Um Esquema para Acessar Objetos em Ambientes Distribuídos

Luiz Otávio Botelho Lento*
Edmundo Roberto Mauro Madeira†
Departamento de Ciência da Computação
IMECC - UNICAMP
e-mail:otavio(edmundo)@dcc.unicamp.br

Abril de 1995

Sumário

Este artigo tem a finalidade de apresentar uma abordagem para acessar objetos em ambientes distribuídos. Essa abordagem é baseada na arquitetura CORBA , na qual especificamos um esquema de acesso aos objetos localizados nos repositórios de interfaces e implementações desta arquitetura. Com isso, espera-se que o acesso às informações contidas nos repositórios citados sejam obtidas de forma mais simples e eficiente.

Abstract

This paper presents one way to access objects in distributed environments. This approach is based on the ORB architecture, where we specified an access scheme to the objects of this environment. This scheme is based on implementation and interface repositories. This scheme allows that the repository information is easily and efficiently obtained.

1 Introdução

Com o desenvolvimento tecnológico, e com a necessidade de se obter uma quantidade de informações imprecindíveis à realização de diversos serviços, tornou-se necessária a criação de ambientes de processamento de informações distribuído. Esses ambientes devem oferecer e viabilizar o acesso às informações, seja ele local ou remoto, de uma forma transparente e homogênea para o usuário. Com isso, surgiram problemas de distribução e comunicação, dificultando a interoperabilidade e interconectividade na integração de sistemas distribuídos[CD88, Tan92].

A forma mais indicada para estabelecer interconectividade entre recursos é o modelo cliente/servidor, que organiza os sistemas distribuídos como um número de servidores distribuídos e oferece um conjunto de serviços para clientes através da rede.

*Mestrando do Departamento da Ciencia da Computação - UNICAMP
†Professor do Departamento da Ciencia da Computação - UNICAMP

Embora sistemas distribuídos baseados no paradigma cliente/servidor suportem a interoperabilidade entre sistemas, experiências demonstram que somente esse fator não é suficiente para ativá-la entre sistemas globais, nos quais problemas como ambientes heterogêneos, gerenciamento de configuração e monitoramento da rede são críticos. A metodologia baseada na orientação a objetos é uma solução, onde a modelagem de um sistema distribuído é feita através de uma coleção de objetos interagindo entre si, proporcionando assim a melhor forma para a integração no processamento das informações distribuídas em um sistema [TLX90, BGHS91]. A CORBA (Common Object Request Broker Architecture) é uma das soluções para os problemas acima citados, pois ela é uma plataforma para ambientes de serviços abertos que provê mecanismos, através dos quais, objetos podem comunicar-se de forma transparente por meio de solicitações e respostas.

Baseado na CORBA, esse artigo apresenta uma especificação de um esquema de acesso aos objetos armazenados em seus repositórios de interfaces e implementações, utilizando-se de recursos computacionais disponíveis, em conjunto com o paradigma cliente/servidor e a metodologia orientada a objetos. Com isso, espera-se que esse esquema dê uma maior facilidade e eficiência na obtenção das informações contidas nesses repositórios, ou seja, facilitando a oferta de suporte a processamento distribuído pela camada *Middleware* à camada *Groupware* e às aplicações na plataforma *Multiware* [LEM+94].

2 Plataforma Multiware

Para o desenvolvimento da plataforma *Multiware* escolheu-se a CORBA devido a sua simplicidade na arquitetura, por possuir funções ODP (Open Distributed Processing) e por possibilitar o incremento de serviços no sistema sem perder as suas funções.

A plataforma *Multiware* provê uma estrutura de *hardware* e *software* que possibilita o trabalho cooperativo em ambientes de serviços abertos, utilizando-se da maioria dos conceitos do modelo de referência ODP. Essa plataforma é composta de três camadas: **Software/Hardware básico**, **Middleware** e **Groupware** (vide figura 1). A camada *Software/Hardware básico* é composta por um sistema operacional e protocolos de comunicação. A camada *Middleware* é responsável por prover facilidades de processamento distribuído para a camada *Groupware* e para as aplicações. Essa camada é composta por duas subcamadas: **Processamento Multimídia** - possibilita a troca de informações multimídia em tempo real, com uma qualidade de serviço especificada; **ODP** - composta pelos sistemas comerciais ANSA, DCE (Distributed Computing Envoronment) e ORB, e nível ODP que provê funcionalidades para esses sistemas. A última camada é a *Groupware*, que provê funcionalidades através de classes diferentes de aplicações, como CSCW[1], nos quais as aplicações podem entrar ou deixar o sistema dinamicamente sem a degradação da operação, DAI[2] e outras [MM94, LEM+94].

3 RM-ODP

O esquema utiliza-se de idéias apresentadas no **Modelo de Referência ODP** (RM-ODP) [VMW+93, ISO94a, ISO94b], pois utiliza-se da metodologia orientada a objetos e provê uma estrutura para padronizar o processamento distribuído aberto. Com isso, ele

[1]Computer Support Cooperative Work
[2]Distributed Artificial Intelligence

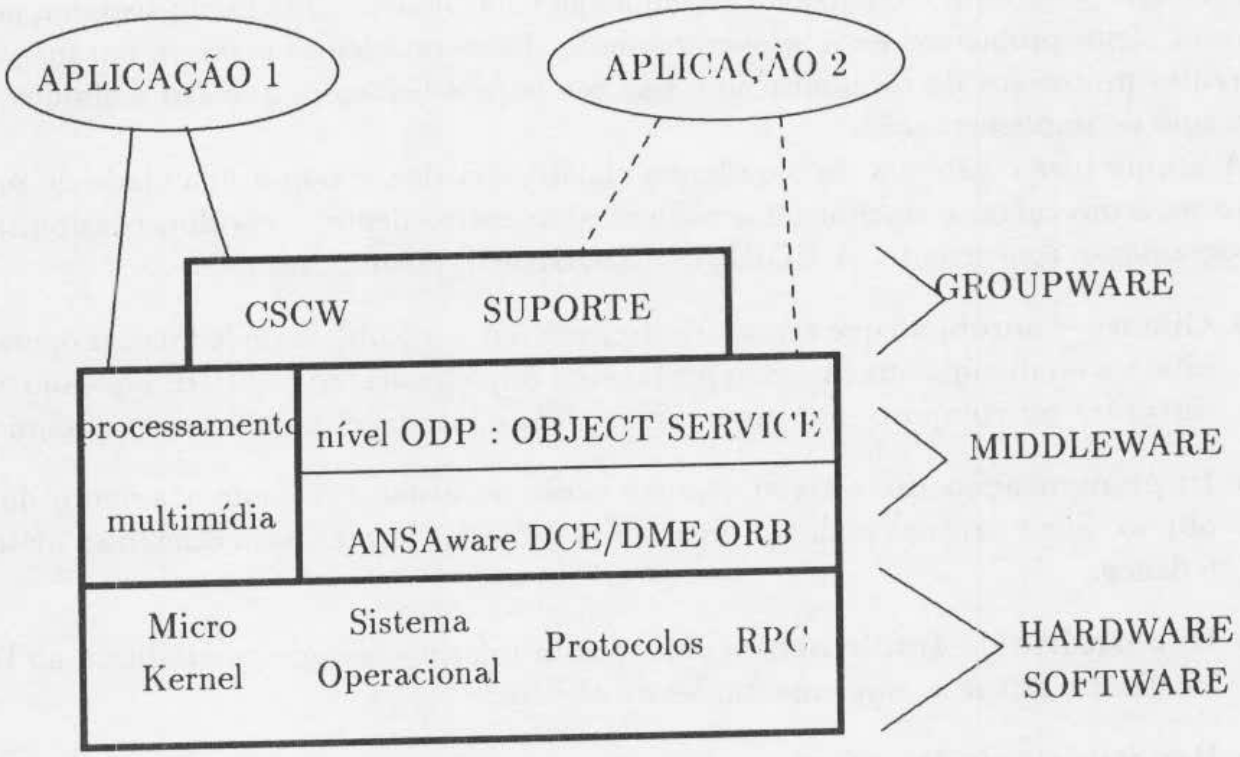

Figura 1: Plataforma Multiware

tenta englobar a heterogeneidade de ambientes, a distribuição física e lógica de equipamentos de computação e de aplicações, a interoperabilidade entre sistemas heterogêneos e a portabilidade das aplicações nos diversos ambientes.

O esquema de acesso foi desenvolvido sob os cinco pontos de vista (empresa, informação, computacional, engenharia e tecnológico) previstos no RM-ODP, bem como também nos baseamos em alguns requisitos básicos desse modelo [K.R93, ISO94a]:

- Prover um modelo consistente para o desenvolvimento em separado de diversos padrões.
- Definir áreas que necessitem de padrões ODP e o relacionamento entre os padrões.
- Descrever um conjunto básico de ferramentas a serem utilizadas na definição de padrões.
- Prover um conjunto consistente de conceitos e terminologias comuns.

4 CORBA

A CORBA, desenvolvida pela OMG (Object Management Group) no fim dos anos 80 e colocada para conhecimento público no ano de 1991, é uma plataforma para ambientes de serviços abertos que provê mecanismos, através dos quais, objetos podem comunicar-se de forma transparente por meio de solicitações e respostas. Sua arquitetura define uma estrutura de suporte à comunicação de objetos em sistemas de objetos [CCODIC93]. A especificação da CORBA é bastante flexível, pois possibilita que projetistas implementem diferentes ORBs (Object Request Brokers), bem como os objetos que os compõem. Essa

flexibilidade possibilita melhoras no desempenho e nas funcionalidades do sistema, porém acarreta sérios problemas de interoperabilidade. Esses problemas ocorrem em função de diferentes protocolos de comunicação e padrões para solicitações que são definidos para cada tipo de implementação.

A arquitetura é baseada no paradgima cliente/servidor, e tem a finalidade de prover meios para direcionar e sincronizar a comunicação entre cliente e servidor, possibilitando a programação concorrente. A CORBA ([CCODIC93]) é composta pelo:

- **Cliente**- é um objeto que através de uma referência do objeto pode invocar operações estática ou dinamicamente. São portáveis à implementação do ORB, podendo a sua estrutura ser composta por um ou mais *stubs* e interfaces de invocação dinâmica.
- **Implementação do objeto** - provê o estado atual e o comportamento de um objeto. Sua estrutura pode ser organizada de várias formas, com diferentes métodos e dados.
- **Repositório de Implementações** - possui informações que possibilitam ao ORB localizar e ativar as implementações de objetos.
- **Repositório de Interfaces** - provê o armazenamento consistente de definições de interfaces.
- **ORB** - definido através de suas várias interfaces, uma interface ORB pode conter operações que são válidas para diversas implementações ORBs, ou específicas a um tipo particular de objetos, ou ainda específicos a modos de implementação de objetos. Suas interfaces são : *stub client* (é o procedimento local correspondente a uma operação); interface de invocação dinâmica (cria e invoca as solicitações de um serviço a serem executadas por uma implementação do objeto); ORB (comum a todos os ORBs, utilizada também para conversões de referências de interfaces em *strings*); *skeleton* (utilizada na recepção das solicitações do cliente e na invocação dos métodos); adaptador de objetos (provê o acesso a um conjunto de funções necessárias à implementação do objeto). O ORB ainda possui um núcleo responsável pelo estabelecimento de comunicação entre o cliente e a implementação do objeto.
- **IDL** - usada para descrever as interfaces que estão disponíveis para os clientes e as implementações de objetos.

5 Esquema de Acesso aos Repositórios de Interfaces e Implementações

A especificação CORBA prescreve vários objetos que necessitam ser analisados e talvez reestruturados para que assim possam servir da melhor forma possível aos propósitos de sistemas distribuídos em geral. Baseado nesse princípio, criamos um esquema de acesso que tem a finalidade de tornar o acesso aos repositórios de interfaces e implementações mais simples e eficiente, bem como prover a interoperabilidade entre repositórios de interfaces num mesmo ORB ou em diferentes ORBs.

5.1 Estrutura

O esquema de acesso foi projetado para atuar em um ambiente ORB, composto pelos objetos repositórios e de acesso, pelo arquivo de acesso, por *caches* de memória (tem a finalidade de armazenar temporariamente informações pertinentes aos escopo de uma máquina) além dos repositórios de interfaces e implementações. A figura 2 apresenta a arquitetura do esquema de acesso ao repositório de interfaces.

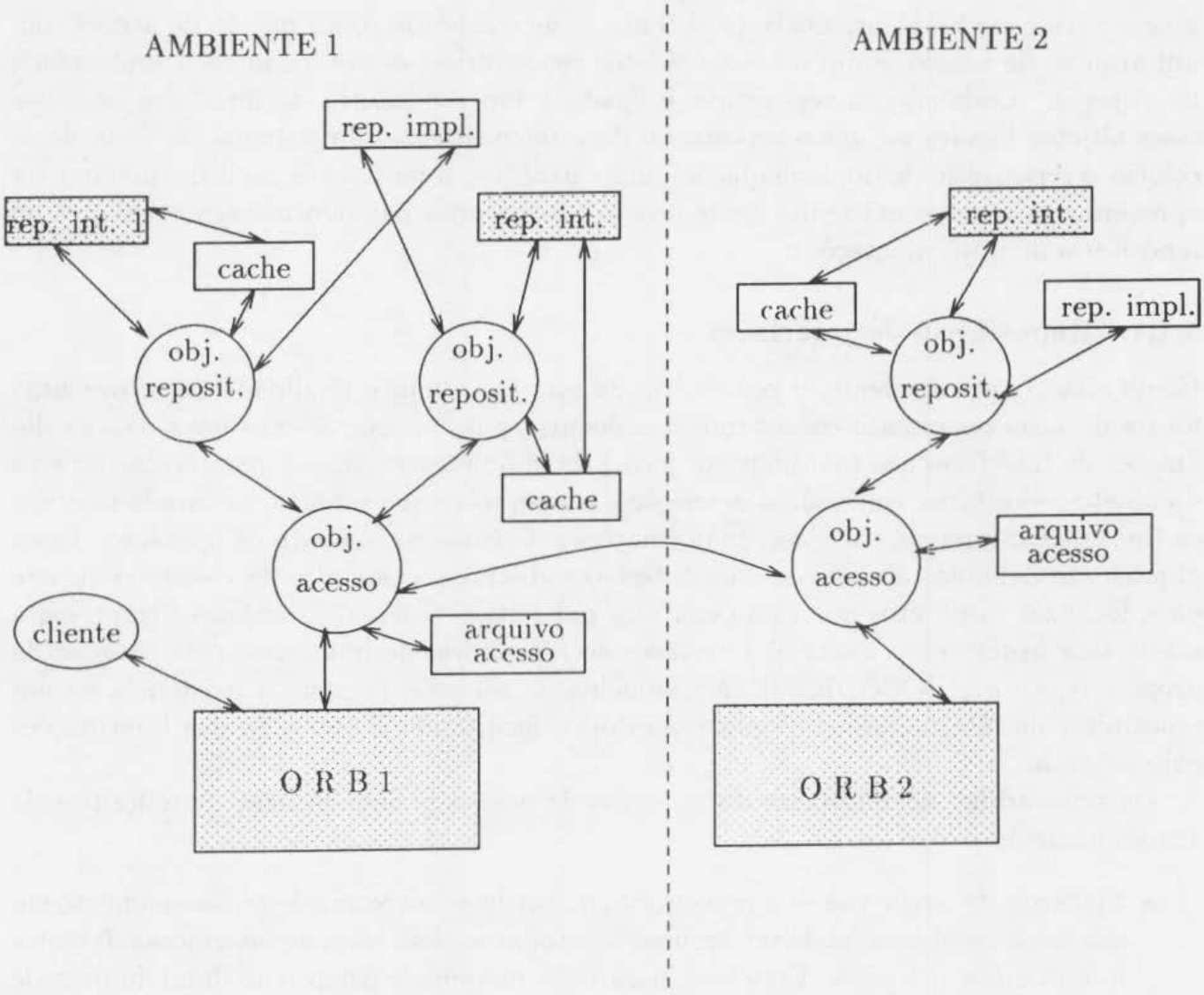

Figura 2: Arquitetura do Esquema de Acesso

Para que o esquema pudesse satisfazer a todas ou quase todas as necessidades de gerenciamento e acesso ao repositório de interfaces, bem como o acesso às informações contidas no repositório de implementações, levou-se em consideração alguns fatores citados abaixo:

- **Tipos e formas de armazenamento** → as informações podem ser armazenadas em banco de dados [ABD+89] ou em sistemas de arquivos, e são estruturadas hierarquicamente em objetos.

- **Tipos de informação** → cada repositório de interfaces possui um determinado tipo de informação armazenada. Esses variam de acordo com o escopo da máquina.
- **Funcionalidade** → a fim de prover uma melhor funcionalidade para CORBA, o esquema possibilita a interoperabilidade entre repositórios de interfaces em diferentes ambientes, ou em um mesmo ambiente ORB.

Levando-se em consideração os fatores acima citados, o esquema de acesso foi projetado para atuar em um ambiente ORB como um todo. Portanto, existe um único esquema de acesso para cada ambiente ORB do sistema, onde existe um único objeto de acesso com um arquivo de acesso, e um ou mais objetos repositórios, dependendo da complexidade do sistema. Cada objeto repositório é ligado a um repositório de interfaces, e todos esses objetos ligados ao único repositório de implementações do sistema. A idéia de se colocar o repositório de implementações junto ao objeto repositório é exclusivamente para aproveitar as funções existentes neste objeto na obtenção das informações existentes no repositório de implementações.

5.1.1 Repositório de Interfaces

Como citado anteriormente, o repositório de interfaces tem a finalidade de prover uma forma de armazenamento consistente das definições de interfaces existentes. Essas definições de interfaces são mantidas em forma de objetos, acessadas e gerenciadas através do objeto repositório, contendo a descrição das operações que suporta, incluindo também os tipos de parâmetros, exceções, e informações referentes ao contexto da operação. Esses objetos são definidos através de duas interfaces abstratas chamadas de *container* (usada para localizar os objetos que estão contidos por outros objetos) e *contained* (representa a interface *header* para todas as interfaces do repositório de interfaces, com exceção da própria *repository*) [CCODIC93]. No esquema de acesso é previsto a existência de um repositório de interfaces por objeto repositório, facilitando a obtenção das informações pelo sistema.

Os repositórios de interfaces do esquema de acesso podem utilizar duas formas de armazenamento :

- **Sistema de arquivos** → o repositório, que utiliza essa forma de armazenamento em sua implementação, pode ter armazenado todas as definições de interfaces referentes a uma única máquina. Com isso, para cada máquina existente no domínio da rede pode existir um repositório de interfaces com as definições referentes ao escopo daquela máquina. O acesso a um repositório por outra máquina torna-se viável conhecendo-se o identificador do repositório desejado. Isso é possível através do arquivo de acesso que contém esse tipo de informação.
- **BDOO**[3] → o repositório, que utiliza essa forma de armazenamento em sua implementação, pode ter múltiplas cópias das definições de interfaces distribuídas em várias máquinas através da rede. Com isso, pode-se ter um único objeto de armazenamento contendo todas as definições de interfaces, independente do escopo das máquinas . Para cada máquina pode existir um *cache* local que conterá algumas definições referentes ao escopo daquela máquina, reduzindo assim o tempo de acesso

[3]Banco de Dados Orientado a Objetos

às definições pertinentes àquele ambiente. O tempo de latência desse *cache* poderá variar de acordo com a vida útil do *request*.

A estrutura do repositório de interfaces é composta pelos objetos do tipo *repository*, que pode ser composto pelos objetos do tipo moduleDef (grupamento lógico de interfaces), interfaceDef (definição de uma interface), constantDef (definição do nome da constante), typeDef (definição do tipo, que não é uma interface) e exceptionDef (definição de uma exceção). O objeto moduleDef pode ser composto pelos objetos do tipo moduleDef, exceptionDef, typeDef, interfaceDef e constantDef. O objeto interfaceDef pode ser composto pelos objetos constantDef, operationDef (definição de uma operação na interface), exceptionDef, attributeDef e typeDef. Com exceção dos objetos repository, interfaceDef e moduleDef, já citados anteriormente, todos os outros são objetos simples. A figura 3 apresenta a estrutura do repositório de interfaces [CCODIC93].

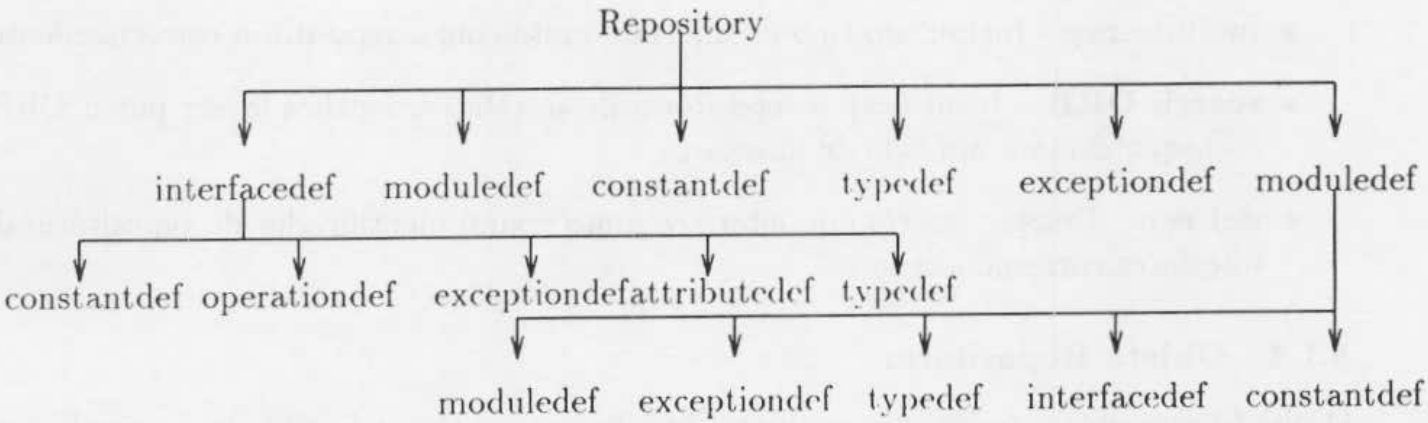

Figura 3: Estrutura Hierárquica do Repositório de Interfaces

Além dessas interfaces, são definidos os *TypeCodes*, que são valores que representam tipos de argumentos e atributos. Eles têm uma variedade de utilizações. São utilizados pelas interfaces de invocação dinâmica para indicar os tipos de argumentos, e pelo repositório de interfaces na especificação de algumas interfaces.

5.1.2 Repositório de Implementações

O repositório de implementações possui as informações referentes à localização e ativação de objetos (endereço da implementação do objeto e sua forma de armazenamento). Essas informações são utilizadas pelo ORB, assim, o objeto adaptador junto com o *skeleton* pode ativar e invocar a correta implementação, referente à solicitação feita pelo cliente, tanto estática quanto dinâmica. No esquema de acesso utiliza-se um repositório de implementações em todo o sistema, onde são armazenadas as informações referentes às suas operações, para possibilitar a utlização das funções existentes no objeto repositório.

5.1.3 Objeto de Acesso

O objeto de acesso possui um conjunto de funções que tem a finalidade de encaminhar as solicitações provenientes do ORB a um objeto repositório específico. Utiliza-se um objeto

de acesso por esquema, em função de um melhor encaminhamento de todas as solicitações feitas ao repositório de interfaces que chegam pela ORB, facilitando assim a aquisição das informações sobre os repositórios em um ambiente. Isso é feito devido à utilização do arquivo de acesso, pois através deste são obtidas informações referentes à localização do tipo da interface. Com isso, otimizamos e reduzimos o custo operacional do sistema (implementação, gerenciamento de solicitações de serviços), porém, existe o problema da centralização da passagem de todos essas solicitações por um único objeto em uma única máquina, reduzindo assim a confiabilidade e disponibilidade do sistema. Esse problema pode ser resolvido através da existência de cópias desse objeto em outras máquinas, bem como a utilização de um *software* de gerenciamento de forma que detecte qualquer falha no sistema. Definimos para o objeto de acesso as seguintes funções:

- **search_rep** - Identificar o repositório de Interfaces correspondente a um tipo de interface.
- **include_rep** - Incluir um tipo de interface junto com o repositório correspondente.
- **search_ORB** - Identificar o repositório de interfaces, localizado em outro ORB, correspondente a um tipo de interface.
- **del_rep** - Deletar um tipo de interface junto com o identificador do repositório de interfaces correspondente.

5.1.4 Objeto Repositório

O objeto repositório possui um conjunto de funções que tem a finalidade de auxiliar no acesso e no gerenciamento dos objetos armazenados nos repositórios de interfaces e implementações. Este objeto (com todas as suas funções) possibilita uma maior disponibilidade na obtenção das informações contidas nesses repositórios. Sendo um mediador entre o objeto de acesso e os repositórios, as solicitações de informações destes são obtidas de forma mais eficiente. Isso é verificado em função do objeto de acesso direcionar a solicitação de um serviço a um objeto repositório específico onde esse poderá dispor de todas as informações contidas no repositório de interfaces a que estiver ligado, bem como no repositório de implementações. Este fato pode ser observado tanto na utilização de um sistema de arquivos como na utilização de um banco de dados distribuído (pode ser utilizado o *cache* de memória), onde em ambos os casos essas informações são obtidas de forma direta através do objeto repositório. Definimos para o objeto repositório as seguintes funções:

- **include** - Incluir uma definição de interface no repositório de interfaces.
- **dynamic_invoc** - Buscar uma definição de interface no repositório de interfaces, com a finalidade de compor a referência do objeto.
- **lookup** - Realizar uma verificação de tipos no repositório de interfaces.
- **del_int** - Retirar definições de interfaces contidas no repositório de interfaces.
- **search_address** - Buscar informações sobre implementações de objetos (forma de armazenamento e localização) no repositório de implementações.

5.2 Interoperabilidade

A questão da interoperabilidade entre repositórios de interfaces foi tratada de duas formas. A primeira entre repositórios num mesmo ambiente ORB, no qual é relativamente simples. Essa simplicidade deve-se ao arquivo de acesso, pois através dele pode-se obter informações de tipos de interfaces pertencentes a outro repositório em um mesmo ambiente ORB. A segunda forma de interoperabilidade é entre repositórios localizados em diferentes ambientes ORBs. Essa forma é mais complexa, pois vários fatores devem ser considerados, tais como a consistência de informações e os conflitos entre identificadores de objetos e conteúdos de informações. Tudo deve ser levado em consideração quando se deseja obter determinadas informações nos repositórios de interface localizados em diferentes ambientes [Zuq95].

Uma forma de solucionar o problema citado no parágrafo acima é a utilização de identificadores padrões de tipos de interface e identificadores padrões de repositórios de interface (localizados no arquivo de acesso), de forma que as informações possam manter-se consistentes, independente do ORB a que estão ligadas. Desta maneira os identificadores de interface seriam valores que representariam os tipos de informação armazenados. Os identificadores de repositórios dariam nome aos repositórios que identificariam exclusivamente módulos, interfaces, constantes, operações, etc.

5.3 Seqüência de Chamada

Para um melhor entendimento do funcionamento geral do Esquema de Acesso, explicaremos como uma solicitação, feita por um cliente, ao chegar no esquema de acesso irá se comportar. A solicitação feita pelo cliente estaticamente, através de stubs (ex: realização de uma verificação de tipos), ou dinamicamente (ex: busca de uma interface), através da DII (interface de invocação dinâmica), chega ao objeto de acesso pelo ORB. O objeto acesso verifica em que repositório estará contida a informação desejada pelo cliente utilizando o arquivo de acesso. No caso da informação desejada não estar contida no ambiente no qual realiza-se a solicitação, o objeto de acesso pode buscá-la em outros ambientes através da interoperabilidade entre esquemas. Já o objeto repositório pode buscar a informação no repositório, desde que o conheça e, obtida a informação solicitada pelo cliente, pode enviá-la a este através do ORB. A solicitação feita pela implementação do objeto chega ao esquema de acesso através do ORB. Com o tipo da interface e o identificador da operação a ser invocada, o objeto de acesso interage com o objeto repositório, e esse com o repositório de implementações obtendo o endereço correspondente à operação a ser invocada pelo *skeleton*.

6 Esquema de Acesso Segundo os Conceitos ODP

Nesta seção realizamos uma análise do desenvolvimento do esquema de acesso sob os conceitos existentes no modelo de referência ODP.

O modelo de referência ODP oferece subsídios para a estruturação dos ambientes de processamento distribuído aberto. Devido às suas características, projetistas de sistemas utilizam-no como um guia na construção de seus sistemas abertos. A CORBA se insere nesse contexto, sendo uma plataforma para ambientes de serviços abertos, e tem também algumas das funcionalidades definidas no RM-ODP [CCODIC93, K.R93, ISO94a].

Como no modelo de referência ODP, a CORBA também fornece uma estrutura consistente e bem definida para o processamento distribuído aberto. Essa estrutura engloba a heterogeneidade dos ambientes, a distribuição física e lógica dos equipamentos e também a portabilidade das aplicações entre os diversos ambientes. Podemos considerar a CORBA como um padrão específico para o ODP, cobrindo determinados conceitos e funções comuns no que tange a especificação da comunicação e da coordenação das informações distribuídas. Baseado na concordância de conceitos entre as duas especificações, estruturamos o esquema de acesso utilizando as idéias existentes no RM-ODP.

6.1 Modelagem de Objetos

A metodologia orientada a objetos utilizada pelo modelo de referência ODP também é utilizada na CORBA. Os objetos do esquema de acesso, de uma forma geral, podem ser definidos para diversas configurações e implementados em várias linguagens de programação. Seus objetos obedecem a algumas propriedades:

- **Abstração** - ocorre quando a descrição de seu serviço abstrai-se da forma como é implementado. Todos os objetos do esquema oferecem serviços que indempendem da forma como são implementados.
- **Interação** - os serviços são oferecidos através de interfaces. Os serviços dos objetos do esquema, bem como os objetos dos repositórios, estão a disposição através de interfaces.
- **Encapsulamento** - todas as informações dos objetos do esquema de acesso são obtidas através de invocações.
- **Hierarquia** - os objetos do repositório de interfaces são classificadas de acordo com o seu serviço, especificados na forma de classes e subclasses.
- **Herança** - essa propriedade é provida pela IDL (Interface Definition Language), durante a definição das interfaces dos objetos contidos nos repositórios.
- **Distribuição** - as implementações dos objetos podem ser relocadas, sem afetar os clientes. Essas mudanças de posicionamento são atualizadas no repositório de implementações.

6.2 Pontos de Vista ODP

Durante o desenvolvimento do esquema de acesso, procuramos visualizar o sistema sob os cinco pontos de vista estabelecidos pelo ODP.

- **Empresa** → Nesse ponto de vista são estabelecidos os objetivos que devem ser alcançados pelo esquema. Foram definidas as políticas de armazenamento das informações nos repositórios de interfaces e implementações. São definidas as políticas de gerenciamento das informações dos repositórios (a forma como serão acessados, atualizados e utilizados), como também o escopo dessas informações. Estabeleceu-se que cada objeto do esquema seria um agente (objeto que desempenha uma ação), e os repositórios junto com o arquivo de acesso seriam artefatos (objetos que não iniciam uma ação).

- **Informação** → Neste ponto de vista, foi definida a semântica das requisições. Definiram-se as linguagens a serem utilizadas na implementação do esquema, bem como as estruturas e quais as funções que seriam desempenhadas por cada um dos objetos do esquema.

- **Computacional** → Sobre esse ponto de vista, visualizou-se o esquema como um objeto computacional que iria interagir com os outros objetos da CORBA.

  A interação entre os objetos do esquema é feita de forma operacional, no qual as interfaces dos objetos são operacionais, provendo um modelo cliente/servidor. Essas interfaces possuem operações com parâmetros, terminações, requisições e resultados. Foram especificados os algorítimos que corresponderiam às funções dos objetos.

  A invocação ao esquema segue o princípio do modelo computacional, que utiliza-se de duas ações atômicas para a sua conclusão. A primeira é a *invocation submit*, quando o cliente submete a invocação ao ORB, e a segunda *invocation deliver*, quando o adaptador de objetos chama a implementação para receber a invocação. Para a conclusão do serviço, a CORBA também segue o conceito do RM-ODP, que utiliza-se de duas ações atômicas para a sua realização. Inicialmente, a *termination submit*, quando o *skeleton* invoca o método correto da implementação, e a *termination deliver*, quando o cliente recebe seus resultados e encerra o serviço. Todas as operações realizadas na CORBA são **interrogações**, onde qualquer invocação é seguida por uma terminação de um serviço.

- **Engenharia** → dentro do ponto de vista de engenharia foi estabelecida a estrutura distribuída do sistema, isto é, foram descritos os aspectos de distribuição do esquema, definindo-se a infraestrutura do modelo. O esquema como um todo é um objeto da infraestrutura de distribuição do sistema. Poderíamos visualizá-lo como uma cápsula no sistema, ocupando um espaço de memória provendo recursos para o sistema concluir um determinado serviço.

  O esquema de acesso provê algumas transparências:

  - **Falha** - A transparência de falha pode ser observada parcialmente, pois caso haja queda no sistema em função da queda da máquina em que se encontra o objeto de acesso, outra máquina pode ser acionada de forma que mascare os erros de comunicação do esquema, proporcionando uma maior confiabilidade e disponibilidade do sistema.
  - **Federação** - Esta transparência pode ser verificada parcialmente, através do compartilhamento de recursos providos pela integração dos repositórios de interfaces.
  - **Localização** - Podemos citar também a transparência de localização, no qual o esquema proporciona para o ORB um acesso às informações nos repositórios de uma forma uniforme e consistente, independente da localização das mesmas.

6.3 Funções

A arquitetura CORBA identifica algumas funções que asseguram a execução dos serviços solicitados pelos clientes. Estamos descrevendo algumas destas funções citadas no RM-ODP como funções de repositório:

- **Armazenamento** - esta função pode ser verificada tanto no repositório de interfaces quanto no repositório de implementações. O gerenciamento destes é feito através de interfaces que possibilitam a realização de modificações, deleções e obtenções de informações

- **Repositório de Tipos** - esta função é verificada através do repositório de interfaces, onde são armazenados os tipos de interface, mantendo-se a hierarquia de tipos e as relações. Esta função possibilita a verificação de tipos durante a invocação dinâmica.

- **Relocador** - a função relocador pode ser observada através do arquivo de acesso que mantém o posicionamento dos tipos de interfaces, mesmo quando existe qualquer alteração desse posicionamento.

7 Implementação

Com a finalidade de validarmos os principais conceitos apresentados pelo modelo conceitual do esquema de acesso aos repositórios de interfaces e implementações, implementamos um protótipo desenvolvido sobre plataformas SUN, no qual foi utilizado o RPC[4] da SUN junto com o NFS[5] da SUN.

A estrutura do protótipo é composta por três *stubs clients* (programas que são responsáveis pelo envio das solicitações de serviços ao esquema de acesso), um objeto de acesso (representado por um conjunto de programas carregados em uma máquina), por dois objetos repositórios (representados por um conjunto de programas e carregados em máquinas diferentes), quatro bases de dados que representam os repositórios de interfaces e implementações, e o arquivo de acesso. A figura 4 demonstra a arquitetura do protótipo criado.

7.1 Stub Cliente

Como já citado, o sistema implementado possui três *stubs client*, sendo que o primeiro tem a finalidade de solicitar uma inclusão de um novo tipo de interface (caso ela não exista) junto com o tipo da operação que ela executa. São passados como parâmetros o tipo da interface a ser incluída, a operação que ela executa e o repositório onde será armazenado.

O segundo *stub client* tem a finalidade de solicitar uma verificação de tipo de interface junto ao objeto de acesso e verificar o tipo de operação a ser executada pela interface junto ao repositório. São passados como parâmetro o tipo da interface e o tipo de operação executada.

O terceiro e último *stub client* tem a finalidade de buscar dinamicamente o endereço referente a uma operação pertencente a uma interface específica. São passados o tipo da operação e da interface como parâmetros.

7.2 Objeto de Acesso

O **objeto de acesso**, dentro do nosso sistema, funciona tanto como um cliente quanto um servidor do RPC da SUN. Como servidor, o objeto de acesso recebe um tipo de dados

[4]Remote Procedure Call
[5]Network File System

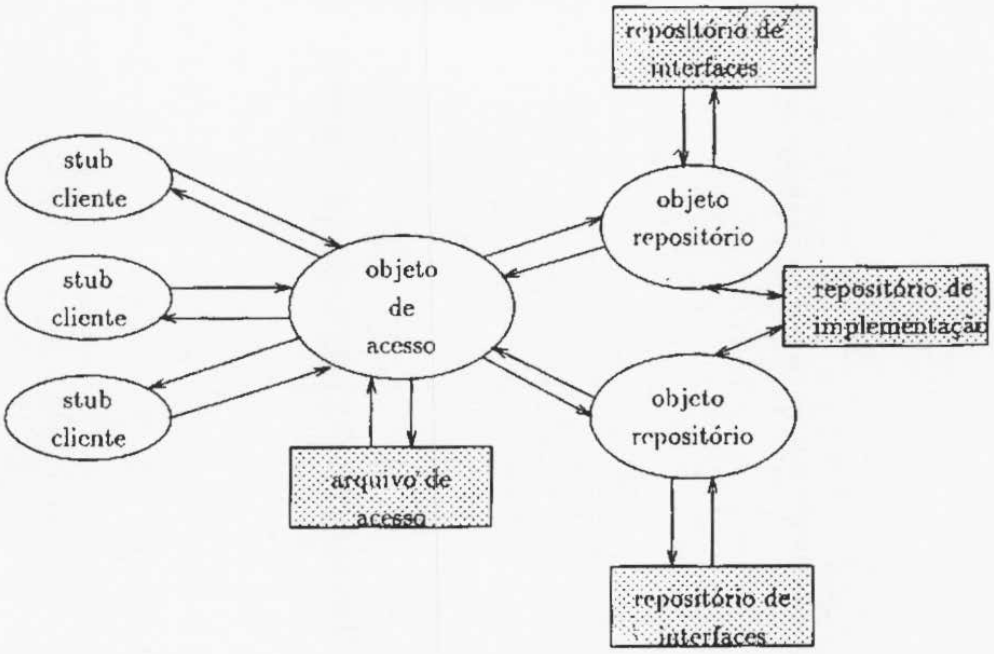

Figura 4: Arquitetura do Protótipo

através da rede e busca alguma informação no arquivo de acesso necessária à invocação de um serviço no objeto repositório. Como cliente, basicamente realiza uma chamada RPC do objeto repositório, passando as informações necessárias à realização do seviço desejado.

Foram implementadas, parcialmente, as funções *search_rep* e *include_rep*, especificadas no modelo conceitual, de forma compatível com o RPC da SUN. A função *search_rep*, no protótipo, tem a finalidade de interagir com o arquivo de acesso, e através do identificador do tipo de interface buscar qual o repositório em que se encontra a interface, bem como a máquina correspondente ao repositório. A função *include_rep* tem a finalidade de incluir um tipo de interface junto com uma operação num repositório de interfaces. Após a realização de sua funcionalidade inicial, cada uma das duas funções realiza uma invocação remota de uma função no objeto repositório correspondente ao repositório de interfaces em que está localizado o tipo da interface que foi passada como parâmetro inicial. A identificação das máquinas no arquivo de acesso é importante devido à necessidade do RPC ter conhecimento destas para realizar uma chamada remota.

7.3 Objeto Repositório

O objeto repositório no nosso protótipo é um servidor do RPC da SUN. O esquema possui dois objetos repositórios representados por diferentes máquinas sendo que cada um desses objetos interage somente com o seu respectivo repositório de interfaces, implementados no nosso protótipo como arquivos. Foram implementadas, parcialmente, as funções *include*, *lookup* e *search_address*, especificadas no modelo conceitual, e compatíveis com o RPC da

SUN. A função *include* tem a finalidade de incluir uma interface junto com a sua operação no repositório de interfaces. Esta função também atualiza o repositório de implementações caso seja necessário a inclusão de uma nova operação. A função *search_address* tem a finalidade de interagir com o repositório de implementações para buscar o endereço e a forma pela qual a operação está armazenada. A função *lookup* procura no repositório de interfaces um determinado tipo de interface, retornando um valor booleano referente ao sucesso ou não da busca. A verificação de tipos realizada por essa função é feita tanto na interface quanto na operação.

7.4 Armazenamento de Informações

Essa seção aborda as quatro bases de dados do esquema. O **arquivo de acesso**, ligado ao objeto de acesso, é um arquivo implementado sob a forma de *strings*. Cada um desses *strings* contém o tipo da interface, o identificador da máquina e o identificador do repositório. Com isso, esse arquivo possui todas as informações referentes às interfaces contidas nos dois repositórios de interface.

O **repositório de implementações**, também ligado ao objeto repositório, é um arquivo implementado com *strings*. Cada uma dessas *strings* contém o endereço (localização), o identificador e a forma como está armazenada uma operação. O identificador da operação é usado como uma entrada na obtenção de outros dados fornecidos por esse arquivo.

Os dois **repositórios de interfaces** estão estruturados em árvores do tipo B, que são de busca balanceada apropriadas para esquemas de armazenamento secundário com acesso direto. Cada nó dessa árvore possui quatro chaves, isto é, pode conter até quatro tipos de interface. Cada chave possui um apontador para um arquivo com o nome da respectiva interface. Cada arquivo possui todas as operações correspondentes à interface, possibilitando que cada tipo de interface possa ter mais de uma operação. Para esse ambiente ORB nenhum repositório aceitará um tipo de interface que já tenha sido especificado em outro repositório. Todas as operações definidas nos arquivos das interfaces são também especificadas no repositório de implementações [FZ92, CLR90].

7.5 Análise do Protótipo do Esquema de Acesso

Nesta seção são abordados alguns fatores suportados pelo protótipo, tentando demonstrar a vantagem da utilização desse esquema de acesso dentro do contexto da CORBA:

- A implementação não faz restrições quanto ao aumento da quantidade de informações a disposição do cliente em tempo de execução. Esse aumento pode ser proveniente da inclusão de um novo tipo de interface com suas operações no repositório de interfaces, bem como a inclusão de uma nova operação no repositório de implementações. Em toda e qualquer inserção ou deleção de tipos de interfaces no repositório de interfaces, o arquivo de acesso é atualizado.

- O acréscimo e a diversificação das informações através do aumento do número de repositórios em tempo de compilação é possível, uma vez que a implementação do protótipo suporta quantos repositórios forem desejados, de forma que cada um esteja agregado a uma máquina. O repositório de interfaces não aceita a inserção de tipos de interfaces já existentes e nem a inserção de operações já existentes em uma determinada interface.

- Outro parâmetro fundamental que valida o nosso modelo é a funcionalidade apresentada pelo objeto de acesso dentro da implementação.
  - **Direcionamento**→ pode-se observar que o objeto de acesso na implementação recebe uma solicitação e envia esta ao objeto repositório, pois tem a sua disposição o arquivo de acesso que possui as informações necessárias para a realização dessa funcionalidade.
  - **Suporte a Falhas**→ refere-se ao problema da queda de um nó ou da máquina onde está localizado o objeto de acesso. Isso pode ser sanado através da manutenção de cópias desse objeto em outro nó ou máquina no sistema resolvendo, assim, esse problema.

- A obtenção das informações pelo objeto repositório, através de suas operações a serem realizadas no repositório de interfaces, passa a ser mais simples. O objeto repositório já recebe do objeto de acesso a operação a ser realizada junto com todas as informações necessárias à execução da mesma.

8 Conclusão

Este artigo apresentou uma forma de acessar objetos em ambientes distribuídos, baseada na CORBA, na qual especificou-se um esquema de acesso aos objetos contidos nos repositórios de interfaces e implementações dessa arquitetura. Esse esquema utilizou-se do conceitos previstos no RM-ODP para a sua especificação.

O esquema foi estruturado para operar em um ambiente ORB, composto pelos objeto de acesso (gerencia as solicitações provenientes do ORB para o objeto repositório), objeto repositório (facilita a obtenção das informações nos repositórios de interfaces e implementações), o arquivo de acesso, repositório de interfaces, repositório de implementações e pelo *cache* de memória.

As vantagens da utilização desse esquema são:

- Gerenciar todos as solicitações provenientes do ORB através de um único objeto de acesso.
- Maior disponibilidade das informações contidas nos repositórios através do objeto repositório, permitindo a existência de vários repositórios no mesmo ORB.
- A utilização do *cache* de memória para otimizar o tempo de busca da informação.

Com isso, pode-se dizer que o esquema de acesso aos repositórios de interfaces e implementações torna mais simples e eficiente o acesso aos objetos contidos nesses repositórios, dando à camada *Middleware* maiores condições de oferecer facilidades de processamento distribuído à camada *Groupware* e às aplicações na plataforma *Multiware*.

Agradecimentos:

Agradecemos à FAPESP pelo apoio financeiro e ao Prof. Manuel de Jesus Mendes pelas discussões e sugestões a este trabalho.

Referências

[ABD+89] Malcon Atkinson, François Bancilkon, David Diltrich, David Maier, and Stanley Zdonik. The object-oriented database system - manifesto. *Report-Technical*, (30), Outubro 1989.

[BGHS91] Gordon Blair, John Gallagher, David Hutchison, and Dong Skepherd. *Object-Oriented Languages, Systems and Applications.* Pitman Publishing, 1991.

[CCODIC93] Digital Equipment Corporation, NCR Corporation, Hyper Desk Corporation Object Design Inc, SunSoft Inc, and Hewlett-Packard Company. The common object request broker: Architecture and specification. *OMG*, 1.2(93.xx.yy), Dezembro 1993.

[CD88] George F. Coulouris and Jean Dollimore. *Distributed Systems Concepts and Desing.* Addison-Wesley, 1988.

[CLR90] Thomas H. Cormem, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* The Mit Press and Mc Graw-Hell Book Company, 1990.

[FZ92] Michael J. Folk and Bill Zoellick. *File Structures.* Addison-Wesley Publishing Company Inc., 1992.

[ISO94a] ISO/IEC. *Basic Model of ODP-Part 1 : Overview and Guide to use.* ISO, 1994.

[ISO94b] ISO/IEC. *Basic Model of ODP-Part 3 : Prescritive Model.* ISO, 1994.

[K.R93] K.Raymond. Reference model of open distributed processing: a tutorial. *International Conference on Open Distributed Processing*, 1:3–14, Setembro 1993.

[LEM+94] W.P.D.C. Loyolla, E.R.M.Madeira, M.J.Mendes, E.Cardozo, and M.F.Magalhães. Multiware platform: an open distributed environment for multimedia cooperative applications. *COMPSAC*, 1994.

[MM94] M.J.Mendes and E.R.M. Madeira. Plataforma multiware: Projeto e desenvolvimento da camada middleware. *SBRC*, (12), Fevereiro 1994.

[Tan92] Andrew S. Tanenbaum. *Modern Operating Systems.* Prentice-Hall, 1992.

[TLX90] Tadao Takakashi, Hans K.E. Liesenberg, and Daniel Tavares Xavier. *Programação Orientada a Objetos - Uma Visão Integrada do Paradigma de Objetos.* USP, 1990.

[VMW+93] V.Tcshammer, M.J.Mendes, W.L.Souza, E.R.M.Madeira, and W.P.Loyolla. Processamento distribuído aberto e o modelo rm-odp/iso. *Anais do Simpósio Brasileiro de Redes de Computadores*, 1(11), Maio 1993.

[Zuq95] Nuccio Marcel Scott Zuquello. *Um Esquema para Possibiltar a Interoperabilidade entre Plataformas ORBs baseado em Proxies.* Tese Mestrado em andamento do Departamento da Ciência da Computação da UNICAMP, 1995.