completamente. Fornece às aplicações confiabilidade.
- **Trading Object Service (Negócio):** este serviço funciona como páginas amarelas de e para objetos. Permite localizar qualquer serviço de qualquer provedor disponível.

CORBAServices

- **Security Service (Segurança):** fornece funcionalidades como autenticação, controle de acesso, auditoria,comunicação segura, criptografia.
- **Concurrency Service (Concorrência):** É como "chaveador" que atua sobre os objetos no ORB. Através de um esquema de "lock" para se manter a integridade no acesso aos objetos.
- **Persistent State Service (Persistência):** interface única para objetos acessarem mecanismos de persistência - bases de dados relacionais, OO, e simples arquivos.
- **Event Service (Evento):** permite registrar, ou desregistrar dinamicamente, os seus interesses por eventos específicos. O emite uma notificação assim que o evento estiver disponível.
- **Time Service (Tempo):** este serviço fornece um mecanismo para sincronizar relógios num ambiente distribuído. Permite ainda definir e gerir eventos acionados pelo tempo.

CORBAServices

- **Notification Service (Notificação):** é uma extensão do Event Service, adiciona funcionalidades como, capacidade de transmitir eventos na forma de dados estruturados; descobrir através dos canais de eventos, quais são os tipos que lhes interessam.
- **Collection Service (Coleções):** criar agrupamentos de objetos que possuem comportamento semelhante, pondendo eles serem acessados via um índice de objetos pertencentes ao grupo.
- **Property Service (Propriedades):** com este serviço é possível associar qualidades (propriedades) dinamicamente aos objetos. Por exemplo, em um sistema de downloads, a cada vez que um download é feito, um contador de download é incrementado ao objeto, mas este atributo não faz parte da implementação do objeto.

CORBAServices

- **Externalization Service (Externalizar):** é a forma de se gravar em um streem em disco, memória ou através da rede um dado estado de um objeto, é uma forma de serializar e depois recarregar um objeto em um outro momento outro servidor.
- **Licensing Service (licenciamento):** alguns servidores podem possuir objetos que devam ter certo controle de acesso baseado em um contrato de licença de uso.
- **Query Service (Busca):** permite a busca de objetos baseado em predicados, está busca e feita e também retorna coleções de objetos que satisfação as condições.

Estudo de Caso - IDL

```
// -*- c++ -*-
interface Account {
    void deposit(in unsigned long amount);
    void withdraw(in unsigned long amount);
    long balance();
};
interface Bank {
    Account create();
};
```

Estudo de Caso - Classes

```
class Account_impl : virtual public POA_Account{
public:
  Account_impl();
  void deposit(CORBA::ULong);
  void withdraw(CORBA::Ulong);
  CORBA::Long balance();
private:
  CORBA::Long bal;
};
Account_impl::Account_impl () {  bal = 0; }
void Account_impl::deposit(CORBA::Ulong amount)
{  bal += amount;  }
void Account_impl::withdraw(CORBA::Ulong amount)
{  bal -= amount;  }
CORBA::Long Account_impl::balance () {  return bal; }
```

Estudo de Caso - Classes

```
class Bank_impl : virtual public POA_Bank{
private:
  PortableServer::POA_var theRootPOA;
public:
  Bank_impl(PortableServer::POA_var);
  Account_ptr create();
};
Bank_impl::Bank_impl(PortableServer::POA_var poa)
{  theRootPOA = poa;  }
Account_ptr Bank_impl::create (){
  Account_impl * servant = new Account_impl;
  CORBA::Object_var ref = theRootPOA->servant_to_reference(servant);
  Account_ptr acc = Account::_narrow(ref);
  return acc;  }
```

Estudo de Caso - Servidor

```
int main(int argc, char *argv[]) {
  CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
  CORBA::Object_var poaobj = orb->resolve_initial_references("RootPOA");
  PortableServer::POA_var poa = PortableServer::POA::_narrow(poaobj);
  PortableServer::POAManager_var mgr = poa->the_POAManager();

  Bank_impl * servant = new Bank_impl(poa);
  CORBA::Object_var bank = poa->servant_to_reference(servant);

  CORBA::Object_var nsobj = orb->resolve_initial_references("NameService");
  CosNaming::NamingContext_var nc =
CosNaming::NamingContext::_narrow(nsobj);
```

Estudo de Caso – Servidor (cont)

```
if (CORBA::is_nil(nc)) {
    cerr << "oops, I cannot access the Naming Service!" << endl;
    exit (1);
 }
 CosNaming::Name name;
 name.length(1);
 name[0].id = CORBA::string_dup("Bank");
 name[0].kind = CORBA::string_dup("");
 cout << "Binding Bank in the Naming Service ... " << flush;
 nc->rebind(name, bank);
 cout << "done." << endl;
```

Estudo de Caso – Servidor (cont)

```
cout << "Running." << endl;
 mgr->activate();
 orb->run();
 /*  Shutdown (never reached) */
 poa->destroy(TRUE, TRUE);
 delete servant;
 return 0;
}
```

Estudo de Caso - Cliente

```
int main (int argc, char *argv[])
{
  CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
  CORBA::Object_var nsobj = orb-
>resolve_initial_references("NameService");
  CosNaming::NamingContext_var nc =
CosNaming::NamingContext::_narrow(nsobj);
  if (CORBA::is_nil (nc)) {
    cerr << "oops, I cannot access the Naming Service!" <<
endl;
    exit (1);
  }
```

Estudo de Caso – Cliente (cont)

```
CosNaming::Name name;
  name.length (1);
  name[0].id = CORBA::string_dup("Bank");
  name[0].kind = CORBA::string_dup("");

  cout << "Looking up Bank ... " << flush;
  CORBA::Object_var obj = nc->resolve(name);
  cout << "done." << endl;
  Bank_var bank = Bank::_narrow(obj);
```

Estudo de Caso – Cliente (cont)

```
Account_var account = bank->create();
  if (CORBA::is_nil (account)) {
    printf ("oops: account is nil\n");
    exit (1);
  }
  account->deposit(700);
  account->withdraw(450);
  cout << "Balance is " << account->balance() << endl;
  return 0;
}
```

Estudo de Caso – Cliente Java

```
public class Client {
  public static void main(String args[]) {
    try{
      Properties props = new Properties();
      props.put("org.omg.CORBA.ORBInitRef",
"NameService=corbaloc::localhost:9999/NameService")
;
      ORB orb = ORB.init(args, props);
      org.omg.CORBA.Object objRef =
orb.resolve_initial_references("NameService");
      NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);
```

Estudo de Caso – Cliente Java

```
String name = "Bank";
    Bank bank =
BankHelper.narrow(ncRef.resolve_str(name));
    Account acc = bank.create();
    acc.deposit(700);
    acc.withdraw(450);
    System.out.println("Balance is: " + acc.balance());
   }
   catch(Exception e) {
    System.out.println("ERROR : " + e);
    e.printStackTrace(System.out);
  }
```