Principais conceitos de CORBA

Tecgraf PUC-Rio

fevereiro de 2011

Common Object Request Broker Architecture

- Uma arquitetura aberta para o desenvolvimento de aplicações distribuídas em um ambiente multilinguagem e multiplataforma.
- Principais características:
  - Arquitetura SOA
  - Tecnologia madura (década de 90).
  - Orientação a objetos.
  - Eficiência no transporte de dados, inclusive binários.

Principais conceitos da arquitetura

- Clientes e servidores
- ORB
- IDL
- Client stubs e Server skeletons
- Servants e Objetos CORBA
- Adaptadores e POA
- Protocolo de comunicação IIOP
- IOR
- Mecanismos de interceptação

Clientes e Servidores

- Um sistema desenvolvido em CORBA adota um modelo de comunicação cliente-servidor:
  - Um servidor é um provedor de serviços
  - Um cliente é um consumidor de serviços
  - Em CORBA, um componente pode atuar tanto como cliente como servidor

ORB – Object Request Broker

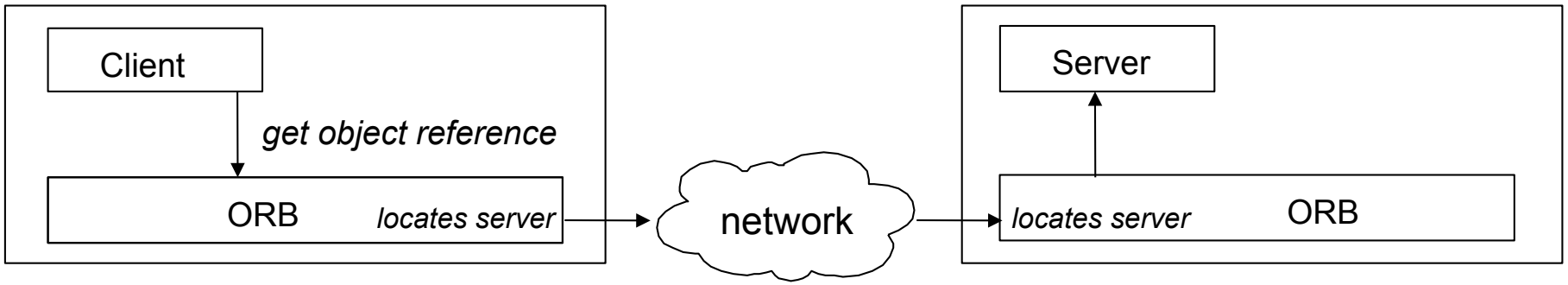
- É o componente da arquitetura CORBA responsável pela comunicação entre os objetos clientes e servidores
  - Localização de um objeto
  - Marshall – traduzir os parâmetros e valores de retorno da requisição para o formato da transmissão (on-the-wire)
  - Unmarshall - traduzir os parâmetros e valores de retorno transmitidos (on-the-wire) para o formato da requisição

ORB – Object Request Broker

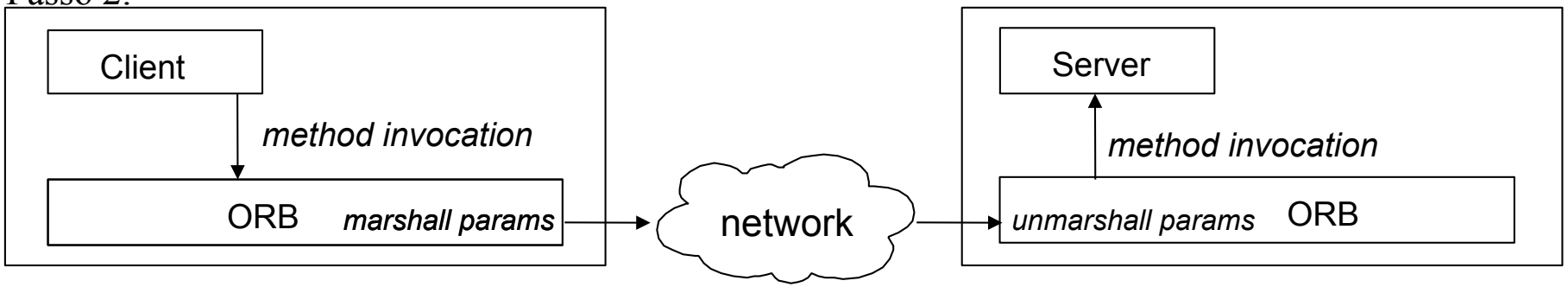
- Um dos principais benefícios do ORB é o tratamento dos dados da requisição de forma independente de plataforma
  - parâmetros são convertidos “on-the-fly” entre formatos de máquinas diferentes, pelo mecanismo de marshall e unmarshall.
- O programador não precisa se precupar em como isso é feito e às chamada aos objetos remotos ocorrem como se fossem chamadas locais.

ORB – Object Request Broker

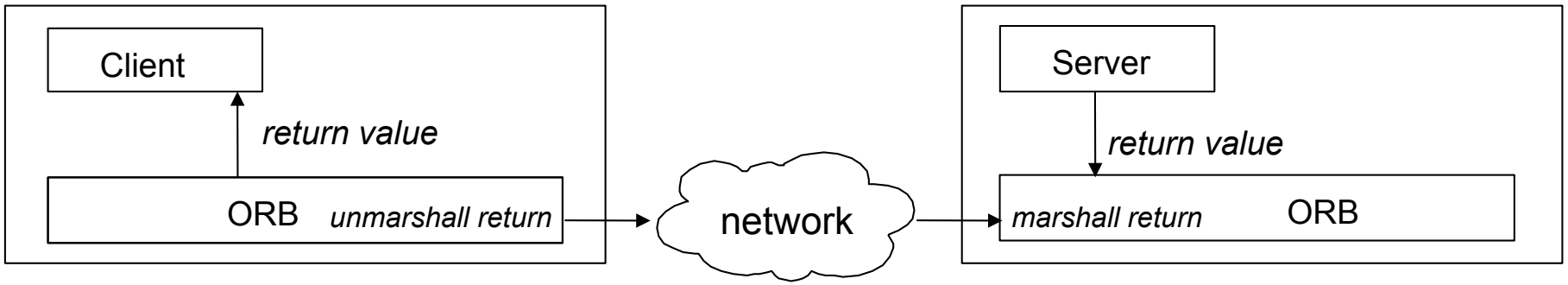
Passo 1:



Passo 2:



Passo 3:



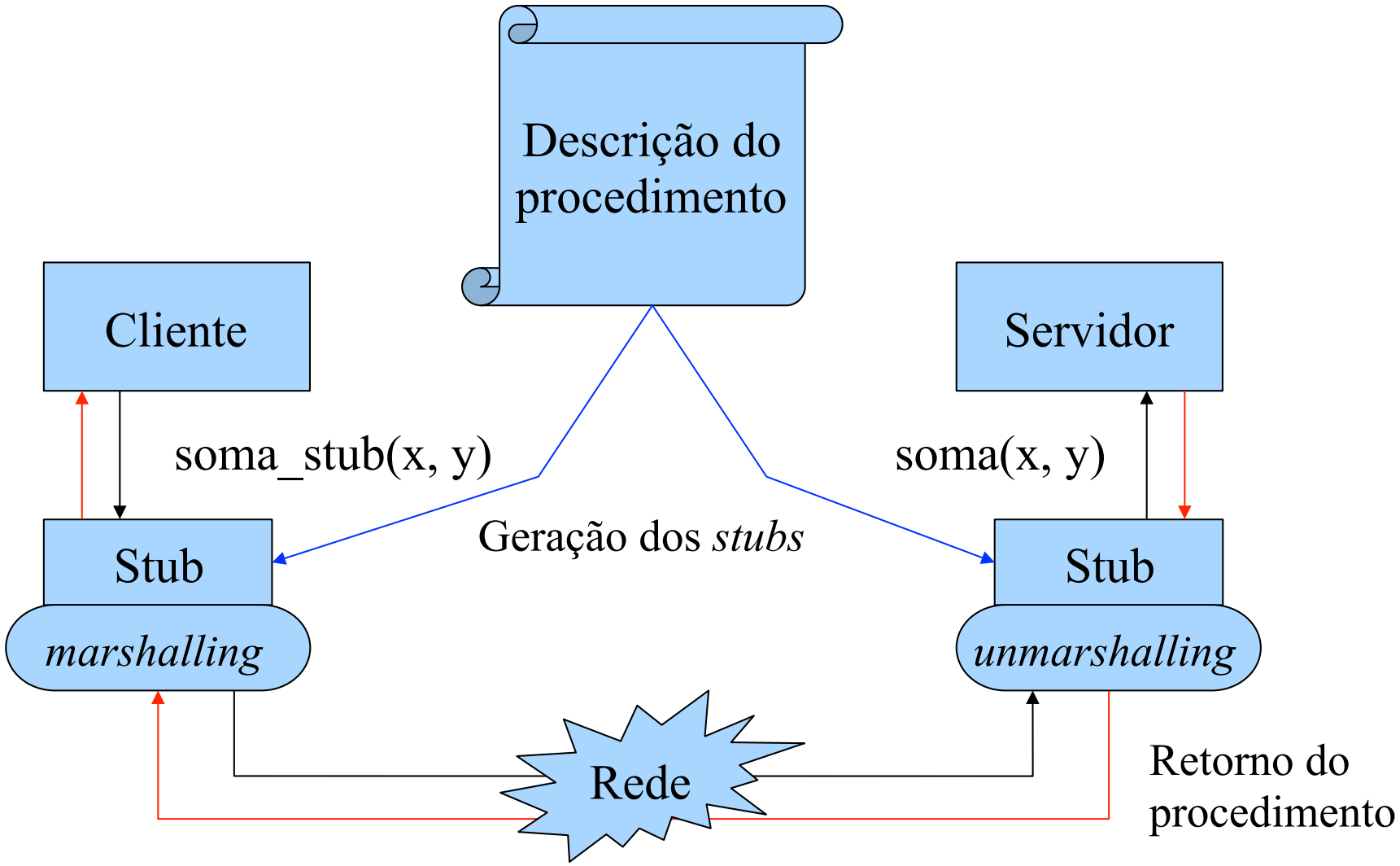
Comunicação Distribuída

- A comunicação distribuída baseada em *Sockets* oferece um nível muito baixo de API para o desenvolvedor
  - Não define nenhum protocolo de comunicação entre o cliente e o servidor.
  - Base para a implementação de outras tecnologias de nível mais alto.
- A comunicação baseada em *Chamada Remota de Procedimentos* (RPC) oferece uma abstração de nível mais alto para comunicação distribuída

RPC - Remote Procedure Call

- Nível mais alto do que sockets.
- Executa chamadas remotas como se fossem locais.
  - Stubs
    - Cliente (proxy).
    - Servidor.
    - Podem ser gerados automaticamente.
- Diversas implementações: Sun RPC, XML RPC, etc.

RPC - Remote Procedure Call



Comunicação entre Objetos Distribuídos

- Evolução do conceito de RPC
  - Noção de objetos distribuídos (OO)
- Integração de ambientes heterogêneos
  - Plataformas de *hardware* e sistemas operacionais
  - Linguagens de programação
- Transparência de localidade dos objetos
- Linguagem neutra para descrição dos serviços

IDL – Interface Definition Language

- É a linguagem usada para definir a interface dos serviços
- Garante a interação entre componentes independente da linguagem de implementação dos clientes e dos servidores
  - Por exemplo, o tipo long definido pela IDL é um valor inteiro de 32-bits com sinal que pode ser mapeado para um long C++ (depedendo da plataforma) ou para um int Java.

IDL – Interface Definition Language

- Mapeada para diversas linguagens: (C, C++, Java, Lua, COBOL, …)
- Orientada a objetos
- Clareza e simplicidade na sua definição
- Familiar para os programadores C++ e Java

Exemplo de IDL

```
struct Book {
    string author;
    string title;
};

typedef sequence<Book> BookSeq;

interface Biblioteca {
    boolean addBook(in Book pbook);
     BookSeq getBooks();
    Book getBook(in string title);
};
```

Stubs e Skeletons

- Client stubs e server skeletons atuam como a "cola" entre a especificação IDL, independente de linguagem, e o código que é específico de uma determinada linguagem
- O processamento das interfaces IDL gera os stubs e os skeletons para cada interface

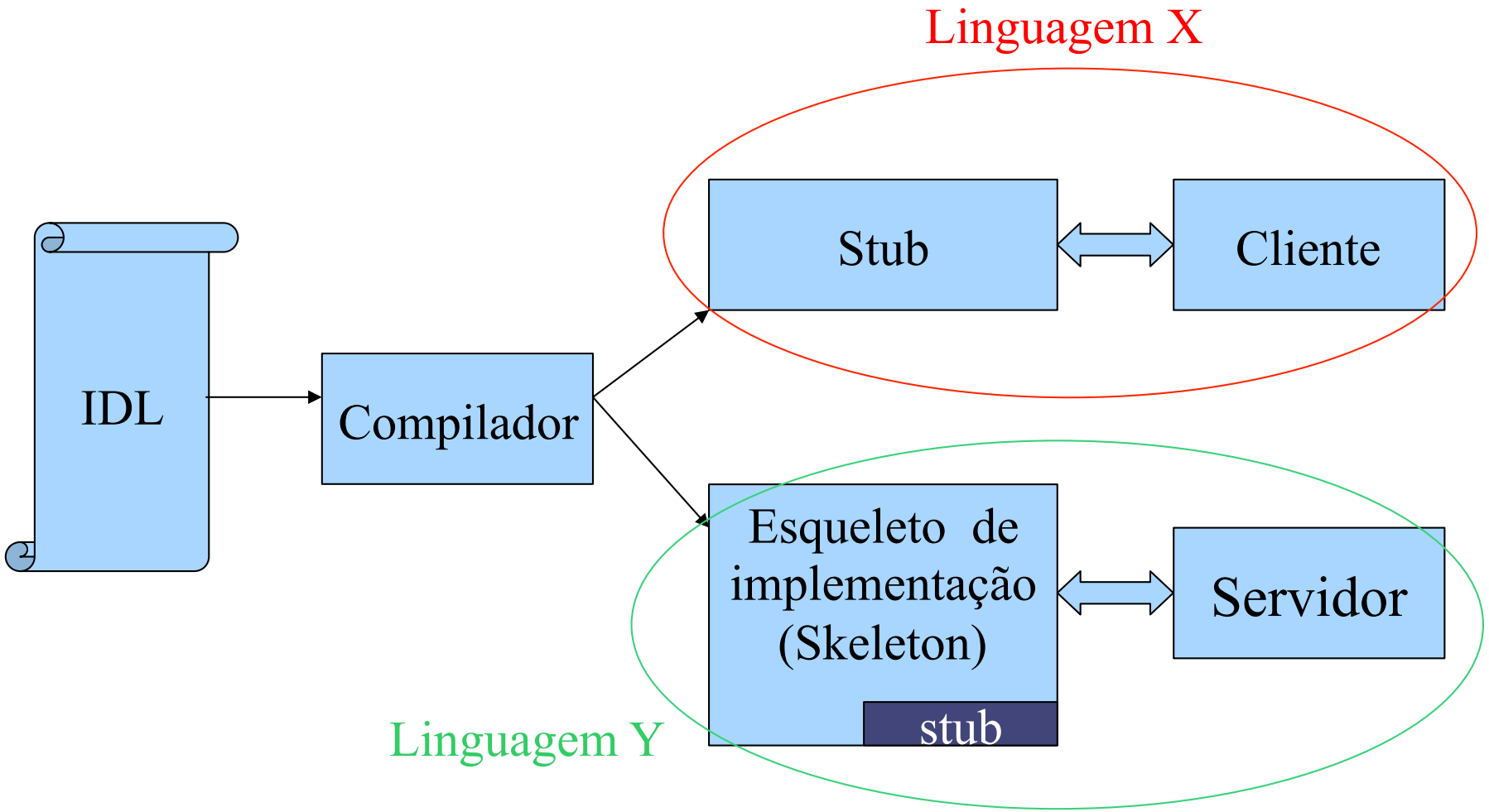
Stubs e Skeletons

- Client stub
  - gerado pelo compilador IDL, é o código que torna uma interface de um objeto CORBA disponível para um cliente
  - possui uma implementação dummy para cada método da interface
    - usa o ORB para transferir a chamada para o método remoto do objeto servidor
  - os stubs são incluídos junto com o código do cliente que usa essas interfaces

Stubs e Skeletons

- Server skeleton
  - também gerado pelo compilador IDL, é o código "framework" que serve de base para a implementação do objeto servidor
  - para cada método, o compilador IDL gera um método "vazio" no skeleton
  - o desenvolvedor é responsável por prover a implementação para cada um desses métodos vazios

Processamento da IDL



Servant e Objeto CORBA

- CORBA utiliza o termo Servant para um objeto, implementado na linguagem hospedeira (C++, Java etc) que implementa a funcionalidade de um objeto CORBA.
- Um Servant **representa** um objeto CORBA

Object Adapter

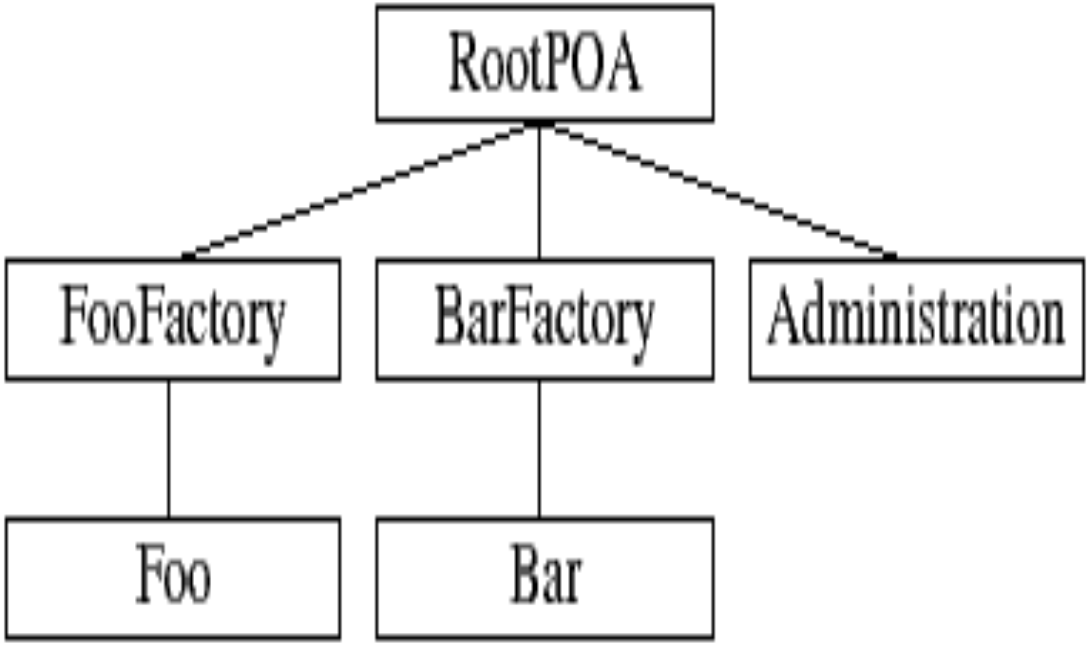
- Um *Object Adapter* serve como mediador entre o ORB e os *Servants*.
- Sua principal função é agrupar e gerenciar um conjunto de *Servants*
  - cria as referências para os objetos CORBA
  - permite a ativação e desativação dos *servants*
  - permite a criação dos *servants* sob demanda

POA – Portable Object Adapter

- O POA (*Portable Object Adapter*) é um *Object Adapter* padrão do CORBA introduzido pela OMG a partir de CORBA 2.0
- A definição do POA permitiu o desenvolvimento de aplicações portáveis entre diferentes ORBs

Hierarquia de POAs

- Podem existir diversas instâncias do POA, organizadas em uma hierarquia
- A raiz desta hierarquia é criada pelo ORB



POA Manager

- O POA Manager gerencia o estado do POA com relação ao tratamento das requisições:
  - HOLDING
    - Estado “off”. As requisições são enfileiradas.
  - DISCARDING
    - Estado “off”. As requisições são descartadas e a exceção CORBA::TRANSIENT é lançada para o cliente
  - ACTIVE
    - Estado“on”. As requisições são encaminhadas para os respectivos servants.
  - INACTIVE
    - Estado “off”. O ORB não executando.

Políticas de um POA

- A vantagem em organizar os servants em POAs diferentes é estabeler políticas (*policies*) de qualidade de serviço diferentes para cada conjunto. Exemplos:
  - O servant precisa ser criado antes das requisições chegarem ou podem ser criados sob-demanda?
  - Uma vez que o servant seja criado, ele permanecerá vivo indefinidamente ou pode ser destruído e substituído por um outro?
  - Existe um mapeamento um-para-um entre os servants e os objetos CORBA ou um mesmo servant pode representar mais de um?

IIOP - Internet Inter-ORB Protocol

- Protocolo de comunicação padronizado entre um *proxy* e seu objeto servidor.
- IIOP (Internet Inter-ORB Protocol)
  - Define o formato de representação dos valores trocados entre cliente e servidor.
  - Permite a comunicação entre objetos em plataformas diferentes.
  - Permite a comunicação entre diferentes implementações de CORBA.

IOR - Inter-ORB Reference

- CORBA define um padrão para a formação de referências de objetos, permitindo que objetos de ORBs diferentes possam se comunicar.
- IOR = endereço IP + porta *socket* + ID do objeto (Inter-ORB Reference).
- Um IOR possui uma representação em forma de *string*.

Interceptação

- Mecanismo que possibilita a interceptação de mensagens CORBA durante o fluxo de uma sequência requisição/resposta.
  - Noção de pontos de interceptação.
    - send_request / receive_request
    - send_reply / receive reply
  - Anexo de informações.