Objetos Distribuídos

Sergio Mainetti Jr., M.Sc.

Curitiba, Junho de 1997

Artigo escrito derivado de aulas nos cursos de Pós-Graduação em Engenharia de Software e Pós-Graduação em Orientação a Objetos (PUC-PR, 1992 a 1995), Pós-Graduação em Engenharia de Software (CEFET-PR, 1997) e Pós-Graduação em Objetos Distribuídos (Visionnaire/FAE-CDE, 1997, 1998), ministradas pelo autor.

Este artigo foi publicado, sem títulos e sub-títulos, sem figuras, sem as referências e com alguns cortes, na edição de Janeiro de 1998 da revista especializada Developer's Magazine. Partes deste artigo foram posteriormente publicados no jornal especializado Computerworld Brasil de 18 de agosto de 1997 e na revista Datamation de Março de 1998.

Objetos Distribuídos

Sergio Mainetti Jr. [*]

Abstract

This paper analyses the "new era" of computing in most of the corporations, where distributed systems and object oriented systems are already a reality. The union of these two technologies, distributed objects, is discussed in more detail, with a description of some models available in the industry today. The components of the OMG's CORBA model are described. Finally, the impact of distributed objects on the Internet is analyzed.

Palavras-chave:

Sistemas Distribuídos; Orientação a Objetos; Cliente/Servidor; Objetos Distribuídos; Java; OMA; CORBA; Internet.

1. Introdução

Existe uma "nova ordem" na computação. Não vivemos mais em um mundo ditado por computadores de grande porte, voltados para o processamento centralizado. Vários computadores pessoais e estações de trabalho (*workstations*) tomaram conta dos ambientes de trabalho de todas as corporações, e a integração entre estes computadores buscando o processamento distribuído é uma necessidade. Os sistemas que devem executar nestes ambientes também têm características diferenciadas, e para atender a estas novas características, o uso da tecnologia de orientação a objetos está passando a fazer parte do processo de desenvolvimento de software de todas as corporações.

Este artigo procura mostrar a nova realidade presente em grande parte das empresas atuais, que estão usando sistemas distribuídos e orientação a objetos, e descrever a união destas duas tecnologias: objetos distribuídos. Modelos de objetos distribuídos são analisados e é dado um enfoque ao modelo CORBA da OMG. A arquitetura OMA e os componentes do CORBA são descritos mais detalhadamente. Finalmente é analisado o impacto da união da área de objetos distribuídos com o imenso mercado da Internet.

[*] Sergio Mainetti Jr. é Mestre em Ciência da Computação pela Northeastern University, Boston, EUA. É consultor em Sistemas Distribuídos e Orientação a Objetos. Foi professor de cursos de Graduação e Pós-graduação na PUC-PR. Atualmente é professor do curso de Pós-graduação em Engenharia de Software pelo CEFET-PR e do curso de Pós-graduação em Objetos Distribuídos pela Visionnaire/FAE-CDE. Coodenador do evento Objetos Distribuídos, o maior da área no Brasil, que é realizado anualmente. Diretor da Visionnaire.
Email: mainetti@visionnaire.com.br.

2. A Nova Geração Cliente/Servidor

Não podemos negar que a influência da filosofia Cliente/Servidor (*Client/Server*) em todas as empresas que utilizam informática foi imensa. Hoje o ambiente cliente/servidor está presente em praticamente todas estas empresas, e contribuiu como um grande passo a partir de sistemas de *mainframe* centralizados.

Ultimamente temos observado alguns autores alertando para a "Nova Onda" de cliente/servidor, ou até mesmo em alguns casos, a "Segunda Geração" de cliente/servidor. Estes autores buscam chamar a atenção de profissionais de desenvolvimento de software para o fato de que, agora que o ambiente cliente/servidor já está presente na maioria das empresas, é importante se preparar para o próximo passo. Mas afinal, o que é esta nova onda, ou esta segunda geração ? Basicamente esta segunda geração de cliente/servidor, é um ambiente onde cliente não tem mais um papel único de ser apenas cliente, e o servidor também não apresenta mais uma função única, por exemplo de apenas servir dados. O que é principalmente alertado é que hoje nós vivemos em uma época da computação muito diferente de, por exemplo, dois anos atrás. Hoje temos como computador de mesa, o nosso cliente, um computador poderoso, com um processador executando a uma velocidade média de 150 Mhz (o que está mudando rapidamente) e com um sistema operacional que não é mais tão limitado como era o MS-DOS, ou o MS-Windows 3.x (que executava em cima do DOS). Hoje temos em nosso computador cliente um sistema operacional poderoso, robusto, de 32-bits, multitarefa, *multithreading*, que suporta vários protocolos de rede nativamente e com uma interface gráfica intuitiva e de fácil uso, provendo poder ao seu usuário. Basicamente, hoje podemos ter em nosso computador cliente, um excelente servidor ! Não temos mais a famosa limitação de não podermos fazer chamadas de volta (também conhecidas como *callbacks*) para os clientes, isso já é coisa do passado. Da mesma forma, nosso servidor já é um computador ainda mais poderoso e que, para prover todos os serviços solicitados pelos clientes, se utiliza de chamadas a outros servidores, espalhados pela rede.

Portanto cliente não é mais apenas cliente e servidor não é mais apenas um servidor, todos os computadores na rede corporativa podem assumir todos os papéis. Na verdade, este tipo de ambiente não é novo, e já tem nome: "Sistemas Distribuídos".

A "segunda geração", ou "nova onda" de cliente/servidor, nada mais é do que a realização, por parte de grandes empresas, de que hoje é completamente possível trabalhar em um ambiente distribuído, transparente, executando tarefas em paralelismo real, obtendo um melhor desempenho no resultado das aplicações. Este tipo de ambiente já é uma realidade, e não faz parte mais de artigos que prevêm as tendências da computação. Isso não é mais uma tendência, é uma realidade.

3. Orientação a Objetos em Todos os Lugares

Além de ambientes distribuídos, outra realidade que está presente na computação moderna é a Orientação a Objetos (OO). Não existe um profissional que esteja envolvido ativamente com a área de desenvolvimento de software que não tenha observado a influência da orientação a objetos. Na nova era da computação em que vivemos, os métodos e processos de desenvolvimento de software, tão famosos e estáveis no passado, já não mais se aplicam para as soluções dos problemas atuais. Nós não vivemos mais em um ambiente apenas voltado para o processamento centralizado, com *mainframes*, terminais burros modo caractere e apenas acessos a dados. A computação mudou, o profissional atualizado precisa apenas olhar para os lados para observar isso. As redes corporativas, a integração entre máquinas de diferentes arquiteturas e sistemas operacionais, a Internet e outras tecnologias modernas já estão presentes em praticamente todas as empresas. Para se desenvolver software em um novo ambiente como este, são necessários métodos e processos mais adequados, um novo paradigma, que é a Orientação a Objetos.

Hoje, mesmo que um profissional da área de desenvolvimento de software queira desenvolver um novo sistema sem usar a orientação a objetos, vai encontrar muitas dificuldades, pois praticamente todas as ferramentas de ponta atualmente usadas no processo de desenvolvimento de software são otimizadas para trabalhar usando-se os conceitos de orientação a objetos. Há muito tempo, vários autores

já vêm fazendo previsões sobre a influência da orientação a objetos nas várias áreas da computação (ver figura 1) [11] e [33], se observarmos os acontecimentos destas áreas nos últimos anos, podemos verificar que todas as previsões se transformaram em realidade.

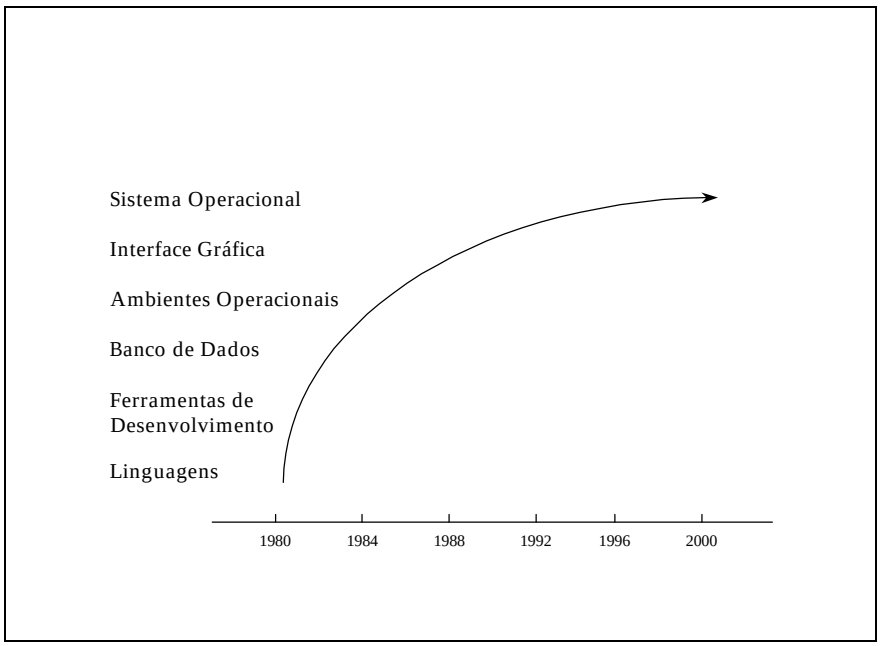

Figura 1 – Influência da Orientação a Objetos nas Áreas da Computação

Vamos verificar a seguir esta influência da orientação a objetos em algumas áreas importantes da computação.

3.1. OO e Linguagens de Programação

Sem dúvida, a primeira influência da orientação a objetos ocorreu através das linguagens de programação. Durante os anos 80, parece que houve uma febre entre criadores de linguagens de programação e praticamente todas as linguagens passaram embutir os conceitos de OO. Linguagens como BASIC, Pascal, C e até mesmo COBOL e LISP passaram a embutir os conceitos de OO (que se transformaram em linguagens como: Object Pascal, C++, Object COBOL e CLOS). Até mesmo as linguagens proprietárias, e baseadas em *scripts*, de ferramentas gráficas de prototipação hoje são consideradas orientadas a objetos (apesar de haver uma grande discussão envolvento os conceitos "baseado" em objetos e "orientado" a objetos, ver [4] e [7]).

Hoje os compiladores mais vendidos no mercado são os compiladores otimizados para linguagens OO, e mais vendidos de longe ! Isto é uma evidência de que a orientação a objetos está realmente sendo usada.

3.2. OO e Metodologias de Análise e Projeto

Os analistas de sistemas, que achavam que a orientação a objetos era coisa de desenvolvedores de software radicais, ou de *hackers*, devem abrir os olhos pois a orientação a objetos já invadiu também a sua área. Hoje, como uma evolução da metodologia de análise estruturada, nós temos a metodologia de análise orientada a objetos, e com vários novos autores descrevendo seus métodos. Autores como Grady Booch [4], James Rumbaugh [30] e até mesmo Edward Yourdon [35] (que não é um autor novo, mas que observou em tempo suficiente que deveria deixar o seu método estruturado e partir para OO [36]) reescreveram os textos de como se desenvolver software seguindo uma metodologia.

Orientação a objetos já é uma realidade em métodos de análise e projeto, e todos que estão planejando o desenvolvimento de um novo sistema devem considerar seriamente o uso destes métodos.

3.3. OO e Bancos de Dados

Outra área importante, e imensa, que vem recebendo a influência da OO há algum tempo é a área de Bancos de Dados. A tentativa de integração entre estas duas áreas identificou um problema, que é conhecido como o problema da impedância (*impedance mismatch*), ou seja, o problema de se armazenar objetos em bancos de dados organizados em tabelas relacionais. A busca da solução deste problema deu origem a uma imensa nova indústria, a dos Bancos de Dados Orientados a Objetos (ou OODBMSs, *Object Oriented Data Base Management Systems*). Hoje temos produtos que permitem se armazenar objetos como objetos, solucionando o problema da impedância.

Outra área que se demonstra em forte crescimento é a dos Bancos de Dados Objeto-Relacionais (ou ORDBMSs, *Object-Relational Data Base Management Systems*). Esta área tem potenciais de ser ainda maior que a indústria de OODBMSs, e tudo indica, que com o advento de novas necessidades de armazenamento advindas da orientação a objetos e da Internet, esta venha a ser a principal tecnologia para os bancos de dados do futuro [32]. Recentemente todos pudemos observar os últimos movimentos das grandes empresas que dominam a área de bancos de dados, como Oracle, Sybase e Informix, que pretendem transformar seus produtos principais em bancos de dados objeto-relacionais, fazendo de ferramentas adicionais que implementam uma camada de persistência praticamente inúteis.

Os agressivos avanços das grandes empresas da área de banco de dados embutindo os conceitos de OO em seus produtos principais, mostram mais uma evidência da aceitação da orientação a objetos.

3.4. OO e a Internet

E não poderíamos deixar de observar a influência da OO no furacão da Internet. Hoje, para se desenvolver software para a Internet (ou uma intranet) existem duas principais tecnologias disponíveis: ActiveX (ou OCX Internet) e Java, e as duas são tecnologias que usam os conceitos de orientação a objetos. Portanto os desenvolvedores de software que pretendem estar presentes no promissor mercado de desenvolvimento de software para a Internet, devem ter conhecimento de OO.

3.5. OO em Todas as Áreas

Além destas áreas, também há a influência da OO na área de Interfaces Gráficas com o Usuário (ou GUIs, *Graphical User Interfaces*), que deu origem às OOUIs (*Object Oriented User Interfaces*) [13], há a influência da OO na área de cliente/servidor (afinal objetos casam muito bem com o ambiente cliente/servidor), na área de ferramentas CASE (as ferramentas CASE mais vendidas no mercado são as que suportam os métodos OO), na área de aplicativos (mesmo o desenvolvedor de macros em uma planilha eletrônica hoje usa OO) e sistemas operacionais (todas as principais empresas que dominam o mercado de sistemas operacionais estão planejando, para as novas versões de seus produtos, sistemas operacionais distribuídos orientados a objetos) [29]. Podemos concluir que hoje já temos uma inteira infra-estrutura orientada a objetos que vai desde aplicativos para usuários finais, passando por ferramentas de desenvolviemnto de software e Internet, chegando ao *kernel* do nosso sistema operacional. Portanto, orientação a objetos é um fato !

4. Objetos Distribuídos

A união destas duas importantes tecnologias (Sistemas Distribuídos e Orientação a Objetos) dá origem à área dos Objetos Ditribuídos (OD).

Como descrito por Robert Orfali, Dan Harkey e Jeri Edwards em [23], [24], [25], [26] e [27], a indústria de objetos distribuídos tem o potencial de ultrapassar a indústria de bancos de dados até o final deste século (ver figura 2), se transformando no principal mercado na computação. Se nos basearmos no andamento desta indústria nos últimos meses, esta previsão tem grandes chances de se tornar realidade brevemente.

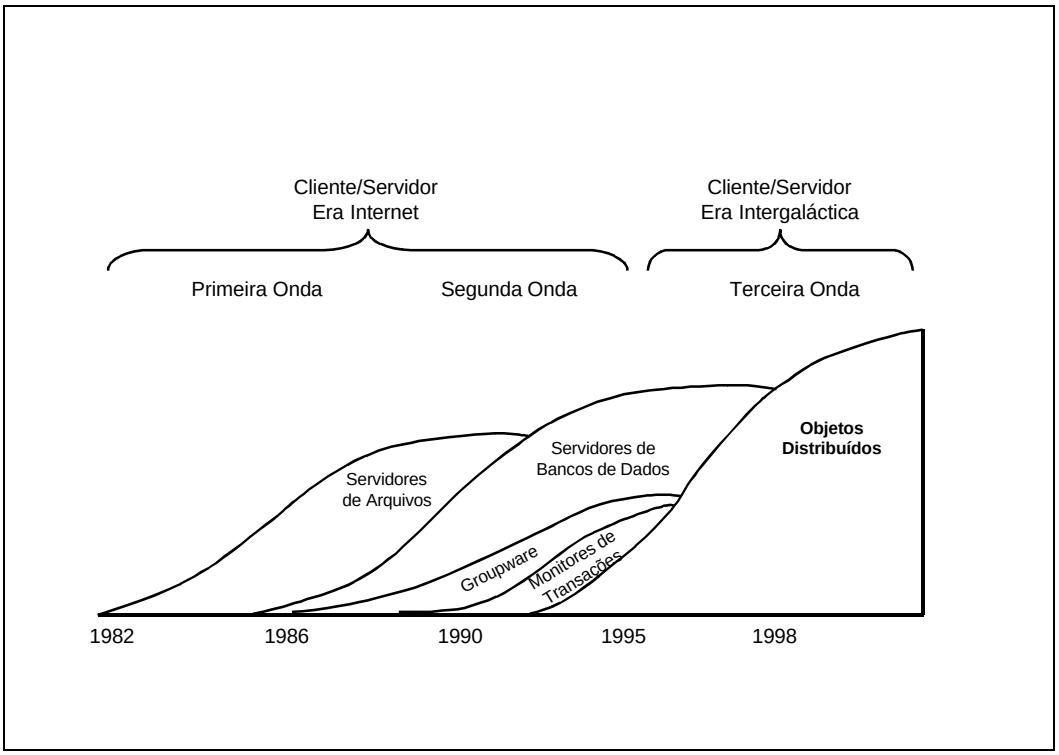

Figura 2 – As "Ondas" de Cliente/Servidor (como originalmente descrito em [22])

Objetos Distribuídos é basicamente usar a tecnologia de orientação a objetos em um ambiente distribuído. Como estas duas tecnologias já estão se transformando em realidade atual na grande maioria das empresas, será uma evolução natural. OD começou a sua trajetória através do *middleware* (a parte de software responsável pela intercomunicação entre os vários componentes distribuídos em uma rede), mas hoje está invadindo todas as áreas, inclusive a Internet.

4.1. Modelos de Objetos Distribuídos

Já existem vários modelos de objetos distribuídos disponíveis no mercado, como: DSOM (*Distributed System Object Model*) da IBM [6], PDO (*Portable Distributed Objects*) da NeXT [10], DCOM (*Distributed Component Object Model*) da Microsoft [14] e [15], CORBA (*Common Object Request Broker Architecture*) da OMG [19], [2], [3], [16], [17], [18], [22], [24], [25], [27], [28] e [31], e outros a nível de pesquisas acadêmicas (como Emerald, Arjuana, ILU, etc). Dentre estes concorrentes, dois estão se transformando nos principais padrões de mercado (seja padrão de fato ou de jure). O DCOM da Microsoft e o CORBA da OMG.

A Microsoft, já há alguns anos, vem promovendo o OLE (*Object Linking and Embedding*) como uma de suas principais tecnologias para desenvolvimento e intergração de aplicações. A camada base para o OLE é conhecida como COM (*Component Object Model*). Até recentemente a Microsoft não tinha uma solução para o uso de objetos em um ambiente distribuído. Com o recente lancamento do Windows NT 4.0 no início de 1997, a Microsoft introduziu o DCOM (*Distributed Component Object Model*) que é uma implementação do COM contemplando características distribuídas de aplicações desenvolvidas em uma rede de computadores com Windows NT. Não podemos ignorar a influência da Microsoft em praticamente todas as áreas da computação, e não podemos ignorar o fato de que a presença de um modelo de objetos distribuídos embutido no Windows NT irá promover ainda mais esta área. Mas o uso de DCOM ainda se limita ao ambiente Windows (DCOM ainda não está disponível para Unix, Macintosh, OS/2 e outros ambientes) e a instalações que disponham de apenas Windows NT versão 4.0 ou superior.

A outra força na área de objetos distribuídos é a OMG, com o seu padrão CORBA, que vem recebendo muita atenção e suporte de grandes empresas nos últimos meses.

5. OMG CORBA

A OMG (*Object Management Group*, ou Grupo de Gerenciamento de Objetos) é uma organização sem fins lucrativos que tem como objetivo difundir a área de orientação a objetos. Sua missão é descrita como: "...estabelecer guias e especificações de gerência de objetos para a indústria, visando prover uma base comum para o desenvolvimento de aplicações distribuídas" [18]. A OMG foi fundada em 1989 e conta hoje com mais de 750 empresas membro, entre elas, IBM, HP, Sun, DEC, Microsoft, Netscape, etc.

A principal contribuição da OMG até o momento é o padrão CORBA (*Common Object Request Broker Architecture*, ou Arquitetura de Corretor de Requisições de Objetos Comum) descrito em [19]. CORBA é um padrão criado pelas empresas membro e endorsado pela OMG que define como objetos devem interoperar em um ambiente distribuído. Uma das características mais importantes deste padrão é a existência de uma linguagem para a definição de interfaces (IDL, *Interface Definition Language*) que é uma linguagem, também padronizada e endorsada pela OMG, independente de arquitetura, que permite se especificar as interfaces dos objetos distribuídos de uma forma que todos possam requisitar serviços a eles.

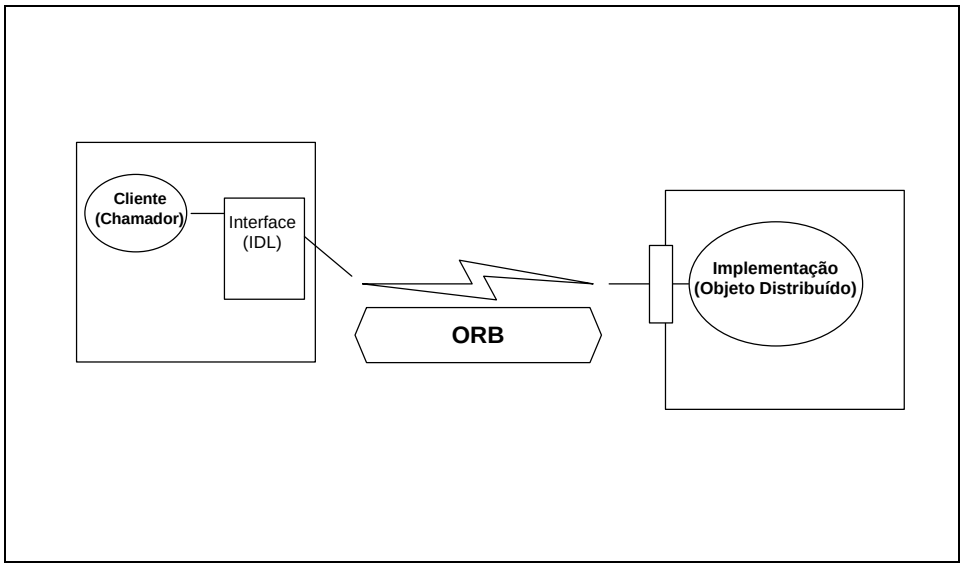

Figura 3 – Funcionamento de uma chamada CORBA

O funcionamento básico de uma chamada CORBA é demonstrado na figura 3. O solicitador do serviço enxerga apenas uma interface de um objeto que está distribuído pela rede, e após executar a chamada, apenas aguarda pela resposta. No CORBA esta interação entre chamador e objeto distribuído chamado é feita através de uma Referência para Objeto (ou *Object Reference*) que o chamador deve obter antes de executar a chamada. Toda a interação entre o chamador e o objeto distribuído chamado deve ser feita pelo *broker* (ORB), fazendo com que os principais participantes dessa interação não precisem se preocupar com detalhes de comunicação, problemas de rede, ordenação/desordenação (*marshalling/unmarshalling*), procura e ativação de objetos, etc. Para o chamador todos os serviços são atendidos de uma forma transparente, para o objeto distribuído chamado, todas as requisições se comportam da mesma forma, como se fossem requisições locais. É importante notar que um objeto distribuído não é a mesma coisa que, por exemplo, um objeto C++. Um objeto distribuído tem uma interface exportada, a sua implementação pode ser feita em qualquer linguagem de programação (inclusive linguagens não orientadas a objetos) e pode executar em qualquer processador. Os objetos distribuídos são independentes de linguagem de programação, *hardware* (máquina), sistema operacional,

rede, protocolo de comunicação, espaço de endereçamento, compiladores, etc. Podem até mesmo ser chamados de “Componentes”.

As chamadas aos objetos distribuídos são feitas após um mapeamento da IDL para a linguagem de programação sendo usada pelo chamador, este mapeamento é feito automaticamente por pré-processadores IDL, e os módulos gerados são ligados ao código do chamador. Isto permite a possibilidade de que um objeto distribuído possa ser desenvolvido em uma linguagem e o chamador possa ser desenvolvido em outra, realmente provocando a independência de chamador e objeto distribuído chamado.

5.1. Estrutura OMA

A arquitetura geral, na qual o padrão CORBA é baseado, é denominada OMA (*Object Management Architecture*, ou Arquitetura de Gerenciamento de Objetos) [18] e é mostrada na figura 4.

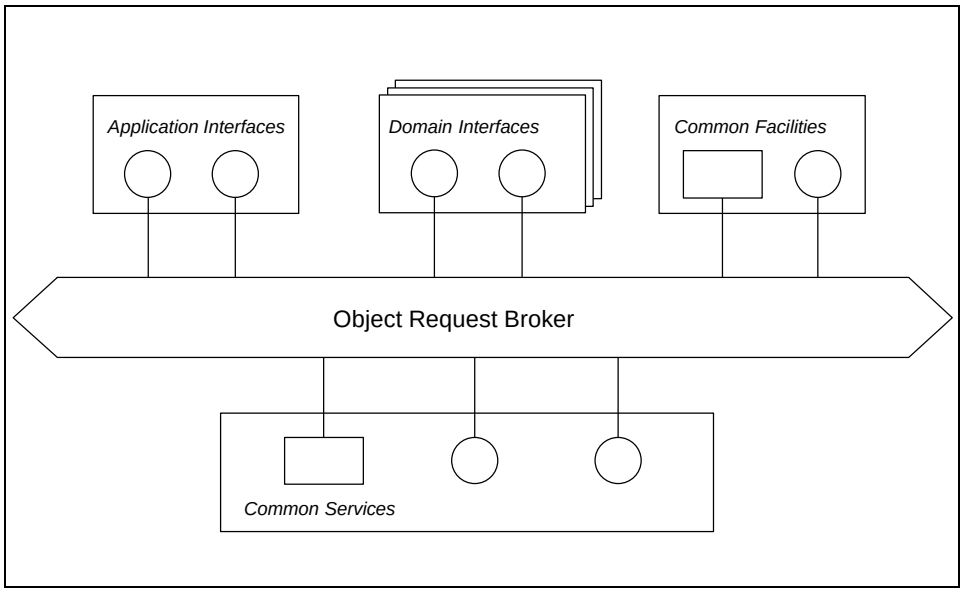

Figura 4 – Estrutura OMA (*Object Management Architecture*)

OMA é a arquitetura definida pela OMG onde todos os padrões são baseados, o componente *Object Request Broker* (ORB) da arquitetura é o coração do padrão. Além do ORB existem outros componentes que são necessários e que simplificam o trabalho dos desenvolvedores.

- ***Object Request Broker***. É o principal componente, o núcleo da arquitetura, comercialmente conhecido como CORBA. Atualmente a OMG está fazendo com que sua principal concentração de trabalho deixe de ser no componente ORB e passe para os outros componentes, pois atualmente o ORB já é considerado robusto e com disponibilidade suficiente para que não necessite tantas atualizações.
- ***Common Services***. O componente de Serviços provê um ambiente genérico no qual objetos podem executar suas tarefas. Este componente provê serviços a nível de sistema com interfaces IDL. Aumenta e complementa a funcionalidade do ORB sendo usado para criar objetos ou componentes, nomeá-los, introduzí-los ao ambiente, etc. Entre os serviços atualmente definidos pela OMG estão: Serviço de Ciclo de Vida (*Life Cycle Service*), Persistência (*Persistence Service*), Nomes (*Naming Service*), Eventos (*Event Notification Service*), Concorrência (*Concurrency Control Service*), Transação (*Transaction Service*), Relacionamento (*Relationships Service*), Externalização (*Externalization Service*), Pesquisa (*Query Service*), Licenças (*Licensing Service*), Propriedades (*Properties Service*), Segurança (*Security Service*), Tempo (*Time Service*), Gerenciamento de

Mudança (*Change Management Service*), Negociador (*Trader Service*) e Coleções (*Collections Service*).

- ***Common Facilities***. O componente de Facilidades provê um conjunto de funções genéricas que podem ser configurados para as necessidades de ambientes específicos, mais voltado para o nível da aplicação. Algumas das facilidades já definidas pela OMG são: Facilidade de Interface com o Usuário (*User Interface Facility*), Gerenciamento de Informação (*Information Management Facility*), Gerenciamento de Sistemas (*Systems Management Facility*) e Gerenciamento de Tarefas (*Task Management Facility*).
- ***Domain Interfaces***. São as interfaces específicas dos Domínios, como: Finanças, Saúde, Manufatura, Telecom, Comércio Eletrônico e Transportes. Esta divisão foi uma recente mudança da OMG, que está visualizando a área de Objetos de Negócio (*Business Objects*) como um dos mercados mais promissores para o futuro, ver [8].
- ***Application Interfaces***. A parte de Objetos da Aplicação da arquitetura representa os objetos do negócio realizando as tarefas necessárias para os usuários. São os objetos de negócio da aplicação. Esta divisão tem o potencial de cumprir todas as promessas realizadas em nome da orientação a objetos, como: compartilhamento, não reinventar a roda, reuso de objetos, etc.

5.2. Estrutura CORBA

O padrão CORBA (*Common Object Request Broker Architecture*) foi criado pela OMG em 1991. Em 1995 a OMG definiu a versão 2.0 do padrão, que fez com que a interoperabilidade entre implementações de diferentes vendedores fosse mandatória [19]. Os componentes CORBA são demonstrados na figura 5.

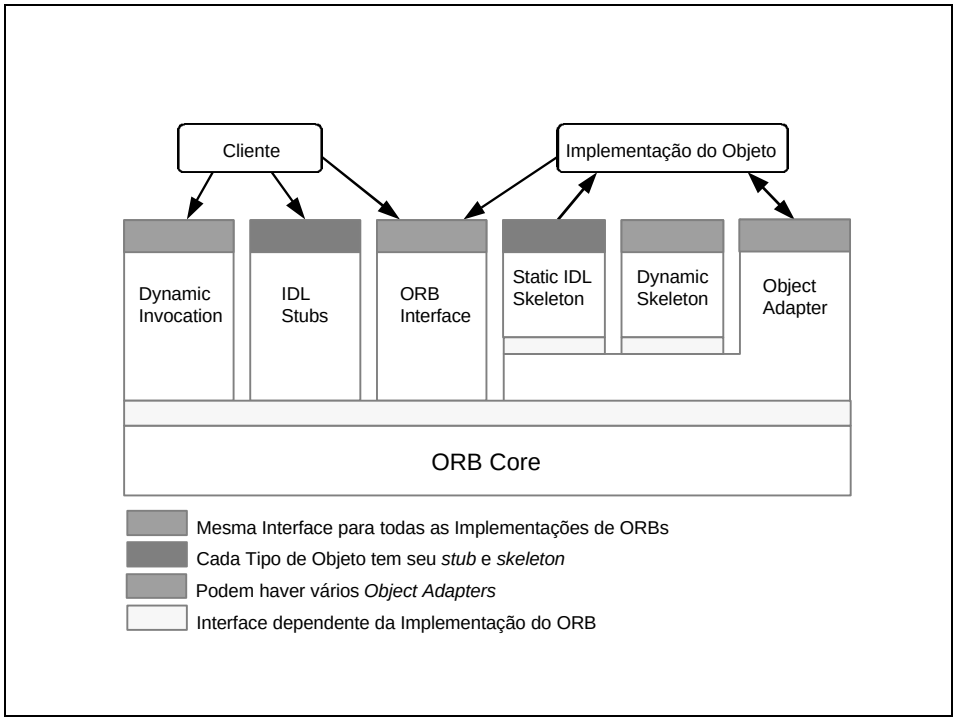

Figura 5 – Estrutura CORBA (*Common Object Request Broker Architecture*)

A especificação CORBA foi projetada para permitir integração em um grande número de sistemas diferentes. Basicamente CORBA é usado para chamadas remotas a objetos, o funcionamento dessas chamadas é semelhante a RPCs (*Remote Procedure Calls*, ou Chamadas de Procedimento

Remoto) do DCE. A camada que se encarrega da comunicação para fazer estas chamadas possíveis é o ORB. Entre as responsabilidades do ORB estão: achar a implementação do objeto para a requisição, preparar a implementação para receber a requisição e comunicar os dados que fazem parte da requisição.

O cliente executa a chamada através da visão da interface da implementação do objeto. O cliente pode usar invocação estática ou dinâmica para acessar o objeto. A invocação estática é feita através dos *stubs* IDL, estes *stubs* são gerados automaticamente para a linguagem desejada a partir da IDL, e transformados, geralmente, em bibliotecas que são ligadas ao código do cliente em tempo de compilação. Invocação estática significa que todas as chamadas do cliente para a implementação do objeto já estão disponíveis no código do cliente, para o cliente a chamada se parece como uma chamada de procedimento local. A invocação dinâmica é feita através do componente de invocação dinâmica com auxílio de um repositório de interfaces. Invocação dinâmica significa que o cliente pode determinar em tempo de execução (dinamicamente) a interface de um objeto, descobrir suas operações, número e tipo de parâmetros, preencher as informações necessárias para executar a requisição e fazer a invocação. Para a implementação do objeto, a chamada é a mesma não importa qual tipo de invocação foi usado no cliente.

O cliente executa uma requisição tendo acesso a uma referência para objeto e ao nome da operação desejada. O ORB localiza a implementação do objeto, transmite os parâmetros e transfere o controle para através dos *skeletons*. Ao executar a requisição, a implementação do objeto pode obter alguns serviços do ORB através do *Object Adapter* (Adaptador de Objeto). Quando a requisição é terminada, o controle e os valores de saída são retornados para o cliente. A seguir os componentes da arquitetura são descritos brevemente:

- **ORB *Core* (*Object Request Broker*)**. O ORB *Core* é a parte que provê a representação básica dos objetos e a comunicação das requisições. O ORB não precisa ser implementado como um componente único, mas tem que obedecer às interfaces definidas no padrão. Pode haver múltiplas implementações de ORBs que têm representações diferentes de referências para objetos e meios diferentes de executar as requisições. O Cliente deve poder acessar referências para objetos em diferentes ORBs sem problemas.
- **Clientes**. Um cliente de um objeto tem acesso a uma referência para o objeto. O cliente conhece somente a estrutura lógica do objeto através de sua interface. Uma implementação de um objeto pode ser cliente de outro objeto. Clientes geralmente vêem as interfaces dos objetos através da perspectiva de uma linguagem de programação. Clientes não têm conhecimento sobre os detalhes da implementação do objeto.
- **Implementações de Objetos**. A implementação do objeto provê a semântica do objeto definindo dados para a instância do objeto e definindo código para os métodos do objeto. Vários meios de se implementar os objetos podem ser suportados, por exemplo, servidores separados, bibliotecas, um programa por método, aplicação encapsulada, banco de dados OO, etc. Geralmente, as implementações de objetos não dependem do ORB, ou de como o cliente as invoca.
- **Referências para Objetos**. É a informação necessária para especificar um objeto em um ORB. O cliente e a implementação do objeto devem ter uma visão opaca da referência, não precisam saber nada sobre sua representação. As implementações de ORBs podem escolher suas próprias representações para as referências, não necessariamente iguais, mas devem permitir a interoperabilidade.
- ***Interface Definition Language* (IDL)**. Linguagem da OMG para especificar interfaces. A interface de um objeto é especificada em IDL, e compõe conjunto de operações e seus parâmetros. Para que um objeto seja acessado, não é necessário que o código fonte IDL esteja disponível, o objeto é acessível via rotinas *stub* ou repositório de interfaces. Para que o objeto possa ser utilizado em um ambiente de programação, sua interface IDL é mapeada para a linguagem de programação usada. Este mapeamento é o mesmo para todas as implementações de ORBs, deve seguir o padrão CORBA. O padrão CORBA 2 contém o mapeamento para C, C++ e Smalltalk. COBOL, ADA e Java estão em processo de padronização, mas implementações proprietárias já existem, como também existem implementações proprietárias para Eiffel, Pascal e linguagens de ferramentas de prototipação (como Delphi, Visual Basic, etc).

- **Interface de Invocação Dinâmica**. A DII (*Dynamic Invocation Interface*) permite que requisições sejam criadas em tempo de execução para os objetos. O cliente deve especificar qual o objeto a ser chamado, a operação desejada, o conjunto de parâmetros necessários e o modo de chamada. Estas informações estão geralmente disponíveis no repositório de interfaces. Permite grande poder para a aplicação, por exemplo, *browsers* de objetos.
- ***Skeleton* da Implementação**. Estes são os *stubs* que irão acessar os métodos que implementam as operações. Cada implementação de objeto terá uma interface estática que, através de mapeamentos para a linguagem usada e dependendo do adaptador de objetos, será usada para acionar os métodos da implementação. O ORB irá chamar os métodos da implementação através desta interface *skeleton*. É uma interface acionada via "*up-call*".
- **Interface *Skeleton* Dinâmica**. A DSI (*Dynamic Skeleton Interface*) permite manipulação dinâmica das invocações aos objetos. *Skeletons* Dinâmicos podem ser chamados via interfaces estáticas ou dinâmicas da mesma maneira e com os mesmos resultados. Este componente foi introduzido na especificação CORBA 2 com o principal objetivo de provar a interoperabilidade entre implementações ORBs de diferentes vendedores.
- **Adaptadores de Objetos**. Os Adaptadores de Objetos (*Object Adapters*) fazem a interação do ORB *Core* com a Implementação. Através do adaptador de objetos a implementação acessa os serviços do ORB. Podem haver vários adaptadores, mas pelo menos um é obrigatório, o Adaptador Básico de Objetos (BOA, *Basic Object Adapter*). Entre os serviços acessíveis através do BOA estão a geração e interpretação de referências para objetos, invocação de métodos, segurança entre as interações, ativação e desativação de implementações, etc.
- **Interface do ORB**. É a interface disponível para o cliente e a implementação acessarem o ORB *Core*. Deve ser a mesma para todas as implementações de ORBs (padrão). Como a maioria da funcionalidade do ORB é provida pelos *stubs*, *skeletons*, invocações dinâmicas ou adaptadores de objetos, apenas poucas operações comuns fazem parte da Interface do ORB.
- **Repositório de Interface**. É um repositório que armazena objetos que representam a informação das IDLs em uma forma disponível em tempo de execução. Pode ser usado pelo ORB para gerar as requisições. É usado para as requisições dinâmicas. Pode armazenar informações adicionais, por exemplo, informações de depuração, bibliotecas, rotinas para seleção de diferentes tipos de objetos, etc.
- **Repositório de Implementação**. Contém informação que permite o ORB localizar e ativar implementações dos objetos. Informações no repositório de implementação podem ser específicas de cada ORB. Entre as possíveis informações no repositório de implementação estão: informação sobre a instalação das implementações, políticas relacionadas a ativação e execução de implementações, métodos de ativação, etc. Pode armazenar informações adicionais, por exemplo, informações de depuração, controle administrativo, alocação de recursos, segurança, etc.

6. Objetos na Internet

Como visto anteriormente, a orientação a objetos já está presente na Internet, e provocando uma forte influência. As duas principais tecnologias usadas para o desenvolvimento de software na Internet atualmente são Active X e Java, as duas usam os conceitos de orientação a objetos.

A influência que Java causou em páginas HTML espalhadas pelo mundo é incontestável. Para os que estão hoje usando a linguagem Java para o desenvolvimento de *applets* que são disponilizados na Internet em páginas HTML, é importante notar que os *applets* nada mais são do que objetos, que podem navegar e executar em milhões de *browsers* espalhados pelo mundo. Este é um exemplo de objetos distribuídos, afinal, qual ambiente mais ditribuído do que a própria Internet ?

O recente lançamento do novo *browser* da Netscape (o conjunto de software chamado Communicator) com suporte embutido a objetos distribuídos é mais um gigantesco impulso para esta indústria, pois em alguns meses, mais de 20 milhões de usuários ao redor do mundo (segundo estimativas

da Netscape [1]) estarão aptos a usar objetos distribuídos. Os produtos da Netscape, inclusive seus servidores Web, passaram a incluir suporte ao protocolo IIOP (*Internet Inter-ORB Protocol*), que foi definido no CORBA 2 como o protocolo padrão para a interoperabilidade entre ORBs. Atualmente qualquer usuário que esteja usando a mais recente versão do *browser* ou dos servidores da Netscape podem fazer chamadas a objetos ditribuídos através do protocolo IIOP, sem a necessidade de comprar ou instalar software adicional. A tecnologia que a Netscape está usando em seus produtos foi licenciada da Visigenic, que produz o Visibroker for Java e Visibroker for C++.

Esta presença praticamente ubíqua é que faz com que todas as previsões com relação à área de objetos distribuídos, como a previsão mostrada na figura 2, estejam se tornando realidade.

7. Conclusão

Este artigo discutiu a evolução dos ambientes de trabalho na grande maioria das empresas com o uso de sistemas distribuídos e orientação a objetos. A união destas duas áreas, ou seja, objetos distribuídos, foi discutida com maiores detalhes e alguns modelos de objetos foram analisados. O modelo CORBA da OMG foi descrito com detalhes.

O impacto de sistemas distribuídos e orientação a objetos em conjunto com a Internet mostra que Objetos Distribuídos é uma tecnologia promissora, que tem grande potencial de entrar na corrente principal (*mainstream*) da indústria da computação nos próximos anos.

8. Bibliografia

[1] Andreessen, M.. ***IIOP and the Distributed Objects Model***. Mountain View, CA: Netscape Communications. http://www.netcape.com. 1996.

[2] Ben-Natan, R.. ***CORBA – A Guide to Common Object Request Broker Architecture***. New York, NY: McGraw-Hill. 1995.

[3] Betz, M.. ***OMG's CORBA***. In: Dr. Dobb's Special Report #225. San Mateo, CA: Miller Freeman, Inc. Winter 1994/95.

[4] Booch, G.. ***Object-Oriented Analysis and Design with Applications - Second Edition***. Redwood City: CA: Benjamin/Cummings Publishing. 1994.

[5] Booch, G., Rumbaugh, J., Jacobson, I.. ***Unified Modeling Language Documentation Set 1.0***. Santa Clara, CA: Rational Software Corporation. http://www.rational.com. 1997.

[6] Campagnoni, F.R.. ***IBM's System Object Model***. In: Dr. Dobb's Special Report #225. San Mateo, CA: Miller Freeman, Inc. Winter 1994/95.

[7] Cardelli, L., Wegner, P.. ***Understanding Types, Data Abstraction and Polimorphism***. ACM Computing Surveys, Vol. 17 No. 4. 1985.

[8] Fingar, P.. ***The Blueprint for Business Objects***. New York, NY: SIGS. 1996.

[9] Fingar, P., Read, D, Stikeleather, J.. ***The Next Generation Computing – Distributed Objects for Business***. New York, NY: SIGS. 1996

[10] Gentry, D.. ***Distributed Applications and NeXT's PDO***. In: Dr. Dobb's Special Report #225. San Mateo, CA: Miller Freeman, Inc. Winter 1994/95.

[11] Henderson-Sellers, B.., ***A Book of Object-Oriented Knowledge – Object-Oriented Analysis, Design and Implementation: A New Approach to Sofware Engineering***. Prentice Hall. 1991.

[12] Jacobson, I., Christerson, M., Jonsson, P., Övergaard, G.. ***Object-Oriented Software Engineering - A Use Case Driven Approach***. Reading, MA: Addison-Wesley. 1992.

[13] Mandel, T.. ***The GUI-OOUI War : Windows Vs. Os/2 : The Designer's Guide to Human-Computer Interfaces***. New York: Van Nostrand Reinhold. 1994.

[14] Microsoft. ***DCOM – A Business Overview***. Redmond, WA: Microsoft Corporation. http://www.microsoft.com. 1997.

[15] Microsoft. ***DCOM – A Technical Overview***. Redmond, WA: Microsoft Corporation. http://www.microsoft.com. 1997.

[16] Mowbray, T.J., Zahavi, R.. ***The Essential CORBA – System Integration Using Distributed Objects***. New York, NY: John Wiley and Sons, Inc. 1997.

[17] Mowbray, T.J., Malveau, R.C.. ***CORBA Design Patterns***. New York, NY: John Wiley and Sons, Inc. 1997.

[18] OMG. ***Object Management Architecture Guide***. Wiley-QED / Object Management Group (OMG). 1991.

[19] OMG. ***Common Object Request Broker Architecture and Specification 2.0***. Object Management Group. 1995.

[20] OMG. ***CORBAservices: Common Object Services Specification***. Object Management Group. 1995.

[21] OMG. ***CORBAfacilities: Common Facilities Architecture***. Object Management Group. 1995

[22] OMG. ***CORBA Academy – Tutorial Notes***. San Francisco, CA: OMG. 1997.

[23] Orfali, R., Harkey, D., Edwards, J.. ***Essential Client/Server Survival Guide – Second Edition***. New York, NY: John Wiley and Sons, Inc. 1996.

[24] Orfali, R., Harkey, D., Edwards, J.. ***Essential Distributed Objects Survival Guide***. New York, NY: John Wiley and Sons, Inc. 1996.

[25] Orfali, R., Harkey, D.. ***Client/Server Programming with Java and CORBA***. New York, NY: John Wiley and Sons, Inc. 1997.

[26] Orfali, R., Harkey, D., Edwards, J.. ***Intergalactic Client/Server Computing***. In: Byte Vol. 20, No. 4. New York, NY: McGraw-Hill. April 1995.

[27] Orfali, R., Harkey, D.. ***Client/Server with Distributed Objects***. In: Byte Vol. 20, No. 4. New York, NY: McGraw-Hill. April 1995.

[28] Otte, R., Patrick, P., Roy, M.. ***Understanding CORBA – The Common Object Request Broker Architecture***. Prentice-Hall. 1996.

[29] Pountain, P.. ***Special Report – Your Next OS***. In: Byte Vol. 21, No. 11. New York, NY: McGraw-Hill. November 1996.

[30] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.. ***Object-Oriented Modeling and Design***. Englewood Cliffs, NJ: Prentice-Hall. 1991.

[31] Siegel, J.. ***CORBA – Fundamentals and Programming***. New York, NY: John Wiley and Sons, Inc. 1997.

[32] Stonebraker, M., Moore, D.. ***Object-Relational DBMSs – The Next Great Wave***. San Francisco, CA: MK - Morgan Kaufmann Publishers. 1996.

[33] Winblad, A. L., Edward, S. D., King, D. R.. ***Object-Oriented Software***. Reading, MA: Addison-Wesley. 1990.

[34] Yang, Z., Duddy, K.. ***CORBA: A Platform for Distributed Object Computing***. In: ACM Operating Systems Review, Vol. 30, No. 2. April 1996.

[35] Yourdon, E.. ***Object-Oriented Systems Design - An Integrated Approach***. Englewood Cliffs, NJ: PTR Yourdon Press. 1994.

[36] Yourdon, E.. ***Auld Lang Syne – Is it Time to Ring out the Old and Ring in the New ?***. In: Byte Vol. 15, No. 10. New York, NY: McGraw-Hill. October 1990.