Um Modelo Para a Implementação de Federação de Traders

Luiz Augusto de Paula Lima Júnior

Um Modelo Para a Implementação de Federação de Traders

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por Luiz Augusto de Paula Lima Júnior e aprovada pela Comissão Julgadora.

Campinas, 06 de outubro de 1994.

Edmundo R. M. Madeira

Prof. Dr. Edmundo Roberto Mauro Madeira
Orientador

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Um Modelo Para a Implementação de Federação de Traders[1]

Luiz Augusto de Paula Lima Júnior[2]

Departamento de Ciência da Computação
IMECC – UNICAMP

Edmundo Roberto Mauro Madeira[3]
(Orientador)

[1]Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação da UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

[2]O autor é Bacharel em Ciência da Computação pela Universidade de São Paulo - Instituto de Ciências Matemáticas de São Carlos.

[3]Professor do Departamento de Ciência da Computação - IMECC - UNICAMP.

Agradecimentos

À Jesus a quem devo a inteligência, a disposição e a capacidade para realizar este trabalho. A Ele seja a glória.

Aos meus pais que me amam, os quais amo também.

Aos meus amigos – amigos mais que irmãos – que sempre estiveram ao meu lado compartilhando das minhas dores e alegrias nestes anos.

Ao Prof. Edmundo cuja orientação foi fundamental na composição deste trabalho.

A todos os colegas e professores do DCC.

À CAPES e à FAPESP pelo apoio financeiro recebido durante o desenvolvimento do projeto.

"Se o Senhor não edificar a casa, em vão trabalham os que a edificam; se o Senhor não guardar a cidade, em vão vigia a sentinela.

Inútil vos será levantar de madrugada, repousar tarde e comer o pão que penosamente granjeastes – aos seus amados Ele o dá enquanto dormem."

Salmo 127:1-2

Resumo

Este trabalho descreve uma proposta de TRADER que é o objeto computacional que recebe ofertas de serviços de outros objetos (chamados "servidores") colocando-as numa base de dados. Assim que um outro objeto qualquer (um "cliente") necessitar de um serviço computacional ele pode perguntar ao trader que então o informa a respeito da "localização" de algum servidor que oferece este serviço. Através de *binding dinâmico*, o cliente pode acessar o serviço na interface do servidor. Estes conceitos são introduzidos no Capítulo 1, juntamente com o modelo de referência para Processamento Distribuído Aberto da ISO (RM-ODP) que fornece a estrutura básica para a formulação dos modelos que serão propostos.

No Capítulo 2 apresenta-se o trader em detalhes a partir dos cinco pontos de vista definidos pelo RM-ODP.

A união de traders estabelecendo *Federações* é discutida no Capítulo 3 apresentando-se vantagens e os detalhes e mecanismos para tal organização, novamente a partir dos cinco pontos de vista do RM-ODP.

No Capítulo 4 são apresentados modelos e mecanismos para a construção de federação de traders. Neste capítulo é proposto um modelo teórico para a implementação de um trader incluindo toda a funcionalidade necessária para a criação de federações e utilização das operações federadas.

Ainda no capítulo 4 são discutidos os detalhes da implementação do protótipo de trader construído no trabalho como parte da plataforma MULTIWARE em desenvolvimento na UNICAMP. Também os principais algoritmos e estruturas de dados são comentados e é feita uma análise comparativa com trabalhos relacionados.

Por fim, no Capítulo 5, são levantadas as principais dificuldades encontradas durante o desenvolvimento do projeto, as principais contribuições do presente trabalho e possíveis futuras extensões.

Abstract

This work describes a proposal of a **TRADER** which is the computational object that receives service offers from other objects (called "servers") storing them in a data base. When another object (a "client") needs a computational service, it can ask the trader about the service, and then the trader may return the "address" of a server which offers that service. Through dynamic binding the client can access the service at the server's interface. All of these concepts are introduced in the Chapter 1 besides the ISO Reference Model for Open Distributed Processing (RM-ODP).

In Chapter 2 the trader is described in detail using the five view-points defined in the RM-ODP.

The grouping of traders creating *Federations* is discussed in Chapter 3, presenting the advantages and the details of this organization, again using the five view-points of the RM-ODP.

In Chapter 4 mechanisms and models for the construction of federated traders are presented. A theoretic model for the implementation of a trader including all the functionality neccessary for federation and federated operations is also proposed in this chapter.

Furthermore, the details of the implementation of the prototype of a trader are discussed. The prototype was built as a part of the MULTIWARE platform which is in development at UNICAMP. The main algorithms and data structures are also commented in Chapter 4. At the end of the chapter, a comparative analysis of related works is made.

At last, in Chapter 5, the main difficulties found during the development of the project are raised and the main contributions of this current work and possible future extensions are commented.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução e Visão Geral

1.1 Sistemas Homogêneos e Heterogêneos

O futuro da computação inclui sistemas distribuídos e abertos. Sistemas distribuídos *homogêneos* já tem sido desenvolvidos e estudados extensivamente porque eles permitem que usuários compartilhem objetos que podem ser : recursos periféricos e computacionais, serviços e informações. Estes sistemas consistem de várias estações de trabalho individuais e recursos periféricos (por exemplo, discos e impressoras) conectados por uma rede local (LAN). Estes computadores têm hardware e software similares e estão sob o controle de um sistema operacional distribuído. Através da interconexão de vários sistemas distribuídos locais para formar um grande sistema distribuído homogêneo, os usuários de cada sistema local se tornam capazes de compartilhar recursos que não estão disponíveis nos seus sistemas locais.

A necessidade de utilizar uma variedade de objetos e o crescimento da diversidade de hardware e software fizeram com que a *heterogeneidade* se tornasse algo corriqueiro. Se usuários individuais estão conectados a um ambiente heterogêneo, o número de objetos que eles podem acessar é significativamente maior do que aqueles disponíveis em suas redes locais (por exemplo, banco de dados, sistemas de arquivos, serviços de diretório, supercomputadores e outros hardwares especializados). Assim, através da interconexão de sistemas distribuídos locais e de diferentes computadores, um grande sistema heterogêneo e distribuído de computadores (sistema aberto) pode ser construído. Este sistema aberto pode ser gerenciado por um sistema operacional aberto.

Apesar das vantagens e do seu potencial, sistemas distribuídos abertos trazem consigo várias dificuldades:

1. Um usuário de um sistema distribuído perde a autonomia que possuía em ambientes "stand-alone", e a necessidade de mecanismos de segurança se torna um verdadeiro desafio;

2. A heterogeneidade acaba com o acesso direto a um objeto, e se usuários querem permitir com que outros usuários tenham acesso ao seus objetos, estes acessos devem acontecer em condições determinadas em comum acordo.

3. Como o número de objetos que oferecem serviços computacionais é normalmente muito grande, deveria existir um meio de localizar determinados servidores de serviços que satisfaçam uma lista de propriedades desejadas.

Para tentar solucionar estes problemas, a ISO[1] está propondo a introdução de uma nova classe de servidores de nomes e gerenciadores de objetos conhecidos como **TRADERS** [ISO7047, ISOTRD] dentro de uma estrutura chamada *Modelo de Referência para Processamento Distribuído Aberto* (RM-ODP[2]) [ISO107462, ISO107463, ISO7056, TSCH92, TSCH93]. É importante ressaltar que ainda não existe a especificação definitiva do trader nos documentos publicados pela ISO. A funcionalidade e os mecanismos para o trader são, até o presente momento, assunto em discussão, fato que abre um espaço para novas propostas e sugestões, o que está sendo feito no presente trabalho.

Poderíamos comparar o trader a um serviço de "classificados" por telefone onde pessoas que estão oferecendo produtos (ou serviços) informam a uma central (o trader) sobre os dados pertinentes àquele tipo de serviço. Assim que a central receber um telefonema de alguém procurando por aquele produto (ou serviço) ela se encarrega de achar a melhor oferta (se é que há alguma) que mais se adequa aos requisitos do cliente que ligou. A central passa então as informações (inclusive o número de telefone) do(s) fornecedor(es) do produto (ou serviço).

Num exemplo mais realista, consideremos a rede heterogênea de computadores da figura 1.1.

Se o usuário (objeto computacional ou ser humano) em A deseja utilizar uma impressora laser que esteja no "Centro de Computação" da empresa em que ele trabalha, cujo preço por página impressa seja mínimo, como descobrir o "endereço" de alguma servidora que ofereça este serviço (se é que existe alguma e se é que A tem autorização para isso) ? E ainda mais : como fazer isto de forma transparente, ou seja, de modo a esconder os detalhes da topologia da rede ? No modelo ODP, é somente através da utilização do trader que objetos podem

[1]ISO : International Organization for Standardization

[2]RM-ODP : Reference Model for Open Distributed Processing.

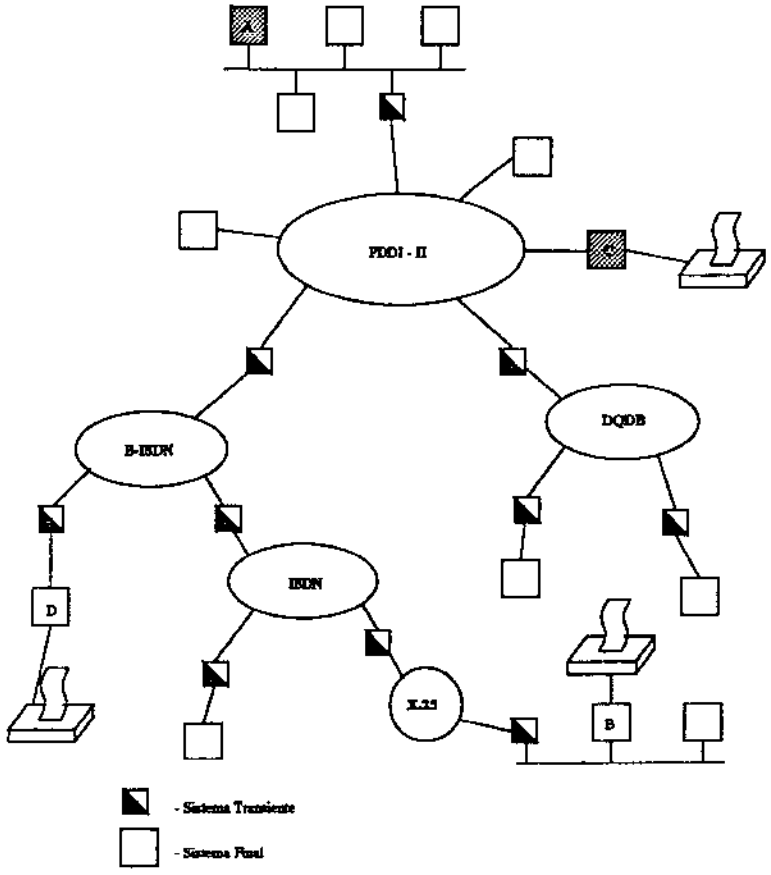


Figura 1.1: Rede heterogênea.

Tipo de Serviço (PRINTER)	Serviço Exportado por C
Printer type : DotMatrix, Laser	*Printer type* = Laser
Identifier : String	*Identifier* = "HP"
Localization : String	*Localization* = "Centro Comp. - Empresa de A"
Paper size : A4, A3 (default A4)	*Paper size* = A4
Cost per page : Float	*Cost per page* = 0.5

Figura 1.2: Tipo de serviço e serviço exportado por C.

saber a respeito da existência de serviços os quais ele deseja requisitar. No exemplo, os servidores de impressão B, C e D podem ter informado previamente ao trader a respeito das características dos serviços de impressão que eles oferecem ("PRINTER" na figura 1.2). Assim, quando A faz o pedido de serviço ao trader, ele se encarrega de achar o servidor (digamos, C) que melhor se adequa às exigências feitas por A, acoplando A a C. O serviço exportado por C também está apresentado na figura 1.2.

1.2 O Modelo de Referência Para Processamento Distribuído Aberto - RM-ODP

1.2.1 Introdução

A Necessidade de Distribuição e Abertura

A informação é a base para a operação bem-sucedida de qualquer empresa porque captura, na sua forma mais abstrata, os objetivos, recursos e atividades da empresa.

Uma parte de informação disponível no local errado não tem nenhum valor para a organização. Acima de tudo, inconsistência descontrolada representa um grande problema para a empresa. Por isso, a informação deve ser transportada do lugar onde ela está disponível, para o lugar onde ela é necessária. A distribuição de informação é, portanto, uma conseqüência necessária das restrições do mundo real.

O objetivo principal da padronização do processamento distribuído aberto é suportar a distribuição de objetivos, recursos e atividades de uma maneira consistente e confiável à medida que usa uma coleção distribuída ou heterogênea de equipamentos.

"Background" Histórico do ODP

O Modelo de Referência OSI[3] forneceu uma estrutura para a padronização de interconexão, mas o seu escopo é limitado pela interconexão de pares de sistemas. Ele tem sido uma ferramenta essencial na criação de uma família de padrões básicos de comunicação, mas o modelo não resolve o problema da distribuição, principalmente por causa da sua concentração na comunicação par-a-par ao invés de uma descrição de configurações arbitrárias de objetos.

[3]OSI : Open Systems Interconection

A ISO tem reconhecido que a padronização de sistemas distribuídos tem crescente importância, e em 1987 ela aprovou um Item de Novo Trabalho em Processamento Distribuído Aberto (ODP), tendo como objetivo a criação de um Modelo de Referência que pudesse integrar uma grande gama de padrões para sistemas distribuídos e resolver o problema de consistência entre tais sistemas. O Modelo de Referência é necessário para dar suporte à especificação de propriedades globais ao invés de somente o comportamento em interfaces específicas.

O Modelo de Referência de Processamento Distribuído Aberto é baseado em conceitos precisos e, na medida do possível, no uso de técnicas de descrição formal para a especificação da arquitetura. Ele contém definições precisas e material de suporte de uma natureza mais tutorial [ISO107462, ISO107463].

Conceitos Básicos de Sistemas Distribuídos Abertos

Sistemas abertos são sistemas capazes de interagir com outros sistemas no ambiente.

Da mesma forma que em grandes sistemas distribuídos, as principais características de um ambiente aberto são :

- **Ilimitado** : público, livre de restrições geográficas e/ou organizacionais;
- **De Livre Acesso** : pronto para admitir qualquer tipo de usuário, componente ou aplicação;
- **Heterogêneo** : caracterizado pela diversidade onde cada usuário decide sobre as próprias características de existência, projeto, implementação, uso e gerenciamento dos seus componentes;
- **Autônomo** : cada usuário ou servidor de aplicações é livre de regulamentação para determinar as propriedades, comportamento e evolução de seus elementos;
- **Descentralizado** : com funcionalidade física e lógica de cada elemento gerenciada a nível local, sem qualquer influência ou controle externo.

Como conseqüência disso, no ambiente aberto nenhum administrador ou usuário tem uma visão completa do sistema, nem pode influenciá-lo como um todo. Neste ambiente, cada observador só pode olhar para uma certa parte limitada do ambiente ilimitado. Portanto, cada observação e cada afirmação nunca são absolutas

e válidas para o ambiente inteiro, mas são somente válidas para um certo ponto e área de observação.

Além disso, deve-se considerar a adoção de um conjunto de regulamentações de acesso e medidas de segurança para garantir a integridade e privacidade dos recursos e para excluir observação externa e controle externo.

Em suma, o modelo de referência para Processamento Distribuído Aberto da ISO tem por objetivo padronizar o compartilhamento transparente de recursos e serviços sobre :

- arquiteturas diferentes,
- redes diferentes,
- sistemas operacionais diferentes.

No entanto, os usuários de serviços devem estar cientes de potenciais fornecedores destes serviços para que haja o desejado compartilhamento. Além do mais, em um grande sistema distribuído, os nós ("sites") e aplicações estão freqüentemente em mudança. Ou seja, em um sistema distribuído aberto, adição de novos serviços e atualização/deleção de serviços antigos são fatos comuns e esperados. Por isso, é vantagem permitir-se acoplamento dinâmico ("late binding") entre clientes e fornecedores de serviços. Uma aplicação deve ser capaz de selecionar servidores adequados que satisfaçam dinamicamente às suas requisições, se é que o acoplamento dinâmico será admitido.

A seleção dinâmica de um serviço apropriado em tempo de execução é fornecida através da **Função de Trading** em ODP. O **Trader** é o agente responsável por prover esta função. Ou seja, "*trading*" pode ser considerado como sendo uma forma melhorada de nomeação e binding (acoplamento) [KUTV93]. No caso de trading, o nome de um serviço é substituído por uma descrição das propriedades desejadas de um fornecedor do serviço.

1.2.2 Pontos de Vista

Ao invés de tentar lidar com toda a complexidade de um sistema distribuído, a especificação ODP considera o sistema a partir de diferentes pontos de vista, onde cada um deles é escolhido para refletir um conjunto de considerações de projeto. Cada ponto de vista representa uma abstração diferente do sistema distribuído original, sem a necessidade de criar um modelo enorme que o descreva.

Informalmente, um ponto de vista é uma representação do sistema com ênfase num aspecto específico. Mais formalmente, a representação resultante é uma abstração do sistema, ou seja, uma descrição que reconhece algumas distinções (aquelas relevantes à consideração) e ignora outras (aquelas não relevantes à consideração).

Atualmente, o trabalho em ODP reconhece cinco pontos de vista (figura 1.3):

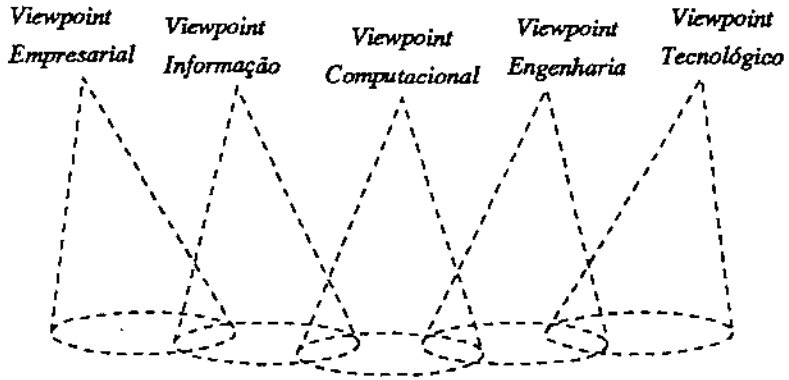


Figura 1.3: Pontos de Vista.

Ponto de Vista Empresarial : está preocupado com os requisitos empresariais para o sistema. Estes requisitos são expressos em termos de políticas de negociação, políticas de gerenciamento e funções de usuários humanos com respeito aos sistemas e ao ambiente com o qual eles interagem. Além disso, fornece uma descrição de onde e como o sistema de informação está colocado dentro da empresa (ou parte dela). O uso da palavra "empresa" aqui não implica numa limitação a uma única organização, mas o modelo construído pode muito bem descrever as restrições colocadas à interação entre várias organizações distintas.

Ponto de Vista de Informação : lida com o modelagem da informação, fornecendo uma visão comum consistente que cobre fontes e "sumidouros" de informação e o fluxo da informação entre eles.

Ponto de Vista Computacional : lida com algoritmos e estruturas de dados que fazem com que o sistema tenha função de sistema distribuído.

Ponto de Vista de Engenharia : lida com os mecanismos de distribuição e com a provisão de várias transparências necessárias ao suporte de distribuição.

Ponto de Vista Tecnológico : lida com detalhes de componentes e links a partir dos quais o sistema distribuído é construído.

1.2.3 Conceitos Básicos Para ODP

Sistemas distribuídos são complexos e mais evoluem do que são repostos, e freqüentemente são usados no processo de seu próprio reprojeto e desenvolvimento. Por isso, é importante que a abordagem básica para o modelagem seja suficientemente poderosa e não-ambígua.

A abordagem geral utilizada é a *orientação a objetos*. Sistemas são modelados em termos de interações de um conjunto de objetos em *interfaces* bem definidas. Uma interface um subconjunto das interações de um objeto [ISO107462]. Uma interface só está bem definida quando estão definidos:

- o conjunto de assinaturas de operações na interface (por exemplo, *int export(Client_Id cliente, Name_Type nome_contexto, int ..., ...)*);
- o comportamento que pode ocorrer na interface;
- restrições do ambiente (como restrições de tempo (deadlines), restrições de falha, disponibilidade, tolerância a falhas, segurança, restrições de localização, etc.);
- papel (cliente ou servidor – nunca ambos).

Para ajudar a extração de funções comuns, um sistema de tipos de objetos e de interfaces é definido. Os "templates" de objetos descrevem as características comuns desses objetos. Ou seja, objetos que fornecem o mesmo serviço podem ser descritos pelo mesmo template. Através da *instanciação*, um novo objeto é produzido a partir de um template[4].

Esta abordagem geral é usada repetidamente nos vários pontos de vista descritos na seção 1.2.2, mas os tipos de objetos identificados variam de componentes concretos (*hardware*) de um subsistema de comunicação até entidades abstratas num modelo empresarial ou de informação.

[4]Tipicamente a instanciação significa atribuir valores iniciais para parâmetros de estado do objeto.

1.2.4 Conceitos de Arquitetura

Os conceitos de arquitetura incluem tipos específicos de estruturação e são apresentados para caracterizar o comportamento de componentes do sistema, adicionando idéias como "*threads*" (ou "processos leves") e atividades para suportar a descrição de concorrência e transação. Isto então é estendido para expressar causalidade e para descrever estruturas computacionais comuns tais como relacionamentos cliente/servidor.

Conceitos específicos são também desenvolvidos para lidar com aspectos particulares de problema de distribuição (como segurança e gerenciamento) que aparecem nos diferentes pontos de vista.

O documento da ISO que apresenta o Modelo de Referência para Processamento Distribuído Aberto define também um conjunto de características que um sistema deve ter para ser caracterizado como um sistema ODP [ISO107463]. Esta definição é feita através de linguagens envolvendo os cinco pontos de vista e através da identificação de funções-chave genéricas que estão relacionadas ao modelo que é construído a partir de cada linguagem.

As definições são construídas em termos dos conceitos básicos e de arquitetura e expressam os requisitos necessários para tornar o processamento distribuído aberto possível.

O modelo criado a partir destas definições está organizado usando os cinco pontos de vista. Em alguns deles (computacional e de engenharia) há muitos requisitos detalhados de projeto. Eles têm o objetivo de assegurar que componentes produzidos separadamente possam ser usados juntos. Já em alguns outros pontos de vista (empresarial, por exemplo) há menos restrições exatamente para permitir políticas e estratégias particulares.

A seguir veremos com mais detalhes a linguagem de engenharia por ser ela que define uma estrutura para a organização de aplicações genéricas no ambiente distribuído e aberto.

1.2.5 Linguagem de Engenharia

Esta linguagem descreve a organização de uma infraestrutura abstrata que permite a execução de um sistema ODP. Também identifica :

- as abstrações necessárias para gerenciar a distribuição física e os recursos do sistema local;
- e define a função dos diferentes objetos que suportam as funções ODP (por exemplo, funções de transparência);

- os pontos de referência dentre os diferentes objetos.

Portanto, o modelo de engenharia – definido a partir da linguagem – revela os mecanismos que regulam e permitem a distribuição.

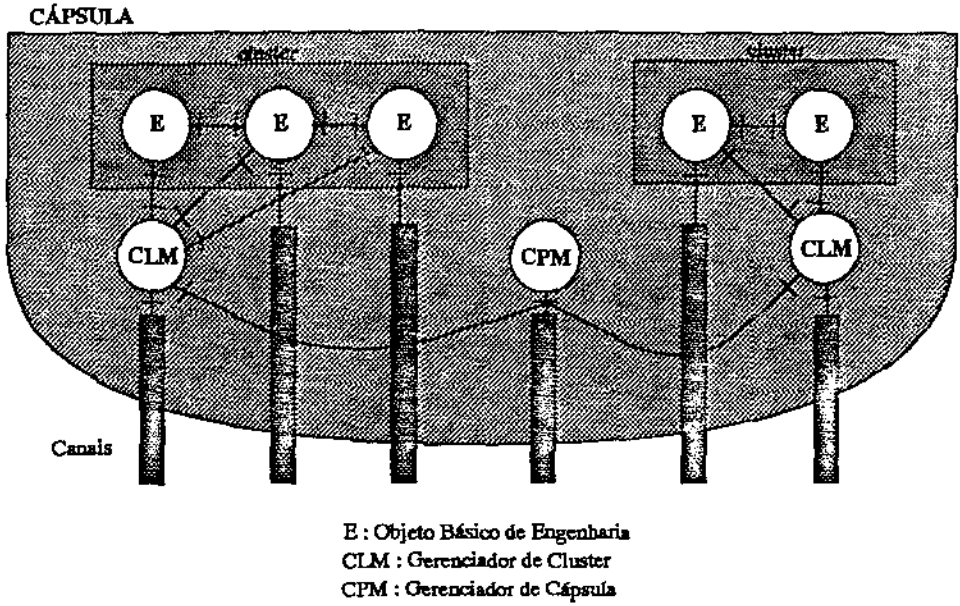


Figura 1.4: Estrutura de uma cápsula.

Conceitos Básicos

Para a apresentação do modelo de engenharia, é necessário o conhecimento de alguns conceitos preliminares :

Objetos Básicos de Engenharia (E) : fornecem a funcionalidade propriamente dita de aplicações ODP. São objetos cujas interfaces estão ligadas ou a outros objetos básicos de engenharia ou a canais.

Canal : uma configuração de objetos (stubs, binders, objetos protocolo e interceptadores) que estabelecem o acoplamento entre um conjunto de interfaces a objetos básicos de engenharia. É através de canais que a interação entre objetos pode ocorrer.

Stub : objeto que modifica a informação trocada numa interação dentro do canal, contribuindo para transparência de distribuição.

Binder : objeto que controla o *binding*[5] entre objetos básicos de engenharia cliente e servidor dentro de um canal, contribuindo para transparência de distribuição.

Interceptador : objeto que permite interações que atravessam diferentes domínios administrativos e de comunicação.

Uma *Cápsula* é a unidade de engenharia de alocação de recursos e encapsulamento. A estrutura das cápsulas (figura 1.4) define, em ODP, a estrutura que as aplicações devem ter para fazerem parte do ambiente distribuído aberto sendo devidamente gerenciadas.

Uma cápsula consiste de :

- um conjunto de objetos básicos de engenharia (E) particionados em clusters ativos;
- um conjunto de objetos gerenciadores de cluster (CLM), um para cada cluster na cápsula;
- um conjunto de objetos stub e binder para cada canal suportando interação em uma interface de um objeto básico de engenharia dentro de um cluster;
- um gerenciador de cápsula (CPM).

Os objetos básicos agrupam-se em *clusters* que representam as unidades de engenharia que podem ser ativadas e que são passíveis de migração. A interação entre objetos de engenharia em clusters diferentes de outra forma que não através de canais é proibida.

Uma *cápsula* é, então, um conjunto de clusters e de objetos de transparência (stubs e binders) residentes num certo nó e sujeitos a um núcleo comum.

Um *Nó* é a unidade de engenharia de independência de recursos. Ou seja, todos os objetos em um nó compartilham recursos de processamento, armazenamento e comunicação.

[5] Um *Processo de Binding* constitui-se de um comportamento através do qual um dado contrato é estabelecido entre duas ou mais interfaces (e, portanto, entre os objetos que as suportam). Quando existe um *binding*, existe a aparência de um caminho fim-a-fim entre os objetos ligados. O *binding* é definido no ponto de vista computacional e corresponde ao canal do ponto de vista da engenharia, ou seja, é constituído de stubs, binders, objetos protocolo e interceptadores que são gerados como resultado da compilação da definição de uma interface [ISO107462, ISO107463].

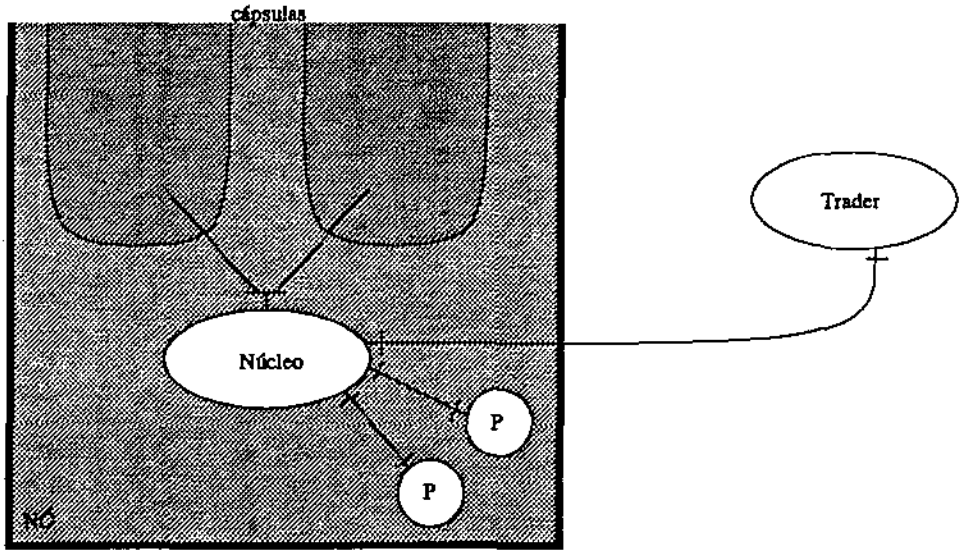


Figura 1.5: Estrutura de um nó.

Na estrutura de um nó (figura 1.5), estão definidos, além das cápsulas, os seguintes objetos:

Núcleo : objeto que coordena as funções de processamento, armazenamento e comunicação para o uso de outros objetos básicos de engenharia usando os recursos do nó ao qual ele pertence. Além disso, possui interfaces de interpretação (para gerenciamento de threads dos objetos básicos) e interfaces de recursos (colocando à disposição as funções de recursos do núcleo).

Objeto Protocolo : objeto que se comunica com um objeto protocolo par para obter interação dentro de um canal.

Trader : lida com a negociação de serviços entre objetos. O trader pode estar localizado em outro nó conforme mostra a figura 1.5 e surge no ponto de vista computacional, tendo, evidentemente, implicações aqui na linguagem de engenharia.

1.2.6 Funções ODP

Uma função ODP é uma função necessária para suportar Processamento Distribuído Aberto. A necessidade de funções ODP particulares surge das definições

das linguagens dos pontos de vista.

Atualmente as principais funções são :

- **Funções de Repositório :** Função de Armazenamento, Função de Relocação, Função de Repositório de Tipos, Função de Trading.
- **Funções de Gerenciamento :** Função de Núcleo, Função de Gerenciamento de Objetos, Função de Gerenciamento de Cluster e Cápsula, Função de Gerenciamento de Domínio de Comunicação e Função de Gerenciamento de Interface de Referência.
- **Funções de Segurança :** Função de Controle de Acesso, Função de Auditoria, Função de Autenticação, Função de Confidencialidade, Função de Integridade, Função de Não-Repúdio.
- **Funções de Transparência :** Transparência de Acesso, de Concorrência, de Falha, de Federação, de Migração, de Grupo, de recursos.
- **Função de Transação.**
- **Função de Grupo.**

A maioria destas funções ainda se encontra em estágio de desenvolvimento preliminar nos documentos da ISO. Nas próximas subseções detalharemos as funções de *Armazenamento* e de *Repositório de Tipos* que envolvem o Trader que constitui o tópico central abordado na dissertação.

1.2.7 Funções ODP Relativas ao Trader

Estas funções dão suporte ao trader, porém não são de uso exclusivo dele. Por isso, elas são funções genéricas apresentando grande utilidade para potencialmente todo objeto do modelo de engenharia ODP.

Função de Armazenamento

Esta função fornece um repositório de dados. Ela "encapsula" os dados criando um objeto chamado *data object* (figura 1.6) que possui uma interface ("x") com operações para :

- retornar uma cópia dos dados;
- escrever por cima dos dados;

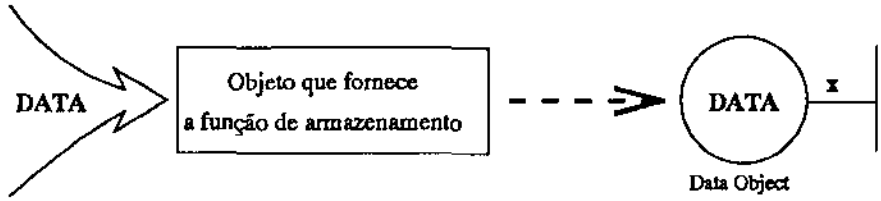


Figura 1.6: Função de Armazenamento.

- liberar espaço dos dados, liberando a interface.

Esta é uma das funções mais básicas do modelo, sendo usada, por exemplo, no momento da desativação de um cluster.

Função de Repositório de Tipos

Esta função provê um meio de dar nomes a tipos e de estabelecer relacionamentos entre eles. Além disso, permite a combinação dinâmica de tipos que inclui :

- decidir quando dois tipos são equivalentes;
- decidir quando um tipo é um subtipo de outro.

Do ponto de vista computacional, o repositório de tipos fornece uma interface de administração e uma interface para clientes.

A interface de administração fornece operações para :

- modificar o contexto de nomes do gerenciador de tipos;
- modificar o conjunto de associações $< nome\_tipo, tipo >$;
- modificar o conjunto de relacionamentos entre tipos de nomes.

A interface do cliente fornece operações para :

- recuperar o contexto de nomes suportado pelo gerenciador de tipo;
- procurar nomes de tipos e tipos usando as associações $< nome\_tipo, tipo >$;
- navegar através de grafos de relacionamentos.

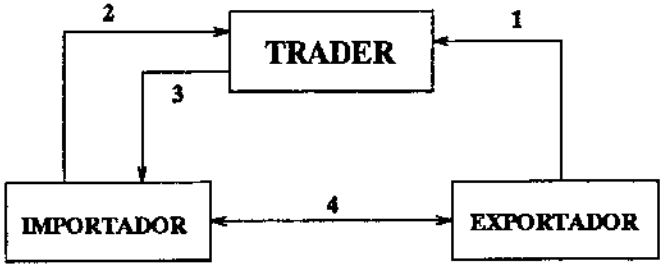
Esta função também provê os meios para descrever e comparar tipos de interfaces. Cada tipo tem a sua própria descrição de tipo e pode ser comparado com outras descrições de tipo.

1.3 Visão Geral do Trader

O propósito da Função de Trading é permitir a identificação e o acoplamento dinâmico de interfaces computacionais, e o **trader** é o objeto a partir do qual outro objeto pode importar ou exportar serviços em um ambiente distribuído[6]. Assim, um objeto (o *exportador*) informa ao trader da sua existência e fornece informações a respeito de si mesmo (de serviços que ele quer colocar "à disposição"). Outro objeto (o *importador*) contacta o trader, solicita e recebe o nome de um objeto que satisfaz as características do serviço desejado. O objeto exportador pode, a qualquer momento, retirar a sua oferta de serviços conforme lhe convier.

Portanto, a função do trader é selecionar um exportador de serviços adequado a partir de um conjunto de objetos que exportam serviços, de forma que sejam satisfeitas as características (estáticas ou dinâmicas) requeridas pelo importador.

1.3.1 Interações do Trader



1. Exportador informa trader sobre serviços oferecidos.
2. Importador solicita serviços.
3. Trader informa importador a respeito de exportador do serviço solicitado.
4. Interação entre importador e exportador.

Figura 1.7: Interações do Trader com seus usuários.

De acordo com a figura 1.7, um trader :

1. Aceita **ofertas de serviços** de exportadores de serviços (servidores) quando estes desejam anunciá-las. Uma **oferta de serviço** contém as características de um serviço que um exportador deseja fornecer. As ofertas

[6]Simplificadamente, o trader é um repositório de anúncios de serviços com operações para incluir e listar estes anúncios.

de serviço são armazenadas pelo trader num banco de dados centralizado ou distribuído.

2. Accita uma **requisição de serviço** de importadores de serviços (clientes). Uma **requisição de serviço** é uma expressão dos requisitos de um serviço necessário ao importador.

3. Procura no seu banco de dados por uma oferta de serviços que combine com a requisição de serviço do importador. E, se necessário, um trader pode selecionar a oferta de serviços mais apropriada (se existe uma) que satisfaz as restrições do importador. A lista de serviços encontrados ou o serviço selecionado é retornado ao importador.

Depois de uma busca bem sucedida, o cliente pode interagir com um servidor cujas características satisfazem ao seu pedido (interação 4 na figura 1.7).

É importante notar que o mesmo objeto pode ser exportador de um serviço e importador de um outro serviço. Além disso, clientes do trader (importadores, exportadores ou outros traders) podem requisitar importações ou exportações em nome de outros objetos.

O importador e o exportador podem estar localizados em nós diferentes e as funções de transparência de distribuição do ODP é que são as responsáveis por esconder estas diferenças de localização.

1.3.2 Federação de Traders

Uma Federação de Traders é uma coleção de traders cooperativos mas autônomos.

Através da federação, uma oferta de um trader componente será conhecida para uma audiência mais ampla e um trader componente da federação terá um mercado maior para suprir as suas necessidades.

Uma federação de traders também pode ser formada quando o número de objetos administrados por um trader se torna tão grande que não é possível que a administração sirva eficientemente aos seus usuários. Através do particionamento das responsabilidades administrativas do banco de dados do trader em traders cooperativos, uma federação de traders pode ser formada.

Numa federação, devem existir mecanismos para permitir cooperação entre traders que são autônomos, heterogêneos e distribuídos. Além do mais, os objetivos do ODP exigem a transparência de distribuição de modo que um usuário de um trader federado será associado a somente um trader e acessará transparentemente outros traders através daquele trader.

Aspectos Fundamentais de Traders Federados

Os aspectos fundamentais de traders federados incluem separação, heterogeneidade, autonomia e transparência para serem coerentes com o modelo de referência ODP.

- **Separação :**

  O banco de dados de cada trader e o gerenciamento dos componentes individuais e de federação são distribuídos. Esta separação freqüentemente causa a perda de performance. A replicação de dados introduz o problema da manutenção da consistência entre as cópias. Entretanto, a separação aumenta a disponibilidade e a confiabilidade do sistema como um todo.

- **Heterogeneidade :**

  Cada trader pode ter os seus próprios mecanismos para controle de acesso, de concorrência e recuperação e também pode ter diferenças nas semânticas dos serviços e atributos. Por exemplo, a qualidade de impressão pode ser expressa com um dígito de 0 a 9, onde 0 significa a melhor qualidade num trader e a pior em um outro.

- **Autonomia :**

  Quando um trader entra para a federação, ele deve ser capaz de manter seus serviços exportados e tipos de serviços, e deve ser capaz de executar operações locais sem interferência externa. Um trader também decide quando deve agregar-se ou sair de uma federação e quanto do seu banco de dados será compartilhado com os outros traders.

- **Transparência :**

  A transparência assegura que uma solicitação do usuário estará no mesmo formato e terá o mesmo significado quando ele acessa um trader local ou remoto. Para obter esta transparência, funções de mapeamento são necessárias entre o trader local e o remoto.

Destes aspectos fundamentais podemos tirar cinco princípios básicos que caracterizam o relacionamento federativo entre os traders componentes :

1. Um componente não pode ser forçado a executar uma atividade para outro componente;

2. Um componente deve ser livre para entrar e sair de uma federação;

3. Um componente deve ser capaz de determinar quais os dados que ele deseja compartilhar com os outros;

4. Um componente determina como ele vê e combina os dados existentes;

5. Usuários existentes, atividades e dados devem sofrer alterações mínimas (se que há) quando se incluem os mecanismos de federação.

Estes princípios devem reger toda a formulação de modelos e algoritmos para a federação de traders.

A Proposta de Modelo Para Federação de Traders

O modelo proposto para federação de traders é um modelo descentralizado que tem a característica de garantir a **autonomia** de cada trader (que é um dos princípios básicos vistos na seção 1.3.2) [BEAR93].

Para ser parte de uma federação, um trader deve importar serviços de pelo menos um outro trader ou deve exportar serviços para pelo menos um outro trader na federação.

Um trader que importa serviços de um trader remoto tem um *contrato de importação* com aquele trader remoto. O contrato de importação define :

- os tipos de interfaces disponíveis no trader remoto;
- as regras para mapear requisições e respostas entre o trader local e o remoto de forma que sejam inteligíveis para ambos.

Um trader que exporta serviços para um trader remoto tem um *contrato de exportação* com o trader remoto. O contrato de exportação define :

- a extensão da informação do trader local que estará acessível ao trader remoto;
- os tipos de interface disponíveis no trader remoto;
- as regras para mapear requisições e respostas entre o trader local e o remoto de forma que sejam inteligíveis para ambos.

Para cada contrato de exportação existe um contrato de importação correspondente no trader remoto.

Um trader que exporta para um trader remoto oferece uma interface de *Estabelecimento-Federação* e uma interface de *Interações-Federadas* para o seu trader remoto [ISO7047].

O procedimento para fazer uma requisição de serviço é o seguinte : um cliente de um trader procura por um serviço no banco de dados do trader e, se necessário, o trader local busca através dos seus contratos de importação. Se o tipo de serviço solicitado está disponível num trader remoto, então o trader local mapeia a requisição em uma requisição remota e a envia para o trader remoto através da interface Interações-Federadas desse trader remoto. O trader remoto inicia uma busca no seu banco de dados, de acordo com o contrato de exportação correspondente. Quando a busca for completada, os resultados (transformados, se necessário) são retornados ao trader que está importando os serviços. A figura 1.8 mostra as interfaces entre traders com os contratos.

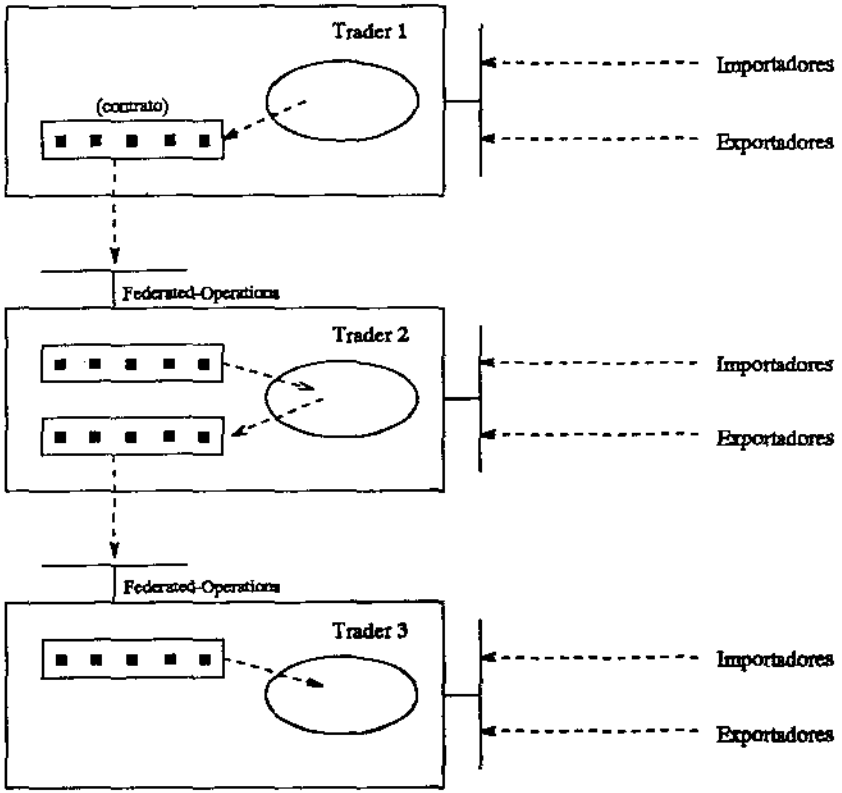


Figura 1.8: Federação de Traders.

Em resumo, as funções do trader permitem que :

- objetos exportem (anunciem) seus serviços;
- objetos importem informações sobre um ou mais serviços exportados, de acordo com algum critério;
- o conjunto de serviços exportados seja particionado de acordo com alguma política;
- haja interação com outros traders.

Através da utilização dos mecanismos de federação, problemas de performance podem ser resolvidos. Por exemplo, alguns traders podem ser pequenos em termos dos recursos que eles consomem. Eles podem manter umas poucas ofertas somente para serviços locais e podem depender da federação com outros traders em outros nós para dar acesso a um conjunto maior de ofertas de serviços.

Capítulo 2

Descrição do Trader

Neste capítulo, estaremos analisando com detalhes cada aspecto do trader a partir dos pontos de vista definidos no RM-ODP (seção 1.2.2).

2.1 Ponto de Vista Empresarial

No ponto de vista empresarial, os objetivos, as requisições funcionais e as políticas que governam as atividades de um trader são identificadas.

Um trader é o centro de uma comunidade estabelecida com o propósito de negociação[1]. As atividades negociadas na comunidade são importações e exportações de serviços, e são governadas por uma *política de negociação* do trader. A comunidade possui membros (agentes) que desempenham os papéis de :

- **Exportadores** : fazem uma oferta de serviço a um trader descrevendo a função fornecida em uma determinada interface. A função pode ser fornecida pelos próprios "exportadores" ou por outros objetos.
- **Importadores** : solicitam ao trader para informá-los a respeito de ofertas satisfazendo algum critério. Os importadores podem utilizar a oferta de serviço para o seu próprio uso ou passá-la a outros objetos.
- **Administradores** : exercem controle sobre quais objetos podem ser importadores e exportadores determinando restrições de acesso e tomando decisões sobre as políticas de negociação do trader.
- **Controladores de Política** : exercem controle externo sobre o uso das ofertas de serviço.

[1] Chamada *Trading Community*.

- **Traders :** fornecem um diretório de ofertas de serviços sob a direção de um administrador com as suas próprias políticas de negociação.

Um objeto pode ter múltiplos papéis dentro da mesma comunidade podendo ser, por exemplo, exportador e importador, simultaneamente.

Cada importador tem também a sua própria *política de importação* que restringe o conjunto de serviços considerados por um trader no processo de importação. Cada exportador tem também a sua própria *política de exportação* que restringe o conjunto de importadores para os quais um trader pode indicar ofertas de serviço.

Portanto, na importação de um serviço, o processo de busca da oferta pelo trader é governado pela :

- política de negociação da comunidade;
- política de exportação do exportador da oferta de serviço;
- política de importação do importador.

A política é expressa por regras. Os membros de uma comunidade de negociação são obrigados a obedecer a estas regras, e elas formam a linha mestra para decisões que satisfazem os requisitos funcionais da comunidade. Algumas regras importantes são :

- **Requisições de Segurança Para Importadores e Exportadores :**

  Regras para prevenir uso não autorizado, abertura e modificação de ofertas de serviços, etc.

- **Requisições de Contabilidade :**

  Regras para determinar taxas para importação e exportação, certificação de crédito, etc.

- **Requisições de Transferência :**

  Regras para dizer que importadores podem importar ofertas de serviço para o seu próprio uso ou para passá-las para outras entidades ou para dizer que exportadores podem exportar serviços que eles fornecem ou serviços fornecidos por outras entidades e coisas semelhantes.

- **Requisições de Qualidade de Serviço do Trader :**

  Regras de requisições para performance, consistência entre o que é fornecido e o que é pedido ou entre valores anunciados e reais, etc.

A especificação de um requisito empresarial tem conseqüências nos outros pontos de vista. Por exemplo, para satisfazer ao requisito empresarial de qualidade de serviço de rápido tempo de resposta, o modelo de informação pode necessitar de um particionamento especial da sua informação, o modelo de engenharia pode requerer replicação de dados, e o modelo tecnológico pode requerer transmissão de dados em faixa larga ("broadband").

Em suma, no ponto de vista empresarial ocorre a definição da política de negociação e, para determinar uma política de negociação, as seguintes questões devem ser consideradas :

- **Que qualidade de serviços será suportada ?**

  Diferentes requisições de qualidade de serviços devem ser consideradas na especificação da política. Os aspectos a serem considerados são :

  - disponibilidade;
  - performance;
  - consistência;
  - remoção de ofertas de serviços que não estão disponíveis.

- **Esta empresa é responsável pela qualidade de serviços anunciados através deste serviço de negociação ?**

  Os traders não precisam verificar a correção ou a confiabilidade de servidores individuais disponíveis para a exportação.

- **Qual é o critério para permitir federação ?**

  Discutido no capítulo 3.

2.2 Ponto de Vista de Informação

Neste ponto de vista são definidos os tipos de informações trocadas e manipuladas no processo de trading.

Para a negociação, deve haver um método para descrever :

- referências de objetos;
- referências de interfaces;

- tipos e qualidade de ofertas de serviços solicitados (especificação do importador);
- características dos objetos (propriedades estáticas de serviço dos exportadores);
- disponibilidade dos objetos (propriedades dinâmicas de serviço dos exportadores);
- seleção (combinação e acoplamento entre importadores e exportadores);
- particionamento do domínio de informação da negociação;
- efeito da federação no domínio de informações da negociação.

2.2.1 Serviços

Um *serviço* é uma função fornecida por um objeto numa interface computacional e é uma instância de um **tipo de serviço** como apresentado na figura 2.1.

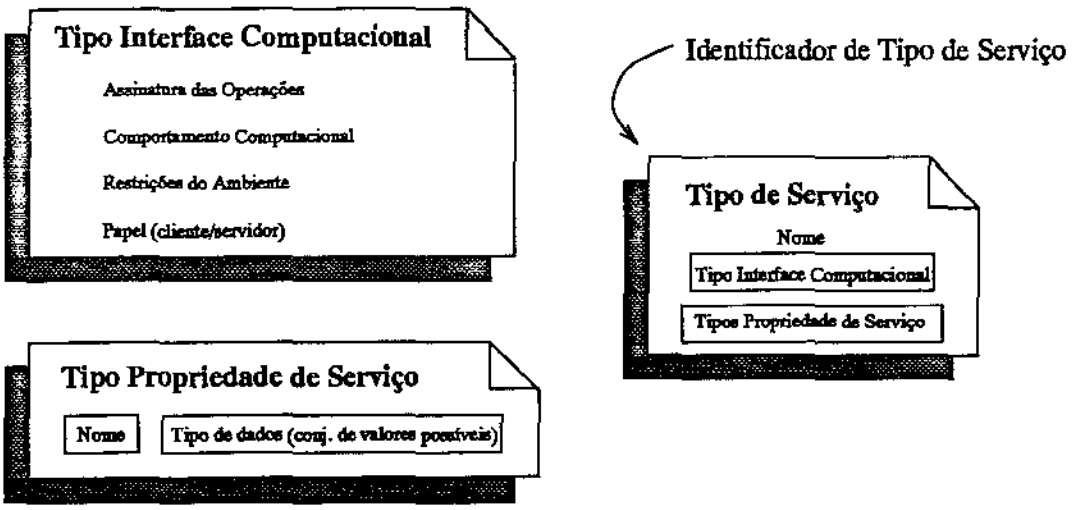


Figura 2.1: Tipo de Serviço.

Um tipo de serviço é um predicado que distingue uma classe cujos membros são serviços que satisfazem o predicado. O predicado especifica as características

comuns de uma coleção de objetos que são descritas pelas **propriedades do serviço.**

Cada propriedade do serviço consiste de um nome que identifica aquela propriedade (por exemplo, *PRINTER_TYPE*, *COST_PER_PAGE*, etc) e um tipo de dado (que é um conjunto de valores possíveis, podendo ter valores default). Para que um **serviço exportado** seja uma instância de um tipo de serviço particular, ele deve especificar o valor para todas as propriedades de serviço ou um valor default deve ser usado, como no exemplo da figura 1.2.

Cada tipo de serviço deve ter um *nome de tipo de serviço* distinto.

2.2.2 Ofertas de Serviço

Uma oferta de serviço descreve um serviço que está sendo negociado. É uma declaração feita pelo servidor a respeito de um serviço que é oferecido para ser usado por outros objetos numa interface computacional.

Os elementos da oferta de serviços estão apresentados na figura 2.2.

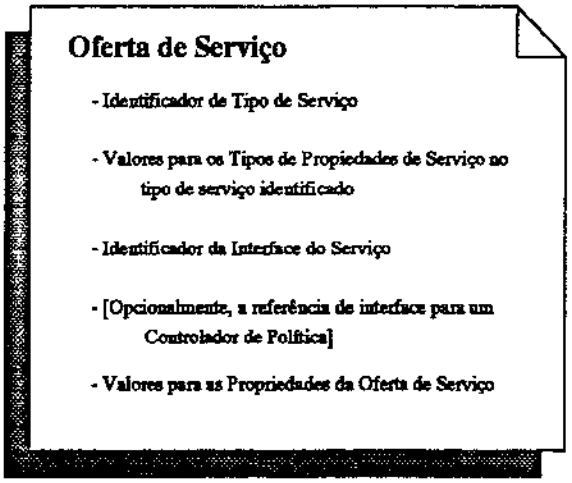


Figura 2.2: Oferta de Serviço.

É necessário que cada oferta de serviço oferecida através de um trader seja uma instância de um tipo de serviço conhecido para aquele trader.

Uma oferta de serviço faz parte de uma das seguintes classes :

- **exported** : a oferta de serviço está disponível através de trading; ou

- **withdrawn** : a oferta de serviço não está disponível através de trading.

As seguintes operações podem ser executadas sobre uma oferta de serviços :

1. *criar* - uma nova oferta é criada como um membro da classe "exported";
2. *destruir* - uma oferta de serviço é destruída;
3. *reclassificar* - uma oferta de serviço é transferida de classe para outra.

Além disso, uma oferta de serviço pode ter propriedades como tempo de duração da oferta e restrições específicas de acesso, por exemplo.

2.2.3 Contextos

As ofertas de serviços podem ser estruturadas em grupos chamados **contextos** (figura 2.3).

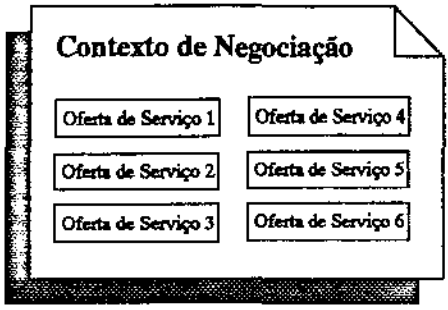


Figura 2.3: Um contexto de negociação.

À cada contexto estão associadas propriedades de grupo que podem descrever:

- algumas características do próprio grupo (p. ex., estrutura física das localizações de armazenamento das ofertas de serviços);
- algumas propriedades comuns de ofertas de serviços (p. ex., ofertas disponíveis para importadores com certas restrições de segurança);
- algumas propriedades comuns de serviços (p. ex., serviços de certos tipos).

As propriedades de grupo podem ser usadas como base para determinar a localização (ou busca) de ofertas de serviços. Por isso, a operação de exportação de uma oferta de serviço também requer um contexto para a colocação da oferta. O contexto também restringe a visibilidade da oferta de serviço.

O *Domínio de Ofertas de Negociação* é o contexto que não está contido em nenhum outro, ou seja, é o conjunto de todas as ofertas de serviços disponíveis no trader (TC-A na figura 2.4). E os relacionamentos entre os vários contextos podem ser representados através de um grafo acíclico direcionado (GAD), como apresentado na figura 2.4.

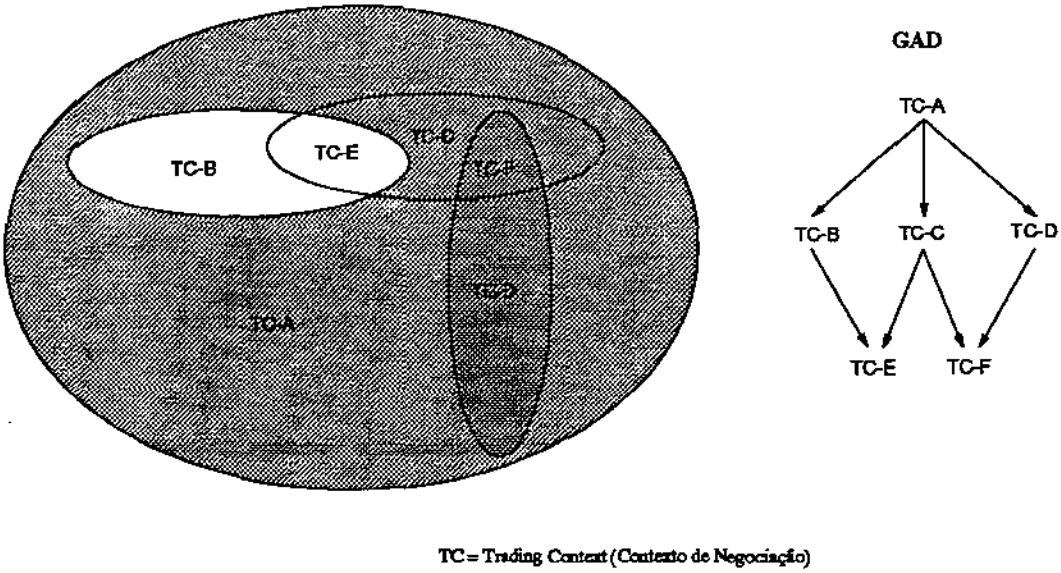


Figura 2.4: Estrutura de contextos.

De forma resumida, contextos são úteis para :

- organizar as ofertas de serviços dando-lhes certa estrutura;
- restringir acessos às ofertas de serviço;
- identificar a localização da base de dados onde estão armazenadas as ofertas de serviço, bem como os meios para acessá-las.

Contextos representam domínios de ofertas de serviços sobre os quais as operações definidas no ponto de vista computacional poderão ser executadas.

2.2.4 Requisições de Importação

Uma *requisição de serviço* é um pedido de um cliente a respeito de ofertas de serviços que satisfazem uma especificação dada. Esta especificação dos requisitos pode ser vista como sendo formada de duas partes :

- o comportamento requerido do serviço computacional que é descrito por um identificador de tipo de serviço ou um identificador de tipo de interface;
- as propriedades requeridas expressas por uma restrição de igualdade[2].

Uma requisição de serviço faz parte de uma *requisição de importação*, que, além do citado acima, também inclui :

- a identidade do importador;
- o escopo de busca que limita a busca a um conjunto de contextos (um limite de tempo e/ou um limite de tamanho da lista a ser retornada na busca também podem ser considerados);
- um método de busca que especifica os requisitos de como a busca será feita (por exemplo, uma ordem de contextos para a busca dentro do escopo).

Meios para representar o escopo de busca e o método de busca estão ainda sendo discutidos e não são abordados nesta dissertação. A política de importação identifica um escopo de busca e/ou um método de busca predefinidos [ISO107463].

2.3 Ponto de Vista Computacional

No ponto de vista computacional, são descritas as interfaces do trader. O modelo prescreve algumas interfaces computacionais do trader que serão utilizadas por outras aplicações do ambiente num sistema ODP. No nosso modelo, identificamos a necessidade de algumas outras interfaces e operações que serão discutidas com detalhes no capítulo 4.

As interfaces (não considerando aqui, federações de traders), estão divididas em interfaces de *gerenciamento* e interfaces de *negociação* ("trading").

[2]Uma *Restrição de Igualdade* (ou "matching constraint") é uma expressão booleana que consiste de valores de propriedades de serviços e (opcionalmente) valores de propriedades das ofertas de serviços. Por exemplo, uma restrição de igualdade pode ser :

$$Printer\_Type = Laser \text{ and } Cost\_per\_Page < \$0.02$$

2.3.1 Operações das Interfaces de Gerenciamento

São operações de *gerenciamento de contextos* e *políticas de importação.* Tanto os administradores como os usuários (sujeitos a algumas políticas de negociação) podem manipular a estrutura dos contextos. As operações de *gerenciamento de contexto* são :

- **Adiciona Contexto** : adiciona um contexto na estrutura de contextos com ligações para os super-contextos[3].
- **Deleta Contexto** : remove um contexto (as ligações associadas) da estrutura de contextos.
- **Lista Contexto** : lista os detalhes de um contexto (o identificador do contexto, as propriedades do contexto, e os relacionamentos com outros contextos, etc).
- **Lista Conteúdo do Contexto** : lista as ofertas de serviços armazenadas no contexto.

As operações de *gerenciamento de política de importação* são :

- **Cria Política de Importação** : define um escopo e método de busca.
- **Deleta Política de Importação** : remove uma política de importação determinada.
- **Adiciona relação** : adiciona uma relação[4] entre um determinado contexto e uma lista de contextos numa política de importação.
- **Deleta relação** : remove relação entre contextos.

2.3.2 Operações das Interfaces de Negociação

As operações destas interfaces podem ser divididas em dois grupos : operações que dizem respeito aos importadores e operações para exportadores.

Para os exportadores, o conjunto de operações é :

[3]Super-contextos são os contextos que englobam totalmente o contexto corrente (figura 2.4)

[4]Uma relação pode ser vista como sendo uma seta de um contexto para outro no caminho de busca.

- **Export** : anuncia uma oferta de serviço num determinado contexto (ou num contexto "default"). O serviço exportado é associado a um identificador de exportação que é único no contexto.

- **Withdraw** : remove uma oferta de serviço de um contexto. A oferta de serviço não está mais disponível aos importadores.

- **Replace** : uma seqüência atômica de um *withdraw* de uma oferta de serviço seguido de uma exportação de uma oferta de serviço dentro de um determinado contexto. A nova oferta de serviço tem o mesmo identificador de exportação da oferta original. Esta operação pode ser usada para alterar propriedades de serviço como, por exemplo, o custo de um serviço.

A nossa proposta para os parâmetros e valores de retorno para estas operações estão apresentados na tabela da figura 2.5.

OPERAÇÕES DO EXPORTADOR		
Operação	Parâmetros	Resultados
EXPORT	Identificador do cliente, Nome do tipo do serviço, Nome do contexto, Id. da interface do serv., Val. das prop. serviço, Id. da interface do controlador de política, Val. das prop. da oferta.	Retorna o id. da oferta de serviço ou indicação de erro.
WITHDRAW	Identificador do cliente, Nome de contexto, Id. da oferta.	Retorna ok ou ind. de erro
REPLACE	Idem WithDraw + lista de nomes de prop. modificadas e novos vals., Lista de serv. adicionais e valores, Lista de serv. a remover.	Retorna ok ou ind. de erro.

Figura 2.5: Operações disponíveis para exportadores.

Para os importadores, as operações são :

- **List** : uma requisição para retornar os *valores das propriedades de serviço* correntes ou os *valores das propriedades de oferta de serviço* de uma determinada oferta de serviço.

- **Search** : uma requisição para retornar um conjunto (possivelmente vazio) de ofertas de serviços em determinado(s) contexto(s) que satisfazem a descrição de tipo de serviço do importador e a restrição de igualdade. Um importador pode especificar o escopo da busca.

- **Select** : uma requisição para retornar a "melhor" oferta de serviço (de acordo com um critério de seleção do importador) a partir de um conjunto de ofertas de serviços que satisfizeram a restrição de igualdade.

A nossa proposta para os parâmetros e resultados para estas últimas operações estão apresentados na tabela da figura 2.6. Este conjunto final de parâmetros está em conformidade com a funcionalidade requerida para cada função, segundo o modelo ODP [ISOTRD].

2.3.3 Operações Disponíveis Para o Trader

Há ainda operações que estão disponíveis somente para o trader : "*Status Inquiry*" e "*Exporter Policy*". Elas estão disponíveis para o trader somente se o exportador incluiu um identificador da interface do seu *controlador de política de exportação* na sua oferta de serviço. Este controlador de política de exportação é um objeto associado ao exportador responsável por informar ao trader sobre propriedades dinâmicas de serviço do exportador (como filas de impressão, etc) assim que ele (o trader) requisitar.

A operação *Status Inquiry* consiste de uma requisição do trader para obter informações do estado corrente de determinada propriedade dinâmica de serviço de um serviço exportado.

Com a operação *Exporter Policy* o trader pede ajuda ao controlador de política de exportação a respeito de um serviço selecionado. O trader envia um identificador de uma interface de serviço e recebe :

- o identificador da interface de serviço, se tudo estiver bem;

- o identificador de uma interface de serviço substituta, devido a congestionamento na rede, por exemplo;

- um identificador de interface nulo, indicando falha na operação devido a, digamos, falta de recursos.

OPERAÇÕES DO IMPORTADOR		
Operação	Parâmetros	Resultados
LIST_OFFER_DETAILS	Id. do cliente, Nome de contexto, Id. da oferta, Lista de prop. de serviço consultada.	Retorna os valores das prop. de serviço da oferta pedida ou código de erro.
SEARCH	Id. do cliente, Nome do(s) contexto(s), Nome do tipo de serv., Critério(s) de igualdade, Lista de prop. de serv. a retornar, Lista de prop. das ofertas a retornar.	Resulta na lista de serv. exportados e suas props. Cada oferta tem valores que satisfazem aos critérios de igualdade.
SELECT	Id. do cliente, do contexto, do tipo de serv., Restrição de iguald., Critérios de seleção, Lista de props., Limites de busca.	Retorna oferta de serv. exportada que satisfaz as restrições de igualdade sendo a mais adequada em relação aos critérios de seleção.

Figura 2.6: Operações disponíveis para importadores.

Para selecionar ofertas de serviços adequadas às requisições de serviços, um trader interage com a função de repositório de tipos fornecida pela infraestrutura ODP (seção 1.2). O conjunto de todos os tipos de serviços conhecidos de um trader é referido como sendo o seu **repositório de tipos** (seção 1.2.7). O repositório de tipos também conhece os tipos de interfaces associados para cada tipo de serviço e mantém um conjunto de relacionamentos (grafo) para os tipos de interfaces. Os relacionamentos incluem relações de equivalência e de super/sub-tipos. Além disso, o repositório de tipos mantém a relação de super/sub tipo entre os tipos de serviços conhecidos. Um trader precisa das seguintes operações oferecidas pelo repositório :

Type Check : para verificar a validade dos tipos nas ofertas de serviços e requisições de serviços;

Compatibility Check : para verificar se um serviço ou um tipo de interface é

compatível com outro tipo de serviço ou de interface;

List Types Relationship : para obter a lista de subtipos de um determinado tipo de serviço ou de interface.

O trader interage com o repositório de tipos utilizando-se destas operações em basicamente três situações :

- **Exportação de Serviços**

  O trader recebe uma oferta de serviço de um exportador. Então ele checa com o seu repositório de tipos se o tipo de serviço (ou de interface), as propriedades de serviços e as propriedades de ofertas de serviço especificados são válidas. A oferta de serviço é armazenada na base de dados do trader incluindo os identificadores de tipo de serviço das ofertas (se fornecidos), identificador de tipo de interface e propriedades de serviço e de ofertas de serviço.

- **Importação de Serviços**

  O trader recebe uma requisição de um serviço de um cliente. Então ele verifica com o seu repositório de tipos se a requisição contém um tipo de serviço ou de interface conhecido e se as propriedades nas restrições de igualdade são válidas.

- **Seleção de Ofertas**

  O trader retorna uma lista de ofertas (possivelmente vazia) para o importador. Esta lista de ofertas satisfaz as especificações requeridas pelo importador. O trader precisa primeiramente consultar o seu repositório de tipos submetendo o identificador de tipos (de uma requisição de serviço válida). O repositório de tipos retorna então uma lista de tipos (identificadores) que são compatíveis (idênticos e/ou sub-tipos) com o tipo que fora requisitado.

  Para achar uma solução o trader procura no seu banco de dados por ofertas de serviços que se encaixam no(s) tipo(s) de serviço ou de interface (como fornecido pelo seu repositório de tipos), nos requisitos de propriedades de serviço e de oferta de serviço (como especificado na restrição de igualdade) e nos requisitos de gerenciamento (questões de segurança, por exemplo). Esta busca tem o escopo limitado pela *política de importação* e a visão de ofertas limitada pela *política de exportação.*

  A partir da lista de ofertas de serviços que se encaixaram no que foi pedido, o trader seleciona (se for requisitado) uma oferta de serviço baseado

num critério de seleção determinado. Se a lista não for vazia, então o trader retorna o identificador da interface do serviço selecionado para o seu importador.

Se a seleção de propriedades dinâmicas for pedida, então o identificador da interface do controlador de política de exportação é usado para determinar os valores das propriedades dinâmicas.

Para usar o serviço, o importador precisa mapear o identificador de interface de serviço em uma localização de serviço, estabelecer um binding com o servidor na localização do serviço e, finalmente, chamar o serviço.

2.4 Ponto de Vista de Engenharia

Neste ponto de vista as estruturas e os mecanismos necessários para implementar as funções computacionais estão visíveis. Obviamente, diferentes abordagens para a implementação são possíveis. A distribuição da informação mantida pelo trader também está visível neste ponto. Os detalhes para este ponto de vista serão considerados com detalhes na seção 4.2.

Capítulo 3

Federação de Traders

A *Federação de Traders* faz com que ofertas de serviços remotas pareçam fazer parte do espaço local de busca. Em outras palavras, traders formam federações para permitir que ofertas de serviços de um trader sejam disponíveis para usuários de outros traders. Um usuário é um cliente direto de um único trader e acessa os outros traders indiretamente através da federação.

Cada contrato de federação estabelecido entre dois traders determina qual será o papel desempenhado por cada um deles : o trader que estiver oferecendo a sua base de dados (ainda que com restrições) é chamado de "*trader exportador*", enquanto que o trader que estiver utilizando os serviços é chamado de "*trader importador*".

Neste capítulo, estaremos considerando o mecanismo para a formação de federações de traders estabelecendo também pontos que os documentos da ISO deixaram em aberto. No entanto, faremos isto sem ferir os princípios gerais que regem um ambiente que se diz adequado para processamento distribuído aberto.

3.1 Ponto de Vista Empresarial de Traders Federados

A partir do ponto de vista empresarial, uma federação de traders é uma comunidade de comunidades de negociação estabelecida com o propósito de fazer com que ofertas de serviço de cada comunidade estejam disponíveis para outras comunidades. As atividades de negociação são exportações e importações federadas além do estabelecimento de federações. Estas atividades são controladas por *políticas de negociação de federação* de comunidades de negociação individuais. Alguns pontos importantes destas políticas são :

- *Requisitos de Exportação Federada* - regras para determinar as partes do espaço de ofertas de serviço do trader a ser exportada através da federação.
- *Requisitos de Acesso Federado* - regras para determinar quando atividades federadas se tornam disponíveis para usuários locais. Por exemplo, uma importação federada só acontecerá se o banco de dados do trader local não puder satisfazer as requisições do usuário.
- *Requisitos Financeiros* - regras para determinar quanto cobrar e quanto pagar por um determinado serviço.
- *Requisitos de Segurança.*

3.2 Ponto de Vista da Informação de Traders Federados

Para haver a interação entre traders numa federação, eles precisam saber :

- a identificação dos outros traders da federação;
- o que está disponível para exportação por um trader;
- os tipos de serviços acessíveis para um trader importador;
- restrições de acesso à base de dados de um trader exportador.

Estas informações são fornecidas por *catálogos*, *contratos de federação*, *contratos de importação* e *contratos de exportação.*

3.2.1 Catálogos

O catálogo contém informações de tipos de serviços e do sub-espaço que o trader colocará à disposição de outros traders. Também determina a política do trader exportador para operações permitidas via federação e a extensão do "linking" para outros traders, ou seja, se o trader exportador passará o pedido adiante a outro trader da federação ou não.

O catálogo é uma *propriedade de serviço* da oferta de serviço padrão de *Estabelecimento de Federação* (veja seção 2.2). Assim, quando um trader deseja compartilhar o seu espaço de busca com outros traders, ele deve primeiramente preparar e divulgar o seu catálogo como uma propriedade da oferta de serviço de Estabelecimento de Federação. Quando um trader quer acessar o espaço de

busca de um outro trader, ele deve primeiro obter uma oferta de serviço de Estabelecimento de Federação com o catálogo do trader exportador.

Um catálogo contém entradas. Propomos que cada uma delas contenha as informações contidas na figura 3.1.

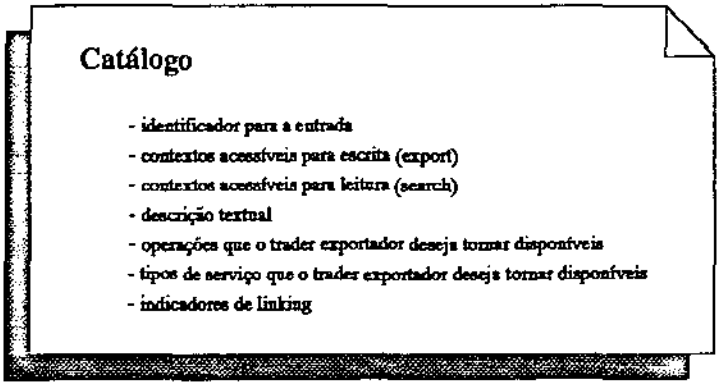


Figura 3.1: Uma entrada do catálogo.

Os *indicadores de linking* indicam quando um trader exportador passará adiante requisições de outros traders. Há dois tipos de indicadores de linking :

- **Upstream-linking** : desejo de passar adiante (para exportadores) ou de aceitar (para importadores) requisições de outros traders.
- **Downstream-linking** : desejo de requisitar (para exportadores) ou de passar adiante (para importadores) requisições de outros traders.

A descrição textual na entrada do catálogo tem por objetivo informar usuários humanos e administradores a respeito de ofertas disponíveis através deste trader e quaisquer condições relacionadas ao acesso a estas ofertas.

3.2.2 Contrato de Federação

Se um trader importador estiver interessado num catálogo de um trader exportador, ele propõe um contrato de federação baseado nas ofertas de trader exportador que estão presentes no catálogo e nas suas próprias necessidades de serviços. Um **contrato de federação** registra o acordo negociado entre os dois traders que fazem parte da federação. As informações contidas num contrato de federação estão apresentadas na figura 3.2.

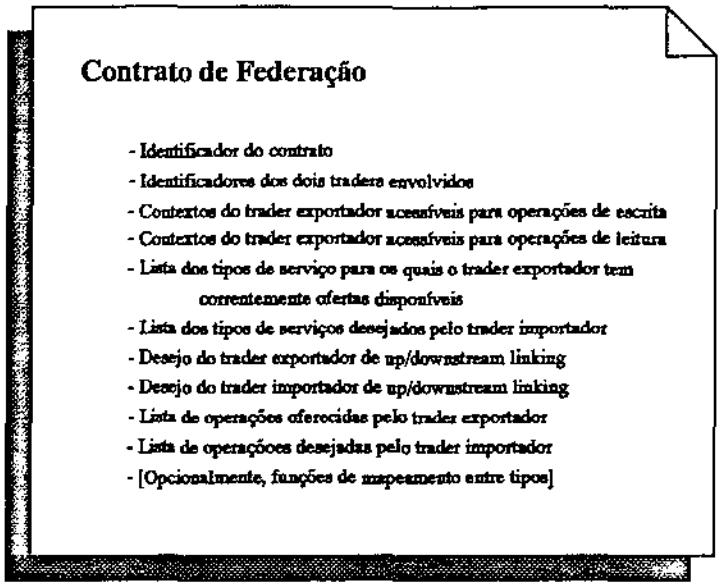


Figura 3.2: Um contrato de federação.

As operações federadas disponíveis através do contrato são aquelas resultantes da interseção das operações oferecidas pelo trader exportador com aquelas que o trader importador deseja. As ofertas de serviço disponíveis são aquelas que pertencem à interseção dos tipos de serviço disponíveis no espaço de busca do trader exportador com aquelas as quais o trader exportador deseja tornar visíveis.

Um contrato de federação é mantido em duas partes : o *contrato de importação* que está no trader importador e o *contrato de exportação* no trader exportador. O contrato de importação contém informações do contrato de federação que são úteis apenas para o trader importador, e é armazenado no banco de dados do trader importador como sendo uma propriedade de serviço padrão do *Serviço de Interação Federada* oferecida pelo trader exportador. O mesmo ocorre para o contrato de exportação com relação ao trader exportador. A principal diferença entre eles é que o contrato de exportação pode estar armazenado numa base de dados independente, mas o contrato de importação precisa, necessariamente, fazer parte de uma oferta de serviço localizada num determinado contexto do diretório. Isto, porque o contrato de importação deve ser identificado no momento da busca de serviços que satisfaçam a determinadas propriedades num dado contexto, enquanto o contrato de exportação só é analisado no mo-

mento em que se faz a requisição de uma operação numa interface de interações federadas.

As informações dos contratos de importação e exportação estão na figura 3.3.

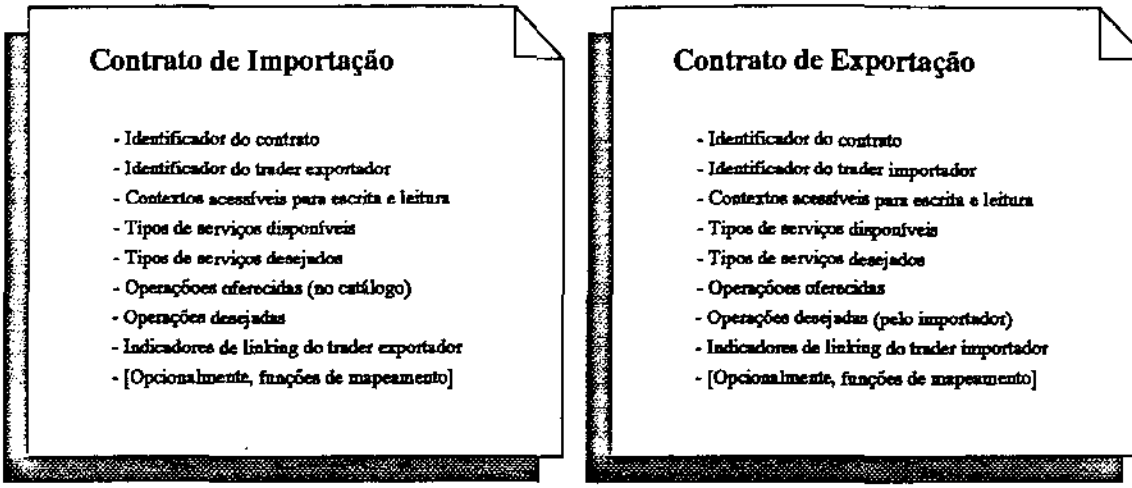


Figura 3.3: Contratos de importação e de exportação.

3.3 Ponto de Vista Computacional de Traders Federados

Neste ponto de vista, para federação, três conjuntos de interfaces estão visíveis :

- **Interface de Negociação** : usado pelo trader agindo como se fosse um cliente direto para exportar/importar o catálogo, para listar/repor/buscar contratos de importação ou de exportação e para remover um contrato de federação.
- **Interface de Estabelecimento de Federação** : usada por um trader agindo como cliente direto para negociar o contrato de federação.
- **Interface de Interação de Negociação Federada** : usada por um trader agindo em nome de um dos seus usuários locais para executar operações como "search" e "export" via federação.

3.3.1 Estabelecimento de Federação

O primeiro passo para estabelecer uma federação entre dois traders é fazer com que o trader importador obtenha o catálogo do trader exportador. Isto pode ser conseguido de duas maneiras :

- o trader exportador "divulga" o seu catálogo para um potencial trader importador; ou
- o trader importador requisita o catálogo de um trader exportador.

Distribuindo o Catálogo

Se um trader exportador deseja federar-se com algum outro trader, ele precisa oferecer o seu serviço de *Estabelecimento de Federação* com um catálogo para um possível trader importador. O catálogo é enviado pelo trader exportador agindo como um cliente do trader importador e usando a operação *EXPORT* do trader importador para anunciar o seu serviço de *Estabelecimento de Federação* com a propriedade de serviço padrão "*Catalogue*" (figura 3.4).

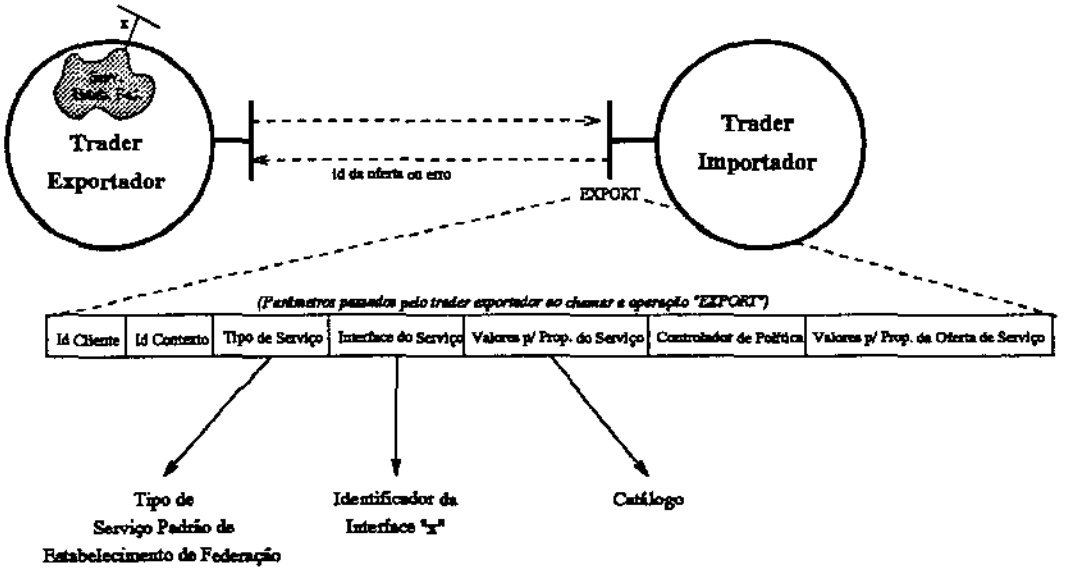


Figura 3.4: Distribuindo o catálogo.

Se não houver erros, o catálogo será conhecido pelo trader importador que então poderá iniciar a negociação do contrato se estiver interessado nas exportações listadas no catálogo. Para isso, usa a operação *EXCHANGE CONTRACT* (item "*Estabelecendo um Contrato de Federação*", abaixo).

Requisitando o Catálogo

A requisição de um catálogo é feita por um trader importador agindo como um cliente do trader exportador usando a operação "*SEARCH*" do trader exportador para obter a oferta de serviço de *Estabelecimento de Federação* com o catálogo do trader exportador como uma propriedade de serviço (figura 3.5).

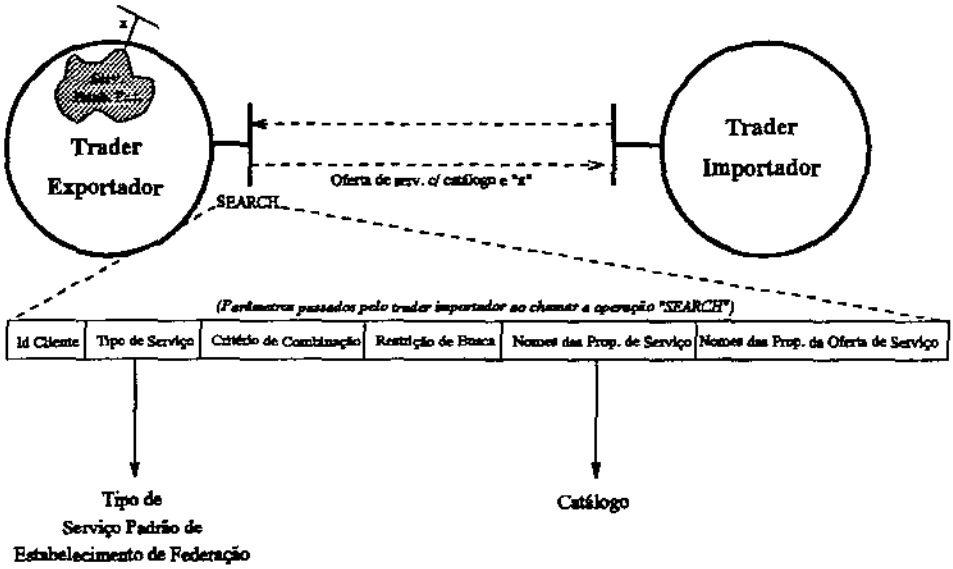


Figura 3.5: Requisitando o catálogo.

A execução bem-sucedida retornará a oferta de serviço de estabelecimento de federação do trader exportador (juntamente com o catálogo) ou um indicador que não há tal oferta de serviço (e, portanto, nenhum catálogo). No segundo caso, um contrato de federação não pode ser estabelecido com aquele trader. Se o trader importador estiver interessado nas entradas do catálogo, ele pode iniciar a negociação do contrato de federação.

Estabelecendo um Contrato de Federação

Se o trader importador estiver interessado no catálogo, ele pode iniciar uma federação utilizando-se da operação "*EXCHANGE_CONTRACT*" da interface de *Estabelecimento de Federação* do trader exportador. Esta operação tem como parâmetros a proposta de contrato de federação feita pelo trader importador e informações de segurança. O trader exportador examina a proposta de contrato à luz das suas políticas internas de acesso e aceita, rejeita ou rejeita propondo um outro contrato (reduzido). No caso de aceitar o contrato proposto, o trader exportador instancia um "*Serviço de Interação de Negociação Federada*" com uma interface associada, cria e armazena o contrato de exportação juntamente com o serviço e retorna um identificador da interface de serviço criada (figura 3.6). Um trader importador reage a uma proposta positiva criando o contrato de importação correspondente e armazenando o contrato de importação com a oferta de serviço de interação de negociação federada. Se uma resposta negativa é recebida com um contrato reduzido, o trader importador pode avaliar o contrato reduzido, e, se for aceitável, pode invocar novamente a operação "*EXCHANGE_CONTRACT*" com o contrato reduzido como parâmetro. O trader importador não faz nada se somente uma resposta negativa é recebida. O parâmetro de informações de segurança é opcional e não está especificado na documentação ODP.

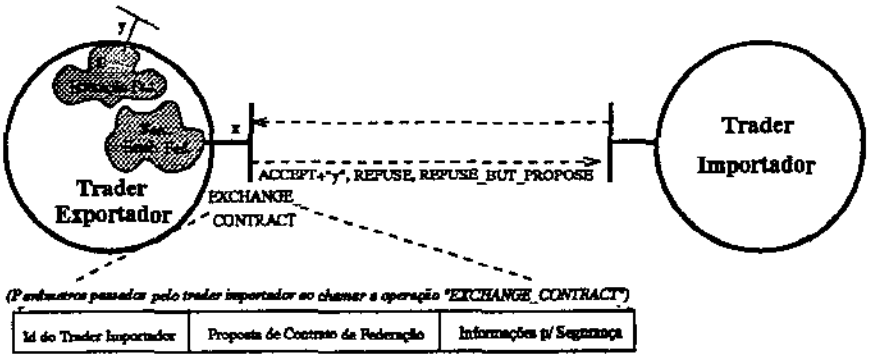


Figura 3.6: Estabelecimento de federação.

3.3.2 Gerenciamento de Federação

Usando a interface de trading (veja seção 2.3), qualquer um dos traders (importador or exportador), pode listar, buscar, modificar ou remover um contrato de importação ou exportação. Isto é possível, porque um contrato faz parte de uma oferta de serviço, e portanto, todo gerenciamento destes contratos pode ser feito através das operações sobre as ofertas de serviços.

Um trader exportador pode modificar suas "ofertas", por exemplo, quando há mudanças nos tipos de serviços disponíveis ou mudanças na política empresarial quanto a operações permitidas. Um trader importador pode modificar suas "requisições" quando, por exemplo, houver mudanças nos tipos de serviços que ele deseja ou mudanças na política empresarial. O contrato implementa a interseção entre "ofertas" e "requisições".

Qualquer um dos traders envolvidos pelo contrato pode terminar uma federação. Usando a interface de trading, um trader pode remover a oferta de serviço local de interação federada (com a parte local do contrato de federação) e pode remover a correspondente oferta de serviço remota de interação federada (com a parte remota do contrato de federação). Quando um Serviço de Interação de Negociação Federada é removido do trader exportador, o trader exportador deve remover a sua interface de serviço de interações federadas.

3.3.3 Interações Federadas

Usando a interface de interações federadas e agindo em nome dos seus usuários, um trader federado pode executar qualquer operação padrão do trader (i.e., *search, list_offer_details, select, export, withdraw, replace*). Entretanto, as operações que o trader pode utilizar na realidade, são aquelas que foram negociadas e que estão prescritas no contrato de federação. Ou seja, o conjunto de operações que um trader pode usar em nome dos seus clientes depende das políticas dos traders para federação e é negociado durante o estabelecimento do contrato. Para as interações federadas, segurança e contabilidade são responsabilidades dos usuários finais.

3.4 Pontos de Vista de Engenharia e Tecnológico de Traders Federados

No ponto de vista de engenharia, as estruturas e mecanismos necessários para o armazenamento e recuperação de contratos de importação ou exportação estão

visíveis. Mecanismos para otimizar o mapeamento entre formas canônicas e locais também são consideradas.

O ponto de vista tecnológico preocupa-se com o hardware e software que contém traders federados implementados. O sistema de software e hardware disponível restringe escolhas específicas de como trocar dados, como armazenar dados e como implementar as estruturas e funções especificadas nos pontos de vista da informação, computação e engenharia.

Capítulo 4

Aspectos de Modelagem e Implementação

4.1 Modelagem

4.1.1 Mecanismo Para Federação

Modelo

Em nosso trabalho identificou-se basicamente dois grupos de atividades desempenhadas por um trader: o gerenciamento de bases de dados relacionadas às informações (estáticas e dinâmicas) mantidas pelo trader e a execução propriamente dita das operações oferecidas pelo trader utilizando-se das informações armazenadas (figura 4.1).

O modelo que propomos consiste de um refinamento destes dois módulos, considerando separadamente funções locais e aquelas relativas ao estabelecimento e "manutenção" de federações, bem como separando em módulos diferentes o gerenciamento/obtenção das informações estáticas e dinâmicas. Assim chegamos a um modelo (figura 4.2) que supre toda a funcionalidade requerida de um trader (inclusive para federação), conforme especificado pela ISO [ISO7047]. Neste modelo identificam-se três elementos básicos (agentes, no ponto de vista empresarial): o trader, o administrador (local e de federação) e o cliente (importador ou exportador). Estes três serão alvo da implementação descrita na seção 4.2.

O administrador local e o de federação, que podem ser um só, são responsáveis pela definição e cumprimento de certas políticas de negociação locais e a nível de federação, respectivamente. O *administrador local* acrescenta novos tipos de serviços a um dado contexto, cria, destrói e renomeia contextos de ofertas de

Figura 4.1: Atividades básicas de um trader.

serviços, além de autorizar determinados clientes a utilizarem (para busca ou exportação) os vários contextos existentes. Já o *administrador de federação* é responsável por preparar o catálogo, requisitar o catálogo, decidir quando e com quem federar-se, e aceitar, recusar ou formular propostas de contratos definindo assim políticas para o estabelecimento da federação. A razão pela qual os administradores estão localizados exteriormente em relação ao trader é para que eles possam gerenciar vários traders em diversos nós. Ou seja, um trader pode ter somente um administrador, porém um administrador pode gerenciar vários traders.

Dois módulos são responsáveis pelo armazenamento e/ou obtenção de informações no trader. São eles :

- **Módulo de Informações Estáticas**

  Concentra as operações que manipulam as informações estáticas que correspondem a tipos de serviços (no Repositório de Tipos) e ofertas de serviço (no Diretório).

- **Módulo de Informações Dinâmicas**

  Responsável por atualizar as propriedades dinâmicas de uma dada oferta de serviço. Este módulo entra em contato com o *controlador de política* do objeto exportador para obter as informações dinâmicas. Um exportador de um serviço não tem que ter necessariamente um controlador de política associado à sua oferta de serviço. Se o identificador do controlador de

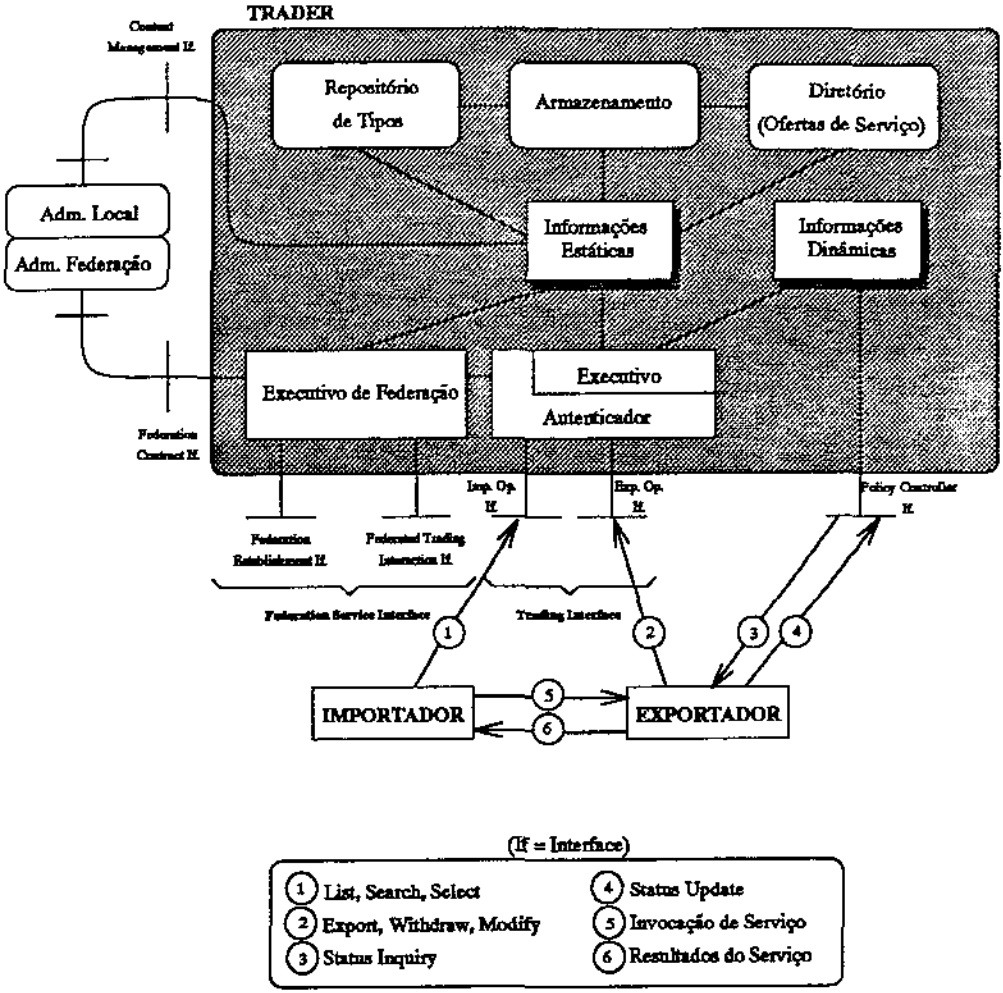


Figura 4.2: Esquema interno de um trader.

política é nulo (indicando sua inexistência) nenhuma operação é feita. Pode ser implementado usando o X.700 da ITU-T.

O *Repositório de Tipos* suporta o armazenamento (utilizando-se do módulo de Armazenamento) e a combinação dinâmica de tipos de serviços para indicar, por exemplo, quando dois tipos são equivalentes, quando um tipo é sub-tipo de outro e assim por diante (veja seção 1.2.7). Na realidade, o repositório de tipos pode ser um objeto independente do trader com suas próprias interfaces e operações.

O *Diretório* armazena as ofertas de serviços exportadas para o trader. Pode ser implementado utilizando-se o serviço de diretório X.500 da ITU-T.

O módulo *Executivo* é o responsável por executar as operações locais do trader utilizando-se de informações estáticas e dinâmicas. O papel do *Executivo de Federação* é analisar as requisições que chegam a uma interface de interações federadas baseado no seu respectivo contrato de exportação e passar (caso as restrições do contrato estejam satisfeitas) a operação para o módulo *Executivo*. Além disso, ele também é o responsável pela negociação e estabelecimento do contrato de federação.

O *Autenticador* é o módulo que verifica a permissão do importador/exportador para executar uma dada operação. Também é responsável por aplicar as restrições de acesso às bases de dados do trader, como definido pelo(s) administrador(es).

O *Administrador Local* utiliza o módulo de *Informações Estáticas* para desempenhar o seu papel de criar e deletar contextos, bem como gerenciar questões de segurança relativas às informações estáticas. Já o *Administrador de Federação* utiliza-se do módulo *Executivo de Federação* para estabelecer contratos de federação. Estes contratos são armazenados como sendo parte das informações estáticas do trader.

O trader, ao receber uma requisição normal (não federada) de uma exportação, verifica, através do autenticador, se há permissão para tal exportação. Em caso positivo, o módulo *Executivo* solicita a exportação ao módulo de *Informações Estáticas*, que, por sua vez, verifica com o *Repositório de Tipos* a coerência do tipo do serviço que será exportado. Se tudo estiver em ordem, a oferta de serviço é armazenada no *Diretório* que pode ter uma função de armazenamento de uso exclusivo seu, ou utilizar-se do módulo de *Armazenamento* proposto no modelo.

Ao receber um pedido de importação, após verificar as permissões de acesso, o módulo executivo tenta obter informações estáticas e dinâmicas dos respectivos módulos, informações essas que satisfazem às requisições do importador. Se não existirem localmente ofertas de serviços nessas condições, um pedido de importação federada é encaminhado ao módulo *Executivo de Federação*. Este módulo verifica a existência de contratos com outros traders e envia o pedido de

uma busca federada (se houver algum contrato) ao trader remoto.

A partir deste modelo, identificamos a necessidade de uma série de interfaces que o trader precisa ter para ser administrado e para atender à funcionalidade que é dele requerida. Este conjunto de interfaces do trader está representado na figura 4.3. As operações de cada interface estão na tabela da figura 4.4.

As interfaces de *operações de importação* e *de exportação* formam juntas a interface de "*trading*" que é, normalmente, a interface mais utilizada numa Comunidade de Negociação. É através desta interface que importações e exportações podem ser feitas num trader.

As interfaces de *Gerenciamento de Contexto* e de *Contrato de Federação* são de uso exclusivo dos administradores local e de federação, respectivamente.

A interface do *Controlador de Política* permite ao trader obter as informações dinâmicas de uma oferta de serviço.

A interface de *Estabelecimento de Federação* oferece a outro trader a possibilidade do estabelecimento de um contrato de federação entre eles.

Finalmente, a interface de *Interações Federadas* recebe requisições de importações e exportações de outros traders que estão agindo em nome dos seus clientes, e com os quais existe um contrato estabelecido. Existe uma interface de *Interações Federadas* distinta para cada contrato de exportação do trader.

4.2 Implementação

Idealmente a implementação do modelo descrito na seção 4.1.1 deveria ser feita sobre a plataforma ODP utilizando-se das estruturas, transparências e facilidades para a construção de aplicações distribuídas lá fornecidas. Uma outra alternativa seria o uso de plataformas como DCE[1] [OSF1 90, OSF2 90, OSF3 90, OSF4 90, OSF5 90, OSF6 90], ORB[2] [OMG1, OMG2] ou Multiware [MEND94] que fornecem um ambiente adequado para processamento distribuído aberto. A plataforma ANSAware[3] [ANSA89, ANSA91, ANSA93], já possui um trader, mas mesmo este necessita de modificações para comportar especialmente os mecanismos de federação definidos anteriormente.

A proposta inicial foi implementar o trader sobre a plataforma Multiware (seção 4.3) (baseada em ORB) em desenvolvimento na UNICAMP. Devido ao fato desta plataforma estar em fase de projeto e início de implementação, inicial-

[1] DCE : Distributed Computing Environment
[2] ORB : Object Request Broker
[3] ANSA : Advanced Network Systems Architecture

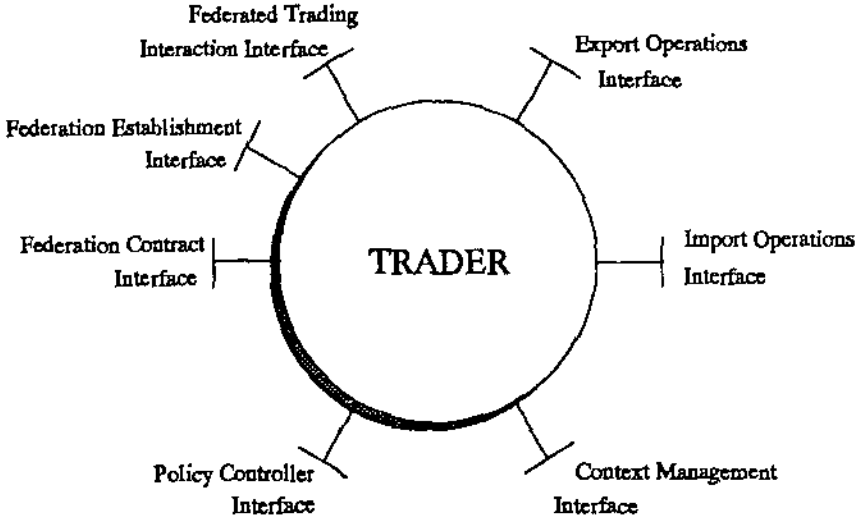


Figura 4.3: Interfaces do Trader (proposta).

mente foi construído um protótipo de um trader diretamente em cima do sistema operacional UNIX. Também foram implementados um *cliente* (importador e exportador) para testes e o administrador. Para tanto, são utilizadas as chamadas de sistema padrão e chamadas a procedimentos remotos (RPC's). Além disso, utiliza-se na implementação *sockets* e uma simulação de "processos leves" (threads) para a comunicação entre os objetos. Com isso, procurou-se tornar o protótipo o mais portável possível para outros ambientes dos quais a corrente implementação possa vir a fazer parte no futuro. O próximo passo será transportar o modelo implementado para utilizar as facilidades de uma das plataformas citadas acima.

Estão implementadas as funções básicas de um trader e aquelas que permitem com que uma federação seja formada. Por este motivo, nas seções seguintes, concentraremos a análise aos aspectos que foram alvo da implementação ressaltando ainda as soluções encontradas no âmbito de federação de traders. O modelo implementado atende às funcionalidades básicas da proposta geral.

Esquematicamente, o protótipo construído está representado na figura 4.5.

4.2.1 Comunicação – o "Broker"

O módulo de *simulação do trader* faz com que o cliente do trader (um importa-

Interfaces	Operações
Export Operations Interface	EXPORT, WITHDRAW, REPLACE.
Import Operations Interface	LIST_OFFER_DETAILS, SEARCH, SELECT.
Policy Controller Interface	STATUS_INQUIRY, EXPORTER_POLICY.
Context Management Interface	CREATE_CONTEXT, DELETE_CONTEXT, LIST_CONTEXT, LIST_CONTEXT_CONTENT, AUTHORIZE.
Federation Contract Interface	ESTABLISH_FEDERATION, DISTRIBUTE_CATALOGUE, REQUEST_CATALOGUE.
Federation Establishment Interface	EXCHANGE_CONTRACT.
Federated Interaction Interface	EXPORT, WITHDRAW, REPLACE, SEARCH, LIST_OFFER_DETAILS, SELECT.

Figura 4.4: Tabela das interfaces e respectivas operações.

dor ou exportador de serviços) tenha a impressão de que todas as suas operações estão sendo feitas localmente, tornando assim totalmente transparente todos os detalhes de comunicação (o que é especialmente adequado para o lado do cliente). Chamamos a parte da implementação que lida com a transparência para a comunicação via rede de "Broker"[4], um termo já consagrado pela CORBA[5] [VOGT93]. O *Broker* corresponde ao resultado da compilação das definições das interfaces do trader escritas em uma linguagem adequada (IDL[6], por exemplo).

Um cliente é construído utilizando-se uma biblioteca que esconde os detalhes de toda a comunicação com o trader remoto. Cada invocação de uma operação do trader é feita localmente ao *Broker* do lado do cliente que, por sua vez, comunica-se com o *Broker* do lado do trader via RPC passando os seguintes parâmetros:

[4] "Broker" é uma pessoa que faz um determinado serviço no lugar de outra pessoa.
[5] CORBA : Common Object Request Broker Architecture
[6] IDL : Interface Definition Language (CORBA)

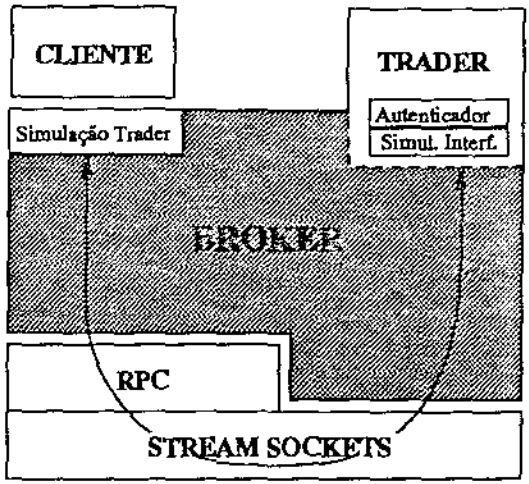


Figura 4.5: Ambiente construído juntamente com o trader.

- identificador da interface;
- identificador do objeto que está requisitando a função (cliente);
- lista de parâmetros adicionais (pode ser vazia);

que são fixos para todas as operações. Caso os parâmetros a serem transmitidos para o trader sejam muito complexos, então o *Broker* se encarrega de transmiti-los via rede através de *stream sockets*. Se não houver permissão para dado cliente efetuar a dita operação, os parâmetros complexos nem são enviados, o que melhora o desempenho nestes casos.

Está também embutido no *Broker* uma implementação básica do mecanismo de interfaces. Assim, ao chamar uma determinada função, é imprescindível que se especifique em qual interface aquela função está localizada. O *Broker* do lado do trader, antes de tudo, sempre verifica a existência de tal operação na interface especificada, e então tenta chamar localmente a função do trader correspondente. Se o cliente tiver autorização, a função é executada pelo trader.

O *Broker* é também responsável por retornar os resultados das operações. Aqui novamente, se os resultados forem complexos (no caso da operação "search", por exemplo), eles são transmitidos para o cliente também através de *stream sockets*.

4.2.2 Simulação do Mecanismo de Interfaces

Um objeto genérico no protótipo implementado é capaz de :

- criar e destruir interfaces dinamicamente;
- verificar a autorização de clientes para usarem as operações das interfaces;
- autorizar clientes a utilizarem operações nas interfaces;
- associar um determinado tipo de serviço a uma interface.

O trader cria interfaces dinamicamente no momento do estabelecimento do contrato de federação, verifica a autorização de clientes para usarem as operações das suas interfaces e associa um contrato de exportação (parte de um tipo de serviço) à interface de interações federadas. A capacidade de autorizar clientes é passada pelo trader ao seu administrador.

4.2.3 O Trader

Antes que qualquer operação seja executada, um módulo chamado "autenticador" verifica a autorização do cliente para executar a determinada operação. A partir das informações do nome (identificador) do cliente, nome da operação, nome do contexto sobre o qual a operação se dará e o tipo de serviço, o autenticador tem condições de determinar a viabilidade da execução da operação. Se não há permissão um código de erro apropriado é retornado, caso contrário, a operação é executada.

4.2.4 O Administrador

A comunicação entre o administrador e o trader se dá essencialmente da mesma forma que entre o cliente o trader. O *Broker* construído para o cliente é também utilizado para implementar a comunicação com o administrador, quando este usa um subconjunto das operações disponíveis aos clientes. Porém, existem diferenças importantes nos aspectos de comunicação que fazem com que o administrador seja mais que um cliente comum do trader. Na verdade, o administrador tem à sua disposição um conjunto maior de operações (operações de administração) e funciona, simultaneamente, como cliente e servidor do trader para conjuntos distintos de operações.

Em nossa implementação os administradores local e de federação estão unidos num mesmo objeto com uma interface que oferece (ao trader) as seguintes operações (figura 4.6) :

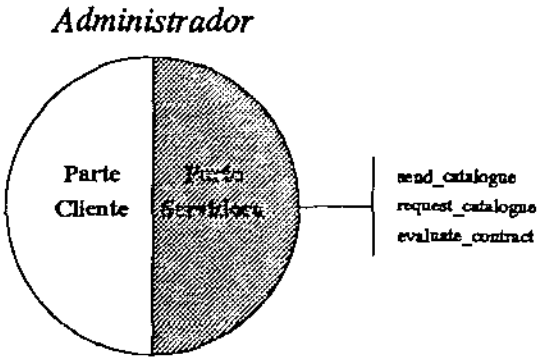


Figura 4.6: O administrador implementado.

- **send_catalogue :** através desta operação o trader pode informar ao administrador a respeito de um catálogo recebido.
- **request_catalogue :** pede ao administrador para elaborar um catálogo.
- **evaluate_contract :** pede ao administrador para avaliar uma proposta de contrato de federação. O Administrador pode responder : "aceito" ou "recusado".

O administrador possui, em nossa implementação, duas partes (figura 4.6): uma que oferece os serviços (descritos acima) ao trader e outra parte que desempenha unicamente o papel de cliente do trader. Esta última, além de utilizar operações como *LIST_OFFER_DETAILS*, *WITHDRAW*, etc. (que são comuns a um cliente qualquer), é também usuária das seguintes operações de gerenciamento fornecidas pelo trader :

- **Interface do Repositório de Tipos :**
  - **add_service_type :** adiciona um tipo de serviço ao repositório de tipos.
  - **display_types :** apresenta os tipos de serviços contidos no repositório de tipos.
- **Interface de Gerenciamento de Contexto :**
  - **create/delete_context :** cria/destrói determinado contexto no diretório de ofertas de serviços.

- **authorize** : determina o conjunto de interfaces, de operações e de contextos disponíveis a um determinado cliente.

• **Interface de Contrato de Federação :**

- **distribute_catalogue** : pede ao trader para anunciar o seu catálogo para um outro determinado trader.
- **request_catalogue** : pede ao trader para obter o catálogo de um outro determinado trader.
- **establish_federation** : pede ao trader para tentar o estabelecimento de um contrato de federação com outro trader fornecendo uma proposta de contrato.

Existe ainda um processo de inicialização do administrador e do trader que precisa ser feito antes de qualquer outra atividade. Neste processo, o administrador humano do trader informa a parte cliente do administrador a respeito da localização do trader que será administrado, bem como o seu nome. A localização do trader é repassada à parte servidora do administrador, ao mesmo tempo em que o nome do trader e a localização do administrador são transmitidas ao trader (figura 4.7). Como a localização do trader é o nome de um nó, este fato impõe a restrição de poder existir somente um trader por nó, o que não chega a ser um problema na grande maioria dos casos.

4.2.5 Federação de Traders

A comunicação entre dois traders para se construir uma federação ocorre da mesma forma descrita para a comunicação entre um trader e o seu cliente[7] (figura 4.8).

A Negociação e o Estabelecimento do Contrato

Para o estabelecimento do contrato, é necessária a intervenção do *Administrador de Federação* (veja figura 4.2), pois é dele que partem as decisões de formar a federação e de avaliar e propor contratos.

O esquema da figura 4.9 apresenta o cenário para a negociação e o estabelecimento de um contrato de federação entre dois traders, onde um faz o papel de "exportador" de serviços e o outro de "importador" de serviços.

[7]No processo de negociação do contrato de federação, somente se torna necessária a inclusão de uma nova operação ("Exchange Contracts") na comunicação entre os dois traders. Toda a troca de informações que precede o estabelecimento do contrato é feita utilizando-se as operações normais do trader.

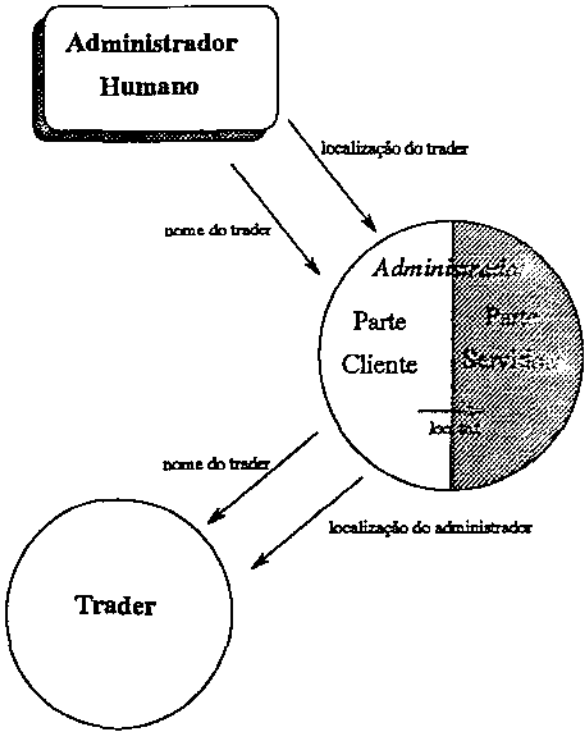


Figura 4.7: Processo de inicialização do trader e do administrador.

Antes de mais nada, para se estabelecer um contrato de federação entre dois traders é preciso que o trader que fará o papel de importador obtenha o *catálogo* do trader exportador. E isto pode ocorrer de duas maneiras.

No primeiro caso, o *administrador 1* (que é o administrador de federação do trader exportador na figura 4.9) decide enviar o seu catálogo para um potencial trader importador. O trader exportador ao receber do seu administrador a requisição para distribuir o catálogo (operação *distribute_catalogue*), juntamente com o próprio catálogo, age como um cliente qualquer do trader importador usando a operação "*export*" tendo o catálogo como propriedade do serviço, e o tipo padronizado "ESTABELECIMENTO DE FEDERAÇÃO" como tipo de serviço. Ao identificar este tipo de serviço padronizado, o trader importador informa ao seu administrador (operação *send_catalogue*) a respeito do recebimento do catálogo. O que é retornado ao *trader importador*, ao *trader exportador* e ao *administrador 1* é apenas um indicador de que o administrador do trader

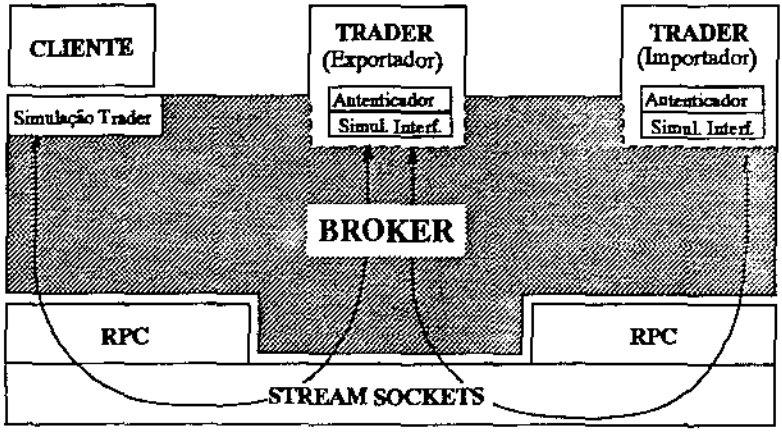


Figura 4.8: Ambiente expandido para federação.

importador está ciente (ou não) do catálogo enviado.

Como o trader exportador age como um cliente qualquer do trader importador, ele precisa ter um nome (identificador) que tenha autorização para utilizar os serviços do trader importador (neste caso, o serviço é a operação *export*).

Uma outra maneira de fazer com que o catálogo seja conhecido pelo administrador do trader importador, é quando o próprio administrador faz um pedido de catálogo a um potencial trader exportador (operação *request_catalogue*). Esta operação, no trader exportador, provoca a execução de um "*search*" com o tipo de serviço padronizado "ESTABELECIMENTO DE FEDERAÇÃO". O catálogo é então passado do *administrador 1* até o *administrador 2* (veja figura 4.9).

Uma vez que o administrador do trader importador possui o catálogo, ele pode agora iniciar a negociação do contrato de federação fazendo uma proposta ao administrador do trader exportador (operação *exchange_contract*), disparando assim o processo de estabelecimento de federação. O administrador do trader exportador pode aceitar a proposta plenamente, recusá-la ou recusá-la e ainda fazer uma contra-proposta com um contrato reduzido. Esta contra-proposta pode ser feita utilizando-se a operação *distribute_catalogue*. Deste modo um contrato de federação pode ser negociado e estabelecido, e as operações federadas podem ser então executadas.

Neste processo definiram-se operações para a comunicação entre o administrador e o trader. Elas estão apresentadas na tabela da figura 4.10.

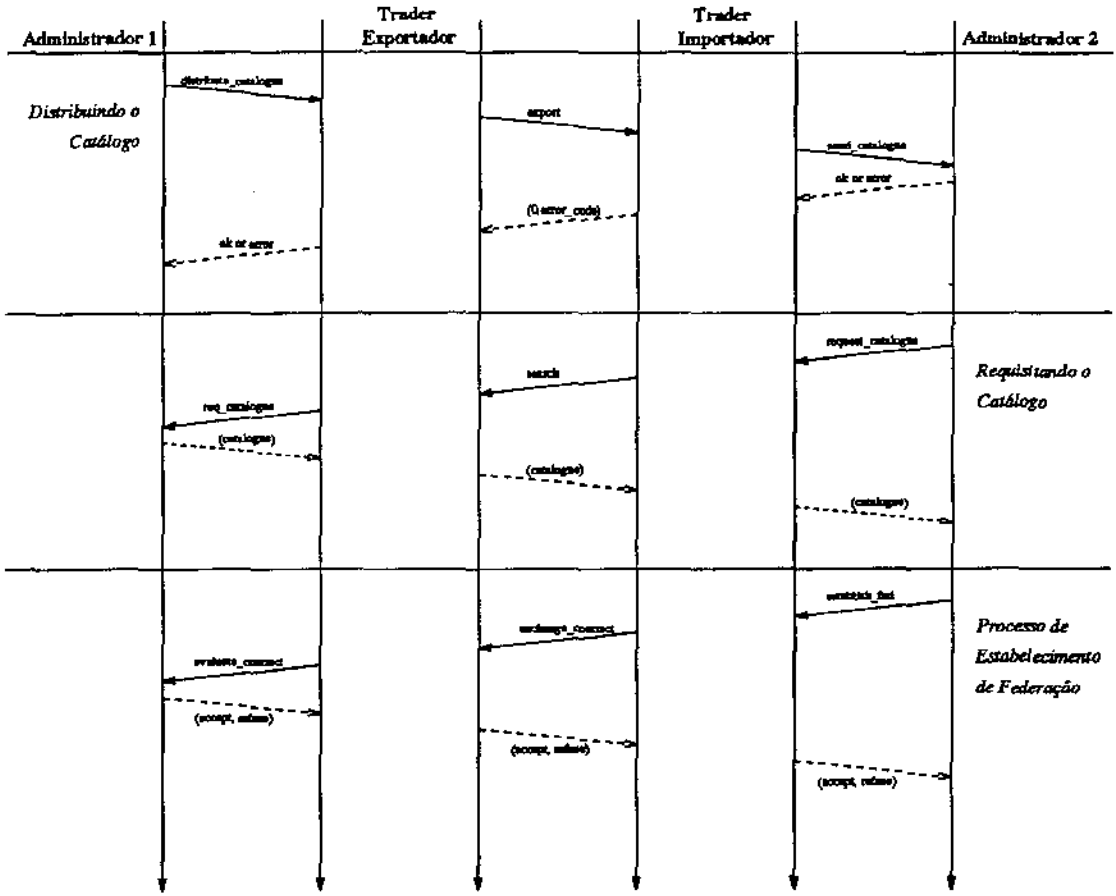


Figura 4.9: Cenário para a negociação de contratos de federação.

Operação	Parâmetros	Valores de Retorno
distribute_catalogue	Localização do trader imp.; Catálogo	OK ou erro
send_catalogue	Catálogo; Id. da interf. de estab. de federação	OK ou erro
request_catalogue	Localização do trader exportador	Catálogo (Possivelmente vazio)
req_catalogue		Catálogo
establish_fed	Id. da Interface de Est. de Fed.; Localização do trader exp.; Proposta de contrato de federação; Contexto local para manter contrato de importação	Aceita ou recusada
evaluate_contract	Proposta de contrato de Federação	Aceita ou recusada

Figura 4.10: Operações para comunicação Administrador-Trader.

Operações Federadas

O módulo "*Executivo de Federação*" (veja figura 4.2) do trader exportador, ao receber a requisição de qualquer operação, identifica o contrato de exportação que está associado à interface de *interações federadas* que foi criada no momento do estabelecimento do contrato e na qual se requisita a dada operação. Depois de identificado o contrato de exportação correspondente, aplicam-se as restrições ali contidas. Para tanto, o *executivo de federação* precisa das seguintes informações:

- identificador do trader importador;
- nome da operação requisitada;
- contexto no qual se dará a dada operação e
- tipo de serviço.

Se as restrições forem satisfeitas, a execução propriamente dita é passada ao módulo "*Executivo*" que faz o procedimento usual. Depois disto, o próprio *executivo de federação* se encarrega de retornar os valores resultantes da operação (transformados, se necessário) ao trader importador.

No lado do trader importador, o módulo *Executivo de Federação* é acionado pelo módulo *Executivo* quando este identificar a necessidade de alguma operação federada. Por exemplo, no caso da operação *search*, se o trader não puder satisfazer aos critérios requeridos pelo cliente localmente e houver um contrato com algum outro trader dentro do escopo de busca, o *executivo* aciona o *executivo de federação* para que ele faça um *search* federado.

De um modo geral, o papel do *executivo de federação* do trader importador é transformar uma requisição local em uma operação federada que é passada ao trader exportador. Também, as operações executadas via federação passam pelo módulo *Executivo de Federação* que encarrega-se de interpretar o contrato de exportação aplicando as restrições e operações combinadas.

Os algoritmos das operações que formam o conjunto mínimo a ser fornecido aos clientes de um trader (importador, exportador ou outro trader) estão comentados a seguir.

4.2.6 Operações Disponíveis na Interface "Exporter Operations"

As operações disponíveis nesta interface são :

- **EXPORT** : corresponde à criação de uma oferta de serviço no diretório do trader;
- **WITHDRAW** : remove uma oferta de serviço do diretório;
- **MODIFY** : corresponde à uma operação de *withdraw* seguida de uma operação de *export* feitas atomicamente.

As operações *withdraw* e *modify* não sofrem modificações para incorporar os mecanismos de federação propostos neste trabalho, por isso não discutiremos aqui os seus algoritmos.

A Operação "EXPORT"

Inicialmente, o trader que recebeu uma solicitação da operação de *export* aciona o autenticador que verifica a permissão do determinado cliente para executar esta operação. Se não há permissão ou se a interface através da qual o cliente fez a requisição não possuir a operação *export*, então um erro apropriado é retornado e nada mais é feito.

Depois disto, o trader analisa o tipo de serviço que está sendo exportado. Se for do tipo padrão de *Estabelecimento de Federação*, então informa ao seu

administrador a respeito do recebimento de um catálogo de um outro trader que desempenhará o papel de exportador. O catálogo é passado ao administrador através da operação *send_catalogue* (seção 4.2.4). Isto disparará no administrador um processo de estabelecimento da federação. O valor retornado pela operação *export*, neste caso, será apenas para indicar que o administrador está ou não informado a respeito do catálogo do trader exportador.

Se o tipo de serviço for qualquer outro, trata-se de uma exportação local comum. Então o trader cria em sua base de dados uma oferta de serviço a partir dos parâmetros recebidos do cliente, retornando um identificador da oferta de serviço que é único num dado contexto. Se o nome do contexto não for especificado, um contexto *default* é assumido.

Um código de erro será retornado se a operação não puder ser executada por algum motivo (se não há permissão, se o contexto não está registrado, etc.).

4.2.7 Operações Disponíveis na Interface "Importer Operations"

As operações disponíveis nesta interface são :

- **LIST_OFFER_DETAILS** : retorna os detalhes de uma determinada oferta de serviço;
- **SEARCH** : retorna todas as ofertas de serviços que satisfazem os requisitos do importador;
- **SELECT** : seleciona a melhor oferta de serviço (a partir de critérios determinados) dentre as que satisfazem às propriedades requeridas pelo importador.

Aqui o algoritmo da operação "*search*" é o único alterado para permitir o estabelecimento de federação e as operações federadas.

A Operação "SEARCH"

Após verificada a permissão do cliente para efetuar a operação, o tipo de serviço pedido é analisado. Se for do tipo padrão de *Estabelecimento de Federação*, o trader pede ao seu administrador que crie um catálogo com as possíveis operações via federação e restrições de acesso. Isto é feito através da operação *request_catalogue* (seção 4.2.4). Então retorna o catálogo ao cliente (que é um outro trader neste caso) juntamente com o identificador para a sua interface de estabelecimento de federação, se a operação for bem-sucedida.

Se o tipo de serviço pedido não for de estabelecimento de federação, o trader faz a busca dentro de um escopo adequado (lista de contextos) para obter as ofertas de serviços que satisfazem aos critérios pedidos pelo cliente. É neste momento que são obtidas tanto as informações estáticas quanto as dinâmicas das ofertas de serviços.

Se não acha ofertas de serviços adequadas localmente, verifica os seus contratos contidos nos contextos do escopo de busca para fazer uma requisição de busca ao *Executivo de Federação* que se encarregará de executar o mesmo "search" no trader remoto.

Caso o cliente seja um outro trader (ou seja, trata-se de um *search remoto* feito através do *Executivo de Federação*), o respectivo contrato de exportação é analisado para impor as restrições de busca lá contidas. Neste caso, se não houver ainda ofertas que se adequam às exigências do cliente, e o contrato especificar *downstream linking*, o trader pode passar a requisição adiante para outros traders da federação.

É retornada ao cliente uma lista de ofertas de serviços com as propriedades pedidas juntamente com os identificadores das interfaces onde os serviços estão disponíveis. Se nenhum critério de igualdade é especificado pelo cliente, todas as ofertas do tipo pedido são retornadas.

4.2.8 A Operação "EXCHANGE_CONTRACT"

Esta operação está disponível na *interface de estabelecimento de federação*. Através desta operação, um trader importador poderá fazer uma proposta de um contrato para a federação. O trader exportador, ao receber esta proposta, pede ao seu administrador para avaliá-la (operação *evaluate_contract*) após verificar permissões e interfaces. Se a proposta de contrato for aceita pelo administrador, o trader cria uma nova interface para interações federadas e exporta para a sua própria base de dados o tipo de serviço padronizado "INTERAÇÃO DE NEGOCIAÇÕES FEDERADAS" tendo o contrato de exportação como sendo a propriedade do serviço. O identificador da interface é então retornado ao trader importador se não houver problemas. Caso contrário, um código de erro é retornado.

Uma resposta positiva do trader exportador, faz com o trader importador exporte para a sua própria base de dados, no contexto determinado, a oferta de serviço do tipo padrão "INTERAÇÕES DE NEGOCIAÇÕES FEDERADAS" tendo o contrato de importação como propriedade do serviço.

Deste modo, o diretório do trader está dividido em contextos que podem conter tanto ofertas de serviços comuns quanto contratos de importação para acessar ofertas de serviços em traders remotos.

4.2.9 Outras Operações Implementadas

As operações "*withdraw*" e "*list_offer_details*" também foram implementadas seguindo o modelo da figura 4.2. Ao ser requisitada uma operação *withdraw* tendo um identificador de oferta de serviço como parâmetro, o *autenticador* verifica a permissão do cliente para tal operação e o *executivo* pede ao módulo de *informações estáticas* para remover a oferta especificada do *diretório*, se houver autorização. Assim, a oferta de serviço é marcada como "deletada".

A operação "*list_offer_details*" obtém as propriedades da oferta de serviço requisitada do módulo de *informações estáticas* e, então, pode obter as informações dinâmicas (se é que há um identificador da interface do controlador de política na informação estática da oferta encontrada anteriormente) retornando os valores encontrados ao cliente que fez o pedido.

A tabela de todas as operações implementadas com respectivos clientes e servidores está representada na figura 4.11. A representação em IDL das interfaces e operações do trader implementado estão no apêndice A.

Operações	Trader	Administrador	Cliente
search	S		C
export	S		C
withdraw	S	C	C
list_offer_details	S	C	C
add_serv_type	S	C	
display_repository	S	C	
authorize	S	C	
create_context	S	C	
distr_catalogue	S	C	
request_catalogue	S	C	
establish_fed	S	C	
exchange_contract	S	C	
send_catalogue	C	S	
req_catalogue	C	S	
evaluate_contract	C	S	

Figura 4.11: Operações Implementadas. C = cliente; S = servidor.

O protótipo implementado a partir destes algoritmos segue o modelo proposto na figura 4.2 e está incluído no projeto da plataforma Multiware [MEND94].

4.3 A Plataforma Multiware

As plataformas hoje disponíveis estão sendo expandidas para dar suporte ao processamento distribuído aberto de informação multimídia. A plataforma Multiware, em desenvolvimento na UNICAMP, adota o modelo de referência para ODP da ISO/ITU-T como modelo funcional e incorpora os conceitos de produtos já existentes no mercado (ANSAware, DCE-OSF, CORBA-OMG, etc.).

A arquitetura básica da plataforma (figura 4.12) agrega idéias defendidas na literatura e procura adotar produtos já existentes no mercado.

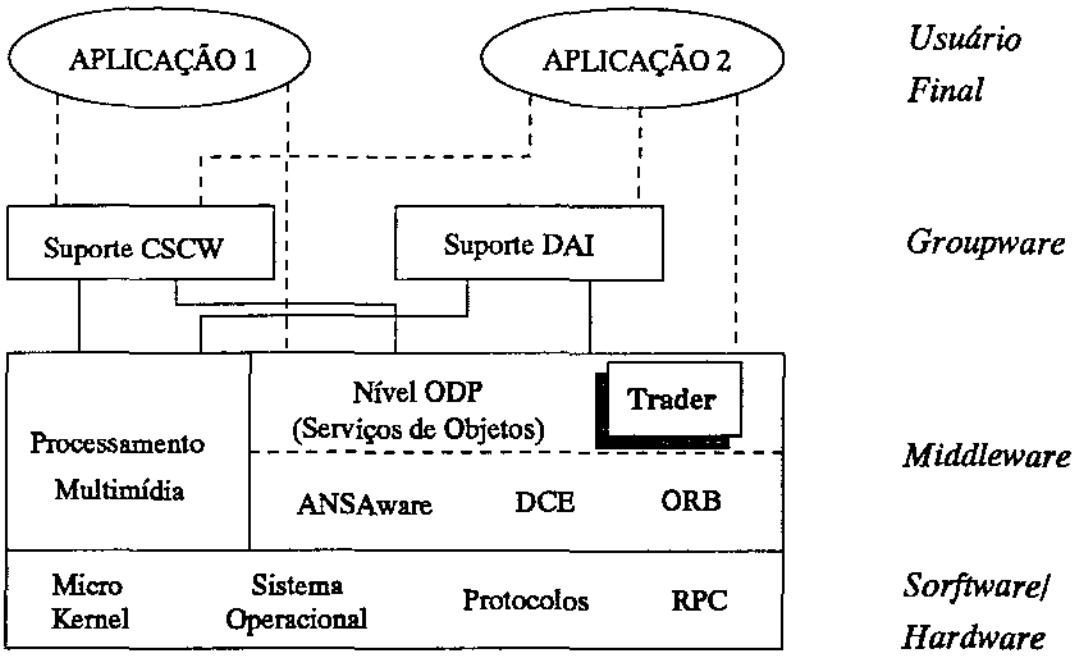


Figura 4.12: Estrutura da plataforma Multiware.

A camada básica de *Software/Hardware* é composta de um sistema operacional (construído eventualmente sobre um *microkernel*) e protocolos de comunicação.

A camada *Groupware* fornece as funcionalidades requeridas por diferentes classes de aplicações, como CSCW, *Distributed Artificial Inteligence (DAI)*, entre outras. Os serviços típicos oferecidos por esta camada são gerenciamento de diálogo, protocolos de interação e manipulação de documentos multimídia. Atualmente, apenas o suporte a CSCW está sendo considerado.

A camada *Middleware* é responsável por fornecer facilidades de processamento distribuído à camada *Groupware* e às aplicações. Esta camada está dividida em duas sub-camadas:

- *Sub-camada de Processamento Multimídia :* permite a troca de informações multimídia em tempo real com uma determinada qualidade de serviço;
- *Sub-camada ODP :* que é composta de dois níveis :
  - Sistemas Distribuídos Comerciais (como ORB, DCE e ANSAware);
  - Nível ODP : agrega as funcionalidades ODP aos sistemas distribuídos comerciais.

Logicamente, neste projeto, o trader encontra-se localizado na camada *middleware*, no nível ODP e é responsável por prover as funções de *trading* às camadas superiores. Neste nível ainda existem outras funções, como por exemplo, suporte a grupos e suporte à transação, entre outras, que, juntamente com o trader, procuram prover o ambiente de processamento distribuído aberto às aplicações, independendo da infraestrutura utilizada pela sub-camada ODP (Serviços de Objetos).

Inicialmente, o trader foi implementado diretamente sobre a camada *Software/Hardware*, sendo as funções intermediárias necessárias (*threads*, simulação do mecanismo de interfaces, etc.) implementadas no "Broker" descrito na seção 4.2.1. Com esta abordagem foi assumido que o trader será, a princípio, colocado sobre a plataforma ORB. Por este motivo também a definição das interfaces do trader está em OMG-IDL no apêndice A. Neste trabalho, as funções de repositório e de armazenamento foram implementadas e incorporadas ao próprio trader.

Futuramente, o trader será portado para o ambiente da figura 4.12 para utilizar as facilidades das plataformas construídas. Para fazer isto, é necessário que o *Broker* construído no protótipo da figura 4.5 seja substituído por um outro *broker* que é o resultado da compilação da definição da interface em uma linguagem apropriada. Assim, para transpor o protótipo implementado para a plataforma Multiware (que utilizará inicialmente ORB), deve-se substituir o *broker* pelo resultado da compilação da definição da interface apresentada no Apêndice A. Este

novo broker será composto de *stubs* que chamam o núcleo da ORB gerenciando toda a parte de comunicação entre clientes, traders e administradores.

4.4 Trabalhos Relacionados

Analisaremos brevemente duas outras implementações de trader – o trader da ANSA [ANSAMAN] e o trader da plataforma Y [TSCHTRD] – fazendo uma comparação entre eles e o protótipo implementado neste trabalho. Consideramos estas duas plataformas como representantes dos diversos esforços no desenvolvimento de plataformas de serviços abertos que possuem traders.

Primeiramente, é importante ressaltar que os três traders (ANSAware, Y e o protótipo) apresentam a funcionalidade básica requerida de um trader, sendo que o último, obviamente, é o menos sofisticado dos três. No entanto, no que tange os aspectos de formação e gerenciamento de federação, o protótipo construído é o que possui o mecanismo mais elaborado (pelo menos, em comparação com as versões do trader da ANSA e do trader da plataforma Y que analisamos na literatura).

O modelo ODP herdou muitos conceitos da plataforma ANSAware, inclusive o trader. Por este motivo, o nosso trader (que segue os princípios do modelo ODP) possui muitas semelhanças com o trader da ANSA. Algumas diferenças são :

- Os contextos estão organizados em hierarquias (como diretórios, em sistemas de arquivos) e as importações das ofertas de serviços podem ser feitas através do "*path*" juntamente com o identificador da oferta, ao invés de somente pelos atributos do serviço. Esta possibilidade inexiste em nosso modelo.

- O administrador inexiste no ambiente ANSA como entidade separada do trader, o que impõe a restrição de haver um único trader por sistema ANSA, uma vez que não poderia existir um único administrador para vários traders dentro de um mesmo domínio.

- O mecanismo para a formação de federação de traders existe no ambiente ANSA, mas é consideravelmente mais simples. Existem basicamente, dois meios de se estabelecer federações : acoplando o espaço de contextos de um trader a um determinado contexto em outro trader, ou usando "*proxy offers*", que são versões simplificadas de contratos de importação. Além disso, a ANSAware forma federações de traders em uma hierarquia de dois

níveis : um é o "trader mestre" e todos os outros acoplam um determinado contexto local a este "trader mestre". Em nosso modelo, uma configuração arbitrária de traders é permitida.

O trader da plataforma Y também possui o mesmo objetivo do trader do protótipo implementado, porém não segue a padronização ODP. Algumas diferenças com o modelo proposto são :

- As importações sempre retornam somente a melhor oferta de serviço. Isto exige que o trader verifique a disponibilidade do servidor em cada importação, pois, caso contrário, surgiria o seguinte problema : se o cliente tentar fazer o binding com o servidor cujo endereço foi obtido da oferta de serviço retornada pelo trader, e encontrar um erro, o cliente não teria como requisitar ao trader um outra opção que ainda satisfizesse seu pedido, mas que não fosse a "melhor".

- Grande preocupação com questões de performance, com a existência de memória *cache* para armazenar as últimas ofertas encontradas e outras estratégias para reduzir o tempo da importação.

- Na versão analisada, ainda não há qualquer mecanismo para federação.

Capítulo 5

Conclusão

Vantagens de Um Ambiente Para Processamento Distribuído Aberto

Devido aos avanços na tecnologia de comunicação com altas taxas de transmissão, taxas de erros reduzidas e segurança melhorada, devido às futuras (e presentes) necessidades dos usuários como integração de ilhas de automação, trabalho cooperativo (CSCW) e arquiteturas para serviços distribuídos abertos e também devido ao progresso na padronização (RM-ODP), um ambiente para processamento distribuído aberto se torna, não somente possível, mas principalmente imprescindível para atender às novas demandas dos usuários.

O ambiente aberto tem a vantagem de ser ilimitado e livre para admitir todos os tipos de usuários, componentes ou aplicações. Como conseqência dessas características temos heterogeneidade e descentralização onde a autonomia deve ser uma garantia para cada componente ou ambiente local.

Vantagens do Trader no Ambiente Aberto

O **Trader** é um componente chave em um ambiente distribuído aberto. Ele fornece um serviço de diretório que permite com que serviços sejam descobertos em tempo de execução através de uma busca baseada nos atributos deste serviço. Ou seja, o propósito da função de trading é manter o conhecimento dos serviços disponíveis correntemente, encontrar ofertas de serviços que satisfaçam requisitos de clientes e selecionar a melhor oferta de serviço para os clientes. Em um Sistema Distribuído Aberto é altamente desejável que existam meios de fazer a seleção dinâmica de serviços computacionais que satisfaçam a determinadas pro-

priedades. E é através do trader que importadores de serviços (clientes) podem entrar em contato com exportadores de serviços (servidores).

A união de traders em federações permite com que sistemas distribuídos distintos trabalhem juntos, mas, ao mesmo tempo, mantenham controle dos seus próprios domínios. Há empresas, departamentos e domínios separados que podem usar diferentes tecnologias de modos diferentes. A federação permite com que sistemas diferentes cooperem-se mutuamente.

Vantagens do Modelo Proposto

O presente trabalho propôs um modelo de implementação do trader dando atenção especial aos aspectos que dizem respeito à **Federação de Traders** apresentando-se algoritmos e detalhes da implementação do protótipo construído. Este modelo apresenta as seguintes vantagens :

- segue o padrão estabelecido pela ISO para a função de trading;
- é adequado para um Ambiente de Processamento Distribuído Aberto, onde autonomia, descentralização e encapsulamento são princípios fundamentais;
- clareza na divisão e especificação dos módulos que compõem o trader, o que leva a uma maior facilidade na implementação;
- definição do protocolo de comunicação entre administradores e traders no processo de estabelecimento de federações;
- um administrador pode gerenciar vários traders em nós distintos (mas dentro de um certo domínio);

Conclusão de Implementação

O protótipo implementado, apesar das suas limitações (daí o nome "protótipo"), trouxe à tona uma série de aspectos de implementação e modelagem que se tornaram absolutamente úteis no processo de elaboração da dissertação. Além disso, ele foi fundamental para o compreensão dos problemas e necessidades de um ambiente aberto de serviços.

A maior dificuldade encontrada na fase de implementação foi com relação à parte de comunicação entre objetos através da rede. Foi necessária a construção de uma camada (o "*BROKER*", figura 4.1) que tornasse os aspectos de comunicação transparentes para o trader e especialmente para o cliente do trader. Com

o futuro transporte do protótipo do trader para o ambiente *Multiware* [MEND94], a camada BROKER (aqui, de uso específico do trader) será substituída por outra camada que fornecerá todas as facilidades de comunicação necessárias para atender não só ao trader, mas também a um objeto computacional genérico.

O Protocolo de Comunicação Administrador-Trader

Todo o protocolo entre o trader e os seus administradores foi definido, especialmente no caso de negociação dos contratos de federação. Definiu-se as operações que foram colocadas à disposição do administrador para que ele pudesse se encarregar de toda tomada de decisão relativa ao trader.

Trabalhos Futuros

Existe uma série de modificações que podem ser feitas no protótipo para melhorar questões de eficiência, segurança, clareza, modularidade e para corresponder plenamente ao modelo proposto pela ISO.

- Eliminar o uso de *stream sockets* na passagem de parâmetros. À medida que a implementação prosseguia percebeu-se que o mecanismo de comunicação construído com sockets poderia ser substituído (com vantagens para clareza e performance) pelo mecanismo de RPC;
- O esquema de segurança não é adequado, pois as restrições de acesso são baseadas apenas em um "nome" do agente requisitante. Deveriam haver mais informações e/ou confirmações sobre a identidade dos clientes, ou seja, serviço de autenticação.
- O diretório onde são armazenadas as ofertas de serviços e os contratos deveria ser substituído pelo X500 da ITU-T, por exemplo, onde os dados estariam armazenados em memória não-volátil, o que não corresponde ao protótipo atual (há perda na performance, mas ganho em segurança).
- O modelo de contextos implementado não leva em consideração o relacionamento entre os contextos. Isto deveria ser feito criando e gerenciando grafos de relacionamentos entre os contextos.
- O repositório de tipos construído deveria verificar a validade dos tipos de serviços requisitados, comparar tipos de serviços e determinar relações entre tipos e sub-tipos.

- A construção de um "administrador inteligente" que automatizasse algumas tomadas de decisão poupando o administrador humano de atividades redundantes ou padronizadas.

- Integrar o trader implementado à plataforma *Multiware* de forma a utilizar os recursos que as camadas inferiores oferecem e de forma a prover serviços de trading ao ambiente distribuído aberto.

- Melhorar a performance nas importações de serviços, utilizando-se, por exemplo, uma memória *cache* para armazenas as últimas ofertas (e/ou as mais requisitadas).

Apêndice A

As Interfaces do Trader em OMG IDL

A.1 Tipos Básicos Para os Parâmetros

A seguir estão definidos os tipos de parâmetros que são utilizados nas interfaces do trader. Os tipos de parâmetros "Interface_Id", "Identifier_T" e "Name_Type" são tipos básicos e para este apêndice, são considerados pré-definidos.

```
interface BasicTypes {

    typedef sequence <Name_Type>  Name_List;

    typedef string                Searching_Constraint;

    typedef Name_Type             Service_Type_Name;

    typedef string                Matching_Criteria;


    // --- Servico Exportado ---

    typedef struct ExpServItem {
        Name_Type prop_type;
        string    prop_value;
    };
```

```
typedef sequence <ExpServItem> Exported_Service;


// --- Propriedades de Servico ---

typedef struct Service_Property {
     Name_Type prop_type;
     Name_List prop_value_list;
};

typedef sequence <Service_Property> Serv_Prop_List;


// --- Tipo de Servico ---
// Descrito pelo id. do tipo de servico.

typedef struct Service_Type {
     Name_Type serv_type_name;
     Serv_Prop_List sp_list;
};


// --- Contrato ---

typedef Serv_Prop_List Contract;


// --- Codigos de Erro ---

typedef enum Error {
     OKAY,
     service_offer_not_found,
     context_not_registered,
     no_authority,
     do_not_interest,
     cannot_establish_contract,
     offer_does_not_exist
};
```

```
    // --- Excessoes ---

    typedef SecurityException {
        NO_AUTHORITY
    };
    exception SECURITY_EXCEPTIONS {SecurityException};

};
```

A.2 Import Operations Interface

Aqui são descritas as operações SEARCH e LIST_OFFER_DETAILS juntamente com os tipos de dados específicos de cada operação.

```
interface ImportOperationsInterface : BasicTypes {

    // --- Excessoes ---

    typedef enum SearchExceptions {
        service_type_not_registered,
        invalid_return_service_property_names,
        invalid_return_service_offer_property_names,
        invalid_matching_service_property_names,
        invalid_matching_constraint
    };
    exception SEARCH_EXCEPTIONS {SearchExceptions};

    typedef enum ListOfferDetailsExceptions {};
    exception LIST_OFFER_DETAILS_EXCEPTIONS {ListOfferDetailsExceptions};


    // --- Estruturas de dados resultantes das operacoes ---

    typedef struct SearchResultItem {
        Interface_Id     if_id;
        Exported_Service serv_prop_values;
        Exported_Service serv_offer_prop_values;
    };
```

```
    typedef sequence <SearchResultItem> SearchResult;

    typedef struct ListResult {
        Service_Type_Name stn;
        Exported_Service serv_prop_values;
        Exported_Service serv_offer_prop_values;
        Interface_Id if_id;
    };


    // SEARCH

    SearchResult search {
        in Identifier_T         client_id,
        in Service_Type_Name    serv_type,
        in Matching_Criteria    m_criteria,
        in Searching_Constraint s_constraint,
        in Name_List            serv_prop_names,
        in Name_List            serv_offer_prop_names,
        out Error               error
    } raises (SECUTIRY_EXCEPTIONS,SEARCH_EXCEPTIONS);


    // LIST_OFFER_DETAILS

    ListResult list_offer_details {
        in Identifier_T client_id,
        in Name_Type    context_name,
        in Identifier_T offer_id,
        in Name_List    serv_prop_names,
        in Name_List    serv_offer_prop_names,
        out Error error
    } raises (SECURITY_EXCEPTIONS,LIST_OFFER_DETAILS_EXCEPTIONS);

};
```

A.3 Export Operations Interface

Aqui são descritas as operações EXPORT e WITHDRAW juntamente com os tipos de dados específicos de cada operação.

```
interface ExportOperationsInterface : BasicTypes {

    // --- Excessoes ---

    typedef enum ExportExceptions {
        service_type_not_allowed,
        service_offer_already_exported,
        invalid_exported_service,
        service_type_contraint_failure
    };
    exception EXPORT_EXCEPTIONS {ExportExceptions};

    typedef enum WithdrawExceptions {
        service_offer_not_exported
    };
    exceptions WITHDRAW_EXCEPTIONS {WithdrawExceptions};

    // EXPORT
    Identifier_T export {
        in Identifier_T      client_id,
        in Name_Type         context_name,
        in Service_Type_Name serv_type_name,
        in Interface_Id      service_if_id,
        in Exported_Service  exp_service,
        in Interface_Id      policy_controller_if_id,
        in Exported_Service  serv_offer_prop_values,
        out Error            error
    } raises (SECURITY_EXCEPTIONS,EXPORT_EXCEPTIONS);

    // WITHDRAW
    Error withdraw {
        in Identifier_T client_id,
        in Name_Type    context_name,
        in Identifier_T offer_id
```

```
    } raises (SECURITY_EXCEPTIONS,WITHDRAW_EXCEPTIONS);

};
```

A.4 Federation Establishment Interface

Nesta interface a única operação presente é EXCHANGE_CONTRACTS.

```
interface FederationEstablishmentInterface : BasicTypes {

    // --- Excessoes ---

    typedef ExcContrExce {
        invalid_contract
    };
    exception EXC_CONTR_EXCEPTIONS {ExcContrExce};


    // --- Estrutura de dados resultante ---

    typedef enum ExcContractResult {
        ACCEPTED,
        REFUSED
    };


    // EXCHANGE_CONTRACT

    ExcContractResult exchange_contract {
        in Contract fed_contract,
        in Security security_info
    } raises (SECURITY_EXCEPTIONS,EXC_CONTR_EXCEPTIONS);

};
```

A.5 Type Repository Interface

Em nossa implementação, esta interface foi incorporada ao trader, pois o repositório de tipos está imbutido no trader.

```
interface TypeRepositoryInterface : BasicTypes {

     typedef sequence <Service_Type> Serv_Type_List;

     // ADD_SERVICE_TYPE

     void add_service_type {
          in Service_Type stype;
     } raises (SECURITY_EXCEPTIONS);


     // DISPLAY_REPOSITORY

     Serv_Type_List display_repository {
     } raises (SECURITY_EXCEPTIONS);

};
```

A.6 Context Management Interface

```
interface ContextManagementInterface : BasicTypes {

     // CREATE_CONTEXT

     void create_context {
          in Name_Type context_name;
          in Name_Type super_context;
     };


     // AUTHORIZE

     void authorize {
          in Identifier_T client_id;
          in Interface_Id interface_id;
          in Name_Type context_name;
     };
```

```
};
```

A.7 Federation Contract Interface

```
interface FederationContractInterface : BasicTypes {

     // --- Estrutura de dados resultante ---
     typedef enum EstFedRes {
          ACCEPT,
          REFUSE
     };

     // DISTR_CATALOGUE

     int distr_catalogue {
          in string imp_trader_location;
          in Contract catalogue;
     } raises (SECURITY_EXCEPTIONS);


     // REQUEST_CATALOGUE

     int request_catalogue {
          in string exp_trader_location;
     } raises (SECURITY_EXCEPTIONS);


     // ESTABLISH_FED

     ExcContrResult establish_fed {
          in Interface_Id fed_estab_if_id;
          in string exp_trader_location;
          in Contract fed_contr_proposal;
     } raises (SECURITY_EXCEPTIONS);

};
```

Bibliografia

[ANSA89] "ANSA : An Engineer's Introduction to the Architecture" – Release TR.03.02 – Architecture Projects Management Limited – Nov. 1989

[ANSA91] "ANSA Work-Programme – Phase III" – Issue 1 – Architecture Projects Management Limited – June 1991

[ANSA93] "The Open Systems Newsletter – ANSA : Object Pioneers" - Technology Appraisals Ltd – Volume 7 - Issue 2 - Feb. 1993

[ANSAMAN] ANSA, ANSA Reference Manual, APM Ltd., 24 Hills Road, Cambridge, CB2IJP, UK, March 1989

[BEAR92] *Bearman, Mirion; Raymond, Kerry* "Federating Traders: An ODP adventure", IFIP 1992, pages 125-141.

[BEAR93] *Bearman, M.* "ODP Trader" - Proceedings of the ICODP - sept. 1993 - pp 19-23

[BEIT94] *Beitz, A.; Bearman, M.* "An ODP Trading Service for DCE" - Proceedings of The International Workshop on Services in Distributed and Networked Environments (SDNE) - IEEE - Prague - Czech Replubic - June 1994

[BIGGS94] *Biggs, C.; Brookes, W.; Indulska, J.* "Enhancing Interoperability of DCE Applications : a Type Management Approach" - Proceedings of The International Workshop on Services in Distributed and Networked Environments (SDNE) - IEEE - Prague - Czech Replubic - June 1994

[BLAIR92] *Blair, Gordon S.; Rodden, Tom* "The Impact of CSCW on open distributed Processing", IFIP 1992, pages 143-153.

[BLAIR93] *Blair, G.; Rodden, T.* "The Challenges of CSCW for Open Distributed Processing" - Proceedings of the ICODP - sept. 1993 - pp 99-116

[BOSCO93] *Bosco, P. G. et al* "A Distributed Object Oriented Platform based on DCE and C++" - Proceedings of the ICODP - sept. 1993 - pp 176-187

[CARD93] *Ferri, C. S.; Cardoso, E.* "A Fully Distributed Name Server for Reliable Distributed Applications" - Anais 11o. Simpósio Brasileiro de Redes de Computadores - 10 a 13 maio - 1993

[CELES93] *Celestino, J.; Claude, J. P.* "Executivo - Um Suporte de Processamento Distribuído para Gerenciamento de Redes de Telecomunicações" - Anais 11o. Simpósio Brasileiro de Redes de Computadores - 10 a 13 maio - 1993

[CHEN94] *Cheng, W. C. H; Jia, X.* "A Reliable Trading Service in Open Distributed Systems" - Proceedings of The International Workshop on Services in Distributed and Networked Environments (SDNE) - IEEE - Prague - Czech Replubic - June 1994

[CHOR1] *Rozier, M. et al* "Overview of CHORUS Distributed Operating Systems", chorus systems, april 1990.

[CHOR2] *Armand, Francois, et al* Revolution 89 or "Distributed Unix brings it back to its original virtues", chorus systems, august 1989.

[COOL1] *Habert, Sabine; Abrossimov, Vladim and Mosseri, Laurence* "COOL: Kernel suport for object oriented environemnts".

[COUL92] *Coulson, G.* "Extensions to ANSA for Multimedia Computing" - Computing Networks and ISDN Systems - 25,3 (Sept. 92)

[DOMV92] *Domville, I.* "The Distributed Application Environment- An architecture Based on Enterprise Requirements", IFIP 1992, pages 191-202.

[DREO92] *Dreo, G.; Heverhagen, U.; Leischmer, M.* "Using the OSI management information model for ODP", IFIP 1992, pages 203-214.

[DRUM93] *Drummond, R.; Gonçalves, C.; Teles, A. P.* "Aspectos da Implementação de Objetos Distribuídos" - Anais 11o. Simpósio Brasileiro de Redes de Computadores - 10 a 13 maio - 1993

[ELLIS91] *Ellis, C. A.; Gibbs, S. J., Rein, G.L.* "Groupware : Some Issues and Experiences" - Communications of the ACM - Jan 1991 - Vol. 34, no.1

[GANT91] *Gantenbein, R. E. J.* "Dynamic Binding in Strongly Typed Programming Languages" - System Software - 14,1 (jan. 91)

[GIM94] *Gimenez, G et al* "Ecology in Global Distributed Systems" - Proceedings of The International Workshop on Services in Distributed and Networked Environments (SDNE) - IEEE - Prague - Czech Replubic - June 1994

[GOSCI93] *Goscinski, A.; Ni, Y.* "Object Trading in Open Systems" - Proceedings of the ICODP - sept. 1993 - pp 117-128

[HEB92] *Hebert, Andrew* "The Challenge of ODP", IFIP 1992, pages 15-27.

[HEINE93] *Popien, C; Heineken, M.* "Trading Enhancement by Service Combination in ODP" - Proceedings of the ICODP - sept. 1993 - pp 312-323

[INDU93] *Indulska, J. et al* "A Type Management System for an ODP Trader" - Proceedings of the ICODP - sept. 1993 - pp 141-152

[ISO7053] "ISO/IEC JTC 1/SC 21/N 7053"
Basic Reference Model of ODP - Part 1 : Overview and Guide to Use

[ISO107462] "ISO/IEC DIS 10746-2 - ITU-T Draft Rec. X.902 - Feb. 1994"
Basic Reference Model of ODP - Part 2 : Descriptive Model

[ISO107463] "ISO/IEC JTC 1/SC 21/N 10746-3.2 - ITU-T Draft Rec. X.903 - Feb. 1994"
Basic Reference Model of ODP - Part 3 : Prescriptive Model

[ISO7056] "ISO/IEC 21/N 7056" (Recomendation X.905)
Basic Reference Model of ODP - Part 5 : Architectural Semantics

[ISO7047] "ISO/IEC JTC 1/SC 21/N 7047, 1992-06-30"
Working Document on Topic 9.1 - ODP Trader

[ISO7057] "ISO/IEC JTC 1/SC 21/N 7057, 1991-06-20"
WG7 Project Management Document : ODP list of open and resolved issues - June 1992

[ISOTRD] "ISO/IEC JTC 1/SC 21 - Nov. 1993"
Information Technology - ODP Trading Function

[KOSA92] *Kosa, T.; Kreuzer, A.* "Application on programming platforms: From Message Passing to Open Distributed Processing", IFIP 1992, pages 253-266.

[KUTV93] *Kutvonen, L.; Kutvonen, P* "Broadening the User Environment with Implicit Trading" - Proceedings of the ICODP - sept. 1993 - pp 129-140

[LIMA94] *Lima Jr., L. A. P.; Madeira, E. R. M.* "Modelo Para Implementação de Federações de Traders" - artigo submetido à revista da PUCCAMP - 1994

[LINI92] *Linington, P. F.* "Introduction to the Open Distributed Processing Basic Reference Model", IFIP 1992, pages 3-13.

[LOYO94] *Loyolla, W. P. D. C; Madeira, E. R. M.; Mendes, M. J.; Cardoso, E.; Magalhães, M. F.* "Multiware Platform : an Open Distributed Environment for Multimedia Cooperative Applications" - COMPSAC'94 - IEEE Computer Software & Application Conference, Taipei, Taiwan, Nov. 1994

[LUM92] *Lumer, E. D.* "Binding Hierachies : a Basis for Dynamic Perceptual Grouping" - Neural Computation - 4,3 (May 92)

[MEND93] *Mendes, M. J.; Loyolla, W. P. D. C.; Madeira, E. R. M.* "Demos : A Distributed Decision-Making Open Support System" - 4th IEEE Workshop on Future Trends in Distributed Computing Systems - Sept. 1993 - Lisboa - Portugal - pp 208-214

[MEND94] *Mendes, M.; Madeira, E. R. M.* "Plataforma Multiware : Projeto e Desenvolvimento da Camada Middleware" - artigo aceito para XII SBRC - Curitiba - maio de 1994

[MEYER93] *Meyer, B.; Popien, C.* "Object Configuration by ODP Traders" - Proceedings of the ICODP - sept. 1993 - pp 425-430

[OMG1] HP Company, Sun Microsystems, inc, "The Object Management Group, Object Request Broker, RFP joint response", 1991.

[OMG2] HP company, Sun Microsystems, inc, "Distributed Object Management Facility Core Especification", 1991.

[OMG3] Object Request for Proposal 1, OMG TC Document 92.8.6 - October 1992

[OSF1 90] "OSF Distributed Computing Environment" - Set. 1990

[OSF2 90] *Fauth, Dr. Dietmar; Gossels, Jonathan; et al* "Distributed Computing Environment Evaluation Team - OSF" – May 1990.

[OSF3 90] "Distributed Computing Environment Overview"– OSF.

[OSF4 90] "Directory Services for a Distributed Computing Environment" – OSF, September 1990.

[OSF5 90] "File Systems in a Distributed Computing Environment" - OSF - September 1990.

[OSF6 90] "Remote Procedure Call in a Distributed Computing Environment" - OSF - October 1990.

[POPI93] *Popien, C. et al* "Service Management - The Answer of New ODP Requirements ?" - Proceedings of the ICODP - sept. 1993 - pp 431-436

[PROC91] "Proceedings of the IFIP TC6/WG6.4 International Workshop on Open Distributed Processing" - Berlin, Germany, 8-11 October 1991

[SCHO92] *Schoo, Peter; Tonnby, Ingmar* "The ROSA object model", IFIP 1992, pages 291-300.

[SINH92] *Sinha, A.* "Client-Server Computing" - Communications of the ACM - July 1992 - Vol 35 - no. 7

[SLON93] *Slonim, J. et al* "Does Middleware Provide an Adequate Distributed Aplication Environment ?" - Proceedings of the ICODP - sept. 1993 - pp 34-46

[STAI92] *Stainov, Rumen* " Transport services management an add-on distributed operating system layer", IFIP 1992, pages 313-323.

[TAN91] *Tanenbaum, Andrew S.* "Modern Operating Systems" - Prentice Hall - 1991

[TERS92] *Tersiev, Atanas* "Object-induction development of open distributed processing systems", IFIP 1992, pages 325-336.

[TSCH91] *Tschammer, A. W.; Eckert, K.-P.; Hall,J.; Schurman, G. and Strick* "OAI- Concepts for Open Systems Cooperation", LNCS 433, pages 174-191, 1989.

[TSCH92] *Tschammer, A. W.* "CODE - Cooperating Open Systems, Open Distributed Processing, Distributed Computing Environment, Experiments and Projects" - Nov. 1992

[TSCH93] *Tschammer, A. W.; Mendes, M. J.; Souza, W. L.; Madeira , E. R. M.; e Loyolla, W.* "Processamento Distribuído Aberto e o Modelo RM-ODP/ISO" - XI SBRC - maio 1993

[TSCHTRD] *Tschammer, A. W.* "Trading in Open Distributed Systems" - Anais de Artigos Convidados - XI SBRC - 1993

[VOGT93] *Vogt, F.; Andrae, C.* "Middleware for Distributed Applications Support : ODP and/or CORBA" - Proceedings of the ICODP - sept. 1993 - pp 405-410

[YEMI92] Yemini, Shaula; Slonim, Jacob; et al, "Distributed Programming Environments: Challenges", IFIP 1992, pages 379-394.