

**Suporte a Grupos  
Cooperativos em Ambiente  
Distribuído Aberto**

**Fábio Moreira Costa**

# Suporte a Grupos Cooperativos em Ambiente Distribuído Aberto

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. Fábio Moreira Costa e aprovada pela Comissão Julgadora.

Campinas, 03 de outubro de 1995.



Prof. 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.

UNIDADE	AC
N.º CHAMADA:	I. UNICAMP
	L. 833.2
V.	5
TOMBO	26017
PROC.	433195
C	<input type="checkbox"/>
D	<input checked="" type="checkbox"/>
PREÇO	R\$ 11,00
DATA	02/11/95
N.º CPDC	m.00079295-4

FICHA CATALOGRAFICA ELABORADA PELA  
BIBLIOTECA DO IMECC DA UNICAMP

Costa, Fabio Moreira

0823s Suporte a grupos cooperativos em ambiente distribuido  
aberto/ Fabio Moreira Costa. -- Campinas, [SP : s.n.],  
1995.

Orientador: Edmundo Roberto Mauro Madeira.

Dissertacao (mestrado) - Universidade Estadual de Campinas,  
Instituto de Matematica, Estatistica e Ciencia da Computacao.

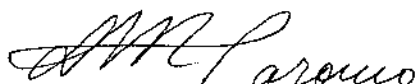
1. Processamento eletronico de dados - Processamento  
distribuido. 2. Redes de computadores 3.\* Comunicacao de grupos.

I. Madeira, Edmundo Roberto Mauro. II. Universidade Estadual de  
Campinas. Instituto de Matematica, Estatistica e Ciencia da  
Computacao. III. Titulo.

FÁBIO MOREIRA COSTA

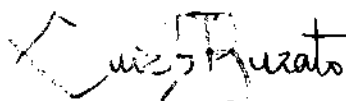
Tese defendida e aprovada em, 03 de OUTUBRO de 1995

Pela Banca Examinadora composta pelos Profs. Drs.



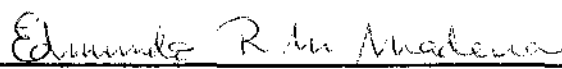
---

Prof(a). Dr(a). LIANE MARGARIDA ROCKENBACH TAROUCO



---

Prof(a). Dr(a). LUIZ EDUARDO BUZATO



---

Prof(a). Dr(a). EDMUNDO ROBERTO MAURO MADEIRA

# Suporte a Grupos Cooperativos em Ambiente Distribuído Aberto<sup>1</sup>

Fábio Moreira Costa<sup>2</sup>

Departamento de Ciência da Computação  
IMECC – UNICAMP

Banca Examinadora:

- Edmundo Roberto Mauro Madeira (orientador)<sup>3</sup>
- Liane Margarida Rockenbach Tarouco<sup>4</sup>
- Luiz Eduardo Buzato<sup>3</sup>
- Nelson Luís Saldanha da Fonseca (suplente)<sup>3</sup>

---

<sup>1</sup>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.

<sup>2</sup>O autor é Bacharel em Ciência da Computação pela Universidade Federal de Goiás

<sup>3</sup>Professor do Departamento de Ciência da Computação - IMECC - UNICAMP.

<sup>4</sup>Professora do Instituto de Informática da Universidade Federal do Rio Grande do Sul

Este trabalho teve o apoio financeiro das seguintes agências: CNPq, CAPES e FAPESP.

*A Jesus Cristo,  
Autor e Senhor de todas as coisas*

# Agradecimentos

Aos meus pais, sem palavras para agradecer o quanto fizeram e fazem por mim. Sem eles, este trabalho não seria possível.

À Betânia, minha mana querida.

À Ester, amor da minha vida, pelo carinho e compreensão durante todo esse tempo.

Ao Professor Edmundo Madeira, meu orientador, cuja temperança e conhecimentos foram fundamentais no desenvolvimento deste trabalho.

Ao Professor Waldomiro Loyolla e ao Fernando Sica, pelas discussões sobre CSCW durante a elaboração do modelo de suporte a grupos.

Aos colegas da “salinha de estudos”, Clevan, Mateus e Ronaldo, pela convivência agradável, que muito me ensinou e, mais que isto, pela amizade desfrutada.

Ao Nuccio, grande companheiro no “desbravamento” da CORBA e do ORBeline.

A todos os colegas do DCC, que fazem deste um ambiente agradável de se conviver.

Aos colegas de moradia durante o tempo do mestrado, Cláudio, Ernesto, Humberto e Walter.

Ao Departamento de Estatística e Informática da Universidade Federal de Goiás.

Aos amigos e, mais do que isto, irmãos do grupo Kerygma, em especial ao Denilson, Marcelo, Carol, Alexandre e Wagner.

A Deus, porque ele é bom.



## Resumo

O estilo de interação em grupo entre os usuários é característica básica de aplicações de trabalho cooperativo. Neste contexto, é fundamental a existência de suporte apropriado de sistemas distribuídos para a estruturação das aplicações em termos de grupos de objetos, permitindo a interação cooperativa entre eles. Este trabalho propõe um modelo para um serviço de suporte a grupos de objetos, a ser usado como ferramenta na construção de aplicações cooperativas. O modelo provê mecanismos para a manutenção consistente dos conjuntos de membros associados aos grupos, bem como para o suporte à coordenação e transparência da comunicação de grupo. A proposta fundamenta-se no Modelo de Referência para Processamento Distribuído Aberto da ISO (RM-ODP) e sua implementação utiliza os recursos da arquitetura CORBA (*Common Object Request Broker Architecture*), de forma a permitir o uso dos serviços em ambientes computacionais heterogêneos. O trabalho descreve a implementação de um protótipo, no ambiente da Plataforma Multiware, o qual provê os serviços básicos do modelo de suporte a grupos.

# Abstract

Computer supported cooperative work applications can be characterized by the style of group interactions among their users. In this context, it is essential the provision of mechanisms that offer suitable support to structure applications in terms of cooperating object groups. This work proposes a model of a group support service to be used as a tool for building cooperative applications. The model provides mechanisms to consistently maintain group membership, as well as to support the coordination and transparency of group communication. The work is based on the ISO Reference Model for Open Distributed Processing (RM-ODP) and its implementation uses resources from the Common Object Request Broker Architecture (CORBA), allowing the services to be used in a heterogeneous computing environment. It is also described the implementation of a prototype, in the Multiware Platform environment, which provides the basic services from the group support model.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Suporte a Aplicações Cooperativas . . . . .	2
1.1.1	Abordagens Existentes . . . . .	3
1.1.2	A Solução Proposta . . . . .	3
1.2	Visão Geral da Dissertação . . . . .	4
<b>2</b>	<b>Processamento Distribuído Aberto - Conceitos Básicos</b>	<b>5</b>
2.1	O Modelo de Referência para ODP . . . . .	6
2.1.1	Pontos de Vista ODP . . . . .	7
2.1.2	Transparências de Distribuição . . . . .	10
2.1.3	Funções ODP . . . . .	11
2.1.4	A Função de Grupos . . . . .	12
2.2	A Arquitetura de Objetos Distribuídos do OMG . . . . .	13
2.2.1	CORBA . . . . .	14
2.3	As Plataformas DCE e ANSAware . . . . .	18
2.4	A Plataforma Multiware . . . . .	19
2.4.1	A Camada Middleware . . . . .	19
2.4.2	A Camada Groupware . . . . .	22
<b>3</b>	<b>Grupos em Sistemas Distribuídos</b>	<b>23</b>
3.1	O Conceito de Grupos . . . . .	24
3.1.1	Classificações de Grupos . . . . .	26
3.2	A Função de Grupos do RM-ODP . . . . .	28
3.2.1	Interação . . . . .	29
3.2.2	Colagem de Interações . . . . .	32
3.2.3	Ordenação . . . . .	33
3.2.4	Controle de Membros . . . . .	38
3.3	Abordagens Relacionadas . . . . .	39
3.3.1	ISIS . . . . .	39
3.3.2	V . . . . .	40
3.3.3	Amoeba . . . . .	41

3.3.4	Emerald . . . . .	41
3.3.5	Grupos de Interfaces em ANSA . . . . .	42
3.3.6	Grupos em CORBA . . . . .	43
<b>4</b>	<b>O Modelo de Grupos de Objetos</b>	<b>44</b>
4.1	O Modelo de Grupos . . . . .	45
4.1.1	Aspectos da Comunicação de Grupo . . . . .	47
4.2	O Modelo de Serviços de Suporte a Grupos . . . . .	50
4.2.1	Serviços de Estruturação de Grupos . . . . .	51
4.2.2	Serviços de Comunicação de Grupo . . . . .	54
<b>5</b>	<b>O Modelo de Implementação</b>	<b>57</b>
5.1	Conceitos Preliminares . . . . .	58
5.1.1	Representação de Informações de Grupos . . . . .	58
5.1.2	Interfaces dos Objetos . . . . .	59
5.1.3	Replicação Passiva de Objetos . . . . .	60
5.2	A Configuração de Objetos de Suporte a Grupos . . . . .	61
5.3	Estrutura e Interface dos Objetos . . . . .	63
5.3.1	Objeto SGsvc . . . . .	64
5.3.2	Objeto SGcoordenador . . . . .	67
5.3.3	Objeto SGlocal . . . . .	69
5.3.4	Objeto Tratador de Falhas . . . . .	73
5.3.5	A Interface dos Objetos de Aplicação . . . . .	73
5.4	Descrição da Implementação dos Serviços . . . . .	73
5.4.1	Ordenação de Eventos . . . . .	75
5.4.2	Detecção e Tratamento de Falhas . . . . .	77
5.4.3	Serviços de Estruturação de Grupos . . . . .	78
5.4.4	Serviços de Distribuição de Mensagens . . . . .	86
<b>6</b>	<b>O Protótipo Implementado</b>	<b>91</b>
6.1	Abrangência do Protótipo . . . . .	92
6.2	Aspectos da Implementação em CORBA . . . . .	93
6.2.1	Definição das Interfaces dos Objetos . . . . .	94
6.2.2	Geração dos <i>Stubs</i> e <i>Skeletons</i> . . . . .	96
6.2.3	Implementação dos Objetos . . . . .	100
6.2.4	Ativação e Desativação de Objetos . . . . .	102
6.2.5	Identificação dos Objetos . . . . .	104
6.2.6	Conexão entre os Objetos . . . . .	105
6.2.7	Comunicação entre os Objetos . . . . .	106
6.3	Integração na Plataforma Multiware . . . . .	107
6.3.1	Usuários do Suporte a Grupos . . . . .	108
6.4	Exemplo de Aplicação do Suporte a Grupos . . . . .	108

<b>7 Conclusão</b>	<b>111</b>
<b>A Definição das Interfaces dos Objetos</b>	<b>114</b>
A.1 Estruturas de Dados dos Parâmetros . . . . .	114
A.2 Interfaces do SGsvc . . . . .	116
A.2.1 Interface Externa do SGsvc . . . . .	117
A.2.2 Interface Interna do SGsvc . . . . .	117
A.2.3 Interface de Implementação do SGsvc . . . . .	118
A.3 Interfaces do SGlocal . . . . .	118
A.3.1 Interface Externa do SGlocal . . . . .	118
A.3.2 Interface Interna do SGlocal . . . . .	119
A.3.3 Interface de Implementacao do SGlocal . . . . .	120
A.4 Interface do SGcoordenador . . . . .	120
A.5 Interface dos Objetos de Aplicação . . . . .	120
<b>Referências</b>	<b>123</b>

# Lista de Tabelas

5.1	Operações na interface externa do SGsvc . . . . .	66
5.2	Operações na interface interna do SGsvc . . . . .	67
5.3	Operações na interface (interna) do SGcoordenador . . . . .	68
5.4	Operações na interface externa do SGlocal . . . . .	72
5.5	Operações na interface interna do SGlocal . . . . .	72
5.6	Operações na interface (interna) do Tratador de Falhas . . . . .	73
5.7	Operações na interface dos objetos de aplicação . . . . .	74

# Lista de Figuras

2.1	Um objeto <i>binding</i> conectando as interfaces de dois objetos computacionais . . . . .	9
2.2	A arquitetura de gerenciamento de objetos da OMG . . . . .	13
2.3	A comunicação entre cliente e servidor através do ORB . . . . .	14
2.4	A estrutura das interfaces do Object Request Broker . . . . .	15
2.5	Arquitetura da Plataforma Multiware . . . . .	20
2.6	Estrutura da sub-camada ODP da camada Middleware . . . . .	20
3.1	Comunicação de grupo . . . . .	25
3.2	Grupos fechados(a) e abertos(b) com respeito às interações de grupo . . . . .	26
3.3	Diferentes estilos de interação em grupo . . . . .	27
3.4	Interação de grupo . . . . .	29
3.5	<i>Binding</i> multi-ponto conectando as interfaces dos objetos de um grupo . . . . .	31
3.6	Colagem de interações . . . . .	32
3.7	Ordenação de eventos de grupo: (a) cenário sem ordenação; (b) cenário com ordenação. . . . .	34
3.8	Ordenação causal das interações de grupo . . . . .	35
3.9	Ordenação total das interações de grupo . . . . .	37
4.1	Um ambiente de suporte a grupos cooperativos . . . . .	45
4.2	Estilo de interação entre os membros de um grupo . . . . .	47
5.1	Estrutura de dados para representar mensagens de grupo . . . . .	59
5.2	Correspondência entre eventos e interações de grupo . . . . .	59
5.3	Estrutura de dados para representar eventos de grupo . . . . .	60
5.4	Modelo de objetos de Suporte a Grupos . . . . .	63
5.5	Interações entre os objetos envolvidos no Serviço de Suporte a Grupos . . . . .	64
5.6	Listas de mensagens pendentes no SGlocal . . . . .	71
5.7	O mecanismo de ordenação causal-total de eventos . . . . .	76
5.8	Criação de Grupo - diagrama temporal de execução do serviço . . . . .	79
5.9	Entrada de Membro - diagrama temporal de execução do serviço . . . . .	80
5.10	Saída de Membro - diagrama de temporal de execução do serviço . . . . .	83
5.11	Mudança de Políticas, Papéis e Autorizações - diagrama temporal de execução dos serviços . . . . .	84

5.12	Estabelecimento e Fechamento de Canais - diagrama temporal de execução . . . . .	85
5.13	Finalização de Grupo - diagrama temporal de execução do serviço . . . . .	86
5.14	Distribuição de Mensagens Com Ordenação - diagrama temporal de execução . . . . .	87
5.15	Distribuição de Mensagens Sem Ordenação - diagrama temporal de execução . . . . .	89
6.1	A comunicação entre os objetos através do ORB . . . . .	93
6.2	Os objetos de suporte a grupos sobre a arquitetura CORBA . . . . .	94
6.3	Hierarquia de interfaces em IDL . . . . .	95
6.4	Arquivos para construção dos clientes e servidores no ORBeline . . . . .	100
6.5	Hierarquia de herança na definição das classes dos objetos de suporte a grupos . . .	101
6.6	Posicionamento do Suporte a Grupos na Plataforma Multiware . . . . .	107



# Capítulo 1

## Introdução

A evolução das tecnologias de redes de computadores permite hoje a comunicação eficiente de grandes volumes de informação entre sistemas computacionais distintos e separados por distâncias consideráveis. Em conseqüência, surge um enorme potencial para o desenvolvimento de aplicações que explorem a natureza distribuída e os recursos de comunicação presentes nestes ambientes. Aplicações podem usufruir de uma série de benefícios decorrentes do fato de serem distribuídas, entre eles, alto grau de disponibilidade e tolerância a falhas, confiabilidade e aumento do poder de processamento, em virtude da execução de tarefas em paralelo.

Entretanto, uma rede de computadores provê um ambiente distribuído onde a existência dos recursos tradicionais que permitem sincronizar e coordenar a execução de tarefas, como memória compartilhada e relógio global, torna-se impraticável, por questões de desempenho. Isto faz com que o desenvolvimento de aplicações distribuídas se torne consideravelmente mais complexo do que no caso de aplicações centralizadas, pois o funcionamento correto da aplicação passa a depender de protocolos que garantam a coordenação das ações dos componentes distribuídos das aplicações. Normalmente, o único recurso básico disponível em um ambiente distribuído para a implementação destes protocolos é a própria comunicação através da rede.

Observa-se, contudo, que os requisitos e problemas de coordenação, em geral, são comuns a várias classes de aplicações distribuídas, podendo ser resolvidos de maneira genérica e independente das características específicas de cada aplicação. Esta observação resultou no surgimento de sistemas de suporte a distribuição, que fornecem um ambiente adequado para que as aplicações sejam desenvolvidas sem lidar diretamente com questões não relevantes aos seus objetivos e, em especial, com os problemas de coordenação do processamento distribuído.

Em especial, destacam-se as aplicações de trabalho cooperativo suportado por computador (CSCW - *Computer Supported Cooperative Work*), que procuram apoiar a cooperação entre usuários organizados em grupos. Estas aplicações apresentam um conjunto comum de requisitos quanto ao suporte de distribuição, que justificam a definição de plataformas de suporte a cooperação.

Este trabalho propõe um modelo de ambiente de suporte de distribuição para aplicações co-

operativas. O modelo está baseado em *grupos de objetos cooperantes*, os quais representam, no nível de sistemas distribuídos, os grupos de usuários da aplicação. São abordados os aspectos de estruturação e comunicação de grupos, juntamente com o problema de coordenação distribuída com relação a estes aspectos. Faz parte do trabalho a implementação dos principais aspectos do modelo proposto. A definição do modelo e sua implementação integram a arquitetura da Plataforma Multiware, atualmente em desenvolvimento na UNICAMP.

## 1.1 Suporte a Aplicações Cooperativas

Uma definição amplamente aceita para *aplicação cooperativa* é encontrada em [EGR91]:

“Sistema baseado em computador que apóia grupos de pessoas envolvidas em uma tarefa ou meta comum e que provê uma interface para um ambiente compartilhado.”

Esta classe de aplicações tem como objetivo explorar o potencial para cooperação existente em ambientes de trabalho em grupo, através da provisão de tecnologias para enfatizar a capacidade de interação entre os usuários membros de grupos. Exemplos de aplicações cooperativas são: sistemas de mensagens, sistemas de conferência computadorizada, sistemas de sala de encontro e sistemas de co-autoria e argumentação.

O suporte ao desenvolvimento de aplicações de trabalho cooperativo está fundamentado em duas áreas de conhecimento distintas, mas complementares [LFSM94]: a área *sociológica*, que estuda os aspectos de cooperação e do comportamento do trabalho em grupo; e a área *tecnológica*, que procura aplicar recursos de processamento de informação e comunicação de dados para implementar e manter o ambiente de cooperação e as interações entre os usuários.

A especificação do suporte tecnológico para aplicações cooperativas envolve dois aspectos distintos: suporte no nível de sistemas distribuídos (devido à necessidade de suportar múltiplos usuários) e suporte ao compartilhamento de recursos.

Especialmente importantes para o presente trabalho são as questões de *suporte no nível de sistemas distribuídos* para aplicações cooperativas (as questões de compartilhamento de recursos são normalmente resolvidas utilizando o suporte de sistemas distribuídos). O desenvolvimento destas aplicações depende das facilidades providas pelas plataformas de sistemas distribuídos, principalmente com relação aos aspectos de controle e comunicação. Aplicações de trabalho cooperativo necessitam de mecanismos de controle flexíveis, capazes de se ajustarem à dinâmica dos ambientes cooperativos. Em [BR91], propõe-se que o suporte de distribuição para aplicações cooperativas seja baseado em transparências de distribuição seletivas, e na separação clara entre mecanismos e políticas, permitindo que as próprias aplicações definam a forma mais adequada em que as facilidades de suporte devem ser providas.

Quanto ao aspecto de comunicação, é necessário prover facilidades que suportem os ricos padrões de interação entre os usuários de aplicações cooperativas. Em particular, é necessário suportar o estilo de *comunicação em grupo* (ou “muitos-para-muitos”), para permitir que vários usuários de um grupo cooperativo participem de uma mesma interação [SST94]. Além disso, os mecanismos de comunicação em grupo podem suportar a manutenção da consistência do ambiente cooperativo,

através da coordenação da comunicação. O suporte de comunicação deve ser flexível o suficiente, permitindo o seu ajuste às necessidades dinâmicas das aplicações (através de políticas e transparências seletivas).

Um outro requisito emergente de aplicações cooperativas é ressaltado em [MNR92] e se refere à necessidade de garantir a interoperabilidade entre aplicações diferentes em um ambiente heterogêneo.

Estes requisitos básicos mostram que os modelos tradicionais de sistemas distribuídos não oferecem o suporte adequado para aplicações cooperativas. Os mecanismos para interação normalmente presentes nestes sistemas permitem apenas a comunicação entre pares de entidades (estilo de comunicação “um-para-um”), o que exige mais esforço durante o desenvolvimento de aplicações cooperativas. Além disso, estes mecanismos, em geral, não suportam a flexibilidade necessária para expressar as diversas formas de interações cooperativas.

### 1.1.1 Abordagens Existentes

Duas abordagens básicas podem ser seguidas na construção de aplicações cooperativas. Uma primeira abordagem utiliza os mecanismos tradicionais de sistemas distribuídos. Os problemas de controle e coordenação da comunicação são resolvidos no próprio nível da aplicação, sendo obtidas soluções específicas para cada caso. Esta abordagem pode ser exemplificada através do sistema GROVE [EGR91], onde são empregados mecanismos próprios de coordenação distribuída.

Na segunda abordagem, procura-se obter um consenso quanto aos serviços e funcionalidades comumente utilizados por aplicações cooperativas em geral, os quais são reunidos e implementados em uma plataforma de suporte. Com isto, é possível reduzir o esforço necessário para o desenvolvimento de aplicações cooperativas, uma vez que os serviços básicos de apoio à cooperação são providos pela plataforma. Um exemplo desta abordagem é encontrado em [SR92], que propõe a implementação dos mecanismos comuns de suporte a trabalho cooperativo (tanto mecanismos de suporte a distribuição, quanto facilidades de apoio à cooperação) no nível de sistemas operacionais, sob a forma de um módulo ou camada construído sobre um núcleo com serviços básicos (*microkernel*).

### 1.1.2 A Solução Proposta

O modelo de suporte a aplicações cooperativas aqui adotado corresponde à segunda abordagem mencionada acima, refletindo a arquitetura da Plataforma Multiware [LMM<sup>+</sup>94] em suas camadas superiores (capítulo 2). O problema de suporte ao trabalho cooperativo é dividido em dois níveis: o suporte aos aspectos de cooperação, como o compartilhamento de recursos e as interfaces com os usuários, e o suporte de sistemas distribuídos, cujo propósito é fornecer mecanismos de comunicação consistentes e confiáveis para uso no nível de suporte à cooperação.

Neste trabalho, é apresentada uma solução para o problema de suporte a aplicações cooperativas no nível de sistemas distribuídos. É proposto um conjunto de funcionalidades que permitem a estruturação de aplicações cooperativas como conjuntos de componentes distribuídos, capazes de cooperação através de mecanismos de comunicação de grupo. Desta forma, as interações entre os

usuários cooperantes, que normalmente seguem um padrão do tipo “muitos-para-muitos”, podem ser diretamente mapeadas para os mecanismos de suporte à comunicação distribuída.

A solução proposta está baseada no conceito de *grupos de objetos* (ou grupos de processos), empregado em certos sistemas distribuídos como recurso para suportar aplicações distribuídas tolerantes a falhas ou que envolvem replicação [Isi93, Maf93, Bir93]. Neste modelo, os objetos membros de um grupo de objetos podem corresponder, diretamente, aos usuários membros de grupos de trabalho cooperativo, sendo que os aspectos de comunicação de grupo ficam transparentes para o nível das aplicações. No restante do trabalho, referimo-nos ao conjunto de funcionalidades para o suporte de distribuição a aplicações cooperativas simplesmente como *suporte a grupos*.

O conjunto de funcionalidades de suporte a grupos, embora elaborado para as necessidades específicas de aplicações de trabalho cooperativo, pode ser utilizado como suporte para outros tipos de aplicações que envolvam algum tipo de cooperação (não necessariamente no sentido empregado em CSCW) entre componentes distribuídos para prover algum serviço comum. Exemplos de outras aplicações que podem utilizar o conceito de grupos (e dos mecanismos de suporte a grupos) são encontrados no capítulo 3.

O modelo de suporte a grupos proposto é definido em um ambiente de processamento distribuído aberto, o que possibilita atender ao requisito da interoperabilidade entre aplicações cooperativas heterogêneas.

## 1.2 Visão Geral da Dissertação

O próximo capítulo procura caracterizar ambientes de processamento distribuído aberto, fornecendo os conceitos básicos utilizados na definição do modelo de suporte a grupos e em sua implementação. São descritos, em linhas gerais, o Modelo de Referência para Processamento Distribuído Aberto (RM-ODP) proposto pela ISO, juntamente com a arquitetura CORBA (*Common Object Request Broker Architecture*), proposta pela OMG, que oferece serviços básicos para a construção de aplicações distribuídas em ambientes abertos. O capítulo descreve também a Plataforma Multitware, na qual está inserido o presente trabalho.

No capítulo 3 é definido o conceito de grupos de objetos cooperativos, sendo apresentados exemplos de sistemas distribuídos que empregam este conceito. O capítulo apresenta a Função de Grupos do RM-ODP, propondo um refinamento das suas funcionalidades, utilizando os conceitos de suporte a grupos propostos na literatura.

O capítulo 4 apresenta o modelo de suporte a grupos proposto neste trabalho, enquanto que o capítulo 5 descreve a abordagem utilizada para implementação deste modelo. Aspectos da implementação do protótipo sobre a arquitetura CORBA são descritos no capítulo 6.

Finalmente, o capítulo 7 apresenta um resumo das contribuições do trabalho, juntamente com possíveis extensões ao modelo de suporte a grupos e à sua implementação.

## Capítulo 2

# Processamento Distribuído Aberto - Conceitos Básicos

As tendências e necessidades atuais em processamento de informação têm levado a uma mudança radical na forma de estruturação de sistemas computacionais. De arquiteturas altamente centralizadas no passado, evolui-se para ambientes onde a distribuição dos recursos de processamento e armazenamento de dados, entre outros, é a característica marcante. Um fator decisivo para esta mudança de rumos foi o desenvolvimento recente das tecnologias de redes de longo alcance e alto desempenho, que tornou possível a troca de grandes volumes de dados entre computadores, independentemente de sua localização geográfica. As conseqüências mais visíveis da abordagem distribuída seriam um considerável aumento na flexibilidade e tolerância a falhas dos sistemas, além de um substancial ganho na relação entre custo e desempenho.

Por outro lado, novas complicações, inexistentes na abordagem centralizada tradicional, tornam mais difícil a tarefa de desenvolvimento de sistemas de processamento distribuído. Em primeiro lugar, é necessário ocultar do usuário os problemas decorrentes da distribuição dos recursos, de maneira que ele seja capaz de usá-los sem se preocupar com os detalhes de localização e acesso aos mesmos. Deve-se ainda levar em conta a necessidade de reutilização de sistemas já existentes, preservando investimentos já realizados.

Desta forma, as novas tecnologias deveriam não apenas permitir a distribuição transparente dos recursos, mas fazê-la com a integração de sistemas heterogêneos, com modularidade, flexibilidade e com atenção aos requisitos específicos de cada tipo de aplicação. Isto torna evidente o fato de que a primeira geração de sistemas distribuídos não fornece um suporte adequado às novas necessidades dos usuários, principalmente pelo fato de serem sistemas fechados e inflexíveis (dos pontos de vista de arquiteturas, tecnologias e políticas de uso), constituídos apenas por componentes homogêneos. Os próprios sistemas distribuídos tradicionais não foram desenvolvidos com base em conceitos comuns, o que gerou arquiteturas distribuídas incompatíveis entre si e incapazes de interoperabilidade.

Ficam claros, portanto, os seguintes aspectos essenciais para o desenvolvimento dos novos sistemas distribuídos [TMM<sup>+</sup>93, ISO94a]:

- *abertura*: os novos ambientes de processamento distribuído não devem apresentar restrições de acesso (qualquer usuário ou componente pode, em princípio, ser admitido no sistema) ou de natureza geográfica e organizacional (os componentes do sistema podem estar espalhados em várias organizações e em locais diferentes);
- *heterogeneidade*: devem permitir a integração de componentes heterogêneos e autônomos (com suas próprias normas e políticas, que definem suas propriedades, comportamento e evolução);
- *descentralização de controle e gerenciamento*: não existe uma autoridade única no sistema distribuído;
- *padronização*: é extremamente importante que os futuros sistemas distribuídos sejam construídos sobre bases e conceitos arquiteturais comuns, que permitam a sua interação cooperativa.

Os esforços em Processamento Distribuído Aberto (ODP - *Open Distributed Processing*), notadamente a padronização do Modelo de Referência para ODP (RM-ODP) da ISO [ISO94a], têm como objetivo fundamental adicionar abertura à nova geração de sistemas distribuídos, permitindo a integração de componentes heterogêneos e autônomos e eliminando as restrições de acesso potencial ao sistema [TMM<sup>+</sup>93]. O RM-ODP provê um modelo uniforme para a construção de sistemas distribuídos abertos que os torne interoperáveis entre si.

Em paralelo aos esforços em torno do RM-ODP, outras organizações têm elaborado propostas de arquiteturas padronizadas para sistemas distribuídos abertos. Destacam-se as iniciativas do OMG (*Object Management Group*), através da especificação da arquitetura CORBA (*Common Object Request Broker Architecture*) [OMG91]; da OSF (*Open Software Foundation*) através da plataforma DCE (*Distributed Computing Environment*) [OSF90], além do projeto ANSA (*Advanced Networked Systems Architecture*) [Lin93]. Estas arquiteturas têm como objetivo oferecer suporte para o desenvolvimento de sistemas distribuídos em ambientes heterogêneos. Embora não apresentem todas as funcionalidades especificadas no RM-ODP, estas arquiteturas não contradizem o modelo, podendo até mesmo serem usadas como base para a construção de ambientes com as funcionalidades ODP.

A próxima seção apresenta os conceitos básicos do Modelo de Referência para ODP, com ênfase no ponto de vista computacional, usado em capítulos posteriores. A seção 2.2 descreve a arquitetura CORBA, com base na qual foi desenvolvido o protótipo de suporte a grupos (capítulo 6), enquanto que a seção 2.3 apresenta uma visão geral de outras plataformas para processamento distribuído aberto. Finalmente, na seção 2.4 é apresentada a arquitetura da Plataforma Multiware [LMM<sup>+</sup>94, Mad95], atualmente em desenvolvimento na UNICAMP, cujo objetivo consiste em prover suporte à construção de sistemas distribuídos abertos com base no RM-ODP, utilizando os recursos fornecidos por plataformas comercialmente disponíveis.

## 2.1 O Modelo de Referência para ODP

O Modelo de Referência para Processamento Distribuído Aberto (RM-ODP) representa o esforço de padronização da ISO (*International Organization for Standardization*) e ITU-T (*International*

*Telecommunications Union*) no sentido de estabelecer os requisitos que um sistema distribuído deve satisfazer para ser considerado um sistema ODP. O modelo fornece as linhas mestras para o projeto de sistemas distribuídos abertos, de forma a garantir a abertura, autonomia e interoperabilidade dos sistemas. Atualmente, o RM-ODP encontra-se em processo de padronização internacional na ISO.

Não é objetivo do RM-ODP padronizar todos os aspectos de todos os tipos de sistemas de processamento distribuído aberto, nem de influenciar a escolha de tecnologias [Ray93]. Ele pretende ser usado como um modelo para a construção de padrões para áreas de aplicação específicas. Desta forma, o RM-ODP pode ser visto como um conjunto de conceitos e regras para a especificação de padrões, restringindo as características destes padrões específicos e tornando-os compatíveis entre si. Por outro lado, o modelo pretende também ser usado por especificadores de sistemas como uma base para projeto, definindo a estrutura dos sistemas a serem construídos.

O RM-ODP consiste de quatro documentos básicos de padronização:

- ISO 10746-1 / ITU-T X.901 [ISO94a] - "Part 1: Overview and Guide to Use" : apresenta uma visão geral e motivações para a definição e uso de padrões ODP, bem como os conceitos principais utilizados no modelo de referência; este documento apresenta também alguns exemplos de especificação de sistemas ODP.
- ISO 10746-2 / ITU-T X.902 [ISO94b] - "Part 2: Descriptive Model" : apresenta as definições dos conceitos necessários para a especificação de sistemas distribuídos.
- ISO 10746-3 / ITU-T X.903 [ISO94c] - "Part 3: Prescriptive Model" : prescreve um conjunto de conceitos, estruturas, regras e funções usados na especificação de sistemas de processamento distribuído aberto.
- ISO 10746-4 / ITU-T X.904 [ISO93] - "Part 4: Architectural Semantics" : descreve como os conceitos de modelagem da Parte 2 podem ser representados usando técnicas de descrição formal.

### 2.1.1 Pontos de Vista ODP

Para lidar com a complexidade do processo de construção de sistemas distribuídos, o RM-ODP adota uma abordagem baseada em *pontos de vista*. Em cada ponto de vista um sistema de processamento distribuído aberto pode ser descrito completamente, de acordo com um particular conjunto de interesses relevantes. O Modelo Prescritivo do RM-ODP [ISO94c] descreve cinco pontos de vista, brevemente comentados a seguir.

Cada ponto de vista possui uma *linguagem*, com conceitos e regras de estruturação próprios, capaz de descrever completamente um sistema distribuído aberto de acordo com o respectivo ponto de vista. Todas as cinco linguagens de pontos de vista empregam conceitos uniformes de modelagem orientada a objetos. Desta forma obtém-se uma base comum de projeto, que possibilita relacionar as especificações de um sistema distribuído aberto, realizadas segundo pontos de vista diferentes, estabelecendo correspondências entre as representações nestes diferentes pontos de vista [ISO94a,

ISO94b]. Cada linguagem de ponto de vista define pontos de conformidade a partir dos quais a consistência de uma especificação em relação aos outros pontos de vista pode ser verificada. Outra vantagem do uso de um modelo comum orientado a objetos é que ele facilita manter sob controle a complexidade do sistema distribuído em processo de especificação; além disso, os conceitos de orientação a objetos têm se mostrado apropriados para a modelagem de sistemas distribuídos [NWM93].

### **Ponto de Vista de Empresa**

Neste ponto de vista um sistema de processamento distribuído aberto (sistema ODP) é descrito em termos de políticas de negócios e de gerenciamento, sendo os componentes do sistema modelados através dos conceitos de *agentes* (componentes ativos, que realizam funções do sistema) e *artefatos* (componentes passivos, que representam os recursos utilizados na realização das funções do sistema). Na especificação de empresa de um sistema, pode-se também expressar os *papéis* que os agentes (que podem representar usuários) desempenham no sistema, juntamente com a relação entre estes agentes e suas interações com o ambiente externo do sistema.

A especificação de empresa descreve onde e como o sistema está localizado dentro da empresa [TMM<sup>+</sup>93]. O termo empresa, neste contexto, não se restringe a uma organização específica, pois o sistema ODP pode estar espalhado em vários ambientes organizacionais diferentes.

### **Ponto de Vista de Informação**

No ponto de vista de informação, um sistema ODP pode ser representado em termos de objetos de informação (que representam as entidades do mundo real) e dos relacionamentos entre estes objetos (através de fluxos de informação). A linguagem de informação contém conceitos que possibilitam a especificação da semântica das informações armazenadas em um sistema ODP, bem como da semântica de sua manipulação através das atividades de processamento do sistema. Técnicas de modelagem de dados são utilizadas para especificar a composição das estruturas e fluxos de dados do sistema.

### **Ponto de Vista de Computação**

Este ponto de vista trata da especificação do sistema ODP em termos de seus objetos componentes, suas atividades e as interações que ocorrem entre eles. São também descritos os algoritmos, os tipos e os fluxos de dados usados na realização das funções do sistema distribuído. Neste ponto de vista, pode-se especificar e construir um sistema distribuído aberto de maneira transparente em relação aos aspectos e problemas do ambiente distribuído.

As construções usadas para a especificação e implementação de sistemas ODP na linguagem de computação são baseadas nos conceitos de *objeto computacional*, *interface* e *interações*. Desta forma, segundo o ponto de vista de Computação, um sistema ODP consiste de uma *configuração de objetos* [ISO94b], capazes de interagirem entre si através de suas interfaces.



Objetos computacionais são os componentes básicos que realizam as funcionalidade dos sistemas ODP, sendo que um objeto é composto de *estado* (as informações por ele mantidas) e *comportamento* (as funções por ele realizadas sobre seu estado).

Objetos computacionais interagem entre si através de suas *interfaces*, sendo que um mesmo objeto pode possuir várias interfaces diferentes, cada uma fornecendo acesso a um subconjunto dos serviços oferecidos pelo objeto. Interfaces podem ser descritas a partir de interfaces já existentes através do mecanismo de *herança* de interfaces. Três categorias de interfaces computacionais são descritas no modelo prescritivo:

- *interface operacional*, que descreve um conjunto de operações que podem ser invocadas sobre um objeto; interações através de uma interface operacional podem ser de dois tipos:
  - **interrogações**, que representam interações do tipo “cliente-servidor”, constituídas de uma invocação de serviço seguida do retorno de uma *resposta*;
  - **anúncios**, que representam interações constituídas pela invocação de uma operação, para a qual não existe retorno de resposta;
- *interface de fluxo*, que permite interações no estilo “produtor-consumidor”, através de fluxos contínuos de informação entre os objetos;
- *interface de sinal*, utilizada para sinalizar eventos de controle relativos ao estabelecimento e manutenção da comunicação entre objetos.

A interação entre objetos ocorre, portanto, a partir da conexão de suas interfaces, o que é realizado pelos objetos *binding*, conforme ilustrado na Figura 2.1. Um objeto *binding* pode ser usado para conectar múltiplas interfaces, habilitando a comunicação “multiponto” entre os objetos portadores das interfaces. Observe que um objeto *binding* pode possuir uma interface operacional de controle, que permite a manutenção da conexão, além de interfaces de sinal, através das quais os objetos computacionais conectados interagem com o objeto *binding*.

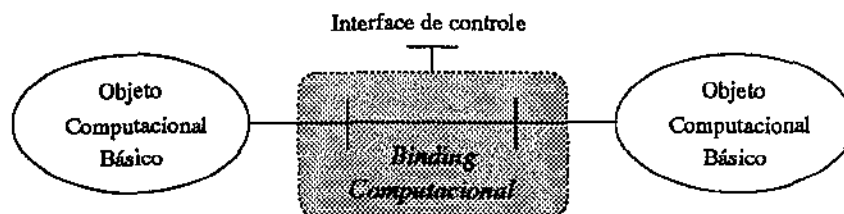


Figura 2.1: Um objeto *binding* conectando as interfaces de dois objetos computacionais

### Ponto de Vista de Engenharia

O ponto de vista de engenharia trata da provisão da infraestrutura de suporte para sistemas ODP, com ênfase nos mecanismos que realizam as transparências de distribuição. Neste ponto de vista, um

sistema distribuído aberto é considerado em termos dos aspectos de distribuição dos componentes do sistema, de modo a prover um ambiente transparente para o ponto de vista de computação.

A linguagem do ponto de vista de Engenharia introduz uma série de conceitos relacionados com o suporte aos aspectos de estruturação e gerenciamento dos recursos de sistemas distribuídos, bem como aos aspectos de estruturação da comunicação entre os componentes do sistema. Estes conceitos de estruturação, juntamente com as regras de especificação relacionadas são descritos extensivamente em [Gar94].

### Ponto de Vista de Tecnologia

Neste ponto de vista são tratados os aspectos de implementação, instalação e manutenção dos componentes e meios de comunicação dos quais o sistema ODP é constituído.

#### 2.1.2 Transparências de Distribuição

Diz-se que um ambiente distribuído é transparente quando a construção e uso de aplicações neste ambiente podem ser realizados sem a necessidade de tratar aspectos que surgem em virtude da distribuição dos componentes da aplicação. Em um ambiente ODP, transparências são providas de forma a tornar possível a especificação de aplicações distribuídas, nos pontos de vista de empresa, informação e computação, abordando apenas os aspectos próprios da aplicação. Em especial, no ponto de vista de computação, uma aplicação distribuída pode ser especificada e construída com base em objetos que interagem entre si, sem considerar o fato de que cada objeto pode estar em um local diferente.

A provisão de transparências de distribuição em ODP é seletiva, o que significa que uma aplicação distribuída pode utilizar apenas as transparências necessárias. Estas transparências são providas como parte da infraestrutura de distribuição do ponto de vista de engenharia. Desta forma, a especificação de um sistema do ponto de vista de engenharia envolve a descrição de um conjunto de objetos que cooperam entre si para prover as transparências selecionadas.

O modelo de referência para ODP prevê um conjunto de transparências consideradas básicas para suportar processamento distribuído aberto:

- *Transparência de Acesso*: mascara a heterogeneidade dos componentes do sistema distribuído, permitindo que o acesso a seus serviços seja feito de maneira uniforme com respeito às representações de dados e mecanismos de invocação utilizados.
- *Transparência de Localização*: permite que um objeto interaja com outros objetos sem necessariamente conhecer a localização destes objetos (isto implica em que o mecanismo de identificação de objetos deve ser independente da localização dos objetos).
- *Transparência de Relocação*: permite que um objeto seja mudado de localização sem afetar outros objetos que dele façam uso.
- *Transparência de Migração*: permite que um objeto seja movido de um lugar para outro sem afetar o seu próprio comportamento; a transparência de migração garante que o ambiente

externo com o qual o objeto migrado interage não é alterado (do ponto de vista do objeto).

- **Transparência de Falhas:** oculta de um objeto a provisão de mecanismos para torná-lo tolerante a falhas.<sup>1</sup>
- **Transparência de Transação:** oculta, de uma configuração de objetos, aspectos da coordenação de operações transacionais que atuam sobre o estado compartilhado e necessitam ser executadas atômicamente, mesmo em presença de falhas.
- **Transparência de Replicação:** permite o uso de um grupo de objetos (comportamentalmente compatíveis) através de uma interface única como se constituíssem um único objeto.
- **Transparência de Persistência:** oculta de um objeto o uso de mecanismos para ativação e desativação deste objeto, permitindo que seu tempo de vida seja independente de seu ambiente de suporte.

### 2.1.3 Funções ODP

O modelo de referência apresenta um conjunto de funções consideradas fundamentais para a construção de sistemas ODP. O uso das funções ODP na especificação de sistemas é seletivo, sendo que especificações de funções individuais podem ser combinadas para formar a especificação de cada um dos componentes de um sistema (juntamente com as funcionalidade específicas da área de aplicação do sistema). A identificação das funções a serem utilizadas em um sistema ODP normalmente é feita durante a especificação do sistema nos pontos de vista de engenharia ou de computação.

As funções ODP prescritas no modelo de referência podem ser agrupadas em quatro conjuntos básicos:

- **Funções de Gerenciamento:** permitem a criação, utilização e desativação dos recursos e componentes básicos da infraestrutura de engenharia sobre a qual um sistema ODP é construído. As funções de gerenciamento possibilitam a manutenção e controle das propriedades dos serviços oferecidos por estes componentes da infraestrutura. Em [Gar94] as funções de gerenciamento são descritas em detalhe, sendo proposto um modelo para sua implementação.
- **Funções de Coordenação:** conjunto de funções que tratam do controle e coordenação do comportamento dos componentes de um sistema distribuído aberto. A provisão destas funcionalidades é feita com o uso das funções de gerenciamento. As funções de coordenação do RM-ODP são as seguintes:
  - funções de *checkpoint* e recuperação, que coordenam a restauração da consistência do estado de objetos afetados por falhas;
  - funções de desativação e reativação, que permitem coordenar a ativação e desativação dos componentes básicos de um sistema distribuído;

---

<sup>1</sup>Note que o ocultamento da falha de um objeto em relação aos clientes deste objeto falho pode ser obtido através de outras transparências do RM-ODP, como, por exemplo, a transparência de replicação.

- função de notificação de eventos, que mantém e disponibiliza um registro dos eventos que ocorrem no ambiente do sistema ODP;
  - função de grupos, que oferece funcionalidades para coordenar a interação cooperativa entre objetos de um grupo;
  - função de migração, que coordena a migração dos componentes de um sistema distribuído;
  - função de replicação, que trata dos aspectos de utilização e manutenção da consistência de recursos replicados; é um caso especial da função de grupo, onde todos os membros de um grupo são idênticos entre si;
  - função de transação, que trata da garantia das propriedades transacionais de operações que acessam recursos compartilhados e necessitam de tolerância a falhas.
- **Funções de Repositório:** oferecem facilidades para o armazenamento de informações relativas a um sistema ODP. Em particular, há funções para gerenciamento dos recursos de armazenamento em geral e funções para gerenciar repositórios de localizações de interfaces e repositórios de tipos. Uma outra função importante desta classe corresponde ao *Trader*, que mantém informações sobre os serviços disponíveis no ambiente ODP, permitindo a sua divulgação para os potenciais clientes destes serviços (esta funcionalidade é tratada em detalhe em [Lim94]).
  - **Funções de Segurança:** oferecem facilidades para controle de acesso às interfaces dos objetos, para a obtenção de informações de segurança sobre o sistema (para auditoria), para autenticação dos objetos participantes em interações, para a manutenção da integridade de dados (contra alterações não-autorizadas), bem como para garantir a confidencialidade dos dados.

#### 2.1.4 A Função de Grupos

A comunicação entre múltiplos objetos é prevista no RM-ODP através do mecanismo de *binding multiponto*, que permite a um objeto interagir transparentemente com um grupo de objetos. Entretanto, este mecanismo não provê funcionalidades para a coordenação das interações entre os objetos, pois apenas os aspectos de comunicação são tratados. Isto implica em que a coordenação das interações em grupo deve ser provida em um nível acima dos *bindings*.

A *Função de Grupos* do RM-ODP provê estas funcionalidades de coordenação, tratando questões como a consistência e confiabilidade das interações em grupo (através de funcionalidades para a distribuição, ordenação e colagem de interações), oferecendo também funcionalidades para a manutenção dos conjuntos de membros dos grupos. A Função de Grupos constitui a base para o presente trabalho e é analisada em detalhes no capítulo 3.

## 2.2 A Arquitetura de Objetos Distribuídos do OMG

Com o objetivo de padronizar o desenvolvimento de aplicações orientadas a objetos, foi formado, em 1989 através de um consórcio de empresas, o *Object Management Group* (OMG). Inicialmente, foi proposta a Arquitetura para Gerenciamento de Objetos (OMA - *Object Management Architecture*) [OMG92a], que apresenta um modelo genérico do ambiente de objetos adotado pelo OMG. A OMA define, em um nível de abstração elevado, os vários componentes utilizados na construção de aplicações em um ambiente de computação distribuída orientada a objetos [Vin93]. Estes componentes são organizadas em quatro classes, ilustradas na Figura 2.2 e descritas a seguir:

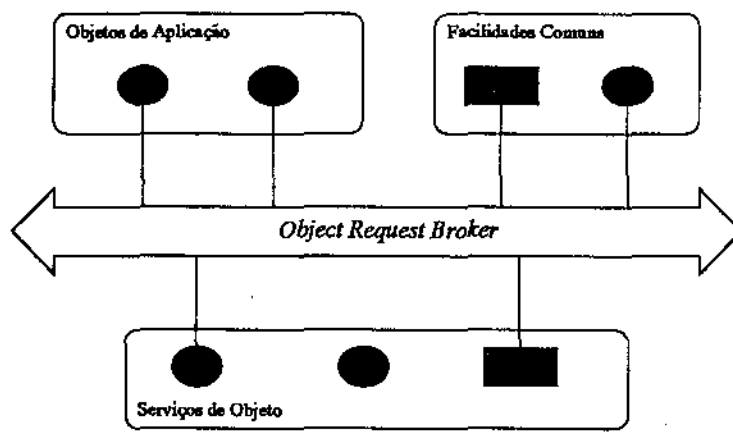


Figura 2.2: A arquitetura de gerenciamento de objetos da OMG

- **Object Request Broker (ORB):** constitui o núcleo da OMA, o qual provê mecanismos de transparência para a localização e ativação de objetos, bem como para a comunicação entre eles. O ORB oferece uma estrutura que permite a comunicação entre objetos, independentemente das tecnologias usadas para implementar os objetos. Toda a comunicação entre os demais componentes da OMA é realizada através do ORB, o que garante a interoperabilidade e portabilidade destes componentes em uma rede de sistemas heterogêneos.
- **Serviços de Objeto (*Object Services*):** estes componentes oferecem funcionalidades padronizadas para o suporte ao desenvolvimento das aplicações, permitindo a criação de objetos, o controle de acesso aos objetos e a manutenção de informações sobre a localização de objetos, entre outros.
- **Facilidades Comuns (*Common Facilities*):** provê um conjunto de funções genéricas, identificadas no nível das aplicações, que podem ser configuradas para atender aos requisitos de particulares configurações de aplicação [OMG94a]. As facilidades comuns estão diretamente relacionadas com os usuários das aplicações, sendo que a ênfase de sua padronização está na uniformização das interfaces de acesso às facilidades. Exemplos são os serviços de impressão, gerenciamento de documentos, correio eletrônico, e bancos de dados.

- **Objetos de Aplicação:** representam os componentes da arquitetura OMA que executam as tarefas específicas de cada aplicação particular. Embora não seja o objetivo da OMG padronizar estes componentes, é adotado um *modelo de objetos* comum [OMG95a] a ser utilizado na especificação das aplicações. Este modelo de objetos segue um estilo de interação do tipo “cliente-servidor”, onde um cliente invoca objetos servidores para a obtenção de serviços.

Os dois primeiros componentes da arquitetura, o ORB e os serviços de objeto, estão relacionados com o suporte às aplicações, no nível de sistemas distribuídos. Para estes componentes a OMG promoveu a definição de padrões para a unificação das interfaces de acesso aos serviços por eles providos<sup>2</sup>. Este esforço de padronização resultou na especificação do modelo de Serviços de Objeto Comuns (*Common Object Services*) [OMG95b] e da CORBA, Arquitetura Comum para o ORB (*Common Object Request Broker Architecture*) [OMG91], esta última descrita a seguir.

### 2.2.1 CORBA

A arquitetura CORBA define um padrão a ser seguido na implementação do componente *Object Request Broker* (ORB) da OMA. A abordagem consiste em prover, no ORB, um conjunto mínimo de mecanismos para a comunicação entre objetos distribuídos, sendo que as funcionalidades mais elaboradas são providas através dos Serviços de Objeto opcionais (que, por sua vez, são implementados sobre o ORB). A especificação da CORBA prescreve um padrão para os serviços fornecidos pelo ORB, bem como para as *interfaces* de acesso a estes serviços. Várias implementações diferentes do ORB podem existir, em conformidade com a CORBA, contanto que suportem o padrão para as interfaces e serviços.

O serviço básico provido pelo ORB consiste na condução, através da rede, das invocações dos clientes para os objetos servidores, bem como dos resultados das invocações no sentido oposto, conforme mostra a Figura 2.3. O ORB permite que a comunicação entre clientes e servidores

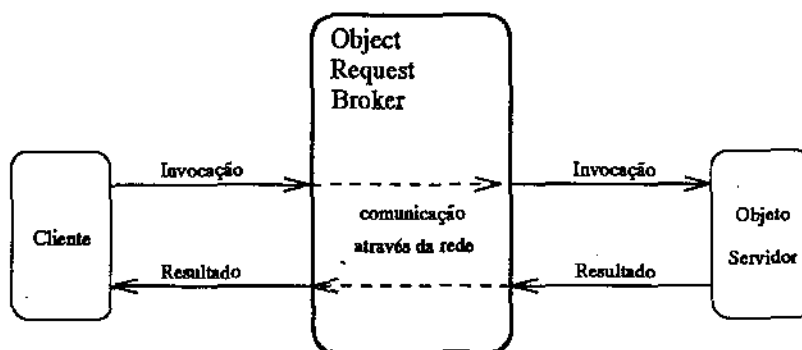


Figura 2.3: A comunicação entre cliente e servidor através do ORB

seja transparente com respeito aos aspectos de localização dos objetos e formas de acesso aos

<sup>2</sup>A OMG também especifica um padrão para as Facilidades Comuns

mesmos, de modo que, do ponto de vista dos participantes nas interações, é como se as invocações ocorressem localmente. A transparência é obtida fazendo-se com que tanto clientes como servidores se comuniquem através de interfaces intermediárias uniformes.

Os componentes propostos na especificação da CORBA, notadamente as interfaces para acesso aos serviços da arquitetura, são mostrados na Figura 2.4. A especificação não prescreve um padrão para a implementação do núcleo do ORB; exige-se apenas que este apresente as interfaces padronizadas.

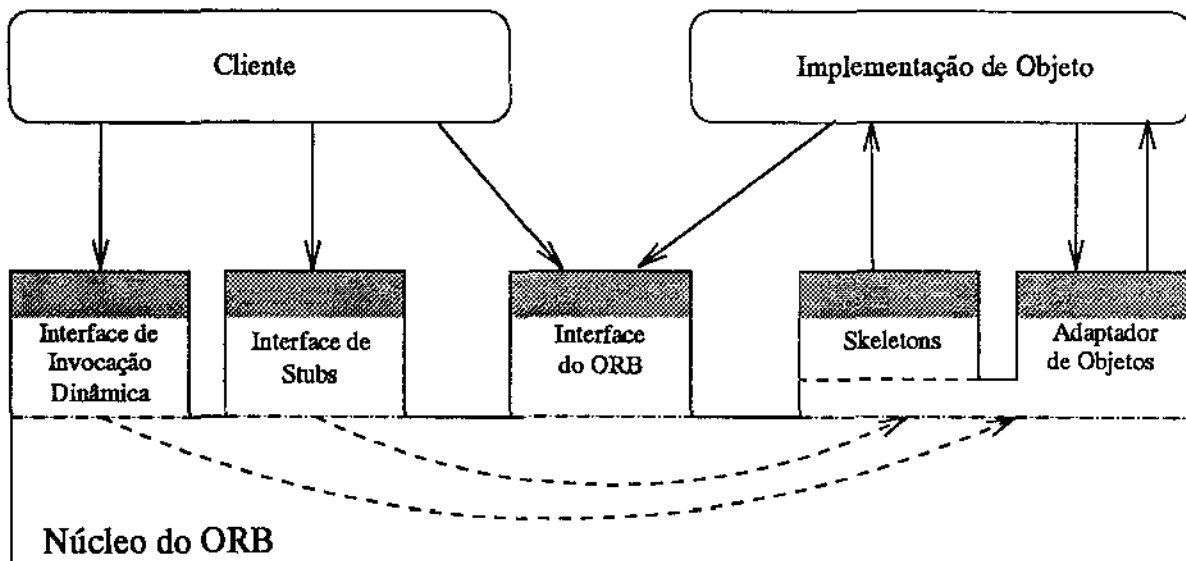


Figura 2.4: A estrutura das interfaces do Object Request Broker

O modelo de objetos da CORBA separa os aspectos de interface dos aspectos de implementação dos objetos. As *interfaces* descrevem o comportamento dos objetos conforme observado pelos clientes, e são especificadas em uma *linguagem de definição de interfaces* (IDL) padrão, descrita em [OMG91]. Uma interface pode ter várias implementações, que definem formas diferentes de prover os serviços especificados na interface. Para os clientes, entretanto, não é possível distinguir qual implementação da interface está sendo usada. Desta forma, a especificação da CORBA identifica o conceito de *implementação de objeto*, que estabelece a semântica de um objeto, através da definição das estruturas de dados para o objeto, juntamente com o código para as operações de sua interface. Uma implementação de objeto pode utilizar (como cliente) serviços de outras implementações de objeto. A partir de uma dada implementação de objeto, podem ser instanciados vários objetos provedores de serviços. Em um particular ambiente de sistema operacional, uma implementação de objeto é representada através do conceito de *servidor*, que representa a imagem executável da implementação de objeto (um servidor pode corresponder a um processo em um ambiente UNIX, por exemplo). Vários objetos, instanciados a partir de uma dada implementação de objeto, podem ser localizados em um servidor.

Para realizar uma invocação, um cliente tem duas alternativas. A primeira consiste em utilizar a

interface de *Stubs*, gerada estaticamente para cada interface definida em IDL. A segunda alternativa consiste em usar a Interface de Invocação Dinâmica (DII - *Dynamic Invocation Interface*), que é genérica para todas as interfaces definidas em IDL (devendo ser usada quando a interface do objeto a ser invocado somente é conhecida em tempo de execução), mas implica em que o cliente precisa construir explicitamente a invocação. Em ambos os casos, o cliente deve identificar o particular objeto invocado, o que é feito através de uma *referência de objeto*.

O ORB é o responsável por interpretar esta referência de objeto associada, de forma a determinar a implementação de objeto para a qual direcionar a invocação.

Do lado da implementação de objeto, o Adaptador de Objetos define (a partir da referência de objeto) qual o particular objeto que receberá a invocação. A invocação é então direcionada para o *Skeleton* apropriado (o qual foi gerado estaticamente a partir da definição da interface do objeto em IDL). O *Skeleton* então se encarrega de invocar o método apropriado do objeto.

A interface do ORB pode ser usada pelos clientes e implementações de objetos para obter serviços básicos do ORB, notadamente para a manipulação de referências de objetos. A interface do Adaptador de Objetos, por outro lado, é acessível às implementações de objetos para obter serviços como a ativação e desativação de objetos, para obter e manipular informações sobre implementações de objetos, e para a autenticação de clientes. A especificação CORBA prevê a co-existência de adaptadores de objetos com características e serviços diferentes, apropriados para cada categoria de aplicações. Em particular, é especificado um **Adaptador de Objetos Básico** (BOA - *Basic Object Adaptor*), que provê os serviços básicos necessários para as classes mais comuns de aplicações.

## A Linguagem de Definição de Interfaces - IDL

A linguagem IDL constitui um elemento fundamental da CORBA. Todas as interfaces de objetos provedores de serviços devem ser definidas em IDL, de forma que possam ser publicadas para a comunidade de clientes interessados. Através da definição em IDL, os clientes podem saber quais as operações são providas por um objeto, bem como quais os tipos dos parâmetros e valores de retorno destas operações. A IDL, contudo, é uma linguagem que permite apenas definições, sendo que a implementação dos objetos provedores de serviços e dos clientes deve ser realizada em uma linguagem de implementação, para a qual exista um mapeamento a partir da IDL. Um compilador é responsável por traduzir as definições IDL em definições na linguagem de implementação, de forma que possam ser utilizadas tanto por clientes quanto pelas implementações dos objetos. Clientes e implementações de objetos podem ser construídos em linguagens diferentes, desde que estas possuam um mapeamento a partir da IDL. Encontram-se definidos, ou em processo de adoção, mapeamentos de IDL para as linguagens C [OMG91], C++ [OMG94b], Ada e Smalltalk, entre outras.

A IDL da CORBA possui um conjunto de tipos de dados primitivos e oferece construções para a definição de tipos estruturados. A partir destes tipos, é possível definir a estrutura (parâmetros e valores de retorno) das operações acessíveis através de uma interface. Também podem ser definidos os atributos de uma interface. A IDL é uma linguagem orientada a objetos, o que permite a derivação de novas interfaces a partir de interfaces existentes, através do mecanismo de herança



de interfaces. Em IDL, todas as interfaces herdam a definição de uma interface raiz, denominada *Object*, a qual oferece operações e atributos comuns para permitir a manipulação uniforme de qualquer objeto instanciado na CORBA.

Por definição, um objeto em CORBA pode ser instanciado apenas a partir de uma única interface. Entretanto, o mecanismo de herança pode ser usado para contornar esta limitação, permitindo que um objeto seja implementado a partir de uma interface completa, derivada (por herança múltipla) a partir de interfaces parciais, as quais são publicadas para os clientes (um exemplo do uso deste recurso é mostrado no capítulo 6).

### *Stubs e Skeletons*

A partir da definição em IDL da interface de um objeto e com base no mapeamento de IDL para a linguagem ou linguagens de implementação, o compilador IDL gera as interfaces de *stubs* e *skeletons*.

Um *stub* representa, para os clientes, os objetos provedores de serviço correspondentes a uma determinada interface definida em IDL. O cliente realiza invocações ao *stub* e dele recebe os resultados, como se estivesse interagindo diretamente com o objeto alvo da invocação. O *stub* se encarrega de converter as invocações do cliente em chamadas ao ORB (que podem ser dependentes de implementação), de forma que este possa transferi-las, sob a forma de mensagens, através da rede. O funcionamento da interface de invocação dinâmica (DII) é semelhante, exceto que grande parte do trabalho de construir as estruturas de invocação é realizado, em tempo de execução, pelo próprio cliente.

Os *skeletons*, por outro lado, atuam como representantes dos clientes para as implementações de objetos. O *skeleton* recebe invocações que chegam através do Adaptador de Objetos (o qual, por sua vez as recebe do ORB) e as converte em invocações dos métodos apropriados na implementação do objeto. Os resultados da invocação são retornados, pela implementação de objeto, ao *skeleton*, como se este fosse o próprio cliente. Os resultados são enviados pelo *skeleton*, através do ORB, que os entrega ao *stub* original e este, por sua vez, entrega os resultados ao cliente.

O retorno produzido por uma invocação de objeto na CORBA compreende, normalmente, duas partes: um resultado de uso específico da aplicação, gerado pelo método invocado na implementação de objeto, e um resultado de exceção, gerado pelo ORB ou pela implementação de objeto para indicar a ocorrência de condições anormais na transmissão ou execução de uma invocação. A CORBA prevê um conjunto bem definido de exceções para sinalizar falhas que ocorram no escopo do ORB; outras exceções podem ser definidas pelas implementações de objetos.

Observe que existe, em geral, uma correspondência de um para um entre *stubs* e *skeletons*. Para cada operação definida na interface de um objeto em IDL, existe uma operação correspondente no *stub*, para receber as invocações dos clientes, e outra, no *skeleton*, para colocar as invocações em um formato apropriado e invocar a respectiva operação (método) na implementação do objeto. *Stubs* e *skeletons* podem ser gerados em linguagens diferentes, dependendo das linguagens utilizadas para implementação dos clientes e objetos servidores.

## Os Repositórios de Interfaces e Implementações

As definições de interface feitas em IDL podem ser armazenadas em um *repositório de interfaces*, de forma que fiquem acessíveis, em tempo de execução, para os clientes. O repositório de interfaces é um objeto, com interface definida em IDL, contendo operações que permitem aos clientes obterem a informação necessária para construir invocações dinâmicas (a serem realizadas através da Interface de Invocação Dinâmica).

O *repositório de implementações*, por sua vez, é utilizado para manter informações sobre as implementações de objetos, as quais são utilizadas pelo ORB para localizar e ativar os objetos servidores. As informações mantidas no repositório de implementações também são disponíveis para uso das implementações de objetos

## 2.3 As Plataformas DCE e ANSAware

Nesta seção, observamos as características gerais de outras duas plataformas de processamento distribuído comercialmente disponíveis.

A plataforma DCE (*Distributed Computing Environment*) da OSF [OSF90] se destaca como uma das primeiras iniciativas bem sucedidas em processamento distribuído heterogêneo. Ela consiste de um conjunto de serviços integrados para o suporte às atividades de desenvolvimento, manutenção e uso de aplicações distribuídas, independentemente de redes ou sistemas operacionais em particular. A arquitetura da DCE obedece uma estrutura em camadas, onde os níveis inferiores oferecem os serviços básicos, enquanto que os níveis superiores são consumidores destes serviços (aplicações). Estes serviços podem ser agrupados em duas categorias:

- Serviços fundamentais, que oferecem ferramentas para a programação de serviços de computação distribuída para o usuário final. São eles os serviços de RPC (chamada de procedimento remoto), serviços de nomes, serviços de temporização, serviços de segurança e serviços de *threads*.
- Serviços de Compartilhamento de Dados, que oferecem aos usuários finais ferramentas construídas sobre os serviços fundamentais. São eles os serviços de arquivos distribuídos, de diretório global, de suporte a estações sem disco, e de integração de PCs.

Apesar de prover considerável suporte para processamento distribuído, a plataforma DCE não satisfaz todos os requisitos de funcionalidade de sistemas ODP [BKR93]. Além disso, as abstrações de modelagem usadas não suportam orientação a objetos, o que gera certas incompatibilidades com o modelo ODP.

Por outro lado, os conceitos sobre os quais foi construída a plataforma ANSA (*Advanced Networked Systems Architecture*) [Lin93] estão intimamente relacionados com aqueles presentes no modelo ODP. Sua própria definição é feita com referência aos cinco pontos de vista do RM-ODP, o que a torna um ambiente de suporte próprio para construção de sistemas baseados naquele modelo. ANSA segue uma visão baseada em linguagem de programação, que procura colocar uma distinção clara entre programação de sistemas de suporte e programação de aplicações, oferecendo, neste

segundo caso, um modelo simples e transparente (com relação aos aspectos de distribuição) para o programador.

## 2.4 A Plataforma Multiware

Encontra-se atualmente em desenvolvimento na UNICAMP, em conjunto com a UNESP, o projeto Multiware, cujo objetivo básico é a especificação e implementação de uma arquitetura de suporte a aplicações em um ambiente de processamento distribuído aberto [LMM<sup>+</sup>94, Mad95]. A especificação funcional da Plataforma Multiware segue os princípios do RM-ODP, incorporando seus conceitos e funcionalidades, e definindo uma estrutura para a implementação das funções ODP. A abordagem consiste em utilizar plataformas comercialmente disponíveis (como ANSAware, DCE e ORB), integrando os serviços providos por estas plataformas de maneira uniforme para o programador de aplicações. Funcionalidades previstas no RM-ODP, mas não suportadas por estas plataformas, são adicionadas ao ambiente, de forma a prover o suporte necessário às aplicações distribuídas.

A plataforma Multiware está estruturada, conforme mostra a Figura 2.5, em três camadas:

- camada de *hardware* e *software* de processamento e comunicação, que compreende os sistemas operacionais e protocolos de comunicação utilizados; esta camada não provê as funcionalidades necessárias para o suporte a sistemas distribuídos abertos;
- camada Middleware, que incorpora as plataformas comerciais e os demais serviços de suporte a ODP;
- camada Groupware, que oferece funcionalidades de suporte para diferentes classes de aplicações, notadamente aplicações de trabalho cooperativo suportado por computador (CSCW).

As aplicações utilizam os serviços da plataforma Multiware através da camada de Groupware ou diretamente, dependendo da necessidades de refinamento dos serviços oferecidos.

### 2.4.1 A Camada Middleware

Esta camada é a responsável por prover o ambiente de processamento distribuído aberto, com serviços de propósito geral, que podem ser usados por todo tipo de aplicação. A estrutura da camada Middleware é constituída por duas sub-camadas (verticalmente dispostas): a sub-camada de Processamento Multimídia e a sub-camada ODP.

A sub-camada de Processamento Multimídia oferece funcionalidades para o transporte de informações de áudio e vídeo com suporte de tempo real para comunicação e sincronização de objetos multimídia. São providos serviços para a criação de canais multimídia do tipo “um-para-muitos” (para permitir o fluxo de informações multimídia de um objeto produtor para vários objetos consumidores), com garantia das qualidades de serviço.

A sub-camada ODP, descrita em [MM94], está estruturada em dois níveis, conforme mostra a Figura 2.6: um nível inferior, composto de plataformas comerciais de processamento distribuído, e o nível superior, de Funções ODP.

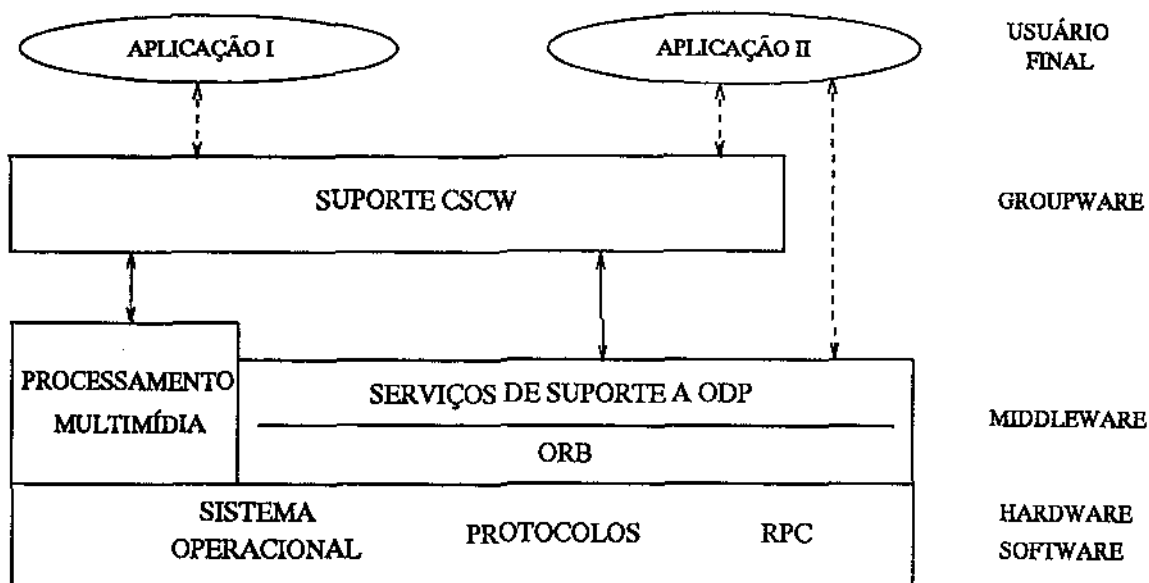


Figura 2.5: Arquitetura da Plataforma Multiware

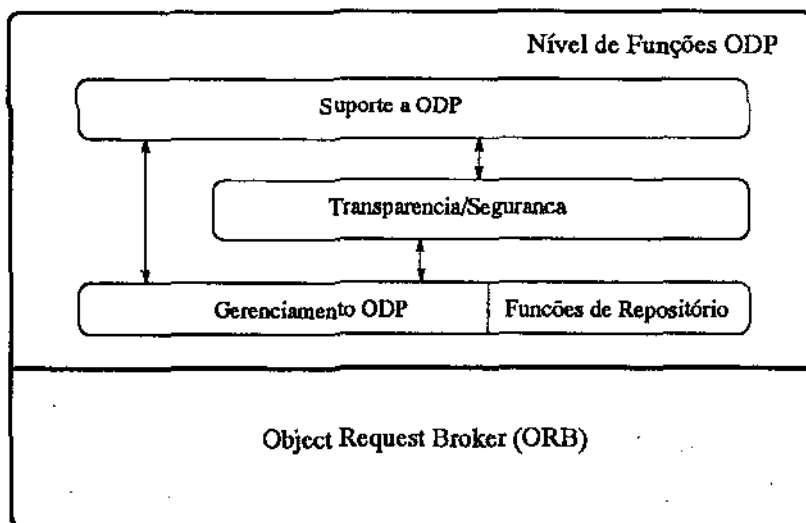


Figura 2.6: Estrutura da sub-camada ODP da camada Middleware

As plataformas que constituem o nível inferior oferecem os serviços básicos de processamento distribuído. É previsto o uso das seguintes plataformas: ANSAware, DCE e ORB, sendo que os trabalhos de implementação atualmente em desenvolvimento se restringem ao uso da plataforma ORB.

O nível de Funções ODP procura prover um conjunto de serviços independente das plataformas do nível inferior, com interfaces de acesso uniformes para os usuários. Isto implica na necessidade de integrar serviços possivelmente diferentes, providos no nível das plataformas, e adicionar serviços complementares para prover às aplicações um ambiente ODP.

Este nível foi estruturado em três sub-níveis, conforme mostrado na Figura 2.6, de maneira a organizar as funcionalidades ODP segundo seu uso e as interações entre os componentes provedores destas funcionalidades:

- Subnível de Gerenciamento ODP – oferece as funções básicas de gerenciamento, permitindo a criação e utilização dos componentes de engenharia do sistema ODP. O suporte às funções de gerenciamento é auxiliado pelas funções de repositório da especificação ODP.
- Subnível de Transparência/Segurança – oferece as transparências e as funções de segurança previstas na especificação ODP, facilitando o uso das funções de gerenciamento. A provisão destas facilidades deve ser flexível para se ajustar aos diferentes requisitos de transparência e segurança das diversas aplicações.
- Subnível de Suporte a ODP – provê às aplicações um conjunto expressivo de funcionalidades ODP (notadamente as funções de coordenação do RM-ODP) para uso direto pela camada superior da plataforma. Os componentes deste subnível podem utilizar os serviços de gerenciamento do subnível inferior de maneira direta ou através do subnível de transparência/segurança, dependendo da necessidade ou não de transparência ou segurança para as interações.<sup>3</sup>

O subnível de Suporte a ODP é composto de blocos funcionais básicos, entre os quais:

- **Suporte a Aplicações:** é responsável pela estruturação de aplicações distribuídas a partir de especificações fornecidas pelos usuários (normalmente no ponto de vista de computação). A estruturação das aplicações é realizada em termos dos conceitos e construções do ponto de vista de engenharia. As aplicações são estruturadas através da definição e instanciação de objetos do ponto de vista de engenharia, que realizam as funções específicas de aplicação, e da organização destes objetos em subsistemas, incluindo requisitos de transparência e segurança, bem como o uso de objetos de suporte.
- **Trader:** permite a solicitação de informações sobre serviços oferecidos por objetos servidores (exportadores de serviços) por parte dos potenciais clientes destes serviços (importadores de serviços); o *trader* permite aos clientes especificarem as características dos serviços desejados,

---

<sup>3</sup>Na primeira versão da plataforma Multiware, entretanto, as funções do subnível de suporte a ODP podem utilizar diretamente os recursos do nível de plataformas comerciais, no caso, a plataforma ORB

retornando, em seguida a identificação dos servidores em conformidade com as especificações solicitadas.

- **Suporte a Transações:** garante que a execução de operações transacionais obedeça às propriedades usuais de transações (atomicidade, consistência, isolamento e durabilidade).
- **Suporte a Grupos:** provê serviços de suporte à interação cooperativa entre objetos membros de um grupo. A definição dos serviços providos por este componente e sua implementação constitui o objetivo básico deste trabalho.

### 2.4.2 A Camada Groupware

Esta camada provê funcionalidades de ambientes cooperativos para a construção de aplicações que envolvem múltiplos usuários autônomos e distribuídos, trabalhando cooperativamente sobre informações compartilhadas. São oferecidos serviços genéricos de suporte à cooperação para várias classes de aplicações, incluindo co-editoração de documentos, tele-conferência e sistemas de apoio a decisão em grupo. A descrição dos componentes e serviços da camada Groupware é apresentada em [LFSM94].

Todas estas funcionalidades da camada Groupware são compostas a partir do uso dos serviços da camada Middleware, notadamente, do nível de Suporte a ODP.

## Capítulo 3

# Grupos em Sistemas Distribuídos

Aplicações distribuídas que envolvem algum tipo de cooperação entre seus múltiplos componentes, em geral, necessitam de abstrações de comunicação mais poderosas que o modelo tradicional de comunicação em sistemas distribuídos, em que apenas duas entidades se comunicam entre si. Isto se deve ao fato de que a execução de tarefas cooperativas geralmente requer a participação de vários componentes da aplicação em uma mesma interação. A complexidade destas interações “multi-ponto” é consideravelmente maior que no modelo tradicional “ponto-a-ponto”, em virtude da necessidade de coordenar as atividades dos participantes das interações, mantendo a consistência entre as sucessivas interações. Com isto, a programação de aplicações cooperativas torna-se excessivamente complicada na ausência de algum tipo de suporte apropriado.

Grupos constituem uma abstração útil neste caso, pois permitem que uma única ação de comunicação envolva um conjunto qualquer de participantes. Desta forma, uma aplicação cooperativa é estruturada como um grupo cujos membros são os componentes cooperantes da aplicação. Facilidades para a comunicação de grupo e coordenação das interações cooperativas são providas como parte do suporte às aplicações. A presença de uma abstração de grupos no nível de suporte a aplicações (em uma plataforma de sistemas distribuídos, por exemplo), permite a existência de uma solução uniforme, que pode ser utilizada por aplicações diferentes, tornando transparentes as questões envolvidas nas interações de grupo [BCG91].

A maioria das abordagens existentes emprega alguma forma da abstração de grupo para estruturar aplicações que envolvem replicação de informação ou de processamento (para tolerância a falhas e aumento do desempenho, por exemplo). Grupos constituem uma forma natural de garantir a consistência entre as réplicas. Entretanto, outros tipos de aplicações podem igualmente se beneficiar da existência de uma abstração de grupos, notadamente aplicações de CSCW (*Computer Supported Cooperative Work*), onde a cooperação entre os usuários é realizada através de comunicação de grupo [SST94].

A consistência dos mecanismos de grupo depende de um tratamento adequado, no nível de suporte, de aspectos como a confiabilidade das interações de grupo, a ordenação das interações, a

manutenção do conjunto de membros de grupos e a transparência das interações de grupo. Cada tipo de aplicação apresenta requisitos diferentes com respeito a cada um destes aspectos, sendo que um modelo de grupos pode optar por servir a um tipo específico de aplicações ou oferecer alternativas quanto a estes aspectos.

Este capítulo está estruturado como se segue. A seção 3.1 apresenta o conceito de grupos de objetos ou grupos de processos, juntamente com uma classificação de grupos segundo vários aspectos. A seção 3.2 descreve a abordagem de grupos de objetos introduzida no Modelo de Referência para Processamento Distribuído Aberto (RM-ODP), sendo proposta uma interpretação para a Função de Grupos do Modelo, segundo conceitos encontrados na literatura. Outras abordagens para o problema de suporte a grupos, relacionadas com trabalhos significativos em sistemas distribuídos, são mostradas na seção 3.3.

### 3.1 O Conceito de Grupos

A literatura define *grupos de processos* como “conjuntos de processos que atuam de maneira cooperativa para prover algum serviço comum” [CZ85, BJ87b]. *Grupos de objetos* são definidos de maneira semelhante, diferindo apenas pelo fato de que, neste caso, os membros de um grupo encapsulam informações de estado além de comportamento [LCN90]. Os membros de um grupo geralmente apresentam algumas características em comum, e seu agrupamento permite que eles sejam tratados pelas aplicações, através do *identificador do grupo*, como se constituíssem uma única entidade, com interface e funcionalidades uniformes.

A cooperação entre os objetos (ou processos) membros de um grupo exige que eles interajam com frequência entre si de modo a coordenar suas atividades. A interação entre eles deve ser suportada através de mecanismos de comunicação “multi-ponto”, permitindo o envio de uma mensagem para os vários membros através de uma única ação, bem como a recepção, em um único evento, de múltiplas mensagens relacionadas, enviadas por vários membros diferentes. Estes estilos de comunicação de grupo são indicados na Figura 3.1, que mostra também um caso de interação do tipo “muitos-para-muitos”, que pode ocorrer, por exemplo, na comunicação entre dois grupos distintos. Estes mecanismos podem ser construídos diretamente sobre primitivas de comunicação da rede, através do mapeamento dos identificadores de grupos sobre endereços de *multicast*, podendo também ser simulados usando comunicação “ponto-a-ponto”.

Os mecanismos de comunicação de grupo podem ser apresentados às aplicações sob a forma de primitivas semelhantes às aquelas usadas para comunicação “ponto-a-ponto”. Dois estilos de comunicação entre componentes de um sistema distribuído são comuns: troca de mensagens e chamada de procedimento remoto (RPC). No primeiro, as primitivas de envio de mensagens devem ser estendidas para interpretar um identificador de grupo como o destino das mensagens, de forma a enviar as mensagens para todos os membros do grupo [CZ85, BC90]. Igualmente, o recebimento de mensagens deve ser feito através de primitivas que permitam receber mensagens de um grupo (de qualquer um de seus membros ou de todos eles).

Por outro lado, a implementação de comunicação de grupo no estilo de chamada de procedimento remoto permite às aplicações invocarem serviços na interface de um grupo, sendo a invocação trans-



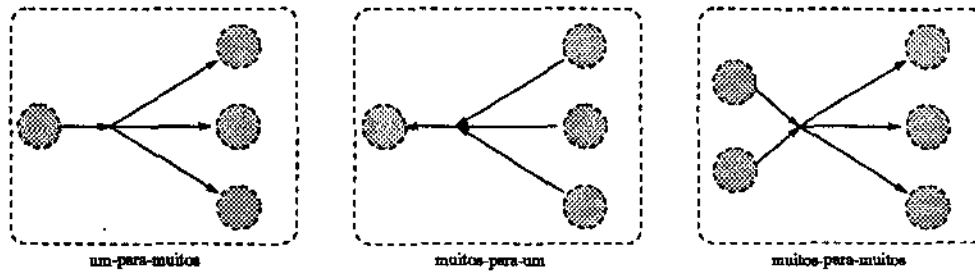


Figura 3.1: Comunicação de grupo

parentemente distribuída para os membros do grupo [Par92b, Maf93]. Isto implica na existência de um mecanismo específico para o tratamento das múltiplas respostas geradas por cada membro do grupo, as quais devem, idealmente, ser combinadas e entregues à aplicação como uma resposta única.

Grupos são estruturas dinâmicas, o que implica que a todo instante pode ocorrer a entrada de novos membros, bem como a saída de membros existentes. Desta forma, além dos mecanismos de comunicação, são necessários recursos de estruturação que permitam relacionar corretamente os membros de um grupo, mantendo a estrutura sempre atualizada e consistente (de forma que todos os membros tenham sempre a mesma visão do grupo).

A motivação para o uso de grupos de objetos decorre do fato de que várias categorias de aplicações distribuídas apresentam características ou requisitos que podem ser modelados naturalmente através da estruturação em grupo dos componentes da aplicação, facilitando a cooperação entre eles. Alguns dos principais usos de grupos de objetos estão relacionados com aplicações que envolvem:

- *Publicação de informação*<sup>1</sup> para múltiplas entidades distribuídas através de uma rede; enquadram-se nesta categoria aplicações que requerem a colaboração, através da troca e compartilhamento de informação, entre múltiplos usuários ou agentes que realizam uma tarefa comum (como aplicações de CSCW);
- *Paralelismo e compartilhamento de carga*, onde ocorre a distribuição das atividades de processamento entre os vários objetos membros de um grupo servidor, o qual é observado pelos clientes como se fosse um único objeto; a abstração de grupos, neste caso, possibilita uma forma consistente e efetiva de dividir as tarefas entre os membros de um grupo e coordenar a atividade dos membros;
- *Replicação e tolerância a falhas*, onde recursos que necessitam de elevada disponibilidade podem ser replicados entre os membros de um grupo; neste caso, mesmo em presença de falhas o recurso pode ainda estar disponível; a manutenção correta das réplicas é facilitada pelo uso de comunicação de grupo confiável;

<sup>1</sup>Tradução do termo em inglês *data publication*

- *Monitoramento e gerenciamento distribuído*, onde grupos constituem uma forma conveniente de coletar e distribuir informação de gerenciamento em um sistema distribuído;
- *Encapsulamento de estado* compartilhado pelos múltiplos membros de um grupo, ocultando as interações entre eles (bem como as informações manipuladas internamente), e permitindo a exportação de uma interface única, através da qual clientes do grupo podem invocar operações de serviço bem definidas.

### 3.1.1 Classificações de Grupos

Com respeito a interações, um grupo é dito *fechado* quando apenas seus membros podem participar das interações de grupo. Por outro lado, quando usuários externos ao grupo podem interagir diretamente com ele, o grupo é dito *aberto*. A Figura 3.2 ilustra estes dois conceitos.

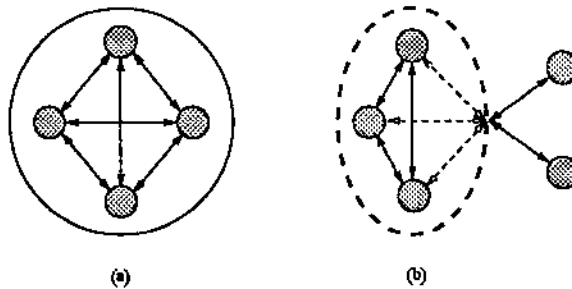


Figura 3.2: Grupos fechados(a) e abertos(b) com respeito às interações de grupo

Quanto ao uso, grupos são classificados como [Bir93]: *grupos anônimos*, quando não se conhece (ou não se tem controle sobre) o conjunto de membros de um grupo, sendo um caso típico de aplicações como grupos de discussão baseados em mensagens (*Newsgroups*); *grupos explícitos*, quando existe um controle sobre os membros que entram e saem do grupo, podendo a informação sobre o conjunto de membros ser utilizada pelos membros em algum processamento.

De acordo com [LCN90], quatro tipos básicos de grupos podem ser definidos, tendo como base o relacionamento entre as operações e dados mantidos pelos membros de um grupo:

- *Grupos com dados e operações homogêneos*: cada membro do grupo mantém as mesmas informações (o estado do grupo é replicado entre seus membros) e implementa as mesmas operações para sua manipulação. A manutenção da consistência entre as réplicas de informação requer que cada membro execute exatamente a mesma seqüência de operações. Exemplos de aplicações: aumento da disponibilidade de acesso e confiabilidade com relação a recursos críticos.
- *Grupos com operações homogêneas*: em grupos deste tipo, o conjunto de informações é particionado entre os membros, com cada membro suportando um conjunto de operações idêntico sobre a partição dos dados por ele mantidos. Com isto, quando uma operação é invocada

sobre um grupo, apenas os membros que mantêm a informação relevante precisam executar a operação. Este tipo de grupo é normalmente utilizado para compartilhamento de carga de processamento entre os componentes de um sistema distribuído.

- *Grupos com dados homogêneos*: o conjunto de informações mantido por cada membro é idêntico. Entretanto, as operações executadas por cada um dos membros sobre estas informações podem ser diferentes entre si.
- *Grupos heterogêneos*: as informações e operações mantidas por cada membro do grupo podem ser distintas entre si. Os membros são independentes para processar as informações, podendo cooperar na realização de tarefas comuns. Aplicações de publicação de informação (conferência computadorizada e grupos de discussão, por exemplo) são típicas de grupos heterogêneos.

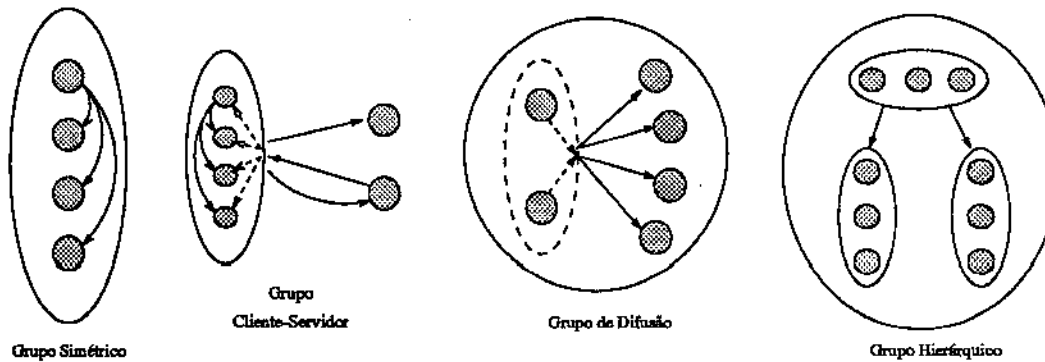


Figura 3.3: Diferentes estilos de interação em grupo

Outra classificação ortogonal à anterior, e que diz respeito à estrutura de grupos de objetos é apresentada em [Bir93]. Grupos são então divididos em quatro categorias, ilustradas na Figura 3.3, com base nos estilos de interação entre seus membros:

- *Grupos simétricos*<sup>2</sup>: um grupo é considerado simétrico se cada um de seus membros tem capacidades semelhantes de interação, e atuam com forte cooperação entre si; qualquer membro pode participar de todas as interações do grupo.
- *Grupos cliente-servidor*: parte dos membros do grupo (os servidores) oferecem serviços para os demais membros (os clientes), ou para usuários externos ao grupo; normalmente, os membros servidores cooperam entre si para oferecer os serviços.
- *Grupos de difusão*: um ou mais membros do grupo enviam mensagens para os demais membros, os quais atuam como receptores passivos de informação.

<sup>2</sup>Tradução do termo em inglês "peer groups"

- *Grupos hierárquicos*: são constituídos de subgrupos estruturados em árvore. A interface externa de serviços do grupo normalmente é implementada pelo grupo raiz, o qual pode dividir as tarefas entre os subgrupos.

Quanto ao comportamento dos membros de um grupo, [LCN90] define duas classes principais de grupos: determinísticos e não-determinísticos. Um grupo é *determinístico* se todos os seus membros devem receber e atuar uniformemente sobre cada interação de grupo. A atividade dos membros deve ser coordenada e sincronizada, de forma a resultar em um comportamento global consistente. Por outro lado, em grupos *não-determinísticos* cada membro pode atuar de maneira diferente sobre as mesmas interações, podendo também não atuar de forma alguma. Este tipo de grupo não requer coordenação entre seus membros, podendo ser tolerado certo nível de inconsistência entre eles.

A proposta de um modelo de grupos envolve a escolha de alternativas para cada um destes aspectos, as quais dependem do tipo de aplicações a serem suportadas. No capítulo seguinte, é apresentado o modelo de grupos de objetos proposto para o suporte a aplicações de trabalho cooperativo. Na descrição do modelo são indicadas as alternativas adotadas, de acordo com os requisitos desta classe de aplicações.

## 3.2 A Função de Grupos do RM-ODP

O conceito de *grupo*, no RM-ODP, tem uma definição genérica que permite a especificação de diferentes tipos de grupos para diferentes classes de aplicações [ISO94a]. Em particular, o conceito pode ser especializado para a definição de grupos como estrutura de suporte a aplicações cooperativas. Os aspectos fundamentais para o suporte a grupos no RM-ODP são contemplados através da *Função de Grupos*, uma das funções de coordenação do RM-ODP [ISO94c]. Seu objetivo consiste em coordenar as interações entre objetos membros de um grupo, os quais atuam de forma cooperativa entre si.

O suporte fornecido pela Função de Grupos é realizado através de quatro funcionalidades básicas, relacionadas com os aspectos fundamentais de coordenação de grupos: **interação**, **colagem de interações**, **ordenação de eventos** e **controle de membros**. Um ambiente de suporte a aplicações cooperativas em conformidade com o RM-ODP deve, portanto, implementar estas funcionalidades básicas, fornecendo uma interface adequada para sua utilização. Entretanto, a especificação da função de grupos, tal como encontrada no RM-ODP, apresenta apenas as linhas gerais destas funcionalidades. Isto implica em que uma implementação real deva ser feita com base em uma interpretação própria da Função de Grupos, de acordo com as necessidades específicas a serem satisfeitas, mantendo, ainda, a conformidade com o RM-ODP. Por outro lado, a definição da Função de Grupos permite um elevado grau de flexibilidade na especificação de modelos de suporte a grupos adequados a cada tipo de aplicação. Esta flexibilidade é reforçada pelo uso de *políticas*, que especificam formas alternativas de controle para cada uma das funcionalidades de suporte a grupos.

As seções seguintes apresentam cada uma das funcionalidades especificadas para a função de grupos, com sua descrição original (extraída do RM-ODP), juntamente com a interpretação aqui

adotada para cada uma delas. A partir da definição da Função de Grupos do RM-ODP, é feita uma análise de cada uma de suas funcionalidades, com base em conceitos bem definidos, encontrados na literatura sobre grupos em sistemas distribuídos.

### 3.2.1 Interação

“Determinação de quais membros de um grupo participam em quais interações, de acordo com uma política de interação.”

Esta funcionalidade tem como objetivo permitir aos membros de um grupo interagirem de maneira transparente em relação aos aspectos da comunicação multi-ponto, possibilitando aos membros abstraírem-se da constituição do grupo, de maneira a não precisarem endereçar as interações a cada membro em particular.

A Figura 3.4 ilustra o estilo de interação em grupo, mostrando a participação de um elemento de suporte a interações, responsável por distribuir as interações para o grupo de maneira transparente para os seus membros.

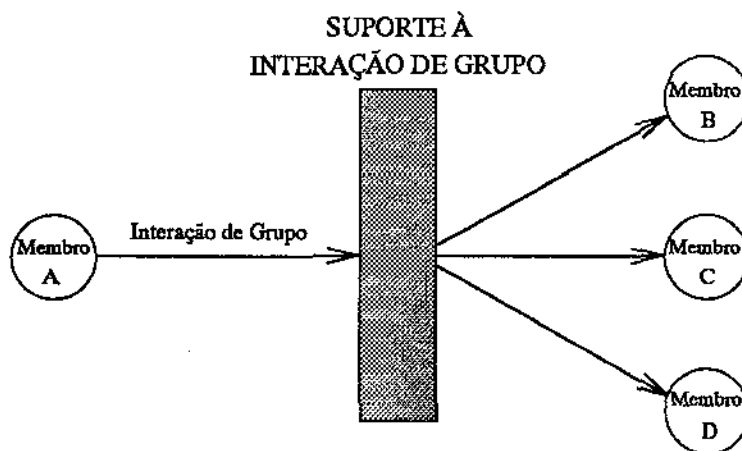


Figura 3.4: Interação de grupo

Interações, no Modelo Prescritivo do RM-ODP [ISO94b], são definidas como quaisquer ações envolvendo um objeto e seu ambiente (que pode ser constituído de outros objetos). Para o propósito de suporte a grupos, uma *interação de grupo* será considerada como qualquer ação de comunicação envolvendo os membros (objetos) de um grupo.

No modelo computacional do RM-ODP [ISO94c], dois estilos básicos de interação são possíveis entre os objetos de um sistema (lá chamados de objetos computacionais):

- *Interações Operacionais*, que consistem na invocação de operações na interface dos objetos para solicitar a realização de serviços ou para a transferência de informações. Operações são classificadas em dois tipos:

- **Interrogações:** interações que envolvem um relacionamento *cliente/servidor* entre os participantes, onde um objeto (cliente) solicita um serviço ou informação a outro objeto (servidor), o que implica em comunicação nos dois sentidos, de forma a possibilitar o retorno de **respostas** por parte do objeto servidor.
  - **Anúncios:** interações envolvendo comunicação em um único sentido, do objeto fonte para o objeto destino da interação; normalmente caracterizam a transferência de informação entre dois objetos.
- **Interações de Fluxo**, que permitem a transferência de informações de natureza contínua entre as interfaces de objetos produtores e consumidores de informação.

Em ambientes de suporte a grupos, ambos os estilos de interação são igualmente úteis. Interações operacionais são apropriadas para a invocação de serviços de grupo bem como para a comunicação de informações de natureza discreta entre os membros de um grupo. Por outro lado, interações de fluxo podem ser utilizadas, por exemplo, em aplicações que demandam interação multimídia entre os membros de um grupo (teleconferência computadorizada, por exemplo).

A comunicação entre objetos é suportada, do ponto de vista computacional do RM-ODP, através de *bindings*, que conectam dois ou mais objetos através de suas interfaces (ver capítulo 2), tornando a interação entre eles transparente quanto aos aspectos de distribuição. No caso de grupos de objetos, as interfaces dos objetos membros devem ser conectadas através de *bindings* multiponto, definidos no RM-ODP, permitindo a qualquer objeto do grupo interagir transparentemente com os demais, sem a necessidade de endereçar cada um deles. A Figura 3.5 ilustra este tipo de conexão entre objetos. Observe, entretanto, que o conceito de *binding* do RM-ODP não suporta a semântica de grupos cooperativos (em termos da coordenação das atividades dos membros). Neste sentido, o modelo de referência especifica a Função de Grupos, que pode ser construída sobre as funcionalidades de “baixo nível” do mecanismo de *bindings* multiponto.

Um aspecto importante relacionado à comunicação de grupo, de acordo com a definição da Função de Grupos diz respeito ao fato de que *interações não necessariamente envolvem todos os membros de um grupo*. Isto exige alguma forma de se determinar o destino das interações, seja explícita ou implicitamente, o que pode ser especificado através de *políticas de interação*. Os mecanismos para a realização destas políticas de interação podem ser suportados por capacidade semelhante de endereçamento disponível no nível dos *bindings* que conectam os membros do grupo.

Igualmente importante é a necessidade de se garantir a confiabilidade das interações. Este aspecto é especialmente complicado em se tratando de interações de grupo, uma vez que é necessário garantir que todos os membros que sejam alvo de uma dada interação realmente participem dela. Em outras palavras, é necessário garantir a propriedade de *atomicidade de interações* de grupo: “Ou todos os membros (não-falhos) endereçados recebem a interação ou nenhum deles a recebe” [JB89]. Caso esta propriedade não seja satisfeita, o estado do grupo pode se tornar inconsistente.

A atomicidade de interações é um problema que surge especificamente em situações de comunicação “multi-ponto”, podendo ser afetada em caso de falhas, sendo possíveis várias situações [BJ87b]:

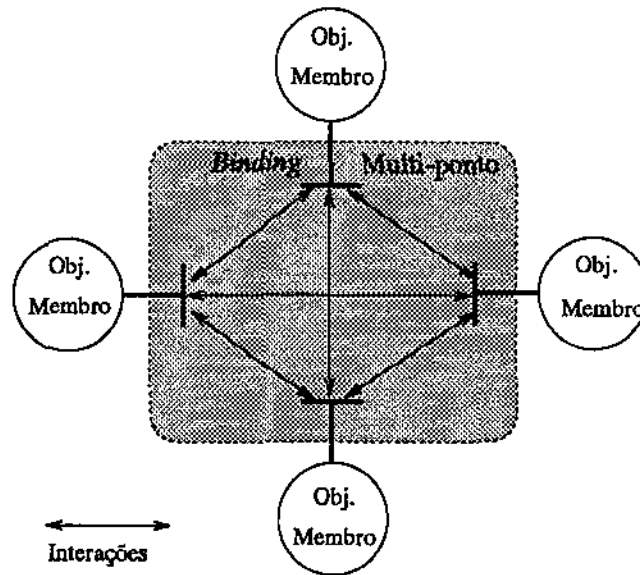


Figura 3.5: *Binding* multi-ponto conectando as interfaces dos objetos de um grupo

- falhas de comunicação: quando uma interação não atinge algum membro endereçado devido a perda das mensagens correspondentes; neste caso, após detectada a falha, as mensagens perdidas devem ser retransmitidas;
- falha de um dos membros alvo de uma interação: neste caso, o mais comum é remover o membro do grupo, sendo a propriedade de atomicidade mantida para os membros não afetados por falhas;
- falha do membro originador de uma interação: esta falha é especialmente grave nos casos em que acontece antes que o processo de distribuição da interação seja concluído, pois apenas alguns dos membros do grupo poderão ter recebido a interação, o que pode deixar o estado do grupo, conforme visto por cada um de seus membros, inconsistente.

O tratamento deste último caso requer soluções mais elaboradas, que envolvem algum protocolo para determinar quando uma interação teve sua entrega efetivada em todos os seus destinos, sendo que cópias das interações devem ser mantidas enquanto isto não acontece. Um protocolo simples que garante atomicidade de interações de grupo, porém a um custo razoável em termos de trocas de mensagens, é exposto em [JB89] e consiste em exigir que todo membro do grupo que recebe uma interação pela primeira vez, envie uma cópia da mesma para cada um dos membros destino. Desta forma, após retransmitir com sucesso uma interação, o membro pode imediatamente consumi-la (aplicando seus efeitos), com a garantia de que todos os outros membros não falhos também o farão. (O protocolo envolve também o descarte das interações repetidas devido às retransmissões).

Outra forma de se garantir atomicidade da comunicação de grupo é adicionando tolerância a falhas aos elementos do grupo que são fontes de interações. Isto pode ser feito através de técnicas

de replicação passiva (capítulo 5), onde cada elemento gerador de interações (que pode corresponder, por exemplo a um objeto membro de um grupo) tem uma réplica, a qual é mantida sempre atualizada, podendo assumir o seu lugar em caso de falhas.

### 3.2.2 Colagem de Interações

“Detecção e recuperação de falhas de interação, de acordo com uma política de colagem de interações.”

Em versões anteriores do RM-ODP, assim como na literatura sobre grupos em sistemas distribuídos, o conceito de *colagem* é descrito como a capacidade de combinar interações de grupo relacionadas [OE93, Isi93]. Como exemplo, tem-se o caso de múltiplas respostas originadas por uma invocação (interrogação) de grupo; tais respostas, por corresponderem a uma invocação comum, podem ser coladas e entregues como uma única resposta (a qual equivale ao conjunto de respostas originais e conduz toda a informação nelas expressa) enviada pelo grupo ao membro originador da invocação. A Figura 3.6 ilustra o conceito.

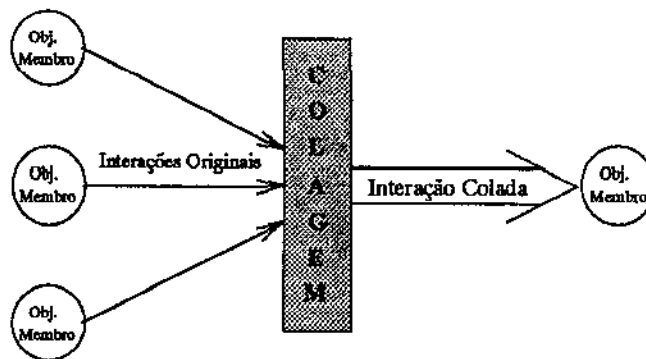


Figura 3.6: Colagem de interações

Este conceito pode efetivamente ser utilizado no tratamento de falhas de interação, conforme expresso na definição acima, fazendo-se com que uma interação válida seja obtida através da colagem das interações originais (relacionadas) que não sofreram falhas (as interações afetadas por falha são ignoradas).

Os mecanismos utilizados na colagem de interações são especificados por meio de *políticas de colagem*, as quais podem expressar vários níveis de tratamento semântico das interações a serem combinadas. Uma política pode, por exemplo, determinar apenas a concatenação das interações originais, enquanto, em outro extremo, outra política pode determinar a combinação do conteúdo real das interações. Entretanto, o segundo enfoque exige conhecimento no nível das aplicações, o que nem sempre é disponível em uma implementação da Função de Grupos.



### 3.2.3 Ordenação

“Para garantir que interações entre membros de um grupo sejam corretamente ordenadas, com respeito a uma política de ordenação.”

A atividade concorrente entre os múltiplos componentes de um sistema distribuído demanda algum tipo de protocolo ou mecanismo de coordenação, de forma a garantir a consistência das interações entre estes componentes e a correção das informações por eles mantidas. Além disso, em ambientes de grupos cooperativos, o número de eventos concorrentes aumenta consideravelmente devido às interações multi-ponto [Isi93], o que pode dificultar a manutenção do ambiente compartilhado pelos membros de um grupo.

A coordenação das interações pode ser realizada através de mecanismos que imponham certo sincronismo ao ambiente, de forma que haja uma relação de seqüência ou *ordem lógica* entre os eventos gerados concorrentemente. A ordenação de eventos distribuídos foi originalmente proposta por Lamport como uma ferramenta útil na implementação de sistemas distribuídos [Lam78] e tem sido largamente utilizado no contexto de grupos cooperativos [Bir93, KTV93, Isi93]. Nestes contextos, a implementação dos protocolos de cooperação torna-se substancialmente mais simples, em virtude do sincronismo virtual resultante da ordenação dos eventos.

A ordenação de eventos em um grupo permite que todas as interações entre seus membros sejam ordenadas com respeito a uma seqüência lógica de eventos, reconhecida no contexto global do grupo. Com isto, garante-se que os membros do grupo observarão todas as interações e reagirão a elas na mesma ordem. Desta forma, evitam-se situações onde membros diferentes observam duas interações em ordens diferentes: caso estas interações correspondam à atualização de algum recurso compartilhado, as visões de grupo observadas pelos membros tornam-se inconsistentes. A figura 3.7 ilustra dois cenários distintos. No cenário (a), as interações de grupo não são ordenadas, podendo atingir membros diferentes em ordens diferentes, devido a atrasos ou falhas na comunicação de mensagens na rede. No caso, B e C recebem as interações 1 e 2 em ordem inversa. Por outro lado, no cenário (b), é garantida uma ordem uniforme de recepção das interações. Nos diagramas, as linhas verticais representam membros de um grupo, enquanto que as setas representam interações entre eles; o tempo aumenta de cima para baixo. A ordenação de eventos, conforme observado na figura 3.7(b) pode implicar no retardo da entrega de interações aos membros de um grupo. Isto se faz necessário nos casos em que interações são recebidas fora de ordem (ou seja, antes do recebimento de interações precedentes, conforme a ordem global dos eventos).

Desta forma, um ambiente de suporte a grupos cooperativos deve controlar a atribuição de ordem às interações, garantindo também que esta ordem seja obedecida pelas aplicações. As diversas formas em que este suporte a ordenação pode ser provido devem ser expressas, conforme a definição da Função de Grupos, através de *políticas de ordenação*, de forma que as próprias aplicações possam definir qual o tipo de ordem mais apropriado. As principais abordagens encontradas na literatura para tratar ordenação de eventos são descritas a seguir.

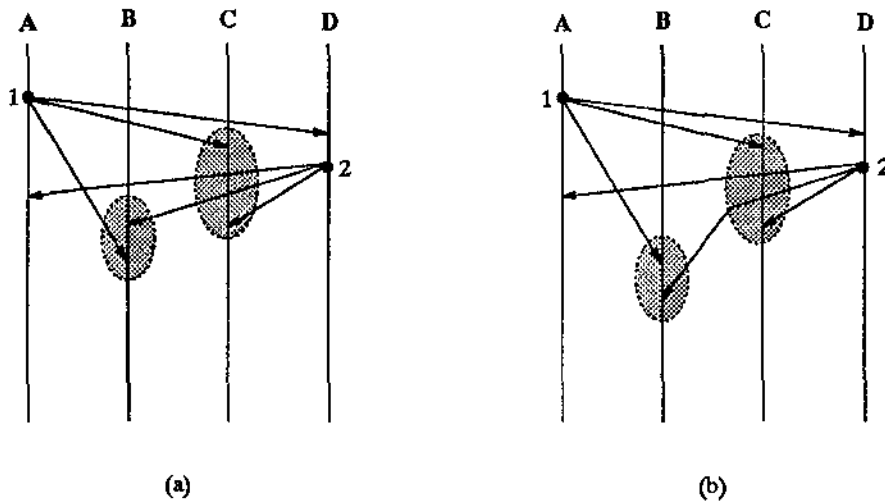


Figura 3.7: Ordenação de eventos de grupo: (a) cenário sem ordenação; (b) cenário com ordenação.

### Ausência de Ordenação

Nesta abordagem, interações são sempre entregues aos membros de grupos na ordem em que elas são recebidas, o que permite que elas sejam manipuladas por eles em ordens mutuamente diferentes. Este tratamento tem como base o argumento de que uma ordenação correta e eficiente das interações de grupo pode ser expressa apenas em termos do conteúdo das interações, o que somente é possível no nível das aplicações [CS93]. Este tratamento de ordenação, contudo, pode tornar o processo de construção de aplicações consideravelmente mais complexo, em virtude da necessidade de lidar com aspectos distintos do objetivo real das aplicações. Esta abordagem é adotada no sistema V [CZ85].

### Ordenação FIFO (*First In First Out*)

Neste tipo de ordenação, todas as interações originadas por um mesmo membro são observadas por todos os demais membros do grupo na mesma ordem. Por outro lado, interações geradas por membros diferentes não são ordenadas entre si, podendo ser entregues aos membros em ordens diferentes. Ordenação FIFO é uma generalização de protocolos de transporte “ponto-a-ponto” com garantia de ordenação “fim-a-fim” de mensagens, tal como TCP (*Transmission Control Protocol*) [Com91].

### Ordenação Causal

Esta abordagem está baseada na ordem dos eventos em sistemas distribuídos, através da *relação de precedência* [Lam78], usada para estabelecer um relacionamento de causalidade entre as interações (tratadas em termos dos eventos correspondentes) que ocorrem em um grupo. A relação precedência permite determinar todas as situações de potencial relacionamento de *causa e efeito* entre

as interações em um sistema distribuído, sendo determinada com base nas duas sentenças abaixo:

1. Um evento  $a$  precede um evento  $b$  (expresso como  $a \rightarrow b$ ) se ambos os eventos são gerados, nesta ordem, pelo mesmo membro de grupo (mais precisamente, por um mesmo *thread* de execução)
2. Se  $a$  é o evento de gerar uma interação e  $b$  é o evento de recebê-la (em um outro processo ou *thread*), então  $a$  precede  $b$  (expresso como  $a \rightarrow b$ )

Pode ser mostrado que a relação de precedência é transitiva, o que permite verificar a causalidade entre qualquer par de eventos (e, conseqüentemente, interações) ocorridos em um grupo, ou seja, sendo  $a$ ,  $b$  e  $c$  eventos ocorridos no grupo, tem-se:

$$(a \rightarrow b) \wedge (b \rightarrow c) \implies a \rightarrow c$$

Eventos não relacionados pela relação de precedência são ditos serem *concorrentes* entre si.

Um protocolo de ordenação causal garante que todas as interações causalmente relacionadas em um grupo são entregues na mesma ordem relativa para todos os membros do grupo. Por outro lado, interações concorrentes podem ser entregues aos membros em ordens mutuamente diferentes. A Figura 3.8 ilustra a relação de precedência entre as interações de um grupo. Na parte (a)

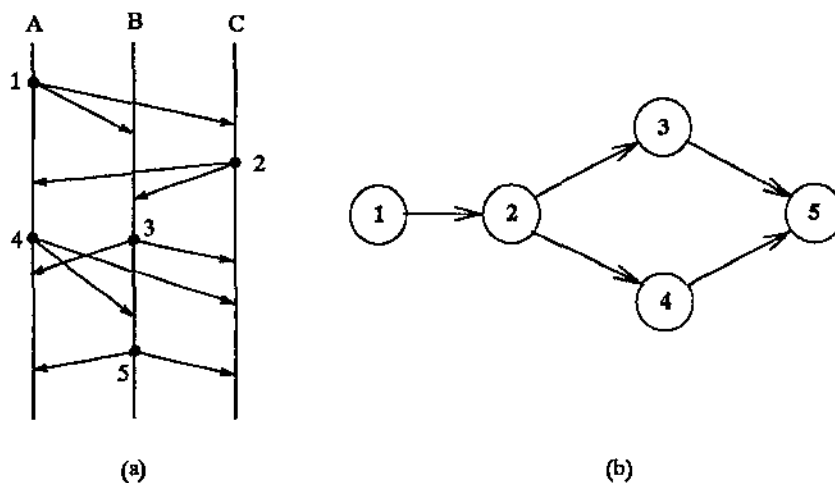


Figura 3.8: Ordenação causal das interações de grupo

da figura é mostrada a seqüência temporal de eventos, enquanto que, na parte (b), mostra-se o relacionamento de causalidade entre eles. Como se pode notar, a interação 1 precede a interação 2 (pela transitividade da relação de precedência), uma vez que o evento correspondente ao envio da interação 1 precede o evento correspondente à sua recepção (pelo membro C), o qual, por sua vez, precede o evento de enviar a interação 2. O mesmo ocorre com as interações 3 e 4, que são precedidas pela interação 2 e, conseqüentemente, pela interação 1. Diz-se que as interações 3 e 4 *dependem* da interação 2 e que esta depende de 1. Por outro lado, 3 e 4 são interações concorrentes

entre si, uma vez que uma não depende da outra, podendo ser observadas pelos membros em ordens diferentes. Finalmente, a interação 5 é dependente de todas as demais, só podendo ser entregue aos membros do grupo após todas as interações precedentes. Neste exemplo, portanto, duas ordens diferentes de entrega de interações a cada um dos membros destinos são possíveis (e corretas):

- 1 2 3 4 5
- 1 2 4 3 5

Ordenação causal de eventos está entre os mecanismos principais de coordenação de eventos de grupo usados no sistema ISIS [Bir93]. Esta abordagem é usada como forma de prover um ambiente com *sincronismo virtual* [BJ87a], no qual aplicações podem interagir de maneira segura e consistente através de mecanismos assíncronos (não-bloqueantes) de comunicação, sem terem que lidar com as complicações decorrentes da concorrência dos eventos. Entretanto, a ordenação causal é completamente adequada somente para aplicações em que as únicas dependências possíveis entre membros de grupos são aquelas decorrentes da comunicação entre eles, que podem ser detectadas através da relação de precedência definida acima. Isto significa que apenas os eventos causalmente relacionados se comportam como em um ambiente síncrono. Por outro lado, nos casos em que existem outros tipos de dependências, devidas, por exemplo, a recursos compartilhados, mesmo as interações causalmente concorrentes podem ser críticas para a manutenção da consistência da aplicação (pois podem atualizar algum recurso compartilhado pelo grupo). Com isto, torna-se necessário o uso de algum mecanismo, adicional ao mecanismo de ordenação causal (exclusão mútua, por exemplo), definido no nível da aplicação, para garantir a consistência [Bir93].

### Ordenação Total

Nesta abordagem, também conhecida como *ordenação atômica*, todas as interações (inclusive aquelas causalmente concorrentes) são ordenadas em uma seqüência única, a qual é observada por todos os membros do grupo. O critério para ordenação destes eventos pode ser arbitrário, desde que a ordem estabelecida seja única.

A ordenação total de eventos permite que aplicações distribuídas sejam programadas como se sua execução se desse em um ambiente centralizado, onde as complicações decorrentes da concorrência entre os membros de grupos são eliminadas. Em outras palavras, garantindo que todos os membros de um grupo observem todas as interações na mesma ordem, garante-se a consistência das informações manipuladas através destas interações, pois todos os membros observarão, sempre, visões de grupo idênticas. A ordem total não necessariamente obedece a relação de causalidade entre os eventos de um grupo, desde que pode ser atribuída uma ordem arbitrária a interações geradas sucessivamente pelo mesmo membro de um grupo (que, por definição, são causalmente relacionadas) [BCG91].

Na Figura 3.9 observa-se as mesmas interações da Figura 3.8, em um ambiente onde os eventos são ordenados totalmente. A ordenação total de eventos implica em que, mesmo o membro que gerou uma interação precisa esperar até que a ordem total seja determinada, antes que ele possa aplicar o efeito daquela interação (como é o caso da interação 4, que precisou ser atrasada até a

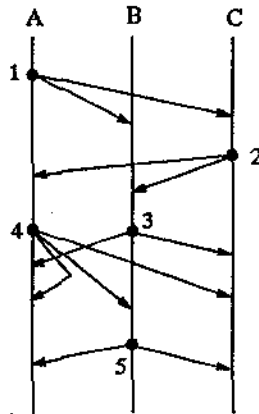


Figura 3.9: Ordenação total das interações de grupo

recepção da interação 3, precedente na ordem total determinada). Como se observa na figura, a ordem total definida para as interações é: 1, 2, 3, 4, 5.

Esta abordagem de ordenação de eventos é empregada no sistema Amoeba [KTV93], resultando em um mecanismo de comunicação de grupo notável por sua eficiência. ISIS também dispõe de uma primitiva para comunicação de grupo com ordenação total [BSS91]. A maioria das implementações de ordenação total existentes empregam algum tipo de elemento centralizador, o qual é responsável por observar todos os eventos e atribuir-lhes ordem [Par92a].

### Ordenação Causal-Total

Esta abordagem consiste em combinar as propriedades dos dois tipos de ordenação vistos anteriormente, resultando em um mecanismo de ordenação semanticamente mais forte [Par92a, BSS91]. Com isto, procura-se reduzir as desvantagens apresentadas pela aplicação isolada destes mecanismos em certos tipos de ambientes. Na ordenação causal, eventos concorrentes, mas que podem afetar o estado compartilhado pelo grupo não são ordenados, gerando a possibilidade de inconsistências. Por sua vez, na ordenação total, a seqüência atribuída aos eventos, por ser arbitrária, permite a ordenação incorreta de eventos causalmente relacionados. Isto ocorre em virtude de mensagens enviadas por um mesmo membro (que, por definição, são causalmente relacionadas) podem chegar fora de ordem.

Uma política de ordenação causal-total *garante a entrega das interações a todos os membros de um grupo em uma ordem única, preservando a relação de causalidade entre elas.*

A ordenação causal-total de interações pode ser obtida através da aplicação de um protocolo de comunicação com ordem causal para a distribuição das interações; entretanto, tais interações têm sua entrega (para a aplicação) retardada até a recepção de uma mensagem (também ordenada causalmente) que contém informação para ordenação total das mensagens causalmente concorrentes. Esta é a abordagem empregada no sistema ISIS [BSS91].

Por outro lado, pode-se utilizar uma abordagem mais simples para a obtenção de ordem causal-

total de eventos, a qual consiste em adicionar a propriedade de ordenação FIFO a um protocolo de ordenação total [Ren93] (pois, na ordenação total, com seqüenciador, a causalidade das interações é violada apenas quando ocorre a inversão na ordem de interações enviadas por um mesmo membro de um grupo). Esta é a abordagem empregada no presente trabalho [CM95].

### 3.2.4 Controle de Membros

“Para o tratamento de casos de falha e recuperação de membros de grupos, bem como para a adição e remoção de membros de acordo com uma política apropriada.”

A definição acima refere-se à manutenção da estrutura de grupos no que diz respeito ao seu conjunto de membros. Esta definição é motivada pelos seguintes fatores básicos:

- grupos, em grande parte de suas aplicações, são estruturas dinâmicas, que podem variar ao longo do tempo por meio da entrada de novos membros ou da saída de membros existentes;
- objetos membros de um grupo podem ser independentemente afetados por falhas, sendo que um membro falho fica impossibilitado de participar (pelo menos corretamente) das interações de grupo; a consistência do grupo depende de um tratamento adequado destas situações, o que pode envolver a remoção do membro falho do grupo;
- aplicações de grupo dependem, comumente, de uma correta visão da estrutura de membros do grupo (por exemplo, para evitar que um membro novo no grupo seja ignorado no contexto de uma interação), o que exige um controle adequado das alterações na estrutura do grupo.

A forma como estas questões são tratadas pela infraestrutura de suporte a grupos é definida pelas *políticas*, de forma a oferecer às aplicações a flexibilidade de escolher a alternativa mais apropriada. A seguir, alguns dos diversos aspectos e alternativas de políticas são apresentados.

Um primeiro aspecto de uma política de entrada e saída de membros diz respeito à necessidade de se manter a consistência destas operações no contexto das interações de grupo [JB89]. Um modelo que preserva consistência deve garantir, por exemplo, que um novo membro de um grupo receba todas e somente aquelas interações de grupo posteriores à sua entrada. Desta forma, é necessário sincronizar as mudanças no conjunto de membros de um grupo (bem como outras alterações na estrutura do grupo, como a mudança de políticas) com as interações entre seus membros, o que pode ser feito através da ordenação dos eventos correspondentes juntamente com os eventos de comunicação. Esta abordagem é adotada em ISIS [BSS91], onde exige-se que todas as mensagens anteriores a uma entrada ou saída de membros sejam entregues a todo o grupo antes que a operação possa ocorrer (esta abordagem é também utilizada em [Maf93]). Por outro lado, o sistema V [CZ85] emprega uma abordagem onde a manutenção da consistência é deixada a cargo das aplicações.

Outro aspecto que deve ser tratado em uma política de controle de membros refere-se à determinação da possibilidade da realização de alterações no conjunto. As políticas podem representar alternativas que vão desde a liberdade total de entrada e saída de membros do grupo até à necessidade de se obter permissões do restante do grupo ou de membros específicos. Além disso, podem

ser especificados limites mínimos e máximos para o número de membros de um grupo, o que pode restringir a saída ou entrada de membros.

As condições de saída de membros de um grupo devido a falhas constituem outro aspecto que pode ser tratado no nível de políticas. As alternativas consistiriam, basicamente, na determinação dos tipos de falha que impossibilitam um membro de continuar a participar de um grupo, devendo dele ser removido. Em [RB93], é proposto um modelo em que sucessivas falhas de comunicação com um membro são consideradas como uma falha persistente, que determinam a remoção do membro do grupo.

### 3.3 Abordagens Relacionadas

O conceito de grupos de objetos ou grupos de processos está presente em alguns sistemas distribuídos existentes, como ISIS [Bir93], V [CZ85], Amoeba [KTV93] e Emerald [Par92a]. Neste contexto, grupos são utilizados, principalmente, como ferramenta, no nível de sistemas operacionais, para suportar a replicação de recursos compartilhados, com o objetivo de prover tolerância a falhas e alto nível de disponibilidade dos recursos, bem como de aumentar o desempenho das aplicações. No entanto, outras aplicações distribuídas que envolvam algum tipo de cooperação entre seus componentes podem igualmente ser estruturadas como grupos de objetos. No contexto de plataformas de processamento distribuído aberto, grupos têm sido propostos com um propósito semelhante, destacando-se o modelo de grupos implementado na plataforma ANSA [OE93] e a proposta de grupos de objetos para CORBA [Isi93]. Outra abordagem diferente consiste em prover uma abstração de grupos (simplificada) no nível de redes, sendo um bom exemplo o protocolo IGMP (*Internet Group Management Protocol*), que adiciona algumas funcionalidades de controle de membros ao protocolo IP (*Internet Protocol*) [Com91].

Grupos constituem uma forma conveniente de lidar com os problemas de coordenação de aplicações distribuídas, uma vez que os mecanismos de estruturação e comunicação de grupo podem prover um ambiente confiável no qual tais aplicações podem se abstrair destes problemas. A seguir, são apresentados aspectos dos modelos de grupos adotados por cada um dos exemplos acima mencionados.

#### 3.3.1 ISIS

O sistema ISIS [Bir93, BJ87a] consiste de uma camada de software sobre o sistema operacional, provendo um conjunto de ferramentas para a construção de aplicações distribuídas com alto grau de confiabilidade. Em ISIS, grupos de processos constituem a principal abstração no sentido de simplificar o desenvolvimento de aplicações confiáveis, sendo que um subconjunto das ferramentas é destinado especificamente à estruturação e gerenciamento da configuração de grupos. São suportados os vários estilos de grupos descritos na seção 3.1.1.

Todas as ferramentas de ISIS estão baseadas na existência de primitivas para *multicast* (comunicação “multi-ponto”) confiável entre grupos de processos: CBCAST e ABCAST. A primitiva CBCAST (*causal broadcast*) permite a um processo enviar mensagens para um grupo qualquer de

processos empregando um protocolo de *ordenação causal* de mensagens, enquanto que a primitiva ABCAST (*atomic broadcast*) atua segundo um protocolo de *ordenação causal-total* (em versões anteriores de ISIS [BJ87b], esta primitiva provia apenas ordenação total). Uma terceira primitiva, GBCAST (*group broadcast*), destina-se à comunicação de informações de configuração de grupos (como alterações na sua estrutura devido, por exemplo, à entrada ou saída de membros), sendo estas mensagens sujeitas à ordem total e causal entre as demais mensagens de grupo (transmitidas via CBCAST ou ABCAST).

A escolha entre o uso de ABCAST ou CBCAST pelas aplicações depende do nível de confiabilidade por elas exigido. ABCAST deve ser usada quando existe a necessidade de que todos os eventos (mesmo aqueles concorrentes entre si) sejam vistos na mesma ordem por todos os membros do grupo, resultando em um ambiente *síncrono*. Por outro lado, aplicações com requisitos menos exigentes quanto à semântica de ordenação podem utilizar a primitiva CBCAST, que provê um ambiente *virtualmente síncrono*, onde apenas as mensagens causalmente relacionadas são ordenadas entre si, de forma que, sob certas restrições, aplicações podem assumir um ambiente síncrono. Sincronismo virtual é considerado em ISIS como um modelo suficiente para prover confiabilidade às aplicações distribuídas, ao mesmo tempo isolando-as dos detalhes e problemas de coordenação distribuída.

### 3.3.2 V

O Sistema V [CZ85] consiste de um núcleo de sistema operacional distribuído, cujo objetivo é prover um ambiente transparente para a construção de sistemas distribuídos. V é considerado o precursor na utilização explícita e ampla do conceito de grupos como forma de estruturação de aplicações distribuídas, tendo influenciado modelos posteriormente desenvolvidos. A abordagem, contudo, é substancialmente diferente daquela encontrada em ISIS, principalmente pelo fato de que V não oferece nenhum tipo de mecanismo embutido, como ordenação de mensagens ou atomicidade de *multicasts*, para garantir a consistência de interações de grupo. Considera-se que o melhor lugar para se inserir qualquer tratamento para preservar a consistência das aplicações encontra-se no próprio nível de aplicações [CS93].

Um grupo de processos em V consiste de um conjunto de um ou mais processos idênticos, tratados uniformemente através de um identificador de grupo. As primitivas para o tratamento de grupos são integradas às primitivas para o tratamento de processos, sendo que um identificador de grupo é idêntico a um identificador de processo. Entretanto, um conjunto especial de primitivas foi introduzido no núcleo para criação de grupos e entrada e saída de membros, sendo que a finalização de grupos é feita através da mesma primitiva usada para destruir processos (além disso, um grupo é finalizado automaticamente quando seu número de membros cai para zero). A comunicação de grupo é feita através das mesmas primitivas usadas na comunicação inter-processos: *Send*, para o envio síncrono de mensagens (o processo que enviou a mensagem fica bloqueado até receber a primeira resposta); *Receive*, para a recepção de mensagens; *Reply*, para o envio de respostas; e *GetReply*, para a recepção de respostas (o tratamento das múltiplas respostas recebidas de um grupo é feito através de invocações sucessivas desta primitiva). Mecanismos de comunicação “multi-



ponto” disponíveis no nível de hardware são amplamente utilizados.

Grupos no sistema V são abertos quanto à interação (qualquer processo pode enviar mensagens para um grupo), mas podem oferecer restrições quanto à entrada de membros. Neste sentido, um grupo pode ser *local*, se todos os seus membros devem ser processos na mesma máquina, ou *global* em caso contrário; além disso, um grupo pode ser restrito a processos de um mesmo usuário.

### 3.3.3 Amoeba

Amoeba [Tan92] consiste de um sistema operacional distribuído orientado a processos e que segue o modelo *cliente-servidor*, sendo este o estilo de grupos suportado [KTV93, KT92]. Grupos em Amoeba são *fechados*, o que significa que processos externos não podem se comunicar diretamente com um grupo.

A abstração de grupos é suportada por um conjunto de primitivas para criação de grupos, entrada e saída de membros, recuperação de falhas e para comunicação de grupo. O envio de mensagens para um grupo é *atômico*, mesmo na presença de falhas, e *totalmente ordenado*, o que significa que todos os membros não falhos de um grupo recebem todas as mensagens na mesma ordem. As mudanças na estrutura de um grupo são também ordenadas com respeito à comunicação de grupo. Sempre que possível, mecanismos de comunicação “multi-ponto” disponíveis no nível de hardware são utilizados

### 3.3.4 Emerald

Emerald consiste de um ambiente de objetos distribuídos (resultado da combinação de uma linguagem baseada em objetos com um sistema operacional distribuído), o qual suporta a abstração de grupos de objetos [Par92b, Par92a]. Várias alternativas quanto a comunicação de grupo (atomicidade, ordenação) são suportadas, podendo haver sincronização entre grupos diferentes que possuem membros em comum.

Grupos são estruturados através de objetos com funcionalidades diferenciadas, interconectados através de *canais*, que permitem a comunicação multi-ponto entre os objetos. Três classes de objetos são definidas em um grupo: *objetos grupo*, que realizam a interface entre um grupo e seus clientes, possibilitando aos clientes visualizar o grupo como um único objeto; *objetos membro*, que executam o processamento cooperativo e interagem apenas entre si e com os objetos grupo; e *objetos cliente*, que solicitam serviços a um grupo através da interface oferecida pelos objetos grupo.

O processo de especificação de um grupo em Emerald inclui a definição de protocolos específicos para manutenção do conjunto de membros, de forma semelhante ao conceito de políticas de grupo.

Um mecanismo de *invocação múltipla de objetos* é proposto para permitir aos objetos cliente se comunicarem com um grupo (através do objeto grupo, que distribui a mensagem para os objetos membros do grupo), sendo que várias alternativas são possíveis para a coleta das respostas associadas a uma invocação.

### 3.3.5 Grupos de Interfaces em ANSA

De acordo com o modelo de objetos de ANSA [Lin93], as interfaces dos objetos são entidades de primeira ordem, que participam de forma ativa das interações. Um objeto pode ter mais de uma interface, podendo exibir comportamentos diferentes em cada uma delas. Em virtude disto, os membros de grupos em ANSA são *interfaces*, em lugar de objetos, sendo que todas as interfaces de um grupo devem ter conformidade com um tipo de interface comum [OE93].

O modelo de grupos de interfaces é especificado em uma linguagem computacional, sendo possíveis várias implementações diferentes. Entretanto, é fornecida uma especificação do modelo no nível de engenharia, que considera as funcionalidades de grupos como sendo implementadas acima do núcleo da plataforma, o que permite a portabilidade para outras arquiteturas de processamento distribuído (como CORBA ou DCE). O modelo compreende diversos aspectos, entre eles:

- interação inter-grupo, para permitir a clientes externos invocarem os serviços de um grupo servidor; tais clientes podem ser tanto objetos simples quanto outros grupos; são suportados os conceitos de *distribuição* de invocações (para o os membros do grupo) e *colagem* de múltiplas mensagens equivalentes;
- interação intra-grupo, para cooperação entre os membros de um grupo; são tratadas questões como ordenação da comunicação entre os membros, falhas de interação e controle de concorrência;
- sincronização das mudanças na estrutura de grupos (através da entrada ou saída de membros, por exemplo) com as demais interações entre os membros;
- separação entre mecanismos e políticas, permitindo a definição de diversas alternativas de execução, por exemplo, para os mecanismos de estruturação de grupos;
- transparência de grupo, visto como uma forma de se definir tipos diferentes de grupos através da seleção de níveis adequados de transparência para cada um dos aspectos relevantes de um grupo; dependendo do nível de transparência adotado, os clientes podem interagir com um grupo de interfaces como se este representasse uma entidade única;
- qualidades de serviço, permitindo controlar a execução dos serviços providos por um grupo, de modo a satisfazer os requisitos dos clientes.

O modelo descrito, entretanto, adequa-se completamente apenas a certos tipos bem definidos de grupos, notadamente, grupos de réplicas ativas, onde os vários membros de um grupo mantêm cópias de um mesmo recurso, sendo capazes de atenderem individualmente ou por cooperação a invocações de serviços sobre estes recursos. Especificações para outros tipos de grupos são consideradas necessárias.

Em [CSB92] é descrita a implementação de uma facilidade para invocação de grupos na arquitetura ANSA, de modo a permitir a criação de grupos de interfaces, de acordo com o modelo brevemente mencionado acima.

### 3.3.6 Grupos em CORBA

Em [Isi93] é proposto um modelo de grupos para integrar a arquitetura CORBA. A proposta encontra-se sob a forma de resposta a um RFI (*Request For Information*) editado pela OMG (*Object Management Group*) com o fim de coletar sugestões de extensões para o *Object Request Broker* da CORBA 2.0 [OMG92b]. Juntamente com o modelo de grupos, são propostas alterações ou extensões em vários aspectos da arquitetura CORBA para suportar grupos de objetos. A intenção básica é suportar mecanismos de grupo diretamente no ORB, considerando grupos como entidades de primeira classe, cujas interfaces são definidas em IDL.

Portanto, nesta proposta, grupos de objetos são considerados no mesmo nível dos objetos de CORBA, podendo participar normalmente em interações como se fossem objetos comuns. A ênfase está no suporte a grupos do tipo *cliente-servidor*, com a provisão de transparência às interações entre os clientes e os grupos servidores. Desta forma, as interações com um grupo se comportam de maneira similar a interações “ponto-a-ponto”, o que simplifica a construção dos clientes.

Outra questão tratada diz respeito ao suporte de comunicação fornecido pelo ORB às interações de grupo. Considera-se a necessidade de prover, no nível do ORB, suporte a comunicação “multi-ponto”, possivelmente explorando as características do hardware subjacente.

Os mecanismos de grupo previstos no modelo oferecem funcionalidades para invocação de grupo, colagem de resultados, ordenação das interações entre membros de um grupo e entre grupos diferentes, controle da entrada e saída de membros e detecção de falhas. Para cada um destes mecanismos são especificadas várias alternativas, as quais devem ser selecionadas e especificadas na definição em IDL de cada (tipo de) grupo em particular, sendo que estas definições podem ser armazenadas no repositório de interfaces da CORBA.

## Capítulo 4

# O Modelo de Grupos de Objetos

Uma aplicação de trabalho cooperativo (*Computer Supported Cooperative Work - CSCW*) normalmente consiste de um conjunto de objetos ou componentes distribuídos, cada um suportando um usuário da aplicação. Estes objetos realizam funções que fornecem aos usuários um ambiente com recursos e atividades compartilhados, como por exemplo, um ambiente para editoração colaborativa de documentos, ou um ambiente de tele-conferência suportada por computador. Para tanto, torna-se necessário um nível intenso de cooperação entre estes objetos, de forma que possam implementar, de maneira efetiva, as interações entre os usuários, bem como o controle das atividades sobre o ambiente compartilhado. Uma estruturação natural destes objetos da aplicação consiste em considerar cada um deles como membro de um grupo de objetos cooperativos, capaz de propagar informações entre os mesmos de maneira consistente, bem como de manter certo controle sobre a constituição do próprio grupo.

Observa-se que aplicações de trabalho cooperativo podem sempre ser estruturadas segundo a abstração de grupos [BR91, SST94], devendo, portanto, de alguma forma lidar com as questões de comunicação e estruturação de grupos. Por outro lado, o tratamento de tais questões envolve aspectos de baixo nível, distintos dos objetivos reais das aplicações, mas que necessitam de soluções uniformes e confiáveis, de forma que aplicações desenvolvidas independentemente possam ser integradas em um mesmo ambiente [MNR92]. Desta forma, torna-se razoável prover um nível ou camada de suporte com facilidades para controlar a estrutura de grupos e permitir a interação consistente entre seus membros [BCG91], tornando transparentes, no processo de construção das aplicações, estes aspectos de baixo nível. A Figura 4.1 ilustra uma configuração de suporte a grupos em um ambiente de trabalho cooperativo (edição colaborativa de documentos).

O modelo de grupos aqui proposto procura oferecer estes recursos de interação e estruturação para uso na construção de aplicações cooperativas. O modelo engloba as quatro funcionalidades definidas para a Função de Grupos, vistas no capítulo anterior, sendo que cada um dos aspectos destas funcionalidades é modelado através das alternativas apropriadas para o suporte a aplicações cooperativas. O conceito de *serviços de suporte a grupos* é definido como a forma de acesso a estas

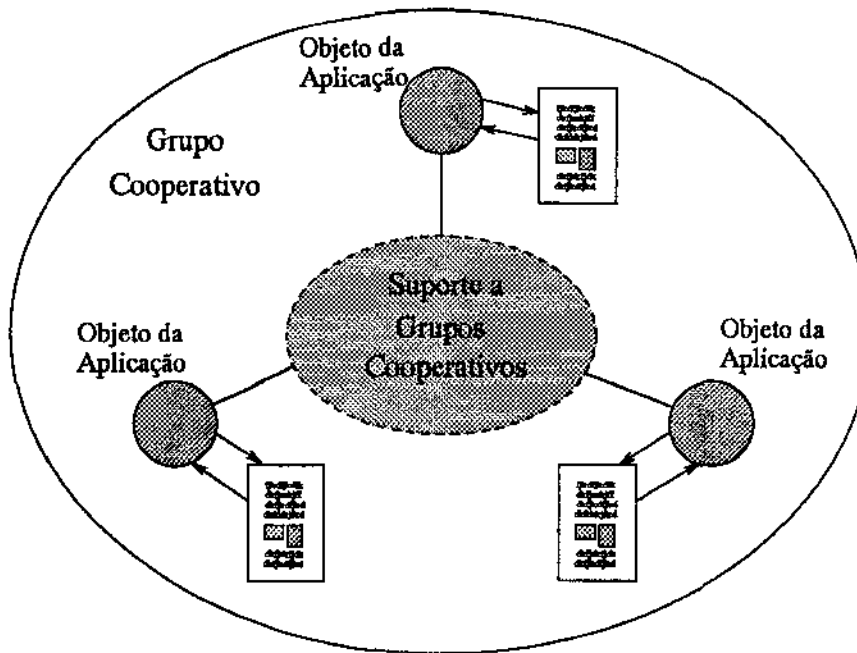


Figura 4.1: Um ambiente de suporte a grupos cooperativos

funcionalidades.

O modelo define *suporte a grupos* no nível de sistemas distribuídos, não envolvendo, portanto, conhecimentos específicos das aplicações ou mesmo dos complexos processos envolvidos na cooperação entre os usuários membros de um grupo, como o compartilhamento de recursos ou a resolução de conflitos no acesso aos mesmos. Estas tarefas seriam de responsabilidade das aplicações de trabalho cooperativo, as quais utilizariam os recursos da infraestrutura de suporte a grupos.

Neste capítulo são abordadas as características e propriedades dos grupos do modelo, sendo que os aspectos de implementação da infraestrutura de suporte a estes grupos são descritos nos capítulos seguintes. Na seção 4.1 deste capítulo são apresentadas as características fundamentais do modelo de grupos, sendo tratados aspectos relacionados com a constituição de grupos e as interações entre seus membros. A seção 4.2, por sua vez, descreve o modelo de acesso à abstração de grupos através de um conjunto limitado, mas flexível, de serviços de suporte a grupos.

## 4.1 O Modelo de Grupos

Esta seção define o modelo de grupos proposto neste trabalho, tratando questões como a estrutura dos grupos e a comunicação entre seus membros.

Os aspectos relacionados com a estruturação de grupos são tratados de maneira *explícita* no modelo. A *criação* de um novo grupo ocorre durante o processo de estabelecimento de um ambiente de trabalho cooperativo, conforme especificado pela aplicação, sendo que um grupo é criado com

um conjunto inicial de membros (possivelmente vazio). Devido à natureza dinâmica dos ambientes de trabalho cooperativo [BR91], é possível realizar, também explicitamente, alterações na estrutura de um grupo, através da *entrada de novos membros*, bem como da *saída de membros existentes*, durante o processo de cooperação. A *finalização* de um grupo é também determinada de maneira explícita pela aplicação, normalmente no final da sessão de trabalho cooperativo. A alteração de outros aspectos da configuração de um grupo pode também ser especificada pela aplicação.

Todos os membros de um grupo são considerados como entidades do mesmo nível, com potencial para participarem igualmente de todas as interações de grupo. Isto classifica os grupos do modelo como *simétricos* (“*peer groups*” [Bir93]).

Entretanto, pode haver uma certa distinção entre os membros, com base na função que eles desempenham no contexto da aplicação cooperativa. Esta distinção pode ser representada no nível de suporte a grupos em termos do conceito de *papéis*, que permite às aplicações estabelecerem um tratamento diferenciado para os membros de um grupo. O conceito de papéis é utilizado de maneira restrita em [BS94] para classificar os objetos membros de um grupo de acordo com sua função, existindo um conjunto pré-definido e limitado de papéis.

Como exemplo do uso de *papéis* por uma aplicação cooperativa, tem-se o caso em que determinado usuário membro de um grupo é responsável por coordenar as atividades dos demais membros, tendo certas características ou prerrogativas especiais. Neste caso, pode-se associar o papel de *coordenador* a este membro, de forma que ele possa ser diferenciado nas interações de grupo. O papel de *coordenador* é o único com semântica explicitamente reconhecida no nível de suporte a grupos, sendo utilizado na execução de políticas de serviços. Este papel é normalmente atribuído ao usuário criador do grupo (caso este seja um membro do grupo criado). A associação de papéis aos membros é dinâmica, podendo ser alterada durante o tempo de vida de um grupo.

No nível de suporte a grupos, papéis são utilizados em duas situações básicas. Na verificação de autorizações para a execução dos serviços de suporte a grupos (seção 4.2), a associação apropriada de papéis determina quais membros de um grupo são autorizados a invocar determinado serviço. Por outro lado, na comunicação de grupo, papéis podem ser utilizados como forma de referenciar um subconjunto dos membros de um grupo, para determinar quais os membros que devem participar de uma dada interação de grupo. Em determinados contextos torna-se necessário a um membro se comunicar com apenas um subconjunto dos membros do grupo, o que pode ser feito associando um *papel específico* a estes membros, e usando este papel para especificar o destino das interações em questão. Neste sentido, um papel pode ser entendido como uma forma simplificada de implementar o conceito de *sub-grupos* dentro de um grupo. Estes sub-grupos, entretanto, não implementam as funcionalidades genéricas de suporte a grupos, como a semântica de ordenação de mensagens. Observe que a ênfase principal do conceito de *papéis*, neste modelo, está em traduzir uma noção existente no nível das aplicações para o nível de suporte a grupos.

O modelo de grupos proposto é genérico com respeito ao comportamento dos membros de um grupo, uma vez que esta é uma questão que se manifesta somente no nível das aplicações. Desta forma, de acordo com a classificação apresentada na seção 3.1.1, o modelo suporta tanto grupos *determinísticos* quanto *não-determinísticos*. Além disso, as funcionalidades do modelo permitem a especificação de grupos que se conformam a qualquer um dos quatro tipos definidos naquela seção

com respeito à *homogeneidade* das operações e dos dados manipulados pelos membros.

No que diz respeito à comunicação, grupos são *fechados*, o que significa que todas as interações em um grupo devem necessariamente ser geradas por membros do grupo. Além disso, não é provido suporte para interações entre grupos distintos. Desta forma, o suporte oferecido pelo modelo procura atender ao estilo de cooperação em que os vários usuários de um grupo colaboram entre si, sem influência direta do meio externo. Um usuário (objeto de aplicação) não-membro que queira se comunicar com um determinado grupo teria, então, duas alternativas: ele poderia entrar para o grupo, ou poderia enviar as mensagens para um dos membros do grupo, o qual propagaria as referidas mensagens para os demais membros.

O modelo contempla, em primeiro plano, apenas interações do tipo “um-para-muitos”, conforme ilustrado na Figura 4.2, que é considerado o caso mais freqüente em aplicações cooperativas. Interações do tipo “muitos-para-um”, que também ocorrem nesta classe de aplicações, são suportadas no tratamento de respostas de grupo, quando os membros de um grupo devem responder a uma invocação de grupo.

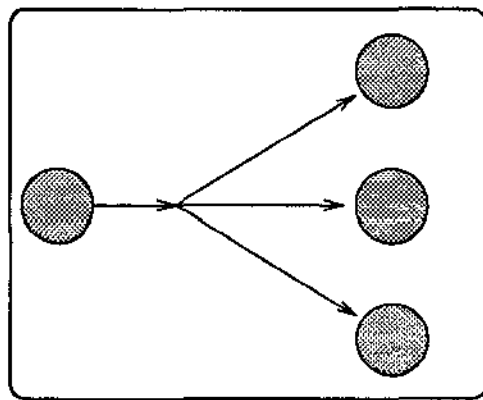


Figura 4.2: Estilo de interação entre os membros de um grupo

#### 4.1.1 Aspectos da Comunicação de Grupo

A comunicação entre os membros de um grupo se dá por meio de trocas de mensagens entre eles, através de um mecanismo conhecido como **distribuição de mensagens**. São possíveis três tipos de mensagens, usados para representar as duas formas de interação operacional, *interrogações* e *anúncios*, definidas no capítulo 3. Uma interação de interrogação envolve uma mensagem para invocação de alguma funcionalidade dos membros do grupo (referida como *mensagem de interrogação*) e, possivelmente, várias *mensagens de resposta*, retornadas por cada um dos membros que receberam a interrogação. Por outro lado, um anúncio envolve apenas comunicação em um sentido (através de *mensagens de anúncio*). A representação distinta das duas formas de interação operacional permite associar corretamente mensagens correspondentes à mesma interação, tanto no nível de suporte quanto no nível das aplicações, de forma a tratá-las adequadamente (especialmente

no caso de mensagens de interrogação e das mensagens de resposta associadas). Interações de fluxo são tratadas de maneira diferente, através do uso de *canais de fluxo contínuo*, conforme descrito a seguir.

Três aspectos fundamentais relacionados à comunicação de grupo em ambientes cooperativos são tratados no modelo: transparência das interações de grupo, ordenação das trocas de mensagens e confiabilidade da comunicação. A seguir são mostradas as soluções adotadas para estes aspectos.

### Transparência das Interações de Grupo

Este aspecto refere-se ao grau de ocultamento dos detalhes e problemas da comunicação “multi-ponto”, no sentido de simplificar a interação entre os membros de um grupo no nível da aplicação. É necessário tornar o envio e a recepção de mensagens de grupo, tanto quanto possível, semelhantes à comunicação “ponto-a-ponto”. Por outro lado, deve-se permitir a cada membro ter uma visão do conjunto de membros do grupo, possibilitando certo nível de controle sobre a comunicação de grupo, por exemplo, para coordenar as atividades cooperativas e o compartilhamento de informações [Rod92].

Desta forma, a *distribuição de mensagens* para um grupo pode ser feita de três maneiras diferentes (semelhante ao proposto em [Isi93]), correspondendo à definição da funcionalidade de *interação* da Função de Grupos do RM-ODP, que prevê alguma forma de definir quais membros de um grupo participam de quais interações:

- Envio de mensagens para todos os membros do grupo, usando o identificador do grupo;
- Envio de mensagens para todos os membros associados a determinados *papéis*, conforme visto anteriormente;
- Envio de mensagens para membros especificados individualmente.

Estes três estilos de distribuição apresentam níveis distintos de transparência quanto ao uso da abstração de grupos, permitindo modelar os diversos padrões de interação encontrados em um ambiente de trabalho cooperativo [SR92]. Entretanto, em todos estes casos, é provido um conjunto comum de transparências de distribuição, para ocultar das aplicações os aspectos de localização, acesso, migração e tratamento de falhas dos objetos.

Com respeito à recepção de interações “multi-ponto”, no caso de *mensagens de resposta*, o modelo de grupos aqui proposto segue o conceito de *colagem de interações* apresentado na Função de Grupos do RM-ODP. Através da colagem das mensagens de resposta correspondentes a uma determinada interrogação, é possível entregar ao objeto de aplicação (que gerou a interrogação), uma única resposta de grupo, equivalente em conteúdo às respostas individuais. Neste sentido, as interações de grupo podem ser consideradas transparentes, pois um objeto (membro) pode interagir com o grupo como se estivesse interagindo com um objeto individual. Entretanto, um nível de transparência total das interações pode não ser desejável; além disso, em determinadas circunstâncias, a colagem de respostas depende de conhecimento disponível apenas no nível de aplicação. Desta forma, o modelo de suporte a grupos realiza a separação entre o mecanismo



de colagem de respostas e as *políticas* usadas para determinar o tipo de colagem desejado. Três políticas de *colagem* de respostas são providas no suporte a grupos, as quais não dependem de conhecimento no nível de aplicação (políticas de colagem semelhantes são propostas em [Par92b] e [Isi93]):

- coleta de todas as respostas, as quais são concatenadas e entregues à aplicação;
- coleta das  $n$  primeiras respostas, as quais são concatenadas e entregues à aplicação ( $n$  é especificado como parte da política de colagem), sendo que as demais respostas são descartadas;
- coleta sem colagem: cada resposta é individualmente entregue à aplicação imediatamente após ser recebida

Políticas mais elaboradas podem ser definidas no nível das aplicações. Neste caso, poderia ser selecionada a terceira política acima, dando às aplicações a liberdade para manipular as respostas recebidas.

### Ordenação de Eventos

Os requisitos quanto à ordenação de eventos de grupo variam de acordo com o tipo de aplicação cooperativa em questão [SST94]. Algumas aplicações dependem essencialmente da ordem em que os usuários observam os acontecimentos de um grupo (mensagens ou mudanças na estrutura do grupo). Exemplo disto são aplicações de edição colaborativa de documentos, onde a ordem de entrega das mensagens permite manter a consistência do documento editado. Como visto no capítulo anterior, a existência de um mecanismo de ordenação de eventos proporciona um ambiente de comunicação com sincronismo implícito, o que simplifica a construção de aplicações confiáveis [BJ87a].

Por outro lado, certas aplicações podem dispensar a provisão de ordenação de eventos, seja porque a consistência do ambiente compartilhado não depende da ordem dos eventos, ou porque o custo envolvido na determinação da ordem pode não ser tolerável à aplicação. Neste último caso, as aplicações podem usar conhecimento de alto nível para impor ordem apenas aos eventos necessários, de uma maneira mais eficiente.

Além disso, uma mesma aplicação pode utilizar ambas as abordagens, onde mensagens não críticas são distribuídas sem ordenação, enquanto que mensagens críticas para a consistência da aplicação são distribuídas com ordenação.

Desta forma, o modelo de suporte a grupos provê as duas *políticas de ordenação* de eventos de grupo: distribuição com ordenação e distribuição sem ordenação. O tipo de ordenação adotado no modelo consiste na *ordenação causal-total* dos eventos, que, conforme visto no capítulo 3, estabelece uma ordem única para a entrega de todos os eventos, preservando a relação de causalidade entre eles. Além das mensagens críticas, para as quais a aplicação determina a distribuição com ordenação, eventos que implicam em mudança na estrutura de um grupo (devido à entrada ou saída de membros, por exemplo) são *sempre* ordenados. Por outro lado, no caso de comunicação não-ordenada, o modelo garante apenas que a entrega das mensagens segue a ordem em que foram enviadas por um membro individualmente (segundo o modelo de ordenação *FIFO*).

### Confiabilidade da Comunicação de Grupo

Outro aspecto do qual depende a consistência das aplicações cooperativas refere-se à propriedade de *atomicidade* da comunicação de grupo, conforme vista no capítulo 3. O modelo de suporte a grupos garante esta propriedade no caso da comunicação com ordenação. Na comunicação sem ordenação, por outro lado, a atomicidade não é garantida. Esta abordagem se deve ao fato do mecanismo de comunicação sem ordenação ser provido como uma alternativa de baixo custo, que permite maior eficiência na distribuição de mensagens em detrimento do aspecto de confiabilidade. Por outro lado, a comunicação com ordenação é considerada uma alternativa com semântica confiável, garantindo que, caso um evento seja entregue para um dos membros de um grupo, ele será entregue para todos os demais, sendo que a ordem em que isto é feito é idêntica em todo o grupo.

## 4.2 O Modelo de Serviços de Suporte a Grupos

A manipulação da estrutura de grupos e o acesso às funcionalidades de comunicação de grupo é feito, no modelo de suporte a grupos aqui proposto, através de um conjunto de serviços específicos. Existem serviços para criação de grupos, para entrada de novos membros em um grupo, além de outros, conforme visto adiante.

O modelo de serviços está fundamentado na idéia de separação entre os **mecanismos** de execução dos serviços e as **políticas** que controlam estes mecanismos [Rod92]. Este conceito de políticas, como visto no capítulo 3, é fundamental na definição das funcionalidades da Função de Grupos. Para cada serviço de suporte a grupos existe um conjunto básico de políticas, as quais representam alternativas de execução diferentes que podem ser adicionadas ao núcleo funcional básico do serviço. Desta maneira, é possível dar às aplicações certa flexibilidade para determinar a forma como grupos são estruturados, permitindo ainda controlar certos aspectos da comunicação de grupo.

Este conjunto básico de políticas é definido de tal forma a não exigir conhecimentos do nível das aplicações, tornando as políticas genéricas e permitindo seu tratamento no nível de suporte a grupos. Entretanto, para aplicações que necessitam de políticas mais sofisticadas, uma política especial é provida para cada serviço, a qual não impõe nenhum comportamento adicional ao núcleo do serviço, permitindo a definição, no nível das aplicações, de uma política mais adequada. Para implementar tais políticas, as aplicações normalmente farão uso das funcionalidades de comunicação de grupos do modelo. Com isto, antes de invocar um serviço de suporte a grupos, a aplicação executaria a política por ela definida para o serviço.

O aspecto de *segurança* de grupos é tratado no nível de suporte a grupos no que diz respeito ao *controle de acesso aos serviços*. Para tanto, é usado o conceito de **autorizações** de serviço, que permite às aplicações determinarem quais os membros de um grupo podem invocar os serviços. As autorizações de acesso são especificadas individualmente para cada serviço, sendo expressas em termos dos *papéis* cujos membros associados estejam autorizados a invocar o serviço, ou através dos próprios nomes dos membros autorizados. Além das restrições de acesso que podem ser especificadas pelas aplicações através do mecanismo de autorizações, o modelo de suporte a grupos define, para

cada serviço, restrições de acesso intrínsecas, que permitem reforçar o caráter *fechado* dos grupos, conforme observado na próxima seção.

A seguir, é apresentado o conjunto de serviços de suporte a grupos, com a descrição do núcleo funcional de cada serviço, juntamente com o conjunto básico de políticas associadas. São também consideradas as possíveis situações de uso de cada serviço por parte das aplicações cooperativas. Neste trabalho, os serviços de suporte a grupos são classificados em *serviços de estruturação de grupos* e *serviços de comunicação de grupos*.

#### 4.2.1 Serviços de Estruturação de Grupos

Estes serviços são usados para construir grupos de objetos cooperativos, permitindo ainda alterar a configuração de grupos existentes. São providos serviços para alteração dos diversos aspectos dinâmicos de um grupo, como o conjunto de membros do grupo, as políticas e autorizações de serviço, e os papéis associados aos membros.

À exceção do serviço de criação de grupos, as seguintes alternativas de políticas básicas estão disponíveis para serem associadas aos serviços de estruturação de grupos:

- realização do serviço após consulta e aprovação do grupo, sendo que duas alternativas são disponíveis para avaliar o resultado das consultas: maioria simples, ou um número mínimo de aprovações dos membros, o que é definido no momento em que a política é selecionada;
- realização do serviço após consulta e aprovação do membro que tem o papel de *coordenador* do grupo;
- política livre, que não exige nenhum tipo de consulta, permitindo a realização imediata do serviço; esta deve ser a alternativa adotada no caso da especificação de políticas mais elaboradas no nível da aplicação.

Uma política consiste de uma forma de alterar o comportamento de um serviço, através de procedimentos complementares, adicionados ao núcleo funcional básico do serviço. No presente modelo, a execução das políticas ocorre no início do processo de execução dos serviços.<sup>1</sup>

Para a manutenção da consistência dos grupos, a execução dos serviços de estruturação (exceto criação de grupos) deve ser ordenada com respeito à comunicação de grupo. Desta forma, é possível obter consenso entre os membros do grupo com respeito às mensagens que foram recebidas antes e depois da execução do serviço [Isi93, KTV93].

#### Criação de Grupo

Este serviço é invocado pela aplicação como parte do processo de estabelecimento de uma sessão de trabalho cooperativo, determinando a formação de um novo grupo para representar o conjunto dos usuários cooperantes. Um objeto de aplicação, correspondente ao usuário criador do grupo, ao invocar este serviço, deve especificar a configuração do grupo, informando o nome do grupo, o

---

<sup>1</sup>Futuras extensões do modelo poderão permitir a aplicação de políticas em outros pontos do processo de execução dos serviços.

conjunto inicial de membros do grupo e seus respectivos papéis, além das políticas e as autorizações de acesso aos serviços. O usuário criador não necessariamente deve estar presente entre os membros do novo grupo.

As funções realizadas por este serviço incluem:

- confirmação de participação no grupo de cada um dos membros propostos no conjunto inicial de membros (de acordo com a política e com o estado do objeto de aplicação associado, ou seja, se este objeto não está falho);
- a disseminação de informações sobre o grupo para cada um de seus membros.

As políticas para o serviço de criação de grupos determinam suas funcionalidades adicionais, as quais são executadas no início do processo de criação de grupos, imediatamente após a invocação do serviço. Duas políticas básicas são providas, as quais estão relacionadas com os aspectos de confirmação do conjunto inicial de membros do grupo, sendo descritas a seguir:

- criação com consultas aos usuários: permite a cada usuário determinar se participará ou não do novo grupo; esta política possui ainda duas variações:
  - permitir ao usuário criador do grupo avaliar o resultado das consultas para determinar a efetiva criação do grupo, e
  - criar automaticamente o grupo consistindo do conjunto de membros que confirmaram a participação;
- criação sem consultas: determina a criação do grupo com a participação (obrigatória) dos usuários propostos como membros; esta alternativa deve ser adotada pelas aplicações caso seja provida uma política de mais alto nível (e que seria executada antes do início do processo de criação no nível de suporte a grupos).

### **Finalização de um Grupo**

Este serviço permite à aplicação determinar explicitamente o término de um grupo existente, o que normalmente ocorre ao final da sessão de trabalho cooperativo. A invocação do serviço é feita por um objeto de aplicação, o qual necessariamente deve representar um membro do grupo a ser finalizado (as autorizações associadas a este serviço definem quais membros podem determinar a finalização de um grupo).

O processo de finalização envolve a distribuição de um evento de notificação a todos os membros do grupo, o qual deve estar sujeito à ordenação de eventos do grupo. Desta forma, obtém-se consenso entre os usuários sobre o momento (em termos da seqüência de eventos do grupo) em que o grupo foi finalizado. Além disso, este evento de finalização deve ser o último evento observado pelos membros do grupo.

### Entrada de Membro em um Grupo

A entrada de um novo usuário em uma sessão de trabalho cooperativo é feita através da entrada do respectivo objeto de aplicação no grupo correspondente. O serviço de entrada de membro pode ser invocado pelo objeto de aplicação correspondente ao novo membro, ou por algum outro objeto de aplicação que já é membro do grupo (neste caso, para determinar a entrada de um outro membro). O segundo caso está sujeito ao mecanismo de autorizações, que determina quais membros de um grupo podem determinar a entrada de novos membros.

O processo de entrada envolve a provisão de informações sobre o grupo para o novo membro, sendo, em seguida, gerado um evento de notificação ao restante do grupo, para informar a entrada do novo membro, assim como informações sobre ele. Este evento é distribuído com ordenação (com respeito aos demais eventos do grupo), de forma que todos os membros o observem no mesmo instante lógico, obtendo assim uma visão consistente do conjunto de membros do grupo. A ordenação do evento de entrada com respeito aos demais eventos de grupo permite que o novo membro observe apenas aqueles eventos posteriores à sua entrada no grupo. O processo de entrada envolve também a atualização da estrutura dos canais que eventualmente estejam associados aos papéis do novo membro.

### Saída de Membro de um Grupo

Para um usuário sair de uma sessão de trabalho cooperativo, o correspondente objeto de aplicação deve ser removido do grupo, através do serviço de *saída de membro*. O serviço pode ser invocado apenas por objetos de aplicação do grupo, não necessariamente por aquele que vai ser removido do grupo. O mecanismo de autorizações determina quais membros podem solicitar a saída de outros membros do grupo (entretanto, um membro pode sempre solicitar a própria saída, mesmo que não esteja na lista de autorizações do serviço). Além disso, a efetivação da saída de um membro pode estar sujeita a restrições expressas pela política associada (como a necessidade de aprovação do grupo, por exemplo).

O processo de saída de um membro envolve um evento de notificação de saída do membro, o qual é distribuído, com ordenação, para os membros restantes do grupo. Além disso, caso o membro que saiu do grupo estivesse ligado a algum canal, a estrutura do canal deve ser atualizada.

### Mudança da Política de um Serviço

Este serviço permite substituir a política associada a um determinado serviço de suporte a grupos por uma das outras políticas pré-definidas, permitindo alterar o comportamento de execução do serviço. A execução do serviço de mudança de política está também sujeita ao mecanismo de políticas, bem como ao mecanismo de autorizações. Além disso, o modelo define que apenas membros do grupo, através de seus respectivos objetos de aplicação, podem (caso autorizados) determinar a execução deste serviço.

A mudança de política é notificada a todos os membros do grupo sob a forma de um evento ordenado.

### Mudança das Autorizações de um Serviço

O conjunto de autorizações associadas a um serviço pode ser alterado através da remoção ou adição de autorizações, de forma a determinar novas restrições ou privilégios de acesso aos serviços no contexto de um grupo específico. Conforme visto anteriormente, autorizações são expressas em termos dos nomes dos papéis ou membros autorizados. Apenas membros do grupo (desde que autorizados) podem invocar com sucesso este serviço, o qual está também sujeito ao mecanismo de políticas.

A mudança das autorizações de um serviço é notificada a todos os membros do grupo sob a forma de um evento distribuído com ordenação.

### Alteração de Papéis de um Membro

O conjunto de papéis associados a um membro pode ser alterado através da adição ou remoção de papéis. O serviço de alteração de papéis pode ser invocado apenas por membros do grupo, estando, ainda, sujeito aos mecanismos de políticas e autorizações.

A alteração dos papéis de um membro é notificada aos membros do grupo através de um evento distribuído com ordenação. Além disso, caso a alteração envolva um papel associado a algum *canal*, a estrutura deste canal deve ser atualizada.

#### 4.2.2 Serviços de Comunicação de Grupo

O acesso às facilidades de comunicação de grupo é possível através do serviço de **distribuição de mensagens**, no caso de *interações operacionais*, e através do serviço de **estabelecimento de canais**, no caso de *interações de fluxo*. O modelo de suporte a grupos define restrições de acesso a estes serviços através do mecanismo de autorizações, que permite definir quais membros de um determinado grupo podem enviar mensagens para os demais membros (ou estabelecer canais). Além disso, devido ao caráter fechado de grupos, apenas os objetos de aplicação que correspondem a membros do grupo em questão podem invocar estes serviços.

#### Distribuição de Mensagens

O serviço de distribuição de mensagens é definido em termos de dois aspectos fundamentais: o tipo das mensagens, e a utilização ou não de um mecanismo para ordenação das mensagens.

As mensagens distribuídas, conforme visto na seção 4.1.1, podem ser de três tipos:

- *mensagens de interrogação*, usadas pela aplicação para a solicitação de algum serviço ou informação aos membros de um grupo (por exemplo, para iniciar uma votação entre os membros do grupo);
- *mensagens de resposta*, usadas pela aplicação para o envio de respostas correspondentes a interrogações recebidas (uma resposta pode, por exemplo, conduzir o voto do membro); as respostas podem ser *coladas* antes de sua entrega ao usuário gerador da interrogação (conforme visto na seção 4.1.1);

- *mensagens de anúncio*, para interações que envolvem comunicação em apenas um sentido entre os usuários, para transferência de informações entre eles (usadas, por exemplo, para um membro propagar atualizações do estado compartilhado do grupo).

Mensagens de interrogação e mensagens de anúncio podem ser distribuídas tanto com ordenação, quanto sem ordenação, cabendo às aplicações determinarem a modalidade a ser utilizada, o que é feito para cada nova mensagem a ser distribuída. Por outro lado, mensagens de resposta apenas são distribuídas sem ordenação (pois sua ordem está implícita nas respectivas mensagens de interrogação).

O serviço de distribuição de mensagens pode ser usado pelas aplicações tanto para realizar a comunicação cooperativa entre os usuários, quanto para a comunicação de mensagens de controle da aplicação (como, por exemplo, as mensagens usadas para implementar uma política de serviço definida no nível da aplicação).

As **políticas** para o serviço de distribuição de mensagens estão divididas em duas classes: as políticas estaticamente associadas ao serviço, e as políticas associadas dinamicamente a cada mensagem de grupo distribuída.

As *políticas estaticamente associadas* definem duas alternativas para a distribuição de mensagens ordenadas, que especificam, respectivamente, *a necessidade ou não do envio de uma mensagem para o membro que a gerou*, caso este pertença ao conjunto de membros endereçados na mensagem. Nos casos em que a consistência das aplicações dependa de um sincronismo forte entre todos os membros, pode ser necessário que todos os usuários (objetos de aplicação), *inclusive* aquele que gerou uma determinada mensagem, possam atuar sobre a mesma *apenas* após ter sido determinada a ordem desta mensagem entre as demais mensagens do grupo. Nestas situações, a primeira alternativa de política deve ser adotada.

As *políticas dinamicamente associadas* às mensagens permitem definir quais os membros de um grupo participam das interações (ou seja, recebem as mensagens). Duas políticas estão disponíveis para mensagens de *interrogação* e mensagens de *anúncio*:

- distribuição para todo o grupo, e
- distribuição para um subconjunto dos membros de um grupo (especificados pelos identificadores dos membros ou pelos identificadores dos *papéis* associados aos mesmos).

No caso de mensagens de resposta, estas são associadas às interrogações originais, o que determina, antecipadamente, o membro do grupo para o qual será enviada a resposta.

### Manutenção de Canais

Para aplicações com demanda de comunicação em tempo real, para a interação multimídia entre os usuários, o modelo de suporte a grupos provê facilidades para o estabelecimento e manutenção de *canais*. Canais permitem conectar as interfaces dos objetos de um grupo e possibilitam a troca de informação de natureza contínua entre eles.<sup>2</sup>

---

<sup>2</sup>O termo "canal" aqui não necessariamente corresponde ao conceito de *canal* da linguagem de engenharia do RM-ODP.

Entretanto, o suporte efetivo aos canais é realizado fora do escopo do modelo de suporte a grupos, por um componente de suporte a comunicação multimídia (como a sub-camada de Processamento Multimídia da Plataforma Multiware), no qual são tratadas questões como a manutenção da qualidade de serviço dos canais.

O gerenciamento dos canais pelo suporte a grupos se restringe à manutenção do conjunto de membros ligados ao canal. Um canal é associado a um *papel* específico existente em um grupo, fazendo a ligação de todos os membros associados ao papel. Desta forma, quando um novo membro é associado ao papel em questão, este membro é automaticamente ligado ao canal (o inverso acontece quando um membro deixa o grupo ou é dissociado do papel).

Um exemplo de uso de canais ocorre em aplicações de tele-conferência, onde é necessária a comunicação multimídia entre os membros de um grupo, com restrições de tempo real. Observe, entretanto, que mesmo esta classe de aplicações não deve prescindir do serviço de distribuição de mensagens, que pode ser necessário, por exemplo, para a distribuição de mensagens para controlar a comunicação através dos canais.

Um canal é especificado em termos do conjunto de membros por ele conectados, pela direção do fluxo de informações, pelas qualidades de serviço, além de outras características. O modelo de suporte a grupos fornece dois serviços para o *estabelecimento* e *fechamento* de canais, respectivamente, além de facilidades para atualizar o conjunto de membros ligados por um canal, à medida em que é alterado o conjunto de membros associados ao papel correspondente. As demais características que especificam um canal estão fora do escopo do modelo e são apenas repassadas para o componente da plataforma responsável pela comunicação multimídia entre objetos.



## Capítulo 5

# O Modelo de Implementação

O Serviço de Suporte a Grupos é estruturado como uma configuração de objetos distribuídos, no sentido expresso no Modelo Descritivo do RM-ODP [ISO94b], os quais interagem entre si para o desempenho das funcionalidades de estruturação e comunicação em grupo. As informações a respeito de grupos existentes encontram-se distribuídas entre estes objetos, sendo que a consistência do conjunto total de informações é mantida através da interação apropriada entre eles. A comunicação entre os objetos de suporte a grupos, bem como entre eles e seus usuários, ocorre através de invocações de métodos (ou operações) nas suas interfaces. São considerados como usuários das funcionalidades de suporte a grupos aplicações que demandem interação cooperativa entre diversos membros de um grupo e, mais especificamente, a camada Groupware da plataforma Multiware (capítulo 2).

Os objetos do Serviço de Suporte a Grupos atuam como intermediários entre as aplicações de trabalho cooperativo e os recursos de comunicação subjacentes. O propósito destes objetos consiste em fornecer um ambiente com serviços de comunicação e estruturação que implementem a abstração de grupos de objetos, tornando mais simples o desenvolvimento deste tipo de aplicação no que diz respeito aos aspectos de distribuição e manutenção da consistência dos grupos. Os serviços de suporte a grupos foram apresentados no capítulo 4.

Este capítulo descreve um modelo para implementação do Serviço de Suporte a Grupos [CM96]. A seção seguinte apresenta algumas questões e conceitos básicos do modelo de implementação. Na seção 5.2 é definida a configuração de objetos de suporte a grupos, mostrando o papel de cada classe de objetos em um contexto de grupos, juntamente com o relacionamento entre eles. A estrutura interna destes objetos (em termos das estruturas de dados por eles mantidas) e suas interfaces são descritas na seção 5.3. Finalmente, a seção 5.4 apresenta a seqüência de interações entre os objetos na realização de cada um dos serviços de grupo. Nesta seção são também descritas as soluções adotadas para ordenação de eventos e tratamento de falhas.

## 5.1 Conceitos Preliminares

Esta seção apresenta algumas questões importantes relacionadas com a definição do modelo de implementação, mostrando o tratamento adotado para cada uma delas.

### 5.1.1 Representação de Informações de Grupos

As informações sobre grupos encontram-se distribuídas entre os objetos de suporte a grupos, de maneira a tornar o acesso a elas local (mais eficiente). Com frequência, esta distribuição é feita através da replicação destas informações entre os objetos que as utilizam, o que implica na necessidade de mecanismos para manter a coerência destas réplicas. Um exemplo de replicação ocorre com as informações sobre membros de um grupo, que devem ser acessíveis de maneira eficiente, pelos objetos de suporte a grupos para realizar funções como a resolução de endereços de grupo na distribuição de mensagens. Cópias da lista de membros são mantidas pelos objetos da estrutura de suporte a grupos que dela façam uso intensivo, sendo a coerência destas réplicas mantida a partir dos eventos de alteração do grupo (entrada de um novo membro no grupo, por exemplo) distribuídos para estes objetos.

Um outro aspecto importante da representação de informações diz respeito à forma como são identificadas as entidades envolvidas em um ambiente de grupos. Neste sentido, deve ser definida a composição dos identificadores de grupos e membros de grupos, bem como dos objetos que constituem a configuração de suporte a grupos. Tais identificadores devem ser únicos no ambiente de grupos. Um grupo é identificado através do seu *nome de grupo*, sendo que, ao ser criado um novo grupo, deve-se garantir que seu nome ainda não exista no contexto. A identificação de membros é também feita através de *nomes de membros*, os quais são atribuídos pelas aplicações, devendo haver uma convenção para a atribuição de nomes únicos. Finalmente, os nomes dos objetos de suporte a grupos são compostos a partir dos nomes dos respectivos grupos ou membros aos quais estão associados, com a adição de qualificadores de contexto ao nome apenas quando necessário (conforme será visto na seção 5.3).<sup>1</sup>

A seguir são apresentadas as estruturas de dados relevantes no ambiente de suporte a grupos.

### Mensagens de Comunicação de Grupo

A comunicação entre os membros de um grupo é expressa em termos de mensagens trocadas entre eles. O Serviço de Suporte a Grupos é o responsável pela transferência destas mensagens do membro fonte para os vários membros do grupo, o que é feito encapsulando a mensagem em uma estrutura de *evento* (vista na próxima seção). A estrutura de mensagem de grupo compreende, além do conteúdo próprio da mensagem, um cabeçalho de controle adicionado pela aplicação para expressar a forma com que a mensagem deverá ser distribuída, conforme mostrado na Figura 5.1.

---

<sup>1</sup>Além da identificação por nomes, os objetos são também identificados através de referências únicas que permitem invocar operações nas interfaces dos objetos. Entretanto, esta é uma questão dependente de implementação e será tratada no capítulo seguinte.

CABEÇALHO
Nome do Grupo a que se refere a mensagem
Membro Remetente da mensagem, identificado pelo nome
Modo de Distribuição, que pode ser: (1) para o grupo todo, ou (2) para um subconjunto dos membros do grupo (especificados através dos nomes dos membros e/ou dos papéis associados)
<i>Caso o Modo de Distribuição seja para membros ou papéis específicos:</i>
Lista de Destinos da mensagem, composta pelos nomes dos membros ou papéis que deverão receber a mensagem
Modo de Ordenação, que determina se a distribuição da mensagem deve ser feita com ordenação ou sem ordenação
<i>Caso a mensagem corresponda a uma interogação de grupo:</i>
Modo de Colagem de Respostas, que especifica a forma como serão coletadas e entregues as respostas correspondentes à interogação
CONTEÚDO DA MENSAGEM
Seqüência de bytes, que representa a informação trocada entre os membros

Figura 5.1: Estrutura de dados para representar mensagens de grupo

### Eventos de Comunicação de Grupo

Toda a comunicação entre os objetos de suporte a grupos é realizada em termos de *eventos*, os quais constituem a representação, no nível de suporte a grupos, das informações trocadas entre os usuários. Invocações dos usuários para a realização de algum serviço de grupo, tais como mensagens a serem distribuídas ao grupo e pedidos de entrada ou saída de membros, são transformados em eventos ao serem manipulados pelos objetos do Serviço de Suporte a Grupos. Ao chegarem aos seus destinos, os eventos são novamente transformados à sua forma original. A Figura 5.2 ilustra a relação entre eventos e invocações de serviços. Eventos consistem de estruturas de dados que encapsulam



Figura 5.2: Correspondência entre eventos e interações de grupo

os pedidos de serviços originados no contexto dos usuários, adicionando a eles informações usadas pelo Suporte a Grupos para permitir o controle da execução dos serviços, conforme mostrado na Figura 5.3.

#### 5.1.2 Interfaces dos Objetos

O acesso aos serviços que um objeto provê é feito através de operações definidas em suas interfaces. A troca de informações entre os objetos é também realizada através da invocação de operações

CABEÇALHO
<p>Nome do Grupo em que ocorre o evento</p> <p>Nome do Membro em favor do qual o evento foi gerado</p> <p>Tipo do Evento, que consiste de uma enumeração com os seguintes valores, representando o tipo de serviço cuja invocação gerou o evento:  <i>MSG_INTERROGAÇÃO, MSG_RESPOSTA, MSG_ANÚNCIO, ENTRADA_MEMBRO, SAÍDA_MEMBRO, MUDANÇA_POLÍTICA, MUDANÇA_PAPEL, MUDANÇA_AUTORIZAÇÃO, FIM_GRUPO, ESTAB_CANAL, FECHA_CANAL</i></p> <p>Número de Seqüência Local, que determina a ordem do evento entre os demais eventos gerados pelo mesmo objeto</p> <p>Número de Seqüência Global, que determina a ordem do evento no conjunto de eventos do grupo em questão</p>
CORPO DO EVENTO
<p>Representado por uma <i>união discriminada</i>, tendo como seletor o Tipo do Evento:</p> <p>Caso <i>MSG_INTERROGAÇÃO, MSG_RESPOSTA</i> ou <i>MSG_ANÚNCIO</i>:  Estrutura de Mensagem</p> <p>Caso <i>ENTRADA_MEMBRO</i>:  Nome e Papéis do novo membro</p> <p>Caso <i>SAÍDA_MEMBRO</i>:  Nome do membro a sair</p> <p>Caso <i>MUDANÇA_POLÍTICA, MUDANÇA_PAPEL</i> ou <i>MUDANÇA_AUTORIZAÇÃO</i>:  Nome do serviço cujas políticas serão alteradas  Tipo da alteração (adição ou remoção)  Nome da política, papel ou autorização a ser adicionada ou removida</p> <p>Caso <i>FIM_GRUPO</i>:  (nenhuma informação adicional)</p> <p>Caso <i>ESTAB_CANAL</i>:  papel associado ao novo canal  parâmetros do canal (opacos ao serviço de suporte a grupos)</p> <p>Caso <i>FECHA_CANAL</i>:  identificador do canal a ser fechado</p>

Figura 5.3: Estrutura de dados para representar eventos de grupo

apropriadas nas suas interfaces, sendo as informações passadas como parâmetros das operações.

Os objetos de suporte a grupos com os quais as aplicações podem interagir possuem duas interfaces distintas: uma *interface externa*, com operações para o provimento dos serviços de grupo, e outra, *interna*, exclusiva para a comunicação entre os objetos de suporte a grupos para a realização dos serviços. O isolamento das duas interfaces possibilita adicionar segurança ao objeto, no sentido de não permitir que objetos externos à configuração de um grupo possam invocar operações internas e alterar indevidamente o estado do grupo. Por outro lado, os objetos que interagem apenas com outros objetos de suporte a grupos possuem apenas uma interface (a interface interna), para uso pelos outros objetos de suporte a grupos. O acesso a estas interfaces é obtido através da *referência à específica interface* de cada objeto.

### 5.1.3 Replicação Passiva de Objetos

A tolerância a falhas de alguns dos objetos de suporte a grupos é obtida através de uma técnica de *replicação passiva* (ou *replicação de cópia primária* [Jal94]), onde cada objeto tolerante a falhas tem várias cópias: uma ativa (que efetivamente provê os serviços do objeto) e as demais, passivas

(capazes de tomar o lugar da cópia ativa caso ela falhe). No caso dos objetos de suporte a grupos, cada objeto crítico têm *uma réplica passiva*. O estado da réplica é mantido sincronizado com o estado do objeto ativo através do seguinte protocolo para a recepção e processamento de eventos pelos objetos:

1. o objeto ativo recebe um evento (as aplicações possuem a referência de interface apenas para a réplica ativa)
2. o objeto ativo envia o evento para sua réplica passiva
3. caso a réplica falhe neste ponto, uma nova réplica é criada
4. a réplica passiva armazena o evento e confirma sua recepção
5. o objeto ativo efetiva o processamento do evento

Caso ocorra uma falha da réplica ativa, a réplica passiva é ativada e seu identificador (para fins de interação) é difundido para todos os clientes do objeto. Desta forma, a menos que ocorra uma falha simultânea do objeto e de sua réplica, garante-se que um objeto replicado pode sempre ser recuperado (sem perder informações de estado), de maneira transparente com relação a seus clientes.

## 5.2 A Configuração de Objetos de Suporte a Grupos

Conforme visto no capítulo 4, o conjunto de serviços de suporte a grupos está classificado em dois subconjuntos relacionados: *serviços de estruturação* e *serviços de comunicação*. Esta classificação serviu também como base para a modelagem e estruturação das classes de objetos que compõem o Serviço de Suporte a Grupos.

Os serviços de estruturação de grupos são realizados através de um objeto *servidor de estruturação de grupos*, denominado de *SGsvc*, cuja existência é permanente, independentemente de existirem ou não grupos em funcionamento. Este objeto é o responsável por servir pedidos de criação e finalização de grupos, bem como de alteração da estrutura de grupos existentes com a entrada ou saída de membros, sendo também responsável por assegurar a unicidade dos nomes usados para identificar os objetos de suporte a grupos. Um objeto *SGsvc* atua no contexto de um *domínio de grupos*, definido como um ambiente distribuído (não necessariamente determinado pela estrutura da rede subjacente) onde se localizam os potenciais usuários das aplicações cooperativas em questão. Esta definição está em conformidade com o conceito de *domínio de objetos*, introduzido no Modelo Descritivo do RM-ODP [ISO94b]. Um determinado grupo existe dentro de um domínio específico, não sendo suportados grupos com membros em domínios diferentes (por outro lado, um dado usuário pode fazer parte de grupos em domínios diferentes). O objeto *SGsvc* é replicado (segundo o protocolo de replicação passiva acima definido), de forma que, caso sua réplica ativa falhe, a réplica passiva pode tomar o seu lugar, mantendo a normalidade da operação dos grupos do domínio.

A realização dos serviços de comunicação de grupo se dá por meio de uma estrutura de objetos particular para cada grupo criado, a qual é estabelecida e gerenciada pelo servidor de estruturação de grupos (SGsvc). A interface entre os usuários e o Serviço de Suporte a Grupos é definida de tal forma que se torna independente da configuração que se adote para a estruturação dos objetos de suporte a comunicação em grupo, seja ela distribuída ou centralizada. A abordagem aqui apresentada adota uma configuração mista, onde os serviços que dependem de coordenação global no contexto do grupo são controlados por um elemento central, denominado de *objeto de suporte a grupos coordenador* (SGcoordenador). Os demais serviços são realizadas de maneira distribuída por objetos dedicados aos usuários membros de grupos, os quais são denominados de *objetos de suporte a grupos locais* (objetos SGlocal).

Para cada grupo é estabelecido um objeto SGcoordenador, replicado para tolerar falhas, o qual atua na imposição de ordem global a eventos que possam alterar o estado compartilhado do grupo, de forma que todos os seus membros observem a mesma seqüência de eventos. A realização destas funções de coordenação por meio de um objeto centralizado, que foi também adotada no sistema Amoeba [KTV93], tem a vantagem de simplificar a implementação, não afetando significativamente o desempenho da execução dos serviços.

A interação direta de um usuário com os grupos dos quais ele participa é realizada pelo seu respectivo objeto SGlocal, o qual é responsável por receber invocações dos usuários, servindo aquelas que não necessitam de coordenação global e repassando as demais para o SGcoordenador do grupo apropriado. A identificação do grupo para o qual uma interação deve ser direcionada é feita através do *nome do grupo*, que é único no seu domínio. Observe que, para cada usuário membro de grupos, existe apenas um SGlocal, que atua com relação a todos os grupos dos quais o usuário é membro. Os objetos SGlocal tratam também da recepção de eventos e colagem de mensagens que chegam para o seu usuário associado. No caso de eventos ordenados, ele é responsável por sua entrega ao usuário na ordem estabelecida pelo SGcoordenador.

Nesta estrutura de objetos, *eventos de comunicação de grupo* são gerados tanto pelos SGlocal (no caso de eventos que encapsulam mensagens de grupo) quanto pelo SGsvc (no caso de eventos que encapsulam informações sobre um pedido de serviço de estruturação).

Um usuário é representado, no contexto de cada grupo do qual é membro, por um particular objeto de aplicação, cuja referência de objeto é conhecida pelo sistema de suporte a grupos. Este objeto constitui, efetivamente, o componente da aplicação cooperativa que interage com o suporte a grupos em favor do usuário, podendo ou não ser reutilizado para representar o usuário em mais de um grupo. A identificação dos usuários é feita através de seu *nome de usuário*, que deve ser único no domínio e é usado para identificar os objetos de aplicação que ele usa para interagir com os grupos dos quais é membro (o nome deste objeto da aplicação cooperativa é composto pelo nome do membro, concatenado com o nome do grupo em questão).

Um outro objeto da estrutura de suporte a grupos consiste do *Tratador de Falhas*, que, como o SGsvc, existe permanentemente no ambiente de grupos. O tratador de falhas é invocado pelos demais objetos de suporte a grupos sempre que for notada alguma falha na configuração de um grupo. Sua função consiste, então, em identificar a falha ocorrida e realizar as devidas ações para a recuperação do sistema.

A Figura 5.4 ilustra, em uma notação semelhante à introduzida por [RBP<sup>+</sup>91], o relacionamento entre estas três classes de objetos de suporte a grupos, juntamente com as entidades “usuário membro” e “domínio de grupos” (as entidades que aparecem em pontilhado não constituem objetos propriamente ditos, mas os conceitos associados).

A Figura 5.5, por sua vez, ilustra o padrão de interações entre os objetos de suporte a grupos e os usuários, através dos *bindings* estabelecidos entre eles (representados pelas setas). A figura ressalta o papel dos objetos SGlocal como interface entre usuários e o Serviço de Suporte a Grupos (para os serviços de comunicação de grupo), bem como a função do SGcoordenador na ordenação da comunicação entre os membros do grupo. Observe-se, entretanto, que, nos casos em que não se exige coordenação global, a comunicação se dá diretamente entre os SGlocal envolvidos. O objeto SGsvc é acessível a todos os demais objetos do sistema (inclusive aos usuários, para invocação de serviços de estruturação de grupos). Por outro lado, o tratador de falhas é um objeto interno ao suporte a grupos, não podendo ser invocado pelos usuários. Além disso, conforme se observa na figura, para efeitos de suporte a grupos, um *membro de grupo* é considerado como a associação entre um usuário (no caso, o objeto de aplicação) e seu respectivo SGlocal.

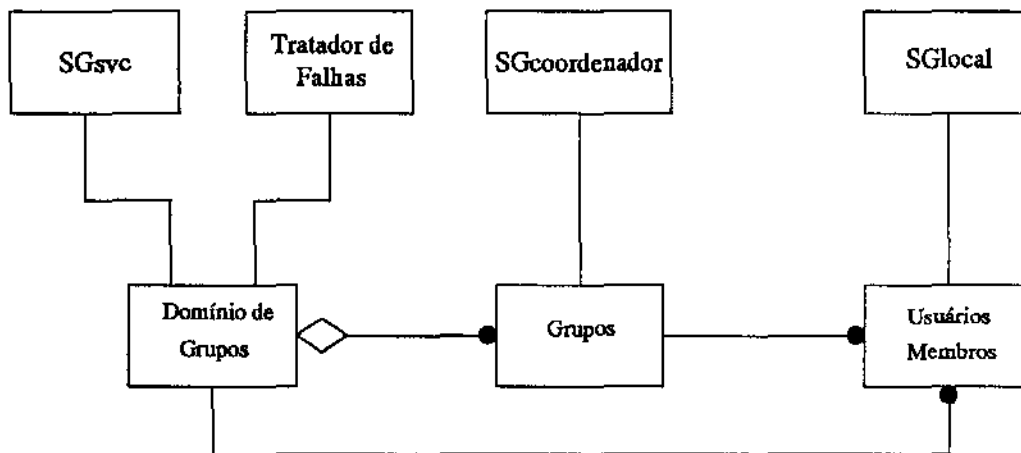


Figura 5.4: Modelo de objetos de Suporte a Grupos

As seções seguintes descrevem em detalhes os aspectos internos dos objetos e da realização dos serviços de suporte a grupos, considerando as funções desempenhadas por estes objetos, bem como as interações entre eles.

### 5.3 Estrutura e Interface dos Objetos

A seção anterior mostrou a estrutura global do modelo de objetos de suporte a grupos adotado, apresentando conceitualmente os inter-relacionamentos existentes entre estes objetos, bem como entre eles e seus usuários. Esta seção tratará de apresentar a composição destes objetos em termos das estruturas de dados neles encapsuladas, juntamente com suas interfaces para acesso aos serviços

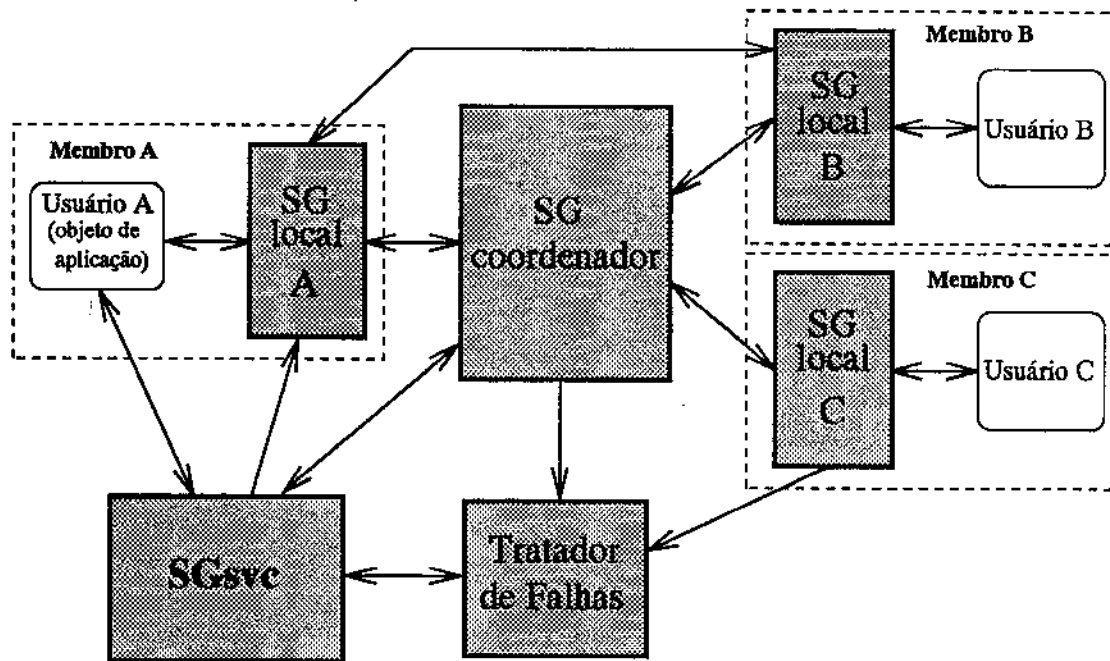


Figura 5.5: Interações entre os objetos envolvidos no Serviço de Suporte a Grupos

de suporte a grupos.

### 5.3.1 Objeto SGsvc

A função básica deste objeto está relacionada com a criação e manutenção da infra-estrutura de suporte aos grupos de um domínio. O SGsvc é o objeto responsável por atender a pedidos de criação de grupos, entrada de novos membros em grupos existentes e saída de membros de grupos, entre outros, bem como pela aplicação das políticas referentes a estes serviços. Para tanto, ele encapsula informações que descrevem a composição dos grupos existentes em seu domínio, assim como informações sobre os membros destes grupos. Basicamente, estas informações são organizadas em duas listas, a lista de grupos e a lista de usuários do domínio.

Sobre um determinado grupo são mantidas as seguintes informações na lista de grupos do domínio:

- nome do grupo;
- nome do usuário criador do grupo;
- referência à interface interna do objeto SGcoordenador do grupo;
- lista de membros do grupo, cujos nós mantêm as seguintes informações sobre os membros, as quais são específicas do grupo em questão:



- índice da entrada correspondente ao membro na lista de usuários do domínio;
  - lista com os nomes dos papéis associados ao usuário membro neste grupo;
  - referência à interface do objeto de aplicação que representa o usuário neste grupo específico;
- lista de políticas de serviço associadas ao grupo, consistindo de nomes que identificam uma política particular entre o conjunto de políticas disponíveis;
  - lista de *canais* de fluxo contínuo estabelecidos no grupo juntamente com o *papel* associado a cada canal.

Além destas informações, o SGsvc mantém um *contador usado para atribuir números de seqüência aos eventos* por ele gerados e enviados para o SGcoordenador do grupo. Este número é útil para garantir que o SGcoordenador processe estes eventos na ordem em que foram gerados (ordenação *FIFO* com respeito ao SGsvc).

A lista de usuários do domínio, por sua vez, mantém informações sobre usuários, as quais são constantes para todos os grupos dos quais o usuário é membro. Informações sobre um usuário são inseridas nesta lista por ocasião da primeira vez que ele se torna membro de um grupo no domínio do SGsvc. Estas informações consistem de:

- nome do usuário, através do qual outros usuários ou o próprio Serviço de Suporte a Grupos podem identificar um membro de um grupo;
- referência à interface interna do objeto do SGlocal associado ao usuário;
- lista com os nomes dos grupos dos quais o usuário participa.

Estas informações são mantidas no SGsvc para que possam ser propagadas para os demais componentes da infra-estrutura de suporte a grupos, bem como para as aplicações que representam os membros de grupos em situações como a criação de novos grupos, entrada de novos membros em grupos e recuperação de falhas. O SGsvc mantém também uma referência à interface do Tratador de Falhas para a notificação de possíveis falhas em algum objeto da estrutura de suporte a grupos.

A *interface externa* do objeto SGsvc é composta das operações mostradas na Tabela 5.1, as quais oferecem às aplicações acesso aos serviços de estruturação de grupos. A descrição dos serviços de estruturação de grupos é apresentada no capítulo anterior. A tabela mostra a lista de parâmetros usados na invocação destas operações, sendo que o primeiro parâmetro de cada uma sempre identifica o usuário invocador (exceto no caso da última operação), o que é usado para fins de verificação de autorização. Os serviços de manutenção de canais, embora relacionados com a comunicação de grupo, têm, no nível de suporte a grupos, um tratamento meramente estrutural, o que justifica a colocação das operações correspondentes na interface externa do SGsvc. Observe que estas operações não retornam resultado (são operações assíncronas ou não-bloqueantes). A confirmação de realização dos serviços, neste caso, é retornada através de operações correspondentes na interface dos objetos de aplicação. Isto impede que os objetos invocadores de serviços permaneçam bloqueados

Tabela 5.1: Operações na interface externa do SGsvc

OPERAÇÃO	PARÂMETROS
<b>Criação de Grupo</b>	nome do usuário que propôs a criação do grupo, nome do grupo a ser criado, lista de membros proposta para o novo grupo, lista de políticas proposta para o grupo, lista de autorizações para os serviços de grupo
<b>Finalização de Grupo</b>	nome do usuário que propôs a finalização do grupo, nome do grupo a ser finalizado
<b>Entrada de Membro em um Grupo</b>	nome do usuário que propôs a entrada do membro, nome do grupo, nome do novo membro, lista de papéis do novo membro
<b>Saída de Membro de um Grupo</b>	nome do usuário que propôs a saída do membro, nome do grupo, nome do membro a sair do grupo
<b>Mudança da Política associada a um Serviço</b>	nome do usuário que propôs a alteração, nome do grupo, nome do serviço envolvido, nome da nova política
<b>Alteração dos Papéis de um Membro</b>	nome do usuário que propôs a alteração, nome do grupo, nome do membro cujos papéis serão alterados, tipo da alteração (adição ou remoção de um papel), nome do papel (novo ou a ser removido)
<b>Mudança das Autorizações associadas a um Serviço</b>	nome do usuário que propôs a mudança, nome do grupo, nome do serviço envolvido, tipo da mudança (adição ou remoção de autorização), nome do papel ou membro autorizado (ou não)
<b>Estabelecimento de Canal</b>	nome do usuário que propôs o canal, nome do grupo, papel associado ao canal, características do canal (opacas ao serviço de suporte a grupos, são apenas repassadas para a camada de suporte a canais)
<b>Fechamento de Canal</b>	nome do usuário que propôs o fechamento do canal, nome do grupo, identificador do canal (que é fornecido a cada um dos membros ligados, durante o estabelecimento do canal)
<b>Informações sobre Grupos</b>	tipo da informação solicitada (lista de grupos ou informações completas sobre um grupo específico), nome do grupo (opcional - caso informações sobre um grupo) Retorno: (depende do tipo da informação solicitada) lista de grupos do domínio, ou lista de membros, políticas e autorizações do grupo

durante a execução dos serviços. Uma operação adicional é provida na interface para que usuários do suporte a grupos possam obter informações sobre os grupos de um domínio.

A *interface interna* do SGsvc, mostrada na Tabela 5.2 contém uma operação que permite ao SGcoordenador confirmar a distribuição ordenada de um evento que, neste caso, representa a efetiva execução de um serviço que altera a estrutura de um grupo. (O SGsvc deve considerar a alteração na estrutura do grupo efetivada apenas a partir deste momento, quando todos os membros do grupo observaram o evento, de forma a evitar inconsistência entre ele e os membros do grupo).

Tabela 5.2: Operações na interface interna do SGsvc

OPERAÇÃO	PARÂMETROS
Confirmação da Distribuição de um Evento Correspondente a um Serviço de Estruturação	<i>nome do grupo em questão, cópia do evento distribuído</i>

### 5.3.2 Objeto SGcoordenador

O SGcoordenador é o objeto central em um grupo. Através dele passam mensagens e outros eventos que devem ser distribuídos ao grupo obedecendo a ordem global de eventos (eventos não sujeitos à ordem global não passam pelo SGcoordenador). Basicamente, sua função consiste em receber pedidos de distribuição de eventos (procedentes dos SGlocal do grupo ou do SGsvc), atribuir a estes eventos um número de seqüência global no contexto do grupo e, em seguida, distribuí-los para os (SGlocal) membros apropriados no grupo. O número de seqüência atribuído ao evento exprime a sua ordem entre o conjunto total de eventos ordenados do respectivo grupo.

Para realização de suas funções o SGcoordenador mantém as seguintes informações sobre a composição do grupo, bem como sobre outros objetos de suporte a grupos:

- nome do grupo;
- nome do usuário criador do grupo;
- lista de membros do grupo, cujos nós são compostos pelas seguintes informações para endereçamento dos membros do grupo na distribuição dos eventos:
  - nome do membro;
  - lista com os nomes dos papéis associados ao usuário membro neste grupo;
  - referência à interface interna do objeto SGlocal associado ao membro;
- referência à interface interna do objeto SGsvc;
- referência à interface do Tratador de Falhas, para a notificação de possíveis falhas no grupo;
- política para o serviço de distribuição de mensagens com ordenação.

Tabela 5.3: Operações na interface (interna) do SGcoordenador

OPERAÇÃO	DESCRIÇÃO	PARÂMETROS
<b>Informações sobre o Grupo</b>	Invocada pelo SGsvc para fornecer ao SGcoordenador informações sobre o grupo (na criação do grupo ou na recuperação de falhas)	<i>nome do grupo, nome do usuário criador do grupo, lista de membros do grupo (com os respectivos papéis), política de distribuição de mensagens ordenadas</i>
<b>Distribuição Ordenada de um Evento</b>	Usada pelos objetos SGlocal para solicitar a distribuição ordenada de mensagens, bem como pelo SGsvc para a distribuição de eventos de mudança na estrutura de um grupo (que exige ordenação)	<i>estrutura de evento de grupo (que pode corresponder tanto a uma mensagem quanto a uma notificação de serviço de estruturação de um grupo a ser distribuída)</i>

Além destas informações específicas sobre a constituição do grupo, o SGcoordenador mantém também informações para uso na ordenação dos eventos por ele distribuídos ao grupo, bem como para possibilitar a recuperação de falhas de comunicação com os membros. Uma vez que o modelo de grupos adotado (ver capítulo 4) permite a distribuição de eventos apenas para parte dos membros de um grupo, é necessário que estas informações sejam particulares para cada membro. Desta forma, o lugar mais apropriado para mantê-las é a própria lista de membros do grupo, cujos nós são adicionados dos seguintes campos:

- contador de seqüência dos eventos enviados para o membro correspondente, incrementado a cada novo evento para ele distribuído; é usado para atribuir ordem global entre os demais eventos do grupo;
- número de seqüência do último evento recebido deste membro (atribuído pelo SGlocal do membro); é usado para detectar eventos repetidos (cujo envio foi feito mais de uma vez pelo SGlocal devido à detecção de possíveis falhas), bem como para possibilitar a ordenação *FIFO* dos eventos originados por este membro;
- lista de eventos cujo envio para o membro em questão (no processo de distribuição) foi retardado devido à ocorrência de falhas de comunicação;
- lista de eventos cuja distribuição com ordenação foi solicitada pelo SGlocal do membro em questão (ou pelo SGsvc), mas que foram recebidos, pelo SGcoordenador, fora de seqüência, com respeito à ordem local dos eventos gerados pelo membro (ordem *FIFO*); a distribuição destes eventos fica pendente até que os eventos precedentes, na ordem *FIFO* sejam recebidos e distribuídos.

O SGcoordenador possui apenas a interface interna, com operações disponíveis apenas para os demais objetos da configuração de suporte ao grupo. As operações desta interface são apresentadas na tabela 5.3.

### 5.3.3 Objeto SGlocal

O SGlocal é o objeto do Serviço de Suporte a Grupos que está em contato direto com os objetos das aplicações cooperativas para o acesso aos serviços de comunicação de grupo. Este objeto constitui o componente distribuído do suporte a grupos e sua função principal consiste em dar acesso ao serviço de distribuição de mensagens, enviando, recebendo e fazendo a *colagem* de mensagens de grupo. No caso da distribuição de mensagens com ordenação, elas são enviadas para o SGcoordenador, que se encarrega da efetiva comunicação com o grupo; no caso de mensagens sem ordenação, todo o serviço de distribuição é realizado no escopo dos objetos SGlocal que fazem parte do grupo. Além destas funções relativas à comunicação de grupo, o SGlocal é também responsável pela realização de funções adicionais na execução dos serviços de estruturação de grupos, conforme mostrado na seção seguinte. Um SGlocal é criado quando seu usuário torna-se, pela primeira vez, membro de algum grupo (quando ainda não existe um SGlocal associado a ele) e seu tempo de vida está relacionado com o período em que o usuário permanece como membro de algum grupo (no domínio). O nome do objeto SGlocal é o mesmo do usuário correspondente.

O estado do objeto SGlocal é constituído de informações que permitem a ele identificar os grupos dos quais o usuário é membro, juntamente com a composição do conjunto de membros destes grupos. Basicamente, estas informações são expressas na *lista de grupos* dos quais o usuário é membro, cujos nós têm a seguinte constituição:

- nome do grupo;
- referência à interface do objeto de aplicação que representa o usuário neste grupo específico;
- referência à interface do SGcoordenador do grupo, que permite a comunicação com este objeto para a distribuição de mensagens com ordenação;
- nome do usuário criador do grupo;
- lista de membros do grupo, cujos nós têm a seguinte composição:
  - nome do membro;
  - lista com os nomes dos papéis associados ao membro;
  - referência à interface interna do SGlocal associado ao membro;

esta lista é usada para o endereçamento dos membros do grupo na execução do serviço de distribuição de mensagens sem ordenação;

- políticas associadas ao serviço de distribuição de mensagens.

Além destas, o SGlocal mantém outras estruturas de dados que são usadas para efetuar a ordenação dos eventos de grupos (a ordem dos eventos é determinada pelo SGcoordenador, mas precisa ser obedecida pelos membros que recebem os eventos). Outras informações são também mantidas para permitir a recuperação de falhas de comunicação passíveis de ocorrerem no envio e recebimento de mensagens ou outros eventos (falhas estruturais são notificadas ao Tratador de

Falhas, sendo que o *SGlocal* mantém uma referência à interface deste objeto). Para cada grupo na lista de grupos são mantidas as seguintes informações:

- informações sobre a ordem dos eventos recebidos:
  - número de seqüência do último evento recebido do *SGcoordenador*; esta informação permite ao *SGlocal* detectar eventos que chegaram fora da ordem causal-total e também eventos repetidos, dando-lhes o tratamento adequado;
  - para cada membro do grupo: número de seqüência da última mensagem recebida do respectivo *SGlocal* (pelo serviço de distribuição sem ordenação), para permitir o descarte de mensagens repetidas, além de permitir ao *SGlocal* definir uma ordem de entrega das mensagens de acordo com a ordem em que foram enviadas por cada membro (ordenação *FIFO*); estes números de seqüência são mantidos na lista de membros do grupo em questão;
- informações sobre a ordem de envio das mensagens para o grupo:
  - número de seqüência local da próxima mensagem com ordenação a ser distribuída para o grupo(ou seja, a ser enviada para o *SGcoordenador* para distribuição), de forma a permitir a detecção, pelo *SGcoordenador*, de falhas ou de mensagens repetidas, além de permitir ao *SGcoordenador* ordenar as mensagens recebidas deste membro (em uma ordem do tipo *FIFO*);
  - para cada membro do grupo: o número de seqüência da próxima mensagem (sem ordenação) a ser enviada para o respectivo *SGlocal*, de forma a permitir a detecção de falhas ou mensagens repetidas, além de permitir ao *SGlocal* (receptor) entregar estas mensagens segundo uma ordem *FIFO* (estes números de seqüência são mantidos na lista de membros de cada grupo);
- listas de eventos recebidos do grupo, mas pendentes por algum motivo:
  - *eventos com ordenação* que chegaram fora de ordem ou, no caso de mensagens, que ainda não foram entregues à aplicação devido a falhas de comunicação; esta lista é ordenada pelo número de seqüência atribuído pelo *SGcoordenador* aos eventos;
  - *eventos sem ordenação*, correspondentes a mensagens (não ordenadas) com entrega pendente devido a falhas de comunicação;
- listas de mensagens com envio pendente:
  - mensagens a serem distribuídas com ordenação cujo envio para o *SGcoordenador* está pendente devido à ocorrência de alguma falha;

- para cada membro do grupo: uma lista de mensagens cujo envio para o SGlocal correspondente, no processo de distribuição sem ordenação, está pendente devido a falhas de comunicação (estas listas são mantidas dentro da lista de membros do grupo).

Estas listas são ordenadas de acordo com a ordem de geração das mensagens pela aplicação (para assegurar ordenação *FIFO*);

- lista de mensagens correspondentes a interrogações, que foram enviadas para o grupo e cujas respostas estão em processo de *colagem*; cada nó desta lista mantém uma sub-lista com as respostas já recebidas; as mensagens são retiradas da lista quando a condição de colagem é satisfeita, sendo as respostas (coladas) entregues à aplicação;
- lista de mensagens correspondentes a interrogações recebidas do grupo e cujas respostas ainda não foram enviadas; esta lista permite a correta associação de uma resposta (a ser retornada) à interrogação que a originou.

A Figura 5.6 ilustra o esquema de listas de mensagens pendentes utilizado pelo SGlocal, mostrando os objetos fonte e destino das mensagens armazenadas nas listas.

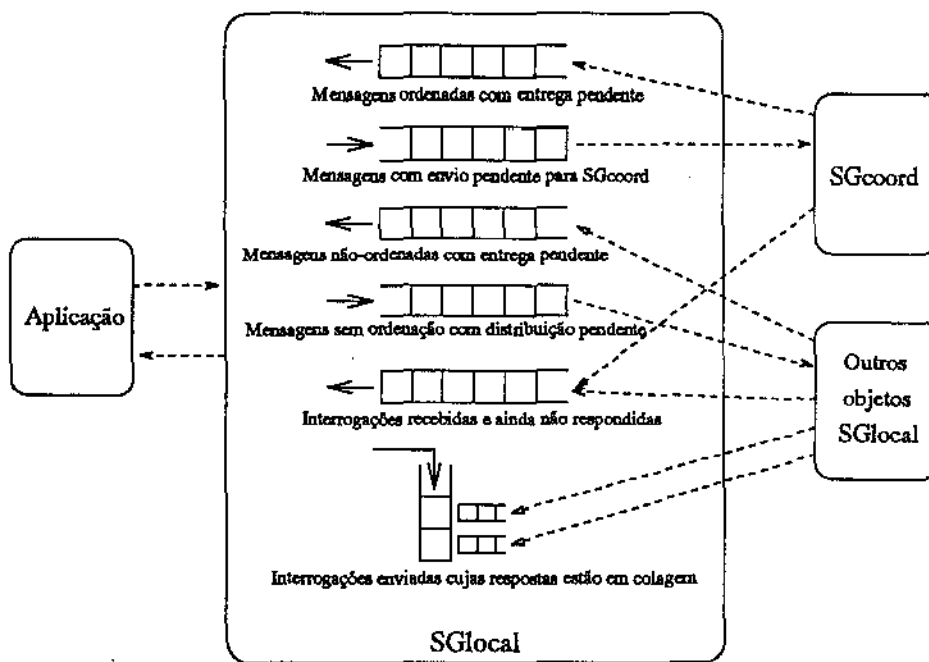


Figura 5.6: Listas de mensagens pendentes no SGlocal

O comportamento do objeto SGlocal observado pelas aplicações está definido na *interface externa* do objeto, que consiste, basicamente, de operações para comunicação de grupo, como mostrado na tabela 5.4. A *interface interna* consiste das operações definidas na tabela 5.5, que permitem ao SGlocal receber eventos de grupo.

Tabela 5.4: Operações na interface externa do SGlocal

OPERAÇÃO	PARÂMETROS
Distribuição de Mensagem de Interrogação	<i>estrutura de mensagem</i> <b>Retorno:</b> identificador da interrogação distribuída; para a aplicação receber e associar corretamente a(s) resposta(s) (um valor de retorno igual a 0 indica que a mensagem não foi aceita para distribuição pelo SGcoordenador)
Distribuição de Mensagem de Anúncio	<i>estrutura de mensagem</i> <b>Retorno:</b> booleano - indica se a mensagem foi aceita para distribuição
Envio de Mensagem de Resposta	<i>estrutura de mensagem (apenas o campo conteúdo é requerido), identificador da interrogação respondida</i> <b>Retorno:</b> booleano - indica se a mensagem foi aceita para envio
Solicitação de Informações sobre a Estrutura de um Grupo	<i>nome do grupo</i> <b>Retorno:</b> listas de membros, papéis, políticas e autorizações

Tabela 5.5: Operações na interface interna do SGlocal

OPERAÇÃO	DESCRIÇÃO	PARÂMETROS
Informações sobre novo Grupo	Invocada pelo SGsvc, durante o processo de criação de um grupo ou entrada do membro neste grupo, para transferir informações sobre a constituição do grupo	<i>nome do grupo, nome do usuário criador do grupo, lista de membros do grupo (com os respectivos papéis), lista de políticas de grupo, lista de autorizações de serviço</i>
Indicação de Evento	Operação através da qual o SGlocal recebe eventos de grupo (invocada pelo SGcoordenador para a entrega de eventos ordenados e por outros SGlocal para eventos sem ordenação)	<i>estrutura de evento de grupo</i>
Inicialização de Novo Membro	Invocada pelo SGcoordenador para transferir informações de inicialização para novos membros, indicando o momento em que o novo membro pode começar a interagir com o grupo	<i>nome do grupo, informações de inicialização da aplicação (sob a forma de seqüência de bytes)</i>
Solicitação de Informações para Inicialização de Novo Membro	Invocada pelo SGcoordenador para obter (normalmente do membro coordenador) informações da aplicação para inicialização de novos membros de um grupo	<i>nome do grupo, nome do novo membro</i> <b>Retorno:</b> informações de inicialização (seqüência de bytes)



Tabela 5.6: Operações na interface (interna) do Tratador de Falhas

OPERAÇÃO	PARÂMETROS
Notificação de Falha	<i>classe do objeto falho (SGlocal, SGcoordenador, etc.), referência à interface do objeto falho</i>

### 5.3.4 Objeto Tratador de Falhas

O **Tratador de Falhas** é o objeto responsável por identificar a causa de falhas envolvendo os objetos do ambiente de suporte a grupos, provendo o necessário tratamento para a recuperação do sistema como um todo. As falhas são normalmente detectadas pelos demais objetos de suporte a grupos por meio da observação de ocorrências anormais na comunicação entre eles e, em seguida reportadas ao tratador de falhas.

Este objeto possui apenas a interface interna, mostrada na Tabela 5.6, com uma operação para notificação de falhas, que permite aos seus clientes (os demais objetos da configuração de suporte a grupos) informarem sobre a suspeita de falha de determinado objeto ou membro do grupo (o qual é identificado através do nome da sua classe e de sua referência de interface).

Como estruturas de informação, o **Tratador de Falhas** mantém uma referência à interface interna do objeto **SGsvc** do domínio, de forma que possa obter as informações necessárias sobre os grupos e membros afetados por falhas, e proceder com as ações de recuperação junto a este objeto.

### 5.3.5 A Interface dos Objetos de Aplicação

No sentido de permitir aos objetos de suporte a grupos invocarem os objetos de aplicação (que representam os membros de grupos) para a notificação de mensagens e outros eventos de um grupo, faz-se necessário que os objetos de aplicação exportem uma interface padronizada contendo um conjunto bem definido de operações com esta finalidade. Estas operações são descritas na tabela 5.7.

## 5.4 Descrição da Implementação dos Serviços

Os serviços de suporte a grupos são realizados de maneira distribuída, o que implica na cooperação entre os objetos **SGlocal**, **SGcoordenador** e **SGsvc**. Uma invocação de serviço originada por um usuário, ao ser recebida em uma das interfaces de serviço (dos **SGlocal** ou do **SGsvc**), recebe um tratamento apropriado e, dependendo do contexto envolvido, é repassada para outro(s) objeto(s) da estrutura de suporte a grupos, para completarem a execução do serviço.

Esta seção examina em detalhes os aspectos envolvidos nas interações entre estes objetos para a realização de cada um dos serviços implementados. São abordadas também questões de tolerância a falhas e ordenação de eventos, bem como as soluções adotadas no modelo de implementação para os problemas relacionados.

Tabela 5.7: Operações na interface dos objetos de aplicação

OPERAÇÃO	DESCRIÇÃO	PARÂMETROS
Consulta sobre Novo Grupo	Invocada pelo SGsvc para consultar os usuários sugeridos como membros sobre a criação do grupo (um usuário pode aceitar ou não participar do grupo)	nome do grupo, lista de membros proposta, nome do usuário criador do grupo Retorno: ACEITO/NÃO ACEITO
Resultado das Consultas	Usada pelo SGsvc para informar o usuário criador do grupo sobre o resultado das consultas de criação do grupo (permitindo a ele confirmar ou não a criação)	nome do grupo, lista de membros com participação confirmada no grupo. Retorno: CONFIRMA/ NÃO CONFIRMA
Aviso de Novo Grupo	Invocada pelo SGsvc para notificar a criação efetiva de um novo grupo; também usada pelo SGlocal para informar sobre a entrada do usuário em um grupo	nome do grupo, nome do usuário criador, lista de membros do grupo, lista de políticas, lista de autorizações, informações de inicialização da aplicação (usado apenas em caso de entrada do usuário em um grupo)
Consulta sobre Alteração na Estrutura de um Grupo	Invocada pelo SGsvc para consultar os membros do grupo sobre alguma mudança na estrutura do grupo, bem como sobre a finalização do grupo	nome do grupo, nome do usuário responsável, tipo da alteração, Caso entrada ou saída de membro: nome e papéis do membro Caso mudança de papéis: nome do membro afetado e nova lista de papéis Caso mudança de política ou autorização: nome do serviço e da política ou autorização a ser adicionada/removida Retorno: ACEITO/NÃO ACEITO
Notificação de Alteração na Estrutura de um Grupo	Invocada pelo SGlocal ou SGsvc para informar aos usuários a ocorrência de mudanças na estrutura de um grupo, bem como para notificar a saída do membro em questão de um grupo ou a finalização de um grupo	nome do grupo, tipo da alteração, Caso entrada ou saída de membro: nome e papéis do membro Caso mudança de papéis: nome do membro afetado e nova lista de papéis Caso mudança de política ou autorização: nome do serviço e da política ou autorização a ser adicionada/removida
Indicação de Mensagem	Invocada pelo SGlocal para entregar mensagens à aplicação (usado para mensagens dos três tipos - interrogações, anúncios e respostas)	Caso Interrogação: estrutura de mensagem, identificador da interrogação (para identificar a resposta a ser retornada) Caso Anúncio: estrutura de mensagem, Caso Resposta: identificador da interrogação que gerou a resposta lista de mensagens de resposta
Identificador de Canal Estabelecido ou Fechado	Invocada pelo SGsvc para informar o estabelecimento ou fechamento de um canal aos membros por ele ligados	nome do grupo, identificador do canal, papéis associados ao canal, características do canal
Solicitação de Informações para Inicialização de Novos Membros	Invocada pelo SGlocal para obter, de algum membro já existente no grupo, informações de inicialização para novos membros	nome do grupo, nome do novo membro Retorno: informações de inicialização (sob a forma de seqüência de bytes)

### 5.4.1 Ordenação de Eventos

Conforme visto no capítulo 4, o modelo de grupos para suportar aplicações cooperativas compreende a *ordenação causal-total* dos eventos de comunicação e de mudança da estrutura de grupos. Esta ordenação de eventos pode ser obtida através de uma combinação de *ordenação total* com *ordenação FIFO* [Ren93].

A componente de ordenação total é obtida através de uma abordagem semelhante àquela utilizada no sistema Amoeba [KTV93, Tan92], na qual um processo *seqüenciador de eventos* atua como intermediário de toda a comunicação de grupo que necessita de ordenação. Em nossa abordagem, o SGcoordenador desempenha esta função de seqüenciamento de eventos. O SGcoordenador adiciona aos eventos informação de ordem, representada por um **número de seqüência global**, que determina a ordem de cada evento entre os demais eventos do grupo; em seguida, o SGcoordenador distribui estes eventos para os membros do grupo. Os eventos a serem distribuídos com ordenação são gerados concorrentemente pelos objetos SGlocal do grupo, bem como pelo SGsvc. Estes eventos são enviados ao SGcoordenador através da operação de *Distribuição Ordenada de Eventos* da sua interface. Invocações concorrentes desta operação são servidas na ordem de chegada (isto é garantido pelos mecanismos de suporte à comunicação entre objetos, conforme será tratado no capítulo 6), o que impede a postergação indefinida de invocações.

A componente de ordenação FIFO é obtida através dos *números de seqüência locais* associados aos eventos (pelos SGlocal ou pelo SGsvc) antes de serem enviados ao SGcoordenador. Estes números de seqüência são utilizados pelo SGcoordenador para servir os eventos enviados por um objeto na ordem em que eles foram enviados.

Uma vez que o modelo de grupos adotado permite a distribuição de mensagens (eventos) para subconjuntos dos membros de um grupo, tem-se situações em que nem todos os membros participam de todas as interações de grupo. Com isto, os números de seqüência global associados aos eventos devem ser particularizados para cada membro do grupo, através da existência de um contador de seqüenciamento para cada um deles (seção 5.3). Portanto, um contador de seqüência global é mantido pelo SGcoordenador para cada membro do grupo; estes contadores são incrementados a cada novo evento a ser distribuído para os respectivos membros, sendo seu valor atribuído como o **número de seqüência global** aos eventos. Um SGlocal, ao receber um evento ordenado, examina o número de seqüência global do evento e, caso este seja o próximo evento esperado, entrega-o ao usuário. O evento deve ter sua entrega retardada pelo SGlocal, sendo colocado em uma lista de eventos pendentes, caso seu número de seqüência seja maior que o esperado (devido ao atraso na chegada de eventos anteriores na ordem global). Um evento permanece nesta lista até que todos os eventos anteriores na ordem global sejam devidamente recebidos e entregues. Após a recepção de um evento ordenado, a lista de eventos pendentes é examinada para verificar se o evento recebido libera outros eventos da lista, os quais devem então ser entregues à aplicação. O número de seqüência é útil também para que o SGlocal descarte eventos repetidos (enviados mais de uma vez pelo SGcoordenador).

A Figura 5.7 ilustra este mecanismo de ordenação causal-total de eventos, mostrando o papel do SGcoordenador como seqüenciador da comunicação de grupo, bem como o uso das filas de eventos

dos SGlocal (descritas na seção 5.3) na entrega ordenada de mensagens para os membros. As filas do SGcoordenador, por outro lado, são utilizadas para preservar a ordem *FIFO* dos eventos. A ordem em que os eventos são distribuídos para o grupo é determinada pela fila de invocações pendentes ao SGcoordenador, conforme visto na figura (a ordem estabelecida para os eventos, no exemplo da figura foi:  $S_2$ ,  $B_2$ ,  $D_3$ ). Cada evento é representado pelo nome do objeto que o gerou ( $S$  no caso do SGsvc, que gera os eventos de estruturação de grupos) associado a um subscrito que representa o número de seqüência local atribuído pelo objeto gerador do evento (por exemplo,  $S_2$  representa o segundo evento na seqüência de eventos gerados pelo SGsvc). Os números de seqüência global, representados pelo segundo subscrito associado aos eventos (na parte direita da figura), são atribuídos pelo SGcoordenador de forma a ordenar totalmente os eventos. Estes números de seqüência são atribuídos a partir de um contador de eventos individual para cada membro do grupo, conforme visto acima. O mesmo evento pode receber números de seqüência global diferentes quando distribuído para membros diferentes (por exemplo,  $S_{2,2}$ , enviado para o membro A, e  $S_{2,3}$ , enviado para o membro B); entretanto, a seqüência destes números observada por cada membro é consistente com os demais membros. Observe que nem todos os eventos são endereçados (e distribuídos) para todos os membros do grupo (por exemplo,  $D_3$  é distribuído apenas para os membros A, B e D). Cada objeto gerador de eventos do grupo está duplamente representado na figura, respectivamente no conjunto de geradores de eventos e no conjunto de receptores de eventos.

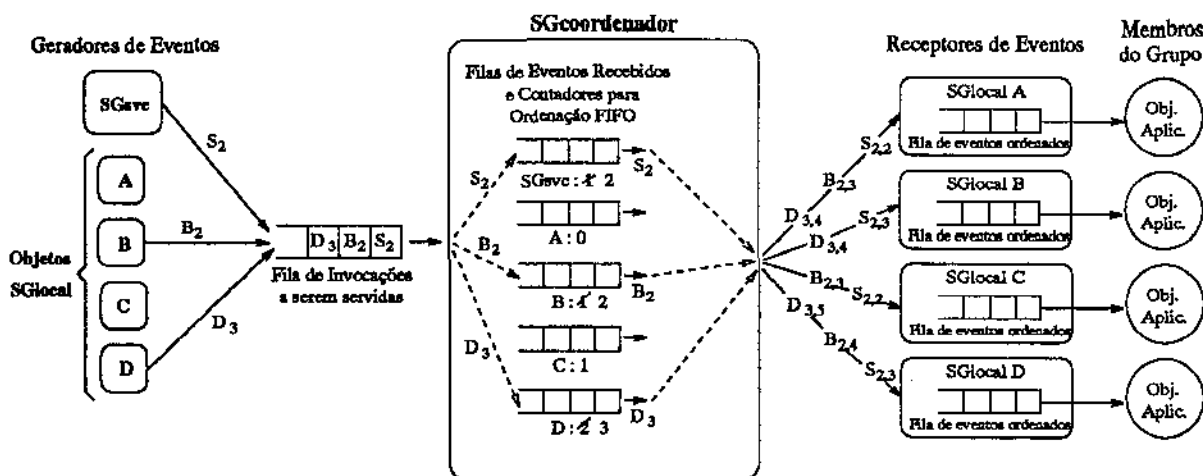


Figura 5.7: O mecanismo de ordenação causal-total de eventos

Uma limitação inerente dos protocolos que implicam em ordenação total de eventos refere-se ao fato de que o mecanismo de seqüenciamento de eventos, baseado em um coordenador central, atua apenas a partir do momento em que o coordenador recebe o evento (ou, mais precisamente, no momento em que a invocação correspondente ao evento é colocada na fila de invocações pendentes a serem servidas) [BCG91]. Desta forma, não é possível inferir sobre a ordem (cronológica) dos eventos antes deste momento. Exceção se faz no caso de eventos originados por um mesmo membro,

quando o número de seqüência local, adicionado pelo SGlocal, permite ao SGcoordenador colocar em ordem eventos recebidos de um mesmo membro. Entretanto, esta limitação não interfere na consistência das interações de grupo, uma vez que seja estabelecida uma ordem global (no contexto do grupo), que fará com que todos os membros observem todos os eventos na mesma ordem relativa.

#### 5.4.2 Detecção e Tratamento de Falhas

Dois tipos de falhas são passíveis de ocorrência em um ambiente de objetos distribuídos. O primeiro se refere a perdas de mensagens, enquanto que o segundo, mais grave, está relacionado com a impossibilidade de acesso a determinados objetos (devido a falhas na rede ou no próprio objeto). Estas falhas podem comprometer o funcionamento correto e consistente do sistema distribuído caso não sejam devidamente detectadas e tratadas, tendo seus efeitos removidos. O objetivo principal das ações de tratamento de falhas consiste em preservar a propriedade de *atomicidade da comunicação de grupo* (capítulo 3): ou todos os membros do grupo (aqueles endereçados) recebem um evento ou nenhum deles recebe.

No ambiente de suporte a grupos o tratamento de perdas de mensagens é confiado à plataforma de sistemas distribuídos subjacente que, no caso, deve prover um serviço de transporte confiável de mensagens. Por outro lado, falhas envolvendo a integridade ou acessibilidade de um objeto são também evidenciadas através de falhas na comunicação com o mesmo. A abordagem aqui adotada para determinar a ocorrência deste tipo de falha está baseada na realização de novas tentativas de comunicação com o objeto. Se após um certo número, pré-determinado, de tentativas a falha persistir, considera-se uma possível falha relativa ao objeto em questão. Neste caso, o Tratador de Falhas é invocado para confirmar a falha e executar as ações necessárias para o seu tratamento. A verificação do tipo de falha é então realizada através de tentativas de comunicação com o objeto suspeito, observando-se o resultado obtido. Caso confirmada a falha do objeto, o Tratador de Falhas procede com o seu tratamento de recuperação, que pode envolver desde a criação de um novo objeto para substituir o objeto falho, até a remoção do membro envolvido do grupo. A ação apropriada depende de qual seja o objeto envolvido na falha, sendo possíveis os seguintes casos:

- Falha dos objetos de suporte a grupo:
  - SGsvc e SGcoordenador: é ativada a réplica do objeto que falhou (cujo identificador é difundido para os clientes), sendo criada uma nova réplica passiva para o objeto;
  - SGlocal: neste caso, tem-se a possibilidade da perda dos eventos que porventura estejam nas filas de eventos pendentes do SGlocal (por exemplo, mensagens em processo de ordenação); isto impossibilita a recuperação consistente do objeto, devendo o membro correspondente ser removido do grupo;<sup>2</sup>
- Falha dos objetos de aplicação: neste caso, a recuperação da falha está fora do escopo do Suporte a Grupos, sendo que o único tratamento viável consiste na remoção do membro

---

<sup>2</sup>Uma alternativa seria fazer com que cada SGlocal registrasse todos os eventos com entrega pendente em um dispositivo de armazenamento não-volátil (em disco), de forma que um novo SGlocal poderia ser criado, sendo os eventos recuperados a partir deste registro.

correspondente do grupo, de forma a manter o restante do grupo em condições normais de interação; a remoção do membro falho é feita utilizando-se do *serviço de saída de membros de grupos*.

O tratamento de falhas ocorre concorrentemente com as atividades dos demais objetos da estrutura de suporte a grupos, o que significa que, durante o processo de detecção e recuperação da falha, o grupo pode estar em atividade (possivelmente com certa degradação dependendo do objeto falho e seu papel na estrutura de suporte ao grupo). Durante este período de recuperação, eventos que deveriam ser enviados ao objeto falho são mantidos em estado pendente pelo objeto responsável por seu envio. Estes eventos pendentes são liberados após a recuperação da falha, sendo enviados ao objeto destino, caso a recuperação da falha tenha mantido o membro no grupo, ou descartados em caso contrário.

### 5.4.3 Serviços de Estruturação de Grupos

Serviços de estruturação compreendem os serviços de criação e finalização de grupos, juntamente com os serviços que permitem a alteração da configuração original do grupo, como entrada ou saída de membros, e mudança de políticas, papéis ou autorizações. As funções básicas na execução destes serviços são realizadas no objeto *SGsvc*, permitindo isolar a execução dos serviços de estruturação da execução dos serviços de comunicação de grupo, de forma a não prejudicar a eficiência da comunicação de grupo. Por outro lado, a centralização das funcionalidades de estruturação de grupos no objeto *SGsvc* não deve prejudicar o desempenho dos grupos, se considerada a frequência relativamente baixa em que estes serviços são executados quando comparados com os serviços de comunicação.

Os eventos correspondentes a alterações na estrutura de um grupo devem ser ordenados com respeito aos eventos de comunicação de grupo, de forma a manter consistente o estado do grupo, conforme visto no capítulo 3. Isto implica na participação do *SGcoordenador* e dos *SGlocal* do grupo, para a *distribuição ordenada* dos eventos de alteração da estrutura do grupo para todos os seus membros.

Padrões de execução diferentes para estes serviços podem ser especificados através da associação apropriada de políticas a cada um deles. Um conjunto de políticas para cada um destes serviços é provido no nível de suporte a grupos, as quais foram descritas no capítulo 4. Nas seções subsequentes, os padrões de execução dos serviços de estruturação de grupos são descritos com base na opção de política mais elaborada, ou seja, aquela que determina algum tipo de consulta a todos os membros de um grupo para confirmar a realização de um serviço (as demais políticas definem padrões de execução que podem ser considerados subconjuntos daqueles descritos).

#### Criação de Grupos

O processo de criação de um novo grupo tem início através da invocação da operação de Criação de Grupos da interface externa do *SGsvc*. Os procedimentos que se seguem envolvem as três classes de objetos de suporte a grupos, sendo mostrados no diagrama da Figura 5.8 e descritos em seguida (os números entre parênteses referem-se às interações correspondentes no diagrama).

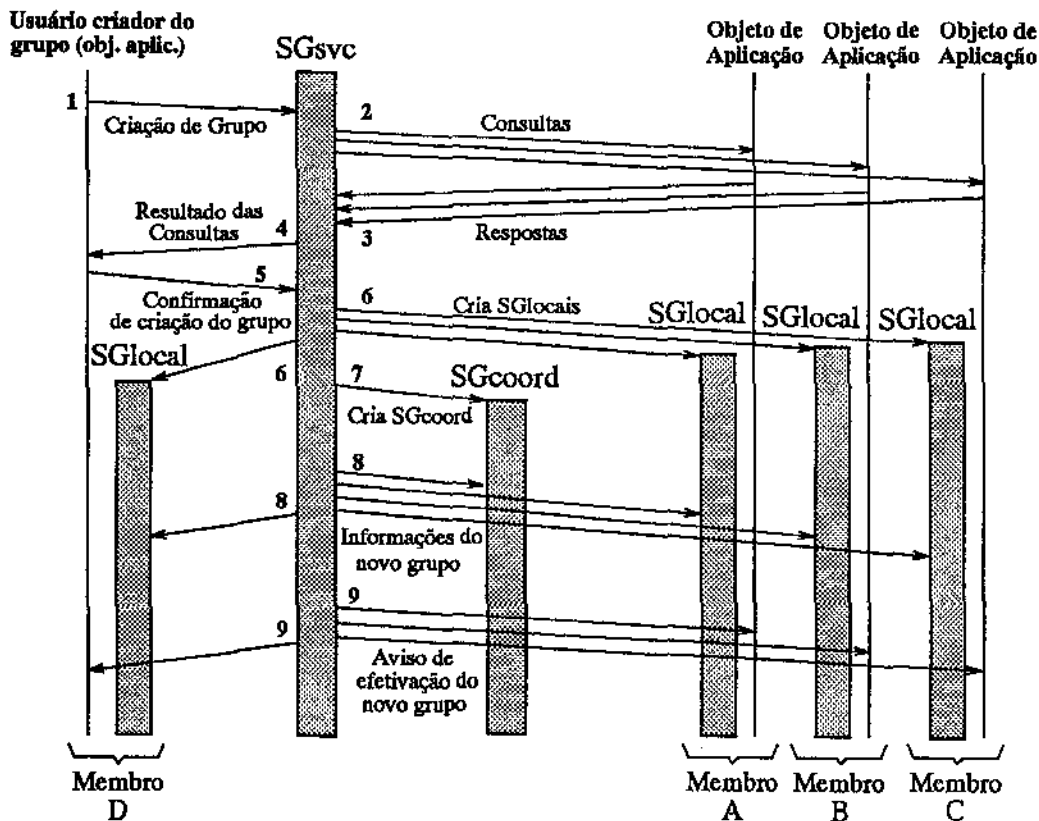


Figura 5.8: Criação de Grupo - diagrama temporal de execução do serviço

Inicialmente, o SGsvc precisa estabelecer uma conexão (*binding*) entre ele e cada um dos objetos de aplicação que representam os usuários membros sugeridos, de forma a possibilitar sua comunicação com eles. Falhas no estabelecimento de alguma conexão ou falhas dos objetos de aplicação implicam na impossibilidade do usuário em questão participar do grupo. Os usuários com os quais a conexão foi estabelecida com sucesso são consultados (2) e somente aqueles que aceitarem participar do grupo (3) permanecerão no processo de criação. Os resultados destas consultas são retornados para o usuário criador do grupo (4), de forma que ele possa determinar (5) se o processo de criação do grupo deve ou não prosseguir com os membros que confirmaram participação.

Em seguida, deve ser estabelecida a estrutura de suporte ao novo grupo, a qual consiste do SGcoordenador (7) e dos vários SGlocal (6) associados aos usuários membros do grupo. O objeto SGcoordenador normalmente é criado no mesmo local (*host*) onde se situa o membro com o papel de coordenador do grupo (a réplica passiva deste objeto é criada em um *host* diferente), sendo que uma conexão deve ser estabelecida entre o novo SGcoordenador e o SGsvc. Por sua vez, os objetos SGlocal são criados junto aos respectivos usuários membros do grupo. Observe que, uma vez que cada usuário tem apenas um SGlocal associado (que serve a todos os grupos dos quais ele participa), este só será criado caso ainda não exista. Se já existir o SGlocal associado a um

membro, são obtidas apenas referências às suas interfaces. O inter-relacionamento dos objetos desta estrutura de suporte é realizado nesta fase de inicialização, quando o SGsvc propaga para o SGcoordenador e para os SGlocal informações sobre o novo grupo, através das operações apropriadas nas interfaces internas destes objetos (8).

Finalmente, após estruturado o grupo, os membros (objetos de aplicação) que chegaram até o final do processo são informados (9), pelo SGsvc, da efetivação da criação do grupo, bem como de seus respectivos pontos de acesso ao mesmo, que correspondem aos objetos SGlocal associados. A partir deste instante, o grupo está pronto para entrar em operação. Note que, do ponto de vista dos usuários membros, o grupo só passa a existir como estrutura funcional após este aviso (as consultas sobre participação representam apenas a intenção de se criar o grupo, não significando necessariamente a sua efetivação).

### Entrada de Membros em um Grupo

O processo de entrada de membro em um grupo pode ser iniciado pelo próprio usuário a entrar no grupo ou por algum usuário já membro do grupo, através da invocação da operação de Entrada de Membros definida na interface externa do SGsvc. O diagrama da Figura 5.9 mostra as interações entre os objetos envolvidos no grupo para a execução do serviço (no diagrama, o processo de entrada de membro é iniciado pelo próprio usuário que deseja entrar no grupo).

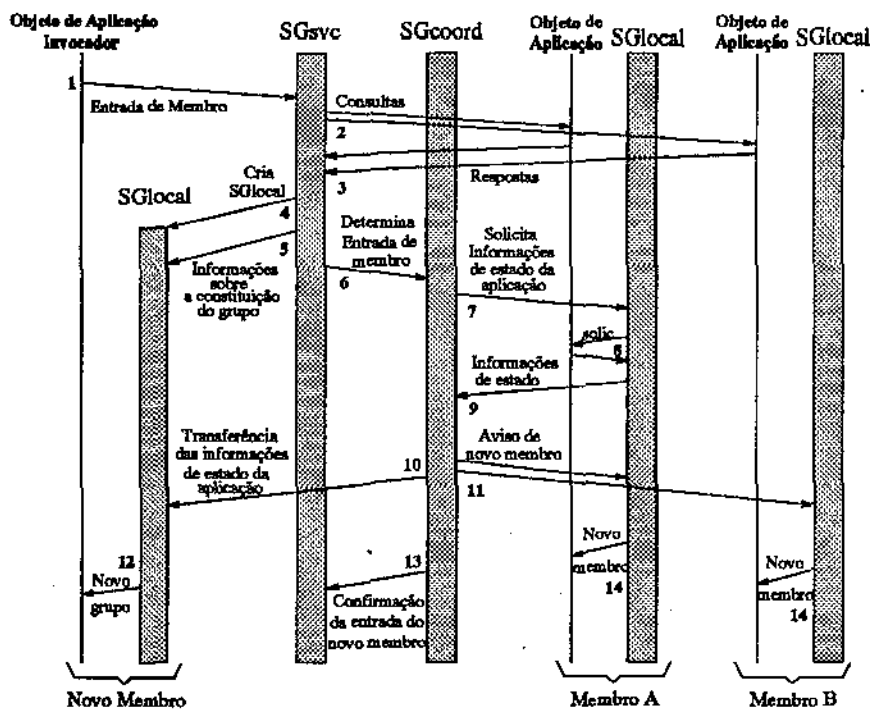


Figura 5.9: Entrada de Membro - diagrama temporal de execução do serviço



Inicialmente, o SGsvc verifica a validade dos parâmetros fornecidos na invocação (1) do serviço (nome do novo membro e nome do grupo) e verifica se o usuário invocador do serviço tem autorização para determinar sua execução. Em seguida, é iniciada a fase de consultas (2) aos membros do grupo (pela aplicação da política de entrada de membros) para determinar a continuação ou não do processo. A avaliação do resultado das consultas ao grupo (3) é feita segundo um critério de maioria simples (alternativamente, um critério diferente poderia ser estabelecido pela aplicação como parte da política).

Uma vez confirmada a entrada do novo membro no grupo, o SGsvc estabelece uma conexão (*bind*) com o objeto de aplicação que representa o membro (caso a conexão ainda não exista), de forma a habilitar futuras interações, servindo também para determinar se o objeto realmente existe e está acessível. Uma falha em obter esta conexão impossibilita a entrada do membro no grupo. Nos casos em que o novo membro é também um usuário novo no domínio do SGsvc, seu SGlocal é criado e devidamente inicializado (4), e suas informações são inseridas na lista de usuários de grupos do SGsvc. Caso o SGlocal já exista, apenas suas referências de interface são obtidas. Em qualquer dos casos, o referido SGlocal é provido com informações sobre a constituição do grupo em que está entrando (5), tais como a lista de membros e as políticas do grupo. O restante do processo de entrada de membro é dependente de coordenação entre os componentes do grupo, de forma que o novo membro perceba apenas aqueles eventos de grupo que ocorrerem após o instante lógico de sua entrada (como acontece em [ADM92] e [Maf93]. Portanto, o SGsvc invoca a operação de Distribuição de Eventos (6) na interface do SGcoordenador para que este distribua o evento de entrada de membro, com ordenação, para os SGlocal do grupo (inclusive para o novo SGlocal).

O SGcoordenador primeiramente adiciona as informações do novo membro (obtidas a partir das estruturas de dados do evento distribuído) à lista de membros por ele mantida. Em seguida, é iniciada a fase de **transferência do estado da aplicação** para o novo membro, quando o SGsvc invoca o SGlocal do membro coordenador do grupo (7) (o qual, por sua vez, invoca o objeto de aplicação correspondente (8)) para obter as informações de estado a serem transferidas. Caso ocorra alguma falha nesta solicitação, o SGcoordenador elege outro SGlocal do grupo para fornecer as informações. Durante esta fase de transferência de estado, toda a distribuição ordenada de mensagens é suspensa pelo SGcoordenador, o que é necessário para evitar que alterações sejam feitas no estado da aplicação que possam não ser observadas pelo novo membro<sup>3</sup>. Em [BJ87b] e em [Isi93] é apresentada uma abordagem semelhante para a transferência do estado de um grupo para novos membros. Após a obtenção das informações de estado (9), o SGcoordenador as transfere (10) para o SGlocal correspondente ao novo membro (o qual, por sua vez, as transfere para o correspondente objeto de aplicação (12)) e, no mesmo instante lógico, distribui o evento de entrada de novo membro para os demais membros do grupo (11). Após este instante, as interações de grupo através de mensagens ordenadas são novamente liberadas. Finalmente, é retornada a confirmação de distribuição do evento para o SGsvc (13) (através da invocação da operação apropriada na interface interna deste objeto).

---

<sup>3</sup>A distribuição não-ordenada de mensagens continua ativa neste período, o que não deve afetar a consistência da aplicação, pois as mensagens que alteram o estado compartilhado são, normalmente, distribuídas com ordenação apenas.

Cada SGlocal, ao receber o evento que indica a entrada do novo membro, realiza o tratamento de ordenação, de forma que a entrega do evento obedeça à ordem global do grupo. Em seguida, as informações do novo membro são inseridas na lista de membros do respectivo grupo, sendo gerada uma indicação de novo membro para o objeto de aplicação (14) (através da operação de Notificação de Alteração na Estrutura do Grupo). No caso do SGlocal que acaba de entrar no grupo, a recepção do evento de transferência de estado da aplicação sinaliza o início efetivo de sua atividade no referido grupo (a partir deste momento, o SGlocal pode aceitar invocações de serviço e interações correspondentes a este grupo).

O SGsvc somente insere o novo membro nas estruturas de dados do grupo após obter, no final do processo, a confirmação do SGcoordenador, quando tem-se a garantia de que todos os SGlocal do grupo terão recebido a notificação de entrada do novo membro. Neste ponto, caso existam *canais* estabelecidos entre os membros do grupo, o SGsvc verifica se os papéis associados ao novo membro correspondem a algum destes canais, devendo, em caso afirmativo, ligar o objeto de aplicação do novo membro ao canal.

### Saída de Membros de um Grupo

O processo de saída de um membro de grupo é iniciado através da invocação da operação correspondente, disponível às aplicações na interface externa do objeto SGsvc. A Figura 5.10 mostra a seqüência de eventos envolvidos na saída de um membro do grupo, ressaltando a participação de cada um dos componentes de suporte a grupos no processo. O diagrama ilustra a invocação do serviço (1) pelo próprio membro a sair.

Antes que o processo de saída de membro seja efetivamente iniciado, o SGsvc valida os parâmetros da invocação do serviço, verificando se o grupo em questão realmente existe e se o membro a ser removido realmente participa do grupo (é também verificado se o usuário invocador está entre aqueles autorizados a determinar a execução do serviço). Em seguida, é aplicada a política de saída de membros, que neste caso determina a consulta aos membros do grupo para confirmar ou não a possibilidade de realização do serviço (2). O resultado das consultas ao grupo (3) é avaliado segundo um critério de maioria simples, sendo que critérios alternativos podem ser especificados como parte da política. Caso a saída do membro seja confirmada, o SGsvc invoca o SGcoordenador, através da operação de Distribuição de Eventos (4), passando como parâmetro o evento correspondente à saída do membro, para que seja distribuído, com informação de ordem, para os SGlocal do grupo (5), inclusive para aquele correspondente ao membro que está saindo do grupo. Em seguida, o SGcoordenador atualiza a lista de membros do grupo (a partir das informações sobre o membro que saiu do grupo, obtidas a partir das estruturas de dados do evento) e invoca o SGsvc para confirmar a distribuição do evento (6).

Cada SGlocal do grupo, ao receber o evento de saída de membro, deve processá-lo segundo a ordem de eventos do grupo, expressa na informação de ordem adicionada ao evento pelo SGcoordenador. O SGlocal tem duas alternativas para processar este evento: caso seja o SGlocal do membro que está saindo do grupo, é gerada uma notificação de *saída do presente membro* do grupo (7), sendo removidas as estruturas de dados, neste SGlocal, correspondentes ao grupo do qual ele

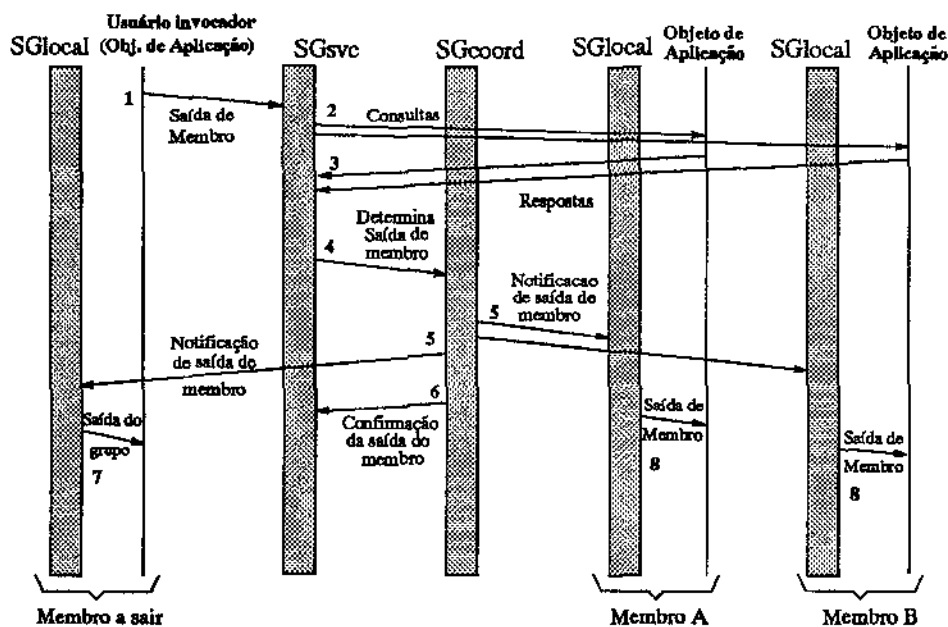


Figura 5.10: Saída de Membro - diagrama de tempo de execução do serviço

foi removido; em caso contrário, é gerada uma notificação de *saída de um membro* do grupo (8), sendo as respectivas informações removidas, pelo SGlocal, da lista de membros do grupo em questão (incluindo as mensagens pendentes relativas ao membro que saiu do grupo).

O SGsvc, ao receber a confirmação de distribuição do evento de saída de membro, atualiza as informações sobre o grupo, com a remoção do membro, sendo que, caso o respectivo usuário não seja membro de nenhum outro grupo, seu objeto SGlocal é finalizado (*deleted*) e sua entrada removida da lista de usuários do domínio. Além disso, caso o objeto de aplicação removido do grupo esteja ligado a algum *canal* (no contexto do grupo em questão), as ações apropriadas são realizadas junto ao componente de suporte a multimídia para atualizar a estrutura do canal.

### Mudança de Políticas, Papéis e Autorizações

A execução destes serviços é iniciada a partir da invocação das respectivas operações na interface externa do SGsvc, seguindo padrões semelhantes de interação entre os objetos envolvidos no grupo, conforme mostrado no diagrama da Figura 5.11.

Inicialmente, o SGsvc valida os parâmetros da invocação do serviço (1), verifica as autorizações e aplica a política do serviço correspondente (2,3). Em seguida, é invocada a operação de Distribuição de Evento na interface do SGcoordenador (4), para que a execução do serviço seja ordenada com respeito aos demais eventos de grupo. Após distribuir o evento para os membros (5), o SGcoordenador atualiza as estruturas de dados relativas ao grupo (o que vale apenas para o caso de alterações no conjunto de papéis de um membro ou mudança da política de distribuição ordenada de mensagens) e, em seguida, invoca a operação de Confirmação de Distribuição de Evento na

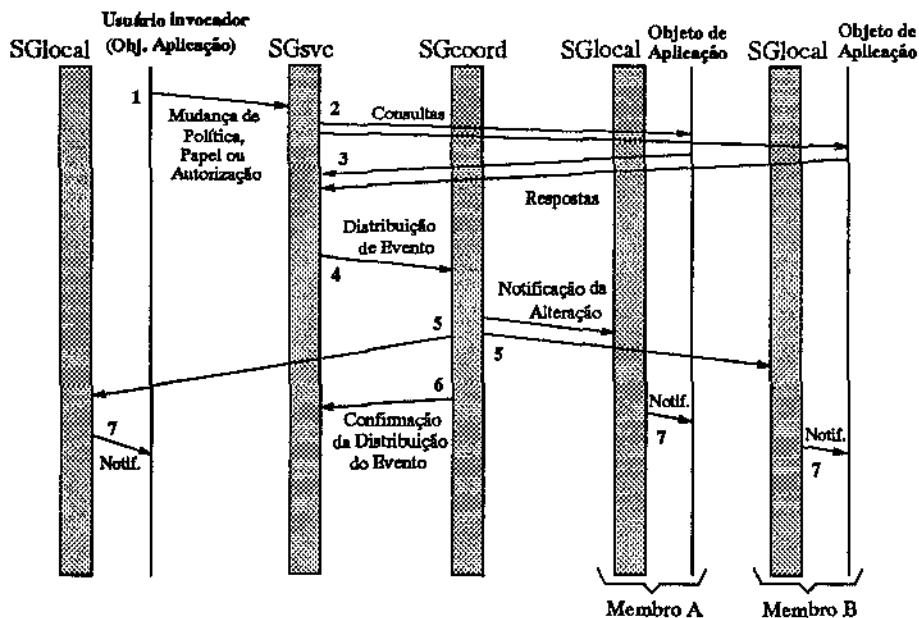


Figura 5.11: Mudança de Políticas, Papéis e Autorizações - diagrama temporal de execução dos serviços

interface interna do SGsvc (6).

O processamento do evento por parte de cada SGlocal do grupo é feito segundo a ordem de eventos do grupo e compreende a atualização das estruturas de dados referentes ao grupo (políticas de um serviço, papéis de um membro ou autorizações de um serviço, dependendo de qual dos três serviços foi invocado) e a geração de uma notificação para o objeto de aplicação correspondente (7). No SGsvc, após a recepção da confirmação do SGcoordenador, ocorre também a atualização das estruturas de dados do grupo. Além disso, no caso de mudança no conjunto de papéis de um membro, o SGsvc deve verificar se o papel adicionado ou removido está associado a algum canal, sendo que, em caso afirmativo, a estrutura do canal deve ser atualizada.

### Estabelecimento e Fechamento de Canal

Canais de comunicação multimídia podem ser criados para conectar os membros de um grupo. Um determinado canal, como visto no capítulo anterior, está associado a um *papel* específico, possibilitando a interação através de fluxos contínuos de informação entre os membros que desempenham este papel. No nível de suporte a grupos são tratados apenas os aspectos de estruturação de canais, no que diz respeito à manutenção do conjunto de membros ligados pelos canais. Os aspectos de comunicação através dos canais, bem como o controle das propriedades destas conexões é realizado pelo componente de processamento multimídia da arquitetura de suporte às aplicações.

A Figura 5.12 ilustra, no mesmo diagrama, os processos de estabelecimento e fechamento de canais, os quais envolvem padrões idênticos de interação entre os objetos envolvidos no grupo. Note

que não são tratados os aspectos de como canais são criados ou fechados, pois tais questões estão fora do escopo do presente trabalho.

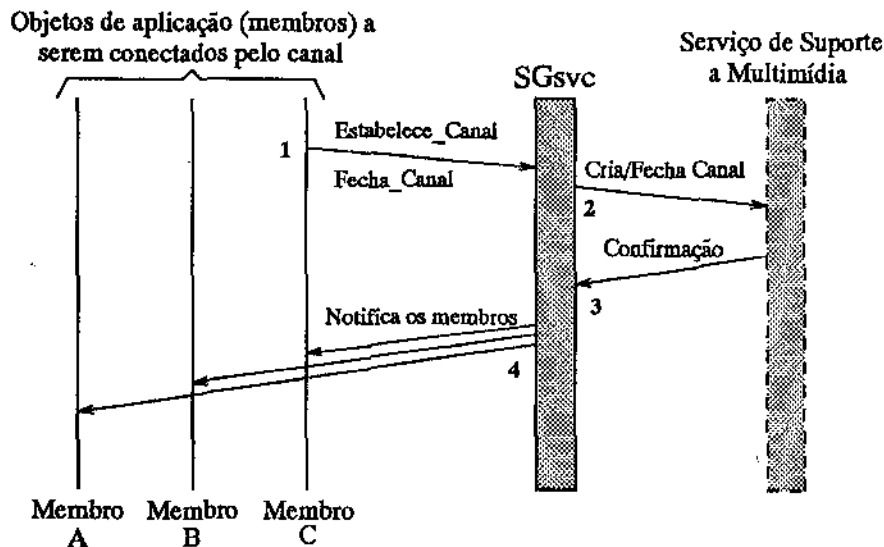


Figura 5.12: Estabelecimento e Fechamento de Canais - diagrama temporal de execução

Conforme se pode notar no diagrama, o estabelecimento e fechamento de canais não é ordenado entre os eventos de grupo (o processo não envolve o SGcoordenador).

No caso de estabelecimento de um novo canal, o SGsvc inicialmente verifica se os parâmetros, notadamente o papel a ser associado ao novo canal, estão corretamente especificados (o papel deve existir no contexto do grupo, isto é, deve haver membros associados ao papel). Em seguida, o SGsvc invoca o serviço de suporte a multimídia para a criação do canal (2), passando, como parâmetros, o conjunto de membros a serem conectados e as características do canal (recebidas sob a forma de parâmetros do usuário invocador). No caso de fechamento de um canal, o SGsvc invoca o serviço de suporte a multimídia (2), passando o identificador do canal a ser fechado.

Em ambos os casos, após receber resposta do serviço de suporte a multimídia (3), o SGsvc envia uma notificação (4) a cada um dos objetos de aplicação do grupo que estão envolvidos no canal, informando o identificador do canal (estabelecido ou fechado).

### Finalização de Grupos

Um grupo pode ser extinto através da invocação da operação de Finalização de Grupo na interface externa do SGsvc. O diagrama da Figura 5.13 mostra a seqüência temporal das interações entre os objetos envolvidos em um grupo para a execução deste serviço. Inicialmente, o SGsvc valida a invocação do serviço (1), verifica as autorizações e aplica a política de finalização de grupos (que, neste caso, determina a consulta aos membros do grupo) (2,3). Uma vez tendo sido confirmada a execução do serviço, o SGsvc invoca o SGcoordenador para a distribuição ordenada do evento

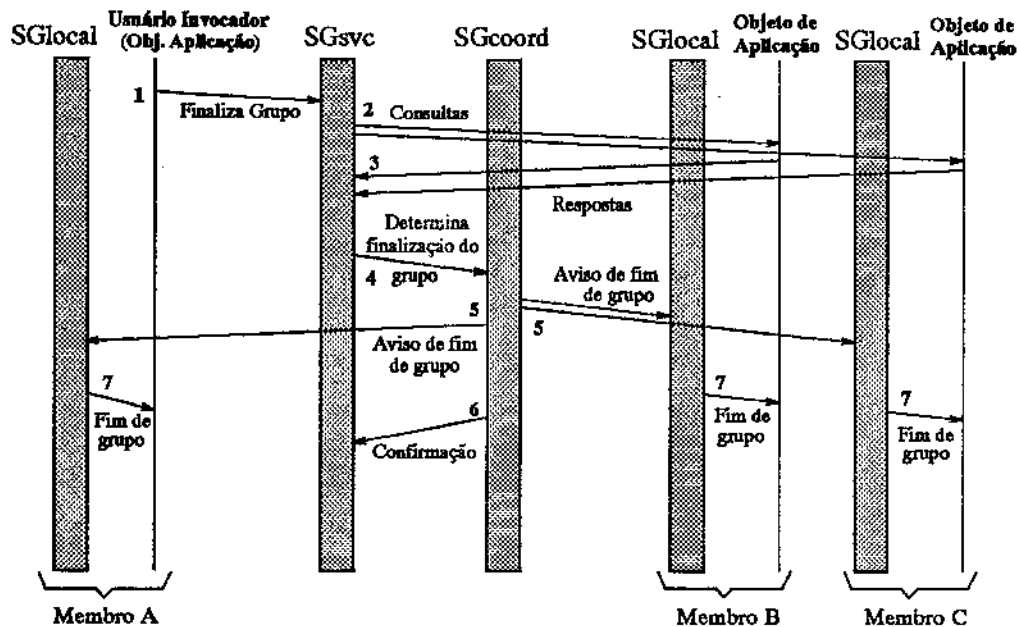


Figura 5.13: Finalização de Grupo - diagrama temporal de execução do serviço

correspondente (4).

Após ter distribuído o evento para todo o grupo (5), o SGcoordenador inibe a distribuição de outros eventos para este grupo (para assegurar que o evento de finalização seja de fato o último evento do grupo) e invoca a operação de Confirmação de Distribuição de Evento (6) na interface interna do SGsvc, confirmando a finalização efetiva do grupo. O SGsvc, por sua vez, remove as informações que mantinha sobre o referido grupo e desativa (“destrói”) o respectivo SGcoordenador, bem como os objetos SGlocal de usuários que não sejam mais membros de grupo algum em seu domínio.

Ao receber o aviso de finalização de grupo, e tendo sido processados todos os eventos anteriores a ele na ordem global, cada SGlocal inibe a distribuição de eventos para o grupo finalizado. Além disso, são removidas as pendências relativas ao grupo (notadamente, as filas de invocações com resposta pendente e as filas de respostas em processo de colagem), juntamente com as informações sobre o grupo que eram mantidas na lista de grupos. Em seguida, o SGlocal invoca o objeto de aplicação para notificar a finalização do grupo (7).

#### 5.4.4 Serviços de Distribuição de Mensagens

A comunicação entre os membros de um grupo é realizada por meio de operações da interface externa do SGlocal, que permitem aos objetos de aplicação membros do grupo enviarem mensagens para o grupo todo ou para parte dele através de uma única ação de comunicação. Os serviços de comunicação permitem a detecção e recuperação transparentes de falhas ocorridas no envio ou

recepção das mensagens, garantindo a eventual entrega das mensagens aos objetos de aplicação correspondentes aos membros do grupo (desde que estes não estejam falhos).

A invocação destes serviços é feita através de três operações presentes na interface dos objetos SGlocal (seção 5.3.3). Estas operações foram definidas segundo o modelo de interações entre objetos descrito no RM-ODP [ISO94b] (ver capítulo 3), de forma a permitir a distribuição de interações de *interrogação* (juntamente com as respostas associadas) e interações de *anúncio*.

A distribuição de mensagens pode ser feita segundo dois modelos: *com ordenação* ou *sem ordenação*. No primeiro caso, a distribuição é feita através da colaboração entre os objetos SGlocal e o objeto SGcoordenador do grupo, usando o mecanismo de *ordenação causal-total* de eventos. No segundo caso, a distribuição é realizada apenas com a participação dos objetos SGlocal. Estes dois modelos são descritos a seguir, juntamente com o mecanismo de coleta das múltiplas *respostas* geradas por *interrogações* de grupo. Observe que apenas mensagens correspondentes a *interrogações* ou *anúncios* podem ser distribuídas com ordenação (o que é especificado no campo “modo de ordenação” da estrutura de mensagem); mensagens de *resposta*, por outro lado, estão sujeitas apenas ao mecanismo de *colagem* de mensagens, sendo sempre distribuídas sem ordenação (a comunicação de respostas utiliza a ordem já estabelecida para as interrogações originais).

### Distribuição com Ordenação

Conforme mostrado na Figura 5.14, a distribuição de mensagens com ordenação é feita com a participação fundamental do SGcoordenador, que é o responsável por atribuir informação de ordem às mensagens e por sua efetiva distribuição.

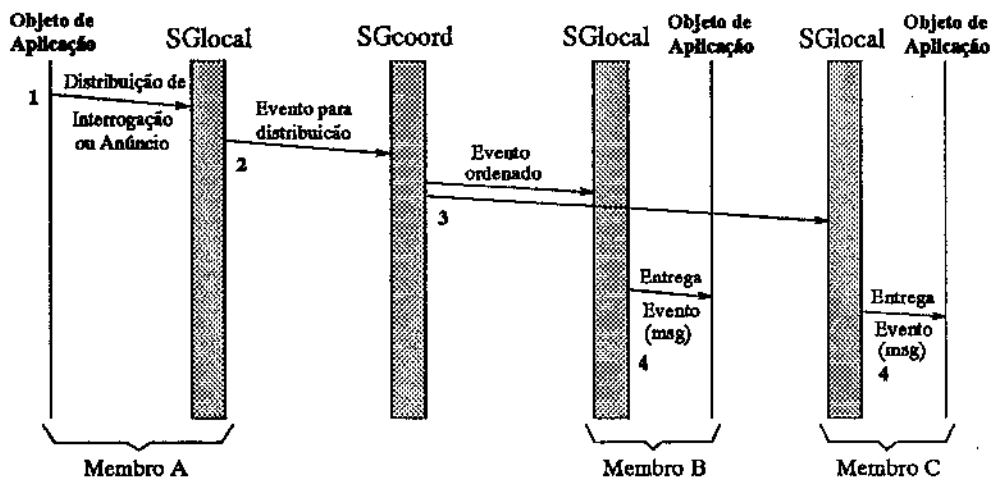


Figura 5.14: Distribuição de Mensagens Com Ordenação - diagrama temporal de execução

Ao receber a invocação de uma operação para distribuição de mensagem com ordenação (1), o SGlocal cria um *evento* correspondente à mensagem, enviando-o para o SGcoordinador (2). Caso esta mensagem corresponda a uma interrogação, o SGlocal cria também um registro que permitirá

fazer o tratamento de coleta das eventuais respostas (este tratamento é descrito a seguir).

Após receber este evento de comunicação de grupo, o SGcoordenador efetua a resolução e validação dos endereços de grupo que exprimem o destino da mensagem, mapeando o nome de grupo e os nomes de papéis para os nomes dos membros correspondentes. Em seguida, o evento é enviado para todos os SGlocal destino (3), juntamente com informação de ordem que o posiciona na seqüência de eventos de grupo recebidos por cada respectivo membro.

A recepção de mensagens com ordenação pelos objetos SGlocal deve ser feita segundo o mecanismo de ordenação de eventos descrito na seção 5.4.1, sendo que a mensagem pode apenas ser entregue à aplicação (4) quando sua informação de ordem indicar que ela corresponde ao próximo evento na seqüência ordenada de eventos recebidos do grupo. Novamente, caso a mensagem recebida seja uma interrogação, é criado um registro que permite ao SGlocal associar corretamente a eventual resposta (a ser retornada) a esta interrogação.

Observe que não existe confirmação explícita da distribuição de mensagens. Uma vez aceita para distribuição pelo SGlocal, os mecanismos de tratamento e recuperação de falhas (seção 5.4.2) do Suporte a Grupos garantem a eventual entrega da mensagem a todos os membros do grupo que foram endereçados. Desta forma, desde que o objeto de aplicação correspondente a um membro do grupo esteja operacional, tem-se a certeza de que este objeto receberá todas as mensagens para ele enviadas.

### Distribuição sem Ordenação

Mensagens não-críticas com relação à consistência do grupo (o que é determinado pela aplicação) podem ser distribuídas diretamente pelos objetos SGlocal, sem o intermédio do SGcoordenador e, portanto, sem ordenação no contexto do grupo. Este tipo de distribuição oferece um nível inferior de confiabilidade, em relação à distribuição com ordenação, mas permite a distribuição mais eficiente de mensagens.

A Figura 5.15 mostra a seqüência de eventos envolvidos no processo.

Ao receber um pedido de distribuição sem ordenação (1), o SGlocal cria um *evento* correspondente e o envia para os SGlocal destino (2) com informação de ordem local, que posiciona o evento na seqüência de eventos por ele gerados e distribuídos para o grupo. Esta informação de ordem local consiste de um número de seqüência mantido individualmente para cada outro SGlocal do grupo (seção 5.3.3) e permite aos SGlocal receptores da mensagem:

- determinarem a ordem em que as mensagens foram geradas pelo SGlocal de origem, estabelecendo uma ordem *FIFO* de entrega das mensagens ao objeto de aplicação; caso uma mensagem chegue com um número de seqüência maior que o esperado, sua entrega pode ser retardada até que as mensagens anteriores sejam recebidas;
- detectar (e descartar) mensagens repetidas, enviadas mais de uma vez por um SGlocal devido à suspeita de falhas de comunicação.

A entrega de mensagens sem ordenação à aplicação (3) é feita segundo a *política* de distribuição sem ordenação, que pode determinar ou não a entrega das mensagens em ordem *FIFO*. Em ambos



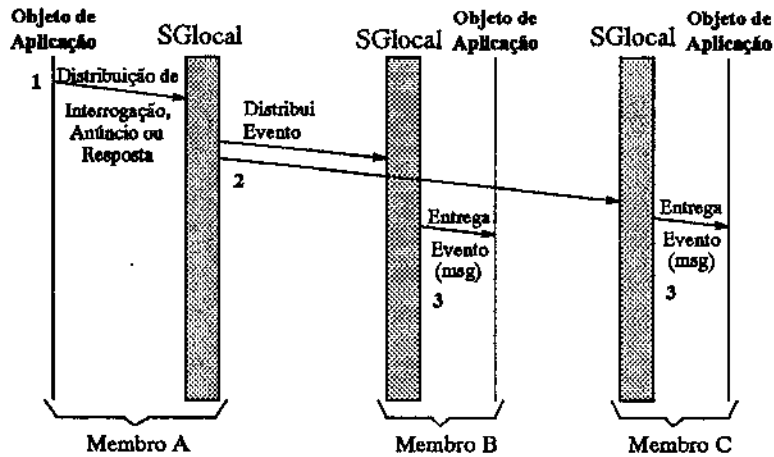


Figura 5.15: Distribuição de Mensagens Sem Ordenação - diagrama temporal de execução

os casos, o modelo de distribuição sem ordenação tende a ser mais eficiente que o modelo de distribuição com ordenação causal-total.

A propriedade de *atomicidade da comunicação de grupo* (que garante que todos os membros destino de uma mensagem de grupo eventualmente a receberão) não é garantida neste modelo de distribuição sem ordenação. A situação crítica ocorre quando um SGlocal falha antes de completar a distribuição de uma mensagem sem ordenação, quando apenas parte dos destinos da mensagem podem tê-la recebido. A atomicidade poderia ser garantida utilizando um protocolo de *broadcast confiável* como proposto em [JB89], onde cada SGlocal, após receber uma mensagem sem ordenação pela primeira vez, realizaria a sua retransmissão (*multicast*) para todos os membros relacionados na lista de destinos da mensagem, de forma que, mesmo na presença de falha do SGlocal original, as mensagens não seriam perdidas. Alternativamente, poderia-se utilizar um mecanismo de replicação passiva, semelhante àquele usado para o SGcoordenador, ou empregar técnicas para o armazenamento persistente (não-volátil) de mensagens pendentes. Entretanto, o uso destas soluções afetaria o desempenho da distribuição de mensagens sem ordenação, o que contraria o objetivo de eficiência (em detrimento de aspectos de confiabilidade) deste modelo de distribuição. Além disso, espera-se que apenas mensagens não-críticas para a aplicação sejam distribuídas segundo este modelo.

### Coleta de Respostas

A semântica de comunicação de grupo exige um tratamento adequado das múltiplas *respostas* geradas por uma *interrogação* enviada para os membros de um grupo. Com a finalidade de facilitar o tratamento destas múltiplas respostas pela aplicação, o Suporte a Grupos pode realizar um tratamento de *colagem*, colocando-as sob a forma de uma única resposta, equivalente em conteúdo [ISO94c, Isi93]. O mecanismo utilizado para combinar as respostas é definido na *política de colagem de respostas*, a qual, diferentemente das demais, é expressa individualmente para cada mensagem de interrogação a ser distribuída para um grupo (através do campo “*modo de colagem*” da estrutura

de mensagem). O conjunto de políticas disponíveis está definido no capítulo 4.

O mecanismo de colagem de respostas está intimamente relacionado com o serviço de distribuição de mensagens. Para cada mensagem do tipo *interrogação* a ser distribuída, é criado, no SGlocal de origem, um registro contendo um identificador local da interrogação e a política de colagem associada (ver Figura 5.6). Desta forma, ao receber as respostas, o SGlocal pode associá-las corretamente à interrogação original. As respostas correspondentes a uma mesma interrogação são mantidas em uma lista até que a condição de colagem (expressa na política) seja satisfeita. Considerações são feitas no sentido de não causar a espera indeterminada por respostas em caso de falha ou de saída de algum dos membros do grupo que deveriam enviá-las. O tratamento de falhas nestes casos envolve o cancelamento da espera por estas respostas, sendo que a resposta colada final é obtida a partir apenas das respostas recebidas com sucesso. Isto introduz um nível adicional de tolerância a falhas na comunicação de grupo, como proposto no próprio RM-ODP [ISO94c], onde uma interação de grupo correta pode ser obtida a partir de um conjunto de interações possivelmente afetado por falhas.

## Capítulo 6

# O Protótipo Implementado

Com o objetivo de validar os conceitos e o modelo apresentado, foi desenvolvido um protótipo com as funcionalidades básicas de suporte a grupos. A implementação segue o modelo de objetos e serviços descrito no capítulo 5, estando em conformidade com as interfaces, funcionalidades e estrutura interna dos objetos então especificados.

Uma variedade de tecnologias pode ser utilizada na construção de um protótipo baseado no modelo de implementação, incluindo RPC (*Remote Procedure Call*) [BN84] e comunicação por passagem de mensagens. Na presente implementação, entretanto, foi adotada a arquitetura CORBA (*Common Object Request Broker Architecture*) [OMG91] como plataforma de distribuição, o que permitiu uma implementação orientada a objetos distribuídos, fundamentada em abstrações muito semelhantes às aquelas usadas na descrição do modelo. A implementação baseada na CORBA possibilita o uso do protótipo em ambientes heterogêneos, permitindo o suporte a grupos de usuários distribuídos através de uma rede (uma *internet*) com recursos computacionais de arquiteturas distintas. Além disso, a definição em IDL (*Interface Definition Language*) das interfaces dos objetos torna uniforme a interação do protótipo com outros objetos distribuídos nesta rede, incluindo os clientes de suporte a grupos, independentemente da linguagem de implementação.

O protótipo integra o contexto maior da Plataforma Multiware [LMM<sup>+</sup>94, CM95], estando posicionado no nível de suporte às funcionalidades ODP [ISO94a] da Camada Middleware [MM94] da plataforma. As funcionalidades de suporte a grupos, neste contexto, deverão servir à Camada Groupware [LFSM94] na criação e manutenção de ambientes de trabalho cooperativo, podendo também ser utilizadas diretamente pelas aplicações. Entretanto, da forma como se encontra definida a implementação, procurou-se minimizar as dependências em relação a outros componentes da arquitetura Multiware (que também se encontram em fase de desenvolvimento), de forma que o uso do protótipo fora deste contexto é também possível.

O restante deste capítulo descreve os aspectos fundamentais da implementação. A seção 6.1 trata da abrangência do protótipo em termos dos objetos e serviços implementados. A seção 6.2 descreve o uso das facilidades da arquitetura CORBA na implementação dos conceitos do modelo,

tratando aspectos como a estruturação distribuída dos objetos de suporte a grupos e a comunicação entre eles. A seção 6.3, por sua vez, trata das questões de integração do protótipo no contexto da Plataforma Multiware. Finalmente, a seção 6.4 apresenta um exemplo de como os serviços implementados no protótipo podem ser utilizados no ambiente de uma aplicação cooperativa típica.

## 6.1 Abrangência do Protótipo

Dentre os objetos e funcionalidades especificadas no modelo de implementação (capítulo 5), foi selecionado um subconjunto básico para compor o protótipo. O objetivo foi construir um protótipo com recursos suficientes para demonstrar a aplicabilidade dos conceitos propostos em ambientes de trabalho cooperativo.

Entre os serviços de suporte a grupos especificados, foram implementados os seguintes:

- criação de grupos;
- entrada de membros;
- saída de membros;
- finalização de grupos, e
- distribuição de mensagens.

Os demais serviços têm implementação semelhante a estes e sua incorporação ao protótipo é considerada como uma futura extensão. No caso do serviço de estabelecimento e fechamento de canais, a implementação é dependente de outros componentes da arquitetura Multiware, dedicados ao suporte à comunicação multimídia, ainda em desenvolvimento.

No nível das funcionalidades de cada serviço, foram seguidas na íntegra as especificações do modelo de implementação, exceto o mecanismo de *autorizações de serviço*, que não está contemplado no protótipo. O tratamento de *políticas de serviço* é realizado para cada serviço, tendo sido implementadas as políticas especificadas no capítulo 4.

Com respeito às funcionalidades de suporte aos serviços, foram implementados os mecanismos de *ordenação causal-total* de eventos e *colagem* de respostas. A manutenção da consistência dos grupos é realizada com a execução dos serviços de estruturação de grupos sujeita ao mecanismo de ordenação de eventos. As interações entre os membros de um grupo são suportadas com o uso dos dois estilos de interação entre objetos (*interrogação* e *anúncio*), sendo realizada a distribuição transparente das interações para o grupo (o que pode ser feito com ou sem ordenação) e a colagem das respostas às interrogações (de acordo com a política de colagem especificada para cada mensagem de interrogação distribuída). Na presente versão do protótipo, não foram implementados mecanismos de tolerância a falhas com todas as características especificadas no modelo de implementação (mecanismos do tratador de falhas e mecanismo de replicação passiva de objetos). Apenas falhas na comunicação entre objetos são tratadas, sendo os demais aspectos considerados como extensões futuras ao protótipo.

Desta forma, o conjunto de objetos implementado se resume às classes *SGsvc*, *SGcoordenador* e *SGlocal*, cujas interfaces (interna e externa) e estruturas de dados são mantidas em conformidade com sua especificação no capítulo 5 (com respeito aos serviços implementados).

## 6.2 Aspectos da Implementação em CORBA

Esta seção aborda os aspectos de implementação do Serviço de Suporte a Grupos sobre a arquitetura CORBA. São descritas as alternativas adotadas para a implementação das funcionalidades de suporte a grupos, notadamente com relação às questões de estruturação distribuída dos objetos e da comunicação entre eles. Em especial, é mostrada a forma como estas questões foram tratadas no desenvolvimento do protótipo sobre o ambiente ORBeline [PMC94].

A configuração dos objetos de suporte a grupos no protótipo se manteve em conformidade com aquela definida no modelo de implementação. Cada objeto do modelo corresponde a uma *implementação de objeto CORBA*, sendo a interação entre esses objetos realizada através do *Object Request Broker* (ORB). A Figura 6.1 ilustra o papel do ORB como um “barramento de comunicação” entre os objetos envolvidos em grupos cooperativos, proporcionando transparência à comunicação entre objetos remotamente localizados, de forma a possibilitar as interações mostradas anteriormente na figura 5.5.

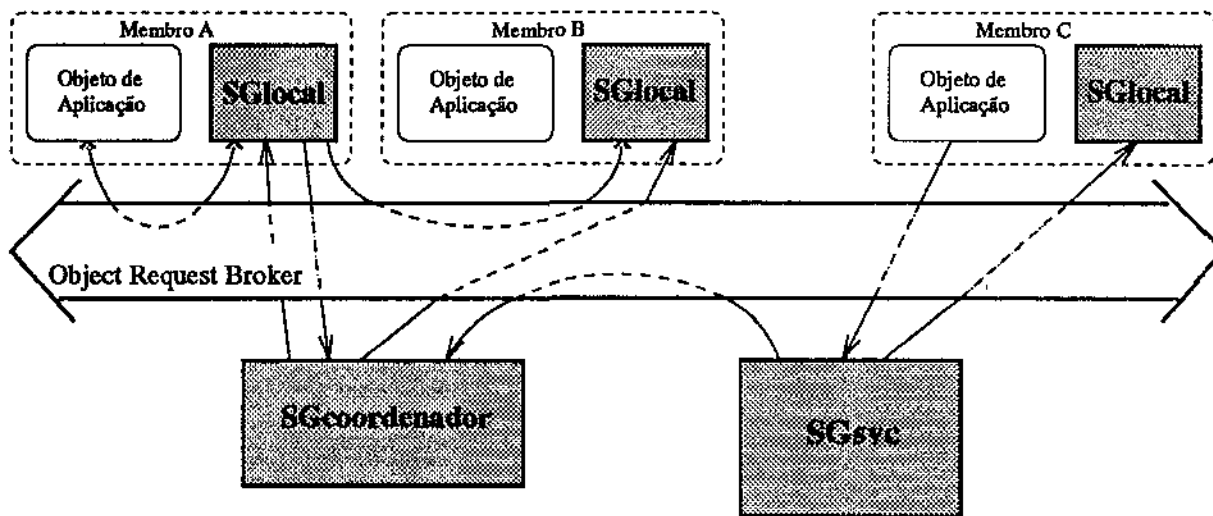


Figura 6.1: A comunicação entre os objetos através do ORB

Neste ambiente, cada objeto se comporta, dinamicamente, como cliente e servidor, dependendo das interações em que participa (conforme será visto adiante). A Figura 6.2 ilustra o posicionamento dos objetos sobre as interfaces de acesso às facilidades da CORBA. A figura representa, simbolicamente, as três classes de objetos de suporte a grupos, juntamente com os objetos de aplicação, nos papéis de clientes e servidores. Um objeto se comporta como cliente de outro em duas situações

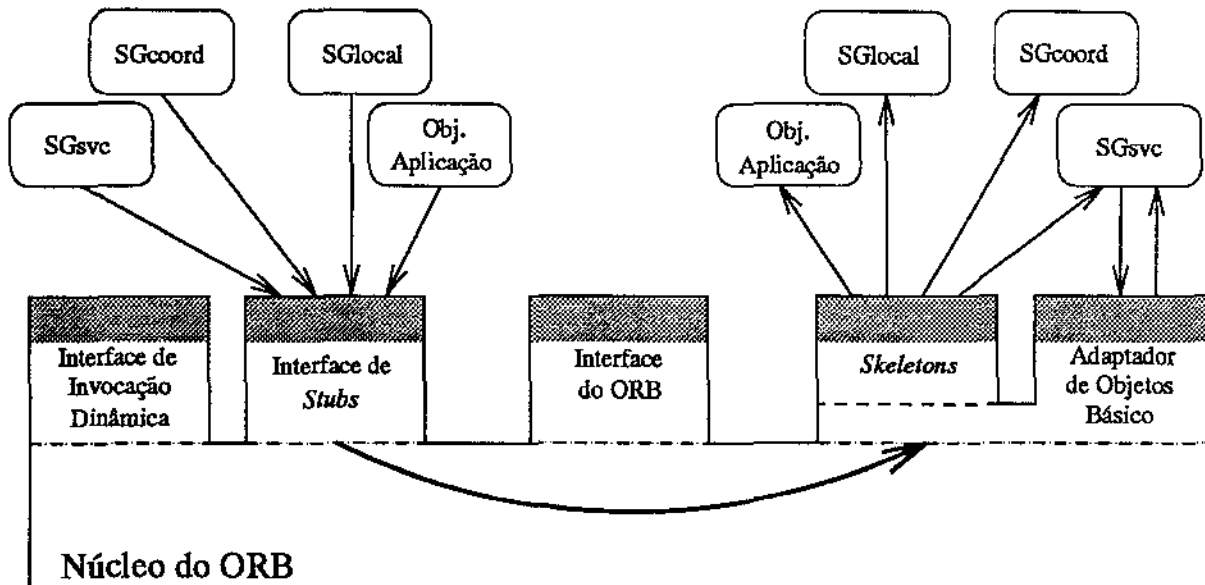


Figura 6.2: Os objetos de suporte a grupos sobre a arquitetura CORBA

distintas: na solicitação de serviços e no envio de informações para outros objetos. Em qualquer dos casos, o objeto alvo desempenha o papel de servidor na interação. Como exemplo, tem-se o caso da distribuição de mensagens com ordenação, quando o SGcoordenador (cliente, nesta interação) invoca os objetos SGlocal (que assumem o papel de servidores), para o envio das mensagens.

Todas as invocações de objetos são realizadas, do lado dos clientes, através da interface estática de *stubs*, os quais são gerados a partir da definição em IDL das interfaces dos objetos. A interface de *stubs* do ORB suporta apenas interações entre pares de objetos, o que significa que a comunicação de grupo (multiponto) deve ser simulada através de múltiplas invocações de objetos. Do lado dos servidores, a entrega das invocações é feita transparentemente pelos *skeletons IDL*. O uso dos mecanismos do ORB através destas interfaces torna transparentes, com relação aos aspectos de distribuição, as interações entre os objetos, tanto do ponto de vista dos clientes, quanto do ponto de vista dos servidores.

A interface do Adaptador de Objetos Básico (BOA - *Basic Object Adaptor*) é também utilizada pelos objetos de suporte a grupos para o acesso a serviços como a notificação, ao Adaptador de Objetos, da instanciação de um objeto, e o registro de implementações de objetos para ativação em tempo de execução.

### 6.2.1 Definição das Interfaces dos Objetos

Os tipos dos objetos de suporte a grupos são definidos através da especificação das interfaces dos objetos na Linguagem de Definição de Interfaces da CORBA (CORBA-IDL). A interface determina o comportamento de um objeto, conforme observado pelos clientes, definindo as operações

que podem ser invocadas, juntamente com os tipos das estruturas de dados que representam os parâmetros e valores de retorno das operações.

No modelo de implementação (capítulo 5), os objetos de suporte a grupos possuem duas interfaces diferentes, sendo uma para acesso às funcionalidades de grupos pelas aplicações, e a outra para uso interno entre os objetos de suporte a grupos. Entretanto, por definição, um objeto em CORBA possui apenas uma interface, a qual pode, em princípio, ser acessada por qualquer um de seus clientes. Desta forma, o uso de um artifício auxiliar é necessário para separar logicamente partes de uma interface com restrições de acesso diferentes. Podem ser usados mecanismos disponíveis sob a forma de Serviços de Objeto, para o *tratamento de eventos*, que permitem a execução de procedimentos de filtragem imediatamente antes da invocação de um objeto para determinar se uma invocação pode prosseguir ou não. As facilidades de manipulação de eventos (*event handling*) disponíveis no ORBeline [PMC94] são um exemplo disso. Por outro lado, pode ser utilizada uma alternativa “mais limpa”, baseada no mecanismo de *herança de interfaces*. Nesta alternativa, as diferentes interfaces (parciais) de serviços apresentadas por um objeto são definidas como interfaces distintas em IDL. Estas interfaces são exportadas para os respectivos clientes, de forma que eles possam utilizá-las para se conectarem ao objeto e invocarem seus serviços. A implementação do objeto, entretanto, é feita a partir de uma interface derivada das interfaces parciais, herdando todas as definições destas interfaces. Esta é a abordagem utilizada no protótipo. Por exemplo, para o objeto `SGlocal`, são definidas duas interfaces: a interface externa `SGlocal_ext` e a interface interna `SGlocal_int` (as quais não são implementadas diretamente); a interface a ser efetivamente implementada, `SGlocal_impl` é então derivada (por herança múltipla) das interfaces externa e interna. A Figura 6.3 ilustra o relacionamento entre as interfaces parciais e as interfaces de implementação dos objetos de suporte a grupos (notadamente os objetos `SGlocal` e `SGsvc`).

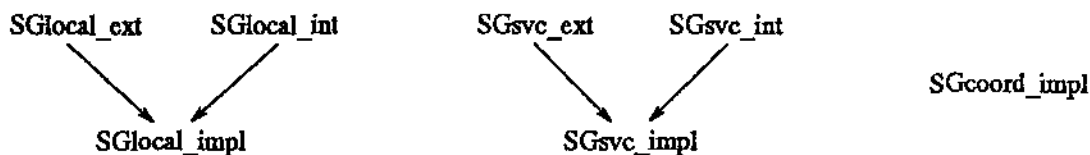


Figura 6.3: Hierarquia de interfaces em IDL

No apêndice A encontram-se definidas as interfaces dos objetos de suporte a grupos. São também especificadas as operações que devem ser suportadas na interface dos objetos de aplicação, de forma que os objetos de suporte a grupos (`SGlocal` e `SGsvc`) possam interagir com eles. A definição das interfaces está em conformidade com as especificações do capítulo 5, sendo complementada com definições das estruturas de dados compartilhadas pelos objetos.

Como exemplo, é mostrada abaixo uma parte extraída do módulo de definição IDL da interface externa dos objetos `SGlocal`.

```

module GroupCommf
// --- Estruturas de dados auxiliares:
// politicas de distribuicao (associadas dinamicamente):
  
```

```

enum DistributionMode {WHOLE_GROUP, SUBGROUP};
typedef sequence <string> DestinationList;

// políticas de colagem (associadas dinamicamente):
enum CollationMode {ALL_REPLIES, FIRST_REPLIES, NO_COLLATION};

typedef sequence <octet> MessageContents

// --- Estrutura de mensagem:
struct MessageStruct{
    string group_name;
    string message_sender;
    DistributionMode distr_mode;
    DestinationList destinations;
    CollationMode collation;
    boolean order_required; // politica de ordenacao
    MessageContents msg_contents;
};

interface SGlocal_ext{

    // --- Operacoes da Interface Externa:

    // distribuicao de mensagens de interrogacao
    unsigned long group_invoke (in MessageStruct message);

    // distribuicao de mensagens de anuncio
    boolean group_announce (in MessageStruct message);

    // distribuicao de mensagens de resposta
    boolean group_reply (in MessageStruct message,
                        in unsigned long interrogation_id);
}; // end interface
}; // end module

```

A definição da interface está contida no módulo de comunicação de grupos (GroupComm), o qual inclui também definições das estruturas de dados utilizadas nas operações da interface. A interface contém as três operações para invocar o serviço de distribuição de mensagens (conforme definidas no capítulo 5). Os parâmetros das operações são definidos como parâmetros de entrada (in).

### 6.2.2 Geração dos *Stubs* e *Skeletons*

A geração dos *stubs* para uso dos clientes, bem como dos *skeletons* usados na conversão das invocações (recebidas do ORB sob a forma de mensagens) para os objetos servidores, é realizada a partir da compilação das definições das interfaces, usando um compilador que gere código na linguagem de implementação dos clientes ou dos servidores. No caso particular do protótipo, foi



usado o compilador de IDL para C++ disponível no ORBeline, o qual obedece o mapeamento entre estas duas linguagens definido em [PMC94]. A especificação deste mapeamento é ligeiramente diferente daquela que está em processo de padronização pela OMG [OMG94b]. As diferenças consistem, basicamente, na forma em que são mapeados alguns tipos estruturados de IDL para C++. No ORBeline, todo tipo estruturado de IDL é mapeado para uma classe de C++, enquanto que no mapeamento da OMG, procura-se, sempre que possível, mapear o tipo IDL para um tipo correspondente em C++ (por exemplo, o tipo `struct` de IDL é mapeado, no ORBeline, para uma classe contendo métodos de acesso aos membros da estrutura, enquanto que, no mapeamento da OMG, o tipo `struct` de IDL é mapeado para o correspondente tipo `struct` de C++). Entretanto, considera-se que estas diferenças não afetam significativamente a portabilidade do protótipo.

O compilador IDL do ORBeline gera quatro arquivos, com código em C++, para cada arquivo de especificação de interface em IDL. As seções seguintes descrevem o uso das declarações e definições encontradas nestes arquivos para a implementação e instanciação dos objetos:

- Interface de *stubs*, contendo declarações das operações que podem ser invocadas sobre os objetos correspondentes a uma interface em IDL, juntamente com declarações de operações que permitem invocar serviços do ORB através do *stub*; neste arquivo são também colocadas as declarações das estruturas de dados definidas em IDL, com seus respectivos métodos de acesso, conforme o mapeamento para C++. Este arquivo deve ser incluído no código fonte dos clientes. Abaixo, são mostrados trechos relevantes deste arquivo, correspondendo à interface do `SGlobal` mostrada anteriormente em IDL:

```
#include <corba.h>
class GroupComm {
public:
    // --- Mapeamento das estruturas de dados usadas neste modulo:
    enum DistributionMode {WHOLE_GROUP, SUBGROUP};
    // ....
    class MessageStruct: public CORBA::Structure
    {
        // .... (mapeamento da estrutura de mensagens)
    };
    class SGlobal_ext_stub: public virtual CORBA::Object
    {
        // ....
    protected:
        // --- Construtor:
        SGlobal_ext_stub(const char *obj_name = NULL) :CORBA::Object(obj_name) {}
        // ....
    public:
        // --- Operacoes para acesso a servicos do ORB:
        // operacao para realizar o bind (conexao) dos clientes com os servidores:
        static SGlobal_ext_stub *_bind(CORBA::Environment &_env,
                                       const char *object_name = NULL,
                                       const char *host_name = NULL,
                                       const CORBA::BindOptions* opt = NULL);
    };
};
```

```

// ....
// --- Operacoes da interface dos objetos:
CORBA::ULong group_invoke(const GroupComm::MessageStruct& message,
                          CORBA::Environment& _env);
CORBA::Boolean group_announce(const GroupComm::MessageStruct& message,
                              CORBA::Environment& _env);
CORBA::Boolean group_reply(const GroupComm::MessageStruct& message,
                           CORBA::ULong interrogation_id,
                           CORBA::Environment& _env);
};
// tipo das referencias de objeto:
typedef SGlocal_ext_stub* SGlocal_ext_stub_Ref;
};

```

O módulo `GroupComm` definido em IDL é mapeado para uma classe C++ de mesmo nome, enquanto que a interface `SGlocal_ext` é mapeada para uma classe C++, `SGlocal_ext_stub`<sup>1</sup>, derivada da classe `Object` (a qual, por sua vez define as propriedades comuns a todo objeto em CORBA). Esta classe `SGlocal_ext_stub` representa a interface de *stubs* para os objetos `SGlocal`. A classe possui, além dos métodos correspondentes às operações da interface, métodos especiais para manipulação dos objetos; o exemplo mostra o método estático `_bind`, que é utilizado pelos clientes para obterem referências de objeto para os objetos que implementam a interface. Uma referência de objeto corresponde a um ponteiro para um objeto C++ do tipo especificado pela classe que define os *stubs*, e é usada para realizar invocações na interface da implementação de objeto (transparentemente, através do ORB). Cada método desta classe contém um parâmetro adicional do tipo `Environment`, que pode ser usado pelos clientes para verificar a ocorrência de *exceções* no decorrer das invocações aos objetos servidores.

- Implementação dos *stubs*, que contém código para a serialização (*marshalling*) das invocações de objetos, permitindo colocá-las em formato apropriado para serem conduzidas, pelo ORB, através da rede. Para cada operação na interface do objeto, existe uma operação de conversão correspondente no *stub*. Este arquivo deve ser compilado e ligado com o código objeto dos clientes.
- Interface de *skeletons*, contendo declarações correspondentes às operações da interface dos objetos, juntamente com as operações para obter informações sobre a implementação de objeto e para a conversão (desserialização) das mensagens que chegam do ORB para a forma de invocações aos objetos. Para cada interface em IDL é gerada uma classe correspondente aos *skeletons*, da qual deve ser derivada a classe das implementações de objeto (servidores). Este arquivo deve ser incluído no código fonte dos servidores. Abaixo são mostradas as definições relevantes neste arquivo, conforme gerado pelo compilador IDL a partir da interface (externa) do `SGlocal`:

```
#include <SGlocal_c.hh>
```

<sup>1</sup>O sufixo “\_stub” foi adicionado ao nome da classe gerada apenas para melhorar a clareza do exemplo (o compilador IDL normalmente gera classes de *stubs* com nome idêntico ao da interface em IDL correspondente)

```

#include <corba.h>

class GroupComm_impl {
public:
    class SGlocal_ext_skeleton: public virtual GroupComm::SGlocal_ext_stub
    {
    protected:
        SGlocal_ext_skeleton(const char *object_name = NULL);
        virtual ~SGlocal_ext_skeleton();
    public:
        // --- Operacoes para acessar informacoes sobre a implementa\c{c}\~ao do objeto
        // ....

        // --- Operacoes a serem implementadas pelo servidor:
        virtual CORBA::ULong group_invoke(const GroupComm::MessageStruct& message);
        virtual CORBA::Boolean group_announce(const GroupComm::MessageStruct& message);
        virtual CORBA::Boolean group_reply(const GroupComm::MessageStruct& message,
                                           CORBA::ULong interrogation_id);

        // --- Operacoes para a desserializacao (unmarshalling) das invocacoes
        // ....
    };
};

```

Como no caso da interface de *stubs*, o módulo `GroupComm` e a interface `SGlocal_ext` são mapeados para classes correspondentes em C++ (mas com nomes diferentes dos originais). A classe `SGlocal_ext_skeleton` é derivada da classe que representa a interface de *stubs* (note que o arquivo que contém a definição da interface de *stubs*, `SGlocal_c.hh`, é incluído no presente arquivo). Todas as operações da interface em IDL são mapeadas para métodos virtuais da classe `SGlocal_ext_skeleton`, indicando que a implementação será realizada em uma classe derivada desta (a classe de implementação do objeto).

- Implementação dos *skeletons*, contendo código para a conversão das mensagens que chegam via ORB em invocações das operações apropriadas na interface dos objetos. Para cada operação definida na interface de um objeto, existe uma operação de conversão correspondente, a qual, após ser invocada pelo ORB (para a entrega de uma invocação ao objeto), realiza os procedimentos de desserialização da invocação e faz uma chamada à operação real na interface do objeto. Este arquivo deve ser compilado e ligado ao código objeto dos servidores.

A figura 6.4 ilustra a relação entre os arquivos gerados pelo compilador IDL, a partir da especificação da interface, e os arquivos contendo declarações e código de clientes e servidores. Observa-se que os clientes não necessariamente precisam ser objetos no nível da CORBA (ou seja, com interfaces definidas em IDL). Como se pode notar pela figura, a geração dos clientes e servidores depende da ligação com bibliotecas do próprio ORBeline. Isto reflete a abordagem distribuída adotada no ORBeline para implementação do ORB através de rotinas residentes no espaço de endereçamento

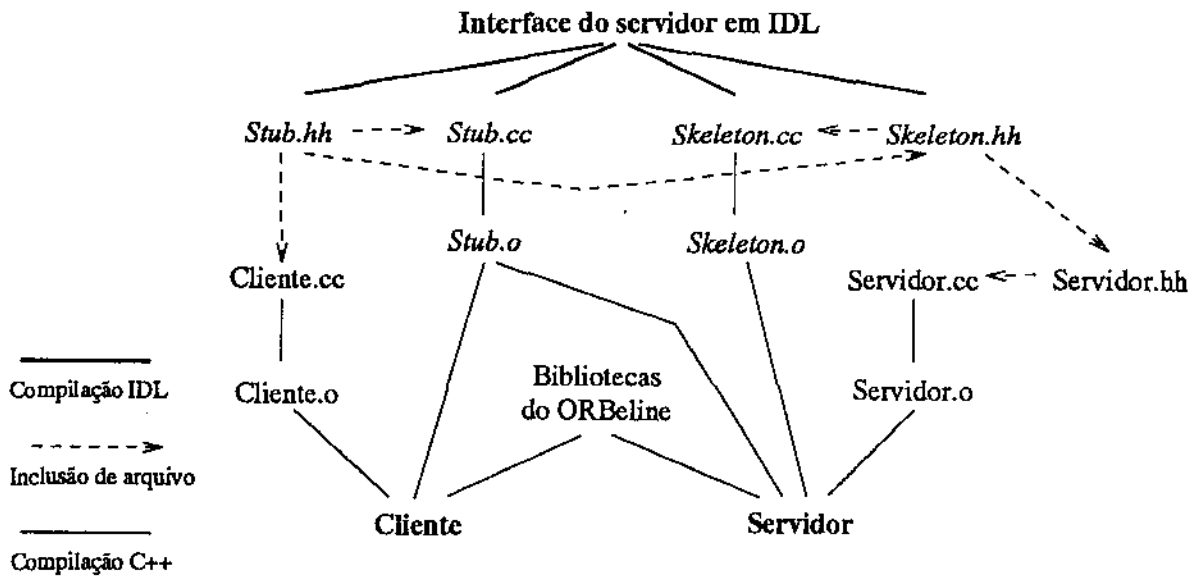


Figura 6.4: Arquivos para construção dos clientes e servidores no ORBeline

dos servidores e clientes. A manutenção dos nomes e referências de objetos únicos é implementada através de um servidor de nomes distribuído, permitindo a transparência na localização dos objetos.

### 6.2.3 Implementação dos Objetos

Os objetos de suporte a grupos foram implementados na linguagem C++. Cada tipo de objeto é definido através de uma *classe* particular, derivada, pelo mecanismo de *herança* de C++, a partir da classe onde é definido o tipo da interface de *skeletons* correspondente (gerada a partir da definição em IDL da interface de implementação dos objetos). A Figura 6.5 mostra a hierarquia de derivação das classes dos objetos de suporte a grupos. Como mostrado na figura, as classes de implementação dos objetos *SGlocal* (*SGlocal\_impl*) e *SGsvc* (*SGsvc\_impl*) são obtidas, por herança múltipla (indiretamente), a partir das classes que representam os *skeletons* e *stubs* correspondentes às interfaces parciais (externa e interna) dos objetos. Observe que as classes intermediárias da hierarquia são geradas automaticamente pelo compilador IDL a partir das interfaces em IDL dos objetos (ver figura 6.3).

A declaração de cada uma das classes de implementação, *SGlocal\_impl*, *SGsvc\_impl* e *SGcoord\_impl*, é constituída, basicamente de três partes principais:

- uma parte privada, contendo declarações dos tipos e objetos usados na implementação do servidor, mas que não estão envolvidos na comunicação através do ORB;
- outra parte privada, contendo declarações das operações (métodos) usados internamente à implementação;

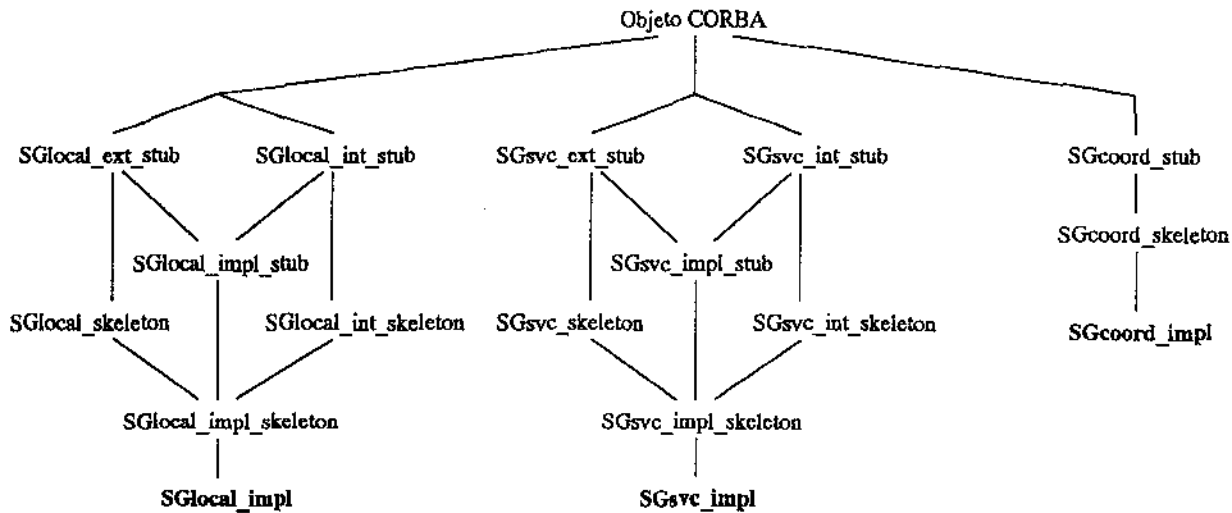


Figura 6.5: Hierarquia de herança na definição das classes dos objetos de suporte a grupos

- uma parte pública, contendo declarações das operações da interface dos objetos servidores, as quais são acessíveis para invocação pelos clientes; estas operações são declaradas como virtuais (sem implementação) na classe que define os *skeletons* e precisam ser redeclaradas e devidamente implementadas nesta classe que define a implementação dos objetos servidores.

Observe que as estruturas de dados definidas como parte da interface em IDL são herdadas pela classe de implementação através das classes que representam os *stubs* (pois a classe de implementação é derivada indiretamente destas classes). Operações para acesso aos serviços do ORB são também herdadas desta forma.

A implementação destas classes contém código para realizar cada uma das operações definidas nas interfaces em IDL, o que efetivamente define o comportamento dos objetos de suporte a grupos. Este comportamento, como visto anteriormente, pode envolver a comunicação com outros objetos através do ORB, onde um objeto atua como cliente e o outro como servidor. Completando a seqüência de exemplos, a seguir é mostrada a declaração da classe de implementação do SGlocal, com os trechos de código relevantes.

```

class SGlocal_impl: public SGlocal_impl_skeleton{
private:
    // --- Tipos e estruturas de dados usados internamente
    // ....

private:
    // --- Declaracao das operacoes de uso interno
    // ....

public:
    // --- Construtor e destrutor da classe
  
```

```

SGlocal_impl(const char* object_name = NULL):SGlocal_impl_skeleton(object_name){
    // .... (inicializacao das estruturas do objeto)
}

~SGlocal_impl(){}

public:
    // --- Declaracao das operacoes da interface externa do objeto
    //      (especificadas na interface em IDL do objeto)
    CORBA::ULong group_invoke(const GroupComm::MessageStruct& message);

    CORBA::Boolean group_announce(const GroupComm::MessageStruct& message);

    CORBA::Boolean group_reply(const GroupComm::MessageStruct& message,
                               CORBA::ULong inv_id);

    // --- Declaracao das operacoes da interface interna do objeto
    // ....
};

```

A classe de implementação dos objetos `SGlocal`, `SGlocal_impl`, é derivada da classe que define os *skeletons* gerados pelo compilador IDL a partir da interface genérica (a qual, por sua vez, é derivada a partir das interfaces interna e externa, conforme o artifício descrito na seção 6.2.1 para prover múltiplas interfaces para um mesmo objeto). Observe que, na seqüência de exemplos é considerada apenas a interface externa do `SGlocal`.

O construtor da classe de implementação (`SGlocal_impl`) pode receber, como parâmetro, o nome do objeto a ser criado. Para cada um dos métodos declarados como virtuais nas interfaces de *skeletons* (correspondentes às interfaces interna e externa em IDL), é declarado um método correspondente, com os mesmos parâmetros e tipo de retorno, cuja implementação é realizada no escopo desta classe.

#### 6.2.4 Ativação e Desativação de Objetos

Cada objeto de suporte a grupos é contido no espaço de endereçamento de um processo particular (um servidor, no sentido de CORBA), de forma a tornar os objetos independentes para comunicação e processamento.

O ORBeline permite a criação de objetos de dois tipos: objetos *persistentes* (também suportados na especificação do Adaptador de Objetos Básico [OMG91]) e objetos *transientes*. O primeiro tipo refere-se a objetos cujo tempo de vida pode ser maior que o do processo servidor no qual foram criados.<sup>2</sup> Por outro lado, objetos transientes deixam de existir no momento em que o processo correspondente é desativado. Os objetos de suporte a grupos são criados como objetos persistentes

<sup>2</sup>A manutenção de armazenamento persistente para as estruturas de dados de objetos persistentes está fora do escopo do ORB e do BOA, devendo ser realizada pela aplicação, ou através de um *serviço de objeto* específico para a manutenção de objetos persistentes [OMG95b].

(embora não tenha sido implementado nenhum mecanismo de persistência).<sup>3</sup>.

Duas formas distintas são disponíveis no ORBeline para a criação de objetos: estática e dinâmica. A criação *estática* ocorre a partir da instalação do processo servidor (por meios externos ao ORB), o qual realiza a instanciação da classe que especifica a implementação do objeto e, em seguida, notifica a criação do objeto ao BOA. A segunda forma de criação de objetos, aqui referida como *dinâmica*, consiste em registrar a implementação do objeto junto ao BOA, o qual, por sua vez adiciona a definição da implementação do objeto ao Repositório de Implementações. A criação efetiva do objeto, neste caso, ocorre apenas quando o objeto for invocado pela primeira vez, sendo que o BOA, através de um mecanismo particular de ativação de objetos, se encarrega de criar um processo servidor correspondente e, dentro dele, instanciar o objeto. O ORBeline oferece uma interface de programação para o BOA que permite que implementações de objetos sejam registradas em tempo de execução, o que possibilita a criação “dinâmica” dos objetos de suporte a grupos no momento em que os ambientes de trabalho cooperativo são estabelecidos.

Uma outra questão importante diz respeito à criação de objetos remotamente, uma vez que a CORBA não prevê esta facilidade no ORB. Desta forma, é necessário algum mecanismo complementar, possivelmente implementado usando os recursos do ORB, para permitir a um objeto determinar a criação de outro objeto em um *host* diferente. No protótipo de suporte a grupos existe um objeto auxiliar, denominado de *fábrica*, o qual é instalado (estaticamente) em cada *host* da rede onde possa existir um membro de grupo. Este objeto é responsável por atender a pedidos do SGsvc para a criação de objetos SGcoordenador ou SGlocal. O uso deste objeto auxiliar poderia ser dispensado caso houvesse disponível um *serviço de objeto de Ciclo de Vida* [OMG95b], que implementaria as funções de criação e desativação de objetos, entre outras.

A desativação de objetos ocorre explicitamente através da invocação de uma operação apropriada na interface do BOA. Além desta forma de desativação de objetos, o ORBeline permite que um objeto seja desativado através da destruição do objeto em C++ correspondente no processo servidor.

A seguir é descrito o uso destas alternativas de criação e destruição de objetos para o caso específico de cada um dos objetos de suporte a grupos.

### Objeto SGsvc

Uma vez que existe apenas um objeto SGsvc em um domínio de grupos, ele é criado, *estaticamente*, através de um servidor persistente [OMG91], que realiza a instanciação de um objeto da classe correspondente à implementação do SGsvc. O processo servidor deve ser criado em um *host* confiável e de fácil acesso, uma vez que a execução dos serviços de suporte a grupos pode depender dele. Embora um servidor persistente possa abrigar vários objetos, no caso do SGsvc é criado apenas um objeto. O trecho de código abaixo foi extraído da função *main* do programa correspondente ao processo servidor e é explicado em seguida.

```
main(){
    class SGsvc *SGsvc_object;
```

---

<sup>3</sup>Objetos persistentes foram utilizados em detrimento de objetos transientes, em virtude de estes últimos não poderem ser localizados pelo servidor de nomes do ORBeline (*osagent*)

```
SGsvc_object = new SGsvc("SGserver");  
  
CORBA::BOA::impl_is_ready();  
  
return(1);  
}
```

A instanciação do objeto é feita pelo código da segunda linha, onde é chamado o construtor da classe de implementação, passando como argumento o *nome do objeto* a ser instanciado (“SGserver” neste caso). Em seguida, o BOA é notificado, através da operação estática `impl_is_ready`, de que a implementação de objeto foi instanciada e está pronta para receber invocações. A invocação de `impl_is_ready` tem o efeito de passar o controle do processo servidor para o BOA (e deste para o ORB), de forma que invocações da interface do objeto (que chegam sob a forma de mensagens da rede) possam ser corretamente recebidas e entregues ao objeto.

### Objetos SGcoordenador e SGlocal

A criação destes objetos é *dinâmica* e realizada pelo objeto *fábrica* à medida que solicitado pelo SGsvc. O objeto *fábrica* registra as implementações de objeto correspondentes junto ao BOA (através da operação `create`), de forma que os objetos sejam efetivamente criados no instante em que precisarem ser utilizados pela primeira vez.

Um objeto SGcoordenador é registrado com o BOA no momento da criação do grupo correspondente, sendo que sua criação efetiva ocorre quando o SGsvc tenta se conectar a ele para a transferência de informações de inicialização. O SGcoordenador normalmente é (registrado e) criado no mesmo *host* onde se localiza o membro com o papel de coordenador do grupo.

Um objeto SGlocal, por outro lado, é registrado junto ao BOA no momento em que o usuário (objeto de aplicação) correspondente se torna, pela primeira vez, membro de algum grupo. O objeto SGlocal é criado no mesmo *host* onde se localiza o respectivo usuário.

A desativação de um SGcoordenador ocorre ao término do processo de finalização do grupo correspondente, enquanto que a desativação de um SGlocal é feita quando o usuário correspondente é removido de todos os grupos dos quais era membro. Em qualquer dos casos, a desativação é realizada pelo objeto *fábrica*, que invoca o BOA (através da operação `dispose`) para remover o registro daquele objeto.

#### 6.2.5 Identificação dos Objetos

Segundo a arquitetura CORBA, objetos são identificados através de *referências de objeto* únicas e não-ambíguas, as quais são atribuídas pelo Adaptador de Objetos em cooperação com o ORB, e devem ser usadas para identificar os objetos nas interações. Além desse tipo de identificação, o ORBeline permite também que *nomes* sejam associados aos objetos no momento em que estes são criados. Nomes de objetos não são usados para interação, mas podem ser utilizados pelos clientes,



juntamente com os nomes das interfaces correspondentes, para a obtenção de referências de objeto para objetos específicos.

Estas duas formas de identificação de objetos são usadas para tratar os objetos de suporte a grupos. O objeto *SGsvc* possui um nome que pode ser divulgado para todos os usuários de um domínio, de forma que as aplicações e outros objetos de suporte a grupos possam utilizá-lo para obter a referência de objeto do *SGsvc* e, conseqüentemente, para interagirem com o objeto através dela. O nome de um objeto *SGcoordenador* equivale ao nome do grupo correspondente e pode ser usado pelo *SGsvc* e *SGlocal* para obterem sua referência de objeto. O mesmo ocorre com um objeto *SGlocal*, que recebe o mesmo nome do usuário correspondente. Por outro lado, um objeto de aplicação (que interage diretamente com os objetos de suporte a grupos) deve também ser identificado (pela aplicação) através de um nome, formado a partir da concatenação do nome do respectivo usuário com o nome do grupo em questão.

O ORBeline provê um *serviço de diretório distribuído*, denominado de *osagent* (*ORBeline Smart Agent*), o qual é o responsável por mapear as referências de objetos usadas nas interações para as respectivas instâncias (localizações) dos objetos, bem como por mapear os nomes de objetos para as respectivas referências de objetos (durante o estabelecimento de conexões). O Smart Agent mantém ainda informações dinâmicas sobre os objetos, permitindo a recuperação de falhas de conexão, bem como o chaveamento dinâmico das interações para as réplicas ativas de objetos falhos.

No caso de identificação de grupos, o protótipo de suporte a grupos implementa um serviço de nomes diferenciado, uma vez que o serviço de diretório do ORBeline não provê diretamente facilidades para a resolução de nomes (endereços) de grupo. Além disso, o *osagent* não provê uma API (*Application Programmer Interface*) para uso explícito de suas funcionalidades. O serviço de nomes de grupo, portanto, mantém tabelas (replicadas) que permitem converter nomes de grupos e nomes de papéis nos correspondentes nomes de membros e referências de objetos.

### 6.2.6 Conexão entre os Objetos

Antes que um objeto da aplicação possa utilizar os serviços de suporte a grupos, é necessário que uma conexão seja estabelecida entre ele e seu respectivo *SGlocal*, bem como entre ele e o *SGsvc*. Conexões são também necessárias para a comunicação entre os próprios objetos de suporte a grupos.

A posse de uma *referência de objeto* por um cliente permite a comunicação entre ele e o objeto correspondente à referência, sendo que a conexão entre os objetos é efetivada, transparentemente, através do ORB. Em ORBeline, uma referência de objeto pode ser obtida de duas formas:

- através de uma ação explícita de estabelecimento de conexão (*binding*), onde o cliente invoca a operação estática `_bind` definida na interface do *stub* correspondente ao objeto servidor (ver definição desta interface acima), a qual retorna a referência de objeto desejada; ao invocar esta operação, o cliente pode identificar o particular objeto desejado através de parâmetros como o nome do objeto e o nome do *host* em que o objeto está localizado;
- através da comunicação de referências de interface entre objetos, onde o objeto (cliente) que recebe a referência passa a ter acesso ao objeto (servidor) por ela referenciado.

No caso dos objetos de aplicação, referências de objetos para o SGsvc e para os SGlocal devem ser sempre obtidas através do primeiro método. Por outro lado, no caso dos objetos de suporte a grupos, são usados os dois métodos, dependendo da situação. O SGsvc obtém referências de objeto, tanto para os objetos SGcoordenador e SGlocal, quanto para os objetos de aplicação, de forma explícita, através da operação de `_bind`. Uma vez obtidas estas referências, elas são comunicadas aos demais objetos (SGcoordenador e SGlocal), de acordo com a segunda abordagem.

### 6.2.7 Comunicação entre os Objetos

O mapeamento de IDL para C++ usado no ORBeline define que referências de objeto são tratadas como ponteiros para objetos em C++. Desta forma, uma vez de posse de uma referência de objeto, um cliente pode utilizá-la como se fosse um ponteiro normal para invocar, transparentemente, operações na interface do objeto servidor. A comunicação de mensagens entre objetos é realizada através de invocações do objeto fonte para o objeto destino, sendo que as mensagens são encapsuladas como parâmetros das invocações.

Este mecanismo de invocações permite apenas a comunicação entre pares de objetos, o que significa que a comunicação de grupo deve ser realizada através de *múltiplas invocações*. Desta forma, no processo de distribuição de mensagens com ordenação, por exemplo, o SGcoordenador deve efetuar uma invocação particular para cada SGlocal cujo usuário membro deva receber a mensagem. A unidade das múltiplas invocações de objeto que correspondem a uma mesma interação de grupo é mantida através de um identificador de interação comum, associado a cada uma das invocações (mais precisamente, às mensagens distribuídas através das invocações).

A natureza da comunicação por meio de mensagens pode ser modelada apropriadamente através de um tipo especial de invocações provido pela CORBA, denominado em IDL de *one-way*, que não envolve o retorno de resultados das invocações (a comunicação ocorre em apenas um sentido, do objeto fonte para o objeto destino das invocações). Desta forma, os objetos não precisam ficar bloqueados à espera de respostas após as invocações, o que possibilita um maior grau de paralelismo entre eles. Um outro efeito do uso deste tipo de invocações refere-se à eliminação de possíveis condições de *deadlock* geradas por objetos mutuamente dependentes que poderiam ficar bloqueados como resultado da realização de invocações.

Na interface interna dos objetos SGcoordenador e SGlocal, todas as operações usadas para a comunicação de eventos são declaradas, em IDL, como sendo deste tipo “não-bloqueante” (*one-way*).

Após cada invocação de objeto, a confirmação de sucesso ou falha da comunicação é obtida através do mecanismo de *exceções* de CORBA, o que dispensa o uso de reconhecimentos com esta finalidade no nível da comunicação entre objetos e viabiliza o uso seguro de invocações “não-bloqueantes” do tipo mencionado acima. Para que isto seja possível, cada invocação aceita um parâmetro adicional (passado por referência), o qual é usado pelo ORB para reportar condições de erro ou exceção ocorridas durante a realização da invocação. Desta forma, caso seja sinalizada uma condição de terminação anormal de uma invocação, o cliente pode verificar o tipo de exceção ocorrida e, dependendo deste tipo, tentar realizar novamente a invocação ou iniciar um processo

de tratamento de falha.

### 6.3 Integração na Plataforma Multiware

O serviço de Suporte a Grupos integra a arquitetura da camada Middleware da Plataforma Multiware [LMM<sup>+</sup>94], constituindo um dos módulos da subcamada de Funções ODP [ISO94c], conforme ilustrado na Figura 6.6.

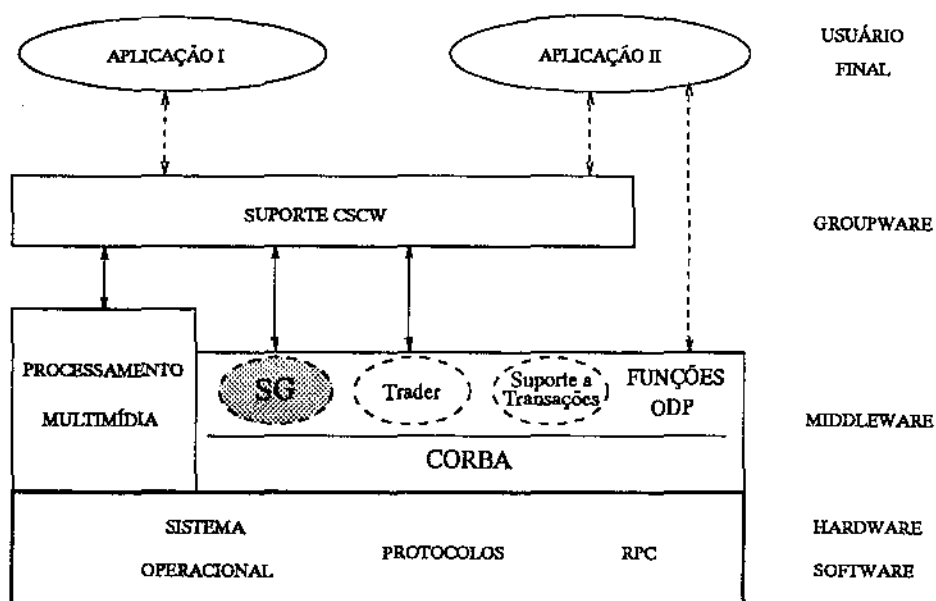


Figura 6.6: Posicionamento do Suporte a Grupos na Plataforma Multiware

Nesta arquitetura, as funcionalidades de suporte a grupos podem ser implementadas através do uso direto de outros componentes da plataforma. Para comunicação entre objetos, são usados os recursos de plataformas comercialmente disponíveis, que oferecem transparência de distribuição e outros serviços básicos. No estágio atual do projeto Multiware, CORBA tem sido a alternativa adotada para este nível, sendo utilizados duas implementações independentes, ORBeline [PMC94] e Orbix [ION95].

No nível das funcionalidades ODP, a implementação do Suporte a Grupos pode interagir com outros módulos, como o *Trader* [LM95]. O Suporte a Grupos (mais especificamente, o objeto SGsvc) pode, por exemplo, se registrar junto ao *Trader*, de modo que as identificações (nomes) dos objetos provedores dos serviços de suporte a grupos possam ser obtidas pelos usuários através das funcionalidades de localização de serviços do *Trader*.

No caso da comunicação multimídia entre os membros de um grupo (através de *canais*, como visto no capítulo 4), o componente de Processamento Multimídia da plataforma é o responsável pelo provimento de canais de comunicação entre os membros, bem como pelo controle das qualidades de

serviço destes canais. Em tais situações, o Suporte a Grupos deve interagir com este componente de Processamento Multimídia para controlar o conjunto dos membros ligados pelos canais.

Observe, entretanto, que o protótipo atual foi implementado sem o uso destas funcionalidades adicionais da plataforma. Futuras extensões, tais como para implementar o serviço de Estabelecimento e Fechamento de Canais, deverão considerar a interação com outros serviços da plataforma.

### 6.3.1 Usuários do Suporte a Grupos

Conforme visto na Figura 6.6, as aplicações podem utilizar as funcionalidades da camada Middleware (mais precisamente as funções de suporte a ODP) diretamente ou através da camada Groupware [LFMS94]. No caso de aplicações cooperativas, a situação mais freqüente deve ser a interação através do componente de Suporte CSCW da camada Groupware, uma vez que este provê às aplicações as facilidades e mecanismos utilizados para a interação cooperativa entre os usuários.

Desta forma, considera-se como principal usuário do Suporte a Grupos a Camada Groupware da plataforma. Esta camada provê objetos que representam os usuários das aplicações cooperativas (referidos como *objetos de aplicação* no capítulo 5), os quais interagem com os objetos de suporte a grupos para obter as funcionalidades de estruturação e comunicação de grupos em favor das aplicações. Entretanto, aplicações, através de seus componentes distribuídos, podem também interagir diretamente com o Suporte a Grupos. Em qualquer dos casos, os objetos clientes do Suporte a Grupos devem prover uma interface em conformidade com aquela especificada no capítulo 5, de forma a possibilitar a comunicação entre estes objetos clientes e os objetos servidores de suporte a grupos.

## 6.4 Exemplo de Aplicação do Suporte a Grupos

Nesta seção é descrito um cenário simplificado, que mostra o uso das funcionalidades de suporte a grupos por uma aplicação cooperativa típica. A aplicação considerada consiste de uma ferramenta para edição colaborativa de documentos, os quais podem ser consultados e atualizados concorrentemente pelos vários usuários membros de um grupo cooperativo.

Um ambiente de edição colaborativa de documentos normalmente consiste de múltiplos editores distribuídos em uma rede, sendo que cada editor é utilizado por um particular usuário para acessar o documento editado cooperativamente. O documento em edição é replicado em cada editor e alterações no mesmo podem ser efetuadas por vários usuários simultaneamente, através das operações normais de edição, como, por exemplo, a inserção ou remoção de caracteres, palavras ou linhas de texto do documento. As alterações efetuadas por um usuário devem ser propagadas (por seu respectivo editor) de maneira consistente, através da rede, para os demais editores, de forma que todos os usuários do grupo observem sempre cópias idênticas do documento. Interações de controle entre os editores são necessárias para a coordenação das operações de edição, por exemplo, para definir qual usuário tem acesso de escrita a determinada parte do documento.

As funções de edição de documentos e controle do ambiente cooperativo são realizadas pelos editores distribuídos, os quais utilizam recursos da infraestrutura de comunicação para realizar

estas funções. Dois exemplos de ferramentas de edição cooperativa com características semelhantes a estas são DistEdit [KP90], que utiliza o suporte de comunicação “multiponto” confiável e com ordenação total fornecido pelo sistema ISIS [BC90], e GROVE [EGR91], que implementa um seqüenciador de operações para propagar as alterações na mesma ordem para todos os usuários. Na plataforma Multiware, as funcionalidades intrínsecas de edição colaborativa de documentos são implementadas na camada Groupware, a qual utiliza os serviços de Suporte a Grupos para a comunicação entre os editores cooperantes, bem como para a manutenção consistente do conjunto de usuários de um grupo cooperativo. A seguir, são mostrados os passos principais envolvidos no estabelecimento e operação de um ambiente para edição colaborativa, utilizando os serviços de Suporte a Grupos propostos neste trabalho.

O processo de estabelecimento do ambiente ou *sessão* de edição colaborativa pode ser iniciado por um objeto da camada Groupware que é responsável por atender a pedidos de estabelecimento de sessão dos usuários. Alternativamente, a própria aplicação de edição pode iniciar o processo a partir de uma solicitação de seu usuário. Inicialmente os objetos da camada Groupware (ou da própria aplicação distribuída) podem realizar uma fase de negociação entre os usuários para definir as características da sessão a ser criada. Em seguida, é invocada a operação de *criação de grupos* da interface do objeto SGsvc, com a especificação da lista dos usuários que formarão o grupo cooperativo correspondente à nova sessão. Estes usuários são representados, no nível de suporte a grupos, pelos respectivos objetos da camada Groupware (ou da própria aplicação cooperativa) que atuam em favor dos usuários. Como exemplo, um usuário pode ser representado pelo editor de documentos por ele utilizado. Juntamente com a lista de usuários, são especificados os papéis dos usuários, bem como as políticas e autorizações dos serviços. Caso não sejam especificadas, políticas e autorizações *default* (políticas livres e autorizações sem restrição) são utilizadas.

A entrada e saída de novos usuários no grupo de edição colaborativa é realizada através das operações correspondentes na interface do SGsvc. Como exemplo, o usuário a entrar no grupo pode, de alguma forma, determinar ao seu objeto de aplicação a entrada na sessão cooperativa. Este objeto de aplicação, por sua vez, é o responsável por invocar o serviço do SGsvc. A efetivação do serviço está sujeita aos mecanismos de políticas e autorizações, bem como à ordenação dos eventos de grupo. Desta forma, o usuário somente pode se considerar dentro (ou fora) do grupo após seu objeto de aplicação receber a notificação de execução do serviço (ver a interface dos objetos de aplicação na seção 5.3). Uma notificação de realização do serviço é distribuída com ordenação para todos os membros do grupo em questão.

Um aspecto importante no caso da entrada de usuários em um grupo diz respeito à *transferência do estado* do grupo para os novos membros. No caso da aplicação de edição colaborativa, o novo usuário deve ser provido com uma cópia do documento sendo co-editado. Esta cópia deve ser consistente com as cópias mantidas pelos demais membros, devendo ser transferida para o novo membro durante o processo de entrada no grupo, enquanto a distribuição de mensagens com ordenação (que podem alterar o documento) está suspensa. A cópia do documento é obtida junto a um dos objetos de aplicação do grupo (normalmente, por aquele correspondente ao membro coordenador do grupo). O protocolo de transferência de estado foi visto no capítulo 5 como parte do serviço de entrada de membros.

As alterações efetuadas no documento por cada membro do grupo cooperativo são propagadas para os demais membros do grupo através do serviço de distribuição de mensagens com ordenação. Desta forma, todas as alterações no documento são observadas por todos os membros do grupo na mesma ordem, o que mantém consistentes as diversas cópias mantidas no grupo. A operação de distribuição com ordenação (na interface dos objetos *SGlocal*) é invocada pelos objetos de aplicação a cada nova alteração a ser propagada para o grupo. Cabe ao objeto de aplicação definir a granularidade das alterações propagadas (caracteres, palavras, etc.).

Por outro lado, o serviço de distribuição de mensagens sem ordenação pode ser utilizado pelas aplicações cooperativas para implementar a comunicação informal [SR92] entre os membros de um grupo.

O fechamento da seção de edição colaborativa é normalmente concluído com a finalização do grupo correspondente através do serviço de finalização de grupos na interface do *SGsvc*. Este serviço é invocado por um dos objetos de aplicação (correspondente a um membro que possua a devida autorização). Observa-se, entretanto, que uma sessão de trabalho cooperativo (no nível das aplicações) pode se estender por tempo indeterminado, podendo haver períodos de inatividade, durante os quais o trabalho cooperativo é suspenso. Nestas situações, a aplicação pode armazenar, em um banco de dados, o estado corrente do ambiente cooperativo (em termos do conjunto atual de membros e dos recursos compartilhados), podendo, em seguida finalizar o grupo (no nível de suporte a grupos). Posteriormente, ao ser reiniciada a sessão de trabalho cooperativo, a aplicação pode criar um novo grupo (usando o Serviço de Suporte a Grupos) com a configuração de membros anteriormente existente. Após estruturado o grupo, os serviços de comunicação de grupos podem ser utilizados para propagar as demais informações de estado da aplicação. Uma outra alternativa para lidar com esta situação seria permitir que os membros de uma sessão de trabalho cooperativo possam se retirar do grupo correspondente à medida em que deixam a sessão. O grupo então poderia ser finalizado (explicitamente) quando não houvesse mais membros no mesmo. Nesta abordagem, o reinício das atividades em uma sessão cooperativa se daria também dinamicamente, através da entrada de sucessivos membros no correspondente grupo. Neste trabalho, deixou-se para as aplicações a tarefa de definir a maneira mais apropriada de tratar estes aspectos de interrupção e reinício de sessões.

## Capítulo 7

# Conclusão

No desenvolvimento de aplicações cooperativas, é fundamental a existência de serviços de suporte que ofereçam facilidades básicas e amplamente utilizadas por esta classe de aplicações. O objetivo do presente trabalho consistiu em prover um conjunto de serviços de suporte, no nível de sistemas distribuídos, para simplificar a construção de aplicações cooperativas. Foram desenvolvidos serviços para a estruturação de aplicações sob a forma de grupos de objetos cooperativos, bem como serviços para a comunicação de grupo entre estes objetos. A abstração de grupos de objetos é considerada apropriada para modelar aplicações cooperativas, uma vez que torna possível associar, diretamente, as entidades do nível das aplicações (os usuários cooperantes), bem como as interações entre elas, a entidades e interações no nível de sistemas distribuídos.

Os serviços de suporte a grupos cooperativos foram definidos com base na Função de Grupos do Modelo de Referência para Processamento Distribuído Aberto da ISO. A Função de Grupos foi adotada em virtude de esta prescrever um conjunto de funcionalidades necessárias para a coordenação de configurações de objetos cooperantes. Entretanto, a descrição da Função de Grupos provida no RM-ODP é superficial, apresentando, apenas em linhas gerais, os serviços por ela previstos. Desta forma, o presente trabalho propôs, com base em conceitos bem definidos na literatura sobre grupos, uma interpretação ou refinamento para os diversos aspectos encontrados na Função de Grupos. Para cada funcionalidade especificada no RM-ODP para a Função de Grupos, é apresentado um conjunto de alternativas para realização das mesmas.

A partir deste refinamento da Função de Grupos e da seleção de funcionalidades apropriadas nele presentes, foi especificado um modelo de suporte a grupos, que propõe serviços básicos para o desenvolvimento de aplicações de trabalho cooperativo. O modelo de suporte a grupos foi especificado com base em conceitos definidos no *ponto de vista computacional* do RM-ODP. Esta abordagem coincide com a abordagem geral adotada no RM-ODP, que determina o desenvolvimento de modelos (ou, em situações de consenso, padrões) específicos para as diversas áreas de aplicação de processamento distribuído aberto.

Uma característica importante do modelo de suporte a grupos proposto diz respeito à possibilidade de configuração das funcionalidades de suporte a grupos, o que permite considerável flexibilidade na adequação dos serviços a ambientes cooperativos com requisitos específicos. Esta

flexibilidade é provida, principalmente, a partir do mecanismo de *políticas*, tendo sido definido um conjunto básico de políticas associadas a cada particular serviço de suporte a grupos. Com relação à comunicação de grupo, foi definido um mecanismo de interações multiponto, com ordenação de mensagens, o que permite garantir, no nível de comunicação, a consistência dos recursos distribuídos manipulados através dos serviços de suporte a grupos. A transparência da comunicação de grupo é reforçada através do mecanismo de *colagem* de respostas que, juntamente com o serviço de distribuição de mensagens, permite às aplicações tratarem a comunicação de grupo de maneira semelhante à comunicação entre pares de entidades. Por sua vez, o conceito de *papéis* de membros, aqui introduzido em uma forma genérica, permite a definição de subgrupos virtuais dentro de grupos existentes, podendo ser usados para fins de endereçamento transparente, bem como para a atribuição de autorizações aos membros.

A implementação do suporte a grupos é proposta segundo um modelo de objetos distribuídos, entre os quais são particionadas (ou replicadas) as funcionalidades e informações de suporte. A replicação das informações sobre grupos nos objetos SGlocal aumenta a disponibilidade de acesso, enquanto que a centralização das funções de ordenação de mensagens no objeto SGcoordenador permite a implementação de algoritmos simples e eficientes para comunicação multiponto confiável. A comunicação entre os objetos é suportada pela arquitetura CORBA, o que permite a utilização dos serviços de suporte a grupos em ambientes de recursos computacionais heterogêneos. A implementação sobre uma plataforma ORB representou um ganho com relação aos aspectos de distribuição dos objetos de suporte a grupos, uma vez que as transparências providas pelo ORB efetivamente ocultaram os problemas de comunicação através da rede. As interações entre os objetos são tratadas, transparentemente, como invocações de métodos em suas interfaces. Entretanto, esta transparência não é completa, uma vez que precisam ser consideradas as possíveis falhas que podem ocorrer no transcurso de uma invocação como, por exemplo, falhas de comunicação. A detecção destas condições de falha é normalmente realizada no nível da implementação dos objetos, através do mecanismo de exceções definido na CORBA.

Por outro lado, uma deficiência da arquitetura CORBA em contextos de suporte a comunicação de grupo refere-se à inexistência de mecanismos para a comunicação multiponto entre objetos, através de invocações múltiplas, por exemplo. Neste sentido, o Serviço de Suporte a Grupos pode ser considerado como uma adição às funcionalidades da CORBA, podendo ser classificado como um *Serviço de Objeto* implementado sobre o ORB.

Com o objetivo de demonstrar os conceitos do modelo e prover um ambiente de suporte a grupos com recursos básicos, foi desenvolvido um protótipo que oferece os serviços de criação e finalização de grupos, entrada e saída de membros, e distribuição (ordenada e não-ordenada) de mensagens. O protótipo foi desenvolvido sobre a plataforma ORBeline, com as interfaces dos objetos definidas em CORBA-IDL, enquanto que suas funcionalidades são implementadas na linguagem C++. As interfaces providas pelo ORBeline para acesso aos serviços do ORB são consistentes com aquelas propostas pelo OMG, o que facilitará a portabilidade do protótipo para outras implementações da CORBA. A versão do ORBeline utilizada (1.0) mostrou-se eficiente na comunicação das invocações de objetos, apresentando um nível de confiabilidade e tolerância a falhas aceitável.

Os serviços oferecidos pelo protótipo, embora direcionados especificamente para o suporte a



aplicações de trabalho cooperativo na plataforma Multiware, podem ser utilizados, em princípio, na construção de outros tipos de aplicações que envolvam interação cooperativa entre objetos.

Alguns aspectos não abordados no presente trabalho são considerados como possíveis extensões, que podem melhorar tanto o modelo de suporte a grupos quanto sua implementação.

Com relação ao modelo de suporte a grupos, uma extensão seria permitir a comunicação entre grupos diferentes, com manutenção das propriedades de atomicidade e ordenação das mensagens. O modelo poderia também ser estendido para suportar o estilo de interações do tipo “muitos-para-um” para qualquer tipo de mensagem (não apenas para mensagens de respostas, como na versão atual). A definição de um mecanismo de ordenação causal, não baseado em coordenador centralizado, pode vir a ser adicionada ao modelo, constituindo uma terceira alternativa para a distribuição de mensagens. Esta adição tornaria o serviço de suporte a grupos menos suscetível a problemas de escalabilidade, permitindo a formação de grupos de tamanho consideravelmente grande. Além disso, um conjunto mais abrangente de políticas pode vir a ser definido para os serviços de suporte a grupos.

Quanto ao protótipo, uma extensão imediata seria a implementação dos serviços de mudança de políticas, papéis e autorizações, definidos no capítulo 4. Considera-se que a incorporação desses serviços ao protótipo seja uma tarefa relativamente simples, uma vez que os mecanismos básicos (como ordenação de eventos) já foram implementados. Por outro lado, a implementação do serviço de manutenção de canais depende da disponibilidade da camada de suporte a multimídia da Plataforma Multiware. Uma outra adição interessante ao protótipo será a implementação dos mecanismos de tolerância a falhas de objetos descritos no capítulo 5, com destaque para a replicação dos objetos SGsvc e SGcoordenador.

A integração efetiva no contexto da Plataforma Multiware deverá ser realizada à medida em que os demais componentes da plataforma estiverem disponíveis. Esta tarefa deverá ser consideravelmente facilitada em virtude do uso uniforme dos recursos da CORBA na definição de todos os componentes da Plataforma Multiware.

O protótipo foi testado através de uma interface que permite simular o uso dos serviços de suporte a grupos pelas aplicações cooperativas. Neste ambiente simulado, foi possível verificar questões como a confiabilidade e o desempenho do protótipo. Entretanto, considera-se a necessidade de testar o protótipo em um ambiente real, através do desenvolvimento de aplicações cooperativas que utilizem os serviços oferecidos. Desta forma, pode-se verificar seu comportamento, juntamente com a adequabilidade dos conceitos do modelo de suporte a grupos, na presença dos requisitos reais das aplicações. Com isto, poderá ser ressaltada a necessidade de extensões ou alterações que não foram observadas no ambiente simulado. Também seria interessante a realização de testes de escalabilidade do protótipo, para determinar o tamanho máximo de grupos suportado para cada classe de aplicações.

## Apêndice A

# Definição das Interfaces dos Objetos

Este apêndice apresenta as definições, em CORBA-IDL, das interfaces dos objetos de suporte a grupos, segundo o protótipo implementado. Estas interfaces constituem um subconjunto daquelas definidas no capítulo 5, de acordo com os serviços suportados. As definições compreendem as operações disponíveis nas interfaces de cada objeto, juntamente com as estruturas de dados que constituem os parâmetros destas operações. É também apresentada a definição da interface que deve ser provida pelos objetos de aplicação.

Conforme visto no capítulo 5, os objetos *SGsvc* e *SGlocal* possuem duas interfaces, uma externa (para comunicação com os objetos de aplicação – para acesso aos serviços de suporte a grupos) e outra interna (para comunicação entre os objetos de suporte a grupos). A definição de múltiplas interfaces é possível em CORBA-IDL através do mecanismo de herança de interfaces, sendo primeiro definidas as várias interfaces que o objeto deverá apresentar (as quais não são diretamente implementadas) e, em seguida, uma interface derivada daquelas, a qual corresponderá à implementação do objeto. Esta interface de implementação, no caso dos objetos *SGsvc* e *SGlocal*, não contém definições adicionais àquelas existentes nas interfaces parciais (as interfaces interna e externa dos objetos). É necessário observar que este artifício é restrito, por não permitir que as interfaces parciais tenham atributos ou operações em comum.

As seções seguintes apresentam as definições das interfaces de cada objeto em particular. As estruturas de dados que correspondem aos parâmetros das operações disponíveis nas interfaces dos objetos são também descritas em CORBA-IDL. Estas definições são apresentadas na próxima seção (no protótipo, estas definições são colocadas em um arquivo próprio, o qual é incluído nos arquivos que contêm as definições de cada interface).

### A.1 Estruturas de Dados dos Parâmetros

Esta seção define as estruturas de dados utilizadas na formação dos parâmetros das invocações de objetos. Estas definições são contidas no arquivo *defs.idl*, que é incluído (via diretiva do pré-processador IDL) nos demais arquivos que definem as interfaces dos objetos.

```
// Lista de papeis de membros
typedef sequence <string> RoleList;

// Representacao externa de um membro de um grupo
struct Member{
    string member_name;
    RoleList member_roles;
};

// Lista de membros (segundo a representacao externa)
typedef sequence <Member> MemberList;

// Representacao interna (para uso entre os objetos
// de suporte a grupos) de um membro de um grupo
struct MemberInt{
    string member_name;
    RoleList member_roles;
    SGlocal obj_SGlocal;
};

// Representacao interna da lista de membros de um grupo
typedef sequence <MemberInt> MemberListInt;

// Politicas de servicos
enum Policies { ... };

// Lista de politicas
typedef sequence <Policies> PolicyList;

// Modos (politicas dinamicas) de distribuicao de mensagens:
enum DistributionMode {WHOLE_GROUP, BY_ROLE, BY_MEMBER};

// Lista de destinos de uma mensagem
typedef sequence<string> DestinationList;

// Modos (politicas dinamicas) de colagem de respostas
enum CollationMode {ALL_REPLIES, FIRST_REPLIES, NO_COLLATION};

// Estrutura de mensagens de grupo
struct MessageStruct{
    string group_name;
    string message_sender;
    DistributionMode distr_mode;
    DestinationList destinations;
    CollationMode collation;
    boolean order_required;
```

```

    string message_contents;
};

// Lista de mensagens (para armazenar respostas coladas)
typedef sequence <MessageStruct> MessageList;

// Tipos de eventos de grupo
enum EventType {MSG_INTERROGATION, MSG_REPLY, MSG_ANNOUNCE,
                JOIN, LEAVE, FINISH_GROUP};

// Uniao discriminada que define a estrutura
// contida no corpo de um evento
union EventContents switch(EventType){
    case MSG_INTERROGATION:
    case MSG_REPLY:
    case MSG_ANNOUNCE:
        MessageStruct message;
    case JOIN:
    case LEAVE:
        MemberInt member;
};

// Estrutura de eventos de grupo
struct Event{
    // Cabecalho do Evento:
    string group_name;
    string member_name;
    EventType type;
    unsigned long local_seq_number;
    unsigned long global_seq_number;

    // Corpo do Evento:
    EventContents event_contents;
};

// Informacoes para inicializacao de novos membros
// (representadas como uma sequencia de octetos)
typedef sequence <octet> InitData;

```

## A.2 Interfaces do SGsvc

As duas interfaces parciais do SGsvc, a interface externa e a interface interna, são aqui apresentadas, juntamente com a interface “de implementação”, derivada das interfaces externa e interna.

### A.2.1 Interface Externa do SGsvc

```
#include <defs.idl>
module GroupStruct{
  interface SGsvc_ext{

    // --- Criacao de novo grupo
    //
    oneway void create_group (in string creator_name,
                              in string group_name,
                              in MemberList members,
                              in PolicyList policies);

    // --- Entrada de membro em um grupo
    //
    oneway void join_group (in string invoker_name,
                            in string group_name,
                            in Member new_member);

    // --- Saida de membro de um grupo
    //
    oneway void leave_group (in string invoker_name,
                             in string group_name,
                             in Member member);

    // --- Finalizacao de um grupo
    //
    oneway void finish_group (in string invoker_name,
                              in string group_name);
  };
};
```

### A.2.2 Interface Interna do SGsvc

Esta interface contém uma operação utilizada pelo SGcoordenador para informar ao SGsvc a efetivação da realização de um serviço que corresponde a uma alteração na estrutura de um grupo.

```
#include <defs.idl>
module GroupStruct_int{
  interface SGsvc_int{

    // --- Confirmacao da distribuicao de um evento
    // de estruturacao de grupo
    //
    oneway void event_confirm (in Event event);
  };
};
```

### A.2.3 Interface de Implementação do SGsvc

```

module GroupStruct_impl{
    interface SGsvc_impl : GroupStruct::SGsvc,
                          GroupStruct_int::SGsvc_int {};
};

```

## A.3 Interfaces do SGlocal

Esta seção descreve a interface externa (que apresenta operações para acesso aos serviços de comunicação de grupo) e interface interna (que possui operações para permitir aos demais objetos de suporte a grupos indicarem ao SGlocal a ocorrência de eventos). É também definida a interface de implementação, derivada das interfaces interna e externa do objeto.

### A.3.1 Interface Externa do SGlocal

```

#include <defs.idl>
module GroupComm{
    interface SGlocal_ext{

        // --- Distribuicao de mensagens de interrogacao
        //
        unsigned long group_invoke (in MessageStruct message);

        // --- Distribuicao de mensagens de anuncio
        //
        boolean group_announce (in MessageStruct message);

        // --- Distribuicao de mensagens de resposta
        //
        boolean group_reply (in MessageStruct message,
                            in unsigned long interrogation_id);

        // --- Operacao atraves da qual os objetos de aplicacao
        // podem obter a configuracao de um grupo
        //
        void query_group (in string group_name,
                        out MemberList member_list,
                        out RoleList role_list,
                        out PolicyList policy_list);

    }; // end interface
}; // end module

```

## A.3.2 Interface Interna do SGlocal

```

#include <defs.idl>
module GroupComm_int{
  interface SGlocal_int{

    // --- Operacao atraves da qual o SGlocal recebe notificacao
    // da criacao de novos grupos dos quais ele seja um membro;
    // esta operacao e' invocada pelo SGsvc e permite a
    // transferencia de informacoes sobre o novo grupo (entre
    // elas, a representacao interna da lista de membros e as
    // referencias de objeto do SGcoordenador e do objeto de
    // aplicacao)
    //
    oneway void new_group (in string group_name,
                          in string group_creator,
                          in MemberListInt member_list,
                          in SGcoord obj_SGcoord,
                          in Applic application_object,
                          in PolicyList policy_list);

    // --- Operacao atraves da qual os demais objetos de suporte
    // a grupos (SGsvc, SGcoordenador e SGlocal) podem informar
    // ao SGlocal a ocorrencia de eventos no grupo (tais eventos
    // podem representar mensagens ou a execucao de algum servico
    // de estruturacao de grupo)
    //
    oneway void event_indication (in Event event);

    // --- Operacao atraves da qual um objeto SGcoordenador informa
    // ao SGlocal a entrada deste em um grupo, transferindo-lhe
    // as informacoes de contexto para inicializacao do novo
    // membro (as quais sao repassadas para a aplicacao)
    //
    oneway void member_init (in string group_name,
                            in InitData init_data);

    // --- Operacao usada pelo SGcoordenador para solicitar ao
    // SGlocal informacoes de contexto de um grupo usadas na
    // inicializacao de novos membros (o SGlocal obtem estas
    // informacoes junto ao respectivo objeto de aplicacao)
    //
    InitData query_init_data (in string group_name,
                              in string new_member_name);

  }; // end interface
}; // end module

```

### A.3.3 Interface de Implementacao do SGlocal

```
module GroupComm_impl{
    interface SGlocal_impl : GroupComm::SGlocal,
                          GroupComm_int::SGlocal_int {};
};
```

## A.4 Interface do SGcoordenador

O objeto SGcoordenador possui apenas uma interface, que contém operações para uso interno ao suporte a grupos.

```
#include <defs.idl>
module GroupCoord{
    interface SGcoord{

        // --- Operacao usada pelo objeto SGsvc para inicializar
        // o SGcoordenador apos a criacao de um novo grupo
        //
        oneway void group_info (in string group_name,
                               in string group_creator,
                               in MemberListInt member_list,
                               in PolicyList policy_list);

        // --- Operacao utilizada pelos objetos SGlocal e SGsvc
        // para solicitar a distribuicao ordenada de eventos
        //
        oneway void distrib_event (in Event event);
    };
};
```

## A.5 Interface dos Objetos de Aplicação

Os objetos de aplicação devem prover uma interface uniforme (padronizada) para uso pelos objetos de suporte a grupos. Esta interface contém, basicamente, operações que permitem ao suporte a grupos entregar mensagens às aplicações, realizar consultas (determinadas pelas políticas de serviços), bem como indicar a ocorrência de eventos como a entrada ou saída de membros de um grupo.

```
#include <defs.idl>
interface Application{

    // --- Operacao que permite consultar os usuarios (objetos de
    // aplicacao) sugeridos como membros de um grupo em criacao
    // sobre sua participacao no grupo
```



```
//
boolean query_new_group (in string group_creator,
                        in string group_name,
                        in MemberList member_list);

// --- Operacao usada pelo servico de Suporte a Grupos para
// informar ao usuario criador o resultado das consultas
// realizadas durante o processo de criacao do grupo e
// solicitar determinacao sobre efetivacao ou nao da
// criacao do grupo
//
boolean query_result (in string group_name,
                    in MemberList confirmed_members);

// --- Operacao que permite informar o objeto de aplicacao
// sobre a criacao de um novo grupo
//
oneway void new_group (in string group_creator,
                     in string group_name,
                     in MemberList member_list,
                     in PolicyList policy_list,
                     in SGlocal obj_SGlocal);

//
// Operacoes que permitem consultar os usuarios (objetos
// de aplicacao) sobre a realizacao de alteracoes na
// estrutura de um grupo:

// --- consulta sobre a entrada de um novo membro em um grupo
//
boolean query_join (in string group_name,
                  in string responsible_member,
                  in Member new_member);

// --- consulta sobre a saida de um membro de um grupo
//
boolean query_leave (in string group_name,
                   in string requester,
                   in Member leaving_member);

// --- consulta sobre a finalizacao de um grupo
//
boolean query_finish_group (in string group_name,
                          in string requester);
```



# Referências

- [ADM92] Y. Amir, D. Dolev, e D. Malki. Membership algorithms for multicast communication groups. *Lecture Notes in Computer Science - 647*, pp. 292–312, 1992.
- [BC90] K. Birman e R. Cooper. The ISIS project: Real experience with a fault tolerant programming system. Technical Report TR90-1138, Department of Computer Science, Cornell University, 1990.
- [BCG91] K.P. Birman, R. Cooper, e B. Gleeson. Design alternatives for process group membership and multicast. Technical Report TR 91-1257, Department of Computer Science, Cornell University, dezembro de 1991.
- [Bir93] K.P. Birman. The process group approach to reliable distributed computing. *Communications of the ACM*, 36(12):37–53, dezembro de 1993.
- [BJ87a] K.P. Birman e T.A. Joseph. Exploiting virtual synchrony in distributed systems. *Operating Systems Review*, 21(5):123–138, novembro de 1987.
- [BJ87b] K.P. Birman e T.A. Joseph. Reliable communication in the presence of failures. *ACM Trans. Computer Systems*, 5(1):47–76, fevereiro de 1987.
- [BKR93] A. Beitz, P. King, e K. Raymond. Is DCE a support environment for ODP? In *International Conference on Open Distributed Processing*, Berlin, setembro de 1993.
- [BN84] A. Birrel e Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1):39–59, fevereiro de 1984.
- [BR91] G. Blair e T. Rodden. CSCW and distributed systems: The problem of control. Technical Report CSCW.1.91, Lancaster University, UK, 1991.
- [BS94] O. Babaoglu e A. Schiper. On group communication in large-scale distributed systems. Technical Report UBCLS-94-19, Laboratory for Computer Science, University of Bologna, julho de 1994.
- [BSS91] K. Birman, A. Schiper, e P. Sephenson. Lightweight causal and atomic group multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, agosto de 1991.

- [CM95] F.M. Costa e E.R.M. Madeira. Suporte a grupos cooperativos na plataforma Multiware. In *XXII Software and Hardware Symposium / PANEL'95*, pp. 563-574, Canela, RS, Brasil, agosto de 1995. Sociedade Brasileira de Computação.
- [CM96] F.M. Costa e E.R.M. Madeira. An Object Group Model and its Implementation to Support Cooperative Applications on CORBA. In *ICDP'96 - International Conference on Distributed Platforms*, 1996. (aceito para publicação).
- [Com91] D.E. Comer. *Internetworking with TCP/IP - Volume I - Principles, Protocols, and Architecture*. Prentice-Hall International, 2nd edição, 1991.
- [CS93] D.R. Cheriton e D. Skeen. Understanding the limitations of causally and totally ordered communication. In *Proc. of the 4th Symposium on Operating Systems Principles*, pp. 44-57. ACM, dezembro de 1993.
- [CSB92] G. Coulson, J. Smalley, e G.S. Blair. The design and implementation of a group invocation facility in ANSA. Technical Report MPG-92-34, Department of Computing, Lancaster University, 1992.
- [CZ85] D.R. Cheriton e W. Zwaenepoel. Distributed process groups in the V Kernel. *ACM Transactions on Computer Systems*, 3(2):77-107, maio de 1985.
- [EGR91] C.A. Ellis, S.J. Gibbs, e G.L. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1), janeiro de 1991.
- [Gar94] C.M. Garcia. Uma proposta para modelagem de funções de gerenciamento para processamento distribuído aberto. Master's thesis, Departamento de Ciência da Computação - UNICAMP, Campinas, SP, dezembro de 1994.
- [ION95] IONA Technologies Ltd, Dublin, Ireland. *Orbix Programmer's Guide - Release 1.3.1*, fevereiro de 1995.
- [Isi93] Isis Distributed Systems Inc. *Object Groups: A response to the ORB 2.0 RFI*, abril de 1993. OMG Doc. 93-4-11.
- [ISO93] ISO / ITU-T. *ISO/IEC JTC1/SC21/WG7 N839. ODP Reference Model - Part 4: Architectural Semantics*, julho de 1993.
- [ISO94a] ISO / ITU-T. *ITU-T X.901 — ISO/IEC CD 10746-1. ODP Reference Model - Part 1. Overview*, julho de 1994.
- [ISO94b] ISO / ITU-T. *ITU-T X.902 — ISO/IEC CD 10746-2. ODP Reference Model - Part 2: Descriptive Model*, fevereiro de 1994.
- [ISO94c] ISO / ITU-T. *ITU-T X.903 — ISO/IEC CD 10746-3. ODP Reference Model - Part 3: Prescriptive Model*, fevereiro de 1994.

- [Jal94] P. Jalote. *Fault Tolerance in Distributed Systems*. Prentice Hall, 1994.
- [JB89] T.A. Joseph e K.P. Birman. Reliable broadcast protocols. In Sape Mullender, editor, *Distributed Systems*, capítulo 14, pp. 293–317. ACM Press, 1989.
- [KP90] M.J. Knister e A. Prakash. Distedit: A distributed toolkit for supporting multiple group editors. In *CSCW'90*, pp. 343–355, 1990.
- [KT92] M.F. Kaashoek e A.S. Tanenbaum. Efficient reliable group communication for distributed systems. Technical Report IR-295, Vrije University, Amsterdam, junho de 1992.
- [KTV93] M.F. Kaashoek, A.S. Tanenbaum, e K. Verstoep. Group communication in Amoeba and its applications. *Distributed Systems Engineering Journal*, 1:48–58, julho de 1993.
- [Lam78] L. Lamport. Time, clocks, and the ordering of events in distributed systems. *Communications of the ACM*, 21(7):558–565, julho de 1978.
- [LCN90] L. Liang, S.T. Chanson, e G.W. Neufeld. Process groups and group communications: Classifications and requirements. *IEEE Computer*, 23(2):57–66, fevereiro de 1990.
- [LFSM94] W.P.D.C. Loyolla, I.R. Fontes, F.C. Sica, e M.J. Mendes. Plataforma Multiware: Especificação da Camada de Suporte para Aplicações CSCW. In *12o. Simpósio Brasileiro de Redes de Computadores*, pp. 407–425, Curitiba, Maio de 1994.
- [Lim94] L.A.P. Lima. Um modelo para a implementação de Federação de Traders. Master's thesis, Departamento de Ciência da Computação - UNICAMP, Campinas, SP, 1994.
- [Lin93] R.J. van der Linden. An overview of ANSA. Technical Report AR.000, Architecture Projects Management Limited, UK, maio de 1993.
- [LM95] L.A.P. Lima e E.R.M. Madeira. A model for a Federative Trader. In *Proceedings of the International Conference on Open Distributed Processing*, pp. 155–166, Brisbane, Australia, fevereiro de 1995.
- [LMM<sup>+</sup>94] W.P.D.C. Loyolla, E.R.M. Madeira, M.J. Mendes, E. Cardozo, e M.F. Magalhães. Multiware platform: An open distributed environment for multimedia cooperative applications. *IEEE COMPSAC'94, Taipei, Taiwan*, novembro de 1994.
- [Mad95] E.R.M. Madeira. Multiware Platform: Some Issues about the Middleware Layer. In *7th IASTED - International Conferences on Parallel and Distributed Computing and Systems*, outubro de 1995. (aceito para publicação).
- [Maf93] S. Maffei. Distributed programming using object-groups. Technical Report IFI TR93.38, University of Zurich, Dept. of Computer Science, Zurich, Switzerland, setembro de 1993.

- [MM94] M.J. Mendes e E.R.M. Madeira. Plataforma Multiware: Projeto e desenvolvimento da Camada Middleware. In *12o. Simpósio Brasileiro de Redes de Computadores*, pp. 93-112, Curitiba, Maio de 1994.
- [MNR92] M. Medina, L. Navarro, e T. Rodden. Environment support for cooperative working. Technical Report CSCW.14.92, Lancaster University, UK, 1992.
- [NWM93] J.R. Nicol, T.C. Wilkes, e F.A. Manola. Object orientation in heterogeneous distributed computing systems. *IEEE Computer*, 26(6), junho de 1993.
- [OE93] E. Oskiewicz e N. Edwards. A model for interface groups. ANSA Architecture Report AR002, Architecture Projects Management Limited, fevereiro de 1993.
- [OMG91] Object Management Group. *The Common Object Request Broker: Architecture and Specification - Rev. 1.1*, dezembro de 1991.
- [OMG92a] Object Management Group. *Object Management Architecture Guide - Rev. 2.0*, setembro de 1992.
- [OMG92b] Object Management Group. *Object Request Broker 2.0 Extensions - Request For Information*, dezembro de 1992. OMG Doc. 92-12-10.
- [OMG94a] Object Management Group. *Common Facilities Architecture - Issue 3.0*, novembro de 1994. OMG TC Doc. 94-11-9.
- [OMG94b] Object Management Group. *Revised C++ Mapping Submission*, setembro de 1994. OMG TC Doc. 94-9-14 (Final submission to the Object Request Broker 2.0 C++ mapping RFP).
- [OMG95a] Object Management Group. *Object Model*, janeiro de 1995. OMG TC Doc. 95-1-13 (Draft of the revised OMG Object Model).
- [OMG95b] Object Management Group. *Object Services Architecture - Rev. 8.1*, janeiro de 1995. OMG TC Doc. 95-1-47.
- [OSF90] Open Software Foundation. *Distributed Computing Environment Overview*, setembro de 1990.
- [Par92a] P. Pardyak. Group communication in a distributed object-based system. Master's thesis, University of Mining and Metallurgy, Cracow, Poland, setembro de 1992.
- [Par92b] P. Pardyak. Group communication in an object-based environment. In *Proceedings of the 2nd International Workshop on Object Orientation in Operating Systems*, pp. 106-116, Dourdan, France, setembro de 1992. IEEE Computer Society Press.
- [PMC94] PostModern Computing Technologies, Inc., Mountain View, California, USA. *ORBe-line User's Guide*, 1994.

- [Ray93] K. Raymond. Reference model for open distributed processing: a tutorial. In *International Conference on Open Distributed Processing*, Berlin, setembro de 1993.
- [RB93] A.M. Ricciardi e K.P. Birman. Process membership in asynchronous environments. Technical Report TR-93-1328, Department of Computer Science, Cornell University, fevereiro de 1993.
- [RBP<sup>+</sup>91] J. Runbaugh, M. Blaha, W. Premerlani, F. Eddy, e W. Lorenzen. *Object Oriented Modeling and Design*. Prentice Hall Inc., 1991.
- [Ren93] R. van Renesse. Causal controversy at le mont st.-michel. *Operating Systems Review*, 27(2), abril de 1993.
- [Rod92] T. Rodden. Supporting cooperative applications. Technical Report CSCW.11.92, Computing Department, Lancaster University, UK, 1992.
- [SR92] K. Schmidt e T. Rodden. Putting it all together: Requirements for a CSCW platform. Technical Report TR-CSCW-9-92, Computing Department, Lancaster University, 1992.
- [SST94] R. Simon, R. Sciabassi, e Z. Taieb. Communication control in computer supported cooperative work systems. In *CSCW'94*, pp. 311 -321, 1994.
- [Tan92] A. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 1992.
- [TMM<sup>+</sup>93] V. Tschammer, M.J. Mendes, E.R.M. Madeira, W.L. Souza, e W.P.D.C. Loyolla. Processamento distribuído aberto e o modelo RM-ODP / ISO. In *11o. Simpósio Brasileiro de Redes de Computadores*, Campinas, 1993.
- [Vin93] S. Vinoski. Distributed object computing with CORBA. *C++ Report*, julho/agosto de 1993.