Ayrton Maia Neto

COMMERCEPIPE

UM FRAMEWORK PARA CRIAÇÃO DE CANAIS COMERCIAIS CONSUMER TO BUSINESS NA INTERNET

Dissertação apresentada ao Departamento de Informática da Puc-Rio como parte dos requisistos para a obtenção do título de Mestre em Ciências em Informática

Orientador: Carlos José Pereira de Lucena

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 16 de junho de 2000

Este trabalho é dedicado a

Francisca Aquino Beleza,
Sheila Aquino Beleza,
Roberta Aquino Beleza Villaça Maia,
Airton José Villaça Maia e
Renata Farah.

Agradecimentos

A execução e finalização deste trabalho de pesquisa de dois anos de duração não se dariam sem a participação de algumas pessoas para as quais gostaria de mostrar aqui meu agradecimento sincero.

Agradeço ao Orientador, Professor e Amigo Carlos J. P. de Lucena por diversas vezes clarear dúvidas em minha mente e minimizar os erros durante todo o trabalho de pesquisa. As conversas de amigo e os inúmeros “bom momentos” foram especiais e caracterizam seu estilo *leve* e *sério* de orientar.

Agradeço aos Professores Bruno Feijó e Ruy Milidiú por participarem de minha banca e por contribuírem com análises e críticas construtivas para que este trabalho de pesquisa ficasse ainda mais completo.

Agradeço a Vera A. S. Menezes, *anjo*, a quem devo muito por toda a ajuda prestada em muitos momentos durante estes dois anos. *Tenho todo o tempo do mundo para quitar a minha conta com você, minha amiga!*

Agradeço a Marcus Felipe, Sérgio Crespo, Viviane Torres, Clécio Guaranys e Bruno Mello pela amizade, pelas boas e produtivas conversas no LES e nas reuniões semanais, pela contribuição em diversos pontos da pesquisa e principalmente pelos momentos de descontração, combustível para inspiração e criatividade.

Agradeço a todos do LES e do TecComm pelo convívio saudável e troca de conhecimento durante o último ano de mestrado.

Agradeço a CAPES, ao PRONEX e ao Departamento de Informática pelo apoio financeiro que possibilitou a realização deste trabalho.

Agradeço a minha mãe Sheila Aquino Beleza, a meu pai Airton J. V. Maia, a minha irmã Roberta Maia e meus familiares pelo que considero *o grande pilar da minha vida*, o que certamente me possibilitou ser capaz de estar aqui escrevendo estes agradecimentos neste momento.

Agradeço a Renata Farah, pelos momentos finais deste trabalho. Seu carinho contribuiu para que o trabalho de pesquisa fosse concluído da melhor forma possível.

Agradeço a Deus por todos os momentos.

Abstract

Based on the study of different electronic commerce software applications and principally inspired by the research work and software evolution carried out in the Vmarket framework, this dissertation presents the Commercepipe Object Oriented Framework as an alternative for the instantiation of Consumer to Business Virtual Markets on the Internet. Using Software Agents Technology combined with the WAP (Wireless Application Protocol) as a wireless interface to access Internet services, the framework proposes a new approach for C2B markets, bringing a new vision of how sellers and buyers can interact to realise commercial transactions through the Internet.

Resumo

Baseado no estudo de diferentes sistemas de comércio eletrônico e, principalmente, inspirado no trabalho de pesquisa e evolução de software realizado no *framework* Vmarket, esta dissertação apresenta o framework Commercepipe como solução tecnológica alternativa para a instanciação de mercados virtuais C2B na Internet. Utilizando a tecnologia de Agentes de Software combinada ao protocolo WAP (*Wireless Application Protocol*) para acesso a serviços Internet, o framework propõe uma nova abordagem para mercados C2B, trazendo uma nova visão de como vendedores e compradores podem interagir para realizar transações comerciais pela Internet.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com a explosão de aplicações para o comércio eletrônico na *World Wide Web*, a presença e a participação, neste ambiente, tornaram-se essenciais para as empresas. Devido às diversas vantagens oferecidas pela *Internet* na realização de transações comerciais, na aproximação de compradores e vendedores, na diminuição de custos, na personalização de serviços e na automação de tarefas, o desenvolvimento de novas tecnologias que permitam a criação de sistemas inovadores para o desenvolvimento do comércio eletrônico torna-se importante neste novo cenário.

O comércio eletrônico encontra-se divido em quatro principais segmentos de mercado: B2B (transações comerciais entre empresas), B2C (transações comerciais entre empresas e consumidores), C2B (consumidores expõem suas intenções de consumo) e C2C (transações comerciais entre pessoas). O quadro, abaixo, apresenta alguns exemplos dos quatro segmentos de aplicações de comércio eletrônico [The Economist, 2000].

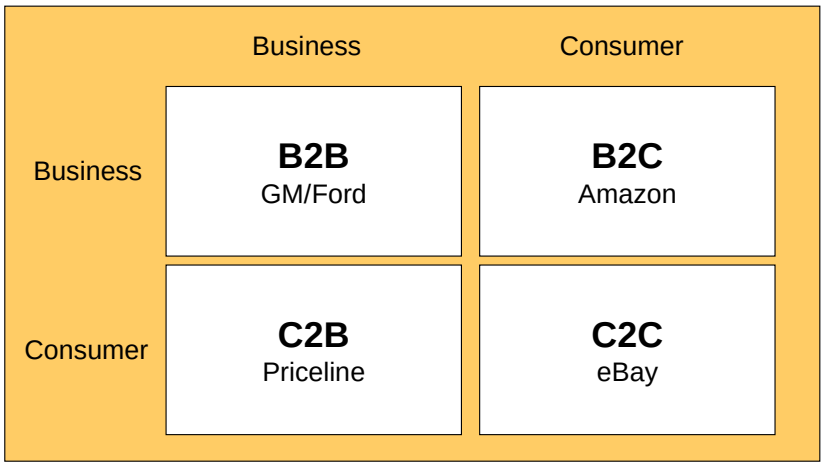


Figura 1 Principais segmentos de mercado

O foco principal deste trabalho de pesquisa é experimentar a utilização da tecnologia de Agentes de Software [Maes, 1994][Maes, 1994] e da tecnologia WAP (Wireless Application Protocol) [WAP, 1999b] em aplicações de comércio eletrônico através da criação de um framework orientado a objetos. O framework desenvolvido utilizará a Tecnologia de Agentes de Software para promover principalmente a automação de etapas do processo de negociação e a Tecnologia WAP para permitir a interação com as aplicações não só através de computadores pessoais mas também através de celulares e PDAs (Personal Digital Assistants) compatíveis com o protocolo WAP. O framework proposto neste trabalho de pesquisa possibilitará futuramente a experimentação com o uso da tecnologia de agentes de software e da tecnologia de acesso a Internet por dispositivos móveis através da criação de diferentes aplicações para o mercado C2B.

1.1 Motivação

Motivando-se na demanda atual das aplicações para o comércio eletrônico, este trabalho de pesquisa analisa a utilização de duas principais tecnologias em aplicações de comércio eletrônico.

- *Tecnologia de agentes de software* – Permite, principalmente, a automação de tarefas [Bradshaw, 1997][Maes, 1994]. Quando aplicado ao domínio de comércio eletrônico, os agentes de *software* podem automatizar etapas do processo de negociação entre compradores e vendedores, desde a busca por produtos específicos, até a própria negociação [Guttman, 1998c];

- *Tecnologia wireless para acesso a Internet* – O acesso a serviços de comércio eletrônico por dispositivos móveis mostra-se cada vez mais viável e já acontece em escala em alguns países do mundo, principalmente na Europa, onde sua popularidade é maior uma vez que se utiliza a mesma tecnologia em toda a

Comunidade Européia (GSM é a tecnologia escolhida para a telefonia móvel na maioria dos países europeus) [Durlacher, 2000].

Com a possibilidade de criação de aplicações para o comércio eletrônico utilizando ambas as tecnologias, este trabalho propõe a modelagem e o desenvolvimento de um *framework* [Fontoura, 1999] orientado a objetos, no qual compradores e vendedores podem realizar transações comerciais por meio da utilização de agentes de *software* como seus mediadores em todo o processo de negociação em um mercado virtual centralizado. A possibilidade de interagir com o sistema por meio de dispositivos móveis é incorporada ao *framework*, permitindo que vendedores e compradores interajam com o sistema, em qualquer lugar e a qualquer momento, para acompanhar o andamento de negociações, realizar propostas e contrapropostas entre outras possibilidades.

A instanciação de uma primeira aplicação do *framework*, parte do trabalho de pesquisa, é realizada validando a usabilidade do *software*. Esta aplicação representa um mercado virtual de carros no qual consumidores em um *Website* expõem propostas de compra de carros, e vendedores, por celulares, são informados sobre novas intenções de consumo, podendo realizar, por conseguinte, contrapropostas imediatamente, independente de sua localização.

1.2 Organização do documento

Este documento está organizado em seis capítulos, sendo o primeiro o de introdução à dissertação. O segundo apresenta a tecnologia de agentes de *software* e seus conceitos básicos. Neste capítulo, apresentam-se a importância da utilização de agentes em aplicações de comércio eletrônico, algumas arquiteturas de sistemas para este domínio de aplicações e a segmentação das aplicações de comércio eletrônico com a apresentação de exemplos para cada segmento. Nestes exemplos, são mostradas as possíveis vantagens da

utilização da tecnologia de agentes e as principais mudanças que ocorreriam com a sua utilização.

No capítulo 3, apresenta-se a tecnologia WAP (*Wireless Application Protocol*) [WAP, 1999a] para permitir o acesso móvel à *Internet*. A arquitetura WAP é apresentada identificando-se as principais partes e mostrando-se o funcionamento de todo o protocolo. As características particulares desta nova forma de acesso também são apresentadas, mostrando as principais barreiras para o desenvolvimento de aplicações WAP. O conceito de m-Commerce é introduzido, mostrando como será possível realizar transações comerciais, fazendo-se uso de dispositivos móveis, além de mostrar algumas aplicações viáveis.

A modelagem do *framework* CommercePipe é apresentada no capítulo 4, no qual uma visão geral da arquitetura do sistema é, inicialmente, apresentada. O capítulo organiza-se em seções, e cada subsistema do *framework* é detalhado. As tecnologias utilizadas no desenvolvimento e o critério de escolha são apresentados no final do capítulo.

O processo de instanciação do *framework* é descrito no capítulo 5. Os pontos de flexibilização [Fontoura, 1999] [Fayad, 1999a] do *framework* são o foco deste capítulo, e o procedimento de instanciação de todos eles são apresentados e exemplificados utilizando-se a aplicação AutoPipe criada utilizando-se o *framework*. Finalmente, no capítulo 6, estão as conclusões do trabalho. Já no capítulo 7, encontra-se a proposta de trabalhos futuros para o *framework* CommercePipe.

Capítulo 2

Agentes de *software* e Mercados Virtuais

O comércio eletrônico já é uma das mais importantes aplicações da *Internet*, potencializando até mudanças na forma como empresas devem se estruturar para esta nova realidade e criando um mercado com características marcantes que não podem ser desconsideradas. Entre as principais características deste novo mercado estão:

- *Velocidade de adaptação a novas tecnologias* – Esta é uma característica marcante e sempre presente neste novo mercado. Empresas precisam ser velozes na adaptação a tecnologias emergentes, de forma a acompanhar a evolução natural do mercado;

- *Personalização* – Esta é uma característica importante no que se refere ao relacionamento com o usuário final. As empresas que atuarem neste novo mercado devem considerar o fato de que a utilização dos serviços é fortemente individualizada, uma vez que existe um único usuário a frente de cada computador e, conseqüentemente, todos os serviços devem ser customizados e percebidos como se houvessem sido feitos sobre medida para cada um;

- *Globalização* – Apesar de não ser regra para as empresas atuarem em um mercado global, o mercado quase que conduz a este posicionamento, já que não existem mais fronteiras geográficas para a venda de produtos e serviços. Assim, as empresas devem estar preparadas para uma eventual atuação em um mercado globalizado.

Percebendo estas características, este capítulo apresenta a tecnologia de agentes de *software*, também chamados “empregados digitais”, como uma tecnologia promissora

para aplicações de comércio eletrônico e mostra como esta pode agregar valor nas aplicações de comércio eletrônico [Guttman, 1998c], sendo aplicável em diversas etapas envolvidas no processo de comercialização, principalmente com relação à personalização e automação do processo de negociação. A utilização de agentes de *software* no domínio de aplicações para o comércio eletrônico envolve a solução de diversos problemas, sendo que os principais serão apresentados neste capítulo.

Posteriormente, serão apresentadas as principais aplicações hoje existentes no mercado virtual, enquadradas em quatro principais segmentos de mercado. As aplicações são detalhadas apresentando-se suas principais características como, por exemplo, tecnologias utilizadas e descrição do serviço oferecido.

2.1 Agentes de Software

O conceito de agenciamento eletrônico, inicialmente abordado com a utilização de *hardwares* específicos como robôs e andróides, migrou, posteriormente, para o paradigma de *software*, permitindo, de forma geral, que se delegasse e automatizasse a execução de tarefas de acordo com as necessidades do usuário [Bradshaw, 1997]. Atualmente, a pesquisa na área de agentes de *software* divide-se em duas linha principais. A primeira caracteriza-se pela pesquisa de agentes de *software* com habilidades na área de Inteligência Artificial. Estes agentes, muitas vezes denominados “agentes inteligentes” [Bradshaw, 1997], são capazes de realizar inferências, aprender com o comportamento do usuário e, baseados nestes dados, alterar sua linha de execução, caracterizando, assim, um comportamento autônomo e adaptativo. Em contraste, a segunda linha de pesquisa enfatiza agentes de *software* capazes de realizar tarefas bem definidas, de forma automática, sem a utilização de conceitos da área de Inteligência Artificial como, por exemplo, *softwares* automáticos (robôs) de busca por conteúdo específico na *Web* e agentes administradores de caixas postais, sentinelas de pontos de rede entre outros.

Apesar de ser um conceito já razoavelmente difundido[1] [Bradshaw, 1997] e empregado em diversas áreas de pesquisa da Ciência da Computação, a definição de agentes de *software* ainda não é clara e varia, dependendo da área na qual se aplica. A proliferação de agentes de *software* em áreas como Interface com Usuário, Banco de Dados, Redes, Computação Gráfica, Engenharia de Software provocou a utilização deliberada do termo "agente". Alguns programas são denominados "agentes" simplesmente por terem a capacidade de executar remotamente em outras máquinas, por informarem ao usuário quando o sistema de arquivos está com problemas, por informarem problemas de comunicação em nós de uma LAN (*Local Area Network*) ou por filtrar informações desejadas na *Web* aliviando a sobrecarga de informação [Maes, 1994].

Dependendo do escopo de atuação ou da tarefa a ser realizada, agentes de *software* podem possuir um subconjunto das características abaixo:

1. *Reatividade* – Habilidade de perceber situações diferentes no ambiente computacional e de reagir de forma a modificá-las. Esta característica permite que agentes reajam a variações do ambiente eletrônico em que atuam e não somente a eventos externos ao sistema;

2. *Autonomia* – Habilidade de completar tarefas sem a necessidade de intervenção do usuário. É uma característica importante uma vez que permite uma menor intervenção do usuário com o sistema, automatizando a execução de tarefas;

3. *Comunicação* – Habilidade de se comunicar com o usuário ou com outros agentes. A comunicação com o usuário pode ser feita por meio de linguagens de alto nível, mais próximas da linguagem natural, enquanto que a comunicação entre agentes de *software* pode ser realizada de diversas formas, com a utilização

[1] A idéia de agentes de *software* foi, inicialmente, criada por John McCarthy na década de 1950 e, alguns anos depois, o conceito foi refinado por Oliver G. Selfridge no Massachusetts Institute of Technology. Naquela ocasião, a percepção era a de que um sistema no qual houvesse um objetivo especificado, o *software*, em situações específicas, poderia detalhar as operações executadas pelo programa, permitindo que o usuario indicasse um caminho de execução.

de linguagens específicas como KQML, TeleScript [MayField, 1997] [Bradshaw, 1997];

4. *Capacidade de Inferir* – Habilidade de adaptar a execução de tarefas com base em conhecimento previamente adquirido, não somente por informação passada pelo usuário, mas também por aprendizado próprio;

5. *Continuidade Temporal* – Capacidade de persistir seu estado de execução por longos períodos de tempo, permitindo que sua execução seja interrompida por algum motivo e prosseguida posteriormente;

6. *Personalidade* – Habilidade de demonstrar características de personalidade como, por exemplo, emoções, utilizando personagens eletrônicos por interfaces gráficas;

7. *Personalização* – É a possibilidade de seu comportamento ser customizado considerando-se as preferências de seu usuário. Muitas vezes, a personalização pode ocorrer não só no momento inicial de sua execução, mas também ao longo de todo o período de atividade;

8. *Adaptabilidade* – Habilidade de aprender e evoluir com o ganho de experiência;

9. *Mobilidade* – Habilidade de migrar de uma máquina para outra, mesmo que sejam plataformas distintas. Agentes com estas características podem viabilizar o balanceamento de carga de máquinas em rede, ou viabilizar a coleta de alguma informação em um ambiente distribuído [Voyager, 1999] [Bradshaw, 1997];

10. *Cooperação e Interação* – Agentes de *software* podem trabalhar em conjunto cooperando para atingir seus objetivos. Agentes de compra e venda podem interagir em um ambiente multiagente, com o objetivo de chegar a uma negociação interessante para ambas as partes. Agentes de busca podem cooperar trocando informações úteis para seus usuários, coletadas na *Internet*, otimizando o trabalho de busca.

A figura 2, mostra o escopo das duas linhas de pesquisa de agentes comparado a sistemas especialistas utilizando um gráfico de dois eixos, um caracterizando agência e outro inteligência.

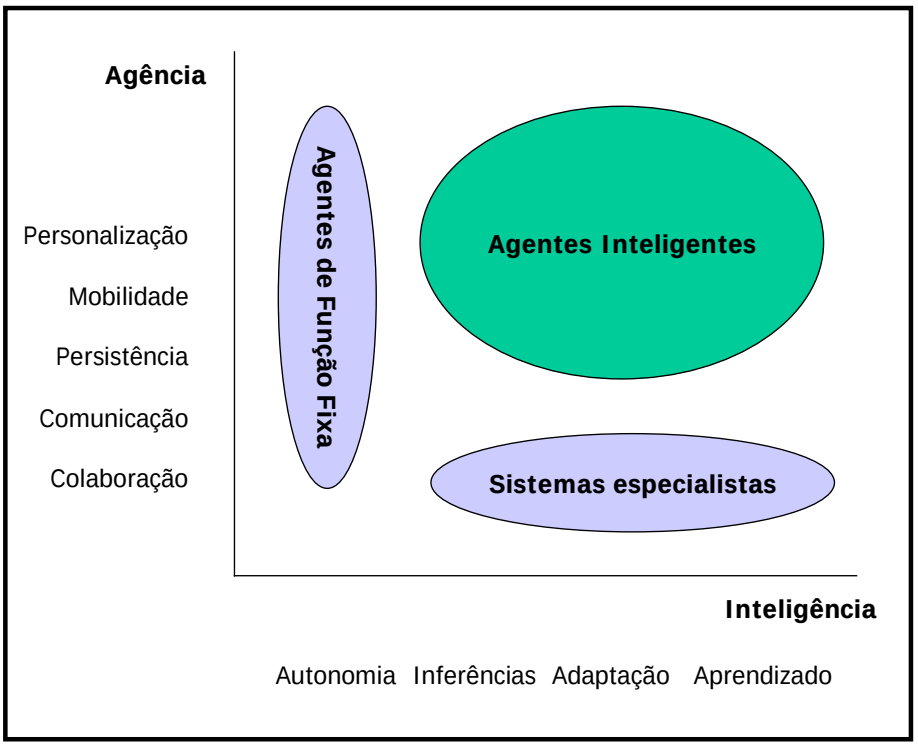


Figura 2 Posicionamento das linhas de pesquisa de agentes de software

Na figura 2, Agentes de Função Fixa representam a linha de pesquisa que não aplica características de Inteligência Artificial. Nota-se que o desenvolvimento de agentes inteligentes envolve, além dos aspectos de agência, aspectos relacionados, principalmente, à capacidade de aprendizado e adaptação utilizados também por sistemas especialistas em geral.

No escopo deste trabalho de pesquisa, agentes de *software* são definidos como entidades automatizadoras de processos capazes de realizar tarefas específicas e bem definidas de forma autônoma, ou seja, sem a necessidade de intervenção do usuário. Com um certo grau de personalização e pela customização da tarefa a ser realizada, tais agentes devem

ser capazes de perceber e reagir a variações do ambiente, adaptando-se para cumprir o objetivo. Dentre as características listadas acima, *Reatividade*, *Autonomia* e *Cooperação* foram consideradas como essenciais para a modelagem dos agentes de *software* neste trabalho.

Com base nesta visão geral do conceito de agentes de *software*, esta pesquisa tem como uma de suas principais motivações investigar a utilização de agentes no domínio de aplicações de comércio eletrônico.

2.1.1 Agentes de *Software* e o Comércio Eletrônico

Com a capacidade de automatizar tarefas diversas, a possibilidade de automatizar etapas no processo de negociação entre compradores e vendedores, faz com que a tecnologia de agentes de *software* seja propícia a aplicações de comércio eletrônico [Moukas, 1998]. Apesar de possuírem muitas diferenças, os diversos mecanismos de negociação utilizados em aplicações de comércio eletrônico possuem, um subconjunto das etapas identificadas como fundamentais no Modelo de Comportamento de Compra do Consumidor (*Consumer Buying Behavior - CBB*) [Guttman, 1998b] [Guttman, 1998a]. Diversos modelos tentam capturar o comportamento de compra do consumidor como o Nicosia, Howard-Sheth, Engel-Blackwell, Bettman e Andreasen [Guttman, 1998b] e as etapas comuns em todos eles são:

1. *Identificação da necessidade* – Este estágio caracteriza-se pela identificação da necessidade de consumo do usuário comprador. Alguns sistemas, nesta etapa, disponibilizam informação sobre produtos para estimular o consumo;

2. *Procura por produto* – Nesta etapa, o sistema provê informações diversas, ajudando o consumidor a avaliar a melhor escolha com relação à variedade de produtos disponíveis para sua necessidade de consumo. Como resultado desta etapa, o consumidor define o subconjunto de produtos que satisfazem sua necessidade;

3. *Procura por vendedor* – Utilizando o conjunto de produtos definidos como satisfatório pelo consumidor, na etapa anterior, o sistema deve auxiliar a busca pelo vendedor mais adequado utilizando informações específicas dos vendedores. Baseando-se em critérios definidos pelo consumidor como, por exemplo, prazo de entrega, preço, reputação e forma de pagamento, o sistema deve ajudar a identificar o conjunto de vendedores aptos a vender o produto desejado;

4. *Negociação* – Esta etapa determina os termos da transação comercial. O processo de negociação varia em duração e complexidade, dependendo do mercado em questão. A negociação pode ser multidimensional [Kumar, 1999a] [Ripper, 1999], ou seja, pode levar em consideração não somente o preço, mas também outras dimensões de importância para o consumidor, como prazo de entrega e forma de pagamento, por exemplo. A consideração de outras dimensões, que não o preço, influencia diretamente o resultado final da negociação, dando maior poder de negociação para consumidores e vendedores;

5. *Compra e Entrega* – Nesta etapa, o consumidor realiza, efetivamente, o pagamento pelo produto adquirido. A entrega deste produto deve ser realizada sinalizando o término de uma negociação bem sucedida;

6. *Avaliação* – Sendo esta a última etapa, a avaliação consiste no retorno de informação do consumidor ou vendedor avaliando todo o processo de negociação. Com esta avaliação, é possível classificar vendedores e consumidores por reputação e qualidade de serviço, o que pode ser utilizado também como uma nova dimensão de negociação.

Desta forma, dentre as etapas do CBB, a tecnologia de agentes de *software* pode ser utilizada em todas, sem exceção, permitindo a automação de etapas importantes do processo de negociação. Para as aplicações de comércio eletrônico, as principais motivações para automatizar etapas do CBB são:

- *Diminuição da sobrecarga de informação* – A disponibilidade de muita informação (comum na *Internet*) prejudica o processo de decisão. A utilização de agentes permite a criação de um filtro automático de informação, permitindo que o usuário final só analise informações relevantes [Maes, 1994]. As etapas de Procura por Produto e Procura por Vendedor são automaticamente otimizadas com a utilização de agentes;

- *Diminuição da necessidade de interação com o sistema* – Principalmente com a possibilidade de automatizar a etapa de negociação, a tecnologia permite que os usuários tenham, cada vez menos, a necessidade de interagir com o sistema, trazendo comodidade para os usuário. Eventuais comunicações podem ser feitas pelos agentes do sistema informando a situação de suas ou de qualquer outra informação de relevância para o usuário;

- *Redução de Custos* – O custo por transação e o tempo gasto para a realização de transações comerciais podem ser drasticamente reduzidos com a utilização de agentes;

- *Comodidade para o usuário* – A possibilidade de delegar tarefas para "empregados digitais" permite a criação de aplicações com um nível maior de conforto para o usuário final.

Além das vantagens citadas na utilização desta tecnologia para aplicações de comércio eletrônico, existem aplicações, com apelo ainda maior, para a utilização de agentes de *software*, como aplicações desenhadas para situações nas quais o usuário não tem grande capacidade de interação com o sistema, seja por falta de recursos tecnológicos ou por falta de tempo disponível.

Aplicações desenhadas para o usuário em movimento são um exemplo claro no qual a tecnologia de agentes de *software* agrega muito valor. Com a possibilidade de se ter acesso à Internet por meio de dispositivos móveis como celulares e PDAs (*Personal*

Diaigtal Assistants), as aplicações desenhadas para esta nova realidade enfrentam alguns obstáculos e devem ser adequadas a duas principais características do acesso móvel atual:

1. *Escassez de recursos do dispositivo* – O acesso móvel caracteriza-se, hoje, pela falta de recursos do dispositivo de acesso [Nokia, 1999c]. Celulares e PDAs são dispositivos com limitação de memória, pouca capacidade computacional, interface de entrada de dados e interface visual limitadas. Este conjunto de características define novos pré-requisitos para o desenvolvimento de aplicações para este novo mundo. A utilização de agentes de *software* permite criar aplicações ideais, nas quais a redução de interação com o sistema é essencial para aplicações desta natureza;

2. *Tempo disponível de interação reduzido* – Por estar em movimento e não mais a frente de um computador pessoal, muitas vezes, o tempo disponível para interagir com o sistema é limitado. A utilização de agentes de *software* possibilita a delegação de tarefas a serem executadas remotamente, sem a necessidade de interação do usuário.

Desta forma, a tecnologia de agentes de *software* pode ser aplicada agregando valor às diversas etapas envolvidas em aplicações de comércio eletrônico. O apelo para a utilização de agentes aumenta, consideravelmente, com relação às aplicações especificamente desenhadas para o usuário em movimento, sendo uma tecnologia que ajuda a resolver os problemas básicos das aplicações móveis. O capítulo seguinte apresentará os aspectos principais do acesso móvel à *Internet*.

2.1.2 Arquiteturas de Agentes de *Software* para o Comércio Eletrônico

Existem muitas arquiteturas desenvolvidas para utilização de agentes de *software* em aplicações de comércio eletrônico. As principais arquiteturas existentes possuem, em comum, três conceitos principais: O Mercado Virtual, Usuários e Agentes.

- *Mercado Virtual* – Representa o ambiente eletrônico no qual compradores e vendedores têm a possibilidade de se aproximarem para a realização de transações comerciais. Deve ser um ambiente seguro e semanticamente centralizado (apesar de poder ser distribuído fisicamente em máquinas distintas para balanceamento de carga) [Ripper, 1999];

- *Usuários* – São os estimuladores do sistema, responsáveis pela criação de intenções de compra e venda. A partir dos usuários do sistema, que podem ser consumidores e vendedores, dependendo do segmento de atuação (C2C, B2C, C2B, B2B), agentes de *software* são criados e personalizados para a realização de tarefas específicas de compra ou venda, cujos objetivos deverão ser atingidos pela interação multiagente no mercado virtual;

- *Agentes* – Responsáveis, muitas vezes, pela realização da transação eletrônica, os agentes de *software* são os mediadores dos usuários no mercado virtual em todas as etapas do processo de negociação.

Quando utilizados em arquiteturas multiagentes, os agentes de *software* devem ser capazes de interagir eletronicamente. Esta interação é implementada, freqüentemente, por linguagens desenvolvidas especificamente para este fim como a linguagem KQML [MayField, 1997]. Desenvolvida pelo KSE (Knowledge Sharing Effort) [MayField, 1997], a linguagem KQML promove a comunicação entre agentes por um mecanismo de troca de mensagens com formato específico. A linguagem baseia-se no uso de ações que definem a intenção básica da mensagem enviada. A tabela 1, abaixo, mostra as ações definidas pela linguagem.

CATEGORIA	AÇÕES
BASIC QUERY	Evaluate, ask-if, ask-one, ask-all
MULTI-RESPONSE QUERY	Stream-about, stream-all, eos
RESPONSE	Reply, sorry
GENERIC INFORMATION	Tell, achieve, cancel, untell, unachieve
GENERATOR	Standby, ready, next, rest, discard, generator
CAPABILITY-DEFINITION	Advertise, subscribe, monitor, import, export
NETWORKING	Register, unregister, forward, broadcast, route

Tabela 1 Ações KQML

Com o uso desta ações, é possível estabelecer um padrão de comunicação entre agentes de *software*, cabendo ao modelador do sistema definir o conteúdo enviado em cada mensagem do sistema.

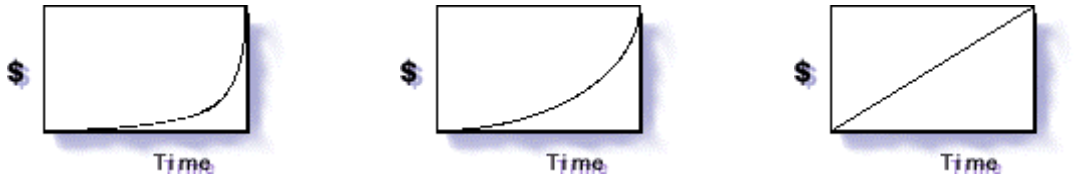
Quando utilizados em sistemas não multiagentes, a principal tarefa, na maioria das vezes, envolve as etapas de *Procura por Produto* e *Procura por Vendedor* do CBB. Agentes desta natureza, quando utilizados na *Web*, devem ser capazes de navegar de forma eficiente entre os diversos *hosts* na busca pela informação desejada.

Entre as principais arquiteturas construídas para a utilização de agentes de *software* no comércio eletrônico temos:

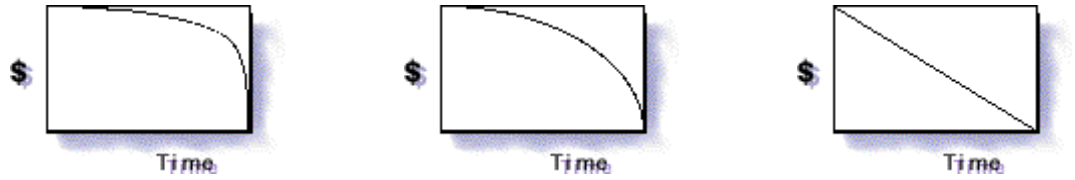
2.1.2.1 KASBAH

Desenvolvido no grupo *Software Agents*(http://agents.www.media.mit.edu/groups/agents) do *Media Lab* no MIT [AMEC, 1999], o Kasbah [Chavez, 1996ª] [Chavez, 1996b] é uma arquitetura multiagente para compra e venda de livros. Basicamente, os usuários criam agentes de compra e venda que interagem para atingirem seus objetivos. A modelagem utilizada define somente a possibilidade de se utilizar agentes proprietários do sistema

(agentes desenvolvidos junto à aplicação) em contraste com ambientes multiagentes que permitem a execução de agentes externos desenvolvidos por terceiros. Os agentes do sistema são mediadores de todo o processo de negociação, incluindo a própria etapa de negociação na qual o acordo sobre o preço final é estabelecido. O processo de negociação para agentes de compra e venda envolve a utilização de funções contínuas para a variação do preço no envio de propostas interagentes, durante o tempo de atividade, modelando as possíveis estratégias de negociação do sistema.



Estratégia de Compra Greedy | *Estratégia de Compra Cool-Headed* | *Estratégia de Compra Frugal*



Estratégia de Venda Greedy | *Estratégia de Venda Cool-Headed* | *Estratégia de Venda Frugal*

Figura 3 Funções contínuas para modelagem de estratégia de negociação

O usuário do sistema interage e personaliza as tarefas de cada agente por uma interface *Web*. Da mesma forma, e por uma *applet* Java, o administrador tem a possibilidade de gerenciar o mercado virtual utilizando serviços de administração disponíveis no sistema. A tabela, abaixo, lista alguns dos serviços de administração disponíveis no sistema.

FUNÇÕES DE ADMINISTRAÇÃO
Número de agentes do mercado virtual (ACTIVE, EXPIRED e DEAL MADE)
Alteração de senha de usuários
Número de transações consolidadas
Shutdown do mercado virtual

Tabela 2 Kasbah - Funções de Administração

2.1.2.2 MAGMA (MINNESOTA AGENT MARKETPLACE ARCHITECTURE)

Trata-se de um sistema multiagente para o comércio eletrônico desenvolvido na Universidade de Minnesota [Tsvetovatyy, 1996]. O sistema dispõe de agentes de software implementados em Java denominados *Trader Agents* que são responsáveis pela compra e venda de itens utilizando a *Web* como espaço de busca. Os agentes do sistema são capazes de comercializar itens eletrônicos facilmente transmitidos pela rede como música, artigos eletrônicos e softwares. A arquitetura do sistema consiste em múltiplos *Trader Agents*, *Advertising Servers* e um Banco Virtual.

- *Trader Agents* – Agentes de Software responsáveis pela compra e venda de produtos na Internet;
- *Advertising Server* – Prove um serviço de classificados eletrônico com funcionalidades de busca e recuperação de anúncios por categoria;
- *Bank* – Prove um conjunto de serviços bancários como adminstração de contas correntes, linhas de crédito e dinheiro eletrônico.

Toda a comunicação entre os agentes é feita por *sockets*, utilizando-se um Servidor centralizado para manter as conexões ativas e rotear as mensagens de forma correta. A arquitetura ainda implementa um Banco que disponibiliza um conjunto de serviços para os agentes, como a utilização de dinheiro eletrônico, crédito e débito e empréstimos.

A modelagem utilizada no sistema, permite que agentes implementados em diferentes linguagens de programação interajam no mercado virtual através da utilização de uma API comum de comunicação disponível na própria arquitetura.

2.1.2.3 VMARKET

Desenvolvido no Laboratório de Engenharia de Software da Puc-Rio, o *framework* Vmarket [Ripper, 1999] [Ripper, 2000]é uma evolução da arquitetura KASBAH [Chavez, 1996[a]]. Permitindo da mesma forma o encontro de agentes de compra e venda em um mercado virtual multiagente, o *framework* possibilita a venda de múltiplos itens de forma simples e automática. Utilizando XML [Jaenicke, 1999[a]] para a construir uma linguagem de definição de itens, o sistema permite que novas categorias de itens sejam adicionadas ao mercado virtual entregando a descrição da nova categoria a um *software* instanciador de aplicações. Diferente do KASBAH, no qual a negociação baseava-se, unicamente, no preço do item, o *framework* permite a criação de aplicações nas quais múltiplas dimensões de negociação [Guttman, 1998b] [Ripper, 1999] podem ser consideradas, pelos agentes de *software*, para o fechamento de acordos. Estas dimensões poderiam ser prazo de entrega, forma de pagamento, algum atributo especial do item, ou qualquer outra dimensão considerada importante pelo usuário.

2.1.2.4 IMPULSE

Um dos projetos mais recentes também desenvolvido no *Software Agents Group* do MIT, IMPULSE [AMEC, 1999] tem como objetivo principal permitir que pessoas realizem compras por meio de um sistema disponível em dispositivos móveis como PDAs. Pela utilização de agentes capazes de negociar considerando múltiplas dimensões de

negociação, os usuários podem localizar ofertas de itens, negociá-las eletronicamente por agentes do sistema em qualquer lugar, eliminando-se, assim, a necessidade de se estar à frente de computadores pessoais de mesa. O projeto foca sua pesquisa em três principais pontos:

1. *Perfil do usuário* – A otimização na modelagem do perfil do usuário pode influenciar as ações tomadas pelos agentes de *software* do sistema;

2. *Negociação constante* –Por um processo contínuo de negociação, os agentes são capazes de avaliar muitas ofertas de itens disponíveis no sistema, conseguindo encontrar produtos mais adequados às necessidades dos usuários;

3. *Personalização constante* – O usuário tem a possibilidade de alterar a execução dos agentes de *software* a qualquer instante por meio da atualização de suas preferências com relação aos produtos oferecidos e procurados no mercado.

2.2 Mercados Virtuais

Nos quatro segmentos de mercado, as aplicações de comércio eletrônico, mostradas no quadro abaixo, exemplificam aplicações da *Internet* brasileira.

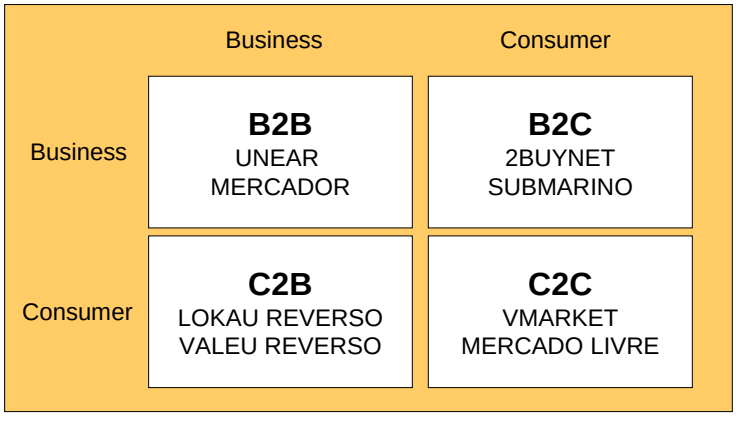


Figura 4 Exemplos de aplicações de comércio eletrônico por segmento de mercado

Dentre os exemplos mostrados na figura 4, O Vmarket é o único que faz uso, explicitamente, da tecnologia de agentes de *software*. Os outros exemplos que serão descritos mais a diante promovem ambientes de comércio nos quais a aplicação de agentes de *software* pode agregar valor em algumas etapas. A seguir, estes exemplos são apresentados, identificando-se os objetivos de cada sistema, suas principais características e os pontos nos quais a tecnologia de agentes poderia ser aplicada.

2.2.1 B2B – Business to Business

O segmento B2B (Business to Business) agrupa todas as aplicações que promovem a aproximação entre empresas, somente entre elas. Neste grupo de aplicações, não há a figura do consumidor pessoa física e, geralmente, a dinâmica de transações caracteriza-se pelo baixo volume de transações de alto valor. Como aplicações deste grupo, podemos citar o modelo implementado pela UNEAR (http://www.unear.net), que permite a realização de venda cruzada entre empresas com *Websites* estabelecidos na *Internet* e o Mercador, aplicação que aproxima Supermercadistas e Vendedores de produtos para supermercados.

2.2.1.1 UNEAR

Recentemente estabelecida no mercado brasileiro, a UNEAR (http://www.unear.net) é uma empresa que permite a aproximação entre empresas para a realização de transações B2B. No modelo utilizado pela UNEAR, é necessário que as empresas estejam presentes na *Internet* pela utilização de *Websites* ou de qualquer outra interface na qual o usuário final possa interagir e utilizar os serviços e produtos oferecidos por elas. No modelo utilizado pela empresa, fica clara a participação de duas figuras no cenário B2B.

- *Empresas* – São os clientes da UNEAR. Estas empresas poderão realizar transações comerciais entre si, utilizando seus *Websites* como veículos potencializadores destas transações;
- *Usuários* – São os clientes das empresas e representam a potencial audiência dos *Websites* das empresas clientes da UNEAR, utilizando os serviços e conteúdos disponibilizados por elas em seus respectivos *Websites*.

Desta forma, a tecnologia desenvolvida pela empresa permite que empresas potencializem a utilização de seus *Websites*, oferecendo-os como principal *frontend* de oferta de *banners*, serviços para suas respectivas audiências. Na realização de um acordo entre empresas (transação B2B) com a tecnologia UNEAR implantada, a disponibilização de serviços e *banners* nos *Websites* de ambas as partes é feita de forma simples e automática através da comunicação da tecnologia implantada.

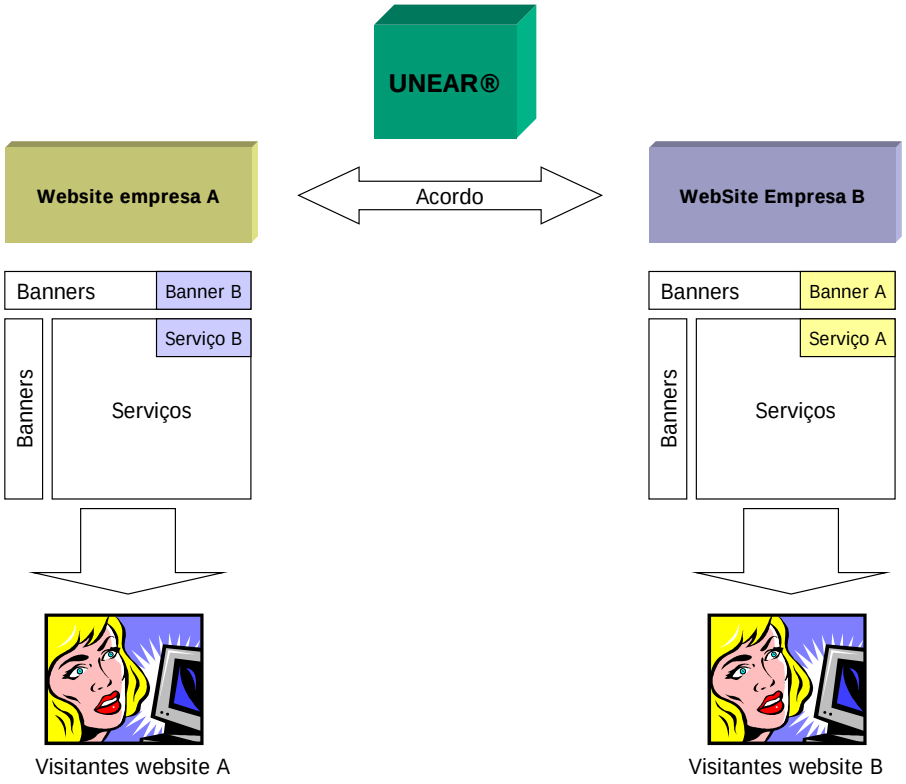


Figura 5 Mecanismo UNEAR

Através da criação de Regras de Negócio, o gerente de marketing de cada empresa pode estabelecer como, quando e para quem um determinado *banner* deve ser ofertado. As regras utilizam informação do perfil dos usuários em conjunto com informações temporais (estação do ano, período do dia) para ofertar o *banner* mais adequado para cada usuário. Uma típica regra de negócio é mostrada na figura 6:

Se Smiles = Possui E Sexo = Homem

Figura 6 Regra de Negócio UNEAR

A regra exemplificada acima, pode estar associada a um *banner* de publicidade que pode ser mostrado em uma determinada área do site. No momento de construção dinâmica de uma página do site para um determinado usuário, esta regra é avaliada automaticamente pelo software junto a outras regras definidas pelo gerente de marketing do site para determinar qual *banner* é o mais adequado para ser exposto ao usuário naquele momento. As regras de negócio podem ser construídas utilizando-se informações do perfil do usuário (características pessoais) e informações diversas que podem influenciar na oferta de produtos como a estação do ano influencia na venda de ar condicionado por exemplo.

Neste mecanismo estabelecido pela UNEAR para potencializar a realização de transações comerciais entre empresas com *Websites* na *Internet*, a tecnologia de agentes de *software* poderia agregar valor, principalmente, na otimização de acordos entre empresas:

- *Otimização de acordos* - A utilização de agentes de *software* permitiria otimizar a realização de acordos no sentido de encontrar as empresas com a audiência mais qualificada para a oferta de um determinado serviço ou produto. Isto é possível, pois o mecanismo da UNEAR permite traçar e acompanhar o perfil dos usuários do *Website* e, utilizando esta informação, os agentes poderiam identificar e classificar empresas de acordo com a qualificação de sua audiência. A utilização de informação de outra natureza, como tamanho da audiência ou área de atuação da empresa, poderia ser utilizada pelos agentes na tarefa de classificação.

A otimização de acordos seria equivalente à etapa de Procura por Vendedor no modelo CBB, no qual a figura do vendedor mais apto seria substituída pela empresa mais apta à realização de um acordo de venda cruzada de serviços e *banners*.

2.2.1.2 MERCADOR

O Mercador (http://www.mercador.com.br) é uma iniciativa brasileira para aproximar supermercadistas de fornecedores de produtos para supermercados, por um *Website* construído para este propósito. Como pré-requisito, o serviço exige que Supermercadistas e Fornecedores de produtos cadastrem-se no *Website*.

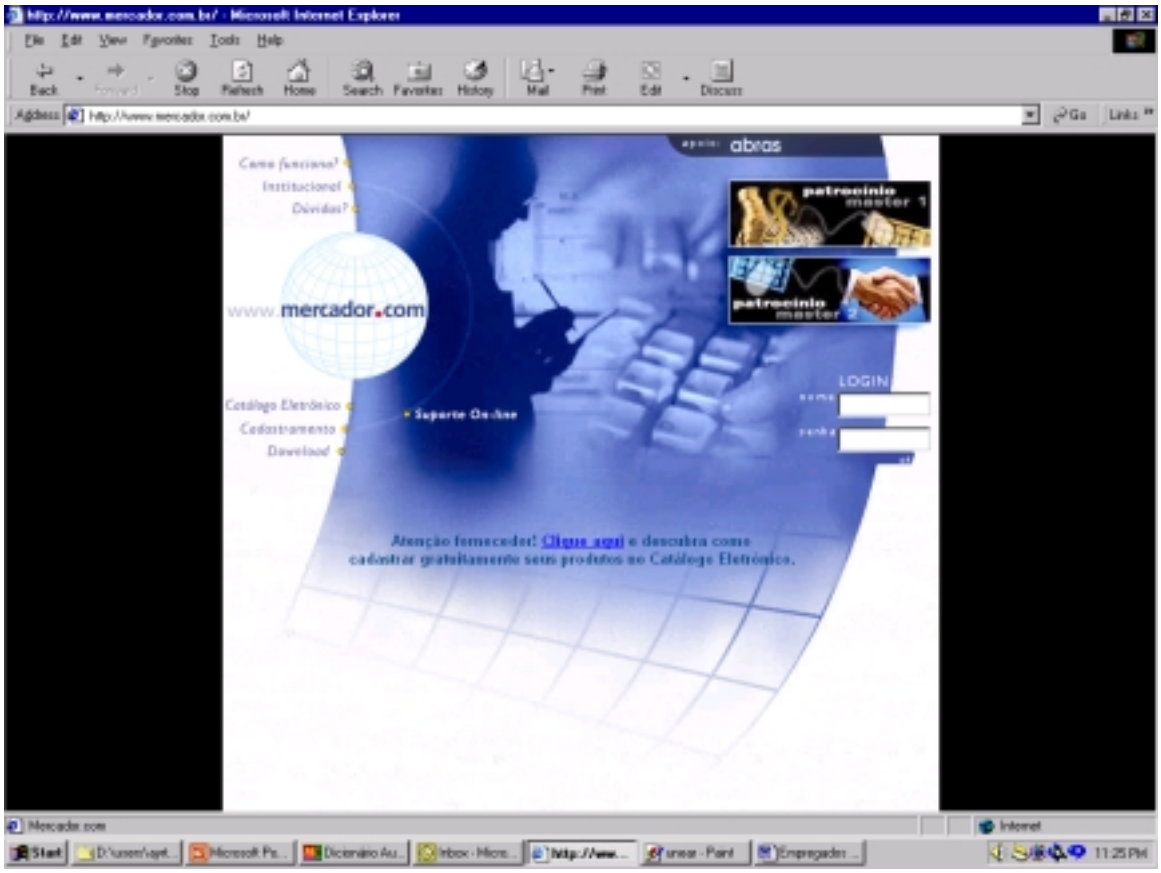


Figura 7 Página inicial do site Mercador

Caso ainda não seja cadastrado e autorizado a utilizar o sistema, a página principal do *site*, como mostra a figura 7, permite que o usuário cadastre-se por meio de um processo simples de três etapas no qual as informações cadastrais devem ser fornecidas.

Os Fornecedores de Produtos devem, então, informar suas ofertas e os respectivos preços em formulários no *Website* para que os Supermercadistas possam, finalmente, utilizar o *site* para encontrar as ofertas de produtos desejadas e escolher entre as mais apropriadas. Desta forma, o serviço, hoje disponibilizado, consiste em um mecanismo de cotação *online*. Esta etapa é equivalente à etapa de Procura por Produto e Procura por Vendedor.

A figura 8, mostra a interface para os supermercadistas na qual o catálogo de produtos é exibido.

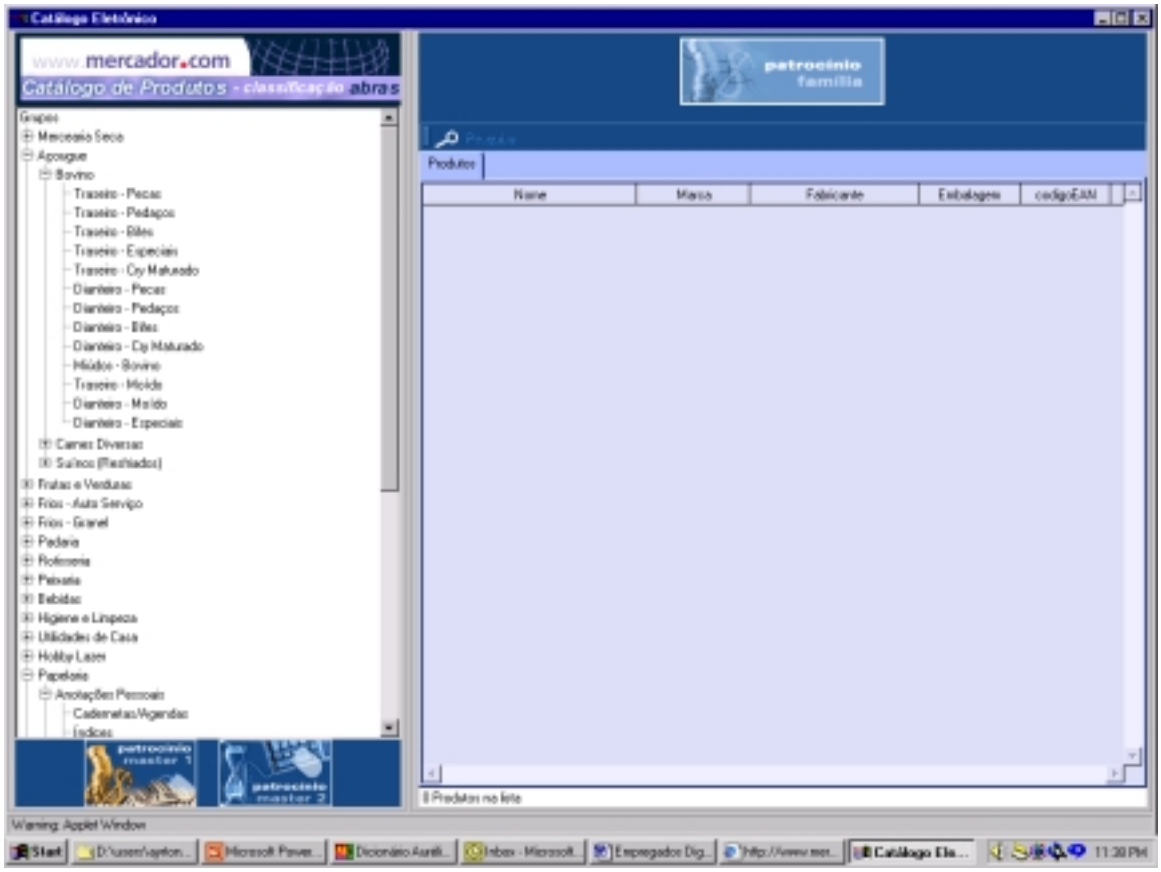


Figura 8 Catálogo de Produtos Mercador

No menu da esquerda, os supermercadistas podem navegar em uma estrutura de árvore (*outline*) que organiza os produtos, previamente cadastrados por fornecedores de produtos, podendo escolher, comparar preços e comprar os produtos mais interessantes por um processo integralmente interativo.

Sendo um mercado caracterizado por produtos de atributos bem definidos e, muitas vezes padronizados, a utilização de agentes de *software* é visível neste cenário, diminuindo o esforço de busca por informação no lado dos supermercadistas. Com a disponibilização de agentes, supermercadistas poderiam, simplesmente, passar uma lista de produtos para seus agentes, e estes buscariam, de forma automática, as melhores ofertas disponíveis no mercado, mostrando-se um sistema mais eficiente para supermercadistas.

Em um segundo momento, os agentes de *software* poderiam realizar, de forma automática, a etapa de negociação, elevando o Mercador a um sistema de negociação dinâmica capaz de ofertar o mesmo produto diferentemente para supermercadistas distintos, ao invés de um catálogo de preços estáticos no qual as diferenças entre supermercadistas não é considerada.

2.2.2 B2C – Business to Consumer

O segmento B2C (*Business to Consumer*) agrupa, hoje, o maior conjunto de aplicações de comércio eletrônico disponíveis atualmente na *Internet*. Este segmento caracteriza-se pelas empresas que ofertam produtos diretamente para o consumidor final, ou em outras palavras, para as pessoas que navegam na *Internet*. Encontram-se neste grupo todas as lojas virtuais da *Internet* que, por catálogos virtuais e formas de pagamento bem definidas, mostram uma solução adequada, porém com algumas barreiras a serem vencidas. O exemplo descrito a seguir, mostra o 2BuyNet (http://www.2buynet.com.br), inicialmente desenvolvido no Laboratório de Engenharia de Software da Puc-Rio (http://les.inf.puc-rio.br), que permite a criação de lojas na *Internet* de forma completamente automática.

2.2.2.1 2BUYNET

O 2BuyNet [Fortunato, 1999] é uma tecnologia capaz de gerar lojas virtuais prontas para operar na *Internet* de forma semi-automática. As lojas geradas pela ferramenta possuem algumas características comuns, definidas como básicas e necessárias para qualquer loja virtual.

1. *Catálogo de Produtos* – Consiste na interface principal de navegação do consumidor. Interagindo com o *Website* pelo catálogo virtual, o consumidor avalia os produtos ofertados pela loja, podendo adicioná-los ao carrinho de compras quando desejar;

2. *Carrinho de Compras* – Permite que o usuário guarde, em uma lista eletrônica, todos os produtos que serão comprados. A metáfora com o carrinho de compras de um supermercado é obvia, e o mecanismo mostra-se perfeitamente adequado a este tipo de aplicação de comércio eletrônico. A gerência do carrinho de compras é completa, permitindo que o usuário retire e adicione produtos em qualquer instante;

3. *Mecanismos de Pagamento* – Com o carrinho de compras supostamente completo, o usuário deve realizar a compra efetiva, por meio de mecanismos de pagamento variados. As lojas geradas pela ferramenta têm, à princípio, disponíveis cartão de crédito, boleta bancária e depósito bancário como alternativas para a realização do pagamento.

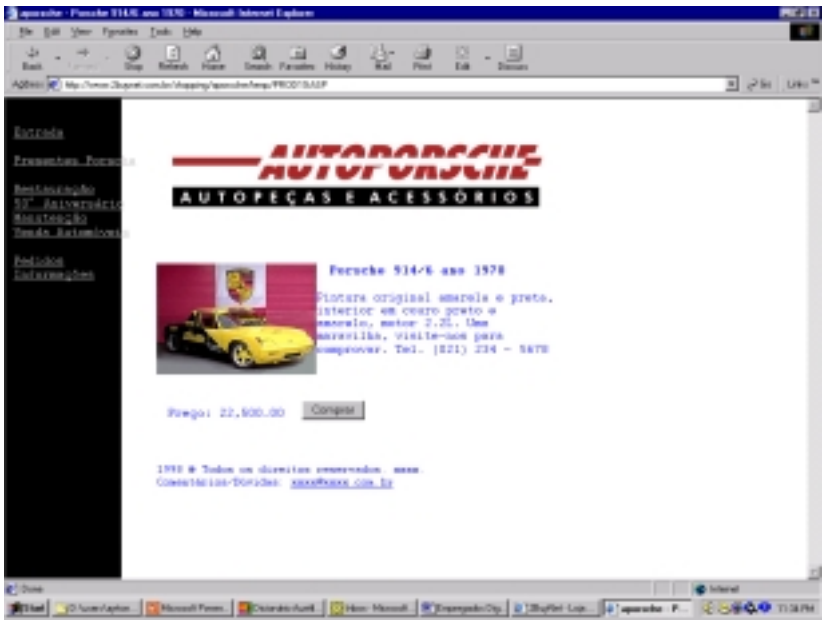


Figura 9 Visão do Catálogo de produtos 2BuyNet na loja Autoporsche

O público alvo principal do 2BuyNet são lojas de pequeno e médio porte, porque, à princípio, não precisam de uma solução absolutamente customizada. O uso da tecnologia de agentes de *software* em lojas virtuais permitiria a criação de lojas virtuais com catálogos de produtos dinâmicos, ou seja, com a possibilidade de se barganhar o preço de etiqueta de um produto eletronicamente, sem a necessidade de contato direto com vendedores. Por um mecanismo automático de negociação, uma nova concepção de lojas virtuais poderia ser criada, dando espaço para a criação de promoções inteligentes. Desta forma, esta nova concepção de lojas teria as seguintes características:

1. *Catálogos dinâmicos* – Aproveitando a informação do perfil do usuário, agentes de *software* poderiam dinamizar o catálogo virtual da loja oferecendo por exemplo, preços diferenciados para usuários com perfis distintos;

2. *Opção de negociação automática* – Pela automação da etapa de negociação do CBB, estas novas lojas virtuais poderiam oferecer a chance de uma "barganha eletrônica", na qual o usuário poderia barganhar o preço do produto antes de realizar a compra.

2.2.3 C2B – Consumer to Business

As aplicações C2B diferem das B2C na orientação do modelo de negociação. Neste segmento de mercado, o usuário final define o preço e todas as suas preferências relacionadas à sua intenção de consumo e, após a publicação destas informações, vendedores competem e jogam com suas margens de lucro para conquistar a venda. Este modelo tornou-se famoso, inicialmente, nos EUA com a iniciativa da Priceline (http://www.priceline.com) no mercado de passagens aéreas, pela qual o usuário final definia o preço do bilhete. No Brasil, iniciativas semelhantes começam a aparecer como o Valeu Reverso, um serviço do *site* de leilão Valeu (http://www.valeu.com.br) que permite que o comprador de itens faça seu preço para algumas categorias disponíveis no Leilão. O Lokau, *site* de Leilão (http://www.lokau.com.br), disponibilizou um serviço no qual os usuário têm a possibilidade de dizer o preço desejado na compra de carros.

2.2.3.1 VALEU REVERSO

O *Website* brasileiro de Leilão Valeu (http://valeu.com.br), oferece um serviço especial denominado Valeu Reverso, no qual o usuário tem a chance de informar o preço que gostaria de pagar para um determinado produto ofertado no referido *site*.

A utilização do sistema pode ser descrita em duas principais etapas, que devem ser seguidas pelos usuários do *site*.

- *Informação do usuário* –Por um formulário HTML, o usuário deve informar seus dados pessoais, caso não esteja cadastrado no *site*. Se o usuário já for membro do *site* Valeu, deverá informar somente seu ID e sua senha para que o procedimento de identificação e autenticação sejam realizados;

- *Informação do produto desejado e condições desejadas* – A segunda etapa consiste na informação do produto em questão desejado pelo usuário. Neste momento, o usuário deve informar a categoria a qual o produto pertence e suas condições ideais para que a negociação, o preço desejado, o prazo de entrega adequado e qualquer outra necessidade específica, possam ser expostas nos campos *Características Desejadas e Observações*, como mostra a figura 10.

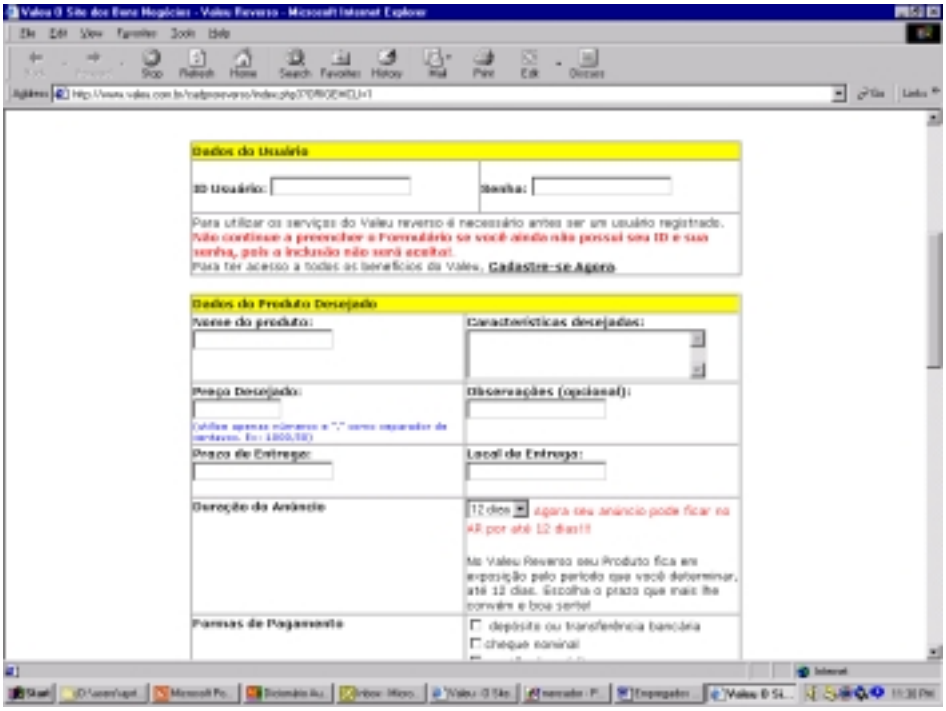


Figura 10 Valeu Reverso – etapa 1

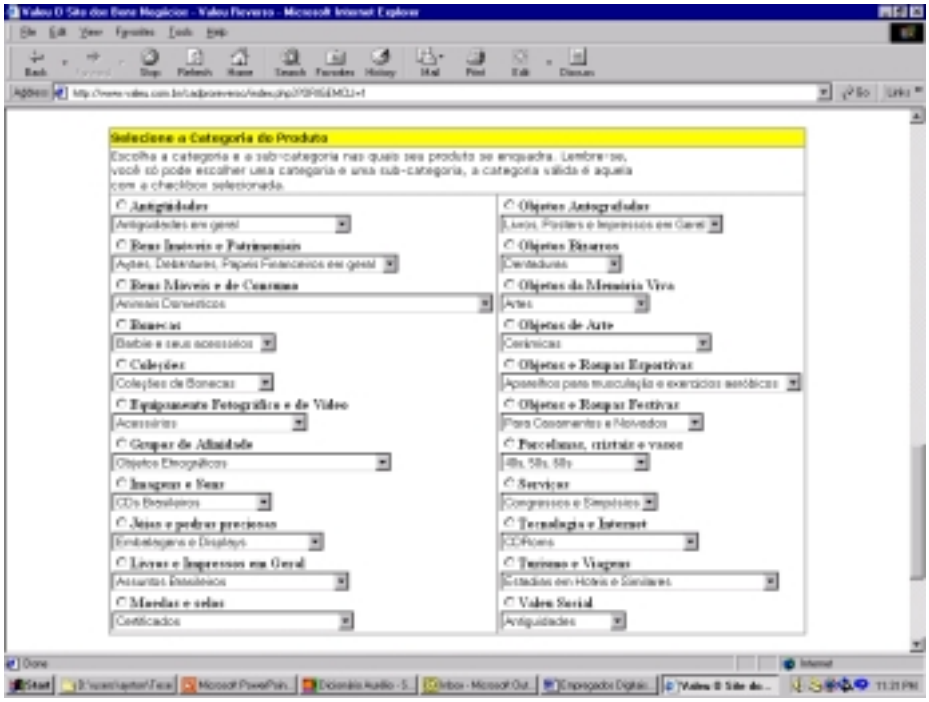


Figura 11 Valeu reverso – etapa 2

Após o cadastramento da intenção de consumo do usuário, o sistema procura viabilizar a negociação aproximando os vendedores aptos a vender o produto desejado. Cabe, então, a estes vendedores analisarem as necessidades do consumidor e disputarem a venda, negociando margem de lucro, prazo de entrega e outras dimensões de negociação.

A utilização de agentes de *software* neste sistema poderia tornar o Valeu Reverso uma aplicação com características semelhantes ao *framework* proposto neste trabalho de pesquisa no qual a procura por vendedores e o processo de negociação são automatizados por agentes de *software* de compra e venda.

2.2.4 C2C – Consumer to Consumer

O segmento C2C é representado na *Internet* principalmente pelos *Websites* de Leilão. Nos EUA, existem, hoje, pelo menos 2000 *Websites* do gênero, sendo o eBay (http://www.ebay.com) o maior deles. No Brasil, já existem pelo menos 10 *sites* de leilão

disponíveis e, entre eles, podemos citar o Lokau (http://www.lokau.com.br), o Valeu (http://www.valeu.com.br), o Arremate (http://www.arremate.com.br), o Mercado Livre (http://www.mercadolivre.com.br), o Mercado 21 (http://www.mercado21.com.br) entre outros.

As aplicações deste segmento procuram aproximar usuários compradores de usuários vendedores. Não existe a presença de empresas neste mercado. Isto mostra que o modelo de leilão não é a única alternativa para aproximar compradores e vendedores. O VBookMarket (http://harper.les.inf.puc-rio.br/vbookmarket), aplicação instanciada a partir do Framework Vmarket desenvolvido no LES (Laboratório de Engenharia de Software), implementa um mercado virtual no qual as pessoas podem vender e comprar livros por meio de um modelo de negociação diferente de leilão.

2.2.4.1 VMARKET

O Framework Vmarket [Ripper, 1999], desenvolvido no Laboratório de Engenharia de Software, é um framework para aplicações do segmento de mercado C2C. A aplicação VBookMarket instanciada a partir do *framework* disponibiliza um mercado de livros na *Internet*.

Os Mercados Virtuais do Framework VMarket permitem o encontro entre compradores e vendedores através do uso de agentes de *software* que representam, de forma completa, o usuário no sistema. No modelo atual do *framework*, todos os usuários do sistema são potenciais compradores e vendedores, dependendo de seus interesses, ou seja, os mercados instanciados a partir do *framework* são, essencialmente, C2C, não existindo, portanto, diferenciação entre consumidores e vendedores no mercado.

A utilização de um modelo centralizado traz grandes vantagens para um sistema de comércio eletrônico como, por exemplo, a utilização de estratégias e protocolos de negociação específicos para o item negociado, permitindo, assim, um alto nível de

especialização do mercado. Outra característica importante do *framework* é a utilização de agentes de *software* como mediadores dos usuários no sistema. Os agentes automatizam duas etapas essenciais do processo de negociação:

- *Localização da parte interessada* – Os agentes localizam, no mercado, os outros agentes interessados em negociar o produto por eles representado;
- *Negociação* – Os agentes encarregam-se de realizar a negociação do produto de forma automática, sem a necessidade de intervenção do usuário.

Agentes de *software*, de forma geral, possibilitam a intervenção mínima possível do usuário no sistema. No Framework VMarket, basicamente, o usuário precisa descrever o item que deseja negociar, a estratégia de negociação e definir como e por quanto tempo deseja negociá-lo no mercado. Eventualmente, o usuário recebe mensagens de seu agente por meio de mecanismos de comunicação como *e-mail*, *pager*, celulares, *Web pages*, informando a consolidação de transações, expiração do tempo de atividade do agente, dificuldade em negociar, entre outras notificações interessantes.

Figura 12 Vbookmarket - Aplicação instanciada a partir do framework VMarket

Capítulo 3

Tecnologia Wireless

O acesso a conteúdos e serviços disponíveis na *Internet* deixa de ser exclusivo para computadores pessoais e *notebooks* à medida que surgem novas tecnologias para acesso à *Internet* por meio de dispositivos móveis. A tendência do mercado mostra que, em um futuro próximo, será comum a utilização de serviços da rede pelas inúmeras interfaces como celulares, PDAs (*Personal Digital Assistant*) e até carros. Até mesmo os próprios acessórios estarão conectados à rede sem fio. Hoje, celulares e PDAs já são utilizados como interface para acessar a *Internet* [Wolk, 2000]. A Europa já se posiciona como pioneira no acesso a internet por dispositivos móveis.

As possibilidades deste novo cenário são inúmeras e, como mostra esta dissertação de Mestrado, o impacto no comércio eletrônico é significativo. Algumas características importantes são listadas, abaixo, com relação à influência desta nova tecnologia no comércio eletrônico [Durlacher, 2000]:

1. *Ubiquidade* – O usuário final tem acesso aos serviços e conteúdos a qualquer momento e a qualquer hora, independente de sua localização, facilitando, conseqüentemente, transações comerciais;

2. *Acesso ao usuário final* – Com prévia autorização do usuário do dispositivo móvel, é possível contatá-lo para fins diversos como promoções, alertas, etc.;

3. *Conveniência* – O conteúdo e os serviços devem ser remodelados com relação à forma de apresentação e de interação. O usuário será mais exigente com relação à

utilidade do conteúdo, a facilidade de utilização e de forma alguma aceitará "ampulhetas de espera"[2] ou GPFs (*General Protection Faults*)[3].

4. *Localização* – Com a disponibilização do serviço de localização de terminais móveis [Durlacher, 2000], otimizações e novos serviços surgirão, trazendo ainda mais valor agregado para o comércio eletrônico móvel, ou melhor, para o m-commerce (nome dado a todo o comércio eletrônico realizado através de dispositívos móveis conectados a Internet) [Durlacher, 2000][Nokia, 1999c].

Com a percepção de todas estas novas possibilidades, as principais empresas fabricantes de telefones celulares criaram um Fórum com o objetivo de estabelecer a especificação de um padrão tecnológico que permitisse o desenvolvimento de aplicações e serviços capazes de integrar celulares, *pagers*, PDAs à *Internet*. As empresas envolvidas são: *Nokia* (http://www.nokia.com), *Ericsson* (http://www.ericsson.com), *Phone.com* (http://www.phone.com), *Motorola* (http://www.motorola.com) e outras. Denominado WAP (*Wireless Application Protocol*), esta especificação define um *framework* para o desenvolvimento de aplicações e um conjunto de protocolos desenvolvidos para a rede sem fio capazes de integrar dispositivos móveis à *Internet* [WAP, 1999a] [WAP, 1999b]. O fórum, denominado WAP Forum, pode ser acionado no endereço http://www.wapforum.org, e, atualmente, desenvolve a versão 1.2 do protocolo WAP.

Os principais objetivos do WAP Forum são:

- Permitir que se tenha acesso a serviços e conteúdos disponíveis na *Internet* por telefones celulares e outros terminais móveis;

- Estabelecer um protocolo de aplicação padrão independente das diferentes tecnologias de rede sem fio existentes, como GSM (*Global System for Mobile*

[2] Ícone utilizado em interfaces gráficas para alertar, ao usuário, de um computador pessoal que o programa encontra-se processando

[3] Nome dado a erros de implementação de *softwares* desenhados para a plataforma Microsoft Windows.

Communications), CDMA (*Code Division Multiple Access*) e TDMA (*Time Division Multiple Access*);

- Sempre que possível, utilizar e seguir padrões existentes já absorvidos pelo mercado, diminuindo a inércia de aprendizado.

Toda a definição da arquitetura leva em consideração alguns dos principais aspectos devido às limitações existentes nos dispositivos e nas redes móveis. Ainda por limitação tecnológica, dispositivos móveis tendem a ter:

- *Pouca capacidade de processamento*
- *Pouca memória disponível*
- *Consumo de energia restrito*
- *Telas pequenas*
- *Variadas e limitadas formas de entrada de dados*

Da mesma forma, redes de dados sem fio apresentam características, completamente diferentes das características de redes cabeadas, que devem ser consideradas:

- *Pouca largura de banda*
- *Maior latência*
- *Menor estabilidade na conexão*

Desta forma, é fundamental que estes pontos sejam considerados na definição da arquitetura WAP, permitindo que aplicações sejam desenvolvidas e apresentadas de forma adequada nos dispositivos dos usuários finais que satisfaçam as limitações tecnológicas da infra-estrutura de redes sem fio atual.

3.1 A arquitetura WAP

Seguindo o objetivo de utilizar padrões já absorvidos com sucesso pelo mercado, a arquitetura WAP foi definida espelhando-se no modelo de programação da *Web*. Sendo um modelo já bastante conhecido, aplicações desenvolvidas para a *Web* são visualizadas em um formato simples de apresentação no qual o usuário final utiliza *Web browsers* como o *software* cliente responsável pela apresentação destas aplicações.

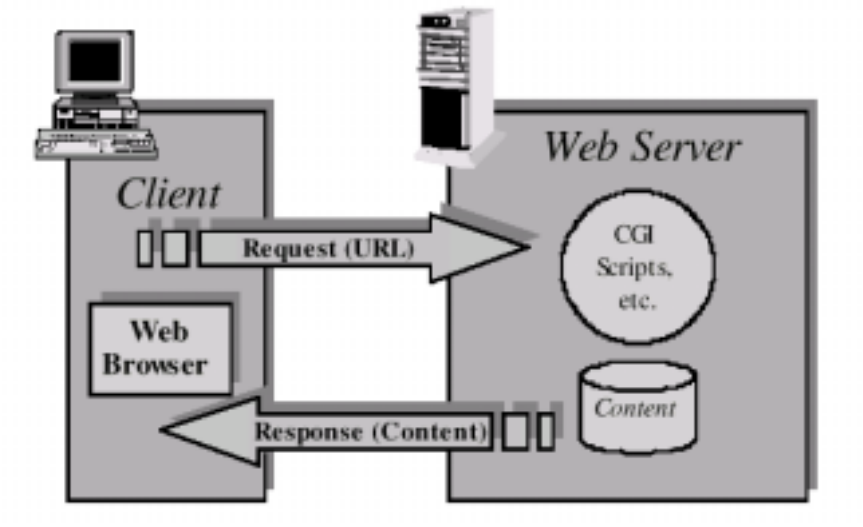


Figura 13 Modelo de programação Web

O modelo de programação WAP foi definido baseando-se neste modelo, no qual, agora, clientes fazem requisição de documentos por *micro browsers* [Nokia, 1998], e servidores respondem entregando o conteúdo desejado.

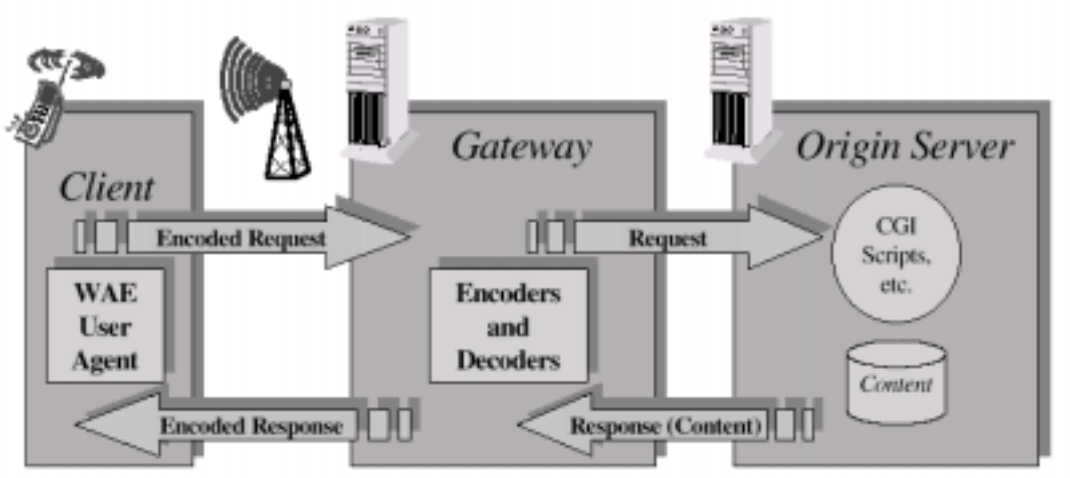


Figura 14 Modelo de programação WAP

A principal diferença é a necessidade de implementar um Gateway como ponte entre a rede cabeada e a rede sem fio. O papel principal deste Gateway, muitas vezes denominado WAP Gateway ou WAP Server, é codificar e decodificar o conteúdo trafegado em ambas as redes.

Da mesma forma que no modelo da *Web*, no modelo WAP usuários finais fazem uso de *micro browsers* embutidos em dispositivos móveis para interagir com serviços na *Internet*. O pedido codificado pelo micro *browser*, de acordo com o protocolo WAP, é transmitido pela rede sem fio até o WAP *Gateway*, onde é decodificado de acordo com a pilha de protocolos utilizados na WWW (*HTTP*, *TCP/IP*) para a execução do serviço. Após a execução do serviço, a resposta é codificada novamente para a pilha de protocolos da rede sem fio e enviada para o dispositivo móvel do usuário.

Para a comunidade de desenvolvedores de *software*, a criação de aplicações WAP para o usuário final faz uso de duas tecnologias básicas:

1. *WML* (*Wireless Markup Language*) [Nokia, 1999d]– Da mesma forma que o HTML, o WML define um conjunto restrito de *tags* capaz de gerar documentos com apresentação adequada para dispositivos com pouca capacidade de interface

e com pouca memória. Equivalente a metáfora de *Web sites* e documentos HTML utilizada no modelo WWW, no modelo WAP, as aplicações são organizadas em Cards (Elemento básico de visualização. Cards são visualizados um a um no micro browser) e Decks (Conjunto de cards). WML é uma linguagem definida utilizando-se XML como linguagem para a definição de sua gramática;

2. *WMLScript* [Nokia, 1999e]– Linguagem de *script* capaz de executar no cliente, ou seja, no dispositivo móvel. Equivalente ao JavaScript ou VBScript, utilizados no modelo WWW, o WMLScript é utilizado para validar a entrada de dados do usuário final, realizar ligações automaticamente dentre outras operações úteis ao cliente.

Para exemplificar, a figura 15 mostra uma aplicação desenvolvida para o protocolo WAP. Trata-se de uma aplicação bastante simples na qual criou-se um único Deck com um único Card que é visualizado na tela do telefone celular.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
   <card id="First_Card" title="First Card">
      <p>
      The first WML example
      </p>
   </card>
</wml>
```

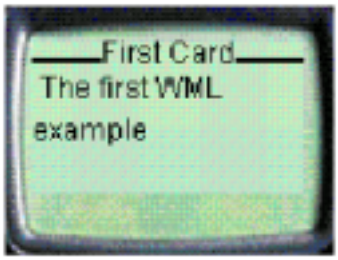


Figura 15 Exemplo de aplicação em WML

A semelhança com o modelo de desenvolvimento de aplicações para a *Web* é proposital, uma vez que visa à rápida absorção das ferramentas de desenvolvimento pela comunidade de desenvolvedores *Web*.

A pilha de protocolos WAP é organizada em uma estrutura em camadas. Cada camada da arquitetura é visível para as demais camadas e também diretamente para outros serviços e aplicações [WAP, 1999b].

1. *Aplicação* – Camada na qual as aplicações para o usuário final são desenvolvidas. Fazendo uso de WML e WMLscript, aplicações são disponibilizadas nesta camada do protocolo;

2. *Sessão* – Define dois tipos de serviço de sessão. O primeiro é com conexão e funciona sobre a camada de transação que utiliza o protocolo WTP. O segundo, sem conexão, utiliza o serviço de transporte disponibilizado pela camada de transporte, denominado protocolo WDP;

3. *Transação* – Implementa o protocolo de transação WTP. O WTP funciona sobre o serviço de datagrama e disponibiliza três tipos de transações: *Unreliable One-Way*, *Reliable One-Way e Reliable Two-Way*. O WTP foi desenhado, especialmente, para atender aos requisitos de memória e capacidade de processamento limitada de terminais móveis;

4. *Segurança* – Implementa o WTLS (*Wireless Transport Layer Security*). Este protocolo foi criado com base no TLS (*Transport Layer Security*), também conhecido como SSL. Esta camada tem como objetivo permitir o transporte de dados com privacidade, integridade e autenticação na rede sem fio;

5. *Transporte* – A camada de transporte é denominada WDP (*Wireless Datagram Protocol*) e opera sobre os portadores (*Beaerers*) da rede sem fio. Sendo uma camada de transporte genérica, oferece um serviço consistente que torna transparente, para camadas superiores, a utilização de diferentes portadores.

A figura 16 apresenta uma visão em camadas da arquitetura WAP.

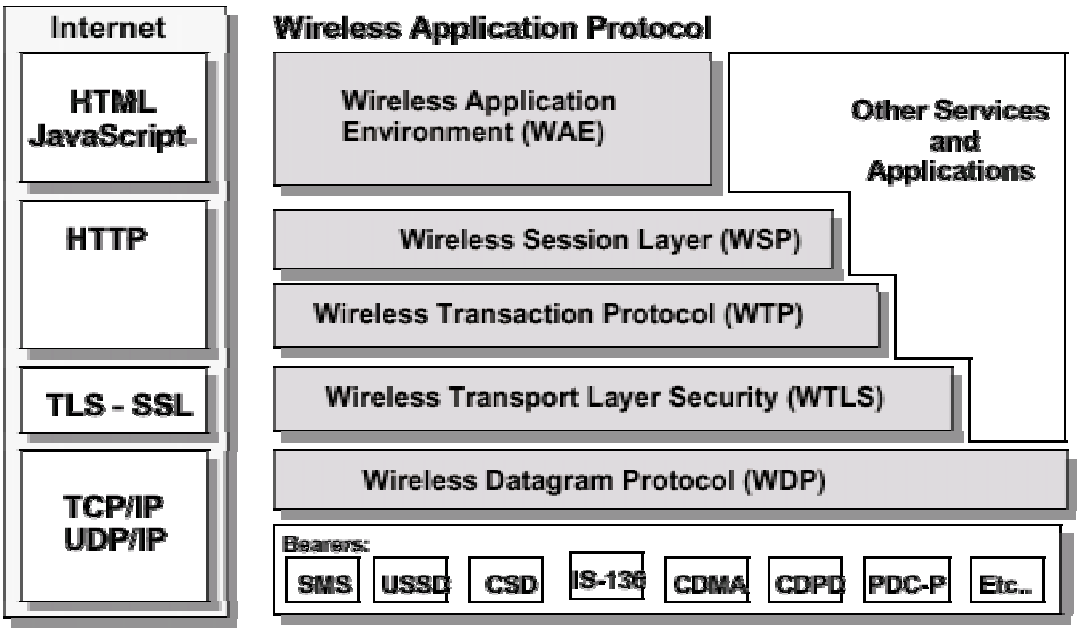


Figura 16 WAP - Arquitetura em camadas

Como é mostrado na figura 16, serviços e aplicações externas podem utilizar, diretamente, as camadas do protocolo WAP, utilizando, por exemplo, a camada de segurança, sem a necessidade de se utilizar todas as camadas superiores. Este acesso direto a camadas do protocolo acontece de acordo com um conjunto de interfaces funcionais bem definido. Este mecanismo permite que aplicações como Calendário, Email e Agenda possam ser adaptadas para a utilização do protocolo WAP.

3.2 m-Commerce com o protocolo WAP

O termo *m-commerce* define todo o comércio eletrônico realizado por dispositivos móveis [Durlacher, 2000] [Wolk, 2000]. O protocolo WAP como viabilizador de um cenário no qual usuários podem ter acesso a serviços na *Internet*, independentemente de onde estiverem, levou à criação deste termo que engloba todas os serviços de comércio eletrônico desenvolvidos para os usuários de dispositivos móveis.

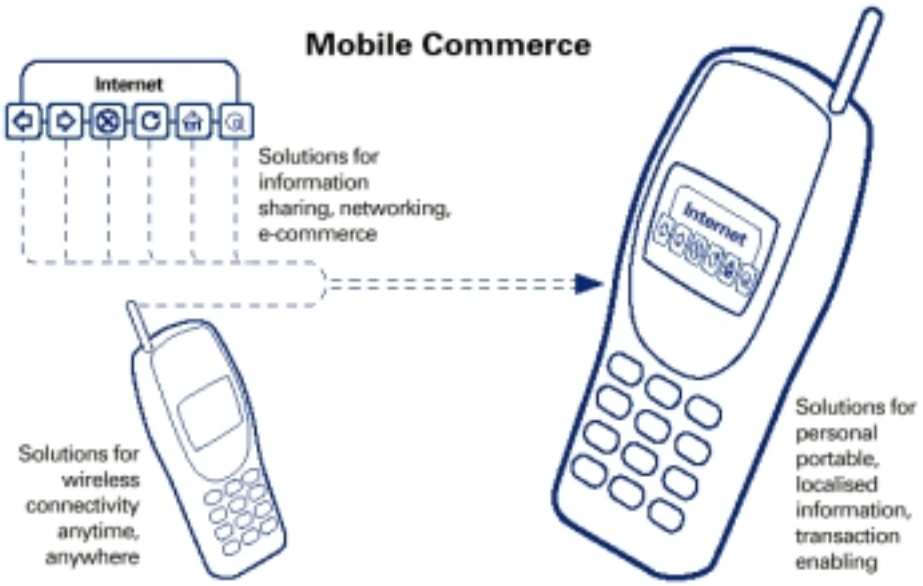


Figura 17 m-Commerce

Pelas características já descritas previamente, aplicações de comércio eletrônico para a comunidade de usuários de dispositivos móveis terão características diferentes das hoje implementadas para a *Web*, e deverão priorizar fatores não tão importantes no modelo da *Web* no qual, por exemplo, o usuário tem mais tempo disponível para interagir com a aplicação.

As aplicações de m-Commerce também podem ser divididas em duas categorias principais: *Consumer m-Commerce Applications* e *Business m-Commerce Applications*. Para exemplificar estas duas categorias de aplicações, a tabela 3, abaixo, lista possíveis aplicações para ambas as categorias.

Consumer m-Commerce Applications	Business m-Commerce Applications
Compra de tickets (compra de tickets em geral)	Telemetria (Gasolina, Kilometragem, ...)
Banco (controle da conta bancária)	Serviços de entrega (vendedores on line)
Leilões (realização de lances)	Integração de processos (suppply chain)
Cash (Retirada de dinheiro)	Vendas (alerta sobre consumidores)

Reservas (Reserva de hoteis)	WASP (Wireless App. Service Provider)

Tabela 3 Aplicações para o m-Commerce

Considerando a maior penetração de celulares e dispositivos móveis na população mundial, principalmente pela facilidade de operação do aparelho e pelo baixo custo de produção, o protocolo WAP mostra-se uma iniciativa bastante interessante para aplicações de comércio eletrônico e caminha para se tornar o padrão para o desenvolvimento de aplicações para dispositivos móveis. Considerando o aspecto tecnológico, a tecnologia *wireless* integra-se perfeitamente ao uso de agentes de *software*, principalmente pela possibilidade de combinar a automação de tarefas que facilitem a utilização de dispositivos com pouca capacidade de interação.

Desta forma, o capítulo, a seguir, apresenta o Framework CommercePipe desenvolvido para instanciação de aplicações nas quais a possibilidade de utilização de dispositivos móveis como forma de interação com mercados virtuais foi incorporada.

Capítulo 4

O Framework CommercePipe

Baseado no estudo de diferentes sistemas de comércio eletrônico e, principalmente, inspirado no trabalho de pesquisa e evolução de *software* realizado no *framework* Vmarket nos últimos 12 meses [Ripper, 2000] [Vbookmarket, 2000], este capítulo apresenta, detalhadamente, o Framework Commercepipe como solução tecnológica alternativa para a instanciação de mercados virtuais C2B na *Internet*. Utilizando a tecnologia de Agentes de Software combinada à Interface Wireless para acesso a serviços *Internet*, o *framework* propõe uma nova abordagem para mercados C2B, trazendo uma nova visão de como vendedores e compradores podem interagir para realizar transações comerciais pela *Internet*.

Enquadrando-se na categoria de Leilões Reversos já apresentada, o *framework* permite a criação de mercados virtuais nos quais consumidores estabelecem suas preferências, como prazo de entrega, preço, forma de pagamento, quantidade entre outras dimensões de negociação, e vendedores competem para satisfazer, da melhor forma possível, às demandas do mercado.

Uma apresentação breve sobre o conceito de *frameworks* para desenvolvimento de *software* é feita, inicialmente, dando uma visão sobre os principais objetivos a serem alcançados no desenvolvimento do mesmo.

Serão apresentados, em detalhes, todos os subsistemas do *framework*, seus principais pontos de flexibilização [Fontoura, 1999] e as características principais do núcleo do *software*, dando uma visão detalhada de toda a arquitetura do *software* desenvolvido.

Posteriormente, uma análise sobre as principais tecnologias escolhidas para o desenvolvimento do *software* é feita no final do capítulo, como linguagem de programação, banco de dados, linguagem de *script*, ferramenta de desenvolvimento, ferramenta de modelagem e outras.

4.1 Frameworks

Frameworks proporcionam a agilidade necessária para a evolução constante de *softwares*, o que caracteriza uma enorme vantagem competitiva em sistemas para o comércio eletrônico, nicho que se encontra em evolução, com constante surgimento de novas tecnologias e tendências. *Frameworks* focalizam, basicamente, duas principais características do desenvolvimento de *softwares*:

- *Reutilização* – O desenvolvimento de *software* reutilizável é um dos principais objetivos da tecnologia de Frameworks. Por meio do desenvolvimento de *software* orientado a objetos [Gamma, 1995], o uso de frameworks permite que classes, subsistemas ou até o próprio sistema possam ser reutilizados para a instanciação de novas aplicações, acelerando, por conseguinte, o processo de desenvolvimento de aplicações similares ou pertencentes ao mesmo domínio;
- *Flexibilidade* – Esta característica permite que variações e extensões possam ser realizadas no *software*, permitindo a instanciação de um amplo conjunto de aplicações a partir do *framework*. Tais variações e extensões, denominadas Pontos de Flexibilização [Fontoura, 1999], ocorrem sem a necessidade de nenhuma mudança fundamental no *design* do *software*.

Frameworks são constituídos por um núcleo de software e diversos pontos de flexibilização [Fayad, 1999a]. O desenvolvimento de um framework deve ser iniciado com uma análise de requisistos do domínio de aplicações para o qual o framework será modelado [Fayad, 1999b]. Esta análise de requisistos deve definir as características comuns

entre as aplicações do domínio em questão e quais são as características que variam de aplicação para aplicação. Com a realização desta análise de requisitos, é possível modelar o framework por completo, construindo o núcleo do software com base nas características comuns e deixando em aberto a implementação dos pontos de flexibilização com base nas características variáveis das diversas aplicações do domínio em questão.

Desta forma aplicações podem ser instanciadas a partir do framework através da instanciação dos pontos de flexibilização. Geralmente esta instanciação se dá através da implementação de classes concretas e através da customização de trechos de código. Em algumas situações é possível criar softwares capazes de automatizar o processo de instanciação de aplicações, gerando código automaticamente para a customização dos pontos de flexibilização.

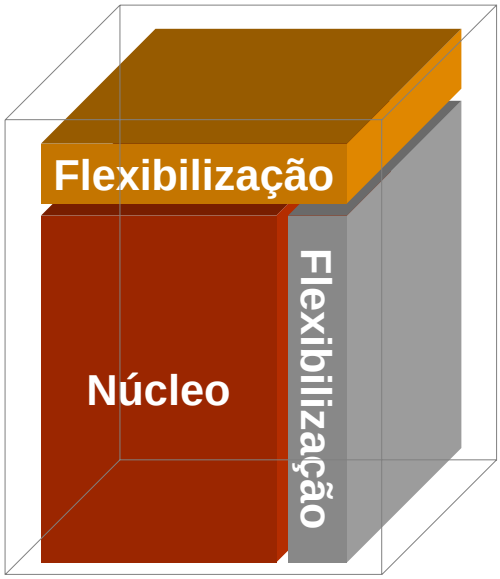


Figura 18 Estrutura de frameworks

4.2 Objetivo

Motivado, principalmente, pela demanda de tecnologia para criação de aplicações C2B, o Framework Commercepipe tem como principal objetivo dar suporte tecnológico para a criação ou instanciação de aplicações baseadas em mercados virtuais com as seguintes características:

1. Competição entre vendedores e não mais entre compradores para a viabilização de uma transação comercial;

2. Possibilidade de realizar transações comerciais por meio de dispositivos móveis, utilizando WAP como protocolo de aplicação padrão;

3. Definição do processo de negociação como automático ou semi-automático, de acordo com as características do mercado virtual, baseado na utilização de agentes de *software* como mediadores do processo de negociação;

Para atingir o objetivo proposto, uma abordagem mais formal de desenvolvimento de *software* é utilizada, por meio do desenvolvimento de um *software* com flexibilidade para instanciar aplicações distintas C2B, baseadas em mercados virtuais nos quais agentes de compra e venda interagem buscando consolidar transações comerciais.

Os pontos de flexibilização identificados para o framework Commercepipe são:

- *Item negociado* – Com o objetivo de permitir a negociação de itens diversos em diferentes mercados virtuais instanciados, o Framework Commercepipe utiliza uma abordagem na qual o processo de instanciação de uma nova aplicação permite definir as características do item a ser negociado no mercado virtual. Nesta abordagem, existe um forte acoplamento entre o item negociado e subsistemas do *backend*; porém, este acoplamento é justificável levando-se em

consideração o esforço necessário para a criação e manutenção de um gerador de instância de aplicações nesta fase embrionária na qual o próprio núcleo do *framework* está sujeito a muitas modificações*;*

- *Processo de negociação automático/semi-automático* – Diferente de mercados C2C, mercados C2B ou B2C, em alguns casos, necessitam de um processo semi-automático de negociação, no qual o usuário final, por um processo interativo, avalia e cria contrapropostas. Alguns mercados nos quais o valor do item em negociação é razoavelmente alto, o usuário final tem a necessidade de estar avaliando propostas antes de realizar contrapropostas. Neste cenário, é necessário modelar o *framework* de forma que seja possível controlar até que etapa os agentes de *software* podem automatizar o processo;

- *Negociação em múltiplas dimensões* [MCDA, 1999]– Da mesma forma que em mercados C2C nos quais a negociação em múltiplas dimensões é uma necessidade real que deve ser modelada em sistemas de *software*, mercados C2B e B2C também possuem esta característica. O consumidor final, em muitas negociações com empresas, coloca, como prioridade, o prazo de entrega ou a forma de pagamento ao invés do próprio preço do item. Desta forma, o Framework Commercepipe deve possibilitar a instanciação de mercados virtuais nos quais múltiplas dimensões [Guttman, 1998b] sejam levadas em consideração durante o processo de negociação;

- *Interface de serviços* – É claro e real o surgimento de diferentes e inovadoras possibilidades de interfaceamento do usuário final com serviços *Internet*, como a crescente utilização de celulares e PDAs na Europa e, mais recentemente, nos Estados Unidos. Desta forma, a modelagem do Framework Commercepipe preocupa-se com a necessidade de ser flexível e facilmente adaptável a novas interfaces que venham surgir com o avanço tecnológico;

- *Métodos de Filtragem* – Faz parte do *framework* a inteligência necessária para aproximar os agentes vendedores dos agentes compradores. Esta aproximação deve ser feita de forma que agentes compradores só sejam aproximados de

agentes vendedores capazes de atender todas as necessidades do consumidor. A modelagem do framework Commercepipe deve permitir que diferentes métodos de filtragem possam ser implementados e disponibilizados para uma aplicação instanciada, constituindo um ponto de flexibilização do *framework.*

4.3 A Arquitetura

Semelhante à arquitetura do Framework Vmarket, o Framework Commercepipe pode ser subdividido em dois principais macro subsistemas, o *frontend* e o *backend*.

O *frontend* define toda a interface funcional de utilização do mercado virtual. Por ele, os usuários do sistema são capazes de interagir e utilizar as principais funcionalidades de aplicações instanciadas. Sendo ponto de flexibilização do *framework,* o *frontend* está ligado a dois principais subsistemas do *backend*, o Subsistema Frontend e o Subsistema Serviços. Estes subsistemas, apresentados mais detalhadamente ainda neste capítulo, são essenciais para a instanciação de novas aplicações e devem ser customizados no processo de instanciação do *framework*. A modelagem escolhida permite que sejam utilizadas, entre outras, duas principais interfaces para utilização do mercado virtual: Web Sites (HTML) ou Wireless Sites (WML) [Nokia, 1999d]. Pela utilização de servidores de aplicação nos quais são executados *scripts* Java (JSP) [Cornell, 1998], a primeira instância do *framework*, criada como prova de conceito deste trabalho, gera, para consumidores e vendedores, respectivamente, seus *Web sites* e *Wireless sites* dinamicamente.

O *backend* implementa todos os subsistemas que compõem o mercado virtual no qual as transações comerciais são consolidadas pela mediação dos agentes de *software* de compra e venda. Trata-se de um programa desenvolvido completamente na linguagem Java [Cornell, 1998] [Grand, 1998] e executado em uma máquina virtual diferente da utilizada pelo *frontend*, permitindo que o *frontend* e o *backend* possam ser executados em máquinas

distintas. Os principais subsistemas do *backend* estão organizados de acordo com as principais entidades e papéis do sistema. São eles:

- *Agentes de Compra e Venda* – São as entidades mediadoras do processo de negociação. Agentes de compra e venda são entidades ativas criadas pelos usuários do sistema e, que por um processo de intercomunicação baseado em chamada de método, implementam o processo de negociação por meio da troca de propostas e contrapropostas;

- *Facilitação* – O Facilitador possui a inteligência para aproximar os vendedores corretos dos compradores do sistema. O Facilitador, em seus gerentes de agentes, possui o controle completo sobre como os agentes de compra e venda devem ser escalonados para execução no *backend*;

- *Usuários* – O *framework* modela, basicamente, três usuários distintos. O Consumidor é o usuário que inicia qualquer processo de negociação nas aplicações instanciadas pelo *framework*. O Vendedor, sempre associado a uma empresa, é o usuário responsável pela interação com o consumidor, buscando fechar transações comerciais. A Empresa modela a entidade responsável por vendedores no sistema, e sua modelagem no *framework* busca viabilizar uma interface de administração completa sobre todas as vendas realizadas no mercado virtual;

- *Serviços* – Os serviços representam a interface funcional do mercado virtual, utilizadas por vendedores, consumidores e empresas. Os serviços podem ser reutilizados em diferentes instâncias do *framework*;

- *Comunicação* – Toda a forma de comunicação entre o *backend* e o *frontend*, visto que estes executam em máquinas virtuais Java distintas, está definida e é realizada pelos comunicadores do *framework*. Basicamente são entidades que encapsulam funcionalidades para comunicação entre máquinas virtuais Java [Cornell, 1998] distintas pela utilização de TCP-IP [Cornell, 1998] e o mecanismo de serialização de Java [Cornell, 1998];

Três famílias de *threads* são executadas na máquina virtual do *backend*. O atentimento à requisição de serviços é feita por duas destas famílias: A *BackendCommunicator Thread* e *CommunicationHandler Thread*. A *BackendCommunicator Thread* fica, basicamente, esperando novas requisições de serviços do *frontend* e, na chegada destas, inicia *CommunicationHandler Threads* para a execução dos mesmos. A outra família é a *Marketplace Engine Thread*, responsável pela execução dos agentes de *software*. Estas três famílias compõem um ambiente altamente concorrente priorizando a criação de um mecanismo de controle de execução das mesmas para o funcionamento correto do mercado virtual.

A comunicação entre o *frontend* e o *backend* realiza-se pela utilização dos comunicadores definidos no subsistema de comunicação cuja funcionalidade é encapsulada e utilizada por JavaBeans [Cornell, 1998], definidos no subsistema Frontend. Esses *beans* são utilizados nos *scripts* JSP para a geração de páginas HTML/WML, permitindo a recuperação de informação do *backend* por meio da chamada de serviços disponíveis. A utilização da tecnologia de componentes através de *JavaBeans* neste cenário é interessante porque através de interfaces bem definidas e do encapsulamento da lógica, os *beans* podem ser utilizados de forma desacoplada no ambiente Java de programação ou em qualquer outra linguagem de programação preparada para o uso de componentes. Para uma melhor visão dos principais subsistemas do *framework*, a figura 19 mostra, em alto nível, toda a arquitetura do Commercepipe, seus subsistemas e famílias de *threads*.

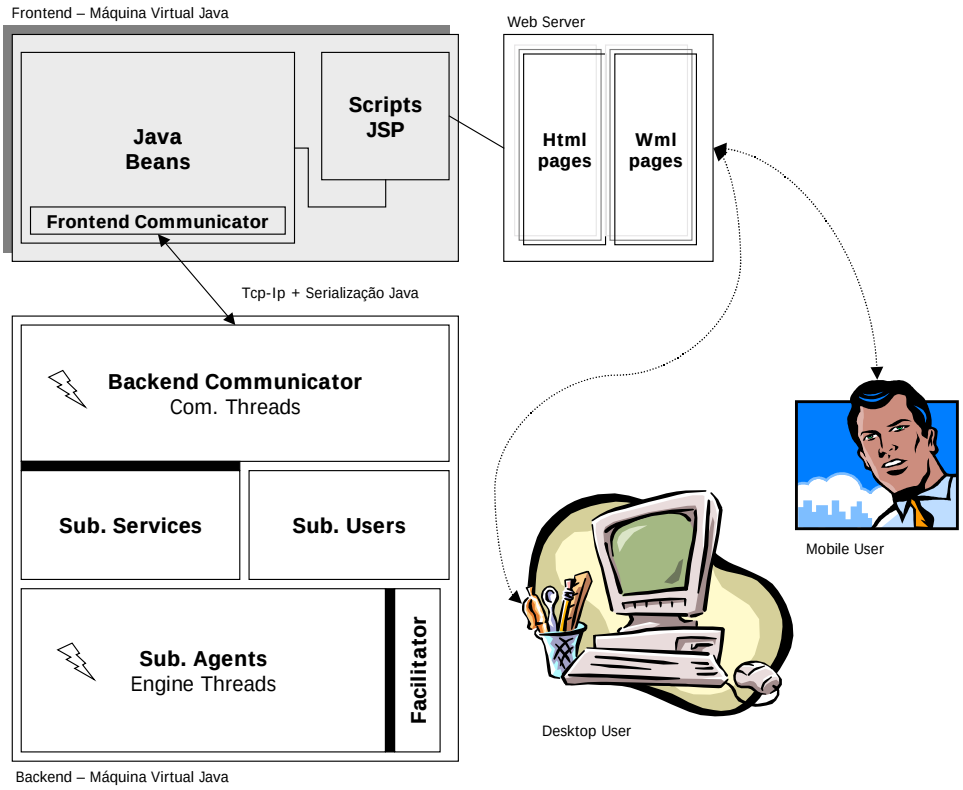


Figura 19 Visão da macro arquitetura do framework Commercepipe.

As seções seguintes apresentam, detalhadamente, os subsistemas modelados no *framework*. São apresentados os modelos UML [Conallen, 1999] [Fontoura, 1999] de cada subsistema com a apresentação das principais classes modeladas e suas funcionalidades.

4.3.1 Subsistema de Usuários

Para um melhor entendimento da arquitetura do *framework*, o subsistema de usuários é apresentado, inicialmente, definindo os principais usuários de um mercado virtual C2B. Mercados virtuais C2B possuem, basicamente, três atores principais: Consumidores, Vendedores e Empresas. A modelagem utilizada no Framework Commercepipe é apresentada na figura 20 em um diagrama de classes UML:

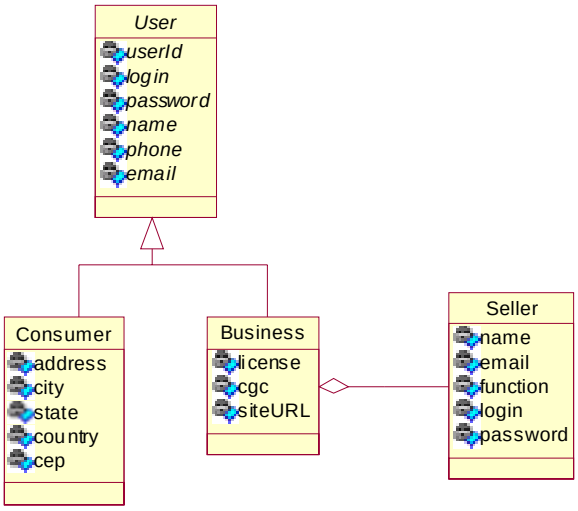


Figura 20 Diagrama UML – Modelagem Usuários

- *Consumer* – São usuários compradores de produtos em mercados virtuais. A iniciação de uma nova negociação sempre parte de consumidores pelo cadastramento de uma nova necessidade de consumo;
- *Business* – São as empresas responsáveis, indiretamente, pela venda de produtos nos mercados virtuais e pela administração de todas as operações realizadas no mercado virtual. As vendas em si são realizadas por vendedores autorizados pelas mesmas para atuação no mercado virtual;
- *Seller* – Responsáveis diretos por qualquer venda realizada no mercado virtual. Obrigatoriamente, qualquer vendedor deve estar associado a uma, e somente uma, empresa registrada no mercado virtual.

A abordagem utilizada pelo *framework* define como usuários do sistema Consumidores e Empresas, sendo Consumer e Business classes concretas de User. Apesar dos vendedores associados a empresas serem os participantes ativos no processo de negociação, desta forma, uma aplicação instanciada pelo *framework* só permite que vendedores atuem no mercado se estes estiverem associados a empresas previamente cadastradas e autorizadas a operar no mercado virtual.

Outra classe fundamental neste subsistema é o UserManager. O UserManager é a entidade responsável pelo gerenciamento da coleção de usuários, provendo funcionalidades como criação e exclusão de usuários, retorno de referências de usuários do sistema para uso de outros subsistemas do *backend*.

A modelagem do subsistema de usuários permite que novos tipos de usuários de mercados virtuais C2B possam ser incorporados ao *framework*, simplesmente por meio da implementação de novas classes que herdem diretamente de User, sem a necessidade de modificações na classe UserManager, facilitando o processo de manutenção e evolução do *framework*.

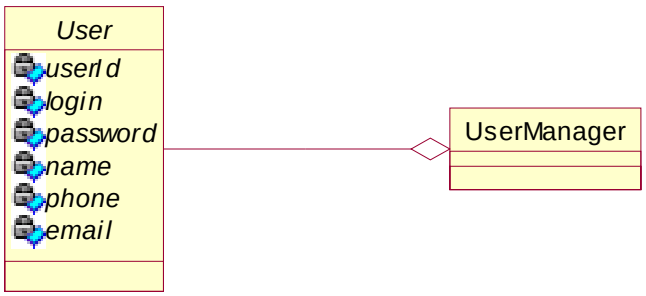


Figura 21 Diagrama UML – UserManager

Este subsistema implementa ainda uma interface Java [Cornell, 1998] que define o protocolo a ser seguido para que um usuário possa realizar negociações no mercado virtual. Toda classe que estenda User e implemente a interface Trader deve implementar o método trade(). No modelo atual do *framework*, Consumers e Sellers são as entidades ativas no processo de negociação e implementam o método trade().

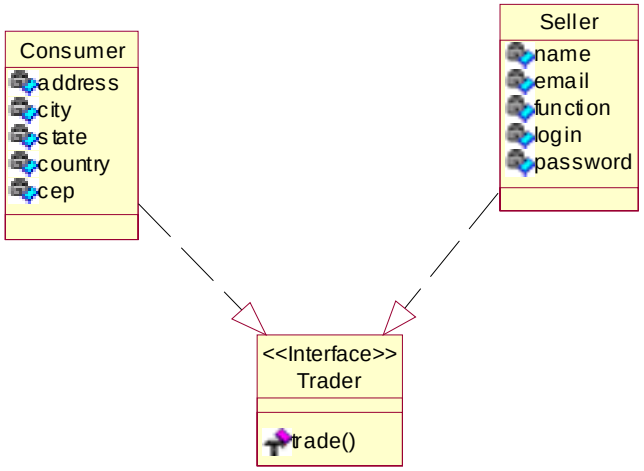


Figura 22 Interface Trader

A implementação do método trade() é dependente da escolha entre negociação automática ou semi-automática para uma aplicação a ser instanciada pelo *framework*. No caso de negociação semi-automática, a implementação para Consumers e Sellers é a mesma, visto que os usuários finais são os responsáveis pela criação de propostas e contrapropostas. No caso de negociação automática, a implementação de trade() para Sellers é diferente, uma vez que, agora, a criação de contrapropostas é feita diretamente pelo agente de venda. O quadro, abaixo, exemplifica a implementação de trade() em ambas as situações:

	Negociação Semi Automática	Negociação Automática

Consumer	Public void
	trade(commercePipe.agent.TradeMessage m) {
	
	agent.putOutBoxNewTradeMessage(m);
	}
		public void
		trade(commercePipe.agent.TradeMessage m) {
		
		agent.putOutBoxNewTradeMessage(m);
		}

Seller	public void
	trade(commercePipe.agent.TradeMessage m) {
	
	agent.putOutBoxNewTradeMessage(m);
	}
		public void
		trade(commercePipe.agent.TradeMessage m) {
		
		//No code necessary
		}

Tabela 4 Situações de implementação de trade()

Observa-se que, na negociação automática, não é necessário codificar o corpo do método trade() do objeto Seller, visto que a criação e o envio de uma contraproposta serão feitos diretamente pelo agente de venda. O objeto Consumer, por sua vez, ainda necessita da implementação do método, pois a primeira proposta de consumo é criada pelo usuário.

O diagrama de interação na figura 23 mostra o contexto de execução do método trade() para aplicações com negociação semi-automática.

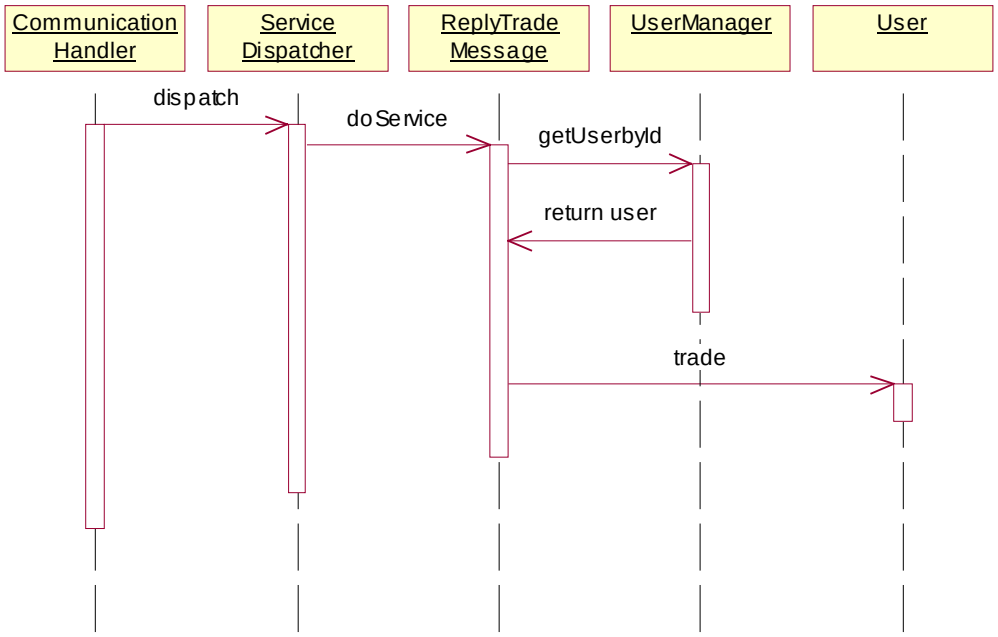


Figura 23 Diagrama de interação - Método trade()

4.3.2 Subsistema de Agentes

O subsistema de Agentes pode ser considerado um dos principais subsistemas do *framework*. Este subsistema modela os agentes de *software*, ou "empregados digitais", que realizam todo o processo de negociação entre compradores e vendedores.

Na etapa de análise, foram identificados alguns principais pré-requisitos que deveriam ser modelados neste subsistema e que encontram-se descritos abaixo:

1. *Facilidade de variação no comportamento dos agentes* – Deve ser possível alterar o comportamento dos agentes de forma simples, sem a necessidade de alterações no núcleo do *software*;
2. *Protocolo de negociação*– A opção de escolha no grau de liberdade na negociação deve existir e ser feita através de uma customização simples e bem definida;
3. *Suporte a diferentes itens* – O subsistema deve dar a facilidade de instanciação de aplicações para itens diversos, permitindo a criação de mercados virtuais diversos.

Desta forma, a modelagem do subsistema de agentes é composta por 17 classes, e o diagrama UML completo do subsistema é mostrado logo adiante:

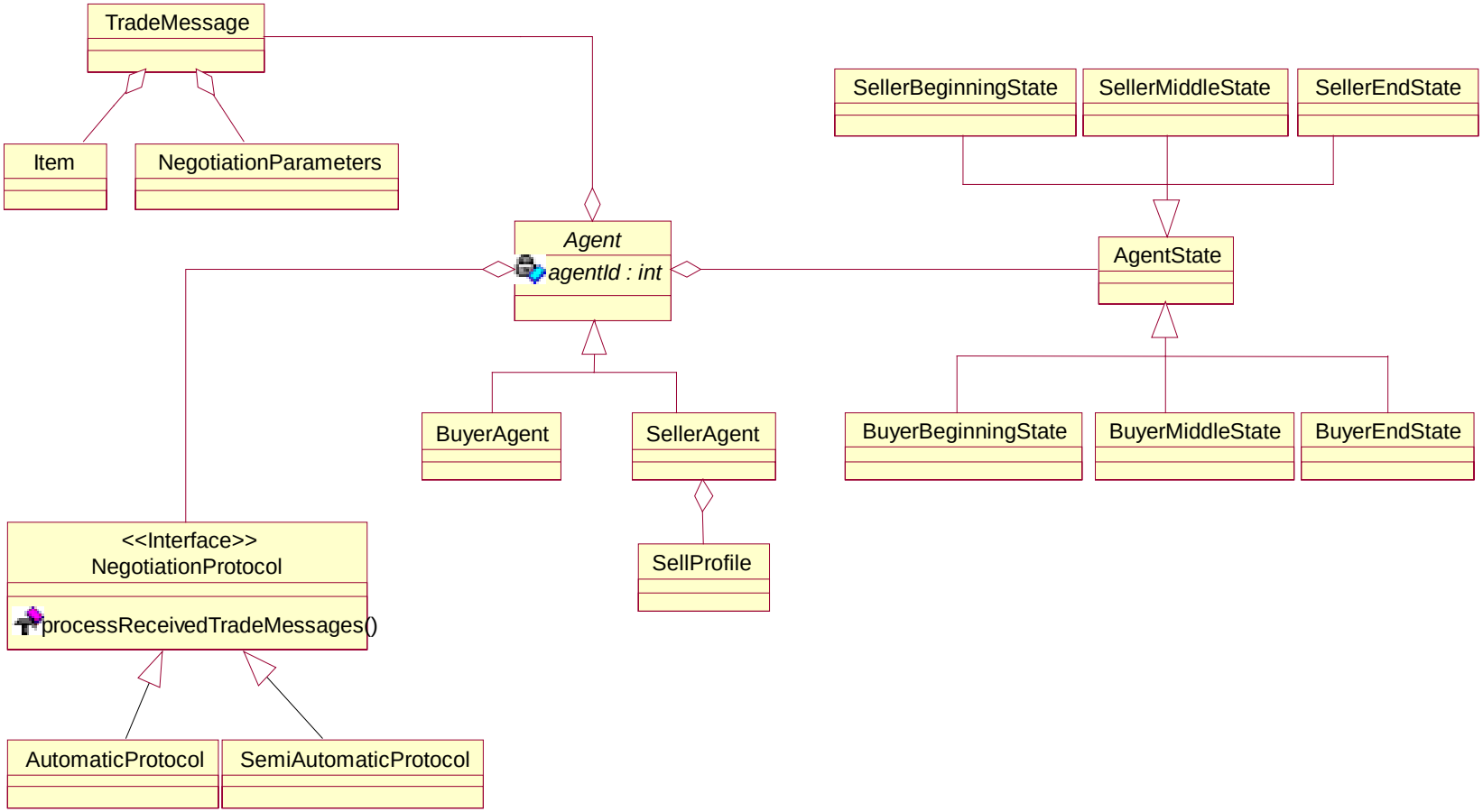


Figura 24 Diagrama UML – Subsistema de Agentes

Os agentes de *software* são modelados por uma classe abstrata Agent, a partir da qual classes concretas podem ser implementadas. Na implementação atual do *framework*, existem duas classes concretas de agentes: BuyerAgent e SellerAgent. Os agentes possuem os seguintes atributos:

- *AgentId* – Identifica unicamente o agente de compra/venda no mercado virtual, sendo utilizado como identidade no sistema;

- *AgentName* – Define um nome ou apelido permitindo um melhor mecanismo de associação desta entidade na interface para o usuário final;

- *State* – Define o estado corrente de execução do agente. O comportamento dos agentes de compra e venda do sistema é definido por uma máquina de estados modelada pelo *pattern* State [Gamma, 1995]. A figura 25 mostra a máquina de estados decorrente da modelagem atual para o *pattern* State.

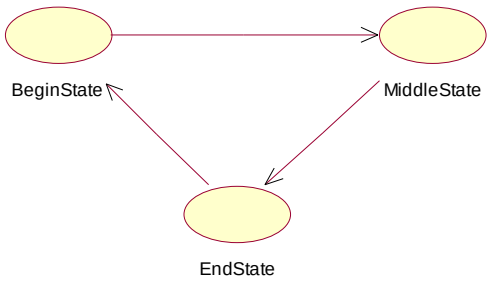


Figura 25 Máquina de estados dos agentes de compra e venda

Esta modelagem permite não só que o comportamento dos agentes de *software* fique mais visível e melhor definido, mas também permite que seja facilmente extensível e customizável com a possibilidade de criação de sub-estados para instanciação de novas aplicações. A figura 26 exemplifica a criação de sub-estados.

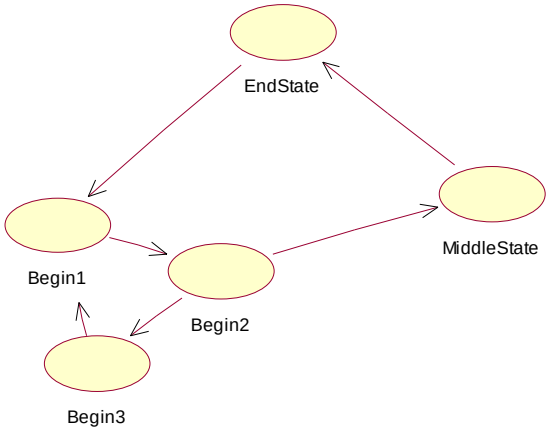


Figura 26 Customização do comportamento dos agentes

- *Inbox* – Modela a caixa postal de entrada dos agentes. Nela serão armazenadas as mensagens de negociação modeladas pela classe TradeMessage;

- *Outbox* – Modela a caixa postal de saída dos agentes. Da mesma forma que a Inbox, são armazenadas TradeMessages, enviadas para outros agentes, durante o processo de negociação;

- *NegotiationProtocol* – Define o grau de autonomia de negociação dos agentes para uma aplicação instanciada utilizando-se o *framework*. Na modelagem atual, estes podem ser automático ou semi-automáticos, como mostra a figura 27:

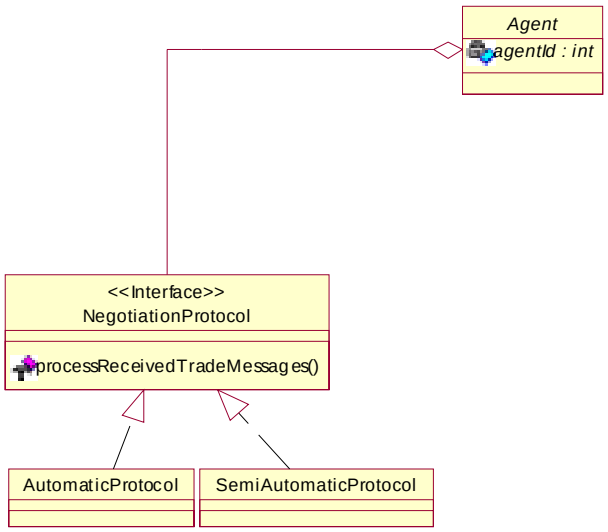


Figura 27 Protocolo de negociação

A modelagem define a interface NegotiationProtocol. Classes concretas como AutomaticProtocol e SemiAutomaticProtocol devem implementar o método processReceivedTradeMessages(), o qual deve conter a semântica de como tratar TradeMessages recebidas. Em outras palavras, processReceivedTradeMessages() define a estratégia de negociação para criação de contrapropostas. Para a escolha de que processo de negociação deve ser utilizado em uma instanciação do *framework*, é necessárias somente uma tarefa.

1. *Modificação do construtor de BuyerAgent/SellerAgent* – No momento de instanciação de um agente, define-se que protocolo de negociação deve ser utilizado. A modificação necessária é a simples alteração da linha de código setNegProtocol(new SemiAutomaticProtocol()), conforme desejado.

Isto é possível, pois, em algum momento, os agentes de *software* chamam getAgent().getNegProtocol().processReceivedTradeMessages(), permitindo que, neste ponto, a criação de contrapropostas ocorra, seja ela automática ou não.

Representando a unidade comunicação no processo de negociação, as TradeMessages contêm toda a informação necessária para a avaliação de uma proposta de negociação.

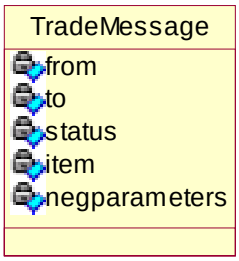


Figura 28 Classe TradeMessage

Os atributos *from* e *to* definem o remetente e o destinatário da mensagem. São do tipo Agent e contêm a referência para os mesmos no mercado virtual. O *status* de uma TradeMessage pode ser um dos listados abaixo:

- *NEW* – Mensagem nova ainda não enviada;
- *READ* – Mensagem recebida e lida;
- *SENT* – Mensagem enviada.

Os principais atributos de uma TradeMessage são os que contêm a informação sobre a negociação em si. O atributo item, modelado pela classe Item, contém todos os atributos que descrevem o item em negociação. O atributo negParameters, modelado pela classe NegotiationParameters, contém todos os atributos que descrevem a proposta para a negociação do item.

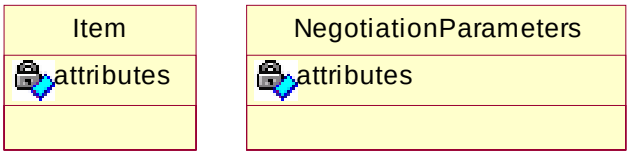


Figura 29 O Item e os parâmetros de negociação

Supondo um mercado de carros, os atributos de Item poderiam ser Modelo, Ano, Marca, Portas, e os atributos de NegParameters poderiam ser Preço, Forma de Pagamento, Prazo de entrega. Esta modelagem define toda a informação necessária para a avaliação de uma proposta de compra ou venda e satisfaz a necessidade de mercados C2B. O diagrama de interação mostrado na figura 30, mostra o ciclo de execução dos agentes de software de compra e venda no sistema.

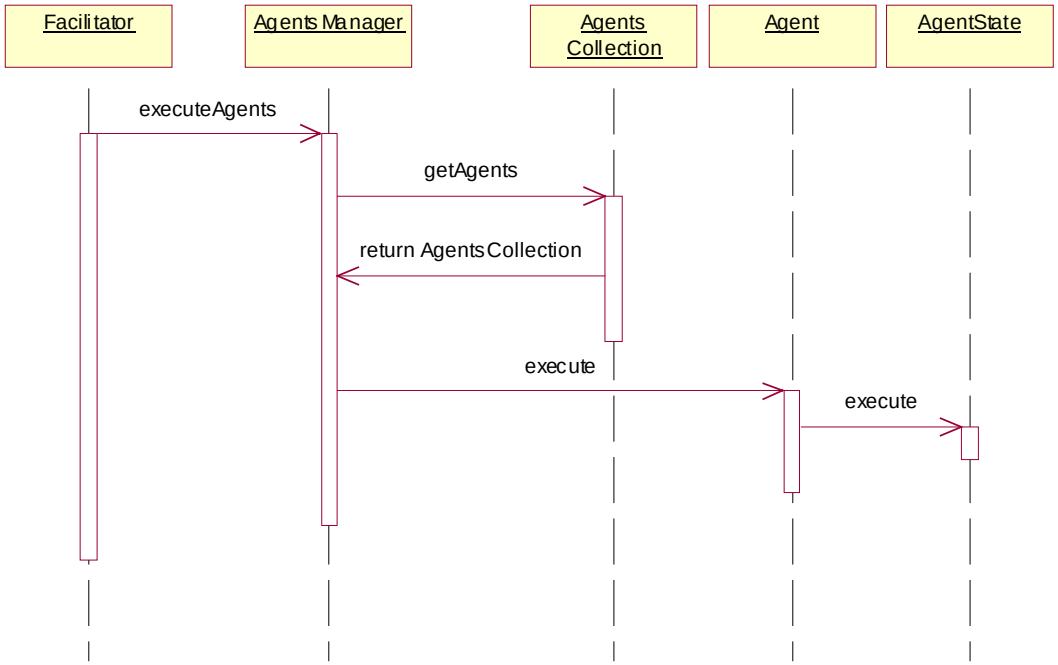


Figura 30 Diagrama de Interação – Execução dos Agentes de Software

Por fim, é necessário que agentes vendedores expressem sua capacidade de venda, ou seja, definam para que itens ou subconjunto destes itens, estão aptos a negociar. Esta informação é absolutamente necessária para que agentes de venda adequados sejam aproximados de agentes de compra. A classe SellProfile modela esta informação e, como veremos no subsistema de facilitação, é necessária para os mecanismos de filtragem utilizados para a aproximação eficiente de agentes de compra e venda.

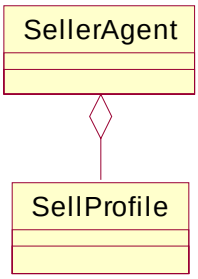


Figura 31 Modelagem Perfil de venda

4.3.3 Subsistema de Facilitação

O subsistema de facilitação é bastante simples e é modelado por seis classes principais. Responsável, basicamente, por papéis de gerência no mercado virtual, este subsistema possui duas principais funções:

1. *Gerência das coleções de agentes* – O papel de gerência dos agentes é modelado pela classe abstrata AgentsManager, que permite a criação de novos gerentes de agentes caso surjam novos tipos de agentes no *framework*. Este constitui mais um ponto de extensão do *framework*, e a modelagem atual contempla duas classes concretas BuyerAgentsManager e SellerAgentsManager, visto que os tipos concretos de agentes disponíveis no *framework*, na versão atual, são dois: BuyerAgent e SellerAgent.

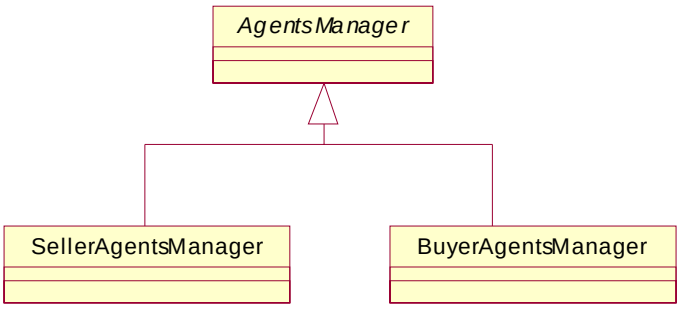


Figura 32 Gerentes de Agentes

Os AgentsManagers são responsáveis pela gerência da coleção dos agentes concretos do *framework*, criando, identificando, removendo agentes do mercado virtual. Os AgentManagers são responsáveis também pela execução dos agentes, escalonando a coleção. O método executeAgents() define a política de escalonamento de cada gerente. A implementação atual utiliza um *loop* simples, iterando sobre toda a coleção permitindo que cada agente execute de acordo com sua máquina de estados corrente;

2. *Aproximação de compradores e vendedores* – O facilitador modelado pela classe Facilitator é responsável pela aproximação de agentes compradores e vendedores.

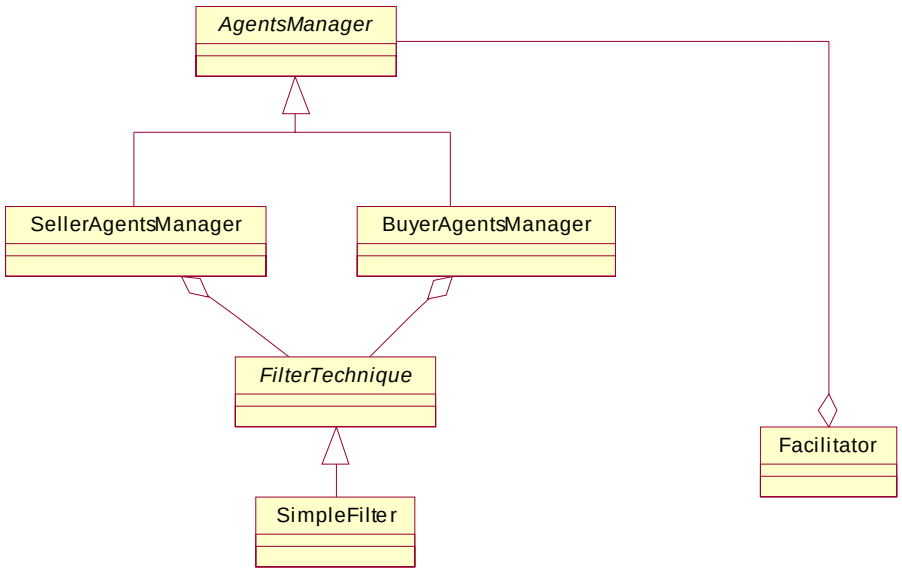


Figura 33 O Facilitador

Pelos AgentsManagers, o Facilitator é capaz de localizar os vendedores corretos para compradores. O que é feito utilizando-se uma classe concreta de FilterTechnique, responsável por filtrar agentes de venda de acordo com seu perfil modelado por SellProfile. A técnica de filtragem concreta disponível no *framework* é modelada pela classe SimpleFilter.

Ainda neste subsistema, encontram-se o modelo de controle de transação do mercado virtual. Sendo um ambiente com três famílias de *threads* bem definidas, foi necessária a

modelagem de um controle de transação evitando que estados inconsistentes aconteçam pela ausência de controle na execução das *threads*. A classe abstrata TransactionManager define três métodos principais para o controle de transação (*readLock*, *writeLock*, *done*), os quais devem ser implementados por classes concretas estendidas. A classe concreta SingleThreadExecution [Grand, 1998] está implementada e disponível no *framework* e implementa os métodos de controle de transação da seguinte forma:

- *ReadLock* e *WriteLock* – Chamando este método, uma *thread* ganha exclusividade de execução na máquina virtual, caso não exista outra *thread* com exclusividade para leitura ou escrita. Caso exista, ela é bloqueada e entra em uma fila de espera até que seja possível sua execução;
- *Done* – Chamando este método, a *thread* dá como concluída a execução na área de exclusão, permitindo que outras *threads* executem.

O controle de transação deve ser utilizado em toda a área de exclusão do *framework*, ou seja, onde o estado de objetos está sujeito a inconsistências. A identificação destas áreas não é uma tarefa trivial e requer uma análise detalhada sobre a linha de execução das *threads* do mercado virtual.

4.3.4 Subsistema de Serviços

O subsistema de serviços é, por inteiro, um ponto de flexibilização do *framework*. Por este motivo o número de classes modeladas é completamente variável, sendo definido de acordo com a aplicação instanciada. Este subsistema está fortemente acoplado à interface funcional com os usuários do sistema, no sentido em que novos serviços podem ser criados para uma aplicação do *framework* por meio da implementação de uma nova classe concreta de Service. A classe abstrata Service modela os serviços do mercado virtual.

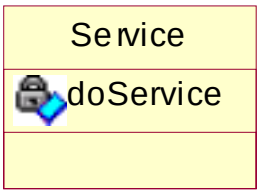


Figura 34 Classe Abstrata Service

Classes concretas de Service implementam serviços que podem ser reutilizados para novas instâncias do *framework*. Outra importante vantagem desta modelagem é que os serviços disponibilizados neste subsistema estão totalmente desacoplados da tecnologia utilizada para a implementação da interface da aplicação, requerendo somente que esta seja capaz de referenciar e utilizar objetos Java. Desta forma, os serviços estão disponíveis para interfaces diversas como WebSites em HTML ou Wireless Sites em WML. Um conjunto de serviços encontra-se implementado no *framework*. São eles:

- *CreateBusinessService* – Cria novas empresas no mercado virtual;
- *CreateConsumerService* – Cria novos Consumidores no mercado virtual;
- *CreateNewTradeMessage* – Cria TradeMessage inicial do processo de negociação. Consumidores são os utilizadores deste serviço;
- *CreateSellerService* – Cria novo vendedor para uma empresa já cadastrada no mercado virtual;
- *ListInboxService* – Retorna as TradeMessages da caixa de entrada dos agentes de compra ou venda
- *ListOutboxService* – Retorna as TradeMessages da caixa de saída dos agentes de compra ou venda;
- *LoginService* – Autoriza Consumidores e Empresas a utilizarem o sistema;
- *LoginSellerService* – Autoriza Vendedores a utilizarem o sistema;
- *ReplyTradeMessageService* – Cria contraproposta e envia para agente destinatário;

A criação de novos serviços requer a implementação de novas classes concretas de Service com a implementação do método doService. Para que o novo serviço esteja disponível em novas instâncias do mercado virtual, é necessário que a classe ServiceDispatcher do subsistema de comunicação saiba de sua existência. Para isso, é necessário que o construtor de ServiceDispatcher seja alterado, acrescentando-se a linha de código dispatchTable.put("NEW_SERVICE_HEADER", new ConcreteService()).

O diagrama de interação mostrado na figura 35 identifica o funcionamento básico das aplicações instanciadas pelo framework, desde a execução de serviços iniciada pelos usuários do sistema.

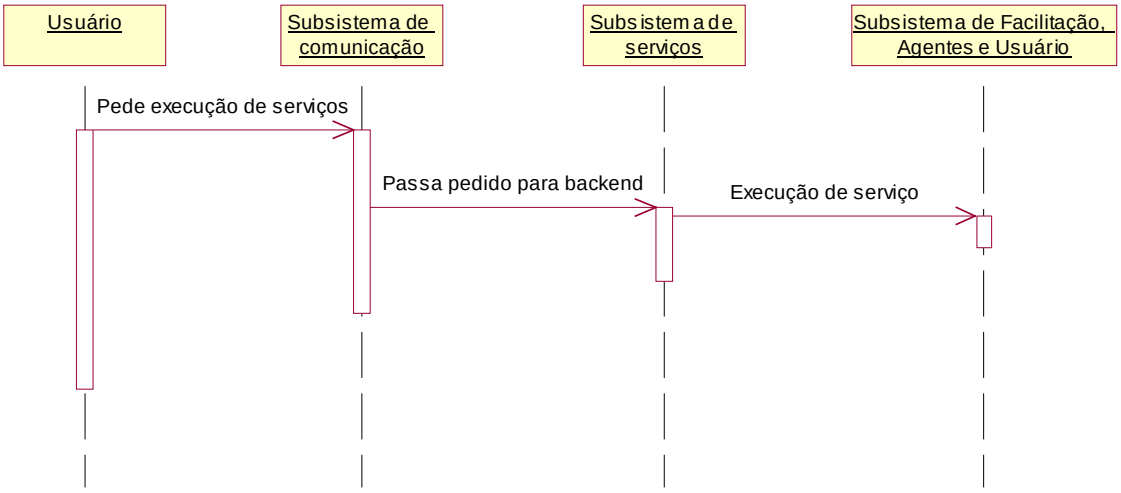


Figura 35 Diagrama de interação – Execução de serviços

Maiores detalhes sobre a utilização de serviços do mercado virtual estão descritas no Subsistema de Comunicação.

4.3.5 Subsistema de Comunicação

O subsistema de comunicação é o responsável por toda a comunicação do *frontend* com o *backend*, sendo, assim, um dos principais subsistemas do *framework*. A modelagem permite o acesso de múltiplos usuários concorrentes. As sete classes do modelo do subsistema são mostradas a seguir.

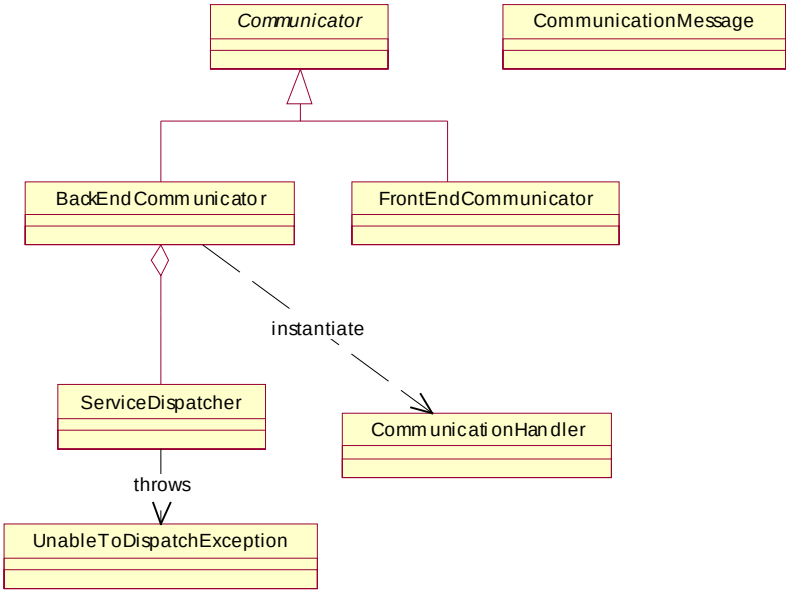


Figura 36 Subsistema de Comunicação

Toda a comunicação é realizada por TCP/IP em conjunto com o mecanismo de serialização de objetos implementado pela máquina virtual Java [Cornell, 1998]. Esta combinação permite que objetos ou coleções de objetos do *backend* sejam serializados e transportados pela rede para outra máquina virtual, no caso, a máquina virtual na qual o *frontend* é executado. Este mecanismo é extremamente poderoso, desacoplando completamente o *frontend* do *backend*;

As classes FrontendCommunicator e BackendCommunicator modelam os objetos responsáveis pela comunicação. Sendo elas classes concretas de Communicator, a comunicação entre eles acontece pela definição do IP e da porta da máquina na qual o *backend* será executado.

O método sendMsg() do FrontendCommunicator permite a chamada de um serviço do *backend* pelo *frontend*. Na implementação atual, foi utilizado JSP (*Java Server Pages*) (http://java.sun.com/products/jsp) para a implementação do *frontend*, e a utilização do FrontendCommunicator acontece de forma encapsulada, pelo uso de *JavaBeans*. Estes *beans* encapsulam em um objeto Java as informações necessárias para a chamada de um serviço do *backend*, além das informações disponibilizadas após a execução do serviço. A figura 37, demonstra todo o processo de chamada de um serviço do *backend*, apontando as classes utilizadas em cada etapa.

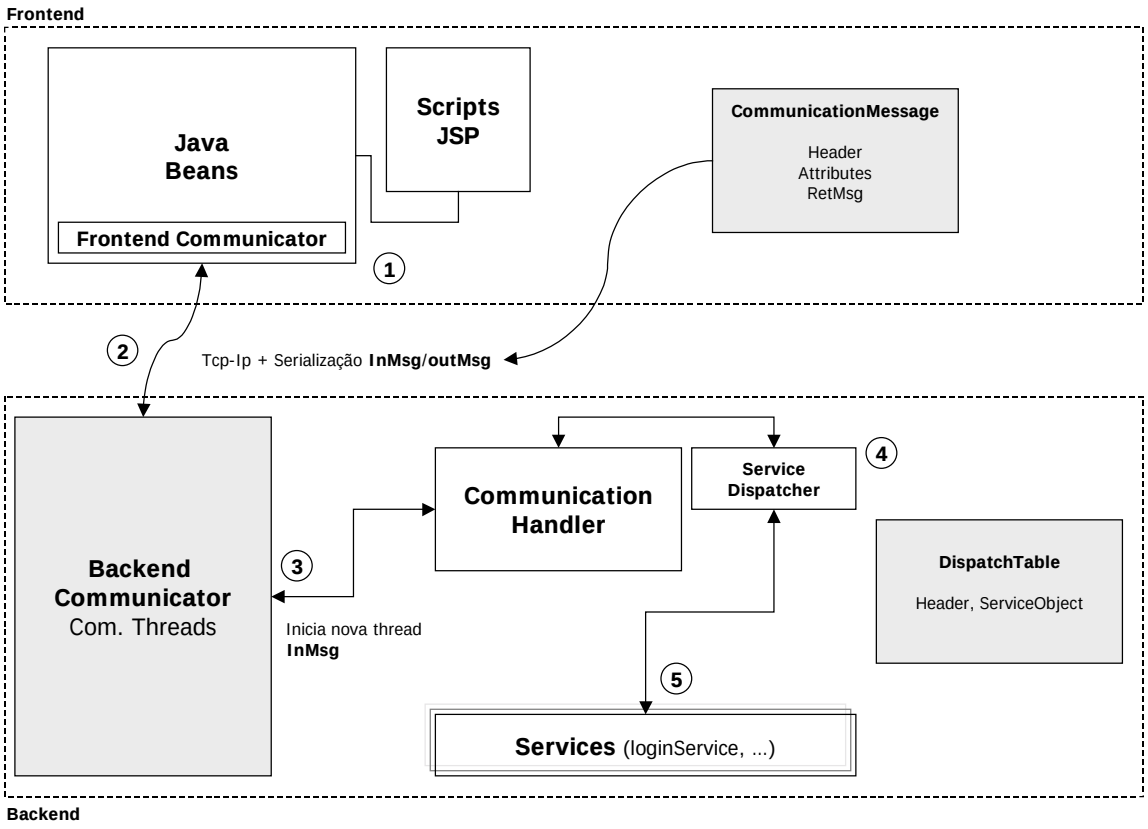


Figura 37 Chamada de serviço do backend: (1) FrontendCommunicator prepara InMsg; (2) InMsg é serializada e enviada via tcp-ip; (3) Criação de nova thread de atendimento a serviços em CommunicationHandler; (4) ServiceDispatcher delega execução ao serviço correto; (5) Serviço é finalmente executado.

Neste processo, o FrontendCommunicator é o primeiro a entrar em ação após estarem disponíveis, na inMsg, todos os dados necessários para a execução do serviço. A inMsg é

serializada e enviada para o *backend*, no qual o BackendCommunicator atende, inicialmente, à requisição de um novo serviço criando uma nova *thread* de execução em CommunicationHandler a qual será responsável pela execução do serviço. O CommunicationHandler chama, então, a execução do ServiceDispatcher, responsável por identificar qual serviço deve ser executado, utilizando-se o atributo *header* da inMsg, ele indexa uma *hashtable* que contém os serviços, fazendo com que o serviço seja executado. Os dados retornados da execução do serviço são encapsulados em outMsg, serializados e enviados para o *frontend*, no qual estarão, finalmente, encapsulados e disponíveis no *bean* para uso da interface com o usuário.

A modelagem utilizada na classe ServiceDispatcher permite que um novo serviço esteja disponível no *backend* simplesmente por meio da introdução de uma linha de código à qual adiciona o novo serviço na dispatchTable do ServiceDispatcher.

O BackendCommunicator possui ainda algumas funcionalidades de controle de utilização do mercado virtual, como número máximo de *threads* de execução de serviços e número de conexões recusadas por ultrapassar o limite máximo de *handlers*.

A classe UnabletoDispatchException é utilizada para alertar se o serviço não pode ser executado, indicando, por exemplo, que este serviço não foi adicionado a dispatchTable do ServiceDispatcher.

4.3.6 Tecnologias utilizadas

Foram utilizadas tecnologias diversas no curso de desenvolvimento deste projeto. O objetivo desta seção é a apresentação compacta das tecnologias escolhidas e dos critérios utilizados para esta escolha.

A tabela 5, abaixo, apresenta, de forma compacta, os pontos de escolha de tecnologia, as opções consideradas e a justificativa pela escolha.

Pontos	Opções consideradas	Escolha	Critérios
Linguagem de programação	Java C++	Java	100% Orientada a Objetos Portabilidade (Sistema em evolução)
Banco de dados	Relacional OO	Relacional (DB2)	Integração com ferramenta de desenvolvimento Menor inércia de aprendizado
Servidor de Aplicação Frontend	IBM Websphere TomCat	TomCat	Executa JSP 1.1 Menor consumo de recursos Freeware
Linguagem Frontend	JSP ASP	JSP	Ambiente Java Integração 100% backend Integração com ferramenta de desenvolvimento
Ferramenta de desenvolvimento	MS Visual J++ IBM VisualAge for Java	IBM VisualAge for Java	Ambiente de desenvolvimento superior Integração com desenvolvimento frontend e banco de dados

Tabela 5 Tecnologias utilizadas no projeto

A familiarização com algumas destas tecnologias já utilizadas em outros projetos do Laboratório também contribuiu para a determinação das escolhas realizadas, levando-se em consideração, principalmente, o prazo para a realização do trabalho. Algumas observações interessantes podem ser feitas analisando a tabela 5:

- A linguagem Java mostrou ser a mais adequada para o desenvolvimento do *framework*, principalmente por ser 100% orientada a objetos. Apesar de ser uma linguagem interpretada, o que, à princípio, poderia levantar dúvidas em relação a seu desempenho, Java possibilita a criação de um ambiente adequado para a pesquisa de *software*, permitindo que este seja desenvolvido e testado em diferentes plataformas, promovendo um nível de agilidade ao processo que C++ não permitiria;

- A escolha pelo modelo relacional de banco de dados deu-se, principalmente, por ter uma menor barreira de aprendizado, visto que esta escolha deve facilitar, da melhor forma possível, o processo de desenvolvimento e evolução do *framework*, foco deste trabalho de pesquisa;

- A linguagem de programação escolhida para o *frontend* também foi Java, utilizando-se JSPs (Java Server Pages). Esta escolha foi fundamental para permitir uma maior integração com o *backend*, trazendo vantagens como o completo desacoplamento entre os dois macro subsistemas;

- Como ferramenta de desenvolvimento, o VisualAge for Java mostrou ser o melhor ambiente de desenvolvimento de aplicações para Java. Além de uma excelente interface gráfica, a ferramenta integra o desenvolvimento do *frontend* (JSPs) e da Persistência, fazendo-se uso de WebSphere Studio e do Persistent Builder, respectivamente.

Todo o desenvolvimento do *framework* foi feito na plataforma Windows NT 4.0 , com IIS 4.0 como WebServer, TomCat 3.0 como Servidor de Aplicação, WebSphere como servidor de aplicação para o desenvolvimento e DB2 Enterprise como SGBD.

Capítulo 5

Instanciação do Framework

Este capítulo apresenta o processo de instanciação de aplicações a partir do *framework* CommercePipe. Na versão atual, o processo de instanciação envolve a customização de pontos de flexibilização do sistema por meio da implementação de trechos de código Java, da escolha de classes concretas já implementadas e da construção de interfaces funcionais para o usuário final. O processo de instanciação de *frameworks*, em geral, pode ser automatizado utilizando-se ferramentas desenvolvidas sob medida para este fim como, por exemplo, as linguagens de domínio específico (DSLs) que permitam a construção de *softwares* capazes de gerar código e configurar pontos do sistema de forma automática. Apesar deste ser um processo viável, ele ainda não foi utilizado no Framework CommercePipe, principalmente devido à grande possibilidade de mudanças no núcleo do sistema.

Uma primeira instância do *framework* foi criada neste trabalho de pesquisa a fim de verificar sua viabilidade como instanciador de aplicações. A aplicação AutoPipe implementa um mercado virtual de carros no qual consumidores, através de um *Website*, podem cadastrar intenções de compra de carros, e vendedores, por meio de dispositivos móveis como telefones celulares e PDAs, podem criar e enviar contrapropostas.

As diversas etapas de customização de pontos de flexibilização serão descritas, neste capítulo, utilizando-se a aplicação AutoPipe como exemplo. Ao longo de todo o capítulo, será utilizada a figura do Instanciador fazendo o papel do profissional responsável pela instanciação de aplicações a partir do *framework*.

5.1 Implementação dos Pontos de Flexibilização

O processo de instanciação de aplicações do *framework* consiste, basicamente, na customização de pontos de flexibilização. A tabela 6, abaixo, resume os passos de customização que devem ser realizados para instanciação de novas aplicações.

Pontos de Flexibilização	Subsistema	Características
Item	Sub. de Agentes	*Possibilitar que sejam criados mercados virtuais para itens diversos*
NegotiationParameters	Sub. de Agentes	*Flexibilidade para defnição dos parâmetros de negociação como prazo, preço, etc...*
Agent States	Sub. de Agentes	*Mudança de comportamento dos agentes de software de acordo com o mercado virtual*
NegotiationProtocol	Sub. de Agentes	*Possibilidade de escolha entre processos de negociação automático ou semi-automático*
FilterTechnique	Sub. Facilitator	*Permitir que algorítmos diversos de matching possam ser utilizados para aproximar agentes de compra e venda*
Services	Sub. Services	*Customização de serviços para os usuários do sistema*

Tabela 6 Pontos de Flexibilização do Framework

Este conjunto de pontos de flexibilização possui interdependência, porque a implementação de alguns *pontos de flexibilização* influenciam na implementação de outros, como mostra o diagrama abaixo:

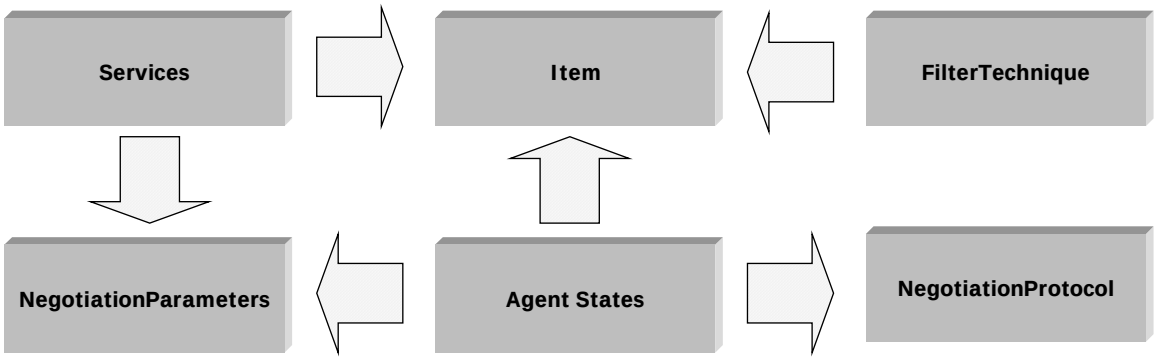


Tabela 7 Interdependência entre pontos de flexibilização

O diagrama mostra os acoplamentos existentes entre os pontos de flexibilização do *framework*, identificando a existência de uma ordem na customização dos pontos de flexibilização. Primeiramente, *Item*, *NegotiationParameters* e *NegotiationProtocol* devem ser customizados, pois não possuem dependência alguma em relação a outros pontos de flexibilização. A seguir, *FilterTechnique*, *Agent States* e *Services*, nesta ordem, podem ser customizados levando-se em consideração as decisões tomadas na customização de *Item*, *NegotiationParameters* e *NegotiationProtocol*.

5.2 Item

A definição do item a ser negociado no mercado virtual é um dos mais importantes pontos de flexibilização do sistema. Desacoplar as características do item do mercado virtual facilita a criação de mercados C2B diversos, permitindo que o *framework* seja aplicável a um grande número de mercados.

Este ponto de flexibilização envolve, principalmente, a customização da classe *Item*. A classes *Item*, mostrada na figura 38, é bastante simples no que diz respeito à sua implementação, e possui, basicamente, dois principais atributos:

Figura 38 Classe Item

- *ItemId* – É utilizado como identificador do item no sistema;

- *Attributes* – Este atributo consiste em uma Hashtable Java e, nela, devem estar os atributos definidos para o item a ser negociado no mercado. Toda a recuperação de informação sobre os atributos dos itens do mercado é feita pelo método getAttribute(String key), que recebe como parâmetro o identificador do atributo que é utilizado como chave única na hashtable attributes para recuperação de seu valor.

A customização da classe *Item* consiste na definição dos atributos do item a serem negociado no mercado e, como sugestão, esta definição pode ser realizada em duas etapas:

1. *Necessidades reais do mercado* – O Instanciador deve levar em consideração o estudos das necessidades reais do mercado para definir quais são os principais atributos a serem modelados na aplicação para a representação do *Item*;

2. *Definição dos Identificadores dos atributos* – Uma vez definido o conjunto de atributos, o Instanciador deve definir identificadores para eles. Os identificadores são *Strings* utilizadas como chaves na *Hashtable attributes*, e que definem no escopo de todo o sistema, como devem ser realizados o acesso ou a modificação de qualquer atributo de um *Item*.

Na aplicação AutoPipe, foram definidos os seguintes atributos para o Item Carro:

Identificador	Descrição
Brand	Marca do fabricante do carro
Model	Modelo do carro

Tabela 8 Atributos do Item Carro na aplicação AutoPipe

A customização de outros pontos de flexibilização do *framework* estão acoplados a Customização de *Item* por utilizarem os atributos dos itens do mercado. São eles:

1. *Filter Technique* – Tem acesso aos atributos de Itens para fazer o *matching* com agentes de venda do mercado virtual;

2. *Services* – Alguns serviços da aplicação estão fortemente acoplados à definição destes atributos, como o cadastramento de novas intenções de consumo (*TradeMessages*);

3. *Agent States* – Para aplicações com o processo de negociação automático, os agentes do sistema utilizam o Item e os Parâmetros de Negociação para construírem contrapropostas de forma automática. Este processamento ocorre na implementação do método *processReceivedTradeMessages*().

5.3 NegotiationParameters

A classe *NegotiationParameters* modela os atributos relacionados à negociação. A definição destes atributos, em conjunto com a definição dos atributos de Item, define o conteúdo principal das instâncias da classe *TradeMessage*, modelada como a unidade de informação utilizada em todo o processo de negociação.

A sugestão para customizar este ponto de flexibilização é idêntica à apresentada para Itens, envolvendo também a importante etapa de análise das necessidades reais de mercado para a negociação do item.

Da mesma forma que o ponto de flexibilização Item, os parâmetros de negociação possuem forte acoplamento com os agentes de compra e venda na implementação do método *processReceivedTradeMessages*() para mercados com negociação automática, nos quais estes atributos são acionados para a criação de contrapropostas.

Na aplicação AutoPipe, foram definidos os parâmetros de negociação indicados na tabela 9:

Identificador	Descrição
Price	Preço do Item
Obs	Observação genérica

Tabela 9 Parâmetros de Negociação na aplicação AutoPipe

Apesar de ser um conjunto simples de atributos no qual somente o preço do item é utilizado na negociação, a modelagem de um atributo genérico, como *Obs*, permite, em um mercado com negociação semi-automática, que o usuário consumidor defina algumas preferências não previstas como, por exemplo, prazo de entrega e forma de pagamento, que podem estar sendo avaliadas por vendedores e influenciando, eventualmente, na definição de contrapropostas.

5.4 Agent States

A máquina de estado dos agentes de compra e venda foi modelada utilizando-se o *pattern State* [Gamma, 1995], principalmente para permitir que a extensão ou variação do comportamento dos agentes fosse feita de forma simples e estruturada. Tornando explícito e estruturalmente bem organizado o comportamento dos agentes de *software*, o

pattern State permite que classes concretas de *AgentState* sejam implementadas sob demanda, definindo novos estados e estendendo o comportamento dos agentes.

No desenvolvimento atual do *framework*, a máquina de estados possui três estados definidos: *BeginningState*, *MiddleState*, *EndState*. Existindo a necessidade de customização destes estados para a instanciação de uma nova aplicação, o Instanciador deve alterar a implementação Java dos estados já definidos. Estas modificações não afetam outras partes do *framework* pela própria construção do *pattern State*, facilitando o processo de customização. Pode ser necessária também, para um determinado mercado, a criação de novos estados além da customização daqueles já existentes e, para isso, o Instanciador deve implementar novas classes concretas, herdando diretamente da classe abstrata *AgentState*, e customizar os estados anteriores e posteriores de acordo com o *pattern State* para inseri-lo na máquina de estados.

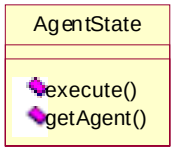


Figura 39 Classe abstrata AgentState

Para a criação do novo estado, o Instanciador deve implementar o método *execute*(), redefinindo a lógica de execução para este estado. O método *getAgent*() permite fazer referência ao agente em questão durante a execução do estado, e não precisa ser adaptado durante a instanciação.

5.5 NegotiationProtocol

O protocolo de negociação define até que ponto os agentes de *software* automatizam o processo de negociação. De acordo com o CBB (*Consumer Buying Behavior Model*), o processo de negociação pode ser modelado definindo-se seis etapas principais:

Figura 40 CBB Model

Este ponto de flexibilização permite que o Instanciador escolha entre automatizar o processo de negociação até a etapa *Merchant Brokering* ou até a etapa *Negotiation*. Esta flexibilidade é importante, pois, em muitos mercados C2B, fica evidente a necessidade de avaliar propostas e construir contrapropostas pessoalmente, ao invés de delegar esta tarefa a agentes de *software*. Automatizando o processo até a etapa *Merchant Brokering*, a localização de vendedores e a mediação do envio e recebimento de propostas são feitas pelos agentes, caracterizando mercados com o processo de negociação semi-automático. Automatizando até a etapa *Negotiation*, a avaliação de propostas e a criação de contrapropostas são feitas pelos agentes de forma automática, caracterizando mercados com processo de negociação automático.

Para aplicar, de fato, a escolha de um dos processos de negociação na instância do *framework*, o Instanciador deve customizar a implementação do método *processReceivedTradeMessages*(). Para o processo de negociação automático, a implementação deste método deve conter a lógica para avaliação de propostas e criação de contrapropostas, ou seja, novas *TradeMessages*. Esta lógica deve avaliar as

informações das *TradeMessages* recebidas (*Itens* e *NegotiationParameters*) para construir contrapropostas. Para o processo de negociação semi-automático, não é necessário implementar o método, uma vez que a análise de propostas recebidas e a criação de contrapropostas são feitas diretamente pelo usuário final.

A definição da escolha de implementação deste ponto de flexibilização interfere pouco na implementação de outros pontos de flexibilização do *framework* e, a princípio, deve ser levado em consideração somente na necessidade de customização da máquina de estados dos agentes do sistema, nas quais o método *processReceivedTradeMessages*() é executado.

5.6 FilterTechniques

Técnicas de filtragem, termo utilizado no *framework* CommercePipe, definem mecanismos de *matching* utilizados para otimizar a aproximação de agentes vendedores de agentes compradores. Em aplicações de comércio eletrônico como as instanciadas a partir do Framework CommercePipe, nas quais a aproximação entre compradores e vendedores é uma etapa bem definida no processo de negociação, a qualidade com que esta aproximação é realizada impacta diretamente nos pontos listados abaixo:

1. *Otimização do processo de avaliação de propostas* – Um mecanismo de filtragem bem construído permite que sejam gerados conjuntos menores de propostas para avaliação. Desta forma, independente da escolha do processo de negociação da aplicação, o usuário final ou o agente de *software* otimizará o tempo de análise de propostas recebidas;

2. *Otimização da qualidade da negociação final* – Um mecanismo de filtragem bem escolhido, teoricamente, direciona a negociação para o caso ótimo, considerando como caso ótimo a negociação na qual todas as preferências do usuário são atendidas. Isto é verdade, pois o mecanismo de filtragem considera, exatamente, as preferências do usuário no momento da seleção do conjunto de vendedores.

Este ponto de flexibilização é modelado seguindo o *pattern Strategy* [Gamma, 1995], no qual as estratégias em questão são técnicas de filtragem. Este *pattern* permite que o Instanciador escolha que técnica de filtragem utilizar na aplicação ou até que ele crie uma nova estratégia implementando uma classe concreta de *FilterTechnique*, como mostra a figura 41:

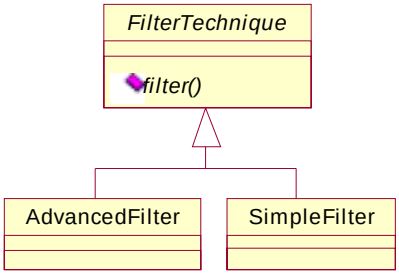


Figura 41 Modelagem para o ponto de flexibilização FilterTechnique

A definição de uma nova técnica de filtragem deve implementar o método *filter*() obrigatoriamente. A implementação deste método faz uso das informações contidas em *TradeMessage*s e da informação contida em *SellProfile* para realizar o *matching*.

Por ser a primeira instância de um *framework* ainda em desenvolvimento, a aplicação AutoPipe utiliza *SimpleFilter* como técnica de filtragem. Este filtro aproxima todos os agentes de venda cadastrados no sistema ao agente de compra.

É interessante notar que a medida que novas aplicações forem instanciadas, um número maior de técnicas de filtragem será, eventualmente, incorporado ao *framework*, permitindo que novas aplicações sejam instanciadas reutilizando técnicas já desenvolvidas anteriormente.

5.7 Services

A customização do subsistema *Services* deve ser realizada de acordo com a decisão de que funcionalidades devem ser implementadas para os usuários da aplicação. Apesar de existir reutilização de alguns serviços já disponíveis no *framework* como *Login*, *ListInbox*, *ListOutbox*, o Instanciador deve definir quais serão os serviços pelos quais os usuários irão interagir com a aplicação. Duas demandas principais podem existir:

1. *Customização de serviços já implementados e disponíveis no framework* – Neste caso, o Instanciador deve customizar o método *doService*() do serviço em questão, adicionando a lógica necessária;

2. *Implementação de novos serviços* – A implementação de novos serviços deve ser feita pela implementação de uma nova classe concreta de *Service*, na qual a implementação do método *doService*() é aberta e cabe, ao instanciador, decidir como implementar a lógica do serviço.

Com a implementação de novos serviços, a medida que novas aplicações são instanciadas, a partir do *framework*, a reutilização de serviços acelera o processo de instanciação de novas aplicações.

5.8 A Aplicação AutoPipe

A aplicação AutoPipe foi instanciada seguindo o processo de instanciação de aplicações descrito neste capítulo para verificar a usabilidade do framework CommercePipe.

Na aplicação AutoPipe, consumidores através de uma interface Web, podem cadastrar intenções de compra, responder a propostas de vendedores e verificar todas as mensagens enviadas e recebidas no mercado virtual. Já os vendedores, através de uma interface WML, podem interagir com o mercado virtual através de telefones celulares criando propostas de venda para todas as intenções de compra que chegam em suas caixas de

entrada. Existe também uma interface Web para a empresa responsável por vendedores, onde é possível verificar informações sobre vendedores cadastrados e autorizados a atuar no mercado virtual. As telas da interface Web e Wap da aplicação são encontradas no final deste documento no Apêndice A.

Para caracterizar o processo de instanciação e o esforço de programação necessário para instanciar a aplicação AutoPipe a partir do framework CommercePipe, a tabela 10 mostra por subsistema o trabalho realizado para a instanciação da aplicação.

Subsistema	Observações
Agentes	Foi necessário implementar o método execute() do estado BeginState dos Agentes de Compra e Venda. Implementação da classe concreta SemiAutomaticProtocol. Definição dos atributos de Item e dos parâmetros de negociação na classes NegotiationParameters. Customização da classe SellProfile.
Serviços	Criação de novos serviços através da implementação de classes concretas de Service. Foram implementados 10 serviços que poderão ser reutilizados em futuras instanciações.
Facilitator	Implementação de classe concreta SimpleFilter. Implementação do mecanismo de controle de transação SingleThreadExecution.
Frontend	Implementação de páginas HTML e WML. Esforço de programação proporcional ao conjunto de serviços implementados. Implementação de 10 JavaBeans utilizados nos scripts JSP como pontos de acesso ao backend.
Communication	Customização da classe ServiceDispatcher

Tabela 10 Esforço para a instanciação da aplicação AutoPipe

Com a utilização do framework, o esforço de programação foi reduzido principalmente pelo foco dado a reutilização de software. Foi necessário a implementação/customização de 28 classes e de toda a interface WEB/WAP para a instanciação da aplicação AutoPipe. Sem a utilização de frameworks, a instanciação da aplicação AutoPipe exigiria maior esforço de programação, e com a visão de instanciar futuras aplicações C2B, o nível de reutilização de software tornaria este esforço ainda maior.

Capítulo 6

Conclusões

Durante o primeiro semestre de 1999, o *framework* VMarket sofreu algumas importantes alterações com o objetivo de levá-lo do estágio embrionário ao de um protótipo estável a ponto de ser possível a instanciação de um primeiro mercado virtual para avaliar sua usabilidade e aceitação.

Grande parte das alterações realizadas objetivaram aumentar a robustez do sistema, e áreas como controle de transação e controle de concorrência foram bastante alteradas. Foi criado um módulo (Log Manager) responsável pelo *log* de transações consolidadas, *log* de ações dos usuários como criação de agentes, modificação de *profile* entre outros eventos. Outro módulo importante adicionado ao sistema foi o de administração (Administrator). O módulo de Administração tem como objetivo prover todas as funcionalidades necessárias para a administração de um mercado virtual em produção. Algumas destas funcionalidades já se encontram implementadas como a verificação de agentes ativos e expirados, lista de usuários do sistema, troca de senha de usuários, exclusão de agentes, dentre outras.

Com este trabalho realizado, o protótipo VBookM@rket [Vbookmarket, 2000] foi lançado no início de setembro de 1999 para a utilização das pessoas integrantes do Departamento de Informática da Universidade, laboratórios associados e alunos de Engenharia de Computação. Trata-se de uma primeira instância do *framework* VMarket. O objetivo deste sistema é possibilitar a negociação de livros na Universidade de forma que calouros gastem menos na compra de livros, e que veteranos consigam vender os que não são mais de interesse, além de promover a venda de livros de maneira geral.

Os oito meses de pesquisa e evolução do Framework Vmarket, que utiliza agentes de *software* para criação de aplicações C2C na *Internet*, foram essenciais para o desenvolvimento do Framework CommercePipe. Por ser um excelente laboratório para a pesquisa na utilização de agentes de *software* no comércio eletrônico, o aprendizado, neste período, acelerou a modelagem e o desenvolvimento do Framework CommercePipe, no que diz respeito à utilização de agentes de *software* no segmento C2B.

Com 65 classes, o Framework CommercePipe permite a criação de aplicações C2B nas quais consumidores por um processo semi-automático/automático de negociação expõem suas intenções de consumo ao sistema no qual vendedores mediados, então, por agentes de *software*, competem neste mercado virtual para realizar a venda de acordo com as necessidades do consumidor.

Por ser um ambiente *multi-thread* e concorrente, o controle de concorrência em mercados virtuais desta natureza faz-se necessário. O *framework* disponibiliza, para as aplicações instanciadas, um controle de concorrência bastante simples por meio da classe *SingleThreadExecution*, bloqueando a execução de *threads* concorrentes em pontos críticos do sistema.

A utilização da tecnologia para acesso móvel aos mercados virtuais traz uma nova possibilidade para realização de transações comerciais, principalmente, para o consumidor. O *framework* permite a instanciação de aplicações nas quais consumidores e/ou vendedores podem utilizar o mercado virtual por dispositivos móveis como celulares e PDAs. O casamento entre agentes de *software* e o acesso móvel foi excelente, permitindo, principalmente, balancear a falta de flexibilidade da utilização dos dispositivos móveis pela automação de tarefas, diminuindo, assim, a dificuldade de utilização destes dispositivos.

A criação da aplicação AutoPipe mostrou a viabilidade de instanciação de aplicações do *framework*. Nesta aplicação, vendedores interagem por meio de dispositivos móveis,

consumidores, assim como empresas de administração e seus vendedores, por meio da interface *Web*,.

De forma geral, o *framework* CommercePipe mostra como agentes de *software* podem ser utilizados em conjunto com a tecnologia para acesso móvel à *Internet* e para instanciar aplicações de comércio eletrônico preparadas para este tipo de acesso móvel. Apesar de estar preparado para o segmento C2B, os benefícios gerados por ambas as tecnologias podem ser atingidos em outros segmentos de mercado, como por exemplo o B2B.

Capítulo 7

Trabalhos Futuros

Sendo o estado atual da implementação do *framework*, alguns trabalhos de pesquisa e desenvolvimento de *software* estão indicados, abaixo, como próximos passos naturais para a criação de uma segunda versão do mesmo.

- A criação de uma interface WML para o consumidor permitiria o acesso por dispositivos móveis, para compradores e vendedores, permitindo a instanciação de aplicações "*full wireless*". Com o reuso desta implementação, o processo de instanciação de novas aplicações seria acelerado;

- Aprimorar a comunicação entre os agentes e os usuários identificando novas situações no processo de negociação nas quais a comunicação é importante e desejável;

- A integração, ao sistema, de um mecanismo de pagamento eletrônico permitiria a instanciação de aplicações com suporte a etapa de pagamento do CBB. A utilização de *smartcards* para a implementação deste mecanismo é viável e seria uma solução com níveis de segurança adequados para esta tarefa;

- Adicionar aspectos de segurança ao framework. Esta é uma característica importante para frameworks de comércio eletrônico e não foi foco neste trabalho de pesquisa;

- Adicionar mecanismos de reputação ao framework, permitindo que futuras aplicações possam fazer uso da reputação de consumidores e vendedores na avaliação de negociações;

- A instanciação de aplicações com mecanismo automático ainda depende da completude da máquina de estado dos agentes do sistema. Os estados necessários para a realização de negociações automáticas ainda precisam ser implementados no *framework*;

- Instanciar aplicações para a comercialização de produtos distintos com a finalidade de identificar os principais atributos a serem modelados para os diferentes produtos. Este trabalho traria o aprendizado sobre como deve ser realizada a modelagem de produtos para aplicações de comércio eletrônico nas quais agentes de *software* são utilizados. A instanciação de novas aplicações permitirá avaliar as vantagens obtidas na instanciação de aplicações a partir do framework comparado ao desenvolvimento de software sem a utilização de frameworks.Experimentar novas técnicas de filtragem e de negociação criando novas classes concretas de *FilterTechnique* e *NegotiationProtocol*;

- Novas funcionalidades poderiam ser adicionadas ao mecanismo de comunicação entre o *frontend* e *backend*, melhorando o controle sobre o acesso concorrente ao *backend*, tornando, principalmente, mais robusta a implementação do *pool* de *threads* do sistema;

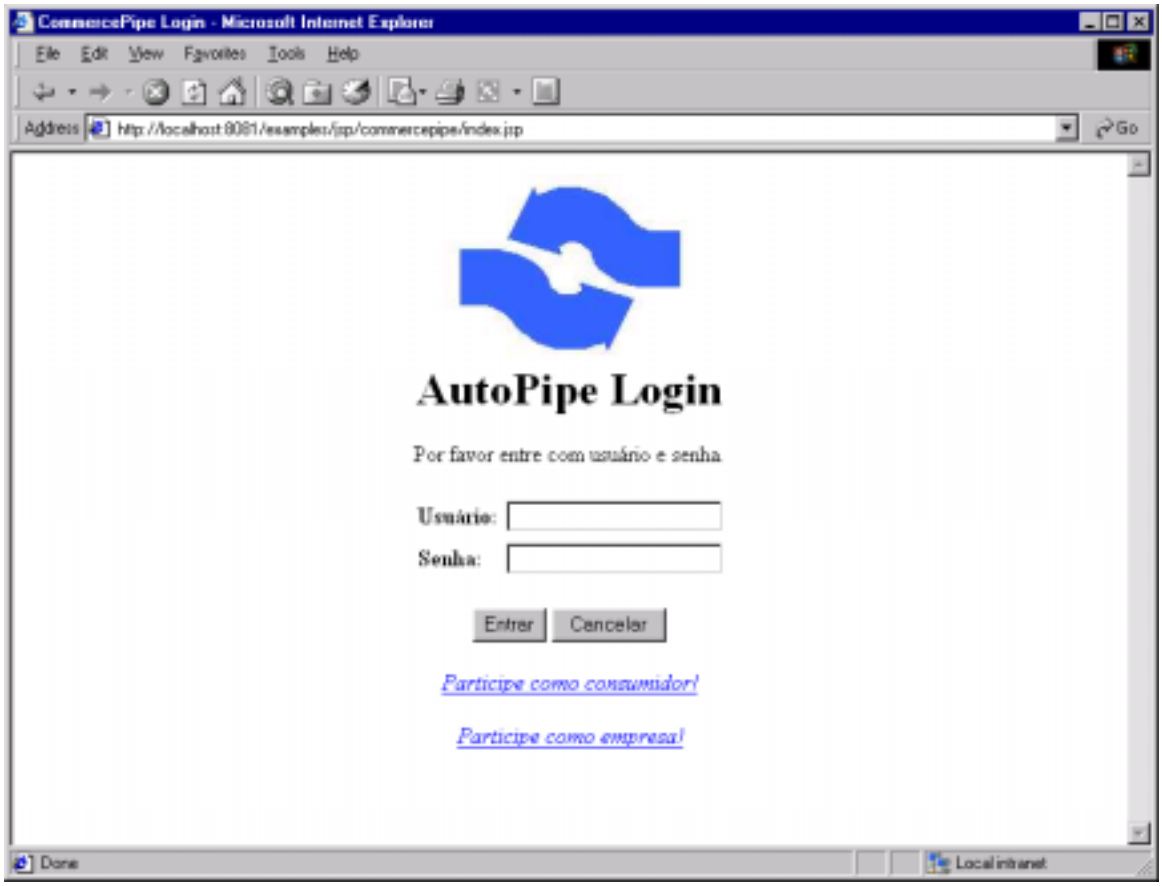
- Criar DSLs para facilitar o processo de instanciação permitindo criar programas capazes de gerar código automaticamente para instanciar novas aplicações.

Apêndice A

Telas da Aplicação AutoPipe

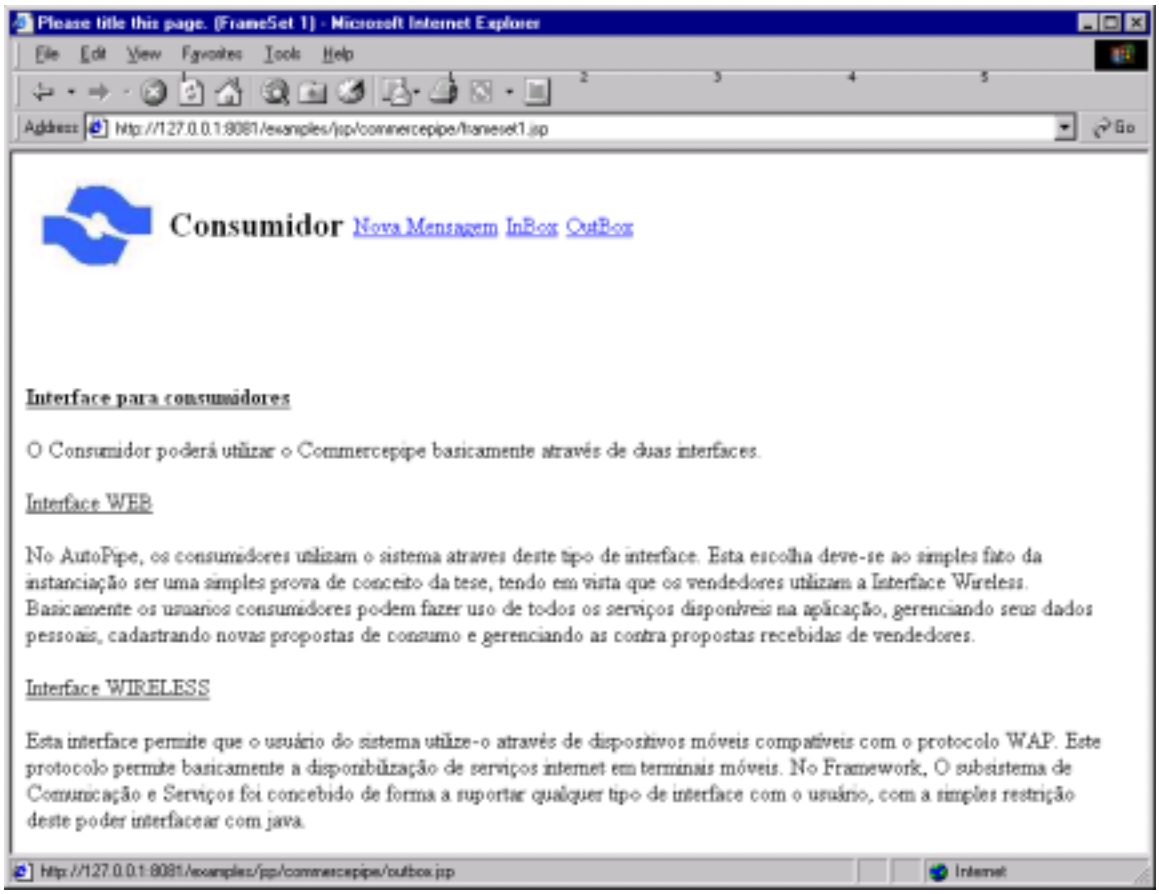
AutoPipe Login

Tela de Login na Interface Web da aplicação. Empresas e Consumidores podem utilizar o sistema através desta interface.



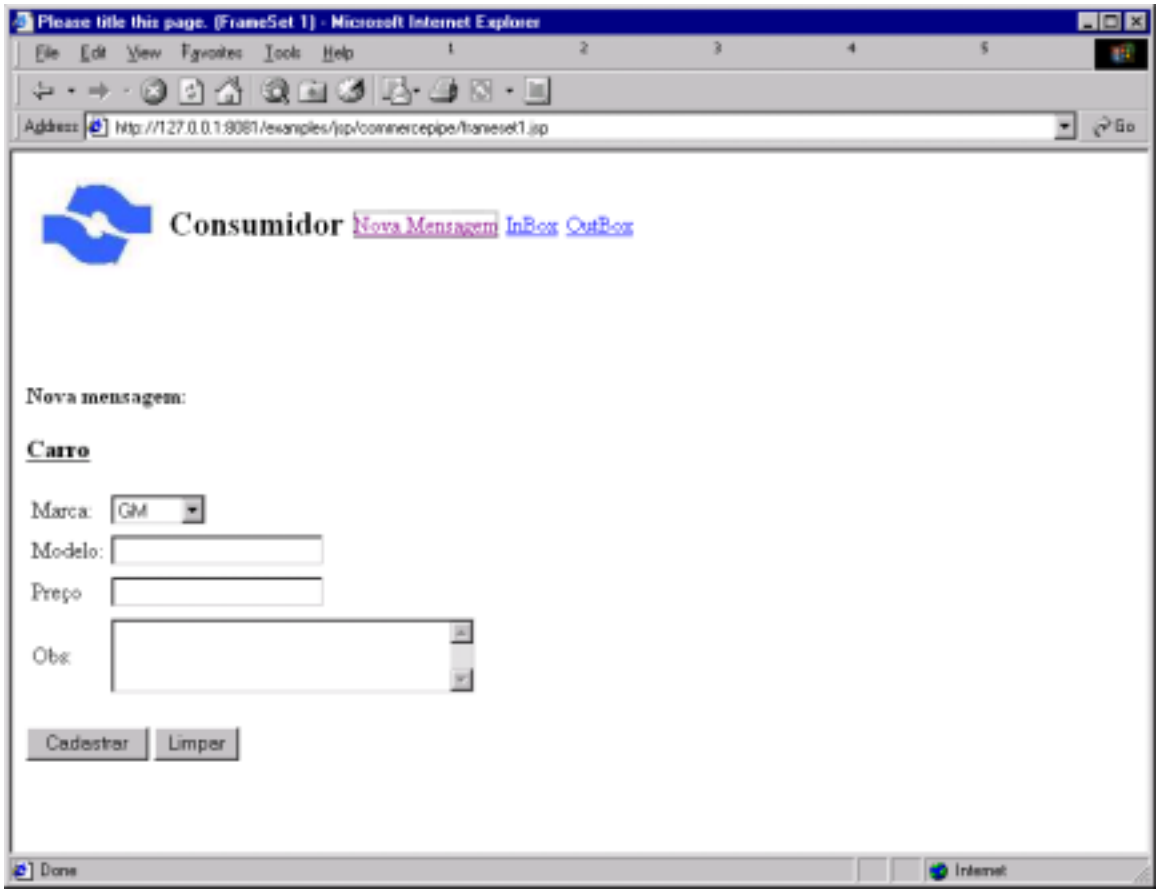
Interface Web Para Consumidores

Utilizando esta interface com o sistema, os consumidores podem criar e cadastrar novas propostas de consumo e checar contra propostas enviadas por vendedores.



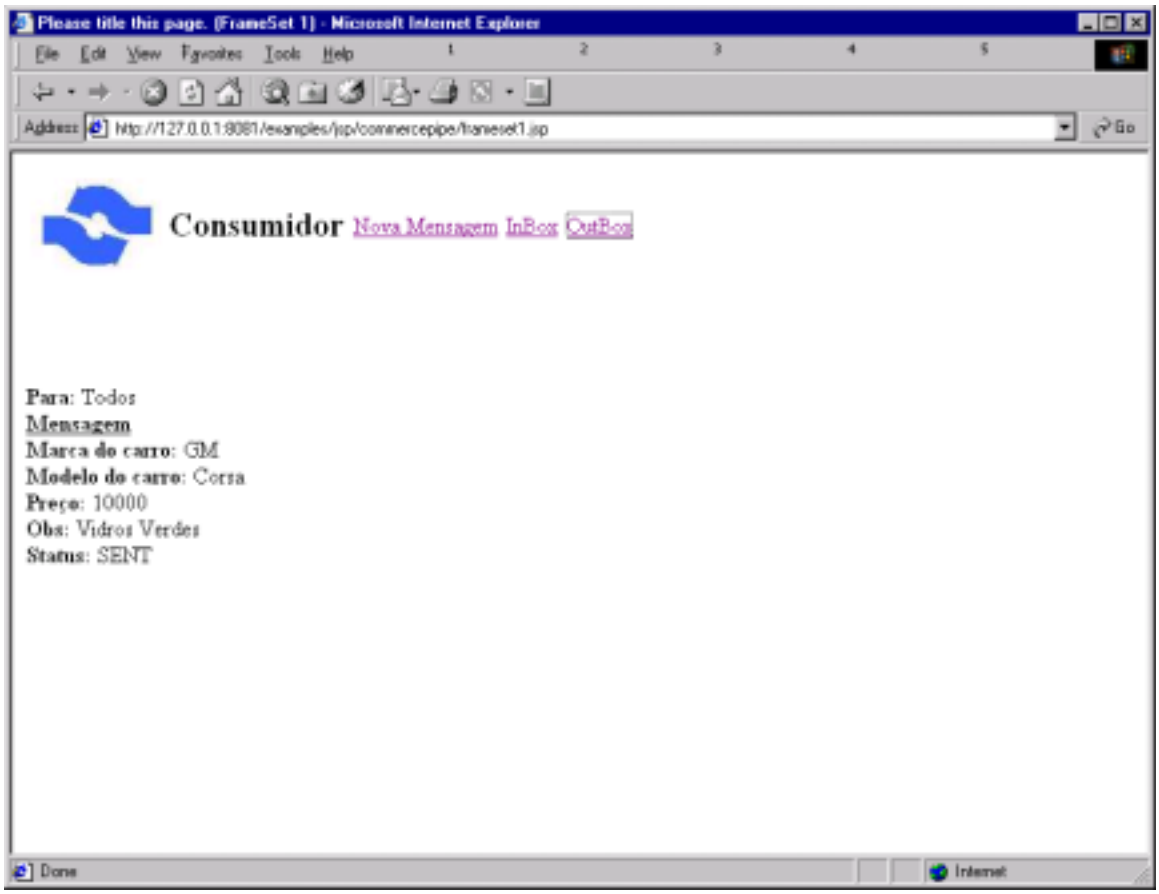
Cadastramento de Proposta de Consumo

Através de formulários, consumidores podem cadastrar no sistema propostas de consumo. Na aplicação AutoPipe, as propostas consiste na descrição da Marca, Modelo, Preço do carro e através de um campo de observação é possível informar qualquer outra exigência com relação ao carro desejado, como forma de pagamento, estofamento, etc.



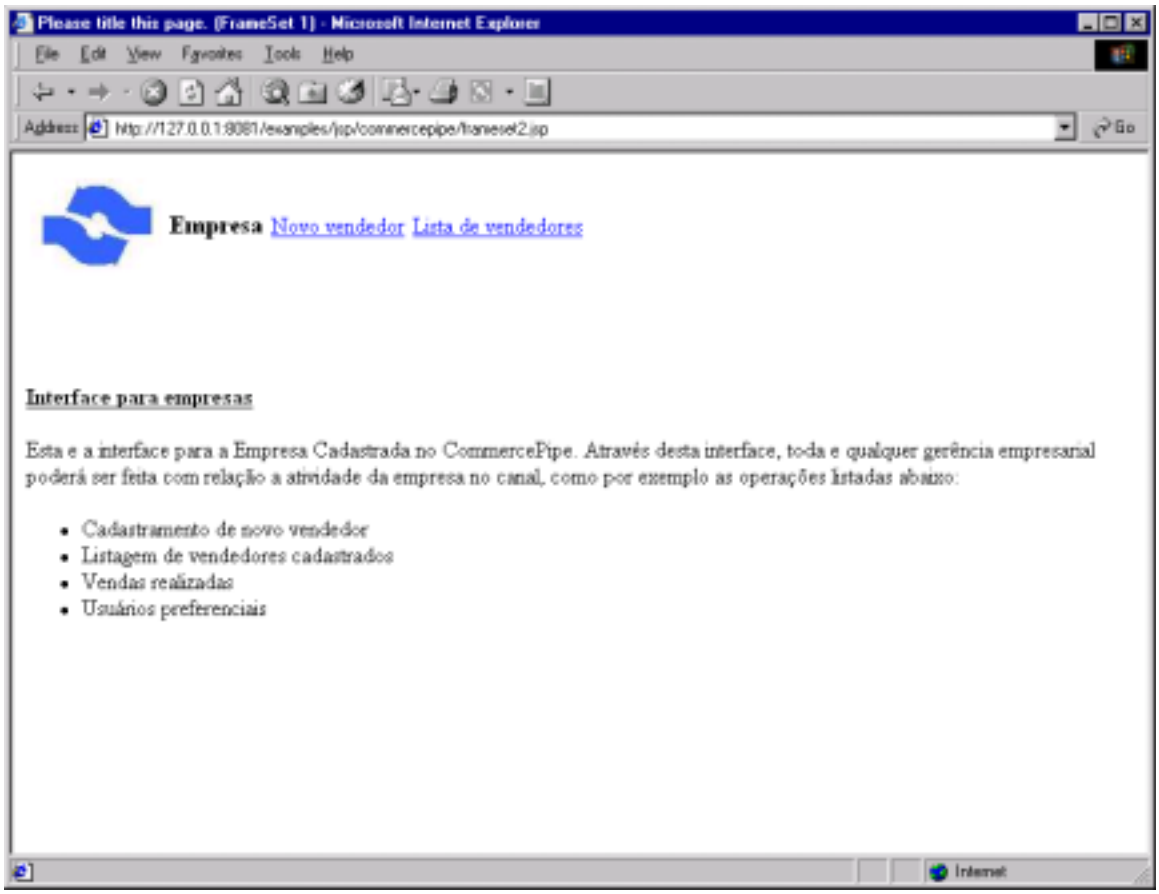
Caixa de entrada com propostas de vendedores

Os Consumidores possuem uma caixa de entrada onde são armazenadas as contra-propostas enviadas por vendedores cadastrados no sistema. Através desta interface é possível verificar todos os detalhes das contra-propostas recebidas.



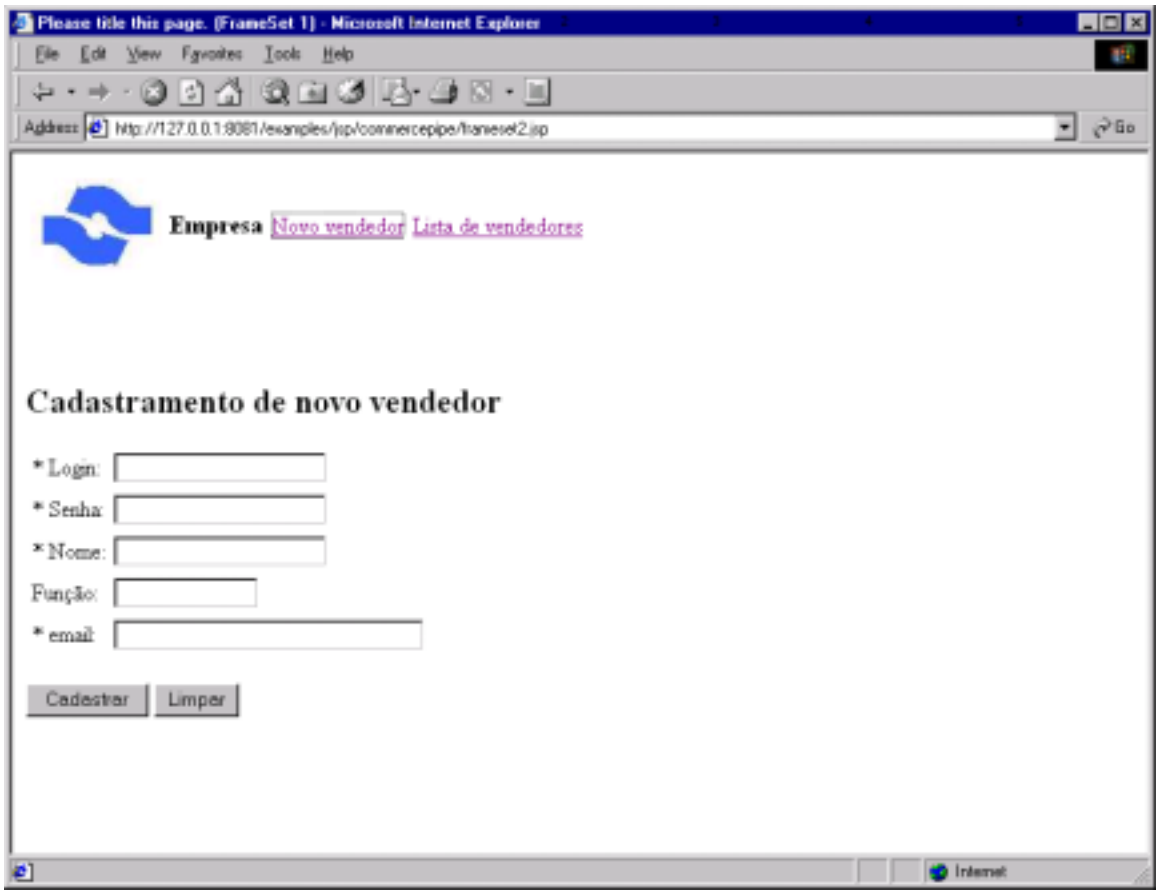
Interface Web para Empresas

Através desta interface, as empresas cadastradas no sistema podem administrar suas operações na aplicação AutoPipe.



Cadastramento de Vendedores

Para cadastrar novos vendedores no sistema, as empresas precisam cadastrar informações básicas sobre o vendedor onde Login e Senha são obrigatórias para a utilização da interface wireless implementada para os vendedores do sistema.



Interface Wireless para vendedores

Página de apresentação

Primeira página do site WAP para vendedores da aplicação AutoPipe.

Login para vendedores

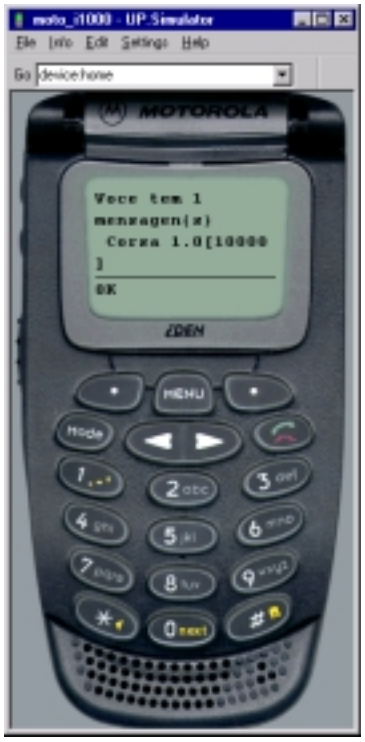
Para que os vendedores possam utilizar a interface wireless, é necesssário que se identifiquem informando o Id da empresa em que trabalham, o Login e a Senha.

Tela de Boas Vindas

Assim que o vendedor realiza o Login no sistema, lhe é apresentado uma tela de boas vindas.

Opções da interface

O vendedor pode através da interface Wireless verificar as propostas de consumo cadastradas no sistema que foram direcionadas a ele na Caixa de Entrada. Na Caixa de Saída ficam armazenadas todas as contra propostas já enviadas por ele no sistema. É possível também verificar todos os Acordos de venda já estabelecidos previamente.



Caixa de Entrada

Na Caixa de Entrada, os venderores visualizam de forma compacta e objetiva as propostas de consumo cadastradas no sistema e enviadas para ele.

Capítulo 8

Referências

[AMEC, 1999] AMEC – Agent Mediated e-Commerce. [online]. <http://ecommerce.media.mit.edu/> [Consulta: 1999]

[AU, 1999] AU System “WAP White Paper” AU Systems <http://www.ausystem.com>, fevereiro 1999.

[Baray, 1999] Baray, C. Wagner, K. “Where do Inteligent Agents come from?” CrossRoads – The ACM Student Magazine, 4-8, Summer 1999.

[Bellman, 1999] Bellman, S. Lohse, G. Johnson, E. “Predictors of the Online Buying Behavior” Communications of the ACM, December 1999/Vo . 42, No. 12.

[Bradshaw, 1997] Bradshaw, J. M. “Sofware Agents” AAAI Press/The MIT Press, 1997.

[Brereton, 1999] Brereton, P. Budgen, D. Bennet, K. Munro, M. Layzell, P. Macaulay, L. Griffiths, D. Stannet, C. “The Future of Software” Communications of the ACM, December 1999/Vo . 42, No. 12 .

[Chavez, 1996ª] Chavez and P. Maes. "Kasbah: An Agent Marketplace for Buying and Selling Goods". Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96). London, UK, April 1996.

[Chavez, 1996b] Chavez, A. Dreilinger D. Guttman, R. Maes, P. “A Real-Life Experiment in Creating an Agent Marketplace” MIT Media Laboratory, 1996.

[Conallen, 1999] Conallen, J. “Modeling WEB Application Architectures with UML” Communications of the ACM, Outubro, 1999/Vo . 42, No. 10.

[Conner, 2000] Conner, M. “IBM Application Framework for e-business: Structuring e-business applications” [online] http://review.software.ibm.com/developer/library/structure/index.html [Consulta: janeiro 2000].

[Cornell, 1998] Cornell, G. Horstmann. C. S. “Core Java” Makron Books, 1998.

[Durlacher, 2000] Durlacher Research “Mobile Commerce” Durlacher Research Report, janeiro 2000.

[Ericsson, 2000] Ericsson “Ericsson Wireless Internet” [online] <http://mobileinternet.ericsson.com> [Consulta: Janeiro 2000].

[Fayad, 1999a] Fayad, M. E. Schmidt, D. C. Johnson, R. E. “Building Application Frameworks” Wiley, 1999.

[Fayad, 1999b] Fayad, M. E. Schmidt, D. C. Johnson, R. E. “Implementing Application Frameworks” Wiley, 1999.

[Fingar, 1999] Fingar, P. “Intelligent Agents: The Key to Open Ecommerce” Middleware & Infrastructure. April 1999.

[Flores, 1999] Flores, R. A. “Towards a standarization of multi-agent system frameworks” CrossRoads – The ACM Student Magazine, 18-23, Summer 1999.

[Fontoura, 1999] Fontoura, M. F. M. C. “A systematic approach to framework development”, Tese de Doutorado, 1999, Departamento de Informática, Puc-Rio.

[Fortunato, 1999] Fortunato, L. “2BuyNet: Um Framework para Instanciação e Administração de Lojas para a Internet” Dissertação de Mestrado, 1999, Departamento de Informática, Puc-Rio.

[Gamma, 1995] Gamma, E. Heml, R. Johnson, R. Vlissides, J. “Design Patterns – Elements of Reusable Object-Oriented Software” Addison Wesley, 1995.

[Guttman, 1998a] Guttman, R. and P. Maes. "Agent-mediated Integrative Negotiation for Retail Electronic Commerce." Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET'98). May 1998.

[Guttman, 1998b] Guttman, R. and P. Maes. "Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce." To appear, Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98). Paris, France, July 1998.

[Guttman, 1998c] Guttman, R. Moukas, A. and P. Maes. "Agent-mediated Electronic Commerce: A Survey." To appear, Knowledge Engineering Review, June 1998.

[Grand, 1998] Grand, M. “Patterns in Java – A Catalog of Reusable Design Patterns Illustrated with UML” Volume 1, Wiley, 1998.

[IAC, 1999] IAC – IBM Institute for Advanced Commerce. [online] <http://www.ibm.com/iac/> [Consulta: setembro 1999].

[Jaenicke, 1999[a]] Jaenicke, C. “Capitalizing on the Benefits of XML” Distributed Computing, 3-4, maio, 1999.

[Jaenicke, 1999b] Jaenicke, C. “Server-side XML: Taming the Tower of Babel” Distributed Computing, 48-50, outubro, 1998.

[Kumar, 1999a] Kumar, M. Feldman S. “Business Negotiation on the Internet” IBM Research Division, 1999.

[Kumar, 1999b] Kumar, M. Feldman, S. “Internet Auctions” IBM Research Division., 1999.

[Kumar, 1999c] Kumar, M. Rangachari, A. Jhingran A., Mohan R. “Sales Promotion on the Internet” IBM Research Division, 1999.

[Larsen, 1999] Larsen, G. “Designing Component-Based Frameworks using Patterns in the UML” The Communications of ACM, October 1999/Vo . 42, No. 10.

[Maes, 1994] Maes, P. “Agents that Reduce Work and Information Overload”, Communications of the ACM, Vol. 37, No. 7, July 1994.

[Maes, 1999] Maes, P. Guttman, R. and A. Moukas. "Agents that Buy and Sell: Transforming Commerce as we Know It." Communications of the ACM, March 1999 Issue.

[MayField, 1997] Mayfield, J. Finin, T. Labrou, Y. “KQML as an Agent Communication Language.” J. Bradshaw (ed.), Software Agents, MIT Press, 1997.

[MCDA, 1999] MCDA – Multi-criteria Decision Analysis. [online] <http://www.concentric.net/~vdwalle/notes/mcda1.htm> [Consulta: agosto 1999].

[MIT, 1999] MIT Media Laboratory’s Kasbah [online] <https://kasbah.media.mit.edu> [Consulta: 1999].

[Moukas, 1998] Moukas, R. Guttman, and P. Maes. "Agent-mediated Electronic Commerce: An MIT Media Laboratory Perspective." Proceedings of the First International Conference on Electronic Commerce (ICEC'98), Seoul, Korea, April 1998.

[Nokia, 1999a] Nokia “Developers Guide, Nokia Wap Toolkit, v1.2” Nokia Wap Developer Forum <http://www.forum.nokia.com>, setembro 1999.

[Nokia, 1998] Nokia “Nokia Developers Guide, Nokia SDK v1.0” dezembro 1998.

[Nokia, 1999b] Nokia “Nokia WAP Server API Specification” maio 1999.

[Nokia, 1999c] Nokia “Wireless Application Protocol – The Corporate Perspective” White Paper, março 1999.

[Nokia, 1999d] Nokia “WML Reference, Version 1.1” Nokia Wap Developer Forum <http://www.forum.nokia.com>, setembro 1999.

[Nokia, 1999e] Nokia “WMLScript Reference, Version 1.1” Nokia Wap Developer Forum <http://www.forum.nokia.com>, setembro 1999.

[Nokia, 1999f] Nokia WAP Developer Forum [online] <http://www.forum.nokia.com/> [Consulta: novembro 1999].

[Oliphant, 1999] Oliphant, M. W. “The mobile phone meets the Internet” IEEE Spectrum, agosto 1999.

[Phone, 2000a] Phone.com, Inc. “Phone.com” [online] <http://www.phone.com> [Consulta: abril 2000].

[Phone, 2000b] Phone.com, Inc. “Understanding Security on the Wireless Internet, How WAP Security Is Enabling Wireless E-commerce Applications for Today and Tomorrow” janeiro 2000.

[Phone, 2000c] Phone.com, Inc. “UP.SDK Developers Guide, Release R4.B2” janeiro 2000.

[Ripper, 1999] Ripper, P. “VMarket: Um Framework para Sistemas de Comércio Eletrônico voltado para Mercados Virtuais Mediados por Agentes de Software” Dissertação de Mestrado, 1999, Departamento de Informática, Puc-Rio.

[Ripper, 2000] Ripper, P. Fontoura, M. F. Neto, A. M. Lucena, C. J. “V-Market: A Framework for e-Commerce Agent Systems” World Wide Web, Baltzer Science Publishers, 3(1), 2000.

[T&R, 1999] Technology and Research – XML: Mastering Information on the Web [online] <http://www.sun.com/980310/xml> [Consulta: setembro 1999].

[The Economist, 2000] The Economist “Ecommerce Survey” fevereiro, 2000.

[Tsvetovatyy, 1996] Tsvetovatyym M., Mobasher, B., Gini, M. And Wieckowski, Z. “MAGMA: An Agent-Based Virtual Market for Eletronic Commerce”, Dep. Of Computer Science University of Minnesota, Minneapolis, MN55455, 1996.

[UP, 1999] Unwired Planet “Why You Should Develop Applications Using The Unwired Planet WAP Solution” February 1999.

[Vbookmarket, 2000] VbookMarket – A Virtual Book Marketplace [online] <http://harper.les.inf.puc-rio.br/vbookmarket> [Consulta: abril 2000].

[Voyager, 1999] Object Space’s Voyager [online] <http://www.objectspace.com/voyager> [Consulta: 1999]

[W3C, 1997] W3C “Extensible Markup Language (XML).” World Wide Web Consortium (W3C) Working Draft, 17 November 1997.

[WAP, 1999a] WAP Forum – Wireless Application Protocol Forum [online] <http://www.wapforum.org> [Consulta: 1999].

[WAP, 1999b] WAP Forum “Wireless Architecture, Wireless Application Protocol Architecture Specification” Abril 1998, Wap Forum <http://www.wapforum.org>.

[WHS, 2000] Wap Hole Sun – WAP Develeper Web Site [online] <http://www.wapholesun.com> [Consulta: abril 2000].

[Wolk, 2000] Wolk, M. R. Bryan, C. K. “Wireless Data, The Next Internet Frontier” Technology Research Report, Robertson Stephens, janeiro 2000.

[YankeeGroup, 2000] The YankeeGroup “Internet Predictions 2000” The YankeeGroup Report, Internet Market Strategies, vol. 6, No 2 – fevereiro 2000.