UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
Convênio UCS/UFRGS

Proposta de Modelo de Agente EDI para uso como ferramenta de apoio a Sistemas de Informação baseado em Web

por

HENRIQUE OLIVEIRA DA SILVA

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Ciência da Computação

Prof. Dante Augusto Couto Barone

Orientador

Porto Alegre, janeiro de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Silva, Henrique Oliveira da
Proposta de Modelo de Agente EDI para uso como ferramenta de apoio a Sistemas de Informação baseado em Web / por Henrique Oliveira da Silva. – Porto Alegre: PPGC da UFRGS, 2002.
8 p.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2002. Orientador: Barone, Dante Augusto Couto.

1. Sistemas de informação. 2. Agentes Web. 3. EDI baseado em Web. 4. Sistemas multiagentes. I. Barone, Dante Augusto Couto. II Título.

Sumário

Lista de Abreviaturas

ASP	*Active Server Pages*
B2B	*Business to Business*
B2C	*Business to Consumer*
BBS	*Bulletin Board System*
BD	Banco de Dados
CGI	*Common Gateway Interface*
DFD	Diagrama de Fluxo de Dados
EDI	*Electronic Data Interchange*
HTML	*Hyper Text Markup Language*
HTTP	*Hyper Text Transport Protocol*
IAD	Inteligência Artificial Distribuída
INSS	Instituto Nacional de Seguridade Social
IP	*Internet Protocol*
JDBC	*Java Database Connectivity*
J2EE	*Java to Enterprise Edition*
JDK	*Java Development Kit*
JRE	*Java Runtime Environment*
JSP	*Java Server Page*
KQML	*Knowledge Query and Manipulation Language*
LAN	*Local Area Network*
MIME	*Multipurpose Internet Mail Extensions*
ODBC	*Open Database Connectivity*
OTP	*Open Trading Protocol*
S-HTTP	*Secure Hyper Text Transfer Protocol*
SBDD	Sistema de Banco de Dados Distribuído
SGBD	Sistema Gerenciador de Banco de Dados
SI	Sistema de Informação
SIG	Sistema de Informação Gerencial
SMA	Sistema Multiagentes
SOHO	*Small-Office/Home-Office*
SER	Serviço de Resolução de Endereços
SSL	*Secure Socket Layer*

TCO	*Total Cost Ownership*
TCP	*Transmission Control Protocol*
URL	*Uniform Resource Locators*
XML	*eXtensible Markup Language*

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho apresenta a proposta de uma arquitetura e o modelo de um Agente de Intercâmbio Eletrônico de Dados, Agente EDI, cuja função é, permitir a troca de dados estruturados entre Sistemas de Informações Distribuídos através da Internet. A estratégia de interação dos agentes possibilita uma maneira alternativa de tratar a recuperação, o armazenamento e a distribuição de dados, permitindo assim, o desenvolvimento de um modelo de Sistema de Informações baseado em Web, igualmente proposto neste trabalho. É apresentado também o desenvolvimento do Agente EDI proposto. O qual poderá ser utilizado por entidades que necessitam disponibilizar ou recuperar dados estruturados via Web, como por exemplo: informações de produtos, listas de preços, dados cadastrais, etc. A relevância deste trabalho está no fato de apresentar uma tecnologia simples e acessível, capaz de ser implementada sem a necessidade de altos investimentos e capaz de facilitar a implementação de Sistemas Distribuídos via Internet.

Palavras-chave: Sistemas de Informação, Agentes Web, EDI baseado em WEB, Sistemas Multiagentes.

TITLE: "AN EDI AGENT MODEL TO SUPPORT WEB INFORMATION SYSTEMS"

Abstract

This work shows a proposal of an architecture and the Agent Electronic Data Interchange model, Agent EDI, whose function, is to allow the structured data interchange between Distributed Information Systems through the Internet. The Agents interaction strategies allow an alternative way to handle the recovery, storage and data sharing. This allows the development of the Information Web System model proposed in this work. It is also showed the deployment of the Agent EDI proposed. This will be used for entities that needs to share or to retrieve structured data through the Internet, for example: product information, cost lists, data forms, etc. The relevance of this work is in the fact that it shows a simple and accessible technology, able to be implemented without need of high investment and able to facilitate the implementation Distributed Systems through the Internet.

Keywords: Information System, Agents Web, Web based EDI, Multi Agents Systems.

1 Introdução

Atualmente, no meio popular, a Web é sinônimo de Internet. Tal afirmação induz a conclusão de que a maior rede da Internet é a Web. Se esta não for a maior pelo menos é a mais significativa para a comunidade em geral.

A Web nos últimos anos tem provocado uma revolução em todos os setores. Esta revolução se torna perceptível quando se observa o desenvolvimento que áreas distintas tem alcançado com o auxílio da Web. Como meio de troca de informações, ela despertou a criatividade dos usuários que passaram a enxergar novos fins para a rede. Hoje a Web é mais do que um grande repositório de documentos HTML. Ela nos permite efetuar transações comerciais, trocar conhecimento com qualquer parte do planeta, enviar e receber documentos hipermídia, além de muitas outras funcionalidades que surgirão no futuro. [JAM99] [SEA 90]

Mesmo assim, muito pouco das possibilidades de um sistema distribuído, como o da Internet, tem sido aproveitado. A Web tem servido principalmente para fins de Marketing e divulgação de material científico. Aos poucos as organizações começam a praticar o Comercio Eletrônico. Muitas novas tecnologias tem sido propostas para atender a novas funcionalidades. O ponto em comum entre as diferentes utilidades da Web é a necessidade de utilização de dados estruturados. [YIX 98]

Em um Sistema de Informações tradicional, os dados são armazenados de forma centralizada. A estrutura Cliente/Servidor consiste em um computador central que mantêm todos os dados armazenados e serve às estações de trabalho. Essa realidade se modificou com as estruturas de redes, nas quais, computadores independentes conectam-se entre si compartilhando recursos. Neste novo ambiente, os dados podem ser armazenados de forma distribuída. A área da Ciência da Computação que estuda os chamados Sistemas de Banco de Dados Distribuídos, SBDD, está baseada nesse conceito e tenta mostrar as vantagens da utilização de tal recurso, mas, também, alerta para todas as peculiaridades inerentes ao projeto de um SBDD. Tais peculiaridades, entre as quais, o alto grau de complexidade de um projeto de SBDD, tem inibido os projetistas a investir no desenvolvimento desses sistemas.

A maioria dos agentes da Web costuma tratar informações desestruturadas encontradas em documentos HTML. Atualmente tecnologias como *Active Server Pages*, ASP, e *Java Server Pages*, JSP, são utilizadas em sites da Web para a construção dinâmica de páginas HTML que contêm informações estruturadas, recuperadas a partir de uma solicitação de consulta que é realizada localmente através de um Browser. Porém essas tecnologias não suprem satisfatoriamente as necessidades de sistemas que manipulam dados estruturados.

Apesar de, dados estruturados serem utilizados nas principais aplicações da Internet, como por exemplo, o Comércio Eletrônico, muito pouca tecnologia têm sido sugerida para aperfeiçoar este processo. Isto, provavelmente, se deve ao fato dos responsáveis pela manutenção destes dados possuírem um conceito enraizado e fundado nos sistemas de computação tradicionais, que funciona perfeitamente até hoje em dia. Fato este que lhes faz considerar não ser necessário qualquer alteração neste processo. [BOR 98] [STR 98]

A infra-estrutura da Internet pode ser utilizada para troca de dados estruturados, sob esse aspecto, ela pode ser considerada como um grande SBDD heterogêneo de fato, onde o intercambio de dados ocorre entre diferentes entidades. Porém ao contrário de um SBDD tradicional, a responsabilidade pela manutenção dos dados é descentralizada. Instituições com Atividades-fim, Procedimentos e Normalização, Sistemas de Informações Gerenciais, SIG, Sistemas Gerenciadores de Banco de Dados, SGBD, e Plataformas Operacionais diferentes são responsáveis pela manutenção e intercâmbio de seus dados. Nesse cenário, os mesmos dados são armazenados por entidades distintas de forma redundante e inconsistente. Esse problema poderia ser solucionado com a possibilidade de recuperação de dados diretamente da fonte geradora sempre que necessário. Por exemplo, os dados referentes a um determinado produto poderiam ser recuperados diretamente do *site* da indústria que produz o produto, sem que fosse necessária a navegação pelo site da empresa através de um Browser. Esse recurso descartaria a necessidade de ter-se que armazenar estas informações em um Banco de Dados, BD, próprio, criado com a finalidade de obter um ambiente de trabalho mais eficiente, porém, mais oneroso.

A Internet permite que informações sejam obtidas por meio de consulta aos dados de forma distribuída. O único problema é a heterogeneidade dos dados de SGBD comerciais, que pode ser contornado pela conversão destes dados, no momento do intercâmbio de dados, em documentos HTML ou XML. [MCG 99] [GRA 98]

Muitas soluções tem sido apresentadas ao longo do tempo para tratar do intercâmbio de dados distribuídos entre entidades distintas que estão conectadas a Internet, porém, a que tem se mostrado mais atraente é a utilização de agentes.

A solução de agentes consiste basicamente no seguinte cenário: cada entidade possui um agente que funciona como uma interface de intercâmbio de dados com outros agentes do mesmo tipo. A interação entre os agentes cria uma rede dinâmica de intercâmbio de informações. Esta solução possui a vantagem de executar a tarefa de forma transparente ao usuário.

Este trabalho apresenta a arquitetura e o modelo de um Agente de Intercâmbio Eletrônico de Dados, chamado de Agente EDI[1] (*Electronic Data Interchange*), cuja função é, permitir a troca de dados estruturados entre Sistemas de Informações Distribuídos através da Internet. Esse agente será uma ferramenta de apoio à Sistemas de Informação baseados em Web. [SIL 01]

Os Agentes possuem um protocolo de troca de dados comum. As informações obtidas de outros Agentes são convertidas para o formato padrão HTML ou XML, que podem ser facilmente manipulados pela entidade.

A estratégia de interação dos Agentes possibilita uma maneira alternativa de tratar a recuperação, o armazenamento e a distribuição de dados, permitindo assim, o desenvolvimento de um modelo de Sistema de Informações baseado em Web, que

[1] No decorrer do trabalho, o termo Agente, iniciado com letra maiúscula, refere-se aos Agentes EDI apresentados no trabalho, enquanto que, agentes, iniciado com letra minúscula, refere-se a agentes em geral.

utiliza esses Agentes como ferramenta de apoio nas operações de intercâmbio de dados estruturados.

Este modelo, além de evitar o problema de redundância e inconsistência, também permite uma redução do *Total Cost Ownership*, TCO, das entidades.

O primeiro passo nesse trabalho é apresentar uma visão geral sobre agentes onde são abordados alguns aspectos relevantes quanto ao projeto de agentes. Antes de especificar o projeto do agente proposto nesse trabalho, é apresentada uma visão geral sobre Sistemas de Informação. Nessa abordagem serão tratados os aspectos relevantes para a fundamentação do modelo de Sistema de Informação baseado em WEB a ser proposto e especificado no decorrer do trabalho. A viabilização desse modelo depende da utilização de um agente de software classificado como um Agente EDI, que é o objeto de estudo desse trabalho. Por fim, o trabalho apresenta uma proposta de arquitetura e o modelo desse agente.

2 Agentes IAD e Agentes Web – Uma visão geral

2.1 Definição

Para definir agentes, Michael Knapik [KNA 98] (pp 3) cita vários pontos de vistas diferentes, encontrados na comunidade da Ciência da Computação, utilizados principalmente pelos pesquisadores de Inteligência Artificial, IA. Uma delas é a definição utilizada por J. A. King (1995, pp 17-19):

"Um agente inteligente é considerado ser um computador substituto para uma pessoa ou processo que cumpre um estado necessário ou ativo. A entidade substituta provê as capacidades de tomada de decisão que são similares as intenções descritas por um humano..." [KNA 98]

Além dessa, são citadas as definições de Russel e Norvig que definem agentes sob três aspectos cuja ordem indica o aumento da sofisticação, ou inteligência:

"1 – Em um genérico, mas poderoso sentido comportamental: 'Um agente é qualquer coisa que possa ser vista, assim como percebida pelo seu ambiente através de sensores e agindo sobre esse ambiente através de seus efeitos.' (Norvig e Russel, 1994)". [KNA 98]

"2 – Com a adição de racionalidade: 'Para cada seqüência perceptível possível, um agente racional ideal poderia fazer o que fosse esperado para maximizar sua medida de performance, baseado nas evidências providas pela seqüência percebida e pelo conhecimento adquirido que o agente tinha.' (Norvig e Russel, 1994)". [KNA 98]

"3 – Um agente racional dá o próximo passo para o comportamento inteligente se ele pode agir autonomamente. Em outras palavras: 'Um agente é autônomo para que a extensão suas ações e escolhas dependam de suas próprias experiências, de preferência ao conhecimento do ambiente que é construído pelo projetista.' (Norvig e Russel, 1994)". [KNA 98]

Ainda segundo Michael Knapik uma definição mais abrangente é dada por Atkinson:

"Agentes inteligentes são entidades de software que realizam um conjunto de operações em nome de um usuário ou de outro programa com algum grau de independência e autonomia, e ao fazê-lo, emprega algum conhecimento ou representação das metas ou desejos do usuário. Agentes inteligentes podem então ser descritos em termos de um espaço definido pelas duas dimensões de agência e inteligência. [...] Agência é o grau de autonomia e autoridade investida no agente, e pode ser medida, no mínimo qualitativamente, pela natureza da interação entre o agente e outras entidades no sistema. No mínimo, um agente pode rodar de forma assíncrona. O grau de agência é incrementado se um agente representa um usuário de alguma forma. Esse é um dos valores chaves do agente. Um agente mais avançado pode interagir com outra entidades tais como dados, aplicações ou serviços. Um agente mais avançado ainda colabora e negocia com outros agentes. [...] Inteligência é o grau de raciocínio e comportamento aprendido: a habilidade do agente de aceitar as declarações

de metas do usuário de executar as tarefas delegadas a ele... Níveis mais altos de inteligência incluem um modelo do usuário ou outras formas de compreensão e raciocínio sobre o que um usuário procura fazer. No topo da escala inteligência estão sistemas que aprendem e se adaptam ao seu ambiente... (Atkinson et al., 1995)" [KNA 98]

Para Pattie Maes, os agentes podem ser utilizados de diferentes maneiras reduzindo assim o trabalho e a sobrecarga de informações. Eles deixam a complexidade e a dificuldade da tarefa transparente ao usuário; executam tarefas no lugar do usuário; podem treinar ou ensinar o usuário, assim como monitorar procedimentos e eventos [MAE 97]. Outra característica importante nos agentes, citada por Thomas Malone é de serem projetados de maneira que usuários não especialistas possam moldá-los conforme suas necessidades [MAL 97].

Tecnologias como lógica *Fuzzy*, Computação Evolutiva e Redes Neurais são algumas das tecnologias de IA utilizadas para o desenvolvimento e elaboração de agentes. [KNA 98].

2.2 Projeto de Agentes

Para James White, algumas considerações devem ser lembradas quando do projeto de um agente: o agente deve possuir uma interface amigável, deve manipular erros, utilizar recursos simples e existentes, sendo que problemas complexos devem ser reduzidos a várias interações entre agentes [WHI 97].

Essas considerações são referentes ao projeto genérico de agente. Em um ambiente onde a computação ocorre de forma distribuída, ou seja, o processamento é executado por máquinas que estão interligadas em rede, as características do agente devem levar em contas outros fatores importantes, principalmente os que dizem respeito à segurança. Ainda para James White, em computação distribuída o ambiente deve prover segurança tal que os agentes respeitem as seguintes restrições funcionais:

a) não mudem componentes do sistema,

b) não apaguem componentes do sistema,

c) não monopolizem a CPU,

d) não utilizem grande largura de banda ou memória,

e) não entrem em conflito com a memória necessária para os aplicativos do usuário [WHI 97].

O projeto do agente deve considerar também as características que dizem respeito ao ambiente de interação do agente. No ambiente de computação distribuída, em uma plataforma Cliente/Servidor, um agente pode ser classificado segundo a sua política de interação com o ambiente de rede, que pode ser uma das seguintes situações:

a) ele roda em um servidor,

b) ele roda em um cliente usando recursos específicos do servidor,

c) ele roda em um servidor específico com referências a recursos de outros servidores

d) ele roda em uma arquitetura cliente-servidor baseada em browser.

Além disso, os agentes ainda podem ser classificados pelo seu grau de mobilidade como agentes fixos ou estacionários, ou agentes móveis ou itinerantes. Essa é outra característica importante a ser observada quando do levantamento de requisitos e na fase de projeto de um agente. [WHI 97]

Por fim, pode ser necessário que os agentes comuniquem-se. Para que agentes possam interagir em um determinado *Framework*, segundo Tim Finin e Philip Cohen, torna-se necessária uma linguagem de comunicação entre os agentes. A *Knowledge Query and Manipulation Language,* conhecida como KQML, que é utilizada largamente para este fim, provê um *Framework* para programas e agentes trocarem informações e conhecimento. A KQML é concebida como um formato para mensagens e como um protocolo de manipulação de mensagens para suportar a troca de conhecimento entre agentes em tempo de execução. [FIN 97] [COH 97]

2.3 Arquitetura de Agentes

Para Michael Knapik, a arquitetura de um agente é: "a forma como os agentes são colocados juntos, a forma ou estrutura de seus relacionamentos e interações.". Já o conceito de Arquitetura de Software, descreve em alto nível a configuração dos componentes que constituem um sistema e as conexões que coordenam as atividades entre esses componentes. O conceito adotado por Michael denota um contraste com o conceito da arquitetura de um sistema completo, a qual usualmente inclui partes relevantes de um sistema de operações, alguma infraestrutura de componentes, a interface do usuário, banco de dados e outros. [KNA 98]

Michael Knapik afirma que uma arquitetura de agentes é considerada adequada se ela possuir as seguintes características: simplicidade, funcionalidade, extensibilidade e portabilidade. [KNA 98]

São alguns exemplos de arquitetura:

a) Arquitetura M de Rieken – uma arquitetura complexa de agentes diversificados integrados. Essa abordagem resultou no realização do Sistema M, um software assistente que tenta reconhecer, classificar, indexar, armazenar, recuperar, explanar e apresentar informações relacionando-a à interface homem-computador em um ambiente de trabalho de conferência multimídia. A Figura 2.1 mostra a arquitetura do software M, na qual diferentes tipos de tarefas são acessadas por diferentes tipos ou classes de agentes e outros componentes funcionais. O sistema M também é um bom exemplo de uma aplicação baseada em agentes que usa várias técnicas de IA. [RIE 97]

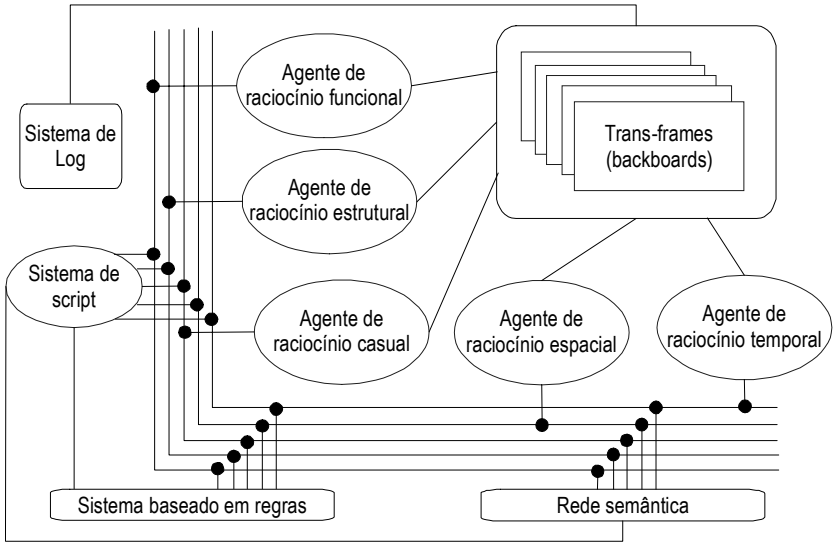

Figura 2.1 - Arquitetura do software M [RIE97]

b) Arquitetura de Genesereth – conceitos arquiteturais que enfatizam a interoperabilidade. Nessa arquitetura de engenharia de software baseada em agentes, o programador escreve seus programas como agentes de software independentes. A comunicação é assegurada através do uso de uma linguagem padrão de comunicação entre agentes. A interação entre agentes é assistida através do serviço de um conjunto coordenado de programas operacionais do sistema, chamado facilitador. Sob esse enfoque agentes e facilitadores trabalham juntos em prol de uma meta a qual é referida como uma federação. A Figura 2.2 mostra essa arquitetura. O facilitador recebe uma mensagem do agente ou facilitador e a direciona para outro facilitador ou a um agente adequado.[GEN 97]

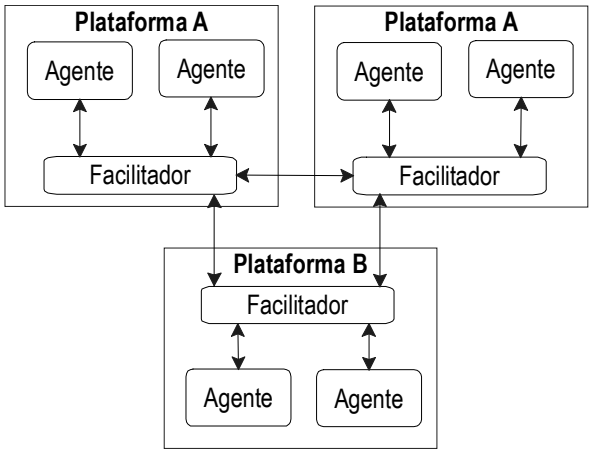

Figura 2.2 - Arquitetura de interoperabilidade de agente

c) Rosenschein – uma arquitetura para negociação de agentes. Essa arquitetura está fundamentada na necessidade de existir um protocolo de comunicação eficiente entre dispositivos de vários níveis de hardware e software que precisam interagir. Esse protocolo permitiria que agentes trabalhassem como sistemas heterogêneos podendo navegar entre diversas Intranet's. O principal objetivo de um protocolo de negociação é permitir que agentes com diferentes motivações colaborem na solução de problemas e tarefas. Um protocolo de saudação de agentes é necessário para uma arquitetura que permite à máquinas inteligentes e seus agentes descobrir que outros

dispositivos, aplicações e serviços estão disponíveis a ele, e como tomar vantagem daquilo que é necessário. Isso é conhecido como problema da descoberta e utilização. A Arquitetura de Saudação (AS) do Consórcio de Saudação prove a base para a negociação da forma e formato da interoperação entre dispositivos, aplicação e serviços, para a descrição de capacidades e disponibilidades, para a descoberta de capacidades de outros, para a procura por capacidades, e para a requisição e estabelecimento de sessões de interoperabilidade com outros para uso de suas capacidades. A Arquitetura de Saudação é formada por: clientes, servidores, um gerenciador de saudações, gerenciadores de transporte e unidades funcionais. A Figura 2.3 mostra um cliente interrogando um servidor de unidades funcionais. A informação trocada é chamada de personalidade. [KNA 98]

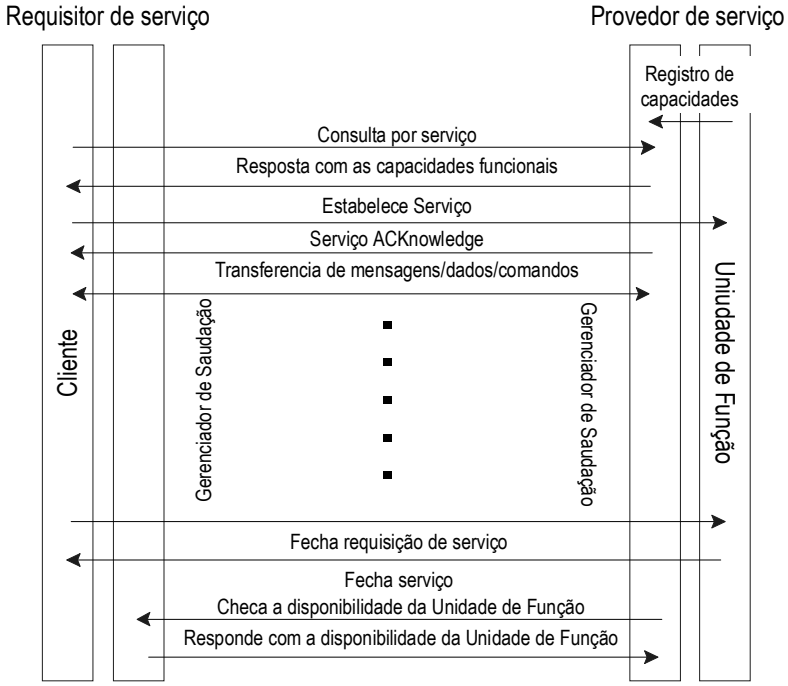

Figura 2.3 - Protocolo de Saudação baseado no Consórcio de Saudação

d) Kautz – Prototipação *Bottom-Up* e interação. Para Kautz o importante é começar a construir seu agente de baixo para cima, sem primeiro comprometer-se com uma arquitetura particular. O projetista mantém um projeto interativo, desenvolvendo, testando junto ao usuário, atualizando em um esforço para aperfeiçoar o sistema continuamente. Dessa forma as interações do agente e as reações do usuário com ele são vistas no mundo real gerando uma idéia melhor de qual a arquitetura mais adequada para produzir um agente mais robusto. Kautz acredita que essa abordagem empírica irá guiar o projetista do agente mais precisamente, definindo os requisitos iniciais para os fins de desenvolvimento dos agentes que são realmente úteis e oportunamente adequados ao ambiente do usuário [KNA 98]

e) Arquitetura ITX para agentes de controle de Kuo-Cho Lee. Essa arquitetura refere-se as Transações Interativas (*Interactive Transactions - ITX*), que é focada na modelagem e controle das interações entre agentes pela integração de dois importantes conceitos do mundo em tempo-real: adaptação via controle baseado em feedback e processamento de transações. Aplicações de software que manipulam computação distribuída e heterogênea e plataformas em rede, requerem um novo paradigma, no qual, um agente dá suporte a um usuário, representando o usuário no

sistema e manipulando complexas interações com outros agentes cooperativos e recursos do sistema. Nessa situação o fator crítico é como juntar objetivos de aplicações quando depara-se com entradas imprevisíveis do usuário e vários tipos de falhas. É necessário que o agente trate questões como: modos de falha; políticas que dizem respeito a propagação de falhas, mensagens de erros e alarme; ter certeza de não existe um único ponto de falha; metas priorizadas; manipulação consistente de estados e superestados, dependências e propagação de mudanças. A Figura 2.4 mostra a ITX que utiliza objetos compartilhados armazenados em banco de dados heterogêneos para suportar agentes através do ambiente. Quando dois agentes precisam cooperar em uma tarefa, modelos ITX coordenam seus esforços como um uma transação interativa em um ou mais objetos compartilhados, que representam fontes e controle de informação. Nesse cenário, agentes trocam conhecimento e colaboram em tarefas sem estar em comunicação direta realmente. [KNA 98]

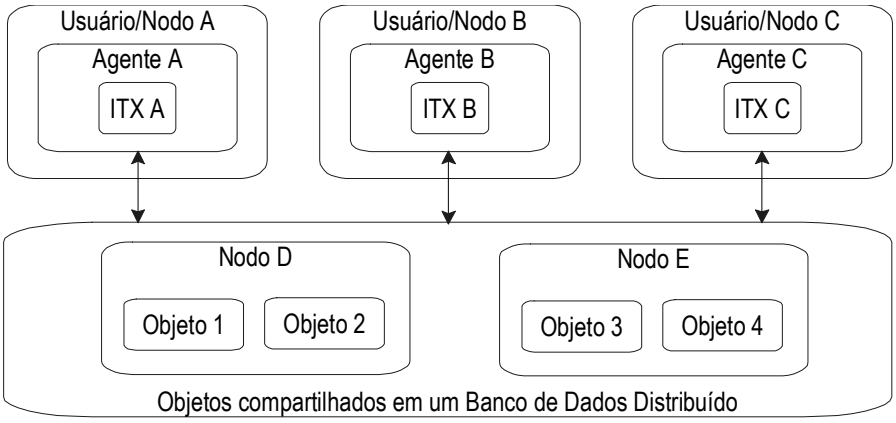

Figura 2.4 - Arquitetura de transações interativas

f) Agentes Colaborativos de Edmonds – uma Arquitetura Federativa. Em sistemas colaborativos são ocupados por agentes desenvolvidos por vários projetistas. O aspecto importante de um mundo heterogêneo é que coisas esperadas e inesperadas podem surgir. Nesse cenário os agentes são capazes de construir e destruir novos objetos que surgem em um processo colaborativo de projeto. A abordagem apresentada por Edmonds incute os agentes com a capacidade de reconhecimento de padrões, baseada em redes neurais, uma importante tecnologia capaz de habitar agentes inteligentes. Alguns dos meios nos quais os agentes podem ser diferentes são os seguintes: técnicas de representação de conhecimento; técnicas para processamento de conhecimento; significação de uma parte do conhecimento. De acordo com Edmonds, uma "sociedade de agentes" e a infraestrutura para suportar grupos de pessoas interagindo deveria: prover um diálogo sensível ao contexto; prover funções de gerenciamento de conferência; ser capacitado para interação de grupo com um sistema de informação geográfico remoto; prover consciência de outros membros do grupo; habilitar preferências individuais; prover conselho na administração do grupo. A "federação" de agentes que Edmonds coloca junto é ilustrada na Figura 2.5. A arquitetura de sistema de suporte de conferência colaborativa consiste em seis classes de agentes: um agente de conferência; um agente de chão; agentes de aplicação; um agente de grupo; um agente usuário e uma extensão desse agente chamada de agente de emergência. [KNA 98]

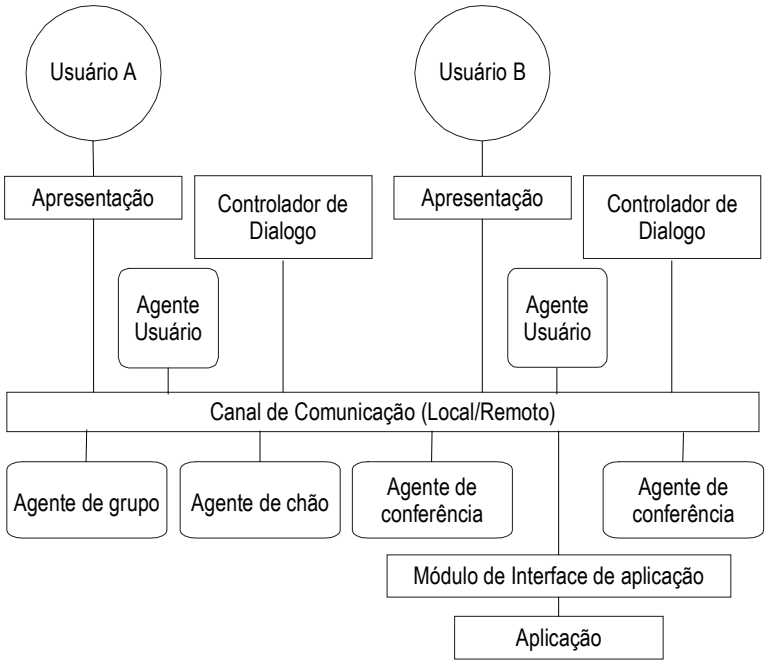

Figura 2.5 - Sistema de Suporte a conferência

g) Conceitos Arquiteturais de Sugawara – Lidando com mudanças. Uma arquitetura robusta precisa dar conta de acontecimentos inesperados, se não houver condição de mantê-lo intacto, pelo menos deveria produzir uma degradação gradativa e ser capaz de recuperação uma vez que o sistema estivesse funcionando novamente. Uma resposta parcial para algumas ocorrências inesperadas, apresentada por Sugawara, chamada de Rede de Informação Flexível orientada a Agentes (RIFA) é baseada no paradigma de Sistemas Multiagentes. Essa proposta arquitetural possui três propriedades essenciais para a realização de uma RIFA: inteligência, homeostase e evolução. Uma RIFA com essas características pode ser modelada considerando-se a rede e seu ambiente como um conjunto de agentes autônomos que se intercomunicam dentro e através de camadas de agentes. Todos os agentes compreendem a coordenação, negociação e os protocolos de comunicação dentro do sistema. Um agente primitivo pode ser especificado com os seguintes módulos ou protocolo: comunicador, especificador de domínio, cooperador e processador de tarefas, como mostrado na Figura 2.6 . Outro conceito nessa arquitetura é a noção de hierarquia de agentes implementada como grupos de agentes. [KNA 98].

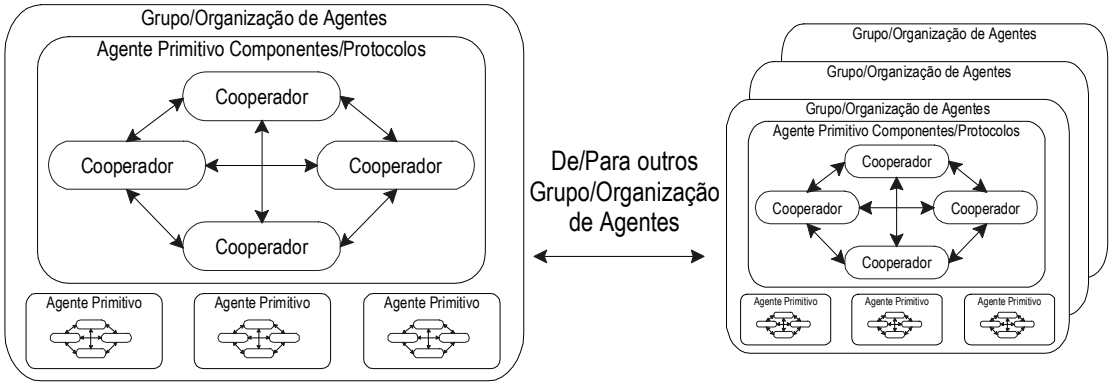

Figura 2.6 - Organização de Agentes

2.4 Agentes, criatividade e emoção

Cheong explica que criatividade envolve um ganho com alguma novidade, algo novo e diferente. Essa nova idéia, para ser interessante, precisa ser inteligível. Não importa quão diferente é a nova idéia, ela precisa ser compreendida em termos do que já era conhecido anteriormente. As regras potenciais do agente relacionadas a criatividade incluem a sugestão, identificação, ou, as vezes, a avaliação de diferenças entre idéias familiares e novidades. [CHE 96]

Segundo Margaret Boden, professora da *University of Sussex's School of Cognitive and Computing Sciences*, nem toda criatividade pode ser entendida como uma nova combinação de idéias familiares. Idéias criativas estão presentes em arquitetura, composição musical, teoremas matemáticos e invenções. Uma maneira de iniciar o pensamento sobre criatividade é considerar a noção de espaços conceituais, que é um plano mental, um estilo de pensamento, que é definido pelo conjunto de restrições que demarcam as margens e dimensões do domínio relevante. Muitas ações criativas resultam da exploração de espaços conceituais de forma sistemática e imaginativa. Agentes podem ajudar a mapear, explorar, talvez até mesmo serem guias na transformação dos espaços conceituais. [CHE 96]

No domínio da arquitetura, por exemplo, o trabalho computacional em estilos arquiteturais sugerem algumas formas que os agentes poderiam utilizar para ajudar um arquiteto. No domínio do Jazz existem programas de computador que ajudam pessoas a melhorar o Jazz. Esses programas entendem as várias dimensões do espaço musical do Jazz e podem viajar entre eles de várias maneiras. Navegando pelo espaço esses programas executam melhorias, sugerindo melodia, harmonia e ritmo, fazendo escolhas aleatórias ao longo de muitas dimensões simultaneamente. No espaço matemático o exemplo mais conhecido é encontrado no programa de Douglas Lenat's, *Automatic Mathematician*, cujas transformações do espaço de resultados heurísticos resultaram na descoberta de dois teoremas previamente desconhecidos sobre números primos. Em projetos de formas, tem sido investigado o suporte de formas emergentes em projetos de comunicação, como por exemplo, como elas poderiam ser manipulada por agentes usando métodos de reconhecimento de padrões. Uma forma emergente apresenta características não presentes na sua fonte. O exemplo preferido dos pesquisadores é a transformação radical do conceito quadro de uma bicicleta na bicicleta *LotusSport*, que usa uma única peça construída em fibra de carbono em substituição do convencional quadro diamante em tubo de aço. [CHE 96]

Para Gheong, "muitos processos psicológicos estão envolvidos no pensamento criativo, desde justaposição combinatória até o mais complexo raciocínio exploratório transformacional aos mais altos pensamentos emergentes não estruturados." [CHE 96]

Além da criatividade, Cheong fala sobre a possibilidade de agentes poderem possuir "emoções". Sua discussão aborda as regras de emoção em agentes e discute como a emoção pode ajudar agentes de software com expressão animada em personagens de desenho animado, fazendo com que eles pareçam mais reais, sem estender sua antropomorfia aos níveis humanos de inteligência e capacidade. [CHE 96]

Cheong coloca que o professor Joseph Bates da *Carnegie Mellon University* acredita que emoções jogam uma importante regra na construção da

credibilidade de agentes. Ele descreve que a credibilidade de um agente não é como aquela que tem um personagem honesto e confiável, mas como aquela que provê uma ilusão de vida convincente ao espectador que quer acreditar que o agente é real. Segundo Bates, a credibilidade de agentes é análoga a credibilidade de personagens discutida em artes de ficção escrita e filmes. Emoção é o primeiro meio de disparar a credibilidade do agente. Um agente que demonstra emoção ajuda as pessoas a entender o que realmente é importante ao seu redor e o que verdadeiramente tinha sido desejado. [CHE 96]

Muitos pesquisadores em Inteligência Artificial (IA) tem procurado construir robôs ou agentes que pareçam pensar, sentir e viver. Nas suas procuras pelas qualidades da humanidade, os pesquisadores de IA, enfatizam os aspectos computacionais das capacidades de recriação tais como raciocínio, aprendizagem e solução de problemas pelo computador. Por outro caminho, artistas de animação procuram reproduzir formas de vida a partir de nada mais do que simples linhas desenhadas, tintas e lâminas de celulose que movem-se quadro a quadro. Bates argumenta que a compreensão dos animadores de personagens nas suas indagações artísticas poderiam ser a chave para construir modelos computacionais de agentes interativos que são críveis. [CHE 96]

O Projeto OZ da *Carnegie Mellon University*, é um trabalho, no qual o trabalho de programadores e animadores poderão ser combinados para criar entidades semelhantes a humanos com as quais humanos poderão eventualmente trabalhar ou jogar. [CHE 96]

2.5 Agentes e a Web

As características e propriedades da Web, junto com suas novas utilizações, acabaram gerando outros problemas. O mais conhecido deles é a segurança, necessária quando se pretende efetuar transações na rede [ZAP 98]. Além disso, em documentos HTML, os dados estão dispostos de forma desestruturada. Os Browsers permitem aos usuários pesquisar através de um grande número de informações mas provêem capacidades muito restritas para localização, combinação, processamento e organização destas informações; nesse caso a utilização de agentes pode sanar estas limitações [KNO 97]. A estrutura básica destes agentes consiste no rastreamento exaustivo e na indexação do conteúdo das páginas.

Segundo Cheong, agentes utilizados na Web, Web *Robots*, como os *Spiders* ou *Wanderers*, são programas que cruzam a Web por recursividade retornando páginas linkadas pelos seus URL's. O *WebCrawler* é um exemplo de *Spider* que procura o que o usuário quer na Web. Já o *Lycos* é um *Spider* que executa a procura baseado em palavras chaves. O sistema conhecido por *Harvest* é um conjunto de ferramentas escalonável, com arquitetura configurável para recuperação, indexação, armazenamento temporário, gravação e acesso a informações na Internet. Os *WebAnts* são exploradores cooperativos que compartilham o trabalho de procurar informações na Internet. Os MUD's (*Multi-User Dungeons*) são mundos virtuais na Internet onde os *Charterbots*, agentes de conversação, trocam informações.[CHE96]

Existem ainda os agentes que ao conectarem-se à rede, constroem um *Framework* de atividade específica. Ao conectarem-se a Internet, esses agentes,

procuram um Servidor específico, onde são registrados a cada vez que entram na rede. A partir do momento que estão conectados à rede eles passam a interagir entre si , disponibilizando ou recuperando informações. Por vezes, esses agentes executam o papel de servidor e de cliente simultaneamente. Uma atividade específica que pode ser executada por esse tipo de agentes, por exemplo, é a disponibilização de arquivos de áudio. Nesse caso o agente que está instalado em uma máquina, ao ser conectado a Internet, passa a disponibilizar os arquivos que estão em seu disco, ao mesmo tempo em que procura por novos arquivos. O servidor que é acessado pelo agente serve apenas para criar uma espécie de comunidade virtual, ou seja, um espaço que é acessado por pessoas com um interesse em comum, através de uma lista dos agentes que estão conectado ao servidor naquele momento. Essa tecnologia de agentes também poder ser utilizada para a criação de conferências ou *chat's* onde a comunidade virtual encontra-se em uma "sala de reuniões" ou em uma *chat's room*.

Um Agente Web possui algumas características peculiares. Estas características são basicamente regidas pelas restrições de ações ditadas pelo protocolo HTTP e pela arquitetura cliente-servidor da Web.

O conceito de ser uma entidade computacional capaz de efetuar tarefas de forma autônoma em nome de um usuário é comum a qualquer tipo de agente. Já a capacidade cognitiva poderia ser implementada em um Agente IAD que opera em uma rede de computadores onde não é utilizado o HTTP como protocolo de comunicação. Para um Agente Web que devesse implementar características de mobilidade, esse recurso poderia incrementar a capacidade de adaptação. Por ser a Internet um ambiente heterogêneo, a adaptabilidade seria uma característica positiva nesse agente. Porém, por utilizar o protocolo HTTP, isso poderia acabar interferindo em questões de desempenho.

Um dos fatores que influenciam no desempenho é a largura de banda. Um agente móvel precisa navegar pela Internet, isso significa que ele precisa se deslocar de um ponto a outro através da rede. O processo de deslocamento consiste em fazer uma cópia do agente para o servidor de destino. Quanto mais características e capacidades tiver um agente, maior será o seu tamanho, interferindo diretamente na quantidade de *bits* que trafegam na rede.

Outro fator é a arquitetura do protocolo HTTP. Segundo Tanenbaum em [TAN97], apesar de ter sido projetado para utilização na *Web*, O HTTP foi, intencionalmente, criado para ser mais genérico do que o necessário com vistas a futuras aplicações orientadas a objetos. Por esse motivo, a primeira palavra na linha de uma da solicitação HTTP completa é o nome do método (comando) a ser executado na página *Web* (ou objeto genérico). Os métodos genéricos podem basicamente:

a) Solicitar a leitura de uma página da *Web*, GET;
b) Solicitar a leitura de um cabeçalho de páginas *Web*, HEAD;
c) Solicitar o armazenamento de uma página da *Web*, PUT;
d) Acrescentar um recurso (por exemplo uma página da *Web*), POST;
e) Remover a página da *Web*, DELETE;
f) Conectar dois recursos existentes, LINK;
g) Desfazer uma conexão entre dois recursos, UNLINK.

Esses métodos denotam que o agente é considerado como um objeto que será manipulado através desses métodos, ou seja, no processo de transferir um agente de um local para outro na rede, ele é considerado como um documento que possui um conteúdo proprietário que deverá ser definido no cabeçalho MIME (*Multipurpose Internet Mail Extensions*) que é uma especificação técnica que descreve a transferência de dados multimídia usando padrões de correspondência (*mail*) da *Internet.*

A MIME define formatos para arquivos e imagens, filmes, arquivos de áudio, arquivos binários, arquivos de aplicação e vários outros tipos de arquivos multimídia. Na verdade, com a MIME você pode definir seu próprio formato de arquivo e usá-lo para comunica-se com um servidor (desde que este reconheça a definição). [JAM99]

O servidor *Web* inclui um *tipo* ou *subtipo* da MIME dentro do cabeçalho de cada arquivo que ele envia para o cliente. O tipo descreve o tipo geral de agrupamentos do arquivo com o qual você está trabalhando. Igualmente, o subtipo informa ao cliente o tipo específico, dentro do agrupamento geral, do arquivo com o qual você está trabalhando. Os tipos e subtipos da especificação MIME estão sempre mudando, porque o MIME também evolui para suportar novos formatos de arquivo ou aplicações.

Quando da implementação de Agentes Móveis a arquitetura Cliente/Servidor da Web traz alguns contratempos, principalmente no que diz respeito a necessidade do sistema ser Tolerante a Falhas. Além disso, para tornar a implementação de Agentes Móveis na Web viável, seria necessário que os Servidores HTTP e os Clientes permitissem a "entrada" de agentes ao invés de simplesmente receber solicitações e de respondê-las transferindo os arquivos solicitados.

Mesmo considerando as restrições, pesquisadores e autores como Jamsa [JAM99], Pattie Maes [MAE 9?], D'Amico [DAM 95], Cerveira [CER 97] e Murch [MUR 98] constatam, ao longo de seus respectivos trabalhos, que atualmente é possível construir-se agentes, que atuam na Web, com características de Agentes IAD, como a cognição por exemplo, porém, a mobilidade dos agentes não poder ser implementada sobre o protocolo HTTP. A fundamentação dessa afirmação está na arquitetura do protocolo descrita anteriormente de forma sucinta.

3 Sistemas de Informação – Uma visão geral

3.1 Definição

Os Sistemas de Informações, SI, são utilizados por empresas para manipular informações, a partir de um BD, que as auxiliem na tomada de decisão e no gerenciamento do negócio. Os SI podem gerar informações de cunho operacional, tático ou estratégico, atendendo, respectivamente, aos níveis administrativos funcional, gerencial e executivo.

3.2 Dados: Relação entre Realidade e Abstrações

Dados são abstrações da realidade utilizados para representar uma determinada característica dessa realidade.

Uma realidade pode gerar *n* abstrações distintas, ou seja, pode gerar inúmeros dados distintos. Essa relação é do tipo *um-para-n*, 1:n, isso significa que uma realidade pode ser representada por *n* dados diferentes que representam várias características da mesma realidade. As características de uma pessoa são exemplos desse tipo de relação. Nesse caso a realidade a ser abstraída é a pessoa. Dados sobre seu nome, idade, sexo, altura, filiação, etc., são as diversas abstrações que podem ser utilizadas para representar características específicas da pessoa, como pode ser visto na Tabela 3.1.

Cada dado gerado precisa ser rotulado com um identificador. Esse identificador é na realidade o nome do campo atribuído a um determinado tipo de dado. Ele pode ser chamado de rótulo, campo, etiqueta ou *tag*.

Tabela 3.1 - Conjunto de abstrações utilizadas para identificar uma pessoa

Rótulo	Dados (Abstrações)
Nome	Maria
Idade	20 anos
Sexo	Feminino

Os rótulos são normalmente agrupados em classes segundo a natureza da característica abstraída, por exemplo, o rótulo “Nome” indica um conjunto de símbolos (letras) que é utilizado para identificar um objeto ou ser vivo. Essa classe pode ser especializada para indicar uma subclasse, por exemplo, a subclasse “Nome_Pessoa” que indica somente os nomes de pessoas, ou, uma subclasse “Nome_Animal” que indicaria todas as classes de animais da natureza. A Figura 3.1 mostra uma hierarquia de classes de tipos de dados. A profundidade da especificação depende somente da necessidade de especificação requerida pelo SI.

Isso demonstra que o mesmo rótulo é utilizado para identificar naturezas semelhantes de diferentes realidades. Esse mecanismo, o inverso da especialização, é chamado de generalização e permite que o mesmo tipo de dado, ou rótulo, seja utilizado

para identificar uma característica comum de diferentes realidades. Como já foi visto, o rótulo “Nome” é utilizado para identificar tanto uma pessoa, como, para identificar um produto.

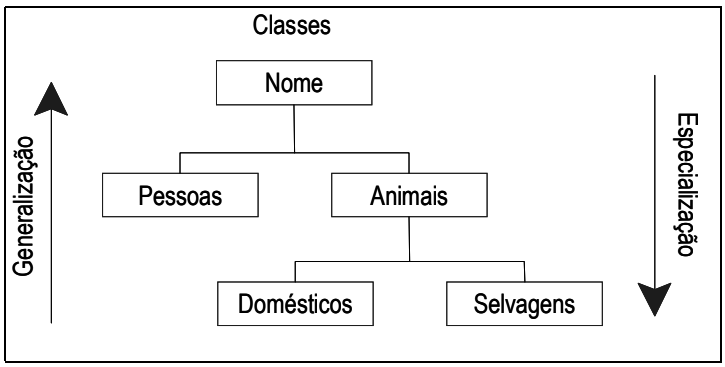

Figura 3.1 - Hierarquia de Classes dos tipos de dados

Esse mesmo princípio é utilizado em técnicas de Orientação a Objeto (OO). Em OO as abstrações da realidade são chamadas de objetos. As noções de classes, generalização e especificação são as mesma apresentadas aqui. Já os rótulos são chamados de propriedades, e, além das propriedades, os objetos possuem métodos, que são ações que podem ser realizadas sobre ou pelos objetos.

3.3 Informação

O conceito de informação está relacionado ao significado atribuído a determinado dado, isto quer dizer que, o dado em si não gera informação. Em um SI a informação existe a partir do momento que um conjunto de dados é processado e gera um novo conjunto de dados, ou um único dado, que possui um significado para a entidade que irá manipulá-lo. Essa definição traz implícita a noção de que a informação é relativa. Ao estar associado ao significado, um conjunto de dados pode representar uma informação para uma entidade, por possuir um significado, enquanto representa apenas um dado para outra entidade, por não possuir significado algum para essa.

3.4 Dados Estáticos x Dados Dinâmicos

As abstrações da realidade (dados) podem ser estáticas ou dinâmicas. As abstrações estáticas, ou dados estáticos, representam características da realidade que não são alteradas com o passar do tempo. Já as abstrações dinâmicas, ou dados dinâmicos, são utilizadas para representar características de uma realidade que sofre alteração com o passar do tempo. Dependendo do tipo de SI os dados dinâmicos podem passar a ser estáticos. Isso acontece quando o SI associa a linha de tempo a um dado dinâmico. Por exemplo, o estoque de um determinado produto é uma informação dinâmica, com o passar do tempo o seu valor pode ser alterado devido a operações de entrada e saída de estoque; quando o SI armazena o estado do estoque a cada período de tempo, essa informação, apesar de ainda representar o estoque, passa a ser estática. O processo de gerenciar versões é uma maneira de transformar dados dinâmicos em estáticos.

Os dados estáticos não podem ser alterados com o passar do tempo, principalmente quando representam um processo evolutivo, ou seja, um processo que

apresenta estados discretos em função do tempo, que necessita do registro de cada etapa para efeitos de controle. Um exemplo de dados estáticos gerados a partir de um processo evolutivo é o estoque de mercadorias de uma empresa. Ao final de cada mês a empresa pode executar uma rotina de balanço de estoque, que armazena o estado atual do estoque. No balanço anual, o estoque final pode ser comparado com o resultado da contabilização do estoque inicial, acrescido das entradas de estoques e debitadas as saídas de estoque dentro de cada período. O resultado final deve ser igual ao levantamento de estoque. Técnicas como essa são utilizadas para testar a consistência do SI ou para detectar desvios de mercadoria. Através desse exemplo é fácil enxergar que a alteração de um dado estático, nesse caso, o registro do estoque mês a mês, pode gerar um problema de inconsistência.

3.5 Taxonomia de Dados em um SI

Em um SI os dados estáticos são representados pelos Dados de Cadastro. A função desse tipo de dados em um SI é de suporte. São eles que identificam os elementos que interagem dentro de um processo.

Os dados dinâmicos são representados pelos Dados de Estado de Processo. Esses dados são gerados a partir dos processos internos da empresa que necessitam ser controlados e avaliados constantemente.

Além da sua classificação temporal, estáticos ou dinâmicos, os dados podem ser classificados quanto à visibilidade. A visibilidade de um dado indica qual o nível de acesso ao dado. Nesse aspecto, os dados podem ser: Públicos – Abertos, ou Privados – Fechados. O que vai definir se o dado é público ou privado é o escopo e a natureza das relações entre as entidades que utilizam os dados.

3.6 Responsabilidades sobre os dados

A responsabilidade de manutenção[2] do dado gerado, por determinada entidade, é de responsabilidade pura e exclusiva dessa entidade. Essa regra pode ser chamada de Princípio de Responsabilidade sobre o dado gerado.

Esse princípio está baseado no fato de que a entidade que gera o dado é quem dispõe de profissionais qualificados para garantir a consistência da informação abstraída. Por exemplo, uma indústria produz um determinado produto. Para comercializar o produto, a indústria, necessita criar um conjunto de dados que especificam e identificam o produto. Se esse produto fosse um medicamento, ele deveria conter informações sobre: o nome comercial, o princípio ativo, a função terapêutica, as interações medicamentosas, a indicação, a posologia, as contra-indicações, o número do registro no ministério da saúde, o fabricante, o farmacêutico responsável, etc. Todas essas informações são impressas na embalagem ou na bula do medicamento. Por ser um conjunto bastante amplo de dados, dos quais, muitos são dados não estruturados (textos extensos), é sensato afirmar que a indústria que produz

[2] Por manutenção considera-se a garantia de consistência e de disponibilidade do dado

esse medicamento é quem é capaz de garantir que as informações sobre o produto são verdadeiras.

O fato é que as informações desse produto serão utilizadas, parcial ou integralmente, por outras empresas, como por exemplo, por distribuidores de medicamentos e farmácias. Para poder comercializar esses produtos essas empresas irão alimentar seus SI com as informações encontradas na embalagem e com informações complementares relativas a transação comercial encontrada nos documentos fiscais (Nota Fiscal). Como esse processo é feito por um, ou vários funcionários, muitas vezes não qualificados para essa tarefa, é possível que as informações alimentadas não sejam consistentes com aquelas produzidas pela indústria, ou seja, não são iguais aquelas indicadas pelo proprietário das informações.

Apesar desse princípio estar relacionado ao conceito de propriedade, que garante ao proprietário, os direitos de propriedade, acarretando, no entanto, na responsabilidade de manutenção do dado. Atualmente, essa responsabilidade fica restrita ao SI do proprietário, isto é, a empresa que produz o produto garante a consistência apenas em seu próprio SI. As demais empresas que precisam utilizar as informações geradas por essa empresa precisam assumir a responsabilidade pela sua interpretação das informações fornecidas. Esse não é um problema de escolha, no qual, o usuário opta por livre e espontânea vontade em assumir a responsabilidade de manutenção das informações produzidas por uma entidade. O fato é que as entidades não oferecem nenhum tipo de infraestrutura que possa ser utilizadas pelos usuários para acessar essas informações.

A possibilidade de assumir a responsabilidade de manutenção efetiva sobre uma informação é algo que interessa tanto a quem produz a informação, no caso de uma indústria, pelo fato de saber que suas especificações do produto não estão sendo distorcidas o que poderia colocar em risco a imagem da empresa, no caso do usuário final, pelo fato de saber que aquelas informações são verdadeiras. Imagine, por exemplo, o caso do Instituto de Identificação de uma região que é uma empresa pública responsável pela identificação do cidadão. Obter as informações direto do Instituto de Identificação é mais confiável para um comerciante do que através de uma cédula de identidade que pode ser falsificada, ou seja, o comerciante se sentiria mais seguro ao cadastrar um cliente se ele soubesse que as informações de identificação daquele cliente estão sendo garantidas pelo próprio órgão que é responsável pela identificação do indivíduo. Essa segurança só é possível a partir do momento que a entidade que gera os dados for responsável pela sua manutenção e disponibilização.

3.7 Analisando um modelo de SI tradicional simplificado

A Figura 3.2 representa um modelo simplificado de SI. Nessa figura estão representados dois SI, o primeiro, pertencente a Empresa B e o segundo pertencente a Empresa A, por exemplo. Ambos os SI armazenam dados sobre a transação de compra de um cliente. A relação entre as compras e o contador da empresa, representa o controle de caixa da empresa. A partir desse modelo é possível extrair informações sobre quantidade de clientes atendidos, movimento de caixa e compras realizadas por clientes. Essas são informações de cunho gerencial.

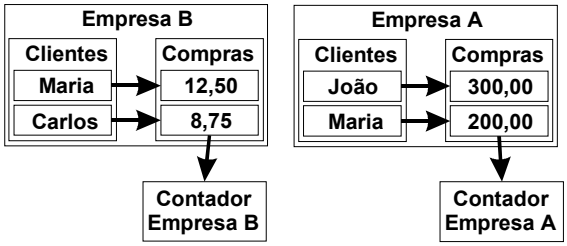

Figura 3.2 - Modelo simplificado de um SI

Nesse modelo os dados sobre o nome do cliente e a compra realizada por ele são estáticos. Enquanto não é feito o fechamento do caixa, o dado *movimento total* é dinâmico. A operação de fechamento de caixa cria uma cópia estática do dado dinâmico, que é passada ao contador da empresa. Para o contador da empresa, o *movimento diário*, que é um dado estático para a empresa, é um dado dinâmico. O contador irá armazenar dinamicamente os dados dos *movimentos diários* para gerar um dado estático do *movimento mensal* e *movimento anual* do *movimento de caixa* da empresa. Quanto à visibilidade, todas essas informações são privadas, porém com diferentes níveis de visão. A gerencia da empresa tem acesso total aos dados, já os funcionários da empresa apenas lançam os dados mas não tem acesso às informações geradas por eles. O contador tem acesso a informação *movimento total* mas não tem acesso aos detalhes de cada venda.

Nesse exemplo a responsabilidade dos dados gerados é de cada uma das empresas. Porém, é possível perceber-se que ambas realizam a abstração do *nome do cliente*. Nesse caso, a mesma característica da mesma realidade está sendo abstraída duas vezes por entidades diferentes, ou ela está redundante, no caso de ter sido feita corretamente pelas duas empresas, ou ela está inconsistente em uma ou em ambas as empresas. O problema real é a inconsistência, pois a redundância apenas acarreta problema no sentido de despender custos de manutenção sobre uma determinada informação que é necessária ao SI da empresa, mas que não é de sua responsabilidade pois não foi gerada por ela.

Isso acontece por que os dados utilizados para identificar o cliente, são abstraídos por entidades, empresas A e B, que não possuem a responsabilidade de identificação de uma pessoa, nesse caso, seu cliente. Essa identificação deveria ser realizada pelo Instituto de Identificação, como foi explicado na seção 3.6.

A identificação do cliente é um dado de suporte. Esse dado, junto com o dado do *valor da compra*, gera a informação da *compra do cliente*. Essa informação, que foi processada a partir de outros dois dados, e que representa uma relação entre a empresa, o cliente e a compra realizada, é uma informação de responsabilidade da empresa. A necessidade de criar uma abstração que não está sob seu escopo de responsabilidade está no fato de não ser possível recuperar essa informação da entidade responsável por determinado dado, ou seja, nesse exemplo não existe uma entidade que seja responsável por manter e distribuir os dados do cliente necessários para a realização de uma transação comercial.

3.7.1 Problemas desse modelo

O principal problema desse modelo é a redundância de dados. Para que a redundância não seja um problema é necessário que haja um processo de sincronização entre as bases repetidas. A redundância é necessária em projetos de Sistemas Tolerantes a Falhas ou em Sistemas Distribuídos que necessitem contornar problemas de infraestrutura tecnológica, como por exemplo a impossibilidade de manter o sistema em atividade o tempo todo.

No caso do exemplo da Figura 3.2, não existe motivo algum que leve as duas empresas a manter o BD de Clientes sincronizada. Esse fato gera inconsistência, pois as representações contidas em cada empresa, apesar de referir a mesma pessoa e de possuírem a mesma finalidade, a identificação do cliente, podem ser diferentes. O principal motivo que pode contribuir para aumentar o risco de inconsistência é o problema da mão de obra desqualificada para proceder a entrada de dados, ou a abstração da realidade.

Além desses dois problemas, ambas as empresas terão o seu TCO aumentado, pois ambas estão mantendo as mesmas informações. Para manter essas informações tornam-se necessários: a manutenção de equipamentos, a realização de cópias de segurança, a contratação ou treinamento do funcionário para realizar a tarefa, além do tempo necessário para realização das tarefas.

3.7.2 Separando as responsabilidades

A Figura 3.3 mostra a separação dos dados utilizados no exemplo anterior. Nessa separação os dados dos clientes são responsabilidade do SI da Entidade de Identificação, enquanto que os dados referentes às compras dos clientes são responsabilidade dos SI das empresas.

Entidade de Identificação

Pessoa	
ID	Nome
01	Carlos
02	João
03	Maria

Empresa B

Compras	
ID	Valor
01	8,75
03	12,50

Empresa A

Compras	
ID	Valor
02	300,00
03	200,00

Figura 3.3 - Dados separados segundo escopo de resposabilidade

No modelo da Figura 3.2, em cada empresa, são criadas duas tabelas: a tabela *Clientes* e a tabela *Compras*. E um relacionamento E-R (Entidade-Relacionamento) 1:n entre as duas tabelas, que pode ser chamado de *Compras_Clientes,* que faz o mapeamento entre os clientes e suas respectivas compras. Com a separação das responsabilidades as empresas A e B manteriam em seu SI apenas as tabelas de *Compras* e o relacionamento E-R *Compras_Clientes*, como pode ser visto na Figura 3.4

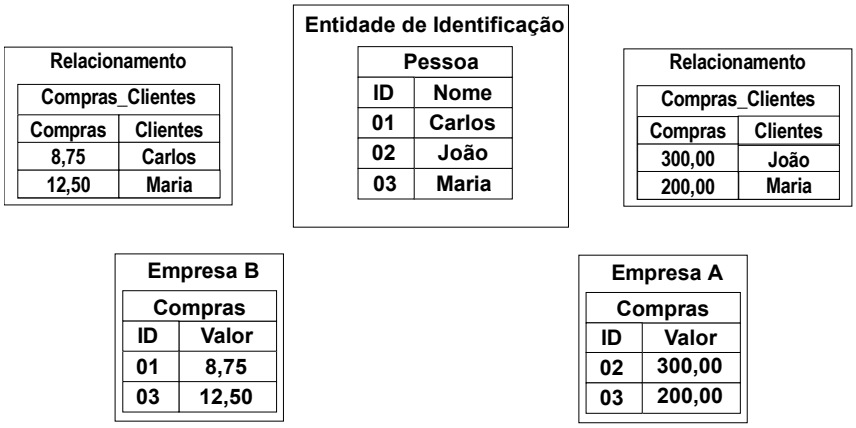

Figura 3.4 - Relacionamento entre entidades com a separação das responsabilidades

Com essa separação, as empresas tem uma considerável economia e melhora na qualidade, pois, a quantidade de dados a ser mantida é menor. A tarefa de entrada de dados é realizada por uma entidade especializada que é aquela responsável pelos dados gerados, segundo o princípio apresentado na seção 3.6. Os funcionários da empresa não precisam ser desviados da atividade-fim da empresa para realizar a manutenção de dados auxiliares.

A diferença essencial é que SI de entidades distintas irão compartilhar dados comuns ao invés de construir e manter uma base privada com dados públicos.

Apesar das vantagens da separação de responsabilidades, poucas, ou nenhuma iniciativa tem sido tomada pelas empresas, ou seja, as empresas continuam desenvolvendo seus SI utilizando dados gerados por outras entidades através da inserção manual ou automática (EDI convencional assíncrono que utiliza troca de arquivos com estrutura padronizada). Dessa forma elas acabam assumindo a responsabilidade de manutenção e, conseqüentemente, os custos envolvidos na manutenção desses dados, processo que deveria ser de responsabilidade de quem gerou os dados. Esse fato deve-se a uma visão arraigada no paradigma de SI centralizado, e na falta de tecnologia desenvolvida especificamente para essa finalidade.

Nesse sentido, a proposta desse trabalho é apresentar um modelo de SI baseado no Princípio de Responsabilidade do Dado Gerado. O modelo a ser proposto visa contornar os problemas citados ao longo dessa seção ou, pelo menos, de otimizar a elaboração de SI diminuindo os custos envolvidos na manutenção de um SI através da divisão da responsabilidade de manutenção de dados do SI. A viabilização do SI a ser proposto será feita através da utilização de tecnologia de agentes discutida na seção 2 que será utilizada para fundamentar a proposta de um Agente EDI específico para essa tarefa, que será apresentado na seção 4 .

3.8 Modelo de Sistema Informações baseado em Web proposto

A Internet permite uma nova perspectiva de acesso a dados distribuídos. Para aproveitar todo o potencial dessa realidade é necessário uma mudança de paradigma. O princípio que rege esta mudança é o Princípio da Responsabilidade sobre o dado gerado, descrito na seção 3.6.

O modelo atual prevê, que SI, mesmo distribuídos, manipulem dados de propriedade da entidade. Mesmo que esses dados sejam uma cópia redundante de dados que são de responsabilidade de outra entidade, mas que são necessários a esta entidade por serem dados de suporte. A cópia, se autorizada, dá o direito ao usuário de manipulá-la da maneira que melhor lhe convier. Por trás dessa liberdade está o risco de inconsistência. Por exemplo, um cliente quando faz um cadastro em uma empresa fornece seus dados pessoais, entre eles, a identificação pessoal e o endereço; como os dados ficam na empresa que o cadastrou, esses dados podem ser manipulados segundo os critérios da empresa. O risco, natural e imediato, de gerar uma inconsistência está no processo de obtenção de dados por parte de funcionários da empresa. O risco não imediato está na manipulação involuntária do cadastro, que também pode gerar inconsistência. Mas além desses, ainda existe o risco indireto, quando, por exemplo, o cliente mudou de endereço não informando a empresa, e isso, ainda descartando-se as atitudes de má fé. A Figura 3.5 apresenta um exemplo, no qual, os cadastros de ambas empresas então inconsistentes. A empresa A está com o endereço errado, o que pode ter acontecido por não ter sido avisada pelo cliente que mudou de endereço, ou porque este forneceu um falso endereço. A empresa B está utilizando o tratamento *Sra.* para referir-se a *um* cliente a não a *uma* cliente. Nesse caso o erro pode ter sido cometido, ou na hora do cadastro, ou por um funcionário desavisado, que ao tentar corrigir o cadastro, acabou criando uma inconsistência pelo fato do nome desse cliente poder ser utilizado tanto para homens quanto para mulheres.

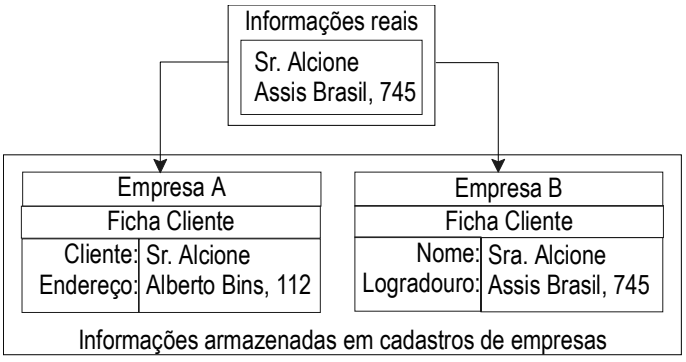

Figura 3.5 - Exemplo de abstração de dados inconsistente

Esse é apenas um exemplo, mas é possível descrever os mesmos problemas nos processos de interação entre empresas, como por exemplo, nas transações comerciais entre varejo, atacado e indústria.

Esses problemas podem afetar transações comerciais do tipo e-Business, isto é, transações entre empresas que vão além de simples transações comerciais, envolvendo trocas de informações estratégicas, táticas e operacionais, sobre processos e produtos, tanto privados quanto governamentais (também chamado de e-Gov). O maior risco nesse caso é a inconsistência. Por exemplo, uma distribuidora que representa e revende os produtos de uma indústria pode conter em seu SI informações erradas sobre os produtos que estão sendo vendidos.

O e-Commerce, conceito que está inserido dentro do conceito de e-Business, é uma transações comercial entre empresa e cliente via Internet. Nesse tipo de transação o cliente seleciona os produtos que deseja comprar e informa seus dados à empresa por intermédio de seu site na Web. Do lado do cliente existe a insegura de

fornecer informações que possam ser utilizadas de forma inadequada. Do lado da empresa existe o problema da consistência das informações fornecidas, por exemplo, se o endereço de entrega e os dados do comprador foram preenchidos corretamente.

Para contornar esses problemas, principalmente em transações que envolvem e-Business e e-Commerce, algumas tecnologias tem sido desenvolvidas, como por exemplo o Web EDI da IBM, uma maneira de trabalhar com dados na Internet, utilizando-se de vários produtos e ferramentas Web da própria IBM, no lugar de EDI convencional e BBS. Porém, tanto essas soluções, quanto soluções alternativas, como páginas pessoais e EDI convencional, são soluções de alto custo. As páginas pessoais, *sites* particulares onde uma pessoa pode manter seus dados pessoais, não são padronizadas e não provêem segurança. Já o EDI necessita de pessoas qualificadas para desenvolver o projeto e prestar manutenção, sendo que é necessário um projeto para cada cliente.

A proposta do modelo de SI baseado em Web, a exemplo de outras tecnologias, é utilizar a Internet, através da Web, como infraestrutura disponível e de fácil acesso, para que se possa fazer a separação dos dados segundo o princípio de responsabilidade.

A diferença está na utilização de um modelo de Agente EDI para recuperação e distribuição de dados estruturados na Web. O objetivo é criar uma rede de Agentes, que se comuniquem entre si, e, que possuam a capacidade de trocar informações, para servirem de interface, de comunicação e de troca de dados estruturados, entre os SI de entidades distintas que partilham dados comuns. Dessa forma é possível disponibilizar os dados de suporte entre SI que executam algum tipo de interação.

A proposta desses Agentes é permitir a qualquer usuário distribuir seus dados de forma privativa ou pública, criando um *Framework* de troca de dados independente, flexível, extensível e descentralizado, permitindo a troca de qualquer tipo de dados estruturados, inclusive os dados de controle da rede.

Esses Agentes tornarão viável às empresas implementarem processos de EDI, B2B (*Business to Business*) e B2C (*Business to Consumer*) que necessitem trocar dados estruturados, de forma menos onerosa.

3.9 Comparando alguns modelos de recuperação de informação na Web

Como a Web é um grande repositório de documentos HTML, a maioria das soluções de recuperação de informações trabalha com dados não estruturados. Para extrair informações desses documentos são utilizados processos manuais ou automáticos. O processo manual consiste na classificação do conteúdo a partir da avaliação de uma pessoa especializada. Os processos automáticos consistem na utilização de um robô de software para executar a leitura automática das páginas. Nessa leitura o robô pode procurar por um identificador (*Tag*) HTML específica, como por exemplo campos de Metainformações. Outra opção é a criação de um índice a partir do documento lido. A Figura 3.6 mostra esse cenário. Nessa situação o usuário, através de um Browser, acessa um s*ite* de busca que possui um índice de páginas que foram previamente indexadas, para fazer uma pesquisa.

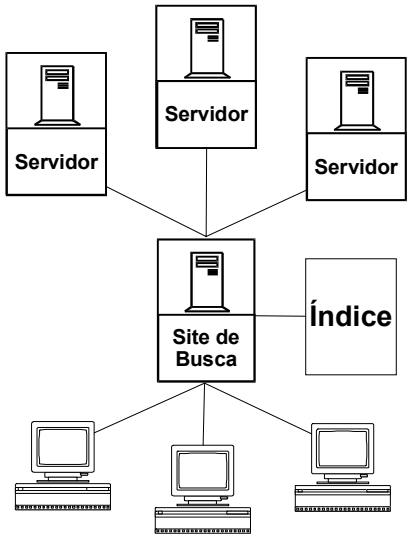

Figura 3.6 - Recuperação de informações na Web através de páginas indexadas

Como existem inúmeros *Sites* de busca através de índices, sendo que cada um pode utilizar um critério diferente para criar o seu índice, uma tecnologia acessória é utilizada para facilitar o trabalho. Trata-se dos robôs de mineração de sites de busca ou Metabusca, Figura 3.7. Como na situação anterior, o usuário acessa o *site* do robô via Browser, quando o usuário solicita uma pesquisa o robô ou Agente acessa os *sites* de busca por indexação distribuindo a requisição do usuário. Quando as consultas são retornadas elas são organizadas e apresentadas ao usuário.

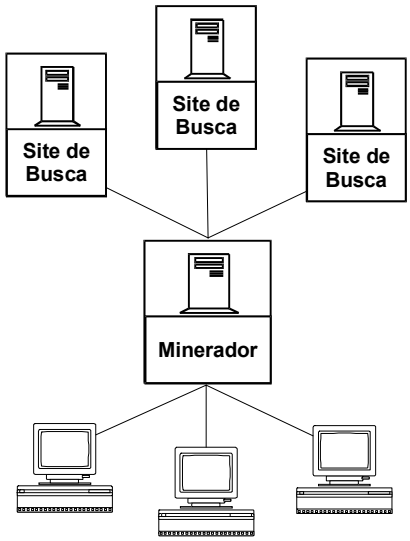

Figura 3.7 - Robôs mineradores de sites de busca

Os exemplo acima demonstram formas de recuperação de dados não estruturados, que estão disponíveis em páginas HTML. Uma alternativa para recuperar dados estruturados através da Web são as bases de dados proprietárias. Nesses BD o usuário cadastra suas informações, quando uma determinada entidade necessita de uma dessas informações ela pode acessar diretamente o *site*, ou utilizar um componente que pode ser inserido na aplicação de SI da entidade, para recuperar as informações. Um exemplo desse tipo de mecanismo é o Microsoft Wallet® que auxilia o usuário no processo de preenchimento de formulários de páginas Web, buscando as informações

que são armazenadas localmente ou em um servidor específico. A Figura 3.8 mostra esse cenário. O fato dos dados serem armazenados em uma empresa privada gera receio aos usuários quanto ao problema da privacidade.

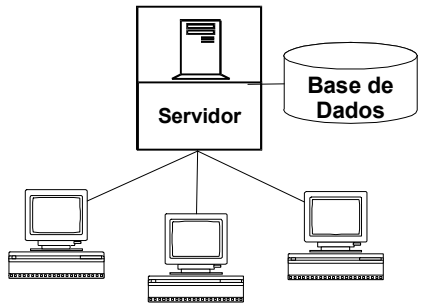

Figura 3.8 - Serviço de BD proprietário

Ao contrário dos modelos apresentados acima, o Agente proposto nesse trabalho, não realiza indexação de páginas, não acessa sites de busca representando o usuário e não mantém uma base de dados privada.

O Agente proposto neste trabalho é um Agente EDI, ou seja, um Agente de Intercâmbio Eletrônico de Dados. O objetivo dos Agentes EDI é descobrir, em uma rede de Agentes EDI quais os Agentes que possuem as informações que ele deseja recuperar. Além disso, eles devem montar uma estratégia de recuperação de informações, entrar em contato com os Agentes encontrados, solicitar as informações, aguardar o retorno das informações solicitadas e formatá-las para serem apresentas ao usuário.

Esses Agentes servem de interface de SI, portanto, descobrir qual o Agente possui determinada informação, significa descobrir qual o SI possui a informação que o usuário está procurando e determinar se o Agente que está representando o usuário possui autorização para acessar tal informação. Isso quer dizer que os Agentes só acessarão informações que são disponibilizadas com o consentimento do responsável pelas informações. A Figura 3.9 mostra esse cenário. Nessa situação o usuário acessa um Browser e faz a requisição, o Browser envia a requisição ao Agente EDI que se comunica com outros Agentes EDI para recuperar a informação.

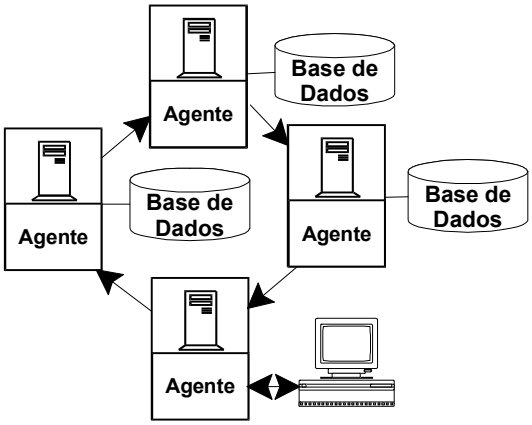

Figura 3.9 - Agente EDI proposto

Com a utilização de Agentes EDI, mesmo que, para recuperar uma determinada informação, sejam necessárias várias interações entre diferentes Agentes EDI, o usuário terá a impressão de estar acessando uma base de dados única e local.

3.10 Descrição operacional do SI Web

A utilização do modelo de SI Web proposto pode ser descrito mais detalhadamente pela seguinte seqüência de passos:

h) O usuário acessa um documento HTML onde estão os campos de dados que ele deseja recuperar.

i) Ele então preenche o campo chave para a recuperação da informação e submete o formulário.

j) O método POST é recebido por um servidor HTTP local, que pode estar situado na própria máquina ou na LAN, a qual o computador pertença.

k) O servidor HTTP encaminha a requisição ao Agente EDI através de uma interface.

l) O Agente EDI identifica os campos solicitados e procura os endereços dos Agentes que possuem as informações solicitadas.

m) O Agente monta um formulário eletrônico (e-Form) contendo os nomes dos campos e os respectivos endereços do Agentes proprietários, que será enviado ao endereço do primeiro Agente EDI relacionado na lista de Agentes do formulário.

n) O Agente EDI que recebe o e-Form preenche os dados dos campos que pertencem ao seu escopo e envia o e-Form ao próximo Agente EDI da lista. Este processo caracteriza uma espécie de anel virtual que se fecha quando o formulário retorna ao solicitante.

o) Enquanto o e-Form está sendo preenchido pelos demais Agentes, o Agente solicitante fica aguardando a resposta.

p) Ao receber o e-Form de volta o Agente solicitante converte a resposta em um documento HTML/XML que poderá ser utilizado pelo usuário da maneira que melhor lhe convier.

3.11 Considerações a respeito do Sistema

A descrição acima mostra o comportamento de uma transação. Para entender como a utilização de Agentes EDI pode tornar-se bastante popular, e para entender como o sistema irá funcionar como um todo, é necessário ponderar alguns pontos, a saber:

a) O objetivo da metodologia de utilização de Agentes EDI como ferramenta de apoio a um Sistemas de Informação baseado na Web, é criar uma rede de agentes independentes que “saibam” onde encontrar as informações desejadas de forma transparente ao usuário.

b) Os agentes poderão tanto recuperar quanto disponibilizar informações, ou seja, ele será tanto Servidor quanto Cliente. Essa é a característica que permite classificá-los como Agentes de Intercâmbio Eletrônico de Dados, ou seja, Agentes EDI.

c) O sistema é dinâmico, podendo, a todo o momento, estar recebendo um novo agente na rede, que irá disponibilizar informações. E, a rede de Agentes é descentralizada. Para poder definir quais os Agentes podem ser consultados, os Agentes irão recorrer a um serviço específico, ao qual, caberá a função de indicar quais os Agentes EDI estão conectados e disponibilizando dados em determinado momento, bem como, determinar quais os tipos de dados que cada um dos Agentes está disponibilizando.

d) A identificação e classificação de cada tipo de dado que será disponibilizado em uma rede, e a classificação do Agente, que ao ser comparada com o tipo de dado definirá o nível de visibilidade do Agente, deverá ser convencionada e normalizada por uma entidade central. Essa entidade ficará responsável, também, por fornecer uma chave de identificação ao Agente e por conceder uma assinatura digital ou certificação para fins de transações seguras.

e) O usuário que necessita de privacidade ao acesso dos dados poderá definir, no momento do registro dos tipos de dados que serão disponibilizados pelo seu Agente EDI, quais as classes de Agentes que poderão acessar os seus dados.

f) Dentro de uma mesma classe de Agentes, um Agente terá ainda um grau de visão. Isto permitirá que determinado Agente EDI disponibilize diferentes quantidades de dados aos Agentes de uma mesma classe.

g) O funcionamento completo do sistema será caracterizado como um Sistema Multiagente (SMA), ou seja, um sistema que é composto por vários agentes que interagem entre si na busca de um objetivo. Porém, os Agentes ao invés de trabalharem de forma cooperativa para alcançar um único objetivo, trabalharão de forma colaborativa para atender as necessidades de cada Agente pertencente a comunidade de Agentes.

3.12 Entidades que podem ser beneficiadas com o sistema

Essas são algumas das entidades que podem ser beneficiadas pelo Sistema de Informação Web proposto:

a) Pessoas Físicas que desejam buscar informações na Web, como por exemplo: preços e descrição de produtos, localizar empresas que fornecem determinados produtos, encontrar as especificações técnicas de determinado produto, etc.

b) Pessoas Jurídicas que desejam disponibilizar informações sobre seus produtos aos seus clientes, para fins de marketing ou para fins de comercialização. Transações do tipo B2C.

c) Pessoas Jurídicas que desejam disponibilizar informações para colaboradores, como por exemplo: listas de preços do produtos aos distribuidores, dados administrativos das filiais que serão acessados pela matriz ou por empresas terceirizadas que são responsáveis por processar estes dados. Transações do tipo B2B.

d) Pessoas Jurídicas Públicas que são responsáveis por disponibilizar informações de direito público, reservados os direitos de privacidade do indivíduo e os direitos de quem tem permissão para acessar determinadas informações. Por exemplo, empresas que trabalham com crediário podem executar uma pesquisa, junto a órgãos específicos, para terem certeza de que a ordem de pagamento que estão recebendo não possui nenhum impedimento legal. No Brasil, as transações de requisição de informações a órgãos governamentais ainda é pouco expressiva, elas

se restringem a órgãos que emitem negativas como por exemplo a Receita Federal ou o INSS, e a órgãos de Justiça como o Supremo Tribunal Federal que publica os acórdãos de julgamentos no seu site, para fins de pesquisa por parte de advogados. Porém, aos poucos, novas iniciativas têm sido disparadas nesse sentido, a mais recente é recente é a legislação que atribui validade jurídica a documentos eletrônicos.

.

4 Modelo do Agente EDI proposto

Para viabilizar o modelo de SI Web proposto na seção 3.8 é necessário a utilização de um Agente EDI. O modelo do agente proposto nesse trabalho consiste em um agente capaz de exercer o papel, tanto de Cliente, quanto de Servidor de informações, isso é, ele é capaz de recuperar e disponibilizar informações com outros Agentes EDI criando um ambiente Cliente/Servidor, no qual, os agentes são capazes de exercer ambos os papéis.

4.1 Modelo Cliente/Servidor Orientado a Agente

A atividade-fim dos Agentes é o intercâmbio de informações. Nesse cenário, os agentes desempenham dois papeis distintos: o de Servidor, quando fornecem os dados a outros Agentes, e o de Cliente, quando solicitam e recuperam os dados de outros Agentes.

Para iniciar o processo é necessário interagir com uma interface. Essa interface pode ser um Browser ou uma interface de requisição do próprio Agente EDI. Após acessar a interface o usuário indica quais são os dados que deseja recuperar e qual o campo chave de pesquisa para a recuperação da informação. A interface pode ter sido previamente configurada, no caso de ser uma interface de requisição do Agente, ou, no caso de ser um documento HTML, ter sido previamente construído. Mas ela também pode ser configurada ou construída dinamicamente. Nesse caso, o Agente identifica na rede quais os campos estão disponíveis, entre aqueles que ele pode ou deseja consultar, para montar a interface de requisição. O Diagrama de Eventos apresentado na Figura 4.1 apresenta a seqüência de eventos de uma interação entre o usuário e o Agente através de uma interface Browser.

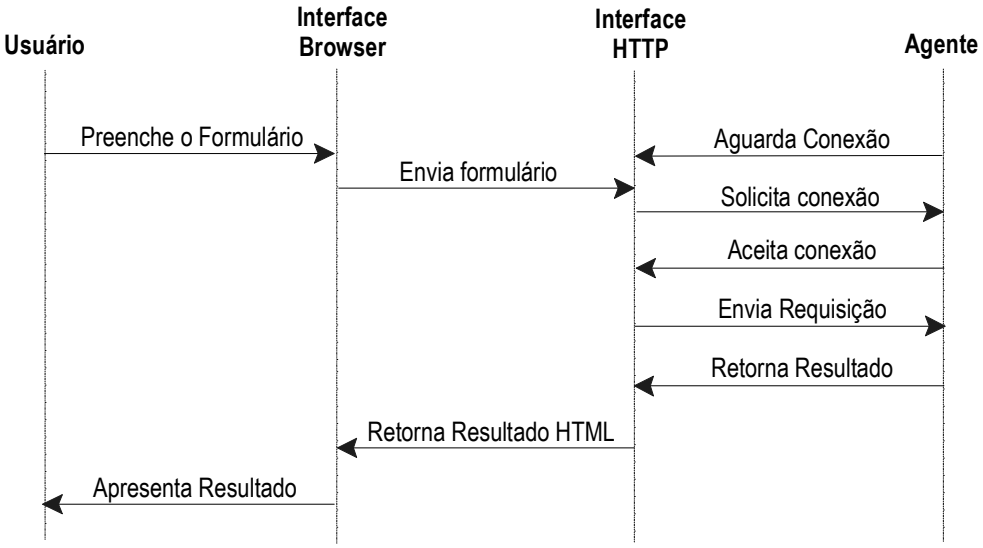

Figura 4.1 - Diagrama de Eventos da interface entre o usuário e o Agente via Browser

Quando a requisição parte de uma interface Browser, o documento HTML envia uma requisição GET/POST. Para processar essa requisição, o servidor HTTP deve executar um programa CGI ou um *Servlet*, que funcionará como interface

entre o Browser e o Agente. Essa interface será responsável pela conexão com o Agente EDI local. Ela também transformará a requisição GET/POST em uma mensagem que é reconhecida pelo Agente EDI. Se a requisição parte de um Agente, ela já é formatada como uma mensagem reconhecida pelo Agente. Portanto, o Agente recebe sempre uma mensagem reconhecida, ou seja, segue um protocolo que o Agente é capaz de identificar. O processo de conversão de uma requisição do método GET/POST para uma mensagem formatada é feito pela interface CGI ou *Servlet* que recebe a requisição do Browser.

Todo o dado estruturado é identificado por um *nome de campo*. Esse nome, apesar de identificar a mesma abstração pode variar de entidade para entidade. Na rede de agentes os nomes para identificar campos deverão ser únicos e padronizados. Isso quer dizer que, por exemplo, o nome "COD_PROD" será usado para identificar o código de um produto. Mesmo que um entidade utilize localmente um nome do tipo: "CODIGO", "CODPRD", quando ela estiver conectada na rede, ela deverá estar utilizando os *nomes de campos padronizados*. Para que não seja necessário fazer mudanças no SI da entidade, o Agente deverá permitir fazer o mapeamento dos *nomes dos campos locais* para os *nomes dos campos da rede*. Esse mapeamento funcionará como uma tradução entre a linguagem utilizada para identificar os campos locais para a linguagem de identificação dos campos dos Agentes.

Ao receber a mensagem formatada o Agente deve identificar e mapear os campos de dados locais para campos de dados da rede de Agentes EDI e criar uma lista de campos que devem ser resolvidos.

O Agente, então, conecta um Servidor de Serviço de Resolução de Endereços de Agentes EDI (SRE) que será descrito mais adiante, do qual, ele recupera os endereços dos demais Agentes que possuem os dados que serão recuperados.

O próximo passo é montar um Formulário Eletrônico, e-Form. Esse formulário possui uma formatação específica, que será discutida na seção 6.2. Depois de construído o e-Form, o Agente identifica quais os campos de dados deste que são locais e que devem ser preenchidas por ele. Se após terem sido preenchidos os campos de dados locais, ainda existirem mais campos a serem pesquisados, o Agente encaminha o e-Form ao primeiro Agente EDI da lista de campos de dados solicitados do e-Form que ainda não foi preenchido. Para que seja possível encaminhar o e-Form, a cada campo de dado solicitado é associado o endereço do Agente EDI proprietário.

O Agente que recebe o e-Form preenche os dados dos campos que pertencem ao seu escopo e envia o e-Form ao próximo Agente da lista. Quando o e-Form estiver completamente preenchido ele é retornado ao Agente solicitante. O e-Form também possui uma seção de identificação do Agente que iniciou a requisição.

Quando o Agente solicitante recebe o e-Form, ele executa o processo inverso de mapeamento de campos, ou seja, ele mapeia os campos de dados da rede de Agentes EDI, que estão no e-Form, em campos de dados locais e cria uma lista contendo o nome do campo e o seu respectivo valor. Essa lista será utilizada para criar o retorno, que é um documento HTML/XML, que poderá ser utilizado pelo usuário da maneira que melhor lhe convier.

O processo de recuperação dos dados é completamente realizado através da interação colaborativa dos Agente EDI, porém, a comunicação entre dois Agentes é feita através de requisições e respostas. Essas duas características, da rede e dos Agentes, respectivamente, caracterizam o modelo como Cliente/Servidor orientado a Agentes.

4.2 A Estratégia de Interação entre os Agentes

Dentro de uma rede de Agentes existem diversas maneiras de interação no processo de recuperação de uma informação. Essas possibilidades de interação são chamadas de Estratégias de Interação. Nessa seção serão examinadas as seguintes estratégias: Requisição Simples, Envio de Formulário Eletrônico e Agentes Móveis.

4.2.1 Requisição Simples

Nesse processo, o Agente envia uma requisição, a outro Agente, para cada campo que deseja recuperar. A Figura 4.2 mostra esse cenário. Nessa exemplo o Agente 100.0.0.1 emite uma requisição para cada um dos demais Agentes. Para cada requisição ele recebe uma resposta que deve ser processada.

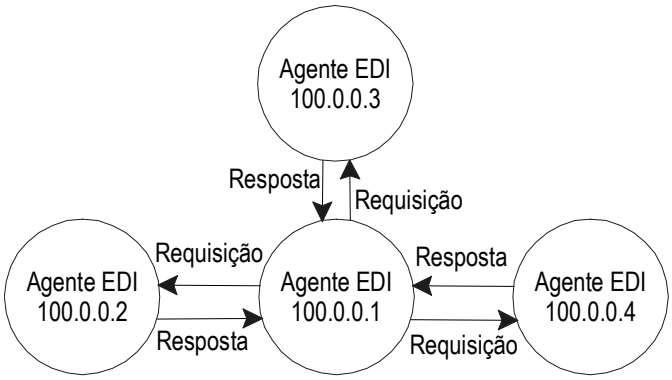

Figura 4.2 - DFD da interação através de Requisição Simples

4.2.2 Envio de Formulário Eletrônico

Nesse processo o Agente solicitante envia um formulário a outro Agente. Neste formulário estão contidos todos os campos de dados que ele pretende recuperar, o Agente que recebe a requisição preenche os campos que estão dentro de seu escopo, depois encaminha o formulário para o próximo Agente que irá completar as demais informações do formulário. Quando o formulário estiver preenchido ele é enviado novamente ao Agente que iniciou o processo. Na Figura 4.3 a requisição é iniciada pelo Agente 100.0.0.1, que envia um formulário para o Agente 100.0.0.2, esse, por sua vez, preenche a informação que lhe pertence e envia o formulário para o Agente 100.0.0.3, que repete o processo e envia o formulário para o Agente 100.0.0.4, que, por fim, completa o formulário e o retorna para o Agente solicitante, 100.0.0.1.

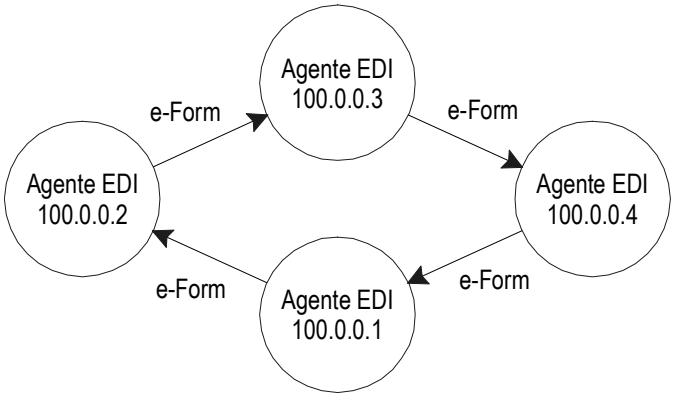

Figura 4.3 - Interação através de Envio de Formulário

4.2.3 Agente Móvel

Nessa opção o Agente sai do seu nodo e vai até um Servidor de Informações, recupera as informações desejadas e continua a navegar através dos outros servidores até conseguir a informação desejada, voltando para o seu nodo de partida quando a tarefa estiver terminada. Nesse modelo de interação o Agente exerce somente o papel de Cliente que navega entre vários servidores. Nesse caso o servidor não é um Agente, é apenas um servidor de dados que mantém uma lista de mapeamento para os dados que ele pode disponibilizar. Esta lista é formatada em um padrão conhecido pelos Agentes. Esse servidor deve também permitir a "entrada" de Agentes Móveis, funcionando como uma estação temporária ou *framework*. Uma complicação neste caso é o fato do Agente ter que "saber" como acessar BD heterogêneos além de ter que possuir a capacidade de "aprender" a acessar novas estruturas de BD sempre que necessário.

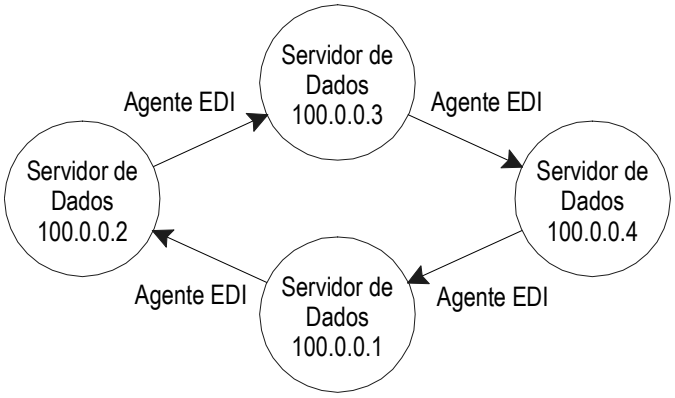

Figura 4.4 - DFD da interação através de Agentes Móveis

4.2.4 Relação entre interações

Nos modelos de interação de Requisição Simples e Envio de Formulário, apresentados acima, existe a interação entre dois Agentes. No primeiro caso a interação depende apenas da comunicação entre os Agentes, e é uma relação de 1:n, ou seja, um Agente comunica-se com *n* Agentes para conseguir recuperar uma informação, conforme mostrado na Figura 4.2.

No segundo caso, além da comunicação, os Agentes precisam manipular um formulário, porém, a relação de interação é 1:1, ou seja, o Agente comunica-se apenas com um Agente de cada vez, cria uma rede e é capaz de recuperar a mesma informação, executando no máximo duas interações, a de solicitação e a de recepção do resultado. Nesse caso é o formulário que é enviado e manipulado *n* vezes, distribuindo a carga de requisições de um Agente entre várias interações do tipo 1:1, conforme mostrado na Figura 4.3.

No modelo de Agentes Móveis, a interação é de 1:1 também, porém, a interação acontece entre um Agente e um Servidor de Dados. Nesse caso o Agente interage com um Servidor de Dados solicitando a conexão e locomovendo-se para ele. Assim como no modelo de Envio de Formulário vão existir *n* interações do tipo 1:1, porém, a carga é distribuída entre os servidores, conforme mostrado na Figura 4.4.

4.2.5 Variações de interação

Outras variações podem surgir a partir desses três modelos de interação. A primeira variação possível é a utilização de múltiplos Agentes para execução da mesma tarefa. No modelo de Requisição Simples mais de um agente pode efetuar requisições simultaneamente. No modelo de Envio de Formulário, o formulário pode ser desmembrado e distribuído entre vários Agentes que ficam responsáveis por distribuir e recuperar os formulários. No modelo de *Agentes Móveis*, mais de um agente poderia sair para buscar as informações.

Nestas variações os Agentes atuariam de forma cooperativa constituindo um Sistema Multiagente, SMA, onde vários agentes trabalham juntos para alcançar o mesmo objetivo. A mudança básica que ocorre nestas variações é a necessidade de distribuir e delegar conjuntos de dados distintos entre os Agentes e depois reorganizá-los novamente.

Outra possibilidade seria a construção de modelos híbridos, conseguida através da implementação da mobilidade nos Agentes dos dois primeiros modelos, Requisição Simples e Envio de Formulários.

A vantagem de utilização dos modelos híbridos está na recuperação de informações que necessitam de longo período de tempo para pesquisa. Nesse caso, o Agente locomove-se para um Servidor que atuaria como estação temporária. A partir daí ele passaria a atuar com seu comportamento original. Esta modalidade permitiria lançar o Agente na rede para executar a pesquisa, desconectar o computador da rede, depois de determinado período conectar-se novamente e receber o Agente com as informações recuperadas.

Nesse cenário, o Agente, apesar de ser móvel, não acessa os dados como no modelo de interação baseado em Agentes Móveis, na realidade o processo de interação continua sendo a Requisição ou o Envio de formulário, apesar dos Agentes terem a capacidade de locomoção.

4.2.6 Analise de custo das Estratégias de Interação

A análise que pode ser feita refere-se às características das interações quanto a alocação de recursos, banda de rede e tolerância a falhas. A Tabela 4.1 faz um comparativo entre as diferentes modalidades de interação.

Tabela 4.1 - Questões de desempenho entre as interações

Modelo de Interação	Alocação de Recursos	Banda de rede	Tolerância a Falhas
Requisição Simples	Maior do lado Cliente	-Aumento do número de requisições -Sobrecarga de requisições no lado-Cliente	Desnecessário
Envio de Formulário	Maior do lado Servidor	-Aumento de tráfego de dados na rede -Requisições distribuídas entre os Agentes	Tratar a perda de formulário
Agentes Móveis	Exclusivamente a cargo do Servidor	-Aumento de tráfego de dados na rede -Requisições distribuídas entre os Agentes	Garantir a locomoção do Agente

Dentro das modalidades apresentadas a de Agente Móveis é a mais onerosa. Nesta modalidade o Agente deve deslocar-se por completo do seu nodo para o nodo do servidor, gerando um aumento no tráfego da rede pois, além dos dados que serão recuperados, para cada movimentação entre servidores o agente leva consigo todo o seu "corpo", portanto torna-se necessário tomar todas as providencias para que o sistema seja tolerante a falhas. Além disto, a alocação de recursos fica exclusivamente a cargo dos servidores.

No modelo de Requisições Simples a alocação de recursos é maior no lado do cliente diminuindo a carga sobre o lado servidor. A desvantagem desta modalidade é o aumento de tráfego na rede, podendo vir a provocar atrasos e congestionamento. O aumento deve-se ao fato de que para cada campo é necessário executar uma conexão.

No modelo de Envio de Formulário a alocação de recursos é maior no lado do servidor que recebe o formulário, processando-o e redirecionando-o. Nesta modalidade, o formulário eletrônico, que é constituído por todos os campos que serão processados, representa um maior número de bytes a serem transferidos em cada conexão, porém, é necessário apenas uma conexão entre o servidor e o cliente para transferir todos os dados.

Quanto à privacidade, o modelo de Requisição Simples só necessita, assim como os demais modelos, de um mecanismo de autorização. Já o modelo de Envio de Formulário, além desse, necessita de um mecanismo para garantir um certo grau de privacidade, pois, o formulário é passado por todos os Agentes. Nesse caso, as informações constantes no formulário devem ter seu acesso permitido aos demais Agentes, ou, no caso dos dados possuírem uma hierarquia de acesso, A > B > C, a rota de recuperação da informação deve ir do dado de menor hierarquia ao de maior hierarquia, nesse caso: C→ B → A. O modelo de Agente móvel, apesar de utilizar um mecanismo de autorização, apresenta o risco do Agente estar acessando diretamente a BD, mesmo que seja permitida apenas a leitura ao Agente, ainda assim é necessário

implementar um mecanismo de permissão de acesso à informação, para garantir que o Agente não vá acessar informações para as quais o mesmo não possui permissão.

Essas considerações apontam para as seguinte conclusões:

a) A utilização de Requisições Simples é melhor quando torna-se necessário privacidade.
b) O Envio de Formulário oferece menos riscos de apresentar problemas de congestionamento e de overhead na rede, dede que, as consultas retornem dados discretos e não listas de dados. Além disso, em situações, em que não existe problema de permissão de acesso aos dados ou na qual é possível a implementação de uma hierarquia de acesso, o modelo apresenta menos custos pois distribui o processamento entre vários Agentes.
c) O Agente Móvel é o de maior custo pois depende do uso de tecnologia de gerenciamento da mobilidade do Agente. Além de necessitar de mecanismos de permissão de acesso a dados, visto que é o Agente que recupera diretamente o dado, ao contrário dos outros dois modelos, nos quais, o Agente apenas recebe o dado solicitado.

4.3 O Tipo de Conexão

Para que dois Agentes possam comunicar-se, eles precisarão utilizar um canal de comunicação. Esse canal de comunicação será fornecido pelo protocolo TCP/IP. O protocolo TCP será utilizado para criar uma conexão de fluxo contínuo e confiável através de *Socket*. Essa conexão acontece apenas entre os dois agentes durante o processo de identificação do Agente e de troca de formulários, que caracteriza uma transação. Terminada a transação o canal de comunicação é fechado.

Do ponto de vista da conexão entre dois Agentes o serviço, oferecido pelas camadas de rede, é *orientado a conexão*, enquanto que, do ponto de vista do sistema como um todo, ele se comporta como um *serviço sem conexão*.

4.4 Protocolo de Comunicação

Para poder interagir entre si, os Agentes precisam ter definido um protocolo de comunicação. Com esse protocolo os Agentes identificam os requisitos mínimos para a realização de uma transação.

A comunicação entre os Agentes consiste basicamente no processo de conexão e na troca de mensagens na forma de formulários. A utilização de um protocolo específico a esse fim pode apresentar melhores resultados relativamente a desempenho. A utilização de uma linguagem de comunicação entre Agentes como a KQML aumenta o espectro de atuação dos Agentes, podendo fazer com que eles sejam capazes de comunicar-se com outros tipos de agentes. Na realidade, um protocolo específico de comunicação é apenas um sub-conjunto de uma linguagem. Isso significa que no futuro o protocolo específico pode ser substituído por uma linguagem de agentes, sem perda da funcionalidade do mesmo. Os detalhes do protocolo específico serão discutidos nas seções posteriores.

4.5 O Serviço de Resolução de Endereços de Agentes EDI

Os três modelos necessitam de uma lista contendo os endereços de Agentes e informações sobre os dados que estes possuem. Uma alternativa, para que esta lista não seja responsabilidade de cada um dos Agentes é a utilização de um Serviço de Resolução de Endereços, SRE, de Agentes EDI.

O sistema é descentralizado e dinâmico, isso quer dizer que, a todo o momento, o SRE pode estar recebendo o *login* de um Agente que irá disponibilizar informações, portanto, a quantidade e a localização dos Agentes é variável.

A função do SRE é manter uma lista dos Agentes que estão conectados na rede em determinado momento, além de, informar os tipos de dados que cada agente está dispondo e a classe de Agentes que pode acessar esse dado.

Ao conectar-se ao Servidor SRE, o Agente EDI deve informar: seu código de identificação, seu endereço e a lista dos tipos de dados que ele irá disponibilizar. A cada dado é associado uma lista de Classes de Agentes que podem acessar o dado.

No caso de uma rede privada, estática, e de pequeno porte, não é necessária a utilização de um Servidor SRE. Cada agente pode ter uma estrutura de dados que forneça as mesmas informações que constariam no Servidor SRE.

4.6 Redes e Sub-redes

O número de Agentes que constitui uma rede pode crescer dinamicamente. Para evitar que uma rede fique muito grande, o que levaria a problemas de desempenho, é possível subdividir-se a rede principal em sub-redes funcionais, ou seja, redes orientadas a função, que constituem uma espécie de comunidades de Agentes. A subdivisão da rede serve apenas para fins de distribuição de carga, não sendo esta responsável por restringir o campo de atuação dos Agentes.

4.7 Registrando e Classificando os Agentes

4.7.1 Registrando

Para poder fazer parte de uma rede o Agente deve ser identificado. A identificação do Agente dá-se pelo registro deste em uma entidade centralizada, responsável por registrar os Agentes que podem participar da rede. Esta entidade fornece ao agente um código de identificação válido, uma assinatura digital ou um certificado, que é reconhecido por todos os Agentes que constituirão a rede.

Cada Agente registrado deve fornecer uma lista dos campos que vai disponibilizar. Na realidade o Agente irá relacionar, através de mapeamento, os nomes dos campos locais com os nomes de campos padronizados para a rede de Agentes EDI.

4.7.2 Classificando

Além disso, cada agente será classificado segundo uma Categoria ou Classe, que será previamente definida. As Classes dos Agentes servirão como filtro para determinar quais dados podem ser acessados por qual Agente. Para isso, cada dado disponibilizado será associado a um conjunto de Classes de Agente que podem acessar o mesmo. A operação de interseção entre a Classe do Agente solicitante e as Classes associadas ao dado definirão se o Agente tem permissão de acesso ao mesmo.

Para estudar as relações de hierarquia entre classes utilizaremos a seguinte notação: d(A) → {B, C}, a qual indica que um dado da classe A permite acesso as classes B e C. Implicitamente, a relação permite acesso as classes do seu mesmo tipo. Para efeito de melhor visualização, as relações de acesso implícito serão representadas pelo nome da classe sublinhada, sendo assim, a notação acima fica: d(A) → {A, B, C}.

Para saber se uma Classe Solicitante, B, tem permissão de acesso ao dado da Classe Solicitada, A, basta perguntar se B é subconjunto de d(A). Se o resultado for verdadeiro a Classe Solicitante tem permissão de acesso, caso contrário, o acesso é negado. Essa operação pode ser representada da seguinte maneira:

$$\mathbf{B = d(A) \cap B}$$

na qual, B tem que ser igual ao resultado da interseção entre d(A) e B.

Porém, para definir a ordem hierárquica entre diversas Classes, é necessário realizar o seguinte procedimento: procura-se dentro de d(x) de cada dado, as classes que são comuns entre eles; cada d(x) retorna um subconjunto contendo um certo número de classes comuns, o d(x) que contiver maior número de classes identifica o dado com menor grau de privacidade, enquanto que, aquele que possuir menos classes, isso é, disponibiliza dados de forma mais restrita, identifica o dado com maior grau de privacidade; a hierarquia entre os dados é definida então colocando-se os valores de d(x) em ordem decrescente, ou seja, do menos privado para o mais privado. Essa hierarquia permite criar uma rota de recuperação de dados, onde, serão recuperados primeiro os dados com menor privacidade antes dos dados de maior privacidade.

Por exemplo, dadas as seguintes relações de classes disponibilizadas por cada dado:

d(A) → {A, B, C, M, Z, D},

d(B) → {B, X, Y, C, D},

d(C) → {C, R, D, S, T, W},

considerando-se que um Agente da Classe D faça um pedido de consulta aos dados d(A), d(B) e d(C), o primeiro passo é determinar se a classe D é subconjunto de cada d(x): D = d(A) ∩ D, D = d(B) ∩ D, D = d(B) ∩ D. Nesse exemplo a classe D é subconjunto de todos d(x). O próximo passo é determinar, dentro de d(x), quais as classes comuns entre os dados solicitados. O resultado é o seguinte:

d(A) → {A, B, C, D},

d(B) → {B, C, D},

d(C) → {C, D},

observe, que a classe sempre irá aparecer, mesmo que ela não faça parte de outros conjuntos, isso porque a própria classe do proprietário do dado é considerada como uma classe de consulta. O resultado mostra que d(A) tem 4 classes, d(B) tem 3 classes e d(C) tem 2 classes. Isso significa que o dado da classe A é menos privado que o dado da classe B que é menos privado que o dado da classe C. Enquanto que um dado da classe A pode ser acessado por B, C e D, um dado da classe C só pode ser acessado por D, portanto a rota que garante a melhor privacidade é: A → B → C, considerando-se que A, B e C são Agentes que estão recebendo um formulário, preenchendo-o e o enviando para o outro.

Existe a possibilidade de um dado não possuir nenhuma classe comum dentro de seu d(x). Isso significa que o seu conteúdo não pode navegar pelos demais Agentes sendo necessário que a recuperação do dado seja feita através de uma interação exclusiva entre os dois Agentes.

4.7.3 Escopo das Classes

Cada Classe de Agente deve pertencer a um dos seguintes escopos: Público, Privado, Local e Protegido.

O escopo púbico informa ao SRE que aquele dado pode ser avaliado por todas as Classes. O escopo Privado restringe o acesso a Classe declarada. O escopo Protegido restringe o acesso a uma Classe e a todas as suas subclasses. O escopo local restringe o acesso às Classes que pertencem a uma sub-rede.

O escopo das Classes serve como filtro para facilitar a busca de informações em uma rede de SRE.

4.8 Possíveis cenários da Interface entre Agente e Usuário

4.8.1 Via Browser

O usuário acessa uma Interface HTML, preenche um formulário e envia a requisição. Esse opção permite três configurações distintas:

a) A interface HTML executa um *Applet*. O *Applet*, baixado do servidor HTTP e executado na máquina do browser – o processamento acontece do lado Cliente – conecta-se diretamente ao Agente EDI. O *Applet* pode ser o próprio Agente ou uma interface que irá conectar a um Agente da rede.

b) A interface HTML envia uma requisição a um programa CGI ou *Servlet*. O programa executado pelo servidor HTTP (o processamento acontece do lado do servidor) conecta-se ao Agente EDI

c) A interface HTML envia uma requisição a um programa CGI ou *Servlet* que é o próprio Agente EDI. Nesse caso o Agente funciona apenas como Cliente e é executado apenas quando o browser faz uma requisição.

4.8.2 Via Agente

O usuário acessa uma interface de acesso a dados diretamente no agente. Nesse caso a comunicação é feita inteiramente entre Agentes.

4.9 Interfaces Dinâmicas

O conteúdo da interface pode ser criado dinamicamente. Nesse caso, ao acessar a interface de requisição, o Agente consulta o SRE para ver quais os servidores que estão ativos no momento. O procedimento é válido tanto para interfaces Browser quanto para interfaces de consulta do Agente. Esse recurso permite que só sejam consultados campos de Agentes que estejam conectados à rede, diminuindo assim a chance de consultas inválidas. Como existe uma fração de tempo entre a consulta dos campos disponíveis e a requisição aos Agentes; existe a possibilidade de algum Agente que irá disponibilizar dados ser desconectado. Por esse motivo o módulo Cliente deve ser capaz de tratar requisições feitas a Agentes que não estão conectados a rede.

Quando o sistema estiver utilizando uma interface Browser, tecnologias como ASP, JSP, CGI permitem criar as páginas HTML dinamicamente.

4.10 Localização do Agente

O Agente pode estar tanto na máquina local quanto em uma LAN. O que irá determinar a localização do Agente é o tipo de interface utilizada pelo Agente. Quando a interface é a do próprio Agente. Este encontra-se na máquina local Quando a interface é um Browser, o Agente pode estar em qualquer lugar. Através de um Servidor HTTP o Agente pode estar localizado em uma LAN, em uma *Extranet* ou na própria Internet. Esse recurso aumenta a abrangência de utilização do Agente. Isso quer dizer que, o serviço de um Agente pode ser disponibilizado para usuários sem a necessidade destes terem um Agente instalado em sua máquina. Uma empresa que utiliza o Agente para recuperação de dados pode disponibilizá-lo através de uma página na Internet. A Figura 4.5 mostra o cenário de uma interface Browser acessando um Agente em uma LAN via Servidor HTTP.

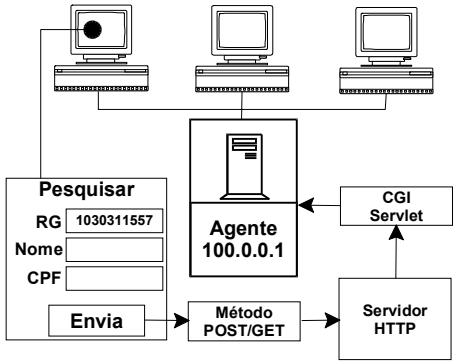

Figura 4.5 - Interface Browser acessando Agente em uma LAN

4.11 Segurança

A segurança dos dados é conseguida através da política de permissão de acesso a dados do sistema, regida pelas operações e escopo das Classes dos Agentes. Além dessas regras que restringem a visibilidade dos Agentes, o sistema ainda conta com a necessidade de identificação do Agente e da possível autorização mediante a apresentação de um certificado digital.

O modelo de envio de formulário só deve ser utilizado quando não é exigido alto grau de privacidade. Quando utiliza-se os e-Form, as informações navegam entre agentes pela rede. Como o agente é uma arquitetura que pode ser criada por qualquer programador, o agente pode ser modificado para visualizar todo o e-Form que passa por ele. Para garantir que o sistema seja seguro, apenas uma instituição deveria fazer a construção dos módulos de comunicação dos agentes, ficando a cargo dos programadores apenas a implementação dos módulos de acesso a dados e interface. No caso disso não ser possível, a opção de Requisição Simples é mais aconselhável.

A identificação do Agente serve apenas para indicar que esse é um Agente válido. Essa informação pode ser útil para gerar um arquivo de *log* que poderá ser utilizado em auditorias para localizar o Agente que fez a utilização das informações solicitadas.

O certificado é um código criptografado que serve para indicar que o Agente Solicitante tem permissão para acessar os dados solicitados.

Porém, estas medidas de segurança referem-se apenas aos processos da aplicação. É necessário que se consiga segurança na Camadas de Aplicação da rede, quando utiliza-se a interface Browser sobre HTTP, e na Camada de Transporte da rede, durante a conexão dos Agentes.

Na Camada de Aplicação a segurança pode ser implementada utilizando-se um servidor HTTP seguro, S-HTTP. Na Camada de Transporte a segurança pode ser feita utilizando SSL (*Secure Socket Layer*). Esse nível de segurança está relacionado ao tráfego da informação sobre TCP, quando o sistema não dispõem desses recursos, a maneira alternativa de conseguir segurança é a utilização de criptografia. O processo de criptografia poderia ser útil em transações baseadas em confiança. Nesse cenário, o Agente ao efetuar o *login*, junto ao SRE, informa uma chave pública para criptografia das mensagens de comunicação. Quando o Agente solicitante entra em contato com o SRE ele identifica-se através de uma senha de acesso. Essa identificação permite que o Agente receba a chave que ele irá utilizar para criptografar suas mensagens. Como o agente já recebeu uma identificação positiva do SRE, o Agente que irá fornecer os dados pode confiar na solicitação, sem a necessidade de manter um cadastro de todos os Agentes que são confiáveis. É claro que as regras de permissão de acesso permanecem valendo. A vantagem dessa técnica é poder criar uma nova chave de criptografia a cada vez que se efetua o *login* na rede.

O mesmo processo pode ser utilizado para a comunicação entre o Agente e o SRE. Nesse caso, o SRE pode criar uma chave pública para cada Agente registrado, aumentado ainda mais a segurança do sistema.

4.12 Requisitos gerais

Os requisitos comum a estes Agentes são:

a) a possibilidade de configuração da estratégia de interação entre os Agentes;

b) a capacidade de determinar a qual SRE ou a quais Agentes recorrer;

c) a capacidade de selecionar os dados desejados em uma lista que contenha os dados disponíveis na rede de Agentes EDI, sem que seja necessário ao usuário conhecer uma linguagem padronizada para executar esta tarefa;

d) facilidade de configuração da característica de Servidor de Informações no que diz respeito ao acesso a banco de dados, seja através de programação complementar ou do uso de componentes através de interfaces.

5 Arquitetura do Agente

5.1 Arquitetura Cliente/Servidor Orientada a Agentes

A arquitetura do agente é definida em cinco módulos: módulo de Configuração, módulo Servidor, módulo Cliente, o módulo de BD e módulo SRE. O DFD mostrado na Figura 5.1 apresenta a interação entre os módulos do Agente. Os detalhes dessas interações serão discutidos no decorrer da seção.

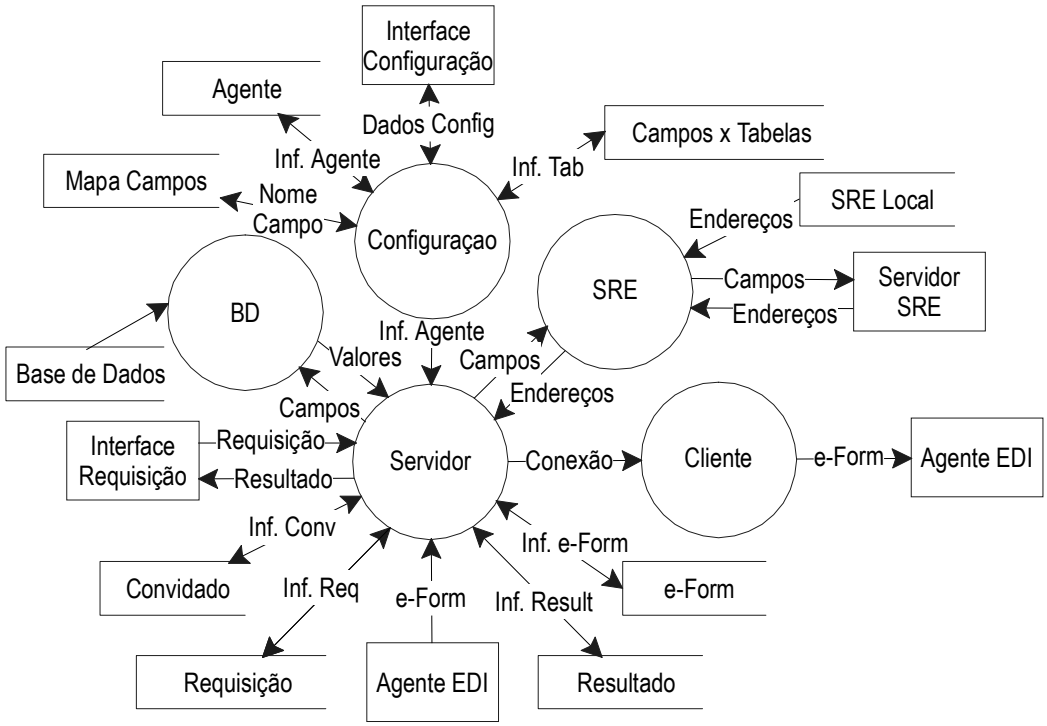

Figura 5.1 - DFD da interação dos módulos do Agente

O DFD da Figura 5.1 mostra informações sobre os dados que são trocados entre os módulos. Para entender melhor o funcionamento da Arquitetura a Figura 5.2 mostra o Diagrama de Estados do Agente durante seu processamento.

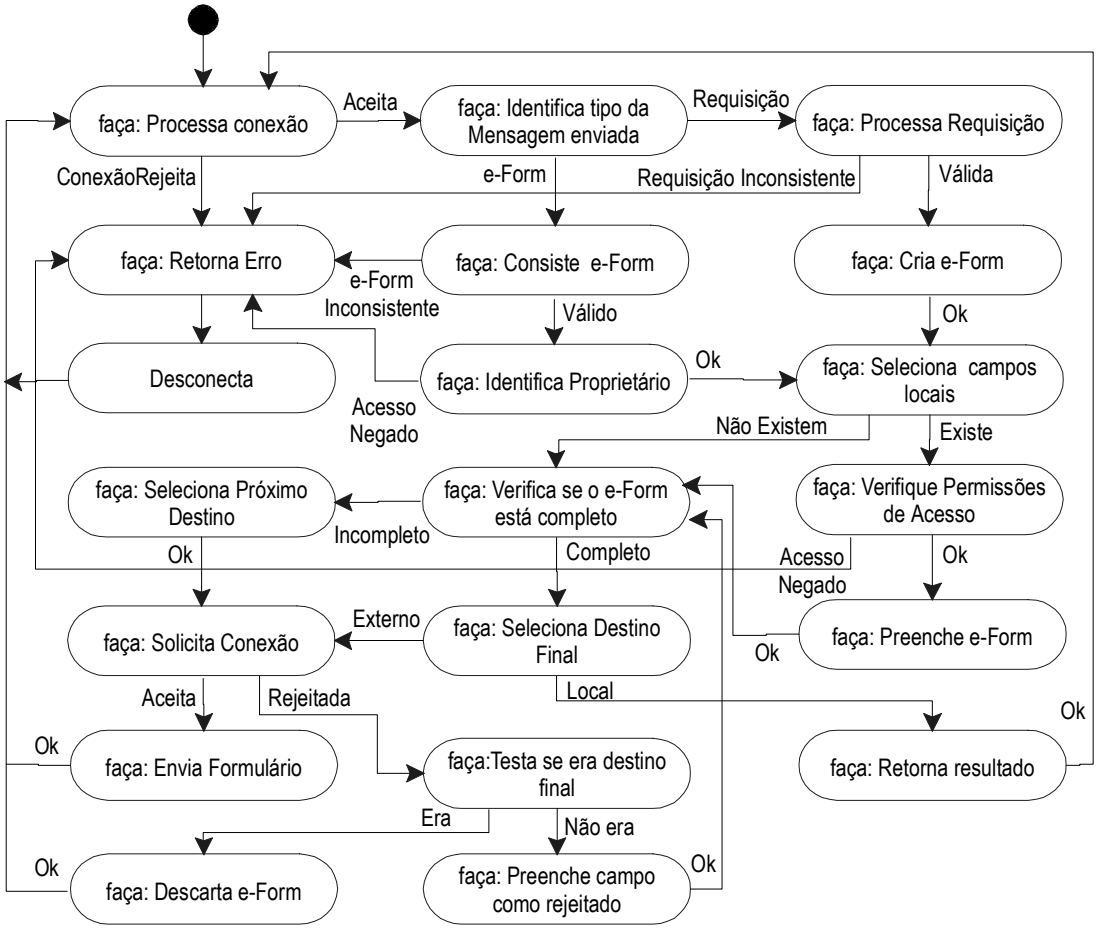

Figura 5.2 - Diagrama de Estados do Agente EDI

5.2 O Módulo de Configuração

O módulo de configuração armazena e manipula as configurações do Agente. Essas configurações são:

a) seus dados de configuração,

b) a lista para mapeamento dos nomes dos campos locais para os nomes dos campos da rede,

c) a lista de nomes dos campos locais relacionadas as suas respectivas tabelas dentro do BD.

Todas essas configurações, quando necessárias, são obtidas através do módulo de configuração. Sendo assim, ele possui duas interfaces, uma que é utilizada pelo usuário para manipular as configurações, e outra, utilizada pelo módulo Servidor para recuperar as informações necessárias para processamento das requisições e dos Formulários Eletrônicos (e-Form). A arquitetura do módulo pode ser vista na Figura 5.3.

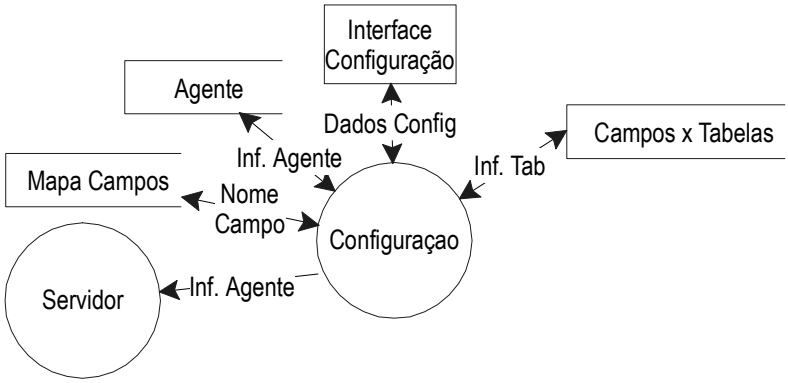

Figura 5.3 - Módulo de Configuração

5.3 O Módulo Servidor

É o módulo responsável pela maior parte do processamento do Agente. O módulo recebe uma requisição e a processa colocando-a em uma *Fila de Requisições*. Quando existir alguma requisição na fila, o processo *Cria e-Form* lê a requisição e a transforma em um e-Form, esse e-Form é colocado numa fila chamada *Fila de e-Form*. Quando houver algum e-Form na fila, o processo *Processa e-Form* lê o e-Form e o processa. Caso o e-Form seja o resultado de uma requisição, ele é transformado em um *Resultado* e é colocado na *Fila de Resultados*. Quando houver um *Resultado* na fila, o processo *Processa Resultados* lê esse resultado, associa-o à requisição que estava aguardando na *Fila de Requisições* e o envia para o processo responsável por encaminhar os resultados às Interfaces que efetuaram as requisições.

Ao processar o e-Form, no processo *Processa e-Form*, pode ser necessário que o e-Form seja encaminhado para outro Agente. Nesse caso o e-Form será encaminhado para o processo *Encaminha e-Form* que lança o e-Form para o Agente EDI de destino.

Existe ainda a possibilidade do Agente receber um e-Form enviado da rede de Agentes EDI. Nesse caso o e-Form é colocado na *Fila de e-Form*, a partir daí ele é processado como os demais e-Form.

A arquitetura do módulo pode ser vista na Figura 5.4.

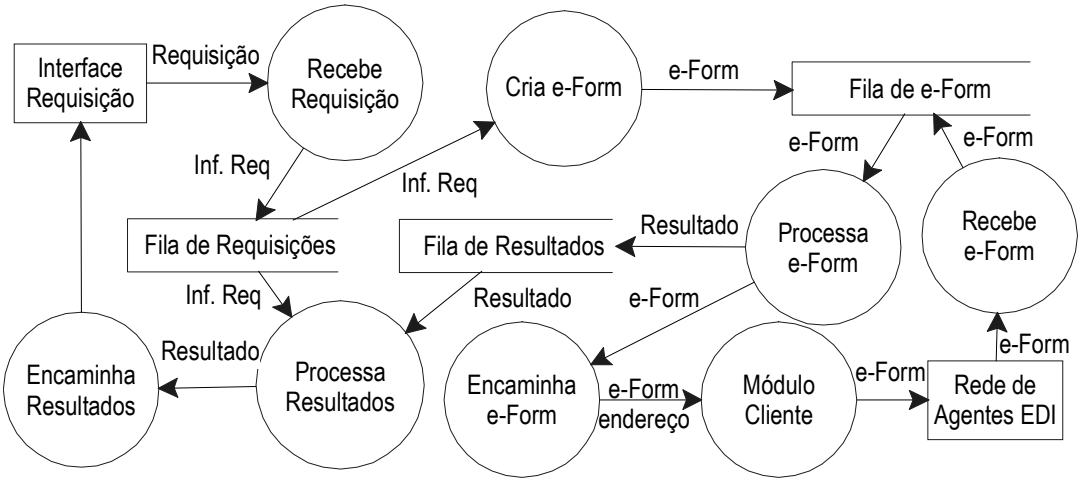

Figura 5.4 - DFD do Módulo Servidor

O módulo servidor é responsável por receber uma requisição ou e-Form, executar o processamento adequado e encaminhar o resultado. Como essas tarefas acontecem simultaneamente, ou seja, o Agente pode estar esperando o resultado de uma requisição enquanto processa outras requisições ou e-Form, os módulos do Agente e seus processos são executados independentemente, através de linhas de execução (*Threads*), significando que o Agente é um sistema *Multithreading*. A utilização de filas, para armazenar os dados que são compartilhados pelos diferentes processos é o mecanismo que permite aos processos trabalharem independentemente uns dos outros. O único cuidado que deve-se ter quando trabalha-se com dados compartilhados entre vários processos simultâneos é a sincronização de acesso aos dados.

5.3.1 Aguardando uma conexão

Para receber a requisição o Agente fica aguardando uma conexão. Ao receber um pedido de conexão é iniciado o processo de comunicação entre o lado-cliente e o lado-servidor. Nesse processo são trocadas informações quanto ao tipo de protocolo a ser utilizado na comunicação, a identificação do Agente que está solicitando a conexão e o tipo de mensagem a ser transferida. A mensagem, ou requisição propriamente dita, pode ser tanto uma requisição feita por uma interface de comunicação, um Browser ou uma interface do próprio Agente, sendo do tipo *Requisição*, ou um formulário enviado por outro Agente, sendo do tipo *e-Form*.

Quando a mensagem é do tipo *Requisição* ela é processada pelo processo *Recebe Requisição*. Quando a mensagem é do tipo *e-Form* ela é processada pelo processo *Recebe e-Form,* ambos mostrados na Figura 5.4.

Toda mensagem do tipo *Requisição* fica aguardando um *Resultado*.

5.3.2 Identificando o Solicitante

A identificação do solicitante da conexão é feita durante o processo de comunicação. No agente, a conexão serve apenas para que se estabeleça uma conexão Socket. A identificação de quem está solicitando a conexão serve para definir se o

solicitante é um Agente ou uma interface de requisição válida. Além de validar o Agente, o sistema pode necessitar identificar explicitamente o solicitante. Se a identificação for válida a conexão é autorizada, caso contrário ela é negada. Porém, para poder identificar o solicitante é necessário manter um BD que contenha as identificações de quem pode acessar os dados.

5.3.3 Recebendo uma Requisição

A mensagem do tipo *Requisição* é processada no processo *Processa Requisição*. Nesse processo, o Agente decompõe a requisição, que é um conjunto de caracteres formatados, no seu *Campo Chave de Pesquisa* e nos *Campos Requisitados*. Essas informações, juntas, geram o objeto *Informações da Requisição*. Esses objetos são enfileirados na *Fila de Requisições*. O DFD do processo pode ser visto na Figura 5.5

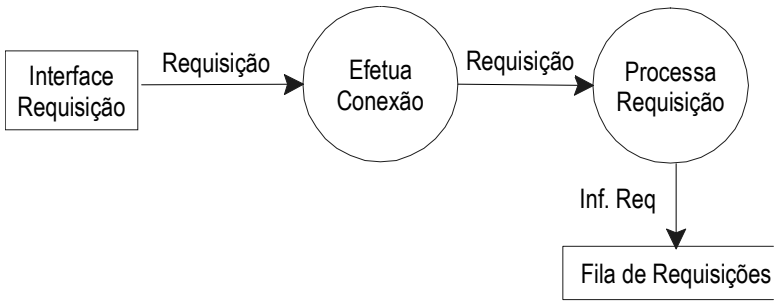

Figura 5.5 - DFD Recebe Requisição

5.3.4 Criando o e-Form

Para criar o e-Form, o processo *Monta Informações do Agente* busca as informações do Agente no módulo de Configuração e as formata numa seqüência de caracteres. O processo *Mapeia Campos* mapeia os nomes dos campos requisitados locais para os nomes de campos da rede. Depois de mapeados, o processo *Resolve Endereços* recupera o endereço de cada campo acessando o módulo SRE. A partir das *Informações das Requisição* esse processo identifica o campo *Chave de Pesquisa* que será inserido no e-Form. Para montar o e-Form, o processo *Monta e-Form* junta as informações do agente e as informações dos campos e cria uma seqüência de caracteres formatada segundo as *Definições do e-Form*. Depois de montado, o e-Form é colocado na *Fila de e-Form*. O DFD do processo *Cria e-Form* pode ser visto na Figura 5.6.

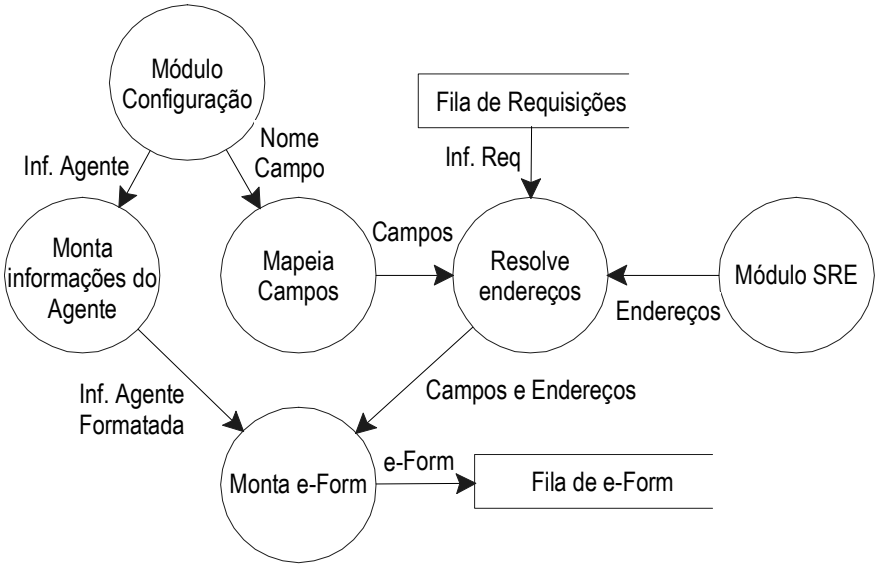

Figura 5.6 - DFD Cria e-Form

Ao criar o e-Form, o Agente pode organizar a seqüência de campos, da lista de campos do e-Form, segundo a hierarquia de permissão de acesso aos dados, explicado na seção 4.7.2. Os dados que podem ser acessados publicamente tem uma hierarquia menor, enquanto que, os dados que necessitam de privacidade tem uma hierarquia maior. Sendo assim a ordem pode ir do menor para o maior. Isso significa que o primeiro agente vai fornecer uma informação que pode ser acessada por todos, enquanto que o último Agente vai fornecer uma informação que pode ser acessada apenas por aqueles dados que tem o mesmo grau de permissão. Essa atitude aumenta a segurança do sistema, garantindo que as informações que navegam pelos Agentes não quebrem nenhuma regra de privacidade.

5.3.5 Processando o e-Form

O processo *Processa e-Form* fica aguardando até que um e-Form seja colocado na *Fila de e-Form*. O e-Form pode ter sido colocado na fila tanto pelo processo *Cria e-Form* quanto pelo processo *Recebe e-Form*. Quando um e-Form é colocado na fila, ele é lido pelo processo.

O e-Form é consistido para verificar se o e-Form é válido, além disso, a Classe do Agente solicitante é comparada com a Classe do Agente local para verificar se o Agente solicitante tem permissão de acesso aos dados. A Classe do Agente solicitante é enviada junto com o e-Form. Porém, para utilizá-la o Agente local terá que realizar uma operação baseada em confiança. Caso o Agente Local precise de uma transação segura, ele pode solicitar a Classe do Agente solicitante à entidade responsável pelo registro dos Agentes EDI. Dessa forma, essa entidade estará funcionando como Servidor de Autenticação. Outra alternativa é manter, junto com a Lista Local de Identificação de Agentes, a Classe do Agente.

Depois de consistido, o Agente procura no e-Form os campos que estão associados ao seu endereço. Esses campos são aqueles que devem ser preenchidos pelo Agente, identificando o campo chave de pesquisa. Então, o campo chave de pesquisa é

enviado, bem como uma lista contendo os campos a serem recuperados, ao módulo de BD que irá pesquisar os dados solicitados na lista de campos. O resultado retornado pelo módulo BD é inserido no e-Form.

Depois de preenchidos os campos do e-Form, o Agente identifica qual o próximo Agente a receber o e-Form, e o encaminha ao Agente através do módulo Cliente. O próximo Agente é indicado pelo endereço relacionado ao próximo campo a ser preenchido no e-Form. No caso do e-Form estar completo, o próximo Agente a recebe-lo é o Agente proprietário do mesmo, que está identificado no cabeçalho. Se o endereço do Agente proprietário for o mesmo do Agente que está processando o e-Form, isso significa que foi recebido o resultado da requisição, nesse caso o e-Form é convertido em uma seqüência de caracteres formatados segundo a *Definição do Resultado* pelo processo *Formata Resultado* e é colocado na *Fila de Resultados*. O DFD desse processo pode ser vista na Figura 5.7.

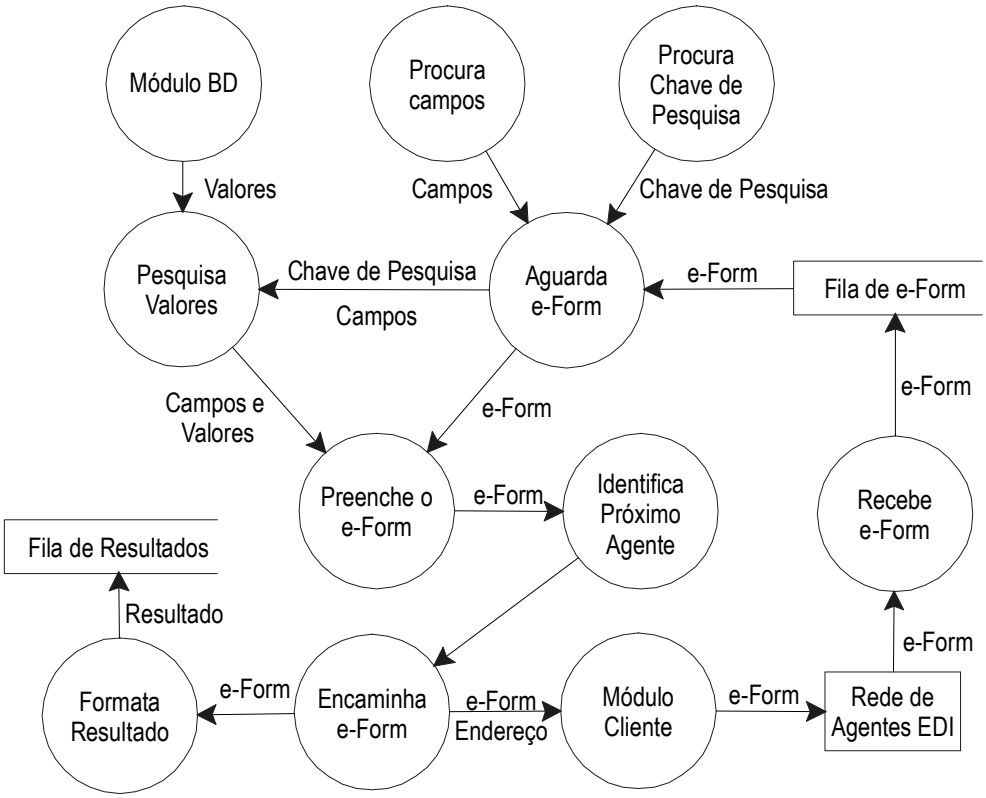

Figura 5.7 - DFD Processa e-Form

5.3.6 Recebendo um e-Form

Quando o agente recebe um e-Form, o formulário foi necessariamente enviado por um outro agente. Esse formulário pode ser apenas um formulário no qual serão preenchidos os dados que estão sob o escopo do Agente, ou, pode ser o e-Form que foi enviado pelo próprio Agente, que está retornando preenchido pela rede de Agentes. O primeiro passo é identificar o Agente proprietário do e-Form e a que Classe esse pertence. Com essas informações o Agente pode definir se o e-Form poderá ser processado ou não.

Nesse momento fica claro que existem duas identificações necessárias durante o processamento de uma requisição. A primeira, serve para identificar se o Agente pode efetuar uma conexão. Em princípio, todos os Agentes podem efetuar uma conexão e enviar uma solicitação, pois em muitos casos, os Agentes que estão solicitando a conexão só estão exercendo a tarefa de enviar um e-Form que não lhes pertence. Isso significa que um Agente que não possua permissão de acesso a dados em relação a um segundo Agente, pode ainda assim, solicitar uma conexão e passar ao Agente um e-Form em nome de outro Agente. A segunda, serve para identificar se o Agente proprietário do e-Form tem permissão de acesso aos dados. No caso desse não possuir permissão de acesso, seus campos devem ser preenchidos com uma mensagem que identifique que seu acesso foi negado. O preenchimento dos campos impede que o e-Form seja encaminhado novamente para o Agente.

Depois de autorizado o e-Form, o Agente coloca-o na *Fila de e-Form,* a partir daí, o processamento é realizado pelo processo *Processa e-Form* que trata e encaminha o e-Form de acordo com a situação, a partir do teste do conteúdo e do endereço do Agente proprietário, como descrito na seção anterior.

5.3.7 Processando o resultado

O processo *Processa Resultado* fica aguardando que um *Resultado* seja colocado na *Fila de Resultados*. Quando o resultado chega, o processo associa o resultado recebido com a sua respectiva requisição contida na *Fila de Requisições.* A requisição é retirada da fila assim como o resultado. Cada requisição está associada a uma linha de execução (*Thread*) que esta aguardando o resultado. O resultado é encaminhado para a interface que fez a solicitação pela *Thread*. A interface então apresenta o resultado para o usuário.

5.4 O Módulo de Manipulação de Dados

O módulo de Manipulação de Dados acessa a Base de Dados local localizando os dados solicitados e os retorna ao processo *Processa e-Form*. O módulo recebe o campo *Chave de Pesquisa* e a *Lista de campos* a serem recuperados, executando a pesquisa no BD local.

Este é o módulo que deverá ser distinto entre os diferentes agentes. Sua programação dependerá do BD utilizado pelo usuário. A tecnologia que torna a tarefa de acesso a BD menos complexa é a utilização de ODBC ou JDBC, porém essas opções podem interferir no desempenho do sistema. Os Agentes que optarem por uma interface de acesso a dados como ODBC, recebem o campo *Chave de Pesquisa* e a *Lista de Campos* a recuperar. Essas informações são utilizadas para montar uma instrução SQL dinâmica. A *tabela resultado* é processada através de operações de *Fecth* que permitem a movimentação do *Cursor* que é o mecanismo disposto pelo SQL, embutido ou dinâmico, que permite o posicionamento nas linhas da *tabela de resultado* para que possam ser processadas individualmente pela linguagem hospedeira. Esse recurso permite que o resultado seja formatado e processado de acordo pelo módulo de DB.

Quando a instrução SQL retorna uma *tabela resultado* vazia, indica que a condição de consulta não foi satisfeita. Nesse caso, o módulo deve retornar uma informação, no lugar do valor de retorno do campo, que indica que o resultado não foi encontrado.

5.5 O Módulo Cliente

O módulo Cliente recebe o e-Form a ser encaminhado, junto com o endereço do Agente a quem enviar o formulário, conecta-se ao Agente destinatário e envia o e-Form. Para poder enviar o e-Form, o agente executa uma conexão *Socket* sobre TCP, depois de conectado os Agentes iniciam uma comunicação através de um protocolo específico, que tem por finalidade identificar: o Agente, o protocolo a ser utilizado na comunicação e o tipo de mensagem a ser enviada.

5.6 O Módulo SRE

O módulo SRE recebe o nome do *Campo* solicitado, procura o endereço do campo numa base local que possui a mesma estrutura de um Serviço SRE. Esse recurso é útil em pesquisas locais ou em sistemas que executam pesquisas estáticas, ou seja, pesquisam sempre os mesmos dados. No caso de o campo não ser localizado na base local o Agente faz uma requisição ao Serviço SRE da rede de Agentes. Para realizar essa requisição, o Agente executa o processo de requisição semelhante a qualquer outra requisição de dados estruturados realizado pelo Agente. Numa requisição normal o Agente informa o campo *Chave de Pesquisa* e os campos a serem recuperados junto com o endereço do Agente a quem enviar a requisição ao módulo Cliente. No caso do SRE o Agente informa ao módulo Cliente a *Chave de Pesquisa,* que é a *Classe do Agente*, a lista de campos e o endereço do Serviço SRE. A diferença acontece no Serviço SRE que utiliza a *Classe do Agente* como filtro para identificar se o Agente tem permissão de acesso aos campos solicitados e retorna, como valor de cada campo requisitado, o endereço do Agente proprietário do campo. Outra diferença é que a conexão executada pelo módulo Cliente utiliza a interação de Requisição Simples, ou seja, o processo fica conectado até receber o resultado. O DFD da Figura 5.8 apresenta o processo SRE.

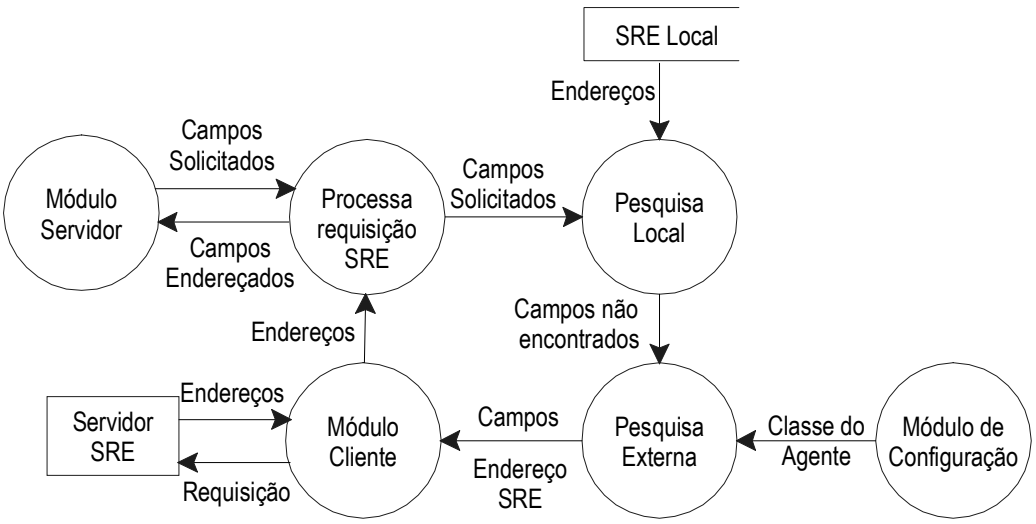

Figura 5.8 - DFD Módulo SRE

Os Servidores SRE podem constituir uma rede para distribuir a carga de processamento e descentralizar o serviço, interagindo entre si quando necessário.

6 Projeto do Agente

6.1 Definindo a Requisição

A requisição é uma seqüência de caracteres que contém informações sobre o campo *Chave de Pesquisa* e a *Lista de Campos* a serem solicitados, a Tabela 6.1 mostra o dicionário de dados da requisição.

Tabela 6.1 - Dicionário de Dados Requisição

Requisição = Campo Chave + Lista de Campos
Campo Chave = Nome + "=" + Valor
Lista de Campos = [Nome]
Nome = String
Valor = String

Os dados são delimitados por *Tags*, para facilitar a identificação dos mesmos. A Tabela 6.2 mostra as tags da requisição. É importante observar que nesta estrutura não existe uma *Tag* para a Lista de Campos. Como só existem dois tipos de dados, o campo chave e os campos requisitados, ao ser identificado o campo chave, o restante são os campos solicitados. Além disso, nessa fase, o nome dos campos solicitados é o mesmo utilizado para identificar os campos locais, ou seja, eles ainda não foram mapeados para nomes de campos da rede de Agentes.

Tabela 6.2 - *Tags* da Requisição

Identificador	***Tag* início**	***Tag* fim**
Campo Chave	<PK>	</PK>
Campos	<F>	</F>

Esquema de uma requisição:

<PK>nome=valor</PK><F>campo_1</F><F>campo_2</F><F>campo_n</F>

6.2 Definindo o e-Form

Ao criar um e-Form, o Agente o coloca na *Fila de e-Form*. A decisão por utilizar uma fila de e-Form permite que o Agente opte por montar um único e-Form, que contém todos os campos solicitados, ou vários e-Form, cada um com apenas um campo. Com isso é possível configurar o Agente tanto para realizar interações de *Envio de Formulários,* quanto para Requisição Simples. É ainda possível trabalhar com servidores híbridos, nos quais, no momento do mapeamento dos campos de dados, é possível saber se o Agente EDI a quem vai ser enviada a requisição, interage sob *Requisição Simples* ou *Envio de Formulários*, ou seja, o agente envia um e-Form para os Agentes que interagem através de *Envio de Formulário*, e envia *Requisições Simples* para os demais. Na opção híbrida, o agente tem que "lembrar" que ele só pode retornar o resultado quando todos e-Form de uma requisição foram retornados. Isso é possível pois o Agente mantém uma *Lista de Requisições* com os identificadores dos e-Form que foram enviados. A cada vez que o agente recebe um formulário, ele tira da lista o

identificador do formulário retornado. Quando a lista estiver vazia o retorno estará completo.

A Tabela 6.3 apresenta o dicionário de dados do e-Form.

Tabela 6.3 - Dicionário de Dados do e-Form

e-Form = Identificação do Agente + Campos Chave + Lista de Campos
Identificação do Agente = Identificador + Endereço IP + Porta TCP + Classe + Chave de Certificado
Identificador = String
Endereço IP = String
Porta TCP = String Numérico
Classe = String
Chave de Certificado = String
Campos Chave = [Chaves]
Chaves = Nome + “=” + Valor
Nome = String
Valor = String
Lista de Campos = [Registro]
Registro = [Campos]
Campos = Nome + Valor + Endereço do Campo + Porta do Campo
Endereço do Campo = String
Porta do Campo = String

A Tabela 6.4 mostra a lista de *Tags* do e-Form

Tabela 6.4 - *Tags* do e-Form

Identificador	***Tag* início**	***Tag* fim**	**OBS**
e-Form	<AEDI>	</AEDI>	1
Agente	<AG>	</AG>	1
Identificador	<ID>	</ID>	1
Endereço Agente	<ADR>	</ADR>	1
Porta Agente	<PORT>	</PORT>	1
Classe Agente	<CL>	</CL>	-
Chave de Certificado	<CK>	</CK>	-
Chave de Pesquisa	<PK>	</PK>	1
Chave	<K>	</K>	2
Nome Campo	<N>	</N>	1
Valor Campo	<V>	</V>	1, 3
Lista de Campos	<L>	</L>	1, 5
Registro	<R>	</R>	1, 5
Campo	<F>	</F>	1
Endereço do Campo	<A>	</A>	1, 4
Porta do Campo	<P>	</P>	1, 4

As seguintes orientações estão indicadas na Tabela 6.4 e referem-se aos seus respectivos identificadores:

1. Esses *Tags* são obrigatórios. No caso de *tags* não obrigatórios, que são usados pelo agente, e que não são informados, o agente considera o menor valor, ou a ausência de valor como sendo o *default*. O *Tag valor* só é obrigatório nas chaves de pesquisa.
2. É obrigatório pelo menos um campo, mas podem ser n campos.
3. Na lista de campos só é obrigatório quando é preenchido pelo agente. A ausência do campo indica que aquele dado ainda não foi processado. Se o agente responsável pelo dado não conseguir resolver o dado, ele deve preencher o campo com uma justificativa do tipo: acesso negado, não encontrado, campo inválido, etc, ou simplesmente com EMPTY.
4. O endereço, junto com a porta servem para identificar o endereço do Agente. Em geral apenas o endereço do Socket seria necessário, porém, enquanto numa rede, cada máquina possui um endereço IP e os processos se comunicam usando a mesma porta, para poder executar mais de um agente na mesma máquina, é necessário que cada agente atenda em uma porta diferente, pois todos possuem o mesmo endereço.
5. Uma lista <L> é composta por uma série de registros <R>. Cada registro é composto por um ou mais campos <F>. A lista conterá mais de um registro quando os mesmo nomes de campos forem encontrados em mais de um agente, construindo assim uma pesquisa composta

Esquema de uma requisição:

<AEDI><AG><ID>Identificador</ID><ADR>Endereço_IP</ADR><PORT></PORT
>Porta_TCP<CL>Classe</CL><CK>Chave_de_Certificado</CK></AG><PK><K><N
>Nome</N><V>Valor</V></K></PK><L><R><F><N>Nome</N><V>Valor</V><A
>Endereço_do_Campo</A><P>Porta_do_Campo</P></F></R></L></AEDI>

6.3 Definindo o Protocolo de Comunicação

O Protocolo de Comunicação serve para que os Agentes possam identificar Agentes Válidos, através da troca de informações depois de efetuada uma conexão *Socket*.

A solicitação de conexão sempre ocorre do lado-cliente para o lado-servidor. A rotina básica de comunicação do lado-servidor consiste em identificar o protocolo a ser utilizado na comunicação, identificar quem está solicitando a conexão, identificar o tipo de requisição que será enviada e receber a mensagem

A Figura 6.1 mostra o diagrama de eventos do processo de comunicação entre o lado-cliente e o lado-servidor dos Agentes. A primeira informação trocada é o tipo de protocolo a ser utilizado na comunicação. Esse recurso permite que sejam desenvolvidos outros protocolos mais complexos para serem utilizados pelos Agentes. Esses protocolos poderão ser privados, para atender a requisitos particulares de entidades, ou públicos, como por exemplo a linguagem KQML. O protocolo definido

como versão básica para a comunicação entre Agentes EDI será chamado de "AEDI1.0".

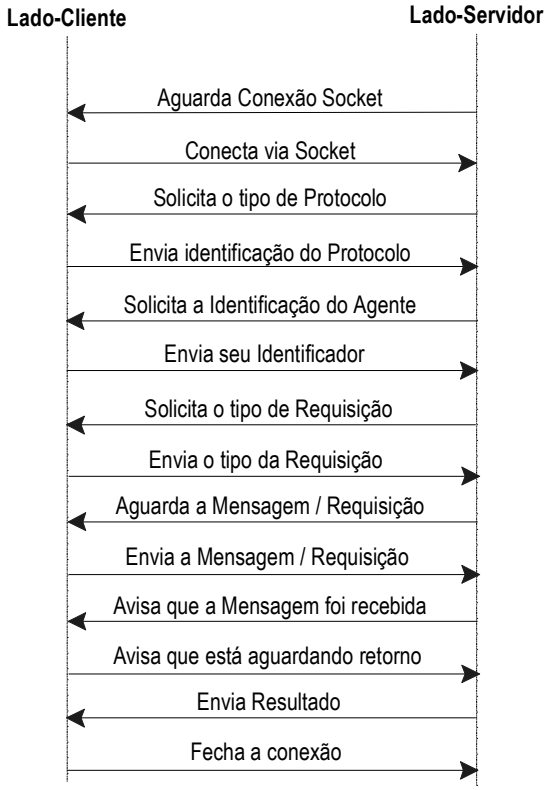

Figura 6.1 - Diagrama de Eventos do Protocolo de Comunicação AEDI1.0

Os identificadores do protocolo AEDI1.0 utilizados pelos Agentes estão na Tabela 6.5. Todos os identificadores são conjuntos de caracteres formatados em Caixa Alta. A mensagem de erro pode vir acompanhada de uma mensagem complementar. Quando o lado-cliente recebe um erro ele é apresentado como sendo o resultado da requisição, isso permite ao usuário identificar onde está localizado o erro. Os erros que o servidor pode enviar são: "ERRO:TIPO INVALIDO" e "ERRO:IDENTIFICACAO INVALIDA". Qualquer mensagem que seja enviada fora desse protocolo é considerada como fim de conexão, o lado-servidor simplesmente finaliza a conexão e o lado-cliente também, mostrando como retorno ao usuário a mensagem: "CONEXAO FINALIZADA".

Tabela 6.5 - Identificadores do Protocolo AEDI1.0

Identificador	Descrição	Escopo
PROTOCOLO	Solicita o tipo de protocolo de comunicação	Servidor
IDENTIFIQUE-SE	Solicita a Identificação do Agente solicitante	Servidor
TIPO MENSAGEM	Solicita o tipo de mensagem a ser enviada	Servidor
AGUARDO MENSAGEM	Informa que está aguardando a mensagem	Servidor
MENSAGEM RECEBIDA	Informa que recebeu a mensagem	Servidor
AGUARDO RETORNO	Informa que está aguardando o retorno	Cliente
QUIT	Finalizou a conexão	Ambos
REQUISICAO	Tipo de mensagem	Cliente
EFORM	Tipo de mensagem	Cliente
ERRO:	Informa que houve um erro	Servidor

6.4 Definindo as Configurações do Agente

As configurações do Agente são armazenadas em um arquivo local chamado "AgentEDI.dat". Os campos desse arquivo são descritos na Tabela 6.6.

Tabela 6.6 - Campos de Configuração do Agente

Campo	Descrição	Exemplo
Caption	Legenda que aparece na Janela	Agente EDI Genérico
Address	Endereço TCP	127.0.0.1
Port	Endereço da porta TCP do agente	5000
QueueLength	Número de conexões Socket simultâneas	100
IdAgent	Identificador do agente (único)	AEDI_1031311557
IdAlias	Nome alternativo para o Agente	Meu Agente
ClassView	Classe do Agente	FARMACIA
Protocol	Versão do protocolo	AEDI1.0
Certificate	Chave de certificação	0001000020000300004
LocalDataBase	Nome ODBC para dados locais	agLocal
localDataBaseSRE	Nome ODBC para dados locais SRE	agLocalSRE

Exemplo de um arquivo de configuração de Agentes:

```
#Propriedades do Agente
#Fri Jul 27 17:22:01 GMT-03:00 2001
queueLength=100
idAgent=AEDI_04
caption=Agente\ EDI
port=5004
address=127.0.0.1
idAlias=Meu\ Varejo
localDataBaseSRE=agSRE
certificate=0001000020000300004
classView=VAREJO
protocol=AEDI1.0
localDataBase=agLocal
```

6.5 Definindo a Lista de Campos x Tabelas

A lista *Campos x Tabelas* é utilizada pelo módulo de configuração para enviar ao módulo de BD o nome das tabelas do BD local relacionadas aos campos solicitados. Essa informação permite montar a instrução SQL que será executada pelo módulo BD.

A lista é armazenada em um arquivo texto chamado "TabelaDB.dat". Cada linha do arquivo contém o nome do campo, seguido do sinal de "=", seguido do nome da tabela do banco de dados. Exemplo:

```
#Tabela de campos do DB
#Thu Jul 26 17:22:45 GMT-03:00 2001
P_VENDA=PRODUTO
EST_ATUAL=PRODUTO
COD_PROD=PRODUTO
P_COMPRA=PRODUTO
PRODUTO=PRODUTO
```

6.6 Definindo a Lista Mapa Campos

A lista *Mapa Campos* é utilizada pelo módulo de configuração para informar ao processo *Cria e-Form* e ao processo *Processa Resultado* o nome dos campos locais e seu mapeamento para o nome dos campos da rede de Agentes.

A lista é armazenada em um arquivo texto chamado "Mapeamento.dat". Cada linha do arquivo contém o nome do campo local, seguido do sinal de "=", seguido do nome do campo da rede de Agentes. Exemplo:

```
#Mapeamento de variáveis
#Wed Aug 01 11:32:22 GMT-03:00 2001
PRECOCOMPRA=P_COMPRA
CODIGO=COD_PROD
ESTOQUE=EST_ATUAL
PRECOVENDA=P_VENDA
DESCRICAO=PRODUTO
ICMS=ICMS
```

6.7 Definindo o Resultado

Quando o Agente recebe o e-Form que ele havia lançado na rede, ele mapeia os nomes dos campos da rede de Agentes contidos no e-Form para os nomes dos campos locais. O Agente, então, seleciona todos os campos com seus respectivos valores e monta o *Resultado*, que é uma seqüência de caracteres, composta pelo campo chave de pesquisa e pela lista de campos recuperados com seus respectivos valores. A Tabela 6.7 mostra o dicionário de dados do *Resultado*.

Tabela 6.7 - Dicionário de Dados do Resultado

Resultado = Campo Chave + Lista de Campos
Campo Chave = Nome + Valor
Nome = String
Valor = String
Lista de Campos = [Registros]
Registros = [Campo]
Campo = Nome + Valor

A Tabela 6.8 mostra as *Tags* do *Resultado*.

Tabela 6.8 - *Tags* do Resultado

Identificador	***Tag* início**	***Tag* fim**
Campo Chave	<PK>	</PK>
Nome	<N>	</N>
Valor	<V>	</V>
Lista de Campos	<L>	</L>
Registros	<R>	</R>
Campo	<F>	</F>

Esquema de uma requisição:

<FK><N>nome</N><V>valor</V></FK><L><R><F>campo_1</F><F>campo_2</F><F>campo_n</F></R></L>

6.8 Definindo o SRE

A estrutura básica do SRE consiste em um BD que deve conter informações sobre o nome do Agente, o campo disponibilizado, o campo chave de pesquisa usado pelo Agente para encontrar o conteúdo do campo, o endereço do Agente, as classes que tem permissão de acesso ao dado e o status que indica se o Agente está conectado ou não. A Tabela 6.9 apresenta a estrutura de dados necessária para um Servidor SRE.

Tabela 6.9 - Estrutura de dados do SRE

Campo	Descrição	Exemplo
AgentID	ID do Agente Proprietário	AEDI_01
Field	Nome do Campo disponibilizado	PRODUTO
FieldKey	Campo Chave de Pesquisa	COD_PROD
Address	Endereço IP do Agente Proprietário	127.0.0.1
Port	Porta TCP do Agente Proprietário	5001
ClassView	Classes com permissão de acesso	VAREJO.ATACADO.INDUSTRIA
Status	Identifica se o agente está conectado	TRUE

6.9 Cenário de utilização do Agente

Para formular uma idéia mais concreta de como será o funcionamento do Sistema de Informação descrito, o cenário seguinte, onde todas as entidades utilizam Agentes EDI, apresenta de forma resumida, a cadeia de transações de compra de um produto, que vai da Indústria, passando pelo Atacado, pelo Varejo até chegar ao consumidor final.

Uma Indústria produz um determinado produto. Esse produto é cadastrado no SI da empresa. As informações que serão cadastradas são aquelas que apresentam alguma funcionalidade para a empresa. Duas são as informações que nos interessam neste exemplo:

a) O preço de venda do produto.

b) As informações que identificam o produto.

A Indústria vende o seu produto ao Atacado. No processo tradicional a Indústria irá cadastrar o produto em seu SI. O Atacado necessita apenas de um subconjunto das informações que a Indústria possui sobre o mesmo. Por exemplo, o Atacado precisará identificar o produto, como essa operação já foi realizada pela Indústria, as informações podem ser reaproveitadas pelo Atacado. A mudança significativa acontece no preço de venda do produto da Indústria, que no Atacado passa a ser identificado como Preço de Compra.

Quando o produto chegar ao Atacado, o Agente EDI vai até a Indústria e busca as informações da Nota de Compra, este formulário possui os códigos que identificam o mesmo. Ao invés de armazenar todos os dados do produto em seu SI, o Atacado armazena somente o código de identificação do produto (ID). Quando for necessário emitir uma listagem, ou algum formulário que utilize dados complementares do produto, o Agente EDI do Atacado solicita ao Agente EDI da Indústria, as informações desejadas. O campo ID é utilizado como chave de pesquisa pelo Agente EDI da Indústria. Esse processo permite ao Atacado diminuir seu TCO por não ser necessário despender tempo no cadastro de novos produtos e na manutenção de informações complementares do seu SI, além de ter certeza da integridade dos dados quando estão sendo utilizados.

A etapa seguinte consiste na transação de compra e venda entre o Atacado e o Varejo, Neste caso, todo o processo descrito na etapa anterior, transação de compra e venda entre a Industria e o Atacado, se repete. Nesta transação, assim como aconteceu na anterior, o preço de venda do produto do Atacado passa a ser o preço de compra do Varejo. A diferença acontece no comportamento do Agente EDI. Nesta etapa o Agente EDI recupera a informação do preço do produto no Atacado, mas recupera as informações do produto na Indústria, assim como faz o Atacado. Neste momento fica claro que a função do Agente EDI é recuperar a informação diretamente na fonte geradora da informação.

Por fim, o Cliente compra o produto do Varejo na Web. O Cliente solicita ao seu Agente EDI o preço e os dados do produto que ele deseja comprar. O Agente EDI encontra o preço do produto no Varejo e os dados do produto na Indústria. A Figura 6.2 apresenta este cenário

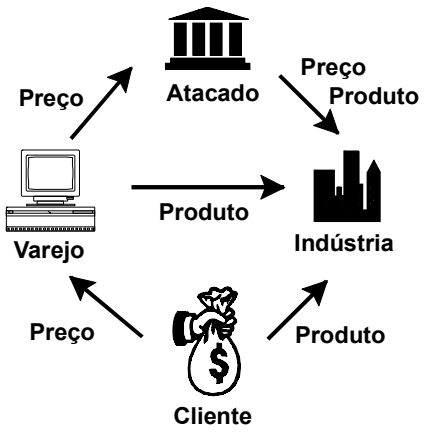

Figura 6.2 - Intercâmbio de dados na cadeia de compra de um produto

O papel dos Agente EDI descrito até agora foi o de solicitante de informação. Porém, durante o processo, os Agente EDI da Indústria, do Atacado e do Varejo exerceram também o papel de Servidores de Informação. O ponto importante a ser salientado é que:

a) O Agente EDI da Indústria é Servidor de Dados do Produto para o Atacado, para o Varejo e para o Cliente, mas é Servidor de Preço do Produto somente para o Agente EDI do Atacado.

b) O Agente EDI do Atacado é Servidor de Preço do Produto somente para o Agente EDI do Varejo.

c) O Agente EDI do Varejo é Servidor de Preço do Produto somente para o Agente EDI do Cliente.

Estas restrições são decorrentes da análise da permissão de acesso através das Classes dos Agentes. A Tabela 6.10 mostra a relação entre as Classes dos Agentes, nesse exemplo identificadas pelo próprio nome do Agente, e as Classes com acesso permitido para cada campo de dado.

Tabela 6.10 - Análise da permissão de acesso através das Classes dos Agentes

Agente	Campo	Classes Permitidas
Industria	Produto	Industria, Atacado, Varejo, Cliente
Industria	Preço	Industria, Atacado
Atacado	Preço	Atacado, Varejo
Varejo	Preço	Varejo, Cliente

Neste cenário, o Agente da Indústria é Servidor de Dados do Produto para o Atacado, para o Varejo e para o Cliente, mas é Servidor de Preço do Produto somente para o Agente do Atacado. O Agente do Atacado é Servidor de Preço do Produto somente para o Agente do Varejo. O Agente do Varejo é Servidor de Preço do Produto somente para o Agente do Cliente. Todas as entidades utilizam os Dados do Produto, fornecidos pela Indústria, porém o Preço do Produto segue uma cadeia hierárquica, pois essa informação só pode ser acessada por Agentes específicos.

Esse exemplo mostra algumas das Classes que podem ser aplicadas aos Agentes: Indústria, Distribuidor, Varejo e Cliente.

6.10O cenário com uma única requisição

Outra possibilidade, onde não é necessário o acesso constante às bases de dados, é utilizar o agente para recuperar dados de forma esporádica. Em sistemas de informações que controlam transações comerciais é comum o armazenamento dos dados da transação. No caso da compra de mercadorias, o usuário lança os dados da nota de compra no sistema. Esse processo pode ser manual ou automático. Os mecanismos de lançamento automático podem ser: a utilização de mídia ou de acesso via intranet ou Internet. O sistema de mídias está sujeito a problemas de troca, extravio, ou dano de mídia. O sistema de acesso à rede prevê a qualificação do usuário. Tanto em um sistema quanto em outro é necessário o desenvolvimento de um canal de comunicação entre o sistema do fornecedor e do usuário, para que a operação torne-se transparente ao último. Levando-se em conta os diferentes níveis de tecnologia utilizados entre as partes, esse pode ser um processo bastante complicado, principalmente para quem desenvolve o sistema para o usuário final. É necessário o desenvolvimento de uma interface para cada fornecedor. A solução adotada é a utilização de documentos EDI. Porém essa alternativa padroniza apenas o formato do documento. As tecnologias de acesso e interação variam de entidade para entidade. Nesse contexto, o Agente EDI serviria como uma camada de abstração que ficaria responsável pela comunicação e acesso a dados entre as entidades. Para quem desenvolve o sistema bastaria configurar e criar uma interface entre o seu

sistema e o Agente EDI. Para quem recebe a requisição do agente, bastaria ter um Agente EDI aguardando as solicitações.

Ao lançar uma Nota Fiscal, o Agente do comprador recebe o número da nota fiscal, que é a chave de pesquisa, e os campos que deseja recuperar; o Agente EDI se encarrega de encaminhar a requisição e recuperar os dados. Independente de quem forneceu os dados, o sistema local precisará apenas processar os dados fornecidos pelo Agente.

6.11 Implementação do Agente

A linguagem a ser utilizada para programar o Agente, pode ser qualquer uma que possua recursos de gravação em dispositivos de entrada e saída padrão, tais como: Visual Basic, C++, Delphi, Java, etc. A escolha da linguagem pode depender também do número de bibliotecas ou classes, que estão disponíveis na linguagem, que facilitem a manipulação de conexões *Sockets* e permita a programação *Multithreading*. Para a programação da interface de comunicação com o Agente via browser, ainda é necessário a elaboração de documentos HTML. Como a interface HTML através de uma camada que pode ser um CGI ou *Servlet*, a linguagem escolhida deve também permitir a criação dessa interface.

Ainda quando é utilizada uma interface Browser, é necessário estar instalado na rede ou na máquina local do usuário um Servidor HTTP, que irá receber a requisição do Browser, acionar o CGI ou *Servlet* que irá se conectar ao Agente.

Para que o Agente funcione, independente da interface utilizada, é necessário que esteja instalado no Sistema Operacional o protocolo TCP/IP, isso para permitir que o Agente possa ser conectado a rede via uma conexão *Socket*.

Na prototipação do Agente apresentado nesse trabalho, a linguagem utilizada foi o Java. A escolha baseou-se nos seguintes fatores da linguagem que:

a) possui o *Package* java.net que possui um conjunto de classes capaz de manipular conexões *Sockets,*
b) possui os *Packages* javax.servlet e java.servlet.http que são utilizados para realizar operações sobre o protocolo HTTP,
c) é capaz de criar *Servlets,*
d) é portável.

As classes do Agente foram programadas em diversos ambientes de programação. O sistema funcionou perfeitamente quando compilado e executado no JBuilder 3.5, JBuilder 4.0, IBM Visualage for JAVA 2.0 e o NetBeans DeveloperX2 2.1. Já o Visual J++ não reconheceu as classes importadas não sendo possível executar o programa com essa linguagem.

Para poder executar as classes dos Agente, foi instalado o JDK (*Java Development Kit*) 1.2.1, o qual, utiliza o JRE (*Java Runtime Environment*) da mesma versão. Para ter acesso as classes *Socket* e *Servlet* foi necessário instalar também o J2EE (*Java to Enterprise Edition*) 1.3.

No protótipo a interface via browser foi a escolhida para executar a comunicação com o Agente. Isso tornou necessário a instalação de um Servidor de HTTP. O servidor deveria ser capaz de interpretar *Servlets*, sendo assim, optou-se pelo *Jigsaw*, um Servidor Web, gratuito, completamente desenvolvido em Java, que foi instalado sobre o Windows 2000.. Além desse poderiam ter sido utilizado o Tomcat Jakarta, também desenvolvido em Java, o Apache ou o IIS (*Internet Information Server*). Sendo que, no IIS seria necessário a inclusão de um componente que permitiria a ele a interpretação de *Servlets*.

Para a validação do Agente, foi simulado um ambiente de recuperação de informações de um produto da indústria farmacêutica, semelhante ao descrito na 6.9. Para isso, foi criado uma página HTML contento uma lista dos campos que podiam ser recuperados. Nesse caso, os campos são: Descrição, Preço de Compra, Preço de Venda, Estoque e ICMS. O campo chave de pesquisa é o código do produto. Cada campo está relacionado a um Agente, como pode ser visto na Figura 6.3.

Figura 6.3 - Browser com interface de entrada de dados do Agente

O próximo passo foi cada um dos Agentes indicados na Figura 6.3 e seus respectivos campos. Para testar o processo, todos os Agentes são inicializados. O browser é aberto. O código do produto a ser pesquisado é inserido no campo *Código*. Os códigos de produtos foram incluídos aleatoriamente para efeito de teste. As caixas de verificação, que são vistas na Figura 6.3, indicam quais os campos que deseja-se recuperar. As caixas que estiverem marcadas indicam os campos que serão pesquisados.

Depois de feita a pesquisa, o *Servlet* retorna uma página HTML criada dinamicamente que é exibida pelo browser como o resultado da pesquisa.

6.12 Desempenho

Em princípio o que pode-se afirmar é que, o desempenho do sistema é equivalente ao desempenho de um sistema pesquisa na Web do tipo Metabusca, isso

pelo fato de ambos os sistemas acessarem vários servidores para recuperar uma informação.

Outro fator que interfere no desempenho, mas para o qual ainda não existem dados estatísticos, é a estratégia de recuperação de informação a ser utilizada. A escolha de umas das estratégias descritas na seção 4.2, está ligada diretamente ao número de informações a serem recuperadas. Por exemplo, a estratégia de Envio de Formulário é mais eficiente em termos de distribuição de carga de processamento do que a estratégia de Requisição Simples, quando a informação a ser recuperada é um conjunto misto de informações de um único objeto.

Por outro lado, quando deseja-se recuperar informações mistas ou uma única informação de vários objetos, como no caso de uma lista de produtos de uma indústria com todos os seus dados ou o nome dos produtos em promoção em todas as farmácias, a estratégia mais indicada é a Requisição Simples. Como esse tipo de pesquisa gera uma lista que possui muitos dados, o número de bytes trocados entre Agentes utilizando-se a estratégia de Envio de Formulário seria diretamente proporcional ao volume de informações, isso iria interferir diretamente no desempenho do sistema.

De certo modo, pode-se afirmar que a estratégia de Envio de Formulário é eficiente quando o conjunto de informações a serem recuperadas é relativo a poucos objetos, mesmo que esse objeto possua várias informações (um produto e todas as suas características). Ou seja, algo que no mundo real seria o equivalente a preencher o formulário de um único produto ou cliente. Enquanto que a estratégia de Requisição Simples é mais eficiente quando deseja-se recuperar informações de uma vários objetos idênticos.

Um resumo do desempenho das estratégias de recuperação pode ser visto na Tabela 6.11. Nessa tabela estão relacionados os três fatores que podem interferir no desempenho do sistema: o número de objetos a ser recuperado, o número de campos que descreve esse objeto e o número de Agentes que devem ser consultados para recuperar essa informação.

Tabela 6.11 - Relação do melhor desempenho segundo a estratégia escolhida

Estratégia de Requisição	Número de Objetos	Número de Campos	Agentes Consultados
Requisição Simples	n	1	1
Envio de Formulário	1	n	n

Na realidade essa tabela apresenta do limites de desempenho, ou seja, os extremos que representam qual seria a melhor estratégia em qual situação. O valor de *n* indica um número qualquer >= 1.

O tempo de processamento é calculado em função do: tamanho do objeto, número de interações realizadas e da taxa de transferência da rede. O tamanho do objeto compreende o tamanho da requisição, incluindo sua estrutura, somado ao tamanho do e-Form. O e-Form varia de tamanho a cada interação, pois o tamanho de cada campo recuperado é variável e a cada interação são acrescentadas novas

informações. O número de interações depende do tipo de estratégia escolhido, mas refere-se a cada uma das conexão entre dois Agentes. Por exemplo, para cada Requisição Simples são necessárias duas interações, uma para transferir a requisição e outra para receber o resultado. Já em uma requisição por Envio de Formulário é feita apenas uma interação entre dois Agentes de cada vez, portanto, o número de interações irá depender do número de Agentes Consultados. Esses valores referem-se ao volume de bytes na camada de aplicação. Para que se utilize a taxa de transferência da rede, que é medida *bps* (*bits per second)*, e que é uma grandeza pertencente a camada física, é necessário, também, levar-se em conta os cabeçalhos dos protocolos da camada de transporte (TCP), de rede (IP) e de link (ARP). Visto que este é um sistema baseado em Web e que, portanto, opera sobre o modelo de referência TCP/IP

7 Considerações finais

7.1 Infraestrutura Necessária

A primeira consideração a ser feita, diz respeito à infraestrutura necessária às entidades que irão disponibilizar os dados através de Agentes EDI. Com a centralização da responsabilidade de distribuição de dados na entidade geradora da informação, é necessário que essas tenham uma infra-estrutura capaz de atender a demanda de requisições. A capacidade necessária à infra-estrutura é proporcional à necessidade de utilização das informações fornecidas pela entidade. Dessa forma, por exemplo, pequenas empresas poderão participar da rede sem que seja necessário um grande investimento em infra-estrutura. Por outro lado, empresas de grande porte, privadas ou públicas, que possuem informações que são utilizadas em grande escala, deverão implementar uma infra-estrutura capaz de atender a necessidade dos usuários. Esta realidade não difere da realidade de empresas que praticam *e-Commerce* ou *e-Business*.

Um dos principais objetivos do sistema descrito, é possibilitar a inserção de empresas do mercado SOHO (*Small-Office/Home-Office*) na Internet, de maneira que elas possam tanto, utilizar efetivamente os benefícios da Internet, quanto, disponibilizar novos recursos, trocando de forma transparente e simples as informações que são de seu interesse.

7.2 Armazenar Cópias Locais ou Não

Outra consideração é quanto a questão de ser ou não melhor armazenar uma cópia das informações localmente. Os fatores que devem ser avaliados para responder esta questão são:

a) Qual a dinâmica de alteração das informações? Informações estáticas, que não mudam ao longo do tempo, como por exemplo o nome de uma pessoa, podem ser armazenadas localmente, depois de terem sido recuperadas da fonte geradora. Informações dinâmicas, que podem sofrer alteração com o passar do tempo, serão melhor aproveitadas com o armazenamento apenas da referência de identificação da informação; um exemplo seria o endereço de uma pessoa.

b) Qual a freqüência de utilização das informações? Armazenar localmente informações que são acessadas freqüentemente melhora o desempenho do sistema. Por outro lado armazenar localmente informações que raramente são acessadas irá apenas aumentar o TCO de manutenção de base de dados. As informações mais utilizadas por uma entidade são aquelas geradas por ela mesmo. Porém estas informações necessitam de Informações de Apoio que também podem ser chamadas de Informações Auxiliares. Por exemplo, para uma Distribuidora, as informações que mais são processadas são as informações fluxo de transações entre a entidade e seus colaboradores. Informações como os dados do produto que é comprado pela distribuidora são acessados com menos freqüência.

Os dados de apoio são *ready-only*. Isto significa que o intercâmbio de informações entre os Agentes não precisa prever condições de concorrência que são necessárias em Sistemas Distribuídos convencionais.

7.3 Privacidade

A privacidade é outro fator importante a ser discutido. Ampliando a utilização do sistema para a distribuição de informações pessoais, como no exemplo de entidades públicas que são responsáveis pelos dados de identificação do indivíduo, nos deparamos de imediato com a questão da privacidade. Porém esta primeira impressão é ilusória mas é natural o seu questionamento. O sistema prevê um protocolo de identificação dos tipos de Agentes que podem trocar informações entre si e qual o tipo de informações que pode ser trocada. As normas que definem esta classificação são as mesmas utilizadas no mundo real. A principal regra que garante a privacidade é a de que o agente tem que permitir o acesso à informação, ou seja, o agente decide para quem ele pode ceder informações.

A assinatura necessária para identificar um agente, fornecida por uma entidade responsável pelo registro dos Agentes, garante que este agente pertence ao grupo que ele afirma pertencer. O papel da entidade que efetua o registro do agente é semelhante ao exercido pela entidade responsável pelo registro de domínios da Internet. A obrigatoriedade de identificação do agente e a capacidade de selecionar a quem ceder informações garante a privacidade no sistema.

7.4 Pesquisas futuras

7.4.1 XML

Os *Tags* utilizados para formatar os documentos manipulados pelo Agente, podem constituir uma linguagem XML. Nenhuma linguagem ainda está definida para a utilização por parte dos Agentes. Uma possibilidade é a utilização da linguagem XML conhecida como *Open Trade Protocol,* OTP, que está sendo elaborada para ser utilizada por empresas que desejam executar transações de Comércio Eletrônico [MCG 99].

7.4.2 Pesquisas de desempenho

Pesquisas para determinar o desempenho do sistema, comparando as diferentes estratégias de interação entre os Agentes, podem dar melhores subsídios para definir qual é a melhor opção. No entanto é importante planejar o sistema, e os Agentes, para que os dois tipos de interação entre Agentes, *Envio de Formulário* e *Requisição Simples*, possam interagir entre si de forma transparente ao usuário. Já a implementação de Agentes Móveis para esta finalidade ainda fica inviável, até que novas tecnologias sejam propostas.

7.4.3 KQML

O agente por enquanto utiliza um protocolo proprietário, porém ele pode ser aperfeiçoado para utilizar a KQML como linguagem de comunicação entre os Agentes. Essa opção tornará o Agente mais genérico, porém não interfere na sua utilização.

7.4.4 Recuperação de documentos não estruturados

Toda a descrição do trabalho foi orientada a recuperação de campos de dados estruturados. Porém, como não existe limite quanto ao tamanho e tipo de arquivo a ser transferido, a não ser os limites físicos, o Agente pode recuperar dados não estruturados, como por exemplo, arquivos de som e vídeo ou documentos HTML. Esse recurso pode ser utilizado para busca de conhecimento na rede. Nesse caso o conhecimento é processado pelo Agente que está disponibilizando as informações. O Agente que faz a solicitação faz apenas uma pergunta, o que anteriormente era constituído pelos campos a serem recuperados, e recebe o resultado que é gerado pelo Agente que está disponibilizando a informação. Essa alternativa faz com que os atuais agentes que recuperam dados não estruturados na Web, sejam apenas um caso específico do Agente EDI. Para que esse tipo de funcionalidade seja implementada no Agente EDI basta tratar o tipo de retorno recebido, por exemplo, utilizando-se os cabeçalhos MIME. É claro que é necessário uma mudança de ponto de vista da resolução do problema, mas essa lógica fica mais clara quando o funcionamento do Agente é compreendido.

8 Conclusões

A utilização da infra-estrutura oferecida pela Internet para o intercâmbio de dados estruturados amplia a utilidade da rede. Este trabalho mostra a arquitetura de um Agente EDI que torna essa possibilidade de utilização uma opção viável. Áreas emergentes, como por exemplo o comércio eletrônico, contarão com mais uma ferramenta de apoio. O trabalho apresenta também o modelo do Sistema de Informações baseado em Web que utilizará estes Agentes, demonstrando com isto a utilidade prática dos agentes.

O Agente pode ser utilizado tanto por pessoas físicas quanto por pessoas jurídicas. Isto lhe permite ser utilizado em diferentes cenários de Intercâmbio de dados estruturados. Essa característica atribui maleabilidade ao sistema e aos Agentes.

A rede de Agentes não possui controle centralizado e é dinâmica. Essa característica caracteriza o sistema como um Sistema de Dados Distribuídos que não é de propriedade de uma única entidade. Mesmo assim, a privacidade das informações é mantida, e o desenvolvimento do sistema utiliza tecnologias existentes.

A dupla capacidade dos Agentes EDI, de recuperar e disponibilizar dados, torna a implementação de um Sistema Informações Web mais simples, além de permitir mais flexibilidade ao usuário.Com este tipo de Agente as empresas diminuirão seu TCO por não precisarem armazenar Informações Auxiliares, sempre que for necessário, é possível recuperar as informações diretamente da fonte geradora. Esta estratégia resolve também o problema da falta de consistência das informações.

Apesar de parecer complexo, o modelo de Sistema de Informações baseado em Web apresentado é bastante simples. A complexidade do sistema surge naturalmente pela interação básica dos Agentes EDI, porém, sem necessariamente tornar o sistema complicado.

E por fim, quanto a implementação do Agente, pode-se afirmar que esta é viável através da utilização de uma linguagem de domínio público e gratuita. Os demais componentes que podem ser necessários a implementação do sistema, como por exemplo, o Servidor HTTP, também pode ser obtido gratuitamente na Internet. As características de: baixo custo de desenvolvimento e operacional, não ser necessária mão de obra extremamente especializada para implementar o sistema e fácil acesso as ferramentas e componentes de desenvolvimento, certamente serão fatores atrativos às empresas, tanto às pequenas e médias, quanto as grandes.

Bibliografia

[BOR 98] BOGER, Marko. **Migrating Objects in Eletronic Commerce Applications** / Marko Boger – Trends in Distribuited Systems for Eletronic Commerce – International IFIP/GI Working Conference, TREC'98, Hamburg, Germany, June 1998.

[CER 97] CERVEIRA, Adriana Justin. **Um estudo de Agentes móveis na Internet**: sistemas linguagens e aplicações. 1997. 51f. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[CHE 96] CHEONG, Fah-Chun. **Internet Agents: Spiders, Wanderers, Brokers, and Bots**. Indianapolis: New Riders, 1996.

[COH 97] COHEN, Philip R.; Levesque, Hector J. **Communicative Actions for Artificial Agents**. In: Software Agents, The MIT Press, Cambridge, Massachusetts, 1997, pp. 419-436.

[DAM 95] D'AMICO, Carmem Barbosa de. et al. **Inteligência Artificial:** Uma abordagem de Agentes. 1995. 87f. Relatório de Pesquisa – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[FIN 97] FININ, Tim; Labrou, Yannis; Mayfield, James. **KQML as an Agent Communication Language**. In: Software Agents, The MIT Press, Cambridge, Massachusetts, 1997, p. 291-315.

[GEN 97] GENESERETH, Michael R. **An Agent-Based Framework for Interoperability**. In: Software Agents, The MIT Press, Cambridge, Massachusetts, 1997, pp. 317-345

[GRA 98] GRAHAM, Isan S. **HTML**: a referência completa para HTML 3.2 e extensões HTML. Rio de Janeiro: Campus, 1998.

[JAM 99] JAMSA, Kris; SULEIMAN, Lalani; WEAKLEY, Steve **Programando para World Wide Web.** São Paulo: Makron Books do Brasil, 1999.

[KNA 98] KNAPIK, Michael; Johnson, Jay. **Developing Intelligent Agents for Distributed Systems**. McGraw-Hill, 1998.

[KNO 97] KNOBLOCK, Craig A.; Ambite, José Luis. **Agents for Information Gathering**. In: Software Agents, The MIT Press, Cambridge, Massachusetts, 1997, p. 347-373.

[MAE 97] MAES, Pattie. **Agent that Reduce Work and Information Overload**. In: BRADSHAW, Jeffrey M. (Ed.) Software Agents. Menlo Park, CA: AAAI Press, 1997. p. 145-164.

[MAE 9?] MAES, Pattie **Modeling Adaptative Autonomous Agents**. Cambridge: MIT Media-Laboratory, [199?].

[MAL 97] MALONE, Thomas W.; Lai, Kum-Yew; and Grant, Kenneth R. **Agents for Information Sharing and Coordination: A History and Some Reflections**. In: Software Agents, The MIT Press, Cambridge, Massachusetts, 1997. p. 109-143.

[MCG 99] MCGRANT, Sean. **XML** : Aplicações práticas. Rio de Janeiro: Campus, 1999.

[MUR 98] MURCH, Richard; Johnson, Tony. **Intelligent software agents**. Prentice-Hall, Inc, 1998

[RIE 97] RIECKEN, Doug. **The M System**. In: Software Agents, The MIT Press, Cambridge, Massachusetts, 1997, p. 247-267.

[SEA 90] SEACORD, Robert C. **User Interface Management Systems and applications Portability.** Computer, New York, v.23, n.10, p.73-75, Oct. 1990.

[SIL 01] SILVA, Henrique Oliveira da, **Agentes EDI como ferramenta de Apoio a Sistemas de Informações baseado em Web**. Revista Iberoamericana de Inteligencia Artificial, Espanha, n.13, p.100-107, Ago 2001.

[STR 98] STRABER, Markus; Rothermel, Kurt; Maihöfer, Cristian. **Providing Reliable Agents for Eletronic Commerce** / Markus Straber; Kurt Rothermel; Cristian Maihöfer – Trends in Distribuited Systems for Eletronic Commerce – International IFIP/GI Working Conference, TREC'98, Hamburg, Germany, June 1998.

[TAN 97] TANENBAUM, Andew S. **Redes de Computadores** : tradução [ds 3. Ed. original] Insight Serviços de Informática. Rio de Janeiro: Campus, 1997.

[WHI 97] WHITE, James E. **Mobile Agents**. In: Software Agents, The MIT Press, Cambridge, Massachusetts, 1997, p. 437-471.

[YIX 98] YI, X.; Wang, X. F.; Lam, K. Y. A **Secure Intelligent Trade Agent System** / X. Yi; X. F. Wang; K. Y. Lam – Trends in Distribuited Systems for Eletronic Commerce – International IFIP/GI Working Conference, TREC'98, Hamburg, Germany, June 1998.

[ZAP 98] ZAPF, Michael; Müller, Helge; Geihs, Kurt. **Security Requirements for Mobile Agents in Eletronic Markets** / Michael Zapf; Helge Müller; Kurt Geihs – Trends in Distribuited Systems for Eletronic Commerce – International IFIP/GI Working Conference, TREC'98, Hamburg, Germany, June 1998.