

[< VOLTAR](#)

Remote Procedure Call: RMI, CORBA e DCOM

Descrever o conceito das soluções baseadas Remote Procedure Call

NESTE TÓPICO

- > JAVA RMI
- > CORBA
- > DCOM
- > Referências



O desenvolvimeno de aplicações Web em diversas plataformas heterogêneas vem crescendo consideravelmente, principalmente o aumento de dispositivos móveis, embarcados e obíquos conectados à Internet. Neste sentido que o surgimento de tecnologias no desenvolvimento de software distribuído também vem evoluindo em direção da reusabilidade, flexibilidade e modularidade. Estes três importantes elementos da qualidade de software também contribui na manutenção evolutiva e corretiva, além da redução de custos.

Neste tópico, veremos três soluções que implementam o conceito de **Remote Procedure Calls** (RPC) ou Chamadas de Procedimentos Remotos no modelo de desenvolvimento de software distribuído e utilizam uma Linguagem de Definição de Interface (IDL) para viabilizar a interoperabilidade e portabilidade da solução. A Figura 1 mostra a estrutura básica do ambiente cliente/servidor, aplicações e serviços em diversas linguagens e as IDLs.

Linguagem de Definição de Interface

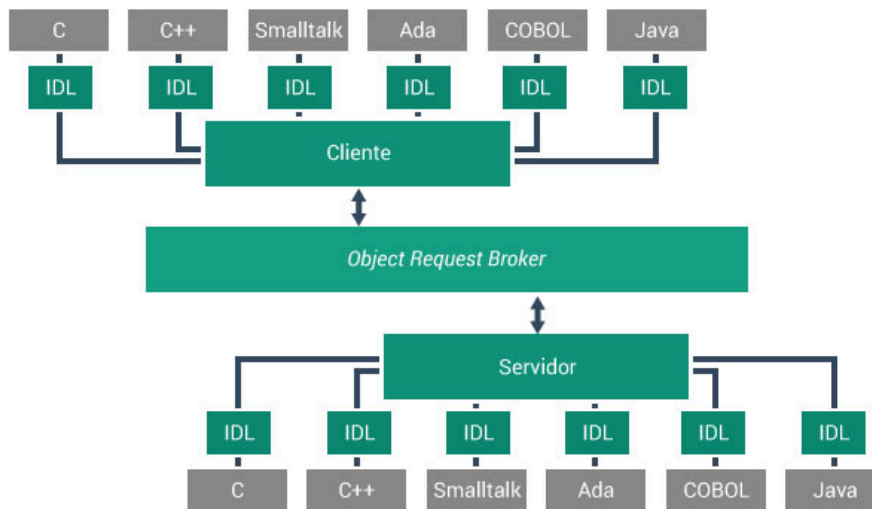


Figura 1 - Arquitetura Cliente/Servidor e a IDL

Fonte: ROQUE, Vitor. Arquiteturas Distribuídas Cliente/Servidor: CORBA, DCOM e Java RMI. Universidade de Aveiro, Portugal, 1999.

JAVA RMI



O JAVA RMI (Remote Method Invocation) é uma solução de RPC para o ambiente de desenvolvimento (JDK - *JAVA Development Kit*) e para a linguagem JAVA. Assim como nas soluções que veremos a seguir, o RMI também é uma solução Cliente/Servidor. A dependência da linguagem do RMI está na sua linguagem que a concebeu, pois deve ser escrita, obrigatoriamente, em JAVA. Mas como a linguagem JAVA é portátil, ou seja, não é limitada a um único sistema operacional, a mesma pode ser executada em computadores com MS-Windows, MAC-OSX, UNIX e GNU/LINUX. Isto é, os objetos podem estar em computadores diferentes, com sistemas operacionais diferentes, que a *JAVA Virtual Machine* (JVM) trará portabilidade na solução.

A Figura 2 a seguir mostra a estrutura dos componentes da arquitetura RMI.

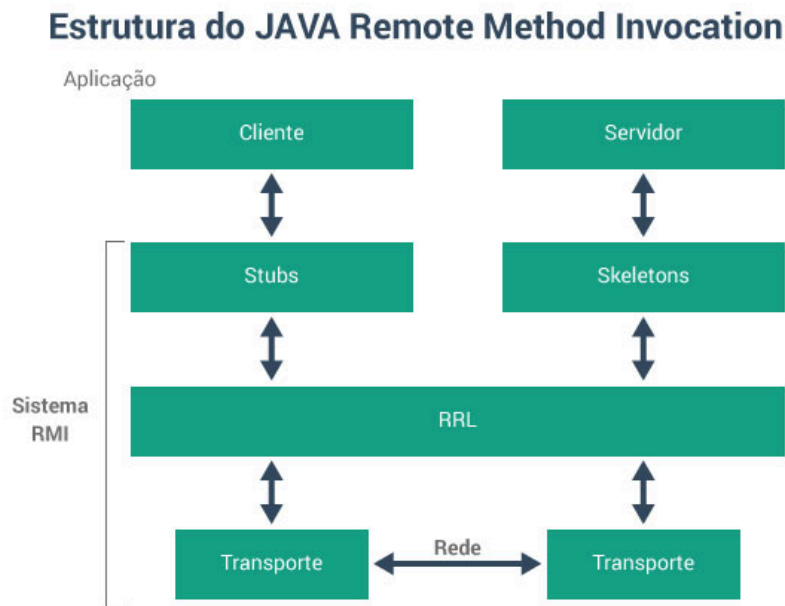


Figura 2 - Estrutura dos componentes da arquitetura JAVA Remote Method Invocation - RMI

Fonte: ROQUE, Vitor. Arquitecturas Distribuídas Cliente/Servidor: CORBA, DCOM e Java RMI. Universidade de Aveiro, Portugal, 1999.



Analizando o ambiente do RMI através da Figura 1, o mesmo possui quatro camadas, sendo a primeira camada (de cima para baixo) dos componentes de interface do Cliente e do Servidor. As demais camadas (três) são referentes ao ambiente RMI.

Na segunda camada, temos os Stub e Skeleton que é responsável pelo gerenciamento das interfaces dos objetos remotos que respondem ao cliente e ao servidor. O *Remote Reference Layer* (RRL) é a camada responsável pelo gerenciamento de atividades e da comunicação entre os objetos remotos e com as JVMs. Finalmente, a última camada de transporte é responsável pela comunicação entre clientes e servidores.

CORBA

O CORBA (*Common Object Request Broker Architecture*) é uma solução de desenvolvimento de software baseado em duas características importantes: a orientação a objetos e o modelo Cliente/Servidor. A solução foi criada em 1989 pela OMG (*Object Management Group*) para o desenvolvimento de soluções heterogêneas e orientadas a objetos CORBA.

As solicitações que são enviadas pelos cliente e as respostas do servidor ocorrem através de ORB (*Object Request Broker* ou Agente de Requisição de Objetos). Cada ORB possui uma especificação de serviços e regras de negócio diferentes. Assim, cada cliente poderá fazer a sua requisição ao objeto ORB que atenderá dentro do que foi especificado. Essas especificações, como vimos nos tópicos anteriores, são denominados conceitualmente

por **Interface Definition Language** (IDL). Como a solução poderá ter diversos objetos ORB, cada um deles recebe um identificador único para que seja possível diferenciá-los tanto do lado do cliente, quanto do servidor.

Cada objeto na arquitetura CORBA pode ser desenvolvido em uma linguagem de programação diferente e estar em localizações geográficas diferentes, e este cenário é totalmente favorável a tudo que estamos discutindo neste disciplina, principalmente nos temas de heterogeneidade, interoperabilidade e portabilidade da solução.

Como sabemos, o cliente possui contato diretamente com a interface do software. A interface, por sua vez, deve ter contato direto com os objetos que devem receber as suas requisições. Pois bem, a dúvida que fica é: De que maneira o cliente se comunica com o objeto?

A resposta está nos dois componentes da arquitetura CORBA: Os **Stubs** e o **Skeletons**

Os **Stubs** ficam na extremidade do cliente e os **Skeletons** nos objetos-servidor. Assim, o cliente apenas informa qual objeto que precisa e o stub se encarrega de empacotar a mensagem para o objeto. Desta forma, o cliente não precisa se preocupar em saber onde o objeto está localizado geograficamente, ou seja, se está local ou distante. O ORB encaminhará a mensagem ao **Skeleton** que entregará ao objeto que processará a requisição. A comunicação da resposta (volta) ao cliente funciona exatamente como vimos na ida.

Em suma, o CORBA é um RPC da OMG e permite realizar invocação remota de objetos pela rede local, de média ou longa distância.

A Figura 3 mostra os componentes e a comunicação entre eles na arquitetura CORBA.

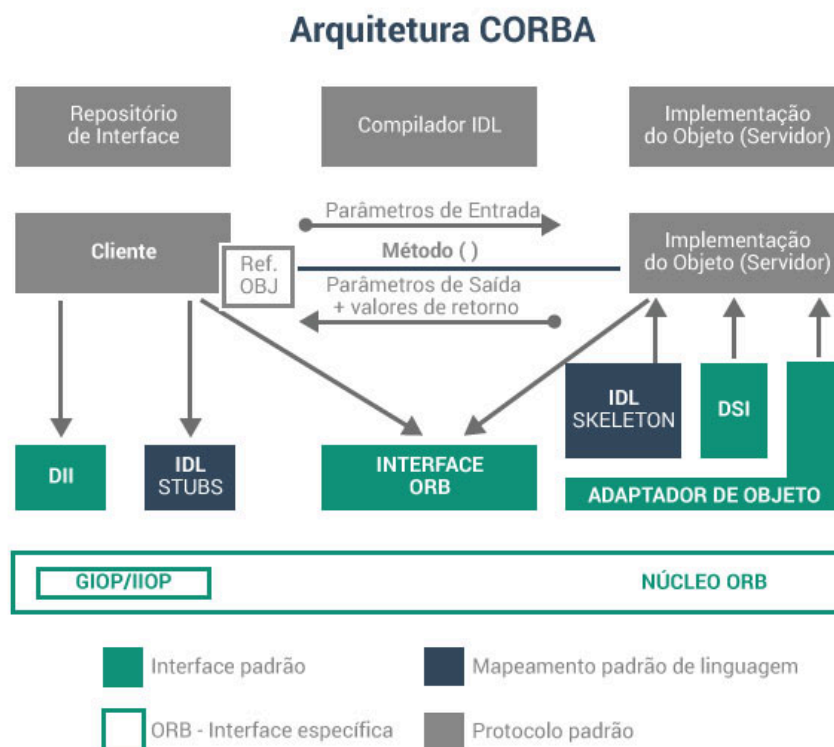


Figura 3 - Estrutura da arquitetura cliente/servidor do CORBA

Fonte: BÓRIO, Rubens. Análise Comparativa entre as Especificações de Objetos Distribuídos DCOM e CORBA utilizando o Ambiente de Desenvolvimento DELPHI, 2000

DCOM

O DCOM (*Distributed Component Object Model*) é uma solução distribuída do COM (*Component Object Model*) da Microsoft e possui dependência apenas do sistema operacional que deve ser o MS-Windows. O início do COM se deu a partir do *Object Linking and Embedding* (OLE) criado em 1990 com objetivo de permitir que documentos em diversas aplicações pudessem manipular tipos de dados multimídia. Além disso, as soluções como OLE *Automation* (sucessor do OLE), ActiveX, COM e COM+ (COM *Plus*) antecederam o DCOM. O COM, por sua vez, foi projetado para ser uma solução orientada a objeto e interoperável.

A Figura 4 mostra a estrutura de comunicação COM.

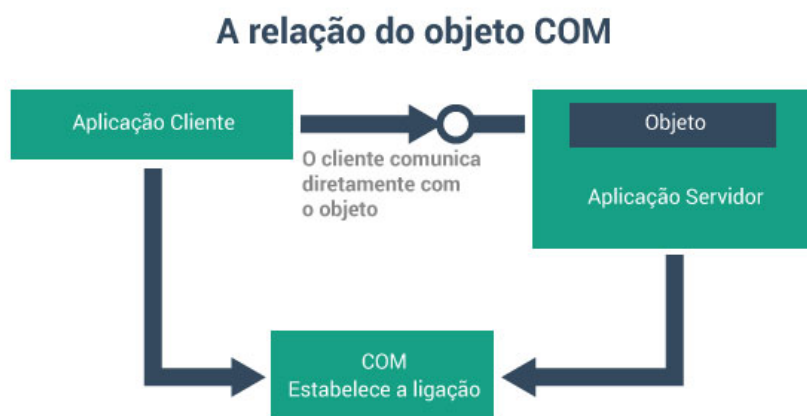


Figura 4 - Comunicação entre o cliente, componente COM e o servidor.

Fonte: ROQUE, Vitor. Arquitecturas Distribuídas Cliente/Servidor: CORBA, DCOM e Java RMI. Universidade de Aveiro, Portugal, 1999.

A Figura 5 mostra a comunicação entre os elementos da arquitetura DCOM. Assim, é importante destacar que o DCOM é uma solução de RPC que oferece serviços de orientação a objetos aos clientes e componentes, além de oferecer os serviços de provedor segurança (*Security Provider*) para que os pacotes gerados estejam alinhados ao padrão DCOM. Os objetos devem ser solicitados através do *Service Control Manager* (SCM) e são identificados da classe chamada *Class Identifier* (CLSID). O DCOM faz o uso do DCE (*Distributed Computing Environment* ou Ambiente de Computação Distribuída) para implementar os conceitos de RPC.

A arquitetura DCOM pode ser desenvolvida nas linguagens de programação C# (C *Sharp*), VB.Net (*Visual Basic .Net*) e ASP.Net (*Active Server Pages .Net*) que são linguagens proprietárias da Microsoft. Neste sentido, o DCOM

pode ser criado para consumir recursos de um *Web Service* (WS) e, neste caso, o WS pode ser criado em outra linguagem de programação, como por exemplo o JAVA e o PHP (*Hypertext Preprocessor*).

Arquitetura DCOM

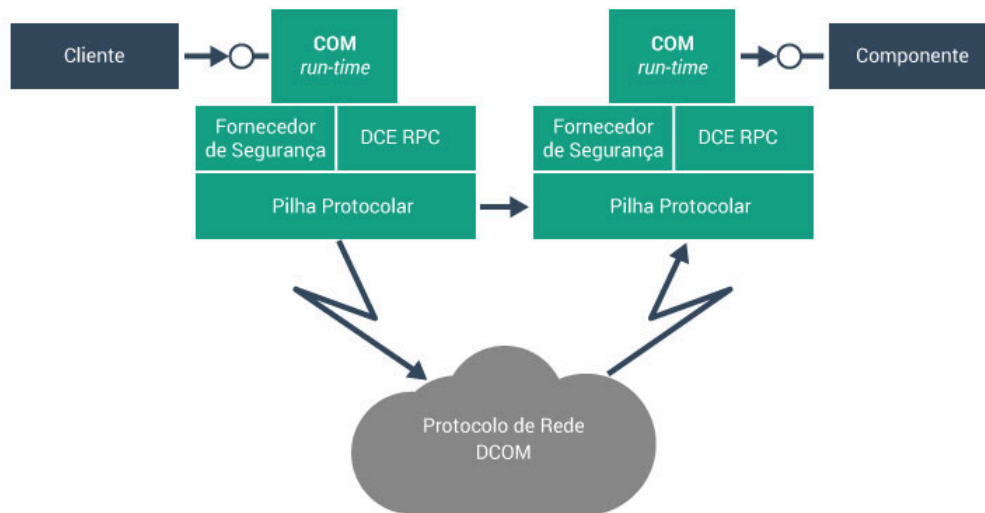


Figura 5 - Comunicação entre o cliente, componente DCOM e o servidor.

Fonte: ROQUE, Vitor. Architecturas Distribuídas Cliente/Servidor: CORBA, DCOM e Java RMI. Universidade de Aveiro, Portugal, 1999.



Quiz

Exercício

Remote Procedure Call: RMI, CORBA e DCOM

INICIAR ➤

Referências

TANENBAUM, Andrew S., STEEN, Maarten van. Sistemas Distribuídos: Princípios e Paradigmas. Segunda Edição. Prentice Hall, 2002.

DE SOUZA, Anamélia Contente; MAZZOLA, Vitório Bruno. Implementando aplicações distribuídas utilizando CORBA: um estudo de caso voltado à área de banco de dados. INFOCOMP Journal of Computer Science, v. 1, n. 1, p. 31-35, 2004.

FREITAS, André Luis Castro de. Proposta de um sistema de banco de dados distribuído e replicado utilizando serviço RMI-JAVA. 2003.

ROQUE, Vitor. Architecturas Distribuídas Cliente/Servidor: CORBA, DCOM e Java RMI. Universidade de Aveiro, Portugal, 1999.

BÓRIO, Rubens. Análise Comparativa entre as Especificações de Objetos Distribuídos DCOM e CORBA utilizando o Ambiente de Desenvolvimento DELPHI, 2000.

ULLRICH, Edgard Charles. Comparativo da Tecnologia DCOM em Diversos Ambiente de Desenvolvimento.



Avalie este tópico



ANTERIOR

Modelo Cliente/Servidor e Sockets em Java

Biblioteca

(<https://www.uninove.br/conheca->

a-

uninove/biblioteca/sobre-

a-

biblioteca/apresentacao/)

Portal Uninove


(<http://www.uninove.br>)

Mapa do Site



Índice

© Todos os direitos reservados

Ajuda? 
(<https://www.uninove.br/ava/uninove/ava.php?idCurso=>)

