Middleware CORBA

Prof. Me. Sérgio Carlos Portari Júnior

CORBA – Introdução

- Ambientes que visam desenvolver aplicações que precisam de um processamento paralelo e distribuído deverão saber lidar com algumas dificuldades.
- Isto decorre da heterogeneidade de hardware, sistemas operacionais, linguagens de programação, protolocos de comunicação, arquiteturas de computadores e diferentes formas de se representar dados.

CODRBA – Introdução

- Para resolver esses problemas, percebeu-se a necessidade de se ter transparência de localização de recursos computacionais, suporte universal de representação de dados, interoperabilidade entre componentes de uma aplicação distribuída, reutilização de componentes, integração de código.

CORBA – Introdução

- O modelo de arquitetura CORBA propõe a interoperabilidade local e/ou remota entre aplicações que têm como base a tecnologia orientada a objetos.
- Com isso, é possível permitir que objetos de sistemas distribuídos comuniquem-se de forma transparente, independentemente do local onde se encontram, do protocolo que utilizam, da linguagem em que foram implementados, da plataforma e do sistema operacional em que estão rodando

CORBA – Conceito

- Common Object Request Broker Architecture (CORBA) – ou Arquitetura Comum de Agente de Requisição de Objetos.
- Trata-se de um padrão proposto pela Object Manager Group (OMG), uma organização internacional da indústria de software que estabeleceu uma estrutura comum para o gerenciamento de objetos distribuídos, conhecida como Object Manager Architecture (OMA) – Arquitetura de Gerência de Objetos.

CORBA – Conceito

- Seus principais componentes são:
- **Núcleo do CORBA o Object Request Broker (ORB)** – manipulam as requisições dos objetos. Um intermediário entre o cliente e o servidor (objeto).

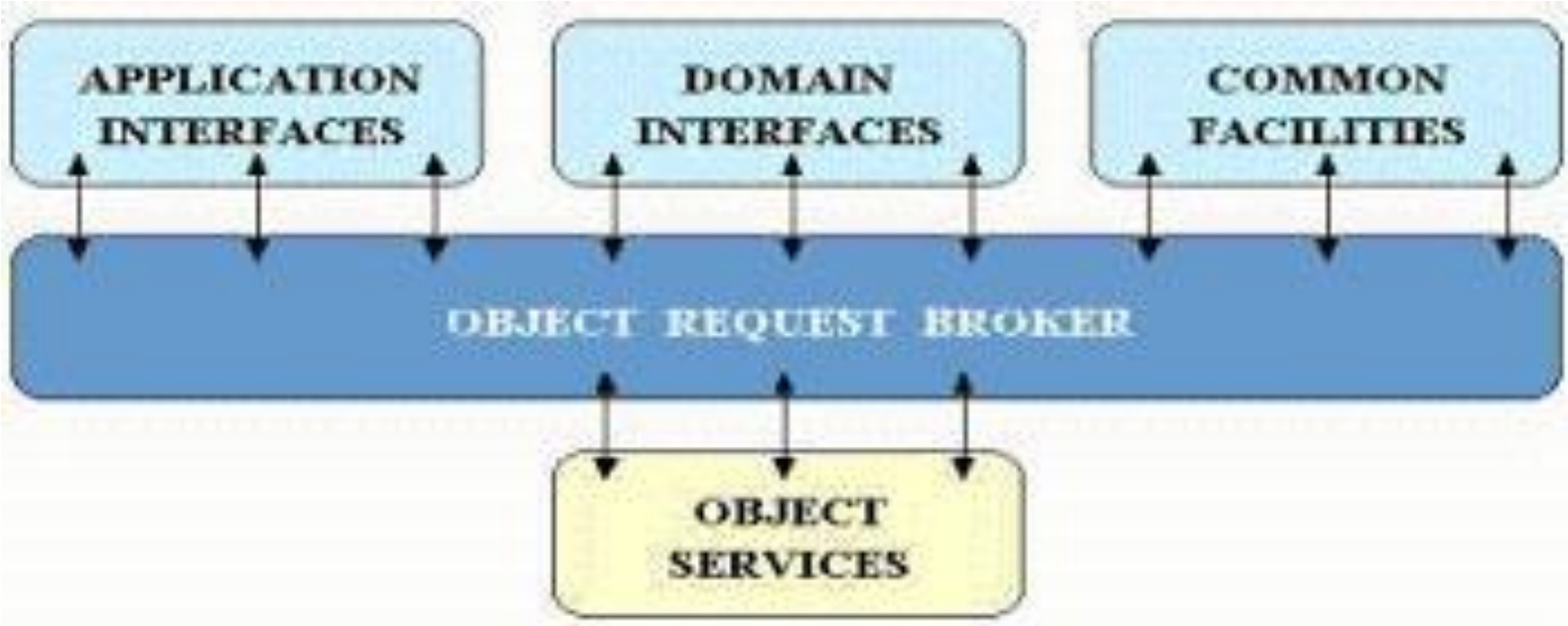
- **Serviços CORBA** – definem serviços que ajudam a gerenciar e a manter objetos. Exemplo: serviço de nomes, eventos, ciclo de vida, tempo, persistência, etc.

CORBA – Conceito

- **Facilidades CORBA** – definem facilidades e interfaces no nível de aplicação, manipulação de dados e armazenamento.

- **Objetos de aplicação** – são os objetos que podem ser considerados visíveis ao nível de aplicação.

CORBA – Conceito



COMO FUNCIONA O CORBA?

- Para cada sistema de rede o CORBA permite que seja definida uma Linguagem de Definição de Interface (IDL).
- Essas interfaces descrevem os serviços que são oferecidos.
- Trata-se de uma linguagem puramente declarativa, que ao ser compilada irá gerar o stubs e o skeletons (responsável pela comunicação entre objetos).
- Desta forma, a aplicação cliente pode acessar facilmente todos os serviços oferecidos.

COMO FUNCIONA O CORBA?

- Além disso, o CORBA inclui uma estrutura de execução, fornecendo diversas funcionalidades básicas importantes como localização e ativação automática de serviços, comunicação, controle de transações e segurança.

- Em linhas gerais, a arquitetura CORBA é composta pelos seguintes componentes :

COMO FUNCIONA O CORBA?

- Do lado do cliente, temos os seguintes itens:
- **Stubs Clientes** – trata-se de uma interface estática gerada através da compilação de uma IDL. Compõe uma mensagem que contém uma identificação e protótipos dos métodos invocados a um servidor.
- **Interface de Invocação Dinâmica (DII)** – permite que o cliente invoque um método no servidor sem que tenha conhecimento, em tempo de compilação, de sua interface.
- **Repositório de Interfaces** – contém uma base de dados com a definição de todas as interfaces de serviços conhecidos pelo ORB.

COMO FUNCIONA O CORBA?

- Do lado do objeto CORBA servidor, temos os seguintes itens:
- **Skeletons** – interface estática para os serviços (métodos) remotos. É responsável por receber as requisições do cliente e repassá-las para o servidor.
- **Interface de Skeletons Dinâmica (DSI)** – é semelhante ao DII, porém ocorre no lado do servidor. Fornece um mecanismo que permite, em tempo de execução, que servidores entreguem requisições de um ORB para uma implementação de objeto sem que tenha conhecimento, em tempo de compilação, de sua interface;
- **Repositório de Implementação** – é um repositório, em tempo de execução, para as classes que um servidor suporta, os objetos instanciados e suas identificações;

COMO FUNCIONA O CORBA?

- **Adaptador de Objetos (OA)** – responsável por: registrar as classes servidoras no repositório de implementação; instanciar os objetos chamados em tempo de execução, de acordo com a demanda dos clientes; receber as chamadas para os objetos e repassá-las aos mesmos; gerar e gerenciar as referências de objetos (ORB).
- **Interface ORB** – permite o acesso às funcionalidades não oferecidas pelas demais interfaces, pois não depende da interface do objeto ou do adaptador do objeto.

IDL – Interface Definition Language

- A IDL é uma linguagem neutra de especificação de interfaces, usada no desenvolvimento de objetos servidores para a declaração dos métodos ou serviços.
- A descrição IDL de um componente/objeto CORBA estabelece uma relação contratual entre este e seus cliente e servidor, definindo a interface pela qual estes poderão interagir, no entanto sem fixar detalhes de implementação dos objetos.

IDL – Interface Definition Language

- Ela é independente de linguagem de programação (C++, Java, Smalltalk, ...).
- Atualmente oferece uma série de mapeamentos como C, C++, Ada, Smalltalk, COBOL e Java.
- Em particular, a IDL é usada para especificação de:
  - atributos de componentes;
  - classes pais (herança);
  - exceções;
  - eventos tipados (sinais).

IDL – Interface Definition Language

- A sintaxe da IDL pode ser caracterizada como um subset de C++ mais diversas palavras chaves para programação distribuída.
- A estrutura geral de um arquivo IDL é ilustrada a seguir:

module <identificador>
// define um escopo de nomes, equivalente
// a namespaces em C# ou packages em Java

IDL – Interface Definition Language

{
<declarações de tipos de dados>;
<declarações de constantes>;
<declarações de exceções>;
interface <identificador> [:<herança pcional>]
{
  <declarações de tipos de dados>;
  <declarações de constantes>;
  <declarações de exceções>;

IDL – Interface Definition Language

[<tipo>] <identificador de método> (<parâmetros>)
[raises <exception>] [context];

.

.

[<tipo>] <identificador de método> (<parâmetros>)
[raises <exception>] [context];

.

.

}