Paula Clark Ribeiro

Modelagem e Implementação OO de Sistemas Multi-Agentes

DISSERTAÇÃO DE MESTRADO

DEPARTARMENTO DE INFORMÁTICA

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Rio de Janeiro, 26 de outubro de 2001.

II

Paula Clark Ribeiro

Modelagem e Implementação OO de Sistemas Multi-Agentes

Dissertação apresentada ao Departamento de Informática da PUC-Rio como parte dos requisitos para obtenção do título de Mestre em Informática: Ciência da Computação.

Orientador: Carlos José Pereira de Lucena

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 26 de Outubro de 2001

Ao Márcio e à milha família.

Agradecimentos

Ao Márcio por todo o seu amor, carinho e incentivo, proporcionando a base necessária para esta conquista.

Aos meus pais por todo o apoio e por tudo que fizeram e fazem por mim dando me condições de realizar o mestrado.

Aos meus irmãos Leo, Ana, Isabel e Júlia pela amizade e companheirismo proporcionando um excelente ambiente familiar de muita tranqüilidade e amor.

À todos os meus amigos, como a Angelinha, a Dani e a Mônica, por todos os momentos bons e divertidos que já passamos juntas em nossas vidas.

À Carlos Lucena, por me acolher em seu grupo, por todo o aprendizado e apoio que tive sob sua orientação e por proporcionar em seu grupo o trabalho em equipe, onde seus alunos de mestrado e doutorado fazem uma valiosa troca de experiência e conhecimento.

Aos amigos do Teccomm por proporcionarem um excelente ambiente de trabalho onde pude aprender muito e por me ajudarem sempre que eu precisava.

À Viviane, Sardinha, Otávio, Guga, Cristina e a todos os amigos da PUC/Teccomm que fizeram valiosas contribuições, sugestões e correções para a minha tese.

À Vera Menezes pela boa vontade de sempre solucionar todos os nossos problemas.

À Lena Ururahy pela sua amizade e por estar sempre pronta a ajudar.

Aos professores Ruy Milidiú e Arndt Von Staa pelas valiosas sugestões e contribuições e por participarem da minha banca.

Aos professores Bruno Feijó e Marcus Poggi e à Isabela por todo o apoio nos momentos críticos relacionados à bolsa da Capes.

À Capes pelo apoio financeiro necessário para a execução desta dissertação.

A todos os professores e funcionários do DI pelo aprendizado, apoio e ajuda.

Resumo

Este trabalho propõe um mapeamento de elementos encontrados na engenharia de software baseada em agentes para elementos da engenharia de software orientada a objetos. Os mapeamentos foram ilustrados através da modelagem de um sistema de mercado virtual na Internet (*marketplace*).

Para as fases de análise e projeto utilizou-se uma metodologia orientada a agentes denominada Gaia que propõe visualizar o sistema a ser modelado como uma organização de papéis que interagem.

São propostos dois mapeamentos dos elementos gerados pela metodologia Gaia para modelos da programação orientada a objetos. O primeiro mapeamento é feito para o *MAS Framework*, framework de desenvolvimento de sistemas multi-agentes, enquanto o segundo é feito para um design pattern de desenvolvimento de sistemas multi-agentes. O *MAS Framework* aplica-se ao domínio de sistemas multi-agentes reativos enquanto o design pattern abrange o domínio de sistemas multi-agentes cognitivos. Ambas tecnologias foram desenvolvidas no laboratório Teccomm.

Ao final, apresenta-se uma análise das vantagens e desvantagens encontradas na aplicação e no mapeamento da metodologia Gaia para as duas abordagens de desenvolvimento OO, através de um estudo de casos sobre o mercado virtual Babilônia.

Abstract

This work proposes a mapping between agent-based software engineering elements and object oriented software engineering elements. The mappings are illustrated through the modelling of an Internet marketplace.

During the analysis and design phases, the Gaia methodology for agent-oriented analysis and design is used. Gaia methodology views the system as an organization of roles that interact.

Two mappings are proposed between the Gaia methodoly elements and object oriented programming models. The first one is done to the *MAS Framework*, a framework to build multi-agent systems, and the other is done to a design pattern to construct multi-agent systems. *MAS Framework* is applicable to reactive agents and the design pattern is concerned with the development of cognitive agents. Both of them were developed at Teccomm labs.

At the end, an analysis of the advantages and disadvantages of mapping Gaia methodology artefacts is presented based on two approaches of developing OO systems. The case studies involve the development of the Babilônia marketplace.

Conteúdo

Lista de Figuras

Lista de Tabelas

Introdução

Agentes de software e sistemas multi-agentes representam uma nova forma de analisar, projetar e implementar sistemas de software complexos. A visão baseada em agentes oferece um poderoso conjunto de ferramentas, técnicas e metáforas que têm o potencial de melhorar consideravelmente a forma de conceituação e implementação de vários tipos de software[17].

Agentes estão sendo usados num crescente número de aplicações, desde sistemas pequenos como filtros de e-mails a complexos sistemas de missão crítica como controle de tráfego aéreo. Em todos os casos, tais sistemas possuem em comum a abstração chave usada que é o agente.

Neste contexto surge a engenharia de software baseada em agentes que consiste em utilizar a tecnologia de agentes para resolver problemas de engenharia de software. A justificativa da engenharia de software baseada em agentes é fundamentada nas vantagens da utilização do paradigma orientado a agentes na aplicação das três estratégias de solucionar sistemas complexos: decomposição, abstração e organização[1].

Dentro desta nova ótica de desenvolver sistemas, é necessário que, a exemplo do que ocorre com o paradigma orientado a objetos, o paradigma orientado a agentes seja também suportado por metodologias e técnicas que garantam a qualidade do processo de produção de software.

Este trabalho tem como objetivo analisar e aplicar a metodologia Gaia[13, 12] de análise e projeto orientado a agentes num estudo de caso de mercado virtual. Para tal, além de propor mapeamentos dos modelos gerados pela metodologia Gaia para modelagens orientadas a objetos, propõem-se extensões para a metodologia Gaia.

Os capítulos estão assim organizados:

- O capítulo 1 apresenta uma introdução aos principais conceitos envolvidos na tecnologia de agentes utilizados neste trabalho.

- O capítulo 2 introduz a metodologia Gaia de análise e projeto orientado a agentes.

- O capítulo 3 apresenta o problema que serviu como estudo de casos que é o mercado virtual Babilônia e em seguida descreve a aplicação da metodologia Gaia no problema. Ao final do capítulo faz-se uma análise das vantagens e desvantagens da metodologia Gaia encontradas durante a sua utilização.

- O capítulo 4 propõe um mapeamento dos elementos da metodologia Gaia para os elementos da modelagem OO para a implementação do mercado virtual Babilônia. A modelagem OO é apresentada em diagramas de UML[10]. O capítulo termina com a análise do mapeamento feito.

- O capítulo 5 relata as principais contribuições desta dissertação e faz algumas considerações a respeito dos trabalhos futuros.

Capítulo 1

Agentes de Software

A pesquisa sobre agentes de software, que se intensificou nos últimos 20 anos, teve início na área de IAD (Inteligência Artificial Distribuída) tornando-se posteriormente também tema de pesquisa da comunidade de Engenharia de Software.

Segundo Wooldridge [7], o termo agente não possui uma definição unânime entre os grupos de pesquisa. O único ponto em que os pesquisadores concordam é que a autonomia é ponto central na característica dos agentes.

Para definir o termo agente será utilizada aqui a definição do Wooldridge em [7]:

“Um agente é um sistema computacional situado em um ambiente e que é capaz de ações autônomas neste ambiente a fim de alcançar seus objetivos”.

Um importante documento nesta área de pesquisa é o Green Paper[18] produzido pelo grupo OMG [19] que tem uma definição de agentes coerente com a definição de Wooldridge:

“Agentes de software são entidades de software autônomas que interagem com seu ambiente. O agente percebe seu ambiente através de sensores e atua no ambiente com efetuadores.“

A Figura 1 ilustra a definição de agentes:

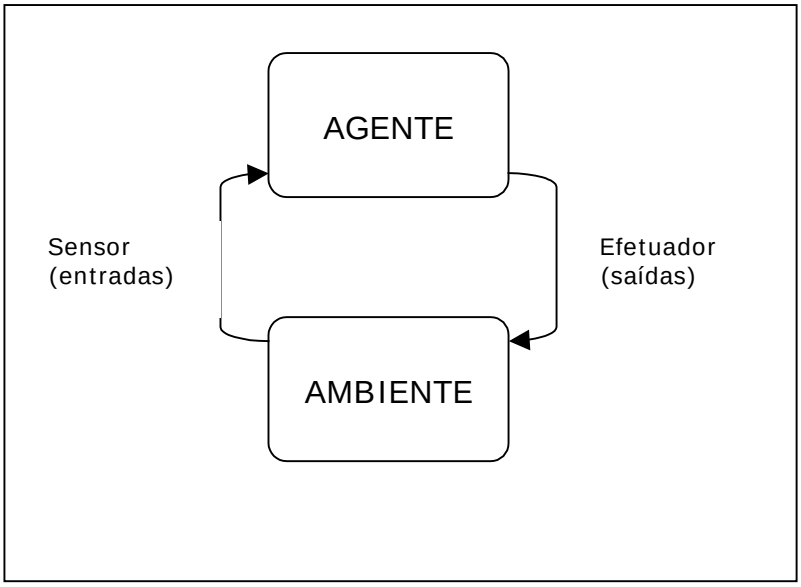

Figura 1 – Visão de agentes

O grupo OMG define uma série de propriedades de agentes, como as listadas a seguir:

Autonomia: É a capacidade de atuar sem intervenção externa direta. Possui dois aspectos independentes: autonomia dinâmica e autonomia imprevisível. Autonomia dinâmica significa que os agentes são pró-ativos e sabem decidir quando agir. Já a autonomia imprevisível significa que não é possível prever as atitudes que serão tomadas pelo agente podendo este, inclusive, decidir por não executar uma determinada tarefa.

Interação: O agente comunica-se com o ambiente e com os outros agentes de forma a atingir seus objetivos.

Adaptação: O agente possui capacidade de responder a outros agentes e ao seu ambiente. Isto significa que o agente pode simplesmente reagir a um determinado estímulo ou pode ir além e refletir sobre o que deve ser feito.

Mobilidade: O agente pode ser transportado para um outro ambiente onde será executado.

Coordenação: O agente possui a capacidade de desempenhar suas atividades em um ambiente compartilhado com outros agentes. As atividades são coordenadas através de planos, fluxo de trabalho ou outro mecanismo de gerência de processo.

Cooperação: Os agentes são capazes de se coordenarem para alcançar um objetivo comum (colaboração é outro termo usado como sinônimo de cooperação).

Dentre tantas propriedades o grupo OMG considera que as três primeiras propriedades descritas acima (autonomia, interação e adaptação) são as mais importantes e portanto fundamentais para caracterizar um agente.

Outro conceito importante da tecnologia de agentes, considerado no escopo deste trabalho, é a distinção que existe entre os agentes reativos e os agentes cognitivos [9]. Os agentes reativos têm o conhecimento construído nas ações a serem executadas. Este tipo de agente não precisa construir uma representação mental do seu mundo pois para ele é suficiente saber reagir às situações apresentadas a ele. Já os agentes cognitivos possuem a capacidade de raciocínio baseada na sua representação interna do mundo. Tal representação se dá através dos componentes cognitivos do agente que também chamamos de estados mentais do agente. Os estados mentais governam todas as atividades internas de um agente: percepção e execução de ações, crenças, desejos e tendências, intenções, métodos e planos, etc.

1.1 Sistemas multi-agentes

Sistemas multi-agentes são compostos por sociedades de agentes que não só interagem entre si, mas que também se coordenam, seja através de cooperação ou de competição ou ainda da combinação de ambas propriedades[18]. Sistemas multi-agentes são então compostos por agentes coordenados através de seus relacionamentos com outros agentes.

O grupo OMG faz algumas considerações sobre os sistemas multi-agentes:

- Não se deve criar agentes que façam tudo sozinho pois dessa forma estes agentes representarão problemas de performance, confiabilidade, manutenção, etc. Dividindo as funcionalidades entre vários agentes obtêm-se uma maior modularidade, flexibilidade, extensibilidade e manutenibilidade.

- O conhecimento que está espalhado em vários agentes pode ser integrado, para obter-se uma visão geral, quando necessário.

- Aplicações que requerem processamento distribuído são melhor suportadas por sistemas multi-agentes. Para isso os agentes podem ser projetados como pequenos componentes autônomos que trabalham em paralelo. Processamento e solução de problemas feitos de forma concorrente trazem soluções para muitos problemas que até então eram tratados de forma linear. E por isso, a tecnologia de agentes é um avanço significativo para a tecnologia de componentes distribuídos.

Outra definição para sistemas multi-agentes encontrada em [17] considera que os sistemas multi-agentes são compostos por entidades autônomos que trabalham juntas (interação) para solucionar problemas que estão além de suas capacidades e conhecimentos individuais. As principais características dos sistemas multi-agentes são:

- Cada agente tem capacidade de solucionar parcialmente o problema;
- Não existe um controle global do sistema;
- Os dados são descentralizados;
- O processamento é assíncrono.

Um dos fatores atuais que promovem a pesquisa de sistemas multi-agentes, é a crescente popularidade da Internet, que proporciona a base para um ambiente aberto onde agentes interagem para alcançar os objetivos coletivos e individuais. Em [2] são apresentadas algumas aplicações de sistemas multi-agentes.

1.2 Engenharia de Software Baseada em Agentes

No contexto da engenharia de software os agentes aparecem como uma nova tecnologia para o desenvolvimento de sistemas distribuídos complexos [16, 15]. A utilização da tecnologia de agentes para resolver problemas de engenharia de software deu origem ao termo “Engenharia de Software Baseada em Agentes”.

A Engenharia de Software orientada a agentes apresenta diversos benefícios quando utilizada para o desenvolvimento de sistemas distribuídos complexos[16]. De uma forma geral a

abordagem orientada a agentes aumenta, consideravelmente, a habilidade de modelar, projetar e construir tal classe de sistemas.

Dentro desta nova ótica, os agentes exercem um papel fundamental na decomposição, abstração e organização do sistema, tornando-se assim um novo paradigma de engenharia de software.

Uma das técnicas de engenharia de software específica para a tecnologia de agentes é a metodologia Gaia[13, 12]. Gaia é uma metodologia para análise e projeto de sistemas orientado a agentes, que é o tema do capítulo 2.

Capítulo 2

Metodologia Gaia

Gaia[1] é uma metodologia para análise e projeto orientados a agentes que se baseia na visão de um sistema multi-agente como sendo uma organização computacional onde vários papéis interagem. Gaia possui duas fases distintas: fase de análise e fase de projeto (design). Na fase de análise, são gerados os modelos de papéis e de interação e, na fase de projeto, são construídos os modelos de agentes, de serviços e de afinidades (Figura 2).

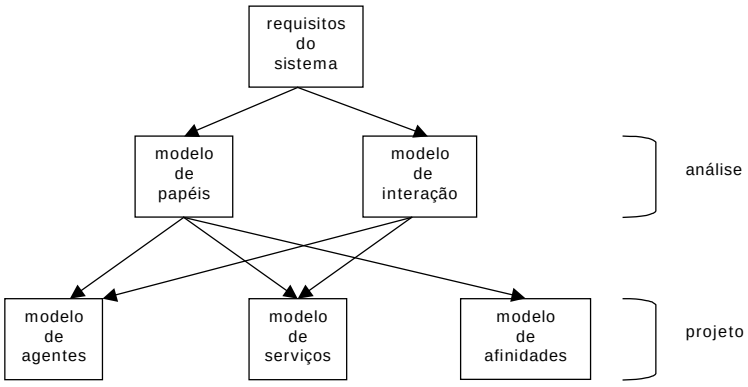

Figura 2 – Os modelos da metodologia Gaia

[1] Na mitologia Grega, Gaia era a figura da mãe Terra. Gaia é também o nome de uma influente hipótese proposta pelo ecologista James Lovelock no sentido de que todo o organismo vivo na Terra pode ser entendido como componente de uma entidade única, que regula o ambiente da Terra. O tema de várias entidades heterogêneas atuando em conjunto para alcançar um único objetivo é um tema central da pesquisa de sistemas multi-agente e foi uma consideração chave no desenvolvimento da metodologia Gaia.

2.1 Fase de análise

Na fase de análise, o objetivo é decompor o sistema em papéis que serão desempenhados na organização (objetivo do modelo de papéis) e definir a maneira como eles interagem de acordo com protocolos específicos (objetivo do modelo de interação). Esta fase promove um melhor entendimento do sistema e de suas estruturas sem considerar detalhes de implementação.

Modelo de papéis

Identifica os papéis existentes no sistema. Um papel é definido por quatro atributos: *responsabilidades, permissões, atividades* e *protocolos.* Responsabilidades determinam as funcionalidades do papel e formam um atributo chave associado ao papel. Responsabilidades se dividem em dois tipos: propriedades de execução[2] e propriedades de segurança. As propriedades de execução especificam o que deve acontecer, enquanto que as propriedades de segurança especificam um conjunto de assertivas que precisam ser mantidas (*invariants*). As propriedades de execução são especificadas através de expressões de execução que definem o "ciclo de vida" de um papel e possuem os operadores descritos na Tabela 1.

Operador	Interpretação
x.y	x seguido de y
x|y	ocorrência de x ou y
x*	x ocorre 0 ou mais vezes
x+	x ocorre 1 ou mais vezes
x^{ω}	x ocorre infinitas vezes
[x]	x é opcional
x||y	x e y ocorrem intercalados

Tabela 1 – Operadores para as expressões de execução

[2] Tradução para liveness properties.

As permissões são os direitos associados ao papel. As permissões de um papel identificam os recursos de informação que estão disponíveis para que o papel possa ser desempenhado. As permissões podem ser de três tipos: *gera*, *lê* e *altera*. *Gera* indica que o papel é quem produz o recurso associado, *lê* indica que o papel tem permissão de leitura do valor do recurso associado e *altera* significa que o papel tem permissão para ler e modificar o valor do recurso em questão.

As atividades são as ações que o papel irá desempenhar sem a necessidade de interagir com outros papéis (no modelo aparecem sublinhados). Um papel é identificado também por seus protocolos que definem a maneira pela qual um papel interage com os demais papéis para desempenhar uma determinada ação (no modelo aparecem sem sublinhado).

As expressões de execução portanto, definem a trajetória de execução de um papel através de suas atividades e de seus protocolos. Podendo ser representada de duas formas, exemplificadas abaixo:

- Forma geral:

NOMEDOPAPEL = Atividade1 . Atividade2 . Protocolo1 . Protocolo2 . Protocolo 3

- Forma Estruturada:

NOMEDOPAPEL = NOME1 . NOME2

NOME1 = Atividade1 . Atividade2 . Protocolo1

NOME2 = Protocolo2 . Protocolo3

O modelo de papéis, que sofreu influências do Fusion Method[3], descreve todos estes atributos num único local (Figura 3).

Esquema do papel: Nome do papel
Descrição: Breve descrição do papel
Protocolos e Atividades: Protocolos e atividades que o papel possui
Permissões: Direitos associados ao papel
Responsabilidades: Execução: Propriedades de execução Segurança: Propriedades de segurança

Figura 3 – Padrão para o modelo de papéis

Modelo de interação

É formado por um conjunto de definições de protocolos, um para cada tipo de interação entre papéis. A definição de um protocolo consiste nos seguintes atributos: *propósito*, *iniciador*, *respondedor*, *entradas*, *saídas*, e *processamento*. O *propósito* é uma breve descrição da natureza da interação. O *iniciador* é o papel responsável por iniciar a interação. O *respondedor* é o papel com o qual o *iniciador* interage. As *entradas* são as informações utilizadas pelo *iniciador* durante a execução do protocolo. As *saídas* são as informações fornecidas para o *respondedor* durante o curso da interação. E o *processamento* é uma breve descrição textual do processamento do *iniciador* durante a interação.

O modelo de interação é um diagrama com a seguinte estrutura (Figura 4):

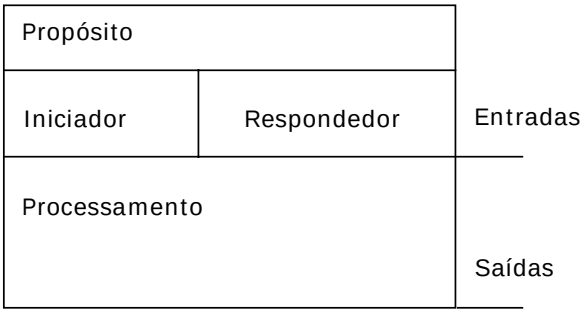

Figura 4 – Padrão para modelo de interação

Processo de Análise

O processo de análise em Gaia deve seguir os seguintes passos:

1. Identificar os principais papéis no sistema, produzindo um protótipo do modelo de papéis.

2. Para cada papel identificar os protocolos associados, ou seja, as interações que ocorrem entre os papéis .

3. Com o modelo de interação, completar o modelo de papéis e, se necessário, repetir os passos anteriores.

Ao final deste processo, o documento esperado é um modelo de papéis bem elaborado, descrevendo cada papel em termos de responsabilidades, permissões, protocolo de interação e atividades, e um modelo de interação, descrevendo cada protocolo em termos de troca de informações e padrões envolvidos.

2.1 Fase de Projeto

Esta fase utiliza os modelos definidos na fase de análise para gerar novos modelos com níveis de abstração mais baixos, obtendo uma definição do sistema multi-agentes mais próxima da implementação. A fase de projeto precisa definir os agentes que irão desempenhar os papéis identificados durante a fase de análise, bem como a cardinalidade destes agentes (tarefa do modelo de agentes). Os serviços que os agentes devem oferecer para desempenhar seus papéis também precisam ser especificados nesta fase (objetivo do modelo de serviços), assim como a topologia das interações entre as diversas classes de agentes de acordo com os modelos de interação da fase de análise e o modelo de agentes da fase de projeto (objetivo do modelo de afinidade). Exemplos dos dois primeiros modelos da fase de projeto podem ser vistos no capitulo 3.

Modelo de agentes

Neste modelo, são identificadas as classes dos agentes que teremos no sistema em execução. Para cada classe de agente definida é preciso especificar a cardinalidade desta. Uma classe de agente pode ser vista como sendo um conjunto de papéis de agentes. Em geral, existirá uma correspondência de um para um entre papéis e classes de agente. Mas também pode existir mais de um papel relacionado a apenas uma classe de agente.

Modelo de serviços

Como o nome já diz, o objetivo do modelo de serviços é identificar os serviços associados a cada classe de agente (derivam das atividades e protocolos do modelo de papéis), e especificar as principais propriedades destes serviços. Um serviço diz respeito à função de um agente. Em termos de OO um serviço corresponde a um método. Para cada serviço é preciso identificar as entradas, saídas, pré-condições e pós-condições. Entradas e saídas são derivadas do modelo de interação. As pré- e pós-condições representam as restrições aos serviços e são derivadas das propriedades de segurança do modelo de papéis.

Modelo de afinidade

Este modelo define os canais de comunicação que existem entre as classes de agentes. Ele não define quais mensagens são enviadas ou quando as mensagens são enviadas, apenas indica quais os caminhos de comunicação existentes. O objetivo deste modelo é identificar um potencial gargalo de comunicação que pode causar problemas durante a execução.

Processo de Projeto

O processo de projeto em Gaia envolve os seguintes passos:

1. Identificar o modelo de agentes, ou seja, agregar papéis a classes de agentes e definir a cardinalidade de cada classe.

2. Identificar os serviços que os agentes devem oferecer para realizar os papéis atribuídos a eles, com a análise das atividades, dos protocolos e das responsabilidades definidas para cada papel.

3. Desenvolver o modelo de afinidade utilizando o modelo de interação e o modelo de agentes para identificar ineficiências do projeto e se necessário, repetir os passos anteriores.

Ao final deste processo, o documento esperado é a atual arquitetura (organização) do sistema de agentes, que pode ser implementado usando técnicas tradicionais de projeto, como por exemplo projeto orientado a objetos.

Capítulo 3

Aplicando a metodologia Gaia

O caso de estudo escolhido para modelagem neste trabalho foi um sistema que implemente um mercado virtual de bens através da Internet. Além de permitir a compra, venda e negociação de bens, o sistema deve permitir ainda a certificação de compradores, vendedores e itens para garantir uma transação mais segura.

3.1 Como funciona o sistema

Para um melhor entendimento do problema, esta seção foi dividida de acordo com os atores que participam da execução do sistema. Para cada ator, são descritas as tarefas que ele deve fazer para desempenhar o seu papel.

Compradores

O sistema permite que um usuário cadastrado crie perfis de compra para busca de bens. Os usuários podem cadastrar vários perfis de compra, mas cada um deles deve fazer referência a apenas uma categoria de item. Por exemplo, um usuário A deseja comprar uma caneta e um livro no mercado virtual. Será necessário criar dois perfis de compra: um com as características da caneta procurada e outro com as características do livro procurado.

Ao criar um perfil de compra o usuário comprador deve informar algumas ou todas as características do item procurado de acordo com a sua categoria. Para a categoria de automóveis do sistema a descrição de um item a buscar está exemplificada na Figura 5. Além disso, o comprador deve informar se será feita a certificação do item e do vendedor ou não. Detalhes da certificação de item e de vendedor são abordados ao final da seção.

Os itens encontrados na busca de um determinado perfil de compra são enviados para o usuário que o criou. O usuário então seleciona um ou mais itens e inicia o processo de negociação. Para iniciar uma negociação é preciso informar ao sistema os seguintes parâmetros de negociação:

- preço inicial da oferta;
- preço máximo de compra;
- parcela de incremento que se adiciona a cada contra-proposta do vendedor.

Um exemplo de escolha de parâmetros de negociação de um determinado item encontrado pode ser visualizado na Figura 88 do apêndice B.

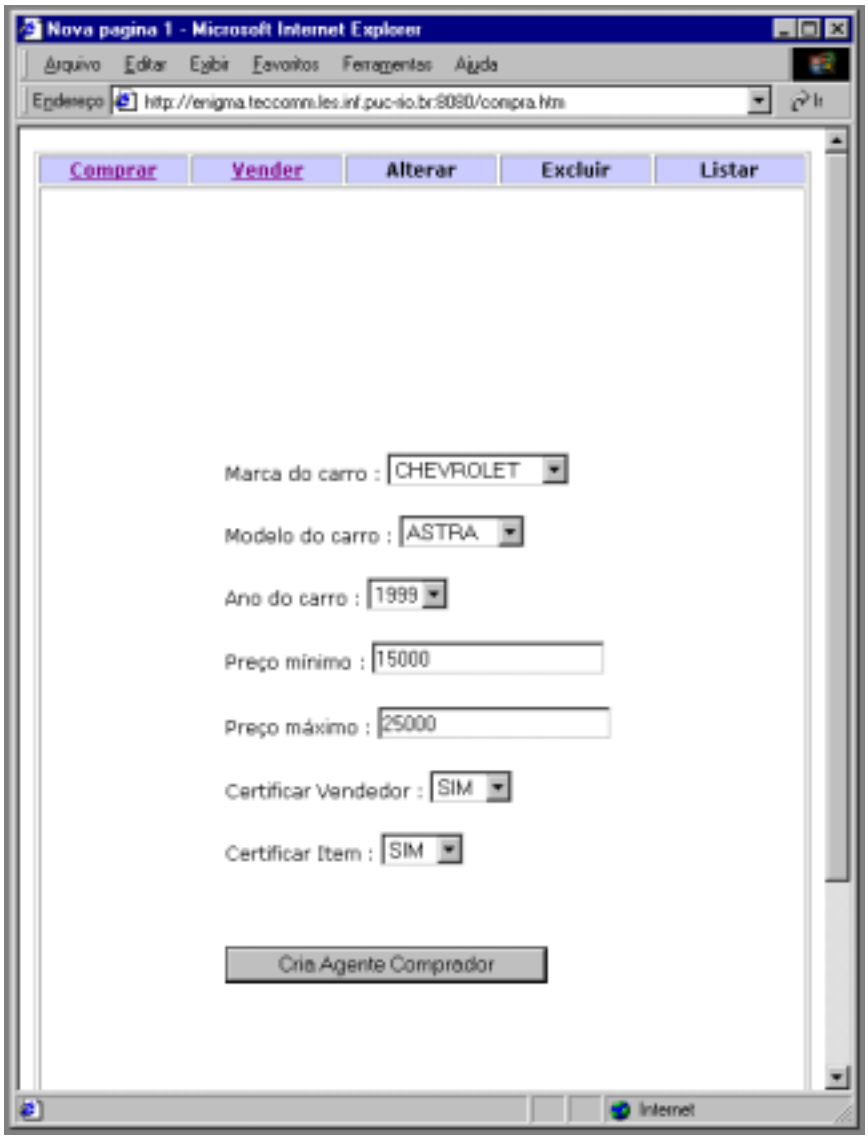

Figura 5 – Perfil de compra de um item da categoria automóvel

Vendedores

Para vender itens no sistema, os usuários vendedores deverão inicialmente criar um perfil para o item que está sendo vendido, informando todas as suas características. Um exemplo de perfil de item a venda da categoria de automóveis é mostrado na Figura 6. Ao criar o perfil do item a ser vendido, o usuário informa de que maneira este item será negociado e se haverá certificação do comprador. Para haver a negociação é preciso informar os seguintes parâmetros de negociação:

- parcela de decremento que se subtrai a cada proposta ou contra-proposta do comprador

- preço mínimo de venda

O usuário pode cadastrar também as suas preferências de comunicação. O sistema permite que seus usuários cadastrem diferentes formas de receber avisos do sistema sobre itens encontrados para a compra, transações efetuadas, etc. A notificação ao usuário pode ser enviada através de uma mensagem direta para o seu celular ou do envio de uma mensagem para a sua caixa de correio eletrônico.

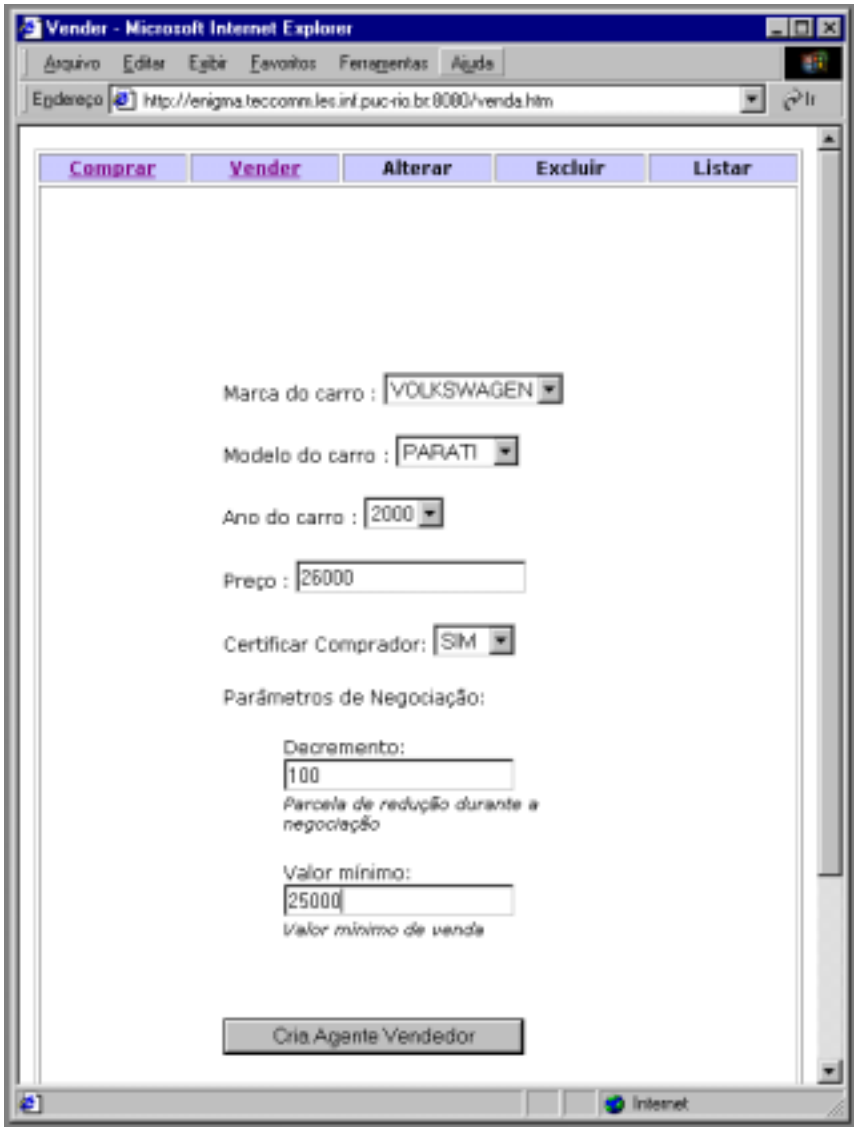

Figura 6 – Perfil de venda de um item da categoria automóvel

Depois de fornecidas as informações, o usuário vendedor autoriza o anúncio do item no ambiente. O item ficará anunciado durante um período de tempo padrão (a ser definido pelo administrador do sistema) a espera de compradores que busquem um perfil de item que case com o anunciado. Se depois do tempo de anúncio o item não for vendido, o sistema notifica o seu dono e remove o seu anúncio do ambiente. Caso o vendedor queira anunciar o item novamente, não será necessário informar os seus dados novamente, bastando selecioná-lo em seu histórico de itens anunciados e solicitar novamente a sua venda.

Certificador

Quando um perfil de compra encontra um correspondente perfil de venda no ambiente, o comprador que solicitou a busca é notificado. Caso seja do interesse das partes, pode ser feito um processo de certificação de usuários e itens. Para tal, é necessário que o comprador solicite a certificação do bem e/ou do seu vendedor, ou que o vendedor solicite a certificação do comprador.

O certificador é o responsável por receber as solicitações de certificação e por autenticar itens e usuários envolvidos na negociação. As funções desempenhadas pelo certificador podem envolver a busca em bases de dados de órgãos governamentais ou a verificação de dados fora do ambiente computacional, como a verificação por telefone das referências bancárias de um usuário.

3.2 Pensando na organização do sistema baseado em agentes de software

A abordagem utilizada de explicação do problema através das funções de seus atores contribui para uma primeira análise de como seria a composição do mercado virtual baseado em agentes.

Cada usuário do mercado virtual pode ser representado por um agente que armazenará as suas preferências e agirá conforme as suas diretrizes (um assistente pessoal). Este agente será o responsável por saber como o usuário quer ser contatado e se é necessário ou não fazer

certificação de itens e usuários para iniciar uma negociação. O agente do usuário também deverá saber criar novos agentes compradores ou vendedores conforme a necessidade.

Outros agentes de fácil identificação são os agentes vendedores e compradores. É interessante notar que cada vendedor ou comprador controla a venda ou compra de um perfil de item a cada momento. Por exemplo, se um usuário tiver dois itens para vender ele será responsável por três agentes no ambiente: 1 assistente pessoal e 2 vendedores (um para cada item vendido).

O certificador também pode ser considerado um agente, mas as suas funções podem estar relacionadas à interação com um ser humano que receba as requisições de certificação e interaja com o agente. As tarefas que o certificador poderá fazer automaticamente serão bem limitadas num primeiro momento, devido à ausência de automação de órgãos de certificação governamentais.

Outros agentes podem surgir das necessidades técnicas do sistema no decorrer de sua modelagem conforme veremos a seguir.

3.3 Modelagem Gaia do problema

Esta seção apresenta a aplicação da metodologia Gaia, apresentada no capítulo 2, no problema descrito.

Fase de Análise

Nesta fase, a descrição do problema será transformada em dois modelos que deverão capturar os papéis a serem desempenhados no sistema e como estes papéis interagem para realizar as suas funções. A construção dos modelos não deve ser feita separadamente. Os modelos devem ser construídos em conjunto e a sua consistência mantida durante o processo.

Modelo de Papéis

Para o problema em questão foram identificados os seguintes papéis:

- Usuário – responsável pelas ordens que são dadas ao sistema. O usuário é o principal cliente do mercado virtual, sendo o seu principal beneficiário. É ele que possui bens para serem vendidos ou está interessado na compra de algum bem.

- Assistente de Usuário – esse papel é responsável pela interface com o papel usuário, recebendo as suas ordens e as transformando em ações no sistema. O assistente de usuário é responsável por manter dados de preferências, e criar compradores e vendedores conforme a necessidade.

- Comprador – busca itens de determinados perfis no ambiente e negocia, de acordo com seus parâmetros de negociação, os itens encontrados na busca do melhor preço.

- Vendedor – anuncia itens de determinado perfil no ambiente e negocia segundo parâmetros de negociação definidos com possíveis compradores.

- Certificador – certifica itens e/ou usuários de acordo com a solicitação de outros agentes.

- Comunicador – envia mensagens de notificação aos usuários através de diferentes meios de comunicação.

Cada um dos papéis definidos acima indica a presença de alguma entidade que poderá ou não ser um agente de software. Por enquanto, a preocupação é apenas o esclarecimento dos papéis e a descrição resumida de suas funções. A definição de que entidade se tornará ou não um agente será feita na fase de projeto do Gaia.

A partir dos papéis identificados e do que eles representam para o sistema, pode ser iniciado o levantamento das funções necessárias ao desempenho destes papéis, que de acordo com a metodologia Gaia são as responsabilidades do papel. Ao indicar quais as funções necessárias, deve-se levar em consideração entre quais papéis estas funções são desempenhadas. Por exemplo, para desempenhar a função criar agente de compra, o assistente de usuário deve interagir com o papel comprador para ativar a compra de um bem de determinado perfil. Em Gaia, funções que exigem interação entre papéis são denominadas protocolos e farão parte tanto do modelo de papéis quanto do modelo de interação.

Existem outras funções do papel, que para serem desempenhadas, não exigem nenhuma interação com outros papéis. Às funções onde não ocorrem interações com outros papéis dá-se o nome de atividade. As atividades fazem parte do modelo de papéis, mas não fazem parte do modelo de interação. A identificação destes dois conjuntos de funções serve de base para a definição dos dois modelos da fase de análise.

O conjunto de figuras começando na Figura 7 e terminando na Figura 12 apresenta a modelagem dos papéis identificados no problema. Cada um dos modelos possui 5 áreas divididas por linhas horizontais. A primeira indica o nome do papel; a segunda, uma descrição geral para este papel; a terceira mostra os protocolos e atividades que o papel possui; a quarta apresenta as informações relacionadas ao papel; a quinta indica as responsabilidades do papel através das propriedades de execução de suas funções (liveness properties) e das propriedades de segurança (safety properties).

Nesta etapa os papéis já estão bem definidos e os protocolos que fazem parte de cada papel estão indicados no modelo. Com base neste modelo é possível construir os modelos de interação.

Esquema do papel: USUARIO (USU)

Descrição: Iniciar as ações de compra e venda de itens através do sistema.

Protocolos e Atividades:
InformarDadosPessoais, InformarPreferencias, OrdenarCompra, OrdenarVenda, ConfirmarCompra, EncerrarCompra, SolicitarHistorico, AlterarDadosPessoais, AlterarPreferencias, ListarPerfisCompra, ListarPerfisVenda, AlterarPerfilCompra, AlterarPerfilVenda, ExcluirPerfilCompra, ExcluirPerfilVenda, SolicitarSituacaoPesquisa, ListarItensNegociados

Permissões:

gera	*dadosPessoais*
	preferências
	perfilCompra
	perfilVenda
	itensSelecionados
	parametrosNegociacao
	itemTransacao
lê	**informado** *itensEncontrados*
	informado *itensNegociados*

Responsabilidades:
Execução:

USUARIO = CRIARUSUARIO . (COMPRAR | VENDER | ALTERAR | EXCLUIR | LISTAR) $^{\omega}$
CRIARUSUARIO = InformarDadosPessoais . InformarPreferencias
COMPRAR = OrdenarCompra . [ConfirmarCompra . EncerrarCompra]
VENDER = OrdenarVenda
ALTERAR = AlterarDadosPessoais | AlterarPreferencias | AlterarPerfilCompra | AlterarPerfilVenda
EXCLUIR = ExcluirPerfilCompra | ExcluirPerfilVenda
LISTAR = ListarPerfisCompra . [SolicitarSituacaoPesquisa] | ListarPerfisVenda | ListarItensNegociados | SolicitarHistorico

Segurança:

- **Verdadeiro**

Figura 7 – Modelo de papéis do usuário

Esquema do Papel: ASSISTENTEUSUARIO (ASU)

Descrição: Responsável por fazer a interface com o usuário e por criar os agentes compradores e vendedores.

Protocolos e Atividades:
ArmazenarDadosPessoais, ArmazenarPreferencias, GerarAgenteCompra, GerarAgenteVenda, ListarPerfisCompra, ListarPerfisVenda, ListarItensNegociados, SolicitarSituacaoPesquisa, ConfirmarCompra, SolicitarCertificacaoItem, SolicitarCertificacaoUsuario, InformarItensEncontrados, InformarResultadoNegociacao, EncerrarCompra, InformarDadosComprador, AlterarDadosPessoais, AlterarPreferencias, AlterarPerfilCompra, AlterarPerfilVenda, ExcluirPerfilCompra, ExcluirPerfilVenda, SolicitarHistorico

Permissões:

lê	**informado** *dadosPessoais*
	informado *preferencias*
	itensEncontrados
	itensSelecionados
	parametrosNegociacao
	itemTransacao
	perfilCompra
	perfilVenda
	itensNegociados
	comprador
	listaPerfiscompra
	listaPerfisvenda
	resultado
gera	*itensNegociacaocorrente*
	historicoPerfiscompra
	historicoPerfisvenda
	dadosCertificacao
	dadosUsuario

Responsabilidades:
Execução:

ASSISTENTEUSUARIO = INICIARAGENTE . (COMPRAR | VENDER | ALTERAR | EXCLUIR | INFORMAR) $^{\omega}$
INICIARAGENTE = ArmazenarDadosPessoais . ArmazenarPreferencias
COMPRAR = GerarAgenteCompra . [SolicitarCertificacaoItem] . [SolicitarCertificacaoUsuario] . InformarItensEncontrados . ConfirmarCompra . InformarResultadoNegociacao . EncerrarCompra
VENDER = GerarAgenteVenda . [SolicitarCertificacaoUsuario] . InformarDadosComprador
ALTERAR = AlterarDadosPessoais | AlterarPreferencias | AlterarPerfilCompra | AlterarPerfilVenda
EXCLUIR = ExcluirPerfilCompra | ExcluirPerfilVenda
INFORMAR = ListarPerfisCompra | ListarPerfisVenda | ListarItensNegociados | SolicitarSituacaoPesquisa | SolicitarHistorico

Segurança:

- **Verdadeiro**

Figura 8 – Modelo de papéis do assistente de usuário

Esquema do papel: COMPRADOR (C)

Descrição: Responsável por buscar um item de determinado perfil e mediar a compra do mesmo.

Protocolos e Atividades:
BuscarItem, InformarListaItensEncontrados, EnviarMensagemNegociacao, AvaliarPropostaRecebida, EnviarResultadoNegociacao, EncerrarCompra, InformarTermino, InformarTempoExpirado, Terminar

Permissões:

lê	**informado** *assistenteUsuario*
	perfilCompra
	itensSelecionados
	parametrosNegociacao
	itemTransacao
	proposta
gera	*itensEncontrados*
	itensNegociados
	comprador
	tempoVida
altera	*proposta*

Responsabilidades:
Execução:

COMPRADOR = INICIAR . (PESQUISAR . [NEGOCIAR . COMPRAR] | PESQUISAR) $^{\omega}$. TERMINAR
INICIAR = BuscarItem
PESQUISAR = InformarListaItensEncontrados
NEGOCIAR = (EnviarMensagemNegociação . AvaliarPropostaRecebida) $^{\omega}$
COMPRAR = EnviarResultadoNegociação . EncerrarCompra
TERMINAR = (InformarTérmino | InformarTempoExpirado) . Terminar

Segurança:

- TempoVida < limiteVida

Figura 9 – Modelo de papéis do comprador

Esquema do papel: VENDEDOR (V)

Descrição: Anuncia um item no ambiente e media a sua venda.

Protocolos e Atividades:
PublicarItem, AguardarPropostas, EnviarMensagemNegociacao, AvaliarPropostaRecebida, InformarTermino, InformarFimPrazoVenda, Terminar

Permissões:

lê	**informado** *assistenteUsuario*
	perfilVenda
	comprador
	proposta
gera	*compradoresNegociacao*
	prazoVenda
altera	*proposta*

Responsabilidades:
Execução:

VENDEDOR = INICIAR . (ANUNCIAR .[NEGOCIAR])$^{+}$. TERMINAR
INICIAR = PublicarItem
ANUNCIAR = AguardarPropostas
NEGOCIAR = (EnviarMensagemNegociação . AvaliarPropostaRecebida)$^{\omega}$
TERMINAR = (InformarTermino | InformarFimPrazoVenda) . Terminar

Segurança:

- **Verdadeiro**

Figura 10 – Modelo de papéis do vendedor

Esquema do papel: COMUNICADOR (C)

Descrição: Enviar mensagens através de um tipo de meio de comunicação.

Protocolos e Atividades:
TransmitirMensagem, ConfirmarEnvio

Permissões:

lê	**informado** *assistenteUsuario*
	informado *meioComunicação*
	dadosUsuario
	itensEncontrados
	comprador

Responsabilidades:
Execução:

COMUNICADOR = (TransmitirMensagem. ConfirmarEnvio)$^{\omega}$

Segurança:

- CanalDisponível = true
- UsuárioExiste = true
- Mensagem <> vazio

Figura 11 – Modelo de papéis do comunicador

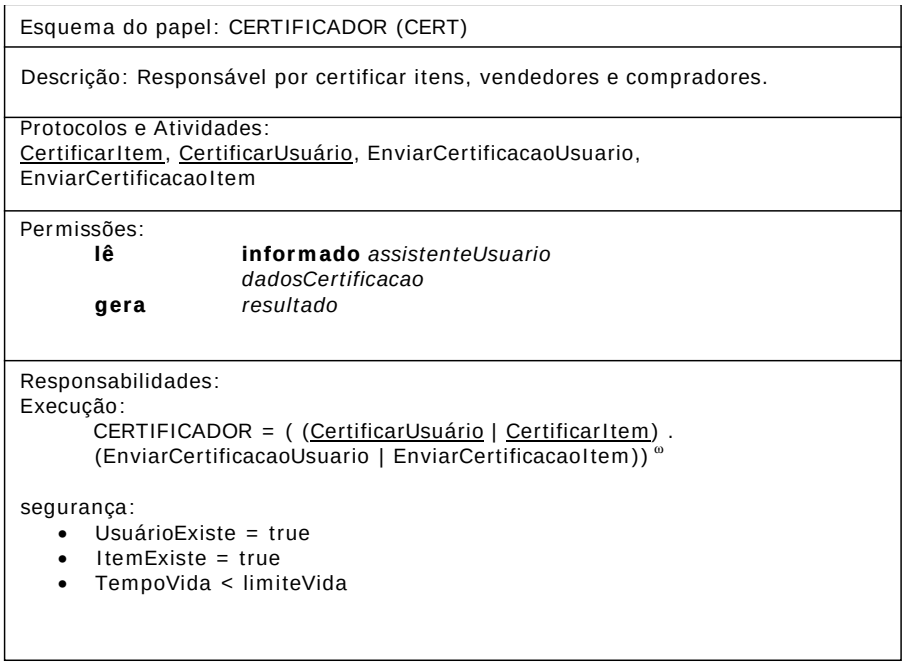

Figura 12 – Modelo de papéis do certificador

Modelo de Interação

Cada um dos protocolos presentes no modelo de papéis participará de uma interação no modelo de interação. O objetivo é indicar como cada protocolo será executado explicitando os papéis que participam da sua execução.

Os modelos de interação podem ser agrupados quando representam uma seqüência de execução de protocolos de diferentes papéis. A representação dos protocolos no modelo de interação sempre é feita a partir do papel iniciador da interação.

Para facilitar o entendimento do modelo de interação, iremos utilizar como exemplo a compra de um item no mercado virtual. Todos os modelos de interação, inclusive os que não tem relação com a compra de um item podem ser verificados no apêndice A. O conjunto de figuras começando na Figura 13 e terminando Figura 21 apresenta os modelos de interação dos protocolos identificados nos modelos de papéis que estão envolvidos na seqüência de compra de um item.

Ao iniciar uma compra, o usuário ordenará a compra de um item de determinado perfil, usando o protocolo do Usuário *OrdenarCompra* (Figura 13). Este protocolo termina gerando um agente comprador para o perfil de item informado. O comprador gerado iniciará a busca

do item no ambiente através de uma atividade interna ao seu papel, indicada por *BuscarItem* no modelo de papéis mostrado na Figura 9.

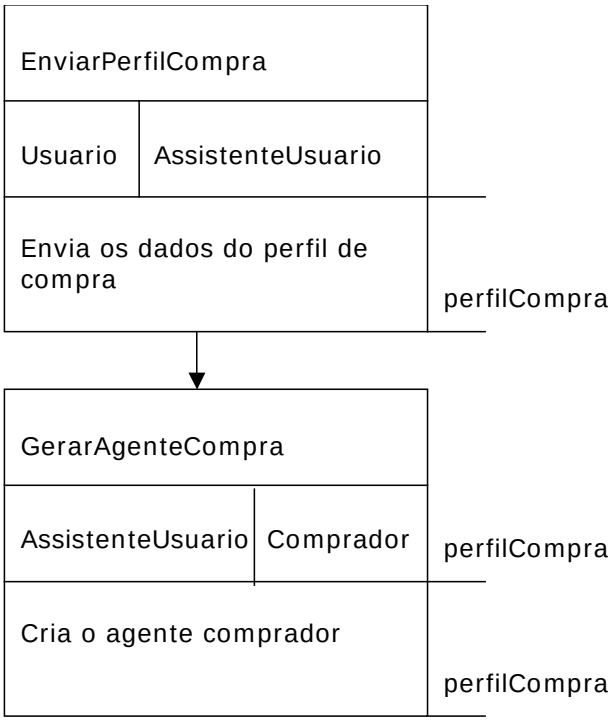

Figura 13 – Modelo de Interação do Protocolo do Usuário *OrdenarCompra*

Ao encontrar um ou mais itens que satisfazem o perfil do bem a comprar, o comprador executará o protocolo *InformarListaItensEncontrados* (Figura 14).

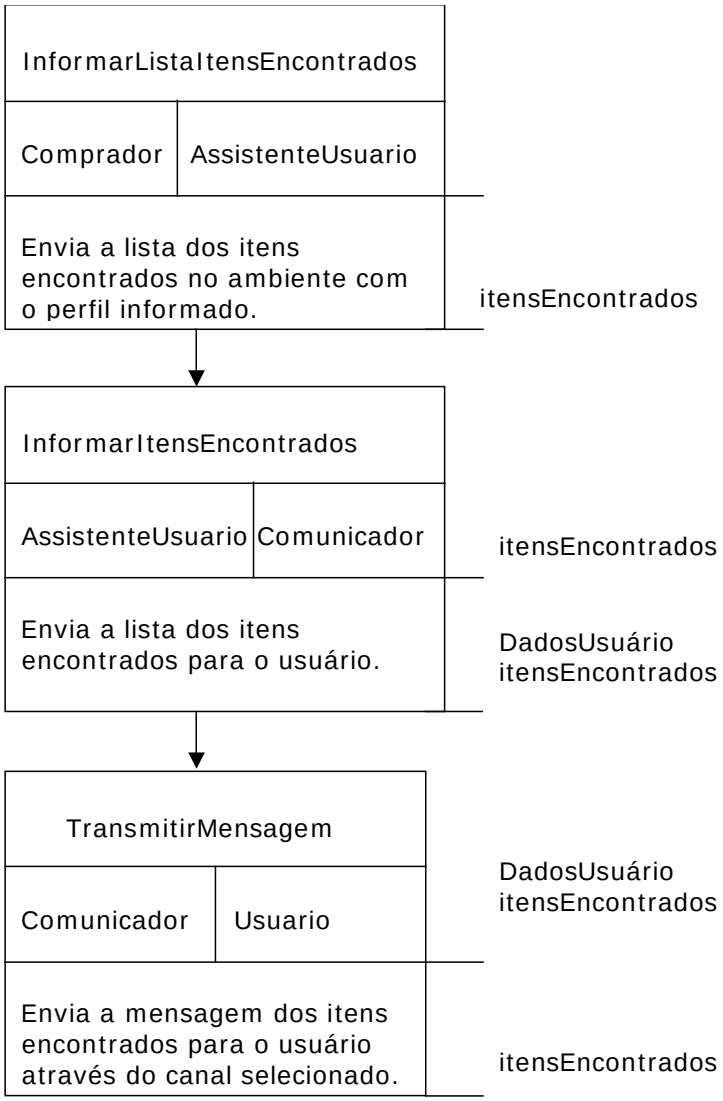

Figura 14 – Modelo de Interação do Protocolo do Comprador *InformarListaItensEncontrados*

O usuário irá então avaliar os itens encontrados e informará, através do protocolo *ConfirmarCompra* (Figura 15) os itens selecionados e seus respectivos parâmetros de negociação.

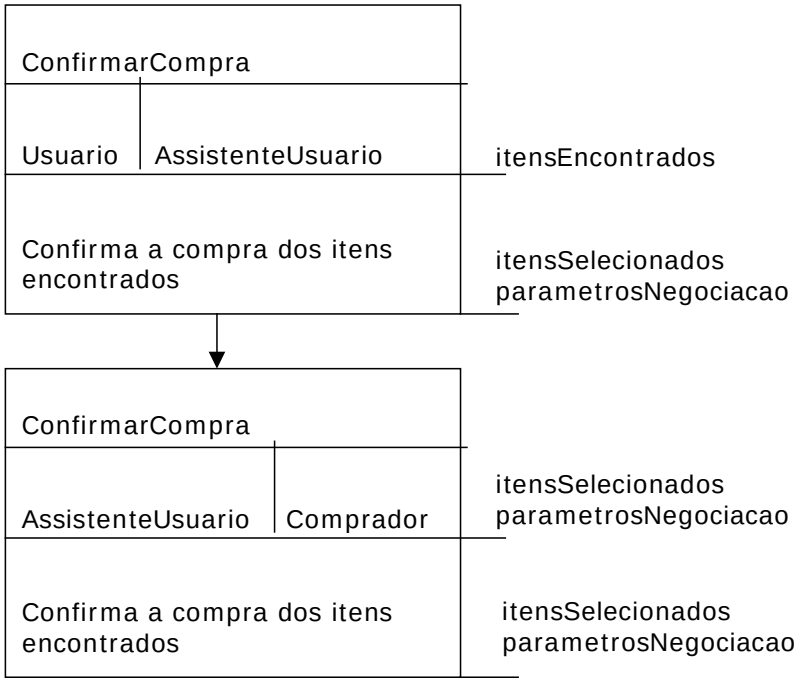

Figura 15 – Modelo de Interação do Protocolo do Usuário *ConfirmarCompra*

O assistente de usuário, com base em suas preferências, executará a certificação do item a ser comprado através do protocolo *SolicitarCertificaçãoItem* (Figura 16).

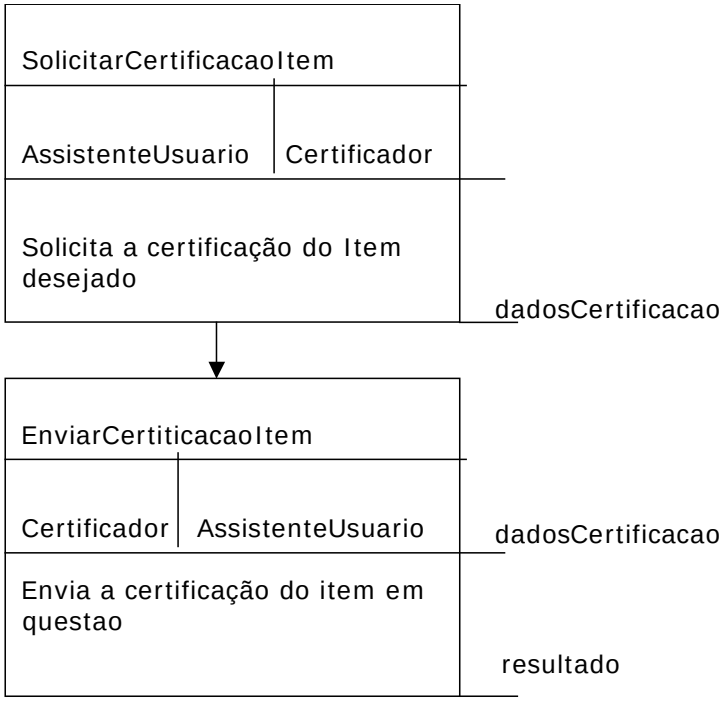

Figura 16 – Modelo de Interação do Protocolo do Assistente de Usuário *SolicitarCertificacaoItem*

Depois de confirmada a autenticidade do item, o agente comprador poderá iniciar a negociação com o agente vendedor em busca do melhor preço utilizando o protocolo do comprador *EnviarMensagemNegociação* (Figura 17).

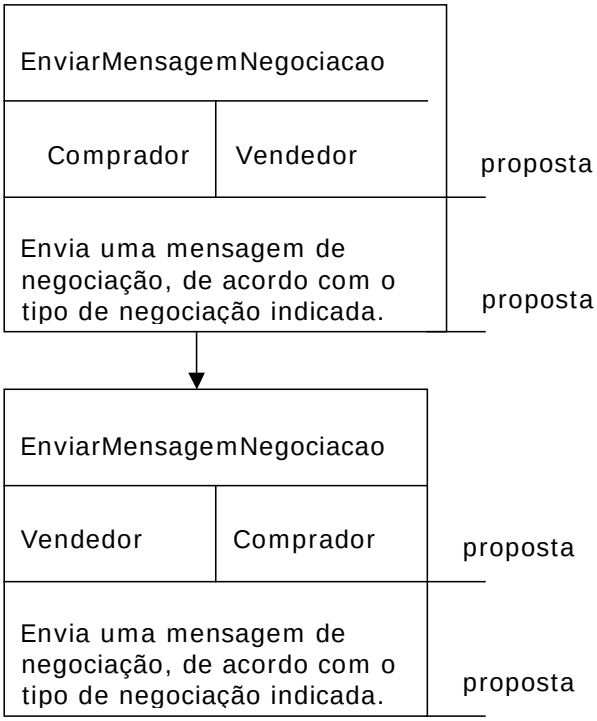

Figura 17 – Modelo de Interação do Protocolo do Comprador *EnviarMensagemNegociação*

Após a negociação, o comprador informa ao usuário, através do protocolo *EnviarResultadoNegociação* (Figura 18), o resultado da negociação.

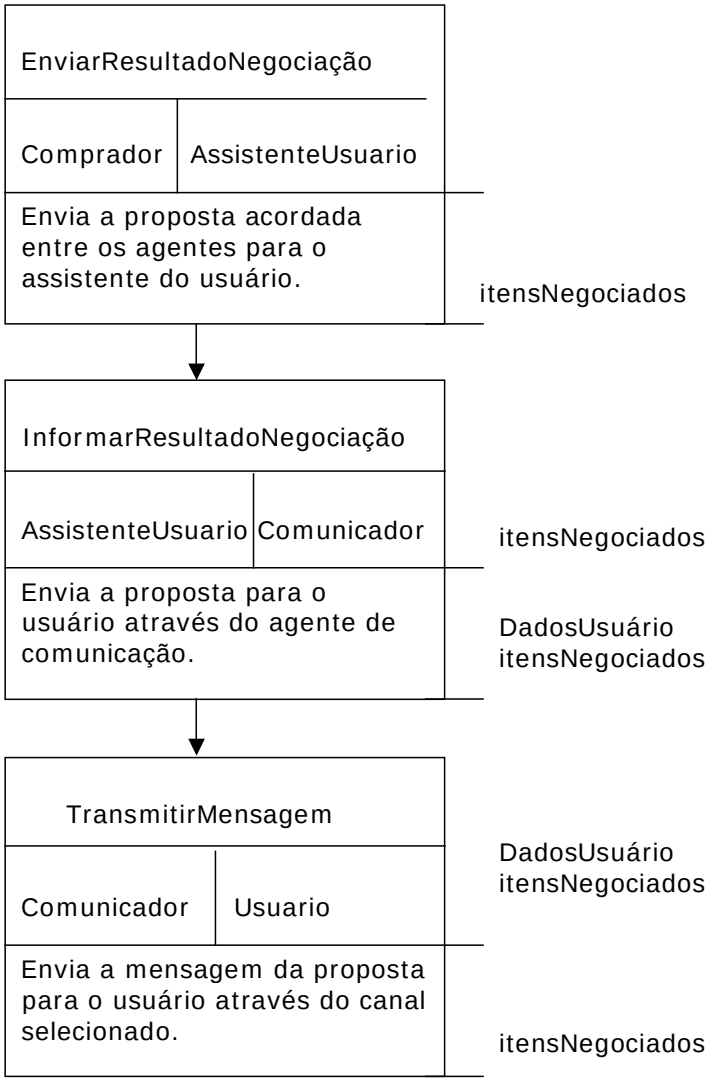

Figura 18 – Modelo de Interação do Protocolo do Comprador *EnviarResultadoNegociação*

O usuário encerrará então a compra do item negociado (protocolo *EncerrarCompra* da Figura 19).

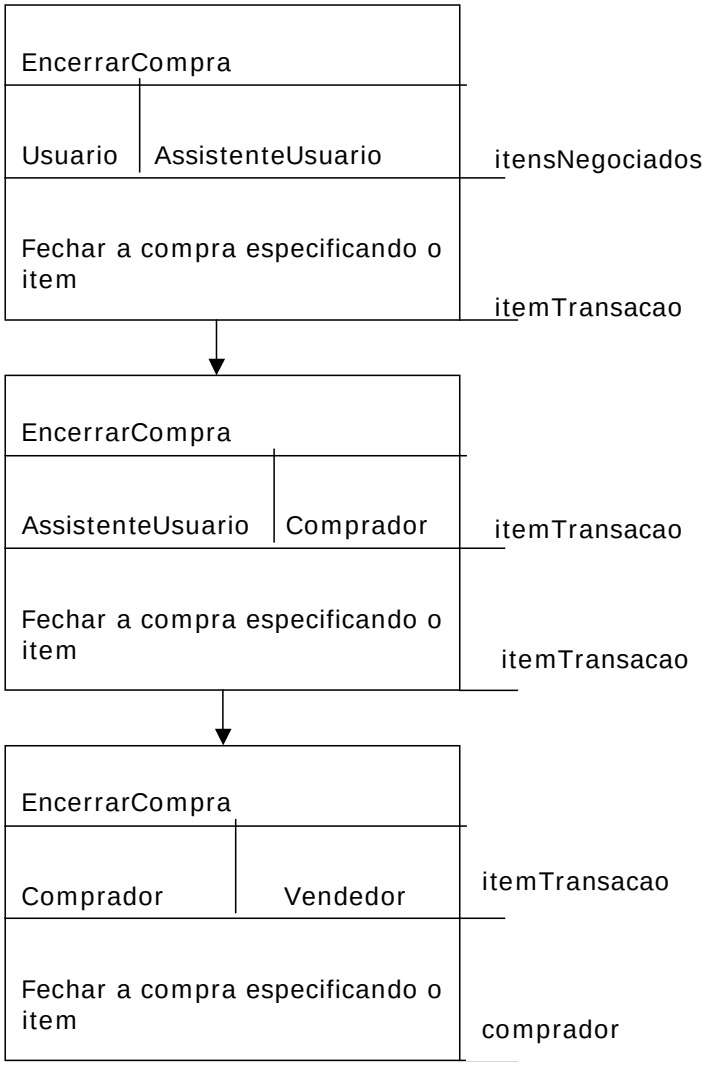

Figura 19 – Modelo de Interação do Protocolo do Usuário *EncerrarCompra*

Após a indicação do usuário, o comprador termina a sua execução informando ao assistente de usuário que a transação está completa, através do protocolo *InformarTérmino* (Figura 20).

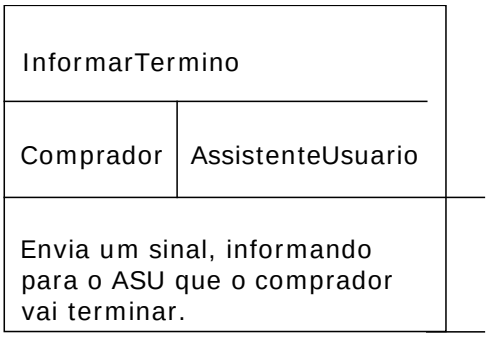

Figura 20 – Modelo de Interação do Protocolo do Comprador *InformarTermino*

O vendedor, por sua vez, informa o usuário que ele representa qual é o resultado da transação através do seu protocolo *InformarTérmino* (Figura 21).

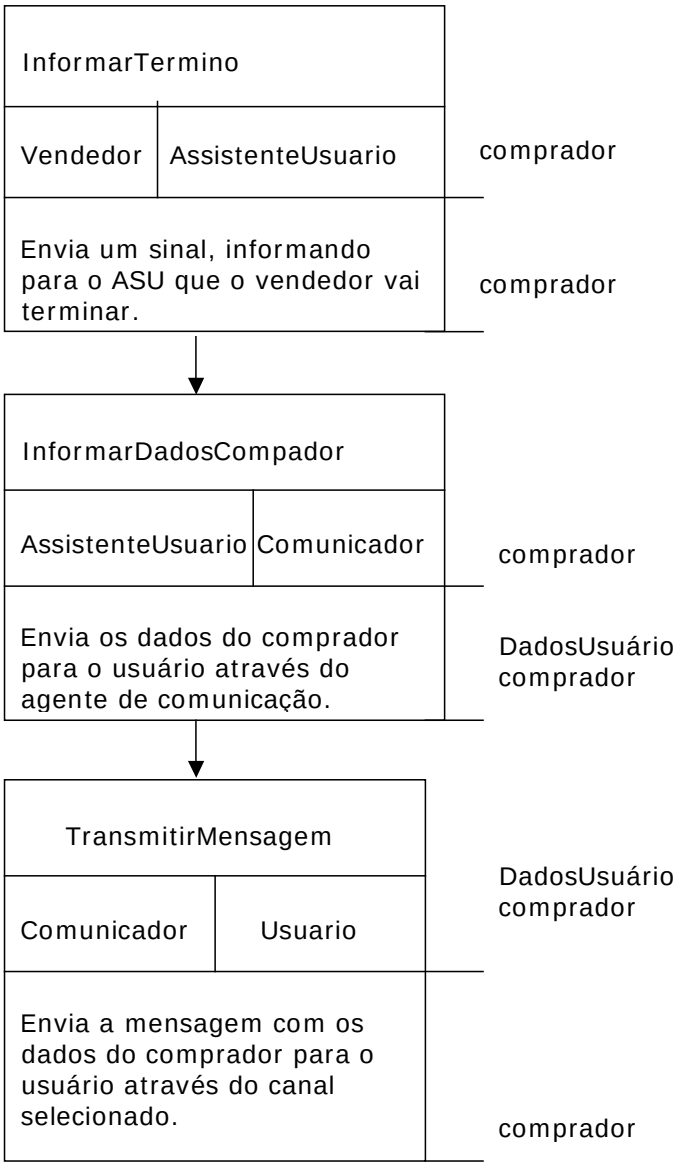

Figura 21 – Modelo de Interação do Protocolo do Vendedor *InformarTermino*

Após a construção dos modelos de papéis e de interação, podemos passar para a fase seguinte da modelagem (fase de projeto). Neste ponto, já foram identificados os papéis, protocolos de interação entre eles, atividades internas a eles, as informações que precisam, que alteram e que geram e o fluxo e as restrições de sua execução.

Fase de Projeto

Com base nos modelos e nas informações geradas durante a análise, dois novos modelos podem ser construídos: o modelo de agentes, usado para mapear papéis em agentes de software, e o modelo de serviços, que identifica os serviços que serão oferecidos por cada tipo de agente. Na aplicação da metodologia Gaia não foi utilizado o modelo de afinidades.

Modelo de Agentes

O modelo de agentes é bastante simples e serve para formalizar o que, durante a fase de análise, já podia ser percebido – que papéis se transformarão em agentes de software e qual a cardinalidade dos agentes de cada tipo no sistema. A atribuição de papéis a agentes segue a lógica funcional de cada papel, em geral teremos um mapeamento de um para um entre papéis e agentes. A Figura 22 apresenta o modelo de agentes referente ao problema proposto neste trabalho.

Observando a Figura 22, pode-se notar que os papéis Usuário e Assistente de Usuário tornaram-se um único agente de software que representará o usuário no mercado virtual. Além disso, é possível notar que só existe um agente Certificador para todo o ambiente e que podem existir de um a três agentes Comunicadores. Um para comunicação através do e-mail, outro para comunicação através do celular e outro para comunicação através do *browser*.

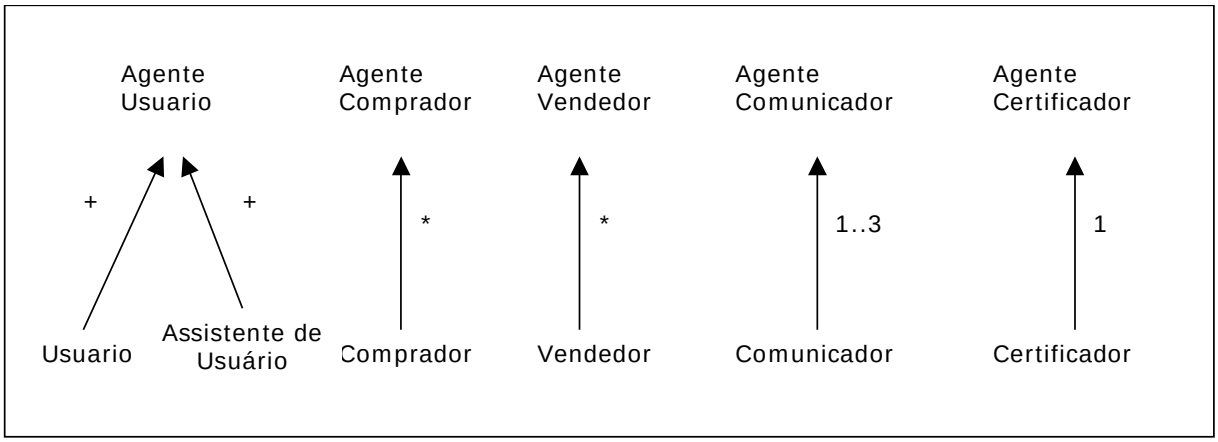

Figura 22 – Modelo de Agentes

Modelo de Serviços

Este modelo captura, a partir dos dados de análise, quais serão os serviços oferecidos por cada agente. No conjunto de tabelas, começando na Tabela 2 e terminando na Tabela 6, listam-se os Modelos de Serviço do mercado virtual, onde a maioria dos serviços foram derivados das atividades e protocolos do Modelo de Papel.

Recordando o exemplo visto anteriormente nos modelos de interação, é possível notar que o protocolo *GerarAgenteCompra* do *AssistenteUsuario* (Figura 8 e Figura 13) deu origem ao serviço *gerar agente compra* (Tabela 2), que recebe o *perfilCompra* com o perfil do item a ser comprado e não possui pré-condições ou pós-condições para execução.

A busca de um item no mercado virtual, representada pela atividade *BuscarItem* do *Comprador* (Figura 9), deu origem ao serviço de mesmo nome do agente Comprador da Tabela 3, que recebe o *perfilCompra*, retorna os *itensEncontrados* e não possui pré- ou pós-condições para execução.

Já o protocolo *InformarListaItensEncontrados* (Figura 9 e Figura 14) derivou o serviço de mesmo nome no modelo de serviços do agente Comprador da Tabela 3. Este serviço recebe a lista de itens encontrados, envia a lista de itens encontrados e os dados do usuário e não possui pré- ou pós-condições.

O protocolo seguinte é *TransmitirMensagem* do agente Comunicador (Figura 11 e Figura 14) que recebe os dados do usuário e os itens encontrados e envia a mensagem com estes dados para o usuário final. Este protocolo derivou o serviço de mesmo nome no modelo de serviços do agente Comunicador da Tabela 5 que recebe *dadosUsuario*, envia a mensagem e possui a pré-condição que o canal esteja disponível, que o usuário exista e que exista uma mensagem.

O protocolo *ConfirmarCompra* do Assistente de Usuário (Figura 8 e Figura 15) deriva o serviço de mesmo nome no modelo de serviços do agente Usuário da Tabela 2. Este serviço recebe os *itensSelecionados* e os *parametrosNegociacao*, e envia os mesmos dados com a pré-condição de que o número de itensSelecionados seja diferente de zero.

O passo seguinte é o processo de negociação (não houve certificação) que equivale aos protocolos do comprador e vendedor *EnviarMensagemNegociacao* (Figura 9, Figura 10 e

Figura 17). Estes protocolos derivam os serviços de mesmo nome no modelo de serviços do agente Comprador (Tabela 3) e do agente Vendedor (Tabela 4) recebendo a proposta e enviando a proposta alterada (contra-proposta). Em ambos os casos, as pré- e pós-condições asseguram que não haja nenhuma negociação, envolvendo o agente, em andamento.

Após a negociação, o comprador informa ao usuário, através do protocolo *EnviarResultadoNegociação* (Figura 9 e Figura 18), o resultado da negociação. Este protocolo deriva o serviço de mesmo nome no modelo de serviços do agente Comprador (Tabela 3) que recebe e envia os *itensNegociados* não possui pré-condições e a pós-condição assegura que ao final da negociação novas negociações envolvendo o agente possam ser iniciadas.

O protocolo *InformarResultadoNegociacao* do Assistente de Usuário (Figura 8 e Figura 18) deu origem ao serviço de mesmo nome no modelo de serviços do agente Usuário (Tabela 2) que recebe *itensNegociados*, envia *dadosUsuario* e *itensNegociados* e não possui pré- ou pós-condições. O próximo protocolo na seqüência é o *TransmitirMensagem* do agente Comunicador (Tabela 5) já descrito acima.

O protocolo *EncerrarCompra* do Assistente de Usuário (Figura 8 e Figura 19) deu origem ao serviço de mesmo nome no modelo de serviços do agente Usuário (Tabela 2) que recebe *resultadoNegociacao*, envia *itemTransação* e possui a pré-condição que hajam itens negociados e a pós-condição que haja pelo menos um item para fazer a transação. O protocolo *EncerrarCompra* do Comprador (Figura 9 e Figura 19) derivou o serviço de mesmo nome do agente Comprador (Tabela 3) que recebe o *itemTransação*, envia *comprador* e não possui pré- ou pós-condições.

O protocolo *InformarTermino* do Comprador (Figura 9 e Figura 20) derivou o serviço de mesmo nome no modelo de serviços do agente Comprador (Tabela 3). Já o protocolo *InformarTermino* do Vendedor (Figura 21 e Figura 10) derivou o serviço de mesmo nome no modelo de serviços do agente Vendedor (Tabela 4) que recebe envia *comprador* e não possui pré- ou pós-condições. O protocolo *InformarDadosComprador* do Assistente de Usuário (Figura 21 e Figura 8) derivou o serviço de mesmo nome no modelo de serviços do agente Usuário (Tabela 2), que recebe *comprador*, envia *dadosUsuario* e *comprador* e não possui pré- ou pós-condições. *TransmitirMensagem*, já descrito acima, é o próximo protocolo da seqüência.

Serviço	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
armazenar dados pessoais	*DadosPessoais*		e-mail válido	verdadeiro
armazenar preferencias	*preferencias*		verdadeiro	verdadeiro
gerar agente compra	*prefilCompra*		verdadeiro	verdadeiro
gerar agente venda	*perfilVenda*		verdadeiro	verdadeiro
confirmar compra	*ItensSelecionados parametrosNegociacao*	*ItensSelecionados parametrosNegociacao*	itensSelecionados <> 0	itensSelecionados <> 0
encerrar compra	*resultadoNegociacao*	*itemTransacao*	itemNegociado <> 0	itemTransacao <> 0
solicitar histórico		*historicoperfisCompra historicoperfisVenda*	verdadeiro	verdadeiro
aterar dados pessoais	*dadosPessoais*		verdadeiro	e-mail valido
aterar preferências	*preferencias*		verdadeiro	Meiocomunicacao valido
listar perfis compra	*usuario*	*listaPerfiscompra*	usuario existe	verdadeiro
listar perfis venda	*usuario*	*listaPerfisvenda*	usuario existe	verdadeiro
alterar perfil compra	*perfilCompra*	*perfilCompra*	verdadeiro	verdadeiro
alterar perfil venda	*perfilVenda*	*perfilVenda*	verdadeiro	verdadeiro
excluir perfil compra	*perfilCompra*		verdadeiro	verdadeiro
excluir perfil venda	*perfilVenda*		verdadeiro	verdadeiro
solicitar situação pesquisa	*perfilCompra*	*itensEncontrados*	verdadeiro	verdadeiro
listar itens negociados		*itensNegociados*	verdadeiro	verdadeiro
solicitar certificação item	*dadosCertificacao*	*resultado*	verdadeiro	verdadeiro
solicitar certificacao usuario	*dadosCertificacao*	*resultado*	verdadeiro	verdadeiro
informar itens encontrados	*itensEncontrados*	*dadosUsuario itensEncontrados*	verdadeiro	verdadeiro
informar resulado negociação	*itensNegociados*	*dadosUsuario itensNegociados*	verdadeiro	verdadeiro
informar dados comprador	*comprador*	*dadosUsuario comprador*	verdadeiro	verdadeiro

Tabela 2 – Modelo de Serviços do Agente Usuário

Serviços	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
buscar item	*perfilcompra*	*itensEncontrados*	verdadeiro	verdadeiro
informar lista itens encontrados	*itensEncontrados*	*itensEncontrados dadosUsuario*	verdadeiro	verdadeiro
enviar mensagem negociacao	*proposta*	*proposta*	Negociando = falso	Negociando = verdadeiro
avaliar proposta recebida	*proposta*	*propostaAlterada*	verdadeiro	verdadeiro
enviar resultado negociacao	*itensNegociados*	*itensNegociados*	verdadeiro	Negociando = falso
encerrar compra	*itemTransacao*	*comprador*	verdadeiro	verdadeiro
informar termino			verdadeiro	verdadeiro
informar tempo expirado			verdadeiro	verdadeiro
terminar			verdadeiro	verdadeiro

Tabela 3 – Modelo de Serviços do Agente Comprador

Serviços	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
publicar item	*perfilVenda*		verdadeiro	verdadeiro
aguardar propostas			verdadeiro	verdadeiro
enviar mensagem negociacao	*proposta*	*proposta*	Negociando = falso	Negociando = verdadeiro
avaliar proposta recebida			verdadeiro	verdadeiro
informar termino	*comprador*	*comprador*	verdadeiro	verdadeiro
informar fim prazo venda			verdadeiro	verdadeiro
terminar			verdadeiro	verdadeiro

Tabela 4 – Modelo de Serviços do Agente Vendedor

Serviços	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
transmitir mensagem	*dadosUsuario*	*mensagem*	CanalDisponível = verdadeiro UsuárioExiste = verdadeiro Mensagem <> vazio	verdadeiro
confirmar envio			verdadeiro	verdadeiro

Tabela 5 – Modelo de Serviços do Agente Comunicador

Serviços	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
certificar item	*dadosCertificacao*		ItemExiste = verdadeiro	verdadeiro
certificar usuario	*dadosCertificacao*		UsuárioExiste = verdadeiro	verdadeiro
enviar certificacao usuario			verdadeiro	verdadeiro
enviar certificacao item			verdadeiro	verdadeiro

Tabela 6 – Modelo de Serviços do Agente Certificador

Vantagens e Desvantagens da metodologia Gaia

O enfoque baseado em papéis da metodologia Gaia é bastante interessante, pois induz a um pensamento voltado a estrutura organizacional do sistema. A identificação dos papéis e de suas funções é fundamental para o entendimento do problema. O fluxo de interação entre os diferentes papéis contribui para o entendimento da forma como os agentes interagem e se relacionam. Embora estes pontos sejam favoráveis a utilização do método, os fatores a seguir tornam difícil a sua utilização, mesmo em problemas com limitado número de agentes como é o apresentado neste trabalho.

Um dos problemas principais que os possíveis usuários da metodologia Gaia irão enfrentar é a falta de documentação com exemplos de uso sobre o método. Na bibliografia disponível, são poucos os artigos que trazem exemplos reais de uso do método.

Os modelos utilizam marcadores em muitos casos que são insuficientes para representar todas as variações possíveis na representação do problema. Um exemplo é o operador de opcional (Tabela 1) das expressões de execução do Modelo de Papéis. Apesar de dizer que a execução é opcional, não é possível representar quando ela está presente e quando não. Isso dificulta o entendimento da seqüência de execução de um determinado papel.

O modelo de interação prevê a indicação das entradas e saídas dos protocolos de interação. Entretanto, a melhor maneira de entender estas entradas e saídas durante a modelagem é fazer as seguintes perguntas:

- O que o protocolo precisa? Ao responder esta pergunta é possível descobrir quais são as entradas das funções.

- O que o protocolo informa? Ao responder esta pergunta é possível descobrir quais são as saídas das funções.

Nos exemplos apresentados na literatura é fácil confundir o que são entradas e saídas em cada função.

O modelo de interação prevê a representação dos protocolos a partir do papel que os origina. Muitas vezes, como é o caso da negociação de bens (Figura 17), dois papéis podem iniciar um

mesmo protocolo. Neste caso, o modelo seria repetido nos dois iniciadores, sendo na verdade componentes uma única seqüência de interação.

A ausência de uma ferramenta CASE para o desenvolvimento de software através da metodologia dificulta e limita seu uso.

Capítulo 4

Mapeamento de Gaia para técnicas OO

A metodologia Gaia gera modelos cujos elementos são diferentes dos elementos encontrados em uma modelagem orientada a objetos. De acordo com [13, 12], ao final da fase de projeto da metodologia Gaia, os modelos gerados constituem a arquitetura do sistema multi-agentes que pode ser implementada usando técnicas de projeto orientadas a objetos. A questão é como partir de modelos da metodologia Gaia, cujos elementos são, entre outros, papéis, responsabilidades, atividades, protocolos, permissões e serviços para modelos com elementos da programação orientada a objetos como classes, métodos, atributos, objetos, etc.

Faz-se necessário, então, um mapeamento dos modelos gerados pela metodologia Gaia para modelos orientados a objetos a fim de viabilizar a implementação, em linguagem orientada a objetos, do sistema modelado de acordo com a metodologia Gaia. A Figura 23 ilustra os elementos da metodologia Gaia e seus relacionamentos.

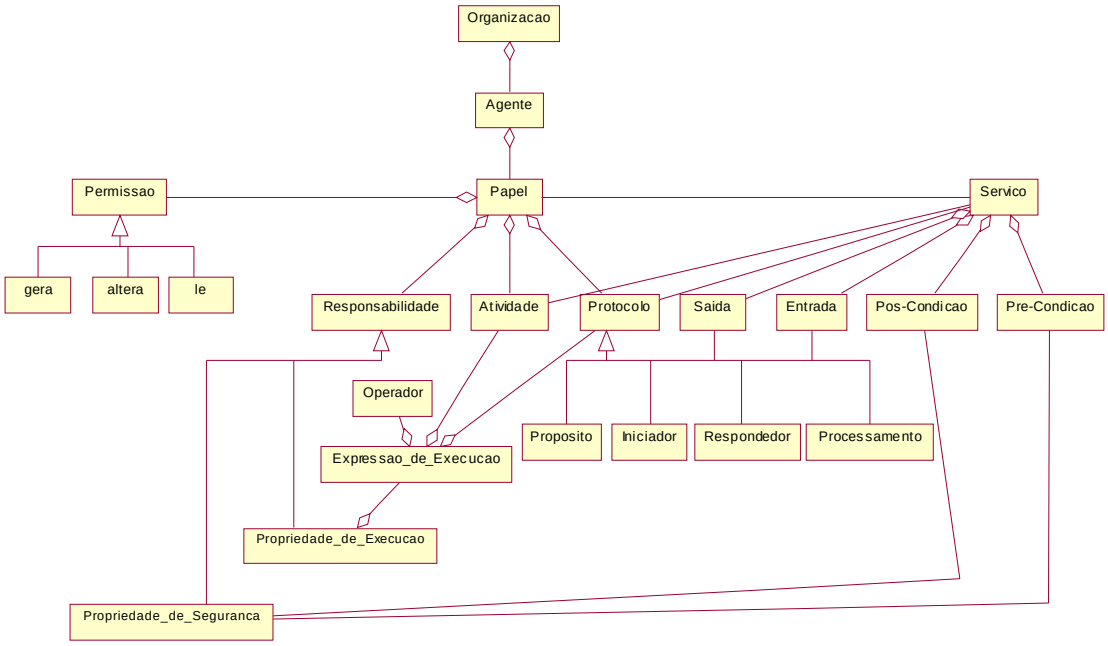

Figura 23 – Elementos da metodologia Gaia e seus relacionamentos

No corrente capítulo avalia-se o mapeamento da metodologia Gaia para duas modelagens orientadas a objetos de desenvolvimento de sistemas multi-agentes. A primeira modelagem orientada a objetos utilizada para o mapeamento da metodologia Gaia foi o *MAS Framework*[14] que é um *framework*[11] para desenvolvimento de sistemas multi-agentes desenvolvido por José Alberto Sardinha no Laboratório Teccomm[8]. Este *framework*, bem como os detalhes do mapeamento, são descritos na seção 4.1. A segunda modelagem orientada a objetos utilizada para o mapeamento da metodologia Gaia foi o design pattern[6] para Sistemas Multi-Agentes[21] desenvolvido por Viviane Torres no Laboratório Teccomm[8] que descreve, através de um diagrama de classes e diagramas de seqüência em UML, o desenvolvimento de sistemas multi-agentes. Esta última abordagem e os detalhes do mapeamento são expostos na seção 4.2.

4.1 MAS Framework

O *MAS Framework* oferece ao usuário do framework o reuso de código e *design* para o desenvolvimento de um sistema multi-agentes. Seu design modela os agentes numa única

classe (Agent) que representa as propriedades dos agentes, como por exemplo autonomia e interação. Além disso, o *framework* oferece uma infra-estrutura para a comunicação entre os agentes através do *Tspaces*[20], que permite o envio de mensagem direta entre os agentes e o envio de mensagem através de *blackboard*.

Os pontos de flexibilização do *MAS Framework* são três: a classe *Agent*, a classe *AgentMessage* e a classe *AgentBlackboardInfo*. Estas classes precisam ser especializadas de acordo com os agentes do problema e de acordo com as mensagens específicas do sistema que se pretende desenvolver. Além disso, o usuário do framework precisa redefinir alguns métodos da classe *Agent* como por exemplo: *initialize*, *run*, *processMsg*, etc. A Figura 24 apresenta o diagrama de classes do MAS Framework.

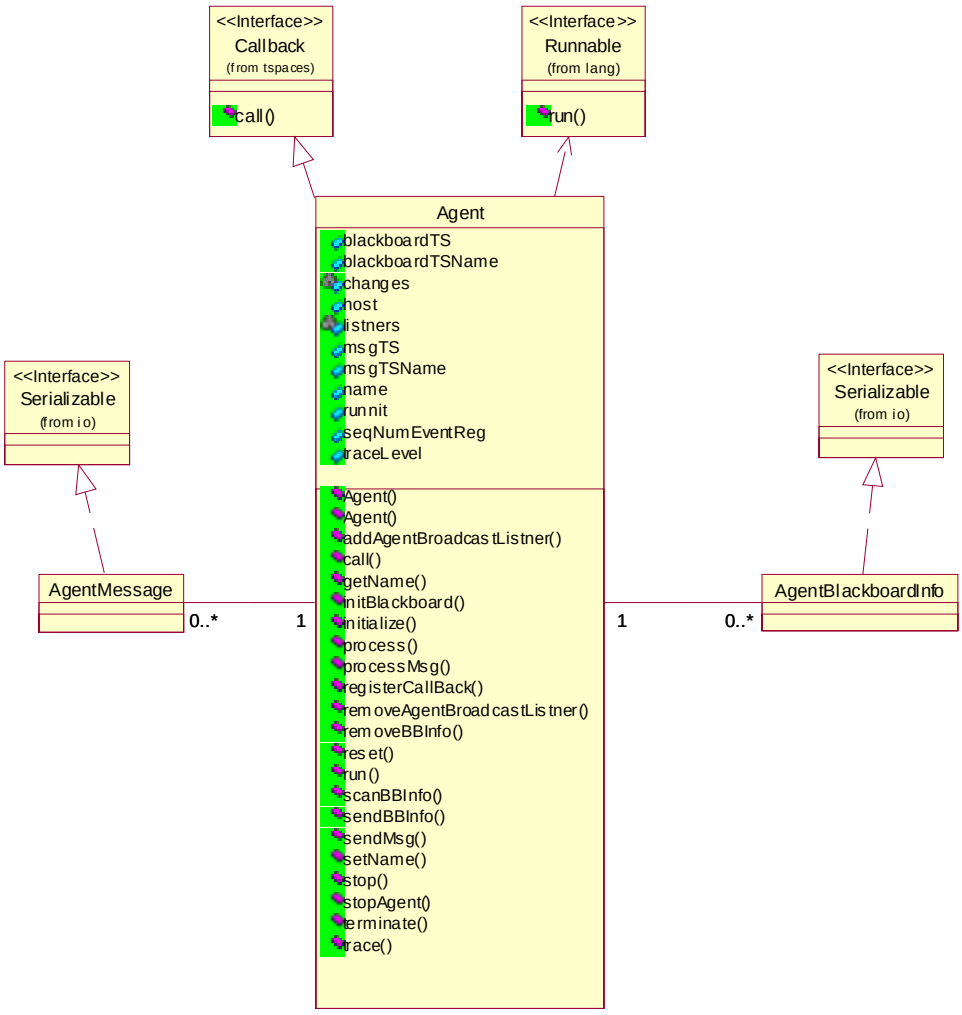

Figura 24 – Diagrama de classes do *MAS Framework*

Durante a implementação do problema surgiu a necessidade de se representar os agentes do sistema em duas classes ao invés de uma única classe como define o *MAS Framework*. Por isso nos diagramas de UML a seguir, é possível verificar que todas as sub-classes de Agent estão associadas a uma outra classe. Ao final desta seção serão expostas as questões de implementação que levaram a esta decisão.

4.1.1 Mapeamento

Esta seção apresenta o mapeamento dos elementos da metodologia Gaia para os elementos de uma modelagem OO feita de acordo com o *MAS Framework*.

Modelo de agentes derivam classes do diagrama de classes

O primeiro modelo da metodologia Gaia escolhido para o mapeamento foi o modelo de agentes definido na Figura 22. Nele foram especificados a existência de cinco agentes participantes do sistema: Agente Usuário, Agente Comprador, Agente Vendedor, Agente Comunicador e Agente Certificador. Como o *MAS Framework* define os agentes numa única classe, houve um mapeamento direto entre os agentes do modelo de agentes do Gaia para os agentes do *MAS Framework*, que são especializações da classe Agent. A Figura 25 ilustra parte do diagrama de classes do sistema multi-agentes resultante deste mapeamento.

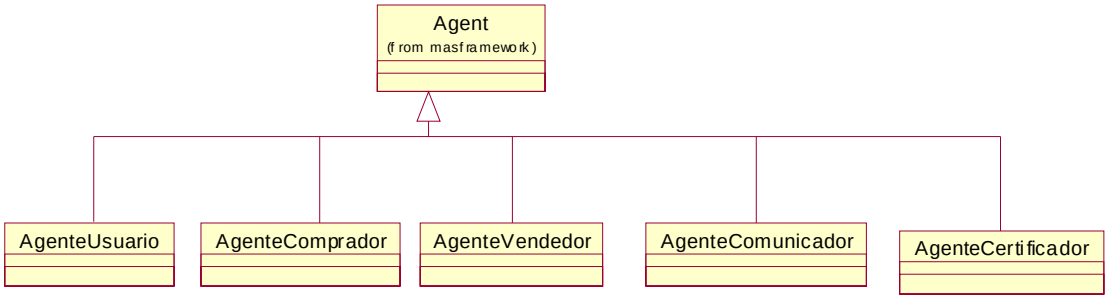

Figura 25 – Diagrama de classes dos agentes do sistema multi-agentes

Serviços do agente derivam métodos das classes

Os métodos das classes que representam os agentes foram derivados dos serviços do Modelo de Serviços. Por exemplo, o serviço *buscar item* do Agente Comprador do modelo de serviços da metodologia Gaia foi mapeado para o método *buscarItem* da classe *AgenteComprador*. A

Tabela 3 apresenta o modelo de serviços do *AgenteComprador* com o serviço *buscar item* e a Figura 26 mostra os métodos da classe *AgenteComprador* com o método *buscarItem*. Os serviços do Agente Comprador tornaram-se métodos ora da classe *AgenteComprador* ora da classe *TrataMsgComp*. O critério utilizado para esta separação leva em consideração o tipo de serviço do agente.

Se for um serviço derivado de uma atividade do papel comprador, então o método correspondente será um método da classe *AgenteComprador*. O método *buscarItem()* é um exemplo deste caso pois corresponde a atividade de mesmo nome e encontra-se na classe *AgenteComprador*.

Se for um serviço derivado de um protocolo do papel comprador onde o comprador é *respondedor* do protocolo, então o método correspondente será um método da classe *TrataMsgComp*. O método *negocia()* é um exemplo deste caso, pois corresponde ao protocolo *EnviarMensagemNegociacao* do papel comprador, e encontra-se na classe *TrataMsgComp*.

Se for um serviço derivado de um protocolo do papel comprador onde o comprador é *iniciador* do protocolo, então o método correspondente será um método da classe *AgenteComprador*. O método *informarListaitensencontrados()* é um exemplo deste caso pois corresponde ao protocolo de mesmo nome e encontra-se na classe *AgenteComprador*.

Cabe ressaltar que durante a implementação do problema, houve alguns casos em que não foi possível separar os métodos desta forma. A conseqüência é que alguns métodos que segundo esta regra estariam na classe *AgenteComprador* se encontram na classe TrataMsgComp. Em geral isto ocorre quando o método foi derivado de um protocolo onde o comprador atua como *respondedor* e em seguida como *iniciador*.

Permissões do papel e entradas e saídas dos serviços derivam atributos das classes

Já os atributos das classes, que compõem os agentes, foram derivados das entradas e saídas do Modelo de Serviços e das permissões do Modelo de Papéis. A permissão de leitura para a informação *perfilCompra* por exemplo, está na Figura 9 do Modelo de Papéis do *Comprador.* Já na Tabela 3 este mesmo elemento aparece como uma entrada do serviço *buscar item* do modelo de serviços do *AgenteComprador*. Com o mapeamento, o elemento *perfilCompra* passa a ser o atributo de mesmo nome da classe *AgenteComprador*, que pode ser verificado na

Figura 26. As permissões do papel *Comprador* e as entradas e saídas do Modelo de Serviços do *AgenteComprador* são distribuídas em atributos ora na classe *AgenteComprador* e ora na classe *TrataMsgComp*. A Figura 26 ilustra as classes que compõem o Agente Comprador com seus atributos e métodos correspondentes.

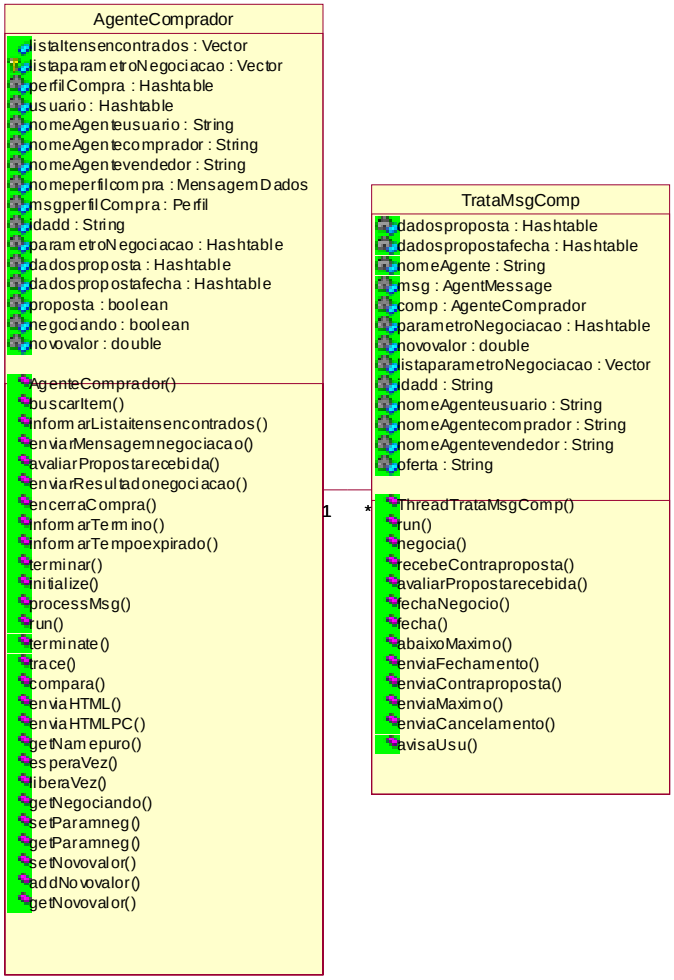

Figura 26 – Diagrama de classes do Agente Comprador

Pré- e pós-condições derivam mecanismo de monitor

Para cada serviço do Modelo de Serviços do Gaia identifica-se as pré- e pós-condições. No serviço *enviar mensagem negociação*, do modelo de serviços do Agente Comprador da Tabela 3, temos a pré-condição que garante que não se têm negociações em andamento no momento de iniciar o processo de negociação (*negociando=falso*) e a pós-condição que garante que após o início do processo de negociação não haverá novas negociações que

envolvam o agente comprador ocorrendo ao mesmo tempo (*negociando=verdadeiro*). Na implementação, esta restrição é traduzida na inclusão de um monitor[5] que garante que um agente comprador só irá executar uma negociação com um agente vendedor de cada vez. Se o agente comprador estiver com uma negociação em andamento e receber um novo pedido de negociação, a *thread*[5], responsável pelo pedido de negociação, ficará em espera (*wait*) até que a negociação corrente se encerre. Quando a negociação corrente é encerrada, as *threads*, responsáveis por pedidos de negociação em espera, são notificadas (*notify*) e, em seguida, tentam novamente iniciar a negociação. A Tabela 7 ilustra o trecho do código que implementa este mecanismo (*wait/notify*). Em [5] este mecanismo é apresentado como uma variação do *pattern Before/After* de programação concorrente, utilizado para resolver problemas de concorrência similares a este da negociação.

```
public synchronized void esperaVez()
{
            while (negociando)
                        wait();
            negociando = true;
}

public synchronized void liberaVez()
{
            negociando=false;
            notifyAll();
}
```

Tabela 7 – Trecho de código que implementa o mecanismo wait/notify do Agente Comprador

Modelo de interação deriva diagrama de seqüência

O modelo de interação da metodologia Gaia foi muito útil durante a implementação do sistema, pois ele representa as interações entre os papéis participantes do sistema. Como os papéis são mapeados para os agentes do sistema, no modelo de agentes da fase de projeto da metodologia Gaia, pode-se dizer então, que este modelo define a interação entre os agentes do sistema. Por isso faz-se um mapeamento bastante direto entre os modelos de interação da metodologia Gaia e os diagramas de seqüência de UML do problema.

Para um completo entendimento dos diagramas de seqüência, é necessário conhecer a estrutura completa do diagrama de classes do problema definido na Figura 27. Neste diagrama, as classes que representam os agentes são apresentadas:

- Agente Usuário é composto pelas classes *AgenteUsuario* e *TrataMsgUsu;*

- Agente Comprador é composto pelas classes *AgenteComprador* e *TrataMsgComp*;
- Agente Vendedor é composto pelas classes *AgenteVendedor* e *TrataMsgVend*;
- Agente Comunicador é composto pelas classes *AgenteComunicador* e *TrataMsgCom*;
- Agente Certificador não foi utilizado nesta implementação e por isso não está com a representação completa no diagrama.

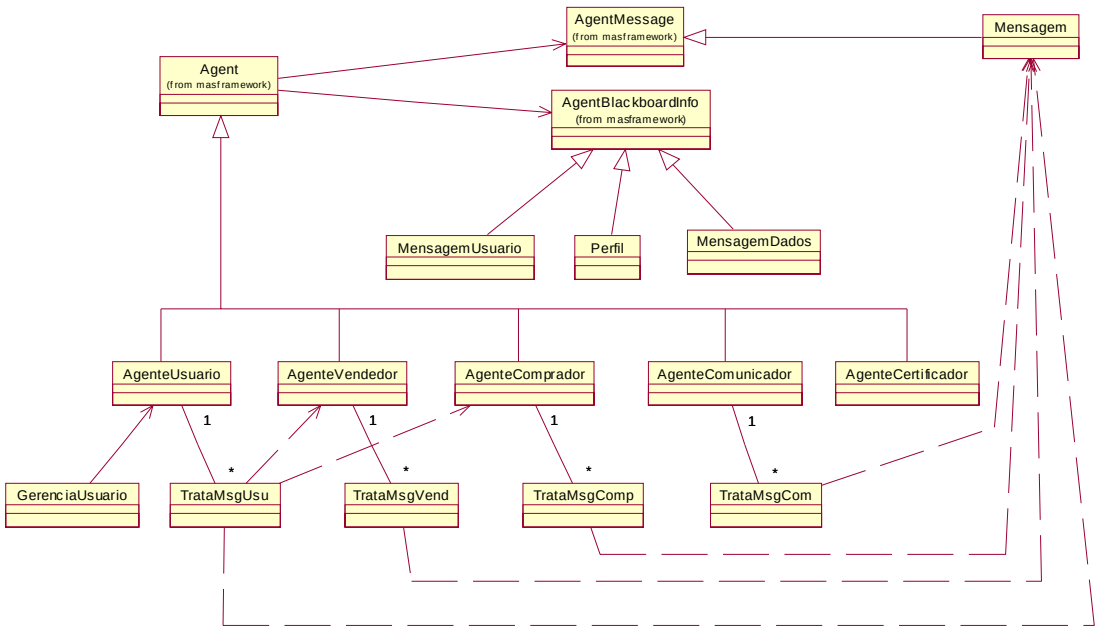

Figura 27 – Diagrama de classes do problema sem os atributos e métodos

Na seqüência, utiliza-se o mesmo exemplo da compra de um item no mercado virtual do capítulo 3 para expor a correspondência entre os modelos de interação do Gaia e os diagrama de seqüência de UML. O protocolo do Usuário *OrdenarCompra* da Figura 13 corresponde ao diagrama de seqüência da Figura 28. Comparando os modelos pode-se perceber grande semelhança entre eles. Este diagrama de seqüência (Figura 28) representa a chegada de um pedido de compra que deve ser tratado pelo *AgenteUsuario*. O *AgenteUsuario* cria a *thread* responsável por processar a mensagem (pedido de compra) recebida (método *start()*). Em seguida o método *criaComprador()* determina a criação de um *AgenteComprador* (construtor da classe) e a ativação deste agente (método *process()*). A lista de agentes compradores do usuário é incrementada através da chamada do método *incluiComprador()*. A principal tarefa

do agente recém criado é a de buscar itens no mercado virtual através do método *buscarItem()* derivado da atividade *BuscarItem* da metodologia Gaia. Este método determina a busca (*scan()*) de *tuplas*, no *blackboard* do Tspaces, que contenham a chave “anuncio”.

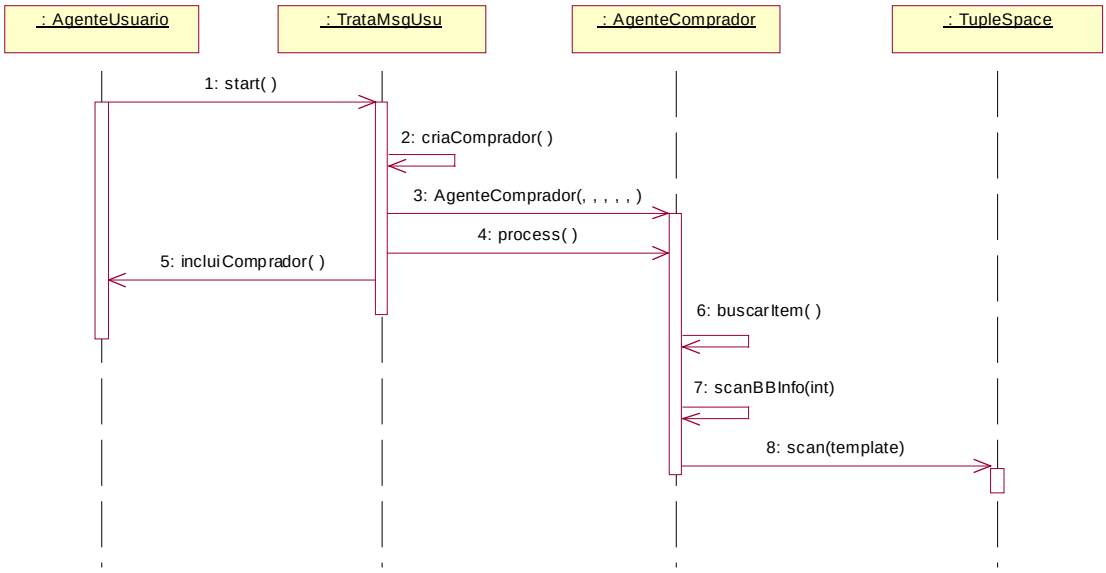

Figura 28 – Diagrama de seqüência equivalente ao protocolo do Usuário *OrdenarCompra*

Já o protocolo do Comprador *InformarListaItensEncontrados* da Figura 14 é mapeado para o diagrama de seqüência da Figura 29. Este diagrama de seqüência representa o instante em que o agente comprador encontra um ou mais itens no mercado virtual e avisa ao seu usuário final através do agente usuário e do agente comunicador. O método *informarListaitensencontrados()* do *AgenteComprador* envia uma mensagem para o *AgenteUsuario* que a recebe através do método *processMsg()* que por sua vez determina a criação de uma *thread* da classe *TrataMsgUsu* para tratamento da mensagem recebida (*start()*). O método *informarItensencontrados()* de *TrataMsgUsu* é executado para enviar os itens encontrados para o usuário final através do *AgenteComunicador* que é o responsável pela comunicação com o usuário final.

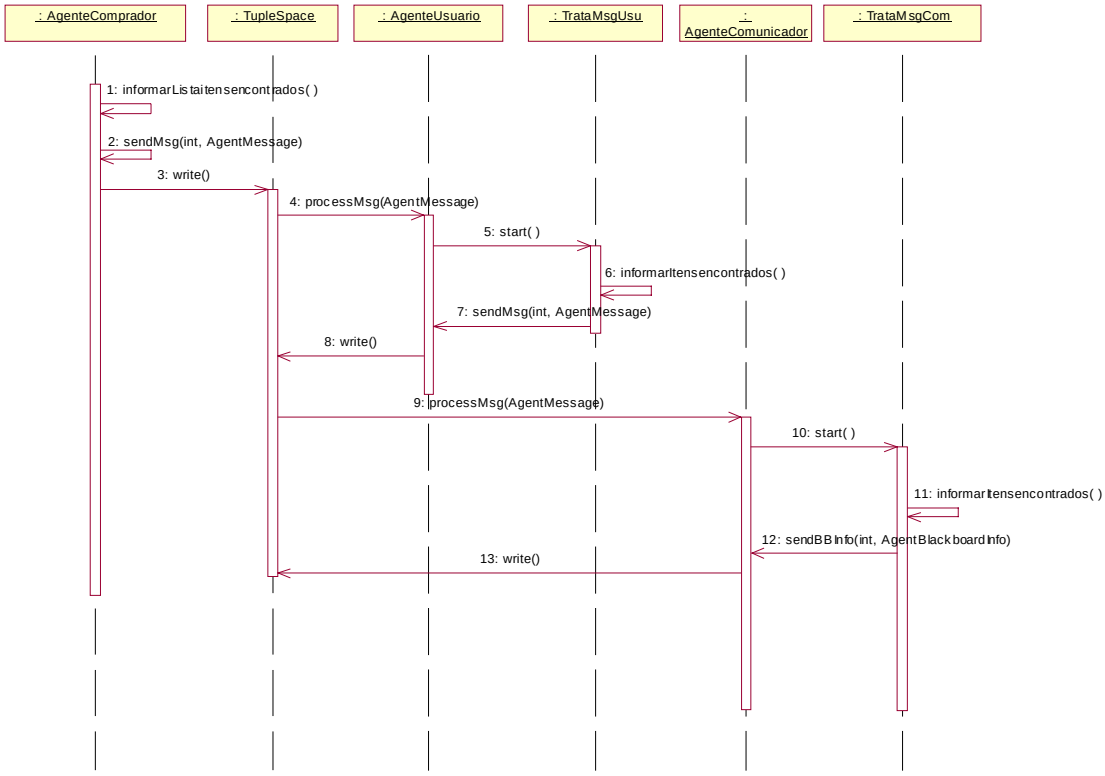

Figura 29 – Diagrama de seqüência equivalente ao protocolo do Comprador *InformarListaItensEncontrados*

A seguir o protocolo do Usuário *ConfirmarCompra* que pode ser observado na Figura 15 é mapeado para o diagrama de seqüência da Figura 30. Este diagrama de seqüência se inicia no momento que o usuário final seleciona um item da lista de itens encontrados e especifica os parâmetros de negociação do item. O agente usuário envia através do *Tspaces* uma mensagem para seu agente comprador dando início ao processo de negociação.

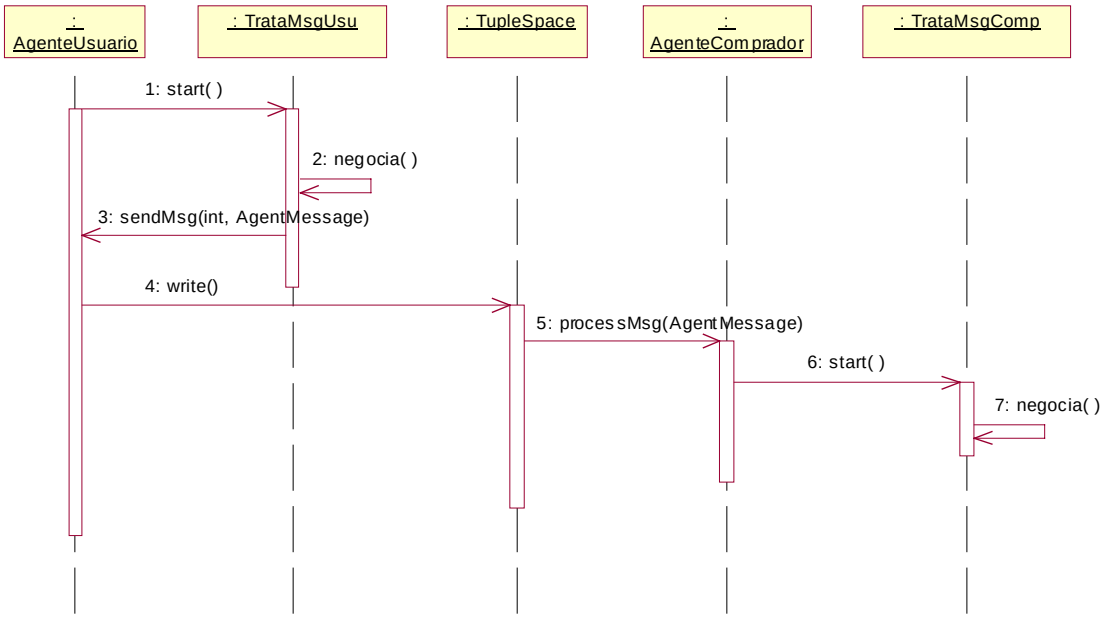

Figura 30 – Diagrama de seqüência equivalente ao protocolo do Usuário *ConfirmarCompra*

A negociação representada pelo protocolo do Comprador *EnviarMensagemNegociação* da Figura 17 deriva o diagrama de seqüência da Figura 31. Neste diagrama de seqüência pode-se observar os passos envolvidos na negociação que começa com uma proposta inicial do agente comprador para o agente vendedor, que por sua vez irá avaliar a proposta e responder com um contra-proposta para o agente comprador, que por sua vez irá avaliar a proposta e encaminhar uma contra-proposta e assim sucessivamente.

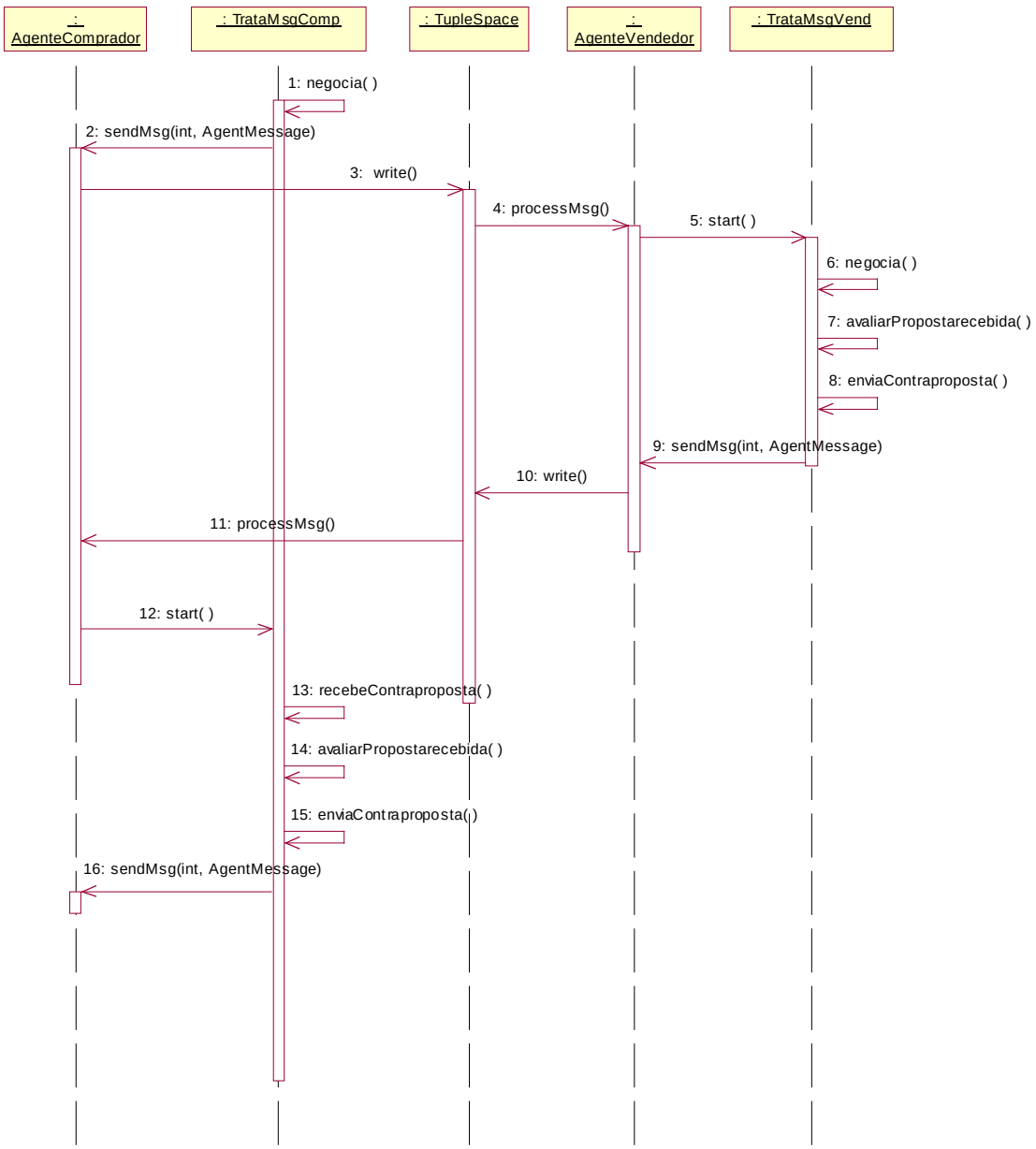

Figura 31 – Diagrama de seqüência equivalente ao protocolo do Comprador *EnviarMensagemNegociação*

O envio do resultado da negociação representado pelo protocolo do Comprador *EnviarResultadoNegociação* da Figura 18 corresponde ao diagrama de seqüência da Figura 32. Este diagrama de seqüência ocorre quando uma negociação chegou ao fim e o agente comprador precisa informar ao seu usuário final os resultados alcançados. Este envio é feito através do agente usuário e do agente comunicador.

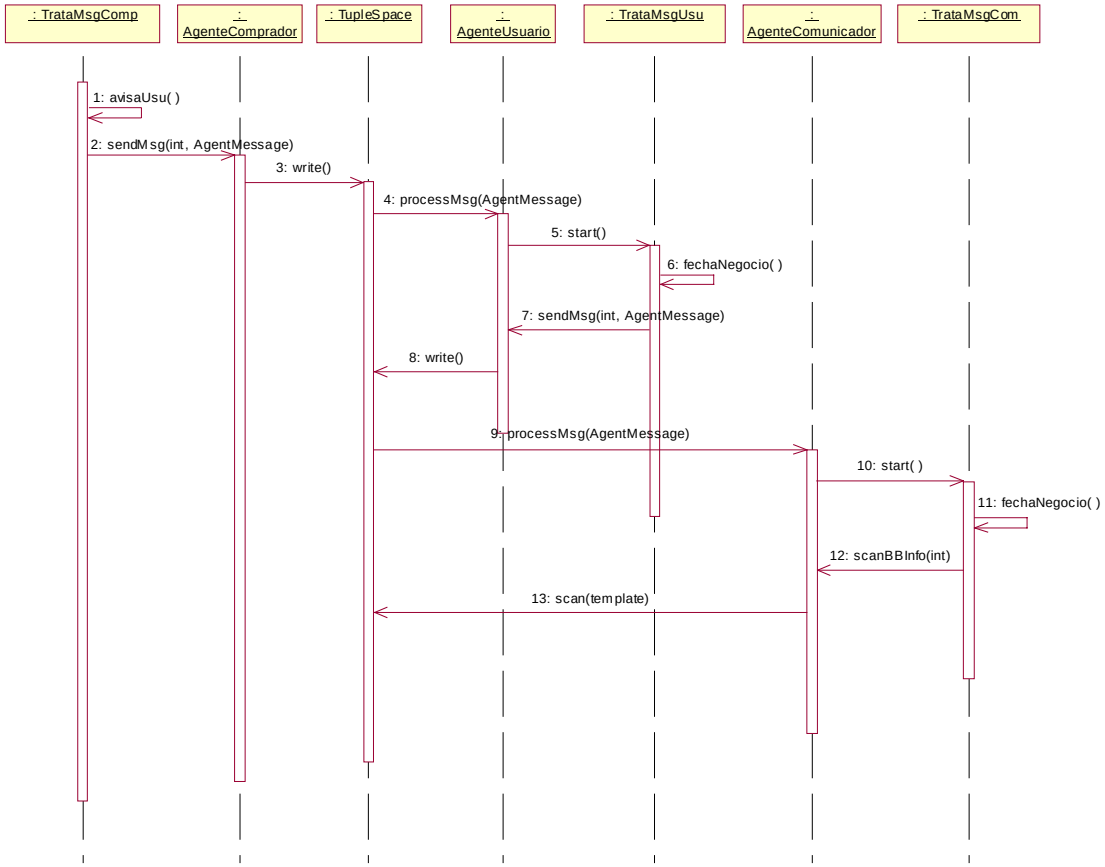

Figura 32 – Diagrama de seqüência equivalente ao protocolo do Comprador *EnviarResultadoNegociação*

Finalmente, o protocolo do Vendedor *InformarTermino* da Figura 21 é mapeado no diagrama de seqüência da Figura 33. Este diagrama de seqüência equivale ao momento em que o agente vendedor avisa ao seu usuário final o resultado da negociação. Isto é feito através do agente usuário e através do agente comunicador.

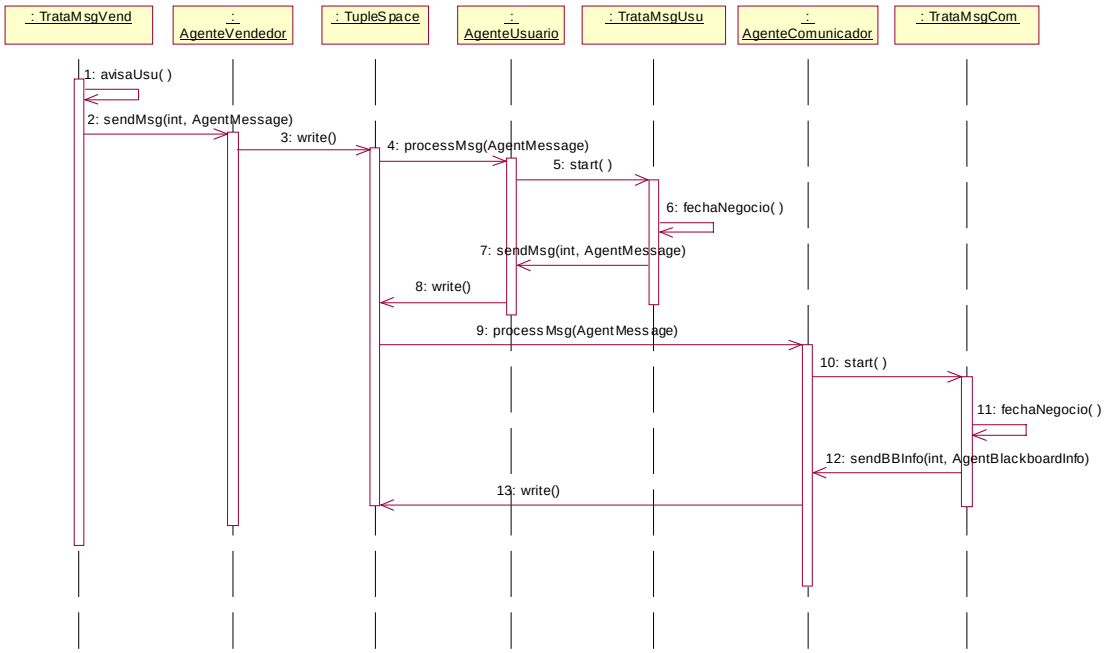

Figura 33 – Diagrama de seqüência equivalente ao protocolo do Vendedor *InformarTermino*

Repare que os agentes que interagem são os mesmos nos Modelos de Interação e nos Diagramas de Seqüência, com a diferença que os diagramas de seqüência possuem mais detalhes de implementação, como por exemplo, as chamadas de métodos entre os objetos e a representação do *Tspaces* que é a infra-estrutura através da qual os agentes se comunicam.

Duas classes para representar cada agente

Nos Diagramas de Seqüência acima e no Diagrama de Classes da Figura 27 podemos observar a existência de uma segunda classe para cada agente do sistema. Ocorre que as classes dos agentes: *AgenteVendedor*, *AgenteComprador*, *AgenteUsuario*, *AgenteComunicador* estão associadas às classes, *TrataMsgVend*, *TrataMsgComp*, *TrataMsgUsu* e *TrataMsgCom* respectivamente. A função principal desta classe, adicionada ao modelo, é a de tratamento das mensagens recebidas pelo agente.

O *Tspaces* possui um mecanismo de envio de mensagem direta que causa a interrupção da seqüência de execução do agente destinatário, a qualquer momento, para que o agente trate a chegada de uma mensagem. Como várias mensagens podem chegar ao mesmo tempo, é preciso estabelecer mecanismos que garantam a consistência dos atributos do objeto agente

receptor da mensagem. A solução adotada foi criar, a cada mensagem que chega no agente, uma nova *thread* para tratamento da mensagem recebida. Esta *thread* é sempre uma instância desta classe de tratamento de mensagem, respectiva do agente destinatário.

Durante a implementação optou-se por inserir, nesta classe de tratamento de mensagem, os métodos de envio de mensagem do agente. Como conseqüência, esta classe trata do envio e recebimento das mensagens do agente, exercendo um papel fundamental na interação do agente. De acordo com a teoria dos agentes, vista no capitulo 1, esta classe trata da propriedade de interação do agente.

Pretende-se estender o *MAS Framework* de forma que a classe *Agent* esteja associada a uma outra classe responsável pelo tratamento das mensagens recebidas. Desta forma, os usuários do *MAS Framework* terão menos trabalho para resolver o problema de concorrência que ocorre quando o agente recebe diversas mensagens do *Tspaces* no mesmo instante.

4.2 Design Pattern para sistemas multi-agentes

Este design pattern foi construído de acordo com a definição de agentes do grupo OMG[19]. Diferente do *MAS Framework*, este design pattern possui o conceito de que os agentes são constituídos de várias classes. Este conjunto de classes do agente representa o próprio agente, suas propriedades (interação, adaptação, autonomia, etc.) e seus estados mentais (metas, planos, crenças e tarefas). O pattern define um diagrama de classes para a representação interna dos agentes e um diagrama de seqüência definindo de que forma ocorre a troca de mensagens entre as classes que compõem o agente. Detalhes de implementação não são especificados neste padrão de modelagem. A troca de mensagens entre os agentes ocorre através do ambiente que possui as características de um *blackboard*. A modelagem define que o agente lê e grava mensagens no ambiente através de sensores e efetuadores respectivamente, mas não define detalhes da implementação deste ambiente (*blackboard*). A Figura 34 ilustra o diagrama de classes desta modelagem. Cabe ressaltar que pequenas adaptações/alterações foram feitas ao diagrama de classes original. A classe *Biblioteca de Planos*, por exemplo, não foi utilizada e alguns relacionamentos entre as classes foram modificados.

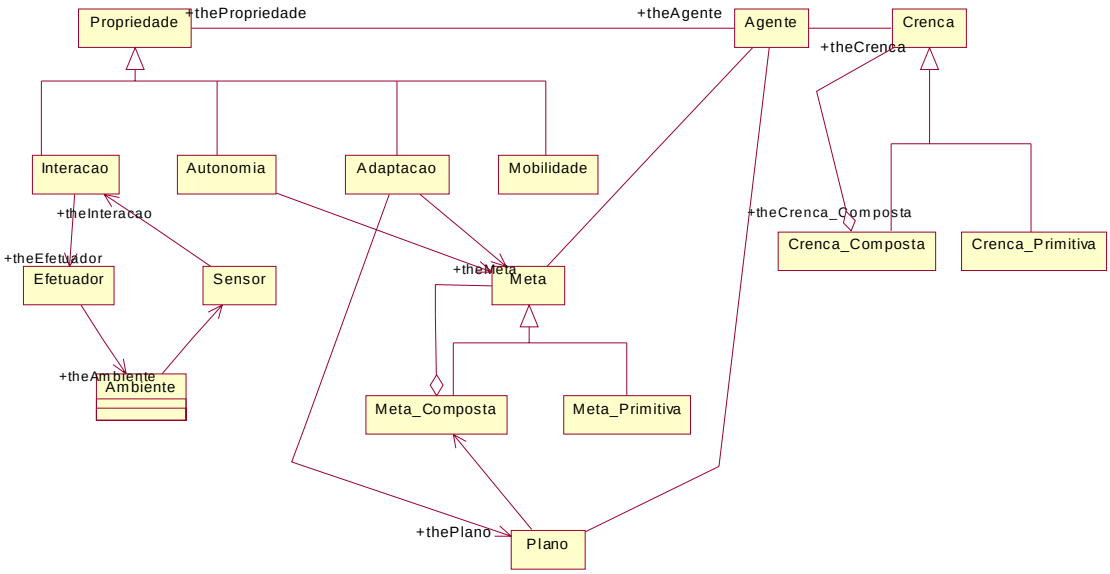

Figura 34 – Diagrama de Classes do Design Pattern para sistemas multi-agentes

Metas, Crenças e Planos

De acordo com o padrão, as metas de um agente são os estados futuros, ou desejos que o agente pretende alcançar, ou satisfazer. As crenças modelam o conhecimento sobre o ambiente externo com o qual o agente interage e modelam o conhecimento sobre os outros agentes do ambiente. Tanto as metas quanto as crenças podem ser sub-divididas em sub-metas e sub-crenças, respectivamente. O plano que é composto por tarefas a serem executadas descreve a estratégia usada para alcançar uma meta. Além disso é com base nas metas e crenças do agente que um plano é selecionado.

Cada meta do agente é uma instância da classe Meta_Primitiva ou Meta_Composta da Figura 34. Esta instanciação ocorre no momento de criação do agente e as demais instanciações de todos os objetos da Figura 34 também ocorrem na criação do agente, exceto a classe Ambiente que é um *Singleton*[6] e deve ser instanciada no início do programa.

Recebendo uma mensagem do ambiente

O padrão estabelece que um agente recebe uma mensagem do ambiente através dos seus sensores, em seguida envia esta mensagem para o objeto responsável pela interação do agente (*Interacao*), que envia a mensagem para o objeto *Agente*. Antes de processar o conteúdo da mensagem, o *Agente* verifica com o objeto *Autonomia* se a mensagem recebida está de acordo com as metas do agente. Se estiver, o agente verifica com o objeto *Adaptação* se é necessário adaptar a meta em questão. Em seguida, o agente seleciona um dos planos de acordo com suas metas. Finalmente, de posse do plano, o agente executa o método *executaplano()* do plano em questão. O diagrama de seqüência da Figura 35 exemplifica a recepção de uma mensagem e o diagrama de seqüência da Figura 36 exemplifica a decisão do agente de tratar ou não a mensagem recebida.

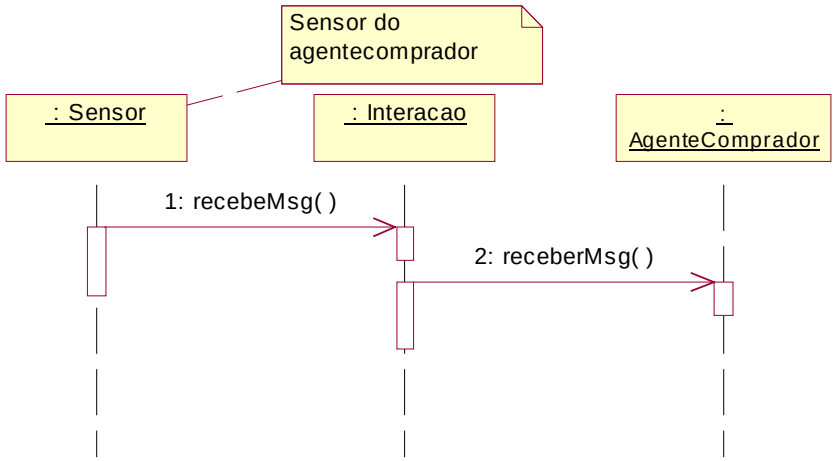

Figura 35 – *AgenteComprador* recebendo uma mensagem do ambiente

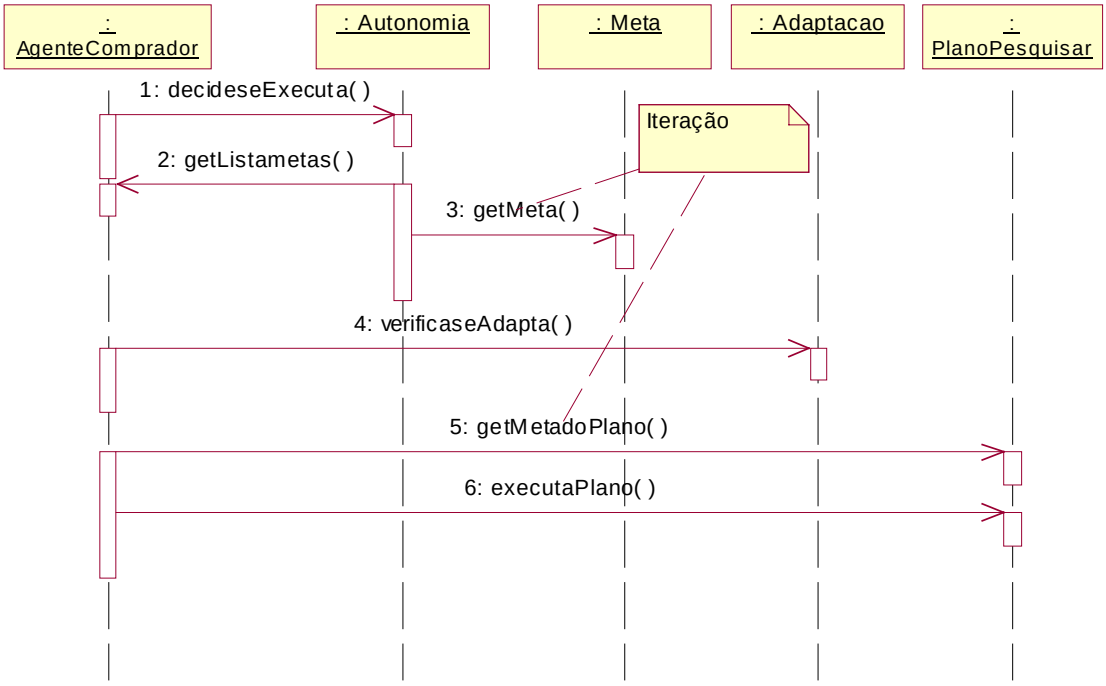

Figura 36 – *AgenteComprador* decide se vai tratar ou não a mensagem recebida

4.2.1 Mapeamento

Nesta seção faremos o mapeamento dos elementos da metodologia Gaia para elementos da modelagem OO feita de acordo com o design pattern para sistemas multi-agentes.

Modelo de agentes derivam classes do diagrama de classes

Como já foi visto anteriormente, o modelo de agentes gerado a partir da metodologia Gaia, especificado na Figura 22, identifica cinco agentes, Agente Usuário, Agente Comprador, Agente Vendedor, Agente Comunicador e Agente Certificador. Os agentes deste modelo são mapeados para o diagrama de classes da Figura 37. Pode-se ver no diagrama de classes que os agentes do sistema são especializações da classe *Agente* do modelo. Para a troca de mensagens entre os agentes optou-se pela utilização de *blackboard* associativos (como Linda [4]). E a implementação do espaço de tuplas adotada foi o *Tspaces*, representado pela classe Tuple Spaces (espaço de tuplas) no diagrama da Figura 37.

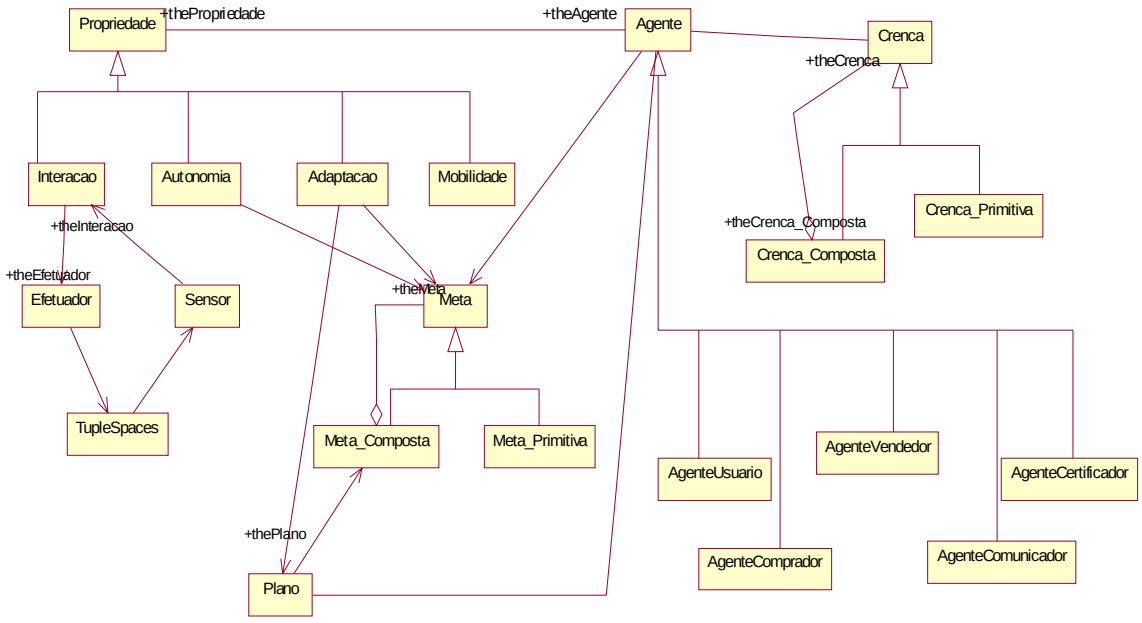

Figura 37 – Diagrama de Classes com os agentes do problema

Serviços do agente derivam métodos da classe

Foi visto anteriormente que muitos métodos das classes que representam os agentes do sistema são extraídos a partir dos serviços do modelo de serviços do Gaia. A dificuldade que se encontra na tentativa de fazer um mapeamento destes serviços para os métodos das classes do modelo, está na definição das classes, do diagrama de classes, que irão conter os métodos, derivados dos serviços. O pattern define o mapeamento de tarefas para métodos, ou seja, os métodos derivados dos serviços são executados pelos planos dos agentes.

Para incluir os métodos no seu respectivo plano, é preciso definir os planos dos agentes e representar suas classes no diagrama de classes. Como todo plano está associado a uma meta, faz-se necessário definir também as metas dos agentes e suas classes. O problema é que a metodologia Gaia não utiliza elementos para representar os estados mentais dos agentes (planos, crenças, metas e tarefas), não se adequando para modelar este tipo de agente cognitivo, utilizado pelo pattern e definido no Capítulo 1.

Extensão da metodologia Gaia

Desta forma, é proposta uma extensão de Gaia para abranger os estados mentais dos agentes, que nos leva a um quinto modelo de Gaia chamado, modelo de estados mentais. O conjunto de figuras começando na Figura 38 e terminando na Figura 42 apresenta os Modelos de Estados Mentais para os agentes do mercado virtual. Este modelo representa os planos, as crenças e as metas do agente. Para a implementação do mercado virtual, o conceito de crença não foi utilizado. As metas são representadas no modelo por uma lista de metas, estruturadas de forma que cada meta primitiva esteja subordinada à sua meta composta correspondente. Toda meta composta possui um plano equivalente que também são representados no modelo por uma lista de planos. As crenças, apesar de não terem sido usadas neste estudo de casos, também devem ser representadas por uma lista de crenças, que assim como as metas, podem ser compostas ou primitivas.

Na tentativa de definir as metas dos agentes do mercado virtual verificou-se semelhança entre estas e a representação estruturada das propriedades de execução dos modelos de papéis da metodologia Gaia. Este fato pode ser observado na comparação das propriedades de execução estruturadas do Assistente de Usuário da Figura 8, com as metas do Agente Usuário da Figura

38. As propriedades de execução do Assistente de Usuário são estruturadas por COMPRAR, VENDER, ALTERAR, EXCLUIR e INFORMAR e as metas do Agente Usuário são, Comprar, Vender, Alterar, Excluir e Listar. Entretanto, utilizar o Modelo de Papéis para especificar as metas não deve ser visto como uma regra, mas apenas como uma sugestão. A questão é que a estruturação das propriedades de execução no modelo de papéis visa apenas simplificar sua representação, sem levar em conta a meta do agente.

Utilizou-se para o modelo de estados mentais um padrão de representação semelhante ao modelo de papéis e às propriedades de segurança. Este novo modelo faz parte da fase de design do Gaia, quando os agentes já foram definidos pelo modelo de agentes. A Figura 43 mostra a incorporação do modelo de estados mentais às fases do Gaia.

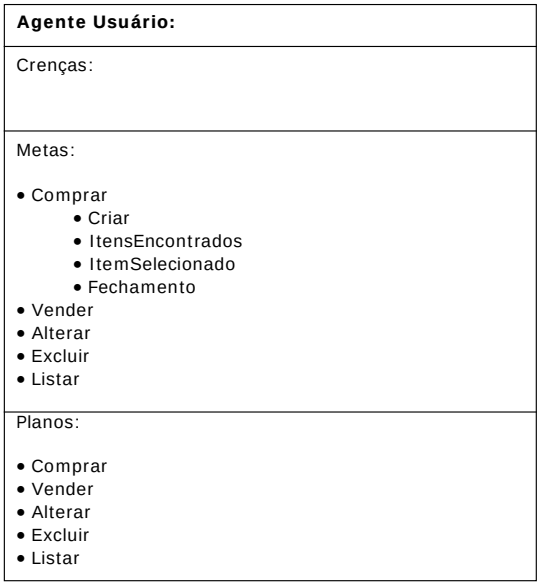

Figura 38 – Modelo de Estados Mentais do Agente Usuário

A Figura 39 apresenta o modelo de estados mentais do agente comprador onde podemos observar que todas as metas compostas têm um plano correspondente. A meta-composta *Pesquisar* significa que o agente comprador tem como objetivo fazer buscas no mercado virtual de acordo com um determinado perfil de compra. Já a meta-composta *Informar* significa que o agente comprador tem a meta de informar ao agente usuário toda vez que forem encontrados itens no mercado virtual.

Ainda na Figura 39 podemos observar a meta-composta *NegociarCompra* que é a meta do agente de efetuar uma negociação quando o agente receber do usuário um item juntamente

com parâmetros de negociação. As metas-primitivas que compõem esta meta-composta são cinco. A primeira delas equivale ao momento que o agente comprador recebe a mensagem do agente usuário contendo o item selecionado para dar início às negociações (*Itemselecionado*). A meta-primitiva *Mínimo* corresponde a recepção de um contra-proposta vinda do agente vendedor contendo o valor mínimo de venda do item. A meta-primitiva *Fechamento* equivale a chegada de uma mensagem do vendedor fechando o negócio e a meta-primitiva *Contraproposta* corresponde ao recebimento de uma contra-proposta que vai ser analisada e respondida. A meta-primitiva *Aborta* é quando uma negociação é abortada por não haver acordo entre comprador e vendedor.

A meta-composta *EfetivarCompra* é a meta do agente comprador de, após a negociação, comprar o item através de seu cartão de crédito ou alguma outra forma de pagamento on-line. E a meta *TerminarCompra* é a meta de finalizar uma compra, após a ocorrência de todos os passos anteriores, avisando o usuário.

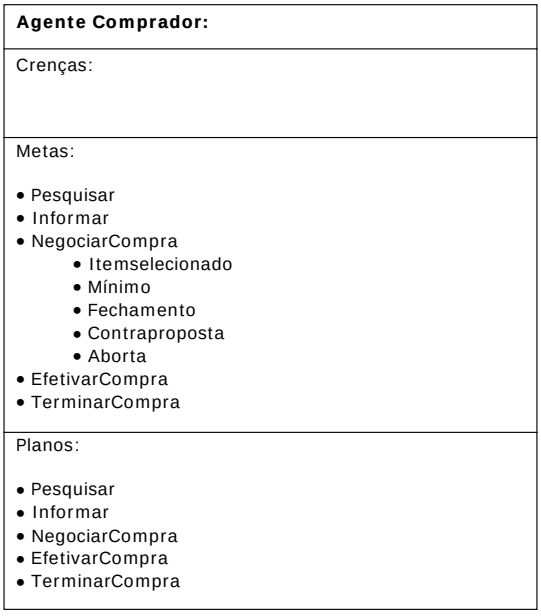

Figura 39 – Modelo de Estados Mentais do Agente Comprador

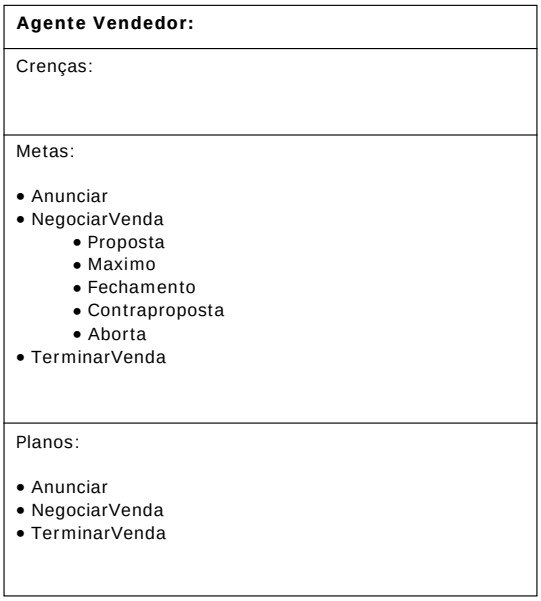

Figura 40 – Modelo de Estados Mentais do Agente Vendedor

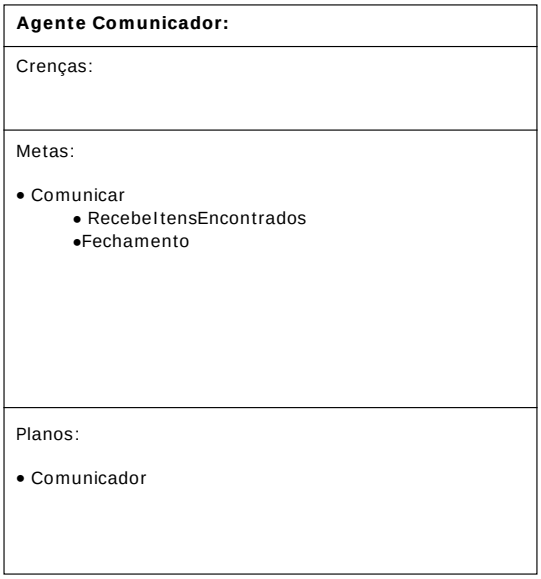

Figura 41 – Modelo de Estados Mentais do Agente Comunicador

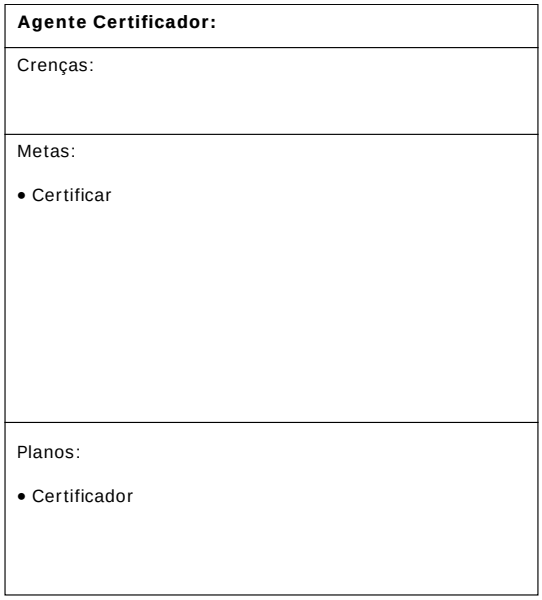

Figura 42 – Modelo de Estados Mentais do Agente Certificador

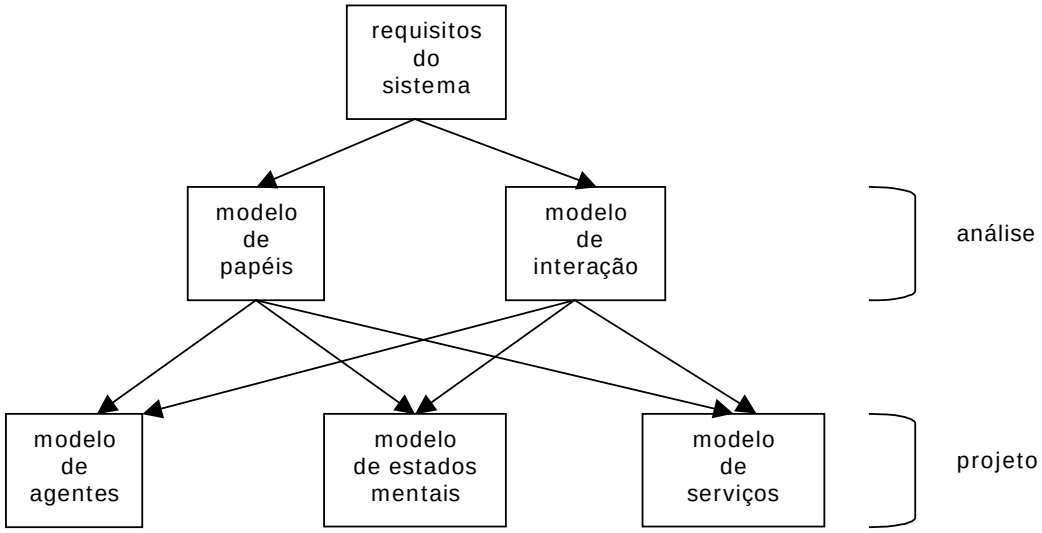

Figura 43 – Os modelos da metodologia Gaia estendida

Criação dos planos

Para cada agente do mercado virtual é criada uma sub-classe da classe plano (Figura 37), onde as tarefas do agente (métodos) são inseridas. Caso um determinado agente apresente mais de um plano no modelo de estados mentais, será necessário estender esta sub-classe criando as sub-classes correspondentes aos planos do agente. O diagrama de classes da Figura 44 ilustra os planos dos agentes do mercado virtual.

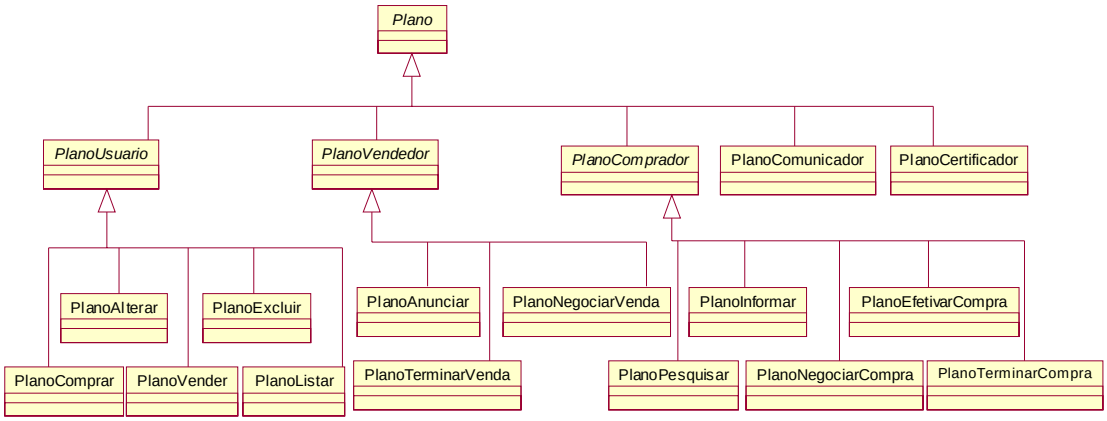

Figura 44 – Planos dos agentes do mercado virtual

Inclusão do plano no modelo de serviços

Em decorrência da existência de planos que contém as tarefas dos agentes, o modelo de serviços sofreu alterações para que todo serviço tenha um plano correspondente. Os modelos de serviços alterados podem ser verificados no conjunto de tabelas que começa na Tabela 8 e termina na Tabela 12. É a partir do modelo de serviços que os métodos, derivados dos serviços, são inseridos nas classes plano correspondentes.

Serviço	**Plano**	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
armazenar dados pessoais		*dadosPessoais*		verdadeiro	verdadeiro
armazenar preferencias		*preferencias*		verdadeiro	verdadeiro
gerar agente compra	Comprar	*prefilCompra*		verdadeiro	verdadeiro
gerar agente venda	Vender	*perfilVenda*		verdadeiro	verdadeiro
confirmar compra	Comprar	*itensSelecionados parametrosNegociacao*	*itensSelecionados parametrosNegociacao*	itensSelecionados <> 0	itensSelecionados <> 0
encerrar compra	Comprar	*resultadoNegociacao*	*itemTransacao*	itemNegociado <> 0	itemTransacao <> 0
solicitar histórico	Listar		*hstoricoperfisCompra historicoperfisVenda*	verdadeiro	verdadeiro
aterar dados pessoais	Alterar	*dadosPessoais*		verdadeiro	e-mail valido
aterar preferências	Alterar	*preferencias*		verdadeiro	Meiocomunicacao valido
listar perfis compra	Listar	*usuario*	*listaPerfiscompra*	usuario existe	verdadeiro
listar perfis venda	Listar	*usuario*	*listaPerfisvenda*	usuario existe	verdadeiro
alterar perfil compra	Alterar	*perfilCompra*	*perfilCompra*	verdadeiro	verdadeiro
alterar perfil venda	Alterar	*perfilVenda*	*perfilVenda*	verdadeiro	verdadeiro
excluir perfil compra	Excluir	*perfilCompra*		verdadeiro	verdadeiro
excluir perfil venda	Excluir	*perfilVenda*		verdadeiro	verdadeiro
solicitar situação pesquisa	Listar	*perfilCompra*	*itensEncontrados*	verdadeiro	verdadeiro
listar itens negociados	Listar		*itensNegociados*	verdadeiro	verdadeiro
solicitar certificação item	Comprar	*dadosCertificacao*	*resultado*	verdadeiro	verdadeiro
solicitar certificacao usuário	Comprar/Vender	*dadosCertificacao*	*resultado*	verdadeiro	verdadeiro
informar itens encontrados	Comprar	*itensEncontrados*	*dadosUsuario itensEncontrados*	verdadeiro	verdadeiro
informar resulado negociação	Comprar	*itensNegociados*	*dadosUsuario itensNegociados*	verdadeiro	verdadeiro
informar dados comprador	Vender	*comprador*	*dadosUsuario comprador*	verdadeiro	verdadeiro

Tabela 8 – Modelo de Serviços do Agente Usuário com os planos do Usuário

Serviços	**Plano**	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
buscar item	Pesquisar	*perfilcompra*	*itensEncontrados*	verdadeiro	verdadeiro
informar lista itens encontrados	Informar	*itensEncontrados*	*itensEncontrados dadosUsuario*	verdadeiro	verdadeiro
enviar mensagem negociacao	NegociarCompra	*proposta*	*proposta*	Negociando = falso	Negociando = verdadeiro
avaliar proposta recebida	NegociarCompra	*proposta*	*propostaAlterada*	verdadeiro	verdadeiro
enviar resultado negociacao	NegociarCompra	*itensNegociados*	*itensNegociados*	verdadeiro	Negociando = falso
encerrar compra	EfetivarCompra	*itemTransacao*	*comprador*	verdadeiro	verdadeiro
informar termino	TerminarCompra			verdadeiro	verdadeiro
informar tempo expirado	TerminarCompra			verdadeiro	verdadeiro
terminar	TerminarCompra			verdadeiro	verdadeiro

Tabela 9 – Modelo de Serviços do Agente Comprador com os planos do Comprador

Serviços	**Plano**	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
publicar item	Anunciar	*perfilVenda*		verdadeiro	verdadeiro
aguardar propostas	Anunciar			verdadeiro	verdadeiro
enviar mensagem negociacao	NegociarVenda	proposta	proposta	Negociando = falso	Negociando = verdadeiro
avaliar proposta recebida	NegociarVenda			verdadeiro	verdadeiro
informar termino	NegociarVenda	*comprador*	*comprador*	verdadeiro	verdadeiro
informar fim prazo venda	TerminarVenda			verdadeiro	verdadeiro
terminar	TerminarVenda			verdadeiro	verdadeiro

Tabela 10 – Modelo de Serviços do Agente Vendedor com os planos do Vendedor

Serviços	**Plano**	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
transmitir mensagem	Comunicador	dadosUsuario	mensagem	CanalDisponível = verdadeiro UsuárioExiste = verdadeiro Mensagem <> vazio	verdadeiro
confirmar envio	Comunicador			verdadeiro	verdadeiro

Tabela 11 – Modelo de Serviços do Agente Comunicador com os planos do Comunicador

Serviços	**Plano**	**Entradas**	**Saídas**	**Pré-condições**	**Pós-condições**
certificar item	Certificador	dadosCertificacao		ItemExiste = verdadeiro	verdadeiro
certificar usuario	Certificador	dadosCertificacao		UsuarioExiste = verdadeiro	verdadeiro
enviar certificacao usuário	Certificador			verdadeiro	verdadeiro
enviar certificacao item	Certificador			verdadeiro	verdadeiro

Tabela 12 – Modelo de Serviços do Agente Certificador com os planos do Certificador

Os diagramas de classes, do conjunto de figuras começando na Figura 45 e terminando na Figura 49, apontam os métodos de cada classe estendida de Plano. Pode-se verificar que os métodos dos planos foram inseridos no plano apontado pelo modelo de serviços. Compare os Modelos de Serviços do conjunto de tabelas que começa na Tabela 8 e termina na Tabela 12 com os diagramas de classes começando na Figura 45 e terminando na Figura 49 e verifique o mapeamento dos serviços para os métodos da classe plano correspondente.

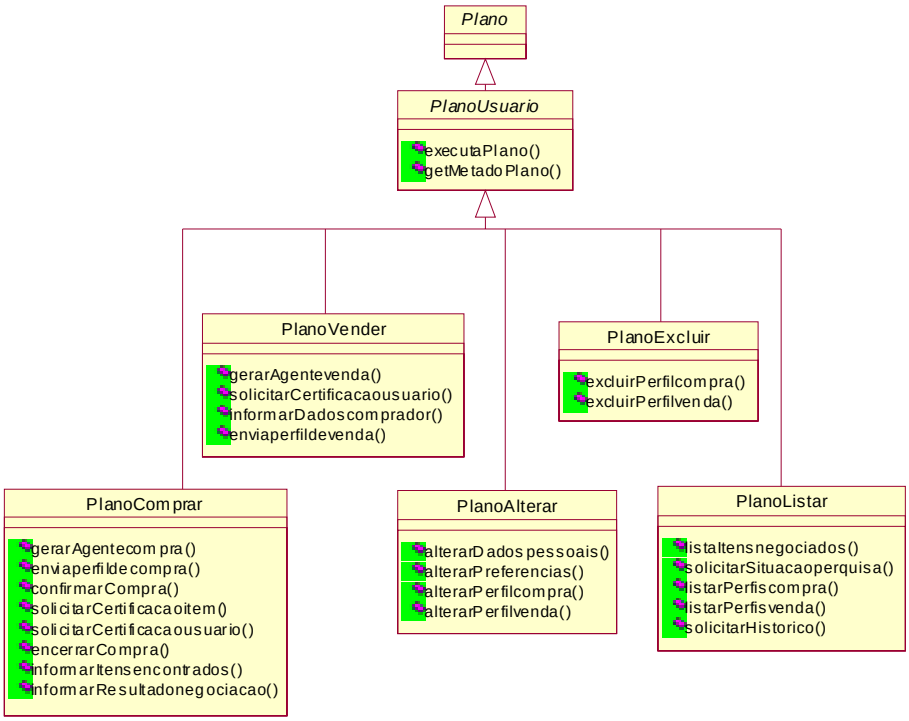

Figura 45 – Diagrama de Classes com os métodos das classes que representam o plano do Agente Usuário

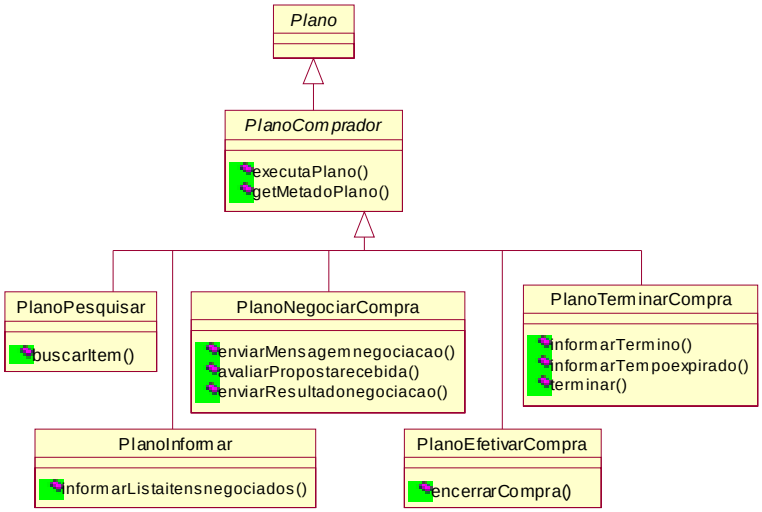

Figura 46 – Diagrama de Classes com os métodos das classes que representam o plano do Agente Comprador

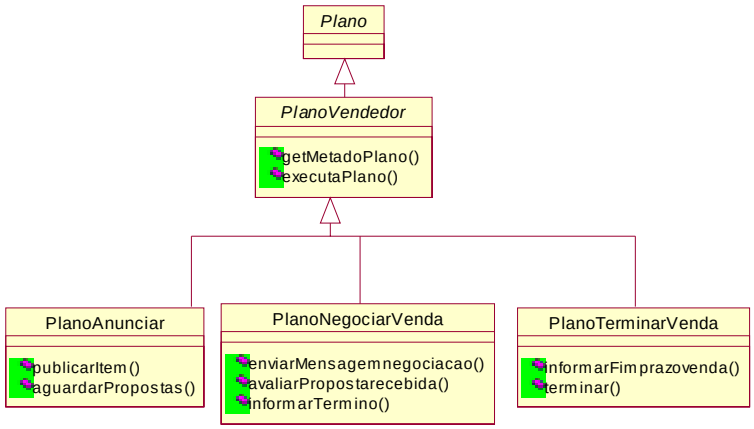

Figura 47 – Diagrama de Classes com os métodos das classes que representam o plano do Agente Vendedor

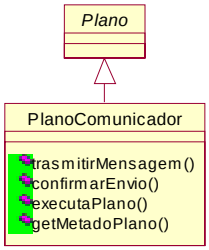

Figura 48 – Diagrama de Classes com os métodos da classe que representa o plano do Agente Comunicador

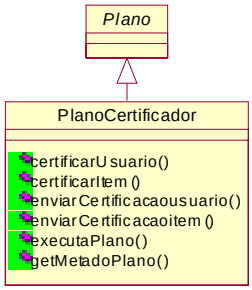

Figura 49 – Diagrama de Classes com os métodos da classe que representa o plano do agente Certificador

Modelo de Interação *versus* Diagrama de Seqüência

No início desta seção, foi feita uma breve introdução de como se dá a seqüência de troca de mensagens entre os objetos que compõem o agente. Percebe-se que não há nenhum modelo em Gaia que modele estas interações. Diferente do *MAS Framework*, não há correspondência entre os Modelos de Interação de Gaia e os Diagramas de Seqüência da implementação do problema. O Modelo de Interação do Gaia especifica a interação entre os agentes do problema, mas não define a interação entre os objetos que compõe o agente.

Veja na Figura 13, o Modelo de Interação do protocolo do Usuário *OrdenarCompra* e compare com os seis Diagramas de Seqüência, do conjunto de figuras começando na Figura 50 e terminando na Figura 55, que modelam esta mesma funcionalidade. Fica claro que não há correlação alguma. Agora compare com o Diagrama de Seqüência da Figura 28, que modela esta mesma funcionalidade no *MAS Framework*.

O diagrama de seqüência da Figura 50 representa o momento em que o usuário final selecionou a opção para criar um agente comprador de acordo com um perfil de compra especificado. O *AgenteUsuario* verifica com o objeto *Autonomia* se esta mensagem proveniente da interface está de acordo com as metas dele. Se estiver o agente verifica com o objeto *Adaptação* se é necessário adaptar a meta em questão. Em seguida o agente seleciona um dos planos de acordo com suas metas e solicita a execução do plano (*executaplano()*).

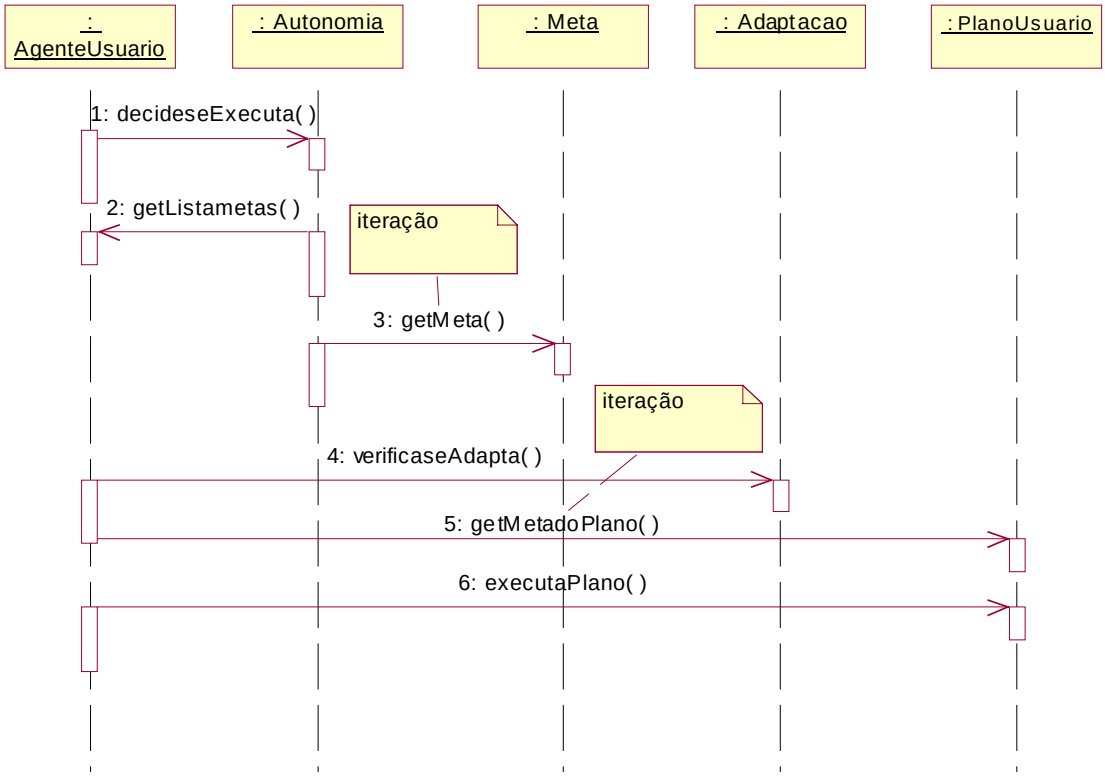

Figura 50 – Diagrama de Seqüência equivalente à parte da funcionalidade do protocolo do Usuário *OrdenarCompra*

Neste exemplo o método *executaplano()* do *PlanoComprar* do *AgenteUsuário* será executado como mostra a Figura 51. Pode-se observar também que a primeira tarefa do plano é criar um agente comprador bem como todos os objetos que compõe o agente comprador. Após a criação do agente comprador a próxima tarefa do plano comprar do agente usuário é o envio do perfil de compra, definido pelo usuário final, para o agente comprador. O envio do perfil de compra deve ser feito através do *Tspaces* como mostra a Figura 52.

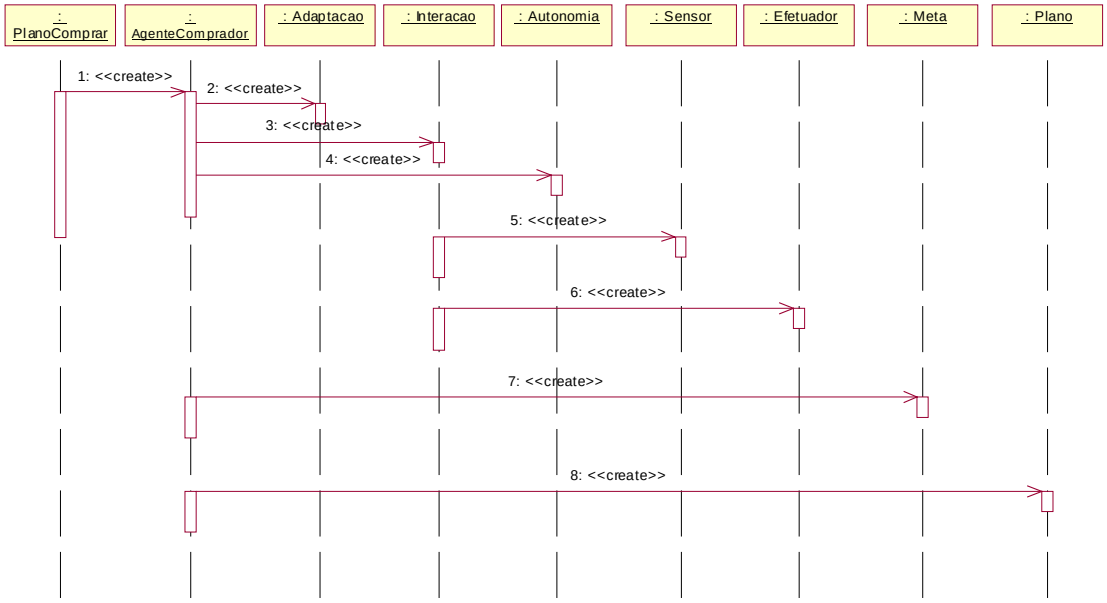

Figura 51 – Diagrama de Seqüência equivalente à parte da funcionalidade do protocolo do Usuário *OrdenarCompra*

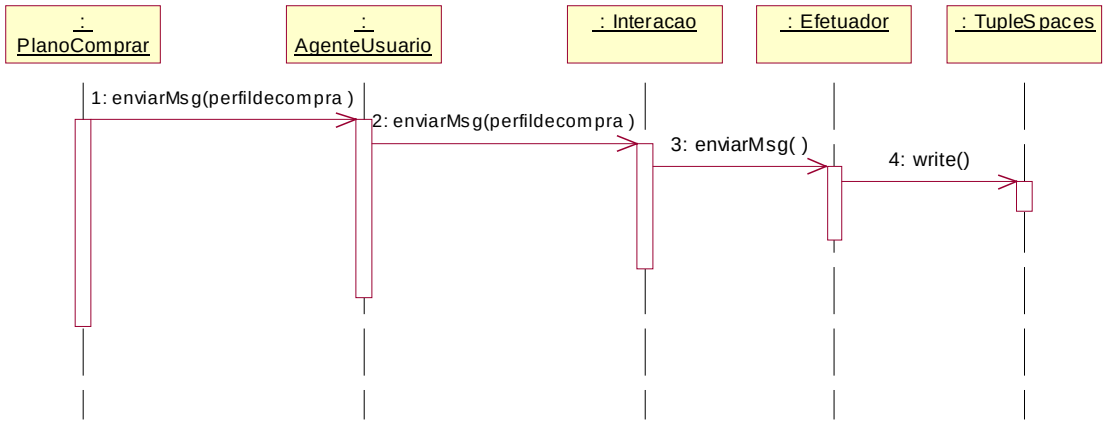

Figura 52 – Diagrama de Seqüência equivalente à parte da funcionalidade do protocolo do Usuário *OrdenarCompra*

O sensor do agente comprador recebe a mensagem contendo o perfil de compra do ambiente (Figura 53) e em seguida executa os passos referentes a recepção padrão de uma mensagem do ambiente (Figura 54). Neste caso a mensagem que continha o perfil de compra determinou a execução do método *executaplano()* do *PlanoPesquisar*. A tarefa do plano que será executada corresponde ao método *buscarItem()* que determina a criação de um sensor específico para anúncios no ambiente (Figura 55).

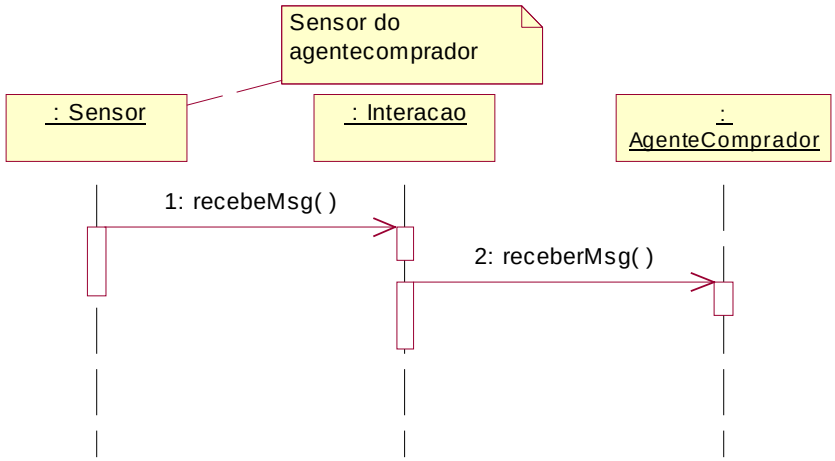

Figura 53 – Diagrama de Seqüência equivalente à parte da funcionalidade do protocolo do Usuário *OrdenarCompra*

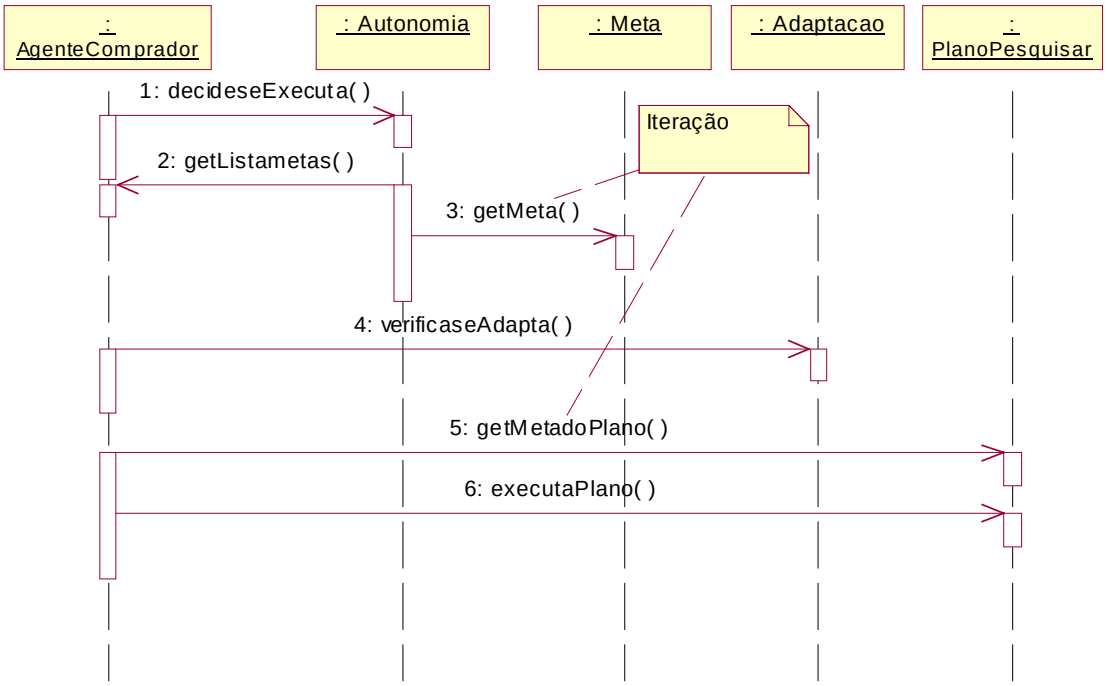

Figura 54 – Diagrama de Seqüência equivalente à parte da funcionalidade do protocolo do Usuário *OrdenarCompra*

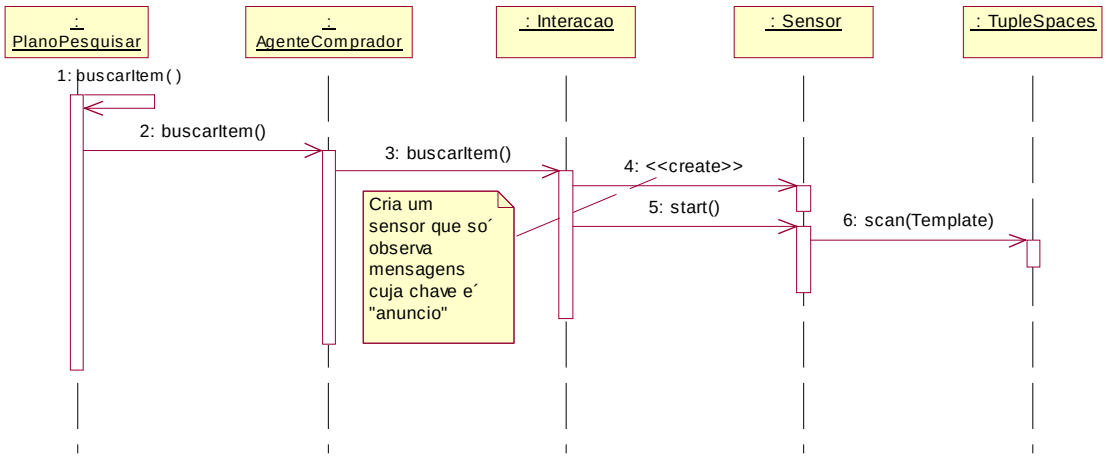

Figura 55 – Diagrama de Seqüência equivalente à parte da funcionalidade do protocolo do Usuário *OrdenarCompra*

4.3 Análise do Mapeamento

A abordagem para implementação de sistemas multi-agentes apresentada na seção 4.2 é mais complexa se comparada com a abordagem apresentada na seção 4.1. Tal complexidade ocorre em função do tipo de agentes que o padrão modela, que são os agentes cognitivos. Tais agentes possuem capacidade de cognição que são expressos através dos estados mentais do agente. Já a abordagem vista na seção 4.1, visa modelar os agentes reativos que não possuem cognição e por isso são mais simples.

O mapeamento de Gaia para a abordagem vista na seção 4.1 é feito de maneira direta e sem a necessidade de se estender a metodologia. Em diversos casos temos uma correspondência clara entre os elementos de Gaia e os elementos do *framework*, de forma que a metodologia Gaia contribui bastante para uma implementação de sistemas multi-agentes, com aspectos semelhantes aos desta abordagem.

A metodologia Gaia separa as funcionalidades do agente, que exigem interação com outros agentes (protocolos), das funcionalidades dos agentes, que não exigem interação com outros agentes (atividades). Esta separação é bastante adequada para fazer o mapeamento da metodologia Gaia para uma modelagem orientada a objetos (diagrama de classes) onde os agentes são representados por duas classes: uma que trata do agente e a outra para tratar as

mensagens recebidas e enviadas entre os agentes (interação do agente). Tal fato reforça a idéia de que Gaia é bastante adequado para modelar um sistema multi-agentes com agentes reativos, especialmente àqueles que são modelados com duas classes como o apresentado na seção 4.1.

O mapeamento de Gaia para a abordagem do design pattern para sistemas multi-agentes foi mais complexa e determinou a extensão da metodologia Gaia para abranger os estados mentais dos agentes. Os modelos de estados mentais são então mapeados para as metas, os planos e as crenças dos agentes cognitivos.

O modelo de interação de Gaia teve pouca utilidade nesta abordagem pois ele não modela as interações entre os objetos que compõem o agente cognitivo, ele só modela a interação inter-agente do problema. A parte intra-agente tratada pela metodologia Gaia, diz respeito aos atributos do modelo de papeis (responsabilidades, permissões, atividades e protocolos), o que para os agentes cognitivos não é suficiente.

É necessário na metodologia Gaia definir modelos que abranjam as interações intra-agente dos agentes cognitivos, onde as metas, os planos, as crenças, as propriedades e o próprio agente constituem o conjunto de objetos que representam o agente e que interagem entre si.

Capítulo 5

Conclusões

O objetivo deste trabalho foi apresentar o mapeamento da metodologia Gaia de análise e projeto orientado a agentes para duas modelagens OO para a geração de um sistema multi-agentes de um mercado virtual.

Para começar foi feita uma introdução dos conceitos envolvidos na tecnologia de agentes, e da metodologia Gaia. O passo seguinte foi apresentar a aplicação de Gaia ao estudo de casos de um mercado virtual na Internet.

Como a metodologia Gaia não define aspectos relativos à fase de implementação de um sistema multi-agentes, foi proposto o mapeamento de Gaia para duas modelagens OO. Uma modelagem trata dos agentes reativos enquanto a outra abrange os agentes cognitivos.

A metodologia Gaia mostrou-se bastante útil para a modelagem de sistemas multi-agentes com características de agentes reativos mas foi ineficiente para tratar da modelagem dos agentes cognitivos. Tal ineficiência determinou a extensão de Gaia com a inclusão de mais um modelo e a alteração de um dos modelos existentes.

5.1 Principais contribuições

Neste trabalho foram apresentadas diversas contribuições para a área de pesquisa de agentes de software e de engenharia de software baseada em agentes.

- Aplicação da metodologia Gaia num problema bastante conhecido de mercado virtual, contribuindo para a análise e validação da metodologia.

- Definição do mapeamento de Gaia para a modelagem definida pelo *MAS Framework* para a implementação de agentes.

- Proposta de extensão do *MAS Framework* para incluir mais uma classe de tratamento das mensagens recebidas, tornando transparente para os usuários do *MAS Framework* o problema de concorrência que ocorre quando o agente recebe diversas mensagens do *Tspaces* no mesmo instante.

- Definição do mapeamento de Gaia para a modelagem definida pelo design pattern para sistemas multi-agentes.

- Proposta de extensões da metodologia Gaia para adequação da metodologia aos sistemas com agentes cognitivos.

- Validação da metodologia Gaia para a implementação de agentes reativos.

- Definição de um método para a geração de um sistema multi-agentes com agentes reativos. Para isso usa-se Gaia para as fases de análise e projeto, em seguida usa-se o mapeamento proposto na seção 4.1 e por fim pode-se implementar o sistema, a partir dos diagramas de classe e de seqüência gerados, numa linguagem OO.

5.2 Trabalhos Futuros

Para trabalhos futuros da presente dissertação incluem-se:

- Proposta de uma extensão de Gaia para incluir um modelo que considere a troca de mensagens entre os objetos que compõem os agentes cognitivos do design pattern para sistemas multi-agentes apresentado em 4.2.

- Implementação de um *framework* para geração de sistemas multi-agentes onde os agentes têm características cognitivas, de acordo com o design pattern para sistemas

multi-agentes visto na seção 4.2. Neste framework a complexidade da interação intra-agente seria abordada pelas partes fixas do *framework* (*frozen spot*).

- Mapeamento de Gaia para outra modelagem OO específica para Gaia, possibilitando a definição de uma metodologia para geração de um sistema multi-agente desde sua concepção até a sua efetiva implementação. Esta metodologia teria duas classes uma para as funções internas do agente e outra para tratar a interação do agente. A primeira classe teria os métodos derivados das atividades do papel já a segunda classe teria os métodos derivados dos protocolos do papel.

- Outros estudos de caso devem ser utilizados a fim de validar a metodologia para um domínio de aplicações maior.

Referências Bibliográficas

1. C. J. Petrie. "*Agent-Based Software Engineering*", in J. Bradshaw and G. Arnold, editors, Proceedings of the 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000), Manchester, UK, 2000.

2. C. Lucena, R. Milidiú, (ed), *"Sistemas Multi-Agentes"*, Editora Papel Virtual, 2001.

3. D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, and P. Jeremaes. *"Object-Oriented Development: The FUSION Method"*, Prentice Hall International: Hemel Hempstead, England, 1994.

4. D. Galernter, "*Generative Communication in Linda*" ACM Transactions on Programming Languages and Systems, vol. 7, No.1, pp 80-112, 1985.

5. D. Lea, "*Concurrent Programming in Java: Design Principles and Patterns*", Second Edition, Addison Wesley, 1999.

6. E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, "*Design Patterns, Elements of Reusable Object-Oriented Software*", Addison-Wesley, 1995

7. G. Weiss, (ed) "*Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*", Cambridge, MA, The MIT Press, 1999

8. http://www.teccomm.les.inf.puc-rio.br

9. J. Ferber, "*Multi-agent Systems. An Introduction to Distributed Artificial Intelligence*", Addison-Wesley, 1999.

10. J. Rumbaugh, I. Jacobson, G. Booch, "*The unified modeling language reference manual*", Adison Wesley, 1999.

11. M. Fayad and D. Schmidt, "*Object-oriented application frameworks*", Communication of the ACM, 40(10), 32-38, 1997.

12. M. Wooldridge, N. R. Jennings, D. Kinny, "*A methodology for agent-oriented analysis and design*", in Proceedings of the Third International Conference on Autonomous Agents (Agents 99), pages 69-76, Seattle, WA, May 1999.

13. M. Wooldridge, N. R. Jennings, D. Kinny, "*The Gaia methodology for agent-oriented analysis and designs*", Journal of Autonomous Agent and Multi-Agent Systems, 2000.

14. MAS Framework: http://www.teccomm.les.inf.puc-rio.br/agentes/

15. N. R. Jennings, M. Wooldridge, "*Agent-oriented software engineering*", in: J. Bradshaw (Ed.), Handbook of Agent Technology, AAAI/MIT Press, 2000, to appear.

16. N. R. Jennings. "*On agent-based software engineering*", Artificial Intelligent 117(2000):277-196, 2000.

17. N.R. Jennings, K. Sycara and M. Wooldridge, "*A Roadmap of Agent Research and Development*", in: Autonomous Agents and Multi-Agent Systems Journal, N.R. Jennings, K. Sycara and M. Georgeff (Eds.), Kluwer Academic Publishers, Boston, Volume 1, Issue 1, pages 7-38, 1998

18. Object Management Group – Agent Platform Special Interest Group. "*Agent Technology – Green Paper*". Version 1.0, September 2000.

19. OMG. The object management group. http://www.omg.org/

20. T. Lehman, S. McLaughry, and P. Wyckoff. "*TSpaces: The Next Wave*". Hawaii International Conference on System Sciences (HICSS-32), January 1999.

21. V. Silva, C. Lucena, "*Modelo Orientado a Objetos para Sistemas Multi-Agentes*", MCC30/01, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, Outubro 2001.

Apêndice A

Modelos de Interação

Nesta seção é apresentado o conjunto completo dos modelos de interação do estudo de casos do mercado virtual Babilônia.

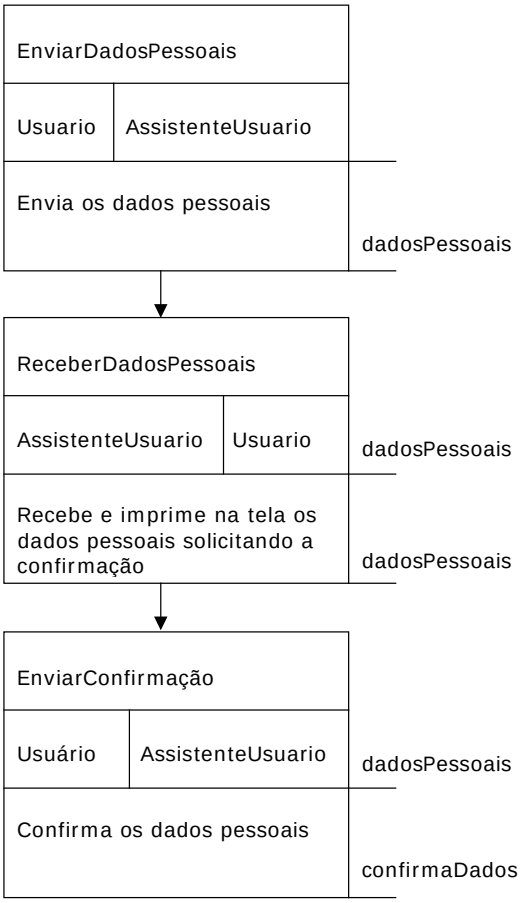

Figura 56 – Modelo de Interação do Protocolo do Usuário *InformarDadosPessoais*

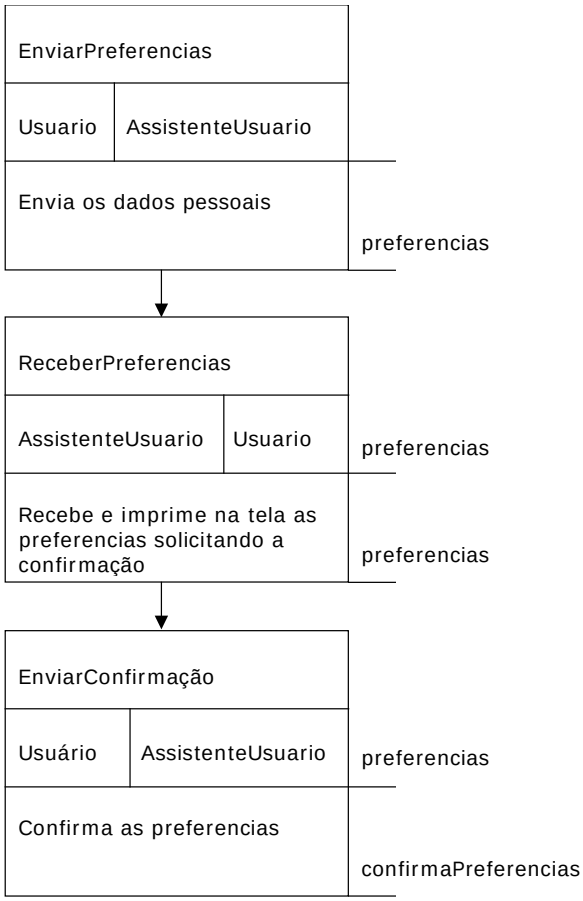

Figura 57 – Modelo de Interação do Protocolo do Usuário *InformarPreferências*

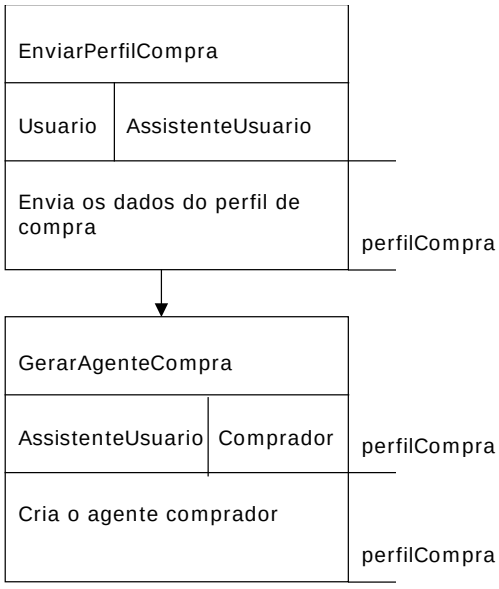

Figura 58 – Modelo de Interação do Protocolo do Usuário *OrdenarCompra*

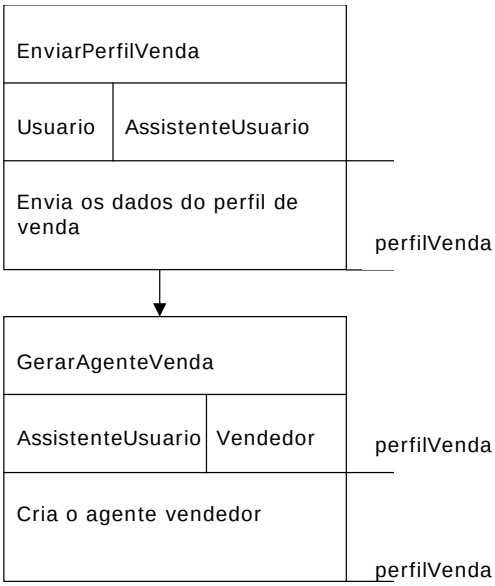

Figura 59 – Modelo de Interação do Protocolo do Usuário *OrdenarVenda*

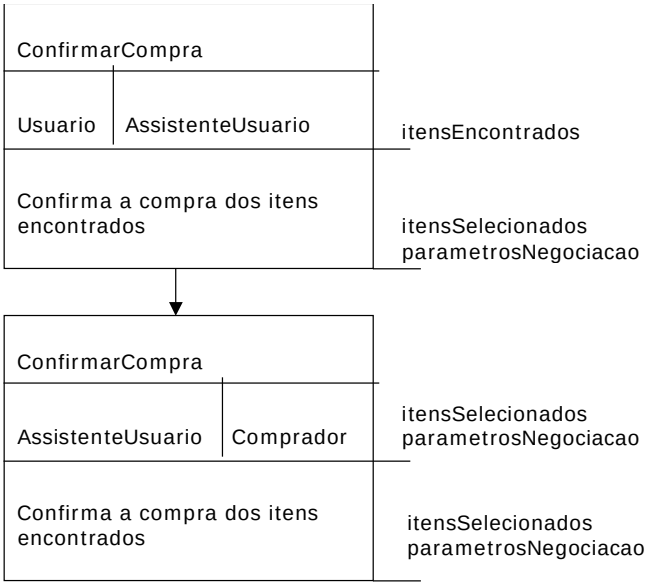

Figura 60 – Modelo de Interação do Protocolo do Usuário *ConfirmarCompra*

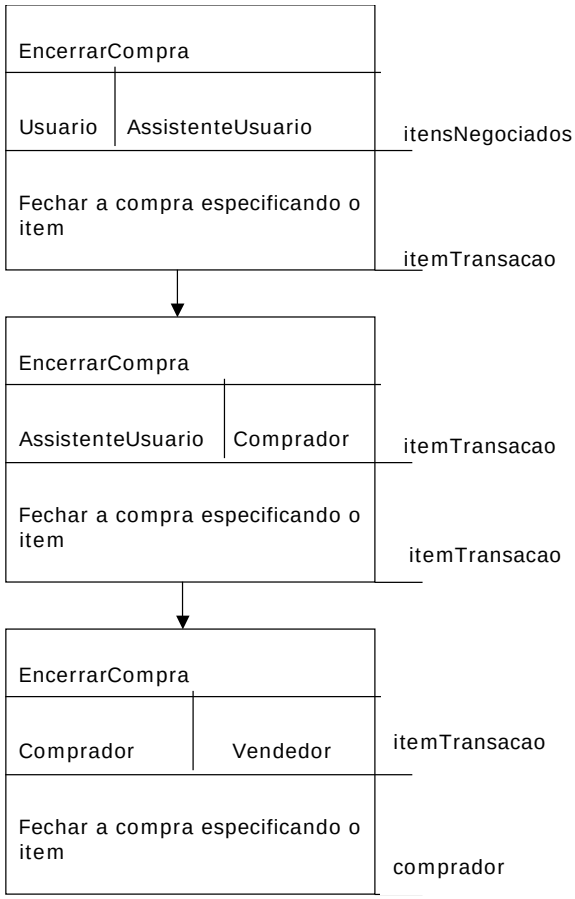

Figura 61 – Modelo de Interação do Protocolo do Usuário *EncerrarCompra*

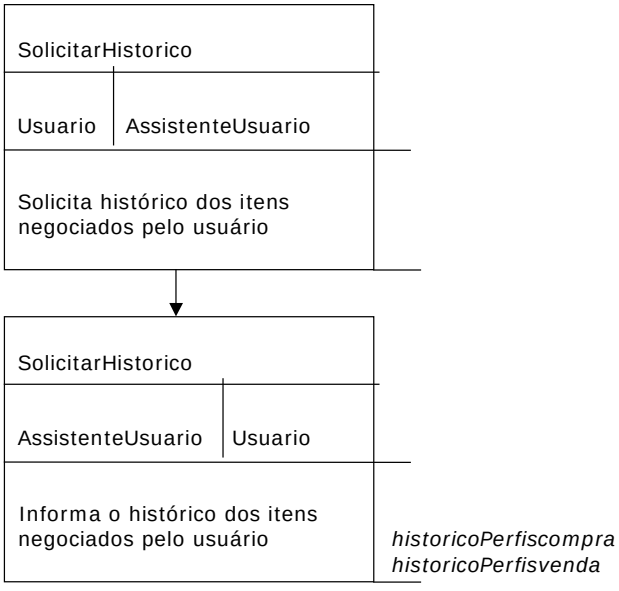

Figura 62 – Modelo de Interação do Protocolo do Usuário *SolicitarHistórico*

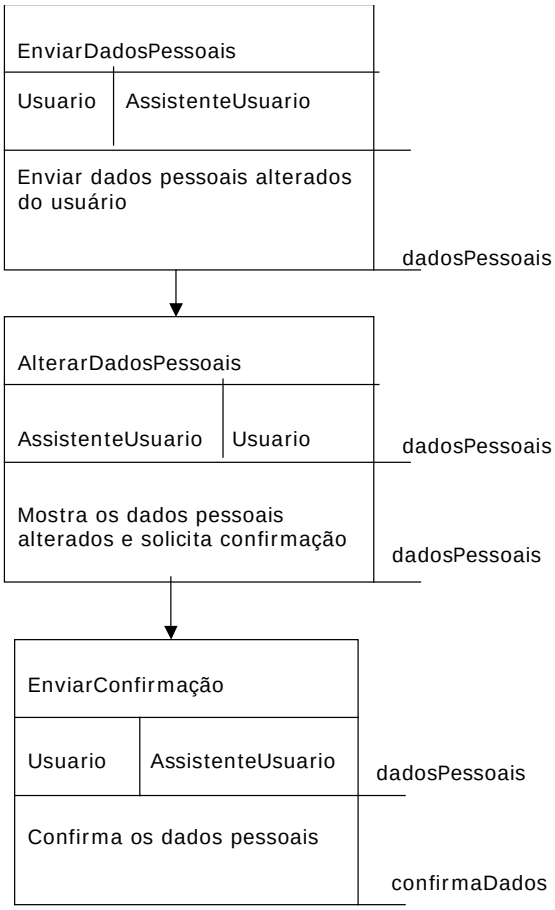

Figura 63 – Modelo de Interação do Protocolo do Usuário *AlterarDadosPessoais*

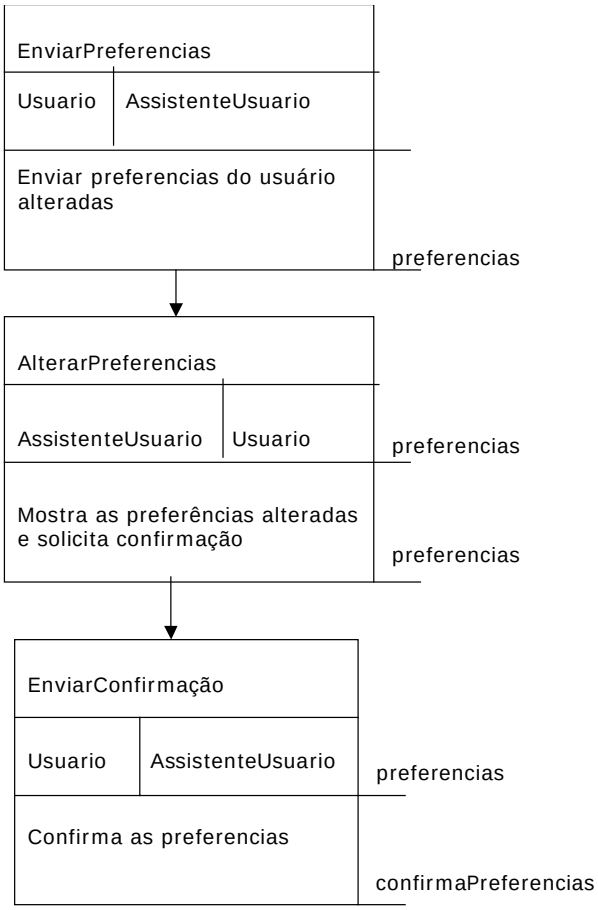

Figura 64 – Modelo de Interação do Protocolo do Usuário *AlterarPreferências*

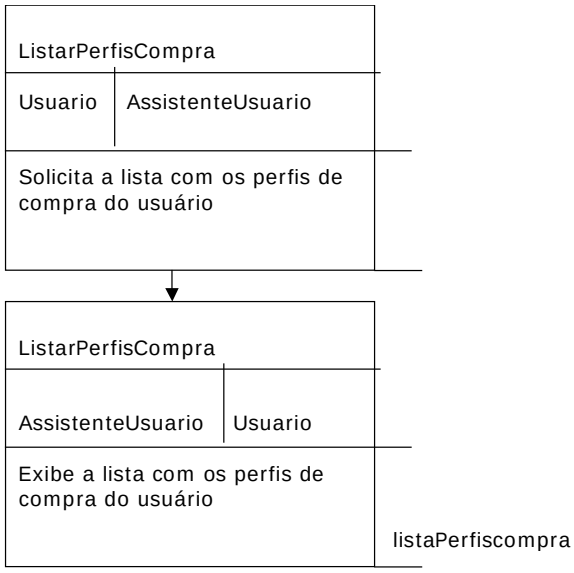

Figura 65 – Modelo de Interação do Protocolo do Usuário *ListarPerfisCompra*

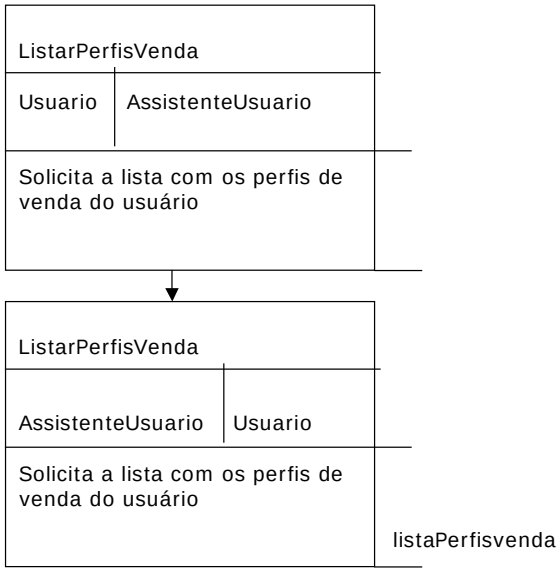

Figura 66 – Modelo de Interação do Protocolo do Usuário *ListarPerfisVenda*

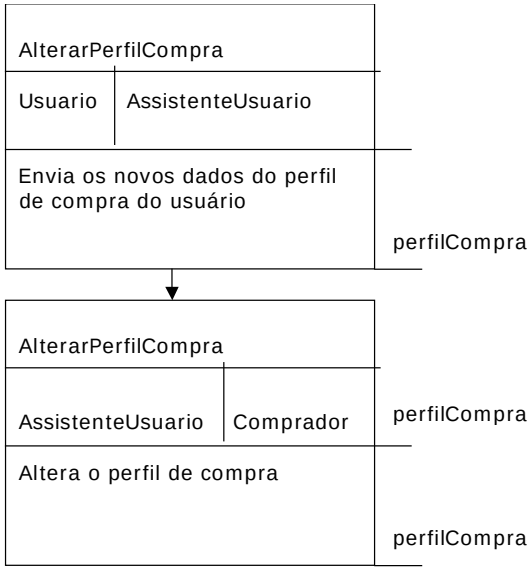

Figura 67 – Modelo de Interação do Protocolo do Usuário *AlterarPerfilCompra*

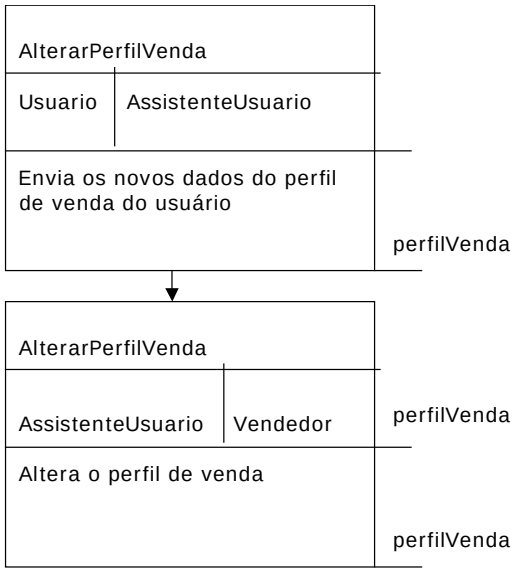

Figura 68 – Modelo de Interação do Protocolo do Usuário *AlterarPerfilVenda*

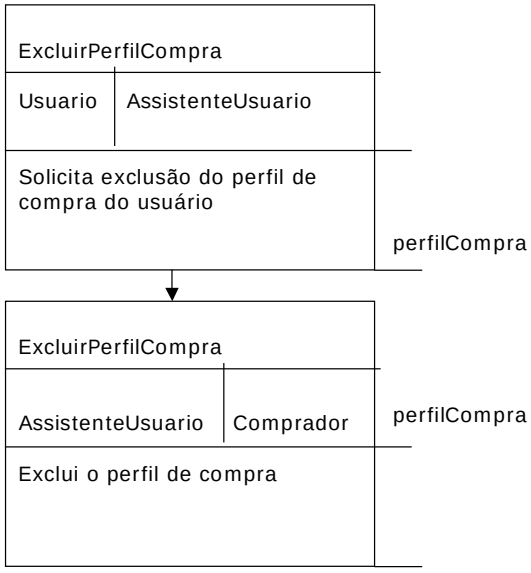

Figura 69 – Modelo de Interação do Protocolo do Usuário *ExcluirPerfilCompra*

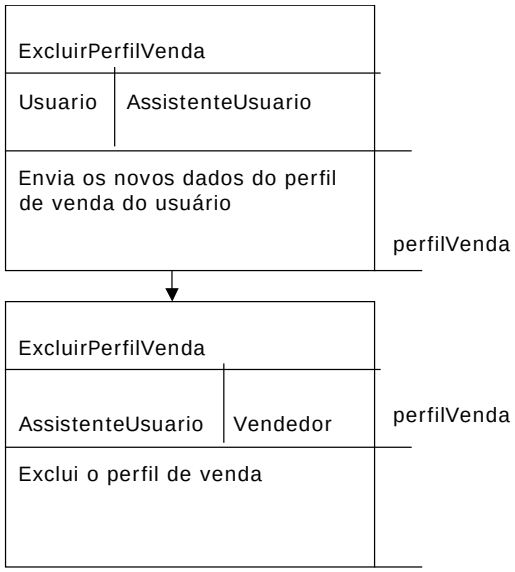

Figura 70 – Modelo de Interação do Protocolo do Usuário *ExcluirPerfilVenda*

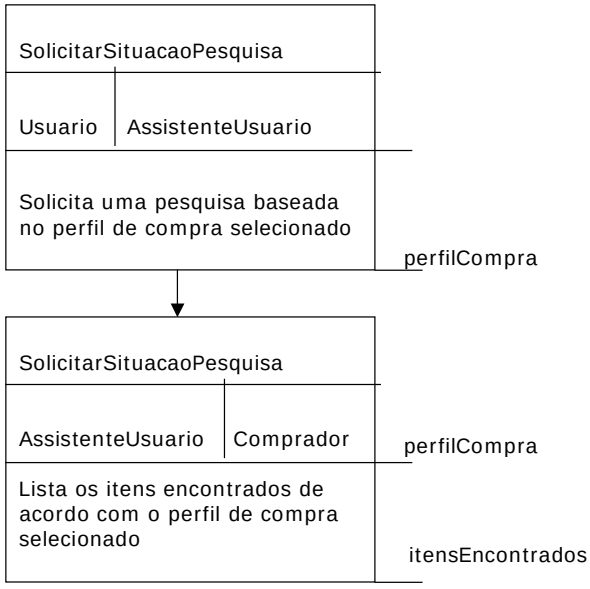

Figura 71 – Modelo de Interação do Protocolo do Usuário *SolicitarSituacaoPesquisa*

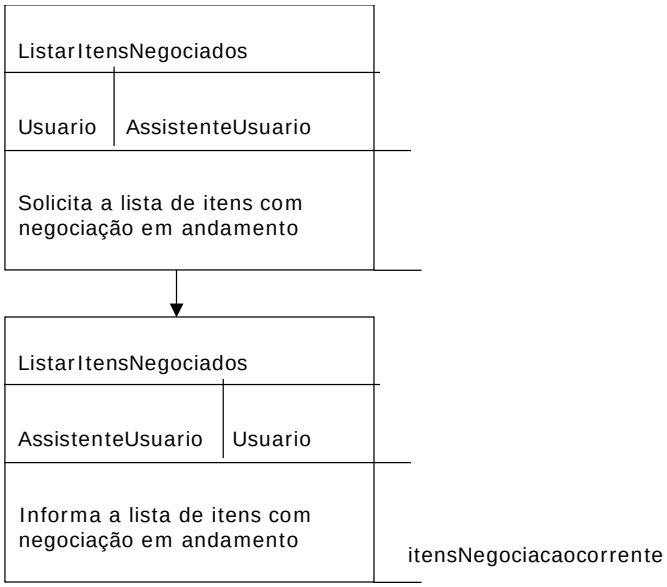

Figura 72 – Modelo de Interação do Protocolo do Usuário *ListarItensNegociados*

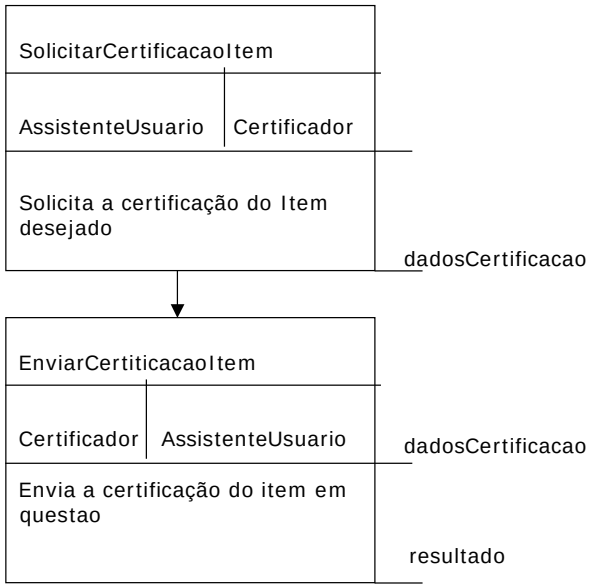

Figura 73 – Modelo de Interação do Protocolo do Assistente de Usuário *SolicitarCertificaçãoItem*

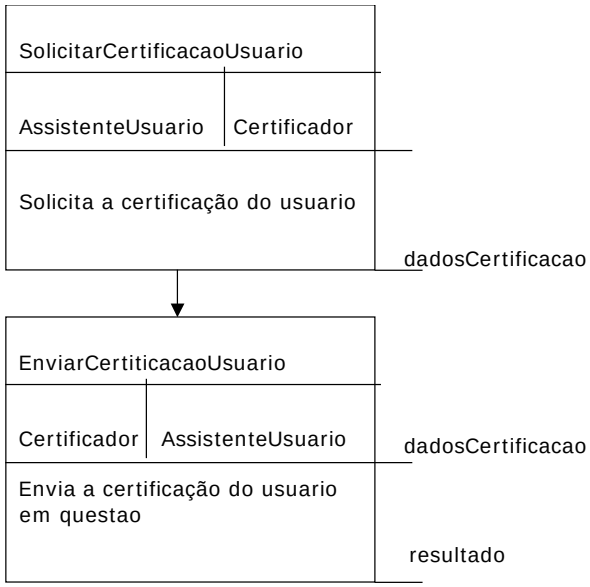

Figura 74 – Modelo de Interação do Protocolo do Assistente de Usuário *SolicitarCertificaçãoUsuário*

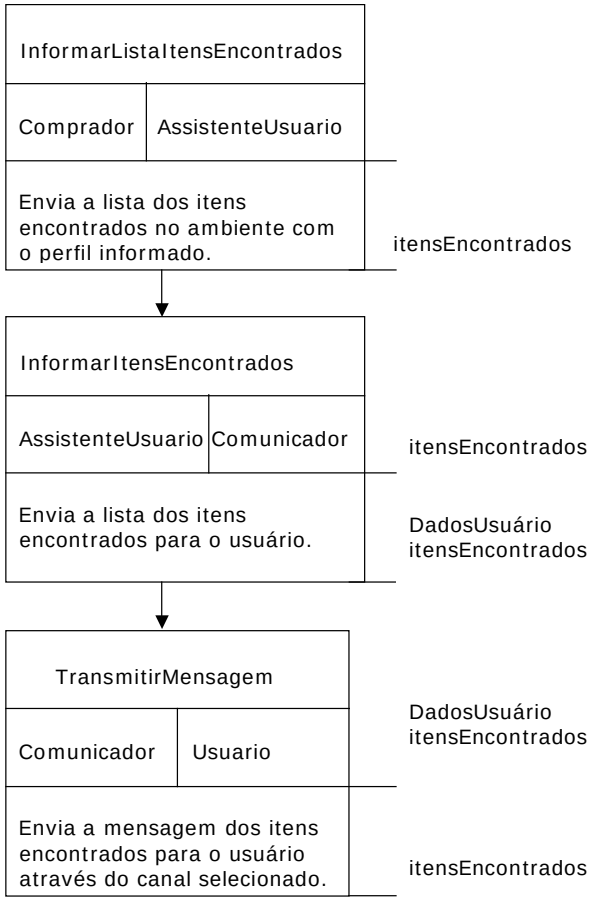

Figura 75 – Modelo de Interação do Protocolo do Comprador *InformarListaItensEncontrados*

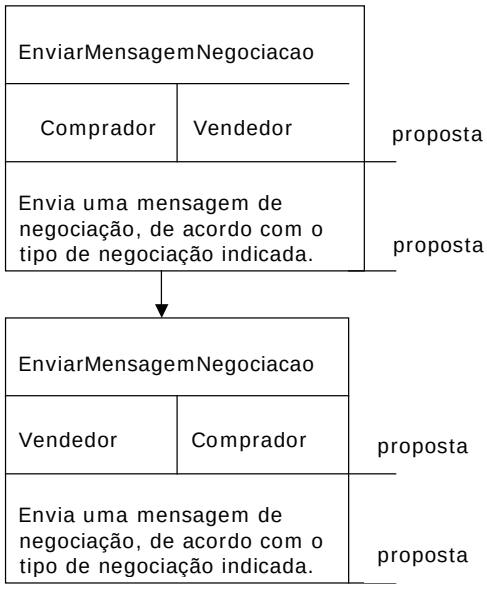

Figura 76 – Modelo de Interação do Protocolo do Comprador *EnviarMensagemNegociação*

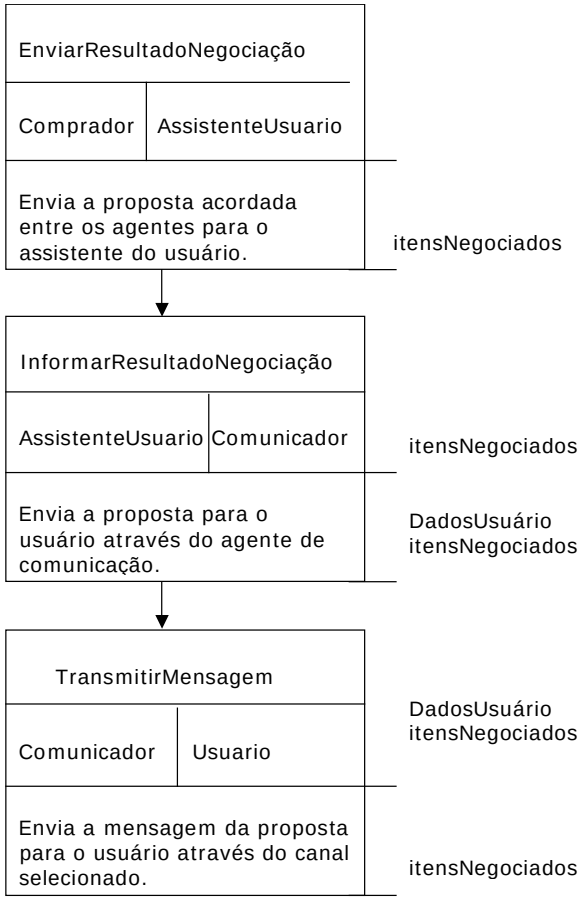

Figura 77 – Modelo de Interação do Protocolo do Comprador *EnviarResultadoNegociação*

InformarTermino	
Comprador	AssistenteUsuario
Envia um sinal, informando para o ASU que o comprador vai terminar.	

Figura 78 – Modelo de Interação do Protocolo do Comprador *InformarTermino*

InformarTempoExpirado	
Comprador	AssistenteUsuario
Envia um sinal, informando para o ASU que o comprador vai terminar por tempo expirado.	

Figura 79 – Modelo de Interação do Protocolo do Comprador *InformarTempoExpirado*

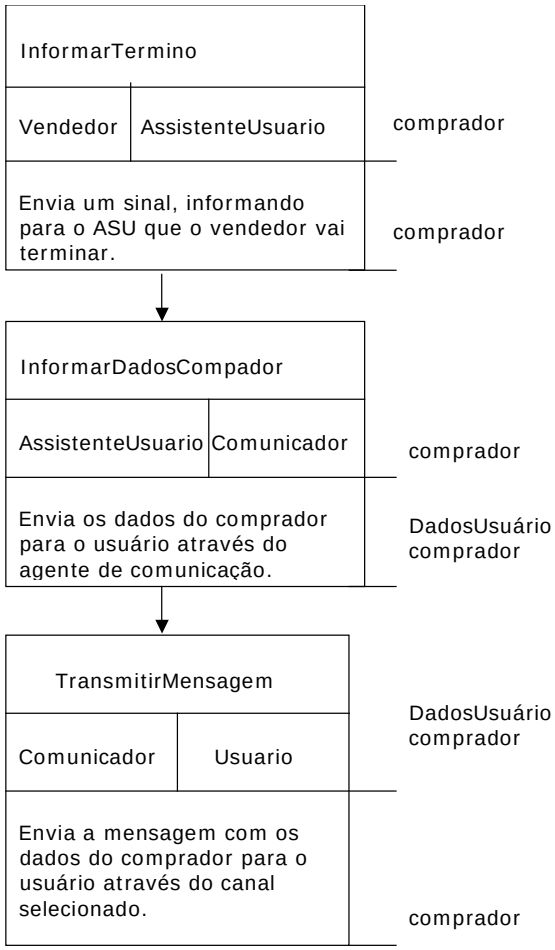

Figura 80 – Modelo de Interação do Protocolo do Vendedor *InformarTermino*

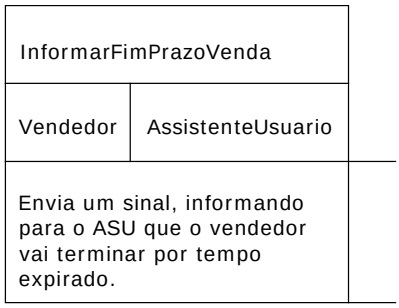

Figura 81 – Modelo de Interação do Protocolo do Vendedor *InformarFimPrazoVenda*

Apêndice B

Telas do Sistema

Nesta seção são apresentas as telas do sistema Babilônia de mercado virtual. A Figura 82 apresenta a tela inicial do sistema, onde os usuário cadastrados podem se registrar no sistema (*login*) e os usuário novos podem se cadastrar (momento de criação do agente usuário). A Figura 83 mostra o formulário de cadastro de usuário.

Após registrar-se no sistema (*login*) o usuário tem acesso ao menu de opções que pode ser visualizado na Figura 84. Neste exemplo foi selecionada a opção *"Comprar"* e o formulário contendo os dados do perfil de compra é apresentado. O perfil de compra neste caso é o de um automóvel Astra da Chevrolet do ano de 1999 cujo preço esteja entre 10.000 e 30.000 reais (a certificação do item e do vendedor não foi implementada). O envio deste formulário implica na criação do agente comprador. A Figura 85 apresenta a tela de visualização do agente comprador com seus dados de perfil de compra e com as opções para listar os itens encontrados e negociados deste agente.

Quando o usuário seleciona a opção de menu "Vender" o formulário da Figura 86 é apresentado. Neste caso o perfil de venda refere-se a um automóvel Astra da Chevrolet do ano de 1999 no valor de 22.000 reais (a certificação do comprador não foi implementada). Os parâmetros de negociação deste automóvel incluem a parcela de decremento de 100 reais e o valor mínimo de venda de 20.000 reais. O envio deste formulário determina a criação do agente vendedor. A Figura 87 apresenta a tela de visualização do agente vendedor com seus dados de perfil de venda e com as opções para listar os itens negociados deste agente.

A Figura 88 apresenta os itens encontrados pelo agente comprador. Neste caso o item selecionado é o primeiro item da lista, e os parâmetros da negociação incluem a parcela de incremento de 100 reais, o valor inicial de 19.000 reais e o valor final de 20.500 reais. A submissão deste formulário (botão Selecionar) determina o início de uma negociação.

Caso uma negociação já tenha chegado ao fim é possível visualizar seu resultado através do link “Listar Itens Negociados” tanto no agente comprador como no agente vendedor. Como pode ser observado na Figura 89 e na Figura 90.

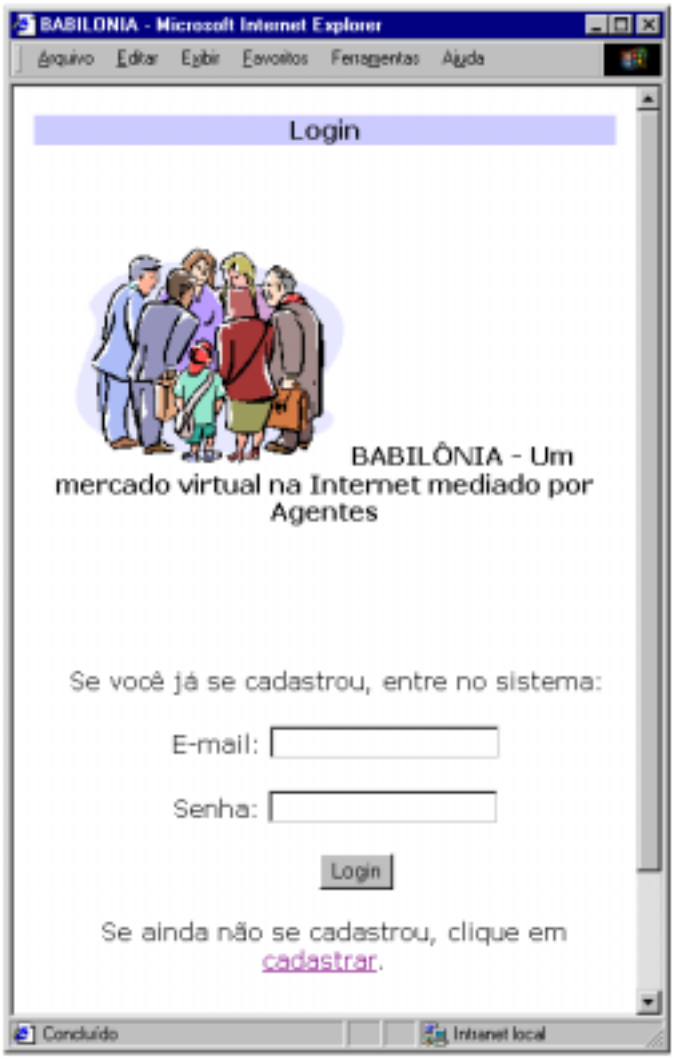

Figura 82 – Tela inicial do sistema

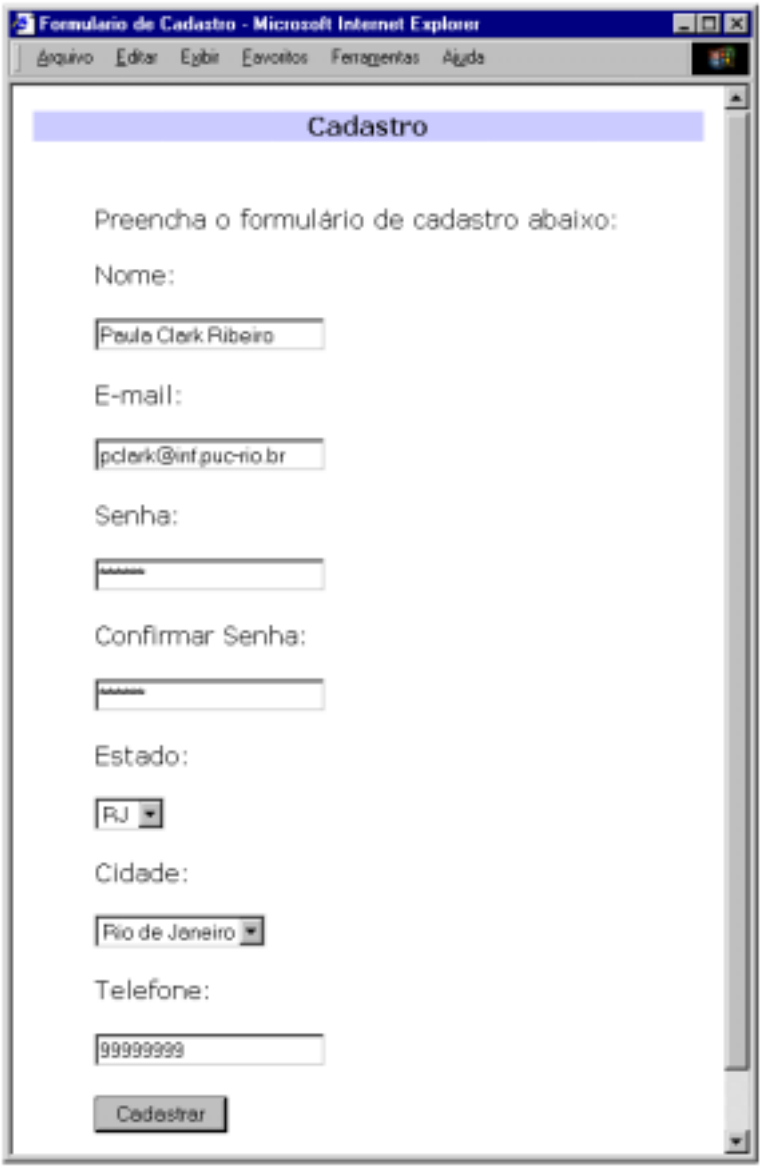

Figura 83 – Formulário de cadastro de usuário

Figura 84 – Formulário de criação do perfil de compra

Figura 85 – Tela de visualização do perfil de compra do agente comprador

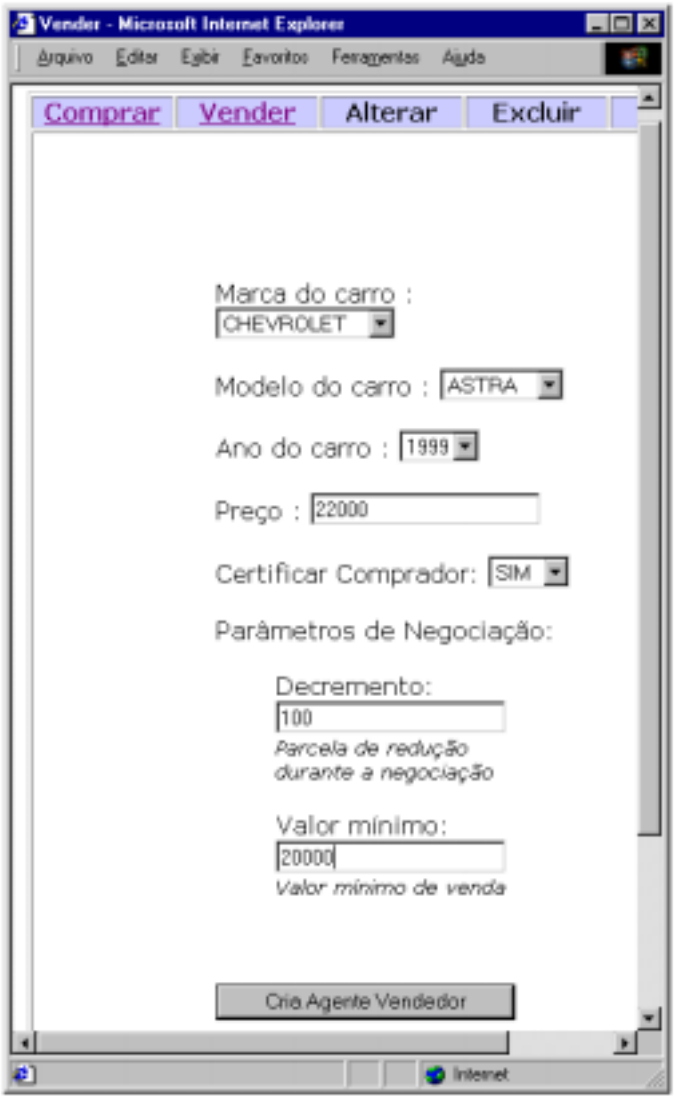

Figura 86 – Formulário de criação do perfil de venda

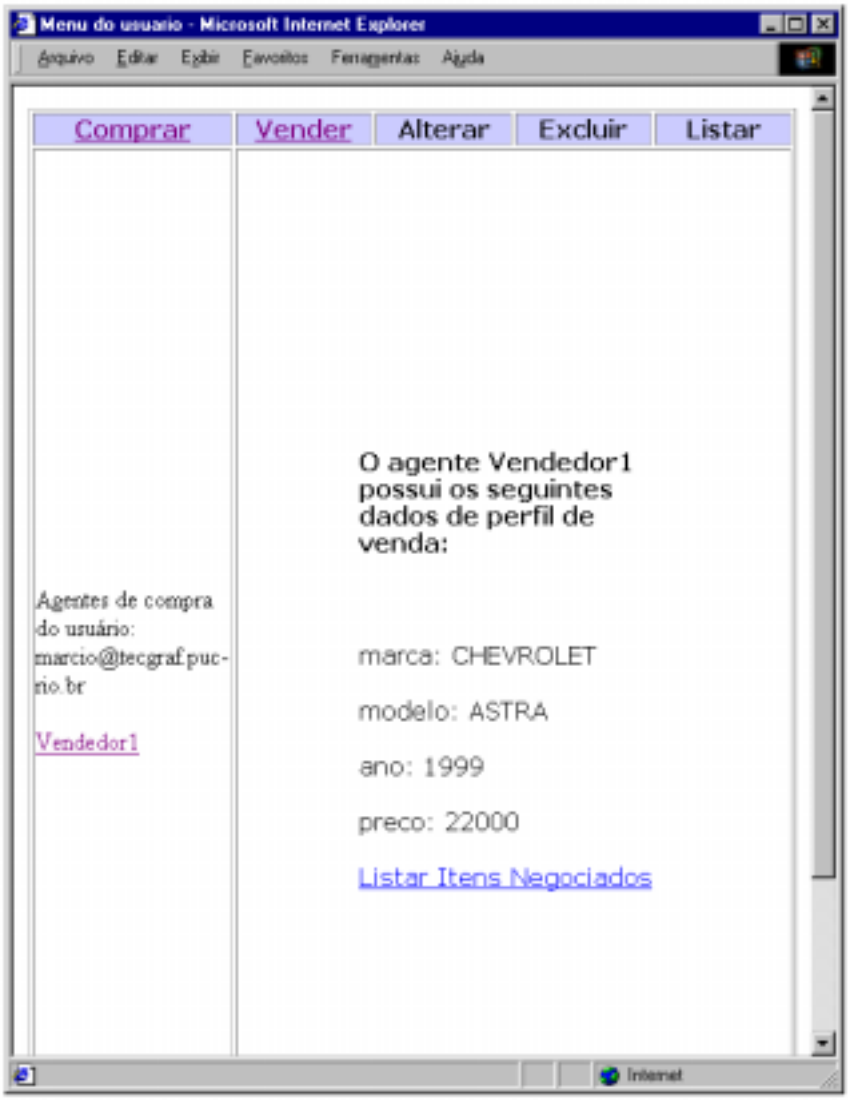

Figura 87 – Tela de visualização do perfil de venda do agente vendedor

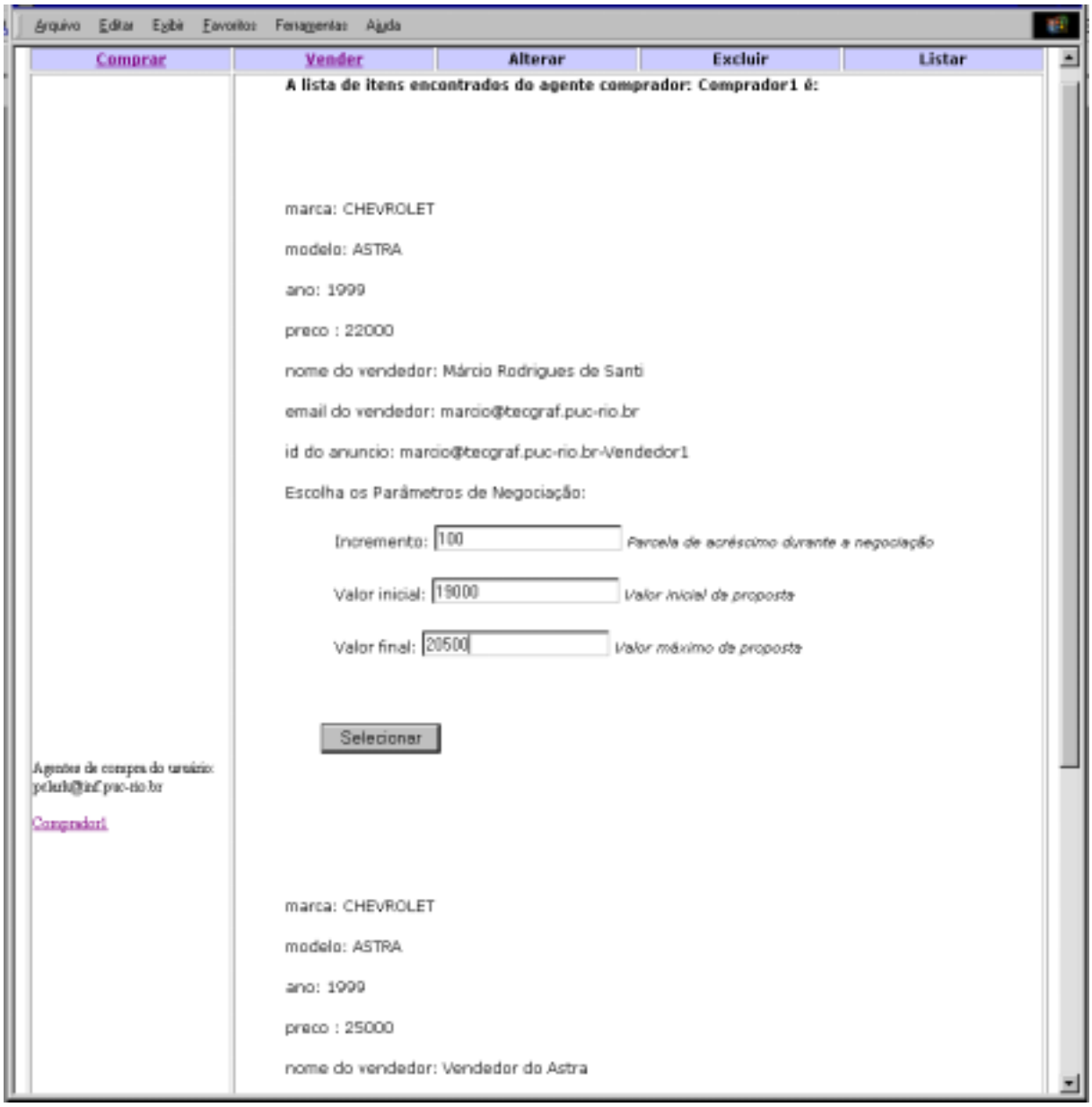

Figura 88 – Tela de visualização da lista de itens encontrados pelo agente comprador

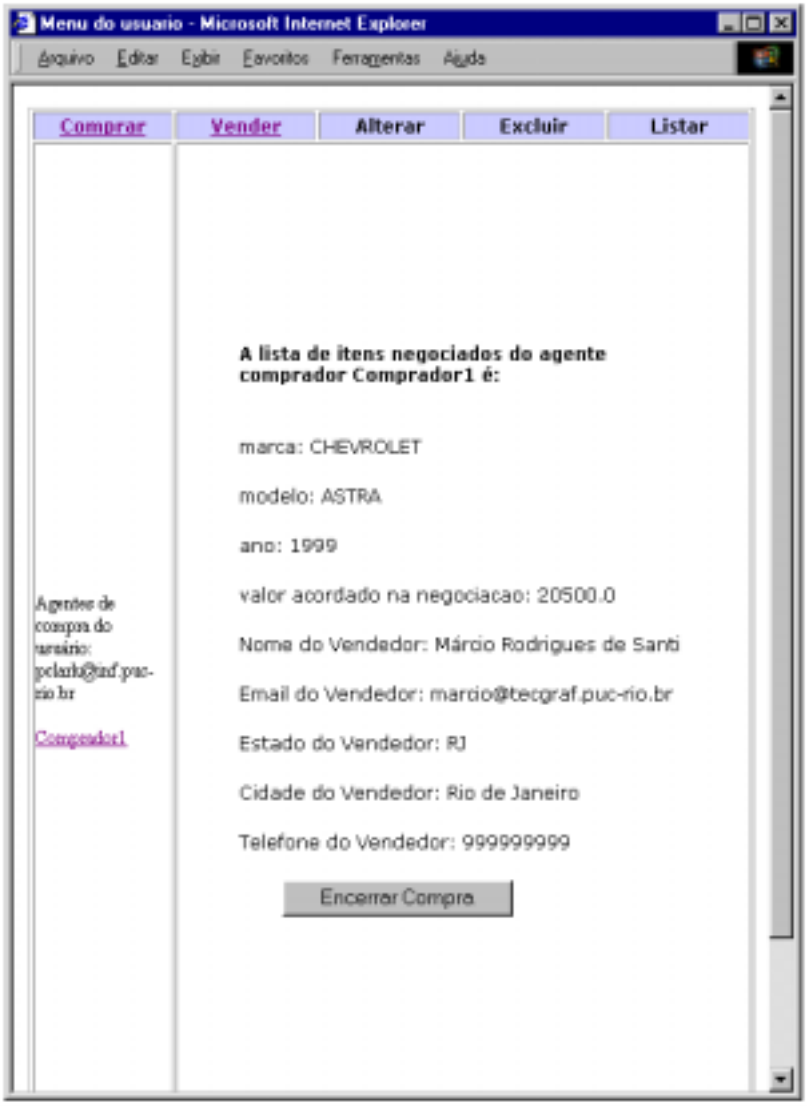

Figura 89 – Tela de visualização dos itens negociados pelo agente comprador

Figura 90 – Tela de visualização dos itens negociados pelo agente vendedor