6
Usando MAS-ML para Modelar um Sistema Multiagentes

Neste capítulo, modelamos um mercado virtual para ilustrar o uso de MAS-ML em uma aplicação prática. Foi escolhido um exemplo de comércio eletrônico, porque há referências na literatura (He et al., 2003; Jennings et al., 1998; Lomuscio et al., 2003) como sendo um *benchmark* de SMAs. O exemplo é descrito e modelado na Seção 6.2. Descrevemos todas as entidades que compõem o sistema de mercado virtual, todas as propriedades associadas às entidades e todos os relacionamentos entre as entidades. Ao ilustrar as entidades de SMAs usando os elementos de diagrama definidos na Seção 4.3.1, omitimos algumas propriedades. Além disso, foram modelados quatro diagramas de seqüência a fim de descrever diferentes aspectos dinâmicos.

Como o intuito de orientar e auxiliar os usuários de MAS-ML durante a aplicação da linguagem para modelar sistemas multiagentes, definimos uma abordagem de modelagem (Seção 6.1). O sistema multiagentes é modelado usando MAS-ML e seguindo a abordagem de modelagem. É importante ressaltar que, da mesma forma que UML, a linguagem de modelagem MAS-ML não possui um modelo de processo associado. A abordagem de modelagem proposta é apenas uma indicação de como a linguagem de modelagem proposta pode ser mais facilmente aplicada em aplicações complexas.

Este capítulo também apresenta a transformação de modelos estruturais de MAS-ML associados ao sistema do exemplo em código Java (Seção 6.2.3). O código reflete todos os conceitos relativos aos modelos, mais todos os componentes que compõem nossa arquitetura abstrata proposta (apresentada na Seção 5.4.1). A descrição textual completa do exemplo usando a gramática de MAS-ML é apresentada no Apêndice II. Finalmente, o capítulo apresenta a avaliação da linguagem MAS-ML e de outros exemplos de SMAs modelados usando MAS-ML.

6.1.
A Abordagem de Modelagem

O objetivo da abordagem de modelagem é ajudar o designer a modelar um sistema multiagentes usando MAS-ML. Ela descreve o que é necessário ser especificado a fim de representar completamente um design por meio de diagramas de MAS-ML. A abordagem de modelagem orienta o designer descrevendo uma seqüência de identificação de entidades.

6.1.1.
Modelando Aspectos Estruturais

Usando MAS-ML, os aspectos estruturais de sistemas são modelados por meio do diagrama de classes, papel e organização, ou seja, os aspectos estruturais das classes, classes de ambiente, classes de papel, classes do agente e classes de organização são modelados usando esses três diagramas. Um diagrama de organização é completamente modelado quando (i) já foram definidas as classes de organização e ambiente, (ii) as classes de papel de objeto e as classes de papel de agente definidas na classe de organização já foram identificadas e (iii) as classes de suborganização, classes de agente e classes que estão exercendo as classes de papel já foram descritas.

Um diagrama de papel é completamente modelado quando todos os relacionamentos entre as classes de papel e entre as classes de papel e as classes já tiverem sido identificados. Um diagrama de classes é completamente definido quando (i) os relacionamentos entre as classes já tiverem sido identificados, (ii) os relacionamentos entre as classes e as classes de ambiente, organização e agente já tiverem sido definidos e quando (iii) os relacionamentos entre as classes de agente, entre as classes de organização e entre as classes de ambiente já tiverem sido descritos.

6.1.1.1.
Identificação do Ambiente e da Organização Principal

O desenvolvimento de um sistema deve começar com a identificação das classes de ambiente e da organização principal. A instância da organização principal residirá na instância da classe de ambiente. A descrição da classe da organização principal inclui a definição de seus objetivos, crenças, planos, ações e axiomas.

A descrição da classe de ambiente depende de suas características. Se as instâncias de ambiente forem elementos ativos, a classe de ambiente deverá ser modelada como um agente com objetivos, crenças, planos e ações. Se forem elementos passivos, a classe de ambiente deverá ser modelada como um objeto com atributos e métodos.

6.1.1.2. Identificação de Papéis

Depois da identificação das classes de ambiente e organização principal, os papéis exercidos na organização principal podem ser identificados. Ao identificar as classes de papel de objeto, os atributos e os métodos de cada classe do papel de objeto devem ser definidos. A fim de identificar as classes de papel do agente, os objetivos, crenças, deveres, direitos e protocolos de cada uma dessas classes devem ser definidos. A definição de suas propriedades sofre a influência dos axiomas das classes de organização que definem o papel.

Os relacionamentos entre as classes de papel não precisam ser definidos ao desenvolver um diagrama de organização porque esses relacionamentos são modelados em diagramas de papel. Ao desenvolver o diagrama de papel, os relacionamentos entre as classes de papel e entre as classes de papel e as classes são modelados. O relacionamento entre os papéis pode ser identificado com base nos protocolos definidos por eles. Um protocolo define as mensagens enviadas e recebidas pelas duas entidades descrevendo suas interações.

6.1.1.3. Identificação das Entidades

As entidades que exercem os papéis precisam ser identificadas. As classes de papel associadas a uma classe de suborganização ou agente podem influenciar sua definição. Os objetivos das classes de papel estão relacionadas aos objetivos das classes de suborganização ou agente. As ações e os planos do agente ou suborganização estão diretamente associados aos deveres, direitos e protocolos definidos nas classes de papel. Além do mais, os axiomas associados à classe de suborganização estão relacionados aos axiomas descritos em sua classe de superorganização. Os axiomas de uma classe de suborganização são definidos com base nos axiomas da classe de superorganização.

Quando um papel do agente é exercido por uma suborganização, deve ser criado um novo diagrama de organização a fim de descrever a organização, os papéis definidos, as entidades que exercem esses papéis e os ambientes em que residem suas instâncias. Para cada classe de suborganização definida no sistema, deve ser criado um diagrama de organização. Durante a definição de um diagrama de organização, só é importante definir as propriedades das entidades e os papéis que exercem. Os relacionamentos entre essas entidades são modelados no diagrama de classes.

Nem todas as classes são modeladas em diagramas de papel e organização. As classes que não exercem papéis e não estão relacionadas a papéis não são modeladas em diagramas de papel e organização. As classes que não são modeladas em diagramas de papel e organização devem ser modeladas em diagramas de classes.

Todas as classes de entidade modeladas em um diagrama de organização estão relacionadas à mesma classe de ambiente. Todas as entidades que exercem papéis em uma organização residem na mesma instância de ambiente dela. Entidades que residem em outro ambiente não podem exercer papéis nessas organizações. Na verdade, não é possível garantir durante o processo de modelagem que todas as instâncias de objeto, suborganização e agente residirão na mesma instância de ambiente em que residem as organizações em que estão exercendo papéis. Essa restrição deve ser garantida no momento da execução.

6.1.2. Modelando Aspectos Dinâmicos

O foco principal de um diagrama de seqüência é modelar interações entre as entidades e as intra-ações relacionadas executadas pelas entidades envolvidas nas interações. Um diagrama de seqüência modela as interações entre (i) agentes que estão exercendo papéis, (ii) organizações que estão exercendo papéis, (iii) ambientes e (iv) objetos, que estejam exercendo papéis ou não. Um diagrama de seqüência também modela planos e ações de agentes, organizações e ambientes ativos. Como o diagrama de seqüência de MAS-ML estende UML, ele também modela tudo que normalmente pode ser modelado no diagrama de seqüência de UML. Um diagrama de seqüência depende da identificação de todas as entidades definidas na aplicação de SMAs.

Ele pode ser usado para modelar um conjunto de interações relacionadas entre instâncias diferentes ou pode ser usado para modelar uma única interação entre duas instâncias. As interações entre as entidades caracterizam-se pelo envio ou recebimento de uma mensagem ou por uma chamada de método. Planos e ações são executados pela entidade que está enviando a mensagem e pela entidade que está recebendo a mensagem. Portanto, os planos e as ações associados às mensagens também devem ser modelados. Ademais, métodos ou ações são executados por entidades que estão chamando métodos. Portanto, esses métodos e ações que dão origem a chamadas e os métodos que são chamados devem ser modelados. Se um método for chamado por uma ação, o plano associado a ela também deverá ser modelado junto com o método chamado.

Um diagrama de seqüência também pode ser usado para modelar um plano específico ou uma ação específica. Nos dois casos, apesar de as interações entre as entidades serem modeladas, o diagrama se concentra na descrição do plano ou da ação.

6.2. O Exemplo do Mercado Virtual

A fim de exemplificar o uso da linguagem de modelagem de MAS-ML e o transformador MAS-ML2Java, usaremos o exemplo do mercado virtual. *Mercados virtuais* são mercados localizados na Web em que os usuários compram e vendem itens. Supomos que há muitos *mercados virtuais* que compartilham as mesmas características. Os *mercados virtuais* modelados nesta Seção têm estas características. Cada um é composto por um *mercado principal* em que os usuários podem negociar qualquer tipo de *item*. Além disso, o *mercado principal* define dois tipos de mercados que negociam itens com determinadas características. Os *mercados de produtos especiais* negociam itens caros e de alta qualidade e os *mercados de produtos usados* negocia itens de baixa qualidade e preço baixo. Os usuários podem comprar *itens* no *mercado principal*, em *mercados de produtos especiais* e em *mercados de produtos usados*. Eles também podem vender seus itens nos *mercados de produtos usados*. No *mercado principal* e nos *mercados de produtos especiais*, os usuários compram os *itens* disponíveis no mercado. O *mercado principal* avalia os lucros. Portanto, os *mercados de*

produtos especiais e usados devem enviar as informações relativas às vendas para o *mercado principal*.

No *mercado principal* e nos *mercados de produtos especiais*, os usuários procuram um vendedor e enviam uma descrição do item desejado. O vendedor, criado pelo mercado para negociar com o comprador, é responsável por verificar se há um item com as mesmas características no *ambiente* (o mercado virtual). O *ambiente* armazena todos os itens que serão vendidos nesses mercados. Se o item for encontrado, o vendedor negociará com o comprador.

Nos *mercados de produtos usados,* os vendedores e os compradores são os usuários. Os usuários que desejam vender itens devem anunciá-los. Os que desejam comprar devem procurar anúncios no mercado. Se o comprador encontrar o item desejado, ele começará uma negociação com o vendedor.

O exemplo do mercado será expresso seguindo a abordagem de modelagem definida na Seção 6.1. A identificação das entidades de SMAs apresentadas nesse exemplo será definida a fim de criar diagramas estruturais e dinâmicos.

6.2.1. Diagramas Estruturais

O diagrama de organização apresentado na Figura 65 ilustra a organização principal usando a representação simplificada que omite os compartimentos inferior e intermediário dos elementos do diagrama. A fim de criar esse diagrama, foram definidas as classes da organização principal e do ambiente, junto com as classes de elemento, suborganização e papel.

O diagrama apresenta a classe da organização principal *General Store* e duas classes de suborganização, *Imported Bookstore* e *Second-hand Bookstore* que exercem os papéis *Market of Special Goods* e *Market of Used Goods*, respectivamente. Ademais, foram modelados nesse sistema dois tipos de agente: *user agent* e *store agent*. O diagrama também ilustra as classes de papel *seller* e *buyer* definidas pela classe da organização principal e exercidas por *store agents* e *user agents*, respectivamente. Foram definidos outros dois papéis pela organização principal: as classes de papel de objeto *desire* e *offer*. As instâncias desses papéis são exercidas pelas instâncias da classe *Book*.

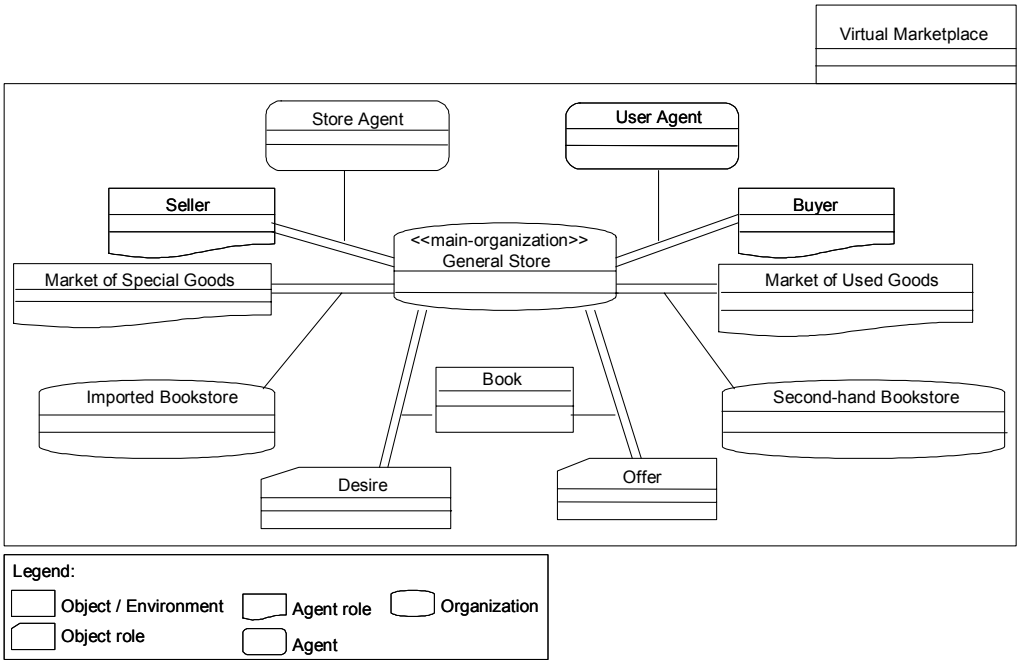

Figura 65 – O diagrama de organização modelando a organização principal.

6.2.1.1. Identificação da Organização Principal e Ambiente

A partir da descrição do problema, é possível identificar uma organização principal que reside no ambiente *Virtual Marketplace*. O ambiente é um elemento passivo modelado como um objeto que armazena itens que serão negociados como um de seus *atributos*. Ele implementa os *métodos* get e set a fim de acessar esses itens. Além disso, também armazena informações sobre os agentes e as organizações que residem nele. Portanto, esses *métodos* são definidos para acessar essas informações.

Como há na Web mercados virtuais que compartilham as mesmas características, o usuário que não conseguir comprar um item em um mercado poderá tentar localizá-lo em outro. Agentes que representam o usuário podem se mover de um ambiente para outro. A fim de possibilitar isso, os mercados devem compartilhar a mesma estrutura e devem se conhecer. O *relacionamento* entre os mercados virtuais é modelado mais adiante no diagrama de classes (Figura 88). Para permitir que um agente ou uma organização se mova de um ambiente para outro, o ambiente deverá ter *métodos* para verificar a permissão relacionada a entrada e a saída dos elementos. O ambiente também define os *métodos* get e set para acessar as demais instâncias de ambiente. A classe Virtual Marketplace está

ilustrada em Figra 66. Alguns métodos e atributos associados a essa classe foram omitidos.

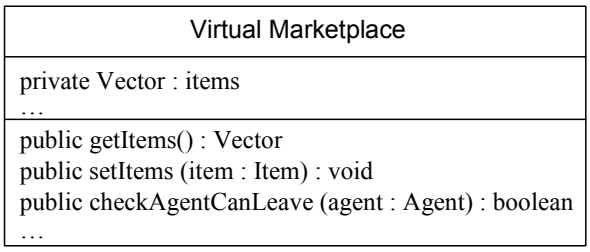

Figura 66 – A classe de ambiente Virtual Marketplace (parcial).

A organização principal *General Store* (parcialmente ilustrada na Figura 67) representará o mercado principal do sistema. Como *General Store* é a organização principal do sistema, ela não exerce qualquer papel e apenas uma instância dessa classe pode ser criada para cada instância de ambiente. Como os usuários podem comparar itens no mercado principal, a organização principal define os papéis do agente *buyer* e *seller*. Esses papéis serão detalhados na Seção 6.2.1.4. Os *objetivos* da organização principal são o gerenciamento dos vendedores, pedidos e lucros. Para alcançá-los, a organização principal define *planos* para (i) criar vendedores para negociar com compradores, (ii) atualizar o ambiente a fim de informar que um item não está mais disponível e (iii) avaliar os lucros resultantes das vendas. Para garantir que a organização principal receba as informações relativas às vendas, é definido um *axioma*. As *crenças* da organização principal estão relacionadas às informações relativas aos compradores, vendedores e vendas.

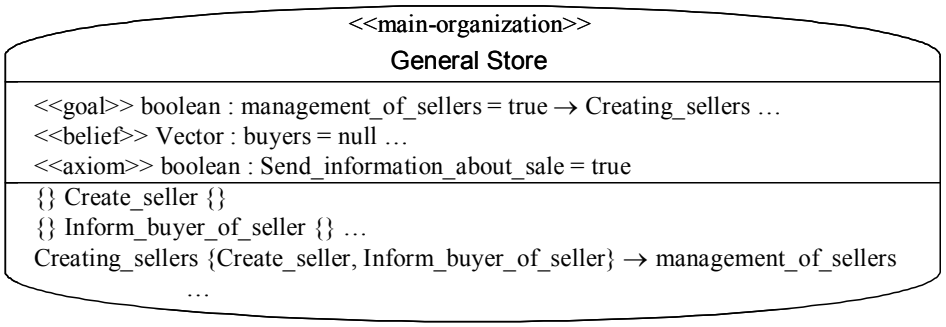

Figura 67 – A classe de organização General Store (parcial).

6.2.1.2. Identificação de Papéis: papéis exercidos por suborganizações

Junto com os papéis *buyer* e *seller*, a organização principal *General Store* também define os papéis *market of special goods* e *market of used goods* exercidos por suborganizações. Como os mercados de produtos especiais vendem

produtos caros, eles devem verificar se os usuários que desejam entrar podem pagar pelos itens que desejam. A fim de garantir que os usuários serão analisados, os mercados de produtos especiais definem um *dever*. Além de outros, o mercado define o dever "analyze entrance". Além disso, ele define alguns *direitos* como "register new buyers" e "update environment".

O gerenciamento de novos compradores é um dos *objetivos* do mercado. Outro objetivo é gerenciar a criação de vendedores quando compradores entram no mercado. Ademais, todos os mercados devem gerenciar os pedidos registrando as vendas. O mercado de produtos especiais define *protocolos* para orientar as interações (i) entre si mesmo e novos compradores (protocolo "to request to enter in market of special goods"), (ii) entre si mesmo e seus próprios compradores, uma vez que interagem para procurar vendedores (protocolos "buyer registration" e "to search for seller") e (iii) entre si mesmo e seus próprios vendedores a fim de receber informações sobre as vendas (protocolos "to register sale"). A Figura 68 ilustra parcialmente o mercado de produtos especiais. São descritos o objetivo "management of buyers", o dever "analyze entrance" e o protocolo "to enter in market of special goods".

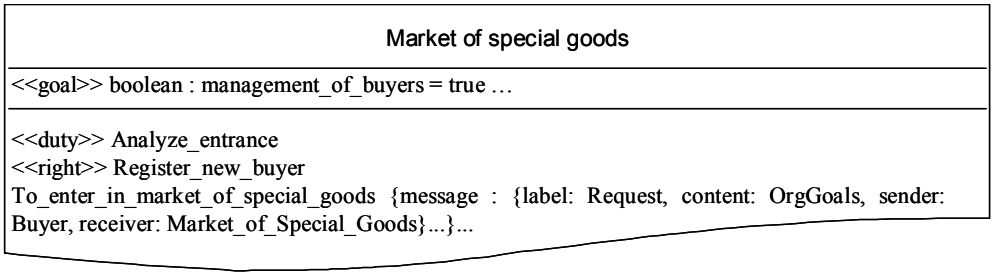

Figura 68 – A classe do papel Market of special goods (parcial).

Nos mercados de produtos usados, os usuários podem vender e comprar itens. Os mercados não controlam a entrada de vendedores porque representam usuários que estão vendendo itens. Um vendedor anuncia itens nos mercados, e os compradores buscam esses anúncios. Apesar de os mercados de produtos usados não definirem qualquer restrição à entrada de novos compradores ou novos vendedores, os mercados registram sua entrada. Além de outros *direitos*, o mercado de produtos usados define os direitos "register announcement" e "register new buyers".

Seus *objetivos* são gerenciar a entrada de vendedores e compradores e gerenciar anúncios e pedidos. Os *protocolos* definidos pelo mercado de produtos

usados estão relacionados às interações (i) entre si mesmo e novos compradores (protocolo “to enter in the market of used goods”), (ii) entre si mesmo e seus próprios vendedores, uma vez que anunciam itens e enviam informações sobre as vendas (protocolos “to announce” e “to register sale”) e (iii) entre si mesmo e seus compradores porque eles procuram vendedores (protocolos “buyer registration” e “to search for announcement”). A Figura 69 ilustra parcialmente o mercado de produtos usados. São descritos o objetivo “management of announcement”, o dever “register announcement” e o protocolo “to announce”.

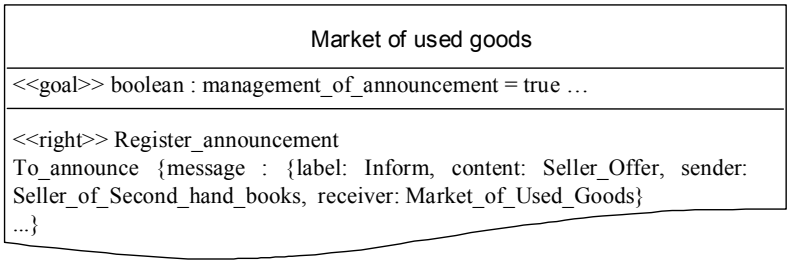

Figura 69 – A classe do papel Market of used goods (parcial).

6.2.1.3. Identificação de Suborganizações

Alguns exemplos de organizações que podem exercer o papel de mercado de produtos especiais são livrarias de livros importados e antiquários. Essas lojas vendem produtos de alta qualidade, normalmente itens caros. Alguns exemplos de organizações que podem exercer o papel de mercado de produtos usados são brechós e sebos. Essas lojas vendem produtos de baixa qualidade e preços baixos. Nesse exemplo, modelaremos uma livraria de livros importados e sebos.

Como essas livrarias exercem os papéis chamados de mercados de produtos especiais, seus objetivos precisam ser compatíveis, isto é, deve haver pelo menos um objetivo na classe da livraria de livros importados relacionado a um objetivo da classe do mercado de produtos especiais. Se os objetivos de uma organização não são compatíveis com os objetivos de um papel, a organização não tem qualquer interesse em exercer o papel. Nesse exemplo, as instâncias da classe da livraria de livros importados exercem apenas instâncias dos papéis de mercado de produtos especiais e, assim, todos os objetivos da classe de organização são compatíveis com todos os objetivos da classe do papel.

Como o papel de mercado de produtos especiais é definido no escopo da organização principal, a classe da livraria de livros importados deve obedecer a

seus *axiomas*. Para garantir que seus vendedores enviarão as informações relativas às vendas, a classe da livraria de livros importados também define o *axioma* "send information about sale".

Os *objetivos* das livrarias de livros importados são gerenciar a inclusão de novos compradores no mercado, gerenciar a criação de vendedores e gerenciar os pedidos. Para alcançá-los, a classe da livraria de livros importados define três *planos*. O plano "analyzing entrance" negocia a entrada de um novo comprador verificando se ele possui as características necessárias e registrando-o. Esse plano é descrito de acordo com o *dever* e o *direito* especificados no papel de mercado de produtos especiais e de acordo com os *protocolos* que definem a interação entre si mesmo e um comprador genérico e entre si mesmo e um comprador de livros importados. Outro plano está relacionado à criação de vendedores de livros importados. O terceiro plano gerencia os pedidos recebendo informações relativas às vendas e atualizando o ambiente. Esse plano é definido de acordo com os *protocolos* que especificam a interação entre si mesmo e um vendedor de livros importados e de acordo com o *axioma* definido pela organização. As *crenças* das livrarias de livros importados são os compradores, os vendedores, as vendas e a organização principal. A Figura 70 ilustra parcialmente essa livraria. A figura mostra (i) o objetivo "management of buyers" relacionado ao plano "analyzing entrance", (ii) a crença que representa os compradores, (iii) o axioma "send information about sale", (iv) a ação "get organization goals" e (v) o plano "analyzing entrance".

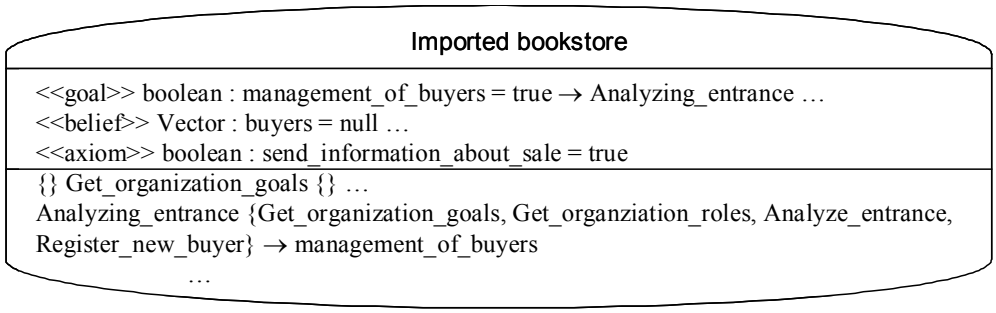

Figura 70 – A classe de organização Imported bookstore (parcial).

Os *objetivos* dos sebos (cujas instâncias exercem os papéis de mercado de produtos usados) são (i) gerenciar a entrada de novos vendedores e compradores, (ii) gerenciar os anúncios de itens que serão vendidos e (iii) gerenciar os pedidos. Os sebos têm de seguir os *axiomas* definidos na organização principal para enviar

informações sobre as vendas. Para isso, a classe de sebos define *axiomas* a fim de garantir que seus vendedores enviarão a quantidade de dinheiro relacionada às vendas. Essa classe define *planos* (i) para gerenciar a entrada de compradores e vendedores, (ii) para armazenar os anúncios e enviá-los aos compradores e (iii) para registrar as vendas. Suas *crenças* são relacionadas às vendas e aos anúncios. A Figura 71 ilustra parcialmente os sebos identificando (i) o objetivo "management of announcement", (ii) a crença que armazena o anúncio, (iii) o axioma "send information about sale", (iv) a ação "register announcement" e (v) o plano "managing the announcement".

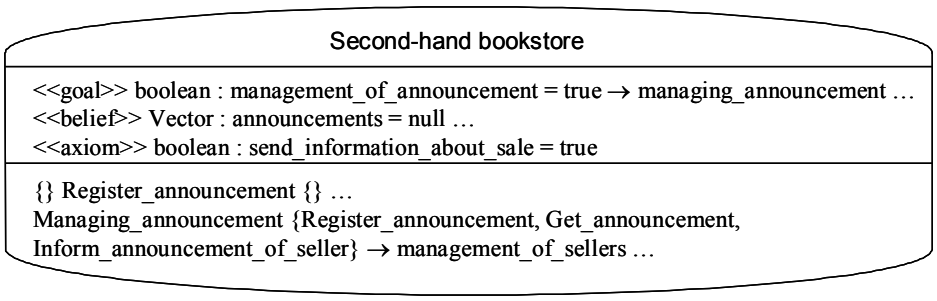

Figura 71 – A classe de organização de Sebos (parcial).

As organizações de livrarias de livros importados e sebos definem os *compradores* e *vendedores* que são diferentes daqueles previamente definidos na organização principal, isto é, *compradores* e *vendedores de livros importados* e *compradores* e *vendedores de livros usados*, respectivamente. O comprador de livros importados é exercido pelos agentes do usuário e o vendedor de livros importados por agentes da loja. O comprador e o vendedor de livros usados são exercidos pelos agentes do usuário. A Figura 72 apresenta um diagrama de organização que modela a classe *imported bookstore*, e a Figura 73 ilustra um diagrama de organização que modela a classe *second-hand bookstore*. A fim de descrever completamente esses diagramas, é necessário descrever os papéis do agente, os papéis de objeto e os agentes ilustrados nos diagramas.

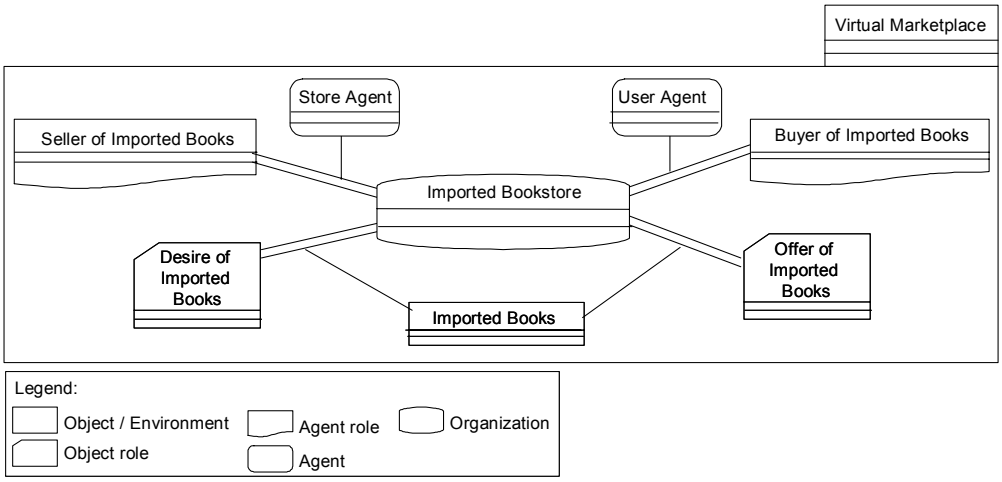

Figura 72 – O diagrama de organização da classe de organização Imported Bookstore.

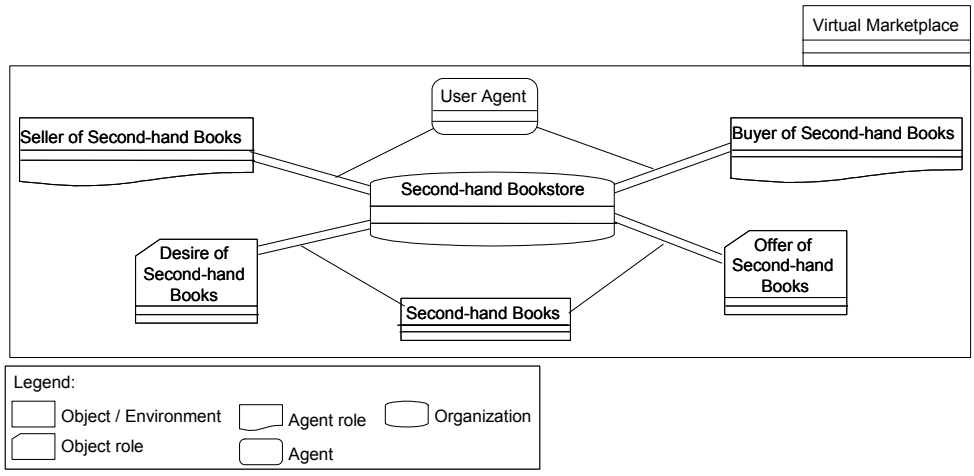

Figura 73 – O diagrama de organização da classe de organização Second-hand Bookstore.

6.2.1.4. Identificação de Papéis: papéis exercidos por agentes e objetos na organização principal

Conforme já visto, a organização principal define o papel de *comprador* (Figura 74) e o papel de *vendedor* (Figura 75) cujos *objetivos* são comprar um item e vender um item, respectivamente. Para alcançá-los, eles negociam itens armazenados no ambiente. O *dever* de um comprador é procurar vendedores. Os papéis de *comprador* e *vendedor* definem o *protocolo* "simple negotiation" que descreve como as entidades que estão exercendo esses papéis devem interagir. Esse *protocolo* define que um comprador deve perguntar a um vendedor o preço de um item. Depois de consultar o ambiente, o vendedor envia o preço ao comprador que pode aceitar ou recusar a proposta. As opções de aceitar ou recusar

uma determinada proposta são os *direitos* do papel de comprador. Se o comprador aceitar a proposta, o vendedor enviará a fatura a ele. Então, o vendedor enviará as informações relativas à venda para a organização principal, conforme especificado em seu *dever*.

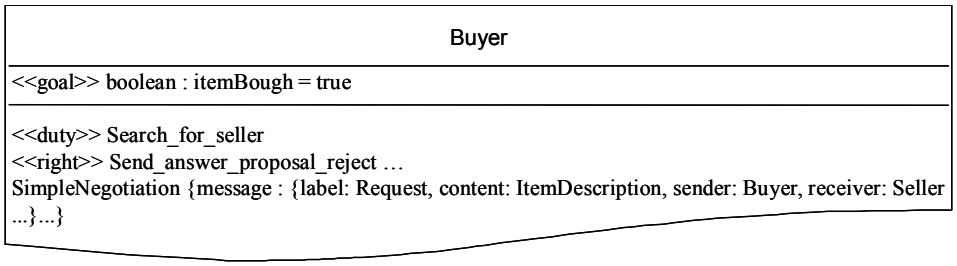

Figura 74 – A classe do papel Buyer (parcial).

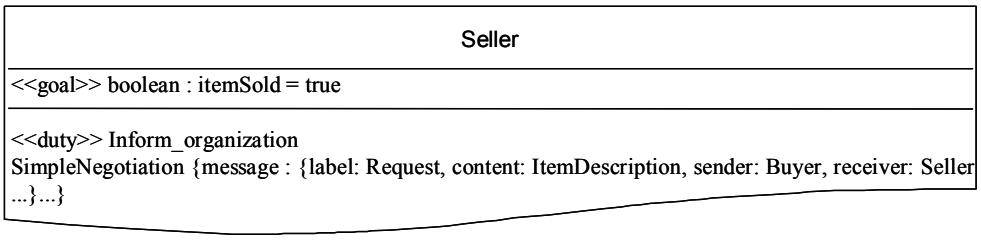

Figura 75 – A classe do papel Seller (parcial).

Os compradores e os vendedores têm diferentes visões dos *itens* que negociam. Um *item* é um *desejo* para os compradores e uma *oferta* para os vendedores. O *desejo* e a *oferta* possuem diferentes características. Suponha que o item negociado no mercado seja um livro. A classe de livro define um conjunto de atributos (título, autor, ISBN, preço) e métodos (getters e setters de cada atributo). O papel de desejo permite que o comprador defina o título, autor e o ISBN de um livro, mas permite que o comprador receba apenas o preço. Por outro lado, o papel de oferta permite que o vendedor receba o preço do livro e as informações do título, autor e o ISBN. A Figura 76 ilustra parcialmente o papel de desejo descrevendo os métodos para receber o preço e definir o título. A Figur 77 ilustra parcialmente o papel de oferta descrevendo métodos para definir o preço e receber o título.

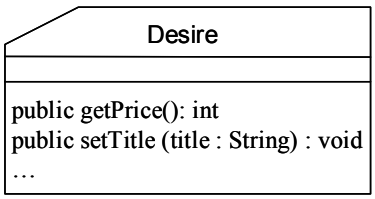

Figura 76 – A classe do papel Desire (parcial).

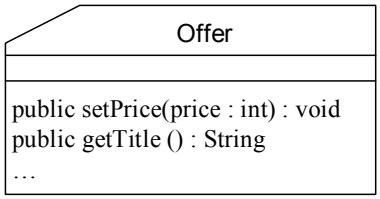

Figura 77 – A classe do papel Offer (parcial).

6.2.1.5. Identificação de Papéis: papéis exercidos por agentes e objetos em livrarias de livros importados e sebos

Conforme mencionado anteriormente, a suborganização livraria de livros importados define as classes de papel *buyer* e *seller of imported books*, e a suborganização sebo, as classes de papel *buyer* e *seller of second-hand books*. Todos os compradores e vendedores no sistema têm os mesmos *objetivos*: comprar e vender um item, respectivamente. Contudo, definem *deveres, direitos* e *protocolos* diferentes. Analisando os protocolos definidos pelos papéis, os relacionamentos entre os papéis são gerados e o diagrama de papel é criado. O diagrama é apresentado mais adiante nas Figuras 84 e 83.

A classe do papel *buyer* definida na organização principal descreve o protocolo "simple negotiation" com o vendedor e o protocolo "to enter organization" com o mercado de produtos especiais e o mercado de produtos usados. A classe do papel *buyer of imported books* (Figura 78) especifica (i) o protocolo "registration" com os mercados de produtos especiais, (ii) o protocolo "to search for seller" com o mercado de produtos especiais e (iii) o protocolo "simple negotiation" com o vendedor de livros importados. Finalmente, a classe do papel *buyer of second-hand books* (Figura 79) define (i) o protocolo "registration" com o mercado de produtos usados, (ii) o protocolo "simple negotiation" com o vendedor de livros usados, (iii) o protocolo "complex negotiation" com o vendedor de livros usados e (iv) o "to search for announcement" com o mercado de produtos usados. O comprador de produtos especiais e o comprador de livros usados têm um *dever* de se registrar em suas organizações.

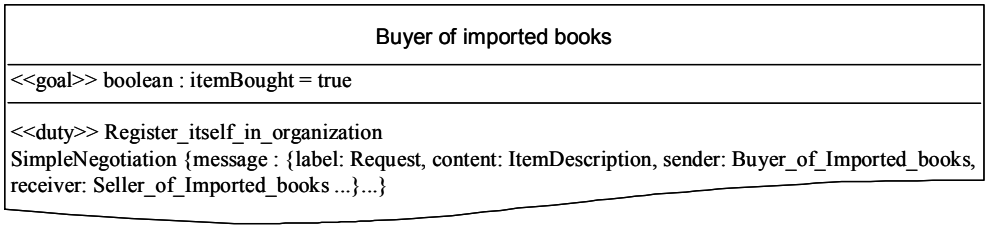

Figura 78 – A classe do papel Buyer of imported books (parcial).

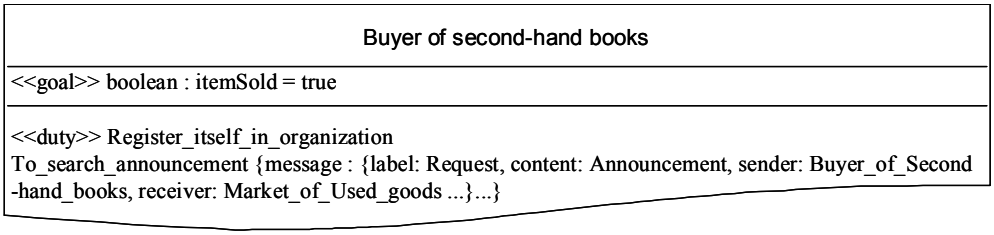

Figura 79 – A classe do papel Buyer of second-hand books (parcial).

A classe do papel *seller* definida na organização principal apenas especifica o protocolo "simple negotiation" com o comprador. Ele é estendido pela classe do papel *seller of imported books* (Figura 80) porque o vendedor de livros importados também define os mesmos objetivos e deveres e especializa o protocolo "simple negotiation". Essa classe *seller of imported books* define o protocolo "simple negotiation" com o comprador de livros importados e o protocolo "to register sale" com o mercado de produtos especiais. Além do mais, a classe do papel *seller of second-hand books* (Figura 81) estende a classe do papel *seller of imported books* ao definir os protocolos "complex negotiation" com o comprador de livros usados e o protocolo "to announce" com o mercado de produtos usados. O protocolo "simple negotiation" também foi especializado.

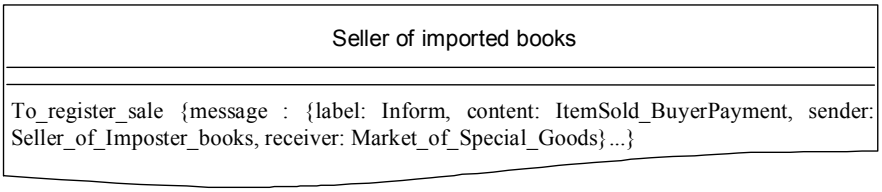

Figura 80 – A classe do papel Seller of imported books (parcial).

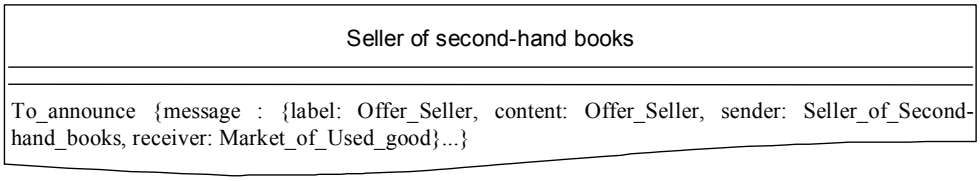

Figura 81 – A classe do papel Seller of second-hand books (parcial).

Como os itens vendidos em livrarias de livros importados e sebos são diferentes, é necessário definir novos objetos para representá-los. As classes *imported book* e *second-hand book* possuem as mesmas características associadas a um livro e outros atributos a fim de indicar o país de origem do livro e um atributo a fim de indicar a aparência do livro, respectivamente. Devem ser definidas novas classes de papel de objeto de desejo e oferta devido à criação desses novos objetos. As classes de papel *desire of imported books* e *offer of imported books* estendem as classes de papel *desire* e *offer*, respectivamente, incluindo métodos para acessar o atributo do país de origem. A *desire of second-hand books* e *offer of second-hand books* estendem as classes de papel *desire* e *offer*, respectivamente, incluindo métodos para acessar o atributo de aparência. As quatro classes de papel de objeto estão ilustradas na Figura 82.

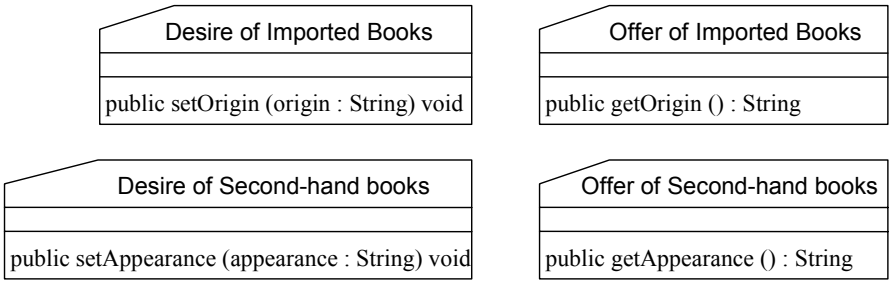

Figura 82 – As classes de papel desire e offer of imported books e as classes de papel desire e offer of second-hand books.

A Figura 83 ilustra um diagrama de papel enfatizando os relacionamentos entre as classes de papel do agente e as classes de papel de objeto. A Figura 84 mostra um diagrama de papel enfatizando o relacionamento *specialization* entre classes de papel de objeto e entre classes de papel do agente. O diagrama de papel na Figura 83 foi criado com base nos protocolos definidos pelas classes de papel. O diagrama ilustra três relacionamentos distintos: *association*, *aggregation* e *control*.

O relacionamento *control* é usado entre a classe do papel *market of special goods* e a classe do papel *seller of imported books* porque cada *market* controla seus *sellers*. A criação e a destruição de vendedores são controladas pelos mercados. Os mercados decidem quando deve ser criado um vendedor. Esse não é o caso dos relacionamentos entre as classes *market of used goods* e *sellers of second-hand books*. Os mercados de produtos usados não têm qualquer controle

sob seus vendedores. Os vendedores são sempre criados quando os usuários desejam entrar em mercados para vender itens.

O relacionamento *aggregation* é usado entre as classes de papel *buyer* e *buyer of second-hand books* e entre as classes *buyer* e *buyer of imported books*. Isso significa que para alcançar o objetivo do comprador, um agente pode ter de exercer os papéis de comprador de livros usados ou de comprador de livros importados. O relacionamento *association* foi usado entre cada dois papéis que interagem durante o envio e o recebimento de mensagens.

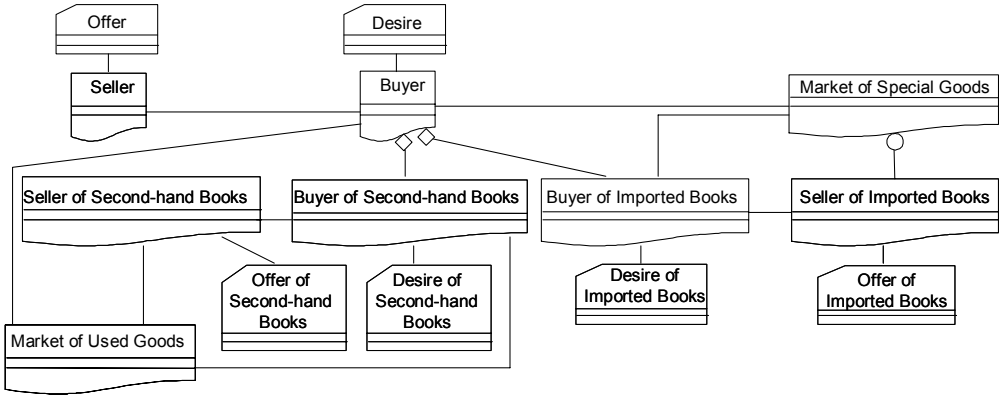

Figura 83 – O diagrama de papel (parte I).

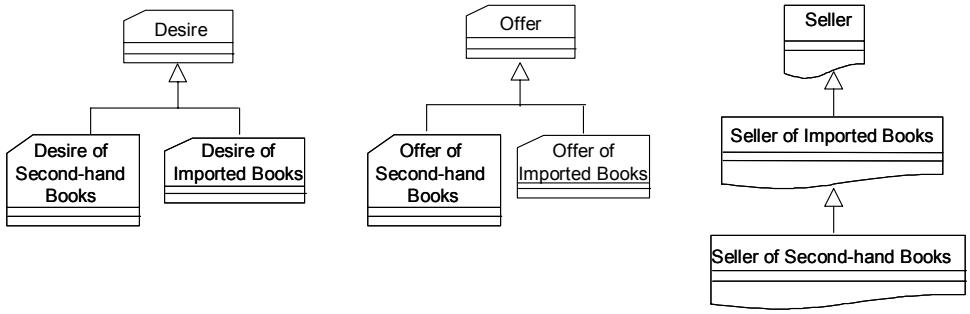

Figura 84 – O diagrama de papel (parte II).

6.2.1.6. Identificação de Classes e Agentes

Conforme descrito anteriormente, os itens vendidos nas organizações são livros. Há três tipos de livros: *book*, *imported book* e *second-hand book*, conforme ilustrado na Figura 85. Livros importados e livros usados possuem propriedades semelhantes descritas na classe book. Ademais, a classe *imported book* define um atributo para descrever o país de origem do item e a classe *second-hand book* define um atributo para indicar a aparência do item.

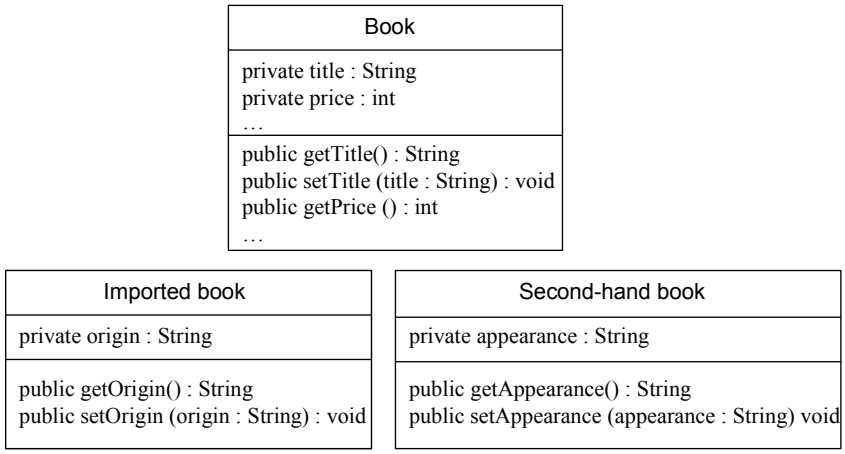

Figura 85 – As classes Book, Imported book e Second-hand book.

Nesse ponto, também é necessário descrever os agentes que exercerão os papéis do agente. Conforme mencionado, há dois tipos de agentes. A classe *user agent* (Figura 86) representa os usuários no sistema. Uma *user agent* é criada quando um novo usuário deseja ter um "item bought" ou um "item sold". A instância do agente de usuário *objetivos* depende dos objetivos do usuário. A classe do agente do usuário descreve os objetivos. Entretanto, se o objetivo de uma instância do agente de usuário for ter um item comprado, o objetivo relacionado a ter um item vendido será excluído. O oposto também é verdadeiro.

Para alcançar seus objetivos, o agente de usuário deve precisar exercer os *papéis* de comprador, comprador de livros importados, comprador de livros usados e vendedor de livros usados. O agente do usuário pode ter *planos* associados a seus papéis e objetivos. Os planos estão relacionados a protocolos definidos nos papéis que o agente pode exercer e respeitam os deveres e direitos definidos nesses papéis.

Como o objetivo "item bought" é um objetivo genérico, ele foi subdividido em subobjetivos a fim de facilitar a definição de planos específicos. Os subobjetivos são "to search for seller", "to negotiate" e "to enter organization". Ao alcançá-los, o objetivo também é alcançado. Antes de tudo, um agente deve tentar alcançar o objetivo "to search for seller". Em seguida, se ele encontrar um vendedor, o agente deverá começar a negociar com ele. Se o agente não puder comprar o item, ele poderá entrar em outra organização. Foram descritos cinco *planos* relacionados a esses objetivos: (i) o plano "searching for seller" (que está relacionado ao *objetivo* "to search for seller"), (ii) o plano "buying item" (que está

relacionado ao *objetivo* “to negotiate” e ao *protocolo* “simple negotiation”), (iii) o plano “complex buying item” (que está relacionado ao *objetivo* “to negotiate” e ao *protocolo* “complex negotiation”), (iv) o plano “entering organization” (que está relacionado ao *objetivo* “to enter organization” e aos *protocolos* “to request to enter the market of used goods”, “to request to enter the market of special goods”, “buyer registration” e “to search for announcement”) e (v) o plano “leaving environment” (que está relacionado ao mesmo *objetivo* e *protocolo* que o plano “entering organization”).

O objetivo “item sold” também foi subdividido. Seus subobjetivos são “to announce item” e “to negotiate”. Um agente de usuário que esteja exercendo o papel de vendedor de livros usados pode anunciar seus itens e deve negociar com os compradores. A classe do agente do usuário define dois planos relacionados a esses objetivos: (i) o plano “announcing item” (que está relacionado ao *objetivo* “to announce item” e ao *protocolo* “to announce”) e (ii) o plano “selling item” (que está relacionado ao *objetivo* “to negotiate” e aos *protocolos* “simple negotiation” e “complex negotiation”). Além de outras crenças, a classe do agente do usuário define a *crença* “item” que é usada para armazenar o item que as instâncias do agente do usuário desejam vender ou o item que desejam comprar. A Figura 86 ilustra a classe User Agent descrevendo (i) o objetivo “itemBought” e seus subobjetivos, (ii) a crença que armazena um item, (iii) a ação “search for seller” e (iv) os planos “searching seller” e “buying item”.

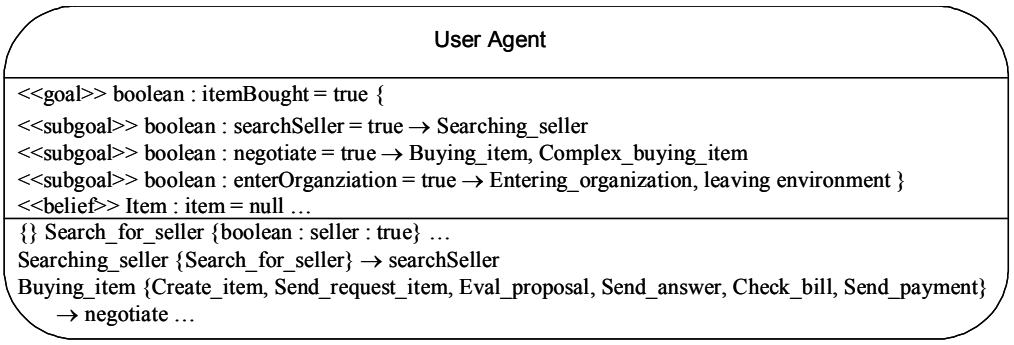

Figura 86 – A classe User agent (parcial).

Uma *store agent* (Figura 87) representa as preferências do sistema. Essa classe descreve um único *objetivo;* ou seja, vender um item. Ela pode exercer os *papéis* de vendedor e de vendedor de livros importados. Independente do papel que o agente está exercendo, ele executa o *plano* “selling item” (que está

relacionado ao protocolo “simple negotiation”) e o *plano* “informing organization about sale” (que está relacionado ao protocolo “to register sale”).

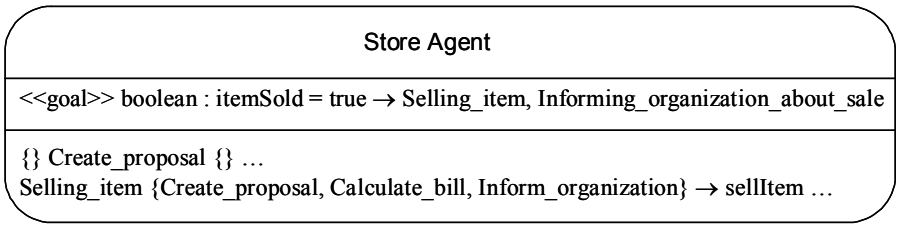

Figura 87 – A classe Store agent (parcial).

O diagrama de classes ilustrado na Figura88 completa a modelagem dos aspectos estruturais do sistema. Ele especifica os relacionamentos entre classes e entre ambientes. Agentes e organizações não são modelados nesse diagrama porque o relacionamento *specialization* não é usado nesse exemplo entre agentes ou entre organizações e porque agentes e objetos não estão diretamente relacionados a classes.

O ambiente possui um relacionamento de auto-referência porque agentes podem se mover de uma instância de ambiente para outra. O sistema não define ambientes com diferentes características e, portanto, todas as classes residem na mesma classe de ambiente. As classes imported book e second-hand book especializam a classe book, que é uma especialização da classe item.

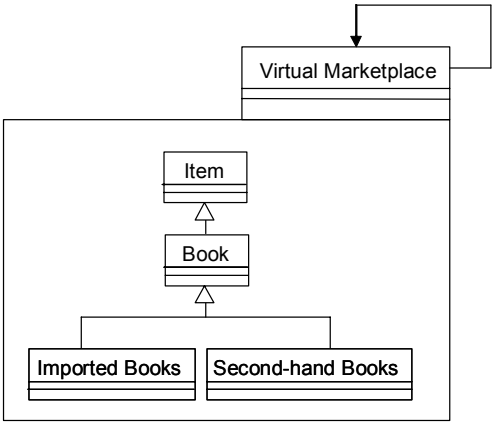

Figura 88 – O diagrama de classes.

6.2.2. Diagramas Dinâmicos

Os aspectos dinâmicos de SMAs modelam as interações entre agentes, organizações, ambientes e objetos. A fim de expressar os aspectos dinâmicos do exemplo, foram criados quatro diagramas de seqüência. O primeiro diagrama se

concentra na definição de planos, o segundo modela um protocolo, o terceiro ilustra um agente do usuário que está entrando em uma organização e o quarto diagrama mostra o movimento de um agente do usuário de um ambiente para outro.

O diagrama de seqüência na Figura 89 ilustra o plano chamado "buying item" definido pela classe *User Agent* e o plano chamado "selling item" definido pela classe *Store Agent*. O diagrama ilustra um agente do usuário que está exercendo o papel *buyer of books* e negociando com um agente da loja que está exercendo o papel *seller of goods*. Os agentes estão exercendo papéis na organização principal *Bookfinder* (instância de *General Store*) que reside em uma instância da classe *Virtual Marketplace*, chamada *Place-A*. O diagrama descreve as ações associadas a cada plano e as mensagens enviadas e recebidas pelos agentes. As mensagens estão relacionadas ao protocolo "simple negotiation" definido nas classes *Buyer* e *Seller*. Esse protocolo foi ilustrado na Figura 32, Seção 4.4.5. Ademais, o diagrama também mostra a interação entre o agente da loja e a organização principal a fim de informá-la sobre o pagamento. Conforme já mencionado, as mensagens enviadas e recebidas pelos agentes não são associadas a métodos (ações ou planos). Agentes recebem mensagens e avaliam se agirão a fim de responder a mensagem ou não.

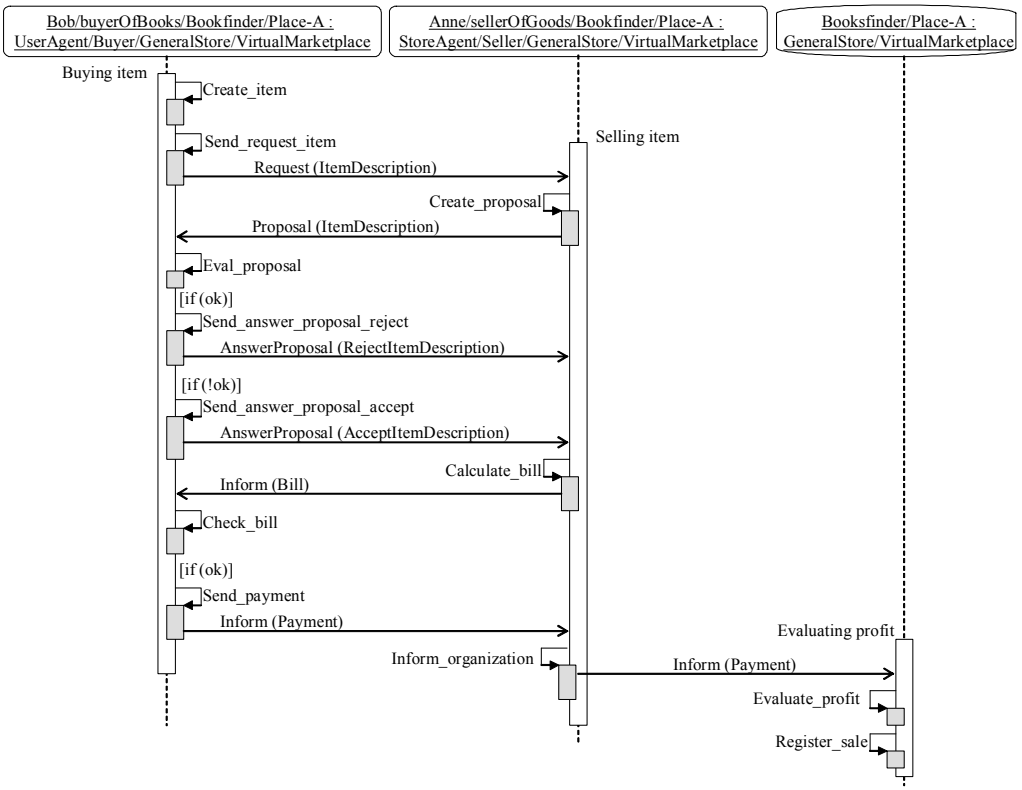

Figura 89 – Um agente do usuário negociando com um agente de loja.

Um agente do usuário pode entrar em livrarias de livros importados para exercer um papel de comprador de livros importados e também pode entrar em sebos para exercer um papel de comprador ou vendedor de livros usados. O diagrama de seqüência modelado na Figura 90 ilustra o protocolo chamado “to enter in market of used goods” definido pelo papel de comprador e mercado de produtos usados. Esse protocolo descreve a interação entre um agente do usuário que esteja exercendo o papel de comprador e um sebo que esteja executando o papel de mercado de produtos usados.

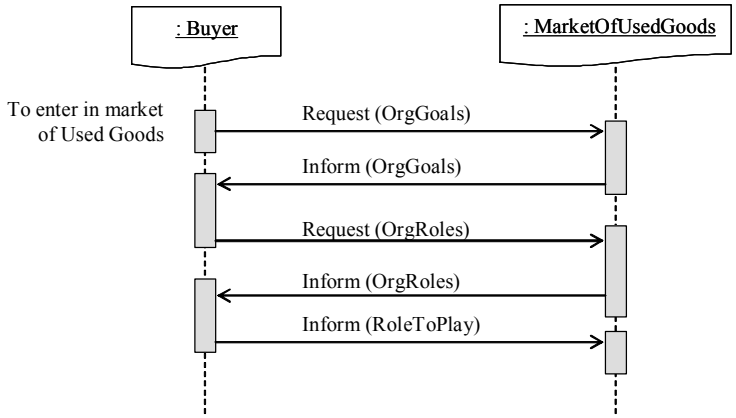

Figura 90 – Um protocolo definido pelos papéis de comprador e mercado de produtos usados.

A Figura 91 mostra um agente do usuário executando os protocolos descritos na Figura 90 e entrando em *Brand New bookstore*, uma instância da classe de organização *Second-hand bookstore*. Essa loja é um departamento de *Bookfinder*. Depois de interagir com a organização, o agente se compromete com o papel *buyer of second-hand books*, mas não pára de exercer o papel *buyer of books*. Esse processo é mostrado pelo estereótipo `<<role_commitment>>`. O agente entra na organização, faz o seu registro e procura um anúncio. Quando o comprador encontra um vendedor do livro que procura, negociará com ele. Caso o comprador compre o livro, o papel pode ser cancelado. O processo de cancelamento de um papel é ilustrado pelo estereótipo `<<role_cancel>>`. Depois disso, o agente está exercendo apenas o papel *buyer of books* na organização principal *Bookfinder*.

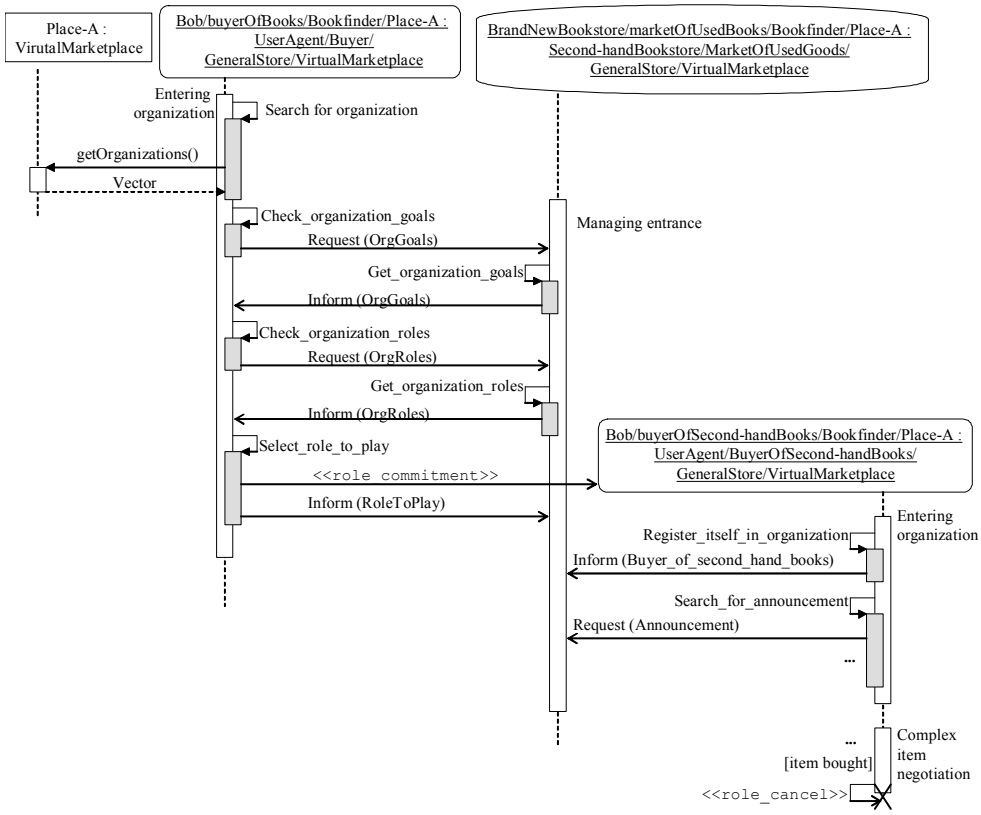

Figura 91 – Um agente de usuário entrando em uma organização, comprometendo-se com um novo papel e cancelando um papel.

Caso o comprador não encontre o livro (consulte Figura 92), ele pode se mover para outro ambiente para tentar encontrá-lo. A fim de se mover para outro ambiente, o agente precisa parar de exercer todos os papéis em *Place-A*. Portanto, o *buyer of books* desativa o papel *buyer of second-hand bookstore* (ilustrado pelo estereótipo `<<role_deactivate>>`). O agente procura outros ambientes no ambiente em que reside. Caso haja outros ambientes, o agente negocia sua entrada em uma organização desse novo ambiente. O agente de usuário pára de exercer o papel *buyer of books* em *Bookfinder* e se compromete com o papel *buyer of books* em *Alibris* (uma instância de *General Store* na instância do mercado virtual *Place-B*). O movimento do ambiente *Place-A* para o ambiente *Place-B* é mostrado pelo estereótipo `<<role_change>>`. A negociação entre o agente e a organização *Alibris* está incorporada no estereótipo `<<role_change>>`.

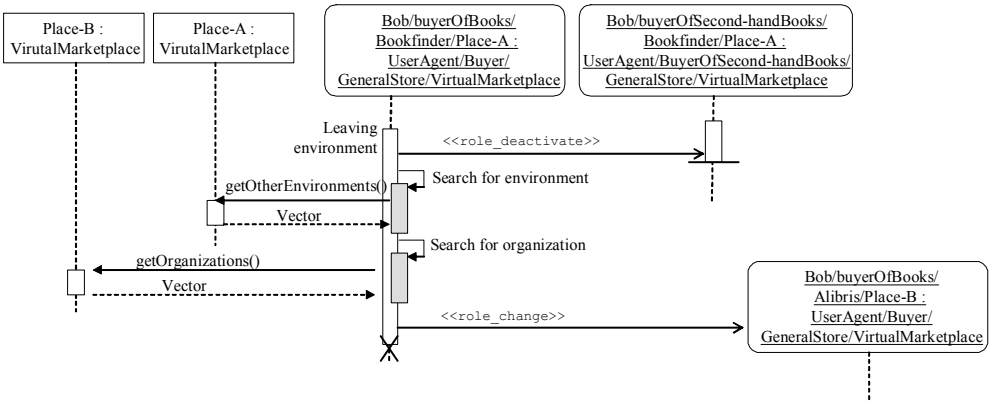

Figura 92 – Um agente de usuário se movendo de um ambiente para outro.

6.2.3. Transformando Modelos Estruturais de MAS-ML em Código Java

O transformador MAS-ML2Java foi usado com o objetivo de transformar os diagramas estruturais de MAS-ML da aplicação modelada na Seção 6.2 em código Java. Usando a ferramenta Together, foi aplicada uma engenharia reversa ao código Java a fim de gerar modelos de UML. Esta seção apresenta partes da descrição textual dos diagramas de MAS-ML e modelos de UML relacionados. A descrição textual completa da aplicação está no Apêndice II.

Para transformar diagramas estruturais em classes, esses diagramas devem ser descritos usando a gramática definida no Apêndice I. Depois disso, o transformador é aplicado ao texto, gerando uma transformação parcial. Na primeira vez em que o transformador é aplicado, as regras de entidade são usadas para criar os componentes definidos na arquitetura abstrata e para transformar as entidades da aplicação e suas propriedades em classes OO. Em seguida, o transformador é aplicado à transformação parcial criando o código Java. Na segunda vez em que ele é usado, as regras dos relacionamentos são aplicadas aos relacionamentos criando atributos e modificando as classes de entidade.

6.2.4. A Transformação do Ambiente

A Figura 93 descreve o ambiente *Virtual Marketplace* definido no exemplo. Como o ambiente é definido como um elemento passivo modelado como uma classe, a descrição textual do ambiente também é uma classe.

```
ENVIRONMENT
(
import java.util.*;
public class Virtual_Marketplace
```

```
{
    public Vector getItems ()
    {        return this.theItem;    }
    public void setItems (Item newItem)
    {        this.theItem.add (newItem);    }
    public Vector getOtherEnvironments ()
    {        return this.theVirtual_Marketplace;    }
    public void setOtherEnvironments (Environment newEnvironment)
    {        this.theVirtual_Marketplace.add (newEnvironment);   }
    public boolean checkAgentCanLeave (Agent newAgent)
    { ...  }
    ...
}
)
```

Figura 93 – A descrição textual da classe de ambiente Virtual Marketplace (parcial).

A classe *Virtual Marketplace* está associada a sete relacionamentos (consulte Figura 94). Seis relacionamentos descrevem os relacionamentos *inhabit* entre a classe *Virtual Marketplace* e as classes *General Store*, *User Agent*, *Store Agent*, *Second-hand Bookstore*, *Imported Bookstore* e *Item*. Quando o transformador é aplicado a esses relacionamentos, são criados dois relacionamentos *dependency* para cada um. Um é associado à classe *Virtual Marketplace* e o outro, à classe de entidade. O outro relacionamento representa o auto-relacionamento da classe *Virtual Marketplace* para ele mesma, conforme ilustrado na Figura 95.

```
RELATIONSHIP General_Store 1..*  INHABITS 1 Virtual_Marketplace
RELATIONSHIP User_Agent 0..* INHABITS 1 Virtual_Marketplace
RELATIONSHIP Store_Agent 0..* INHABITS 1 Virtual_Marketplace
RELATIONSHIP Second_hand_Bookstore 0..* INHABITS 1
                                        Virtual_Marketplace
RELATIONSHIP Imported_Bookstore 0..* INHABITS 1
                                        Virtual_Marketplace
RELATIONSHIP Item 0..* INHABITS 1 Virtual_Marketplace

RELATIONSHIP Virtual_Marketplace 0..* ASSOCIATED WITH 0..*
                                        Virtual_Marketplace
```

Figura 94 – A descrição textual dos relacionamentos da classe Virtual Marketplace.

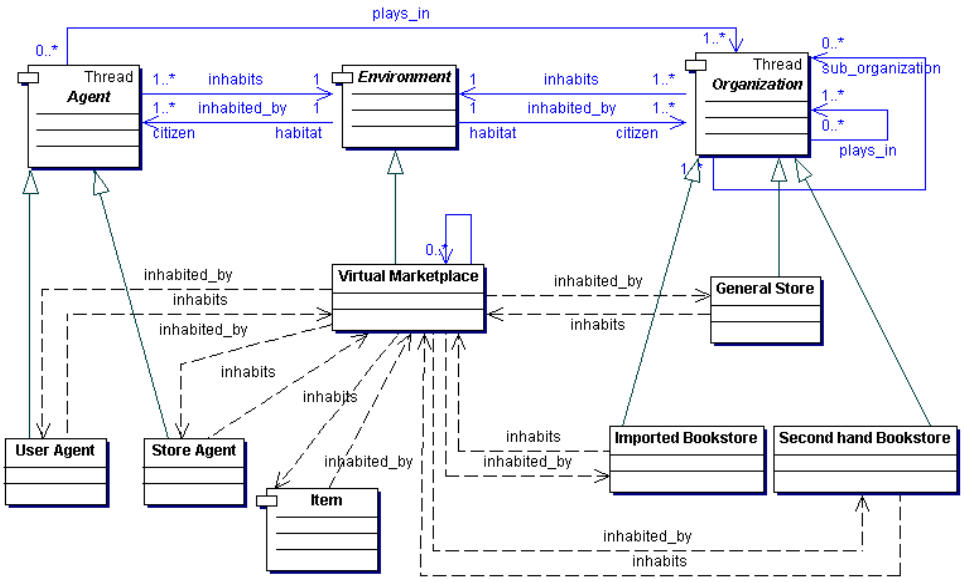

Figura 95 – A classe orientada a objetos Virtual Marketplace e seus relacionamentos.

6.2.5. As Transformações das Organizações

A fim de ilustrar a transformação de uma classe de organização em classes orientadas a objetos, a transformação da classe de organização *General Store* será usada como um exemplo. A Figura 96 mostra a descrição textual da classe de organização *General Store*.

```
ORGANIZATION
( General_Store
      GOAL ( "boolean" :  "management_of_sellers" = "true"
            RELATED TO PLAN "Creating_sellers"
      )
      BELIEF ("Vector" : "buyers" = null)
      PLAN ( Creating_sellers
                  COMPOSED OF ACTION "Create_seller"
                  COMPOSED OF ACTION "Inform_buyer_of_seller"
                  RELATED TO GOAL "management_of_sellers"
      )
      ...
      ACTION Create_seller
      ACTION Inform_buyer_of_seller
      ...
      AXIOM ("boolean" : "Send_information_about_sale" = "true")
)
```

Figura 96 – A descrição textual da classe de organização *General Store* (parcial).

Todas as classes de organização são transformadas em classes orientadas a objetos. A classe de organização *General Store* foi transformada em uma classe orientada a objetos *General Store*. Ao criar a classe OO, o método do construtor

da classe é implementado de acordo com as descrições da organização. Objetivos, crenças, planos, ações e axiomas são criados conforme mostrado na Figura 97.

```
import java.util.*;
public class General_Store extends Organization
{   ...
public General_Store (Environment theEnvironment, AgentRole
                        initialRole, Organization initialOrg)
{
      //creating a goal
      objectGoal = new LeafGoal ("boolean",
                                  "management_of_sellers", "true");
      this.goals.add (objectGoal);
      ...
      //creating a belief
      this.beliefs.add (new Belief ("boolean",
                               "management_of_sellers", "false"));
      ...
      //creating an action
      objectAction = new Create_seller ();
      this.actions.add (objectAction);
      ...
      //creating a plan and associating an action
      objectPlan = new Creating_sellers ();
      objectPlan.setOrganization (this);
      this.plans.add (objectPlan);
      actionAux = null;
      enumActions = this.actions.elements ();
      while (enumActions.hasMoreElements ())
      {
            actionAux = (Action) enumActions.nextElement ();
            if (actionAux.getClass ().getName ().
                                    equals ("Create_seller"))
            {     objectPlan.setAction (actionAux);     }
      }
      ...
      //creating an axiom
      this.axioms.add (new Axiom ("boolean",
                         "Send_information_about_sale", "true"));
}
public void run ()
{    // TO BE IMPLEMENTED    }
}
```

Figura 97 – O código Java para a classe *General Store* (parcial).

Ademais, os planos e as ações definidos pelas classes de organização são transformados em classes. Essas ações e planos concretos especializam as classes abstratas *Plan* e *Action* descritas na arquitetura abstrata. A Figura 98 ilustra duas ações e um plano relacionado à classe *General Store*. Essa figura também ilustra a classe abstrata *Organization* e as classes que representam suas propriedades. Essas classes são descritas pela arquitetura abstrata apresentada na Seção 5.4.1.

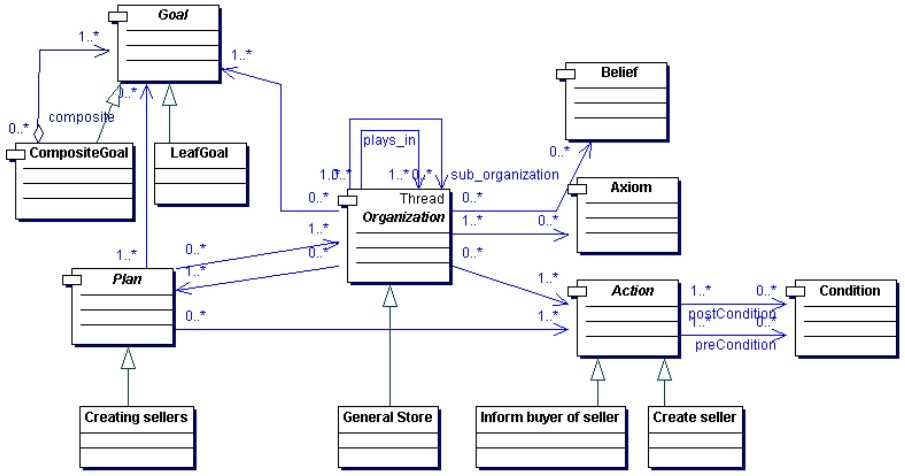

Figura 98 – Duas ações e um plano relacionado à classe *General Store*.

Os relacionamentos associados à classe de organização *General Store* são *inhabit*, *ownership* e *play* (consulte Figura 99). Conforme mencionado anteriormente, o relacionamento *inhabit* liga *General Store* e *Virtual Marketplace*. O relacionamento *ownership* liga *General Store* e os papéis definidos. São eles: papéis *Buyer*, *Seller*, *Market of Special Goods*, *Market* of *Used Goods*, *Desire* e *Offer*.

Esses papéis são exercidos em *General Store* por *User Agent*, *Store Agent*, *Book*, *Second-hand Bookstore* e *Imported Bookstore*, respectivamente. A transformação desses relacionamentos e da classe de organização *General Store* em atributos e em uma classe orientada a objetos está ilustrada na Figura 100.

```
RELATIONSHIP General_Store 1..*  INHABITS 1 Virtual_Marketplace

RELATIONSHIP General_Store 1 OWNS 0..* Buyer
RELATIONSHIP General_Store 1 OWNS  0..* Seller
RELATIONSHIP General_Store 1 OWNS  0..* Market_of_Special_Goods
RELATIONSHIP General_Store 1 OWNS  0..* Market_of_Used_Goods
RELATIONSHIP General_Store 1 OWNS  0..* Desire
RELATIONSHIP General_Store 1 OWNS  0..* Offer

RELATIONSHIP User_Agent 1 PLAYS  0..* Buyer IN General_Store
RELATIONSHIP Store_Agent 1 PLAYS  0..* Seller IN General_Store
RELATIONSHIP Book 1 PLAYS  0..* Desire IN General_Store
RELATIONSHIP Book 1 PLAYS  0..* Offer IN General_Store
RELATIONSHIP Second_hand_Bookstore 1 PLAYS  0..*
                      Market_of_Used_Goods IN General_Store
RELATIONSHIP Imported_Bookstore 1 PLAYS  0..*
                      Market_of_Special_Goods IN General_Store
```

Figura 99 – A descrição textual dos relacionamentos da classe de organização *General Store*.

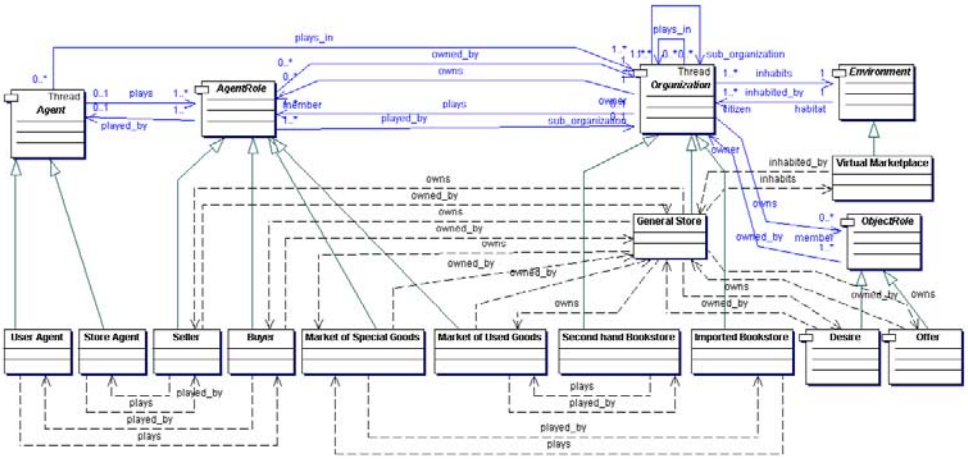

Figura 100 – A classe orientada a objetos *General Store* e seus relacionamentos.

6.2.6.
As Transformações dos Agentes

A transformação dos agentes será demonstrada usando a classe *User Agent*. A descrição textual de *User Agent e*stá ilustrada na Figura 101.

```
AGENT
( User_Agent
      GOAL ( "boolean" :  "itemBought" = "true"
            SUBGOAL (
                  GOAL ( "boolean" :  "searchSeller" = "true"
                        RELATED TO PLAN "Searching_seller"
                  )
                  GOAL ( "boolean" :  "negotiate" = "true"
                        RELATED TO PLAN "Buying_item"
                        RELATED TO PLAN "Complex_Buying_item"
                  )
                  GOAL ( "boolean" :  "enterOrganization" = "true"
                        RELATED TO PLAN "Entering_organization"
                        RELATED TO PLAN "Leaving_environment"
                  )
            )
      )
      ...
      BELIEF ("Item" : "item" = "null")
      ...
      PLAN ( Searching_seller
                  COMPOSED OF ACTION "Search_for_seller"
                  RELATED TO GOAL "searchSeller"
      )
      ...
      ACTION Search_for_seller
            POSTCONDITION ( "boolean" : "seller" = "true")
      ...
)
```

Figura 101 – A descrição textual da classe *User Agent* (parcial).

Semelhante ao que ocorre com as classes de organização, uma classe orientada a objetos é criada a fim de representar uma classe do agente. Uma

classe, chamada *User Agent*, foi criada para representar *User Agent*. O método do construtor da classe também foi implementado e novas classes foram criadas a fim de representar as ações e os planos definidos pela classe do agente, conforme demonstrado na Figura 102.

```
public class User_Agent extends Agent
{ ...
public User_Agent (Environment theEnvironment, Organization
initialOrg, AgentRole initialRole)
{     ...
      this.theEnvironment = theEnvironment;
      this.theEnvironment.setAgent (this);
      setRoleBeing*Play*ed (initialRole, initialOrg);
      ...
      //creating a goal and associating with a plan
      objectGoal = new LeafGoal ("Boolean", "itemBought", "true");
      this.goals.add (objectGoal);
      objectGoal.setPlan ("Searching_seller");
      ...
      //creating a belief
      this.beliefs.add (new Belief ("Item", "item", "null"));
      ...
      //creating an action and its post-condition
      objectAction = new Search_for_seller ();
      this.actions.add (objectAction);
      objectCondition = new Condition ("Boolean", "seller",
"true");
      objectAction.setPostCondition (objectCondition);
      ...
      //creating a plan
      objectPlan = new Searching_seller ();
      this.plans.add (objectPlan);
      ...
      //associating an action with the plan
      actionAux = null;
      enumActions = this.actions.elements ();
      while (enumActions.hasMoreElements ())
      {
            actionAux = (Action) enumActions.nextElement ();
            if    (actionAux.getClass    ().getName    ().equals
("Search_for_seller"))
            {     objectPlan.setAction (actionAux);       }
      }
      //associating a goal with the plan
      goalAuxLeaf = null;
      enumGoals = this.leafGoals.elements ();
      while (enumGoals.hasMoreElements ())
      {
            goalAuxLeaf = (LeafGoal) enumGoals.nextElement ();
            if (goalAuxLeaf.getName ().equals ("itemBought"))
            {     objectPlan.setGoal (goalAuxLeaf);        }
      }
}
public void run ()
{
      // TO BE IMPLEMENTED
}
}
```

Figura 102 – O código Java da classe *User Agent* (parcial).

Na Figura 103, dois planos e ações concretos definidos pela classe *User Agent* são ilustrados juntos com as classes definidas na arquitetura abstrata.

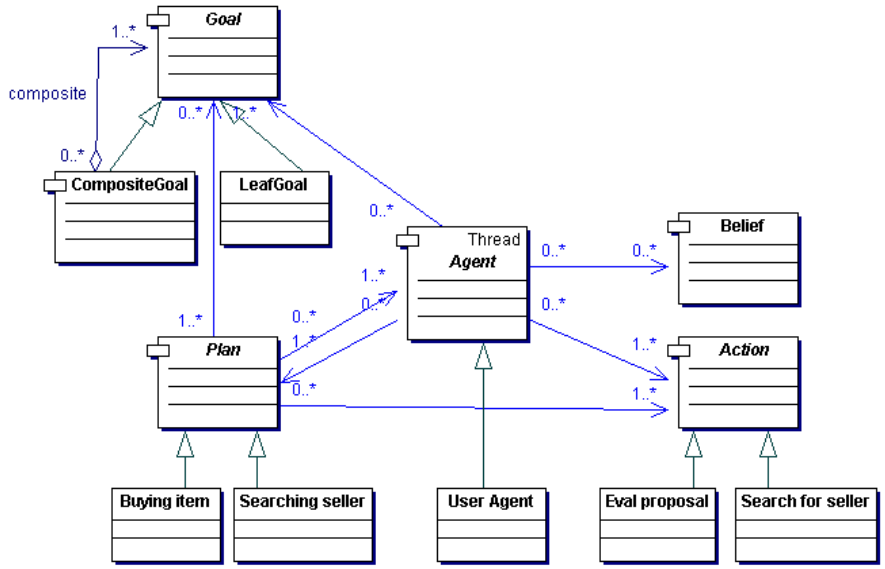

Figura 103 – Duas ações e dois planos relacionados à classe *User Agent*.

A classe *User Agent* está ligada aos papéis *Buyer*, *Buyer of Second-hand Books*, *Seller of Second-hand Books* e *Buyer of Imported Books* pelo relacionamento *play*. Conforme já mencionado, *User Agent* também está ligada a *Virtual Marketplace* pelo relacionamento *inhabit* (consulte Figura 104). A classe *User Agent* é transformada em uma classe orientada a objetos e os relacionamentos relacionados a ela são transformados em atributos, conforme ilustrado na Figura 105.

```
RELATIONSHIP User_Agent 0..* INHABITS 1 Virtual_Marketplace

RELATIONSHIP User_Agent 1 PLAYS  0..* Buyer IN General_Store
RELATIONSHIP User_Agent 1 PLAYS  0..* Buyer_of_Second_hand_Books
                                      IN Second_hand_Bookstore
RELATIONSHIP User_Agent 1 PLAYS  0..* Seller_of_Second_hand_Books
                                      IN Second_hand_Bookstore
RELATIONSHIP User_Agent 1 PLAYS  0..* Buyer_of_Imported_Books
                                      IN Imported_Bookstore
```

Figura 104 – A descrição textual dos relacionamentos da classe *User Agent*.

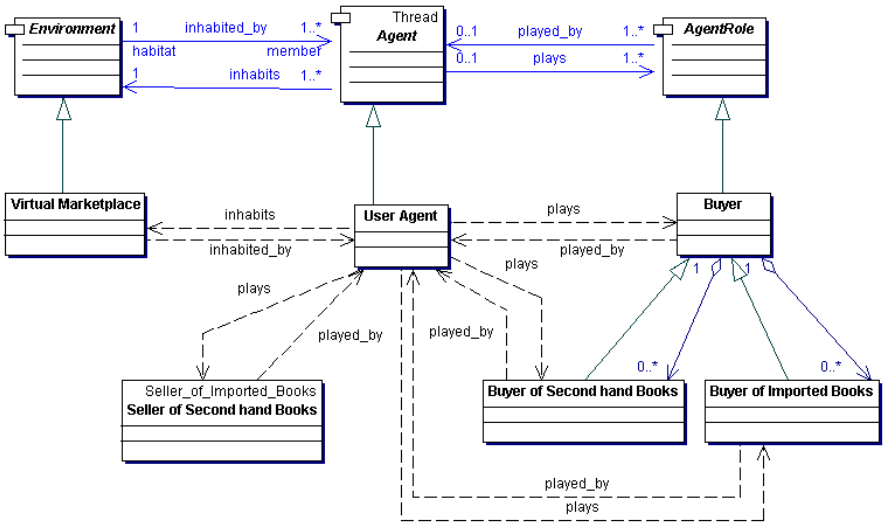

Figura 105 – A classe *User Agent* orientada a objetos e seus relacionamentos.

6.2.7. As Transformações dos Papéis dos Agentes

A fim de ilustrar a transformação de uma classe do papel do agente em classes orientadas a objetos, a transformação da classe do papel do agente *Buyer* será descrita. A Figura 106 mostra a descrição textual da classe do papel *Buyer*.

```
AGENTROLE
( Buyer
      ROLEGOAL ("boolean" :  "itemBought" = "true"
      )
      DUTY "Search_for_seller"
      ...
      RIGHT "Send_answer_proposal_reject"
      ...
      PROTOCOL SimpleNegotiation
            MESSAGE
            (LABEL "Request"
                  CONTENT "ItemDescription"
                  SENDER "Buyer"
                  RECEIVER "Seller"
            )
            MESSAGE
            (LABEL "Proposal"
                  CONTENT "ItemDescription"
                  SENDER "Seller"
                  RECEIVER "Buyer"
            )
      ...
)
```

Figura 106 – A descrição textual da classe *Buyer* (parcial).

Todas as classes de papel do agente são transformadas em classes orientadas a objetos. A classe do papel do agente *Buyer* foi transformada em uma classe orientada a objetos *Buyer*. Ao criar a classe OO, o método do construtor da classe

é implementado de acordo com as descrições do papel do agente. Objetivos, crenças, deveres, direitos e protocolos são criados conforme mostrado na Figura 107.

```
public class Buyer extends AgentRole
{
    public Buyer (Organization owner)
    {
        ...
        //creating a goal
        objectGoal = new LeafGoal ("boolean", "itemBought",
"true");
        this.goals.add (objectGoal);
        ...
        //creating a duty
        this.duties.add (new Duty ("Search_for_seller"));
        ...
        //creating a right
        this.rights.add (new Right
("Send_answer_proposal_accept"));
        ...
        //creating a protocol
        objectProtocol = new SimpleNegotiation ();
        this.protocols.add (objectProtocol);
        ...
        //creating a message
        objectMessage = new Message ("Request", "ItemDescription",
"Buyer", "Seller");
        ...
        //associating a protocol with a message
        objectProtocol.setMessage (objectMessage);
        ...
    }
}
```

Figura 107 – O código Java da classe *Buyer* (parcial).

Ademais, os protocolos definidos pelas classes de papel do agente são transformados em classes. Esses protocolos concretos especializam a classe abstrata *Protocol* descrita na arquitetura abstrata. A Figura 108 ilustra um protocolo relacionado à classe *Buyer* e as classes definidas na arquitetura abstrata.

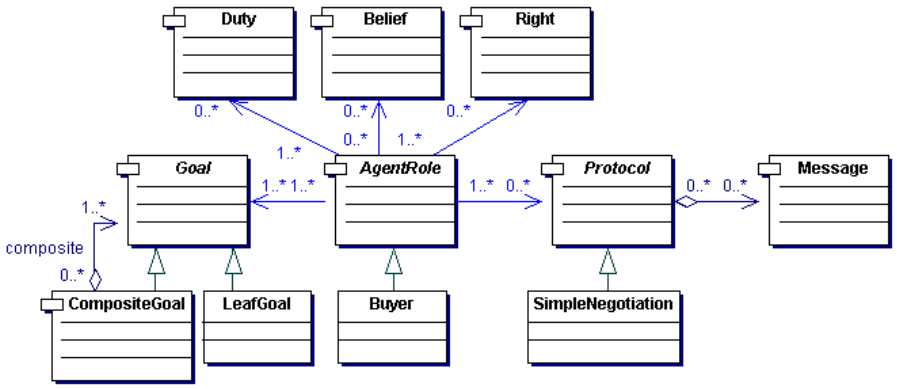

Figura 108 – Um protocolo relacionado à classe *Buyer*.

Os relacionamentos referentes à classe *Buyer* são os relacionamentos *ownership*, *play*, *association* e *aggregation*. Conforme mencionado anteriormente, o relacionamento *ownership* liga as classes *Buyer* e *General Store*, e o relacionamento *play* liga a classe *Buyer* e *User Agent*.

A classe *Buyer* é associada à classe do papel *Desire*, com as classes de papel do agente *Seller*, *Market of Special Goods* e *Market* of *Used Goods*. A classe *Buyer* agrega *Buyer of Second hand books* e *Buyer of Imported Books* (consulte Figura 109). A classe do papel do agente *Buyer* é transformada em uma classe orientada a objetos, e os relacionamentos referentes a ela são transformados em atributos, conforme ilustrado na Figura 110.

```
RELATIONSHIP General_Store 1 OWNS 0..* Buyer

RELATIONSHIP User_Agent 1 PLAYS  0..* Buyer IN General_Store`

RELATIONSHIP Buyer 1 ASSOCIATED WITH 0..* Desire
RELATIONSHIP Seller 0..* ASSOCIATED WITH 0..* Buyer

RELATIONSHIP Buyer 0..* ASSOCIATED WITH 0..* Market_of_Used_Goods
RELATIONSHIP Buyer 0..* ASSOCIATED WITH 0..*
                                          Market_of_Special_Goods
RELATIONSHIP Buyer 1 AGGREGATES 0..* Buyer_of_Second_hand_Books
RELATIONSHIP Buyer 1 AGGREGATES 0..* Buyer_of_Imported_Books
```

Figura 109 – Os relacionamentos da classe *Buyer*.

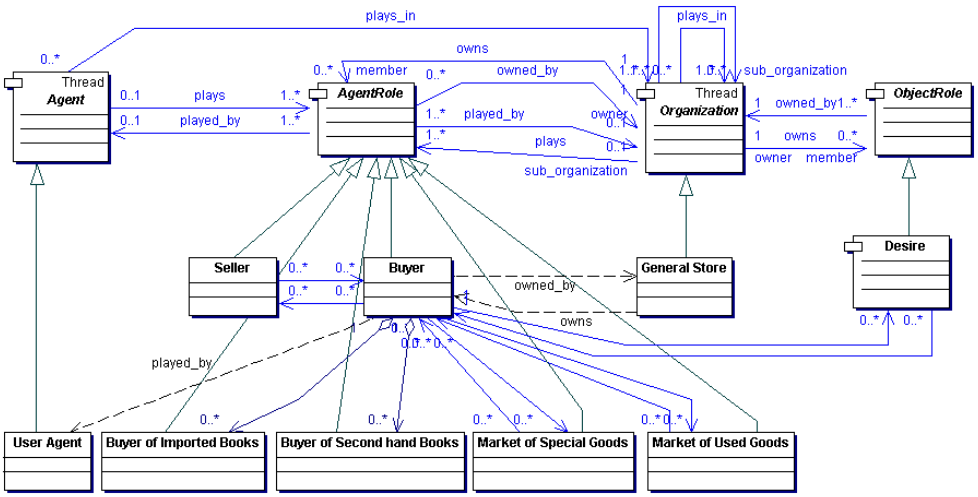

Figura 110 – A classe orientada a objetos *Buyer* e seus relacionamentos.

6.2.8. As Transformações dos Papéis de Objetos

A transformação do papel de objeto será descrita usando a classe do papel de objeto *Desire*. Os papéis de objeto são definidos em MAS-ML como tendo as

mesmas propriedades que uma classe. Portanto, todos os papéis de objeto são transformados em classes orientadas a objetos. A Figura 111 ilustra a classe do papel de objeto *Desire* definindo seus métodos, uma vez que não define qualquer atributo.

```
OBJECTROLE
( Desire (
      public int getPrice ()
      {        return object.getPrice();    }
      public void setTitle (String title)
      {        object.setTitle(title);    }
      ...
))
```

Figura 111 – A descrição textual da classe do papel de objeto *Desire* (parcial).

Os relacionamentos associados à classe do papel de objeto *Desire* são os relacionamentos *ownership*, *play*, *specialization* e *association*. Conforme mencionado anteriormente, o relacionamento *ownership* liga a classe *Desire* à classe *General Store*, e o relacionamento *association* liga a classe *Desire* à classe *Buyer*. O relacionamento *play* liga a classe *Desire* à classe *Book*. *Books* exercem instâncias do papel *Desire* em *General Store*. As classes de papel de objeto *Desire of Second-hand Books* e *Desire of Imported books* especializam a classe *Desire* (consulte Figura 112). O diagrama de UML relacionado à classe *Buyer* está ilustrado na Figura 113.

```
RELATIONSHIP General_Store 1 OWNS 0..* Desire
RELATIONSHIP Book 1 PLAYS 0..* Desire IN General_Store
RELATIONSHIP Desire_of_Second_hand_Books SPECIALIZES Desire
RELATIONSHIP Desire_of_Imported_Books SPECIALIZES Desire
RELATIONSHIP Buyer 1 ASSOCIATED WITH 0..* Desire
```

Figura 112 – Os relacionamentos da classe *Desire*.

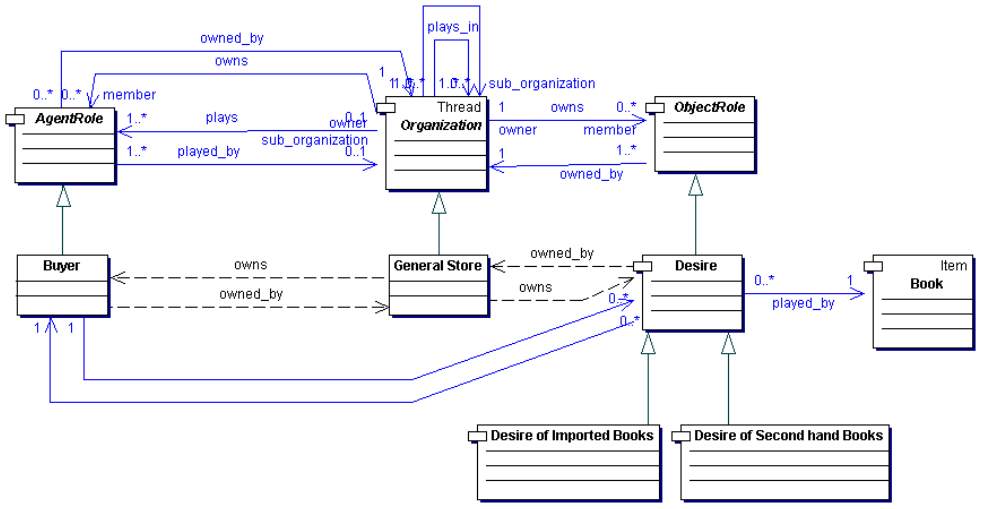

Figura 113 – A classe orientada a objetos *Desire* e seus relacionamentos.

6.2.9.
As Transformações de Classes

Com o objetivo de ilustrar a transformação de uma classe modelada em diagramas de MAS-ML em uma classe modelada em diagramas de UML, a transformação da classe *Book* será ilustrada. A Figura 114 mostra a descrição textual da classe *Book*.

```
public class Book extends Item
{
      protected String title = null;
      protected int price = null;
      ...
      public String getTitle ()
      {        return this.title;    }
      public int getPrice ()
      {        return this.price;    }
      ...
      public void setTitle (String title)
      {        this.title = title;    }
      public void setPrice (int price)
      {        this.price = price;    }
      ...
}
```

Figura 114 – A descrição textual da classe *Book* (parcial).

A classe *Book* está ligada às classes *Desire* e *Offer* pelo relacionamento *play* e às classes *Item*, *Imported Book* e *Second-hand Book* pelo relacionamento *specialization* (consulte Figura 115). A Figura 116 ilustra o diagrama de UML da classe *Book* e seus relacionamento.

```
RELATIONSHIP Book 1 PLAYS  0..* Desire IN General_Store
RELATIONSHIP Book 1 PLAYS  0..* Offer IN General_Store
RELATIONSHIP Book SPECIALIZES Item
RELATIONSHIP Imported_Book SPECIALIZES Book
RELATIONSHIP Second_hand_Book SPECIALIZES Book
```

Figura 115 – Os relacionamentos da classe *Book*.

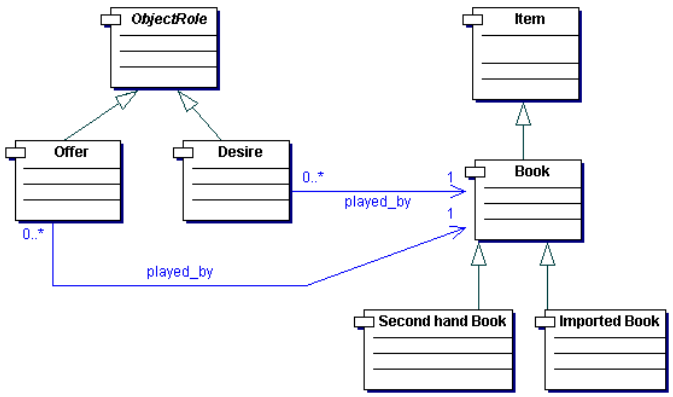

Figura 116 – A classe orientada a objetos *Book* e seus relacionamentos.

Na Figura 117, todas as entidades modeladas no exemplo do mercado virtual estão representadas. Alguns relacionamentos foram omitidos a fim de simplificar a figura e torná-la mais legível.

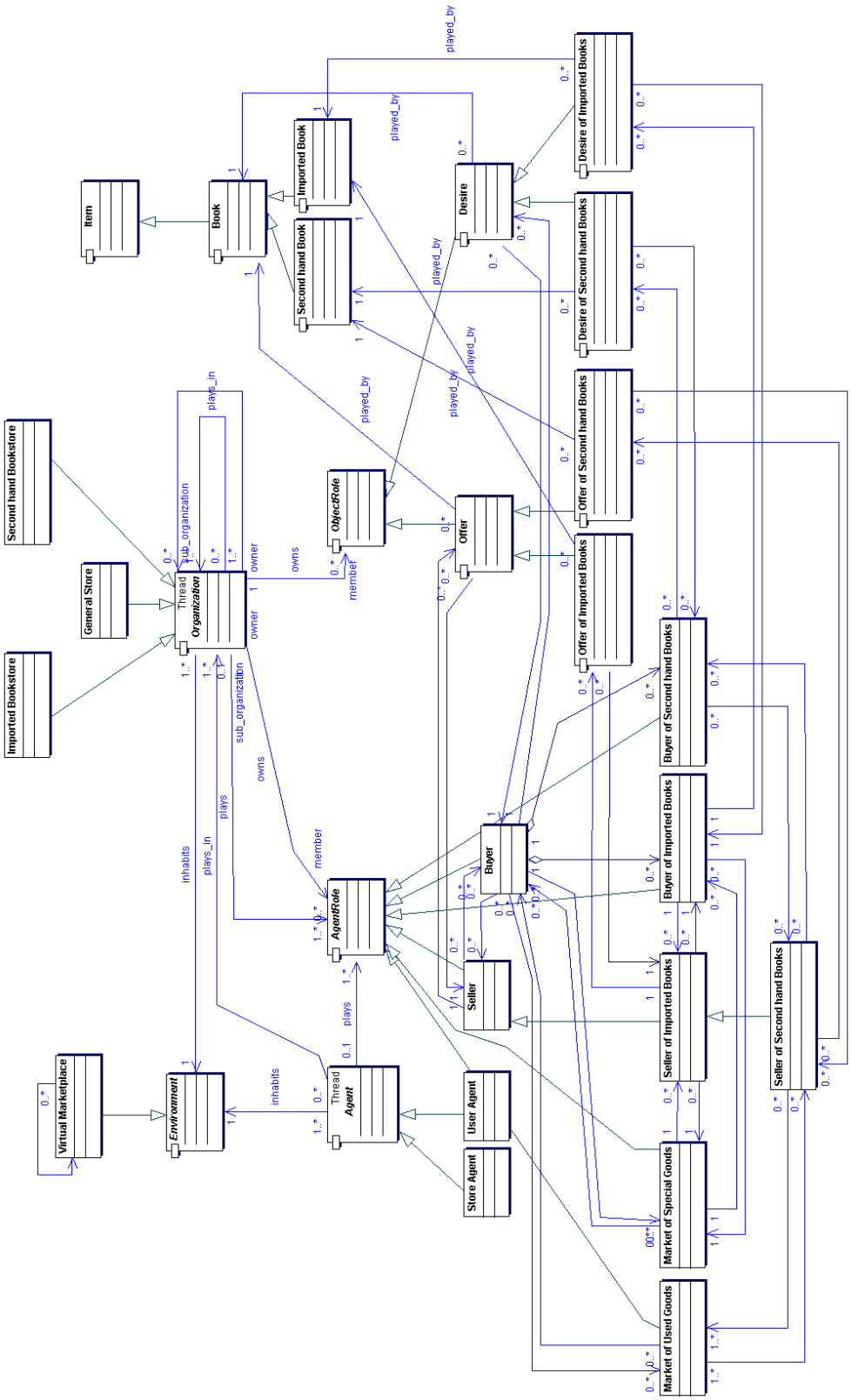

Figura 117 – Um diagrama de classes de UML ilustrando todas as entidades definidas no exemplo.

6.3. Avaliação da Linguagem MAS-ML

A fim de avaliar a linguagem MAS-ML, foram modeladas várias situações de modelagem difíceis e/ou impossíveis de representar nas linguagens de modelagem para SMAs existentes. Com MAS-ML, foi possível modelar os aspectos estruturais de papéis, organizações e ambientes. Suas propriedades e seus relacionamentos são modelados em diagramas de papel e organização. Foi possível descrever os papéis definidos por organizações e os agentes que exercem esses papéis, conforme ilustrado na Figura 63. Esse diagrama também modela a classe de ambiente na qual residem agentes, organizações e objetos.

Os aspectos dinâmicos de organizações e ambientes foram modelados usando o diagrama de seqüência definido por MAS-ML. Esse diagrama, ilustrado na Figura 91, apresenta um agente que está interagindo com uma organização e está entrando nela comprometendo-se com um novo papel. Esse compromisso indica a criação de uma instância de papel e sua associação com o agente. Depois de entrar na organização, o diagrama mostra o agente exercendo dois papéis em duas organizações diferentes. O diagrama também ilustra um agente que está interagindo com um ambiente e o agente que está cancelando um de seus papéis. Além disso, ele modela a execução interna de agentes e organizações enquanto ilustra os planos e as ações executadas por eles.

A Figura 92 ilustra um agente se movendo de um ambiente para outro. O diagrama ilustra um agente que está interagindo com um ambiente a fim de conhecer outros ambientes e um agente que está mudando seu papel. Os diagramas apresentados na Figura 91 e Figura 92 ilustram como as características de paralelismo e distribuição de SMAs podem ser modeladas usando MAS-ML.

AUML, a proposta de transformação gráfica, e AOR não definem ambientes e, portanto, não é possível modelar agentes interagindo com ambientes e agentes se movendo de um ambiente para outro. Usando Tropos, Prometheus e MaSE, também foi possível modelar ambientes.

A proposta de transformação gráfica, Tropos e Prometheus não descrevem organizações. Apesar de MESSAGE definir organizações, não é possível modelar nessa linguagem as interações entre organizações e outras entidades de SMAs porque elas são entidades virtuais. AUML identifica organizações e papéis como

entidades de SMAs, mas não define suas propriedades. Ademais, embora usando AUML seja possível modelar os papéis definidos em uma organização, as organizações não são modeladas em diagramas de seqüência e, portanto, não é possível modelar agentes interagindo com organizações a fim de exercer papéis.

Usando MAS-ML, os protocolos de diagrama de seqüência e interações entre agentes podem ser descritos conforme ilustrado na Figura 90 e Figura 89, respectivamente. Os protocolos e as interações entre agentes podem ser modelados usando AUML. Também é possível modelar agentes comprometidos com papéis, cancelando, mudando, ativando e desativando papéis. Entretanto, não é possível modelar a execução de planos e ações usando o diagrama de seqüência proposto por AUML. Como os diagramas de seqüência identificados por MESSAGE, Tropos e Prometheus baseiam-se em diagramas de seqüência de AUML, também é possível modelar protocolos e interações. Apesar de as interações entre agentes poderem ser modeladas usando AOR, os protocolos não podem.

Associado a MAS-ML, um transformador é definido a fim de refinar os modelos estruturais em código. Tropos e Prometheus mapeiam seu diagrama na plataforma de programação de agentes Jack também gerando um código esqueleto. Em AUML, são propostas diretrizes para gerar código dos diagramas de seqüência ao modelar protocolos com base em um exemplo. AOR, a proposta de transformação gráfica, MESSAGE e MaSE não definem qualquer diretriz a fim de gerar código a partir de modelos de alto nível.

6.4. Outros Exemplos Modelados Usando MAS-ML

Antes de modelar o sistema de mercado virtual descrito neste capítulo, também foram modelados mercados virtuais menos complexos usando MAS-ML. O primeiro mercado modelado usando MAS-ML não descrevia lojas especializadas em negociações de diferentes itens. Havia apenas uma organização no sistema que negociava qualquer tipo de item. Além disso, havia apenas um ambiente em que os agentes podiam residir. Os agentes não podiam se mover de uma organização para outra ou de um ambiente para outro. Foram adicionadas características mais complexas ao sistema a fim de ilustrar um agente se movendo

de um ambiente para outro e se movendo de uma organização para outra no mesmo ambiente.

Outras duas aplicações de SMAs de benchmark foram modeladas usando MAS-ML. Um dos exemplos modelados foi um sistema de revisão e envio de trabalhos baseados na Web (DeLoach, 2002; Zambonelli et al., 2001) usado por comissões de conferências. O sistema oferece suporte a diferentes atividades: o envio de trabalhos, a atribuição de um revisor, o envio do revisor, a notificação da aceitação e recusa e outros. Apesar de esse tipo de sistema usar algoritmos complexos para, por exemplo, atribuir trabalhos a revisores, modelar um sistema assim como um SMA foi menos complexo do que modelar o mercado virtual. O envio de trabalhos e o sistema de revisão foram modelados como uma organização em que agentes não se movem de uma organização para outra ou de um ambiente para outro. Além disso, apesar de o mesmo agente poder exercer três papéis completamente diferentes (autor, revisor e chair), não foi difícil modelar suas características dinâmicas. O agente não muda seus papéis ou se compromete com outro papel a fim de alcançar os objetivos do papel original. Portanto, para modelar a execução de um papel, não foi necessário nenhum dos estereótipos propostos associados a papéis. Conforme já visto, no mercado virtual, um comprador pode se comprometer com o papel de comprador de livros usados a fim de comprar um livro.

O sistema de revisão e envio de trabalhos enfatiza o uso de papéis de objeto. Diferentes visões do trabalho podem ser modeladas. Os revisores podem receber os atributos do trabalho e podem atribuir uma revisão do trabalho, mas não podem definir os atributos. Por outro lado, o autor pode definir os atributos do trabalho, mas não pode alterar nenhuma revisão.

O outro exemplo modelado usando MAS-ML é um sistema de gerenciamento da cadeia de suprimento simples (Huget, 2002b; Fox et al., 2000). Esse sistema envolve o recebimento de um pedido, negociando o preço e a data da entrega, comprando matéria-prima, produzindo o produto e finalmente entregando-o. O sistema de gerenciamento da cadeia de suprimento considera três tipos de atores: (i) os clientes que fazem o pedido, (ii) a empresa que produz e faz a entrega e (iii) os fornecedores da matéria-prima.

O sistema é composto por uma organização principal que é a empresa, os agentes de usuário que representam os clientes e os fornecedores e os agentes

internos que trabalham em nome da empresa. A organização principal define os papéis de cliente e fornecedor exercidos pelos agentes de usuário. Ademais, ela define vários papéis como os papéis de logística, entregador e trabalhador que serão exercidos por agentes internos. A diferença importante entre esse sistema e os demais são os diversos papéis que podem ser exercidos por agentes internos. Cada agente interno pode exercer diferentes papéis ao mesmo tempo. Além disso, eles trocam com freqüência seus papéis ao parar de exercer um papel e se comprometer com outro. Isso ocorre para otimizar a execução do sistema. Por exemplo, se houver muitos produtos a serem produzidos e poucos pedidos, os agentes que estiverem exercendo o papel de logística podem ser realocados comprometendo-se com o papel de trabalhador. Os diagramas de seqüência que modelam esse sistema usam muitos estereótipos de mensagens.

6.5. Discussão

Para exemplificar o uso da linguagem MAS-ML e para demonstrar sua utilidade, foi modelado um sistema multiagentes usando os diagramas estruturais e dinâmicos de MAS-ML. Seus diagramas estruturais foram transformados em diagramas de UML. As transformações mostraram a aplicação do transformador MAS-ML2Java.

Os diagramas estruturais de MAS-ML usados para modelar o sistema ilustram todas as entidades de MAS-ML e quase todos os relacionamentos de MAS-ML. A partir do conjunto de relacionamentos definidos por MAS-ML, o único relacionamento não usado durante a modelagem do exemplo foi o relacionamento *dependency*. Apesar de esse relacionamento não ter sido usado ao modelar o exemplo, seu uso é tão simples quanto o uso do relacionamento *association*, pois os dois estendem relacionamentos de UML bem conhecidos.

Ao modelar os aspectos dinâmicos do exemplo, foram criados quatro diagramas de seqüência, ilustrando as interações entre agentes, organizações e ambientes. Os diagramas de seqüência modelam os três aspectos dinâmicos independentes do domínio de alto nível, a seguir: (i) um agente entrando em uma organização, (ii) um agente saindo de uma organização e (iii) um agente se movendo de um ambiente para outro. Ademais, os diagramas ilustram agentes e organizações que estão enviando e recebendo mensagens, assim como chamadas

de métodos enquanto executa planos e ações. O uso de quatro estereótipos de mensagens também foi ilustrado no conjunto de diagramas de seqüência.

Os diagramas de seqüência da Figura 91 e Figura 92 também ilustram duas importantes propriedades de SMAs: paralelismo e distribuição. O paralelismo é caracterizado por um agente que executa em paralelo com uma organização. A Figura 91 ilustra o agente *Bob* que executa em paralelo com a organização *Brand New Bookstore*. A característica de distribuição é visualizada pelo movimento de um agente a partir de um ambiente para outro. A Figura 92 ilustra o agente *Bob* se movendo de um ambiente *Place-A* para o ambiente *Place-B*.

Depois de modelar os exemplos descritos nas Seções 6.2 e 6.4 e analisando o uso de MAS-ML, foi possível chegar a algumas considerações importantes. Usando os diagramas estruturais, é melhor aplicar a representação compactada dos elementos do diagrama (omitindo os compartimentos intermediários e inferior) e ocultar algumas propriedades. Modelar grandes sistemas (os sistemas que têm muitas entidades) mostrando todos os compartimentos de todos os elementos do diagrama poderia levar a diagramas extremamente grandes. A fim de modelar as propriedades das entidades, podem ser usadas duas abordagens. Cada elemento do diagrama pode ser modelado separadamente dos diagramas. Além disso, uma descrição textual das entidades, definida usando a gramática apresentada no Apêndice I, poderia ser usada.

Uma instância de entidade pode ser representada em diagramas de seqüência usando o *pathname* completo ou simples. Incentivamos o designer dos modelos de MAS-ML a usar o *pathname* completo quando realmente necessário. No Capítulo 4, descrevemos e justificamos quando o *pathname* completo poderia ser usado e quando o simples poderia substituí-lo. O uso dos *pathnames* simples facilita a leitura dos nomes das instâncias e dos nomes das classes. Os *pathnames* completos podem ficar longos, o que dificulta sua leitura.

A fim de definir melhor o exemplo de modelagem, os diagramas de seqüência podem ser usados para enfatizar diferentes características. Esse tipo de diagrama pode ser usado para modelar um protocolo, para modelar um conjunto de interações relacionadas, para modelar um plano ou uma ação. Ele pode, por exemplo, ilustrar a seqüência de ações executadas em um plano e o conjunto de mensagens enviadas enquanto uma ação é executada.

A execução de SMAs é inerentemente concorrente. Os agentes, por exemplo, podem exercer mais de um papel ao mesmo tempo. Isso significa que o agente possui pelo menos uma thread para representar um de seus papéis e que eles estão sendo executados concorrentemente. Ao modelar um agente que está executando mais de um papel em um diagrama de seqüência de MAS-ML, o designer não precisa se preocupar com a concorrência de seus papéis. O designer apenas escolhe usar uma seta com cabeça aberta ou com cabeça cheia para modelar agentes que estão enviando e recebendo mensagens indicando mensagens assíncronas e síncronas, respectivamente. Ao implementar esse sistema, o implementador precisa lidar com problemas de concorrência.

A aplicação de mercado virtual define um conjunto de 24 classes e 61 relacionamentos. Usando o transformador, foram geradas 103 classes de Java. Esse número é muito maior do que o número de classes de entidade. Do conjunto de classes geradas, 17 classes são relacionadas à arquitetura abstrata, 24 classes representam as entidades definidas na aplicação especializando as entidades abstratas *environment*, *agent*, *organization*, *agent role* e *object role*. Outras classes são especializações das propriedades abstratas *plan*, *action* e *protocol* definidas na arquitetura. Como o mercado virtual é um exemplo complexo, muitos planos, ações e protocolos foram definidos (consulte Apêndice II).

O mercado virtual foi implementado usando as classes geradas pelo transformador. Os métodos *run* definidos em agentes e organizações e os métodos *execute* definidos em planos, ações e protocolos foram implementados de acordo com os diagramas de seqüência modelados. O conjunto de classes, atributos e métodos gerados pelo transformador não foi modificado ao implementar o sistema. Entretanto, alguns outros atributos e métodos foram adicionados às classes.