INE5380 - Sistemas Distribuídos

Object Request Broker
e
CORBA

Por: Léo Willian Kölln - 0513227-4

Novembro de 2006

ORB
Object Request Broker

- ORB aqui será tratado como um Middleware que permite a construção de programas que se comunicam através de uma rede de computadores.

ORB
Object Request Broker

- Algumas implementações:
  - **ORBexpress** - *real-time ORBs - Objective Interface Systems*
  - **DCOM** - *Distributed Component Object Model - Microsoft*
  - **RMI** - *Remote Method Invocation Protocol - SUN*
  - **CORBA** - *Common Object Request Broker Architecture - OMG*
  - **OpenFusion** - *real-time, embedded, enterprise CORBA ORBs - PrismTech*
  - **SimpleORB** - *small, "non-CORBA" ORB*

CORBA

Common Object Request Broker Architecture

- CORBA é uma especificação definida pela Object Management Group (OMG).
- Sua principal característica é permitir a interoperabilidade entre componentes de software escritos em diversas linguagens e executando em diversos "h ost".
- O sucesso de sua interoperabilidade é que CORBA define interfaces, e não código!

CORBA

Common Object Request Broker Architecture

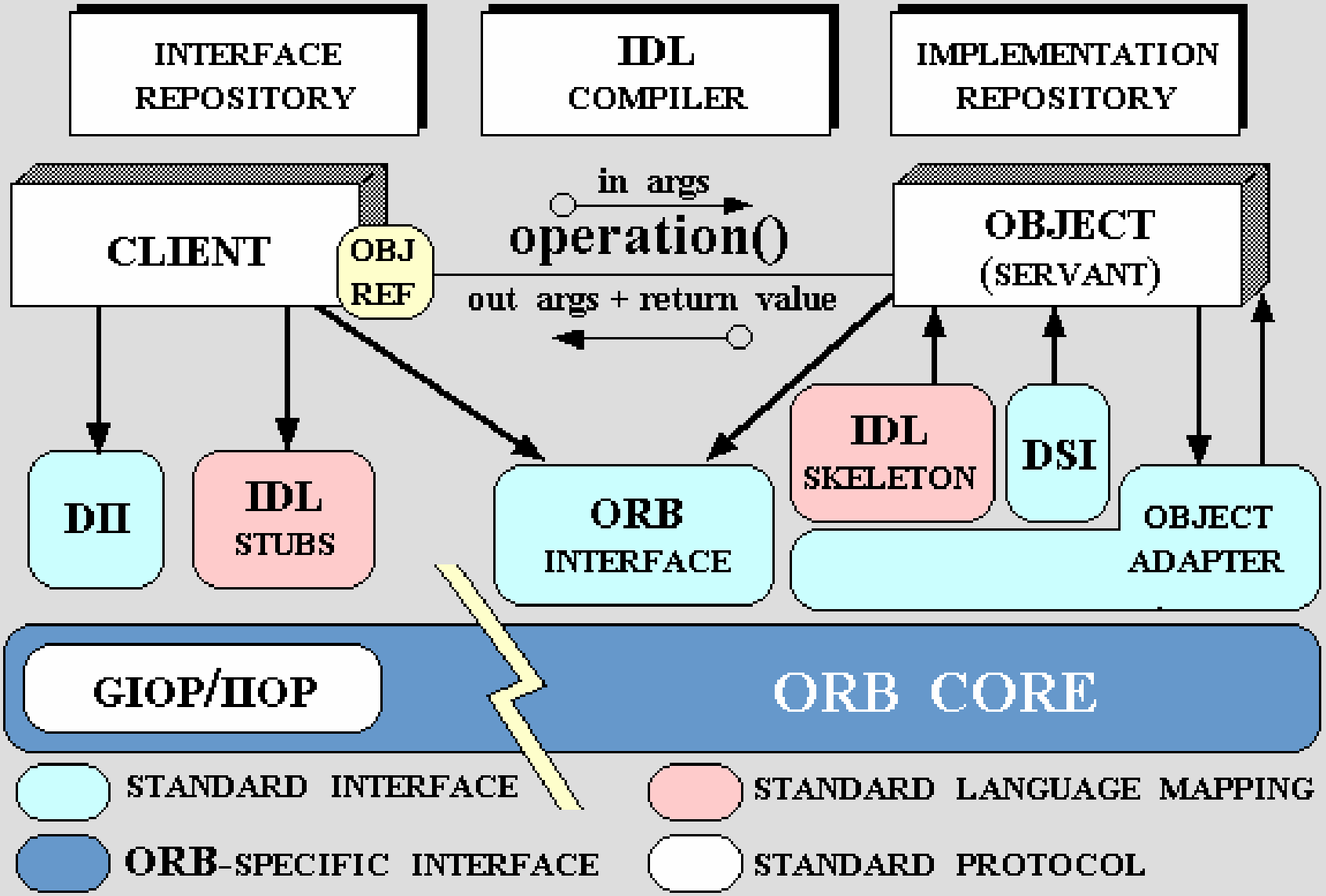
- Define a forma dos componentes e a maneira como eles se inter-relacionam.
- Oferece suporte de execução que mascara camadas intermediárias do sistema distribuído (linguagens, compiladores, SO, máquinas, rede)

CORBA
Common Object Request Broker Architecture

- Pertencem a uma aplicação utilizando CORBA:
  - Objetos
    - São inteligentes e podem estar em qualquer Servidor.
    - Conhecidos por sua interface.
    - Independem da linguagem.
  - Servidor
    - “Hospeda” e “executa” os objetos.
  - Cliente
    - Faz “uso” dos objetos e conhece apenas suas interfaces.
    - Detalhes de localização, implementação busca e outros são completamente abstraídos.

CORBA

CORBA ORB Architecture



CORBA
CORBA ORB Architecture

- **Object** – Esta é uma entidade de programação CORBA que consiste de uma identidade, uma interface e uma imlpementação, que é conhecida como "Servant".
- **Servant** – Esta é a implementação da entidade que define as operações que suportam a interface CORBA IDL.

CORBA
CORBA ORB Architecture

- **Client** – Esta é a "é ntidade programa" que invoca operações sobre implementações de objetos. Acessar os serviços de um objeto remoto é transparente para quem o chama. Na prática, é simples como chamar o método de um objeto.

CORBA
CORBA ORB Architecture

- **Object Request Broker (ORB)** – A ORB prove um mecanismo para a comunicação transparente de requisições entre clientes e uma implentação de um objeto alvo. A ORB simplifica a programação de sistemas distribuídos, desacoplando o cliente dos detalhes da invocação de um método. Isto faz com que as requisições de um cliente pareçam locais. Quando um cliente invoca uma operação, a ORB fica responsável por encontrar a implementação do objeto, sua ativação e se necessário, entregar a requisição ao objeto e retornar qualquer resposta a quem o invocou.

CORBA
CORBA ORB Architecture

- **ORB Interface** – A ORB é uma unidade lógica que pode ser implementada de diferentes maneiras. Para separar a aplicação dos detalhes de implementação, CORBA define uma interface de abstração para a ORB. Esta interface prove várias funções auxiliares que provem, por exemplo, a conversão de referências de objetos em strings e vice versa, e criando listas de argumentos para as requisições feitas atravéz da Interface de invocação dinâmica (Dynamic Invocation Interface).

CORBA
CORBA ORB Architecture

- **CORBA IDL Stubs and Skeletons** – Servem como "cola" entre as aplicações cliente e servidor, e a ORB. A Transformação entre as definições da *CORBA IDL* e a linguagem de programação alvo é automatizada pelo *CORBA IDL Compiler.* O Uso deste compilador reduz o potencial de inconsistências entre os Stubs dos clientes e os Skeletons dos Servers e aumenta a oportunidade para otimizações automáticas pelo compilador.

CORBA
CORBA ORB Architecture

- **Dynamic Invocation Interface (DII)** – Esta interface permite o acesso direto do cliente aos mecanismos de requisição providos pela ORB. Aplicações usam o DII para efetuar as requisições a objetos de maneira dinâmica sem que seja necessária a lincagem de interfaces específicas de stubs. Diferentemente das Stubs IDL (que permitem apenas chamadas no estilo RPC), a DII também permite aos clientes fazer requisições não bloqueantes derreferenciadas síncronas (operações de envio e recebimento separadas) e chamadas “oneway” (apenas envio).

CORBA
CORBA ORB Architecture

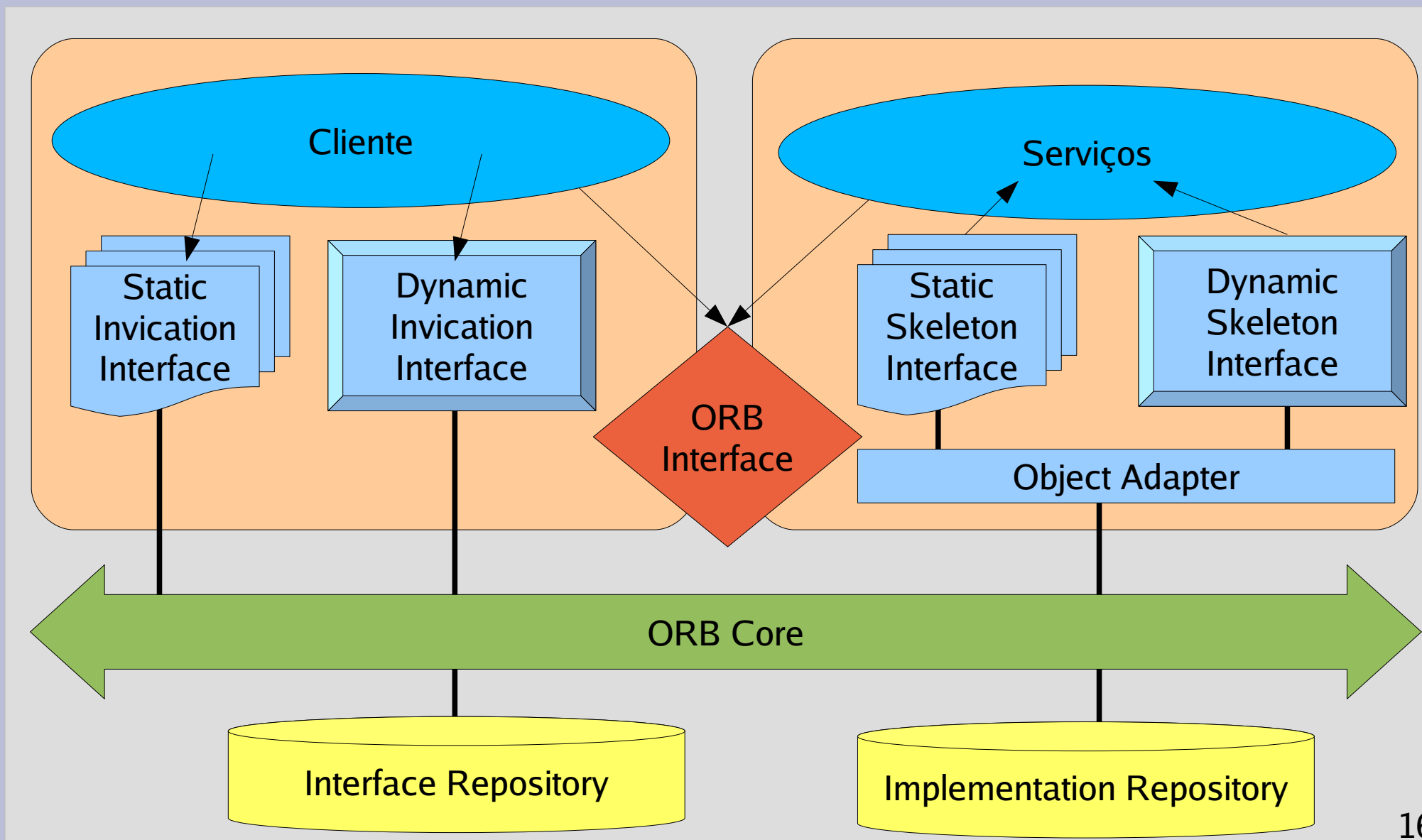
- **Dynamic Skeleton Interface (DSI)** – Esta é a versão “client-side” análoga a DII que é “server-side”. A DSI permite a ORB entregar requisições a uma implementação de um objeto, mas que não tem conhecimento em tempo de compilação sobre que tipo de objeto o esta implementando. O cliente que faz uma requisição não tem idéia se a implementação esta usando um tipo específico de Skeleton IDL ou está usando Skeletons dinâmicos.

CORBA
CORBA ORB Architecture

- **Object Adapter** – Esta interface ajuda a ORB a entregar as requisições a um objeto e a ativalo. Mais importante, um “Object Adapter” associa a implementação de um objeto com a ORG. Os adaptadores de objetos podem ser especializados para prover suporte a certos estilos de implementação de objetos (assim como adaptadores de objetos OODB para persistência e Livrarias de Objetos para objetos não remotos).

ORB (CORBA)

Anatomia



Definição e Ativação de Objetos CORBA

- **Geração dos Serviços**
  - Definição da Interface OMG-IDL
  - Preparação para invocação estática de métodos
  - Preparação para invocação dinâmica de métodos
- **Inicialização**
  - do servidor de serviços
    - do cliente de serviços
- **Ativação dos objetos**

Geração de Serviços
Invocação Estática de Métodos

1. **Definir Interface do serviço em OMG-IDL**
2. **Projeção da interface do serviço para linguagem específica (C++, Java)**
    1. **Executar pré-compilador na interface e gerar várias definições de classes necessárias ao servidor**
        1. classe do skeleton, etc.
        2. Import Files (descreve objetos para o Interface Repositary)
3. **Implementar/adicionar classe correspondente ao serviço**
4. **Compilar todas as classes e gerar**
    1. stubs do cliente, skelletons do servidor, código do serviço, import files
5. **Carregar as definições no Interface Repositary**
    1. Feito por utilitário apropriado
6. **Registrar objetos no Implementation Repositary**
    1. Feito pelo Object Adaptor
7. **Instanciar os objetos no servidor**

Geração de Serviços
Invocação Dinâmica de Métodos

1. **Obter referência e interface do objeto**
    1. Get_interface() : descrição completa da interface
2. **Obter descrição do método**
    1. lookup_name() : Busca no Interface Repositary
    2. describe () : assinatura IDL do método
3. **Preparar lista de argumentos**
    1. create_list(), add_arg()
4. **Preparar a invocação**
    1. creat_request()
5. **Realizar a invocação remota**
    1. Usando RPC: invoke()
    2. Send/receive: send(), get_response()
    3. Datagrama (one-way): send ()s

Servidor de Serviços
Inicialização

1. **Inicializar o bus CORBA**
    1. ORB_init() que retorna pseudo-objeto ORB
    2. Inicializar o Adaptador de Objetos
    3. BOA_init() retorna referência do BOA
    4. Instanciar os objetos que representam os serviços
    5. Obter as referências dos objetos no Adaptador de Objetos
2. **Objetos já foram registrados na etapa de geração do serviço**
    1. Difundir as referências de objetos para clientes
3. **Diretamente (exibição em arquivo ou tela)**
4. **Indiretamente (Serviço de Nomes)**
    1. rebind(serviço)
    2. Colocar-se à espera de requisições
    3. impl_is_ready()

Cliente do Serviço
Inicialização

1. **Inicializar o bus CORBA**
    1. Criar objeto ORB: ORB_init()
2. **Descobre serviços disponíveis**
    1. uso da API string_to_object ()
    2. Diretamente
        1. *Recuperação a partir de arquivo*
    3. Indiretamente
        1. *resolve_initial_references (“Serviço de Nomes”);*
        2. *list_initial_services() retorna string names;*
3. **Obtem referências de objetos para serviços desejados**
    1. Uma referência equivale a uma instância do stub de cliente
        1. *(i) obj = resolve (“nome”);*
        2. *(ii) serviço = narrow(obj)*
4. **Efetuar invocações**

Política de Ativação do Objeto no Servidor

- **BOA Shared Server**
  - Múltiplos objetos podem residir num mesmo
- processo
- **BOA Unshared Server**
  - Cada objeto reside num processo servidor diferente
- **BOA Server-per-Method**
  - Um novo servidor é instanciado a cada invocação de método
- **BOA Persistent Server**
  - Servidores são ativados externamente (fora da BOA)
  - Uso da política shared-server

Bibliografia

- http://twiki.im.ufba.br/pub/MAT167SD/Material/Corba-3pp.pdf
- http://pt.wikipedia.org/wiki/CORBA
- http://en.wikipedia.org/wiki/Object_request_broker
- http://www.cs.wustl.edu/~schmidt/corba-overview.html