Dissertação de Mestrado

Inatel
Instituto Nacional de Telecomunicações

ESTUDO, ANÁLISE E PROPOSTA DE UMA ARQUITETURA DE INTEGRAÇÃO DE SISTEMAS HETEROGÊNEOS EM REDES DE LARGA ESCALA COM ADAPTAÇÃO DE UMA PLATAFORMA DE SUPORTE A AGENTES MÓVEIS (FT-SALE/SMAM)

CRISTIANE MOREIRA DA SILVA

DEZEMBRO / 2006

INSTITUTO NACIONAL DE TELECOMUNICAÇÕES – INATEL

MESTRADO EM TELECOMUNICAÇÕES

ESTUDO, ANÁLISE E PROPOSTA DE UMA ARQUITETURA DE INTEGRAÇÃO DE SISTEMAS HETEROGÊNEOS EM REDES DE LARGA ESCALA COM ADAPTAÇÃO DE UMA PLATAFORMA DE SUPORTE A AGENTES MÓVEIS (FT-SALE/SMAM).

CRISTIANE MOREIRA DA SILVA

Dissertação apresentada ao Mestrado em Telecomunicações do Instituto Nacional de Telecomunicações – INATEL, como requisito parcial para obtenção do título de Mestre em telecomunicações.

ORIENTADOR: PROF. DR. ANTÔNIO MARCOS ALBERTI

CO-ORIENTADOR: PROF. DR. ANILTON SALLES GARCIA

SANTA RITA DO SAPUCAÍ - MG

2006

CRISTIANE MOREIRA DA SILVA

ESTUDO, ANÁLISE E PROPOSTA DE UMA ARQUITETURA DE INTEGRAÇÃO DE SISTEMAS HETEROGÊNEOS EM REDES DE LARGA ESCALA COM ADAPTAÇÃO DE UMA PLATAFORMA DE SUPORTE A AGENTES MÓVEIS (FT-SALE/SMAM).

Esta dissertação foi julgada e aprovada para a obtenção
do título de Mestre em Telecomunicações do Instituto
Nacional de Telecomunicações

Santa Rita do Sapucaí, 15 de Dezembro de 2006.

Membros da Banca

Prof. Dr. **Antônio Marcos Alberti**
Orientador

Profa. Dra. **Rosane Bodart Soares**
Examinador Externo

Prof. Dr. **Carlos Alberto Ynoguti**
Examinador Interno

DEDICATÓRIA

Dedico este trabalho a minha família, em especial a minha mãe, por minha formação moral e por sempre acreditar em mim. À minha filha Camila pelo seu carinho e compreensão. Aos meus amigos e a todos aqueles que direta ou indiretamente contribuíram para a realização de mais esta etapa em minha vida.

AGRADECIMENTOS

A Deus por estar sempre presente em minha vida me dando forças e me fazendo enxergar oportunidades para que eu possa ir cada vez mais adiante.

Ao meu orientador professor Dr. Antônio Marcos Alberti pelo apoio e por acreditar e confiar em minha escolha.

Ao meu co-orientador professor Dr. Anilton Salles Garcia pelo incentivo e pelo apoio moral e técnico que foram essenciais para a conclusão desta pesquisa.

Aos amigos Mara Rúbia, Lara Mendonça e Willian Hisatugu pela amizade, companheirismo e pelo apoio.

Ao Prof. Adonias pela oportunidade.

A Robélia, Aline e a todos os funcionários do INATEL que com uma enorme boa vontade, facilitaram todas as etapas necessárias para tornar possível minha manutenção no Mestrado.

À querida amiga Susiléa Abreu dos Santos Lima pelo grande apoio e por todos os momentos engraçados e divertidos, porém bastante produtivos, durante nossa fase de pesquisa.

Ao amigo Salverino pelo incentivo.

Ao Moacyr Weiss pelo companheirismo.

A amiga Carina Darós pelo apoio necessário para que eu pudesse me dedicar às pesquisas em um dos momentos mais complicados de meu mestrado.

A todos de minha família por acreditar que eu seria capaz.

A minha filha Camila por ser o meu maior incentivo para estar sempre em busca do crescimento pessoal e profissional.

A todos que direta ou indiretamente contribuíram para a realização de mais esta vitória em minha vida.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Uma Empresa Virtual é uma aliança temporária de empresas que se reúnem para partilhar habilidades ou recursos e suas competências principais, visando melhor responder a oportunidades de negócios, e cuja cooperação é provida através de redes de computadores. Nesta dissertação, é apresentada uma proposta de integração de duas plataformas que atuam no contexto das Empresas Virtuais: a **Plataforma de Busca e Seleção de Parceiros**, desenvolvida explorando os benefícios do paradigma de agentes móveis, o que garante maior agilidade e eficiência no processo de busca e seleção de parceiros para a formação de Empresas Virtuais, e a **Plataforma FT-SALE**, cuja proposta é adequar as técnicas de tolerância a falhas, baseadas nas especificações **FT-CORBA**, aos ambientes de Empresas Virtuais. Assim, a principal vantagem desta união é utilização das técnicas de tolerância a falhas na melhoria da confiabilidade do processo de busca e seleção de parceiros. Além disso, a inserção de um **Sistema Multi-Agentes Móveis** (**SMAM**) no contexto da **Plataforma FT-SALE** abre outras oportunidades que podem ser exploradas em trabalhos futuros. Portanto, a principal motivação para este trabalho foi propor melhorias para o processo de detecção de falhas e o gerenciamento de controle de grupo (*membership*) na **Plataforma de Busca e Seleção de Parceiros**, fornecendo uma maior confiabilidade na formação das Empresas Virtuais e na comunicação entre objetos. Assim sendo, esta dissertação apresenta e discute as principais tecnologias por de trás dessas plataformas, bem como propõe uma nova plataforma chamada de **Plataforma FT-SALE/SMAM**. Por fim, são fornecidas algumas diretrizes para a implementação da plataforma proposta.

ABSTRACT

A Virtual Enterprise is a temporary alliance of enterprises that come together to share skills or core competencies and resources in order to better respond to business opportunities, and whose cooperation is supported by computer networks. In this dissertation, it is presented an integration proposal of two platforms that work in the context of Virtual Enterprises: the **Partners Search and Selection Platform**, developed to explore the benefits of mobile agents paradigm, that allows more agility on the partners search and selection process during these enterprises implementation, and the **FT-SALE Platform**, whose proposal is to adequate the **FT-CORBA** failure tolerant techniques to the Virtual Enterprise environment. Thus, the main advantage of this merge is to use the failure tolerant techniques to improve reliability on Virtual Enterprise partner and selection process. Besides this, the insertion of a mobile multi-agent system in the context of the **FT-SALE Platform** opens other opportunities that can be explored on future works. Therefore, the main motivation for his work was to improve failure detection and membership control management on **Partners Search and Selection Platform**, offering more reliability for Virtual Enterprise formation and objects communication. Thus, this dissertation presents and discuss the main technologies behind these platforms as well as proposes a new platform called **FT-SALE/SMAM Platform**. At the end, they are presented guidelines for its implementation.

LISTA DE ACRÔNIMOS

AMEF	Agile Manufacturing Enterprise Forum
AAM	Plataforma de Agentes Móveis
BOA	*Basic Object Adapter*
BP	*Business Process*
CIM	*Computer Integrated Manufacturing*
CORBA	*Common Object Request Broker Architecture*
COSS	*Common Object Service Specification*
DCOM	*Distributed Component Object Model* (*Microsoft*)
DBP	*Distributed Business Process*
DLLs	*Dynamic Link Libraries.*
DII	*Dynamic Invocation Interface*
DSI	*Dynamic Skeleton Interface*
DSOM	*Distributed System Object Model* (IBM)
EA	*Enterprise Activities*
EDI	*Electronic Data Interchange*
EV	Empresa Virtual
FT	*Fault-Tolerant* (Tolerância à Falhas)
FT-CORBA	*Fault-Tolerant CORBA Specification*

GSeq	*Gateway Seqüenciador*
IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
IDL	*Interface Definition Language*
IIOP	*Internet Inter-ORB Protocol*
IOGR	*Interoperable Object Group Reference*
IOR	*Interoperable Object Reference*
IP	*Internet Protocol*
KQML	*Knowledge Query and Manipulation Language*
LAN	*Local Area Network*
MAF	*Mobile Agents Facility*
MAS	*Multi-agent Systems*
MIE	Metodologia de Integração de Empresas
OMA	*Object Management Architecture*
OMG	*Object Management Group*
ON	Oportunidade de Negócio
ORB	*Object Request Broker*
OSI	*Open System Interconnection*
OV	Organização Virtual

RMI	*Remote Method Invocation* (Sun)
SCG	Suporte de Comunicação de Grupo
SMA	Sistema Multi-Agente
SMAM	Sistema Multi-Agente Móveis
SMG	Serviço de *Membership* de Grupo
SSL	*Secure Sockets Layer*
TCP	*Transport Control Protocol*
TI	Tecnologia de Informação
XML	*Extended Markup Language*

1. INTRODUÇÃO

A **Internet** provê uma poderosa infra-estrutura de comunicação, possibilitando assim o surgimento dos sistemas distribuídos. A ampliação da rede pública e o crescimento do número de aplicações que fazem uso deste meio tornaram a Internet peça fundamental para os sistemas com abrangência global. Porém, juntamente com as facilidades providas pela Internet, surgiram novos desafios para o desenvolvimento de aplicações distribuídas, como a flexibilidade, modularidade e escalabilidade, exigindo assim a criação de novas tecnologias.

A Internet possibilitou ainda abertura de mercado e de globalização econômica. Com isso, as empresas têm enfrentado grandes desafios para aumentarem seu nível de eficiência e assim se manterem competitivas [Rabelo00].

Não satisfeitas em usar os computadores para gestão de seus negócios, as empresas querem utilizar seus computadores para aperfeiçoar e administrar os processos de geração de bens e serviços, fazendo com que as máquinas cada vez mais trabalhem de forma integrada, para reconhecer e responder a novas oportunidades que o mercado exige.

Nas últimas décadas, com o advento da Internet, observa-se uma tendência natural das empresas na busca de uma rápida integração, e uma busca constante por melhor desempenho neste espaço virtual. A união temporária de empresas, de forma a colaborar na obtenção de um produto final, leva à formação das chamadas **Empresas Virtuais** (**EVs**). Estas empresas normalmente são formadas por núcleos originalmente independentes, mas que se unem em busca de um objetivo comum, e trocam informação de modo a coordenar as suas ações.

Empresas Virtuais podem ser definidas como sendo uma aliança temporária entre empresas que se dispõem a compartilhar técnicas, ou negócios afins, e pesquisas, de maneira a melhor reagir às oportunidades de negócios, e cuja cooperação é mantida por uma rede de computadores, onde cada empresa é um nó da rede. Neste cenário, os mecanismos de comunicação, para se mostrarem eficientes, devem estar sobre uma infra-estrutura robusta que garanta segurança, desempenho e alta confiabilidade, principalmente no estabelecimento de transações entre as empresas e da empresa com o cliente.

Entretanto, conectar-se em rede significa sujeitar, mesmo que sob condições específicas e com algum tipo de controle, a exposição dos sistemas da organização a possíveis falhas provenientes destas redes, tais como: problemas relativos a atrasos, falhas de comunicação, problemas com o meio de transmissão, falhas de terminais e demais problemas existentes em redes de telecomunicações.

Tais falhas, se não forem devidamente tratadas, podem tornar o sistema vulnerável. Portanto, podem gerar ameaças à integridade e confiabilidade dos dados e serviços oferecidos pela empresa, além de gerar prejuízos e comprometer a produção do produto final. Este fato pode resultar em conseqüências desastrosas, tais como altos prejuízos financeiros e uma perda incalculável da confiança dos clientes e parceiros, visto que, dependendo da gravidade destas falhas, as mesmas podem ocasionar uma perda parcial, ou até mesmo completa, dos dados desta organização.

Várias técnicas de projeto podem ser usadas para aumentar a confiabilidade e a disponibilidade de equipamentos e serviços de computação. Mesmo assim, sistemas totalmente infalíveis são impossíveis, pois falhas são inevitáveis. Desta forma, para garantir alta qualidade e confiabilidade, torna-se necessário o emprego de técnicas de tolerância a falhas nestas redes [Weber02].

Portanto, aplicações distribuídas, como as encontradas em Empresas Virtuais, devem conter mecanismos de tolerância a falhas que garantam propriedades como confiabilidade, eficiência e integridade, para que as mesmas possam operar de forma correta e eficaz. Para isso, precisam dispor de técnicas de tolerância a falhas que permitam tornar as aplicações distribuídas confiáveis no sentido de evitar que as informações das empresas envolvidas sejam perdidas ou danificadas (corrompidas).

A arquitetura **CORBA** (*Common Object Request Broker Architecture*) [OMG02], um padrão aberto para sistemas abertos e distribuídos baseados em objetos, foi proposta pela **OMG** (*Object Management Group*) com o intuito de solucionar as dificuldades encontradas no desenvolvimento de aplicações distribuídas. O **CORBA** fornece funcionalidades para a interoperabilidade e a portabilidade em aplicações distribuídas, tais como a transparência de localização e heterogeneidade de plataforma e de linguagens de programação.

Buscando a confiabilidade no sistema de objetos distribuídos, a **OMG** acrescentou o Serviço de Tolerância a Falhas CORBA (**FT-CORBA,** do inglês *Fault-Tolerant CORBA*) [FT-CORBA01] a especificação de serviços CORBA.

Uma proposta de inserção de técnicas de tolerância a falhas, em sistemas de Empresas Virtuais, baseadas nas especificações **FT-CORBA**, foi a plataforma **FT-SALE** [Lima01a], que une a plataforma **SALE** (Sistema Abertos de Larga Escala para Empresas Virtuais) [Fraga01a] e as experiências do **GrouPac 2.0** [Laurindo01] para tratar tolerância a falhas em redes de larga escala.

O objetivo principal da formação de uma Empresa Virtual é o de várias empresas unirem competências (cooperar) a fim de atender a uma nova **Oportunidade de Negócio** (**ON**) de forma ágil e eficiente, ampliando sua participação no mercado. Contudo, de nada adiantaria a implantação de **técnicas de tolerância a falhas** nos sistemas destas Empresas Virtuais se a escolha das empresas parceiras fosse feita de forma lenta e incorreta (escolha de parceiros que não se encaixam no perfil da nova **ON**). O ideal seria a união das duas coisas, ou seja, um sistema tolerante à falhas e que garanta agilidade e eficiência na escolha de parceiros para a formação de uma **EV**.

Visando prover maior flexibilidade e eficiência na **Fase de Criação** destas empresas, um **Sistema de Busca e Seleção de Parceiros** para a formação de Empresas Virtuais foi proposto em [Schmidt03]. O Sistema explora os benefícios do paradigma de agentes móveis para garantir uma maior agilidade na apresentação das **ONs** ao grupo de empresas e uma maior eficiência na formação e análise das possíveis empresas virtuais a serem constituídas. Para se adequar às redes de larga escala, como a Internet, o Sistema considera requisitos de interoperabilidade, escalabilidade, e portabilidade, além de aspectos de segurança para a comunicação dos agentes móveis, fortalecendo o processo de construção de confiança na formação de **EV**s.

No entanto, o sistema de Busca e Seleção possui um único Gerenciador de Grupo para cada **OV**, o que o torna vulnerável, visto que em caso de falha deste Gerenciador de Grupo, todo o controle relativo aos componentes da Organização Virtual (grupos de empresas para formar **EVs**) é perdido. Uma solução para este problema é a inserção de um controle de membros (*membership*) e de técnicas de tolerância a falhas neste sistema.

1.2 Objetivos

Esta dissertação tem por objetivo principal apresentar uma proposta visando inserir tolerância a falhas na **Plataforma de Busca e Seleção de Parceiros** para Empresas Virtuais, proposta por [Schmidt03], utilizando como base as especificações de CORBA tolerantes à falhas (**FT-CORBA**). Para o desenvolvimento desta proposta foram utilizados como ponto de partida o **Sistema Multi-Agentes Móveis**, base da **Plataforma de Busca e Seleção de Parceiros**, juntamente com a **Plataforma FT-SALE** [Lima01b], que insere técnicas de tolerância a falhas em ambientes de Empresas Virtuais através da inclusão de soluções que visam adaptar as especificações **FT-CORBA** ao uso em ambientes de larga escala.

Assim sendo, um outro objetivo desta dissertação é propor uma nova plataforma baseada em **Agentes Móveis** e técnicas de tolerância a falhas em CORBA. Ou seja, esta plataforma deve permitir a execução do **Sistema de Busca e Seleção de Parceiros,** proposto em [Schmidt03], e superar as limitações de tolerância a falhas existentes neste sistema, principalmente devido a existência de um único **Gerenciador de Grupo**. A proposta se dá através da inserção de um **Sistema Multi-Agentes Móveis** à **Plataforma FT-SALE**, dando origem a uma nova plataforma chamada **Plataforma FT-SALE/SMAM**. Além disso, este trabalho apresenta e discute alguns dos principais aspectos das tecnologias que dão suporte a essas plataformas, bem como fornece diretrizes para a implementação da plataforma proposta.

1.3 Trabalhos Relacionados

Baseado nas definições expostas acima, a formação de uma Empresa Virtual pode ser considerada uma forma de aumentar a eficiência das empresas e de manter estas competitivas no mercado atual. Entretanto, estas empresas devem possuir uma infra-estrutura de suporte que permita seu funcionamento em redes de larga escala e a comunicação entre sistemas heterogêneos. Visando contornar alguns problemas encontrados nos sistemas de Empresas Virtuais, tais como falhas, e garantir assim a confiabilidade do sistema, principalmente na fase de criação, alguns trabalhos foram desenvolvidos no meio acadêmico. Neste trabalho, técnicas de Tolerância a Falhas baseadas nas definições de Tolerância a Falhas do CORBA (FT-CORBA) serão integradas ao uso de agentes móveis, os quais minimizam as diferenças causadas por seus sistemas legados e cooperam na resolução das tarefas. Esta união busca prover confiabilidade,

flexibilidade e interoperabilidade ao sistema de uma EV. A seguir são apresentados trabalhos relacionados ao tratamento de falhas e ao uso de agentes móveis em sistemas de EVs:

Em [Lima01b] é apresentada uma proposta para inserir tolerância a falhas dentro do contexto de Empresas Virtuais utilizando como base as especificações FT-CORBA. Para o desenvolvimento desta proposta foi utilizada a plataforma SALE (Sistema Abertos de Larga Escala para Empresas Virtuais) [Fraga01a] e as experiências do GroupPac 2.0 [Lung01] para tratar tolerância a faltas em redes de larga escala. A plataforma SALE é adequada para tratar novas aplicações que se configuram como sistemas globalmente distribuídos e que tratam com informações de diferentes naturezas. O objetivo principal deste trabalho foi apresentar uma proposta de adequação das especificações FT-CORBA para prover confiabilidade em Empresas Virtuais, o que deu origem à arquitetura FT-SALE, a qual será usada como arquitetura base para a plataforma FT-SALE/SMAM proposta nesta dissertação.

Uma outra base para a arquitetura proposta nesta dissertação é o sistema de Busca e Seleção de parceiros para a formação de uma Empresa Virtual, proposto por Ricardo Schmidt em [Schmidt03]. Este trabalho define uma abordagem para auxiliar a Fase de Criação de EVs baseada em uma arquitetura híbrida de agentes. São explorados os benefícios do paradigma de agentes móveis para garantir uma maior agilidade na apresentação das oportunidades de negócios ao grupo de empresas e uma maior eficiência na formação e análise das possíveis empresas virtuais. O modelo proposto foi adequado aos sistemas de larga escala, como a Internet, através de requisitos de interoperabilidade, escalabilidade e portabilidade considerados durante sua concepção. O trabalho incorporou ainda, aspectos de segurança para a comunicação dos agentes móveis, fortalecendo o processo de construção de confiança na formação de EVs. Para a validação do modelo, foi implementado um protótipo que simula uma Organização Virtual existente onde, dada uma tarefa a ser realizada, efetua-se a busca e seleção de parceiros mais adequados à composição de uma EV, permitindo inclusive a qualquer membro atuar como um coordenador deste processo dentro do grupo [Schmidt03]. Outros trabalhos relacionados à criação de Empresas Virtuais são apresentados em [Schmidt03].

Ainda dentro do contexto do uso de agentes móveis, [Wangham03b] define um esquema de segurança para proteção das plataformas de agentes móveis baseado em redes de

confiança SPKI/SDSI, considerando sistemas distribuídos e de larga escala. O esquema é composto por um protocolo de autenticação mútua para as plataformas envolvidas, por um autenticador de agentes móveis e por um mecanismo para geração de domínios de proteção. Devido à flexibilidade dos mecanismos de delegação de certificados SPKI/SDSI adotados, o esquema proposto forneceu um controle descentralizado de autorização e de autenticação. Foi feita ainda, uma integração do protótipo a uma aplicação de Busca e Seleção de parceiros para a formação de Empresas Virtuais baseada em agentes móveis, o que possibilitou analisar a viabilidade do esquema de segurança proposto para aplicações na Internet.

Em [Macedo01] é proposta uma metodologia de negociação, no contexto dos Sistemas Multi-Agentes, adaptada ao cenário de formação de uma Empresa Virtual. O trabalho apresentou uma proposta de um projeto de um Sistema Multi-Agente que representa o mercado eletrônico, e onde a Empresa Virtual se constitui e apóia. O processo de negociação desenvolvido e proposto entre empresas independentes visa a formação da Empresa Virtual, traduzindo-se em uma negociação multi-atributo, iterativa, com características apropriadas ao contexto de uma atividade comercial.

Foi proposto em [Macedo01] um algoritmo de negociação (negociação-Q) que assegurou duas importantes funcionalidades: a privacidade e a aprendizagem. O sistema manteve a privacidade de informação relativa a utilidade de cada empresa, sem prejuízo do poder de negociação no mercado. A aprendizagem foi incluída no sistema, visando permitir que uma empresa que não conhece, à priori, os seus parceiros de negócio, saiba como se adaptar às constantes alterações do mercado. O algoritmo de negociação-Q é um algoritmo de negociação adaptativa multi-critério, que usa argumentação qualitativa.

Para solucionar o problema da dependência entre negociações simultâneas e interdependentes na formação da Empresa Virtual, foi desenvolvido em [Macedo01] um algoritmo de satisfação de problemas de dependências distribuídas, que se baseou na estratégia de diminuição progressiva da utilidade. Isto resultou na seleção, distribuída, da solução que maximiza o valor da utilidade total dos agentes envolvidos na resolução dessas dependências.

Finalmente, em [Macedo01] também foi desenvolvida uma plataforma computacional, denominada ForEV (Formação de Empresas Virtuais), que incluiu as metodologias de negociação reportadas no trabalho em questão.

Outros trabalhos relativos a este assunto são citados nas referências bibliográficas dos trabalhos mencionadas acima.

1.4 Estrutura da Dissertação

Este trabalho está organizado em seis capítulos. Neste Capítulo 1, foram apresentados as justificativas e os objetivos do trabalho. Em seguida, no Capítulo 2, apresentam-se várias definições e conceitos relativos as Empresas Virtuais, e ainda uma breve descrição do problema de Busca e Seleção de parceiros para a formação destas empresas. Ou seja, são apresentadas algumas definições que envolvem o conceito de Empresas Virtuais e Organizações Virtuais. São apresentados também os principais conceitos de integração de empresas, modelos de empresas e justificativas para a necessidade de suporte a tolerância a falhas em Empresas Virtuais. Busca-se neste capítulo definir vários termos e apresentar os conceitos relevantes ao melhor entendimento do restante do trabalho.

No Capítulo 3 é feita uma breve apresentação dos principais elementos e conceitos referentes à arquitetura **CORBA**, que é utilizada para prover a integração de empresas e o suporte para comunicações entre Agentes. Especificações de técnicas de tolerância a falhas do **CORBA** são destacadas, bem como o uso do **FT-CORBA** em Empresas Virtuais. Busca-se introduzir uma base para a compreensão da **Plataforma FT-SALE**, que é apresentada no Capítulo 5.

No Capítulo 4 é feita uma apresentação dos sistemas baseados em Agentes. Busca-se introduzir uma base para a compreensão da **Plataforma de Busca e Seleção de Parceiros**, que é apresentada no Capítulo 5. Neste capítulo são destacadas as vantagens e desvantagens destes sistemas, bem como os problemas de interoperabilidade entre Agentes. Técnicas de tolerância a falhas em **Sistemas de Multi-Agentes Móveis** são abordadas neste capítulo.

No Capítulo 5 são apresentadas as plataformas: **Plataforma de Busca e Seleção de Parceiros** e **Plataforma FT-SALE**. Uma vez apresentadas estas plataformas, é descrita, de forma detalhada, a plataforma proposta, ou seja, a **Plataforma FT-SALE/SMAM**.

No Capítulo 6 apresentam-se as considerações finais sobre o modelo conceitual proposto, bem como sugestões para trabalhos futuros.

2. EMPRESAS VIRTUAIS

2.1 Introdução

Pode-se encontrar diferentes definições para o conceito de **Empresa Virtual**. O termo em língua inglesa "*Virtual Corporation"* aparece pela primeira vez em 1986, em [Mowshowitz86]. Em seguida surgiram várias tentativas de se difundir o conceito. *Davidow e Malone* [Davidow92] buscaram um consenso em seu livro, que mostra uma visão do desenvolvimento futuro de empresas e afirma que o tipo de cooperação "empresas virtuais" possui a maior chance de sobrevivência no atual cenário de competitividade via Internet.

Para *Davidow* e *Malone*, o termo Empresa Virtual relaciona-se primeiramente, com a arte da organização, no sentido institucional, onde: "O observador externo vê uma estrutura quase sem contornos, com linhas de divisão, constantemente mutáveis, entre empresa, fornecedor e cliente". Por outro lado, há referências onde o termo é comparado ao emprego de recursos computacionais, que de uma maneira geral significa que todas as características de um objeto estão presentes, menos o objeto em si.

Somando-se essas duas definições tem-se um objeto que possui todas as características de uma empresa, sem ser uma empresa com registro legal.

Segundo [Byrne93]:

> "Empresa Virtual é uma **rede temporária de empresas independentes** – fornecedores e clientes, mesmo sendo rivais – ligados pela tecnologia de informação para compartilhar habilidades, custos e permitir o acesso comum aos mercados dos mesmos. Não possuirá nem escritório central e nem organograma. Não terá hierarquia e nem integração vertical. Na mais pura forma do conceito, cada empresa que se liga com outras para criar uma empresa virtual será requisitada por sua essência. Contribuirão apenas aquelas que se ocuparem com suas competências essenciais".

Segundo [Mertens95], uma **EV** pode ser definida como sendo uma:

> "... cooperação entre empresas para a realização de missões, nas quais desiste-se da formação de novas instalações ou da formalização contratual através de *Joint Ventures* ou Consórcios ou mesmo da compra de novas filiais".

Segundo [Arnold95]:

> "Empresa virtual é uma forma de cooperação legal de empresas, pessoas e/ou instituições independentes que formam uma capacidade comum de entendimento sobre um negócio. As unidades cooperadoras dividem o trabalho de acordo com suas competências essenciais e atuam na geração de ofertas sobre terceiros como unidade básica. A institucionalização de órgãos centrais para o desenvolvimento, realização e gerenciamento de EV não é necessária".

Uma definição para o termo Empresa Virtual, formada da combinação entre várias definições encontradas, é apresentada em [Camarinha99]:

> "Uma empresa virtual é uma aliança temporária de empresas que se reúnem para partilhar habilidades ou recursos e suas competências principais, visando melhor responder a oportunidades de negócios, e cuja cooperação é provida por redes de computadores",

Uma **EV** corresponde a uma rede de empresas que se manifestam de três tipos, a saber:

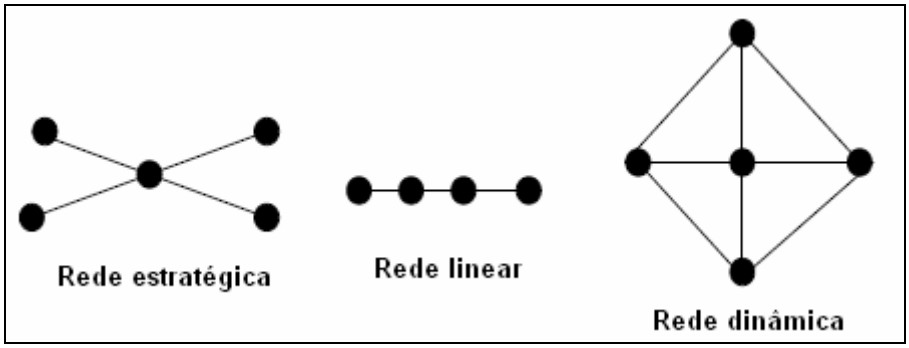

Figura 2.1: Tipos de redes de empresas.

- **Rede estratégica:** orientada para o mercado, servindo para a obtenção de certas metas. É formada com a direção de uma empresa ao centro, a qual controla todas as atividades, como por exemplo, montadoras de automóveis.

- Rede linear: adapta-se de acordo com a cadeia de valores. A operação linear desde o fornecedor de matéria-prima passando pelo produtor até o cliente é apropriada para conseguir o aumento da eficiência no processo de logística.

- Rede dinâmica: caracteriza-se por um intenso e variável relacionamento de empresas, o que é difundido pelo tipo de **cooperação EV.** Exemplos deste tipo de rede são os projetos de cooperação a nível regional, como na área da construção civil.

2.2 Definição de uma Organização Virtual

De acordo com [Toffler80], a história humana pode ser dividida em quatro grandes eras, caracterizadas pelos nômades, pela agricultura, pela indústria e agora pela informação. As civilizações correspondentes a cada período reproduzem determinadas características da forma de organização:

- Primeiro, as pessoas organizavam-se em pequenos grupos, de acordo com as habilidades, tal como a caça;
- A hierarquia cresceu com o advento da agricultura;
- A era industrial trouxe o nascimento da burocracia e
- A era da informação traz as redes.

Uma rede é um tipo de organização que se origina como um projeto básico para a construção de um grupo social. No contexto de empresas, redes significam cooperações estáveis ou dinâmicas, com a finalidade de explorar oportunidades de mercado. A formação de grupos dentro destas redes possui características semelhantes aos antigos grupos nômades, onde determinadas especialidades (competências) para a caça precisavam ser agrupadas.

Desta maneira, uma rede baseada em empresas, apoiada pela informação e baseada em competências, transforma-se nos grupos de caça da era da informação. Neste sentido, uma **Organização Virtual** (**OV**) pode ser definida como "uma rede estável de empresas, destinada a formação de **EVs**, interligadas de acordo com suas competências essenciais e estratégias de mercado e suportada pela utilização da tecnologia de informação" [Netage96].

Em outras palavras, uma Organização Virtual pode ser vista como uma plataforma estável, onde empresas trocam informações a respeito de oportunidades de mercado e conseqüentemente utilizam-se dessa cooperação para a formação rápida de uma **EV**, uma vez que uma estrutura organizacional já está presente.

A Figura 2.2 ilustra o cenário de uma **OV** da qual faz parte um grupo de 11 empresas (**E1, E2, E3, E4, E5, E6, E7, E8, E9, E10** e **E11**), sendo que as empresas **E3, E6, E7, E8** e **E11** constituem uma Empresa Virtual **(EV-1)**.

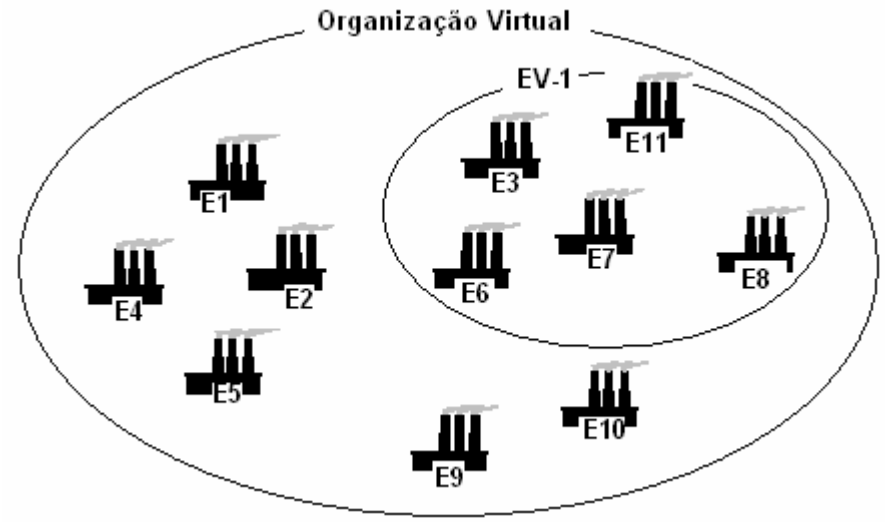

Figura 2.2: Formação de Empresas Virtuais a partir de uma Organização Virtual.

Um modelo para a construção desta plataforma (OV) pode ser observado no projeto *Virtuelle Fabrik* da Universidade de *St. Gallen* – Suíça, disponível em [Katzy97a]. Este modelo contém quatro fases principais para a formação de rede estável de cooperação:

1. **Esclarecimento:** corresponde a interligação de um time básico (o técnico da rede, mais cerca de 10 empresas) e a definição dos limites da área de atuação;

2. **Planejamento:** refere-se ao planejamento de projetos e financiamento;

3. **Construção:** fundação da rede como uma organização, juntamente com a definição de papéis e regras para o processo de coordenação. Nesta fase existe ainda a transferência de *know-how* para a difusão das competências existentes na rede.

4. **Operação:** prolongamento da rede através da admissão de novos parceiros e início de aquisição e desenvolvimento de ordens de pedido.

A iniciativa para a formação de uma rede estável pode ter diferentes origens, dentre as quais ressaltam-se os projetos de pesquisas financiados por instituições governamentais.

Na prática, a formação destas plataformas pode surgir através da expansão da cadeia tradicional de fornecedores a nível regional, mas, no entanto, apresenta uma velocidade menor devido à construção das camadas de infra-estruturas [Griese92].

2.2.1 Ciclo de Vida da Empresa Virtual dentro de uma Organização Virtual

Um modelo de ciclo de vida foi proposto pelo "*Agile Manufacturing Enterprise Forum*" (**AMEF**), no qual foram estabelecidas as fases desde a criação até a dissolução de uma EV [Goranson95]. Cada fase envolve decisões que são suportadas por diferentes métodos e ferramentas, e por um sistema de métricas. No entanto, esta proposta pode ser considerada genérica, servindo como modelo para a formação de EVs independentemente se as empresas estão em uma organização ou não. A Figura 2.3 ilustra as 4 fases do ciclo de vida de uma Empresa Virtual.

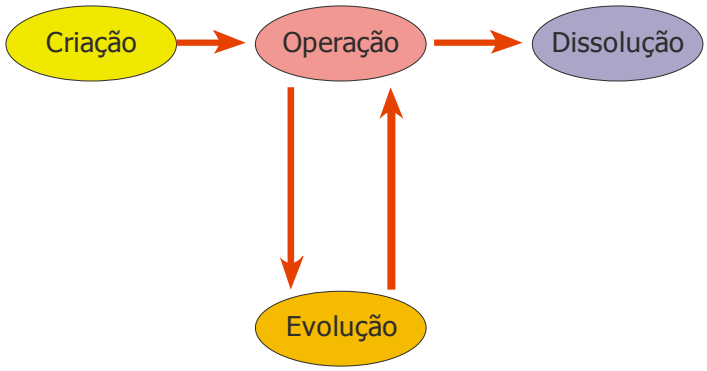

Figura 2.3: Fases do ciclo de vida de uma Empresa Virtual.

Fase de Criação – Fase inicial quando a EV é criada/configurada. Algumas funcionalidades previstas para esta fase são:

- Busca e seleção de parceiros,
- Negociação do contrato,
- Definição dos direitos de acesso e nível de compartilhamento,
- Definição dos procedimentos de cancelamento de contrato, configurações, etc...

Fase de Operação – Fase na qual a EV está realizando o processo de negócios de forma a atingir o objetivo comum. Requer funcionalidades como:

- Mecanismo básico de troca segura de dados,
- Compartilhamento de informações,
- Gerenciamento de pedidos,
- Processo de pedidos incompletos,
- Planejamento de programação distribuída e dinâmica,
- Gerenciamento das tarefas distribuídas, etc...

Fase de Operação – Evoluções são necessárias durante a operação de uma EV, como, por exemplo, quando é necessário adicionar e/ou substituir um parceiro. Isto deve acontecer durante eventos excepcionais, como incapacidade do parceiro, necessidade de crescimento das propostas, etc. Funcionalidades similares às especificadas na criação são necessárias nesta fase.

Fase de Dissolução – Fase na qual a EV finaliza seu processo de negócio e se desmancha. Duas situações podem ser a causa da dissolução. Uma delas é o sucesso de todos os objetivos alcançados, a outra, está na decisão dos parceiros em parar a operação da EV.

2.2.2 Integração de Empresa

Na década de 70, grandes empresas fizeram investimentos vultosos na implantação de sistemas computacionais que de um lado resultaram em um aumento do desempenho, mas por outro apresentaram enormes dificuldades para modificação e integração dos mesmos. Isto provocou uma decepção na relação custo/benefício de implantação de tais sistemas [Scheer95].

Desta maneira, surgiu então o conceito de "**Arquitetura**", com a qual uma empresa poderia representar o seu funcionamento sobre diferentes perspectivas, representadas por recursos humanos, estrutura organizacional, informação e processos de manufatura. Nesta época, as metodologias ainda enfatizavam a utilização de recursos da **Tecnologia da Informação** (**TI**), o que mais uma vez decepcionou as empresas que utilizaram o conceito de Arquitetura sobre esta perspectiva [Rozenfeld96], uma vez que continuava a dificuldade de integração de sistemas.

Um conceito que surgiu na década de 80 e vem sendo pesquisado até os dias de hoje é o "*Computer Integrated Manufacturing*" (**CIM**). Como *Rozenfeld* mostra, este conceito muda o foco de atenção para se ajustar às necessidades das empresas, em busca de melhor desempenho [Rozenfeld96]. Na sigla, o foco saiu da letra **"C"** para letra **"I"**, ou seja, a utilização do computador, da Tecnologia da Informação, passou a ser uma das visões, na qual uma empresa é observada. O termo **Integração** assumiu assim o papel central para a melhora de desempenho de uma empresa. A Figura 2.4, extraída de [Rentes96], mostra a evolução da sigla.

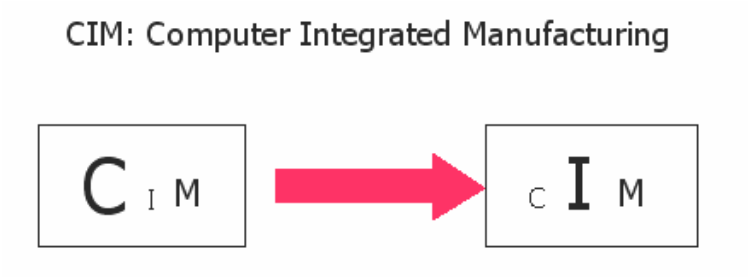

Figura 2.4: Evolução da sigla CIM.

Segundo [Rentes96], a visão holística de uma empresa equivale a se ter uma "imagem única", sintética, de todos os elementos da empresa, tais como estratégias, atividades, informações, recursos e organização, assim como suas inter-relações.

Ressaltada a importância da utilização dos conceitos sobre **integração de empresas** (**IE**), é necessário apresentar os mecanismos que viabilizam os objetivos difundidos por esta área do conhecimento. Entre os principais mecanismos podem ser citados [Williams95]:

- A utilização de uma arquitetura de referência, para a realização da modelagem da situação atual das empresas;
- A utilização de uma ferramenta computacional, para o armazenamento eletrônico das informações levantadas nas empresas;
- A utilização de uma metodologia, para guiar o processo de otimização da empresa.

2.2.3 Metodologia de Integração de Empresas

De acordo com as novas tendências, a Metodologia de Integração de Empresas (**MIE**) baseia-se no conceito de *Change Management* [Rentes96]. Segundo este conceito, as mudanças que ocorrem em uma empresa, acontecem de maneira circular, onde a definição de novos objetivos e metas, juntamente com a proposição de novos processos é sempre reinicializada ao final de cada implementação de um projeto ou sempre que se julgar necessário. O principal objetivo da **MIE** consiste na formalização de uma arquitetura para sistematizar, organizar e integrar vários métodos relacionados.

A primeira é a dimensão **Metodologia de Intervenção**, que envolve passos a serem seguidos no processo de integração da empresa, com a sugestão de uma seqüência a ser implementada. Nesta dimensão é proposta uma lógica de condução para o projeto de integração, assim como um formalismo para o levantamento das informações.

A segunda é a dimensão de **Processos Empresariais** ou de **Processos de Negócios** que focaliza diferentes tipos de empresas, orientando a diferenciação dos métodos de intervenção para cada tipo. De acordo com o tipo da empresa a ser integrada, as classes de Processos de Negócios devem ser identificadas, assim como, as filosofias e conceitos correspondentes para serem utilizados no processos de gerência da mudança (do *Change Management*).

A última dimensão é a de **Suporte Metodológico** que oferece, como o próprio nome diz, suporte metodológico, ferramental e de conhecimento para a aplicação dos métodos de integração. Esta dimensão contém instrumentos para o estabelecimento da **Visão de Processos de Negócios** e para o processo de condução dos métodos de intervenção.

A dimensão **Metodologia de Intervenção** tem um papel central na aplicação da **MIE**, pois ela indica o processo de integração da empresa. Isto ocorre em três etapas, onde cada uma é composta por várias fases específicas:

- A etapa inicial, denominada **Integração de Objetivos**, tem o objetivo de obter uma visão genérica da empresa, através da especificação de suas estratégias, seus fatores críticos de sucesso e a sua forma atual de operação. Baseados nestas informações, os projetos de modernização, que melhor se ajustam à realidade (ou requisitos) da empresa, são definidos. Esta etapa é dividida em três fases:
    - Diagnóstico da Situação Atual,
    - Definição de Estratégias Organizacionais e
    - Definição das Ações de Integração. Cada fase pode ser vista como uma classe de técnicas de intervenção que pode ser encontrada na literatura.
- A segunda etapa, **Integração de Processos,** envolve a realização de um plano de integração funcional da empresa. De acordo com as Ações de Integração, definidas junto com a empresa, os projetos que devem ser adotados são identificados. O conjunto de atividades nesta etapa é normalmente muito complexo, nem sempre sendo possível a aplicação de uma forma global. Assim, de acordo com os objetivos e requisitos obtidos na primeira etapa da metodologia, apenas algumas fases devem ser focalizadas, orientando os esforços da empresa para ações que possibilitam maiores ganhos. As fases para esta etapa podem ser divididas em:

- Desenvolvimento/Reengenharia, Gerenciamento de Custos, Seleção ou Desenvolvimento de Soluções, Seleção de Tecnologias.

- Na última etapa, denominada **Operacionalização**, os projetos de modernização são detalhados, implementados e mantidos. A migração para as novas formas de operação é suportada pelas informações que foram obtidas durante a primeira etapa. As fases desta etapa são:

    - Implantação de Processos, a Implantação de Sistemas e a Melhoria Contínua.

2.3 Processo de Busca e Seleção de Parceiros para a Formação Empresas Virtuais

Como visto anteriormente, uma empresa virtual pode ser definida como uma aliança temporária de empresas que, apoiadas por redes de computadores, se reúnem de forma a compartilhar recursos e suas competências principais, com a finalidade de atender a oportunidades de negócios.

Entretanto, esta união de empresas deve ser feita de forma confiável e ágil, para poder atender a dinâmica imposta no mercado atual. As competências principais das empresas que formarem a parceria devem estar relacionadas à modalidade de serviço exigida em uma nova **Oportunidade de Negócio** (**ON**), não somente em termos de recursos tecnológicos ou de habilidades e serviços, mas também em termos de prazos e custos necessários, para que a mesma possa concluir sua parte da tarefa. Neste caso pode-se excluir a participação de empresas que não se encaixam nestas condições.

O processo de formação de uma Empresa Virtual é iniciado no momento em que acontece uma solicitação do Cliente para a fabricação de um novo produto ou a realização de um serviço. Esta é considerada uma Oportunidade de Negócio, a qual deve ser atendida em um prazo determinado pelo Cliente.

Dada uma nova Oportunidade de Negócio, deve-se dar inicio á primeira fase da Empresa Virtual, ou seja, a fase de Criação. O processo de **Busca e Seleção** de parceiros é realizado durante a fase de criação da Empresa Virtual, visto que é necessário encontrar aqueles que serão os responsáveis por atender esta oportunidade. Outro momento que as atividades de busca e

seleção são necessárias é quando a **EV** já está na sua fase de operação e surge a necessidade de substituição de um ou mais membros [Schmidt03].

As principais etapas do processo de Busca e Seleção de parceiros são descritas de forma detalhada em [Schmidt03]. Algumas considerações sobre este processo, importantes para uma melhor compreensão do modelo proposto neste trabalho, são citadas no Capítulo 5.

2.4 Necessidade de Tolerância a Falhas em Sistemas de Empresas Virtuais

A parceria entre empresas implica que os parceiros devem empregar mecanismos de comunicação para a troca de informações e manutenção do sincronismo de suas produções distribuídas, o que torna estas empresas cada vez mais dependentes de seus sistemas computacionais. Contudo, surge uma grande preocupação em torno da confiabilidade nestes sistemas, uma vez que falhas podem ocorrer.

Em [Cummins02] são sugeridas três técnicas básicas para melhorar a confiabilidade do sistema e ainda alguns mecanismos básicos para minimizar os riscos de falhas:

1) Técnicas para melhorar a confiabilidade do sistema:

Minimizar o risco de o sistema falhar;

Detectar um mau funcionamento logo de início;

Limitar o impacto de falha.

2) Mecanismos básicos para minimizar os riscos de falhas:

Ambiente seguro com uma fonte de energia confiável;

Testes dos equipamentos e *software* antes da operação de produção;

3) Controle de alterações.

Em ambiente de Empresas Virtuais, os processos estão distribuídos pela rede. Uma interrupção nas aplicações destas empresas, provocada por falhas de processos ou de canais de

comunicação, pode causar sérios prejuízos. Estas falhas devem ser devidamente tratadas. Desta forma, as aplicações de uma empresa precisam dispor de técnicas de tolerância a falhas, as quais tornarão possível a continuação de um serviço, mesmo na presença de falhas [Batalha01].

Soluções para falhas em ambiente de empresas virtuais são vistos com maiores detalhes no Capítulo 5.

2.5 Modelos de Empresas

Na literatura podem ser encontradas diferentes definições para o termo **Modelo de Empresa**. Segundo [Kosanke92], o modelo de empresa deve refletir a realidade o melhor possível e ser utilizado para controlar e monitorar suas operações. Essas operações devem ser descritas em termos de sua funcionalidade e comportamento dinâmico. O modelo deve ser constituído por sub-modelos, que representam diferentes aspectos da empresa e, as mudanças ocorridas ao nível de sub-modelos devem se refletir no modelo total. O modelo também deve apresentar diferentes níveis de abstração, pois assim é possível utilizar modelos para suporte nas decisões estratégicas, táticas e operacionais.

[Rozenfeld96] define o modelo de empresa como sendo uma ferramenta que viabiliza e suporta diversas atividades de implantação de novas tecnologias, filosofias e métodos na manufatura. Ele deve servir de suporte para:

- Entendimento do complexo funcionamento de uma empresa;
- Suporte para a integração de objetivos e estabelecimento de estratégias;
- Padronização de nomenclatura e procedimentos;
- Suporte para resolução de problemas existentes, através de uma visão objetiva do sistema;
- Suporte para integração lógica;
- Suporte para o desenvolvimento e funcionamento de um gerenciador de fluxo de informações.

Tais sistemas podem ser implementados de diferentes maneiras, usando uma larga variedade de recursos da **TI** e de infra-estrutura de comunicação. Além disso, os sistemas podem operar em ambientes que vão desde pequenas redes até grandes redes.

Normalmente os integrantes de uma **OV** são empresas independentes, distribuídas dentro de uma região e que possuem diferentes infra-estruturas de **TI**, o que deve ser levado em conta por qualquer solução pretendida para o suporte à Empresas Virtuais. Portanto, padronizar a infra-estrutura de **TI** das empresas não é uma alternativa viável, pois requer o abandono de soluções já implantadas e introduz alterações significativas nos processos das empresas, o que por vezes é inviável de se fazer [Schmidt03].

Na literatura e na prática existem diferentes arquiteturas sendo utilizadas para prover a integração entre sistemas heterogêneos, como os das empresas que constituem uma **OV**. Das diversas possibilidades existentes, neste trabalho opta-se pela plataforma **CORBA**, cujos principais fundamentos são apresentados no capítulo seguinte. A plataforma **CORBA** é utilizada na arquitetura **FT-SALE**, base para a arquitetura proposta neste trabalho. Exemplos de utilização da plataforma CORBA são encontrados na referências [Batalha01], [Fraga01b], [Filho00], [Lima01a], [Lung01] e [Mendonça02].

3. A ARQUITETURA CORBA

3.1 Introdução

O acelerado crescimento das redes de computadores trouxe consigo o aumento do uso de sistemas distribuídos e heterogêneos em diversas áreas da computação.

Desenvolver *software* para um sistema distribuído envolve níveis elevados de esforço, e desenvolver *software* para sistemas distribuídos heterogêneos às vezes chega a ser quase impossível. Projetistas de sistemas distribuídos reais precisam lidar com a heterogeneidade.

Um *software* voltado para aplicações distribuídas heterogêneas deve lidar com quase todos os problemas normalmente encontrados na programação de sistemas distribuídos, tais como: falha de algum dos sistemas na rede; o particionamento da rede; problemas ligados ao controle de uso e ao compartilhamento de recursos na rede; e também com a segurança.

Uma das características destes sistemas é a questão da possibilidade de migração de uma aplicação em rede para uso em outra plataforma. Esta migração irá resultar em duas ou mais versões da mesma aplicação na rede. Se forem feitas alterações em qualquer uma das duas versões, a outra versão deve também ser alterada apropriadamente, e depois as duas devem ser testadas individualmente para que se possa assegurar que elas estão funcionando adequadamente.

O nível de dificuldade que surge com esta situação aumenta significativamente quando aumenta o número de diferentes plataformas na composição da rede. Observa-se que a heterogeneidade, neste contexto, não se refere tão somente ao *hardware* de computação e aos sistemas operacionais.

Desenvolvedores de aplicações distribuídas costumam fazer uso intensivo de ferramentas e bibliotecas disponíveis. Isso implica dizer que os sistemas distribuídos são implicitamente heterogêneos, e geralmente compostos por várias aplicações em diferentes camadas e por várias bibliotecas. Infelizmente, em muitos casos, à medida que o sistema distribuído cresce, as chances de que todas as aplicações e bibliotecas que o compõem tenham sido especificamente projetadas para trabalhar em conjunto, diminuem drasticamente.

De maneira geral, duas regras devem ser observadas no projeto de aplicações para sistemas distribuídos heterogêneos: empregar modelos e abstrações independentes de plataforma; e esconder ao máximo as complexidades de baixo nível sem que se sacrifique demais o nível de desempenho.

A utilização de abstrações e modelos corretos pode, essencialmente, fornecer uma nova camada de desenvolvimento de aplicações que seja homogênea, contribuindo com a complexidade causada pela heterogeneidade. Tal camada irá ocultar detalhes de baixo nível e permitirá que os desenvolvedores solucionem problemas mais urgentes, sem necessitar tratar de imediato de detalhes de baixo nível de rede para todas as diferentes plataformas usadas por suas aplicações.

Para solucionar os vários problemas causados pela heterogeneidade dos sistemas distribuídos, fez-se necessária a criação de tecnologias que provessem o desenvolvimento e a integração das diversas aplicações existentes [Mendonça02].

Entre as diversas tecnologias existentes no mercado, para este fim, pode-se citar:

- *Distributed Component Object Model* – **DCOM** da *Microsoft*;
- *Distributed System Object Model* – **DSOM** da IBM;
- *Remote Method Invocation* – **RMI** da Sun;
- *Common Object Request Broker Architecture* **– CORBA** da OMG.

Estas tecnologias são conhecidas como *Midlleware*[1]. Entretanto, das soluções apresentadas acima, o **CORBA** da **OMG** tem se tornado padrão para a maioria das aplicações. Por esta razão, o **CORBA** foi considerado como foco principal deste trabalho.

[1] Partes dos *softwares* responsáveis pela intercomunicação entre os vários objetos distribuídos em uma rede.

3.2 Grupo de Gerenciamento de Objetos – OMG

A **OMG** (*Object Management Group*) é uma fundação, sem fins lucrativos, formada no final dos anos 80, pela união de várias companhias interessadas em desenvolver um ambiente com a capacidade de manipular a heterogeneidade e a distribuição das aplicações modernas.

Em 1990, a **OMG** criou o **OMA** (*Object Management Architecture*) e o seu núcleo (a especificação **CORBA**), com o objetivo de fomentar o crescimento de tecnologias baseadas em objetos e fornecer uma **infra-estrutura conceitual** para todas especificações **OMG** [Montez97]. O **CORBA** fornece uma **infra-estrutura arquitetural** de suporte ao desenvolvimento, voltada para aplicações distribuídas heterogêneas, orientadas a objetos.

A estrutura da **OMA** é baseada essencialmente em dois elementos: o **Modelo de Objeto**, que define o **objeto** que será distribuído pelo sistema heterogêneo, e o **Modelo de Referência**, que define as características de integração destes objetos.

Segundo [Montez97], os quatro elementos principais que compõem a **OMA** podem ser definidos da seguinte forma:

1. **ORB** (*Object Request Broker*): Conhecido comercialmente como **CORBA**, o **ORB** é o elemento principal do modelo. Pode ser considerado como elemento base para a construção de aplicações utilizando objetos distribuídos que possuem características de interoperabilidade entre aplicações em ambientes heterogêneos ou homogêneos. Assim sendo, o **CORBA** é uma especificação de uma arquitetura que permite as aplicações fazerem solicitações a objetos, de uma forma transparente e independente, sem levar em conta a linguagem, o sistema operacional e/ou a localização. O **ORB** fornece ainda uma interface que pode ser usada diretamente tanto pelos clientes, quanto pelos objetos.

2. **Serviços de Objetos** (*Object Services*): São serviços que contêm objetos distribuídos. Em outras palavras, é uma coleção de serviços (interfaces e objetos) que suportam funções básicas para usar e implementar objetos. Como exemplos destes serviços, tem-se: os servidores de nomes (permitem descobrir objetos

distribuídos através do nome), serviço de transações (permitem descobrir objetos distribuídos através de propriedades), serviços para gerenciamento de ciclo de vida de objetos remotos, segurança, transações, notificação de eventos, entre outros.

3. **Facilidades Comuns** (*Common Facilities*): é uma coleção de serviços que muitas aplicações podem compartilhar, mas que não são tão fundamentais como os serviços de objetos. São dedicadas a usuários finais de aplicações e podem ser divididas em duas categorias: i) **facilidades horizontais:** serviços de e-mail, impressão, interface de usuário, etc.; ii) **facilidades verticais**: facilidades específicas de um domínio de aplicação.

4. **Objetos de Aplicação** (*Application Objects*): são os objetos que podem ser considerados visíveis ao nível de aplicação. Correspondem à noção tradicional de aplicações de usuários que, por esse motivo, não são padronizados pelo **OMG**.

A Figura 3.1, extraída de [Mendonça02], ilustra as categorias de interfaces que estão conceitualmente ligadas por meio de um **ORB**, utilizando suas facilidades de ativação e comunicação: **serviços de objetos**, **interfaces de domínios** e **interfaces de aplicações**.

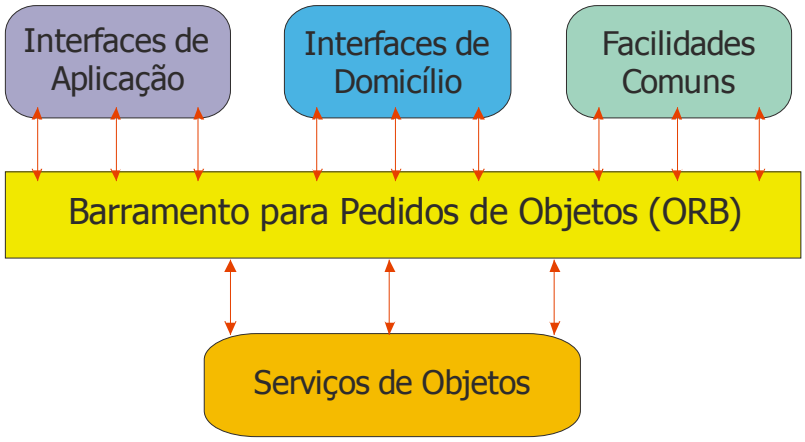

Figura 3.1: Arquitetura OMA.

Segundo [Mendonça02], **interfaces de domínio** "desempenham uma função similar àquelas da categoria de Serviços de Objetos, exceto pelo fato de que as Interfaces de Domínios são específicas para um determinado domínio, ou seja, são orientadas verticalmente, ou seja, este componente prove uma série de interfaces mais dirigidas para indústria e mercados verticais que

possuem um domínio próprio de operações específicas, tais como finanças, manufatura, comércio eletrônico e outras".

3.3 Fundamentos, Terminologias e Conceitos Básicos

Para uma melhor compreensão da arquitetura **CORBA** é necessária a descrição de alguns fundamentos, conceitos e terminologias de objetos usados por esta arquitetura.

Dentre os conceitos básicos, um dos principais é o de **sistema de objetos** que é composto por entidades denominadas **objetos**, as quais possuem a função de fornecer serviços aos **clientes**. Um **cliente de um serviço**, por sua vez, é qualquer entidade capaz de requisitar serviços através de eventos denominados **requisições**. Uma **requisição** possui basicamente as seguintes informações associadas: **operação**, **objeto destino**, **parâmetros** e **contexto da requisição** [Montez97].

Em [Teixeira00] é apresentada uma definição para os termos mais significativos da terminologia da plataforma **CORBA**, os quais são reproduzidos a seguir:

- **Objeto CORBA:** representa uma entidade virtual, exceto quando caracterizada através de uma implementação escrita em uma linguagem de programação. Quando **objetos clientes** requisitam serviços às implementações de **objetos CORBA**, o fazem através do **ORB**, o qual pode localizar esta entidade e entregar pedidos de clientes dirigidos para ela. O **objeto CORBA** para quem o pedido é direcionado é chamado de **objeto Alvo**, o qual é determinado exclusivamente através da referência ao objeto utilizado para que se possa emitir o pedido. Objetos **CORBA** podem ser criados e destruídos através de determinadas requisições. O cliente toma conhecimento da criação de um objeto através de uma **referência de objeto**, que denota o novo objeto.

- **Referência a objeto**: Uma referência a objeto diz respeito somente a um objeto **CORBA**, e representa a forma adequada para identificar, localizar e encaminhar pedidos para este objeto **CORBA**. Uma referência a objeto é utilizada por um cliente para direcionar pedidos para objetos. Porém, eles não

podem criar referências a objetos, nem podem acessar ou modificar o conteúdo de uma referência a objeto.

- **Pedido:** representa a chamada de uma operação por um cliente em um objeto **CORBA.** Os pedidos fluem de um cliente para um objeto alvo em um servidor, e o objeto alvo envia os resultados de volta como resposta, caso o pedido exija uma.

- **Servente** (*Servant*): Objetos **CORBA** são "encarnados" por entidades da linguagem de programação na qual o servidor foi escrito. Ao longo de sua vida, um objeto **CORBA** pode ser representado por diferentes serventes. Em C++ ou Java, um servente é uma instância de objetos de uma determinada classe.

- **Objeto Cliente**: representa uma entidade que emite um pedido para um objeto **CORBA**. Um **cliente** pode estar localizado ou não no mesmo espaço de endereçamento do **objeto CORBA,** ou então o **cliente** e o **objeto CORBA** podem existir na mesma aplicação. Devido a propriedade de **transparência de localização**, um **cliente** pode chamar métodos de objetos **remotos** do mesmo modo que chamam métodos de objetos locais, sendo necessário para isso ter uma **referência para o objeto.** Nas Figuras 3.2 e 3.3, é possível visualizar de forma mais clara o conceito de **transparência de localização**.

- **Servidor** representa uma aplicação na qual existem um ou mais objetos **CORBA**.

Vale lembrar que os termos **cliente** e **servidor**, tal qual apresentado anteriormente, são significativos somente no contexto de um pedido em particular, pois a aplicação que é cliente para um pedido pode também ser o servidor para um outro pedido qualquer, e vice versa.

3.3.1 A Linguagem de Definição de Interfaces

Para que se possa emitir chamadas para operações em um objeto distribuído, o cliente precisa conhecer a interface que é oferecida pelo objeto. Para descrever interfaces, ou seja, especificar um "contrato" entre os objetos clientes e servidores, padronizando suas ofertas e usos

de serviços, o **CORBA** utiliza a Linguagem de Definição de Interfaces (em inglês, ***Interface Definition Language*** ou **IDL**), a qual é puramente declarativa[2] baseada em C++, garantindo desta forma a interação entre objetos distintos de linguagem, sistema operacional e localização. Uma interface de um objeto define as operações que ele suporta e o tipo de dados que podem ser passados de/para essas operações.

3.3.2 Mapeamentos de Linguagens

Mapeamentos de linguagens especificam como a **IDL** pode ser traduzida para diferentes linguagens de programação, definindo que facilidades da linguagem de programação precisam ser empregadas para que se possa viabilizar a construção **IDL** para as aplicações [Teixeira00].

Atualmente uma interface definida em **IDL** está associada a seu mapeamento específico, descrito no **Modelo de Objetos CORBA**, para as linguagens: C, C++, Java, *Smalltalk,* ADA e COBOL [Filho00].

Um compilador específico converte os tipos e interfaces definidas em **IDL** para uma determinada linguagem de programação, gerando um conjunto de objetos chamados ***stubs*** e um conjunto de interfaces que os servidores devem implementar chamados ***skeletons***.

A Figura 3.2, extraída de [Montez97], ilustra a independência de linguagem de programação entre os objetos.7

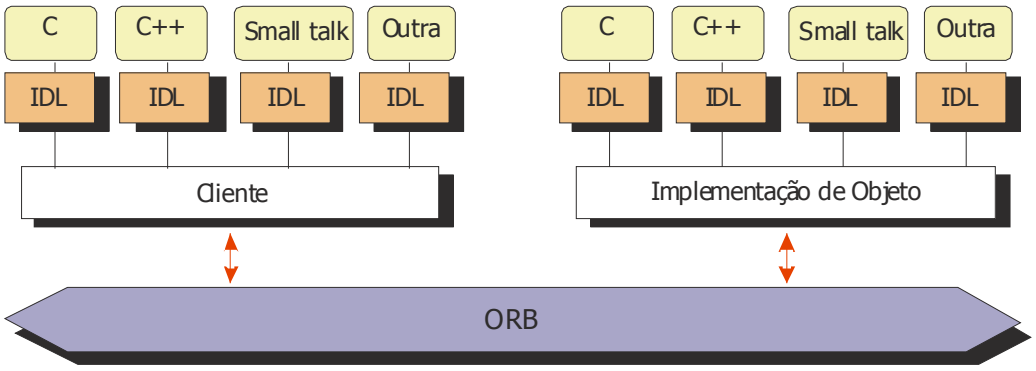

Figura 3.2: IDL provê independência de linguagem de programação entre os objetos.

[2] A **IDL** não é uma linguagem de programação, de modo que, evidentemente, os objetos e as aplicações não podem ser implementados em **IDL** [Teixeira00].

3.3.3 *Stubs* e *Skeletons*

Os ***stubs*** e os ***skeletons*** implementam a arquitetura Cliente/Servidor no **CORBA** e possuem funções similares**,** porém o ***stub*** se encontra na extremidade do Cliente e o ***skeleton***, na extremidade do Objeto [Souza98].

A Figura 3.3, extraída de [Filho00], ilustra o processo de geração de ***Stubs*** e ***Skeletons,*** a partir da interface **A** usando a ***Interface Definition Language*** (**IDL**).

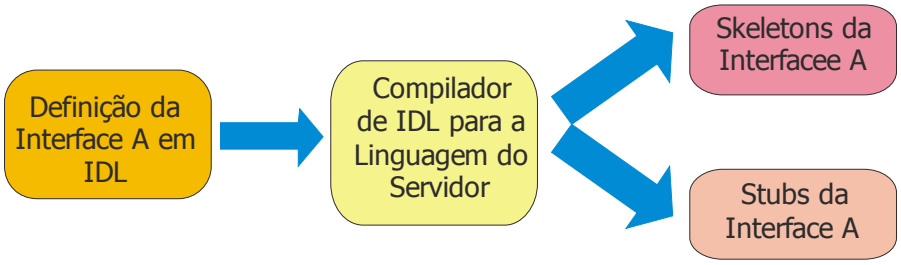

Figura 3.3: Processo de geração de *Stubs* e *Skeletons* a partir de uma interface IDL.

O ***stub*** e o ***skeleton*** são criados a partir da compilação da interface **IDL** do Cliente e do Objeto, respectivamente. Quando um Cliente emite uma solicitação, ele só precisa dizer ao ***stub*** para qual Objeto ela é destinada, não devendo importar se o Objeto está situado a nível local ou não, garantindo desta forma a transparência nos sistemas distribuídos [Souza98].

Os ***stubs*** são também conhecidos como **proxies** por possuírem a capacidade de **localizar** o objeto destino e, junto com o **ORB**, empacotar a mensagem, possibilitando seu envio pela rede, caso necessário. Os ***skeletons***, por sua vez, recebem solicitações do **ORB**, e têm a função de, a partir desta solicitação, desempacotar a mensagem e entregá-la à implementação do Objeto, podendo ainda fazer o processo inverso, caso haja resposta.

3.3.4 Tipos de Invocação de Objetos

Existem duas formas de invocação de objetos em CORBA, são elas:

- **Invocação Estática**: mensagens são construídas na aplicação Cliente e na implementação do Objeto. O ***stub*** transforma a linguagem de programação que é passada da aplicação do Cliente, para um formato que possa ser trafegado pela rede, e o ***skeleton*** faz o inverso, ou seja, transforma o que recebe para a

linguagem de programação da implementação do Objeto. Para que isso seja possível, **stubs** e **skeletons**, precisam ter conhecimento completo das interfaces **OMG IDL** dos Objetos **CORBA** que são invocados. Nesta invocação, para que novos Objetos possam ser acessados, é necessário parar o sistema, ou realizar nova compilação, o que a torna inflexível.

- **Invocação Dinâmica**: busca solucionar o problema da inflexibilidade da invocação estática, a qual não permite que Objetos criados após a compilação do ***stub*** ou ***skeleton***, sejam acessados. Para esta invocação existem duas interfaces suportadas pelo **CORBA**: (i) A **DII** (Interface de Invocação Dinâmica), correspondente ao ***stub*** e a (ii) **DSI** (Interface *Skeleton* Dinâmica), correspondente ao ***skeleton***. Outra característica importante da invocação dinâmica é a transparência, pois a implementação nunca sabe se a sua requisição está sendo feita a um ***stub*** estático ou dinâmico.

3.4 Arquitetura de um Sistema CORBA

Uma vez que as Empresas Virtuais se encaixam no perfil de redes de larga escala e heterogêneas, a busca de uma infra-estrutura de integração para os sistemas distribuídos que forma a EV é um passo fundamental. A arquitetura CORBA provê um ambiente de suporte ao desenvolvimento de novos sistemas e/ou um ambiente para a integração de aplicações, no qual novos sistemas podem ser construídos através da composição de funcionalidades prontas em sistemas (ou subsistemas) já existentes [Teixeira00].

Uma implementação do padrão CORBA é conhecida como um ORB, que é um intermediário que encaminha pedidos dos clientes aos objetos através da rede, de uma forma transparente e independente, sem levar em conta a diferença de linguagens de programação, a diferença entre os sistemas operacionais e a localização. Cada implementação deve poder se comunicar com todas as outras através de um protocolo entre ORBs. O **IIOP** (*Internet Inter-ORB Protocol*) é o protocolo especificado para redes baseadas nos protocolos **TCP/IP**, sendo obrigatório para as implementações CORBA [Filho00].

O protocolo IIOP emprega um formato de mensagem chamado **GIOP** (*General Inter-ORB Protocol*), que define um conjunto de primitivas de comunicação, independentes das camadas de transporte e de rede, e que permite a comunicação entre ORBs.

Em [Montez97], o relacionamento entre o GIOP e o IIOP é comparado ao mapeamento existente entre a IDL e uma linguagem de programação específica. Neste caso, o GIOP pode ser mapeado em diferentes protocolos de transportes, e específica os elementos de protocolo que são comuns a todos os mapeamentos. Contudo, o GIOP não fornece uma completa interoperabilidade, da mesma forma que a IDL não pode ser usada para se construir programas completos.

Em CORBA considera-se que os objetos existam dentro de servidores (implementação), sendo cada objeto associado a um único servidor. Porém, alguns servidores podem ter mais do que um único processo associado a eles. No caso do servidor de um determinado objeto estar "parado" quando é feita uma chamada para este objeto, o CORBA cuida de ativar automaticamente o servidor, dando inicio a um processo ou a uma *threads*[3], ou ainda a carga de uma **Biblioteca de Ligação Dinâmica** (do inglês, *Dynamic Link Libraries*. ou DLL), dependendo do tipo de sistema operacional no qual o servidor estiver registrado [Teixeira00].

O código para um servidor inclui: (i) o código que implementa os objetos que ele contém (seus tipos), (ii) uma função principal (main (), em C++) que inicializa o servidor e, normalmente, cria um conjunto inicial de objetos. Os objetos em um servidor podem ser todos do mesmo tipo, ou de diferentes tipos ou ainda objetos não-CORBA, sem qualquer dificuldade e sem a exigência de se empregar código especial. Na Figura 3.4 a seguir, extraída de [Reverbel06], é possível se visualizar a chamada remota a um objeto CORBA do servidor da Máquina **Y,** feita pelo cliente da Máquina **X.** Observa-se ainda, que um servidor não pode ser invocado pelos clientes, somente seus objetos CORBA podem.

---

[3] ***Threads*** (**Linhas de execução**) fornecidas pelo próprio Sistema Operacional (conhecidas como KLT, pois são em nível de *Kernel*) ou implementadas por bibliotecas de uma determinada linguagem (conhecidas como ULT, pois são em nível de usuário). Um ***thread*** é uma maneira de um processo dividir a si mesmo em duas ou mais **linhas** de tarefas simultâneas. Um uso comum dado aos *threads* é, por exemplo, ter uma *thread* para prestar atenção a um ambiente, enquanto outras *threads* fazem outros cálculos [Wikipedia - http://pt.wikipedia.org/wiki/Thread].

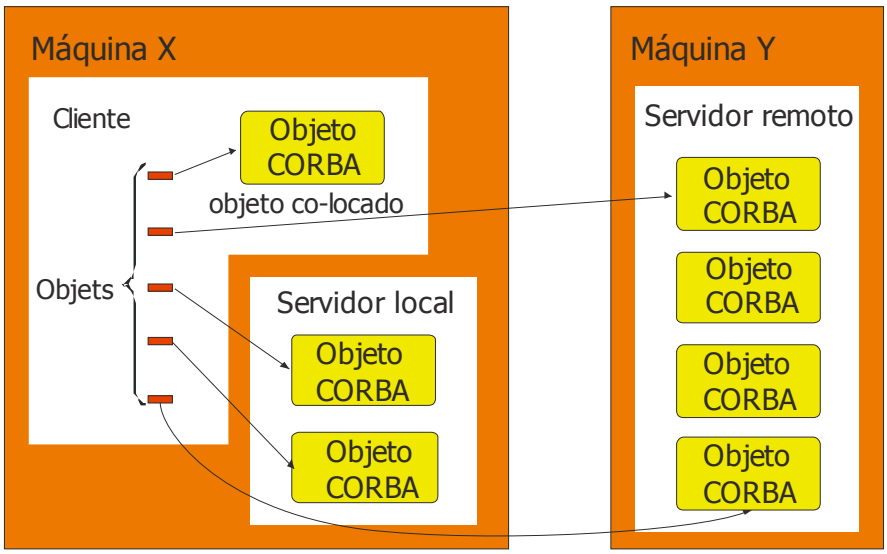

Figura 3.4: Chamada de acesso remoto – transparência de localização.

O cliente simplesmente chama um método sobre uma referência a objeto. O ORB se encarrega de mandar uma mensagem de requisição, esperar a mensagem de resposta e retornar os resultados para o cliente. Desta forma, para o cliente o procedimento parece uma chamada de método normal [Reverbel06].

Normalmente, o termo "remoto" é empregado quando a chamada é direcionada para outro servidor, mesmo que este estiver na mesma máquina onde está o "chamador" (ciente ou outro servidor). O nível de transparência é preservado pelo emprego da mesma sintaxe empregada nas chamadas a objetos.

É possível ainda, que em algumas implementações do CORBA, o código do cliente contenha objetos CORBA, possibilitando aos servidores executar chamadas a estes objetos, de forma que para um servidor poder localizar um objeto em um cliente, este servidor precisa ser informado da existência do objeto. Isto se dá de duas maneiras: (i) por meio do cliente, através uma referência ao objeto (do cliente) para o servidor, de forma simples, (ii) ou na forma de um parâmetro em alguma chamada anteriormente feita do cliente para o servidor em questão, também de forma bastante simples.

Existem disponíveis alguns tipos de chamadas efetuadas por um cliente [Teixeira00]:

- **tipo *blocking* (bloqueante):** tipo mais comum (*default).* Caracteriza-se bloqueante pelo fato de o cliente ficar bloqueado, no aguardo de uma resposta após a execução do código do objeto alvo.

- tipo *non-blocking* (não bloqueante): neste caso, permite-se que o chamador execute em paralelo com o pedido emitido, podendo receber resposta em uma oportunidade posterior.

- tipo *store-and-forward* (armazena e encaminha): o pedido é armazenado em um repositório não volátil (persistente) antes de ser passado para o objeto alvo;

- **tipo *publish-and-subscribe* (publica e faz assinatura):** uma mensagem é enviada tendo como destino um determinado tópico, de modo que qualquer objeto que estiver interessado neste tópico pode receber mensagens (desde que permitido).

3.4.1 Especificação CORBA

As principais funcionalidades da especificação **CORBA** são:

- O núcleo (*core*) – **ORB**;
- A linguagem de definição de interfaces (**IDL**);
- O repositório de interfaces;
- Mapeamentos de **IDL** para linguagens de programação;
- *Stubs* e esqueletos estáticos;
- Interfaces de invocação dinâmica (**DII**);
- Interface de esqueleto dinâmico (**DSI**);
- Adaptadores de objetos (*Object Adapters*);
- O repositório de implementações, e
- Protocolos (**GIOP** e **IIOP**).

O relacionamento entre as funcionalidades CORBA é ilustrado na Figura 3.5.

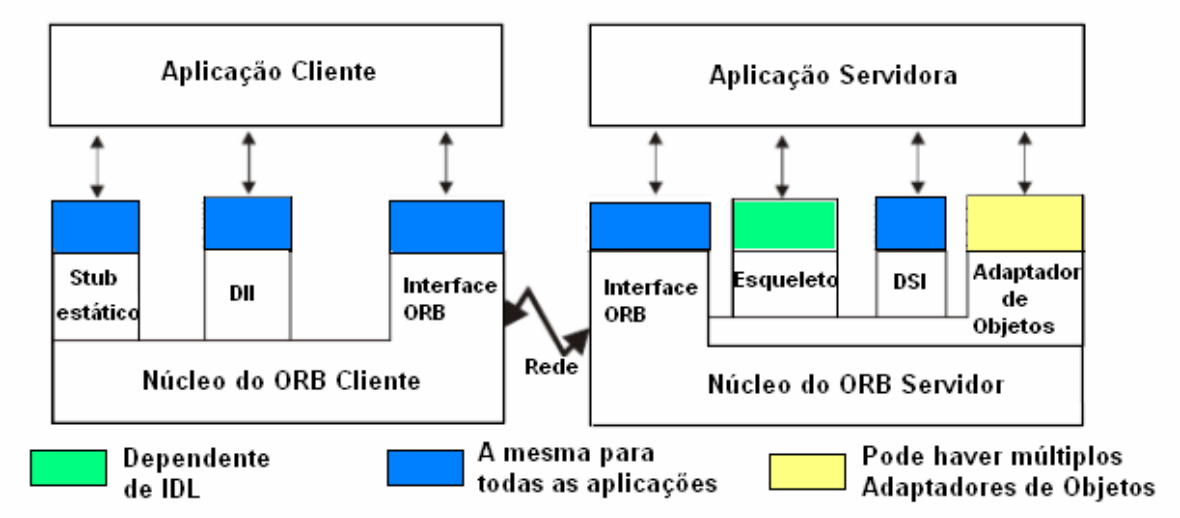

Figura 3.5: Arquitetura CORBA.

O fluxo de pedidos da Figura 3.5 segue os seguintes passos: (i) a aplicação cliente envia pedidos através de *stubs* estáticos ou através da interface DII, direcionando os pedidos para o núcleo do ORB do Cliente. (ii) os pedidos são direcionados, pelo núcleo do ORB do cliente, para o núcleo do ORB do servidor. (iii) o núcleo do ORB do servidor envia o pedido para o adaptador de objetos do servidor, que é responsável pela criação do objeto alvo. (iv) o adaptador de objetos do servidor transfere o pedido para o servente que estiver implementando o objeto alvo (não mostrado na figura). (v) o servente atende ao pedido, e retorna a resposta para o cliente [Teixeira00].

3.5 Tolerância a Falhas no CORBA

O objetivo de tolerância a falhas é alcançar dependabilidade (*dependability*). A dependabilidade indica a qualidade do serviço fornecido por um dado sistema e a confiança depositada no serviço fornecido [Weber02]. Dentre as principais medidas para prover dependabilidade podemos destacar [Weber02]:

- **Confiabilidade** (*reliability*) – capacidade de atender a especificação, dentro de condições definidas, durante certo período de funcionamento e condicionado a estar operacional no início do período. Ou seja, uma vez instalado e parametrizado, o sistema é estável o suficiente para funcionar sem a necessidade de acertos ou adaptações, eliminando o desperdício de tempo com paradas para manutenção do sistema ou atualização de versão.

- **Disponibilidade** (*availability*) **–** probabilidade de o sistema estar disponível (em pleno funcionamento) num dado momento requerido.

- **Segurança de Funcionamento** (*safety*) – medida de prevenção que permite ao sistema descontinuar sua execução caso ocorra alguma falha significativa.

- **Segurança** (*security*) **–** idem ao anterior, porém relativo à proteção de falhas contra a integridade, autenticidade e confidencialidade do sistema.

- **Manutenibilidade** – significa a facilidade de realizar a manutenção do sistema, ou seja, a probabilidade que um sistema com defeitos seja restaurado a um estado operacional dentro de um período de tempo determinado. Restauração envolve a localização do problema, o reparo físico e a colocação em operação.

- **Testabilidade** – capacidade de testar certos atributos internos ao sistema ou facilidade de realizar testes.

- **Comprometimento do Desempenho** (*performability*) – está relacionada à queda de desempenho provocado por falhas, onde o sistema continua a operar, mas degradado em desempenho.

Devido a grande quantidade de aplicações críticas que usam a arquitetura CORBA, tornou-se necessário garantir o funcionamento continuado destas aplicações, mesmo em caso de falhas. Para isso, foram implementadas técnicas de tolerância a falhas. A OMG inseriu na arquitetura CORBA alguns mecanismos que, além de redirecionar o cliente para um objeto replicado ativo, garantisse as propriedades de consistência e integridade dos dados [Mendonça02]. Entre os objetivos principais a serem alcançados pela arquitetura FT-CORBA [OMG02] pode-se destacar:

- Garantir a ausência de pontos de falhas, principalmente em aplicações críticas;

- Tornar a administração de réplicas mais flexível, utilizando técnicas de redundância, detecção de falhas e recuperação;

- Facilitar a administração de grupos de objetos;
- Garantir a consistência das réplicas, mesmo em caso de falhas de larga escala;
- Garantir serviços de descoberta, notificação e recuperação do estado de um objeto falho;
- Prover *checkpoints*[4] controlados pela aplicação ou pelos usuários, no qual agem os mecanismos de *logging*[5] e *recovery*.

Para uma melhor compreensão das técnicas de tolerância a falhas, são listados a seguir alguns conceitos básicos de tolerância a falhas no **FT-CORBA**.

- **Replicação** – consiste na criação de réplicas de um mesmo objeto em meios físicos diferentes, visando garantir a recuperação imediata do sistema, através da reconfiguração automática, permitindo que em caso de falha de objeto, o processo que estava sendo executado seja direcionado para outro caminho alternativo mantendo, desta forma, a comunicação contínua. Cada réplica possui uma referência própria *Interoperable Object Reference* (**IOR**).
- **Grupos de Objetos –** objetos são integrados em sub-grupos diferentes, cada grupo contendo uma referência única *Interoperable Object Group Reference* (**IOGR**)[6], nesta técnica já está implementada a primitiva de comunicação *multicasting,* direcionando o recebimento de uma requisição somente para um determinado grupo de objetos, reduzindo desta forma o tráfego na rede. No contexto de **Replicação de Grupos e Objetos** tem-se que vários objetos

[4] O *checkpoint* é um determinado ponto no qual o estado atual de uma parte do sistema que está no arquivo *log* é verificado e copiado no arquivo *log* das demais partes, permitindo a constante atualização das partes do sistema, mantendo-se assim consistente [Mendonça02].

[5] O *logging* é o mecanismo pelo qual as requisições feitas pelo usuário, a uma parte do sistema que está distribuído em algum meio físico, são gravadas em um arquivo *log*, para que em caso de falha, uma outra parte do sistema que assumir o lugar do anterior, seja capaz de obter o estado corrente e assim dar continuidade ao processo [Mendonça02].

[6] A IOGR (*Interoperable Object Group Reference*) é utilizada para referenciar objetos replicados. Trata-se da representação alfanumérica de todos os membros de um grupo de objetos, a partir do qual os clientes podem localizar dentro de um determinado domínio de tolerância a falhas, o objeto de interesse.

passam a formar um grupo de objetos, passando, portanto, a terem uma referência única (**IOGR**), simplificando desta forma o mecanismo de replicação e contribuindo para tornar as invocações e as falhas totalmente transparentes ao usuário.

- **Domínio de Tolerância a Falhas** – visando facilitar a administração de réplicas, a OMG [OMG02] criou o conceito de domínio de tolerância a falhas, o qual define o agrupamento de vários terminais em redes distintas, onde cada terminal pode conter vários grupos de objetos replicados. Desta forma, cada domínio irá possuir seu gerente de réplica, facilitando a administração das mesmas. Além da comunicação entre objetos de mesmo domínio, a comunicação entre objetos de domínios diferentes também é permitida. Uma abordagem mais detalhada em cima deste conceito, é apresentada a seguir.

- **Propriedades de Tolerância a Falhas** – cada grupo de objetos pode ser diferenciado pelas propriedades específicas que possui. Estas propriedades são definidas no momento da criação de cada grupo e definem a maneira pela qual eles irão atuar. Como exemplos de propriedades temos [Lima01b]: *ReplicationStyles* (*Cold_Passive, Warm_Passive, Active* ou *Active_With_Voting), InitialNumberReplicas, MinimalNumberReplicas, CheckPointInterval* e outras. Estas propriedades podem ser definidas tanto para um domínio, quanto para um grupo específico de objetos dentro do domínio, e ainda, algumas delas podem ser alteradas dinamicamente.

3.5.1 Domínio de Tolerância a Falhas

Aplicações complexas distribuídas em rede quase sempre necessitam de tolerância a falhas. Porém, manipular este tipo de aplicação como uma única entidade é inviável. Com a introdução do conceito de domínios de tolerância a falhas, o gerenciamento deste tipo de aplicação tornou-se mais fácil. Um domínio de tolerância a falhas é formado por várias máquinas e grupos de objetos, sendo que um objeto pertence a apenas um grupo, porém grupos de objetos podem possuir objetos em diversas máquinas.

Cada domínio possui um conjunto de componentes definidos na infra-estrutura de tolerância a falhas FT-CORBA, que atuam exclusivamente dentro deste domínio. Entre eles encontra-se o **Gerenciador de Replicação**, o **Notificador de Falhas** e o **Detector de Falhas**.

A Figura 3.6, extraída de [Batalha01], ilustra o funcionamento dos componentes de tolerância a falhas de um domínio.

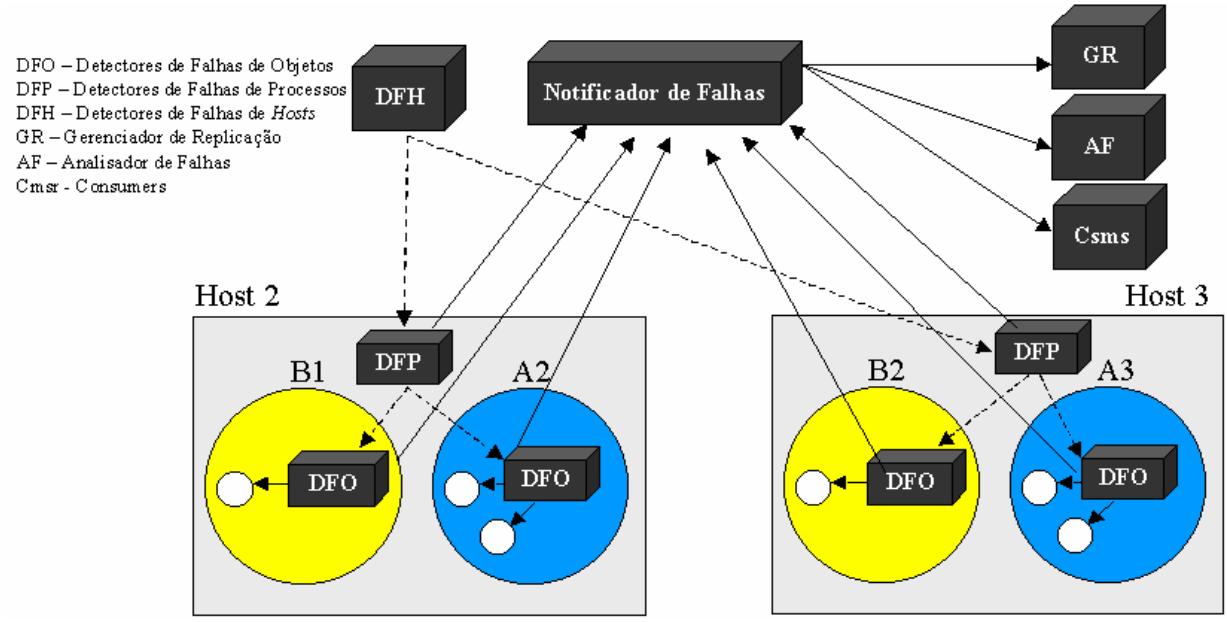

Figura 3.6: Componentes da Arquitetura FT-CORBA.

A Figura 3.6 representa dois terminais de um domínio de tolerância a falhas, o ***Host* 2** e o ***Host* 3**, sendo monitorados pelos componentes de tolerância a falhas do domínio. Observa-se a presença dos **Detectores de Falhas de Objetos** (**DFO**), dos **Detectores de Falhas de Processos** (**DFP**), dos **Detectores de Falhas de *Host*** (**DFH**), do **Notificador** (**NF**) e do **Analisador de Falhas** (**AF**), do **Gerenciador de Replicação** (**GR**) e dos ***Consumers*** (**Csms**), que se registram no notificador para receber informações de eventos de falhas. As setas tracejadas indicam monitoração, e as setas contínuas indicam eventos de falhas que estão sendo reportados.

3.5.2 Componentes da Arquitetura FT-CORBA

- **Gerente de Replicação** (*Replication Manager*) – Componente principal da arquitetura. Interage com os outros componentes enviando e coletando informações necessárias para o gerenciamento eficiente de tolerância a falhas. Deve existir um gerente de replicação em cada domínio de tolerância a falhas.

Entre suas principais funções pode-se destacar o gerenciamento da criação de grupos e a criação de objetos de um grupo [Mendonça02].

- **Notificador de Falhas** (*Fault Notifier*) – Componente responsável por receber relatórios de falhas dos monitores de objetos, para então repassar para uma aplicação cliente ou para o Gerente de Replicação.

- **Detector de Falhas** (*Fault Detector*) – Componente responsável por monitorar os objetos replicados para verificar se estes ainda estão ativos.

- **Mecanismos de Escrita e Leitura** (*Logging* e *Recovery*) **–** O mecanismo de **escrita** (*Logging*) tem a função de anotar o estado atual do objeto primário nos objetos secundários. O mecanismo de **recuperação** (*Recovery*) tem a função de recuperar o estado atual do objeto primário caso o mesmo venha a falhar.

3.5.3 Limitações

Mesmo possuindo diversas técnicas de tolerância a falhas, a arquitetura FT-CORBA apresenta algumas limitações que podem influenciar de forma significativa na garantia de qualidade total dos sistemas que necessitam de tolerância a falhas. Dentre estas limitações [OMG02] pode-se destacar:

- A **comunicação insegura entre ORBs cliente e servidor** – a infra-estrutura de tolerância a falhas, só é aplicada no lado servidor, impossibilitando o cliente de detectar uma falha ocorrida na réplica requisitada;

- A **infra-estrutura comum –** existe uma limitação quanto a interoperabilidade entre ORBs, exigindo, para o funcionamento correto da infra-estrutura de tolerância a falhas, a utilização dos mesmos ORBs por parte do cliente e do servidor;

- O **comportamento determinístico** – o comportamento determinístico das réplicas de objetos servidores não é garantido, fazendo com que as respostas

requisitadas pelos clientes sejam imprecisas, caso tenha ocorrido, durante o processamento, erros que possam ter danificado os dados;

- As **falhas em redes particionadas** – a tolerância a falhas não é garantida caso as réplicas de objetos estejam localizadas em redes diferentes. A comunicação entre redes distintas geralmente necessita da intervenção de equipamentos susceptíveis à falhas (*switches, hubs,* roteadores e outros). Pode ocorrer falha em um destes dispositivos, tornando indisponível a réplica do objeto, localizado na rede particionada.

- A **falha comissionada** – as especificações FT-CORBA não garantem a tolerância a falhas em caso de geração de resultados incorretos pelo objeto replicado. Estes erros podem ser causados por mau funcionamento e ou por fatores externos (ruídos, campos magnéticos etc), interferindo nos equipamentos envolvidos em um domínio de tolerância a falhas.

- A **falha correlata** – falhas causadas pelo mau projeto do sistema ou de programação não são garantidas pelas especificações FT-CORBA. Tais falhas podem, muitas vezes, interferir em todos os objetos, nos ORBs, nos terminais ou no sistema operacional.

3.6 Uso de FT-CORBA no Cenário de Empresas Virtuais

Como visto no Capítulo anterior, as tecnologias necessárias para operação de empresas virtuais são complexas e os sistemas que dão suporte a estas empresas são caracterizados pela heterogeneidade e larga escala. Desta forma, é necessário disponibilizar para estas empresas um suporte computacional que facilite o emprego destas tecnologias, de forma a permitir que interações entre empresas possam ocorrer de maneira segura, confiável, e com um desempenho satisfatório. O suporte deve ainda fornecer mecanismos para a especialização, no sentido de atender os requisitos de qualidade de serviço das empresas virtuais [Fraga01a].

As especificações FT-CORBA apresentam limitações para sistemas distribuídos de larga escala. Nestas especificações, não existe a definição de um serviço de comunicação de grupo que

é fundamental na implementação de replicações ativas em sistemas distribuídos. A tolerância a falhas em redes de larga escala envolve complexidades adicionais tanto na comunicação, como na gestão dos componentes do sistema distribuído.

Em [Lima01b] é apresentada a arquitetura **FT-SALE**, a qual objetiva inserir tolerância a falhas dentro do contexto de Empresas Virtuais através de adaptações nas especificações FT-CORBA. Esta proposta teve como base a plataforma **SALE** [Fraga01a], a qual objetiva integrar e conciliar mecanismos que forneçam diferentes garantias de desempenho, confiabilidade e segurança nas interações das aplicações de larga escala, fazendo uso das especificações **RT-CORBA**[7], **FT-CORBA** e **CORBAsec**[8]**,** respectivamente, em conjunto com as experiências do **GrouPac 2.0** [Laurindo01], para tratar tolerância a falhas em redes de larga escala.

A arquitetura **FT-SALE** é apresentada em maiores detalhes no Capítulo 5. O capítulo seguinte apresenta as principais características dos sistemas de agentes móveis, objetivando a compreensão do sistema proposto em [Schmidt03], o qual é usado como base para a proposta apresentada neste trabalho.

[7] RT-CORBA é a extensão do CORBA para tratar de sistemas em tempo real.
[8] CORBAsec é a extensão do CORBA para tratar de sistemas seguros.

4. SISTEMAS DE AGENTES MÓVEIS

4.1 Introdução

A inteligência humana desperta há muitos séculos a curiosidade no campo das ciências humanas, principalmente na Filosofia, gerando desta forma intenso estudo neste campo. O século passado marcou o nascimento de uma nova ciência também preocupada com este fenômeno, a **Inteligência Artificial** [Russell02]. A Inteligência Artificial estuda a forma como uma entidade computacional é capaz de exibir comportamento inteligente, o que obrigou o estudo exaustivo de técnicas heurísticas, métodos baseados em conhecimento, planejamento, aprendizagem, compreensão e geração de linguagem natural.

O crescimento das redes de computadores possibilitou, entre outras coisas, atacar problemas de natureza distribuída, trabalhar com paralelismo e somar o poder computacional dos nós interligados. A **Inteligência Artificial Distribuída** (**IAD**) busca construir e desenvolver sistemas compostos de múltiplas entidades com graus de autonomia e inteligência variáveis, que interagem entre si, visando solucionar um problema global, e aumentar o grau de desempenho dos sistemas.

A investigação da IAD é baseada nos possíveis comportamentos de múltiplos **agentes**, membros de uma mesma sociedade, e estuda o problema de como aumentar a eficiência de um sistema que busca obter a satisfação de objetivos globais, utilizando múltiplos agentes com objetivos locais.

No mundo real é visível a presença de **múltiplos agentes**, visto que, a todo o momento, as pessoas necessitam planejar as suas ações considerando potenciais ações dos outros membros da sociedade, os quais podem influenciar de alguma forma a sua própria atividade.

Na área de **Solução Distribuída de Problemas**, um agente é construído visando um problema em questão. Em seguida, são estudadas formas de interação, do agente criado, com outros agentes, considerando aspectos individuais, como interesses específicos que conduzem sua ação. Os **Sistemas Multi-Agente** estão relacionados com esta interação entre diferentes agentes, os quais se unem buscando atingir um determinado objetivo [Schmidt03].

Este capítulo aborda alguns conceitos básicos de Sistemas Multi-Agentes e Sistemas Multi-Agentes Móveis, objetivando uma melhor compreensão do sistema de busca e seleção de parceiros para a formação de uma empresa virtual [Schmidt03], o qual é a base da plataforma proposta neste trabalho. São abordados, neste capítulo, alguns aspectos sobre a interoperabilidade entre agentes. São apresentados ainda, os agentes que compõem o sistema de busca e seleção. Por último, alguns métodos de tolerância a falhas aplicados a sistemas de agentes móveis são brevemente descritos.

4.2 Inteligência Artificial Distribuída

A **Inteligência Artificial Distribuída** (**IAD**) estuda os métodos para analisar e desenvolver comunidades inteligentes compostas por grupos de processos que interagem, são coordenados e baseados em conhecimento. Tais comunidades são formadas por nós ou **agentes** com capacidade de processamento, sendo que, dado um problema, devem atuar na sua resolução [Schmidt03].

Estes sistemas visam, entre outras coisas: resolver problemas cujo tamanho ou complexidade são tão grandes, que se torna impossível resolvê-los através do uso de processos simples; facilitar o desenvolvimento e a manutenção dos sistemas; reduzir custos e limitação de recursos e melhorar questões de adaptabilidade, eficiência, velocidade e confiança nos sistemas.

Uma das principais razões que despertou a necessidade de se distribuir inteligência foi o crescimento da complexidade dos sistemas computacionais, como, por exemplo: aplicações de larga escala (como as das Empresas Virtuais), as quais necessitam interligar múltiplos sistemas, quase sempre heterogêneos e normalmente distribuídos geograficamente.

Existem duas correntes principais na IAD:

- **Solução Distribuída de Problemas (SDP)** – Geralmente, este tipo de solução inclui a existência de regras já definidas relativas ao comportamento dos múltiplos agentes da comunidade (a coordenação das atividades dos agentes é centralizada). O planejamento das ações a se desenvolver por cada um dos agentes é efetuado inicialmente por decomposição do problema em sub-problemas e a partir daí é feita a atribuição do problema aos vários agentes.

Assim, a resolução de um problema é alocada a um conjunto de agentes, que cooperam apenas na divisão do esforço e partilha de conhecimento e resultados [LiXin99].

- **Sistemas Multi-Agentes (SMAs) –** É o caso em que se têm vários agentes que não foram desenvolvidos por um único projetista atuando no mesmo sistema [Schmidt03]. Os **Sistemas Multi-Agentes** ou **SMAs** podem ser definidos como uma rede de solucionadores de problemas ou agentes que trabalham coletivamente além das capacidades individuais de cada um de forma a resolver um problema comum [Gomes00]. Entre as vantagens dos SMAs pode-se citar: (i) habilidade em resolver problemas que seriam considerados muito grandes para um único agente, (ii) resolver problemas inerentemente distribuídos[9], (iii) oferecer maior velocidade e confiabilidade, (iv) oferecer claridade conceitual e simplicidade de projeto (em alguns casos) e (v) tolerar dados e conhecimentos incertos.

4.3 Agentes

A principio, ainda não existe uma definição universalmente aceita para o termo "agente". No seu sentido mais restrito, o termo pode ser aplicado a toda uma gama de entidades, incluindo os sistemas de *software* [Gomes00].

Existem na literatura muitas definições publicadas para o termo agente, porém torna-se inconveniente uma definição fechada, visto que a mesma pode limitar o campo de estudo. Desta forma, é apresentada neste trabalho uma definição dada por [Wooldridge99], a qual possui a vantagem de ser genérica o bastante para permitir uma ampla gama de estudos e ainda descreve propriedades nas quais podem ser identificadas definições de agente comumente encontradas na literatura [Gomes00].

> "Um agente é um sistema computacional situado em um ambiente, sendo capaz de agir autonomamente para alcançar suas metas." [Wooldridge, 1999, pág.29].

[9] Alguns problemas devem ser mantidos distribuídos, por questões de controle da privacidade dos dados, por exemplo, organizações independentes mantêm os dados privativamente, por razões competitivas [Gomes00].

As propriedades do agente definem qual é a classificação do agente, ou seja, um agente inteligente, reativo, capaz de aprender, etc. Estas propriedades correspondem as características de um sistema, que são resultados diretos de sua arquitetura ou dos seus níveis de operação. Segundo [Fleischhauer98], entende-se por arquitetura a porção do sistema que fornece e gerência os recursos primitivos de um agente.

Ainda de acordo com [Fleischhauer98], para que o sistema seja considerado agente, é recomendável que o mesmo apresente algumas propriedades, tais como:

- **Autonomia** – É a capacidade de independência do agente em relação ao usuário. Isso inclui aspectos de ação periódica, execução espontânea e iniciativa, em que o agente precisa ser capaz de tomar ações preemptivas (optativas) e independentes que eventualmente beneficiarão o usuário, permitindo ao agente ter algum tipo de controle sobre suas ações e estado interno.
- **Mobilidade** – É a habilidade para mover-se de uma localização para outra, enquanto preserva seu estado interno.
- **Comunicabilidade** – É a capacidade de trocar informações com outras entidades (agentes, humanos, objetos, seu ambiente). Nas interfaces entre os agentes, é necessário decidir que tipo de declaração os agentes são capazes de gerar e compreender. Há ainda um outro problema na comunicação que é a interpretação e significado das declarações do agente. Um termo pode ter diferentes significados para diferentes agentes, o que é chamado de conflito. Diferentes termos podem, entretanto, ter significado similar, o que significa uma correspondência.
- **Aprendizagem** – Agentes são capazes de examinar o ambiente externo (por exemplo, a Web) e o "sucesso" de ações prévias levam condições similares, e adaptam suas ações para melhorar a probabilidade de atingir adequadamente suas metas.
- **Reatividade** – Um agente está inserido em um ambiente que pode ser o mundo físico, um usuário via uma interface gráfica, uma coleção de agentes, a Internet,

ou talvez todos esse combinados. A reatividade é a capacidade de um agente responder de uma maneira oportuna à mudanças que ocorre nele e no ambiente.

- **Iniciativa** – É a habilidade de exibir comportamento direcionado ao objetivo, oportunístico e que não reage simplesmente ao seu ambiente.
- **Sociabilidade** – É a capacidade de interação com outros agentes (e possivelmente humanos) através de algum tipo de linguagem para comunicação de agente.
- **Racionalidade** – É a suposição de que um agente atuará para atingir seus objetivos, e não atuará de modo que impeça que seu objetivo seja alcançado – pelo menos até onde sua opinião permitir.
- **Percepção** – Os agentes sentem o mundo e geram conhecimentos acessíveis para processar o que a razão está dizendo para perceber no mundo.
- **Cooperação** – Agentes inteligentes precisam ter um "espírito" cooperativo para existir e ter sucesso em "sistemas orientados a agentes". O que se quer é que agentes inteligentes trabalhem juntos para que possam executar tarefas mutuamente benéficas mais complexas.

As definições das propriedades parecem ser mais convergentes que as definições de agentes, entretanto podem ser interpretadas de várias formas. Dependendo das características que um agente apresenta, ele pode ser classificado em categorias distintas de agentes. Para poder classificar um agente, é necessário que o agente atenda algumas características peculiares. Por exemplo, para um agente ser considerado um **agente móvel**, ele precisa possuir a propriedade de **mobilidade** [Ferrari99].

4.4 Agentes Móveis

Segundo [Schmidt03], "a disponibilidade de aplicações e sistemas distribuídos acessíveis às pessoas em geral, como a Internet, tem despertado o interesse de usuários e do mercado, motivando o desenvolvimento de novas aplicações, que por sua vez dependem de técnicas, linguagens e paradigmas que ofereçam suporte para sistemas distribuídos. Um paradigma que tem recebido grande destaque e interesse em pesquisas é o de agentes móveis".

Os agentes móveis herdaram algumas características dos agentes definidos anteriormente, tais como: cooperação, autonomia e representatividade. Com a finalidade de suprir as necessidades exigidas para o bom funcionamento de modelos que utilizam o paradigma de agentes móveis, foram acopladas outras características, tais como: **objetos passantes**, **assincronismo**, **interação local**, **operações sem conexão** e **execução paralela**. Segundo [Lange97], estas características são definidas como:

- **Objetos Passantes** – Quando um agente móvel é transferido, todo o objeto é movido, ou seja, o código, os dados, o itinerário para chegar ao servidor necessário, o estado de execução, etc;
- **Assincronismo** – O agente móvel possui sua própria *thread* de execução e não precisa ser executado sincronamente;
- **Interação Local** – O agente móvel interage com outros agentes móveis ou objetos estacionários locais. Se necessário, um agente mensageiro é enviado para facilitar a interação com agentes remotos;
- **Operações sem Conexão** – O agente móvel pode executar tarefas mesmo com a conexão fechada. Quando se faz necessário a transferência de agentes, aguarda-se até que a conexão seja restabelecida;
- **Execução Paralela** – Múltiplos agentes podem ser disparados para diferentes servidores a fim de executar tarefas em paralelo.

Além destas características, um agente móvel implementa os seguintes modelos [Gomes00]: (a) um modelo de agente, (b) um modelo de ciclo de vida, (c) um modelo computacional, (d) um modelo de segurança, (e) um modelo de comunicação e ,finalmente, (f) um modelo de navegação. Estes modelos descrevem formalmente os agentes móveis, e são mapeados para uma implementação em linguagem de programação computacional.

Outro conceito importante relativo aos agentes móveis é o chamado **Ambiente de Agentes Móveis** (**AAM**), que pode ser definido como um sistema computacional distribuído que fornece o suporte necessário para que os agentes móveis executem as suas funções. De acordo com [Green97], "o AAM implementa a maior parte dos modelos que aparecem na definição de

agentes móveis. Ele também suporta serviços que estão relacionados com o próprio AAM, serviços que dizem respeito ao ambiente onde o AAM está implementado, e serviços para suportar o acesso a outros Sistemas de Agentes Móveis".

4.5 Sistemas Multi-Agentes

Como vimos, os **Sistemas Multi-Agentes** (**SMAs**) formam um grupo de *softwares*, cada qual com as suas capacidades, que trabalham de forma integrada para resolver um problema complexo. Dentre as principais características destes ambientes estão [Gomes00]: a definição de uma infra-estrutura de comunicação que permita o inter-relacionamento entre os agentes; não possuem um criador centralizado; e contêm agentes que são autônomos e distribuídos, podendo estes apresentar interesse próprio ou agir de forma cooperativa.

Segundo [Lobato05], "o conhecimento de um agente consiste em informações sobre o seu ambiente e outras entidades que compõem o ambiente, ou sobre o agente em si, e o conjunto de funcionalidades providas pelo agente. Existem dois tipos de conhecimento de um agente, o **conhecimento intrínseco** que é o conhecimento relacionado às informações e funcionalidades básicas do agente e o **conhecimento extrínseco** que é o conhecimento relacionado aos diferentes papéis desempenhados pelo agente nos vários contextos de colaboração. Compreende a informação e os serviços necessários para que o agente possa desempenhar papéis em interações com outros agentes".

Normalmente, existem vários tipos de agentes de *software* em um **SMA**. Segundo [Lobato05], "diferentes tipos de agentes são necessários, porque a estrutura interna de um tipo define o mesmo conjunto de funcionalidades para agentes que pertencem a este tipo. Aos vários tipos também estão associadas diferentes propriedades de agentes como adaptação, colaboração, aprendizagem e mobilidade. As relações mantidas entre estas propriedades também são dependentes de cada tipo".

4.6 Sistemas Multi-Agentes Móveis (SMAs Móveis)

Como visto anteriormente, agentes móveis podem ser definidos como agentes de *software* que possuem a habilidade de se mover entre os diversos terminais em uma rede. De acordo com a aplicação em execução, os agentes móveis são capazes de migrarem para o terminal onde os dados de interesse da aplicação se encontram armazenados, selecionarem informações de que o usuário necessita e retornam com os resultados para o terminal inicial.

Existem alguns cenários, onde o agente é projetado apenas para transferir os dados encontrados, deixando a tarefa de seleção das informações para outros agentes, e ainda alguns cenários nos quais os agentes móveis são projetados para trocar mensagens com os agentes que efetivamente transferem, processam ou retornam as informações para os usuários. As funcionalidades básicas de agentes móveis estão relacionadas aos requisitos das aplicações onde estes serão utilizados.

A movimentação de um agente móvel se dá em um ambiente distribuído, entre terminais de uma rede aos quais possui acesso. A configuração de acesso é feita através de um servidor instalado nos terminais. Entre as funções deste servidor, está a tarefa de fornecer aos agentes um **local** adequado à sua execução, o qual é designado pelo termo **contexto de execução** dos agentes.

O protocolo de mobilidade do agente é dado pelo conjunto de interações entre agentes móveis e o servidor de agentes de suas respectivas plataformas. Ao ser transferido na rede, a execução do agente é interrompida no terminal original, e após o transporte para outro contexto no ambiente distribuído, sua execução é reiniciada a partir do ponto de interrupção anterior. Ao migrar, o agente "carrega" o código e os dados de sua execução. De acordo com [Gomes00], "isto permite que o agente seja transportado na rede, sem que seus atributos internos devam ser reiniciados".

Portanto, para que ocorra a execução de agentes móveis em uma rede é necessário que exista uma infra-estrutura de rede necessária à mobilidade dos agentes, o servidor de agentes e o contexto de execução dos agentes fornecido pelo servidor. A Figura 4.1, extraída de [Gomes00] ilustra a estrutura geral necessária para a execução de agentes móveis sobre uma rede de comunicações.

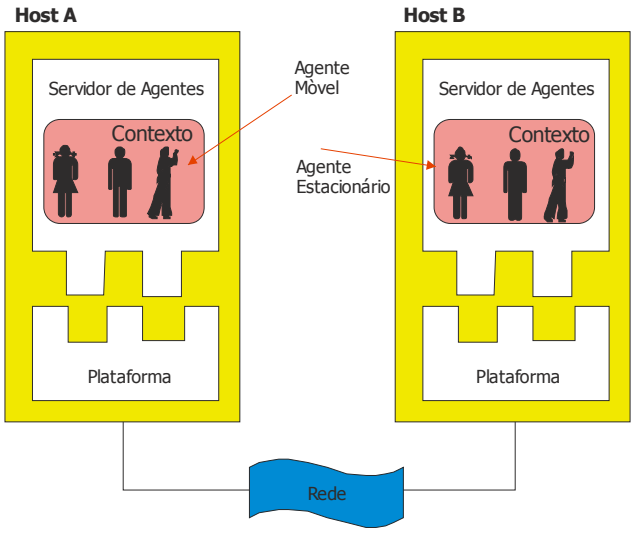

Figura 4.1: Estrutura Geral de SMAs Móveis.

Desta forma, pode-se descrever a existência de um agente móvel através de um **Ciclo de Vida**, que é tipicamente composto das seguintes etapas ou operações [Gomes00]:

1. **Instanciação** – Executada somente uma vez durante todo o ciclo de vida. O agente instanciado recebe um identificador, tem seu estado interno iniciado e é preparado para instruções adicionais;

2. **Inicialização** – Esta operação é efetuada toda vez que um agente chega a um novo terminal;

3. **Movimentação** – A movimentação permite ao agente transferir-se para o contexto onde se localiza um objeto com o qual deseja interagir. As circunstâncias nas quais os agentes devem se movimentar, ou ainda, os instantes em que a ação de movimentação deve ser disparada, são especificados por desenvolvedores e são denominados pontos de movimento das aplicações.

   A **migração** dos agentes pode ser classificada em migração **fraca** quando o agente reinicia a execução em um novo *host* a partir da primeira linha de seu código, ou **forte** quando a execução reinicia exatamente do ponto onde foi interrompida, uma vez que a pilha de execução é integralmente recuperada no

novo contexto do agente. Independente do tipo de migração, o agente tem sua execução reiniciada sem perda dos valores de seus atributos internos.

A migração dos agentes pode também ser classificada em **proativa** ou **reativa**. Na migração **proativa**, o próprio programa decide realizar a migração. Na migração **reativa**, a iniciativa vem de um outro programa que, por exemplo, pode estar sendo executado na máquina de destino.

Em relação ainda ao programa que toma a iniciativa, a migração pode ser **síncrona** ou **assíncrona**, dependendo se o programa espera ou não pela execução do código que migrou [Kazuhiro04]. A Figura 4.2 ilustra os tipos de migração existentes.

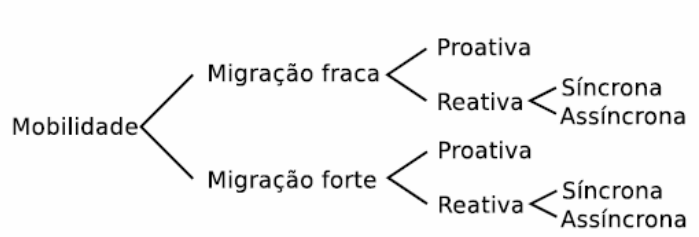

Figura 4.2: Tipos de migração.

4. **Destruição –** Implica na finalização das atividades de um agente, liberando os recursos sendo utilizados por ele. Após esta operação, perde-se o agente.

4.6.1 Considerações sobre Benefícios e Problemáticas

Dentre os principais benefícios do uso da tecnologia de agentes móveis estão:

- **Redução da Carga na Rede** – O uso de agentes móveis permite que o processamento de dados seja feito no próprio sistema aonde os dados estejam localizados, evitando assim a transferência de grandes quantidades de dados através da rede. Após o processamento, somente os resultados são transferidos ao destino.

- **Execução Assíncrona e Autônoma** – Ao invés de manter uma conexão permanente entre duas aplicações, o uso de agentes móveis permite que a conexão seja estabelecida somente para enviar o agente móvel, e posteriormente para retornar o agente com os resultados encontrados. Desta forma, não é exigida a sincronização entre a conexão e o processamento das tarefas. De acordo com [Lobato05], isto é possível porque, após a transferência, os agentes móveis tornam-se independentes da máquina original e operam de maneira assíncrona.

- **Adaptação Dinâmica** – Em SMAs, os agentes podem trabalhar coletivamente de forma a resolver um problema comum. Durante este processo, os agentes podem trabalhar de forma a balancear a carga na rede de comunicações.

- **Tolerância a Falhas** – Uma outra vantagem é a facilidade de construção de sistemas distribuídos robustos e tolerantes a falhas, devido as características de execução autônoma e assíncrona, e a adaptação dinâmica. Em especial, elas permitem que um agente tome uma atitude em situações desfavoráveis. Por exemplo, se uma máquina precisa ser desligada, os agentes nessa máquina podem ser avisados e com isso migrar para outra máquina da rede para que possam dar continuidade a seu trabalho [Kazuhiro04].

Entretanto, a utilização do paradigma de agentes móveis também possui algumas desvantagens. A primeira delas é a necessidade de sistemas adicionais para o suporte ao paradigma. Para o processamento de agentes móveis geralmente são utilizadas plataformas de mobilidade, o que limita a mobilidade de um determinado agente apenas aos locais que possuem a infra-estrutura apropriada a ele. A existência de diversos tipos de ambientes para agentes móveis gera um outro problema, que é a **interoperabilidade** de agentes diferentes entre si e, de agentes e sistemas de agentes diferentes. Isto implica em dificuldades, principalmente na comunicação a ser estabelecida entre estes elementos. A próxima seção aborda, de forma mais detalhada, o problema da interoperabilidade em sistemas de agentes móveis.

Porém, a segurança continua sendo o maior problema quando o assunto em questão é o uso de agentes móveis em sistemas de IAD. Faz-se necessário o uso de técnicas de segurança,

para a garantia da confiabilidade, tais como: restrições de acesso aos recursos das máquinas ou autenticação para garantir a confiabilidade do código, visto que não é sempre que se pode confiar em códigos provenientes de outras máquinas. Desta forma, torna-se possível impedir que um código malicioso provoque danos aos terminais destinatários de agentes móveis [Gomes00].

Os ataques relacionados com agentes podem ser divididos em quatro categorias [Kazuhiro04]: (i) um agente atacando o sistema de agentes em que se encontra; (ii) o sistema atacando um agente que está hospedando; (iii) um agente atacando outro agente no mesmo sistema; e (iv) outra entidade externa (agente em outro sistema, outro sistema ou outro programa) atacando o sistema. Além disso, os tipos de ataque podem ser classificados em: exposição de informação, negação de serviço e corrupção de informação.

4.6.2 O Problema da Interoperabilidade

Como visto, o ambiente no qual residem os agentes móveis é quase sempre heterogêneo. E mais, os agentes durante o seu ciclo de vida migram constantemente de terminal. De acordo com [Schmidt03], nestas redes, freqüentemente, as plataformas para agentes móveis são diferentes, fato que gera o problema de heterogeneidade, e por conseqüência, a não interoperabilidade.

Um dos principais objetivos do consórcio internacional de *software* OMG [OMG03] é a regulamentação de padrões na área de computação de objetos distribuídos. A OMG busca o estabelecimento de linhas guia para a indústria e de especificações de gerenciamento de objetos para prover um arcabouço comum para o desenvolvimento de aplicações, estimulando, desta forma, o crescimento da tecnologia de objetos, principalmente quanto à reutilização, portabilidade e interoperabilidade de softwares baseados em objetos, em ambientes distribuídos e heterogêneos.

Existem alguns padrões já estabelecidos com a finalidade de interoperabilidade, tais como: **CORBA**, **UML** (*Unified Modeling Language*) e o **IIOP**. Os sistemas de agentes, em sua maioria, buscam a interoperabilidade. Visando atender esta necessidade, a **OMG** criou a especificação **MAF** [OMG00] (do inglês, □*óbile Agents Facility*), a qual procura solucionar o problema da interoperabilidade entre sistemas de agentes escritos na mesma linguagem, mas

potencialmente criados por diferentes fornecedores e em diferentes sistemas, utilizando para isto, uma coleção de definições e interfaces que determinam uma interface interoperável para sistemas de agentes móveis.

De acordo com [Schmidt03], "a padronização **MAF** atinge três pontos importantes: (i) gerenciamento de agentes: criação, desativação, encerramento, entre outros eventos; (ii) transferência de agentes: infra-estruturas comuns para permitir o livre deslocamento; e, (iii) nomes de agentes e de sistemas de agentes: uso de sintaxe e semântica padrão para a identificação de agentes".

Os principais conceitos do paradigma de agentes móveis padronizados pela **MAF** podem ser encontrados em [OMG00] e [Schmidt03].

4.6.3 Comunicação entre Agentes

A comunicação entre os agentes é uma questão ainda não abordada pela **MAF**. Entretanto, é uma das questões bastante discutidas na arquitetura **CORBA**, um padrão também estabelecido pela **OMG**. Assim, a comunidade de agentes móveis enxerga a comunicação entre agentes como sendo, na maior parte do tempo, a troca de objetos ou de referências para objetos e requisições de serviços.

A visão de interoperabilidade entre agentes móveis não deve ser considerada como sendo apenas um problema de diferentes sistemas e plataformas possíveis de serem visitadas. Deve-se abordar ainda o lado da comunicação entre estes agentes. Questões como a interação entre agentes em plataformas diferentes, ou entre agentes e fontes de informações ainda precisam ser respondidas. Esta interação inclui a troca de mensagens, a localização das fontes de informações, a sua identificação e avaliações de relevância de informações. Uma última questão ainda, de acordo com [Schmidt03], "é a falta de abordagens declarativas para agentes móveis, onde o agente simplesmente especificaria a tarefa que deve ser realizada, deixando os detalhes de como fazer para quem recebe o pedido".

4.6.4. Linguagem de Comunicação de Agentes

No que diz respeito às linguagens de comunicação em sistemas multi-agente, cabe destacar dois grandes esforços para prover interoperabilidade em sistemas abertos: um promovido pela **FIPA** (*Foundation for Intelligent Physical Agents*) [FIPA] e outro pela **ARPA** (*Advanced Research and Projects Agency*) [ARPA], em meados dos anos 90.

O trabalho desenvolvido pela **ARPA** resultou em um projeto denominado **KSE** (*Knowledge Sharing Effort*) [Neches94], que tem como objetivo o desenvolvimento de metodologias e *software* para o compartilhamento e a reutilização de conhecimento, e que durante muito tempo tornou-se a base para um padrão na implementação de comunicação em sistemas multi-agentes.

A interoperabilidade de agentes envolve três questões principais [Schmidt03]: i) como fazer a tradução de uma linguagem de representação de conhecimento para outra; ii) como manter o significado de objetos, relações e conceitos entre diferentes agentes; iii) como compartilhar esse conhecimento (comunicar);

Para lidar com estas questões, utiliza-se uma linguagem formal de comunicação de agentes chamada **ACL** (*Agent Communication Language*). Em português se utiliza o termo **LCA**, que significa **Linguagem de Comunicação de Agentes**. As linguagens mais populares são **FIPA-ACL**, da **FIPA**, e **KQML** (*Knowledge Query and Manipulation Language*), da **ARPA**.

Do ponto de vista das linguagens de comunicação, um agente é visto como um componente de *software* que se comunica com outros agentes através de mensagens codificadas em **LCA**. Dessa forma, uma **LCA** é uma coleção de tipos de mensagens, cada uma tendo um significado reservado. Uma linguagem de comunicação tem a capacidade de diferenciar trocas de mensagens conforme suas intenções, podendo, por exemplo, distinguir uma mensagem de consulta, de outra que está emitindo uma resposta [Schmidt03].

Na próxima seção é feita uma breve apresentação da linguagem **KQML**, que é utilizada pelo sistema de **busca e seleção de parceiros** em empresa virtuais, conforme será visto no

próximo capítulo. A linguagem **FIPA-ACL** é mais recente que a **KQML**, porém é similar e possui o mesmo objetivo.

4.6.4.1 KQML

A linguagem **KQML**, cuja tradução de acordo com [Schimdt03] seria algo como Linguagem para Manipulação e Consulta de Conhecimento, foi projetada para suportar a comunicação entre agentes. A linguagem **KQML** é hoje considerada como a proposta padrão mais utilizada como linguagem de comunicação entre agentes [Loss03].

O objetivo do **KQML** é dar suporte ao compartilhamento de conhecimento, em tempo de execução, entre agentes. Conceitualmente, uma mensagem **KQML** é organizada em três partes: **conteúdo**, **comunicação** e **mensagem**. As primitivas da linguagem **KQML** são chamadas de "performativas" (*performatives*). As performativas definem ações possíveis ou os tipos de interações que os agentes podem usar na comunicação com outros agentes. Os demais parâmetros das mensagens são vistos como argumentos ou atributos da performativa.

A especificação da linguagem **KQML** assume que a comunicação entre os agentes é confiável, e que a transferência de mensagens ocorre de um modo assíncrono em relação à operação dos agentes. Além disso, ela é neutra em relação ao método usado para codificar a mensagem, que pode ser desde uma cadeia arbitrária de caracteres até uma expressão **SQL** de consulta a um banco de dados [Lobato05].

Um ambiente onde os agentes se comunicam através da **KQML** pode ser enriquecido com agentes "facilitadores", que fornecem aos demais agentes, funções adicionais para trabalhar em forma de rede, como: funções de associação de endereços físicos com nomes simbólicos, registros de agentes e ou outros serviços oferecidos e/ou requisitados pelo agente, melhoria nos serviços de comunicação, como por exemplo, encaminhamento e difusão [Loss03].

4.6.5. Ontologia

Para que agentes possam trocar informações é necessário não somente um formato comum e uma linguagem de comunicação, mas também um modelo comum do mundo, ou seja, é necessário que os agentes estejam de acordo com uma ontologia comum.

Entende-se por ontologia uma especificação explicita de uma conceitualização. Por conceitualização entende-se uma visão simplificada e abstrata do mundo, onde se representam formalmente os objetos, conceitos e outras entidades que existem em alguma área de interesse, e relações entre elas [Macedo01]. Desta forma, a ontologia fornece um vocabulário de representação para um domínio específico e, ainda, um conjunto de definições e axiomas que restringem o significado dos termos nesse vocabulário, permitindo uma interpretação consistente dos dados existentes no vocabulário.

Um sistema que adota uma ontologia comum garante a **consistência**[10] e **compatibilidade**[11] da informação presente neste sistema.

Portanto, uma comunicação ideal entre os agentes se dá com a união de uma linguagem de comunicação comum, um conteúdo de linguagem comum e uma ontologia partilhada, pois desta maneira os agentes conseguem se comunicar na mesma forma, na mesma sintaxe e com o mesmo entendimento do mundo.

4.7 Tolerância a Falhas em Sistemas Multi-Agentes Móveis

Como visto anteriormente, um agente móvel pode ser definido como um programa autônomo de computador que viaja através de uma rede de computadores heterogêneos.

Os agentes móveis são criados em um ambiente de execução, podendo transportar seus estados e códigos. Esta característica permite que estes agentes possam ser executados em outro ambiente que não seja aquele no qual foram criados [Fioreze03]. Caso ocorram falhas nesses ambientes de execução, pode ocorrer perda parcial ou total de um agente. Infelizmente, tanto os *softwares*, quanto os *hardwares* são propícios à falhas. Portanto, qualquer ambiente de execução é potencialmente sujeito à falhas.

Segundo [Fioreze03], três defeitos podem afetar um agente:

- **Colapso do Próprio Agente** – Neste caso, ocorre a perda completa do agente. Porém, quase sempre esta falha não afeta o ambiente de execução ou o

[10] Uma mesma expressão tem o mesmo significado para qualquer agente [Macedo01].
[11] Um conceito é designado pela mesma expressão em qualquer agente [Macedo01].

computador onde ocorre a execução. Nos demais defeitos, o ambiente de execução entra em colapso, direta ou indiretamente, podendo levar a perda total do agente.

- **Colapso no Ambiente de Execução** – Se o ambiente de execução entrar em colapso, obviamente, o agente móvel será perdido.

- **Colapso no Computador** – Também implica na perda do agente.

Os agentes móveis podem ser executados em pequenas redes, ou em grandes redes, tal como a Internet. As grandes redes são ambientes propícios a atrasos, falhas de comunicação e demais problemas existentes em redes de computadores. Desta forma, torna-se necessário à adoção de técnicas de tolerância a falhas que permitam, por exemplo, informar ao proprietário de um agente criado, se o mesmo foi perdido ou se a execução do agente está atrasada devido a problemas no meio de comunicação, evitando desta forma problemas como: i) **execuções simultâneas do mesmo agente** e ii) **situação bloqueante.** No primeiro caso, diante da hipótese de que o agente tenha se perdido, o proprietário recria o agente lançando-o novamente na rede, o que implicará em execuções simultâneas do mesmo agente em determinados terminais, caso o mesmo não tenha sido perdido, possivelmente encontrando-se atrasado. No segundo caso, o proprietário fica aguardando pelo término da execução do seu agente, mas se este foi perdido, ocorrerá uma espera infinita (situação de bloqueio).

As técnicas de tolerância a falhas em sistemas de agentes móveis permitem que problemas como estes sejam eliminados ou reduzidos. Elas podem garantir a chegada do agente ao seu destino ou, pelo menos, notificar ao proprietário do agente sobre problemas ocorridos com o mesmo, evitando desta forma as incertezas quanto à integridade do agente.

4.7.1 Técnicas de Tolerância a Falhas

Nesta seção são apresentadas, de forma sucinta, três técnicas de tolerância a falhas aplicadas em sistemas de agentes móveis. São elas [Fioreze03]:

- **Técnica Baseada em *Checkpointing*** – Ao migrar para um computador A, o agente leva consigo informações sobre seu estado atual e seu código, as quais são

armazenadas, de forma segura, pelo computador A, antes do início da execução do agente. Caso o computador A falhe, estas informações podem ser recuperadas após a restauração do mesmo, permitindo que o agente continue sua execução. Porém, esta técnica é bloqueante, visto que o dono do agente fica no aguardo do final da execução que não ocorrerá até que o computador A seja recuperado.

- **Técnica Baseada em Replicação –** Nesta técnica, o problema do bloqueio é eliminado. Várias cópias do agente, em um determinado estágio, são feitas em vários lugares distintos. Caso um lugar falhe, o agente pode ser executado em outro lugar. Por prevenção, estes lugares são normalmente localizados em computadores diferentes. Entretanto, esta técnica é propensa a violar a propriedade dos agentes móveis executarem somente uma vez, que é conhecida como *exactly-once property*.

- **Técnica Baseada em Replicação com Votação –** Similar a técnica de replicação, porém, uma ordem entre os lugares pertencentes ao mesmo estágio, é definida por uma propriedade associada a cada lugar. Inicialmente, cada estágio seleciona o lugar com a maior prioridade sendo este denominado de **trabalhador**. Os demais lugares são denominados de **observadores** e tem como função monitorar a disponibilidade do trabalhador. Caso o **trabalhador** entre em colapso, um novo **trabalhador** é selecionado entre os **observadores**, por eles mesmos, através de um protocolo de eleição. Para preservar a propriedade dos agentes móveis executarem somente uma vez em um determinado estágio, um **trabalhador** só finaliza uma transação se houver comum acordo entre a grande maioria dos lugares envolvidos na execução de um agente móvel.

4.8 Uso de Agentes no Cenário de Empresas Virtuais

Como visto, dentre as principais razões para prover um comportamento cooperativo em sistemas multi-agentes, se encontra a necessidade de "unir" competências, em termos de recursos e informações dos agentes, para a solução do problema.

Uma cooperação pode ser de vários tipos e ser realizada em vários níveis, de acordo com as capacidades das entidades envolvidas. A cooperação está fortemente ligada à aplicação específica, pois a sua base de trabalho implica na procura coletiva de um objetivo comum (a resolução de um problema naquele domínio). A forma como a **cooperação** se desenrola depende, obviamente, de como o problema a resolver é decomposto e distribuído.

A **comunicação** fornece a base necessária para a realização da **cooperação** entre múltiplos agentes. A comunicação entre os vários agentes resolvedores de problemas permite a exploração em comum dos seus recursos e conhecimentos próprios, tornando possível o trabalho em paralelo de diferentes partes do problema, e a obtenção mais rápida da resolução do problema [Hubner03].

Uma Empresa Virtual se encaixa no perfil dos **SMAs**, pois pode ser considerada como um nó na federação, que mantém sua autonomia local nos dados e define um conjunto de esquemas exportáveis pelo qual os dados são disponibilizados a outros nós específicos. Além disso, todo nó pode importar dados de outros nós através dos seus esquemas importáveis, tendo acesso aos dados deles de acordo com as permissões de acesso predefinidas. Como conseqüência desta facilidade geral de interação, a aproximação permite não só uma cooperação, mas também suporta a coordenação dos nós federados (grupo de empresas envolvidas) para realizar uma tarefa, enquanto são preservadas a autonomia local e independência de cada nó [Rabelo00].

No entanto, agir cooperativamente significa fornecer toda a informação requerida por um outro agente. Se este agente é “externo”, ele pode vir a usar esta informação contra aquele que a forneceu. Por este motivo, no cenário de Empresas Virtuais, durante a negociação ou em algumas fases da execução de certas tarefas, uma empresa não pode estar completamente aberta à outra empresa, uma vez que informações confidenciais estão normalmente circulando [Rabelo01].

Como visto, os **SMAs** permitem resolver problemas de grande proporção ao empregar vários agentes autônomos e cooperantes, garantem a interoperação de diferentes sistemas legados e oferecerem soluções que usam eficientemente fontes de informação espacialmente distribuídas. Já o paradigma de **Agentes Móveis** se mostra adequado ao cenário de **Busca** e **Seleção** de parceiros para a formação de uma **EV**, pois garante uma maior agilidade na apresentação das

oportunidades de negócios ao grupo de empresas e uma maior eficiência na formação e análise das possíveis empresas virtuais a serem constituídas [Schmidt03].

5. PLATAFORMAS PARA EMPRESAS VIRTUAIS

5.1 Introdução

Este capítulo tem por objetivo apresentar a **Plataforma FT-SALE/SMAM**. Ou seja, este capítulo apresenta o modelo conceitual para o processo de inserção de técnicas de tolerância a falhas no ambiente de uma Empresa Virtual formada sob um **Sistema de Multi-Agentes Móveis** (**SMAM**). O modelo proposto segue as especificações gerais dos serviços de tolerância a falhas do **CORBA** (**FT-CORBA**), com a introdução de algumas funções que visam adaptar a plataforma proposta ao ambiente de agentes híbridos.

Para melhor compreensão da plataforma proposta, são apresentadas duas plataformas que atuam no contexto das Empresas Virtuais: a **Plataforma de Busca e Seleção de Parceiros** [Schmidt03], desenvolvida baseada no paradigma de agentes móveis, que garantem maior agilidade e eficiência no processo de busca e seleção de parceiros para a formação destas empresas, e a **Plataforma FT-SALE** [Lima01a], cuja proposta é adequar as técnicas de tolerância a falhas, baseadas nas especificações **FT-CORBA**, aos ambientes de Empresas Virtuais.

A abordagem que está sendo proposta neste trabalho tem como intuito superar as limitações técnicas de tolerância a falhas existentes na **Plataforma de Busca e Seleção de Parceiros**, que se devem principalmente ao fato da existência de um único **Gerenciador de Grupo**. Portanto, a principal vantagem da união destas plataformas é a utilização das técnicas de tolerância a falhas na melhoria da confiabilidade do processo de busca e seleção de parceiros. Além disto, a inserção de um **Sistema Multi-Agentes Móveis** no contexto da **Plataforma FT-SALE** abre outras oportunidades que podem ser exploradas em trabalhos futuros.

A descrição prossegue, identificando quem são as entidades presentes em ambas as plataformas e quais são os seus papéis durante todas as fases do processo de prevenção, tratamento e recuperação de falhas em Empresas Virtuais sob a **Plataforma FT-SALE/SMAM**.

5.2 Plataforma de Busca e Seleção de Parceiros em Empresas Virtuais

5.2.1 Componentes

Nesta seção é feita uma breve apresentação dos agentes e demais entidades componentes do sistema de busca e seleção de parceiros para formação de **EVs** originalmente proposto em [Schmidt03], o qual será base para a plataforma proposta neste trabalho. São apresentadas, de forma sucinta, as definições das funções e o comportamento de cada agente durante o processo de busca e seleção.

O Sistema de Busca e Seleção de Parceiros está sob uma plataforma híbrida, a qual combina agentes estacionários e móveis. Os **agentes estacionários** possuem as funções de **coordenadores** do processo de busca e seleção e representantes das empresas de uma Organização Virtual (**OV**) (atividades que exigem um nível de segurança das informações que os agentes móveis não podem fornecer). Já os **agentes móveis** são utilizados para explorar sua habilidade de deslocamento pela rede, quer para **buscar/levar informação**, quer para **interagir localmente** com outros agentes/sistemas.

O Sistema de Busca e Seleção de Parceiros possui uma arquitetura híbrida de agentes[12], composta por agentes de dois tipos:

- **Agentes estacionários** – Ficam localizados nas plataformas de cada empresa e realizam comunicação com agentes móveis que se deslocam até aquelas plataformas. Esta comunicação é realizada para obter os dados que servirão de base para a seleção de parceiros;

[12] O papel de cada agente existente no sistema de **Busca e Seleção** é descrito em detalhes em [Schmidt03], assim como o papel do *Broker* humano.

- **Agentes móveis** – Deslocam-se entre as empresas e a cada empresa "visitada" coletam e armazenam informações, podendo, inclusive, analisá-las localmente e solicitar reavaliação dos dados recebidos.

A Figura 5.1, extraída de [Schmidt03], ilustra as entidades componentes do sistema de busca e seleção, as quais são brevemente apresentadas a seguir, de acordo com as definições encontradas nesta mesma referência.

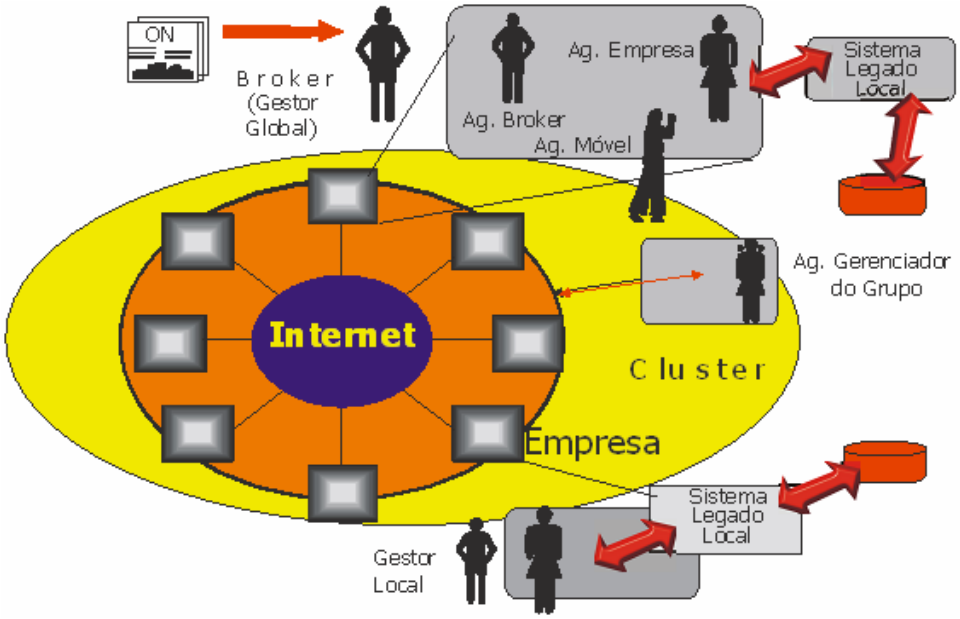

Figura 5.1: Cenário representando o *cluster* e as entidades presentes no Sistema de Busca e Seleção.

- **Gestor Global** – É o representante que recebe a Oportunidade de Negócio (**ON**). O Gestor Global é o responsável pela condução dos processos de busca e seleção para uma Empresa Virtual (**EV**).

- **Agente Gerenciador do Grupo** – É um agente estacionário e persistente responsável pelo controle de grupo (*membership*) no grupo de empresas. Todos os membros componentes do grupo são cadastrados em uma lista, a qual possui as informações necessárias no momento da seleção das empresas que farão parte da nova **EV**. Este agente deve registrar a entrada e/ou a saída das empresas na **OV**, para que a lista se mantenha sempre atualizada, possibilitando ao mesmo fornecer

a relação atualizada de quais empresas compõem o grupo mediante solicitação efetuada por algum membro da **OV**.

- **Agente Empresa (AE)** – É um agente estacionário e persistente que tem a função de representar a empresa onde é criado, permitindo a obtenção e a disponibilização de informações. O agente interage com o repositório de dados local e/ou com o gestor ou especialista responsável da empresa para a solicitação dos dados. Caso uma nova oportunidade de negócio seja anunciada ao Agente Empresa, este determina a criação de um agente ***Broker***, que será responsável pela coordenação dos processos necessários à formação da **EV** voltada para atender a uma nova oportunidade.

- **Agente *Broker* (AB)** – É um agente estacionário criado exclusivamente para cada nova oportunidade de negócio e seu tempo de vida termina no momento da dissolução da EV. O *Broker* é o agente coordenador do processo de busca e seleção a ser feito pelo sistema de agentes.

- **Agentes Móveis *Broker* (AM)** – Criados pelo agente *Broker* para auxiliar no processo que o Agente *Broker* está coordenando. Carregam informações sobre a **ON** em questão e obtém dados nos locais por onde passam com a finalidade de auxiliar na escolha das empresas com capacidade para integrar a EV. Assim como seu criador, é um agente temporário. A tarefa a ser desempenhada pelo agente móvel é modelada como uma missão, obtida em uma **biblioteca de missões**. Este agente pode ter diferentes missões:

    - **Agente Mensageiro** – Quando possui a missão de transportar e disponibilizar as informações de anúncio aos sistemas de agentes dos locais visitados;

    - **Agente Pesquisador – Q**uando possui a missão de percorrer os vários sistemas de agentes solicitando, obtendo e transportando informações;

- **Agente Negociador –** Quando fazendo uso das informações obtidas vai-se em busca de melhores resultados que aqueles fornecidos inicialmente.

Entre os agentes apresentados acima, o que requer mais atenção para a proposta deste trabalho é o Agente Gerenciador de Grupo, responsável pelo controle de *membership.* Este agente cria um ponto de vulnerabilidade no sistema, visto que todo o controle de grupo é feito em um único ponto do sistema, o que não é aconselhável, pois um único ponto de controle torna o sistema susceptível à falhas.

5.2.2 Formação de uma Empresa Virtual

O Sistema de Busca e Seleção de Parceiros para Empresas Virtuais possui uma arquitetura híbrida composta por **agentes estacionários** e **agentes móveis,** e, ainda, sob a perspectiva do fluxo de controle, passa por uma etapa de interação com um **agente humano** (*Broker* humano) responsável por analisar informações de considerável importância para a formação da Empresa Virtual. O **agente humano** toma as decisões relevantes que não podem ser deixadas para o sistema decidir por si só.

Quando uma nova **ON** (**Oportunidade de Negócio**) é identificada, seja através de uma descoberta feita por um membro do grupo ou através de uma solicitação feita pelo cliente, uma nova empresa virtual deve ser criada com a finalidade de atender a esta nova **ON**.

O contato inicial é estabelecido entre o cliente e o representante de alguma empresa membro do grupo. Este representante (*Broker*) passa a ser o responsável por identificar os parceiros ideais (cujo perfil se encaixa melhor em relação a **ON** em questão) dentro do grupo de empresas e selecioná-los para a formação da nova Empresa Virtual. A empresa representante de uma **ON** pode ser qualquer uma das empresas do grupo.

Após o levantamento das característica da **ON**, inicia-se a fase de **anúncio**. O **anúncio** é realizado em dois momentos, para que seja direcionado às empresas que realmente possam e queiram participar da seleção. Nesta fase, as características do produto desejado pelo cliente são divulgadas às empresas participantes do grupo. O objetivo é comunicar que uma **EV** precisa ser

formada, pois há uma tarefa a ser realizada, provendo informações para que a empresa possa decidir em candidatar-se ou não [Schmidt03].

Dentro de um grupo, existem empresas com características diferentes, o que torna desnecessário informar a todas EVs sobre a presença de uma dada **ON**. Isto implica na realização de uma pré-seleção dos destinatários do anúncio, conhecida como **Restrição de Audiência**.

Assim que for criado por um especialista de uma Empresa **X** qualquer, o **Agente Empresa** deve efetuar o registro da empresa **X** junto ao **Gerenciador do Grupo,** possibilitando aos demais participantes tomar conhecimento da existência da empresa **X**.

Após a fase de **Anúncio**, são emitidas as **Respostas ao Anúncio.** Estas respostas servem para definir a relação das empresas interessadas em participar da **EV**. De posse desta relação, parte-se para a fase de **Busca de Informações** referentes às empresas interessadas. Esta procura é feita com o levantamento de dados das empresas, principalmente dados referentes à sua capacidade de produção, como disponibilidade, prazo de entrega dos produtos e ainda, dados de custo. O principal objetivo da fase de **Busca de Informações** é determinar quais são as empresas responsáveis por quais tarefas.

Segundo [Schmidt03] "uma **tarefa** a ser atendida é composta por várias "**subtarefas**" a serem distribuídas entre os participantes da **EV**. Assim, uma empresa pode assumir a realização de mais de uma destas **subtarefas** e, para isso, as informações presentes em **Dados da Tarefa** visam especificar quais itens o candidato pretende realizar. Conforme uma **subtarefa** tenha vários interessados em realizá-la, os "vencedores" são determinados com base nas informações coletadas junto aos candidatos. Isto possibilitará a escolha da empresa que melhor atenda a requisitos como qualidade, prazo ou custo".

A fase seguinte é a de **Seleção**. Nesta fase, as informações coletadas durante a fase de **Busca** servirão de base para identificar as empresas com melhores condições de atender a ON e então, definir quais são as tarefas de cada empresa participante. Ainda, pode ser necessária a realização de uma fase de **Negociação**, que é a barganha por melhores resultados em vista daqueles informados pelas empresas na fase de **Busca**.

A Figura 5.2 ilustra as fases principais do processo de Busca e Seleção.

Figura 5.2: Fases principais do processo de Busca e Seleção.

Terminada a fase de **Seleção**, o *Broker* passa a ser o coordenador da **EV** e tem como função garantir que a **EV** cumpra sua tarefa. Ele é o responsável pelo andamento do processo. O *Broker* deve observar os prazos para a tarefa, lidar com eventuais atrasos de algum integrante e responder pela empresa virtual criada, inclusive nas questões de contrato.

O processo de **Busca** e **Seleção** encerra-se no momento que é realizada a **Concessão de Contrato**. Esta fase ocorre logo após a identificação das empresas vencedoras, através da análise dos dados coletados nas fases anteriores. A agilidade neste processo é de fundamental importância para atender às necessidades dos clientes, e ainda, para que a empresa possa atingir suas metas de prazo de entrega e aumentar sua competitividade no mercado [Schmidt03].

A proposta feita em [Schmidt03] é automatizar parte deste processo, fazendo com que as decisões tomadas pelo responsável, o *Broker*, sejam auxiliadas por dados coletados e pré-processados por agentes móveis e estacionários. Por exemplo, a formação de grupos possíveis entre as empresas candidatas a uma dada tarefa.

No sistema de Busca e Seleção, explora-se o uso de agentes móveis, com capacidades de execução em plataformas heterogêneas. Para isso, são adotados alguns padrões de portabilidade e

interoperabilidade[13], conforme discutido no capítulo anterior. Para a troca de mensagens entre os agentes, são usadas as linguagens de comunicações **KQML** e **XML**.

Uma vez concluído o processo de **Busca e Seleção,** a **EV** passa para o funcionamento propriamente dito, começando sua fase de **Operação**. A Figura 5.3, extraída de [Wangham03b], ilustra o cenário para a seleção de Empresas Virtuais.

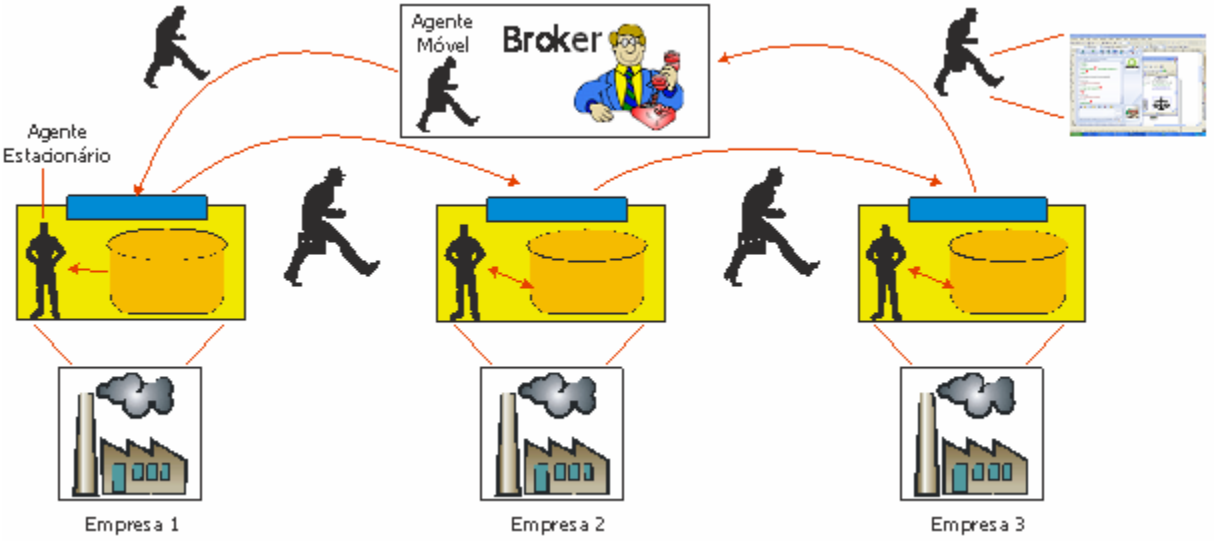

Figura 5.3: Cenário para Seleção de Empresas Virtuais.

Os agentes móveis terão como função deslocar-se pelas empresas em busca de informações de **Evs.** Após obter as informações necessárias, o agente móvel deve retornar para o ***Broker*** trazendo consigo as alternativas de empresas virtuais, classificadas a partir de indicadores de desempenho previamente definidos para avaliá-las.

No modelo proposto em [Schmidt03] a **Restrição de Audiência** é realizada com o auxílio do **Gerenciador de Grupo** através de uma requisição da lista de membros. Durante a requisição, o perfil da **ON** é informado ao Gerenciador, que verifica quais são as empresas adequadas a este perfil.

Em síntese, os seguintes passos são realizados pelo Sistema de Busca e Seleção de Parceiros durante a construção de uma EV [Schmidt03]:

[13] São considerados aspectos de segurança e ainda, a possibilidade que vários *Brokers* existam no sistema, cada um coordenando um processo de formação de EV.

1. Dada uma nova **ON,** o **Agente Empresa** (**AE**) determina a criação de um Agente *Broker*;

2. O **AE** solicita ao Gerenciador de Grupo a relação das empresas registradas, podendo então verificar quais devem ser consultadas, e repassando ao agente ***broker*** (**BR**);

3. O **BR** analisa a lista de membros da **OV** fornecida pelo **Gerenciador de Grupos**;

4. O **BR** identifica as empresas candidatas, que possuem o perfil para atender a nova **ON**, e distribui o pedido entre estes empresas;

5. As empresas selecionadas pelo **BR** recebem o anúncio da nova **ON** e após análise, caso tenham interesse e condições técnicas e temporais (dentro do prazo) para participar da nova **ON**, enviam suas propostas;

6. Um **AM** é enviado para "dialogar" com cada **AE** das empresas interessadas, com a finalidade de apurar o real interesse da empresa em participar da nova **ON**. Se constatado o interesse, o **AM** solicita ao **AE** informações relacionadas às capacidades técnicas (retiradas de um banco de dados local) e aos prazos de entrega. Em seguida, pede a confirmação de preços e prazos junto ao supervisor da empresa do **AE**. Caso contrário, ou seja, se não houver interesse em participar, o **AM** simplesmente desloca-se para a próxima empresa (ver Figura 5.3);

7. O **AM** trafega pelas empresas candidatas coletando e armazenando informações destas empresas, referentes a nova **ON**;

8. Após "visitar" todas as empresas candidatas, o **AM** retorna ao *broker* trazendo as informações coletadas, bem como diferentes alternativas de **EV** geradas (coalizões).

9. O *broker* escolhe a **EV** mais adequada para atender a nova **ON**;

10. As empresas selecionadas são notificadas, pelo *broker*, sobre o resultado final. Isto é feito através do envio de uma mensagem que corresponde ao acordo de negócio.

5.2.3 Componentes de uma Organização Virtual

Em uma Organização Virtual (**OV**), o termo Empresa Virtual (**EV**) refere-se a todo grupo de empresas físicas (**E**) que se unem com a finalidade de executar um serviço e ou um produto final. Vale a pena lembrar, que estas empresas estão conectadas através de uma rede de computadores, quase sempre com arquiteturas e sistemas heterogêneos.

A Figura 5.4, também extraída de [Schmidt03], ilustra o conceito de uma **OV** composta por um grupo de **EV** e algumas **E** participantes. Cada **EV** é responsável pela finalização de seu produto, não tendo necessariamente qualquer relação com as demais empresas virtuais da **OV**. Pode ocorrer ainda que uma determinada empresa **E**, componente de uma **EV-1**, faça parte também do grupo de empresas de uma **EV-2**, pertencente a mesma **OV** onde se encontra a **EV-1**.

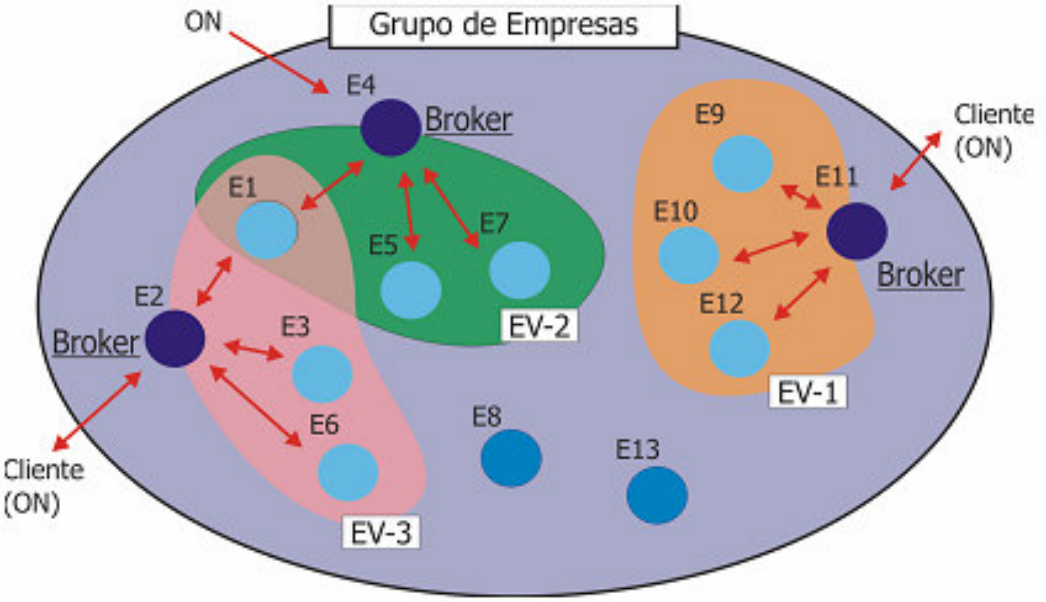

Figura 5.4: Cenário de uma Organização Virtual.

Como exemplo de uma **OV**, temos o Grupo de Empresas da figura, onde existem 3 **Evs**. A **EV-1** é composta pelas empresas **E10**, **E9**, **E11** e **E12**; a **EV-2** é composta pelas empresas **E1, E4, E5** e **E7**; e a **EV-3** é composta pelas empresas **E1, E2, E3** e **E6**, onde cada uma possui o seu próprio *Broker*. Existem ainda as empresas **E8** e **E13** que, embora façam parte da **Organização Virtual**, não se encaixam no perfil das necessidades das **Nos** que originaram as **Evs** existentes no exemplo. Neste caso, estas empresas não participam da composição da **EV-1**, nem da **EV-2** e nem mesmo da **EV-3**.

Com base no modelo proposto em [Schmidt03], cabe notar que cada **EV** possui um *Broker* (elemento de organização) responsável por decidir sobre a composição da empresa virtual que melhor se adapta às necessidades de uma nova **ON**. Desta forma, várias empresas virtuais podem ser criadas dentro de uma **OV**, dividindo a carga de processamento entre os diversos *Brokers* existentes.

Com a existência de mais de um *Broker* dentro de uma Organização Virtual, as atividades de tomada de decisão se encontrarão descentralizadas, o que contribui para a construção de confiança entre os integrantes da **EV**.

Entretanto, a existência de mais de um *Broker* pode acarretar em complexidade na operação e em um grande volume de mensagens trocadas, visto que o *Broker* deve possuir uma relação de quem são os outros membros, onde se localizam e quais são as suas habilidades.

No caso de existir mais de um *Broker* para o mesmo grupo, torna-se necessário que cada empresa adquira informações sobre os outros membros do grupo. E ainda, quando acontece a inclusão de uma nova empresa, esta precisa buscar informações sobre os membros do grupo e em seguida as informações sobre a nova empresa devem ser incluídas na relação de participantes do grupo. Quando ocorre uma exclusão de uma empresa já existente, as informações referentes a esta empresa devem ser excluídas da relação de participantes do grupo.

Em [Schmidt03], a forma encontrada para solucionar este problema foi definir um **Gerenciador de Grupo** onde são feitos os registros e as exclusões dos integrantes do grupo (Controle de *Membership*). Desta forma, a empresa que deseja ingressar no grupo para buscar informações sobre os demais membros, precisa saber apenas quem é e onde está o **Gerenciador de Grupos**.

Entretanto, como visto no Capítulo 2, uma das principais características de uma Empresa Virtual é a localização geográfica das empresas que a compõem, que muitas vezes são diferentes e ou até mesmo distantes. Por este motivo, a comunicação entre estas empresas se dá através de uma rede de larga escala, em ambientes como a Internet. Todavia, isto torna difícil o estabelecimento de um único domínio para gerenciar um grupo de objetos.

5.2.4 Considerações Sobre a Implementação do Modelo de Busca e Seleção

Na implementação da plataforma de Busca e Seleção de parceiros para **EVs** [Schmidt03] foram utilizadas duas plataformas de agentes, sendo elas *Aglets* e *Massyve*:

- **A plataforma *Aglets*** – Foi usada para suportar agentes móveis no sistema, podendo dar suporte a todas ações necessárias a um agente móvel, desde sua criação, seu envio para locais remotos e conseqüente recebimento. Possibilita toda a comunicação entre os agentes, até o momento do seu encerramento.

- **A plataforma *Massyve*** – Visou sobretudo, validar o modelo de Busca e Seleção perante os aspectos de interoperabilidade. Ou seja, desejou-se assumir uma realidade na qual cada empresa do *cluster* teria o seu sistema legado particular, usualmente heterogêneo, com o qual é necessário interagir.

Os agentes da plataforma *Aglets* implementam o sistema de busca e seleção de parceiros, que trabalha "sobre" os sistemas legados das empresas, cada qual representado na plataforma como um agente *Massyve*.

No entanto, visando dar ainda maior "realismo" ao sistema de Busca e Seleção, foi sugerida a integração deste a um sistema maior que cobre outras fases do ciclo de vida de uma **EV**, tais como a Configuração e Operação/Evolução. O sistema escolhido para tal foi o **SC2** (*Supply Chain Smart Coordination*) [Rabelo02], por tratar de um Sistema Multi-Agente implementado na plataforma *Massyve*. O sistema de Busca e Seleção pode ser visto, então, como um módulo/componente "agentificado" do **SC2** para uma função que até então não era suportada. O sistema **SC2** foi implantado em cada um dos "nós"/empresas, podendo interagir com seus módulos internos, com módulos de uma dada plataforma para **EV** e com os sistemas legados de cada empresa.

Para resolver o problema da interoperabilidade, fez-se necessária a interação entre os agentes *Aglets* (do sistema proposto) e os agentes *Massyve* (do **SC2**). O agente *Massyve* é

responsável pela interação com sistemas legados, ou seja, ele trata das questões particulares a cada plataforma onde está situado, realizando comunicação com agentes *Aglets* nos casos de requisição e fornecimento de informações. Porém, este fato ocasionou um problema de interoperabilidade entre agentes criados em plataformas distintas e ainda, de agentes com sistemas legados, visto que a plataforma *Massyve* está escrita em C/C++, enquanto que *Aglets* está escrita em Java.

Para garantir a interoperabilidade entre estes agentes, o modelo proposto em [Schmidt03] utiliza o CORBA como *middleware*. O objetivo do CORBA é prover facilidades de comunicação em ambientes heterogêneos como os encontrados na Internet. Desta forma: "...cada plataforma apresenta uma interface que forma uma espécie de contrato entre clientes de serviços e o objeto servidor destes. Estas interfaces descrevem todos os serviços (métodos e atributos) que serão exportados e visíveis para os potenciais clientes, descrevendo a forma como esses serviços podem ser acessados. Uma vez que a comunicação entre agentes de diferentes plataformas é realizada através de um ORB de comunicação, este traz a vantagem de garantir a heterogeneidade das plataformas envolvidas (diferentes Sistemas Operacionais e diferentes Linguagens de Programação)" [Schmidt03].

5.3 Arquitetura FT-SALE

Como visto no Capítulo 3, a tolerância a falhas no **CORBA** é obtida através da replicação de objetos, técnicas de detecção e recuperação de falhas, aumentando assim a confiabilidade no sistema e a disponibilidade dos recursos.

No entanto, as especificações de tolerância a falhas no **CORBA** (**FT-CORBA**), apresentam algumas dificuldades ao serem aplicadas em redes de larga escala. As especificações **FT-CORBA** são limitadas em termos de detecção de falhas e gerenciamento de replicação (*membership*), que são aspectos essenciais para dar confiabilidade na comunicação entre objetos em um ambiente amplamente distribuído e heterogêneo como o ambiente de uma Empresa Virtual.

Estas limitações incentivaram o uso da plataforma **FT-SALE** [Lima01a] para a adaptação de técnicas de tolerância a falhas no modelo proposto em [Schmidt03], pois no modelo **FT-**

SALE é feita uma adaptação às especificações **FT-CORBA**, sem modificações nas interfaces do padrão **CORBA**, utilizando como base a solução dada pelo **GrouPac 2.0**[14] [Laurindo01] para a implantação de técnicas de tolerância a falhas em redes de larga escala, juntamente com a plataforma **SALE** [Rabelo01], cuja arquitetura integra serviços com diferentes tipos de garantia (desempenho, confiabilidade e segurança), se tornando adequado para Empresas Virtuais.

A seguir serão detalhadas as principais características da plataforma FT-SALE.

5.3.1 Domínio de Tolerância a Falhas

Como visto no Capítulo 3, o conceito de **domínios de tolerância a falhas** foi introduzido para facilitar o gerenciamento de aplicações muito complexas, as quais seria inviável manipular através de uma única entidade.

Segundo a definição de domínio de tolerância a falhas nas especificações **FT-CORBA**, são criadas réplicas de um objeto, as quais são gerenciadas como um grupo de objetos. Existe ainda uma referência para cada objeto e o grupo todo tem uma referência de grupo conhecida como **IOGR** (*Interoperable Object Group Reference*). A **IOGR** é utilizada por um cliente na solicitação de um serviço a um objeto replicado.

A existência de réplicas, suas falhas e a recuperação de falhas destas réplicas é transparente ao cliente, independente do tipo de replicação adotada.

Um conjunto de propriedades de tolerância a falhas é associado a cada grupo de objetos, podendo-se definir propriedades que se aplicam ao grupo todo ou a um tipo específico de objetos dentro do grupo [Lima01a].

A Figura 5.5, extraída de [Lima01b], mostra a composição de um domínio de tolerância a falhas.

[14] Sistema que fornece suporte a grupo e tolerância a falhas na arquitetura CORBA.

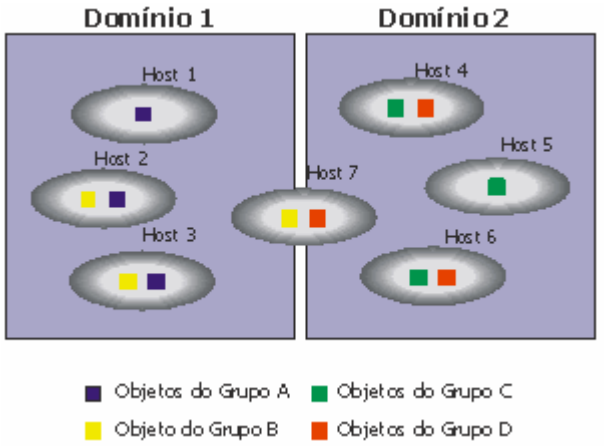

Figura 5.5: Domínio de tolerância a falhas.

A figura ilustra 2 domínios de tolerância a falhas, os quais possuem grupos de objetos replicados. Cada domínio é composto por várias máquinas (*hosts*), nas quais se localizam os objetos. Nos domínios 1 e 2, estão localizados três *hosts*, sendo que cada um destes *hosts* possuem objetos pertencentes a um ou mais grupos. Como exemplo, temos que no domínio 1 o *host* 1 possui um objeto do grupo A, o *host* 2 e 3 possuem objetos dos grupos A e B e o *host* 7 possui objetos dos grupos B e D, sendo que estes pertencem a grupos de diferentes domínios. Portanto, pode-se considerar que o *host* 7 pertence aos dois domínios de tolerância à faltas.

A especificação **FT-CORBA** define que um objeto pertence apenas a um grupo de domínio de tolerância a falhas, porém grupos de objetos podem ter objetos em diversas máquinas.

5.3.2 Definição de Grupos no FT-SALE

De acordo com [Bessani02], "um **grupo** é uma coleção de objetos ou processos. O **Suporte de Comunicação de Grupo** (**SCG**) é responsável pelo gerenciamento e comunicação entre membros de um grupo".

Em sistemas distribuídos, em especial em SCG, é de fundamental importância determinar, em tempo de execução, que processos são membros de um grupo. Esta lista de membros pertencentes ao grupo é chamada de *membership*[15] (pertinência) ou visão. O ***membership*** de um grupo pode ser:

[15] Não confundir com o controle de grupo das Empresas Virtuais.

- ***Membership* Estático** – Neste caso, os membros do grupo são conhecidos previamente, em tempo de compilação.

- ***Membership* Dinâmico** – Processos entram e saem de grupos, o que representa uma pertinência que evolui, possuindo um número de membros variável em função do tempo.

Para prover flexibilidade, os **SCG** trabalham com ***membership*** dinâmico. Existem três razões pelas quais o conjunto de membros de um grupo pode mudar. São elas:

- **Requisitos da Aplicação** – Algum requisito funcional da aplicação atuando sobre o **SCG** deve implicar nas alterações de *membership*. Um exemplo típico é o de uma videoconferência, onde membros podem entrar após o início da mesma, ou podem sair antes de seu término;

- **Falhas de Processos** – Caso sejam detectadas falhas em processos, estes devem ser removidos de grupos;

- **Redimensionamento do Sistema** – Grupos podem crescer ou diminuir segundo demandas. No caso de uma alteração significativa (crescer ou diminuir muito) torna-se necessário um redimensionamento de seus recursos.

Ao disponibilizar o *membership,* o **SCG** torna a lista de membros do grupo visível através do **Serviço de *Membership* de Grupo** (**SMG**). Em *membership* dinâmico, o **SMG** é responsável pelas mudanças e alterações na lista de membros do grupo.

Se as falhas de processos forem inexistentes no ambiente, as operações citadas seriam suficientes para que o **SMG** tivesse uma visão correta do grupo. Porém, na maior parte dos sistemas existentes, não se pode desconsiderar as falhas em processos. Desta forma, através de algum **mecanismo de detecção de falhas**, o **SMG** deve descobrir os processos faltosos de modo a excluí-los da lista de pertinência, mantendo assim um *membership* consistente.

O modelo **FT-SALE** utiliza como base para a identificação de grupos dentro do cenário de uma **EV** as especificações de grupos de domínio do **GrouPac**.

O processo de produção de um determinado resultado (produto ou serviço) dentro do cenário de Empresas Virtuais é conhecido como **DBP** (*Distributed Business Process*). Um **DBP** é executado por uma série de diferentes fornecedores, conectados entre si por uma rede de computadores e representado por um grupo de **BPs** (*Business Processes*) formado dinâmica e temporariamente e que no conjunto, concretizam o produto final da Empresa Virtual.

Para cada **DBP**, um membro (nó) da Empresa Virtual é eleito como "disparador" do negócio (coordenador) e os outros como seus fornecedores. A Figura 5.6, extraída de [Lima01b], ilustra o conceito de um **DBP**, onde a empresa 2 (**nó 2**) atua como coordenadora (**EV** *coordinator*), e as empresas 1, 3 e 4 (**nós 1, 3** e **4**) como seus fornecedores diretos.

Cada empresa é responsável por alguns **BPs,** os quais representam o valor agregado de cada empresa na cadeia de produção e as suas respectivas Atividades da Empresa (***Enterprise Activities*** – **Eas**). Assim como na plataforma **FT-SALE** [Lima01a], cada **BP** é constituído por uma ou mais operações de manufatura "de baixo nível", as quais são representadas pelas **Eas**.

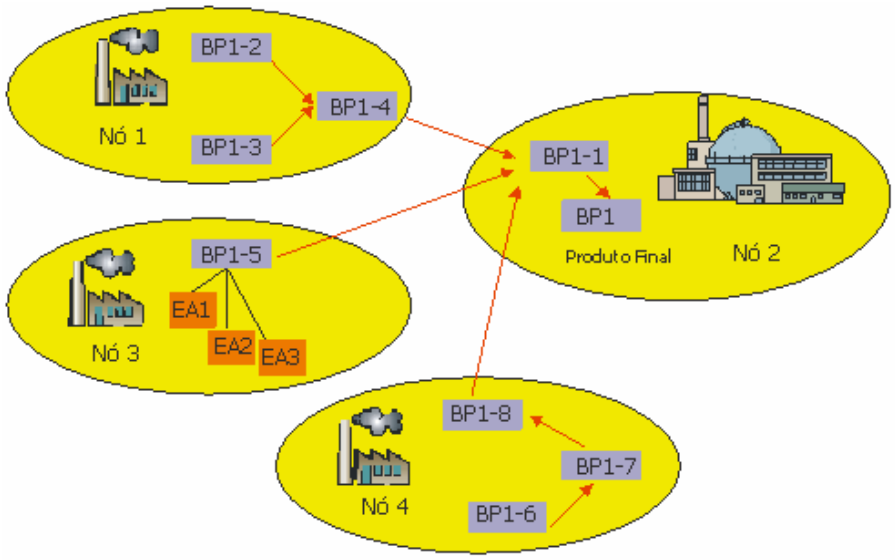

Figura 5.6: Cenário de uma Empresa Virtual.

Na Figura 5.6, a empresa 1 **(fornecedora)** executa 3 **BPs (BP1-2**, **BP1-3** e **BP1-4),** deve enviar, dentro de certo prazo, o resultado para a empresa 2 de forma que o **BP1-1** seja processado, mantendo assim a cadeia de produção em andamento.

Nas especificações de grupo do **GrouPac** são caracterizados dois tipos de grupos de domínio:

- **Grupo Intradomínio** – É aquele no qual todas as réplicas são gerenciadas e se comunicam dentro do próprio domínio;
- **Grupo Interdomínio** – É aquele que contém componentes espalhados em vários domínios.

Por natureza, cada **BP** é constituído por vários (sub) **BPs** interdependentes, distribuídos em várias empresas. Considerando cada empresa um domínio de tolerância a falhas, os **BPs** podem ser caracterizados como grupos **interdomínios.**

As **Eas**, por sua vez, são atividades próprias de cada empresa que podem ou não estar envolvidas como o **DBP** em questão. As **Eas** não estão distribuídas em diferentes domínios (empresas), podendo ser caracterizadas como grupos **intradomínios** [Lima01a].

5.3.3 Serviço de Detecção de Falhas no FT-SALE

As especificações **FT-CORBA** permitem que apenas um detector de falhas monitore um ou mais *hosts* em um domínio de tolerância a falhas. Porém, isto implica na não tolerância a possíveis falhas neste detector, o que se torna um problema. A solução dada pelas especificações **FT-CORBA** seria a monitoração de um *host* a partir de um conjunto de detectores de falhas, gerando uma grande quantidade de mensagens entre *hosts* e detectores, o que seria inviável em redes de larga escala.

A solução proposta em [Lima01a], baseada no projeto **SALE** [Rabelo01], foi a adoção de grupos de **detectores de falhas**[16] dedicados, com o objetivo de monitorar cada *host*. O grupo de detectores controla as entradas e saídas (por falha) dos também detectores membros deste grupo.

[16] Um **detector de falhas** é, de maneira simplificada, um módulo do **SCG** responsável pela monitoração do comportamento dos membros do grupo. A detecção de falhas é baseada normalmente na associação de cada membro de um grupo a um detector específico (seu detector). Cada um destes detectores de falhas monitora os membros, ou um subconjunto dos membros, de um grupo e mantém uma lista de processos suspeitos (membros sob suspeita) [Bessani02].

Neste trabalho, foi descartada a hipótese da utilização de apenas um detector de falhas. A Figura 5.7, extraída de [Lima01b], ilustra o processo de Gerência de Falhas no **SALE**.

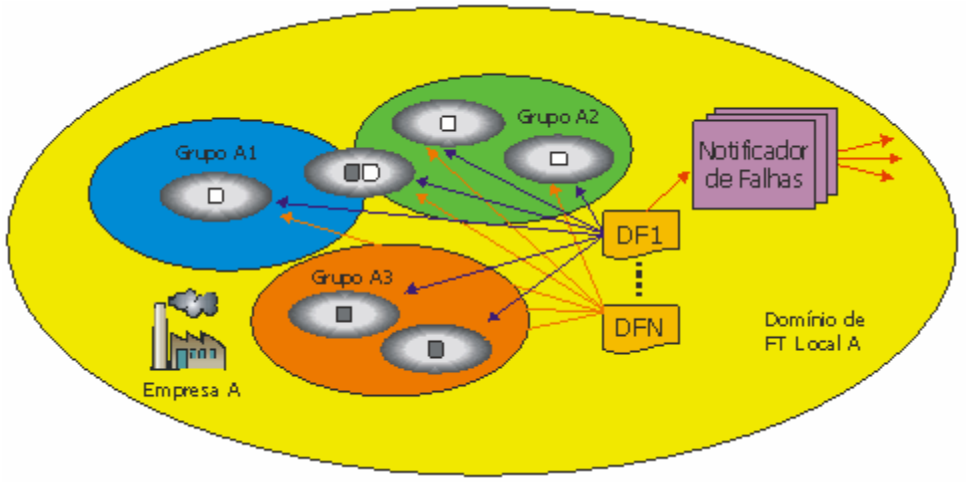

Figura 5.7: Domínio de tolerância a falhas local.

O Notificador de Falhas é responsável por enviar a mensagem de falha para o Sistema Gerenciador de Replicação (**SGR**). Com isso, o **SGR** pode atualizar as **IOGRs .**

Um protocolo baseado em voto majoritário é utilizado para tratar falhas em um determinado objeto. Para que seja admitida a falha de um determinado objeto, é preciso que haja confirmação de falha neste objeto, pela maioria dos detectores em ação. Para isso é necessário o monitoramento dos detectores, objetivando conhecer quantos detectores formam a maioria.

A Figura 5.8, extraída de [Lima01b], ilustra uma falha em um dos detectores que compõem o grupo de detectores do domínio.

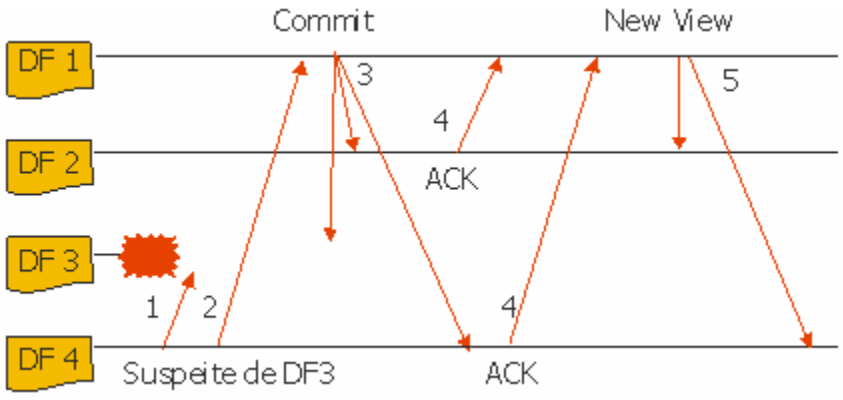

Figura 5.8: *Crash* [17] da réplica DF3.

Uma possível falha (***crash***) no detector **DF3** é detectada pelo objeto detector de falhas **DF4**, o qual envia uma mensagem "suspeite de **DF3**" para o objeto detector de falha primário (**DF1**). O **DF1**, por sua vez, ativa o protocolo de *membership* difundindo uma mensagem de *commit* para detectar quem ainda continua no grupo. Em seguida, os detectores ativos enviam uma mensagem de confirmação (***ACK)*** respondendo ao **DF1** que então, com base nos ***ACKs*** recebidos cria uma nova lista. Caso **DF1** falhe, o detector escolhido para substituí-lo segue a ordem estabelecida por um Anel Virtual, no qual cada membro monitora o parceiro imediatamente anterior da seqüência (ver Figura **5.9**).

No nível de detectores, o protocolo ***commit***[18] baseado em [Ricciardi91] (centralizado no detector primário), de duas fases (três fases no pior caso), é utilizado para obter uma nova lista de membros detectores (*New View*). Neste protocolo, os membros detectores são monitorados pelo detector imediatamente anterior, baseando-se em uma seqüência definida por um **Anel Virtual** formado pelos detectores.

A utilização do protocolo de *commit,* no modelo proposto, integrado ao processo de votação, permite preservar a propriedade dos agentes móveis executarem somente uma vez em determinado estágio (*exactly-once property*) [Fioreze03].

---

[17] A falha por ***crash*** ocorre quando um determinado componente não responde a uma requisição e a mais nenhuma requisição subseqüente.

[18] O protocolo clássico de validação atômica em duas fases (em inglês, *two-phase commit* - 2PC) [Gray78] é a melhor solução até hoje proposta. Infelizmente, em presença de falhas, a solução é bloqueante. Um protocolo de acordo é considerado não-bloqueante quando ele permite a tomada de decisão pelos processos corretos, mesmo na ocorrência de falhas. Os protocolos de validação atômica em três fases (em inglês, *three-phase commit* - 3PC) ([Keider95], [Guerraoui95]) são não bloqueantes [Greve01].

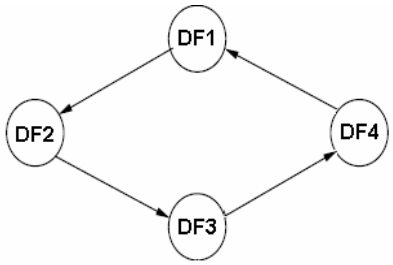

Figura 5.9: Anel Virtual formado pelos detectores de falhas.

Representação da lógica de substituição que forma o Anel Virtual dos detectores de falhas **(DF1, DF2, DF3 e DF4)**. Caso ocorra uma falha no detector **DF1**, por exemplo, o mesmo será substituído pelo detector **DF2** que é o próximo detector na seqüência de substituição imposta pelo Anel Virtual. Em nível de *host*, os responsáveis pela detecção de falhas são o grupo de detectores (**DF1**, **DF2**, **DF3** e **DF4**). Quando ocorre a falha em um *host*, todos os processos e objetos deste *host* são considerados falhos.

Para decidir se um determinado ***host*** é faltoso (***crash***), todos os **DF**s monitoram, periodicamente, dentro de um intervalo de tempo **T**(s), o mesmo conjunto de *hosts* dentro de uma EV. Caso qualquer um dos detectores suspeite de falha em um dos *hosts*, o protocolo executa um procedimento de decisão baseado em voto majoritário para alcançar consenso entre os detectores. A coordenação do processo de votação é sempre realizada pelo primário, como ilustra a Figura 5.10, extraída de [Fraga01b].

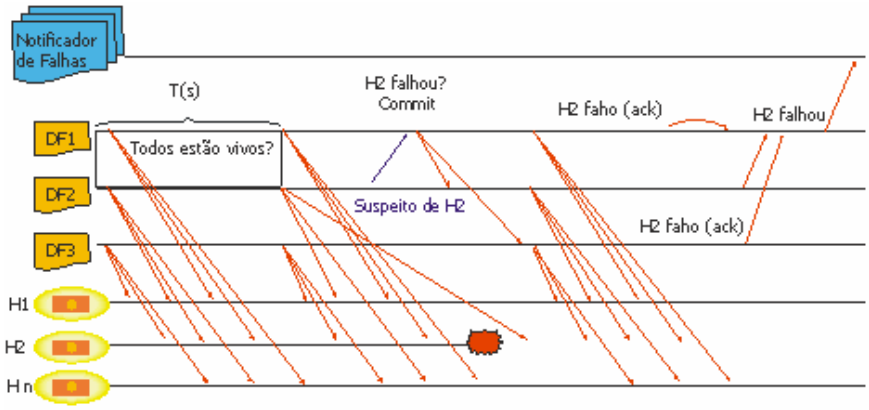

Figura 5.10: Instância do protocolo de detecção de falhas.

O detector **DF2** suspeita do *host* **H2** e avisa ao **DF1** (primário) sobre esta suspeita. A partir daí é iniciado o protocolo para se alcançar o consenso sobre a falha ou não de **H2**; O **DF1** solicita aos outros detectores (**DF2** e **DF3**) para que o próximo intervalo de monitoração (**T**(s)), informe sobre o *status* (vivo ou faltoso) do *host* **H2**. Então, cada detector informa ao **DF1** sobre o *status* de **H2** e finalmente, o detector de falha primário computa os resultados e decide sobre o *status* de **H2**. Então,

o **DF1** informa ao notificador de falha (**NF**) que, por sua vez, informa ao **SGR**, para que este gere uma nova **IOGR**, removendo **H2** da lista. A nova **IOGR** é enviada ao grupo de detectores para que estes atualizem suas listas de processos monitorados [Lima01b].

5.3.4 Tolerância a Falhas no Serviço de Nomes do FT-SALE

O **serviço de nomes** do **FT-SALE** é dividido em dois níveis: **SN-L** e **SN-G**, os quais são responsáveis, respectivamente, por: (i) gerenciar todas as **IOGRs** (**IORs**) dos objetos da aplicação de um domínio de tolerância a falhas, possuindo em seu contexto de nomes, uma cópia da **IOGR** do **SN-G**; e (ii) gerenciar a **IOGR** de cada **SN-L**, possuindo cópias atualizadas das **IOGRs** dos **SN-L** de todos os domínios registrados.

A implementação da tolerância a falhas no **SN-L** de cada empresa é realizada através da técnica de replicação **primária/*backups***. Com relação ao **SN-L**, esta técnica possui as seguintes características [Lima01a]:

- Cada **SN-L** é replicado e gerenciado dentro do próprio domínio de tolerância a falhas;
- O **SN-L** primário mantém os *backups* atualizados utilizando a técnica de recuperação e *logging*;
- O **SN-L** é responsável por receber e disponibilizar todas as referências do domínio;
- Réplicas do **SN-L** ficam espalhadas em diferentes pontos da Empresa, a fim de melhorar a acessibilidade;
- As requisições de operação para obter referência podem ser atendidas por qualquer réplica, visando melhorar o desempenho da empresa virtual;
- Somente o **SN-L** primário se registra no **SN-G**.

A cada substituição do **SN-L** primário por um *backup*, é feito um registro em um banco de dados. A estatística de falhas no **SN-L** pode ser obtida através do número de registros de falhas do **SN-L** da empresa.

A implementação da tolerância a falhas no **SN-G** é realizada através da distribuição de réplicas (ativas) do **SN-G** nas empresas membros. Cada empresa terá ao menos uma réplica do **SN-G** em um dos seus *hosts*, as quais devem ser gerenciadas pelos serviços oferecidos pela empresa coordenadora.

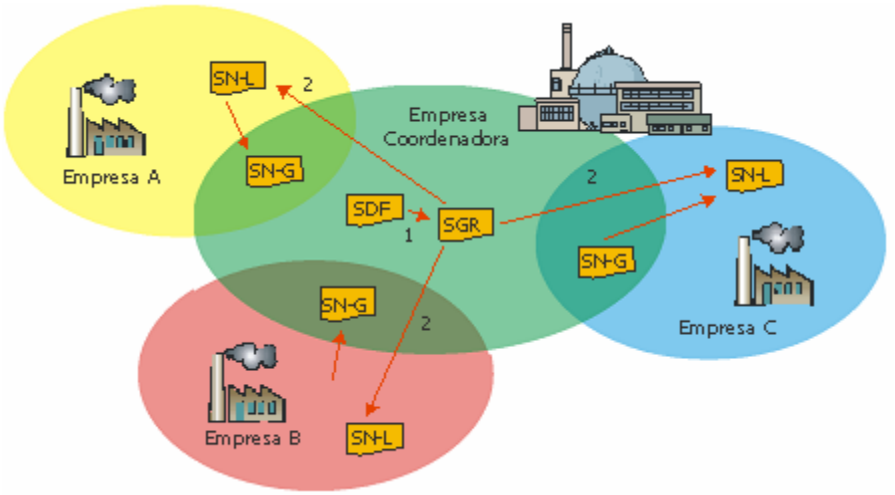

Figura 5.11: Processo de alteração dos membros no SN-G.

A figura ilustra uma alteração dos membros do **SN-G** de uma das empresas. Com esta alteração, uma nova **IOGR** será registrada. Neste caso, o **SDF** da empresa coordenadora manda a informação para o **SGR** (passo 1); o **SGR** por sua vez, se encarrega de enviar a nova **IOGR** para os **SN-Ls** de cada empresa membro (passo 2).

A Figura 5.12 ilustra a alteração de membros da **IOGR** em uma empresa virtual no caso de uma falha em algum objeto ou *host*, que fazem parte do grupo ao qual a **IOGR** pertence.

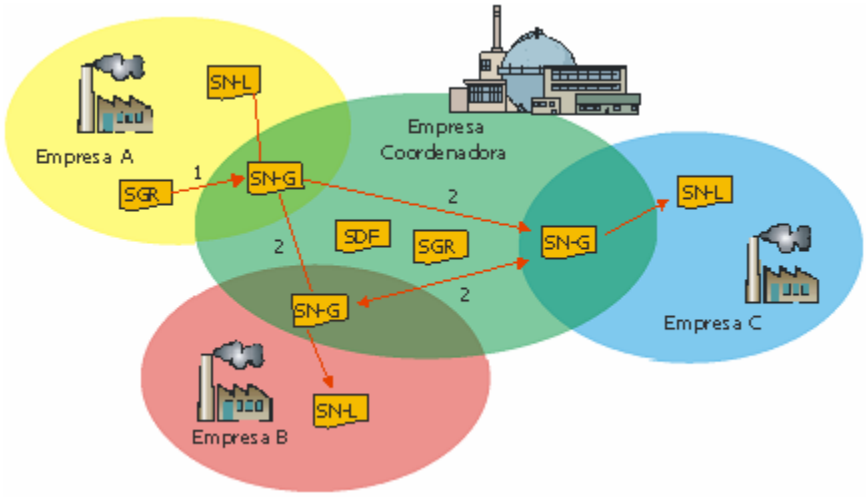

Figura 5.12: Mudança de membros no SN-L da empresa.

A figura ilustra uma mudança de membros no **SN-L** da empresa ª Neste caso, o **SGR** da própria empresa se encarrega de enviar a nova **IOGR** para a réplica do **SN-G** mais próxima **(passo 1)** e esta se encarrega de difundir a alteração para todas as outras réplicas **SN-G** da Empresa Virtual (passo 2).

5.3.5 Comunicação de Grupo no FT-SALE

Definidos os grupos que compõem a Empresa Virtual, torna-se necessário definir a forma de comunicação entre os componentes destes grupos. Como visto, as **Eas** formam grupos intradomínios e os **BPs** formam os grupos interdomínios.

5.3.5.1 Etapas da Comunicação entre Eas

Para fazer a comunicação entre as Eas pertencentes a uma mesma Empresa, basta que o cliente obtenha a IOGR desejada no SN-L da própria empresa, para então fazer a ligação. Para fazer a comunicação entre as Eas pertencentes à Empresas diferentes, as seguintes etapas são necessárias:

- O cliente deve primeiramente, acessar o **SN-L** da sua empresa para obter a **IOGR** do **SN-G**;
- Em seguida deve acessar o SN-G para obter a IOGR do SN-L da empresa desejada;

- Por fim, deve acessar o SN-L da empresa desejada para obter a IOGR para a ligação requerida.

5.3.5.2 Comunicação entre BPs

Todas as **IOGRs** dos **BPs** de um determinado **DPB** devem ter o mesmo nome registrado no **SN-L** de cada empresa. Se um novo **BP** for introduzido em outra empresa (em uma Empresa D, por exemplo), este **BP** terá uma **IOGR** gerada pelo SGR e deverá, também, ser registrado no SN-L desta empresa com o mesmo nome ("AACS") definido para o DBP, como mostra a Tabela 5.1, extraída de [Lima01a].

Empresa	***Nome***	***Referência de Grupo (BPs)***
Empresa A	"AACS"	IOGR2 do BP1-2
Empresa B	"AACS"	IOGR3 do BP1-3
Empresa C	"AACS"	IOGR4 do BP1-4
Empresa D	"AACS"	IOGR5 do BP1-5

Tabela 5.1: Registro de DBP particionado em Quatro Empresas.

A Figura 5.13 ilustra um exemplo de comunicação entre **BPs**. O DBP desta figura é distribuído entre três Empresas (A, B e C), e é formado por um conjunto de **BPs** (**BP1-2**, **BP1-3**, **BP1-4**), respectivamente.

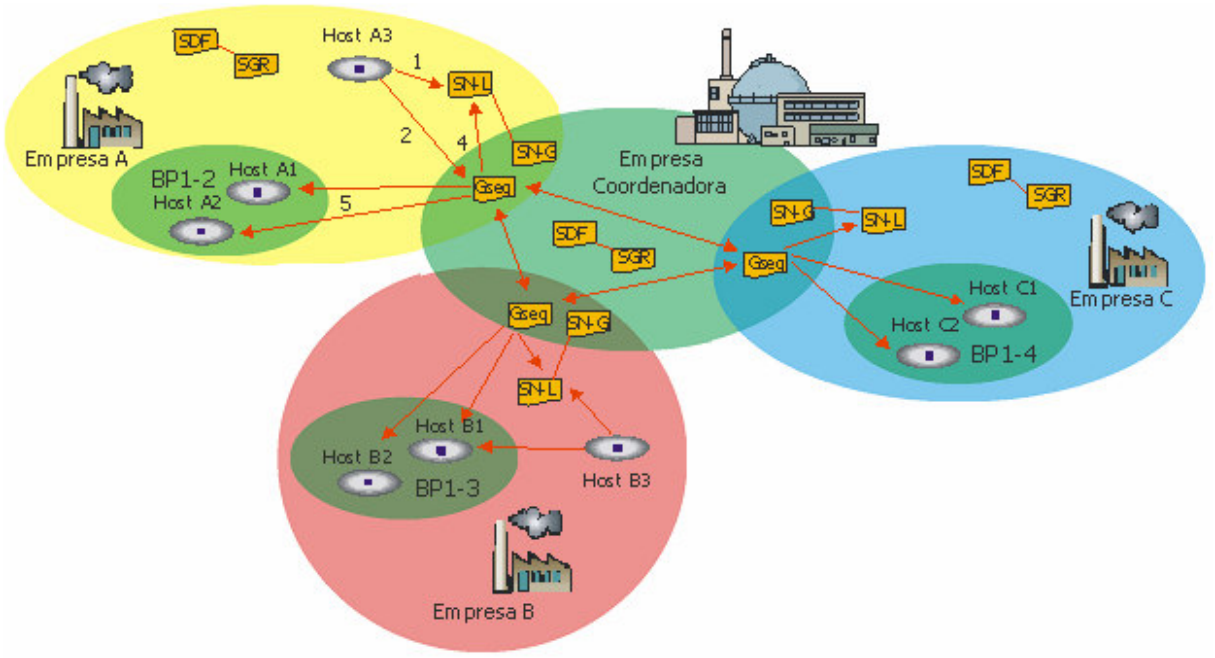

Figura 5.13: Comunicação de Grupo no FT-SALE.

A figura ilustra uma comunicação entre **BPs**, a qual segue os seguintes passos: i) um cliente (***Host A3***) obtém a **IOGR** de um grupo (**seta 1**), no **SN-L** da **Empresa A**; ii) o mecanismo de interceptação de mensagem no cliente verifica se um grupo é interdomínio ou não; iii) caso o grupo não seja interdomínio, é feita a invocação direta para o grupo (intradomínio); iv) caso o grupo seja interdomínio (um DPB), o interceptor do cliente desvia a requisição, incluindo o nome do grupo, para uma das réplicas do **Gseq** (**seta 2**); v) as réplicas, então, executam um protocolo para difusão da requisição segundo a ordem definida pelo parâmetro *OrderingType* [19] (**seta 3**); vi) as réplicas do **Gseq** entram em contato com o **SN-L** (**seta 4**), enviando o nome do **DBP** e obtendo a **IOGR** do **BP** da Empresa a qual pertence; vii) as réplicas do **Gseq** enviam a requisição para **BP** do domínio (**seta 5**).

O **Gseq** (*Gateway Seqüenciador*) é um mecanismo de ordenação de mensagem, que é parte importante na comunicação de grupo interdomínio. O Gseq é um grupo de objetos que faz parte do domínio de tolerância a falhas global. O Gseq tem uma única IOGR gerada pelo SGR do domínio global. Qualquer alteração na lista de membros do grupo Gseq é gerenciada pelo SGR do domínio global, o qual gera e atualiza a lista (IOGR) de réplicas do Gseq.

A principal função do grupo Gseq é servir de ponte para a difusão de uma mensagem entre os diferentes subgrupos de um grupo interdomínio [Lung01].

---

[19] *OrderingType*: determina o tipo de ordenação a ser utilizada pelos *Gateways* seqüenciadores, se *Unreliable*, *Reliable*, Causal ou Total.

5.4 Aspectos Relativos à Implementação da Plataforma FT-SALE e Plataforma de Busca e Seleção de Parceiros

Como base nas definições apresentadas acima, vê-se que as plataformas utilizadas como suporte a plataforma proposta trazem vantagens para aplicações distribuídas em larga escala, como as de uma Empresa Virtual. A seguir são apresentadas as implementações realizadas em cada uma destas plataformas. O objetivo é analisar a utilização das mesmas em um cenário real.

Como visto anteriormente, em [Schmidt03] é apresentado e discutido o problema da Busca e Seleção de parceiros para a formação de Empresas Virtuais (EV). Neste trabalho um protótipo foi definido e implementado tendo em vista uma Organização Virtual existente – a TechMoldes, que é composta por treze empresas ligadas ao ramo de ferramentaria, mais precisamente à fabricação de moldes e matrizes, as quais cooperam de forma temporária para atender oportunidades de negócios.

Na implementação foram utilizadas duas plataformas de agentes, sendo elas *Aglets* [AGLETS02] e *Massyve* [MASSYVE02]. A primeira foi utilizada para prover o sistema com os agentes móveis, dando suporte a todas ações necessárias a um agente móvel, desde sua criação, seu envio para locais remotos e conseqüente recebimento, e possibilitando toda a comunicação entre os agentes, até o momento do seu encerramento. A segunda visou, sobretudo, validar o modelo proposto perante os aspectos de interoperabilidade. Assumiu-se a existência de um sistema legado particular para cada empresa do *cluster*, sistema este usualmente heterogêneo, com o qual é necessário interagir [Schmidt03].

Visando cobrir outras fases do ciclo de vida de uma EV, tais como a Configuração e Operação/Evolução, este sistema foi integrado a uma plataforma maior chamada **SC_2** (*Supply Chain Smart Coordination*) [Rabelo02] e passou a ser visto como um módulo/componente "agentificado" do **SC_2** para uma função que até então não era suportada. O **CORBA** foi utilizado como *middleware*, para garantir a interoperabilidade entre os agentes, provendo a facilidade de comunicação em sistemas heterogêneos. As mensagens de aplicação trocadas entre as entidades do sistema seguem o formato **KQML**, estabelecendo assim, uma padronização e atribuindo um significado ao ato da comunicação. Na representação da mensagem no formato **KQML** foi usada a linguagem **XML**, ou seja, as mensagens que seguem o formato **KQML** foram "empacotadas"

em mensagens **XML**. Um exemplo da dinâmica do sistema é apresentada em [Schmidt03, página 81].

Aspectos relacionados ao esquema de segurança para sistemas de agentes móveis ainda não haviam sido totalmente implementados no sistema. O sistema garantia apenas a integridade e confidencialidade durante os saltos dos agentes, através do uso do **RMI** sobre o **SSL**, integrado a plataforma *Aglets*.

Testes de desempenho foram realizados sob o cenário da TechMoldes a fim de avaliar a aplicabilidade deste sistema em ambientes distribuídos de larga escala.

Em [Lima01b] é apresentada uma proposta para inserir tolerância a falhas dentro do contexto de Empresas Virtuais utilizando como base as especificações **FT-CORBA**. Neste trabalho a plataforma proposta não foi implementada, contudo, é ilustrado o ambiente das soluções do Grouppac e os testes realizados neste ambiente. Estas informações podem ser encontradas no anexo (Anexo I.) no referido trabalho.

5.5 Arquitetura Proposta: FT-SALE/SMAM

Nesta seção, apresenta-se a **Plataforma FT-SALE/SMAM**, que visa integrar as duas plataformas discutidas nas seções anteriores, ou seja, permitir a utilização das técnicas de tolerância a falhas na melhoria da confiabilidade do processo de busca e seleção de parceiros em **Evs**. A arquitetura proposta será apresentada de forma conceitual. No entanto, algumas considerações são feitas no sentido de prover uma implementação futura. A Figura 5.14 ilustra a plataforma proposta.

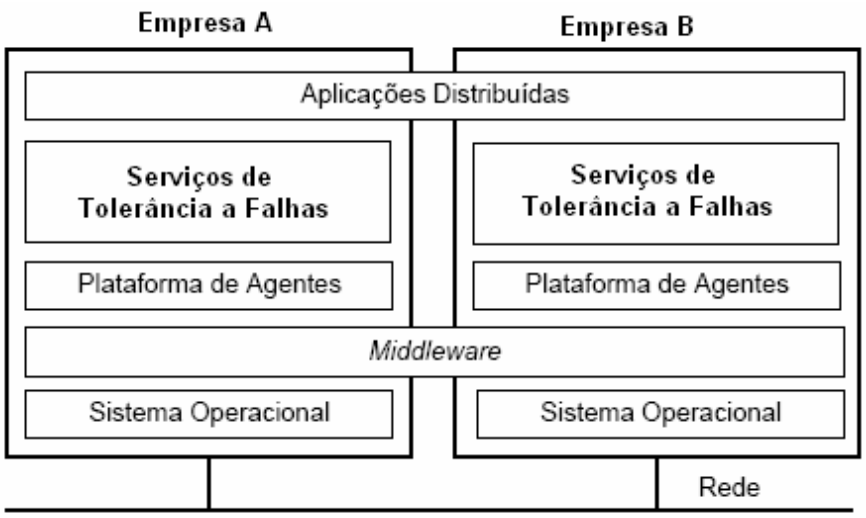

Figura 5.14: Plataforma proposta.

Os serviços de **tolerância a falhas** serão implementados seguindo as especificações de tolerância a falhas propostas na Arquitetura **FT-SALE.**

O padrão **CORBA** (*Common Object Request Broker Architecture*) [OMG 03] foi escolhido para compor a **camada de *middleware*** da plataforma proposta, pois é utilizado em aplicações distribuídas orientadas a objetos, que seguem conceitos de sistemas abertos e que definem padrões para ambientes heterogêneos. O padrão CORBA tem por finalidade fornecer funcionalidades para a interoperabilidade e a portabilidade, como, por exemplo, a transparência de localização, e a heterogeneidade de plataforma e de linguagens de programação. Desta forma, torna possível a comunicação entre as aplicações, independente de localização ou de quem as projetou [Schmidt03]. Esta escolha se deu em razão da grande aceitação desta especificação, por

ser o padrão utilizado tanto na **Plataforma FT-SALE** [Lima01a], quanto na **Plataforma de Busca e Seleção de Parceiros** [Schmidt03].

O padrão **CORBA** incorpora ainda os protocolos Internet para comunicação (pilha **TCP/IP**). A escolha do **CORBA**[20] como *middleware* de comunicação oferece outras vantagens quando seus serviços são integrados a tecnologias de agentes móveis [Wangham03a]. A comunidade de agentes móveis vê a comunicação entre agentes como sendo, na maior parte do tempo, a troca de objetos ou de referências para objetos e requisições de serviços. A comunicação entre os agentes é bastante discutida pelo **CORBA**. Outros detalhes da arquitetura CORBA foram fornecidos no Capítulo 3.

5.5.1 Integração do Sistema de Busca e Seleção

Inicialmente, é necessário adequar as plataformas (ou modelos), identificando como podem ser formados os domínios de tolerância a falhas e os grupos que vão compor estes domínios dentro da Organização Virtual, procurando desta forma eliminar o problema da existência de um único **Gerenciador de Grupo** para uma **OV**, que é discutido logo a seguir.

5.5.1.1 Pontos Fracos do Sistema de Busca e Seleção de Parceiros e Soluções Propostas

Como visto na Seção 5.2.2, durante o processo de formação de uma EV, o **Agente Empresa** (**AE)** solicita ao **Gerenciador de Grupo** a relação das empresas registradas, podendo então verificar quais devem ser consultadas, e repassando ao **Agente *Broker*** (**BR**). Um primeiro problema está na vulnerabilidade do sistema nesta fase devido a presença de apenas um **Gerenciador de Grupos**. Supondo que o mesmo falhe, e que o *broker* não consiga adquirir a lista dos membros da **OV**, não haveria possibilidade de identificação dos candidatos em potencial para a produção da nova **ON**. E mais, não seria possível distribuir o pedido entre as empresas que se encaixam no perfil desejado. Conseqüentemente, este problema comprometeria todos os passos seguintes do processo de busca e seleção de parceiros.

---

[20] É importante ressaltar que a utilização do *middleware* CORBA não obriga que os agentes móveis sejam necessariamente objetos CORBA.

Além disto, o **Agente Móvel** (**AM**) tem como função deslocar-se para as empresas (pré-selecionadas) do *cluster* em busca de informação para a formação da empresa virtual. Para isso é necessário que o **AM** saiba a localização de cada empresa na rede para as quais pretende migrar. Este "endereço" é fornecido pelo serviço de nomes do **CORBA** (*CossNaming*) [OMG97], quando o agente solicita a referência da empresa. Através da interface **CORBA,** o **AM** se comunica com o **AE** representante da empresa visitada, pedindo as informações desejadas, as quais são retiradas do banco de dados local da empresa visitada e repassadas ao **AM**.

O fato é que o Sistema de Busca e Seleção de Parceiros utiliza o serviço de nomes CORBA, que, segundo a especificação *CossNaming*, não é tolerante a falhas. Este é o segundo problema identificado. Em outras palavras, o serviço de nomes, sendo um único ponto de falha, pode comprometer toda a estrutura do FT-CORBA: se este falhar, todos os objetos e grupos de objetos ficam inacessíveis [Lung01]. Isto implica que não haveria possibilidade de uma recuperação imediata da lista dos membros do grupo (IOGR). Desta forma, não haveria como o **AM** saber a localização de cada empresa na rede para as quais pretende migrar.

A **Plataforma FT-SALE/SMAM** propõem se a solucionar estes problemas implementando técnicas de tolerância a falhas nos serviços de nome CORBA, e ainda implementando serviços de nome em cada nó (empresa) da **OV**. Dentre os componentes necessários para esta implementação encontram-se os **SN-Gs** e **SN-Ls** e suas réplicas. O **SN-G** possui uma lista dos membros do grupo (**IOGR**) da **OV**. O **SN-L** possui uma cópia desta lista, assim como suas réplicas. A **IOGR** é alterada dinamicamente, tanto na entrada de um novo membro, quanto na saída de um membro do grupo. Por este motivo, o *broker* possui uma lista sempre atualizada dos membros da **OV**.

Desta forma, o **Gerenciador de Grupos** pode ser eliminado do sistema, e sua função de fornecer ao *broker* a lista de membros da **OV**, pode ser realizada pelo **SN-L,** através da **IOGR** que pode ser acessada pelo *broker* diretamente. Assim, se ocorrer uma falha no **SN-L**, o mesmo pode ser imediatamente substituído por uma de suas réplicas mais próximas, o que evitaria a ocorrência do primeiro problema citado anteriormente. Isto é possível, porque o *broker* poderia solicitar uma cópia da **IOGR** ao **SN-L** substituto, possibilitando assim que *broker* identificasse os candidatos em potencial para a produção da nova **ON**.

Com relação ao segundo problema, a plataforma proposta provê serviço de tolerância a falhas para o serviço de nomes do CORBA, através do uso de técnicas de replicação. Portanto, em caso de falha no serviço de nomes CORBA, ocorre a substituição imediata do SNL ou do SNG, possibilitando ao AM acessar a IOGR que informa a localização de cada empresa na rede para as quais pretende migrar.

5.5.1.2 Definição dos Grupos de Tolerância a Falhas

De acordo com [Schmidt03], as informações a serem prestadas pelas empresas dependem fundamentalmente da natureza da **tarefa**. Por este motivo, o cenário a ser considerado para a exemplificação da adaptação do **Sistema de Busca e Seleção de Parceiros** ao modelo proposto será o de uma **ON** onde o tema principal envolve **produção**.

Como visto, uma **tarefa** é composta por várias "**subtarefas**" a serem distribuídas entre os participantes da **EV**, e uma empresa pode assumir a realização de mais de uma destas **subtarefas**.

Em comparação com as especificações de grupo do **FT-SALE,** é assumido que uma **tarefa** completa equivale a um **DBP** (*Distributed Business Process*), o qual pode ser dividido em um grupo de **BPs** (**subtarefas**). As várias **subtarefas** (**BPs**) podem ser executadas por uma série de diferentes fornecedores, conectados entre si por uma rede de computadores.

Para toda **tarefa**, um membro (nó) da Empresa Virtual é eleito como "disparador" do negócio (coordenador) e os outros como seus fornecedores. O coordenador é a empresa possuidora do *Broker* responsável pela **ON** em questão.

Cada empresa é responsável por algumas **subtarefas** e as suas respectivas Atividades da Empresa (*Enterprise Activities* – **Eas**). As **Eas** representam as operações de manufatura "de baixo nível" dentro de cada **subtarefa** e as **subtarefas** representam o valor agregado de cada empresa na cadeia de produção.

Considerando **cada empresa** como sendo um **domínio de tolerância a falhas local**, as **subtarefas** podem ser caracterizados como grupos **interdomínios.**

As **Eas**, por sua vez, são atividades próprias de cada empresa que podem ou não estar envolvidas como a **subtarefa** em questão. As **Eas** não estão distribuídas em diferentes domínios (empresas) podendo ser caracterizadas como grupos **intradomínios** [Lima01a].

Na seção seguinte é apresentada a definição do domínio de tolerância a falhas do modelo proposto.

5.5.1.3 Definição do Domínio de Tolerância a Falhas

Inicialmente, deve-se definir o **domínio de tolerância a falhas**, para que em seguida seja associado um conjunto de propriedades de tolerância a falhas a cada grupo de objetos, tornando possível definir propriedades que se aplicam ao grupo todo ou a um tipo específico de objetos dentro do grupo.

O modelo proposto neste trabalho considera uma **OV** como sendo um **domínio de tolerância a falhas global**. Cada empresa pertencente ao grupo de empresas desta **OV** constituirá um domínio **local.** Os *hosts* são considerados **locais**, onde se encontram os agentes que fazem parte de todo o processo de busca e seleção de parceiros.

Desta forma, o modelo proposto irá tratar, entre todas as falhas possíveis em uma **SMAM**, falhas no *host*, ou seja, falhas nos espaços físicos onde os componentes se hospedam e ainda possibilitam a execução de componentes de computação (locais) e falhas nos "objetos" agentes.

Assim como no **FT-SALE**, no modelo proposto, cada empresa possuirá sua própria infra-estrutura para tolerância a falhas (domínio de tolerância a falhas local), composta pelos componentes **SRL** (Serviço de Recuperação e *Logging*), **SDF** (Serviço de Detecção de Falhas), **SGR** (Serviço de Gerenciamento de Replicação), **SN-L** (Serviço de Nomes Local), **SN-G** (Serviço de Nomes Global) e **Gseq** (*Gateway* Seqüenciador). As funções de cada componente da estrutura de tolerância a falhas foram apresentadas anteriormente.

Como visto, um domínio de tolerância a falhas é composto por várias máquinas e grupos de objetos. Atuando exclusivamente dentro de cada domínio encontram-se um Gerenciador de Replicação, um Notificador de Falhas e um Detector de falhas, componentes definidos na infra-estrutura do **FT-CORBA**.

Depois de definido o domínio, com sua infra-estrutura de tolerância a falhas, é estabelecido o monitoramento das falhas nos *hosts*.

Assim como explicado anteriormente, o processo de monitoramento é feito pelo **SDF**. No caso de uma falha em um *host*, o **SDF** notifica as falhas ao **SGR**, o qual estabelece uma nova **IOGR** já com a retirada do *host* falho. O **SN-L** possui as **IOGRs** (e **IORs**) do domínio ao qual pertence. As novas **IOGRs** estabelecidas pelo **SGR** devem ser enviadas aos **SN-L** para que este possa atualizar a **IOGR** existente. Para permitir que objetos de um domínio possam localizar objetos de outro domínio, o **SN-G** faz a ligação de todos os **SN-L** [Lima01a].

A Figura 5.15, extraída de [Lung01], ilustra os domínios de tolerância a falhas **local** e **global** com os componentes que fazem parte da infra-estrutura de tolerância a falhas dos dois domínios.

Assim como no **FT-SALE**, o **SN-L** é replicado e gerenciado dentro do próprio domínio de tolerância a falhas (**Empresa**). O **SN-L** primário mantém os *backups* atualizados utilizando a técnica de recuperação e *logging*. O **SN-L** (primário ou *backup*) é responsável por receber e disponibilizar todas as referências do domínio. É importante ressaltar que somente o **SN-L** primário se registra no **SN-G.** Todas as empresas membro possuem uma réplica (ativa) do **SN-G**, ou seja, todas as empresas terão pelo menos uma réplica do **SN-G** em um dos seus *hosts*.

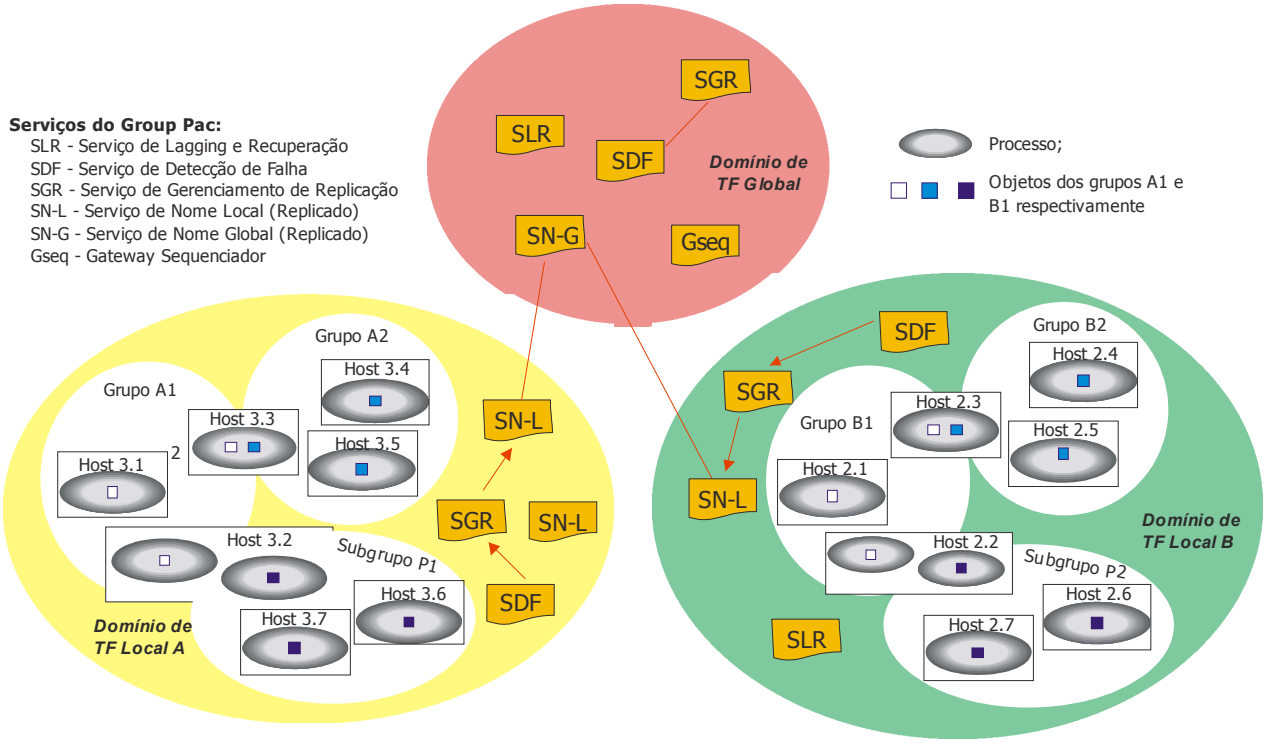

Figura 5.15: Domínios de tolerância a falhas local e global e seus componentes.

No modelo proposto, as réplicas do **SN-G** devem ser gerenciadas pelos serviços oferecidos pela empresa escolhida para ser a responsável (empresa coordenador) pela criação e manutenção da **OV**.

A nova arquitetura acrescenta, ao sistema, técnicas de tolerância a falhas baseadas em especificações do **FT-SALE**, visando melhorar o controle de *membership* e, desta forma, torna o processo mais confiável. Isso permite a eliminação do **Gerenciador de Grupos,** proposto no sistema de Busca e Seleção de Parceiros [Schmidt03], resolvendo assim o problema de falha em um único ponto. Para isso, o Gerenciador de Grupos foi substituído pela presença do **SN-L** e do **SN-G** e suas réplicas, em cada empresa do grupo, permitindo desta forma que no caso de falha em um destes componentes, o mesmo seja substituído imediatamente e de forma transparente ao cliente.

Portanto, na arquitetura **FT-SALE/SMAM**, quando uma nova **ON** (Oportunidade de Negócio) é lançada, o *Broker* precisa apenas consultar a **IOGR** da empresa a qual pertence. Desta forma, a busca aos membros do grupo torna-se mais rápida e confiável, visto que a **IOGR** é

atualizada constantemente e replicada em diversos pontos do domínio. Com a presença da **IOGR** no domínio local (Empresa) ocorre ainda uma diminuição significativa no fluxo de mensagens geradas no sistema.

5.5.1.4 Fase de Inserção de uma Empresa na Organização Virtual

Ao ingressar em uma **OV,** uma **empresa** passa a ser considerada um novo domínio de tolerância a falhas no qual deve ser implementada a infra-estrutura para tolerância a falhas, constituída pelos elementos **SLR**, **SGR**, **SN-L, SNG e Gseq**, definidos anteriormente. O **SN-L** contém as **IOGRs** e as **IORs** do domínio ao qual pertence.

O diagrama de seqüência apresentado na Figura 5.16 ilustra os passos para o **registro** de uma **nova empresa** dentro de uma **OV**. Considera-se no diagrama que todo o processo de configuração dos elementos necessários pra a formação da infra-estrutura de tolerância a falhas na **nova empresa** esteja concluído.

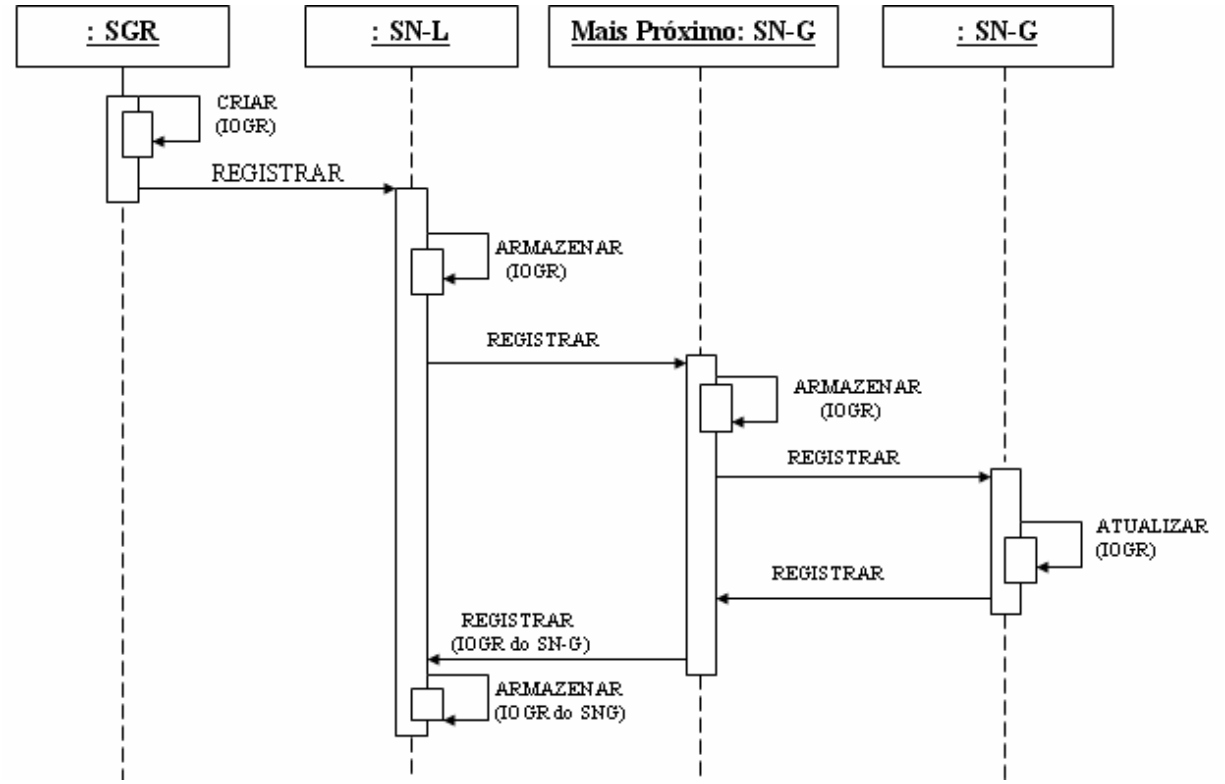

Figura 5.16: Diagrama de Seqüência: o registro de uma nova empresa na OV.

Após a configuração dos elementos necessários para a constituição deste novo domínio, o **SN-L primário** cria a **IOGR da nova empresa** e se registra no **SN-G mais próximo**, que por sua vez se encarrega de difundir a informação para todas as **réplicas do SN-G** existentes na **OV**. Com a atualização do **SN-G** das demais empresas membro da **OV**, todas as empresas que constituem a **OV** tomam conhecimento da inserção da **nova empresa** no grupo.

5.5.1.5 Fase de Anúncio e Resposta ao Anúncio

Após o Registro da empresa na **OV** as demais empresas participantes podem tomar conhecimento da inserção da nova empresa no grupo**.** Desta forma**,** diante do surgimento de uma **ON**, a nova empresa poderá ser comunicada sobre a **ON** e conseqüentemente fazer parte da seleção de empresas que participarão da nova **EV**, que será criada para atender a **ON** em questão.

Como visto na Seção 5.2.2, a primeira fase após a identificação de uma nova **ON** é a fase do **Anúncio**, a qual consiste em comunicar às empresas componentes da **OV**, que há uma tarefa a ser realizada e, portanto, uma **EV** precisa ser formada, provendo informações para que a empresa possa decidir em candidatar-se ou não [Schmidt03]. No Sistema de Busca e Seleção proposto por [Schmidt03] as fases de registro, anúncio e resposta ao anúncio seguem os passos ilustrados no diagrama da Figura 5.17.

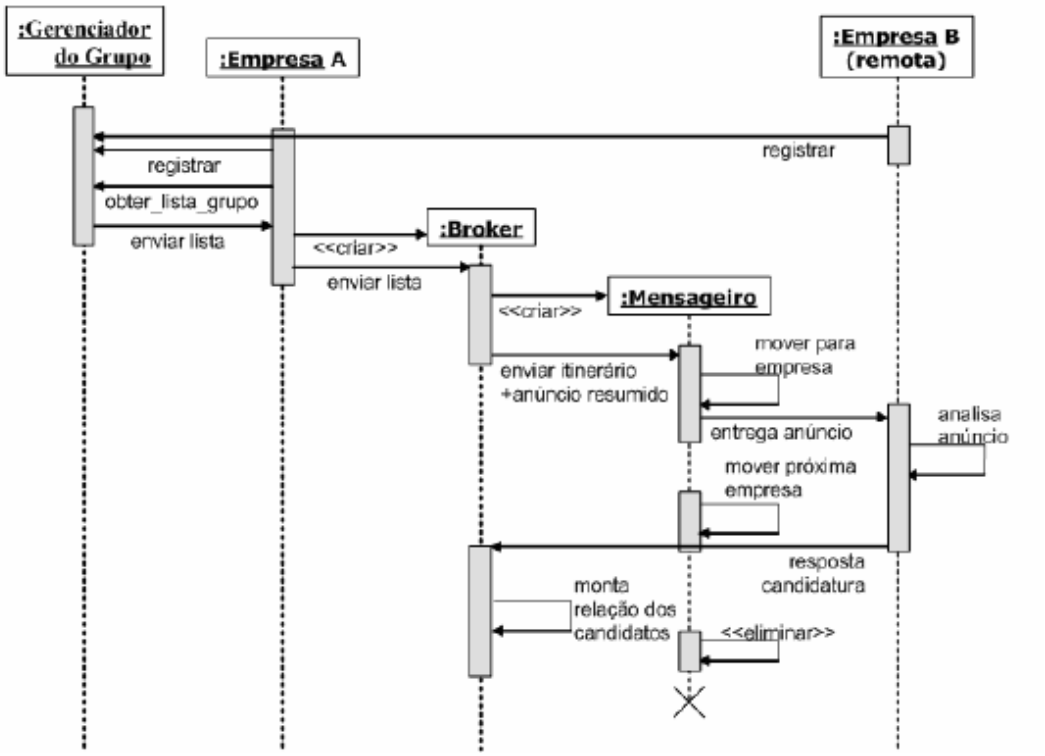

Figura 5.17: Diagrama de Sequência: 1ª fase de anúncio e o envio de respostas no Sistema de Busca e Seleção de Parceiros. Extraído de [Schmidt03].

O diagrama possui os seguintes passos: 1º) Os agentes Empresa A e B efetuam registro junto ao Gerenciador de Grupo (**registrar**); 2º) O agente Empresa A inicia um processo de *brokerage* e solicita a lista de membros do grupo ao Gerenciador de Grupo (**obter_lista_grupo**); 3º) O Gerenciador de Grupo envia lista de membros para o agente Empresas A (**enviar lista**); 4º) Após receber a lista, o agente Empresa A efetua uma pré-seleção das empresas que se encaixam no perfil procurado e então cria um agente *Broker* **(<<criar>>**); 5º) O agente Empresas A repassa ao *Broker* criado, as informações sobre a **ON** e a relação das empresas que serão consultadas (**enviar lista**); 6º) De posse das informações enviadas no 5º passo, o agente *Broker* cria um Agente Móvel *Broker* para auxiliá-lo na escolha das empresas com capacidade para integrar a EV **(<<criar>>**); 7º) O agente *Broker* monta o itinerário a ser percorrido usando a relação das possíveis empresas candidatas e repassa ao Agente Móvel *Broker* juntamente com o anúncio resumido (**enviar itinerário + anúncio resumido**); 8º) O Agente Móvel *Broker* migra até o sistema de agentes de cada empresa candidata e entrega o anúncio a cada uma das empresas (**mover para empresa**) + (**entregar anúncio**); 9º) Após ter recebido o anúncio, o sistema da empresa visitada cria uma interface onde apresentará os dados recebidos, possibilitando ao responsável da empresa avaliar o que está sendo proposto e então decidir pela participação da empresa na seleção (**analisa anúncio**); 10º) Caso esteja de acordo com o perfil da **ON** e opte por participar da **EV**, a empresa em questão envia resposta a candidatura ao agente *Broker* (**resposta candidatura**); 11º) O agente *Broker* monta a relação dos candidatos e acrescenta a empresas a esta relação (**monta relação de candidatos**); 12º) caso a empresa não responda ao anúncio a mesma é eliminada do processo (<<**eliminar**>>).

Depois da fase de anúncio surge a fase de **Resposta ao Anúncio**. É nesta que são realizados registros das respostas aos anúncios, enviadas pelas empresas ao *Broker*. A fase de resposta ao anúncio ajuda a manter o controle de envio de mensagens, permitindo inclusive, através de auditorias, detectar o não recebimento de alguma mensagem por uma determinada empresa. As empresas que responderam aos anúncios são cadastradas em listas. Tais listas são usadas ao final da etapa do anúncio completo para saber quais empresas serão consultadas no levantamento de dados [Schmidt03].

O diagrama de seqüência apresentado na Figura 5.18 ilustra os passos para a primeira fase do Anúncio e o envio de respostas ao Anúncio Resumido na plataforma **FT/SALE-SMAM.** Considera-se no diagrama concluída a fase de **registro da empresa** de forma semelhante a apresentada no diagrama da Figura 5.16.

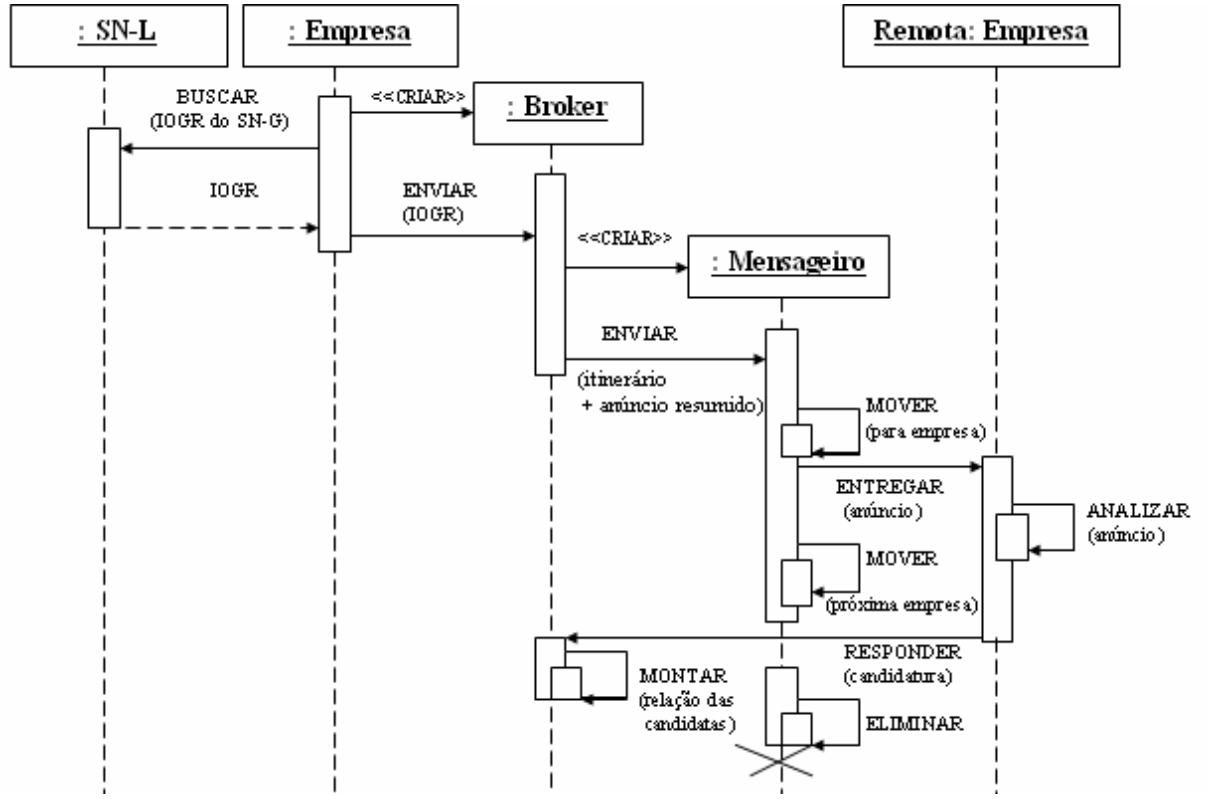

Figura 5.18: Diagrama de Sequência: a busca pela lista de membros do grupo de empresas pertencentes a OV, 1ª fase de anúncio e o envio de respostas.

Comparando os diagramas das Figuras 5.17 e 5.18 é possível observar que diferentemente do **Sistema de Busca e Seleção de Parceiros,** proposto em [Schmidt03], ao iniciar o processo de ***Brokerage***, na plataforma **FT-SALE/SMAM**, a empresa não mais solicita a lista de empresas participantes do grupo ao Gerenciador de Grupos e sim ao **SNL local**, o qual possui armazenada uma cópia da **IOGR** do **SN-G**. Esta cópia contém informações sobre todas as empresas membro do grupo de empresas da OV. Os demais processos, assim como o processo de anúncio permanece como proposto em [Schmidt03]. Esta modificação elimina a vulnerabilidade existente neste sistema, pois substitui a existência de um único **Gerenciador de Grupos** por réplicas do **SN-L** e do **SN-G**, distribuídas pela Organização Virtual, permitindo que em caso de falha do **SN-L primário**, o mesmo seja substituído imediatamente por uma de suas réplicas atualizadas.

5.5.1.6 Configuração do Sistema

Na plataforma **FT-SALE/SMAM** uma empresa do grupo deverá ser escolhida para assumir o papel de **coordenador** da **OV**. O sistema multi-agente do coordenador da **OV** terá a possibilidade de interagir com os sistemas multi-agente das empresas membro.

O Sistema de Busca e Seleção de Parceiros é instalado em cada uma das empresas membros da **OV**, inclusive na empresa escolhida como coordenador da **OV**.

Os serviços de tolerância a falhas referentes ao domínio Global são configurados somente na Empresa coordenadora e servem como ferramenta de controle de tolerância a falhas durante todo o processo de Busca e Seleção e ainda durante todas as fases do ciclo de vida das empresas virtuais membro da **OV**. Os serviços de tolerância a falhas referentes ao domínio local serão configurados em todos os membros da **OV**, inclusive no coordenador, para que possa haver uma garantia de atualização das **IOGRs** local e global.

5.5.2 O *Framework* [21] da Aplicação

A aplicação em questão é o **Sistema de Busca e Seleção de Parceiros na Formação de Empresas Virtuais**, o qual é constituído a partir de um **SMAM**, que visa apoiar o gestor da

[21] O mesmo que arcabouço ou *skeleton*.

empresa virtual (*Broker*) nas várias fases da execução de um processo de Busca e Seleção de parceiros.

A **Plataforma FT-SALE/SMAM** se encaixa aqui como base genérica de serviços de infra-estrutura, na qual os serviços de aplicação (no caso sendo estudado, os do Sistema de Busca e Seleção) podem ser suportados no que se refere à comunicação.

A representação virtual de uma empresa componente da Empresa Virtual é composta por uma *suite* de módulos com serviços de alto nível. O **SC2** é um destes módulos e o Sistema de Busca e Seleção é um dos módulos do **SC2** .

As comunicações *intra-suite* e *inter-suite* são feitas através de um sistema de *workflow*, com as informações trocadas em formato **XML** no *backbone* disponibilizado. Por exemplo, a requisição de alguma informação feita por um agente do Sistema de Busca e Seleção de uma empresa segue os seguintes passos:

1. O agente requerente se inscreve numa lista (por "assunto") do *workflow backbone*;

2. Outro agente que tiver a informação desejada publica a mesma no *backbone*;

3. O mecanismo de *workflow* envia a informação para os agentes clientes da lista.

Deve-se destacar aqui a existência de dois elementos que são de suma importância e servem de base para o funcionamento do sistema: o **banco de dados,** que tem por finalidade armazenar todas as informações que são intercambiadas entre os sistemas multi-agente das empresas e a **ferramenta de *workflow***, que gerencia as mensagens, em formato **XML**, trocadas entre os parceiros.

As plataformas de suporte a interoperação em uma **EV** devem encapsular as plataformas legadas com as quais as empresas componentes já trabalham.

A Figura 5.19 apresenta o *Framework* da **Plataforma FT-SALE/SMAM**.

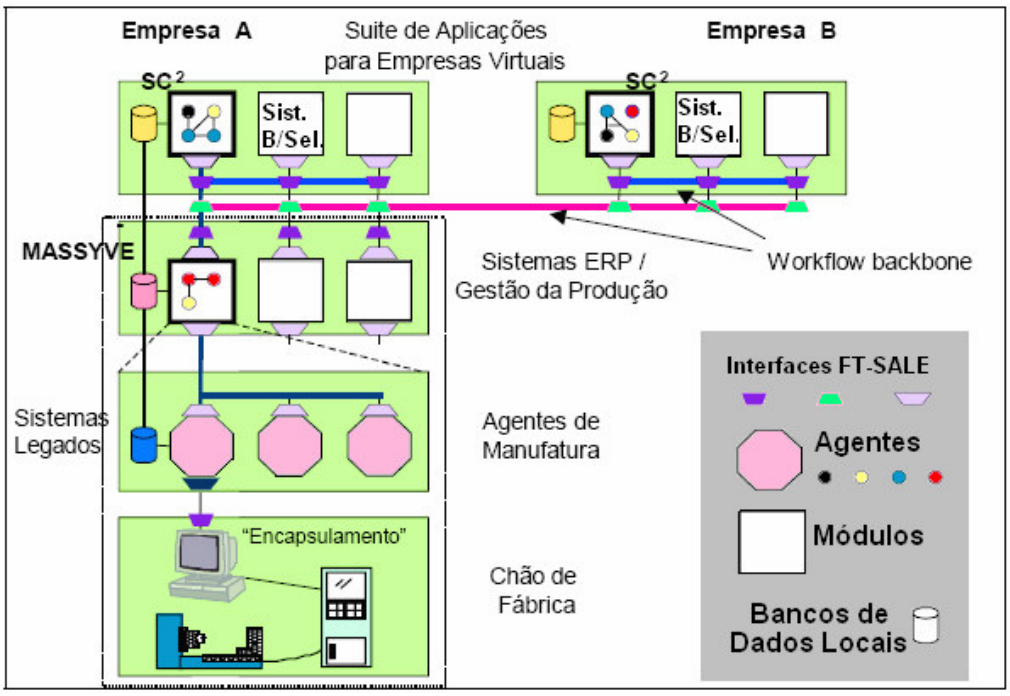

Figura 5.19: *Framework* da plataforma FT-SALE/SMAM.

A plataforma **FT-SALE/SMAM** oferece um conjunto de serviços de infra-estrutura para permitir uma adequada execução de serviços no nível aplicação, focando principalmente no cenário de **EVs**. A Figura 5.19 mostra os componentes da plataforma proposta, que são:

- **Sistema SC2** – O **SC2** (*Supply Chain Smart Coordination*) é um sistema que cobre varias fases do ciclo de vida de uma empresa virtual, tais como a Configuração e Operação/Evolução [Schmidt03]. É constituído a partir de um Sistema Multi-Agente de suporte à decisão, implementado na plataforma Massyve, que visa assistir o gestor da empresa virtual na monitoração da execução de um "processo de negócios" [Rabelo01a]. Este sistema é composto por vários módulos, sendo o Sistema de Busca e Seleção de Parceiros um destes módulos. Os serviços de aplicação do **SC2** são suportados pela plataforma **FT/SALE-SMAM** no que se refere à comunicação. O sistema **SC2** é implantado em cada um dos "nós"/empresas e pode interagir com seus módulos internos, com módulos de uma dada plataforma (*suite*) para EV e com os sistemas legados de cada empresa [Schmidt03].

- **Ferramenta de *Workflow*** – É responsável por gerenciar as mensagens, em formato **XML**, trocadas entre os parceiros. As informações são trocadas através do ***workflow backbone*** disponibilizado. A Figura 5.20 ilustra a comunicação entre agentes através dos mecanismos de ***workflow***.

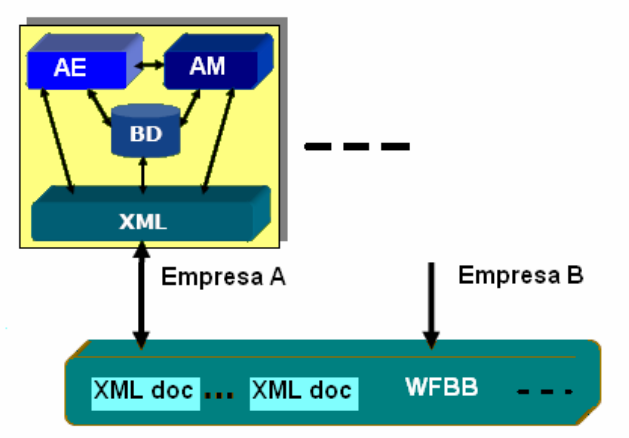

Figura 5.20: Comunicação entre agentes através dos mecanismos de *workflow*.

- **Sistemas Legados** – São os sistemas com os quais as empresas componentes já trabalham (tais como ERPs), e devem ser encapsulados na forma de objetos **CORBA** (agentes, no caso) através do **FT-SALE/SMAM**, pelas plataformas de suporte a interoperação em uma EV [Rabelo01]. Em uma transação entre empresas componentes da **EV**, vários sistemas legados estarão envolvidos (tais como acessar a informação do chão de fábrica, atualizar no banco de dados, encapsular o conteúdo desejado no formato adequado, criptografar, e enviar informações).

- **Sistema HOLOS-MASSYVE** [Rabelo00] – Também é um SMA responsável por coordenar a produção do chão de fábrica de uma empresa componente. Atua como uma espécie de interlocutor entre as necessidades da EV com as disponibilidades do chão de fábrica de uma empresa componente. Suporta uma negociação entre seus agentes (“agentes de manufatura”) visando criar, dinâmica e temporariamente, a mais adequada composição de equipamentos para um dado plano de produção. As interfaces dos agentes de manufatura dependem do tipo de atividade executada pelos sistemas legados encapsulados por estes agentes [Rabelo01].

5.5.2.1 Comunicação entre Agentes

A **Plataforma FT-SALE/SMAM** não sugere alterações no sistema de agentes proposto em [Schmidt03]. Como visto anteriormente, a mesma é vista como uma base genérica de serviços de infra-estrutura, na qual os serviços de aplicação podem ser suportados no que se refere à comunicação.

Neste caso, os agentes existentes no **Sistema de Busca e Seleção de Parceiros** permanecerão cada qual com suas funções pré-definidas. No entanto, a nova plataforma sugere uma modificação no processo de Busca e Seleção, em relação à configuração dos agentes. A mudança está relacionada com a busca da lista de componentes do grupo. No modelo proposto, os agentes devem ser configurados para buscarem as informações sobre os possíveis parceiros, direto nas **IOGR locais** e não mais no **Gerenciador de Grupos**.

A existência do agente *Broker* não é alterada. Portanto, após a formação de uma EV, o mesmo poderá manter o controle sobre seus membros através dos módulos de aplicação (sistema *workflow*).

A linguagem **XML** é adotada no Sistema de Busca para prover comunicação entre os agentes, "empacotando" as mensagens da aplicação, que seguem o formato **KQML**, definindo os parâmetros do **KQML** através de *tags* **XML**. O uso do **KQML** tende a resolver o problema da interoperação semântica, porém não soluciona o da sintática, uma vez que o padrão **KQML** abstrai-se da implementação. O uso do **XML** possibilita que sistemas troquem dados em formatos chamados neutros [Schmidt03].

Portanto, a **Plataforma FT-SALE/SMAM** mantém, para a comunicação entre os agentes, à plataforma **XML**.

5.5.3 Sumário das Melhorias Introduzidas

Nesta seção, é feito um sumário sobre as melhorias introduzidas pela **Plataforma FT-SALE/SMAM** no aspecto tolerância a falhas. Também são resumidas as principais contribuições desta dissertação, bem como maiores esclarecimentos sobre a finalidade da plataforma proposta.

Como visto, a agilidade no processo de busca e seleção de parceiros em EVs é fundamental para atender às necessidades dos clientes, e ainda, para que a empresa possa atingir suas metas de prazo de entrega e aumentar sua competitividade no mercado.

O **Sistema de Busca e Seleção de Parceiros** [Schmidt03] sugere as seguintes soluções:

- Automatizar parte deste processo, fazendo com que as decisões tomadas pelo responsável, o *Broker*, sejam auxiliadas por dados coletados e pré-processados por agentes móveis e estacionários.

- Existência de mais de um *Broker* atuando dentro de uma **OV**, aumentando a confiança entre os membros do grupo e distribuindo a carga de processamento e as comunicações entre eles.

- Arquitetura híbrida de agentes, composta por agentes móveis e agentes estacionários, cada qual com suas devidas funções, tornando o processo mais ágil e flexível.

Entretanto, este sistema possui a seguinte vulnerabilidade: existência de um único **Gerenciador de Grupos**, o qual é responsável por armazenar toda e qualquer informação referente aos membros da **OV**.

Em caso de falha no **Gerenciador de Grupos**, tal vulnerabilidade é constatada, pois os seguintes problemas podem ocorrer:

- Ausência de técnicas de tolerância a falhas, impossibilitando a substituição do gerenciador faltoso;

- O processo de comunicação entre os agentes pode falhar em qualquer uma das fases, afetando todo o processo de Busca e Seleção;

- Perda de toda a informação sobre os membros da **OV**;

- Falta de confiabilidade dos processos, afetando a cooperação entre as empresas;

- Comprometimento da estrutura final da Empresa Virtual.

A integração da **Plataforma FT-SALE** com o **SMAM** gera uma plataforma que provê portabilidade, interoperabilidade e escalabilidade, **com suporte a sistemas multi-agentes**, garantindo maior tolerância a falhas e tornando o sistema de EV mais confiável, tanto na fase do processo de **Busca** e **Seleção** de parceiros, quanto na fase de **Operação**.

A plataforma **FT-SALE/SMAM** sugere a definição dos seguintes processos:

- Eliminação do ponto de vulnerabilidade do sistema de Busca e Seleção através da substituição de um único **Gerenciador de Grupos** pelos componentes de tolerância a falhas do serviço de nomes do FT-SALE (SN-L e SN-G);
- Replicação dos **SN-Ls** e **SN-Gs** de cada empresa, fornecendo maior tolerância a falhas;
- Atualização dinâmica das **IOGRs**;
- Identificação de atividades da Empresa Virtual, que pudessem ser adaptadas a forma de divisão dos grupos de domínios proposta na arquitetura FT-SALE;
- Identificação dos mecanismos e processos que fazem parte do domínio local de tolerância a falhas;
- Identificação dos mecanismos e processos que fazem parte do domínio global de tolerância a falhas;
- Identificação das funções a serem desenvolvidas pelas empresas, no caso de mudanças nos membros do serviço de nomes;

5.5.4 Quadro Comparativo

Nesta seção, é feita uma comparação entre as três plataformas discutidas neste trabalho. A Tabela 5.2 compara as três plataformas, com relação ao problema de busca e seleção de parceiros para a formação da EV. O quadro apresenta as vantagens obtidas na plataforma proposta.

	Sistema de Busca e Seleção de Parceiros		
Características	**[Schimdt03]**	**Plataforma FT-SALE**	**Plataforma FT-SALE/SMAM**
Controle de *Membership*	Sim	Sim	Sim
Técnica de Tolerância a Falhas	Não	Sim	Sim
Pontos de Vulnerabilidade	Sim	Não	Não
Suporte a Agentes	Sim	Não	Sim
Domínio de Tolerância a Falhas	Não	Sim	Sim

Tabela 5.2: Quadro Comparativo do uso do Sistema de Busca e Seleção de Parceiros em três diferentes plataformas.

5.5.5 Confiabilidade, Interoperabilidade e Escalabilidade da Plataforma Proposta

Esta seção procura destacar as vantagens obtidas na plataforma proposta, relacionadas a alguns aspectos de projeto, tais como confiabilidade e interoperabilidade, e ainda, discutir a escalabilidade da plataforma proposta.

Entre as melhorias trazidas pela plataforma proposta destaca-se o considerável aumento na confiabilidade do sistema em relação ao sistema de busca e seleção [Schmidt03], dado que no caso de uma falha em um *host* ou em um objeto, o mesmo pode ser imediatamente substituído (de forma transparente ao usuário) por uma de suas réplicas.

Uma outra melhoria esta relacionada com a interoperabilidade, devido a introdução de uma plataforma de suporte a agentes móveis na arquitetura FT-SALE, a qual provê a troca de mensagens de aplicação entre entidades, através do uso do formato **KQML** empacotadas em **XML,** estabelecendo assim uma padronização. Esta característica torna a plataforma proposta apta a suportar sistemas de agentes móveis, e ainda, tornam a solução independente de linguagem de implementação, como por exemplo, os diversos sistemas legados das empresas de uma OV.

Entretanto, nada se pode afirmar em relação a escalabilidade da plataforma, visto que a comunicação se dá baseada em grupos de objetos. Sabe-se que estes grupos estão sujeitos a sofrerem acréscimos, não somente no número de objetos pertencentes ao mesmo, mas também pode haver um aumento na quantidade de grupos componentes do sistema, como por exemplo, com a entrada de mais organizações no grupo de empresas de uma **OV**.

Por este motivo, a escalabilidade pode vir a ser um problema, pois, a pesar da plataforma FT-SALE/SMAM ser estruturada com base na plataforma FT-SALE, que por sua vez é uma plataforma apropriada para o uso em redes de larga escala, não podemos garantir a escalabilidade perante um aumento considerável de objetos e grupos de objetos. Ou seja, o sistema pode funcionar muito bem enquanto os grupos possuem poucos elementos e existe um número razoável de grupos, mas não se sabe e nada se pode afirmar sob a confiabilidade do sistema no caso de milhares de objetos e milhares de grupos. Ainda não é possível responder perguntas como: O que aconteceria se os grupos tivessem milhares de componentes? O que aconteceria se houvessem milhares de grupos? A resposta a estas perguntas depende da implementação total do sistema e, portanto, estão fora do escopo deste trabalho.

6. CONSIDERAÇÕES FINAIS

A cooperação com as **Empresas Virtuais** (**EV**) é uma solução encontrada por muitas organizações, para vencer desafios trazidos com a nova economia, as novas formas de comércio, os escritórios remotos, os novos padrões monetários e o aparecimento da Internet, entre outros. A **EV** tornou-se uma forma de cooperação que permite que empresas distribuam tarefas a fim de atender a uma **Oportunidade de Negócios** (**ON**). Entretanto, estas empresas possuem estruturas diversas, com propriedades heterogêneas. Portanto, são necessárias ferramentas que cubram tais aspectos.

Com o intuito de ajudar a melhorar a eficiência e a flexibilidade na fase de **Busca** e **Seleção** de parceiros para a formação de Empresas Virtuais, foi proposto em [Schmidt03] um sistema baseado em uma arquitetura híbrida de agentes, que "contempla as atividades de receber as especificações de uma oportunidade de negócio de um cliente, modelá-la, distribuí-la entre as empresas da **Organização Virtual** (**OV**), receber as propostas das empresas interessadas, avaliar as possíveis **EVs** e divulgar o resultado final às empresas proponentes".

O Sistema de Busca e Seleção de Parceiros utiliza agentes móveis para apresentar de forma mais ágil as Oportunidades de Negócio a um grupo de empresas, permitindo maior eficiência e engenhosidade na formação das EVs. Para obtenção das informações desejadas junto aos sistemas legados das empresas reais, são utilizados agentes estacionários.

O sistema utiliza um **Gerenciador de Grupos** que é responsável pelo controle dos membros do grupo de empresas. De acordo com [Schimdt03], o Gerenciador de Grupos "mantém uma lista dos membros que compõem o grupo, que além de dados da empresa sobre identificação e localização, possui informações sobre indicadores da empresa, por exemplo, se está operando normalmente ou se está com a capacidade comprometida. Isto permite uma melhor seleção prévia das empresas que serão consultadas, ou então, que a lista seja um reflexo mais atualizado possível da situação do grupo e seus membros".

Uma empresa que integra uma OV deve se registrar no Gerenciador de Grupos. Quando uma empresa deseja se retirar de uma OV, ela também deve informar o Gerenciador de Grupos. Por fim, o Gerenciador de Grupos mantém uma lista atualizada das empresas membro de uma

OV. Fica claro, portanto, a importância do Gerenciador de Grupos no Sistema de Busca e Seleção de Parceiros.

No entanto, a plataforma proposta em [Schmidt03] trabalha com um único **Gerenciador de Grupos,** o que não é aconselhável, visto que um único ponto de controle torna o sistema susceptível à falhas. Por exemplo, se o Gerenciador de Grupos falhar durante uma consulta de empresas, o *broker* não consiga adquirir a lista dos membros da **OV**, impedindo a identificação dos candidatos em potencial para o atendimento de uma nova **ON**. Além disto, não seria possível distribuir o pedido entre as empresas que se encaixam no perfil desejado. Conseqüentemente, este problema comprometeria todos os passos seguintes do processo de busca e seleção de parceiros.

Outra situação possível envolve o agente móvel, que tem como função deslocar-se para as empresas em busca de informação para a formação da EV. Para isso é necessário que este agente saiba a localização de cada empresa na rede para as quais pretende migrar. Para resolver este problema, o Sistema de Busca e Seleção de Parceiros utiliza o serviço de nomes do CORBA, que não é tolerante a falhas. Se este serviço falhar, todos os objetos e grupos de objetos ficariam inacessíveis. Isto implica que não haveria possibilidade de uma recuperação imediata da lista dos membros do grupo. Desta forma, não haveria como um agente móvel saber a localização de cada empresa na rede para as quais pretende migrar.

A tolerância a falhas no **CORBA** é obtida através da replicação de objetos, técnicas de detecção e recuperação de falhas, aumentando assim a confiabilidade no sistema e a disponibilidade dos recursos. No entanto, as especificações **FT-CORBA** apresentam algumas dificuldades ao serem aplicadas em redes de larga escala. As especificações **FT-CORBA** são limitadas em termos de detecção de falhas e gerenciamento de replicação, que são aspectos essenciais para dar confiabilidade na comunicação entre objetos em um ambiente amplamente distribuído e heterogêneo como o ambiente de uma Empresa Virtual.

Estas limitações incentivaram o desenvolvimento da **Plataforma FT-SALE**, que faz uma adaptação às especificações **FT-CORBA**, sem realizar modificações nas interfaces do padrão **CORBA**. Esta plataforma utiliza como base a solução desenvolvida pelo **GrouPac 2.0** para a implantação de técnicas de tolerância a falhas em redes de larga escala, juntamente com a

plataforma **SALE**, cuja arquitetura integra serviços com diferentes tipos de garantia (desempenho, confiabilidade e segurança), se tornando adequado para Empresas Virtuais.

Neste trabalho apresentou-se a **Plataforma FT-SALE/SMAM**, que visa inserir tolerância a falhas na **Plataforma de Busca e Seleção de Parceiros** para Empresas Virtuais. Como base para esta proposta, foi utilizada a **Plataforma FT-SALE**. A **Plataforma FT-SALE/SMAM** integra a **Plataforma FT-SALE** com a **Plataforma de Busca e Seleção de Parceiros na Formação de Empresas Virtuais,** gerando uma nova plataforma que provê portabilidade, interoperabilidade e escalabilidade, com suporte a **Sistemas de Multi-Agentes Móveis** (**SMAM**), garantindo maior tolerância a falhas e tornando as Organizações Virtuais mais confiáveis, tanto na fase do processo de Busca e Seleção de parceiros, quanto na fase de Operação.

A **Plataforma FT-SALE/SMAM** soluciona os problemas relativos à existência de um único Gerenciador de Grupos no **Sistema de Busca e Seleção de Parceiros** implementando técnicas de tolerância a falhas nos serviços de nome do CORBA, e ainda implementando serviços de nome em cada empresa da **OV**. Dentre os componentes necessários para esta implementação encontram-se os **SN-Gs** e **SN-Ls** e suas réplicas. O **SN-G** possui uma lista dos membros do grupo da **OV**. O **SN-L** possui uma cópia desta lista, assim como suas réplicas. Esta lista é alterada dinamicamente, tanto na entrada de um novo membro, quanto na saída de um membro do grupo. Por este motivo, o *broker* possui uma lista sempre atualizada dos membros da **OV**.

A presença dos elementos da **Plataforma FT-SALE** permite que o **Gerenciador de Grupos** seja eliminado do sistema, e sua função de fornecer ao *broker* a lista de membros da **OV**, seja realizada pelo **SN-L,** através da **IOGR** que pode ser acessada pelo *broker* diretamente. Portanto, se ocorrer uma falha no **SN-L**, o mesmo pode ser imediatamente substituído por uma de suas réplicas mais próximas. Isto é possível, porque o *broker* poderia solicitar uma cópia da **IOGR** ao **SN-L** substituto, possibilitando assim que *broker* identificasse os candidatos em potencial para a produção da nova **ON**.

Quando comparado aos trabalhos anteriores, a **Plataforma FT-SALE/SMAM** fornece os seguintes benefícios: eliminação do ponto de vulnerabilidade do Sistema de Busca e Seleção de Parceiros através da substituição de um único **Gerenciador de Grupos** pelos componentes de

tolerância a falhas do serviço de nomes do FT-SALE (SN-L e SN-G); replicação dos **SN-Ls** e **SN-Gs** de cada empresa, fornecendo maior tolerância a falhas; suporte à **Sistemas de Multi-Agentes Móveis** (**SMAM**), que agilizam o processo de apresentação de Oportunidades de Negócio a um grupo de empresas e melhoram a confiabilidade do sistema. Este benefício não está disponível na Plataforma FT-SALE; combina técnicas de tolerância a falhas CORBA com SMAs e SMAMs, abrindo portas para futuras inovações.

A dissertação também apresentou e discutiu alguns dos aspectos mais importantes por de trás da **Plataforma FT-SALE** e da **Plataforma de Busca e Seleção de Parceiros na Formação de Empresas Virtuais**, detalhando os elementos, funcionalidades e tecnologias utilizadas nestas plataformas. Além disto, foi feita a apresentação e discussão do papel da arquitetura **CORBA** e dos **Sistemas de Multi-Agentes** nestas plataformas. O suporte a tolerância a falhas foi discutido nestas tecnologias e plataformas. Também foram fornecidas algumas diretrizes para a implementação da plataforma proposta.

6.1 Sugestões para Trabalhos Futuros

Como sugestão para trabalhos futuros tem-se a implantação de um conjunto de melhorias e extensões do sistema existente atualmente, as quais estão descritas nos parágrafos seguintes. Tais melhorias estão relacionadas com as limitações observadas na plataforma proposta.

A extensão a outras fases da Empresa Virtual, que não a fase de criação, é um aspecto a ser considerado. A arquitetura FT-SALE/SMAM parece permitir a realização desta extensão, podendo dar suporte também a fase de **operação/evolução** de uma Empresa Virtual, visto que a mesma oferece um conjunto de serviços de infra-estrutura para permitir uma adequada execução de serviços no nível de aplicação.

A inserção de uma plataforma de suporte a **Sistemas Multi-Agentes** no contexto da **Plataforma FT-SALE** abre outras oportunidades que podem ser exploradas em trabalhos futuros, tais como estudos voltados para o desenvolvimento mais aprofundado e genérico de serviços de ontologia, para serem disponibilizados para uso dos Sistemas Multi-Agentes suportados pela plataforma, principalmente os sistemas voltados para aplicações em Empresas

Virtuais. Tais serviços incluiriam processos genéricos de criação de ontologias específicas para o ambiente de Empresas Virtuais e outros ambientes, além de serviços de tradução para ontologias já pré-existentes. A possibilidade de um agente ser capaz de avaliar atributos não numéricos, baseado em conceitos comuns, tornaria mais fácil a eliminação da necessidade da existência do *Broker* humano no processo de Busca e Seleção, o que seria de grande ajuda para a medida do desempenho do sistema e a comprovação dos benefícios da utilização de agentes móveis, visto que a necessidade de especialistas humanos na tomada de decisões dentro do processo de formação de uma EV prejudica estas análises.

A plataforma FT-SALE/SMAM propõe a criação de grupos interdomínios, os quais exigem um suporte mais complexo de comunicação, ainda não oferecido pela plataforma proposta, portanto estudos sobre comunicação de grupo complementariam a solução de detecção de falhas apresentada.

Finalmente, considera-se como proposta para trabalhos futuros a implementação da arquitetura proposta, a fim de comprovar na prática a viabilidade de sua utilização em aplicações distribuídas que embutem a noção de mobilidade de código, e, por conseguinte, validar a plataforma aqui proposta. Também como trabalhos futuros pode-se citar estudos e análises que visem simplificar e otimizar a plataforma FT-SALE/SMAM.

REFERÊNCIAS BIBLIOGRÁFICAS

[AGLETS02] AGLETS; 2002. **Aglets**. http://www.trl.ibm.com/aglets/index_e.htm;

[ARPA] ARPA. "***Advanced Research and Projects Agency***". Atualmente, chamada de DARPA.

[Arnold95] Arnold, O.; Hartling, M. "***Virtuelle Unternehmen: Begrifsbildung und – diskussion***". Arbeitspapier der Reihe Informations- und Kommunikations system als Gestaltungselement Virtueller Unternehmen,1995, n.3, 38 p.

[Batalha01] Batalha, Marcela Santana Guimarães. "**Serviço CORBA de Diagnóstico de Falhas**". Dissertação Mestrado. UFPB, Campina Grande, 2001.

[Bessani02] Bessani, Alysson Neves "**O Padrão UMIOP como base para comunicações de Grupo Confiável em Sistemas Distribuídos de Larga Escala**", Dissertação de Mestrado, UFSC, Florianópolis – SC, 2002.

[Byrne93] Byrne**,** J. A.; Bradt, R.; Port, O. "***The Virtual Corportation"***. Business Week, p. 41. 1993.

[Camarinha99] Camarinha, Matos; Afsarmanesh, H; 1999**. "*The Virtual Enterprise Concept. In Infrastructures for Virtual Enterprises: Networking Industrial Enterprises"***. IFIP TC5 WG5.3/PRODNET Working Conference on Infrastructures for Virtual Enterprises (PROVE' 99) p. 3-14, October 27-28, 1999, Porto, Portugal Kluwer Academic Publishers

[Cummins02] Cummins, Fred A., "**Integração de Sistemas – EAI – *Enterprise Application Integration***", Ed. Campus 2002.

[Davidow92] Davidow**,** W. H.; Malone, M. S. "***The Virtual Corporation: Structuring and revitalizing the corporation for the 21st century"***. (HarperBusiness), New York, 1992.

[Fraga01a] Fraga Joni S; Rabelo, Ricardo J.;. Siqueira, Frank [a], "**Suporte a Sistemas Abertos de Larga Escala para Empresas Virtuais**". Departamento de Automação e Sistemas – UFSC – 2001

[Fraga01b] Lung, Lau C.; Fraga, Joni S.; Padilha, Ricardo; Souza, Luciana; **"Adaptando as Especificações FTCORBA para Redes de Larga Escala"**, XIX Simpósio Brasileiro de Redes de Computadores – SBRC'2001, SBC, Florianópolis – SC. 2001.

[Ferrari99] Ferrari, Débora Nice **"Um Estudo sobre Mobilidade em Sistemas Distribuídos"** Trabalho de conclusão do Curso de Pós-Graduação em Ciência da Computação - UFRG – Porto Alegre, fevereiro de 1999.

[Filho00] Filho, Roberto Silveira Silva; "**Uma Arquitetura Baseada em CORBA para *Workflow* de Larga Escala**" Dissertação de Mestrado - UNICAMP, Campinas – SP - Agosto de 2000

[Fioreze03] Fioreze, Tiago; Porto, Ingrid Jansch; Granville, Lisandro Zambenedetti, "**Tolerância a Falhas em Sistemas de Agentes Móveis**", Anais 1ª Escola Regional de Redes de Computadores (ERRC 2003), Porto Alegre, Brasil, Set. 2003

[FIPA] FIPA. "***Foundation for Intelligent Physical Agents***". www.fipa.org.

[Fleischhauer98] Fleischhauer , L. "**Agentes"**. Dissertação de Mestrado, UFSC, Agosto, 1998.

Disponível em: http://www.eps.ufsc.br/disserta97/amaral/cap3.htm

[FT-CORBA01] OMG, "**Especificação FT-CORBA**". Disponível em:

ftp://ftp.omg.org/pub/docs/formal/01-09-29.pdf.

[Gomes00] Gomes, Antônio Sérgio Ribeiro **"Contribuições ao Estudo de Redes de Agentes"** Tese de Mestrado, Universidade Estadual de Campinas – Faculdade de Engenharia Elétrica e de Computação, Campinas – SP – Brasil, Junho, 2000.

[Goranson95] Goranson, T. ***Agile Virtual Enterprise: Best Agile Practice Reference Base.*** Working Draft, Janeiro, 1995 11p.

[Gray78] J. Gray, "***Notes on Database Operating Systems. In Operating Systems****" Na Advanced Course*, pp 10-17. Lecture Notes in Computer Science (60), Springer-Verlag. 1978.

[Green97] Green, S., Hurst, L., Nangle, B., Cunningham, P., Somers, F., Evans, R., "***Software Agents: A Review***", Relatório técnico TCD-CS-1997-06, 1997.

[Greve01] Greve, Fabíola; Narzul, Jean-Pierre ê **" Um Protocolo de Validação Atômica Não-Bloqueante"** *5th International Symposium on Autonomous Decentralized Systems, March 2001*.

[Griese92] Griese**,** J. "***Auswirkungen globaler Informations- und Kommunikationssysteme auf die Organisation weltweit tätiger Unternehmen*****",** Managementforschung n.2, Berlin-New York., 1992.

[Guerraoui95] R. Guerraoui, A. Schiper, "***The Decentralized Non-Blocking Atomic Commitment Protocol****"*. In 14th IEEE *International Symposium on Parallel and Distributed Processing,* pp. 2-9, San Antonio, October 1995.

[Hubner03] Hubner, Jomi Fred, "**Um Modelo de Reorganização de Sistemas Multi-Agentes**", Tese Doutorado, Escola Politécnica da Universidade de São Paulo. São Paulo, 2003

[Katzy97a] Katzy**,** B. R.“***Optmierung der Wertschöpfungskette***”, Seminário da Universidade de St. Gallen, Fevereiro, Suíça. 1997a

[Katzy97b] Katzy, B. R. “***Das Konzept der Virtuellen Fabrik***”, Seminário da Universidade de St. Gallen, Fevereiro, Suíça, 1997.

[Kazuhiro04] Ueda, Leo Kazuhiro “**Composição e Performance Musical Utilizando Agentes Móveis**”, Dissertação de Mestrado, Instituto de Matemática e Estatística da Universidade de São Paulo – São Paulo – SP, Outubro, 2004.

[Keider95] I. Keider, D. Dolev, “***Increasing the Resilience of Atomic Commit*****, at No Additional Cost**”, ACM PODS’1995: Principles of Database Systems, pp 245-254, May 1995.

[Kosanke92] Kosanke**,** K. “***Unternehmensmodellierung in dynaminschen Netzwerk***”. Seminário da Universidade de St. Gallen, Setembro, Suíça, 1996.

[Lange97] Lange, Danny; Oshima, Mitsuru; et al. **“Aglets: programming Mobile Agents in Java”**. *Worldwide Computing and its Applications* (WWCA’97). International Conference. Proceedings. Tsukuba, Japan, March, 1997.

[Laurindo01] Laurindo, F. J. B., “**Tecnologia da informação como suporte às estratégias empresariais**”, I Workshop: Redes de Cooperação e Gestão do Conhecimento, PRO-EPUSP, 2001.

[Lima01a] Silva, Cristiane M. e Lima, Susiléa A. S.*, “***Adaptando as Especificações FT-CORBA à Plataforma SALE**”, INATEL, UFES, Vitória – ES, 2001.

[Lima01b] Lima, Susiléa Abreu dos Santos. “**Uma Proposta para Inserir Tolerância a Falhas em Ambientes de Empresas Virtuais**”.

Dissertação de Mestrado. UFV, Vitória – ES, 2003.

[LiXin99] Li Xin, "**Relatório Técnico**", Capítulo 2. Disponível em: http://paginas.fe.up.pt/~eol/MARSiMA/RELATORIO/Teoria.html

[Lobato05] Lobato, Cidiane Aracaty, **"Um *Framework* Orientado a Aspectos para Mobilidade de Agentes de Software"**, Dissertação de Mestrado, PUC-RJ, Dezembro de 2005.

[Loss03] Loss, Leandro; **"Integração de Sistemas Multiagente Industriais: Uma Proposta baseada no intercâmbio de Informações entre Padrões"**. Dissertação de Mestrado – Centro Tecnológico, UFSC. Florianópolis, 2003.

[Lung01] Lung, Lau C; "**Experiências com Tolerância a Falhas no CORBA e Extensões ao FT-CORBA para Sistemas Distribuídos de Larga Escala**", Tese Doutorado, UFSC, Florianópolis – SC, 2001.

[Macedo01] Macedo, Ana Paula C. R; **"Metodologias de Negociação em Sistemas Multi-Agentes para Empresas Virtuais"** Universidade do Porto – FEUP, Dezembro, 2001.

[MASSYVE02] MASSYVE; 2002. **Massyve**. http://www.gsigma-grucon.ufsc.br/massyve/;

[Mendonça02] Mendonça, Fabio da Silva; **"Tolerância a Falhas em Sistemas Distribuídos**; **A abordagem do CORBA/FT"** Universidade Tiradentes – Aracaju 2002.

[Mertens95] **Mertens,** P.; Faisst, W., **"*Virtuelle Unternehmen – eine Organisationsform für die Zukunft?*"**, Revista alemã *Technologie & Management*, 1995 n. 44, p. 61-68.

[Montez97] C. Montez, "**Um Modelo de Programação para Aplicações de Tempo Real em Sistemas Abertos**", Monografia do Exame de

Qualificação de Doutorado, DAS, UFSC, Julho de 1997.

[Mowshowitz86] Mowshowitz, A. "***Social Dimensions of Office Automation***", Yovitz, *Avances in Computers* 1997, n.25, p.335-404.

[Neches94] NECHES, R.; "***The Knowledge Sharing Effort***". Disponível em: http://www-ksl.stanford.edu/knowledgesharing/papers/kse-overview.html; 1994.

[Netage96] Netage, "***Four Ages of Organization"***. Disponível em: http://www.netage.com/Knowledge/NetAge_bk/age.html (15 Nov)1996.

[OMG00] OMG. "**Mobile Agent Facility Specification**". OMG Document 2000-01-02;

[OMG02] OMG. "**CORBA V 2.6.1**". Disponível na Internet em:

http://www.omg.org/cgi-bin/doc?formal/02/05/08

[OMG03] OMG; 2003. "**Object Management Group**" – http://www.omg.org/;

[Rabelo00] Rabelo, Ricardo & Klen, Alexandra A. Pereira. "**Escalonamento Cooperativo Interorganizacional**". G-SIGMA, UFSC, Florianópolis, 2000.

[Rabelo01] Rabelo, Ricardo & Klen, Alexandra A. Pereira. **"Suporte a Sistemas Abertos de Larga Escala para Empresas Virtuais"**. G-SIGMA, UFSC, Florianópolis, 2001.

[Rabelo02] Rabelo, Ricardo & Klen, Alexandra A. Pereira. **"*A Multi-agent System for Smart Coordination of Dynamic Supply Chains – In Collaborative Business Ecosystems and Virtual\Enterprises*"** – 3rd Working Conference on Infrastructures for Virtual Enterprises, IFIPTC5/WG5.5 (PRO-VE'02). Sesimbra, Portugal. P. 379-386. Kluwer Academics Publisher; 2002.

[Rentes96] Rentes, A. F.; Souza, F. B.; Campeão, P.; Suga, R. A. "***Uma Proposta de uma Metodologia de Integração de Empresas***". Congreso Latino Americano de Administracion – XXXI Asamblea Anual CLADEA, Santiago, Chile, 1996.

[Reverbel06] Reverbel, Francisco C. R; "**Uma Introdução à Arquitetura CORBA**" nota de aula – ME-USP – Departamento de Ciência da Computação – USP – SP – 2006

[Ricciardi91] A. Ricciardi and K. Birman, **"*Using Process Groups to Implement Failure Detection in Asynchronous Systems*"**. *In Tenth ACM Symposium on the Principles of Distributed Computing. August*, 1991.

[Rozenfeld96] Rozenfeld, H**. "*Para Integrar a Manufatura é Importante o Domínio dos Business Processes*"**. Revista Máquinas e Metais, Outubro, 1996.

[Russell02] Russell, Stuart J. and Norvig, P. "***Artificial Intelligence: A Modern Approach***". Prentice Hall, 2nd Edition, 2002.

[Scheer95] Scheer**,** A.-W. "***Wirtschaftsinformatik - Referenzemodelle für industrielle Geschäftsprozesse***", Springer Verlag, 1995 Alemanha.

[Schmidt03] Schmidt, Ricardo. "**Busca e Seleção de Parceiros para Empresas Virtuais: Uma abordagem baseada em agentes móveis**". Dissertação Mestrado. UFSC, Florianópolis, 2003.

[Souza98] Souza, Anamélia Contente de; Mazzola, Vitório Bruno; "**Implementando Aplicações Distribuídas Utilizando CORBA: Um Estudo de Caso Voltado à Área de Banco de Dados**" INE – UFSC – Campus Universitário - Florianópolis – SC.

[Teixeira00] Teixeira, José Helvécio; Moraes, Luís Felipe M. de; Teixeira, Suzana Ramos "**Uma Abordagem para o Desenvolvimento de Aplicações Distribuídas em CORBA usando C++ e ORBIX**" Relatório Técnico –

COPPE/UFRJ –2000. Disponível em: http://www.ravel.ufrj.br/arquivosPublicacoes/helvecio_rel_corba.pdf

[Toffler80] Toffler, A. "***A Terceira Onda"***, Editora Record, Rio de Janeiro, 1980.

[Wangham03a] Wangham, Michelle Silva; Fraga, Joni da Silva; Santin, Altair Olivo; "**Usando Certificados SPKI/SDSI para geração de domínios de execução de Agentes Móveis**", LCMI-DAS, UFSC, Florianópolis, 2003.

[Wangham03b] Wangham, Michelle Silva; Fraga, Joni da Silva; Santin, Altair Olivo; "**Mecanismos de Segurança para Plataformas de Agentes Móveis, baseados em Redes de Confiança SPKI/SDSI**". Simpósio Brasileiro de Redes de Computadores (SBRC), Natal, 2003.

[Weber02] WEBER, TAISY SILVA **"Tolerância a Falhas: Conceitos e Exemplos"** Programa de Pós-Graduação em Computação - UFRGS – 2002.

[Williams95] Williams**,** T.J.; Bernus, P.; Nemes, L**. "*The Concept of Enterprise Integration*",** Anais do IFIP TC5 Working Conference on Models and Methodologies for Enterprise Integration, Australia, 1995. p.8-19.

[Wooldridge99] Wooldridge, M. (1999) **"*Multiagent Systems*"**, *Chapter 1: Intelligent Agents*. The MIT press.