UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE SISTEMAS DE INFORMAÇÃO

JOSÉ DIÓGENES FREITAS DE SOUSA

MODELAGEM DE ONTOLOGIAS UTILIZANDO A LINGUAGEM MAS-ML 2.0

QUIXADÁ

2013

JOSÉ DIÓGENES FREITAS DE SOUSA

MODELAGEM DE ONTOLOGIAS UTILIZANDO A LINGUAGEM MAS-ML 2.0

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de Concentração: Computação.

Orientador: Prof. Msc. Enyo José Tavares Gonçalves.

QUIXADÁ

2013

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

S696m Sousa, José Diógenes Freitas de
Modelagem de ontologias utilizando a linguagem MAS-ML 2.0 / José Diógenes Freitas de Sousa
– 2013.
97 p. : il. color., enc. ; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2013.
Orientação: Prof. Me. Enyo José Tavares Gonçalves
Área de concentração: Computação

1. Linguagem de programação (Computadores) 2. Inteligência artificial 3. Ontologia (Representação do conhecimento) I. Título.

CDD 005.133

JOSÉ DIÓGENES FREITAS DE SOUSA

MODELAGEM DE ONTOLOGIAS UTILIZANDO A LINGUAGEM MAS-ML 2.0

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: Computação.

Aprovado em: 17/12/2013.

BANCA EXAMINADORA

_____

Prof. Msc. Enyo José Tavares Gonçalves (Orientador)
Universidade Federal do Ceará – UFC

_____

Prof. Dr. Lincoln Souza Rocha
Universidade Federal do Ceará – UFC

_____

Prof. Dr. Marcos Antonio de Oliveira
Universidade Federal do Ceará – UFC

Dedico este trabalho à Deus e à minha família, por serem o motivo maior da minha existência e da minha perseverança.

AGRADECIMENTOS

À Deus por que sem Ele nada disso teria sido possível. Por tudo que sou, por ser minha fortaleza, meu socorro na angústia. Por tão amorosamente cuidar da minha vida e tão sabiamente guiar meus passos. Ao Senhor Jesus, por ser minha salvação, minha esperança, pela grande prova de seu amor por mim. Ao Espírito Santo, por me dar sabedoria e conhecimento de Deus, por me acompanhar em todos os momentos.

À toda minha família por todo amor, apoio e carinho, em especial: à minha mãe, Antonia Maria, por tudo que ela representa na minha vida, à cada um dos meus irmãos, à minha prima Ana Kelly por me ajudar sempre, e ao meu pai, José Diógenes. À minha amada Daniela, pelo seu amor, paciência e compreensão, por acreditar nos meus sonhos e por decidir vivê-los comigo.

Aos meus grandes amigos Leudivan, Giórgia, Maristela e Aparecida por todo apoio prestado desde o início desta caminhada. Vocês tiveram uma importância fundamental nessa minha vitória, obrigado e que Deus retribua tudo que vocês já fizeram por mim.

Ao meu orientador, professor Enyo Gonçalves, por todo apoio prestado, pela disponibilidade em ter me recebido e orientado, pacientemente, neste trabalho. Aos professores Marcos Oliveira e Lincoln Rocha, integrantes da banca examinadora deste trabalho, por todas as suas contribuições.

À todos os amigos que encontrei nesta Universidade, em especial, aos meus grandes amigos: Alexandra Valesca, Thalita Maria, Micaelly Priscila e Bruno Leandro por terem me acompanhado em todos os momentos desta caminhada, por todas as vezes em que me ajudaram e por serem os melhores amigos que já tive em toda minha vida.

Que Deus abençoe a vida de cada um de vocês!

“Há ocasiões em que optamos por acreditar em algo que normalmente seria considerado absolutamente irracional. Isso não significa que seja mesmo irracional, mas certamente não é racional. Talvez exista a suprarracionalidade: a razão além das definições normais dos fatos ou da lógica baseada em dados. Algo que só faz sentido se você puder ver uma imagem maior da realidade. Talvez seja aí que a fé se encaixe.”

(Willian P. Young, do livro *A Cabana*)

RESUMO

A modelagem de Sistemas Multiagentes (SMAs) requer que as características estruturais e comportamentais de elementos específicos da teoria de agentes sejam descritos e modelados para dar suporte à fase de desenvolvimento. A ontologia é uma técnica de representação do conhecimento que define um domínio a partir da descrição dos elementos e dos relacionamentos encontrados nesse domínio. Neste contexto, a ontologia aplicada aos sistemas multiagentes define uma terminologia que descreve os conceitos e como esses conceitos se relacionam dentro de um determinado ambiente do SMA, tornando possível que agentes, com diferentes características e propriedades, entendam os conceitos encontrados no domínio do sistema multiagente. O *framework* conceitual TAO e a linguagem de modelagem MAS-ML 2.0 fornecem os aspectos dinâmicos e estruturais de entidades que são normalmente encontradas na literatura de sistemas multiagentes. Entretanto, ambos não fornecem suporte à modelagem de ontologias. Assim, o presente trabalho apresenta uma extensão ao *framework* TAO e à linguagem MAS-ML 2.0 para que ambos possam fornecer os elementos (entidades e relacionamentos) necessários à modelagem de ontologias.

Palavras-chave: Sistemas Multiagentes. *Framework* Conceitual. Linguagem de Modelagem. Ontologia.

ABSTRACT

Modeling Multi-Agent Systems (MAS) requires structural and behavioral characteristics of specific elements agents' theory to be described and modeled to give support to the development phase. Ontology is a knowledge representation technique which defines a domain from the descriptions of elements and relationships found in this domain. In this context, ontology applied to multi-agent systems defines a terminology which describes the concepts and how they relate inside an environment of the MAS, making it possible for agents, with different characteristics and properties, to understand the concepts found in the MAS domain. The conceptual framework TAO and the MAS-ML 2.0 modeling language provide the dynamic and structural aspects of entities normally found in MAS literature. However, both do not support ontology modeling. Thus, this work presents an extension to the framework TAO and the language MAS-ML 2.0 so that both provide the necessary elements (entities and relationships) to modeling ontologies.

Key-words: Multi-Agent Systems. Conceptual Framework. Modeling Language. Ontology.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AML	Agent Modeling Language
AOR	Agent-Object-Relationship
AORML	Agent-Object-Relationship Modeling Language
AUML	Agent Unified Modeling Language
AVA	Ambiente Virtual de Aprendizagem
EAD	Educação à Distância
ES	Engenharia de Software
IA	Inteligência Artificial
KAoS	Knowledgeable Agent-oriented System
MAS-ML 2.0	Multi-Agent System Modeling Language Estendida
MAS-ML	Multi-Agent System Modeling Language
MDA	Model Driven Architecture
SMA	Sistema Multiagentes
SMAs	Sistemas Multiagentes
TAO	Taming Agents and Objects
UML	Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da pesquisa e desenvolvimento de sistemas inteligentes, o paradigma orientado a agentes tem conquistado destaque significativo, uma vez que este paradigma permite a criação de agentes inteligentes que são capazes de agir da melhor forma possível em uma determinada situação (RUSSEL; NORVIG, 2004). Ainda segundo Russell e Norvig (2004, p. 33), um agente é uma entidade "que pode ser considerada capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores".

Sistemas multiagentes (SMAs) são sistemas compostos por elementos computacionais, conhecidos como agentes de software, capazes de agir de forma autônoma e de interagir com outros agentes para realizar tarefas e alcançar objetivos (WOOLDRIDGE, 2009). Outros elementos que podem compor os SMAs são as organizações, ambientes, papéis de agente e de objetos, normas e ontologias.

A ontologia vêm sendo aplicadas em SMAs como meio para definir uma terminologia que descreva os conceitos e, como esses conceitos se relacionam dentro de um determinado ambiente do SMA. O termo *ontologia* teve origem na Filosofia e possui diferentes definições dentro de Ciência da Computação. Segundo Noy e McGuinness (2001), uma ontologia é uma descrição explícita e formal dos conceitos de um domínio, e constitui uma base de conhecimento que provê uma descrição dos conceitos, das propriedades de cada conceito e das relações entre cada conceito de um domínio.

Existem diversos tipos de ontologias e esses tipos variam de formalidade e especificação. Wooldridge (2009, p. 112) descreve os seguintes tipos: ontologia superior, ontologias de domínio e de aplicação; e as seguintes especificações: informal (especificação simples) e formal (especificação complexa). Além de descrever os domínios de um SMA, as ontologias podem ser usadas para aumentar a base de conhecimento de um agente e facilitar a comunicação do mesmo com outras entidades do sistema, a fim de que um agente possa executar corretamente ações que exercem impacto direto no alcance de seus objetivos.

Para atingir objetivos, um agente necessita conhecer e interagir com o ambiente em que está inserido, bem como interagir com as outras entidades que também habitam neste ambiente. Na interação de agentes podem surgir problemas de comunicação oriundos da falta de entendimento, por parte dos próprios agentes, de determinadas propriedades do ambiente e das entidades que estão interagindo. Neste contexto, dada a necessidade de desenvolver sistemas multiagentes, onde agentes com diferentes características e propriedades possam entender os conceitos encontrados em um ambiente e agir corretamente, seguindo o propósito

para o qual o SMA foi desenvolvido, uma ontologia pode ser utilizada para suprir a necessidade de representar a semântica dos conceitos e dos relacionamentos encontrados no domínio e nas entidades de um sistema multiagentes.

Este trabalho tem como foco a fase de projeto (modelagem) de sistemas multiagentes, considerando a modelagem de ontologias para agentes de software utilizando o *framework* conceitual TAO e a linguagem de modelagem MAS-ML, ambos criados por Silva (2004). Em Gonçalves (2009), a linguagem MAS-ML e o *framework* TAO ganharam uma segunda versão (MAS-ML 2.0) para que ambos possibilitassem a modelagem dos principais tipos de arquiteturas internas de agentes de software.

A escolha de MAS-ML 2.0 deu-se pelo fato desta linguagem se destacar de outras, em relação ao fornecimento dos aspectos dinâmicos e estruturais de entidades que são normalmente usadas e encontradas no domínio de sistemas multiagentes (SILVA, 2004, p. 2).

Baseado nessas considerações, o objetivo principal deste trabalho foi estender o *framework* conceitual TAO e a linguagem de modelagem MAS-ML 2.0, de forma que ambos pudessem fornecer os elementos (entidades e relacionamentos) necessários à modelagem de ontologias. Para isso, a extensão exigiu alterações nas diversas camadas que compõem a arquitetura do TAO e de MAS-ML, ou seja, (i) foi necessário estender o *framework* conceitual TAO para contemplar elementos de ontologia, (ii) estender a sintaxe abstrata da linguagem em conformidade com o *framework*, (iii) estender a sintaxe concreta e representar a extensão nos diagramas, e (iv) realizar um estudo de caso demonstrando o uso da extensão. Com isso, a linguagem MAS-ML além de possibilitar a modelagem das entidades de um SMA, poderá também possibilitar a modelagem das ontologias que serão utilizadas por estas entidades.

Este trabalho está organizado da seguinte forma: na seção 2 é apresentada a revisão bibliográfica, a qual contém a definição de agentes inteligentes e sistemas multiagentes, os conceitos, componentes e as classificações mais comuns de ontologias disponíveis na literatura, bem como a apresentação do *framework* conceitual TAO e da linguagem de modelagem MAS-ML 2.0; na seção 3 é apresentada a metodologia utilizada para a execução deste trabalho; na seção 4 apresentamos toda a extensão proposta ao *framework* TAO e à linguagem MAS-ML 2.0 para modelar ontologias e como ela foi trabalhada nas sintaxes abstrata e concreta desta linguagem; na seção 5 apresentamos um estudo de caso, no qual são identificadas e modeladas classes de ontologia para um sistema multiagente; e finalmente na seção 6 são apresentadas as considerações finais e os trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Esta seção apresenta alguns referenciais teóricos que fundamentaram a extensão proposta. Inicialmente, conceitos relacionados à SMAs são descritos; em seguida, serão abordados conceitos de ontologias. Por fim, o *framework* conceitual TAO e a linguagem de modelagem MAS-ML são descritos de forma sucinta.

2.1 Agentes Inteligentes e Sistemas Multiagentes

A ideia de um agente inteligente (ou racional) denota a criação de agentes que podem agir da melhor forma possível em uma determinada situação (RUSSEL; NORVIG, 2004, p. 29). Russel e Norvig (2004, p. 33) afirmam que "um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores".

De acordo com Russel e Norvig (2004, p. 35-39), os agentes possuem quatro grandes características: racionalidade – capacidade de executar a "ação que fará o agente obter maior sucesso", onisciência – o agente conhece "o resultado real de suas ações e pode agir de acordo com ele", aprendizado – o agente pode adquirir conhecimento a partir de suas percepções, e autonomia – capacidade que faz um agente agir de acordo com as "suas próprias percepções".

Sistemas multiagentes é uma subárea da Inteligência Artificial (IA) que trata da criação de agentes autônomos e inteligentes, e investiga o comportamento dos mesmos na resolução de um determinado problema (JENNINGS, 1996 *apud* GONÇALVES, 2009). Segundo Russel e Norvig (2009, p. 33) um sistema pode ser adequadamente considerado inteligente quando este é composto de agentes bem-sucedidos, ou seja, agentes que cumprem adequadamente o propósito para o qual foram construídos. Wooldridge (2009) define sistemas multiagentes como sistemas que são compostos de vários elementos computacionais (agentes) autônomos e capazes de interagir.

A comunicação e a colaboração são características desejáveis em sistemas multiagentes. A comunicação é uma propriedade que permite aos agentes trocarem informações sobre mudanças e novas descobertas em um ambiente. A colaboração permite que os agentes trabalhem juntos para alcançar um objetivo comum (COPPIN, 2012, p. 481). Desta forma, os SMAs podem ser classificados didaticamente em duas categorias: sistemas multiagentes cognitivos (cooperativos ou colaborativos) e sistemas multiagentes reativos (ou reflexivos). Os SMAs cognitivos são formados por agentes que colaboram entre si para

alcançar objetivos. Já os SMAs reativos são compostos por agentes que respondem a eventos no ambiente a partir de um conjunto de regras predeterminadas (COPPIN, 2012, p. 474).

Segundo Demazeau (1995), os sistemas multiagentes cognitivos possuem as seguintes características:

a) existe uma representação explícita do ambiente e dos demais agentes;
b) os agentes podem manter um histórico das interações e ações passadas para assim, serem capazes de planejar ações futuras;
c) a comunicação entre agentes cognitivos é realizada de modo direto, através do envio e recebimento de mensagens;
d) o mecanismo de controle de um agente cognitivo é deliberativo, ou seja, os agentes cognitivos raciocinam e decidem em conjunto, quais ações devem ser executadas, quais planos seguir e quais objetivos eles devem alcançar;
e) sistemas multiagentes cognitivos são compostos por agentes que se organizam em sociedade, como as organizações humanas.

Ainda segundo Demazeau (1995), os sistemas multiagentes reativos possuem as seguintes características:

a) não há representação explícita do ambiente e nem de outros agentes;
b) não há memória das ações, ou seja, os agentes reativos não guardam o histórico de ações que já foram realizadas;
c) organização etológica, ou seja, os agentes reativos são organizados a partir da observação de seu comportamento;
d) os agentes reativos possuem um comportamento simples: eles recebem um estímulo e respondem à esse estímulo;
e) a comunicação entre agentes reativos é feita através do ambiente pela propagação de sinais.

A comunicação entre agentes cognitivos e reativos é regida por uma linguagem de comunicação entre agentes e segue um protocolo de comunicação. Esse protocolo irá determinar o formato das mensagens que são enviadas e/ou recebidas pelos agentes. Uma das grandes vantagens da comunicação em sistemas multiagentes é que esta promove a interoperabilidade entre os agentes de software.

2.2 Ontologias

Devido ao aumento exponencial da informação, as técnicas de organização da informação tem conquistado destaque significativo nos últimos anos. Essas técnicas buscam

melhorar o tratamento da informação, atuando na sua seleção, no seu processamento, na sua recuperação e na sua disseminação. É neste contexto que as ontologias têm recebido atenção como uma técnica de organização da informação (ALMEIDA; BAX, 2003).

As ontologias podem ser usadas para representar o conhecimento de um domínio, através de regras que regulam a combinação de conceitos e relações encontrados neste domínio. Nesta subseção, são apresentados os conceitos de ontologia disponíveis na literatura de Filosofia e de Computação, os componentes básicos de uma ontologia, bem como as classificações mais comuns de ontologias.

2.2.1 Conceitos

O conceito original de *ontologia* vem da *metafísica* e esta, teve origem na Grécia Antiga. A metafísica consiste em uma parte da Filosofia "que investiga os fundamentos, os princípios e as causas de todas as coisas e o Ser íntimo de todas as coisas, indagando por que existem e por que são o que são." (CHAUI, 2009, p. 181). De acordo com Chauí (2009), a metafísica contemporânea é chamada de ontologia e procura superar as características da antiga metafísica.

Segundo Chauí (2009, p. 183), a ontologia é o "estudo [...] do Ser, dos entes ou das coisas tais como são [...], real e verdadeiramente, isto é, o estudo do Ser enquanto Ser." Ainda segundo Chauí (2009) a metafísica contemporânea (ontologia) possui as seguintes características:

a) investiga os diferentes modos como os entes (seres) existem;
b) investiga a essência (significado) desses entes;
c) investiga a relação necessária entre a existência e a essência dos entes;
d) descreve as estruturas que compõem o mundo e o nosso pensamento.

Em Computação, especialmente na Inteligência Artificial, a *ontologia* vem sendo utilizada como um método de representação do conhecimento cuja finalidade é representar as informações de um domínio através de classes e relacionamentos. Russel e Norvig (2004, p. 253) afirmam que a *ontologia* é "uma teoria específica sobre a natureza de ser ou existir", determinando os tipos de itens que existem em um determinando domínio. Ainda segundo Russel e Norvig (2004, p. 245), "um domínio é simplesmente alguma parte do mundo sobre a qual desejamos expressar algum conhecimento".

A estrutura de uma ontologia é definida a partir das características do domínio que se deseja representar e não segue um padrão, podendo variar de componentes (classes, relações, axiomas etc.) e classificações (ontologias de aplicação, domínio, tarefa etc.). Segundo

Almeida e Bax (2003, p. 8), para elaborar uma ontologia é necessário definir categorias para os elementos que existem em um mesmo domínio visando criar um catálogo desses elementos. Assim, a estrutura de uma ontologia é definida por componentes e classificações, os quais são apresentados nas próximas subseções.

2.2.2 Componentes

Não há na literatura um padrão que determine quais são os componentes básicos de uma ontologia. Desta forma, é possível encontrar diversas definições de componentes dadas por diferentes autores. Segundo Almeida e Bax (2003, p. 9), "as ontologias não apresentam sempre a mesma estrutura, mas existem características e componentes básicos comuns presentes em grande parte delas".

Para os autores Noy e McGuinness (2001) e Gruber (1996), os componentes básicos de uma ontologia são:

a) classes – representam os conceitos de um domínio e podem ser organizadas em taxonomias. Exemplo: considere o domínio *Computação* e que dentro desse domínio existem as seguintes classes (conceitos): *Algoritmo*, *Algoritmo de Busca* e *Algoritmo de Ordenação*;
b) relações – representam o tipo de relação que existe entre as classes (conceitos) de um domínio. Exemplo de uma relação entre classes: um *Algoritmo de Busca* (subclasse) é um tipo de *Algoritmo* (superclasse);
c) axiomas – são regras explícitas que restringem o uso dos conceitos de um domínio. Um exemplo de axioma é dizer que todo *Algoritmo de Busca* retorna um elemento quando este faz parte de uma lista de elementos;
d) instâncias – usadas para representar o conhecimento existente na ontologia. Exemplo: uma instância da classe *Algoritmo* é um conhecimento oferecido pela ontologia de *Computação*.

2.2.3 Classificações

A classificação de ontologias segue uma abordagem, já que existem diversos tipos de ontologias. Os autores Almeida e Bax (2003, p. 10) apresentam as seguintes abordagens, bem como os tipos de ontologia em cada abordagem:

a) quanto à função da ontologia – ontologias de domínio, ontologias de tarefa e ontologia geral.

b) quanto ao grau de formalismo da ontologia – ontologias altamente informais, ontologias semi-informais, ontologias semiformais e ontologias rigorosamente formais.
c) quanto à aplicação da ontologia – ontologias neutras, ontologias de especificação, ontologias de acesso comum a informação.
d) quanto à estrutura da ontologia – ontologias de alto nível, ontologias de domínio e ontologias de tarefa.
e) quanto ao conteúdo da ontologia – ontologias terminológicas, ontologias de informação, ontologias de modelagem do conhecimento, ontologias de aplicação, ontologias de domínio, ontologias genéricas e ontologias de representação.

Wooldridge (2009) classifica uma ontologia de acordo com o seu grau de formalismo (formal ou informal) e o tipo de informação que ela provê (conteúdo da ontologia). Os tipos definidos em Wooldridge (2009, p. 112) são:

a) ontologia superior – define os conceitos que são independentes do domínio. Este tipo de ontologia possui um alto nível de reusabilidade;
b) ontologia de domínio – define os conceitos apropriados para um domínio específico. Por exemplo, uma ontologia de domínio que representa a Computação deve prover a descrição de conceitos relacionados a área da Computação;
c) ontologia de aplicação – define os conceitos usados por uma aplicação específica.

Como visto, as ontologias podem ser classificadas de diversas maneiras. Também pode ser constatado em Almeida e Bax (2003, p. 09-11) que as ontologias podem ser utilizadas nas mais diversas áreas do conhecimento para os mais diversos propósitos, como: gestão do conhecimento, comércio eletrônico, processamento de linguagens naturais, recuperação da informação na web e projetos educacionais.

2.3 *Framework* TAO e a Linguagem de Modelagem MAS-ML 2.0

Embora muitos *frameworks* e linguagens de modelagem para sistemas multiagentes tenham sido propostos, havia a necessidade de criar um *framework* conceitual e uma linguagem de modelagem que contemplasse os aspectos dinâmicos e estruturais de entidades que são normalmente encontradas em sistemas multiagentes (SILVA, 2004, p. 2).

A partir dessa necessidade, o *framework* conceitual TAO e a linguagem de modelagem MAS-ML foram criados para descrever e modelar as entidades que são frequentemente encontradas em sistemas multiagentes. Por exemplo, antes de MAS-ML todas as propostas de linguagens de modelagem que estendem o metamodelo de UML (BOOCH; RUMBAUGH;

JACOBSON, 2005) descreviam a entidade *agente*, entretanto, entidades como *papel*, *organização* e *ambiente* não eram definidas em muitas delas. Uma visão mais detalhada do *framework* TAO e da linguagem MAS-ML é apresentada nas próximas subseções.

2.4 O *Framework* Conceitual TAO

O *framework* TAO (*Taming Agents and Objects*) foi criado por Silva (2004) com o objetivo de prover uma descrição detalhada das entidades e dos relacionamentos que compõem um sistema multiagentes, servindo de base para a construção da linguagem de modelagem MAS-ML. Desde então, este *framework* conceitual vem sendo estendido para incorporar novas características na construção de artefatos de modelagem. Por exemplo, Gonçalves (2009) inseriu no TAO e em MAS-ML, novas metaclasses e novos estereótipos para modelar os principais tipos de arquiteturas internas de um agente: agente reativo simples, reativo baseado em conhecimento, baseado em objetivo e agente baseado em utilidade.

De acordo com Silva (2004), o TAO incorpora a descrição dos aspectos estruturais (entidades e relacionamentos) e dos aspectos dinâmicos (interações entre as entidades) de sistemas multiagentes, os quais são apresentados a seguir.

2.4.1 Aspectos Estruturais

Os aspectos estruturais do *framework* TAO definem as seguintes entidades: agente, objeto, papel de agente, papel de objeto, organização e ambiente; e os seguintes relacionamentos: *inhabit*, *ownership*, *play*, *control*, *dependency*, *specialization*, *association* e *aggregation* (SILVA, 2004, p. 42).

2.4.1.1 Entidades

- *Agent* (Agente): é uma entidade que possui crenças (conhecimento do ambiente), objetivos (estados que o agente deseja atingir), planos (conjunto de ações) e ações (tarefas que um agente deve ou pode executar). Segundo Silva (2004, p.35), "os agentes podem interagir com outras entidades enviando e recebendo mensagens enquanto exercem um papel e executam um protocolo de interação".
- *Object* (Objeto): é uma entidade que possui atributos e métodos. Ao longo de seu ciclo de vida, um objeto executa ações (métodos) que alteram seu estado. Os objetos não são entidades racionais ou autônomas, ou seja, eles não podem escolher qual ação executar, apenas executam as ações que são solicitadas explicitamente (SILVA, 2004).

- *Agent Role* (Papel de Agente): é uma entidade que orienta o comportamento de um agente. Um papel de agente descreve o objetivo que o agente deve alcançar, as ações que o agente deve executar (deveres) e as ações que o agente pode executar (direitos). Além disso, um papel de agente descreve os protocolos que o agente deverá seguir ao exercer um papel e ao interagir com outras entidades do sistema.
- *Object Role* (Papel de Objeto): é uma entidade que orienta o comportamento de um objeto. Assim como um objeto, uma classe de papel de objeto também possui atributos e métodos, isso para fornecer informações adicionais ao objeto e para restringir o acesso as propriedades do objeto que está exercendo um papel de objeto.
- *Organization* (Organização): é uma entidade formada por um grupo de agentes e regida por um conjunto de axiomas (regras que os agentes e as suborganizações devem obedecer). Uma classe de organização possui um estado mental formado por: crenças, objetivos, planos, ações e axiomas; e definem papéis e relacionamentos.
- *Environment* (Ambiente): é uma entidade na qual residem agentes, objetos e organizações. Segundo Silva (2004, p. 37), "um ambiente pode ser uma entidade passiva, como um objeto, ou pode ser uma entidade ativa, como um agente que tenha características de agência como autonomia, adaptação e interação". Um ambiente passivo possui atributos, métodos e relacionamentos. Já um ambiente ativo, possui um estado mental (definido pelas crenças, objetivos, planos e ações) e os relacionamentos que há entre o ambiente e as outras entidades (SILVA, 2004, p. 37-38).

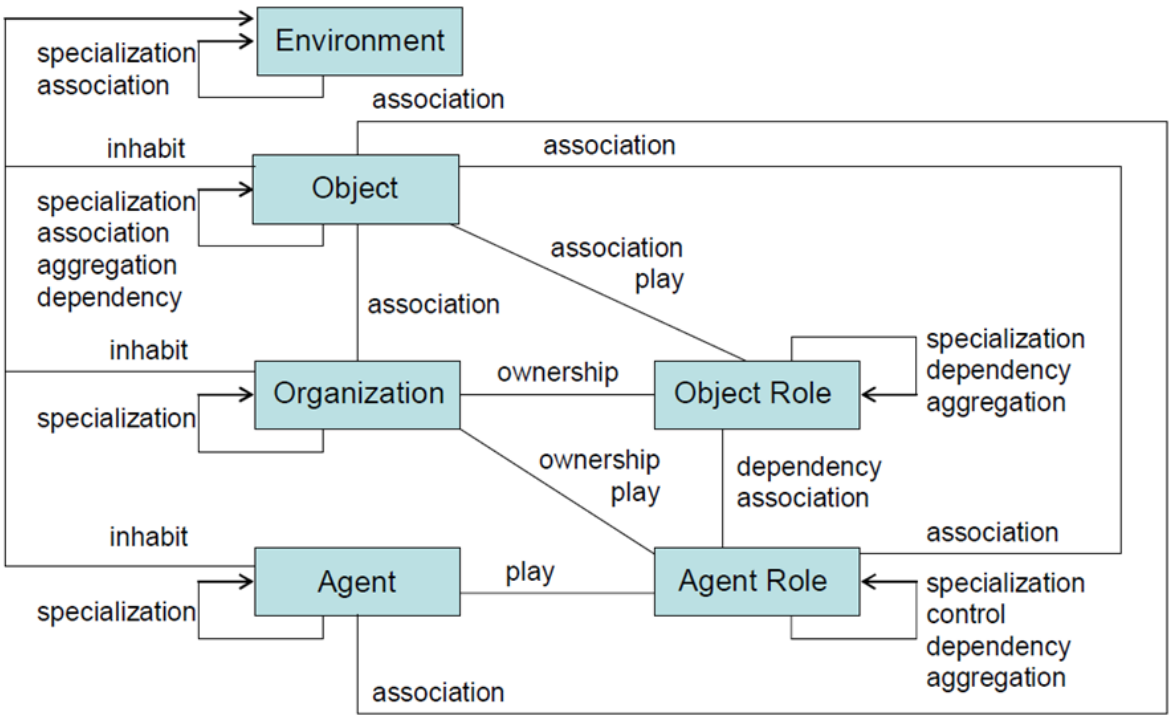
2.4.1.2 Relacionamentos

- *Inhabit* (Habita): este relacionamento é usado para especificar que a instância de uma entidade (agente, objeto ou organização) é criada e destruída em um ambiente, o qual é chamado de habitat. Estas entidades podem entrar e sair de um ambiente, respeitando suas permissões (SILVA, 2004).
- *Ownership* (Proprietário): este relacionamento é usado para especificar que uma entidade (o membro) está definida no escopo de outra entidade (o proprietário). Portanto, um membro deve obedecer a um conjunto de restrições globais definidas pelo seu proprietário e um membro não pode existir fora do escopo de seu proprietário. Esse tipo de relacionamento só pode ser aplicado entre organização e papel de agente, e organização e papel de objeto (SILVA, 2004).
- *Play* (Exerce): este relacionamento é usado para especificar que uma entidade está exercendo um papel. De acordo com Silva (2004), o relacionamento *play* pode ser

aplicado entre agente e papel de agente; objeto e papel de objeto; organização e papel de agente; e organização e papel de objeto.

- *Control* (Controle): este relacionamento é usado para especificar que uma entidade pode controlar outra. Assim, a entidade controlada deve fazer tudo que a entidade controladora solicitar. O relacionamento *control* só pode ser aplicado entre as instâncias de papel de agente (SILVA, 2004).
- *Dependency* (Dependência): este relacionamento é usado para especificar que uma entidade (o cliente) depende de outra (o fornecedor) para alcançar seus objetivos.
- *Specialization/Generalization* (Herança): este relacionamento é usado para especificar que uma subentidade está herdando as propriedades e os relacionamentos definidos em uma superentidade. Silva (2004) afirma que "as propriedades herdadas também podem ser redefinidas pela subentidade. [...] a subentidade também pode definir novas propriedades e novos relacionamentos".
- *Association* (Associação): este relacionamento é usado para especificar que uma entidade pode interagir com outra, por que ambas se conhecem. O relacionamento *association* pode ser aplicado entre todas as entidades descritas no *framework* TAO.
- *Aggregation* (Agregação): este relacionamento é usado para especificar que uma entidade pode agregar outra. Por meio do relacionamento *aggregation*, uma entidade pode utilizar as funções de outra para alcançar seus objetivos.

Figura 1 – As Entidades e os Relacionamentos Descritos no *Framework* TAO.



Fonte: Silva (2004, p. 46).

2.4.2 Aspectos Dinâmicos

"Os aspectos dinâmicos de uma entidade definem a execução da entidade em uma aplicação. Os aspectos dinâmicos caracterizam-se pela execução interna da entidade (intra-ações) e pelas interações com outras entidades." (SILVA, 2004, p. 46). Os agentes, organizações e ambientes proativos executam planos e ações, enquanto os objetos e os ambientes reativos executam métodos (SILVA, 2004, p. 46). A partir da interação com outras entidades e dos objetivos definidos, é que os agentes, as organizações e os ambientes proativos irão selecionar os planos que serão executados. A interação de um objeto com outras entidades é que vai definir quais métodos o objeto irá executar (SILVA, 2004, p. 46).

De acordo com Silva (2004, p. 46-47) podemos classificar os aspectos dinâmicos de sistemas multiagentes, descritos pelo TAO, em dois grupos principais:

a) aspectos dinâmicos dependentes do domínio – são responsáveis por descrever as intra-ações e interações entre as entidades de um domínio específico. Os relacionamentos *association*, *aggregation*, *dependency*, *control* e *specialization* são utilizados para definir as interações que são dependentes do domínio;
b) aspectos dinâmicos independentes do domínio – são responsáveis por descrever as intra-ações e interações entre as entidades independente do domínio. Os

relacionamentos *inhabit*, *ownership* e *play* são utilizados para definir as interações que são independentes do domínio.

O *framework* conceitual TAO provê a descrição de todas as entidades e de todos os relacionamentos que serviram de base para a construção da linguagem de modelagem MAS-ML, a qual é apresentada na próxima subseção.

2.5 A Linguagem de Modelagem MAS-ML 2.0

A linguagem de modelagem MAS-ML (*Multi-Agent System Modeling Language*) foi criada por Silva (2004) para incorporar a modelagem dos aspectos estruturais (entidades e relacionamentos) e dinâmicos (interação entre as entidades) descritos no *framework* TAO. MAS-ML é uma linguagem de modelagem que estende algumas propriedades (diagramas e relacionamentos) de UML (*Unified Modeling Language*) (BOOCH; RUMBAUGH; JACOBSON, 2005), e é formada por cinco diagramas, sendo três diagramas estruturais – classes, organização e papel; e dois diagramas dinâmicos – diagrama de sequência e de atividades.

A proposta de MAS-ML envolveu a extensão do diagrama de classes de UML – usado para representar os aspectos estruturais descritos no TAO; do diagrama de sequência – usado para representar a execução de planos e ações e modelar as intra-ações das entidades *agente*, *organização* e *ambiente*; e do diagrama de atividades – usado para modelar planos e ações, e os demais conceitos definidos pela linguagem MAS-ML. Em seu trabalho, Silva (2004) estendeu o metamodelo da linguagem UML, definindo novas metaclasses e novos estereótipos para representar as entidades e os relacionamentos descritos no *framework* conceitual TAO.

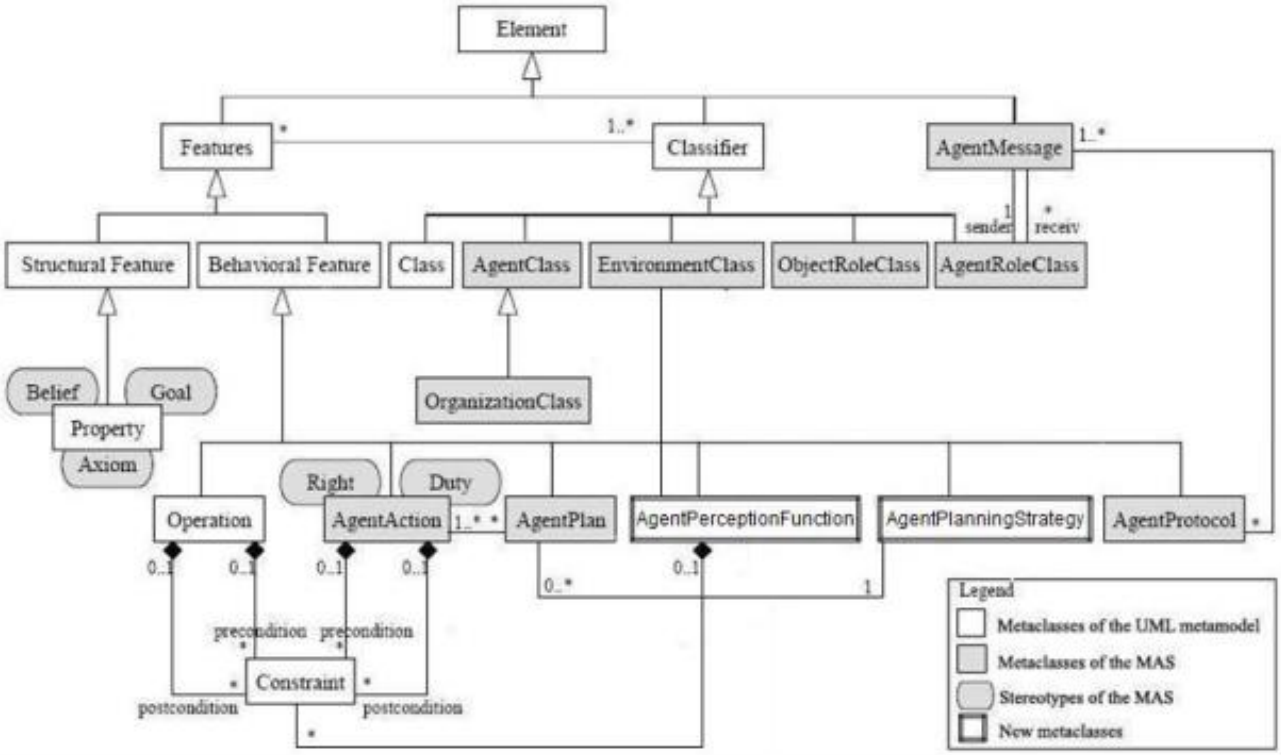
A linguagem MAS-ML 2.0 estende MAS-ML e é a versão resultante do trabalho de Gonçalves (2009), o qual introduziu no metamodelo de MAS-ML, 2 (duas) novas metaclasses e 6 (seis) novos estereótipos para que a linguagem fosse capaz de modelar adequadamente os principais tipos de arquiteturas internas de um agente de software.

Freire *et al.* (2012) também realizou uma extensão na linguagem MAS-ML, definindo 11 (onze) novas metaclasses, novos estereótipos e um diagrama estático (diagrama de normas) para que MAS-ML fosse capaz de fornecer os elementos necessários à modelagem de agentes normativos. Este trabalho será desenvolvido sob o metamodelo da linguagem MAS-ML 2.0, o qual é apresentado a seguir.

2.5.1 O Metamodelo de MAS-ML 2.0

O metamodelo da linguagem MAS-ML 2.0 é formado pelas metaclasses e pelos estereótipos criados por Silva (2004) para representar as entidades e os relacionamentos definidos no *framework* TAO – dando origem a linguagem MAS-ML, bem como pelas metaclasses e pelos estereótipos criados por Gonçalves (2009) – originando a segunda versão da linguagem, a MAS-ML 2.0. Na figura a seguir, são mostradas as metaclasses e os estereótipos, bem como as associações entre eles, que juntos, formam o metamodelo da linguagem. Nas próximas subseções, descrevemos essas metaclasses e estereótipos, a fim de facilitar a compreensão do metamodelo.

Figura 2 – O Metamodelo da Linguagem MAS-ML 2.0.



Fonte: Gonçalves (2009, p. 46).

2.5.1.1 As Metaclasses

Segundo Booch, Rumbaugh e Jacobson (2005, p. 458), uma metaclasse é "uma classe cujas instâncias são classes". Em MAS-ML, as metaclasses: *Class*, *AgentClass*, *EnvironmentClass*, *ObjectRoleClass*, *AgentRoleClass* e *OrganizationClass* foram criadas por Silva (2004) para representarem, respectivamente, as entidades objeto, agente, ambiente, papel de objeto, papel de agente e organização descritas no *framework* conceitual TAO. Todas essas metaclasses estendem a metaclasse *Classifier* – a qual é definida no metamodelo de UML (BOOCH; RUMBAUGH; JACOBSON, 2005).

As metaclasses *StructuralFeature* e *BehavioralFeature* estendem a metaclasse *Features* (ambas são definidas no metamodelo de UML) e definem, respectivamente, as características estruturais e comportamentais das entidades. Por exemplo, as características estruturais da entidade agente são os objetivos e as crenças do agente, e as características comportamentais são as ações e os planos que o agente executa.

As metaclasses *AgentAction*, *AgentPlan* e *AgentProtocol* estendem a metaclasse *BehavioralFeature* e foram criadas por Silva (2004) para representarem, respectivamente, as ações que são executadas por um agente, os planos (conjunto de ações) que um agente executa para alcançar seus objetivos, e os protocolos que um agente deve seguir ao exercer um determinado papel e ao trocar mensagens com outras entidades do sistema.

Como as mensagens enviadas e recebidas por um agente são diferentes das mensagens enviadas e recebidas por um objeto, Silva (2004, p. 76) criou uma metaclasse chamada *AgentMessage* para representar as propriedades das mensagens usadas em um protocolo de comunicação. Essas propriedades incluem o tipo, o conteúdo, o emissor e o destinatário da mensagem.

As metaclasses *AgentPerceptionFunction* e *AgentPlanningStrategy* estendem a metaclasse *BehavioralFeature* e foram criadas por Gonçalves (2009) para representarem, respectivamente, as informações que podem ser percebidas pelos sensores de um agente dentro de um ambiente, e o planejamento das ações que serão seguidas por um agente no alcance de um objetivo.

2.5.1.2 Os Estereótipos

Um estereótipo é um elemento de modelagem usado para rotular tipos de classes de objeto, sendo que uma classe pode ter um ou mais estereótipos. Um estereótipo é representado entre os sinais <<*name_estereotype*>> ou através de um ícone (BOOCH; RUMBAUGH; JACOBSON, 2005).

Em MAS-ML, Silva (2004) criou os estereótipos <<*goal*>> e <<*belief*>> para representar respectivamente, os objetivos e as crenças das entidades: agente, organização, papel de agente e ambiente ativo. A entidade organização, além de possuir as mesmas propriedades que a entidade agente, possui axiomas e define papeis de agente e de objeto. Os axiomas são usados para representar as regras que devem ser obedecidas pelos agentes dentro da organização. Desta forma, Silva (2004, p. 78) criou o estereótipo <<*axiom*>> e o associou às propriedades da metaclasse *OrganizationClass* para representar os axiomas que são definidos por uma organização.

O estereótipo <<*duty*>> foi criado para identificar as ações que um agente deve executar ao exercer um papel, e o estereótipo <<*right*>> foi criado para identificar as ações que um agente tem permissão de executar ao exercer um papel e ambos estão associados à metaclasse *AgentAction* (SILVA, 2004, p. 78).

Em seu trabalho, Gonçalves (2009, p. 46) criou os seguintes estereótipos: <<*next-function*>>, <<*formulate-goal-function*>>, <<*formulate-problem-function*>> e <<*utility-function*>> e os associou a metaclasse *AgentAction* para representar, respectivamente, a função próximo, função de formulação de objetivo, função de formulação de problema e função utilidade. Os estereótipos criados por Gonçalves (2009) representam as ações especiais de um agente e são utilizados conforme a definição da arquitetura interna do agente.

Nesta seção, foi apresentada a definição dos conceitos que fundamentaram este trabalho: sistemas multiagentes, ontologias, o histórico do *framework* conceitual TAO e da linguagem de modelagem MAS-ML. Na próxima seção, apresentamos a metodologia usada na execução deste trabalho.

3 PROCEDIMENTOS METODOLÓGICOS

A primeira fase deste trabalho consistiu em selecionar os elementos (as entidades e os relacionamentos) mais comuns de ontologias e inserir a definição dos mesmos no *framework* conceitual TAO. Além disso, esses elementos foram classificados de acordo com os aspectos estruturais (entidades e relacionamentos) e dinâmicos (interação entre as entidades) descritos no TAO, e posteriormente, foi descrito como as entidades e os relacionamentos de ontologias se relacionam com as entidades e os relacionamentos já existentes no TAO.

A segunda fase consistiu em estender a sintaxe abstrata de MAS-ML 2.0, ou seja, estender o metamodelo da linguagem de forma que o metamodelo passasse a incorporar os elementos (entidades e relacionamentos) que foram definidos na fase anterior. Para isso, a linguagem UML define alguns métodos de extensão, como: criação de novas metaclasses, estereótipos (*stereotypes*), restrições (*constraints*) e *tagged values*. Como MAS-ML estende UML, esses recursos de extensão também foram utilizados como estratégia para estender o metamodelo de MAS-ML 2.0.

Consequentemente, a terceira fase consistiu em estender a sintaxe concreta da linguagem, ou seja, trabalhamos na representação gráfica dos elementos contemplados no metamodelo, mantendo assim, a consistência entre a sintaxe abstrata e a sintaxe concreta. Além disso, a representação desses elementos foi trabalhada individualmente em cada um dos diagramas estáticos (classes, organização e papel) e dinâmico (diagrama de sequência) da linguagem MAS-ML 2.0.

A quarta e última fase consistiu em identificar e modelar classes de ontologias, nos diagramas estáticos e dinâmico, que serão utilizadas pelos agentes de um sistema multiagente desenvolvido para o Moodle (Gonçalves et al., 2013). Esse modelo serve como demonstração de uso da extensão inserida em MAS-ML, e para analisar a expressividade da linguagem após a extensão realizada neste trabalho.

4 MODELANDO ONTOLOGIAS EM MAS-ML 2.0

Nesta seção, apresentamos a extensão proposta ao *framework* conceitual TAO e à linguagem de modelagem MAS-ML 2.0 para modelar ontologias no contexto de sistemas multiagentes.

A modelagem de ontologias em MAS-ML 2.0 envolveu: a extensão dos conceitos de entidades e relacionamentos descritos no *framework* conceitual TAO; a inclusão de novos estereótipos no metamodelo de entidades e de relacionamentos, e a representação desses elementos nos diagramas de classes, organização papel e sequência.

4.1 Estendendo o *Framework* Conceitual TAO

Como extensão ao *framework* conceitual TAO, acrescentamos o conceito de ontologia na definição de três entidades: *Class* (classe de objeto), *ObjectRoleClass* (classe de papel de objeto) e *Package* (pacote), visto que, uma ontologia será formada a partir de instâncias destas entidades. Na definição apresentada por Silva (2004, p. 34), um objeto possui atributos e métodos, executa operações que modificam seu estado durante o seu ciclo de vida, não possui uma estrutura predefinida, e armazena através de seus atributos, informações sobre si mesmo, sobre o ambiente e sobre outros objetos.

Neste trabalho, a entidade *Class* também será usada para representar os conceitos definidos em uma ontologia. E assim como na definição tradicional, os objetos que representam as classes (conceitos) de uma ontologia também possuem uma estrutura genérica, formada por atributos e métodos. Os atributos são usados para armazenar as informações de um determinado conceito da ontologia, já os métodos serão usados para definir o comportamento desses conceitos dentro da ontologia.

A entidade *ObjectRoleClass* será usada para representar os papéis exercidos pelos objetos de uma ontologia. Um papel de objeto descreve as características de um objeto de ontologia que são vistas por outras entidades e orienta o comportamento do mesmo, fazendo com que o objeto aja de acordo com as chamadas de seu papel. Além disso, um papel de objeto pode restringir o acesso ao objeto limitando as informações (atributos) e o comportamento (métodos) que outras entidades podem acessar, bem como adicionar atributos e métodos ao objeto que está exercendo o papel (SILVA, 2004, p. 39).

A entidade *Package* foi originalmente definida no metamodelo da linguagem UML. Ela é usada para organizar os objetos em grupos e armazenar outros componentes de um sistema, como *Interfaces* por exemplo. Introduzimos a entidade *Package* no *framework* TAO

para representar os pacotes criados para armazenar os elementos de uma ontologia. Os pacotes que armazenam ontologias possuem uma estrutura genérica, formada pelo conteúdo (classes de objeto, classes de papel de objeto e subpacotes) que o pacote armazena e adornado com um estereótipo que o identifica como sendo um pacote de ontologia.

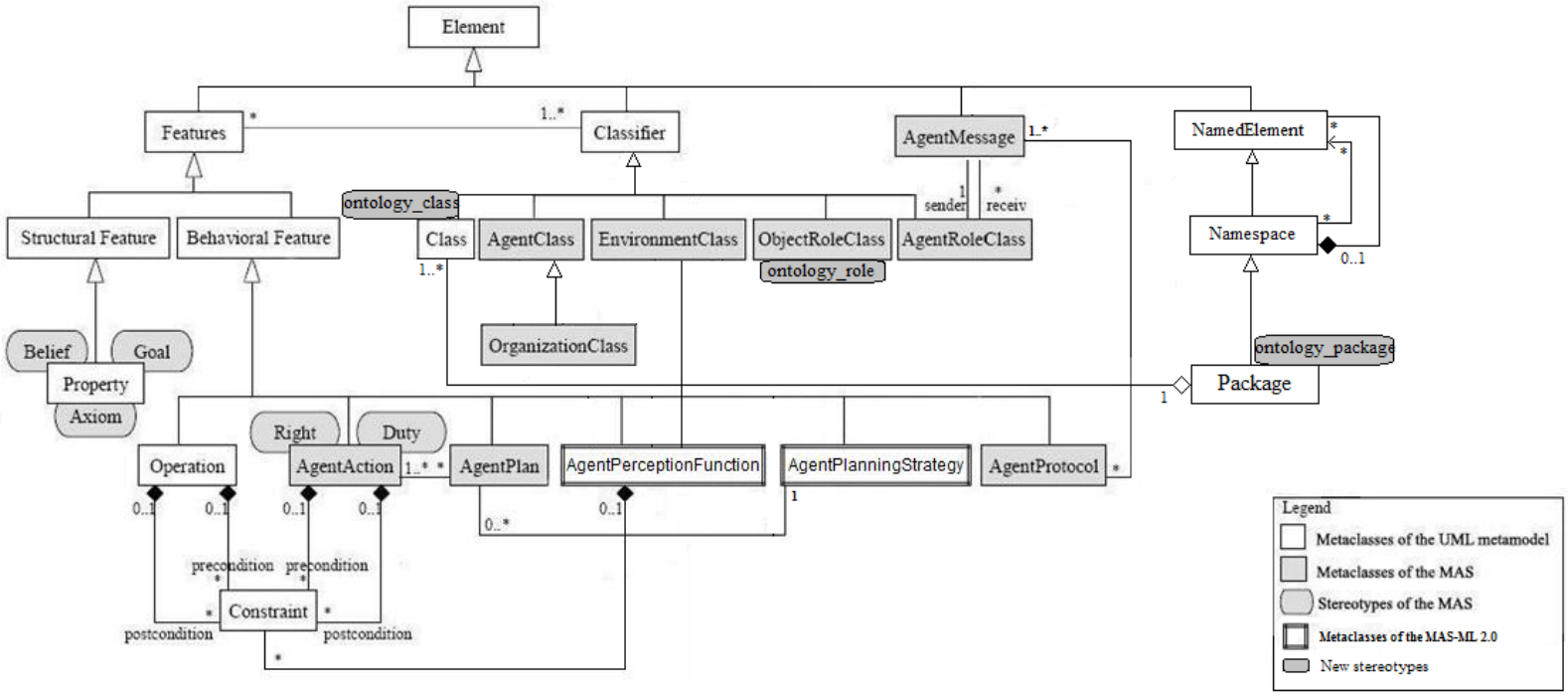
O relacionamento *aggregation*, descrito no TAO, foi estendido para representar um agente obtendo uma ontologia. Utilizamos a agregação por esta ser um tipo de associação na qual uma entidade pode agregar outras entidades para desempenhar suas funções. Nesse caso, quando esse relacionamento for aplicado entre um agente e uma ontologia, isso significa que o agente pode utilizar partes de uma ontologia para desempenhar seu papel.

4.2 Estendendo a Sintaxe Abstrata de MAS-ML 2.0

A sintaxe abstrata de uma linguagem de modelagem está relacionada aos elementos presentes no metamodelo da mesma, descrevendo apenas os elementos relevantes ao entendimento da linguagem. O metamodelo por sua vez, mostra em alto nível, as entidades, os relacionamentos e estereótipos que compõem a linguagem de modelagem.

Neste contexto, três novos estereótipos foram criados e introduzidos no metamodelo da linguagem MAS-ML 2.0 para modelar ontologias. Criamos os estereótipos *<<ontology_class>>*, *<<ontology_package>>* e *<<ontology_role>>* e os associamos, respectivamente, às metaclasses *Class* – para representar as classes de uma ontologia; *Package* – para representar um pacote que armazena todos os elementos relacionados à uma ontologia; e *ObjectRoleClass* – para representar os papéis exercidos por objetos de uma ontologia. A Figura 3 apresenta o metamodelo de MAS-ML após a inserção dos estereótipos.

Figura 3 – O Metamodelo de MAS-ML 2.0 Estendido para Representar Ontologias.



Fonte: adaptado de Gonçalves (2009, p. 46).

4.2.1 Definindo as Associações entre Metaclasses de MAS-ML e Elementos de Ontologia

Nesta subseção, apresentamos as associações entre as metaclasses utilizadas para modelar ontologias e as metaclasses que já estavam definidas no metamodelo de MAS-ML.

A metaclasse *EnvironmentClass* está associada à metaclasse *Class*, adornada com o estereótipo <<*ontology_class*>>, e com *Package* adornada com <<*ontology_package*>>, através do relacionamento *inhabit*, especificando que uma classe e um pacote de ontologia podem residir em um ambiente (*EnvironmentClass*).

A metaclasse *OrganizationClass* está associada às metaclasses:

a) *Class*, adornada com o estereótipo <<*ontology_class*>>, através do relacionamento *play in*, especificando que uma classe de ontologia pode exercer um papel de ontologia em uma organização;

b) *ObjectRoleClass*, adornada com o estereótipo <<*ontology_role*>>, através do relacionamento *define*, especificando que os papéis de uma ontologia são definidos por uma organização.

A metaclasse *AgentClass* está associada às metaclasses: *Class*, adornada com o estereótipo <<*ontology_class*>>, e à metaclasse *Package*, adornada com <<*ontology_package*>>, através do relacionamento *aggregation*, especificando que um agente pode agregar classes e pacotes de ontologia.

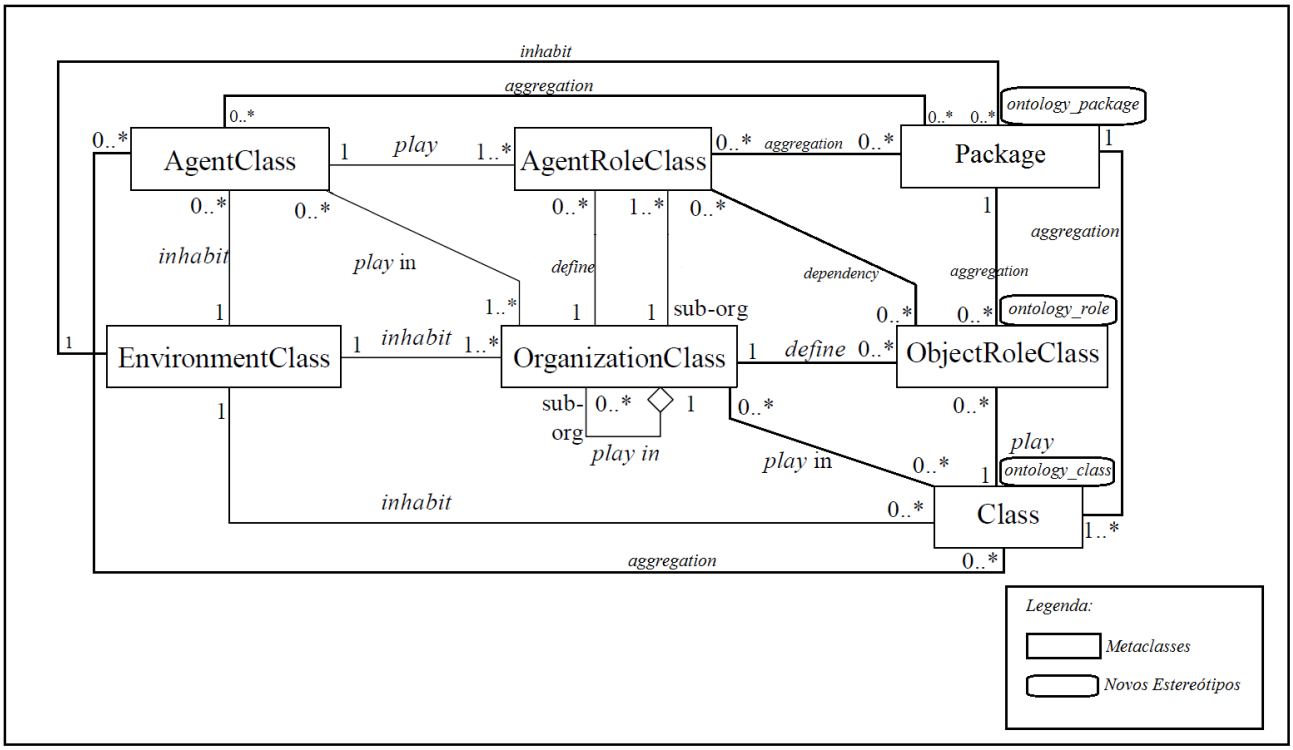
A metaclasse *AgentRoleClass* está associada às metaclasses:

a) *ObjectRoleClass*, adornada com o estereótipo <<*ontology_role*>>, através do relacionamento *dependency*, especificando que o papel de um agente pode depender do papel de uma ontologia;
b) *Package*, adornada com o estereótipo <<*ontology_package*>>, através do relacionamento *aggregation*, especificando que o papel de um agente pode agregar um pacote de ontologia.

A metaclasse *Package* está associada às metaclasses: *Class*, adornada com o estereótipo <<*ontology_class*>>, através do relacionamento *aggregation*, especificando que as classes de uma ontologia fazem parte de um pacote de ontologia; e à metaclasse *ObjectRoleClass*, adornada com o estereótipo <<*ontology_role*>>, através do relacionamento *aggregation*, especificando que as classes de papel de ontologia fazem parte de um pacote de ontologia.

Por fim, as metaclasses *Class* adornada com <<*ontology_class*>> e *ObjectRoleClass* adornada com <<*ontology_role*>> estão associadas pelo relacionamento *play*, especificando que uma classe de ontologia exerce um papel de ontologia. A figura 4 mostra as associações entre metaclasses que foram descritas acima.

Figura 4 – Associações entre Metaclasses de MAS-ML e Elementos de Ontologia.



Fonte: adaptado de Silva (2004, p. 46).

4.2.2 Aplicando os Relacionamentos de MAS-ML aos Elementos de Ontologia

Nesta subseção, apresentamos a forma como os relacionamentos de MAS-ML serão aplicados entre as entidades usadas para modelar ontologia. Alguns desses relacionamentos foram criados por Silva (2004), como é o caso dos relacionamentos: *inhabit*, *ownership*, *play* e *control*; e os demais relacionamentos (*association*, *aggregation*, *composition*, *specialization/generalization*, *realization* e *dependency*) provêm da linguagem UML.

4.2.2.1 Estendendo o Relacionamento Aggregation

De acordo com Silva (2004, p. 83) a linguagem "UML não define uma metaclasse para representar explicitamente o relacionamento *aggregation*". Assim, a agregação em UML é definida como uma associação que possui características particulares: a agregação define um relacionamento total ou parcial entre duas entidades.

Em UML, este relacionamento é aplicado entre classes de objeto. A agregação parcial ocorre quando uma classe contém instâncias de outra classe, de tal forma que a primeira seja o todo, e que as classes referenciadas sejam as partes do todo. Nesse caso, as classes que foram agregadas não deixarão de existir quando a classe agregadora for destruída (BOOCH; RUMBAUGH; JACOBSON, 2005). A agregação total, mais conhecida como composição,

ocorre quando uma classe é responsável pela criação e pelo ciclo de vida de outra classe. Assim, quando duas classes estiverem ligadas pelo relacionamento composição, no caso de uma classe ser destruída, a outra classe também será.

Em MAS-ML, Silva (2004, p. 83) estendeu este relacionamento para também ser usado entre classes de papel de objeto e entre classes de papel do agente. Neste trabalho, estendemos este relacionamento para também ser usado entre classes de papel de ontologia e, para especificar como um agente ou uma organização irão utilizar uma ontologia. Para isso, criamos o estereótipo <<*obtain_ontology*>> e o associamos à sintaxe do relacionamento *aggregation*.

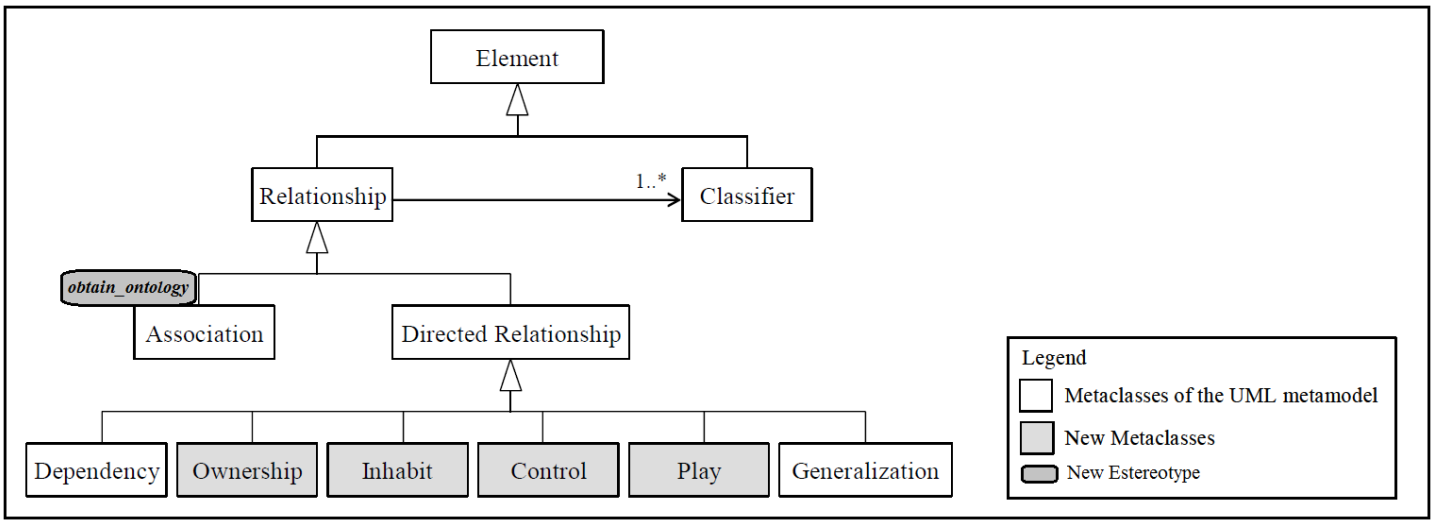
O relacionamento *aggregation* <<*obtain_ontology*>> está relacionado a agentes ou organizações que utilizam ontologias, isso significa que, quando um agente ou uma organização estiverem ligados à uma ontologia por este relacionamento, o agente e/ou a organização está obtendo as instâncias dessa ontologia, e por tanto, os mesmos poderão consultar as informações que a ontologia oferece para alimentar sua base de dados, ou seja, suas crenças. Caso as instâncias de um agente ou de uma organização que utilizam uma determinada ontologia sejam destruídas, as instâncias da ontologia não serão, pois o relacionamento de agregação define que quando o "todo" é destruído, as "partes" que formam o "todo" não são. Entretanto, no caso em que se deseja destruir o agente ou organização e também as instâncias da ontologia que eles utilizam, torna-se necessário explicitar que essas instâncias também devem ser destruídas.

O relacionamento *aggregation* <<*obtain_ontology*>> pode ser aplicado entre classes do agente e classes de ontologia, entre classes do agente e pacotes de ontologia, entre classes de papel do agente e pacotes de ontologia, entre classes de organização e classes de ontologia, e entre classes de organização e pacotes de ontologia.

A linguagem MAS-ML é formada por outros relacionamentos, além do relacionamento *aggregation*, são eles: *association*, *specialization/generalization*, *dependency*, *inhabit*, *ownership*, *play* e *control*. Entretanto, não houve a necessidade de estender a sintaxe destes relacionamentos, pois a representação original dos mesmos é suficiente para modelar os elementos de ontologias.

A figura a seguir mostra todas as metaclasses que formam o metamodelo de relacionamentos de MAS-ML, as quais representam todos relacionamentos descritos na subseção 2.4.1.2 bem como o estereótipo criado e associado à metaclasse *Association* para representar o relacionamento *aggregation* <<*obtain_ontology*>>, descrito anteriormente.

Figura 5 – O Metamodelo de Relacionamentos de MAS-ML Estendido.



Fonte: adaptado de Silva (2004, p. 86).

4.2.3 Representação dos Elementos de Ontologia

Como visto na subseção 2.2.2, uma ontologia possui conceitos, relações entre conceitos, axiomas e instâncias. Para representar os elementos de uma ontologia na sintaxe concreta de MAS-ML 2.0, utilizamos a seguinte estratégia:

a) os conceitos (classes) de uma ontologia poderão ser representados através da metaclasse *Class* adornada com o estereótipo <<*ontology_class*>>;

b) os relacionamentos permitidos entre classes de ontologia são os mesmos definidos por UML e usados em MAS-ML, são eles: *association* (associação), *aggregation* (agregação), *composition* (composição), *specialization/generalization* (herança), *realization* (implementação) e *dependency* (dependência).

c) os axiomas poderão ser representados através do estereótipo <<*axiom*>>, o qual já estava definido no metamodelo de MAS-ML 2.0 e faz parte das propriedades de uma entidade. Ele poderá ser atributo de uma classe de ontologia apenas quando esta estiver adornada com o estereótipo <<*ontology_class*>>. As regras definidas pelos axiomas poderão ser explicitadas através de notações OCL (*Object Constraint Language*) inclusas dentro de *tagged values* e associadas a uma propriedade definida como axioma.

d) as instâncias de uma ontologia são os objetos instanciados a partir de classes da ontologia e fazem parte do corpo de conhecimento da mesma.

Nas próximas subseções, os elementos definidos para modelar ontologias são representados nos diagramas estáticos e dinâmico da linguagem MAS-ML 2.0.

4.3 Estendendo a Sintaxe Concreta de MAS-ML 2.0

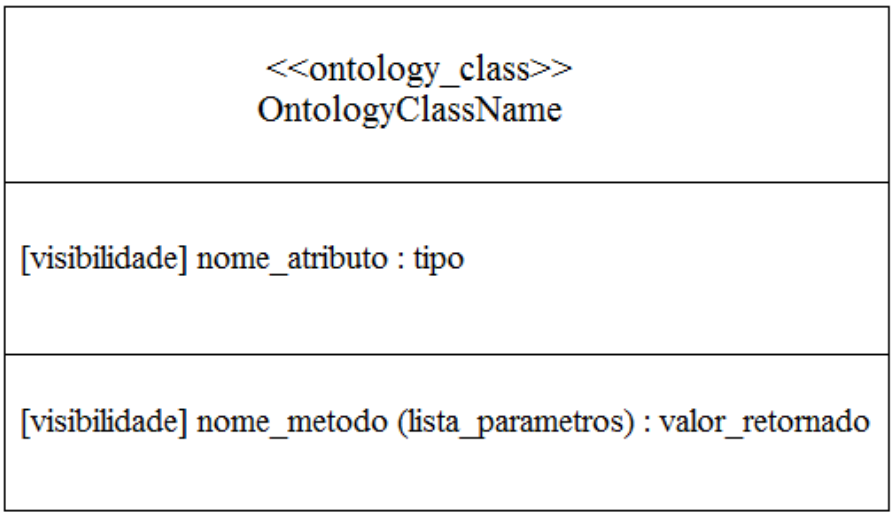
Nesta subseção, mostramos a representação das metaclasses e dos relacionamentos, cuja sintaxe concreta foi estendida para oferecer suporte à modelagem de ontologias em MAS-ML.

4.3.1 Representação de Class <<ontology_class>>

Em UML, uma classe é representada através de uma "caixa" que possui três compartimentos divididos por linhas horizontais: o primeiro compartimento contém o nome da classe, o segundo contém a lista de atributos e o terceiro contém a lista de operações que são executadas pela classe (BOOCH; RUMBAUGH; JACOBSON, 2005).

Segundo Silva (2004, p. 87), no *framework* TAO, a definição da entidade *Class* (objeto) é semelhante à definição da entidade *Class* de UML. Desta forma, as classes de uma ontologia poderão ser representadas por meio da metaclasse *Class* adornada com o estereótipo *<<ontology_class>>* e, os relacionamentos permitidos entre classes que utilizam este estereótipo, são os mesmos definidos por UML e MAS-ML, são eles: *association* (associação entre classes), *aggregation* (agregação), *specialization/generalization* (herança), *dependency* (dependência), *composition* (composição), *realization* (implementação) e *inhabit* (habita) usado entre classes de ontologia e classes do ambiente. A figura a seguir mostra a sintaxe concreta de uma classe de ontologia.

Figura 6 – Representação de uma Classe de Ontologia.



Fonte: adaptado de Booch, Rumbaugh e Jacobson (2005).

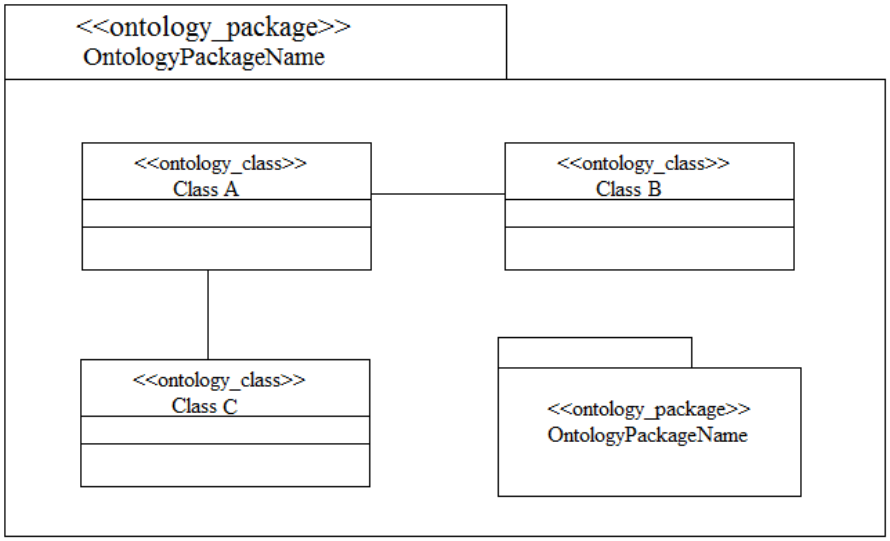
4.3.2 Representação de Package <<ontology_package>>

Em UML, um pacote é um mecanismo usado para organizar elementos em grupos e é representado como um retângulo com uma aba no canto superior esquerdo (BOOCH; RUMBAUGH; JACOBSON, 2005).

A representação gráfica de um pacote de ontologia segue a mesma definição de pacote dada por UML. Entretanto, criamos o estereótipo <<*ontology_package*>> e o associamos à metaclasse *Package*, definida pelo metamodelo de UML, para representar um pacote de ontologia. Um pacote de ontologia pode conter uma ou mais classes de ontologias e, além de classes ele pode conter papéis de objetos de ontologia e pacotes subordinados (subpacotes), sendo possível definir a visibilidade de um pacote bem como a visibilidade dos elementos contidos nele. Neste trabalho, um pacote de ontologia define uma única ontologia. Assim, todos os elementos (classes, papéis e subpacotes) de uma determinada ontologia, devem fazer parte de um único pacote.

Os relacionamentos permitidos entre pacotes de ontologia são os mesmos permitidos entre pacotes em UML, e foi acrescentado um relacionamento de MAS-ML, são eles: *dependency* (dependência simples entre pacotes ou elementos de um pacote), dependência com estereótipo <<*import*>> (importação de pacotes), dependência com o estereótipo <<*merge*>> (união de pacotes), *association* (associação), *aggregation* (agregação), *specialization/generalization* (herança) e *inhabit* (habita) usado entre pacotes de ontologia e classes do ambiente, especificando que um pacote pode residir em um ambiente do SMA. A figura a seguir mostra a sintaxe concreta de um pacote de ontologia.

Figura 7 – Representação de um Pacote de Ontologia.

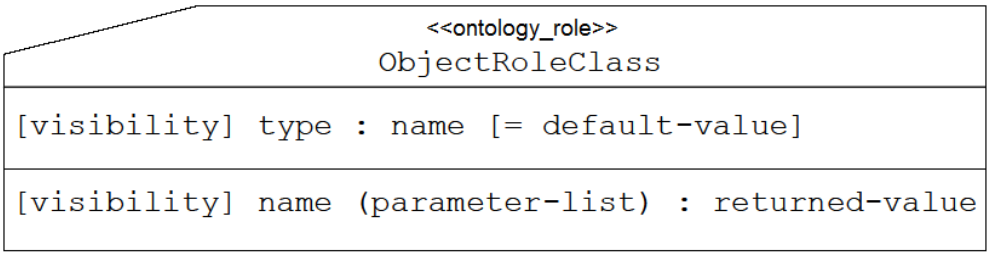


Fonte: adaptado de Booch, Rumbaugh e Jacobson (2005).

4.3.3 Representação de ObjectRoleClass <<ontology_role>>

A metaclasse *ObjectRoleClass* (Papel de Objeto) foi adornada com o estereótipo *<<ontology_role>>* para representar uma *OntologyRoleClass* (Classe de Papel de Ontologia). Uma classe de papel de ontologia segue a mesma definição de uma classe de papel de objeto: é representada através de uma "caixa" que possui três compartimentos divididos por linhas horizontais: o primeiro compartimento contém o nome da classe, o segundo contém a lista de atributos e o terceiro contém a lista de métodos. De acordo com Silva (2004, p. 88), a notação usada para descrever os atributos e os métodos de uma classe de papel de objeto é equivalente à notação usada para descrever os atributos e os métodos de uma classe de objeto. Os atributos são descritos por sua visibilidade, nome, tipo e valor padrão (podendo ser omitido), já os métodos são descritos por sua visibilidade, nome, lista de parâmetros e valor retornado. A figura a seguir mostra a sintaxe concreta de uma classe de papel de ontologia.

Figura 8 – Representação de uma Classe de Papel de Ontologia.

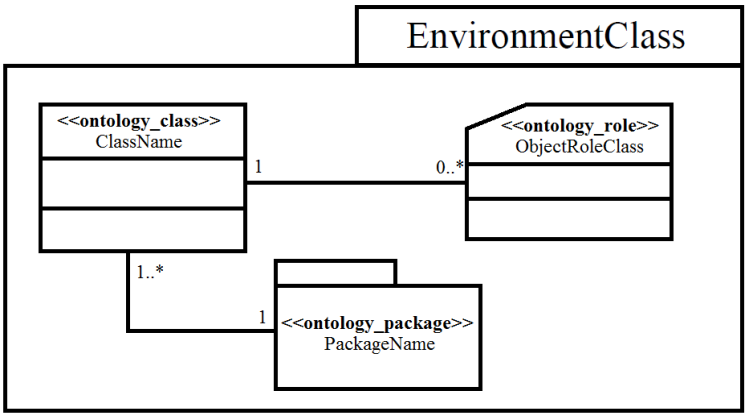


Fonte: adaptado de Silva (2004).

4.3.4 Representação do Relacionamento Inhabit

O relacionamento *inhabit* é mostrado graficamente, inserindo a entidade no compartimento inferior de uma *EnvironmentClass* (SILVA, 2004, p. 91). Uma vez que o relacionamento *inhabit* relaciona *EnvironmentClass* às metaclasses *Class*, *AgentClass* e *OrganizationClass*, e que uma classe de ontologia é representada pela metaclasse *Class* adornada com o estereótipo <<*ontology_class*>>, consequentemente o relacionamento *inhabit* também é aplicável à classes de ontologia.

Figura 9 – O Relacionamento *Inhabit* Aplicado à Elementos de Ontologia.

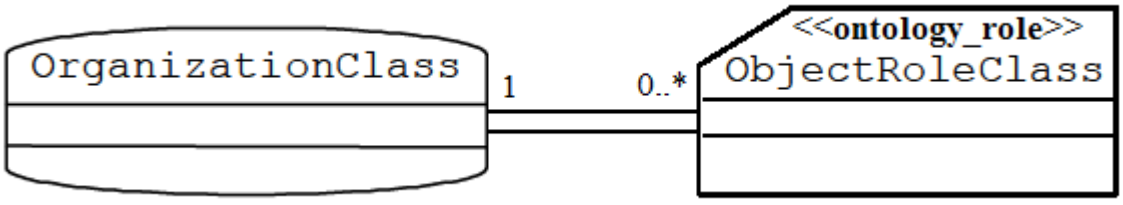


Fonte: adaptado de Silva (2004, p. 92).

4.3.5 Representação do Relacionamento Ownership

O relacionamento *ownership* é mostrado graficamente como uma linha dupla que liga o proprietário e o membro (SILVA, 2004, p. 91). Ele é aplicado entre *OrganizationClass* e *AgentRoleClass*, e entre *OrganizationClass* e *ObjectRoleClass*. Neste trabalho, ele também é aplicado entre uma *OrganizationClass* e uma *ObjectRoleClass* adornada com o estereótipo <<*ontology_role*>>, especificando que um papel de ontologia é também um membro da organização que o define.

Figura 10 – O Relacionamento *Ownership* Aplicado à Elementos de Ontologia.



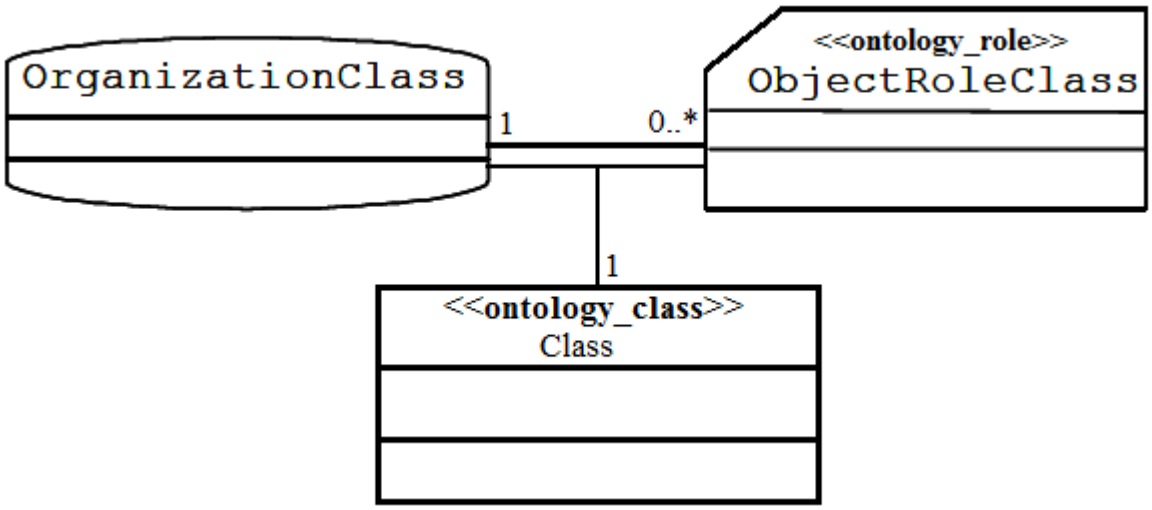
Fonte: adaptado de Silva (2004, p. 92).

4.3.6 Representação do Relacionamento Play

O relacionamento *play* é mostrado graficamente como uma linha simples que liga: um agente (*AgentClass*) e seu papel (*AgentRoleClass*); ou um objeto (*Class*) e seu papel (*ObjectRoleClass*) à uma organização (*OrganizationClass*). A linha que descreve o relacionamento *play* é ligada à linha dupla que descreve o relacionamento *ownership* o qual identifica uma organização e os papéis que ela define (papel de agente ou de objeto) (SILVA, 2004, p. 92).

Quando uma *Class* ou uma *ObjectRoleClass* estiverem adornadas, respectivamente, com os estereótipos <<*ontology_class*>> e <<*ontology_role*>>, e ligadas pelo relacionamento *play* à uma organização, isso significa que um objeto de ontologia está exercendo um papel de ontologia em uma organização.

Figura 11 – O Relacionamento *Play* Aplicado à Elementos de Ontologia.



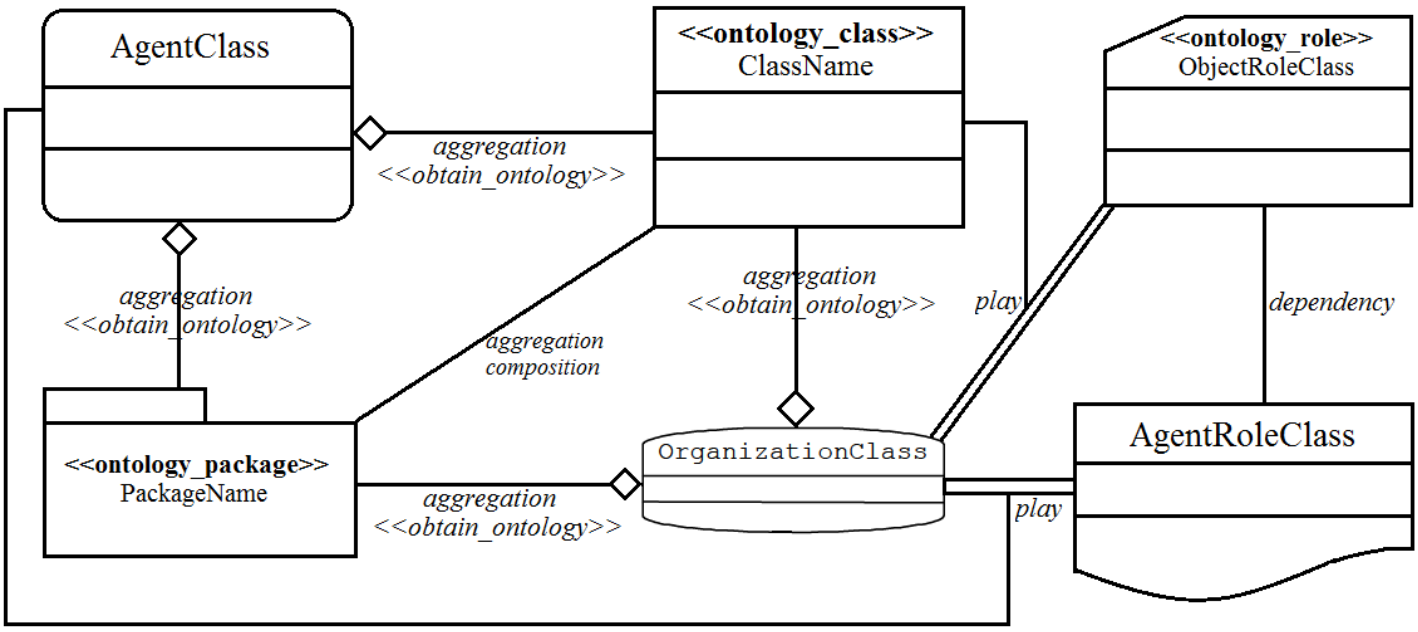
Fonte: adaptado de Silva (2004, p. 93).

4.3.7 Representação do Relacionamento Aggregation <<obtain_ontology>>

O relacionamento *aggregation* <<*obtain_ontology*>> é mostrado graficamente como uma linha simples com um losango vazio posicionado na entidade agregadora, e adornado com o estereótipo <<*obtain_ontology*>>, ligando a entidade agregadora à entidade que está sendo agregada.

Neste trabalho, este relacionamento pode ser aplicado entre classes do agente e classes de ontologia, entre classes do agente e pacotes de ontologia, entre classes de organização e classes de ontologia, e entre classes de organização e pacotes de ontologia.

Figura 12 – O Relacionamento *Aggregation* <<*obtain_ontology*>> Aplicado aos Elementos



de Ontologia.

Fonte: elaborada pelo autor.

4.3.8 Representação de Ontologias no Diagrama de Classes

O diagrama de classes é usado para modelar um conjunto de classes, interfaces e relacionamentos. As classes correspondem a objetos que possuem atributos e métodos. Os atributos correspondem às informações que o objeto armazena, e os métodos correspondem às ações que o objeto realiza (BOOCH; RUMBAUGH; JACOBSON, 2005).

Em MAS-ML, de acordo com Silva (2004, p. 94), as classes que podem participar do diagrama de classes são: classe do agente, classe de organização, classe do ambiente, classe de objeto e outras classes definidas por UML.

Neste trabalho, utilizamos o diagrama de classes para modelar as classes e os pacotes de uma ontologia que residem em um ambiente do sistema, classes e pacotes de ontologia que estão associados a agentes e organizações, e modelar a especialização entre classes e pacotes de uma ontologia. Os relacionamentos usados nesse diagrama são todos aqueles definidos no diagrama de classes de UML, mais os relacionamentos de MAS-ML listados a seguir:

a) *inhabit* – usado entre classes de ontologia e classes do ambiente, e entre pacotes de ontologia e classes do ambiente.
b) *aggregation* <<*obtain_ontology*>> – usado entre classes do agente e classes de ontologia, entre classes do agente e pacotes de ontologia, entre classes de papel do

agente e pacotes de ontologia, entre classes de organização e classes de ontologia, e entre classes de organização e pacotes de ontologia.

4.3.9 Representação de Ontologias no Diagrama de Organização

O objetivo do diagrama de organização é modelar todas as organizações de um SMA. Ele é responsável por modelar as propriedades de uma organização (crenças, objetivos, planos, ações e axiomas), os papéis definidos por ela, as entidades (agentes, objetos e suborganizações) que exercem esses papéis e o ambiente em que a organização reside (SILVA, 2004, p. 95).

O diagrama de organização também pode ser utilizado para descrever as propriedades dos papéis (objetivos, crenças, deveres, direitos e protocolos) definidos na organização e as entidades que exercem cada papel. Entretanto, a modelagem dos relacionamentos entre os papéis e entre os papéis e as classes é feita no diagrama de papel. No diagrama de organização é necessário definir as propriedades de cada agente, objeto e suborganização, contudo, os seus relacionamentos são modelados no diagrama de classes (SILVA, 2004, p. 95).

De acordo com Silva (2004, p. 95), as classes que podem ser usadas no diagrama de organização são: classe de objeto, classe de papel de objeto, classe do agente, classe de papel do agente, classe de organização e classe do ambiente, e os relacionamentos que podem ser usados são:

a) *ownership* – usado entre classes de organização e classes de papel (papel de agente e papel de objeto) definidas pela organização;
b) *play* – usado entre classes do agente e classes de papel do agente, entre classes de suborganização e classes de papel do agente, e entre classes (objeto) e classes de papel de objeto;
c) *inhabit* – usado entre as classes do ambiente e organização, e entre classes do ambiente e classes do agente.

Neste trabalho, utilizamos o diagrama de organização para modelar as organizações que definem e utilizam elementos de ontologias: classes, pacotes e papéis de ontologia. As classes que participam deste diagrama e cuja sintaxe concreta foi estendida são: *Class* e *ObjectRoleClass*, sendo que as mesmas foram representadas, respectivamente, nas subseções 4.3.1 e 4.3.3. Os relacionamentos usados entre classes de organização e elementos de ontologias são os mesmos definidos por Silva (2004), são eles:

a) *ownership* – usado para especificar que uma organização é proprietária de um papel de ontologia. Aplica-se à classes de organização e classes de papel de ontologia;
b) *play* – usado para especificar que um objeto de ontologia exerce um papel de ontologia. Aplica-se à classes de ontologia e classes de papel de ontologia;
c) *inhabit* – usado para especificar que as organizações, os agentes e os elementos de ontologia residem em um ambiente.

4.3.10 Representação de Ontologias no Diagrama de Papel

O diagrama de papel é responsável por ilustrar os relacionamentos entre papéis de agente e papéis de objeto identificados nos diagramas de organização. Esse diagrama também identifica quais classes podem ser acessadas pelos papéis de objeto e pelos papéis do agente.

De acordo com Silva (2004, p. 95), as entidades que podem participar desse diagrama são classe de papel do agente (*AgentRoleClass*) e classe de papel de objeto (*ObjectRoleClass*).

Neste trabalho, utilizamos o diagrama de papel para modelar os papéis exercidos por objetos de ontologia. A classe que participa deste diagrama e cuja sintaxe concreta foi estendida é *ObjectRoleClass*, sendo que a mesma foi representada na subseção 4.3.3.

Os relacionamentos usados nesse diagrama são os mesmos definidos por Silva (2004, p. 95), são eles:

a) *control* (controle) – usado entre classes de papel do agente;
b) *dependency* (dependência) – usado entre classes de papel do agente, entre classes de papel de ontologia, entre classes de papel do agente e classes de papel de ontologia;
c) *association* (associação) – usado entre classes de papel de ontologia, entre classes de papel do agente e classes de papel de ontologia;
d) *aggregation* (agregação) – usado entre classes de papel de ontologia, e entre classes de papel do agente;
e) *specialization/generalization* – usado entre classes de papel do agente, e entre classes de papel de ontologia.

4.4 Representação de Ontologias no Diagrama de Sequência de MAS-ML 2.0

O diagrama de sequência é “um diagrama de interação que dá ênfase à ordenação temporal de mensagens [trocadas entre objetos]” (BOOCH; RUMBAUGH; JACOBSON, 2005, p. 455). Nesta subseção, representamos as metaclasses e os relacionamentos usados para modelar ontologias no diagrama de sequência de MAS-ML 2.0, mostrando as interações

entre as instâncias de entidades do SMA e entidades de ontologia, bem como as intra-ações definidas por cada instância.

Inicialmente, para representar a extensão proposta neste trabalho no diagrama de sequência de MAS-ML, precisamos definir os *pathnames* e ícones que identifiquem as instâncias de ontologias. Posteriormente, é necessário representar a criação, utilização e destruição de instâncias de uma ontologia pelas entidades do SMA. Para isso, utilizamos alguns estereótipos de MAS-ML e criamos novos estereótipos e os associamos à mensagens.

4.4.1 Definindo o Pathname de um Objeto de Ontologia

O *pathname* é uma estrutura de nomes que permite identificar as entidades que estão interagindo no diagrama de sequência. O *pathname* associado aos objetos no diagrama de sequência de MAS-ML descreve, à esquerda dos dois pontos, os nomes das instâncias e, à direita dos dois pontos, os nomes das classes (SILVA, 2004, p. 98).

O *pathname* simples de um objeto de ontologia pode descrever o nome do objeto, o papel que ele está exercendo, e o pacote no qual ele está contido. O *pathname* completo de um objeto de ontologia pode descrever o nome do objeto, o papel que ele está exercendo, o pacote no qual o objeto está contido, a organização na qual o objeto ou o pacote pertence, e o ambiente em que reside a organização. É importante ressaltar que o projetista pode substituir o *pathname* completo de um objeto de ontologia, por um *pathname* simples, quando apropriado.

Quadro 1 – Definindo o *Pathname* dos Objetos de Ontologia.

***Pathname* Simples**	*ObjecName/ObjectRoleName/OntologyPackageName : Class/ObjectRoleClass/OntologyPackage*
***Pathname* Completo**	*ObjecName/ObjectRoleName/OntologyPackageName/OrganizationName/ EnvironmentName : Class/ObjectRoleClass/OrganizationClass/OntologyPackage/Environment Class*

Fonte: elaborado pelo autor.

4.4.2 Definindo o Ícone de um Objeto de Ontologia

Após definir como os objetos de uma ontologia serão identificados, é preciso definir como eles serão representados graficamente no diagrama de sequência. O ícone usado para representar os objetos de uma ontologia no diagrama de sequência de MAS-ML segue a mesma definição dada por Silva (2004, p. 101): eles são representados por um retângulo

preenchido com o *pathname* (simples ou completo) e uma linha simples usada para marcar os eventos do ciclo de vida do objeto.

Quadro 2 – Ícone dos Objetos de Ontologia.

Instância	**Representação no Diagrama de Sequência**
Class (Objeto) adornada com o estereótipo <<*ontology_class*>>	*object/role/package/org/env : Class/ObjectRoleClass/Package/OrganizationClass/EnvironmentClass*

Fonte: adaptado de Silva (2004).

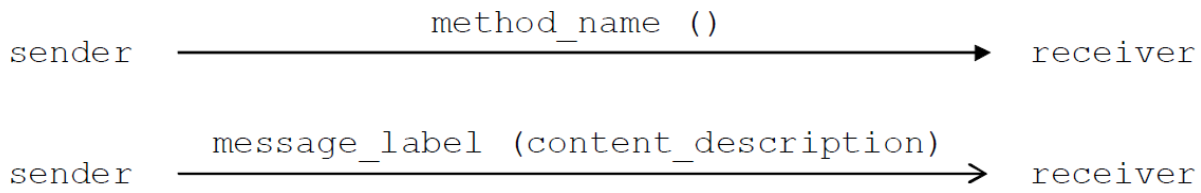
A seguir, são representadas as interações entre as entidades de MAS-ML envolvidas nos processos de criação, destruição e utilização de uma ontologia.

4.4.3 Representando a Troca de Mensagens

Em UML, o diagrama de sequência ilustra como um objeto envia mensagens a outros objetos, ou seja, como ele invoca métodos de outros objetos. Nesse caso, uma mensagem é apresentada como uma linha direta do emissor para o destinatário, quase sempre acompanhada do nome do método que está sendo invocado (SILVA, 2004, p. 101).

MAS-ML estende o diagrama de sequência de UML e também usa uma seta para representar um método de um objeto que está sendo chamado por outra entidade, e ainda, para representar as mensagens trocadas entre agentes, organizações e ambientes. As setas com cabeça fechada representam mensagens síncronas, e as setas com cabeça aberta representam mensagens assíncronas. A seta de cabeça fechada é usada para mostrar um método sendo chamado e indica a entidade que está chamando o método (emissor) e a entidade que executará o método (destinatário). A seta com cabeça aberta é usada para mostrar uma mensagem do agente, associa-se à seta: o rótulo e o conteúdo da mensagem, e a linha liga o emissor ao destinatário da mensagem (SILVA, 2004, p. 101-102).

Figura 13 – Elementos da Troca de Mensagens.



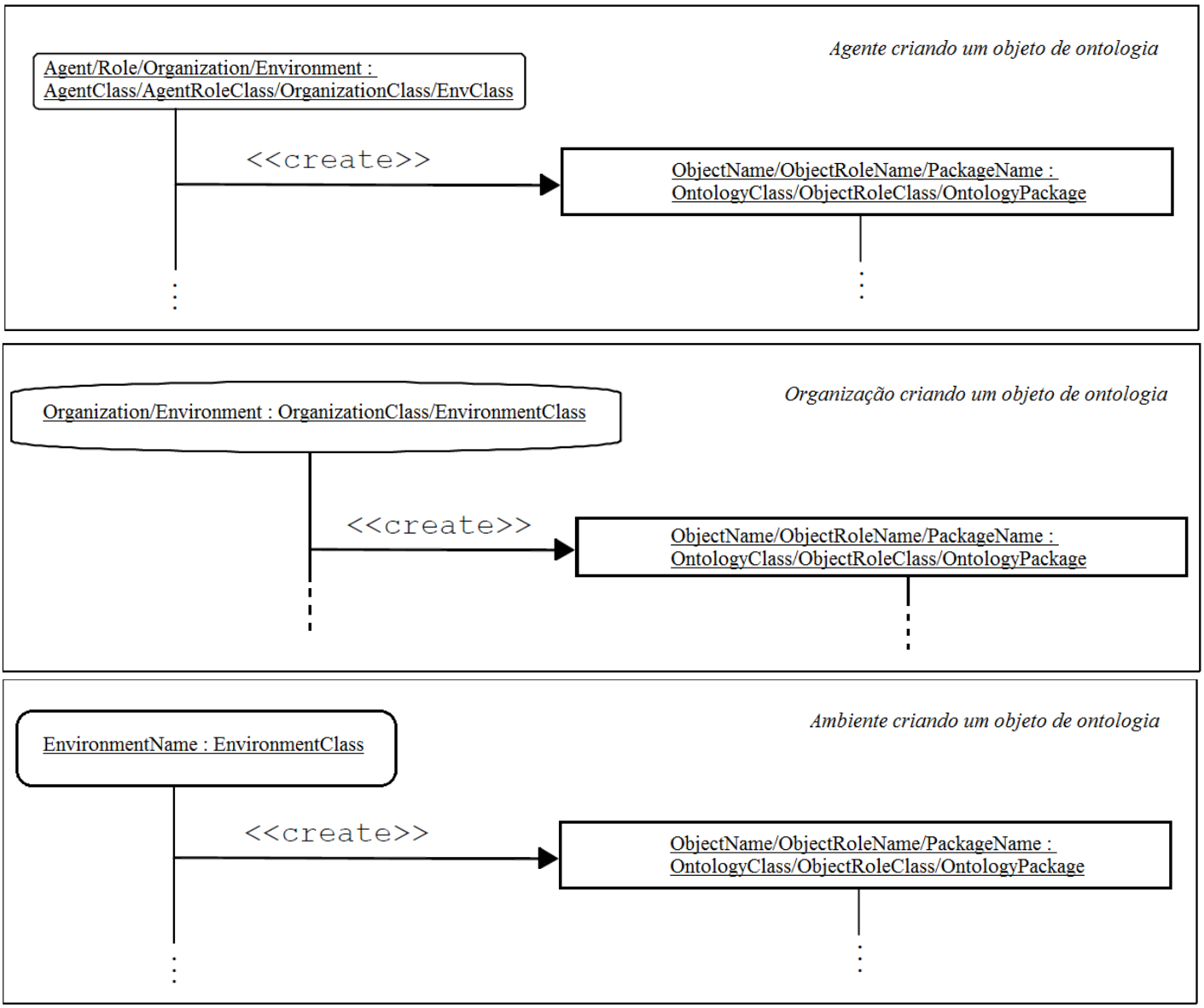
Fonte: Silva (2004, p. 102).

4.4.3.1 Instanciando Objetos de uma Ontologia

Em UML, os estereótipos <<*create*>> e <<*destroy*>> associados à mensagem são usados para representar a criação e a destruição de objetos. MAS-ML utiliza os mesmos estereótipos para representar a criação e a destruição de objetos, agentes, organizações e ambientes.

Neste trabalho, a instanciação das classes (objetos) de uma ontologia é representada no diagrama de sequência de MAS-ML 2.0 por uma seta de cabeça fechada partindo da entidade criadora (agente, organização ou ambiente) para o objeto que está sendo criado, adornada com o estereótipo <<*create*>> e acompanhada de uma mensagem de criação do objeto. Ao instanciar os objetos de uma ontologia pode-se utilizar o *pathname* completo do objeto, quando for necessário mencionar o nome e o papel do objeto, o pacote, a organização e o ambiente nos quais o objeto reside, ou pode-se utilizar o *pathname* simples do objeto, quando for necessário mencionar apenas o nome, o papel e pacote do objeto.

Figura 14 – Entidades de MAS-ML Criando Objetos de Ontologia.



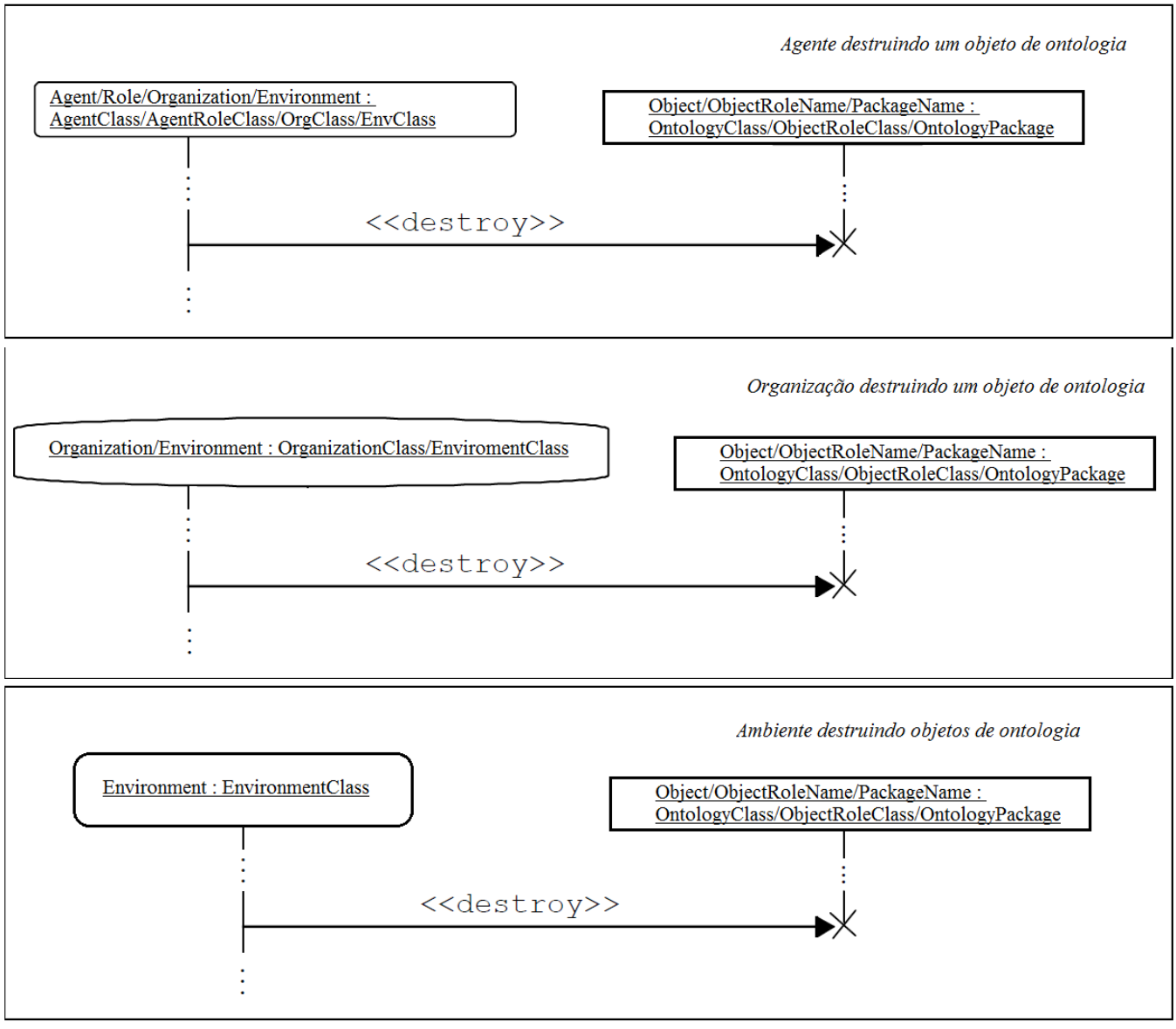
Fonte: adaptado de Silva (2004).

4.4.3.2 Destruindo Objetos de uma Ontologia

A destruição dos objetos de uma ontologia é representada no diagrama de sequência de MAS-ML 2.0 por uma seta de cabeça fechada partindo da entidade destruidora (agente, organização ou ambiente) para o objeto que está sendo destruído, adornada com o estereótipo *<<destroy>>* e acompanhada de uma mensagem de destruição do objeto. O ciclo de vida do objeto termina quando a mensagem rotulada com o *<<destroy>>* é recebida, sendo indicada pela marcação visual de um grande X na entidade destruída.

Ao destruir os objetos de uma ontologia pode-se utilizar o *pathname* completo do objeto, quando for necessário mencionar o nome, o papel do objeto que está sendo destruído, o pacote, a organização e o ambiente nos quais o objeto reside, ou pode-se utilizar o *pathname* simples do objeto, quando for necessário mencionar apenas o nome, o papel e o pacote do objeto destruído.

Figura 15 – Entidades de MAS-ML Destruindo Objetos de Ontologia.



Fonte: adaptado de Silva (2004).

4.4.3.3 Criando um Papel de Ontologia

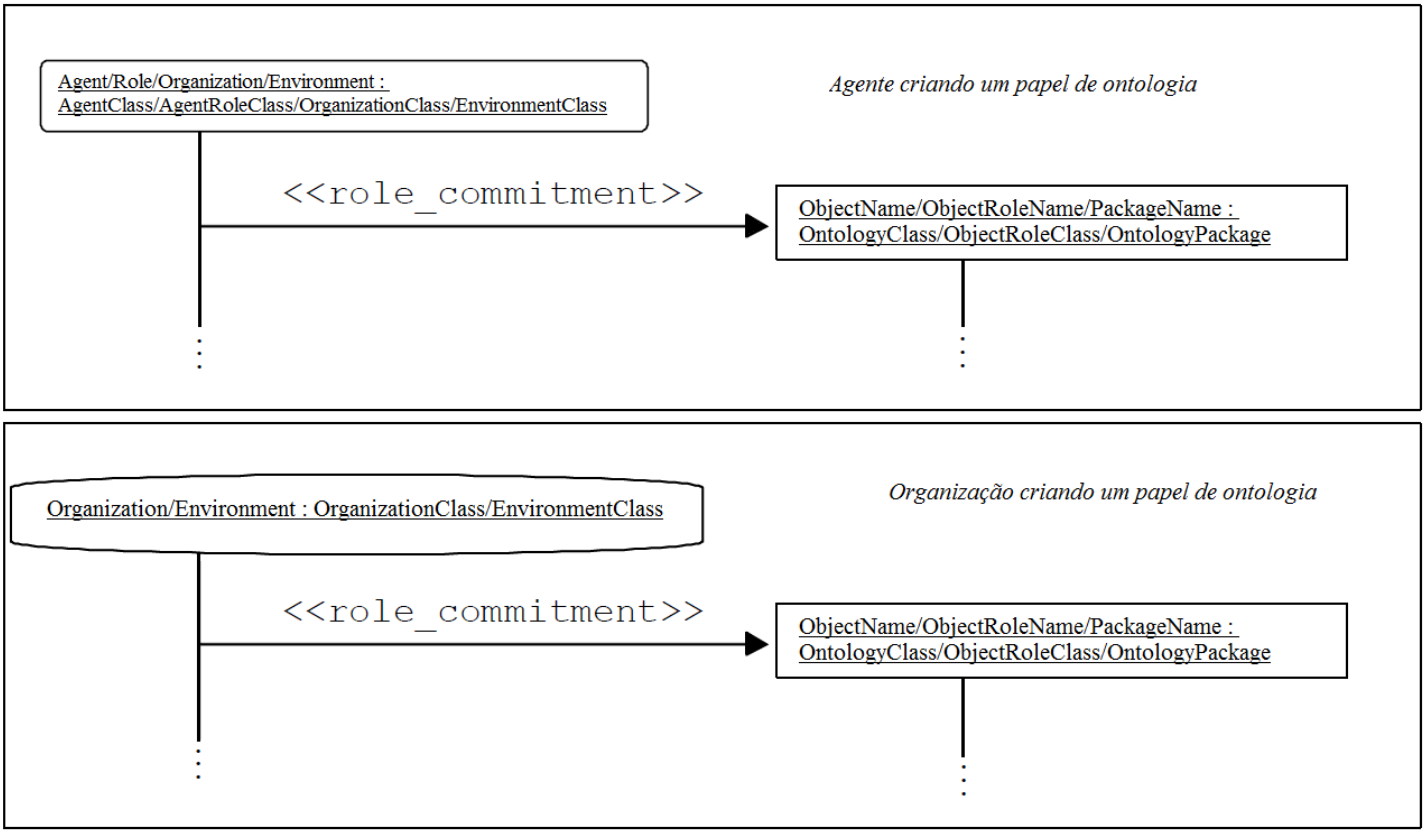
De acordo com Silva (2004, p. 104), os estereótipos <<*create*>> e <<*destroy*>> não conseguem representar a criação e destruição de um papel (papel de objeto e papel do agente), pois com esses estereótipos só é possível representar os processos dinâmicos primitivos de criação e destruição de objetos, agentes, organizações e ambientes. Desta forma, Silva (2004) criou cinco novos estereótipos e os associou à mensagens a fim de representar: um objeto, agente ou organização que está se comprometendo (<<*role_commitment*>>) ou cancelando (<<*role_cancel*>>) um papel; um agente ou uma organização que está alterando seus papéis (<<*role_change*>>) e um agente ou organização que está alterando o estado de um papel – ativo (<<*role_activate*>>) ou inativo (<<*role_deactivate*>>).

Neste contexto, utilizamos o estereótipo <<*role_commitment*>>, criado por Silva (2004, p. 104), associado a uma mensagem, para representar um objeto de ontologia que está se comprometendo com um papel de ontologia. Quando uma instância de papel de ontologia é criada, essa instância é sempre associada a um objeto de ontologia. A entidade que envia a mensagem é aquela que está criando o papel, e a entidade que recebe a mensagem é aquela que exercerá o papel.

Quando um objeto de ontologia se compromete com um novo papel de ontologia, ele não necessariamente deixa de exercer os papéis que já estava exercendo, pelo contrário: uma nova instância de papel é criada e associada ao objeto. O ciclo de vida de um papel de objeto de ontologia começa com o recebimento da mensagem estereotipada como *role_commitment*, indicando que uma nova instância de papel foi criada.

De acordo com Silva (2004, p. 105), as entidades que podem criar um papel de objeto são sempre agentes ou organizações. Assim, seguiremos a mesma restrição neste trabalho: a entidade que envia uma mensagem rotulada com o estereótipo <<*role_commitment*>> a um objeto de ontologia, indicando que ele está assumindo um papel de ontologia, deve ser sempre um agente ou uma organização.

Figura 16 – Entidades de MAS-ML Criando um Papel de Ontologia.



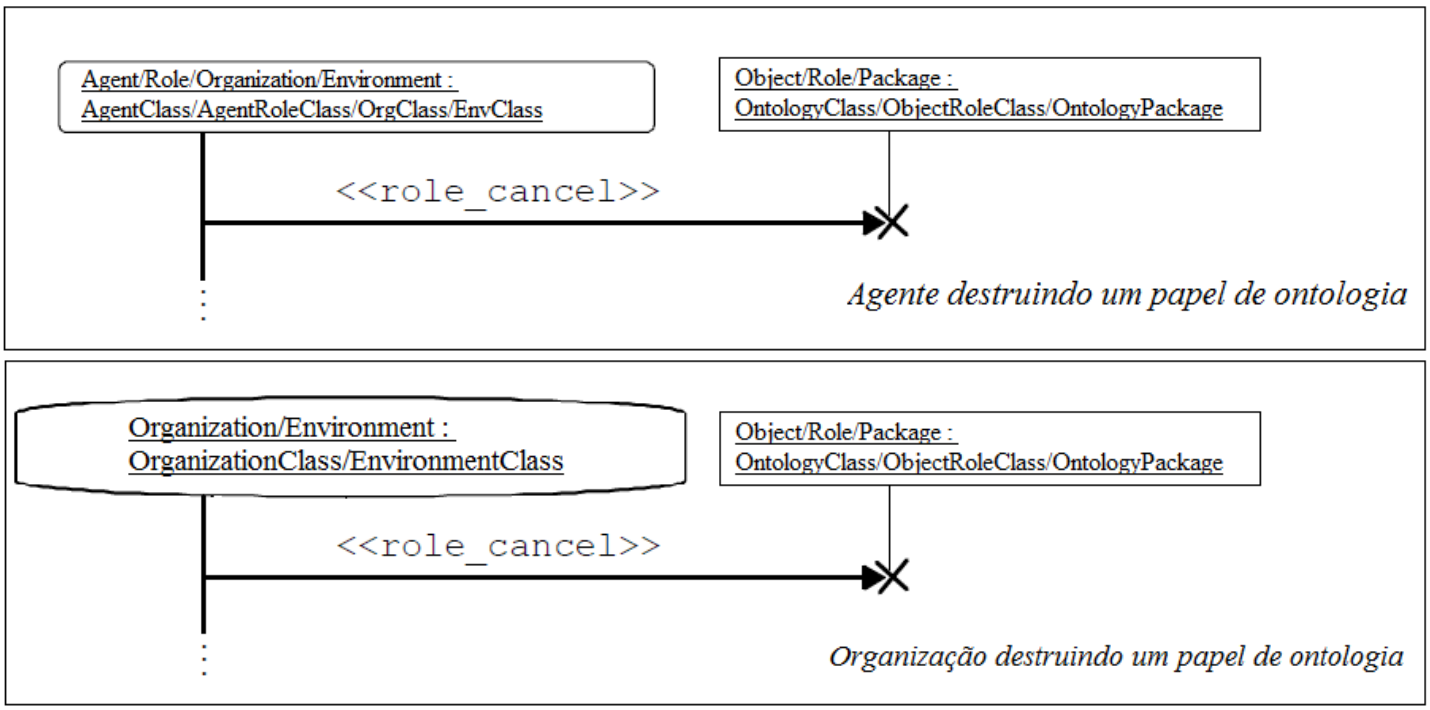
Fonte: adaptado de Silva (2004, p. 106).

4.4.3.4 Destruindo um Papel de Ontologia

O estereótipo <<*role_cancel*>> foi criado por Silva (2004, p. 106) para representar um papel sendo cancelado, ou seja, destruído. Assim, o compromisso entre a entidade e o papel é cancelado, e a entidade deixa de exercer o papel.

Neste contexto, utilizamos o estereótipo <<*role_cancel*>>, associado a uma mensagem, para representar um papel de ontologia que está sendo cancelado (destruído). O ato de cancelar (destruir) um papel de objeto de ontologia só pode ser executado por um agente ou uma organização. O ciclo de vida do papel que está sendo destruído termina com o recebimento da mensagem rotulada com o <<*role_cancel*>> e recebe a marcação visual de um grande X.

Figura 17 – Entidades de MAS-ML Destruindo um Papel de Ontologia.



Fonte: adaptado de Silva (2004, p. 106).

4.4.3.5 Representação de Entidades Obtendo uma Ontologia

Conforme visto na subseção 4.2.2.1 , criamos o estereótipo <<*obtain_ontology*>> e o associamos ao relacionamento *aggregation* para representar, nos diagramas estáticos (classe, organização e papel), um agente ou organização obtendo uma ontologia.

O ato de um agente ou uma organização obter uma ontologia significa que eles estão agregando instâncias dos elementos da ontologia (classes, papéis e pacotes). No diagrama de sequência utilizamos o mesmo estereótipo <<*obtain_ontology*>>, associado a uma mensagem, para representar dinamicamente um agente ou organização obtendo uma ontologia. Nesse caso, tal ato pode ser representado de quatro maneiras:

a) o próprio agente ou organização cria a ontologia e, portanto pode utilizá-la. O ato de criar uma ontologia foi representado na subseção 4.4.3.1 Já o ato de utilizá-la é representado por uma seta de cabeça fechada adornada com o estereótipo <<*obtain_ontology*>> e associada a uma mensagem da entidade (agente ou organização) para o objeto, indicando que o agente ou organização está invocando métodos de objetos de uma ontologia, conforme pode ser visto no primeiro bloco da Figura 18.

b) a ontologia é definida em um papel de agente. Neste caso, quando um agente entrar em um ambiente ou em uma organização e assumir o papel que define a

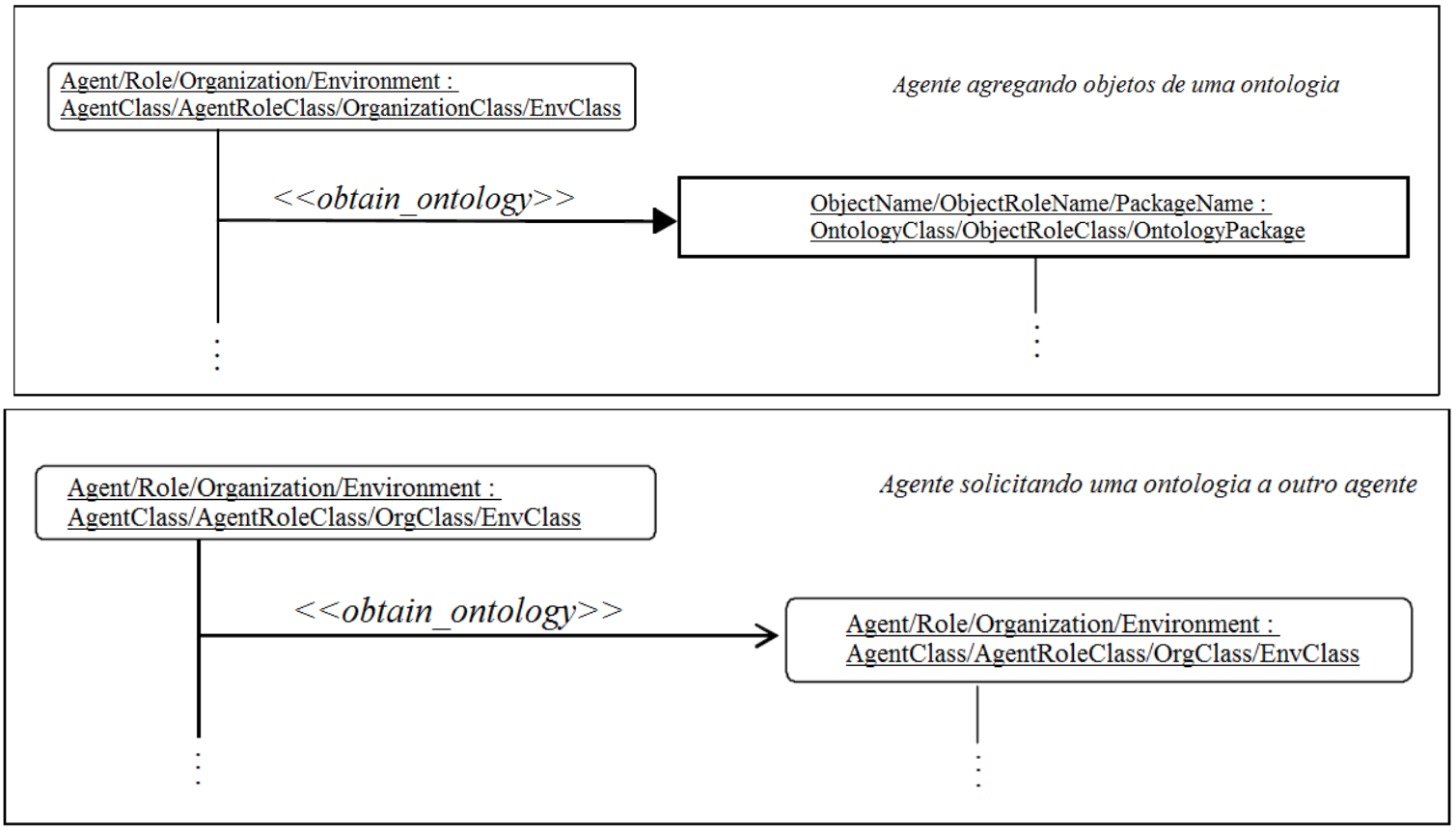
ontologia, automaticamente a ontologia estará disponível para o agente, não sendo necessário que o agente crie ou solicite uma ontologia à outra entidade. Nesse caso, a representação de um papel de agente que define uma ontologia pode ser modelada no diagrama de papel.

c) um agente ou organização solicita uma ontologia a outro agente ou organização. Neste caso, tal ato é representando por uma seta de cabeça aberta adornada com o estereótipo <<*obtain_ontology*>> e associada a uma mensagem trocada entre agentes, indicando que um agente está solicitando uma ontologia a outro agente, conforme pode ser visto no segundo bloco da Figura 18.

d) o agente solicita uma ontologia ao ambiente no qual está habitando. Neste caso, tal ato é representado por uma seta de cabeça fechada adornada com o estereótipo <<*obtain_ontology*>> e associada a uma mensagem do agente para o ambiente.

O estereótipo <<*obtain_ontology*>> pode ser utilizado para encapsular a complexidade de um agente ou organização que esteja obtendo uma ontologia para desempenhar seu papel.

Figura 18 – Obtendo uma Ontologia.



Fonte: adaptada de Silva (2004).

Nesta seção foi apresentada uma extensão à linguagem MAS-ML 2.0 para a modelagem de ontologias. Inicialmente, o conceito das entidades *Class* e *ObjectRoleClass* foram adaptados para incluir a definição de ontologia, e o conceito da entidade *Package* foi introduzido no *framework* conceitual TAO. Em seguida, apresentamos a extensão à linguagem MAS-ML a partir da sintaxe abstrata, passando pela representação dos estereótipos criados e inseridos no metamodelo da linguagem, definindo as associações entre as metaclasses que utilizam esses novos estereótipos e as demais metaclasses da linguagem, e mostrando como os elementos de ontologia estão representados nos diagramas de classes, organização, papel e sequência.

Na próxima seção, é mostrado um estudo de caso, no qual demonstramos o uso da extensão proposta à MAS-ML. Nele, foram identificadas e modeladas classes de ontologia para diferentes tipos de agentes de um sistema multiagente.

5 UTILIZANDO A EXTENSÃO

Nesta seção, apresentamos um estudo de caso, o qual foi utilizado para analisar a extensão proposta neste trabalho à linguagem MAS-ML. Esse estudo de caso consiste em utilizar MAS-ML 2.0 estendida para modelar classes de ontologia em um sistema multiagente. Os diagramas apresentados nesta seção foram criados com o software *Astah Community 6.7* (disponível em http://astah.net/download) que é baseado na linguagem UML, e adequados para MAS-ML 2.0 através de editores de imagens.

5.1 Estudo de caso do Moodle

Um Ambiente Virtual de Aprendizagem (AVA) é um ambiente complexo, muito utilizado na Educação à Distância (EAD), onde são disponibilizados materiais didáticos e ferramentas de comunicação para a formação de alunos sob a supervisão de professores e tutores (PERAYA; 2002). Neste contexto, é possível encontrar vários AVAs na literatura de EAD, dentre eles o *Modular Object-Oriented Dynamic Learning Environment* (*Moodle)*, um ambiente amplamente utilizado em cursos de EAD. O *Moodle* é um software livre, disponibilizado em uma versão padrão que pode ser customizada para o uso em um determinado curso/instituição de ensino.

As ferramentas computacionais autônomas, como por exemplo, sistemas multiagentes, estão sendo utilizadas em ambientes de EAD para acompanhar os diversos problemas que podem surgir no cenário complexo de um AVA, tais como: dificuldades nas interações e no desenvolvimento de trabalhos em grupo *online*.

Neste contexto, em Gonçalves *et al*. (2013) é apresentado o desenvolvimento de um sistema multiagente para o acompanhamento do ensino através do *Moodle*, utilizando técnicas da Engenharia de Software orientada a agentes. No sistema apresentado, foram definidos seis tipos de agentes, onde cada agente desempenha um papel dentro do *Moodle*:

a) Agente Companheiro de Aprendizagem – responsável por acompanhar o processo de aprendizagem de um aluno durante o curso, enviando ao mesmo, mensagens de apoio ou reforço dependendo de seu desempenho, e sugerindo atividades complementares, como: participação em fóruns e chats.
b) Agente Pedagógico – responsável por acompanhar o usuário nos diferentes cursos, disciplinas ou projetos em que este participar, e oferecer dicas e sugestões, como por exemplo, quais atividades devem ser realizadas.

c) Agente Acompanhante de Tutores – responsável por monitorar a participação dos tutores nos fóruns e na postagem de materiais, e dar dicas de como o tutor pode obter melhor desempenho ao ministrar uma disciplina.
d) Agente Buscador de Informações – responsável por enviar ao aluno, conteúdo didático digital que complemente o conteúdo de uma determinada disciplina. Este agente realiza seu trabalho em conjunto com o agente companheiro de aprendizagem, e à medida que o aluno tem rendimento abaixo do esperado, o agente interfere e oferece material complementar na disciplina em questão.
e) Agente Formador de Grupos – responsável por sugerir a formação de grupos de estudo de acordo com a afinidade de temas ou de perfis dos alunos.
f) Agente Auxiliar de Usabilidade – responsável por dar dicas de como os usuários podem fazer melhor uso dos recursos do *Moodle*.

Neste trabalho, modelamos ontologias para cada tipo de agente descrito acima. Essas classes de ontologia são utilizadas por estes agentes dentro do SMA, e esse modelo foi utilizado como um estudo de caso, no qual analisamos como a linguagem MAS-ML se comporta na modelagem de ontologias.

5.2 Modelagem das Ontologias

Nesta subseção, apresentamos a modelagem das classes de ontologia através da ferramenta *Astah*, já que MAS-ML ainda não possui uma ferramenta que contemple a extensão proposta neste trabalho. Inicialmente, foram definidos os conceitos necessários em cada ontologia, bem como, foi definido como esses conceitos se relacionam. Posteriormente, é mostrado os diagramas de classes, sequência, organização e papel de cada ontologia.

5.2.1 Ontologia para o Agente Companheiro de Aprendizagem

O objetivo desta ontologia é oferecer ao agente os conceitos necessários para que ele exerça o papel de companheiro de aprendizagem de um aluno durante um curso.

5.2.1.1 Identificação das Classes

a) *Curso*: descreve para o agente o que é um Curso e quais são suas propriedades.
b) *Aluno*: descreve para o agente o que é um Aluno e quais são suas propriedades.
c) *Disciplina*: descreve para o agente o que é uma Disciplina e suas propriedades.
d) *Processo de Aprendizagem*: descreve para o agente o que é um Processo de Aprendizagem e suas características.

e) *Avaliação de Desempenho do Aluno*: descreve para o agente o que é uma avaliação de desempenho para que ele possa avaliar o desempenho de um aluno.
f) *Fórum*: descreve para o agente o que é um Fórum e como um Aluno participa dele.
g) *Chat*: descreve para o agente o que é um Chat e como um aluno participa dele.

5.2.1.2 Descrição dos Relacionamentos

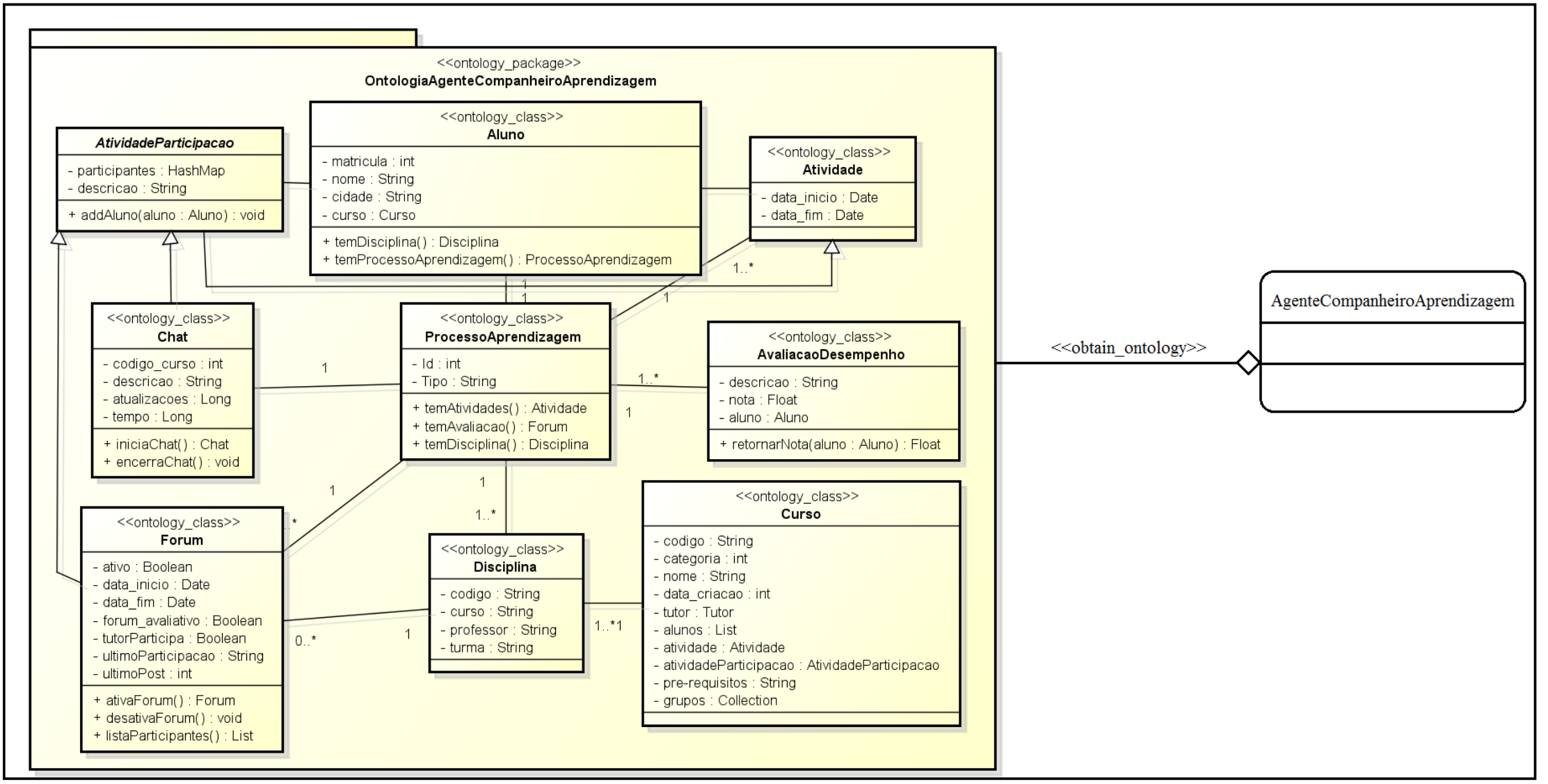
a) Um Curso *tem* Aluno, Disciplina;
b) Um Aluno *tem* Disciplina, Processo de Aprendizagem e Avaliação de Desempenho;
c) Uma Disciplina *tem* Fórum, Chat.
d) Um Aluno *participa* de um Fórum e de um Chat.

5.2.1.3 Diagrama de Classes da Ontologia

Na Figura 19 é ilustrado o diagrama de classes simplificado da ontologia proposta para o agente Companheiro de Aprendizagem. A estrutura interna do agente foi omitida, uma vez que o objetivo desta subseção é modelar as classes/pacote da ontologia.

Figura 19 – Ontologia do Companheiro de Aprendizagem no Diagrama de Classes.

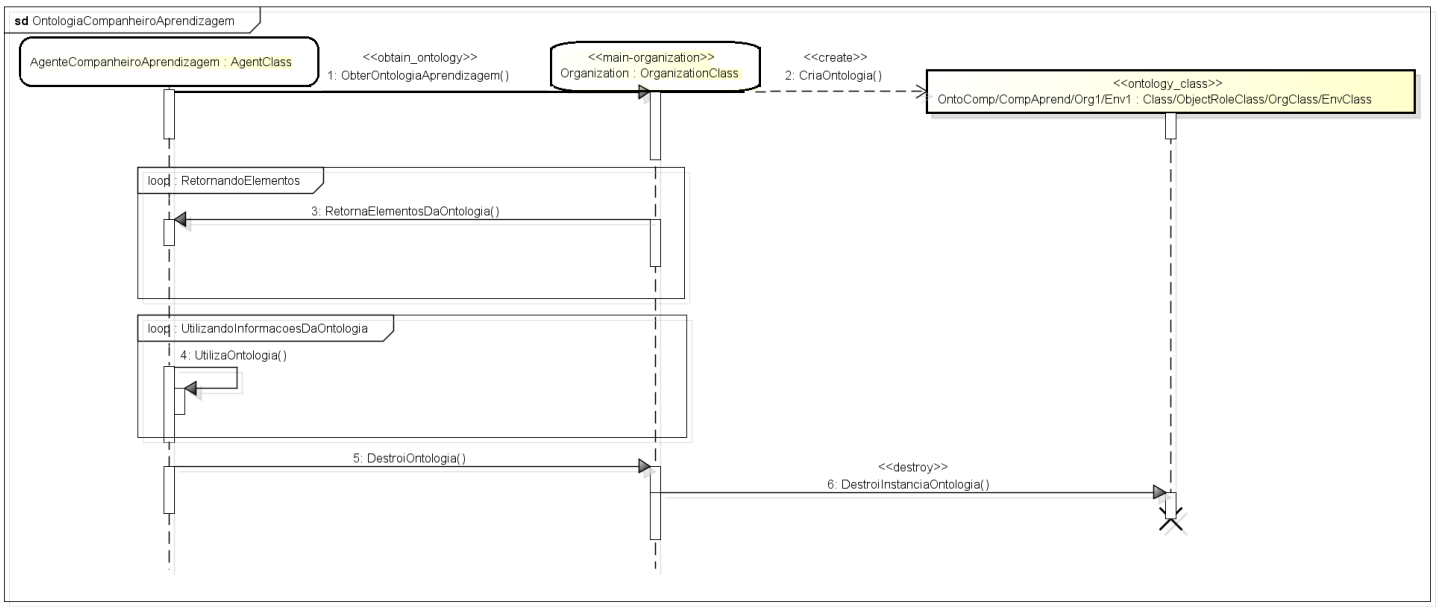
Fonte: elaborada pelo autor.


<<ontology_package>>
OntologiaAgenteCompanheiroAprendizagem
AtividadeParticipacao
- participantes : HashMap
- descricao : String
+ addAluno(aluno : Aluno) : void
<<ontology_class>>
Aluno
- matricula : int
- nome : String
- cidade : String
- curso : Curso
+ temDisciplina() : Disciplina
+ temProcessoAprendizagem() : ProcessoAprendizagem
<<ontology_class>>
Atividade
- data_inicio : Date
- data_fim : Date
<<ontology_class>>
Chat
- codigo_curso : int
- descricao : String
- atualizacoes : Long
- tempo : Long
+ iniciaChat() : Chat
+ encerraChat() : void
<<ontology_class>>
ProcessoAprendizagem
- Id : int
- Tipo : String
+ temAtividades() : Atividade
+ temAvaliacao() : Forum
+ temDisciplina() : Disciplina
<<ontology_class>>
AvaliacaoDesempenho
- descricao : String
- nota : Float
- aluno : Aluno
+ retornarNota(aluno : Aluno) : Float
<<ontology_class>>
Forum
- ativo : Boolean
- data_inicio : Date
- data_fim : Date
- forum_avaliativo : Boolean
- tutorParticipa : Boolean
- ultimoParticipacao : String
- ultimoPost : int
+ ativaForum() : Forum
+ desativaForum() : void
+ listaParticipantes() : List
<<ontology_class>>
Disciplina
- codigo : String
- curso : String
- professor : String
- turma : String
<<ontology_class>>
Curso
- codigo : String
- categoria : int
- nome : String
- data_criacao : int
- tutor : Tutor
- alunos : List
- atividade : Atividade
- atividadeParticipacao : AtividadeParticipacao
- pre-requisitos : String
- grupos : Collection
1..*
1
0..*
1..*1
<<obtain_ontology>>
AgenteCompanheiroAprendizagem

5.2.1.4 Diagrama de Sequência da Ontologia

Na figura a seguir é mostrado, no diagrama de sequência, utilizando a notação de MAS-ML 2.0, a interação entre o agente companheiro de aprendizagem, a organização e a ontologia que ele utiliza para desempenhar seu papel.

Figura 20 – Ontologia do Companheiro de Aprendizagem no Diagrama de Sequência.

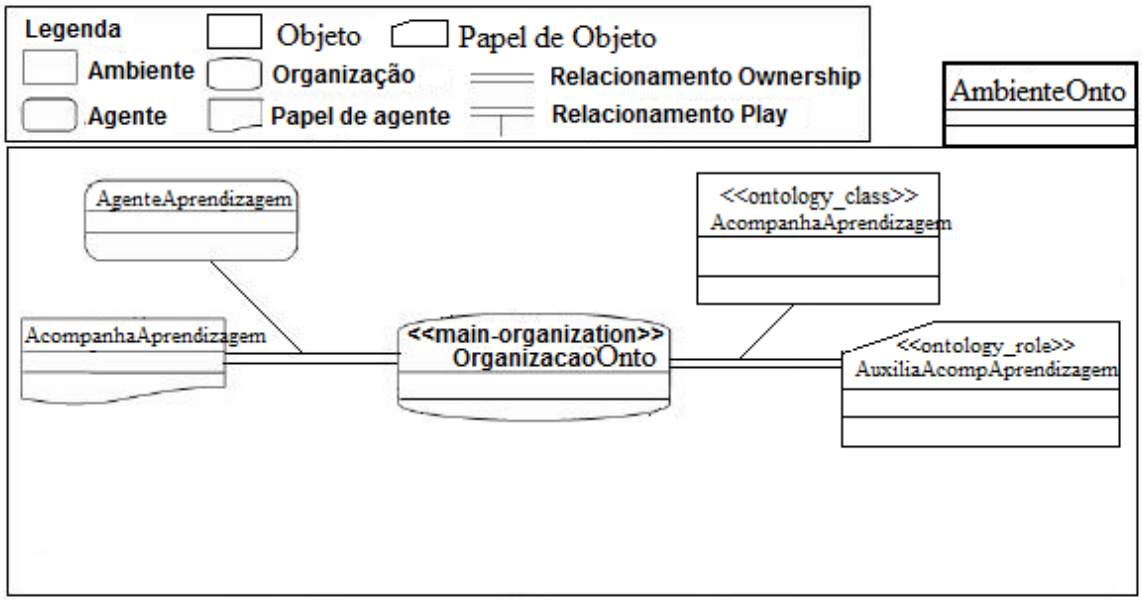


Fonte: elaborada pelo autor.

5.2.1.5 Diagrama de Organização da Ontologia

Na Figura 21 é apresentado o diagrama de organização simplificado da ontologia para o Agente Companheiro de Aprendizagem. Nele são mostrados a organização principal, o agente e seu papel, a ontologia e seu papel.

Figura 21 – Ontologia do Companheiro de Aprendizagem no Diagrama de Organização.

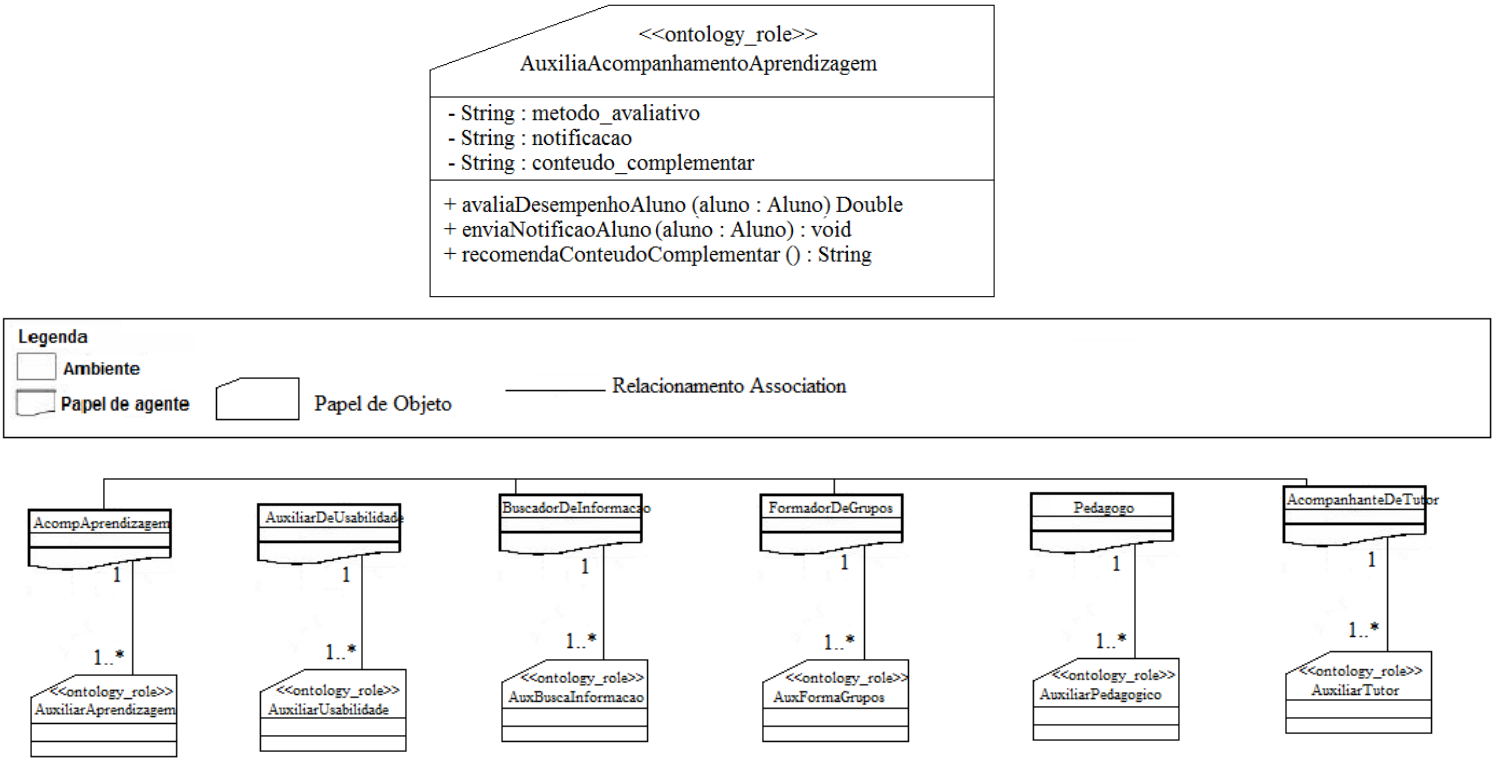


Fonte: adaptado de Gonçalves (2009).

5.2.1.6 Diagrama de Papel da Ontologia

Na Figura 22 é apresentado o diagrama de papel simplificado da ontologia para o Agente Companheiro de Aprendizagem. Primeiro é mostrado a estrutura do papel de ontologia e depois ele é mostrado dentro do referido diagrama.

Figura 22 – Ontologia do Companheiro de Aprendizagem no Diagrama de Papel.



Fonte: elaborada pelo autor.

5.2.2 Ontologia para o Agente Pedagógico

O objetivo desta ontologia é oferecer ao agente os conceitos necessários para que ele desempenhe o papel de agente pedagógico.

5.2.2.1 Identificação das Classes

a) *Objetivo Pedagógico*: descreve para o agente o que é um objetivo pedagógico.
b) *Aluno*: descreve para o agente o que é um aluno e quais são suas propriedades.
c) *Habilidades*: descreve para o agente o que são habilidades para que ele possa trabalhar as habilidades de um aluno.

d) *Disciplina*: descreve para o agente o que é uma disciplina e quais são suas características.
e) *Conteúdo*: descreve para o agente o que é um conteúdo didático e como ele está relacionado a uma disciplina.
f) *Atividade*: descreve para o agente o que é uma atividade e como ela está relacionada a uma disciplina.
g) *Avaliação do Conteúdo*: descreve para o agente o que é uma avaliação de conteúdo para que ele possa avaliar como um determinado conteúdo foi abordado.
h) *Metodologia de Ensino*: descreve para o agente o que é uma metodologia de ensino, quais são suas características (se é formada de estudos de caso, desafios, perguntas e respostas etc.), e como o agente pode aplicá-la.

5.2.2.2 Descrição dos Relacionamentos

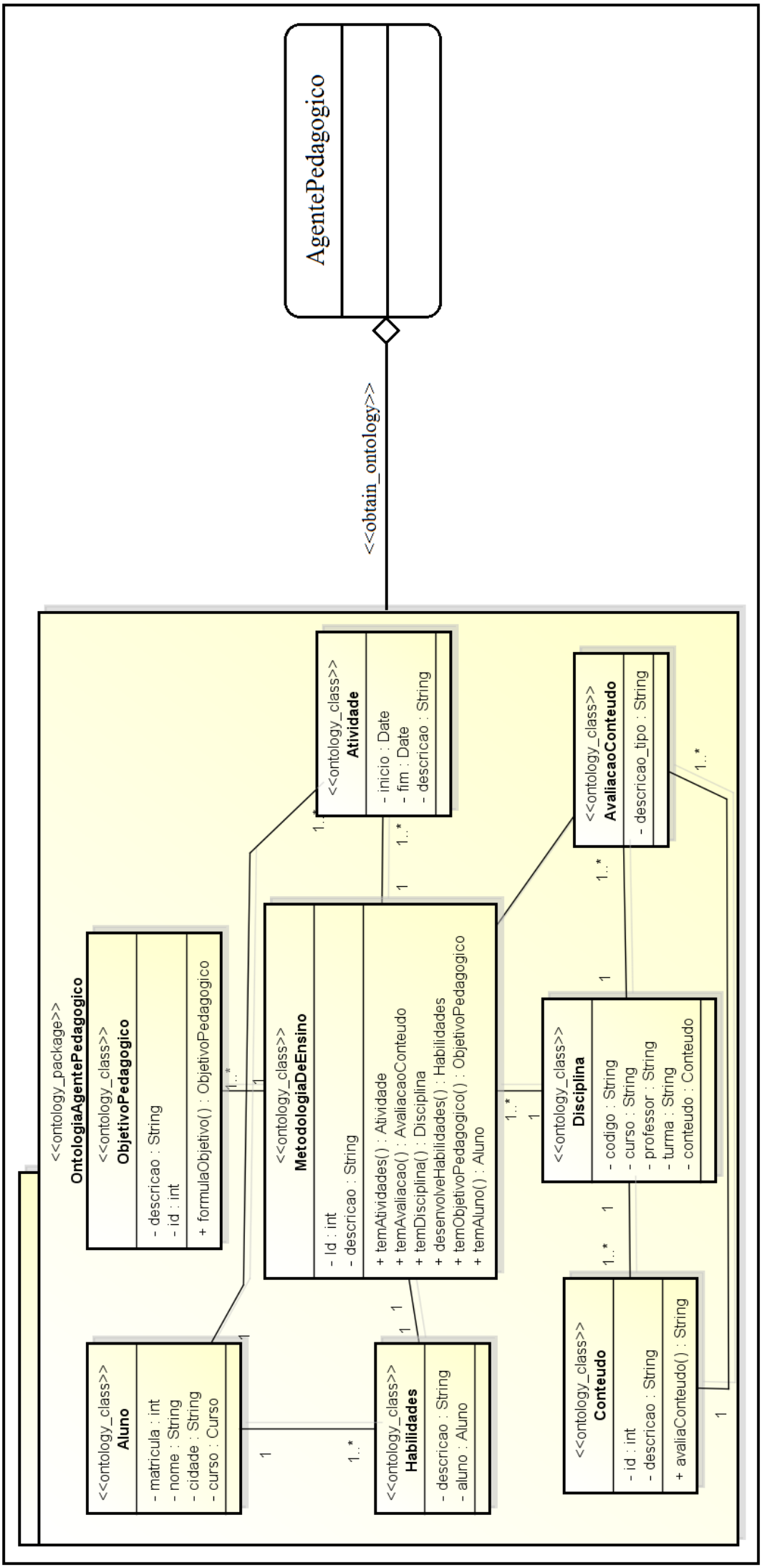
a) Um Aluno *tem* Disciplina e Habilidades.
b) Uma Disciplina *tem* Conteúdo, Metodologia de Ensino, Atividade e Avalição do Conteúdo.
c) Uma Metodologia de Ensino *tem um* Objetivo Pedagógico.

5.2.2.3 Diagrama de Classes da Ontologia

Na Figura 23 é ilustrado o diagrama de classes simplificado da ontologia proposta para o Agente Pedagógico. A estrutura interna do agente foi omitida, uma vez que o objetivo desta subseção é modelar as classes/pacote da ontologia.

Figura 23 – Ontologia do Agente Pedagógico no Diagrama de Classes.

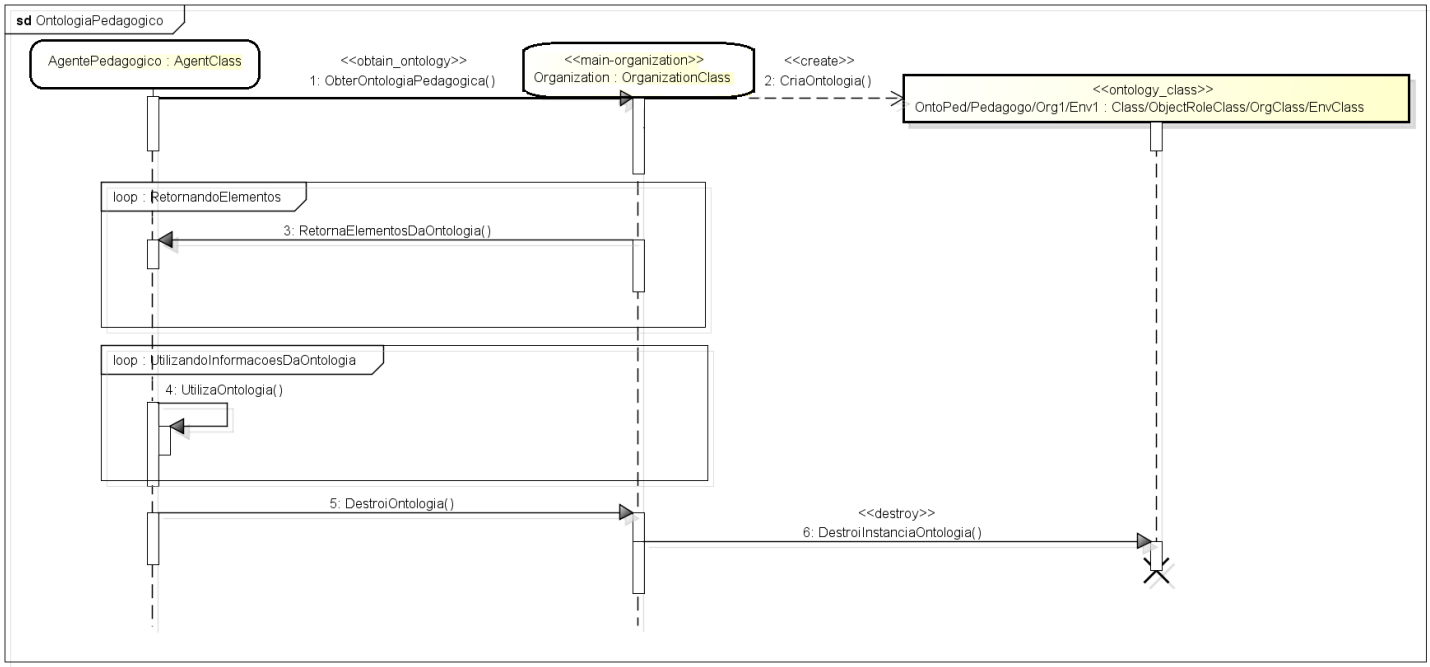
Fonte: elaborada pelo autor.


AgentePedagogico
<<obtain_ontology>>
<<ontology_package>>
OntologiaAgentePedagogico
<<ontology_class>>
ObjetivoPedagogico
- descricao : String
- id : int
+ formulaObjetivo() : ObjetivoPedagogico
<<ontology_class>>
Atividade
- inicio : Date
- fim : Date
- descricao : String
<<ontology_class>>
MetodologiaDeEnsino
- Id : int
- descricao : String
+ temAtividades() : Atividade
+ temAvaliacao() : AvaliacaoConteudo
+ temDisciplina() : Disciplina
+ desenvolveHabilidades() : Habilidades
+ temObjetivoPedagogico() : ObjetivoPedagogico
+ temAluno() : Aluno
<<ontology_class>>
AvaliacaoConteudo
- descricao_tipo : String
<<ontology_class>>
Aluno
- matricula : int
- nome : String
- cidade : String
- curso : Curso
<<ontology_class>>
Disciplina
- codigo : String
- curso : String
- professor : String
- turma : String
- conteudo : Conteudo
<<ontology_class>>
Habilidades
- descricao : String
- aluno : Aluno
<<ontology_class>>
Conteudo
- id : int
- descricao : String
+ avaliaConteudo() : String

5.2.2.4 Diagrama de Sequência da Ontologia

Na figura a seguir é mostrado, no diagrama de sequência, utilizando a notação de MAS-ML 2.0, a interação entre o agente pedagógico, a organização e a ontologia que ele utiliza para desempenhar seu papel.

Figura 24 – Ontologia do Agente Pedagógico no Diagrama de Sequência.

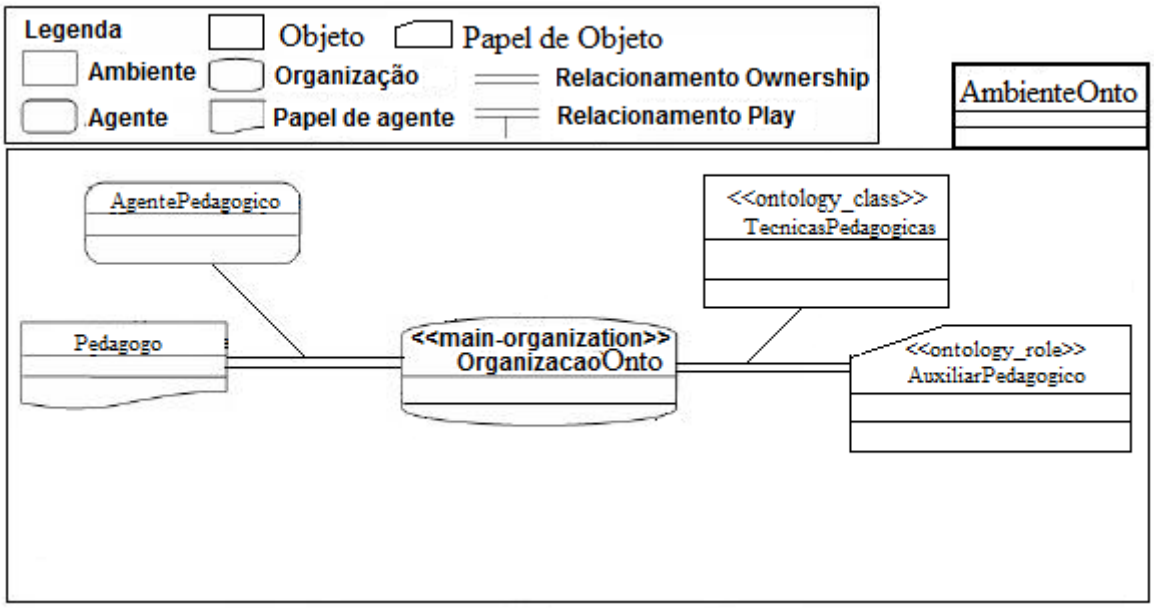


Fonte: elaborada pelo autor.

5.2.2.5 Diagrama de Organização da Ontologia

Na Figura 25 é apresentado o diagrama de organização simplificado da ontologia para o Agente Pedagógico. Nele são mostrados a organização principal, o agente e seu papel, a ontologia e seu papel.

Figura 25 – Ontologia do Agente Pedagógico no Diagrama de Organização.

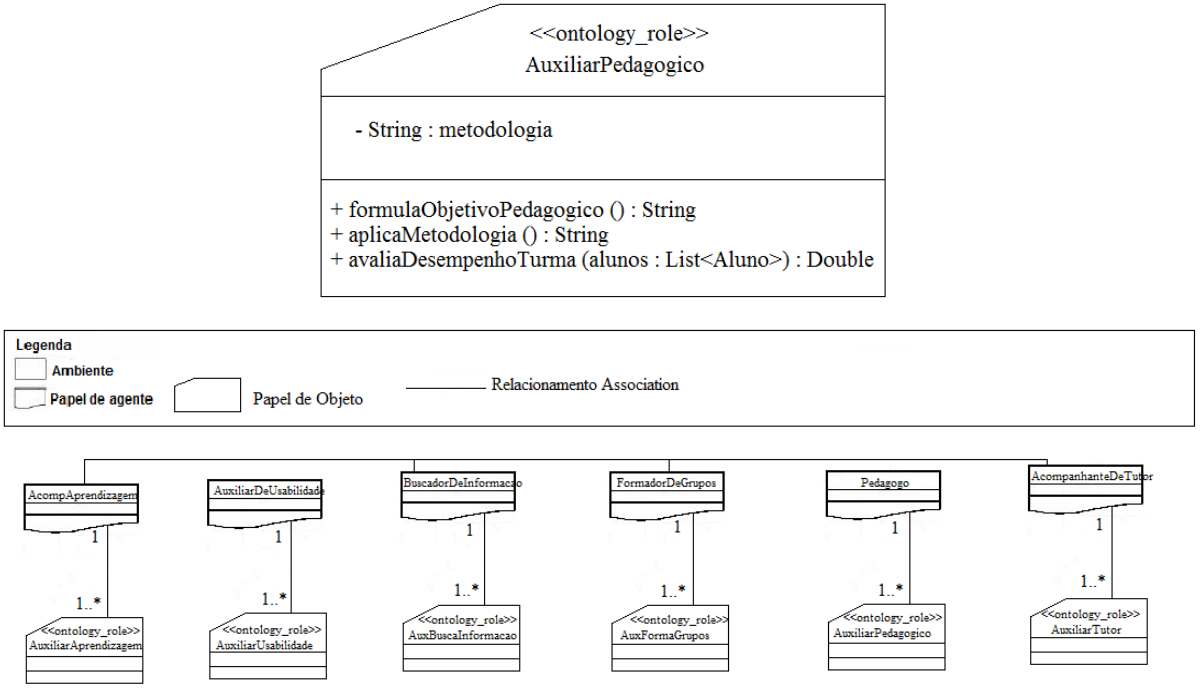


Fonte: adaptado de Gonçalves (2009).

5.2.2.6 Diagrama de Papel da Ontologia

Na Figura 26 é apresentado o diagrama de papel simplificado da ontologia para o Agente Pedagógico. Primeiro é mostrado a estrutura do papel de ontologia e depois ele é mostrado dentro do referido diagrama.

Figura 26 – Ontologia do Agente Pedagógico no Diagrama de Papel.



Fonte: elaborada pelo autor.

5.2.3 Ontologia para o Agente Acompanhante de Tutores

O objetivo desta ontologia é oferecer ao agente a descrição dos conceitos necessários para que ele saiba como auxiliar um tutor.

5.2.3.1 Identificação das Classes

a) *Tutor*: descreve para o agente o que é um Tutor e quais são suas funções.
b) *Fórum*: descreve para o agente o que é um Fórum e como um Tutor participa dele.
c) *Avaliação de Desempenho*: descreve para o agente o que é uma avaliação de desempenho de um Tutor, para que ele possa avaliar e sugerir dicas de como um Tutor pode aumentar seu desempenho em uma determinada disciplina.
d) *Disciplina*: descreve para o agente o que é uma disciplina e quais são suas características.
e) *Oferecer Conteúdo*: descreve para o agente o que é conteúdo (didático) e como ele pode oferecer a um Tutor.
f) *Curso*: descreve para o agente o que é um Curso e quais são suas propriedades.

5.2.3.2 Descrição dos Relacionamentos

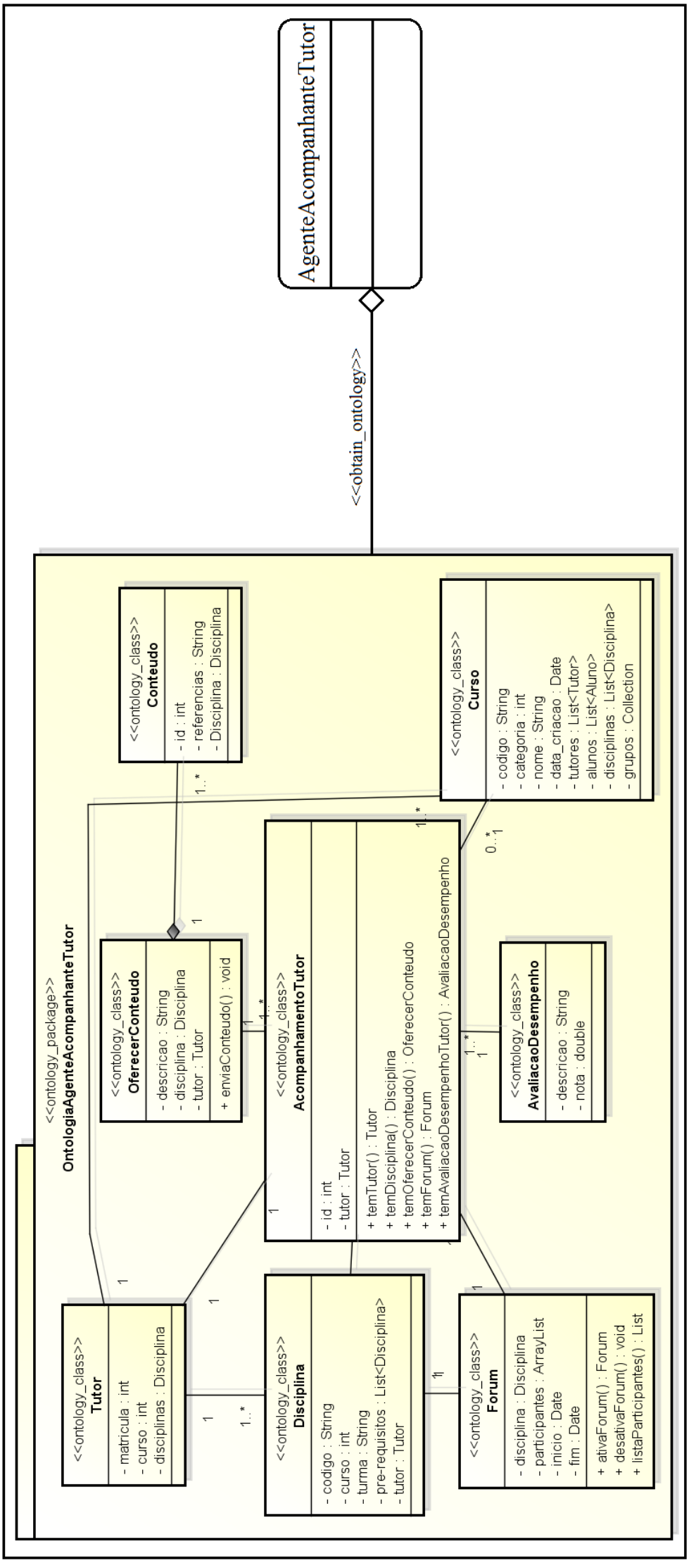
a) Uma Disciplina *tem* Conteúdo e Tutor.
b) Um Tutor *participa* de um Curso, de um Fórum e *tem uma* Avaliação de Desempenho.

5.2.3.3 Diagrama de Classes da Ontologia

Na Figura 27 é ilustrado o diagrama de classes simplificado da ontologia proposta para o Agente Acompanhante de Tutores. A estrutura interna do agente foi omitida, uma vez que o objetivo desta subseção é modelar as classes/pacote da ontologia.

Figura 27 – Ontologia do Agente Acompanhante de Tutores no Diagrama de Classes.

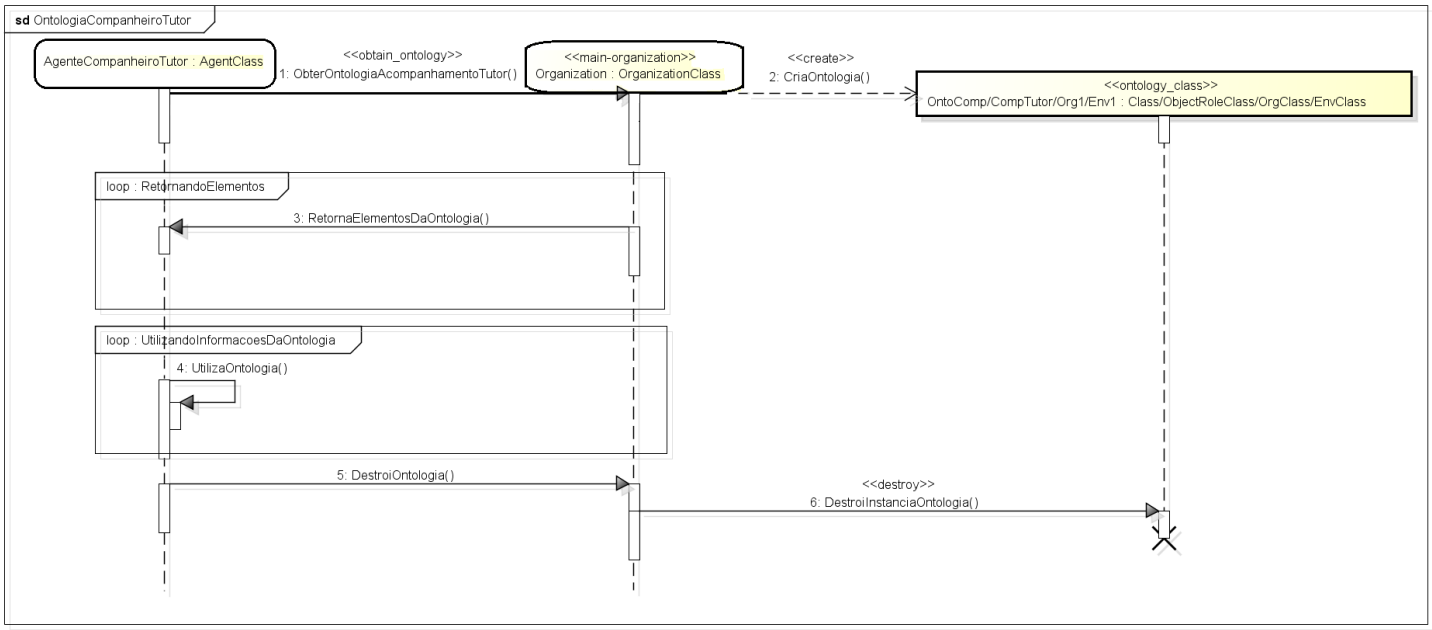
Fonte: elaborada pelo autor.


AgenteAcompanhanteTutor
<<obtain_ontology>>
<<ontology_package>>
OntologiaAgenteAcompanhanteTutor
<<ontology_class>>
Conteudo
- id : int
- referencias : String
- Disciplina : Disciplina
<<ontology_class>>
Curso
- codigo : String
- categoria : int
- nome : String
- data_criacao : Date
- tutores : List<Tutor>
- alunos : List<Aluno>
- disciplinas : List<Disciplina>
- grupos : Collection
<<ontology_class>>
OferecerConteudo
- descricao : String
- disciplina : Disciplina
- tutor : Tutor
+ enviaConteudo() : void
<<ontology_class>>
AcompanhamentoTutor
- id : int
- tutor : Tutor
+ temTutor() : Tutor
+ temDisciplina() : Disciplina
+ temOferecerConteudo() : OferecerConteudo
+ temForum() : Forum
+ temAvaliacaoDesempenhoTutor() : AvaliacaoDesempenho
<<ontology_class>>
AvaliacaoDesempenho
- descricao : String
- nota : double
<<ontology_class>>
Tutor
- matricula : int
- curso : int
- disciplinas : Disciplina
<<ontology_class>>
Disciplina
- codigo : String
- curso : int
- turma : String
- pre-requisitos : List<Disciplina>
- tutor : Tutor
<<ontology_class>>
Forum
- disciplina : Disciplina
- participantes : ArrayList
- inicio : Date
- fim : Date
+ ativaForum() : Forum
+ desativaForum() : void
+ listaParticipantes() : List
1
1..*
1
1..*
1
1..*
1
1..*
1
1..*
0..*
1
1
1..*
1
1
1
1
1

5.2.3.4 Diagrama de Sequência da Ontologia

Na figura a seguir é mostrado, no diagrama de sequência, utilizando a notação de MAS-ML 2.0, a interação entre o agente acompanhante de tutores, a organização e a ontologia que ele utiliza para desempenhar seu papel.

Figura 28 – Ontologia do Acompanhante de Tutores no Diagrama de Sequência.

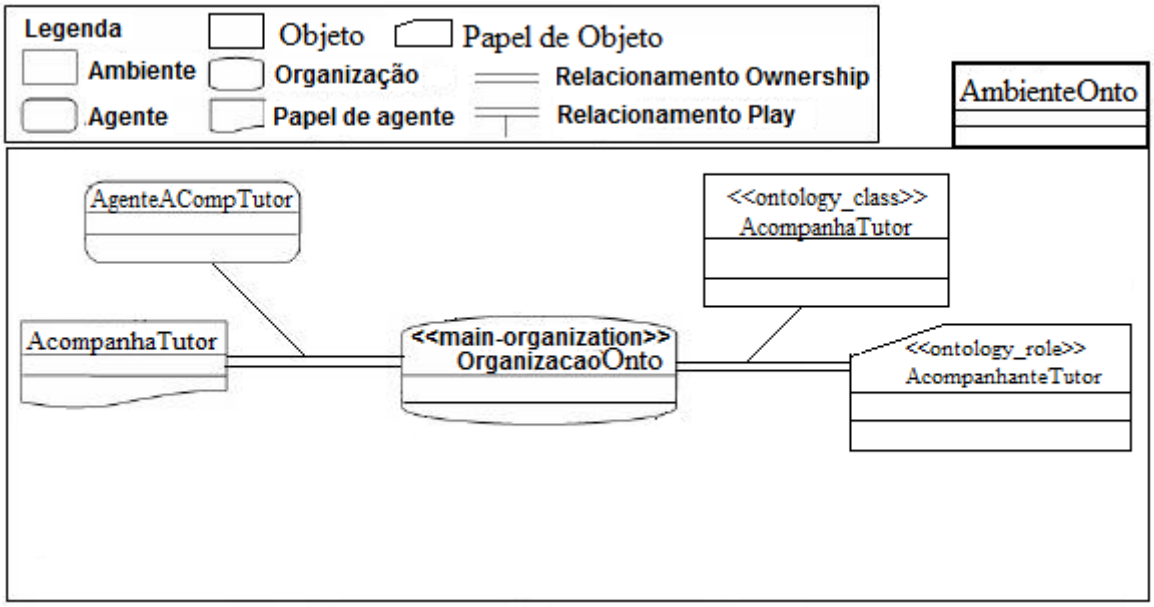


Fonte: elaborada pelo autor.

5.2.3.5 Diagrama de Organização da Ontologia

Na Figura 29 é apresentado o diagrama de organização simplificado da ontologia para o Agente Acompanhante de Tutores. Nele são mostrados a organização principal, o agente e seu papel, a ontologia e seu papel.

Figura 29 – Ontologia do Acompanhante de Tutores no Diagrama de Organização.

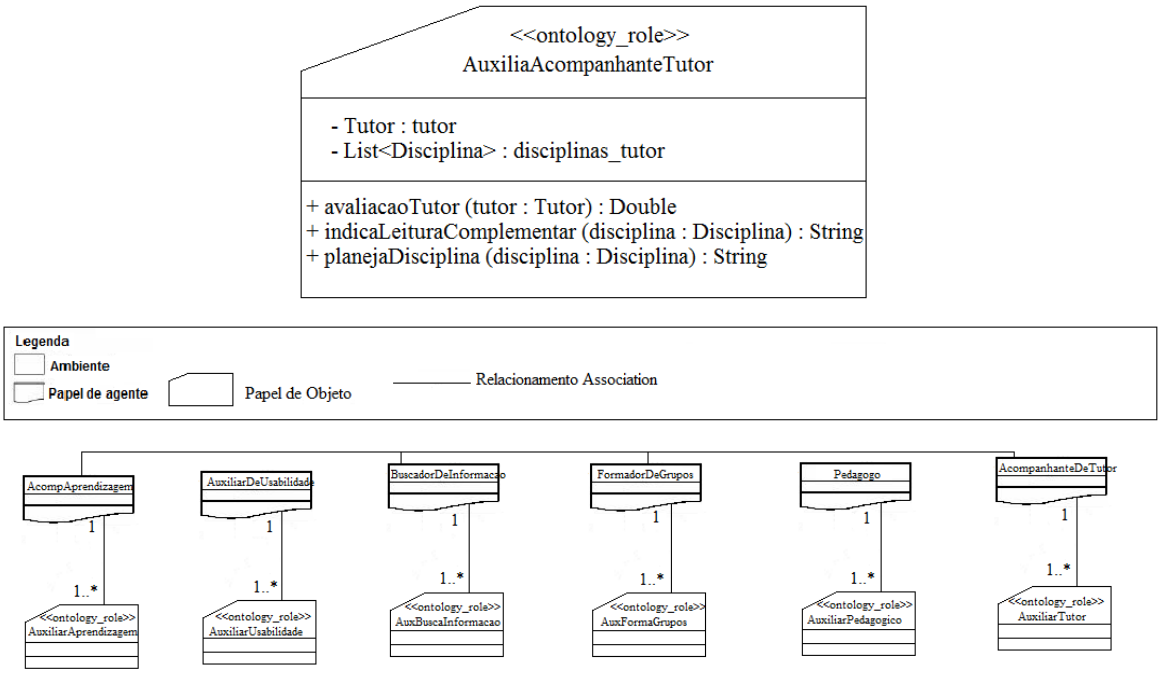


Fonte: adaptado de Gonçalves (2009).

5.2.3.6 Diagrama de Papel da Ontologia

Na Figura 30 é apresentado o diagrama de papel simplificado da ontologia para o Agente Acompanhante de Tutores. Primeiro é mostrado a estrutura do papel de ontologia e depois ele é mostrado dentro do referido diagrama.

Figura 30 – Ontologia do Acompanhante de Tutores no Diagrama de Papel.



Fonte: elaborada pelo autor.

5.2.4 Ontologia para o Agente Buscador de Informações

O objetivo desta ontologia é oferecer ao agente a descrição dos conceitos necessários para que ele saiba como avaliar o rendimento de um aluno, e a partir disso, buscar e oferecer informações que possa ajudar o aluno a obter um rendimento melhor.

5.2.4.1 Identificação das Classes

a) *Aluno*: descreve para o agente as propriedades de um aluno.
b) *Disciplina*: descreve para o agente o que é uma disciplina, para que assim, ele possa oferecer conteúdos referentes a uma determinada disciplina.
c) *Avalição do Aluno*: descreve para o agente o que é uma avaliação de desempenho, para que assim ele possa propor conteúdo didático complementar ao aluno com rendimento abaixo do esperado.
d) *Buscar Informação*: descreve para o agente como ele pode buscar informações a partir do que ele sabe sobre um aluno e seu rendimento em uma determinada disciplina.
e) *Conteúdo Didático*: descreve para o agente o que é são conteúdos didáticos.
f) *Oferecer Material Didático*: descreve para o agente o que é "material" e como ele pode enviar materiais para um aluno.

5.2.4.2 Descrição dos Relacionamentos

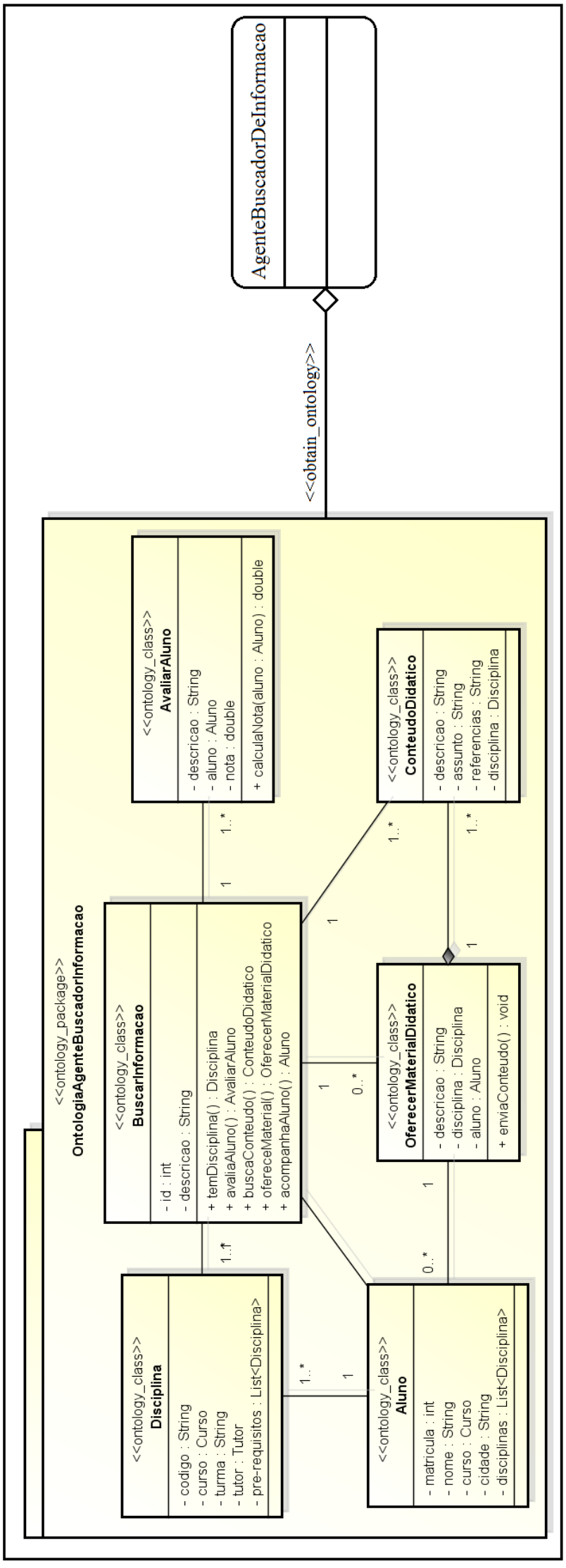
a) Um Aluno *tem* Disciplina e Avaliação do Aluno.
b) Uma Disciplina *tem* Conteúdo Didático.
c) Um Conteúdo Didático *é parte de* Buscar Informação e de Oferecer Material Didático;

5.2.4.3 Diagrama de Classes da Ontologia

Na Figura 31 é ilustrado o diagrama de classes simplificado da ontologia proposta para o Agente Buscador de Informações. A estrutura interna do agente foi omitida, uma vez que o objetivo desta subseção é modelar as classes/pacote da ontologia.

Figura 31 – Ontologia do Agente Buscador de Informações no Diagrama de Classes.

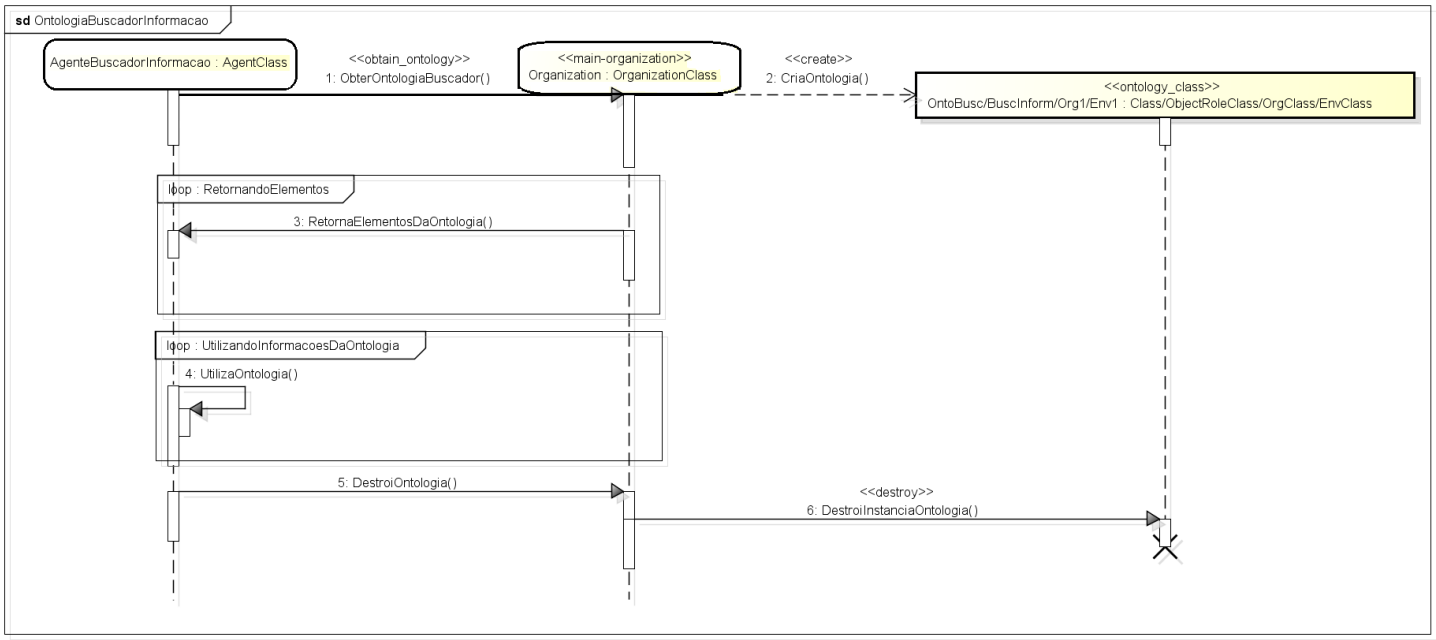
Fonte: elaborada pelo autor.


AgenteBuscadorDeInformacao
<<obtain_ontology>>
<<ontology_package>>
OntologiaAgenteBuscadorInformacao
<<ontology_class>>
AvaliarAluno
- descricao : String
- aluno : Aluno
- nota : double
+ calculaNota(aluno : Aluno) : double
<<ontology_class>>
ConteudoDidatico
- descricao : String
- assunto : String
- referencias : String
- disciplina : Disciplina
<<ontology_class>>
BuscarInformacao
- id : int
- descricao : String
+ temDisciplina() : Disciplina
+ avaliaAluno() : AvaliarAluno
+ buscaConteudo() : ConteudoDidatico
+ ofereceMaterial() : OferecerMaterialDidatico
+ acompanhaAluno() : Aluno
<<ontology_class>>
OferecerMaterialDidatico
- descricao : String
- disciplina : Disciplina
- aluno : Aluno
+ enviaConteudo() : void
<<ontology_class>>
Disciplina
- codigo : String
- curso : Curso
- turma : String
- tutor : Tutor
- pre-requisitos : List<Disciplina>
<<ontology_class>>
Aluno
- matricula : int
- nome : String
- curso : Curso
- cidade : String
- disciplinas : List<Disciplina>
1..*
1
0..*

5.2.4.4 Diagrama de Sequência da Ontologia

Na figura a seguir é mostrado, no diagrama de sequência, utilizando a notação de MAS-ML 2.0, a interação entre o agente buscador de informações, a organização e a ontologia que ele utiliza para desempenhar seu papel.

Figura 32 – Ontologia do Buscador de Informações no Diagrama de Sequência.

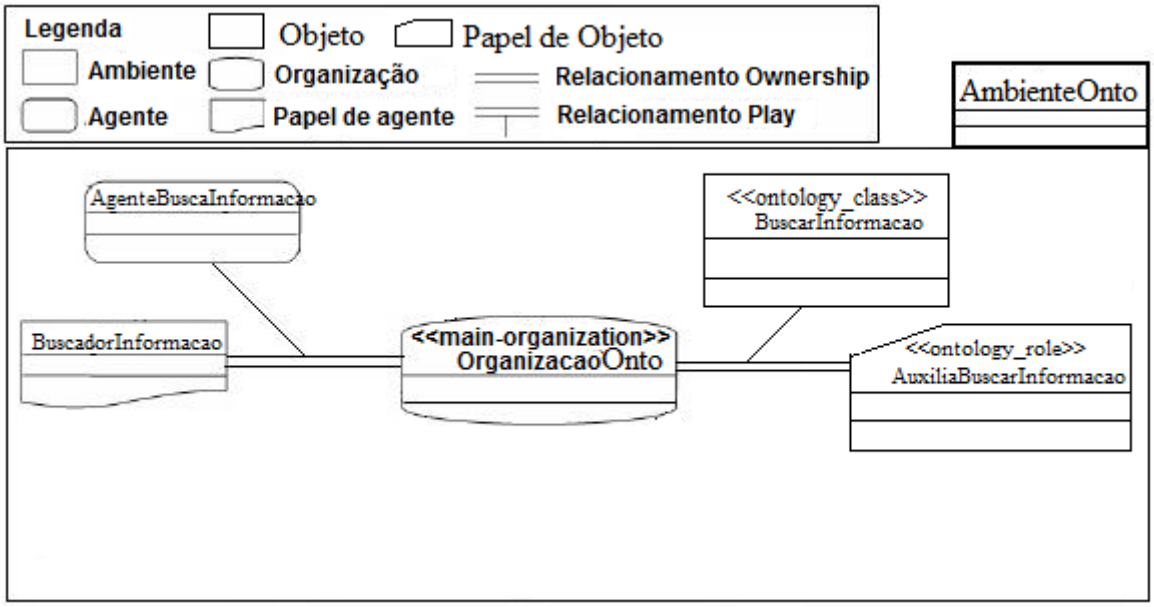


Fonte: elaborada pelo autor.

5.2.4.5 Diagrama de Organização da Ontologia

Na Figura 33 é apresentado o diagrama de organização simplificado da ontologia para o Agente Buscador de Informações. Nele são mostrados a organização principal, o agente e seu papel, a ontologia e seu papel.

Figura 33 – Ontologia do Buscador de Informações no Diagrama de Organização.

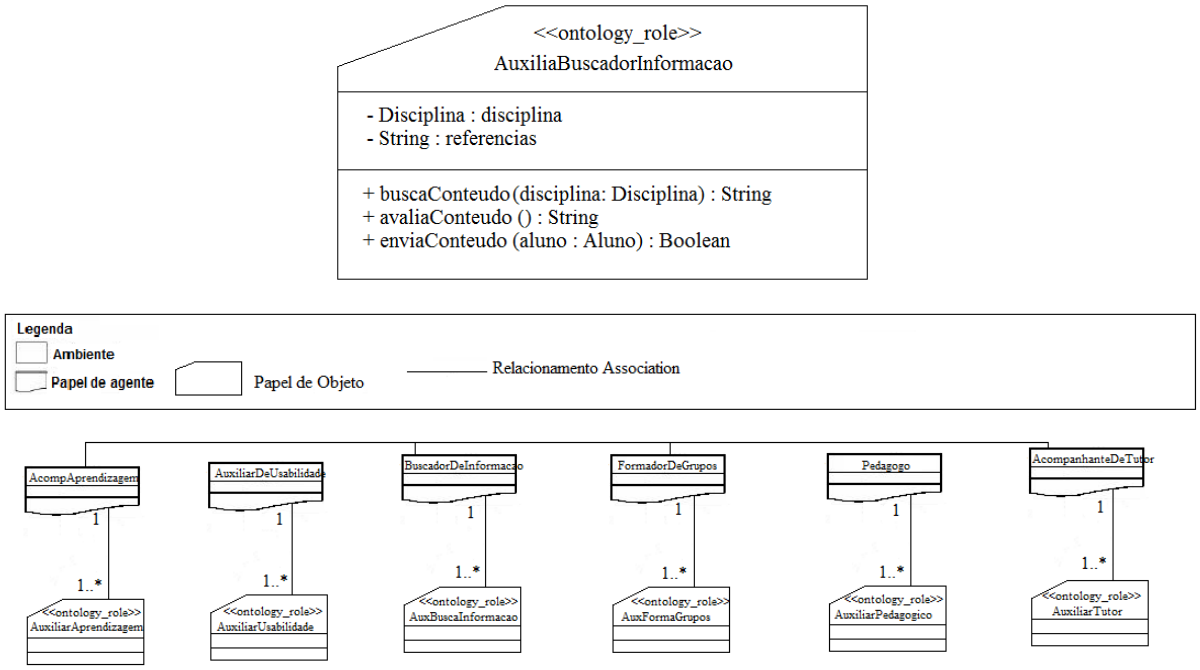


Fonte: adaptado de Gonçalves (2009).

5.2.4.6 Diagrama de Papel da Ontologia

Na Figura 34 é apresentado o diagrama de papel simplificado da ontologia para o Agente Buscador de Informações. Primeiro é mostrado a estrutura do papel de ontologia e depois ele é mostrado dentro do referido diagrama.

Figura 34 – Ontologia do Buscador de Informações no Diagrama de Papel.



Fonte: elaborada pelo autor.

5.2.5 Ontologia para o Agente Formador de Grupos

O objetivo desta ontologia é oferecer ao agente a descrição dos conceitos necessários para que ele saiba como identificar o perfil de um aluno e a partir disso formar grupos de estudo.

5.2.5.1 Identificação das Classes

a) *Aluno*: descreve para o agente o que é um aluno e quais são suas propriedades.
b) *Grupo*: descreve para o agente o que é um grupo de aprendizagem e quais são as suas características;
c) *Perfil*: descreve para o agente o que é um perfil, para que ele possa identificar o perfil dos alunos e formar grupos de acordo com os perfis traçados.
d) *Processo de Aprendizagem*: descreve para o agente o que é um processo de aprendizagem, para que ele possa saber como ajudar um grupo a aprender algo.
e) *Formar Grupos*: descreve para o agente como ele pode formar grupos a partir do que ele sabe sobre aluno, grupo e perfil.

5.2.5.2 Descrição dos Relacionamentos

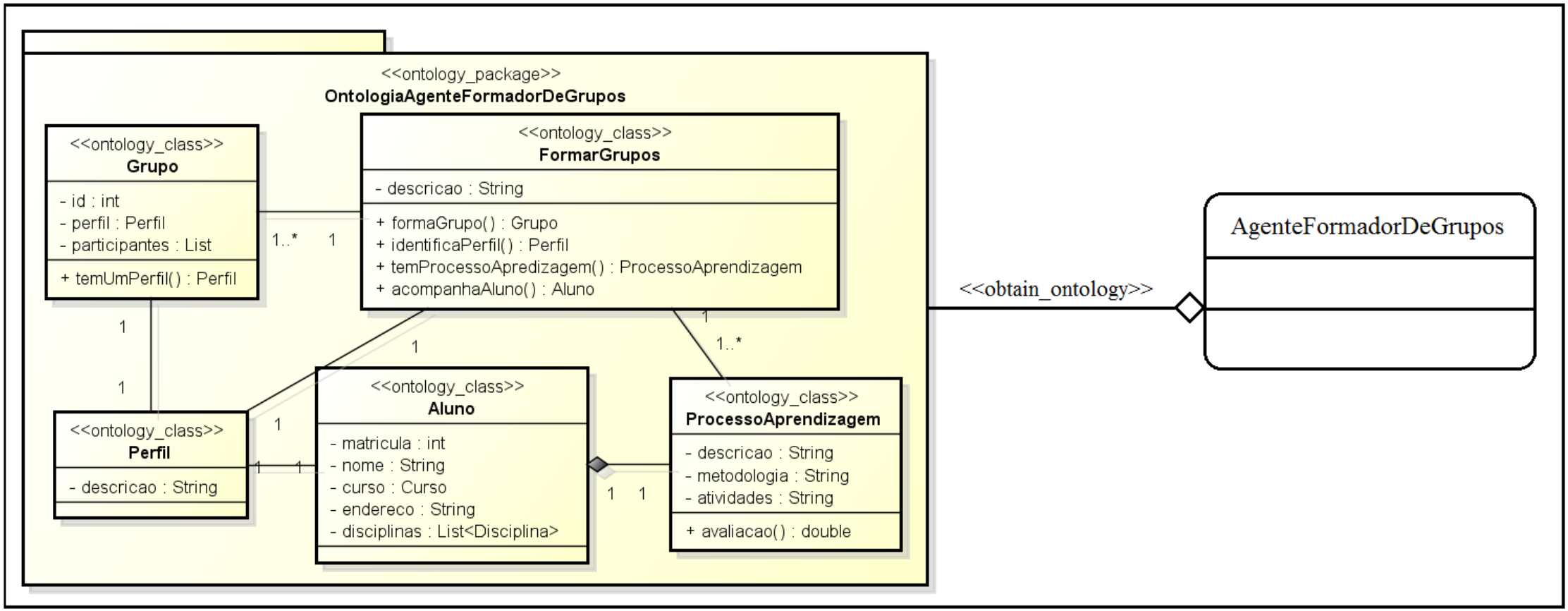
a) Um Aluno *tem um* Perfil, Processo de Aprendizagem e *participa de* um Grupo;
b) Um Grupo *tem um* Perfil *e um* Processo de Aprendizagem.
c) Um Grupo *é parte* de Formar Grupos.

5.2.5.3 Diagrama de Classes da Ontologia

Na Figura 35 é ilustrado o diagrama de classes simplificado da ontologia proposta para o Agente Formador de Grupos. A estrutura interna do agente foi omitida, uma vez que o objetivo desta subseção é modelar as classes/pacote da ontologia.

Figura 35 – Ontologia do Formador de Grupos no Diagrama de Classes.

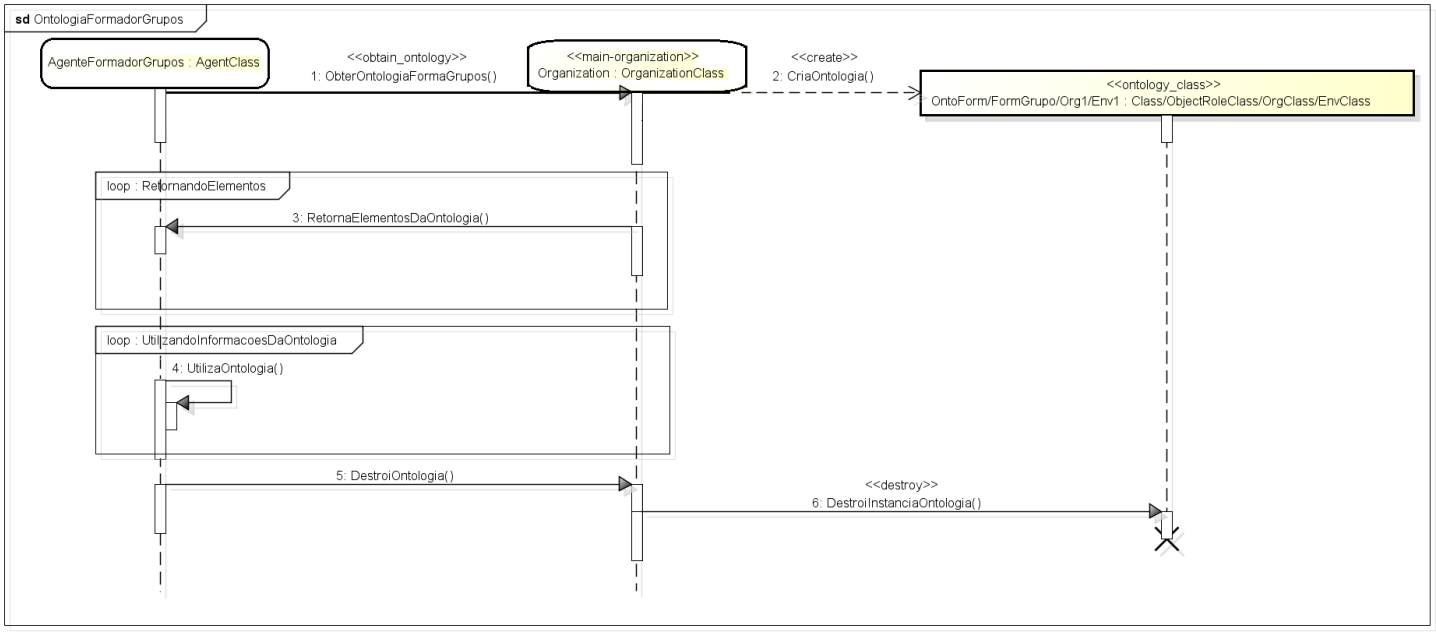
Fonte: elaborada pelo autor.


<<ontology_package>>
OntologiaAgenteFormadorDeGrupos
<<ontology_class>>
Grupo
- id : int
- perfil : Perfil
- participantes : List
+ temUmPerfil( ) : Perfil
<<ontology_class>>
FormarGrupos
- descricao : String
+ formaGrupo( ) : Grupo
+ identificaPerfil( ) : Perfil
+ temProcessoApredizagem( ) : ProcessoAprendizagem
+ acompanhaAluno( ) : Aluno
<<ontology_class>>
Perfil
- descricao : String
<<ontology_class>>
Aluno
- matricula : int
- nome : String
- curso : Curso
- endereco : String
- disciplinas : List<Disciplina>
<<ontology_class>>
ProcessoAprendizagem
- descricao : String
- metodologia : String
- atividades : String
+ avaliacao( ) : double
1..*
1
1
1
1
1
1
1
1
1
1..*
<<obtain_ontology>>
AgenteFormadorDeGrupos

5.2.5.4 Diagrama de Sequência da Ontologia

Na figura a seguir é mostrado, no diagrama de sequência, utilizando a notação de MAS-ML 2.0, a interação entre o agente formador de grupos, a organização e a ontologia que ele utiliza para desempenhar seu papel.

Figura 36 – Ontologia do Formador de Grupos no Diagrama de Sequência.

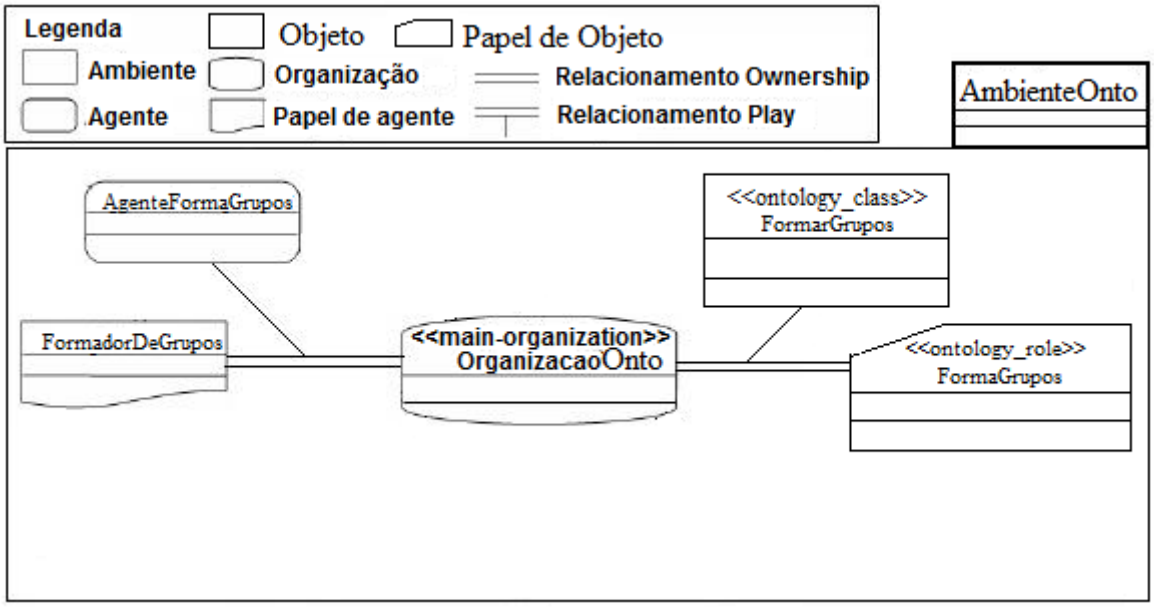


Fonte: elaborada pelo autor.

5.2.5.5 Diagrama de Organização da Ontologia

Na Figura 37 é apresentado o diagrama de organização simplificado da ontologia para o Agente Formador de Grupos. Nele são mostrados a organização principal, o agente e seu papel, a ontologia e seu papel.

Figura 37 – Ontologia do Formador de Grupos no Diagrama de Organização.

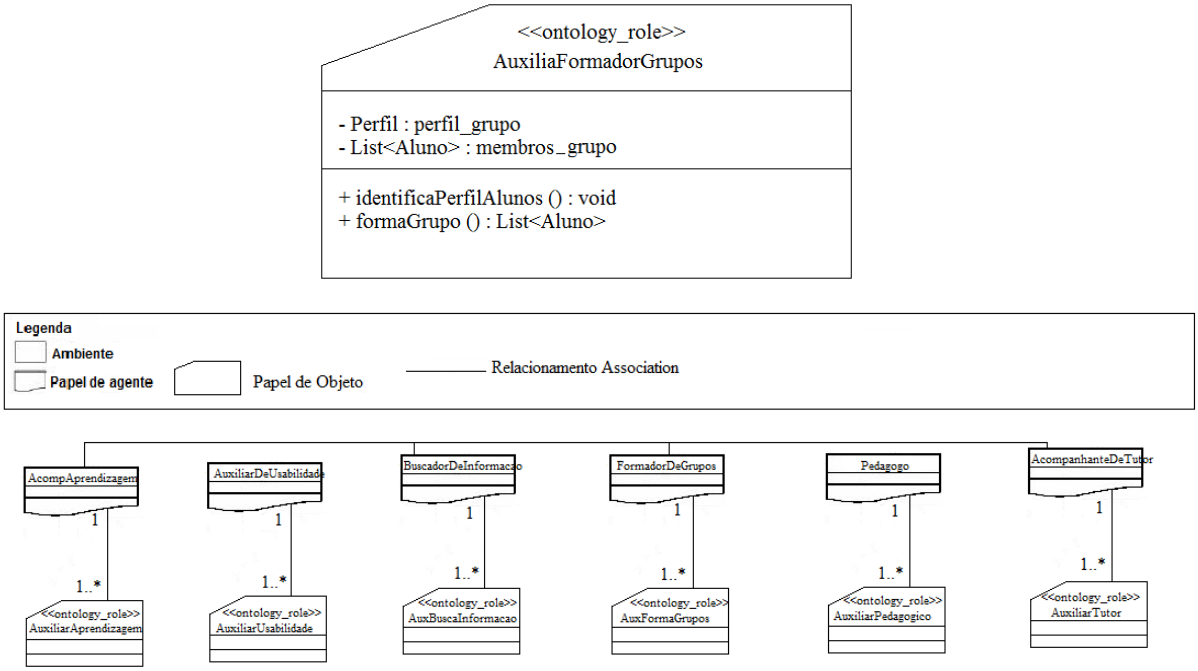


Fonte: adaptado de Gonçalves (2009).

5.2.5.6 Diagrama de Papel da Ontologia

Na Figura 38 é apresentado o diagrama de papel simplificado da ontologia para o Agente Formador de Grupos. Primeiro é mostrado a estrutura do papel de ontologia e depois ele é mostrado dentro do referido diagrama.

Figura 38 – Ontologia do Formador de Grupos no Diagrama de Papel.



Fonte: elaborada pelo autor.

5.2.6 Ontologia para o Agente Auxiliar de Usabilidade

O objetivo desta ontologia é oferecer ao agente a descrição dos conceitos necessários para que ele saiba oferecer dicas de como os usuários podem usar as funcionalidades do Moodle da melhor maneira possível.

5.2.6.1 Identificação das Classes

a) *Técnicas de Usabilidade*: descreve para o agente o que é uma técnica de usabilidade para que assim ele saiba como aplicar uma.
b) *Entrevistas e Questionários*: descreve para o agente como elaborar e aplicar entrevista e questionários com os usuários do Moodle, para que a partir disso ele possa oferecer dicas de usabilidade.
c) *Avaliação Heurística*: descreve para o agente o que é uma avaliação heurística e como ele pode aplica-la.
d) *Inspeção*: descreve para o agente como ele pode realizar uma inspeção de usabilidade.
e) *Checklist*: descreve para o agente o que é um checklist e como ele pode aplica-lo no Moodle.
f) *Auxiliar de Usabilidade*: descreve para o agente como ele pode auxiliar os usuários a partir do que ele sabe sobre os conceitos de usabilidade.

5.2.6.2 Descrição dos Relacionamentos

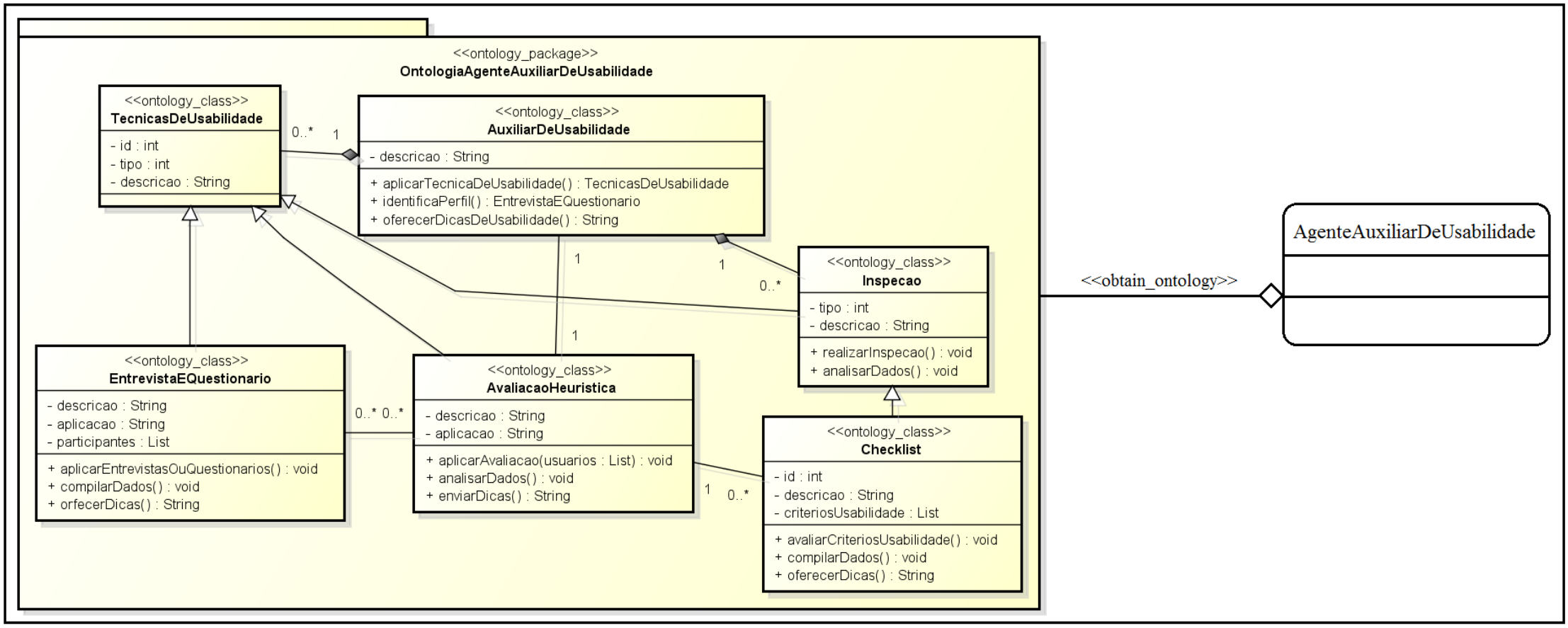
a) Entrevistas e Questionários, Avaliação Heurística e Inspeção são *um tipo de* Técnica de Usabilidade.
b) Checklist é *um tipo de* Inspeção.
c) Um Auxiliar de Usabilidade é *composto por* Técnica de Usabilidade.

5.2.6.3 Diagrama de Classes da Ontologia

Na Figura 39 é ilustrado o diagrama de classes simplificado da ontologia proposta para o Agente Auxiliar de Usabilidade. A estrutura interna do agente foi omitida, uma vez que o objetivo desta subseção é modelar as classes/pacote da ontologia.

Figura 39 – Ontologia do Auxiliar de Usabilidade no Diagrama de Classes.

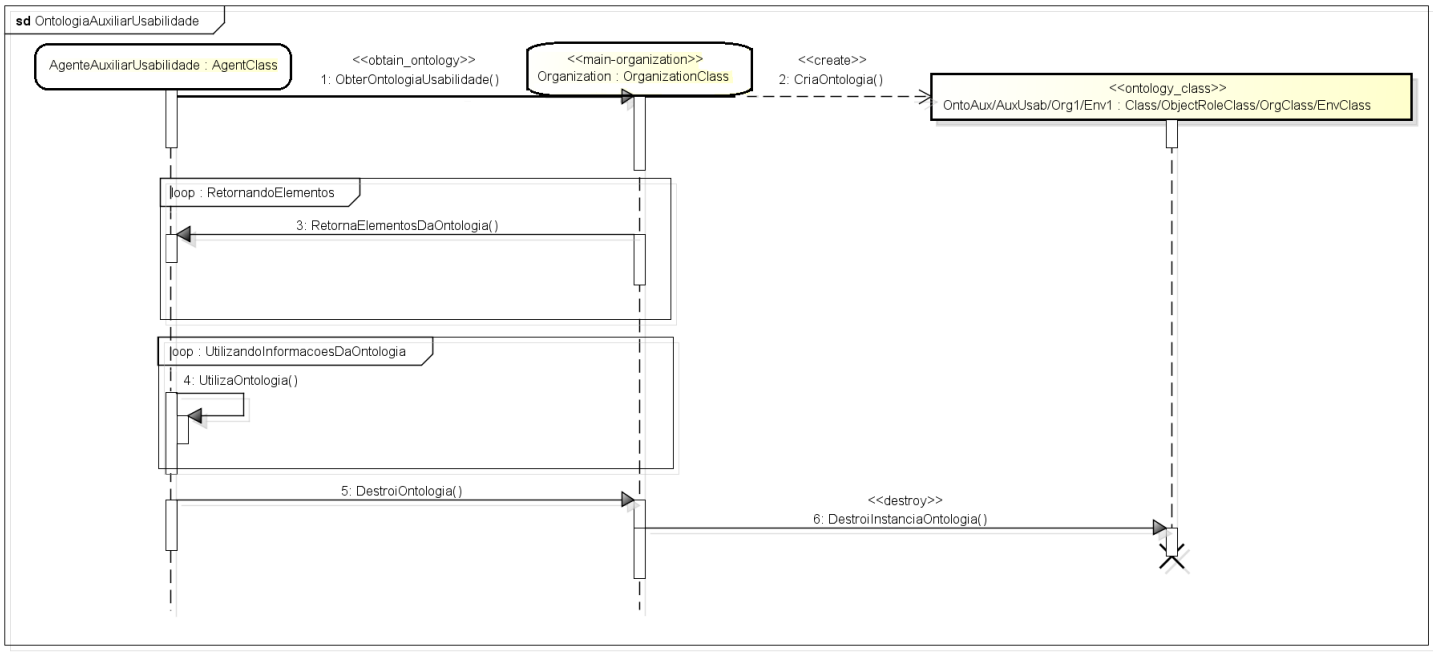
Fonte: elaborada pelo autor.


<<ontology_package>>
OntologiaAgenteAuxiliarDeUsabilidade
<<ontology_class>>
TecnicasDeUsabilidade
- id : int
- tipo : int
- descricao : String
<<ontology_class>>
AuxiliarDeUsabilidade
- descricao : String
+ aplicarTecnicaDeUsabilidade() : TecnicasDeUsabilidade
+ identificaPerfil() : EntrevistaEQuestionario
+ oferecerDicasDeUsabilidade() : String
<<ontology_class>>
Inspecao
- tipo : int
- descricao : String
+ realizarInspecao() : void
+ analisarDados() : void
<<ontology_class>>
EntrevistaEQuestionario
- descricao : String
- aplicacao : String
- participantes : List
+ aplicarEntrevistasOuQuestionarios() : void
+ compilarDados() : void
+ orfecerDicas() : String
<<ontology_class>>
AvaliacaoHeuristica
- descricao : String
- aplicacao : String
+ aplicarAvaliacao(usuarios : List) : void
+ analisarDados() : void
+ enviarDicas() : String
<<ontology_class>>
Checklist
- id : int
- descricao : String
- criteriosUsabilidade : List
+ avaliarCriteriosUsabilidade() : void
+ compilarDados() : void
+ oferecerDicas() : String
0..*
1
1
1
0..*
1
0..* 0..*
1
0..*
<<obtain_ontology>>
AgenteAuxiliarDeUsabilidade

5.2.6.4 Diagrama de Sequência da Ontologia

Na figura a seguir é mostrado, no diagrama de sequência, utilizando a notação de MAS-ML 2.0, a interação entre o agente auxiliar de usabilidade, a organização e a ontologia que ele utiliza para desempenhar seu papel.

Figura 40 – Ontologia do Auxiliar de Usabilidade no Diagrama de Sequência.

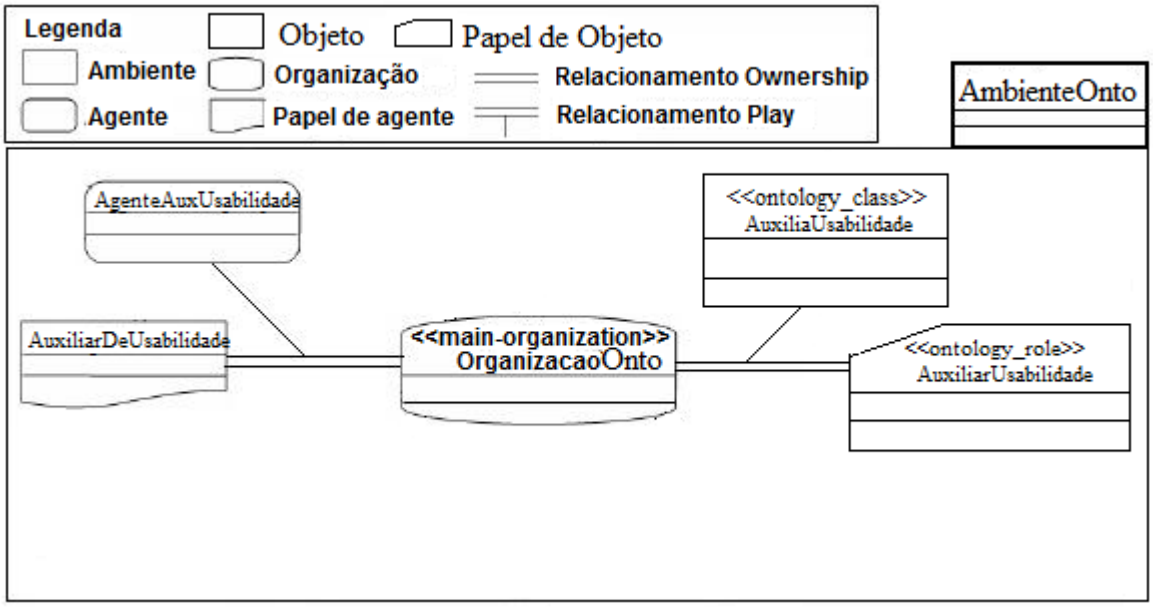


Fonte: elaborada pelo autor.

5.2.6.5 Diagrama de Organização da Ontologia

Na Figura 41 é apresentado o diagrama de organização simplificado da ontologia para o Agente Auxiliar de Usabilidade. Nele são mostrados a organização principal, o agente e seu papel, a ontologia e seu papel.

Figura 41 – Ontologia do Auxiliar de Usabilidade no Diagrama de Organização.

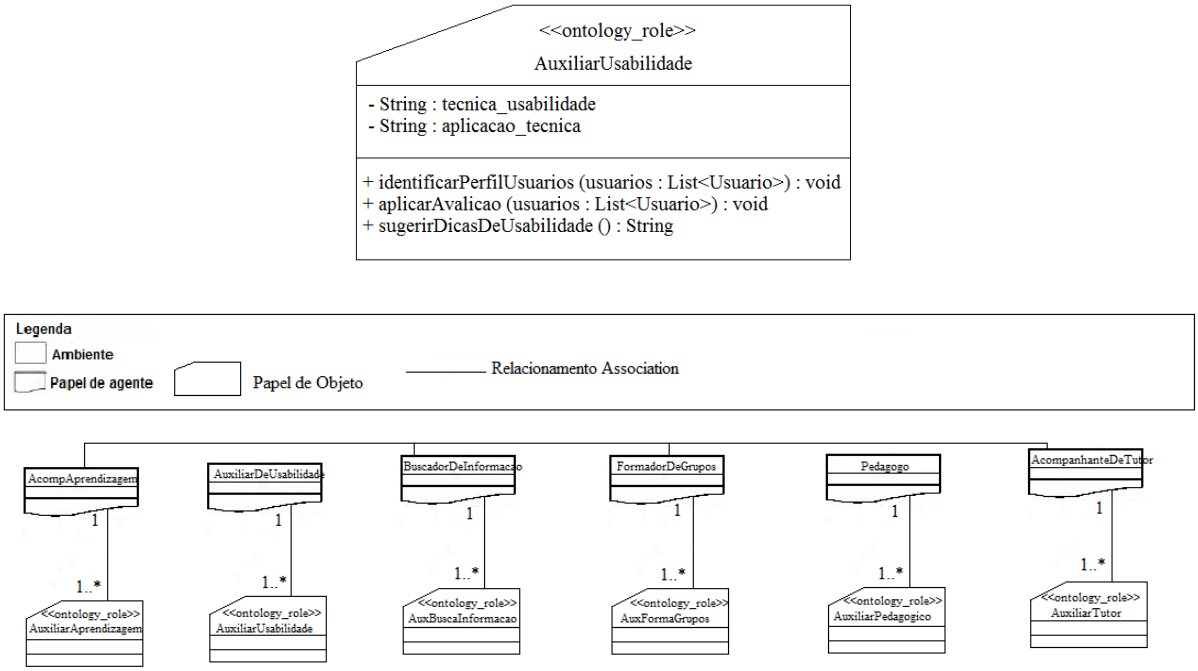


Fonte: adaptado de Gonçalves (2009).

5.2.6.6 Diagrama de Papel da Ontologia

Na Figura 42 é apresentado o diagrama de papel simplificado da ontologia para o Agente Auxiliar de Usabilidade. Primeiro é mostrado a estrutura do papel de ontologia e depois ele é mostrado dentro do referido diagrama.

Figura 42 – Ontologia do Auxiliar de Usabilidade no Diagrama de Papel.



Fonte: elaborada pelo autor.

6 CONSIDERAÇÕES FINAIS

Nesta seção são feitas as considerações finais acerca do trabalho realizado, focando nas contribuições, limitações e trabalhos futuros.

6.1 Conclusão

Este trabalho envolveu uma extensão do *framework* conceitual TAO e da linguagem de modelagem MAS-ML 2.0 para incluir, em ambos, os elementos da modelagem de ontologias no contexto de sistemas multiagentes. A extensão proposta foi construída a partir de um levantamento bibliográfico. Neste levantamento, foram analisados trabalhos relacionados e foram definidos os elementos que seriam utilizados para modelar ontologias em MAS-ML.

Inicialmente, a descrição dos elementos de ontologia foi inserida no *framework* TAO, pois o mesmo é responsável por descrever, detalhadamente, as abstrações (entidades e relacionamentos) que formam a linguagem MAS-ML. Posteriormente, trabalhamos a extensão na sintaxe abstrata de MAS-ML, onde nesta, foram inseridos e analisados os elementos descritos no TAO, bem como foram definidas as associações entre os elementos de ontologia e os elementos que já estavam presentes no metamodelo da linguagem.

Além disso, trabalhamos a extensão na sintaxe concreta da linguagem, onde as entidades e os relacionamentos de ontologia foram representados graficamente. Ainda na sintaxe concreta, foi trabalhada a representação dos elementos de ontologia nos diagramas estáticos (classes, organização e papel) e dinâmico (diagrama de sequência) de MAS-ML. No diagrama de classes, trabalhamos a representação de classes e pacotes de ontologia. No diagrama de papel, trabalhamos a representação dos papéis que podem ser exercidos por objetos de uma ontologia. No diagrama de organização, trabalhamos a representação de objetos e papéis de objetos de ontologia que são definidos no escopo de uma organização. No diagrama de sequência, trabalhamos as interações entre as entidades de MAS-ML e as entidades de ontologias. Nele, representamos os processos de criação, destruição e utilização dos elementos de uma ontologia.

Por fim, como demonstração de uso da extensão, foram identificadas e modeladas classes de ontologia. Essas classes são utilizadas por agentes de um SMA desenvolvido para o Moodle (Gonçalves *et al*., 2013), um ambiente virtual de aprendizagem muito utilizado na modalidade de educação à distância. Essas ontologias foram modeladas tanto nos diagramas estáticos quanto no diagrama dinâmico de MAS-ML 2.0.

Durante a execução deste trabalho nos deparamos com algumas limitações, entre elas podemos citar a dificuldade ocasionada pelo esforço de dominar novas tecnologias que seriam

utilizadas na extensão de uma ferramenta de modelagem. Com isso, o estudo de caso, no qual demonstramos a utilização da extensão proposta à MAS-ML ficou um pouco prejudicado, tendo que ser modelado através de uma ferramenta de modelagem baseada em UML e adaptado à MAS-ML através de editores de imagens.

Apesar das adversidades, este trabalho trouxe algumas contribuições tanto para o *framework* TAO quanto para a linguagem MAS-ML. Estas são citadas a seguir:

a) a extensão traz ao *framework* TAO a descrição dos elementos que podem ser usados na modelagem de ontologias. Isso auxilia bastante no entendimento de como esses elementos são utilizados na linguagem de modelagem;
b) a extensão foi trabalhada em todas as camadas da linguagem MAS-ML, desde a especificação dos elementos inseridos no metamodelo (sintaxe abstrata) até a representação dos mesmos nos diagramas que compõem esta linguagem (sintaxe concreta).
c) a extensão traz à MAS-ML a possibilidade de modelar não somente as entidades que fazem uso de ontologias, como também a possibilidade de modelar as ontologias que são utilizadas por estas entidades.
d) foram modeladas classes de ontologias nos diagramas estáticos e dinâmico de MAS-ML, e estas são utilizadas em um sistema multiagente desenvolvido para o Moodle.

6.2 Trabalhos Futuros

Durante a execução do trabalho proposto foram identificadas algumas possibilidades de trabalhos futuros, são eles:

a) inserir a extensão proposta neste trabalho em uma ferramenta de modelagem. Atualmente existem duas ferramentas para modelar sistemas multiagentes utilizando a linguagem MAS-ML: *MAS-ML tool*, criada no trabalho de Farias *et al*. (2009) e *VisualAgent*, criada por De Maria *et al*. (2005);
b) trabalhar na geração de código-fonte das ontologias a partir de uma ferramenta de modelagem (item a). Para isso, pode-se utilizar uma abordagem baseada em MDA (*Model Driven Architecture*) semelhante à utilizada por De Maria (2005);
c) entender melhor as necessidades da modelagem de ontologias diante da extensão proposta neste trabalho, analisar todas as propostas de melhoria que poderão surgir a partir desta iniciativa e inseri-las em versões futuras da linguagem.

REFERÊNCIAS

ALMEIDA, M. B.; BAX, M. P. **Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção**. Ciência da Informação, Brasília, v. 32, 2003. Disponível em: <http://revista.ibict.br/ciinf/index.php/ciinf/article/view/17/11>. Acesso em: 03 maio 2013.

BARBER, K. S.; FULLAM, K.; KIM, J. Challenges for trust, fraud and deception research in multi-agent systems. In: **Proceedings of the Workshop on Deception, Fraud and Trust, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)**. Italy: Bologna. Publisher in Springer Berlin Heidelberg, 2003.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML:** guia do usuário. 2 ed. Rio de Janeiro: Elsevier, 2005.

CARTER, J.; MOLINA, J. M.; MURO, J. D. **Trust management through fuzzy reputation**. Published in Journal: Cooperative Information Systems, v. 12, n. 1, 2003.

CASARE, S. J. **Uma ontologia funcional de reputação para agentes**. Dissertação (Mestrado em Engenharia Elétrica) – Escola Politécnica, Universidade de São Paulo (USP), São Paulo, SP, 2005.

CASTELFRANCHI, C. Engineering social order. In: **Proceedings of the First International Workshop on Engineering Societies in the Agent World (ESAW'00)**. United Kingdom: London, 2000.

CASTRO, J.; ALENCAR, F.; SILVA, C. Engenharia de software orientada a agentes. In: **Atualizações em Informática**. Rio de Janeiro: PUC-Rio, 2006.

CERVENKA, R.; TRENCANSKY, I.; CALISTI, M.; GREENWOOD, D. AML: Agent Modeling Language. In: **Agent-Oriented Software Engineering V: 5th International Workshop (AOSE 2004)**. United States: New York, NY, 2004.

CHAUI, M. **Convite à filosofia**. 13 ed. São Paulo: Ática. 2009.

CHOREN, R.; LUCENA, C. Agent-oriented modeling using ANote. In: **3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2004)**. United Kingdom: Edinburgh, 2004.

CONTE, R. PAOLUCCI, M. **Reputation in artificial societies:** social beliefs for social order. Kluwer Academic Publishers, 2002.

COPPIN, B. **Inteligência artificial**. 1 ed. Rio de Janeiro: LTC, 2012.

COSTA; E.; SIMÕES, A. **Inteligência artificial:** fundamentos e aplicações. 3 ed. Lisboa: FCA, 2008.

D'INVERNO, M.; LUCK, M. **Understanding agent systems**. New York: Springer. 2001.

DARDENNE, A.; LAMSWEERDE, A.; FICKAS, S. **Goal-directed requirements acquisition**. Science of Computer Programming, v. 20, 1993.

DE MARIA, B. A.; SILVA, V. T.; LUCENA, C.J.P.; CHOREN, R. **VisualAgent: A Software Development Environment for Multi-Agent Systems**. 19º Simpósio Brasileiro de Engenharia de Software (SBES 2005), Uberlândia, MG, 2005.

DEMAZEAU, Y. From interactions to collective behaviour in agent-based systems. In: **European Conference on Cognitive Science**, France: Institute IMAG, 1995.

DERBAS, G.; KAYSSI, A. ; ARTAIL, H. ; CHEHAB, A. TRUMMAR: A trust model for mobile agent systems based on reputation. In: **The IEEE/ACS International Conference on Pervasive Services (ICPS'04)**. Lebanon: American University of Beirut (AUB), July, 2004.

ECLIPSE FOUNDATION. **Eclipse Platform**. [site corporativo]. Disponível em: <http://www.eclipse.org>. Acesso em: 4 jul. 2013.

FARIAS, K.; NUNES, I.; SILVA, V.; LUCENA, C. MAS-ML Tool: um ambiente de modelagem de sistemas multiagentes. In: **V Workshop on Software Engineering for Agent-oriented Systems**. Fortaleza, 2009.

GATTI, M. A. C.; LUCENA, C. J. P.; BRIOT, J. P. On Fault Tolerance in Law Governed Multi-Agent Systems. In: **5th International Workshop on Software Engineering for Large-scale Multi-Agent Systems (SELMAS) at ICSE'06,** 2006.

GONÇALVES, E. J. T. **Modelagem de arquiteturas internas de agentes de software utilizando a linguagem MAS-ML 2.0**. Dissertação (Mestrado em Ciência da Computação) – Centro de Ciências e Tecnologia, Universidade Estadual do Ceará, Fortaleza, CE, 2009.

GONÇALVES, Enyo J. T.; DE OLIVEIRA, Marcos A.; JUNIOR, João H. Freires.; FEITOSA, Guilherme E. S.; MENDES, Douglas H.; CORTÉS, Mariela I.; FEITOSA, Robson G. F. **Desenvolvimento de um sistema multiagente para o MOODLE baseado em técnicas de engenharia de software**. [S.l.]. 2013.

GRUBER, T. **What is an ontology?** [S. l. : s. n.], 1996.

JENNINGS, N. R. Coordination techniques for distributed artificial intelligence. In: **Foundations of distributed artificial intelligence**, John Wiley and Sons, 1996.

MULLER, G.; VERCOUTER, L. **L.I.A.R – Achieving social control in open and decentralised multi-agent systems**. France: Saint-Étienne, 2008.

NARDIN, L. G. **Uma arquitetura de apoio à interoperabilidade de modelos de reputação de agentes**. Dissertação (Mestrado em Engenharia Elétrica) – Escola Politécnica, Universidade de São Paulo (USP), São Paulo, SP, 2009.

NOY, N. F.; McGUINNESS, D. L. **Ontology development 101:** a guide to creating your first ontology. Stanford: Stanford University, 2001. Disponível em: <http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html>. Acesso em: 13 maio 2013.

OBJECT MANAGEMENT GROUP (OMG). **Unified Modeling Language (UML)**. [site corporativo]. Disponível em: <http://www.uml.org>. Acesso em: 08 jul. 2013.

ODELL, J.; PARUNAK, H. V. D.; BAUER, B. Extending UML for Agents. In: **Agent-Oriented Information Systems (AOIS'00) Workshop at the 17th National Conference on Artificial Intelligence (AI'00)**. United States: Austin, TX, 2000.

OSTROM, E. A behavioral approach to the rational choice theory of collective action: presidential address, american political science association. In: **The American Political Review**, v. 92, n. 1, 1998.

RUSSEL, S.; NORVIG, P. **Inteligência artificial**. 2 ed. Rio de Janeiro: Elsevier, 2004.

SABATER, J.; PAOLUCCI, M.; CONTE, R. **Repage:** Reputation and image among limited autonomous partners. Published in: Journal of Artificial Societies and Social Simulation, vol. 9, n. 2, 2006. Disponível em: <http://jasss.soc.surrey.ac.uk/9/2/3.html>. Acesso em: 08 jul. 2013.

SABATER, J.; SIERRA, C. Reputation and social network analysis in multi-agent systems. In **Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)**. Italy: Bologna, 2002.

SILVA, V. T. **Uma linguagem de modelagem para sistemas multiagentes baseada em um framework conceitual para agentes e objetos**. Tese (Doutorado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2004.

SOMMERVILLE, I. **Engenharia de software**. 8 ed. São Paulo: Pearson Addison-Wesley, 2007.

WAGNER, G. **The Agent-Object-Relationship Meta-Model:** towards a unified view of state and behavior. Netherlands: Eindhoven University of Technology, Department Information and Technology. Published in: Journal Information Systems, v. 28, July, 2003.

WOOLDRIDGE, M. **An introduction to multiagent systems**. 2nd. United Kingdom: John Wiley and Sons Ltd, 2009.

YU, B.; SINGH, M. P. An evidential model of distributed reputation management. In: **Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)**. Italy: Bologna, 2002.

YU, L.; SCHMID, B. A conceptual framework for agent-oriented and role-based work on modeling. In: **1th International Workshop on Agent-Oriented Information Systems (AOIS 1999)**. Germany: Heidelberg, 1999.

ZACHARIA, G.; MAES, P. **Trust management through reputation mechanisms**. Published in: Applied Artificial Intelligence Journal, v.14, Oct. 2000. United Kingdom: London, Taylor and Francis Ltd, 2000.