YLANA KIPUNA DOS SANTOS FIGUEIREDO

Orientador: Fernando Cortez Sica

MODELAGEM DE FERRAMENTAS FOCADAS EM ONTOLOGIAS PARA A EAD BASEADAS EM TEORIAS SOCIAIS E AGENTES INTELIGENTES

Ouro Preto

Novembro de 2010

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

MODELAGEM DE FERRAMENTAS FOCADAS EM ONTOLOGIAS PARA A EAD BASEADAS EM TEORIAS SOCIAIS E AGENTES INTELIGENTES

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

YLANA KIPUNA DOS SANTOS FIGUEIREDO

Ouro Preto
Novembro de 2010

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Modelagem de ferramentas focadas em ontologias para a EaD baseadas em teorias sociais e agentes inteligentes

YLANA KIPUNA DOS SANTOS FIGUEIREDO

Monografia defendida e aprovada pela banca examinadora constituída por:

Me. Fernando Cortez Sica – Orientador
Universidade de Campinas

Dr. Carlos Alberto Marques Pietrobon
Pontifícia Universidade Católica/RJ

Dr. José Romildo Malaquias
Universidade Federal de Uberlândia

Ouro Preto, Novembro de 2010

Resumo

Este trabalho demonstra o uso de ontologias baseadas em teorias sociais para o processo de criação de ferramentas computacionais a serem utilizadas pela EaD (Educação a Distância). Com a contextualização da ontologia baseada em teorias sociais, é permissível a modelagem de agentes inteligentes aplicados à educação. Neste caso, os agentes comportam-se como proxies dos cursistas afim de que sejam estabelecidas interações ou comunicações entre os demais agentes pertencentes à comunidade e, também, afim de que sejam inspecionadas as atividades dos cursistas inseridos em um grupo (comunidade) específica. Estes agentes são criados levando-se em conta o conceito de grupos modelados a partir de uma ontologia baseada em teorias sociais, tendo como foco "interesses comuns" entre os integrantes do grupo.

Palavras-chave: Ontologia.Teorias Sociais. EaD. Agentes Inteligentes

Abstract

This work demonstrates the use of ontologies based on social theories for the process of creating computational tools to be used by DE (Distance Education). In the context of ontology based on social theories, it is permissible for modeling intelligent agents for education.

In this case, the agents act as proxies of the course participants in order to be established communications or interactions between other agents belonging to the community and also be inspected in order that the activities of the course participants within a group (community) specific. These agents are created taking into account the concept of groups modeled after an ontology based on social theories,"focusing on common interests" among members of the group.

Keywords: Ontology. Social Theories. DE. Intelligent Agents.

Dedico este trabalho aos meus pais que sempre me apoiaram em todos os momentos, sempre me deram uma palavra de conforto e carinho nas horas mais conflitantes da minha vida e muitas vezes sem mesmo saberem.

E sempre fizeram tudo para me dar uma boa educação e para que nada me faltasse me ajudando a alcançar mais uma meta.

Agradecimentos

Agradeço ao Senhor, pois sem Ele nada sou, sem Ele nada teria sido realizado e não conseguiria alcançar mais uma etapa da minha vida. Pois tudo o que sou e tenho devo ao Senhor, que sempre me amparou em todas as horas, nos momentos mais difíceis, me deu forças para me re-erguer, me deu discernimento para superar os obstáculos e alcançar os meus objetivos. Sempre estiveste presente e sempre foi o meu o Melhor Amigo. Muito obrigado, meu Deus!

Aos meus queridos papás, Humberto e Teresa, pelo amor imensurável que têm por mim, por toda dedicação, por todos os ensinamentos, por todos os conselhos, por todas as palavras de consolo e principalmente, por acreditarem em mim. Acreditarem que a minha vinda ao Brasil seria para alcançar o objetivo de me formar, me transformar na pessoa que hoje sou e colher os frutos advindos desta árvore plantada.

As minhas avós, Joana e Aurora pela constante preocupação. A minha tia do Coração, Guerrita, pelas palavras de conforto.

Ao Osvaldo, pela amizade sincera, pelas palavras de consolo e encorajamento nos momentos de incerteza, por todas as tristezas e algerias divididas, por todo carinho, atenção e bons conselhos doados nos momentos em que precisei.

A Paloma pela suas atitudes de verdadeira amizade e pelo seu companheirismo que se resumem a uma única palavra, irmandade!

As minhas amigas, que sempre estarão no meu coração Âurea, Nelma, Suelaine e Sílvia por todos momentos de alegrias e tristezas compartilhados.

Aos meus colegas e amigos de graduação, em especial Daniel, Igor, Pablo, Rodrigo, Marcos, por dividirem comigo alguns obstáculos e por toda a força e amizade depositidas.

Ao meu Orientador e Professor, Sica, por toda a paciência, por toda ajuda e orientação, durante a realização deste trabalho. Pelas suas palavras amigas nos momentos de tensão e por todos os seus ensinamentos.

A todos os mestres que contribuiram para minha formação acadêmica e engrandecimento como instrumento da sociedade.

A Ouro Preto, que me ensinou a crescer e a todos àqueles que fizeram parte da minha jornada.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Nas últimas décadas, tem-se visto uma grande massa de cidadãos com a preocupação de terem uma educação e para atender a esta grande massa, evidenciou-se a importância da educação a distância (EaD), realizada a princípio por meio de correspondência, posteriormente através do uso de meios de comunicação como o rádio e a televisão associados a materiais impressos enviados pelo correio. O advento das tecnologias de informação e comunicação trouxe novas perspectivas para a educação a distância, levando universidades, escolas, centros de ensino, organizações empresariais e grupos de profissionais de educação a se dedicarem ao desenvolvimento de cursos a distância com suporte em ambientes digitais de aprendizagem acessados via internet, os quais assumem distintas abordagens. Este trabalho tem por objetivo desenvolver uma ferramenta que possa facilitar o estudo e aprendizado do aluno em um ambiente virtual para educação a distância. Com base nisso, pretende-se criar uma ontologia de aluno que propicie uma base para o desenvolvimento do trabalho prosposto para esta monografia.

Para que os pontos inerentes ao desenvolvimento do trabalho sejam conhecidos, esta monografia inicia-se apresentando uma breve introdução sobre o trabalho proposto, posteriormente, no capítulo 2 é apresentada a justificativa do trabalho dando ênfase a sua relevância.

No capítulo 3 são apresentados os objetivos gerais e específicos, ajudando a compreender melhor a relevância deste trabalho.

O capítulo 4 apresenta a contextualiazação da proposta de trabalho, que tem como base regras e ações sociais que partem de teorias de pensadores modernos e sociólogos como Durkheim e Vygotsky. Através destas teorias sociais é possível ter-se a idéia de como se pode criar grupos que atendem um certo objetivo. Estes grupos serão formados por alunos, uma vez que o foco deste trabalho é a criação de uma ferramenta para educação a distância baseada em uma ontologia de aluno, que visa a interação entre alunos de um certo curso, através de grupos constituídos por eles.

Como citado no parágrafo anterior a modelagem dos alunos e dos grupos será feita através da criação de ontologias para aluno e grupo, respectivamente, que posteriormente será utilizada para a implementação dos agentes inteligentes através de um framework do Java apropriado

para este tipo de aplicação, uma vez que este já possuí todas regras para o estabelecimento da comunicação entre agentes. As regras de comunicação são estabelecidas pelos protocolos do FIPA, usados pelo JADE.

O capítulo 5 apresenta o conceito de Ontologia, suas características e relações, de seguida, no capitulo 6 são apresentados alguns conceitos de Inteligência Artificial dando ênfase à área de sistemas multiagentes e suas principais características.

No capítulo 7 será feita uma breve apresentação sobre as especificações produzidas pela fundação FIPA que criou protocolos que facilitam a comunicação entre agentes. O capítulo 8 falará sobre o framework JADE adotado para implementação do sistema multiagente proposto no trabalho, bem como algumas das suas especificações e suas diversas aplicações.

O capítulo 9 trata da implementação do trabalho, onde será feita a descrição da implementação dos agentes inteligentes, modelados através de uma ontologia. Também serão apresentadas as funcionalidades inerentes as classes implementadas.

Ainda no capítulo 9, serão apresentados os resultados obtidos após a execução da aplicação, no capítulo 10 serão apresentadas as conclusões obtidas durante a concepção do trabalho.

Capítulo 2

Justificativa

Este trabalho se justifica pela relevância e contribuição apresentada pela fusão de teorias sociais e informática. Justifica-se, ainda, pela importância que os ambientes virtuais de aprendizagem e a própria EaD vem tomando no cenário educacional e social. Cenário representado por iniciativas privadas e públicas, como por exemplo, o programa Proinfo do governo federal.

Diante da crescente demanda dos ambientes e ferramentas relacionados à EaD, espera-se, com este trabalho, contribuir para que novas ferramentas e metodologias possam ser criadas e consolidadas de forma a estabelecer novos caminhos e horizontes para os processos de ensino-aprendizagem.

Aliado ao fato da importância da representação ontológica no cenário computacional, tem-se, também, a relevância da Educação a Distância EaD no contexto da educação, havendo neste campo a participação em massa da computação representada pelas NTICs - Novas Tecnologias de Interação e Comunicação, dentre as quais incluem-se os AVAs - Ambientes Virtuais de Aprendizagem.

Devido às importâncias e relevâncias apresentadas, pretende-se com esta monografia, focar o uso das ontologias no processo de criação de ferramentas computacionais para a EaD de forma a consolidar perfis de ferramentas baseadas em perfis sociais.

Capítulo 3

Objetivos

Nesta seção serão apresentados os objetivos gerais e específicos do trabalho.

3.1 Objetivo Geral

Desenvolver uma ferramenta para EaD, partindo da criação de uma ontologia baseada em perfis sociais de alunos e agentes inteligentes modelados por meio de um framework do Java, apropriado para o desenvolvimento de aplicações orientadas a agentes e interação entre eles, o JADE (Java Agent Development framework) é um ambiente que propõe toda uma infra-estrutura de suporte e desenvolvimento de sistemas multiagentes, tornando a implementação mais simples e robusta.

3.2 Objetivos Específicos

- Estudar representações na forma de ontologias;
- Estudar pensadores da ciência social relacionados à educação;
- Estabelecer contextos e fronteiras entre a educação mediada por computador e os parâmetros sociais;
- Estudar conceitos inerentes aos agentes inteligentes aplicados à educação;
- Criar uma representação baseada em ontologia;
- Criar metodologias de ensino e agentes inteligentes;
- Implementar mecanismos de interação inter-agentes, com o uso da ferramenta JADE.

Capítulo 4

Contextualização

A partir da contextualização das regras e ações sociais baseadas em teorias de pensadores modernos, como Durkheim, Vygotsky e outros sob a forma de ontologia e do estabelecimento de relações das teorias sociais com o processo de ensino-aprendizagem mediada por computador, pretende-se modelar ferramentas computacionais para a EaD. Esta contextualização será utilizada para alimentar as regras de interação, supervisão, controle e mediação dos agentes inteligentes modelados como proxies dos cursistas. Entende-se como "proxy agent" o modelo de agente modelado para inspecionar as atividades desenvolvidas pelos cursistas de modo a interagir com os demais agentes pertencentes à comunidade e propôr fluxos de execução para as ações de ensino-aprendizagem.

Para estabelecer parâmetros e conceitos afim de facilitar o processo de criação de ontologias, é importante o conhecimento das teorias e contextualização criadas pelos pensadores e sociólogos. A seguir, são citados e retratados alguns pensadores de suma importância ao entendimento da atual conjuntura educacional.

Segundo Durkheim Allahverdi (1999), sociólogo da educação positivista:

> "As pessoas têm incorporadas em si dois seres. O primeiro, é o ser individual que se caracteriza pelos estados mentais de cada um e pelos aspectos de sua vida pessoal. O segundo é o ser social, voltado para os comportamentos relacionados à sociedade em que vivemos."

Para Durkheim a educação deveria ao mesmo tempo, ter uma base comum e diversificada. Apesar das diferenças sociais todas as crianças devem receber idéias e práticas, que são valores do seu povo, da sua nação. Essa seria a base comum da educação, pois contem os conhecimentos que deveriam ser compartilhados por todos.

Durkheim não desenvolveu métodos pedagógicos, mas suas idéias ajudaram a compreender o significado social do trabalho do professor, tirando a educação escolar da perspectiva individualista, sempre limitada pelo psicologismo idealista (Revista Escola - Durkheim (2008)). Segundo Durkheim, o papel da ação educativa é formar um cidadão que tomará parte do espaço público. Não somente o desenvolvimento individual do aluno. A educação tem por

objetivo suscitar e desenvolver na criança estados físicos e morais que são requeridos pela sociedade política no seu conjunto. Tais exigências, com forte influência no processo de ensino, estão relacionadas à religião, às normas e sanções, à ação política, ao grau de desenvolvimento das ciências e até mesmo ao estado de progresso da indústria local (Revista Escola - Durkheim (2008)).

Já neste ponto, pode-se traçar paralelos dos princípios de Durkheim com a utilização de agentes inteligentes como agentes mediadores do processo de ensino-aprendizagem. Como se pôde notar, Durkheim tem como uma de suas sugestões básicas, a introdução de elementos que permitam a interação e a construação colaborativa e cooperativa do conhecimento. Sendo assim, a determinação de ontologias contextualizando ações sociais no âmbito educacional, junto com a implementação de agentes inteligentes, vêm ao encontro dos princípios deste pensador.

Além de Durkheim, outros sociólogos também tiveram uma partcipação extremamente importante no desevolvimento de novas teorias educacionais, que hoje representam um papel muito importante não só na prática da educação presencial, como também na educação a distância. Dentre estes, destaca-se Vygotsky, sociólogo interacionista que elaborou sua teoria tendo por base o desenvolvimento do indivíduo como resultado de um processo sócio-histórico, destacando o papel da linguagem e da aprendizagem nesse desenvolvimento, passando a ser uma teoria histórico-social (Revista Escola - Vygotsky (2008)). Uma idéia principal para a compreensão das concepções sobre desenvolvimento humano como processo sócio-histórico é a idéia de mediação. Vygotsky enfatiza a construção do conhecimento como uma interação mediada por várias relações, sejam elas intra ou interpessoais, ou seja, o conhecimento não está sendo visto como uma ação do sujeito sobre a realidade e sim, pela mediação feita por outros sujeitos (Vygotsky (2001)).

Portanto, para Vygotsky, a aquisição de conhecimentos se opera pela interação do sujeito com o meio, ou seja, o sujeito não é apenas ativo, mas interativo.Tendo em vista sua teoria, Vygotsky considerava o aluno não somente como um sujeito de aprendizagem, mas aquele que aprende junto ao outro o que o seu grupo social produz, tal como: valores, linguagem e o próprio conhecimento (Daniels (2003) ”apud” Vygotsky (2001)). A partir desta teoria e da teoria de Durkheim, tem-se a idéia de desenvolver os agentes inteligentes por meio de um framework específico do JAVA, modelando-os como grupos, sendo que cada grupo detém as próprias características sociais e os elementos que os constituem têm a capacidade de interagir uns com os outros e até mesmo com elementos não percentes ao grupo. Um exemplo de grupo/grupos detendo de características a ele/eles inerentes seria:

- Grupo para alunos que frequentassem uma determinada série.
- O grupo pode ser destinado a discussão e ao aperfeiçoamento do aprendizado em uma determinada disciplina, por exemplo, matemática. Os alunos também podem fazer do

o grupo um repositório de material de estudos, como trabalhos resolvidos, listas de exercícios, provas, etc.

- Estabelecendo-se a disciplina a ser estudada, determinar o grau de dificuldade dos alunos dentro da disciplina e criar formas mais fáceis de mediar os alunos. Neste caso há a necessidade de um agente mediador do grupo, ou então, o próprio grupo já atua como mediador. Ele identifica as dificuldades de cada alunos, internamente ele separa os alunos que têm as mesmas dificuldades e sugere novas formas de aprendizado.

- Os próprios alunos podiam sanar as dúvidas uns dos outros, por meio de troca de mensagens.

- O grupo faz convites de participação.

Após o estudo dessas teorias sociais voltadas para a educação que contribuiram para a modernização do sistema educacional, culminando em ambientes em que os próprios alunos se responsabilizam pelo processo indivídual de aprendizagem,pesquisas apontam que há a necessidade de se criar ambientes que tenham a capacidade de se adaptarem a este contexto e de se personalizarem de acordo com as características dos alunos. Daí, mais uma vez, se pode ver a importância do conceito de grupos para a modelagem dos agentes inteligentes, ou seja, os grupos de cursistas de formas a atender as suas necessidades. Isso, leva à introdução de técnicas de IA com a finalidade de propiciar mecanismos de modelagem do processo de ensino bem como do estado cognitivo do estudante. Cabe a partir deste ponto mostrar um pouco sobre agentes inteligentes, suas propriedades e funcionamento, entretanto, será definido um capítulo para aprofundar mais o estudo sobre o tema.

Agentes inteligentes é um ramo da IA (*inteligência artifial*), constuído por componentes capazes de organizar, selecionar, produzir informações e tomar decisões a partir de alguma fonte de dados. Os agentes devem deter de algumas propriedades para que eles possam se comunicar e cooperar com outros agentes que coexistem no mesmo ambiente. São elas:**Autonomia**, **Reatividade**, **Pró-atividade**, **Intencionalidade**, **Racionalidade**, **Continuidade temporal**, **Sociabilidade**, **Benevolência**, **Adaptabilidade**, **Mobilidade**, que serão explicadas mais adiante.

Para se estabelecer a intereção entre os agentes, é necessário que haja uma arquitetura que define processos que vão possibilitar a intereção dos agentes. Além da arquitetura que facilita a comunicação entre agentes, criando assim, um sistema multiagente, ou seja, vários agentes acoplados dentro de um ambiente, é necessário que haja um meio de comunicação entre esses agentes, com protocolos bem definidos para o processo de troca de mensagens entre os agentes. Para isso, faz-se necessário o uso de ferramentas apropriadas que modelam não só os próprios agentes como também, regras que definem os protocolos de troca de mensagem. E a melhor ferramenta que existe atualmente no mercado é o framework do JAVA conhecido

por JADE (*Java Agent Development framework*) como próprio nome diz, é um framework específico para o desenvolvimento de agentes, ele trata da comunicação, do ciclo de vida e do monitoramento da execução dos agentes. É importante citar que a comunicação entre os agentes feita pelo JADE se baseia nos padrões de comunicação para tecnologias de sistemas multiagentes estabelecidos pela FIPA (*Foundation For Intelligent Physical Agents*), a partir disso é possível então estabelecer-se uma comunicação distribuída entre os agentes.

Mais detalhes sobre o JADE, FIPA e protocolos de comunicação, serão vistos em capítulos específicos que falarão sobre o assunto.

Capítulo 5

Ontologia

É originária da filosofia, ela lida com a natureza e a organização do ser. Esse termo foi introduzido por Aristóteles em Metafísica (Puc-Rio (2003) "apud" Maedche (2002)).

A ontologia deve ser explícita, formal[1] e descreve um conhecimento comum a um grupo determinado. Estruturalmente, deve possuir um conjunto de termos organizados com uma hierarquia associada, ou seja, uma taxinomia[2] que servirá de esquema para a base de conhecimentos.

Atualmente, vem se verificando o uso de ontologias em diversas áreas da informática, dentre elas destacam-se: o uso de ontologias para o processamento de linguagens naturais (Lopes (2005) "apud" Moreno e Hernández (2000)); na área de e-commerce (computeRs (2010)); gerenciamento do conhecimento (computeRs (2010)); web semântica (Lopes (2005) "apud" Maedche (2002)); engenharia de software(Falbo (2005)) e na área de educação (Severo et al. (2009) "apud" Maedche (2002)), que é o foco deste trabalho.

5.1 Definição

A literatura sobre ontologias apresenta uma série de definições distintas com pontos de vista diferentes e até mesmo complementares para uma mesma realidade. Segundo o artigo sobre ontologia da PUC-RIO, Guarino faz uma longa discussão sobre o significado preciso do termo dentro da Ciência da Computação, pois o seu siginificado tende a variar conforme o objetivo do uso da ontologia.

Uma ontologia pode ser definida como algo que fornece um conjunto de conceitos e termos para descrever um determinado domínio. Sendo que os termos definidos servem para descrever uma determinada realidade, que serão usados por uma base de conhecimentos, porém a ontologia permanece inalterada, desde que o domínio se mantenha inalterado.

Uma outra definição de ontologia, seria a definição proposta por Fensel (2001) proveniente do artigo editado pela PUC-RIO :

[1] passível de ser processada por uma máquina

[2] estabelece uma descrição e classificação das coisas

” Uma ontologia é uma especificação formal explícita de uma conceitualização compartilhada.”

Pode-se usar os mesmos conceitos descritos na ontologia para descrever diferentes realidades na base de conhecimentos.

A ontologia possui axiomas, ou seja, regras pertinentes ao domínio em questão.

A seguir, é apresentado um exemplo ilustrativo para diferenciar uma ontologia de uma base de conhecimentos. Neste exemplo, é definida uma ontologia para o domínio de circuitos eletrônicos, que incluiria conceitos específicos (por exemplo: circuito integrado, resistor, transistor processador, diodo, etc) e também relações entre esses conceito (por exemplo: o processador é um tipo de circuito integrado, etc) Puc-Rio (2003) e com isto podemos descrever uma base de conhecimentos que descreva o circuito de um rádio.

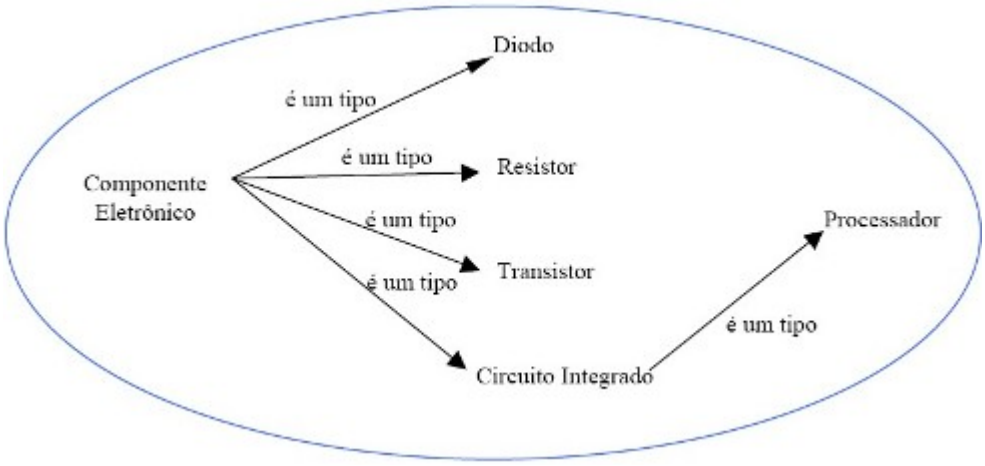

Figura 5.1: Exemplo de ontologia de domínio de circuitos eletrônicos

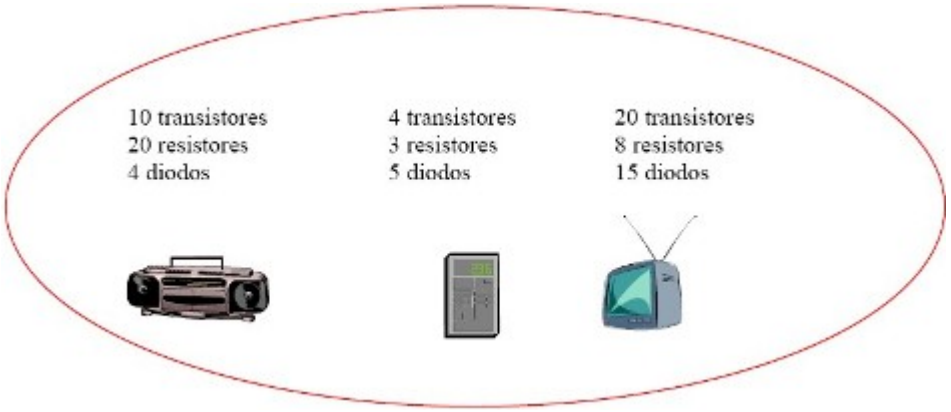

Figura 5.2: Exemplo da base de conhecimento que utiliza a ontologia de circuitos eletrônicos

5.2 Vantagens

Agora que já conhecemos algumas definições de ontologia, é importante destacar as vantagens de seu uso. Uma lista de vantagens da utilização de ontologias na ciência da computação é apresentada a seguir:

- Ontologias fornecem um vocabulário para representação do conhecimento. Esse vocabulário tem por trás uma conceitualização que o sustenta, evitando assim interpretações ambíguas.
- Ontologias permitem o compartilhamento do conhecimento. Por exemplo, considere uma ontologia para o domínio de livraria. Se ela estiver disponibilizada, várias livrarias podem usar o vocabulário fornecido por essa ontologia para construirem os seus catálogos sem a necessidade de refazer uma análise do domínio de livraria.
- Fornece uma descrição exata do conhecimento; cria uma conceitualização comum de palavras; diminuí diferentes interpretações sobre algo.
- Uma mesma conceitualização pode ser expressa em várias línguas.
- Pode-se estender o uso de uma ontologia genérica de forma a que ela se adeque a um domínio específico. Por exemplo, se alguém precisa de uma ontologia sobre carro para construir uma aplicação e só encontrar uma ontologia genérica de veículos, podendo utilizar essa ontologia estendendo-a para o domínio específico da aplicação, que no caso é de carro.

5.3 Componentes de um Ontologia

Uma descrição completa sobre os componentes de uma ontologia proposta por Gómez-Pérez (1999) devido ao seu grau de formalismo (Puc-Rio (2003) ”apud” Gómez-Peréz (1999)) é:

- Um conjunto de conceitos e uma hierarquia entre esses conceitos (taxinomia). Os conceitos podem ser abstratos; exemplo: força, ou concretos; exemplo: carro, elementares; exemplo: elétron ou compostos; exemplo: átomo, reais ou fictício.

  Um exemplo de taxinomia é o conceito ”homem” ser um subconceito do conceito ”pessoa” .
- Um conjunto de relacionamentos entre esses conceitos. Exemplo: pessoa e carro é o relacionamento de **ser dono do carro**.
- Fornece uma descrição exata do conhecimento; cria uma conceitualização comum de palavras; diminuí diferentes interpretações sobre algo.

- Um conjunto de funções. Uma função é um caso especial de relacionamento em que o conjunto de elementos tem uma relação única com um outro elemento. Exemplo: ser-país-biológicos, onde um conceito "homem" e um conceito "mulher" estão relacionados a um conceito "pessoa".

- Um conjunto de axiomas, ou seja, regras que são sempre verdade. Exemplo: toda a pessoa tem uma mãe; toda pessoa morre.

- Um conjunto de instâncias que são um conhecimento prévio existente na ontologia. A grande maioria das instâncias está na base do conhecimento.

5.4 Classificação das ontologias

Será apresentado um sistema de classificação que usa a conceitualização como o principal critério para a classificação. Este modelo foi criado por Maedeche e propõe 4 tipos de classificação de uma ontologia:

- **Ontologias de alto-nível:**

  Descrevem conceitos muito gerais como espaço, tempo, evento, etc. Esses conceitos tipicamente são independentes de um problema particular ou domínio. Sendo assim, é bem razoável ter-se uma ontologia de alto-nível compartilhada por grandes comunidades de usuários.

- **Ontologias de domínio:**

  Descrevem o vocabulário relacionado a um domínio genérico, através da especialização de conceitos introduzidos nas ontologias de alto-nível. Exemplo: ontologias de veículos, documentos, etc.

- **Ontologias de tarefas**

  Descrevem um vocabulário relacionado a uma tarefa ou atividade genérica, através da especialização de conceitos introduzidos nas ontologias de alto-nível.

- **Ontologias de aplicação**

  São as ontologias mais específicas por serem utilizadas dentro das aplicações. Esse tipo de ontologia especializa conceito tanto das ontologias de domínio, como também das de tarefas. Exemplo: uma aplicação que trabalhe com carros de luxo. Essa ontologia especializará o conceito da ontologia de veículos (que é uma ontologia de domínio).

  As ontologias de alto-nível possuem maior capacidade de reúso, por definir conceitos genéricos, enquanto as ontologias de aplicação são as que possuem menor capacidade de reúso, por definirem conceitos relativos a uma aplicação específica.

A seguir é apresentada uma figura que ilustra os diferentes tipos de ontologias e suas relações

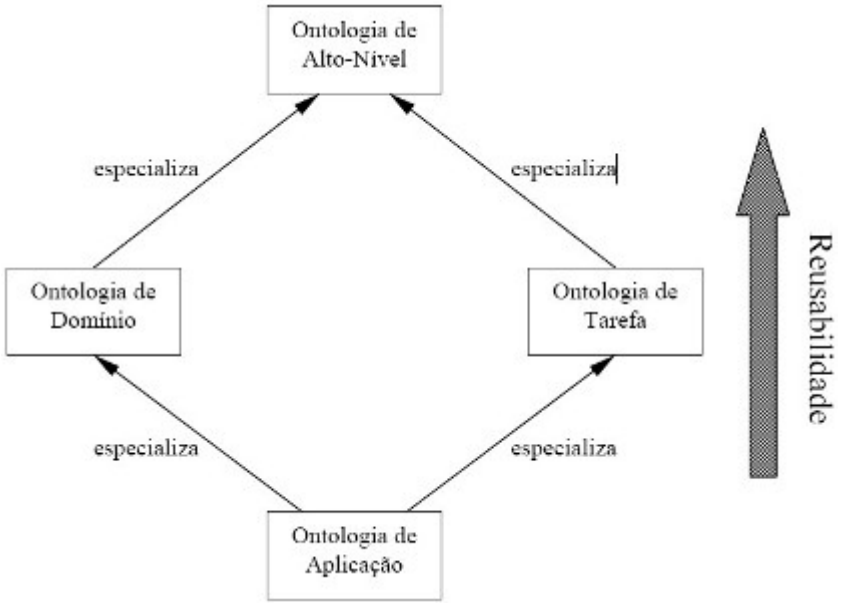

Figura 5.3: Diferentes tipos de ontologias e suas relações

5.5 Principais áreas de aplicação das ontologias

1. **Processamento de linguagem natural:**

   O conhecimento do domínio é muito importante para compreensão do texto, que pode ser dado por meio de uma ontologia sobre o domínio de discurso do texto. O uso de ontologias é de vital importância por 2 motivos:

   - Auxilia a elucidação de ambiguidade de compreensão existentes no texto. Com a utilização de uma ontologia sobre o domínio de discurso do texto se reduzem os problemas de ambiguidade.
   - A ontologia funciona como um dicionário de conceitos dentro do domínio do texto.

2. **Gestão de conhecimento:**

   Aquisição, manutenção e acesso ao conhecimento dentro de uma organização. A tecnologia de ontologias dentro dessa área auxilia das seguintes formas:

   - Ontologias fornecem uma estrutura básica sobre a qual se constroem bases conhecimentos.

- Uma dificuldade dos sistemas de gestão de conhecimento é o fato de que muito do conhecimento presente dentro das organizações se encontra em uma forma não estruturada. Usando ontologias, pode-se notar informações semânticas em artefatos de informação não estruturados, visando assim a obtenção de resultados mais precisos em pesquisas de informação.

3. **Web Semântica:**

   A tecnologia de ontologias entra para fornecer uma estrutura semântica para anotação das páginas da Web. Espera-se que com uma estrutura fornecida pela Web semântica seja possível obter buscas mais precisas (uma vez que a semântica estará colocada em linguagem formal) e dar uma maior capacidade para os agentes de software que utilizem conteúdo da Web.

5.6 Abordagens para o desenvolvimento de uma ontologia

Assim como na atividade de desenvolvimento de software, o desenvolvimento de uma ontologia pode ser feito a partir de várias abordagens, cada uma apresentando suas vantagens e desvantagens.

Conforme o caso deve-se escolher a abordagem mais adequada, segundo a classe da ontologia a ser desenvolvida.

Algumas das principais abordagens para o desenvolvimento de uma ontologia, são apresentadas a seguir:

1. **Abordagem partindo de uma inspiração:** O desenvolvedor começa com uma premissa sobre porquê da necessidade de uma ontologia. Usando a sua imaginação, criatividade e visão pessoal sobre o domínio de interesse da ontologia, o desenvolvedor constrói a ontologia que visa solucionar essa necessidade encontrada.

   A principal vantagem dessa abordagem está no facto de ser possível obter uma ontologia inovadora para a resolução de um problema. A principal desvantagem dessa abordagem é que a ontologia gerada é resultado de uma visão muito subjetiva, do próprio desenvolvedor sobre o domínio, gerando assim dificuldades para a sua adoção.

2. **Abordagem indutiva:** A ontologia é desenvolvida através da observação, exame e análise de casos específicos no domínio de interesse da ontologia. A ontologia resultante para um caso específico é aplicada para todos os outros casos do mesmo domínio.

   A principal vantagem dessa abordagem é ser possível utilizar uma mesma ontologia para solucionar uma série de casos dentro de um domínio. Porém a principal dificuldade é a incapacidade dessa ontologia em cobrir domínios menos específicos, ou seja, mais amplos.

3. **Abordagem dedutiva:** Nessa abordagem, a partir de uma ontologia mais genérica de um domínio, chega-se a uma ontologia mais específica e restritiva dentro de um subconjunto do domínio.

   A vantagem dessa abordagem está na capacidade de utilizar ontologias genéricas para geração de ontologias específicas. Porém a desvantagem é a necessidade de que já existe inicialmente uma ontologia genérica.

4. **Abordagem sintética:** O principal papel do desenvolvedor é fazer, de uma forma coerente, a composição de diversas ontologias em uma ontologia unificada.

   Por unir múltiplas ontologias, essa ontologia unificada tem grande propensão em ser aceita pelos usuários das ontologias usadas para a unificação. Dessa forma, essa ontologia unificada fornece uma forma coerente para a comunicação entre esses usuários das ontologias múltiplas.

   A desvantagem dessa abordagem está na dificuldade muitas vezes existente para estabelecer uma ligação de forma harmónica entre um conjunto grande de ontologias.

5. **Abordagem colaborativa:** Nessa abordagem, o processo de desenvolvimento da ontologia não se encontra ao encargo de uma única pessoa e sim de um conjunto de pessoas.

   A vantagem está em aumentar as chances da ontologia ser aceita como um padrão para um determinado domínio, pois possui a contribuição de um conjunto de pessoas para a sua formação. Outro ponto importante é o fato de a ontologia resultante desse processo ser muito elaborada do que nas abordagens supracitadas devido ao facto de mesma absorver pontos de vista de mais de um indivíduo.

   A principal dificuldade está em coordenar o grupo de desenvolvimento, quando esse é muito grande.

5.7 Metodologias para desenvolvimento de uma ontologia

O processo de construção de ontologias ainda está pouco desenvolvido[3]. Sendo que a maioria dos desenvolvedores usa os seus próprios critérios, e por isso muitos deles passam diretamente da fase de aquisição de conhecimento para a fase de implementação, o que pode causar os seguintes problemas:

- Os modelos conceituais ficam implícitos no código da implementação.
- Dificuldades de reúso da ontologia, porque as decisões de projeto estão implícitas no código.

[3] Algumas ferramentas para construção de ontologias já têm sido desenvolvidas, como é o caso da ferramenta Protégé 2000 com Plug-In OWL (Ontology Web Language)

- Problemas de comunicação devido às dificuldades que o expert no domínio da ontologia tem para entender o código da implementação.
- Dificuldades no desenvolvimento de ontologias complexas, pois a passagem da aquisição de conhecimento para a implementação é muito abrupta.
- Dependendo da linguagem escolhida para a codificação pode-se limitar a capacidade de descrição conceitual do domínio da ontologia.

Dessa forma, se faz necessária a adoção de uma metodologia afim de reduzir as dificuldades acima citadas. Uma metodologia será abordada na seção seguinte.

5.8 Metodologia para criação de uma ontolgia

Esta metodologia foi proposta por Mariano Fernández, Asunción Gomez Pérez e Natalia Juristo. É uma metodologia bastante amadurecida, pois além de descrever mais a fundo a metodologia (passos seguidos e artefatos criados), fornece também um processo de desenvolvimento de ontologia e também propõe um ciclo de vida baseado em evolução de protótipos.

Como criar uma ontologia? Qual a metodologia criada por estes autores?

Eles criaram um conjunto de atividades que devem ser cumpridas e que auxiliam na construção de uma ontologia, este conjunto de atividades pode ser comparado ao conjunto de atividades que devem ser realizadas quando se pretende implementar um projeto de software.

- **Planejamento:** identifica as tarefas principais da ontologia, e planeja a utilização de recursos.
- **Especificação:** Define por que a ontologia está sendo construída, e quem serão os seus usuários.
- **Aquisição de conhecimento:** Adquire conhecimento sobre o domínio da ontologia. Pode ser feito: através de entrevistas com o expert na ontologia a ser criada, análise de livros sobre o domínio, etc.
- **Conceitualização:** criação de um modelo conceitual que descreve o problema e a sua solução.
- **Formalização:** transforma o modelo conceitual em um modelo formal.
- **Integração:** procurar integrar o máximo possível as ontologias existentes à nova ontologia.

- **Implementação:** implementa a ontologia em uma linguagem formal de modo que ela seja computável.
- **Avaliação:** Avalia a ontologia.
- **Documentação:** documentação da ontologia para facilitar futuro reúso e manutenção.
- **Manutenção:** Executar a manutenção da ontologia quando necessária.

As atividades de aquisição de conhecimento, avaliação e documentação, são executadas em todos os estágios do ciclo de vida. O planejamento é a primeira atividade a ser executada. A maior parte da atividade de aquisição é feita simultaneamente com o estágio de especificação da ontologia e vai decaindo conforme o ciclo de vida avança. A maior parte da atividade de avaliação da ontologia é feita durante os estágios iniciais do ciclo, de forma a diminuir a propagação de erro. A atividade de documentação deve ser realizada em todos os estágios.

É importante conhecer os artefactos gerados na fase de desenvolvimento e os que são gerados na parte de conceitualização da ontologia. Os artefatos gerados são de fácil compreensão para o desenvolvedor da ontologia e para o expert no domínio. Os artefatos também fornecem uma boa documentação sobre a ontologia.

O artefato gerado durante o estágio de especificção é um documento que cobre o objetivo principal, o propósito, a granularidade e o escopo da ontologia. Essa especificação deve ser completa e concisa.

Os principais artefatos propostos na metodologia que são gerados durante o estágio de especificação e conceitualização são:

- Glossário de termos: inclui todos os termos do domínio (conceitos, instâncias, atributos,etc) e suas descrições que devem ser claras e concisas.
- Árvore de classificação de conceitos: define relações tais como: subclasse e exclusividade mútua entre classes. Dessa forma são definidas várias taxinomias de domínio, sendo que cada uma deve gerar uma ontologia. Verificar a consistência(devem existir ciclos na árvore) e a concisão (não deve existir repetição de conceitos).
- Diagrama de relações binárias: esse diagrama estabelece os relacionamentos entre conceitos da mesma ontologia e de ontologias diferentes.
- Dicionário de conceito: documento que contém todos os conceitos do domínio, instâncias desses conceitos, e opcionalmente, sinónimos e antónimos dos conceitos, Para cada árvore de classificação deve existir um dicionário.
- Tabela de relações binárias: especifica o nome da relação, o nome dos conceitos de origem e destino da relação, a relação inversa e a cardinalidade da relação. Para cada árvore de classificação deve existir uma tabela desse tipo.

- Tabela de atributos de instância: descreve cada atributo de instância do dicionário de conceitos. Estes atributos são definidos nos conceitos e valorados nas instâncias. Para cada atributo de instância deve ser especificado: nome, tipo de valor, unidade de medida para valores numéricos, precisão de valores numéricos, faixa de valores aceitos, cardinalidade máxima e mínima. Atributos ou constantes utilizadas para inferir o valor do atributo, entre outras.

- Tabelas de atributos de classe: Descreve os atributos de classe no dicionário de conceitos. Esses atributos têm um valor que é dado pelo conceito, ou seja, o valor será o mesmo para todas as instâncias do conceito. Para cada atributo de classe deve ser especificado: nome do atributo, precisão dos valores numéricos, cardinalidade máxima e mínima, atributos que podem ser inferidos desse atributo e referências.

- Tabela de axiomas: define conceitos por meio de expressões lógicas que são sempre verdadeiras.

- Tabela de constantes: especifica para cada constante: nome, descrição em linguagem natural, o tipo lógico do valor, seu valor constante, sua unidade de medida, os atributos que podem ser inferidos usando a constante de referências.

- Tabela de fórmulas: descreve cada fórmula das tabelas de atributos de instância. Essa tabela é usada para inferir os valores numéricos dos atributos de instância. **Manutenção:** Executar a manutenção da ontologia quando necessária.

- Árvore de classificação de atributos: mostra graficamente os atributos e as constantes relacionados a uma sequência de cálculos de atributo raiz. É usado para validar que todos os atributos usados na fórmula fazem sentido e nenhum atributo foi omitido.

- Tabela de instância: Desecreve as instâncias do domínio.

Outras formas de se representar uma ontologia é fazendo uso de ferramentas para criação de ontologias, que apesar de poucas, elas podem ser facilmente obtidas através da internet e também podemos usar diagramas UML, estes também funcionam bem para representação de ontologias.

5.9 Processos de aprendizagem baseados na interação social

Nesta secção pretende-se mostrar como será definido um modelo computacional para o processo de aprendizagem partindo de categorias de mediação e interação social através de uma ontologia. O desenvolvimento do sujeito seja este cognitivo, afetivo, social está estreitamente relacionado com a interação social. O foco de interesse aqui é o aspecto educacional e fator de influência que a interação social exerce sobre o desenvolvimento do sujeito (aluno).

Será mostrada uma ontologia criada para EaD baseada no perfil social do aluno, visto que este é foco do trabalho. Esta ontologia auxilia o aluno em suas ações de aprendizagem. E para isso é importante que se defina um vocabulário e que ele seja conceitualizado. Para além da ontologia de aluno, também será criada uma ontologia para grupo de aluno, que define um vocabulário conceitualizado e possui regras que estabelecem a comunicação entre os alunos pertencentes ao grupo.

A ontologia (aluno e grupo) proposta neste projeto tem por finalidade servir como base de conhecimento para elaboração de uma infra-estrutura de software de apoio a mediação da aprendizagem em ambientes de ensino a distância.

5.9.1 Desenvolvimento em uma prespectiva social

Para muitos autores como por exemplo Vygotsky (2001) p.161 (”apud” Severo et al. (2009) p.2) a ação do homem no mundo tem efeitos físicos de mudanças no mundo e efeitos psicológicos sobre o próprio homem. E. Wertsch(”apud” Severo et al. (2009) p.2), diz que a mediação é um processo dinâmico, no qual intervêm ferramentas e signos numa ação envolvendo o potencial das ferramentas para modelar a ação e os uso das mesmas por parte dos indivíduos.

Assim, do ponto de vista educacional , ações de mediação são importantes, não somente para o desenvolvimento cognitivo do aluno, mas também, para o desenvolvimento de sua autonomia. Em ambientes virtuais de ensino/aprendizagem não poderia ser diferente, eles apresentam mecanismos que permitem o ensino como aprendizagem flexível, independentes de espaço e tempo para que possam ser desenvolvidos (Severo et al. (2009)).

Além disso, aprendizagem, mediação e comunicação são conceitos intimamente ligados e que representam um ponto chave para a qualidade do ensino a distância. O professor exerce uma importante função no desempenho de sua atividade de mediação, buscando a realização de ajustes para que o aluno possa desenvolver um melhor desempenho, de acordo com as necessidades individuais do estudante.

O processo de regularização proposto por Diaz (1993) (”apud” Severo et al. (2009) p.3) e adaptado por Passerino(2005) (”apud” Severo et al. (2009) p.3) é realizado em níveis que partem de um controle, passando por autocontrole e chegando a auto-regulaçao. O controle é externo ao sujeito, realizado pelo sujeito mais experiente e pode assumir duas dimensões: direta ou indireta. O Controlo Direto verifica-se através de ordens, diretivas e perguntas diretivas. Já o Indireto é feito através de perguntas perceptivas, conceituais, procedimentais, e culminam no afastamento físico que entra na categoria de auto-controle.

Segundo Diaz (1993), considera auto-controle, como sendo a realização, por parte do sujeito, de uma ação esperada obedecendo a um tutor internalizado. Ou seja, a figura do sujeito mais experiente que era real e externo no processo anterior, agora é interna. A auto-regulação não pode ser observada de forma direta , pois a mesma acontece internamente ao sujeito, mas considera-se que o sujeito está na categoria de auto-controle quando organiza, planeja

e executa a ação sem intervenção de nenhum mediador externo. Assim a auto-regulação é o plano de ação concebido pelo sujeito que se converte no seu próprio tutor. A diferenciação central entre auto-controle e auto-regulação não passa pela internalização das ordens e diretivas do tutor, mas na capacidade emergente de planejar e definir objetivos próprios organizando funcionalmente sua conduta para os mesmos e adaptando-a de acordo com o contexto.

Algumas evidências de mediação em ambientes de ensino a distância serão apresentadas.

O trabalho de Hack (2009) ("apud" Severo et al. (2009) p.3) aponta que as evidências de mediação em ambientes de ensino a distância mostram-se de grande importância no desenvolvimento de processos educativos . Os dados coletados a partir de pesquisa empírica mostraram que do ponto de vista pedagógico,o papel do mediador e seu ajuste nas ajudas oferecidas dentro das categorias identificadas de controle, auto-controle e auto-regulação são importantes para a autonomia do aluno e apropriação do conhecimento.

Segundo Severo, a hipótese básica do estudo é que as categorias de controle, auto-controle e auto-regulação, podem ser aplicadas de forma produtiva na análise e ajustes dos processos de mediação que ocorrem em ambientes de EaD, pois além de ajudarem na compreensão dos processos de mediação nesses ambientes, podem servir de base para a construção de ferramentas para ambientes de EaD que apóiem tanto professores quanto alunos a atingir processos de mediação produtivos,do ponto de vista pedagógico.

Consequentemente, num processo de educação, muitas vezes "os ajustes" na mediação são feitos no andamento dos trabalhos e seu acompanhamento e regulação só é possível a partir "da leitura" do contexto inter-subjetivo que se estabelece na interação social.

Capítulo 6

Agentes Inteligentes e a Educação

A Inteligência Artificial (IA) é uma área da Ciência da Computação que dá ênfase ao estudo de sistemas computacionais inteligentes, ou seja, sistemas que imitam o comportamento do ser humano - compreensão de linguagem, aprendizado, redes neurais, entre outras. Devido as aplicações computacionais da IA, muitos pesquisadores têm utilizado suas técnicas na área de educação, devido às suas potencialidades, no que se refere ao desenvolvimento de ambientes de ensino-aprendizagem inteligentes(Bassani et al. (2007)).

Segundo Bassani et al. (2007), ambientes de ensino-aprendizagem inteligentes são sistemas que, na interação com o aluno, modificam suas bases de conhecimento através das percepções feitas. Possuem a capacidade de aprender e adaptar estratégias de ensino de acordo com o desempenho do aluno, e se caracterizam, sobretudo, por construir um Modelo Cognitivo desse aluno, através da interação, da formulação e da comprovação de hipóteses sobre seu conhecimento. Possui, contudo, a capacidade de adequar estratégias de ensino-aprendizagem ao aluno e à situação atual Viccari(1990) ("apud" Bassani et al. (2007) p.2). Sendo assim, estes ambientes computacionais têm um grande potencial para uso pedagógico, uma vez que envolve o usuário/aluno no processo, potencializando alto grau de interação. Desta forma podemos ver que o uso do computador na educação pode trazer inúmeras vantagens, tais como: estender o processo de ensino-aprendizagem; dar uma maior qualidade no processo de Educação à Distância (EaD) e ser uma boa ferramenta para o paradigma construcionista, entretanto é importante referir que este também traz desvantagens no que concerne a ausência de contato presencial dificultando o acompanhamento e avaliações informais, porém, podemos usar o próprio sistema (inteligente) que vai auxiliar o professor na tarefa de acompanhar e avaliar o aluno.

Um dos ramos da IA é a Inteligência Artificial Distribuída (IAD) que soluciona problemas partindo da individualidade para a coletividade, ou seja, a inteligência é vista como emergente da ação e interação de entidades (agentes) autônomas.

Segundo alguns autores, a IAD está dividida em 3 áreas: **Resolução Distribuída de Problemas (RDP)**; **Inteligência Artificial Paralela (IAP)** e **Sistemas Multiagentes**

(SMA) que é o foco do nosso trabalho, portanto, os itens seguintes abordarão os seus conceitos básicos, mas antes vale ressaltar o que significa Sistemas Multiagentes.

Os Sistemas Multiagentes são caracterizados pela existência de um certo número de entidades autônomas (agentes), heterogêneas e potencialmente independentes, trabalhando juntas para resolver um problema.

6.1 Agentes Inteligentes

Definir agentes com um conceito único ou padrão é muito difícil, pois existem varias abordagens e pontos de vista de diferentes autores. Além disso, devido às suas mais diversas aplicações uma definição precisa torna-se cada vez mais complicada e variada.

RUSSEL e NORVIG (1995) definem um agente como um sistema capaz de perceber, através de sensores, e agir, através de atuadores, em um dado ambiente. Um agente também pode ser uma entidade à qual se atribuem estados, denominados estados mentais, tais como crenças, decisões, capacidades, compromissos e objetivos (conceitos análogos ou similares aos humanos) (SHOHAM (1993)).

Uma definição bem completa é dada por Bradshaw (1997), que descreve um agente como sendo uma entidade de software que funciona de forma contínua e autônoma em um ambiente. Segundo este autor, um agente também deve ser capaz de perceber e atuar no seu ambiente, de forma flexível e inteligente, sem requerer intervenção ou orientação humana constantes. Idealmente, um agente deve aprender através da experiência, comunicar-se e cooperar com outros agentes que porventura coexistam no mesmo ambiente. Ainda, um agente deve poder mover-se de um local para outro a fim de satisfazer os seus objetivos.

Um sistema pode utilizar-se da interação entre vários agentes (modelo multiagente).

A partir disto, podemos destacar as seguintes propriedades para agentes:

- **Autonomia:** Um agente deve agir sem intervenção humana direta, portanto deve possuir algum tipo de controle sobre suas ações e seu estado interno;
- **Reatividade:** Um agente deve ser capaz de reagir aos estímulos externos produzidos pelo seu ambiente ou por outros agentes;
- **Pró-atividade:** Um agente não somente reage ao seu ambiente, mas também deve exibir um comportamento orientado à satisfação de seus objetivos (Orientação a Objetivos).
- **Intencionalidade:** Capacidade de representação explícita dos seus objetivos;
- **Racionalidade:** Habilidade de agir de forma a atingir seus objetivos e não contra eles;
- **Continuidade temporal:** Persistência de identidade por longos períodos de tempo;

- **Sociabilidade:** Habilidade de interação com outros agentes através de mecanismos de comunicação;
- **Benevolência:** Capacidade de cooperação com outros agentes;
- **Adaptabilidade:** Capacidade de aprender através da experiência;
- **Mobilidade:** Capacidade de mover-se de um ambiente para outro.

Em determinadas aplicações, algumas características são mais importantes que as outras, portanto, um agente não precisa ter necessariamente todas elas. O comportamento do agente é dado pelo ambiente no qual ele está inserido e modelado através do seu respectivo projeto.

A FIGURA 6.2 ilustra de forma esquemática a noção elementar de um agente. Nesta figura podemos ver alguns dos atributos citados no parágrafo anterior como reatividade (percepções e ações) e pró-atividade (estado interno e ações).

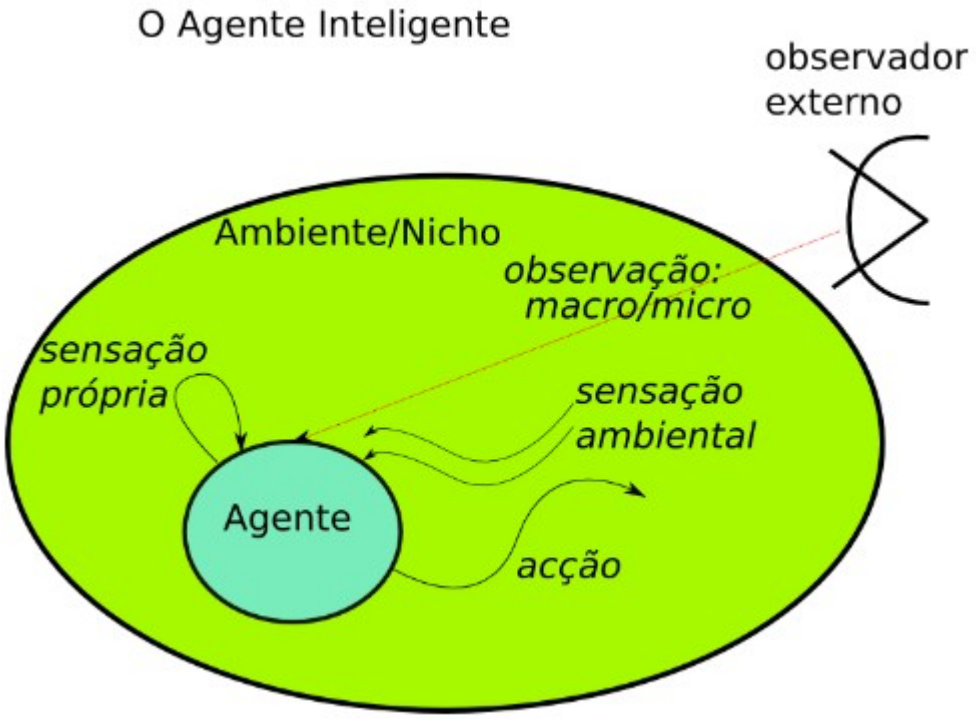

Figura 6.1: Esquema de agente

6.2 Categoria de Agentes

Para melhor compreensão da aplicabilidade da tecnologia de agentes, é interessante referenciar sobre os diferentes tipos de agentes e suas diferenças, a fim de termos uma noção mais clara da utilidade no emprego de agentes. A classificação dos agentes pode ser feita quanto à mobilidade, ao relacionamento entre agentes e quanto a capacidade de raciocínio.

- **Agentes Móveis:** são agentes que tem a mobilidade como característica principal. Isto é, uma capacidade de mover-se seja por uma rede interna local (intranet) ou até mesmo pelo Web, transportando-se pelas plataformas levando dados e códigos. Seu uso tem crescido devido alguns fatos como uma heterogeneidade cada vez maior das redes e seu grande auxílio em tomadas de decisões baseadas em grandes quantidades de informação.

- **Agentes situados ou estacionários:** são aqueles opostos aos agentes móveis. Isto é, são fixos em um mesmo ambiente e ou plataforma. Não se movimentam em uma rede e muito menos na Web.

- **Agentes Competitivos:** são agentes que "competem" entre si para a realização de seus objetivos ou tarefas. Ou seja, não há colaboração entre os agentes.

- **Agentes Coordenados ou Colaborativos:** agentes com a finalidade de alcançar um objetivo maior, realizam tarefas especificas porém coordenando-as entre si de forma que suas atividades se completem.

- **Agentes Reativos:** é um agente que reage a estímulos sem ter memória do que já foi realizado no passado e nem previsão da ação a ser tomada no futuro. Não tem representação do seu ambiente ou de outros agentes e são incapazes de prever e antecipar ações. Geralmente atuam em sociedades como uma colônia de formiga por exemplo. Baseiam-se muito também na "teoria do caos" no qual afirma que até mesmo no Caos existe uma "certa organização". No caso da colônia de formigas, por exemplo, uma única formiga não apresenta muita inteligência mas quando age no grupo comporta-se o "todo" como uma entidade com uma certa inteligência, ou seja, a força de um agente reativo vem da capacidade de formar um grupo e construir colônias capazes de adaptar-se a um ambiente.

- **Agentes Cognitivos:** esses, ao contrário dos agentes reativos, podem raciocinar sobre as ações tomadas no passado e planejar ações a serem tomadas no futuro, ou seja, um agente cognitivo é capaz de "resolver" problemas por ele mesmo. Ele tem objetivos e planos explícitos os quais permitem atingir seu objetivo final. Ferber afirma que para que isso se concretize, cada agente deve ter uma base de conhecimento disponível, que compreende todo os dados e todo o "know-how" para realizar suas tarefas e interagir com outros agentes e com o próprio ambiente. Sua representação interna e seus mecanismos de inferência o permitem atuar independentemente dos outros agentes e lhe dão uma grande flexibilidade na forma de expressão de seu comportamento. Além disso, devido a sua capacidade de raciocínio baseado nas representações do mundo, são capazes de ao mesmo tempo memorizar situações, analisá-las e prever possíveis reações para suas ações.

6.3 Aplicações dos Agentes

Algumas das aplicações dos agentes inteligentes serão apresentadas de seguida:

- **Agentes no processo ensino-aprendizagem:** Utilização de agentes no aprendizado e no processo de treinamento de usuários.
- **Agentes na indústria:** A grande vantagem da aplicação de agentes na produção industrial é o fato de que muitos dos sistemas que atuam neste propósito são complexos e isolados. O papel dos agentes seria integrá-los, afim de compartilhar informações entre os sistemas.
- **Agentes em simulação:** Agentes podem simular situações dando um grau maior de veracidade. Tanto na área de entretenimento quanto na área de pesquisa tecnológica e militar.
- **Agentes em realidade virtual:** Nessas aplicações o agente atua como um participante que auxilia e monitora certas atividades e usuários, assistindo-os ou ajudando-os quando necessário, principalmente em ambientes virtuais muito complexos.
- **Agentes na prestação de serviços:** Nesse contexto os agentes irão agregar valor a certos serviços, gerenciando a informação a fim de satisfazer as necessidades dos clientes. Podemos citar como exemplo:

    1. manutenção e atualização de bases de dados explorando a informação (data mining);
    2. comércio eletrônico, onde agentes procuram preços mais convenientes ou ofertas e produtos com as especificações desejadas pelo usuário;
    3. gerência de correio eletrônico e filtragem de mensagens, funcionando como um assistente pessoal;
    4. gerência de produção e muitas outras aplicações;

- **Agentes em redes de computadores:** As suas funções nessas aplicações variam muito. Varia desde definição de melhores rotas, controle de manutenção de equipamentos, gerenciamento de dados até a monitoramento de atividades e gerência da própria rede.

6.4 Sistemas Multiagentes

Um único agente pode requerer muito conhecimento para resolver problemas complexos. Em alguns casos, o problema é tão complexo que um agente não pode, por ele mesmo, resolvê-lo

(BRENNER et al. (1998)). Além disso, muitos problemas têm características distribuídas e, portanto, necessitam de unidades distribuídas de conhecimento para a sua resolução. Portanto, seria interessante a existência de unidades independentes que resolvessem cada questão individualmente e que juntos resolvessem o problema todo, ou seja, um sistema composto por mais de um agente que trabalham juntos para alcançarem um objetivo comum. Esses sistemas são denominados Sistemas Multiagentes (SMA).

Em um SMA, cada agente tem os seus próprios objetivos e contacta outros agentes para obter informações ou contribuir na solução de um problema maior. Logo, sob o ponto de vista de constituição de sociedade de agentes, a fundamentação dos SMA é baseada na interação social de seus indivíduos.

Segundo BRENNER et al. (1998), uma vantagem dos sistemas multiagentes, é o fato deles permitirem a integração de agentes já existentes. Isso faz com que a solução do problema não demande o desenvolvimento de novos e especializados agentes, ou seja, o conhecimento dos agentes existentes pode ser utilizado para resolver determinado problema.

O conhecimento especializado de cada agente não pode ser utilizado por outro agente e nenhuma estratégia comum de solução pode ser desenvolvida sem mecanismos de comunicação e cooperação (BRENNER et al. (1998)). Deste modo, a resolução de problemas através de SMA só é possível quando os agentes têm capacidades de se comunicarem e de cooperarem uns com os outros. Além disso, é necessária alguma forma de coordenação do comportamento dos agentes.

6.5 Comunicação entre agentes

A comunicação é fundamental para permitir que haja colaboração, negociação, cooperação entre agentes.

Em sistemas multiagentes, é necessário que a comunicação seja disciplinada para que os objetivos sejam alcançados efetiva e eficientemente, necessitando assim uma linguagem que possa ser entendida pelos outros agentes presentes no ambiente. Essa comunicação tem como principal objetivo à partilha do conhecimento com os outros agentes e a coordenação de atividades entre eles, ou seja, ela deve permitir que agentes troquem informações entre si e coordenem suas próprias atividades resultando sempre em um sistema coerente.

Existem diversas maneiras para os agentes trocarem informações uns com os outros em sistemas multiagentes. Para Baker (1997) agentes podem trocar mensagens diretamente, chamada também por alguns autores como comunicação direta, podem comunicar-se através de um agente "facilitador" especial em sistema "federado" (comunicação assistida), podem também utilizar uma comunicação por difusão de mensagens (broadcast) e até utilizar o modelo de comunicação através de blackboard ou quadro negro.

6.5.1 Comunicação direta

A comunicação direta, ou comunicação via troca de mensagens direta, cada agente comunica diretamente com qualquer outro agente sem qualquer intermediário. Na FIGURA 6.3 há um estabelecimento de uma ligação direta (ponto-a-ponto) entre os agentes através de um conjunto de protocolos que garante a chegada de mensagens com segurança. Nesse tipo de comunicação faz-se necessário que cada agente envolvido tenha conhecimento da existência dos outros agentes e da forma de como endereçar mensagens para eles. A principal vantagem deste tipo de comunicação entre agentes é o fato de não existir um agente coordenador da comunicação. Isso porque esses agentes coordenadores podem levar a um ”gargalo” ou até mesmo ao bloqueio do sistema caso haja, por exemplo, um grande número de troca de mensagens. Entretanto, algumas desvantagens podem ser verificadas como o custo da comunicação se tornar muito grande, principalmente quando há um grande número de agentes no sistema, e a própria implementação que se torna muito complexa em comparação às outras formas de comunicação.

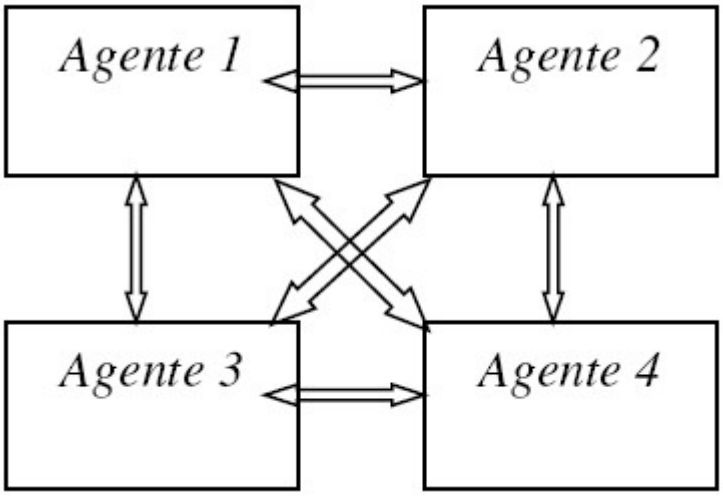

Figura 6.2: Comunicação Direta entre agentes

6.5.2 Comunicação por sistemas federados

Na comunicação por sistemas federados Baker (1997) ou comunicação assistida, os agentes utilizam algum sistema ou agente especial para coordenar suas atividades, ou seja, uma estrutura hierárquica de agentes é definida e a troca de mensagens dá-se através agentes especiais designados ”facilitadores” ou mediadores (Veja a figura 6.4). Essa é uma alternativa bem popular à comunicação direta pois diminui muito o custo e a complexidade necessária aos agentes individuais na realização da comunicação. Geralmente é utilizado quando o número de agentes dentro do sistema é muito grande.

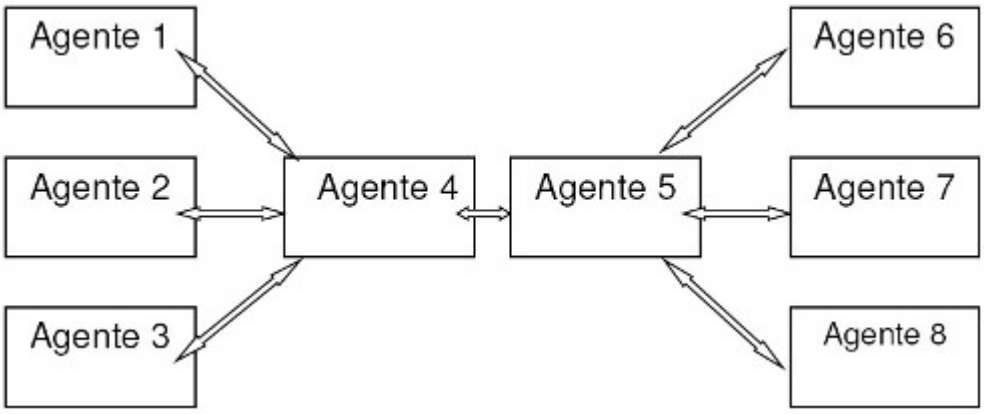

Figura 6.3: Comunicação por sistema federado ou comunicação assistida

6.5.3 Comunicação por brooadcast

A comunicação por difusão de mensagens ou broadcast geralmente é utilizada em situações onde a mensagem deve ser enviada para todos os agentes do ambiente ou quando o agente remetente não conhece o agente destinatário ou seu endereço. Em suma, todos os agentes recebem a mesma mensagem enviada. É semelhante a comunicação por broadcast em redes de comunicação.

6.5.4 Comunicação por blackboard

Comunicação por quadro negro ou blackboard ou ainda quadro de avisos, segundo Baker (1997), é bastante usada na Inteligência Artificial como modelo de memória compartilhada, ou seja, nada mais é do que um repositório onde os agentes escrevem mensagens a outros agentes e obtêm informações sobre o ambiente.

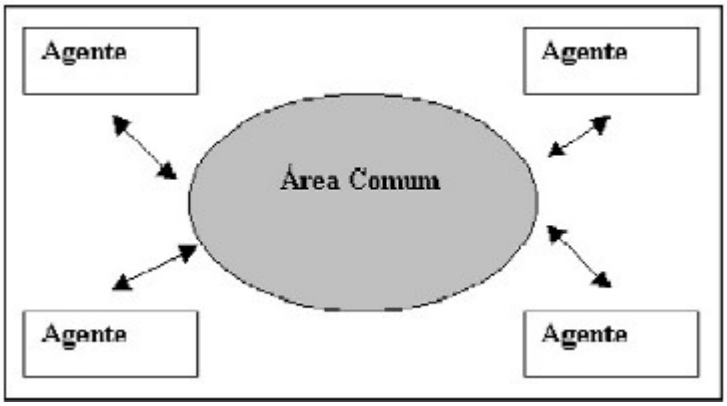

Figura 6.4: Comunicação por quadro negro

6.6 Protocolos de comunicação entre agentes

O mecanismo de troca de mensagens necessita da definição de um protocolo de comunicação que especifique o exato processo de comunicação, um formato para as mensagens e uma linguagem de comunicação. Esses requisitos nos levam às Linguagens de Comunicação de Agentes (ACL) e aos serviços de transportes de mensagens.

Uma ACL determina um meio através do qual uma atitude (afirmação, requisição, consulta, etc) a respeito do conteúdo da troca de conhecimentos é comunicada (Finin et al. (1997)).

Finin et al. (1997) destaca que para uma linguagem de comunicação possa ser adequada à comunicação entre agentes deve possuir as seguintes propriedades:

- **Agentes forma:** Uma ACL deve ser declarativa, sintaticamente simples e legível. Ela deve proporcionar fácil realização de *parsing* ;
- **O conteúdo:** Uma ACL deve fornecer um conjunto bem definido de ações de comunicação (primitivas). Além disso, ela deve poder ser estendida de forma a se adaptar bem com outros sistemas;
- **A semântica:** Uma ACL não deve ter uma semântica ambígua, ela deve ser baseada numa teoria e considerar tempo e local. A descrição da semântica é normalmente feita através da linguagem natural;
- **A implementação:** Uma ACL deve ser eficiente tanto na velocidade como na boa utilização da largura de banda da rede. Deve possuir uma interface amigável, isto é, detalhes devem ficar escondidos.

  Deve também ser possível realizar-se uma implementação parcial da linguagem para que agentes possam utilizar apenas um subconjunto de primitivas de ações de comunicação;
- **A rede:** Uma ACL deve adaptar-se bem com as tecnologias atuais de rede. Deve suportar os tipos básicos de conexão ponto-a-ponto, multicast, broadcast. Conexões síncronas e assíncronas também devem ser suportadas. Um conjunto rico de primitivas deve ser disponibilizado de forma que possam ser desenvolvidos linguagens e protocolos de alto-nível e esses mecanismos devem ser independentes da camada de transporte;
- **O ambiente:** Os ambientes em que os agentes inteligentes são utilizados frequentemente são distribuídos, heterogêneos e dinâmicos. Portanto, deve ser possível a integração com outras linguagens e protocolos;
- **A confiabilidade:** A linguagem deve proporcionar uma comunicação confiável e segura entre os agentes.

O KQML (*Knowledge Query and Manipulation Language*) é um protocolo para troca de informações e conhecimento entre agentes. Foi utilizado na prática, até o ano de 2000, quando foi lançada a primeira implementação da FIPA-ACL, ele era a única linguagem padronizada e com implementação disponível e, portanto, dominava a área de comunicação entre agentes.

A FIPA-ACL é uma linguagem de comunicação de agentes que vem ganhando espaço. Ela foi apresentada inicialmente como uma alternativa bem fundamentada para a KQML. Porém, apesar de ter sido proposta em 1997, foi somente a partir do ano 2000, com os lançamentos do padrão FIPA 2000 e da plataforma FIPA, que ela passou a ter uma implementação disponível e a competir com a KQML, e tomando o seu lugar em muitas aplicações. Um exemplo desta afirmação, é o fato de o JADE usar os protocolos de mensagem do FIPA para estabelcer a comunicação entre os agentes. Devido a sua importância, no próximo capítulo será feita uma maior abordagem sobre a plataforma.

Capítulo 7

FIPA

A linguagem FIPA (Foundation for Intelligent Physical Agents) foi apresentada em 1996 em Genebra, Suíça com o propósito de desenvolver e padronizar as tecnologias de sistemas multiagentes promovida pela organização FIPA (FIPA (2010)). A linguagem FIPA é open source e a estrutura de suas mensagens é muito parecida com a estrutura de mensagens da KQML.

Cada mensagem é composta por um ato comunicativo (performative) e um conjunto de parâmetros, dos quais o único obrigatório é a performativa. Porém, a grande maioria das mensagens contém um emissor, um receptor e um campo de conteúdo. Os parâmetros podem ser alocados em qualquer posição dentro da mensagem.

7.1 Parâmetros das mensagens do FIPA-ACL

Os parâmetros nativos da FIPA-ACL são divididos em cinco categorias: tipo do ato comunicativo, participantes da comunicação, conteúdo da mensagem, descrição do conteúdo e controle de conversação.

A lista completa dos parâmetros e suas respectivas categorias são mostradas na TABELA 7.1:

Tipo do ato comunicativo	
Performative	Denota o tipo do ato comunicativo da mensagem.
Participantes da Comunicação	
Sender	É o emissor da mensagem.
Receiver	É o receptor da mensagem.
Reply-to	Indica que as próximas mensagens dessa conversação que deverão ser enviadas para o agente indicado pelo parâmetro reply-to
Conteúdo de mensagem	
Content	É o conteúdo ou conhecimento transportado pela mensagem.
Descrição do conteúdo	
Language	É a linguagem no qual o conhecimento está expresso.
Encoding	Aponta a codificação utilizada na expressão da linguagem de conteúdo.
Ontology	É a ontologia utilizada para dar significado à expressão de conteúdo.
Controle de conversação	
Protocol	É o protocolo de interação utilizado para essa mensagem pelo agente emissor.
Conversation-id	Introduz uma expressão que será usada para identificar a conversação em andamento.
Reply-with	Introduz uma expressão que será usada pelo agente que responderá a essa mensagem para identificá-la.
In-reply-to	É a expressão que referencia a mensagem à qual se está respondendo.
Reply-by	Explicita um tempo máximo durante o qual o agente emissor estará esperando por uma resposta a esta mensagem.

Tabela 7.1: Parâmetros das mensagens FIPA-ACL

O conteúdo de uma mensagem, indicado pelo parâmetro content, referencia a informação sobre a qual o ato comunicativo se aplica. Em geral, o conteúdo pode ser expresso em qualquer linguagem. A linguagem utilizada na representação do conteúdo pode ser declarada no parâmetro language. O projeto FIPA define e sugere algumas linguagens de representação de conteúdo de padrões, tal como a linguagem SL.

7.2 Especificações FIPA

As especificações produzidas pela FIPA são publicadas em documentos que seguem uma identificação especial quanto ao seu tipo e lançamento. Nesta secção serão apresentados esses documentos. As especificações são divididas em cinco áreas distintas: *Aplicações de Sistemas Multiagentes,Arquiteturas Abstratas*, *Comunicação entre Agentes*, *Gerenciamento de Sistemas Multiagentes* e *Transporte de Mensagens entre Agentes*. A seguir será feita uma análise de cada uma dessas categorias.

7.2.1 Aplicações de Sistemas Multiagentes

As Aplicações de Sistemas Multiagentes são exemplos de domínios onde podem ser utilizados agentes FIPA. Possui definições de ontologias e descrições de serviços para esses domínios.

O padrão FIPA-97 já incluía especificações de aplicações para PTA (Personal Travel Assistence) entretenimento audiovisual, gerenciamento de redes e assistência pessoal. No padrão FIPA-2000 essas especificações foram mantidas, tendo sido incluídas apenas duas novas: suporte a aplicações nômades (agentes móveis ou migratórios) e bufferização de mensagens. Logo após, foi adicionada a especificação de Qualidade de Serviço. A tabela 7.2 mostra esses documentos.

Identificador	Título
SI00014	FIPA Nomadic Application Support Specification
XC00079	FIPA Agent Software Integration Specification
XI00080	FIPA Personal Travel Assistance Specification
XI00081	FIPA Audio-Visual Entertainment and Broadcasting Specification
XI00082	FIPA Network Management and Provisioning Specification
XI00083	FIPA Personal Assistant Specification
XC00092	FIPA Message Buffering Service Specification
SC00094	FIPA Quality of Service Specification

Tabela 7.2: Especificações FIPA: Aplicações de Sistemas Multiagentes

7.2.2 Arquitetura Abstrata para Sistemas Multiagentes

A Arquitetura abstrata para Sistemas Multiagentes trabalha com entidades abstratas necessárias para a construção de serviços e ambientes de agentes. Assim, ela define quais são as características arquiteturais que um sistema multiagente deve estar conforme especificado os requisitos para arquiteturas concretas (implementadas em JAVA, PROLOG, etc.) de sistemas multiagentes.

É incluída, também, uma especificação definindo como sistemas multiagentes poderiam ser estruturados em domínios distintos e como poderiam ser estabelecidas políticas de manutenção.

Identificador	Título
SC00001	FIPA Abstract Architecture Specification
PC00089	FIPA Domains and Policies Specification

Tabela 7.3: Especificações FIPA: Arquitetura Abstrata

7.2.3 Comunicação entre Agentes

O maior avanço e detalhamento em termos de padronização ocorreram no tratamento das questões relativas à comunicação entre agentes. Entre a especificação da linguagem FIPA-ACL feita por um único documento (considerado atualmente obsoleto, a especificação (FIPA00003, 1997)[FIPA 2003]) em 1997 e sua instanciação atual, coberta por quase trinta documentos distintos, percebe-se claramente onde esteve concentrado a maior parte do trabalho da FIPA nesse intervalo de tempo. Mesmo a arquitetura abstrata também poderia ser vista, em grande parte, como um trabalho necessário para uma melhor fundamentação (pragmática) dos processos de comunicação.

O formato básico que deve ser seguido por todas as mensagens FIPA-ACL foi descrito em (FIPA00061, 2001) FIPA (2010).

Esta categoria divide-se em:

- **Atos comunicativos:** define os atos comunicativos padrão da linguagem FIPA-ACL;
- **Protocolos de Interação:** Define uma seqüência lógica de troca de mensagens, especificando atos comunicativos para cada uma delas. Os protocolos de interação são úteis quando alguma conversação necessita um maior poder de interação entre os agentes.
- **Linguagem de Conteúdo:** define as diferentes maneiras de representar ou codificar a informação passada através de uma mensagem ACL;

7.2.4 Gerenciamento de Agentes

A categoria de Gerenciamento de Agentes trabalha com a especificação de ferramentas para controle e gerenciamento de agentes em plataformas de agentes.

A plataforma de Gerenciamento de Agentes descrita no documento mostrado pela TABELA abaixo que define o ambiente onde os agentes FIPA existem e operam. Ele estabelece o modelo lógico de referência para a criação, registro, localização, comunicação, migração e desativação dos agentes.

Identificador	Título
SC00023	FIPA Agent Management Specification

Tabela 7.4: Especificações FIPA: Gerenciamento de agentes

7.2.5 Transporte de Mensagens entre Agentes

O Transporte de Mensagens entre Agentes se preocupa em especificar a forma como as mensagens são transportadas e representadas através de diferentes protocolos de rede.

Esta categoria está divida em:

- **Representações da ACL:** define diferentes formas de representar uma mensagem ACL;
- **Representações de Envelope:** define diferentes formas de representar um envelope;
- **Protocolos de Transporte:** define formas de transporte de mensagens ACL para diferentes protocolos de redes.

Transporte de Mensagens entre Agentes	
SC00067	FIPA Agent Message Transport Service Specification
XC00093	FIPA Messaging Interoperability Service Specification
Representações da ACL	
SC00069	FIPA ACL Message Representation in Bit-Efficient Specification
SC00070	FIPA ACL Message Representation in String Specification
SC00071	FIPA ACL Message Representation in XML Specification
Representações de Envelope	
SC00085	FIPA Agent Message Transport Envelope Representation in XML Specification
SC00088	FIPA Agent Message Transport Envelope Representation in Bit Efficient Specification
Protocolos de Transporte	
SC00075	FIPA Agent Message Transport Protocol for IIOP Specification
XC00076	FIPA Agent Message Transport Protocol for WAP Specification
SC00075	FIPA Agent Message Transport Protocol for HTTP Specification

Tabela 7.5: Especificações FIPA: Transporte de Mensagens entre Agentes

O FIPA-OS[1] é um ambiente que está tendo um crescimento muito grande, com a sua utilização em laboratórios de pesquisa e instituições de ensino no mundo todo. Isso pode ser observado na agilidade, na divulgação e correção de erros, e na disponibilização de material para testes e compreensão do ambiente.

[1]FIPA Open Source

Capítulo 8

JADE

Jade [1] (*Java Agent DEvelopment framework*) é um ambiente para desenvolvimento de aplicações baseado em agentes conforme as especificações da FIPA (*Foundation for Intelligent Physical Agents*) para interoperabilidade entre sistemas multiagentes totalmente implementado em Java. Foi desenvolvido e suportado pelo CSELT da Universidade de Parma na Itália. É open source (LGPL7 [2]) Segundo Telecom-Itália (2010), o principal objetivo do Jade é simplificar e facilitar o desenvolvimento de sistemas multiagentes garantindo um padrão de interoperabilidade entre eles através de um abrangente conjunto de agentes de serviços de sistema, os quais tanto facilitam como possibilitam a comunicação entre agentes, de acordo com as especificações da FIPA: serviço de nomes (naming service) e páginas amarelas (yellow-page service), transporte de mensagens, serviços de codificação e decodificação de mensagens e uma biblioteca de protocolos de interação (padrão FIPA) pronta para ser usada. Toda sua comunicação entre agentes é feita via troca de mensagens. Além disso, lida com todos os aspectos que não fazem parte do agente em si e que são independentes das aplicações tais como transporte de mensagens, codificação e interpretação de mensagens e ciclo de vida dos agentes. Ele pode ser considerado como um "middle-ware" de agentes que implementa um framework de desenvolvimento e uma plataforma de agentes. Em outras palavras, uma plataforma de agentes em complacência com a FIPA para desenvolvimento de agentes em Java.

[1] Jade é uma marca registrada do TILAB (`http://www.telecomitalialab.com`). Foi desenvolvido pelo TILAB juntamente com o AOT (`http://aot.ce.unipr.it`) com a permissão do TILAB.) anteriormente conhecido com CSELT.

[2] LGPL ou Lesser General Public License.

8.1 Características do JADE

Seguem abaixo algumas características do framework JADE:

- *Plataforma distribuída de agentes:* JADE pode ser dividida em vários "hosts" ou máquinas (desde que eles possam ser conectados via RMI). Apenas uma aplicação Java e uma Java Virtual Machine é executada em cada host. Os agentes são implementados como threads Java e inseridos dentro de repositórios de agentes chamados de containeres (Agent Containers) que provêm todo o suporte para a execução do agente.
- *Gui ou Graphical User Interface:* Interface visual que gerencia vários agentes e containeres de agentes inclusive remotamente.
- *Ferramentas de Debugging:* ferramentas que ajudam o desenvolvimento e depuração de aplicações multiagentes baseados em JADE. Suporte a execução de múltiplas, paralelas e concorrentes atividades de agentes - através dos modelos de comportamentos (Behaviours).
- *Ambiente de agentes complacente a FIPA:* No qual incluem o sistema gerenciador de agentes (AMS - Agent Management System), o diretório facilitador (DF - Directory Facilitator) e o canal de comunicação dos agentes (ACC - Agent Communication Channel). Todos esses três componentes são automaticamente carregados quando o ambiente é iniciado.
- *Transporte de mensagens:* Transporte de mensagens no formato FIPA-ACL dentro da mesma plataforma de agentes.
- *Biblioteca de protocolos FIPA:* Para interação entre agentes JADE dispõe de uma biblioteca de protocolos prontos para serem usados.
- *Automação de registros:* Registro e cancelamento automático de agentes com o AMS fazendo com que o desenvolvedor se abstraia desta implementação.
- *Serviços de nomes (Naming Service) em conformidade com os padrões FIPA:* Na inicialização dos agentes, estes obtêm seus GUID (Globally Unique Identifier) da plataforma que são identificadores únicos em todo o ambiente.
- *Integração:* Mecanismo que permite aplicações externas carregarem agentes autônomos JADE.

Além das características acima citadas que por si só já facilitam muito o desenvolvimento de sistemas multiagentes, Jade possui também algumas ferramentas muito úteis que simplificam a administração da plataforma de agentes e o desenvolvimento de aplicações.

8.2 Regras de comunicação FIPA no JADE

O modelo de plataforma padrão especificado pela FIPA, que pode ser visto na figura ilustrada.

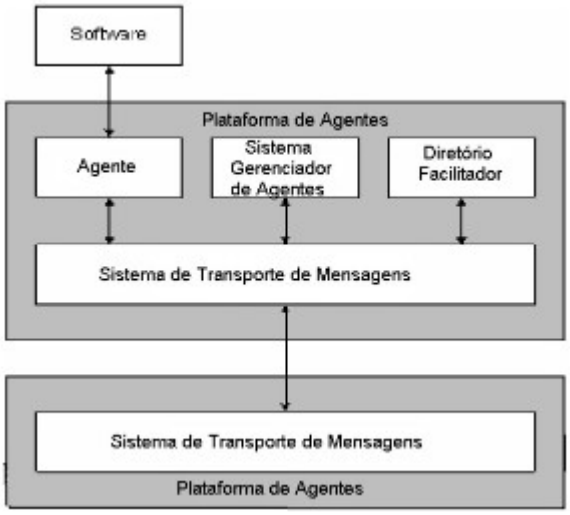

Figura 8.1: Modelo da plataforma de agentes definido pela FIPA

Este modelo pode ser composto por algumas estruturas definidas abaixo:

O *agente*, parte superior esquerda da FIGURA 8.1, é o agente propriamente dito cujas tarefas serão definidas de acordo com o objetivo da aplicação. Encontra-se dentro da plataforma de agentes (Agent Platform) e realiza toda sua comunicação com agentes através de troca de mensagens e relacionando-se com aplicação externa (software).

O sistema gerenciador de agentes ou Agent Management System (AMS), parte superior central da mesma Figura, é o agente que supervisiona o acesso e o uso da plataforma de agentes. Apenas um AMS irá existir em uma plataforma. Ele provê guia de endereços (whitepages) e controle de ciclo-de-vida, mantendo um diretório de identificadores de agentes (Agent Identifier - AID [3]) e estados de agentes. Ele é o responsável pela autenticação de agentes e pelo controle de registro. Cada agente tem que se registrar no AMS para obter um AID válido.

O diretório facilitador (Directory Facilitator - DF), localizado na parte superior direita da FIGURA 8.1, é o agente que provê o serviço de páginas amarelas (yellow-pages) dentro da plataforma.

Na parte inferior da Figura temos o sistema de transporte de mensagens ou *Message Transport System*, também conhecido como canal de comunicação dos agentes(Agent Communication Channel - ACC). Ele é o agente responsável por prover toda a comunicação entre agentes dentro e fora da plataforma. Todos os agentes, inclusive o AMS e o DF, utilizam esse canal para a comunicação.

[3] AID ou Agent Identifier é uma classe de JADE que atribui identificadores únicos aos agentes.

JADE cumpre totalmente com essa arquitetura especificada. Sendo que, no carregamento da plataforma JADE, o AMS e o DF são criados e o ACC é configurado para permitir a comunicação através de mensagens.

8.3 Agentes em JADE

Para o Jade, um agente é autônomo e independente do processo que tem uma identidade e requer comunicação com outros agentes, seja ela por colaboração ou por competição, para executar totalmente seus objetivos (Telecom-Itália (2010)).

Em outras palavras, pode-se concluir que Jade é absolutamente neutro no que diz respeito à definição de um agente, ou seja, ele não limita ou especifica que tipo de agente pode ser construído pela plataforma.

Em termos mais técnicos um agente em Jade tem o funcionamento de uma ”thread” que emprega múltiplas tarefas ou comportamentos e conversações simultâneas. Esse *agente* Jade é implementado como uma classe Java chamada Agent.

Essa classe Agent atua como uma super classe para a criação de agentes de software definidos por usuários. Ela provê métodos para executar tarefas básicas de agentes, tais como:

Passagens de mensagens usando objetos ACLMessage (seja direta ou multicast); Suporte completo ao ciclo de vida dos agentes, incluindo iniciar ou carregar, suspender e ”matar” (killing) um agente; Escalonamento e execução de múltiplas atividades concorrentes; Interação simplificada com sistemas de agentes FIPA para a automação de tarefas comuns de agentes (registro no DF, etc).

O JADE também provê suporte ao desenvolvimento de agentes móveis. Devido a sua própria arquitetura, os agentes podem migrar e clonar-se entre os containeres. Além disso, disponibiliza uma biblioteca (jade.domain.mobility) que contém a definição de ontologias para a mobilidade em JADE, vocabulário com uma lista de símbolos usados e todas as classes Java que implementam essas ontologias.

Do ponto de vista do programador, um agente JADE é simplesmente uma instância da classe Agent, no qual os programadores ou desenvolvedores deverão escrever seus próprios agentes como subclasses de Agent, adicionando comportamentos específicos de acordo com a necessidade e objetivo da aplicação, através de um conjunto básico de métodos, e utilizando as capacidades herdadas que a classe Agent dispõe, tais como mecanismos básicos de interação com a plataforma de agentes (registro, configuração, gerenciamento remoto, etc).

Na FIGURA 8.2 temos a descrição de uma arquitetura interna de um agente em JADE.

Na parte superior temos os comportamentos ativos do agente que representariam as ações/intenções que cada agente tem. O modelo computacional de um agente em JADE é multitarefa, onde tarefas (ou comportamentos) são executadas concorrentemente. Cada funcionalidade ou serviço provido por um agente deve ser implementado como um ou mais

comportamentos. Note que esses comportamentos podem ser diversos uma vez que JADE permite uma variedade de comportamentos em um mesmo agente.

Na parte inferior esquerda da Figura, temos uma fila de mensagens ACL. Todo agente JADE tem essa fila onde decide quando ler as mensagens recebidas e quais mensagens ler.

Ao centro da figura, temos o escalonador de comportamentos e o gerenciador do ciclo de vida. O primeiro é responsável por escalonar a ordem de execução dos comportamentos. O gerenciador de ciclo de vida é o controlador do estado atual do agente.

Uma característica importante do modelo de agentes JADE é a autonomia. Cada agente possui uma autonomia que tem a possibilidade de controlar completamente sua thread de execução. O gerenciador de ciclo de vida é o meio que os agentes utilizam para determinar seu estado atual (ativo, suspenso, etc). No lado direito da figura, temos os recursos de agentes dependentes da aplicação, e é nesse local aonde serão armazenadas as crenças e capacidades que o agente adquiriu na execução da aplicação.

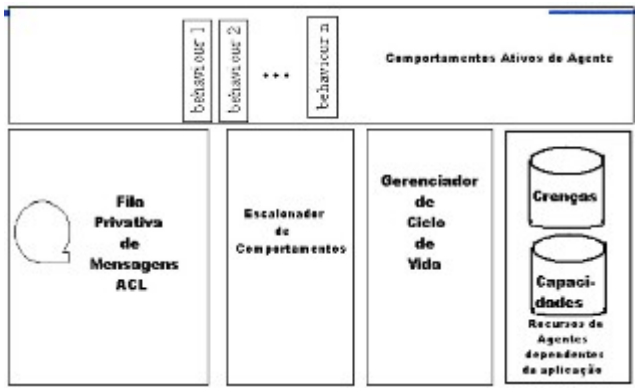

Figura 8.2: Arquitetura interna de uma agente genérico em JADE

8.3.1 Ciclo de vida do Agente

Um agente JADE pode estar em um dos vários estados de acordo com ciclo de vida (Agent Platform Life Cycle) das especificações FIPA. Na FIGURA 8.3, temos a representação do ciclo de vida de um agente definido pela FIPA e seus estados possíveis (FIPA (2010)).

Vale ressaltar que é permitido ao agente executar seus comportamentos (behaviours) apenas quando estiver no estado ativo. Outro fato que também merece ser ressaltado é que se em qualquer dos comportamentos de um agente for chamado o método Em_ espera(), o agente como um todo e todas suas atividades serão bloqueadas, não apenas o comportamento que chamou o método. Também é importante referenciar que o JADE possuí métodos que permitem somente a suspensão de um comportamento, evitando o bloqueio total.

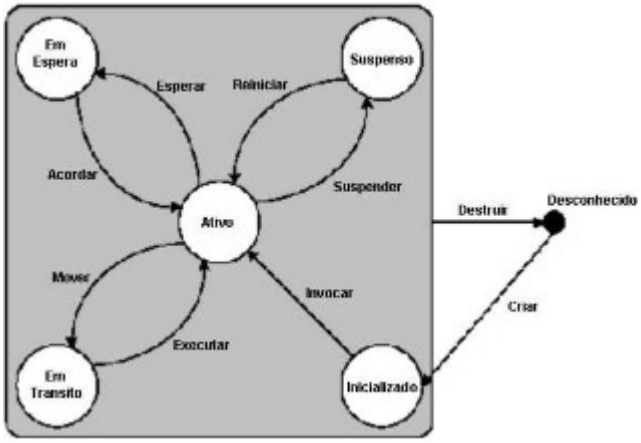

Figura 8.3: Ciclo de Vida de um agente definido pela FIPA

8.4 Troca de Mensagens

Toda a troca de mensagens realizada no JADE é feita através de métodos próprios e com o uso de instâncias da classe ACLMessage. Esta classe possui um conjunto de atributos que estão em conformidade com as especificações da FIPA, implementando a linguagem FIPA-ACL. Assim, um agente que pretenda enviar uma mensagem deve instanciar um objeto da classe ACLMessage, preenchê-los com as informações necessárias e chamar o método *send()* da classe Agent. Caso for receber mensagens, o método *receive()* ou *blockingReceive()* também da classe Agent deve ser chamado. Outra meio de enviar ou receber mensagens no JADE é através do uso das classes de comportamentos *SenderBehaviour()* e *ReceiveBehaviour()*. Fato que torna possível que as trocas de mensagens possam ser escalonadas como atividades independentes de um agente.

8.5 Interoperabilidade

Conforme já foi mencionado antes, a arquitetura de JADE é baseada em Java Virtual Machine em diferentes hosts. Para manter as comunicações entre diferentes ambientes, JADE utiliza-se do protocolo IIOP[4] (Internet Inter-ORB Protocoll). Para comunicação de agentes na mesma plataforma JADE já utiliza outros meios, como RMI ou via eventos, podemos perceber que JADE possui um comportamento variado e distribuído em relação a forma com que realiza sua comunicação, aonde o mecanismo é selecionado de acordo com a situação do ambiente. Isso ocorre com o objetivo único de atingir o menor custo possível de passagens de mensagens. Com isso também podemos perceber a estrita relação entre o JADE que é baseado em agentes inteligentes e a Inteligência artificial Distribuída.

[4]Introduzido na segunda especificação de CORBA (Common Object Request Broker Architecture), o IIOP permitiu que CORBA se tornasse uma solução definitiva para interoperabilidade entre objetos que não estão presos a uma plataforma ou padrão específico.

A seguir é mostrada uma figura que ilustra a interoperabilidade entre os agentes, em que o meio para se estabelecer a comunicação mais eficiente é escolhido de acordo com a localização do agente receptor da mensagem conforme foi explicado anteriormente. Na parte inferior da FIGURA 8.4 existem as formas possíveis de comunicação. O cache local, localizado no ACC (parte central da Figura), armazena as referências dos objetos dos outros containeres. Essas referências são adicionadas ao cache sempre que uma mensagem é enviada.

Figura 8.4: Interoperabilidade entre os agentes

Agora que já foram adquiridos os conhecimentos sobre ontologia, agentes inteligentes/sistemas multiagentes, FIPA e JADE e as relações existentes entre eles, já podemos partir para a implementação da ferramenta proposta neste trabalho que se baseia em ontologias focadas em teorias sociais e agentes inteligentes.

Capítulo 9

Implementação

Conforme citado no capítulo anterior, uma vez adquiridos os conhecimentos necessários sobre os recursos a serem utilizados e suas relações, já é possível detalhar a implementação da ferramenta proposta.

Esta ferramenta tem por objetivo criar ontologias de aluno e grupo baseadas em teorias sociais. Essas ontologias são representadas por agentes, que detêm de uma certa inteligência (no caso, artificial) provendo a interação entre aluno/aluno e aluno/grupo. E para alcançarmos este objetivo, a ferramenta será implementada com um framework poderoso para este tipo de implementação, o Jade, cujas funcionalidades e características foram apresentadas no capítulo anterior.

O Jade tem todo o suporte para a criação de agentes baseados em ontologias e também possuí os protocolos de comunicação do FIPA que permitem a interoperabilidade entre os agentes. Sendo assim, as classes *Aluno* e *Grupo* são os agentes criados com suas respectivas ontologias. Depois serão criadas as classes que irão estabelecer os comportamentos entre os agentes, a comunicação e a interoperabilidade entre os agentes.

Nas próximas seções, serão detalhadas todas as funcionalidades de cada classe que compõem o projeto em si, bemo como os diagramas UML do projeto. Os códigos das classes do pacote ontoCursistas poderão ser vistos em Apêndice A e os códigos das classes setup AssociaAgent e RequesterAgent poderão ser visto em Apêndice B.

9.1 Lista de Requisitos da ferramenta

Neste seção serão apresentados os requisitos funcionais e não funcionais da ferramenta.

9.1.1 Requisitos Funcionais

RF001 **Cadastro de Alunos**

A ferramenta deve permitir aos usuários, o cadastro de alunos contendo os campos nome e identificador.

RF002 **Cadastro do Grupo**

Um grupo deve ser previamente criado na plataforma contendo os atributos nome, identificador, tipo do grupo e lista de alunos que pertencem ao grupo.

RF003 **Material**

A ferramenta deve permitir que o grupo sirva de repositório de material dos alunos, bem como a emissão do mesmo.

RF004 **Dúvidas dos alunos**

A ferramenta, permitirá ao grupo criar fóruns de discussão dos tópicos relacionados a matéria que está sendo estudada pelos alunos.

RF005 **Comunicação do Grupo**

O grupo deverá emitir mensagens aos alunos integrantes quando um novo aluno é associado ao grupo e quando há um novo post nos fóruns.

RF006 **Comunicação entre Alunos**

A ferramenta permitirá que os alunos possam estabelecer uma comunicação direta entre eles, sem intermédio do grupo.

RF007 **Status do Aluno**

A ferramenta deve permitir que o grupo guarde o "Status"de participação dos integrantes (como cada participante está acessando o material, enviando mensagens, etc).

RF008 **Estatísitcas**

A ferramente deve permitir que o grupo emita relatórios de participação dos alunos.

9.1.2 Requisitos Não Funcionais

RNF001 **Usabilidade**

A navegação deve ser simplificada de modo a tornar o sistema produtivo e fácil de usar.

RF002 **Acessibilidade**

O sistema deve permitir acesso de qualquer local que disponha de um computador com acesso a internet .

9.2 Ontologia para aluno e grupo de alunos

Após a análise dos requisitos da ferramenta devemos descrever a implementação mais detalhadamente, começando por apresentar de forma esquemática as ontologias criadas para aluno e grupo. É estelecido o vocabulário e as funcionalidades para modelo de aluno e grupo, que posteriormente serão modelados como agentes (agentes inteligentes) através do JADE [1] que dará suporte a comunicação[2] entre esses agentes (comunicação aluno-aluno e comunicação aluno-grupo ou vice-versa).

- **Ontologia de aluno:** define uma base de conhecimento para o aluno e suas funcionalidades.

  1. **Propriedades do aluno:** Aluno deve ter um id e um nome
  2. **Ações do aluno:**
     - Associar-se a um grupo (o grupo, para teste, será pré-existente);
     - emitir material no grupo;
     - responder ao grupo (como por exemplo, fórum);
     - aluno pode responder diretamente a outro aluno;
     - obter material.

[1] JADE é um framework do Java para o desenvolvimento de aplicações de agentes inteligentes
[2] A comunicação é estabelecidade através dos protocolos de comunicação da FIPA

- **Ontologia do grupo:** define uma base de conhecimento acerca do grupo, bem como suas funcionalidades.

    1. **Propriedades do grupo:**
        - Deve ter um id, nome e tipo de grupo, que define previamente a que tipo de matéria se destina o grupo. Por exemplo, podem ser criados um grupo de matemática, um de física dentro do sistema multiagentes etc.
        - O grupo deve possuír a lista de alunos inseridos
    2. **Ações do grupo:**
        - Subentende-se que o grupo é previamente criado pelo sistema;
        - Enviar mensagem ao aluno que se associou a ele;
        - Enviar mensagem de aviso quando há uma participação verbal(tipo fórum) no grupo;
        - O grupo servirá de repositório de material;
        - Emissão de material;
        - Servirá para discussão de dúvidas entre os integrantes;
        - O grupo deverá guardar o "Status"de participação dos integrantes (como cada participante está acessando o material, enviando mensagens, etc). É importante guardar o status de participação dos integrantes, porque o grupo pode identificar alunos que compartilham das mesmas dificuldades e propor novas formas de aprendizado que se adequem a um determinado grau de dificuldade para certos alunos;
        - Estabelecer relatórios do tipo: quais os materiais (e freqüência de acesso) acessados por cada integrante, com quem o integrante se relaciona mais.

A seguir é apresentado o diagrama que representa a ontologia descrita acima:

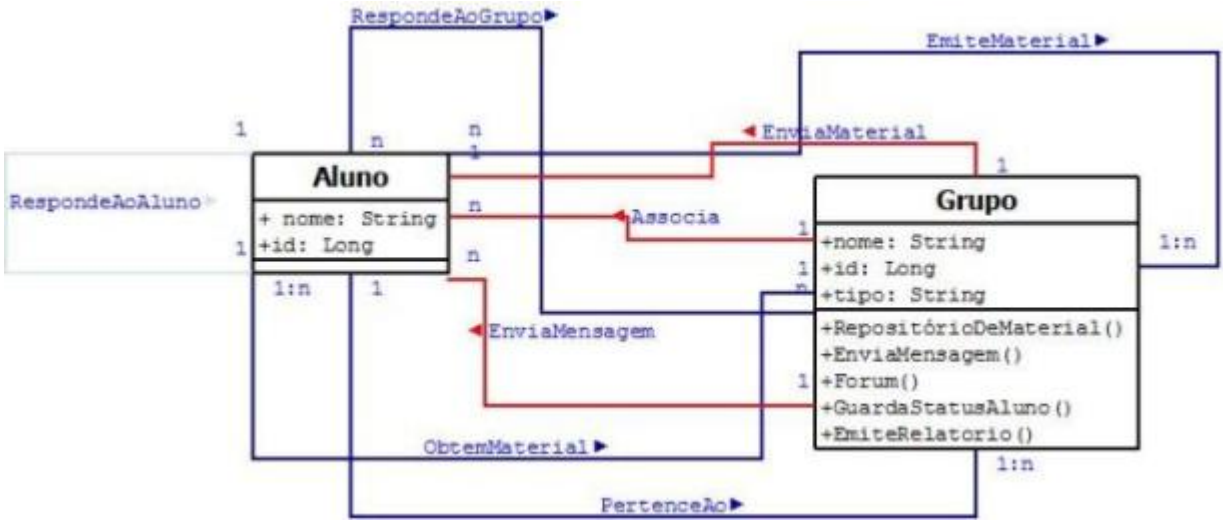

Figura 9.1: Diagrama descritivo da ontologia Aluno-Grupo

Como se pode perceber o desenvolvimento de uma ontologia inicia a execução de um projeto e a partir da sua definição é possível buscar a construção de um software para um ambiente virtual de ensino e aprendizagem, na tentativa de orientar o professor a estabelecer estratégias que visem melhorar a utilização do ambiente para a promoção da aprendizagem e também ajudam o aluno no seu auto-aprendizado.

9.3 Modelo Aluno

Este modelo armazena a base de conhecimentos do Aluno necessária para a modelagem da ontologia. O aluno é constituído por um nome e um identificador para representá-lo dentro do sistema multiagente, sendo que juntos (nome e identificador) deverão ser únicos para cada aluno.

9.4 Modelo Grupo

O modelo Grupo armezana os dados inerentes a cada grupo criado. É constituído por um nome, tipo de grupo e um identificador. É importante fazer a modelagem da base de conhecimentos para o grupo porque esta será necessária para modelagem da ontologia proposta.

9.5 Classe AlunoCursista

Implementa os predicados do aluno necessários para formar a ontologia.

9.6 Classe Associa

Esta classe é implementa para criar a funcionalidade de ”associar” um aluno no grupo e para tal devem ser criados os objetos aluno e grupo.

9.7 Classe AssociaError

Esta classe implementa os predicados para a funcionalidade Associa.

9.8 Classe PertenceAoGrupo

Implementa os predicados e testa se um aluno já está inserido no grupo, para não permitir futuramente que este aluno tente se associar novamente ao grupo.

9.9 Classe Modelo_ Ontologia

Esta classe modela a ontologia proposta. Ela cria o vocabulário e os conceitos da ontologia de Aluno-Grupo, também cria as ações pertencentes a ela, tais como, o grupo associa alunos, os alunos se associam a um determinado grupo.

Para que a ontologia fique bem modelada, também há necessidade de se definir os predicados, nomeadamente: PertenceAoGrupo, AssociaError inerentes a aluno e grupo.

A classe garante que uma única ontolgia é acessada, retornando o objeto ontologia e seus respectivos conceitos de grupo e aluno.

9.10 AssociaAgente

Como próprio nome da classe diz, cria o agente que estabelece a comunicação (ACL) dentro do sistema de acordo com os protocolos FIPA, portanto ela funciona como a classe principal, responsável por toda a comunicação entre os agentes dentro do sistema, manipula de facto, a associação do aluno no grupo após uma ação de Requester e faz a associação do aluno no grupo. Esta classe possui dois comportamentos, a saber:

SimpleAchieveREResponder para lidar com perguntas sobre os alunos que pertencem um determinado grupo através do protocolo FIPA-Consulta. Lida com as requisições de associação através do protocolo FIPA-Request.

É responsável por criar os comportamentos (BEHAVIOURS) do agente, sendo que este comportamento lida com a associação dos alunos ao grupo, seguindo o protocolo FIPA-Query.

A classe obtém os predicados corretos e as ações definidas para a ontologia mediante uma consulta e também cria os métodos dos agentes.

9.11 RequesterAgent

Esta classe obtém as informações detalhadas dos aluno que querem se associar ao grupo e solicita ao *AssociaAgent* para associar alunos ao grupo.

As informações dos alunos são obtidas pelo método onStart(), depois uma consulta é feita em *AssociaAgent* para ver se o aluno já está ou não associado ao grupo. A consulta é feita através dos protocolos FIPA, de acordo com o resultado da consulta uma solicitação é enviada para o /AssociaAgent].

A seguir são apresentados os diagramas UML que representam respectivamente a dependência entre as clasees do pacote OntoCursistas e a dependência entre as classes AssociaAgent e RequesterAgent com o pacote OntoCursista. Estes diagramas representam a modelagem da ferramenta proposta para este trabalho.

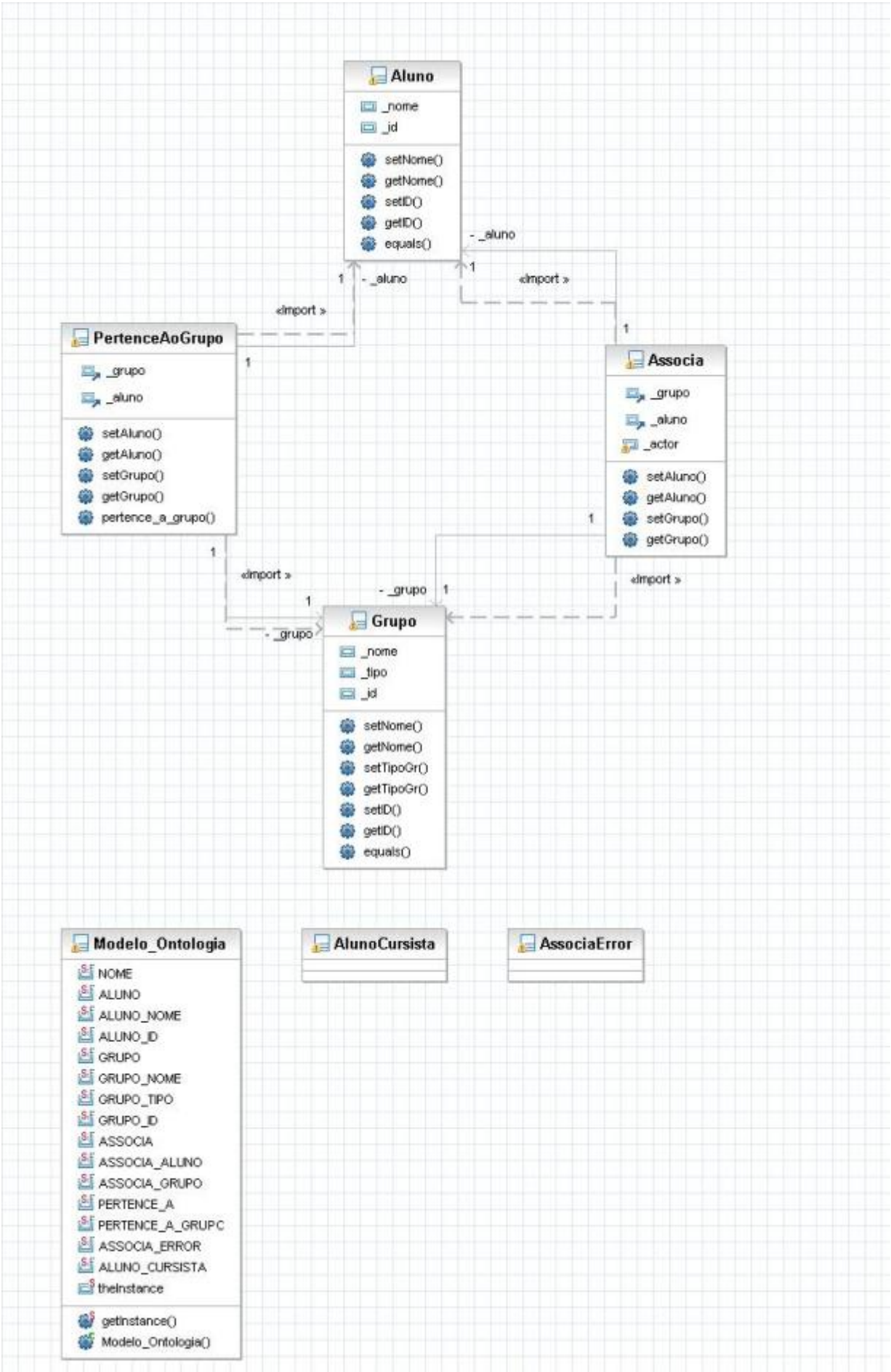

Figura 9.2: UML do pacote OntoCursistas

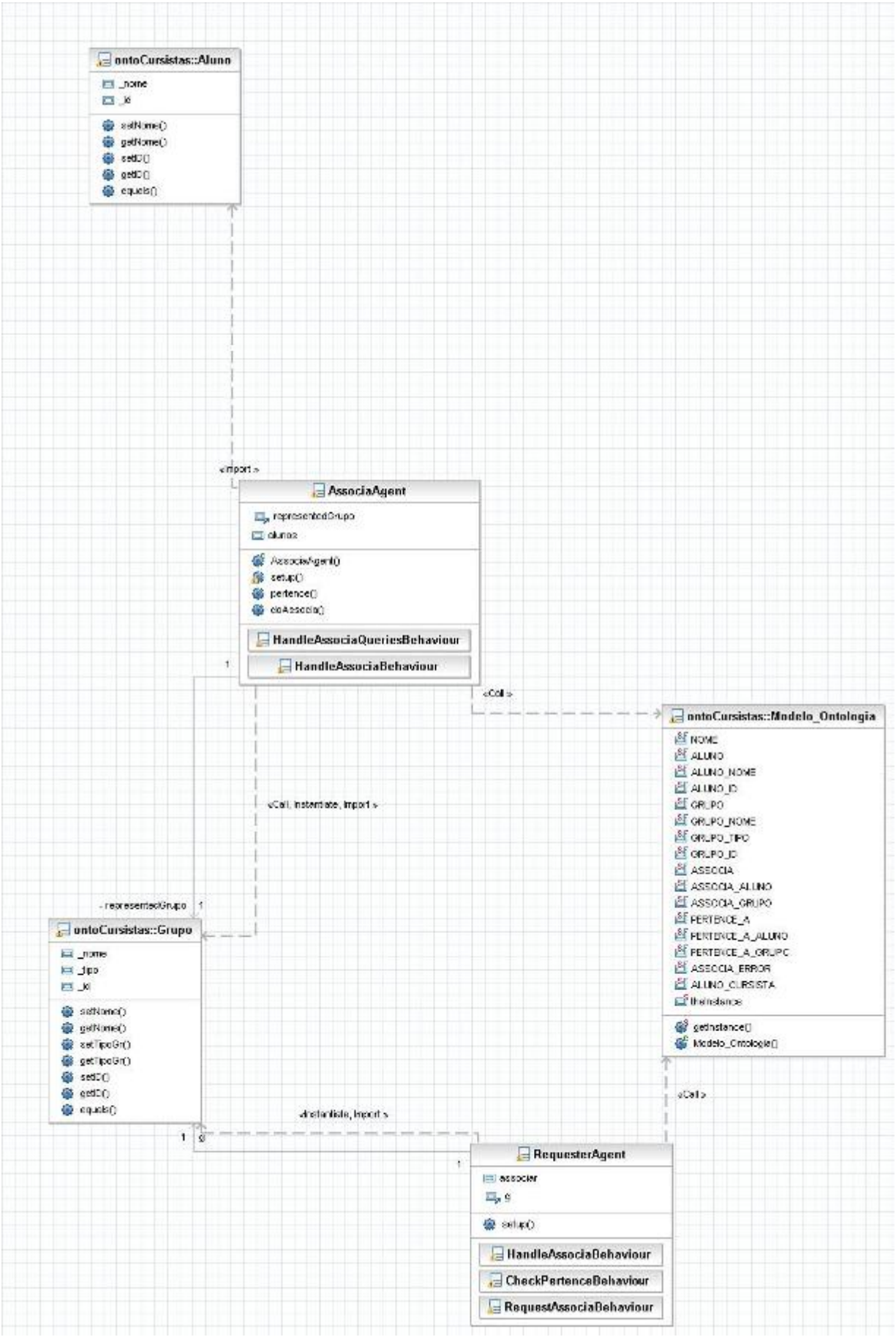

Figura 9.3: Dependência entre as classes AssociaAgent e RequesterAgent e as classes do pacote OntoCursistas

9.12 Comunicação

Um dos principais aspectos do desenvolvimento de sistemas multiagentes diz respeito à forma sob a qual os agentes irão se comunicar. Sem mecanismos de comunicação é praticamente impossível a definição de agentes.

Todo o processo de comunicação entre os agentes se dá por troca de mensagens FIPA-ACL através do ambiente FIPA usado pelo JADE.

A FIGURA 9.4 mostra um esquema de como é feita a comunicação entre os agentes aluno e grupo na plataforma multiagente. Nesta Figura podemos constatar a existência de 3 alunos que pertencem ao grupo matemática. O Aluno1 faz a solicitação de material ao grupo, por sua vez o grupo confirma a exeistência desse material e emite-o ao respectivo aluno.

Podemos constatar outra comunicação entre aluno e grupo que diz respeito a associação de um novo aluno (Aluno3) no grupo e uma outra interação em que uma mensagem é enviada ao Aluno2 contendo um conjunto de dados ou informações pertencentes ao aluno.

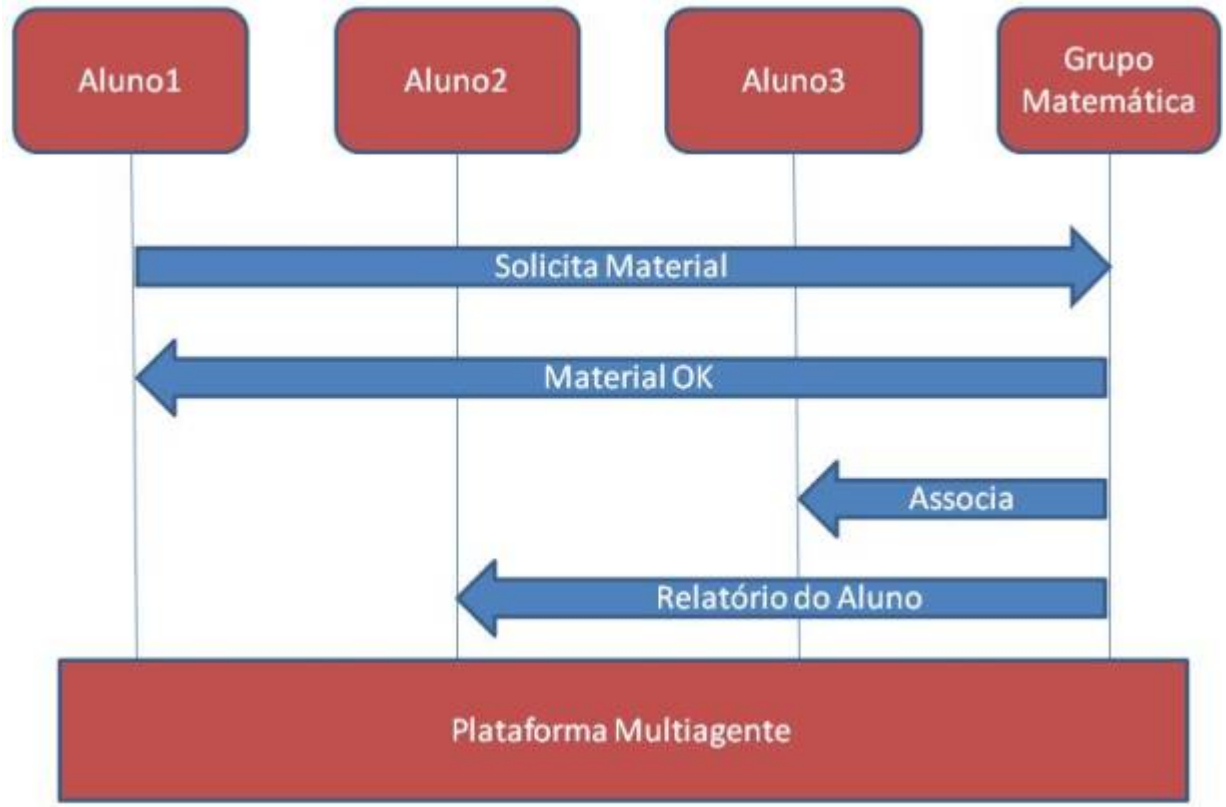

Figura 9.4: Comunicação entre os agentes

9.13 Execução da Aplicação

A primeira Figura mostra o carregamento (boot) de AssociaAgent, ou seja, é criado um agente de Grupo, cujo nome é "Matemática".

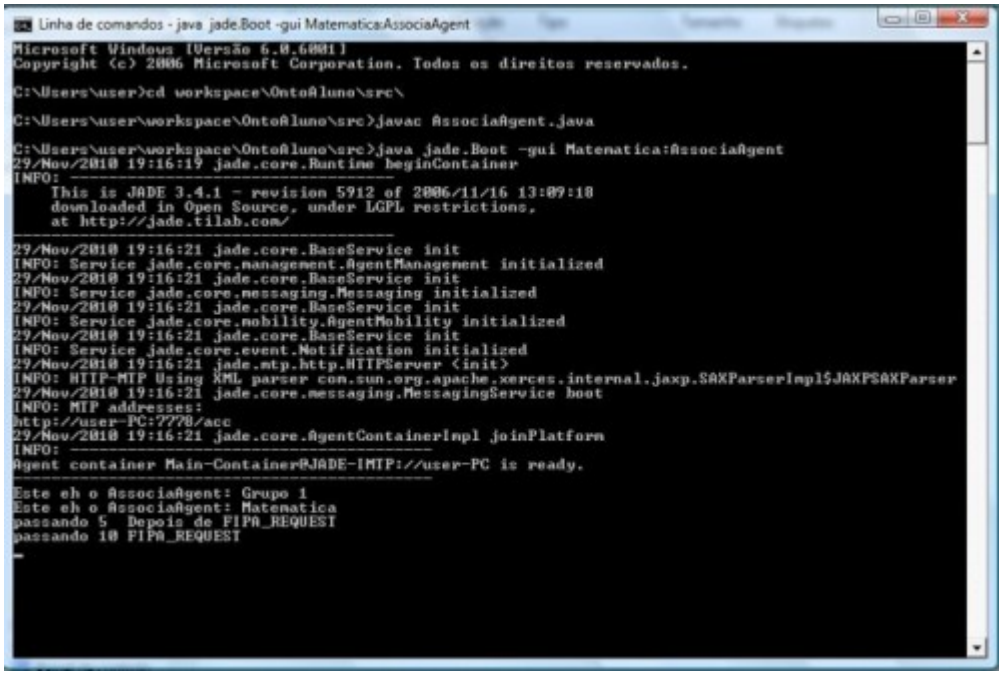

Figura 9.5: Carregando AssociaAgent - Agente Grupo(Matematica)

O agente também é criado na gui de gerenciamento de agentes (RMA) do JADE, o RMA[3] (FIGURA 9.6), assim que o AssociaAgent é carregado no prompt.

O **RMA** é um console gráfico para o controle e gerenciamento remoto da plataforma JADE. Permite o controle dos estados do ciclo de vida de todos agentes em execução inclusive os distribuídos. Além disso, serve como um controle principal onde estão centralizadas algumas funcionalidades, como chamar as outras ferramentas do JADE e temos como exemplo o serviço de yellow pages, clone de agentes, migração de agentes para outras plataformas, dentre outras.

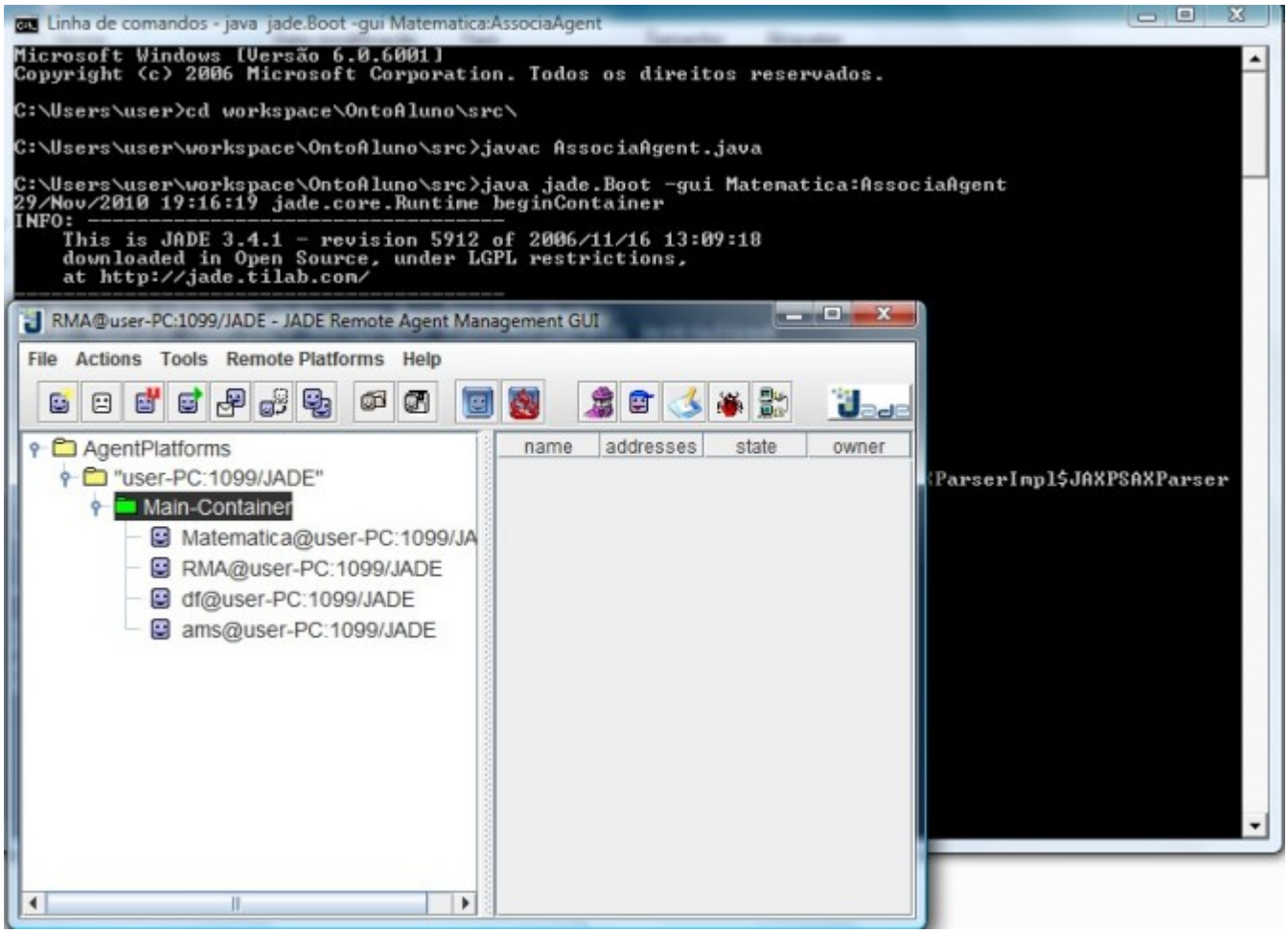

Figura 9.6: Carregando AssociaAgent no RMA - Grupo Matematica

[3]O JADE permite que haja mais de um RMA atuando na mesma plataforma

Na FIGURA 9.7 podemos ver que é carregado o RequesterAgent, ou seja agente de aluno (Ylana) para estabelecer a interação e interoperabilidade dentro do grupo. Podemos perceber que a tela do prompt aonde foi carregado o agente grupo (AssociaAgent) faz o controle de toda a comunicação entre os demais agentes e o controle do ciclo de vida de todos os RequerterAgents , ele também deve se certificar de que o agente que está sendo criado ainda não pertence ao grupo.

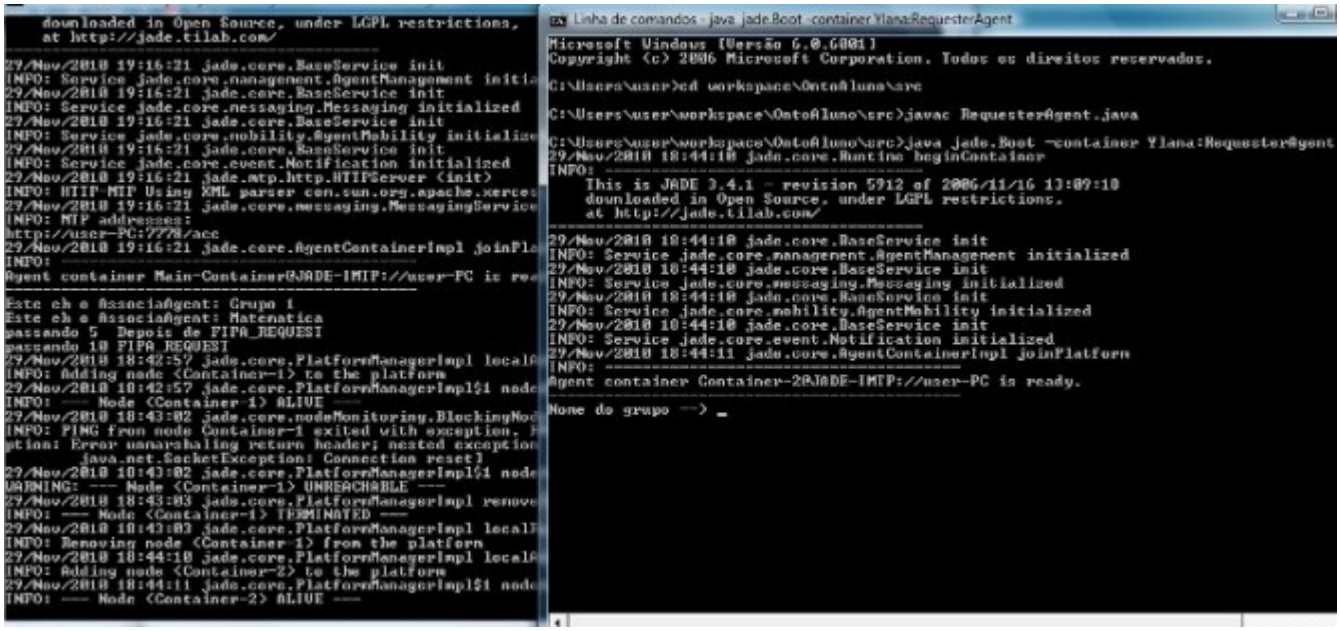

Figura 9.7: Carregando o 1º RequesterAgent - Agente Aluno(Ylana)

A FIGURA 9.8 mostram o agente de aluno Ylana sendo criado no RMA no conteiner 2, este agente tem todas as funcionalidades que um agente Jade deve ter, porém só agente de grupo é que detém o controle do sistema e a comunicação entre os agentes.

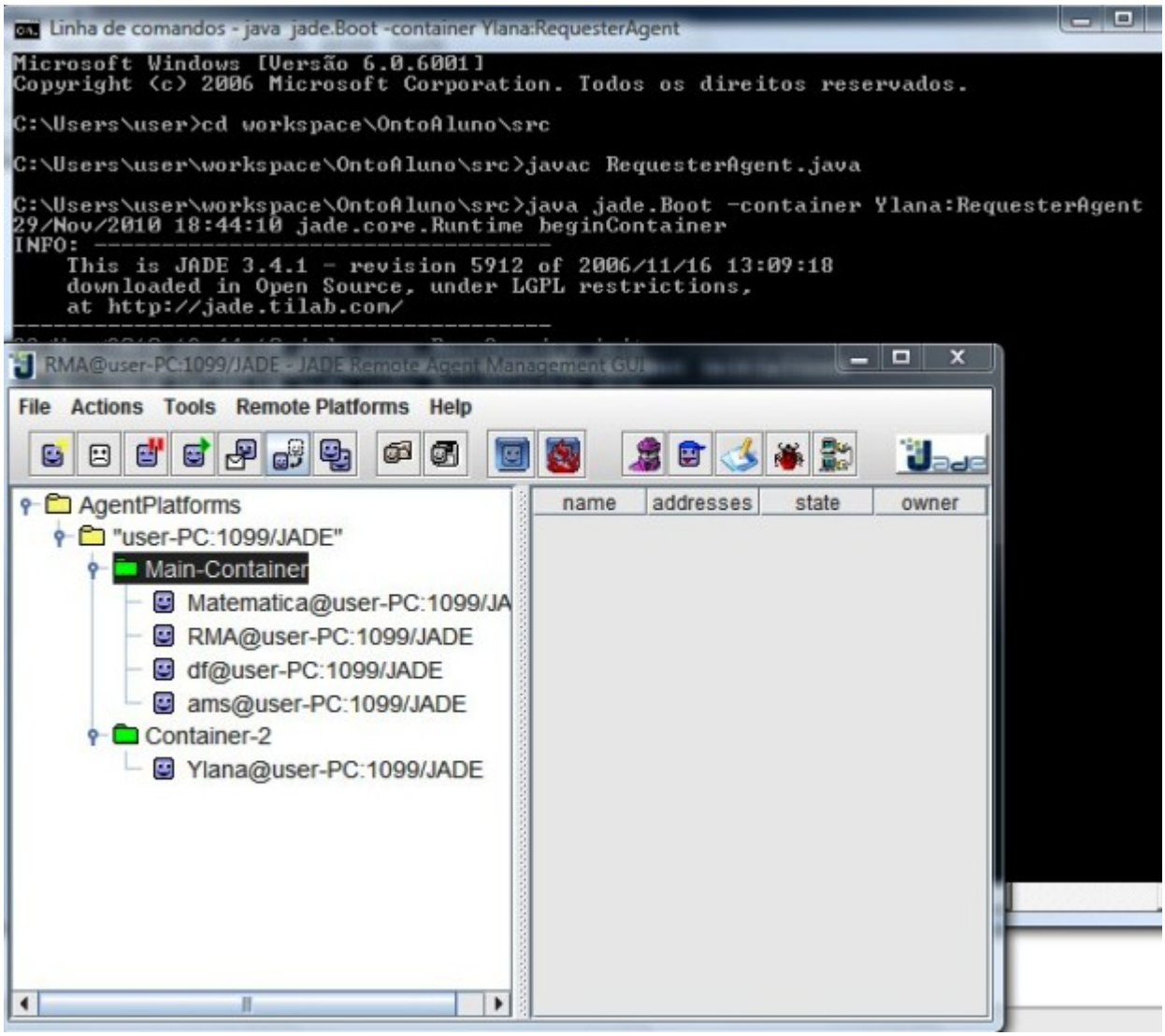

Figura 9.8: Carregando o 1º RequesterAgent no RMA - Aluno Ylana

Na FIGURA 9.9 podemos ver um RequesterAgent sendo carregado no sistema, ou seja agente de aluno (Pedro) para que seja estabelecida a comunicação entre 3 agentes dentro do sistema.

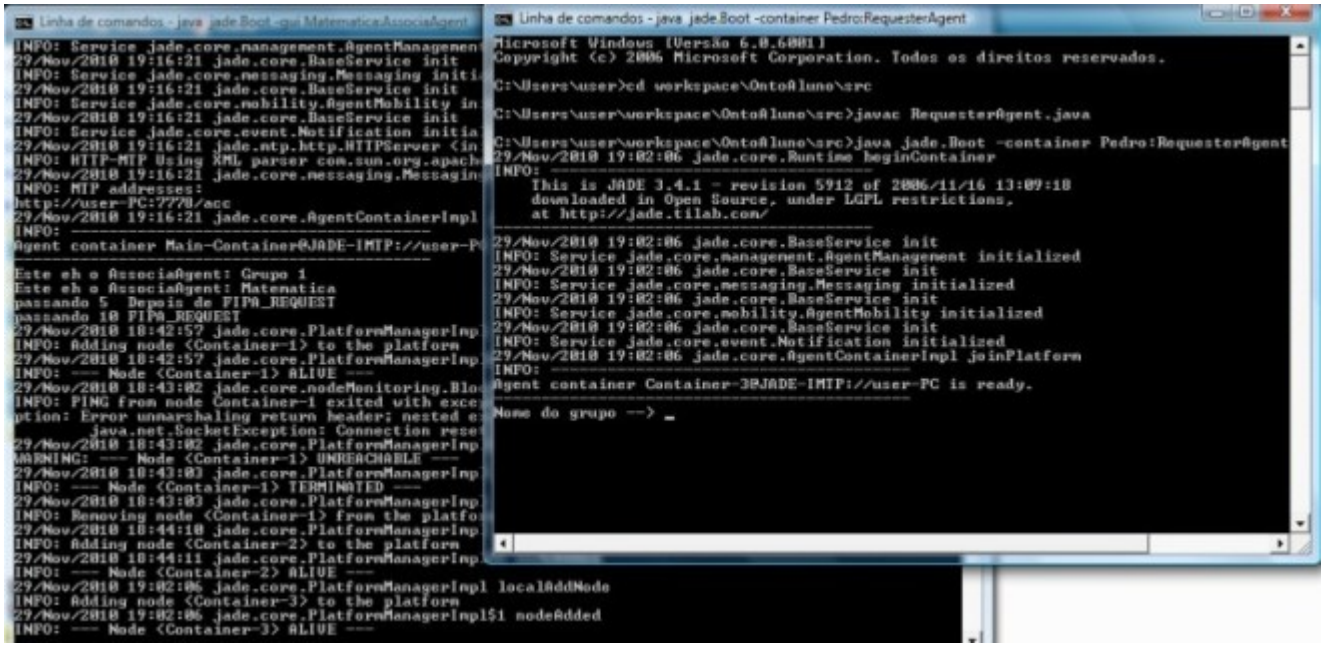

Figura 9.9: Carregando o 2º RequesterAgent - Agente Aluno (Pedro)

A FIGURA 9.10 mostram o agente de aluno Pedro sendo criado no RMA no conteiner 3.

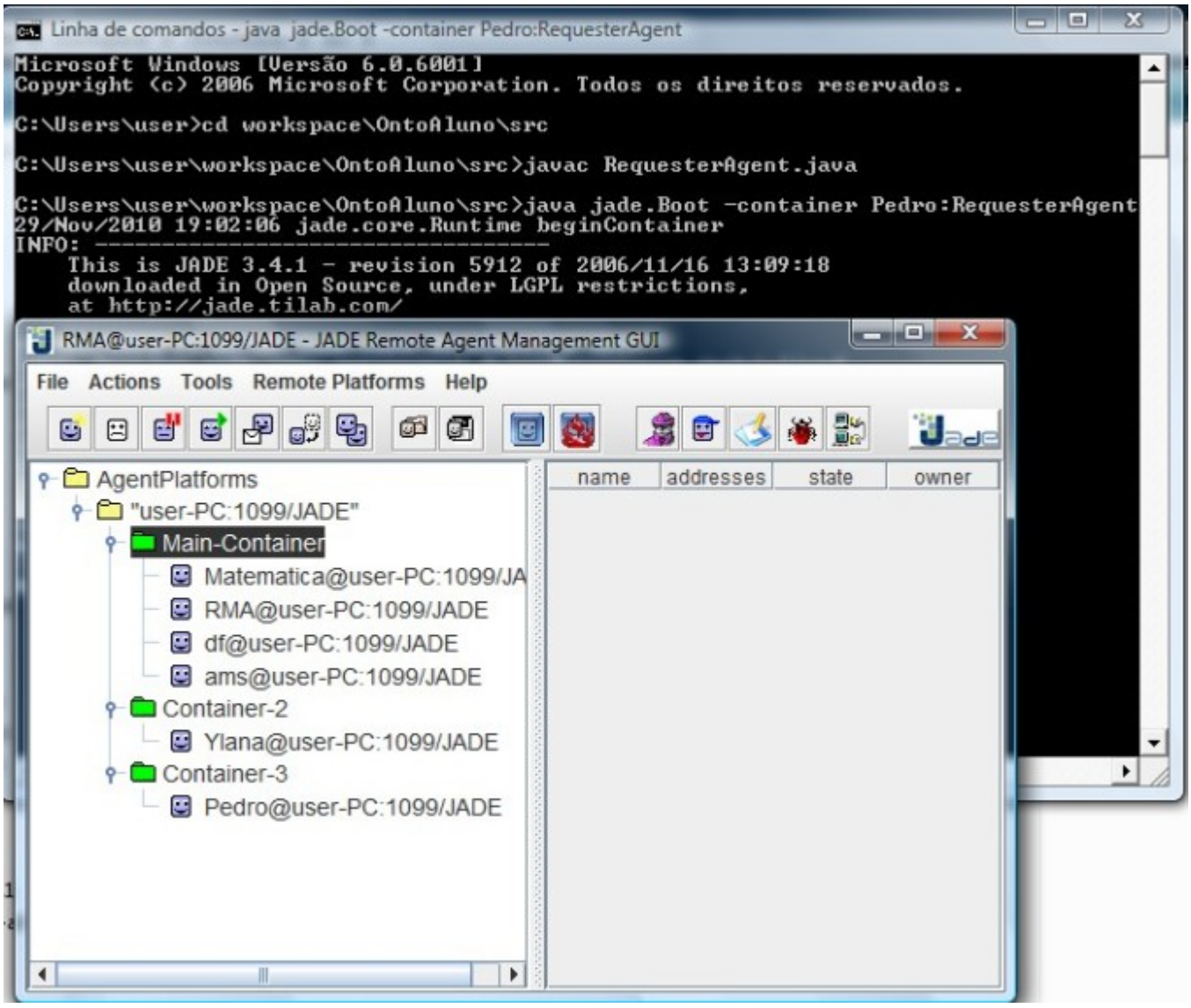

Figura 9.10: Carregando o 2º RequesterAgent no RMA - Aluno Pedro

Na FIGURA 9.11 podemos ver as funcionalidades inerentes aos agentes para aplicação, que já foram apresentadas em seções anteriores, no caso estão sendo testadas 5 funcionalidades para o sistema.

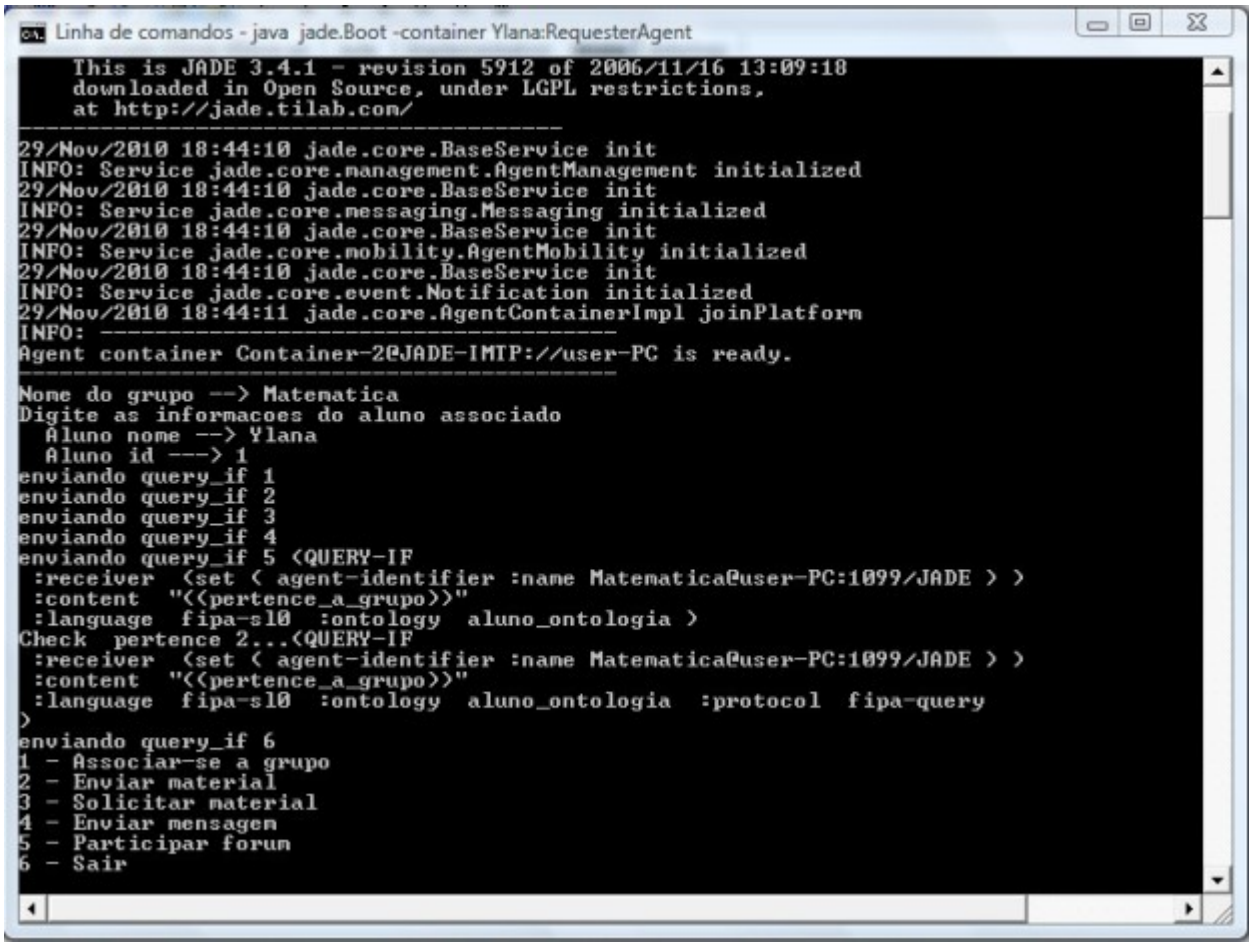

Figura 9.11: Funcionalidades do RequesterAgent - Agente de aluno

Capítulo 10

Conclusões

O presente trabalho teve por objetivo modelar uma ferramenta para EaD baseada em teorias socias e agentes inteligentes, criando-se todos tipos de agentes que estabelecem uma comunicação entre eles.

O framework Jade que comporta toda a infra-estrutura que promove o desenvolvimento e implementação de ambientes ou sistemas multiagentes voltados para a Educação a Distância. Os sistemas multiagentes não estão restritos somente ao uso em ambientes computacionais de ensino. Eles podem ser utilizados nas mais diversas áreas, tais como: controle de fluxo de rede, compartilhamento de arquivos em rede, simulações de ambientes sociais e etc.

Um aspecto muito importante na construção de sistemas multiagentes é o estabelecimento de mecanismos de comunicação e interoperabilidade entre os agentes, que podem ser obtidos pelos protocolos de comunicação da FIPA.

A organização FIPA surgiu como um esforço na tentativa de padronizar as tecnologias de sistemas multiagentes, que até então os mecanismos de comunicação eram completamente heterogêneos. Ela define todas as características que um sistema multiagente deve implementar e possuir para ser considerado interoperável, ou seja, ter a capacidade de se comunicar com os demais agentes presentes na plataforma.

A comunicação dos agentes no JADE é baseada nas regras estabelecidas pela FIPA, o que facilita a comunicação entre os agentes.

O objetivo de se criar agentes *Aluno* e *Grupo* neste trabalho, vem do facto de se propôr que os cursistas sejam mais autônomos no ambiente do seu curso. Que eles possam interagir, troca impressões, trocar material de estudo e sanar as suas dúvidas.

Com a utilização de ontologias baseadas nos perfis sociais dos alunos, é possível identificar os alunos que possuem maiores difuldades dentro do grupo e encontrar novas metodologias de ensino que possam atender o grau de dificuldade dos alunos.

É possível perceber que a criação desta ferramenta pode diminuir um pouco mais o trabalho do tutor dentro do curso, pois ele não precisará de estar permanentemente levantando questões

e tirando dúvidas aos alunos, pois estes têm a possibilidade de aprenderem uns com os outros e o tutor irá intervir quando ele achar necessário.

Espera-se que este trabalho possa servir de grande utilidade para cursos que são ministrados à distância, permitindo a consolidação da EaD.

Capítulo 11

Apêndice

11.1 Apêndice A

Os algoritmos do apêndice A mostram a implementação das classes do pacote ontoCursistas, para que se possa entender melhor como foi desenvolvida a ferramenta.

Aluno.java

```
* @author Ylana Figueiredo

package ontoCursistas;
import jade.content.Predicate;

public class Aluno implements Predicate{

    private String  _nome; //nome do aluno
    private Long    _id;   //identificador de aluno

    //Métodos set e get da classe Aluno
    public void setNome(String nome) {
        _nome=nome;
    }
    public String getNome() {
        return _nome;
    }
    public void setID(Long id) {
        _id=id;
    }
    public Long getID() {
        return _id;
  }


    public boolean equals(Aluno a){
        if (!_nome.equalsIgnoreCase(a.getNome()))
            return false;
        if (_id != null && a.getID() != null) // ID é um campo opcional de Aluno
            if (_id.longValue() != a.getID().longValue())
                return false;
        return true;
    }

}
```

Figura 11.1: Classe Aluno

Grupo.java

```
/**
* @author Ylana Figueiredo
*
*/

package ontoCursistas;

import jade.content.Concept;

public class Grupo implements Concept {

    private String  _nome;
    private String  _tipo;
    private Long    _id;  //identificador de aluno


    public void setNome(String nome) {
        _nome=nome;
    }
    public String getNome() {
        return _nome;
    }
    public void setTipoGr(String tipo) {
        _tipo=tipo;
    }
    public String getTipoGr() {
        return _tipo;
    }
    public void setID(Long id) {
        _id=id;
    }
    public Long getID() {
        return _id;
    }


    public boolean equals(Grupo g){
        if (!_nome.equalsIgnoreCase(g.getNome()))
            return false;
        if (!_nome.equalsIgnoreCase(g.getTipoGr()))
            return false;
        if (_id != null && g.getID() != null) // ID é um campo opcional de Aluno
            if (_id.longValue() != g.getID().longValue())
                return false;
        return true;
    }

}
```

Figura 11.2: Classe Grupo

AlunoCursista.java

```
/**
* @author Ylana Figueiredo
* Classe AlunoCursista
*/

package ontoCursistas;

import jade.content.Predicate;

public class AlunoCursista implements Predicate {
}
```

Figura 11.3: Classe AlunoCursista

Associa.java

```
/**
* @author Ylana Figueiredo
* classe que modela o grupo associando o aluno
*
*/

package ontoCursistas;
import jade.content.Concept;


public class Associa implements Concept{

    private Grupo    _grupo; //Define o grupo que associa aluno
    private Aluno    _aluno; //Define o aluno associado no grupo
    private AID      actor;

    //Metodos usados pelo JADE-framework
    public void setAluno(Aluno aluno) {
        _aluno= aluno;
    }
    public Aluno getAluno() {
        return _aluno;
    }

    public void setGrupo(Grupo grupo) {
        _grupo= grupo;
    }
    public Grupo getGrupo() {
        return _grupo;
    }

}
```

Figura 11.4: Classe Associa

AssociaError.java

```
package ontoCursistas;

import jade.content.Predicate;

public class AssociaError implements Predicate {
}
```

Figura 11.5: Classe AssociaError

```
/**
 * @author Ylana figueiredo
 * Classe PertenceAoGrupo : Teste se o aluno já está associado no grupo
 */
package ontoCursistas;

import jade.content.Predicate;

public class PertenceAoGrupo implements Predicate {

    private Grupo   _grupo;
    private Aluno   _aluno;


    public void setAluno(Aluno aluno) {
        _aluno=aluno;
    }
    public Aluno getAluno() {
        return _aluno;
    }

    public void setGrupo(Grupo grupo) {
        _grupo=grupo;
    }
    public Grupo getGrupo() {
        return _grupo;
    }

}
```

Figura 11.6: Classe PertenceAoGrupo

Modelo_Ontologia.java

```
*@author Viana Figueiredo


package ontoCursistas;

import jade.content.onto.*;

public class Modelo_Ontologia extends Ontology {

    //constante simbolica que contem o nome da ontologia
    public static final String NOME = "aluno_ontologia";

    // Vocabulario
    // Conceitos

    public static final String ALUNO      = "ALUNO";
    public static final String ALUNO_NOME = "nome";
    public static final String ALUNO_ID   = "id";

    public static final String GRUPO      = "GRUPO";
    public static final String GRUPO_NOME = "nome";
    public static final String GRUPO_TIPO = "tipo"; //tipo do grupo (matematica,
Fisica, etc)
    public static final String GRUPO_ID   =  "id";


    // Acções

    public static final String ASSOCIA       = "associa";
    public static final String ASSOCIA_ALUNO = "associa_aluno";
    public static final String ASSOCIA_GRUPO = "associa_grupo";


    // Predicados

    public static final String PERTENCE_A       = "pertence_a";
    public static final String PERTENCE_A_ALUNO = "grupo_contem";
    public static final String PERTENCE_A_GRUPO = "pertence_a_grupo";
    public static final String ASSOCIA_ERROR    = "associa_error";
    public static final String ALUNO_CURSISTA   = "aluno_cursista";

//  private static final String ALUNO_CURSISTA  = null;

    private static Ontology theInstance = new Modelo_Ontologia();

     /** Este metodo garante o acesso a uma unica ontologia.
      * return Ontologia objecto, contendo os conceitos da ontologia
      */

     public static Ontology getInstance() { // A classe Ontology eh propria do JADE
           return theInstance;
       }

     /**
       * Constructor da classe Modelo_Ontologia
       */
      private Modelo_Ontologia(){
          super(NOME, BasicOntology.getInstance());
```

Figura 11.7: Classe Modelo_Ontologia

Modelo_Ontologia.java

```
        try {
                add(new ConceptSchema(ALUNO), Aluno.class);
                add(new ConceptSchema(GRUPO), Grupo.class);

                add(new PredicateSchema(ASSOCIA), Associa.class);
                add(new PredicateSchema(ASSOCIA_ERROR), AssociaError.class);
                add(new PredicateSchema(ALUNO_CURSISTA), AlunoCursista.class);
                add(new PredicateSchema(PERTENCE_A), PertenceAoGrupo.class);

                add(new AgentActionSchema(ASSOCIA), Associa.class);

                ConceptSchema al = (ConceptSchema)getSchema(ALUNO);//para definir os
conceitos do GRUPO e ALUNO

                al = (ConceptSchema)getSchema(ALUNO);
                al.add(ALUNO_NOME, (PrimitiveSchema)getSchema(BasicOntology.STRING));
                al.add(ALUNO_ID, (PrimitiveSchema)getSchema(BasicOntology.INTEGER),
ObjectSchema.OPTIONAL);


                al = (ConceptSchema)getSchema(GRUPO);
                al.add(GRUPO_NOME, (PrimitiveSchema)getSchema(BasicOntology.STRING));
                al.add(GRUPO_TIPO, (PrimitiveSchema)getSchema(BasicOntology.STRING));
                al.add(GRUPO_ID, (PrimitiveSchema)getSchema(BasicOntology.INTEGER),
ObjectSchema.OPTIONAL);

                //Define os predicatos inerentes a GRUPO e ALUNO
                PredicateSchema ps = (PredicateSchema)getSchema(PERTENCE_A);
                ps.add(PERTENCE_A_ALUNO, (ConceptSchema)getSchema(ALUNO));
                ps.add(PERTENCE_A_GRUPO, (ConceptSchema)getSchema(GRUPO));

                AgentActionSchema as = (AgentActionSchema)getSchema(ASSOCIA);
                as.add(ASSOCIA_ALUNO, (ConceptSchema)getSchema(ALUNO));
                as.add(ASSOCIA_GRUPO, (ConceptSchema)getSchema(GRUPO));
            }
            catch(OntologyException oe) {
              oe.printStackTrace();
            }

        }
}
```

Figura 11.8: Classe Modelo_Ontologia (cont.)

11.2 Apêndice B

Os algoritmos do apêndice B mostram a implementação das classes setup AssociaAgent e RequesterAgent, para melhor entendimento da implementação da ferramenta.

AssociaAgent.java

```
/**
* @author Yiana Figueiredo
*
* Esta classe faz a associacao do aluno ao grupo. Ele tem dois comportamentos
* -SimpleAchieveREResponder para lidar com perguntas sobre os aluno que pertencem
*  um determinado grupo atraves do protocolo FIPA-Consulta.
* -O SimpleAchieveREResponder lida com as requisicoes de associacao
* atraves do protocolo FIPA-Request.
*
*/

import jade.proto.SimpleAchieveREResponder;


// esta classe é capaz de associar alunos ao grupo

public class AssociaAgent extends Agent {


    // AGENT BEHAVIOURS
    /**
     *  Comportamento que lida com associação dos alunos a um determinado grupo
     *  Seguindo o protocolo FIPA-Query
    */

    class HandleAssociaQueriesBehaviour extends SimpleAchieveREResponder{

        //Constructor para a classe HandleAssociaQueriesBehaviour
    public HandleAssociaQueriesBehaviour(Agent myAgent){
        super(myAgent, MessageTemplate.and(
                                            MessageTemplate.MatchProtocol(FIPANames.Int
eractionProtocol.FIPA_QUERY),
                                            MessageTemplate.MatchOntology(Modelo_Ontolo
gia.NOME)));
    }

    //Este metodo eh chamado quando uma mensagem QUERY-IF ou QUERY-REF eh recebida
    public ACLMessage prepareResponse(ACLMessage msg) {

        ACLMessage reply = msg.createReply();

        // A mensg QUERY pode ser QUERY-REF. Neste caso responde
        // com NOT_UNDERSTOOD

        if (msg.getPerformative() != ACLMessage.QUERY_IF){
            reply.setPerformative(ACLMessage.NOT_UNDERSTOOD);
                String content = "(" + msg.toString() + ")";
            reply.setContent(content);
            return(reply);
        }

        try {

            // Obtem um predicado correcto mediante a consulta
            Predicate pred =
(Predicate)myAgent.getContentManager().extractContent(msg);
            if (!(pred instanceof PertenceAoGrupo)) {
            // Se o predicado da consulta for PertenceAoGrupo
            // responde com NOT_UNDERSTOOD
                reply.setPerformative(ACLMessage.NOT_UNDERSTOOD);
```

Figura 11.9: Classe AssociaAgent

AssociaAgent.java

```
                String content = "(" + msg.toString() + ")";
            reply.setContent(content);
            return(reply);
        }

        // Responde
        reply.setPerformative(ACLMessage.INFORM);
        PertenceAoGrupo pag = (PertenceAoGrupo)pred;
        Aluno a = pag.getAluno();
        Grupo g = pag.getGrupo();
        if (((AssociaAgent) myAgent).pertence(a, g))
            reply.setContent(msg.getContent());
         else {

            Ontology o = getContentManager().lookupOntology(Modelo_Ontologia.NOME)
            AbsPredicate not = new AbsPredicate(SLVocabulary.NOT);
                not.set(SLVocabulary.NOT_WHAT, o.fromObject(pag));
            myAgent.getContentManager().fillContent(reply, not);
        }

    }
    catch (Codec.CodecException fe) {
        System.err.println(myAgent.getLocalName()+" Fill/extract content
unsucceeded. Reason:" + fe.getMessage());
    }
    catch (OntologyException oe){
        System.err.println(myAgent.getLocalName()+" getRoleName() unsucceeded.
Reason:" + oe.getMessage());
    }

    return (reply);
}// Fim do metodo handleQueryMessage()


}//Fim da classe HandleAssociaQueriesBehaviour


/**
 * Este behaviour lida com uma acao de Associa que jah foi requerida
 * Seguindo o protocolo FIPA-Request
 */

class HandleAssociaBehaviour extends SimpleAchieveREResponder {

    //Constructor para a classe HandleAssociaBehaviour.

    public HandleAssociaBehaviour(Agent myAgent){
        super(myAgent,
MessageTemplate.MatchProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST));
    }

    /**
     * Este metodo implementa a interface FipaRequestResponderBehaviour.Factory.
     * Que sera chamada com o protocolo FipaRequestResponderBehaviour quando for
     * requrida uma nova associacao de aluno para instanciar um nova acao
     * HandleEngageBehaviour requisitada
     * Este metodo manipula a acao de associar o aluno
     */
```

Figura 11.10: Classe AssociaAgent (cont1.)

AssociaAgent.java

```
        public ACLMessage prepareResultNotification(ACLMessage request, ACLMessage
response) {

            // Prepara uma dummy ACLMessage usada para criar
            //o conteudo de todas as mensgens_resposta
               ACLMessage msg = request.createReply();

               try{
                   // Obtem a acao de requeste
                   Action a =
(Action)myAgent.getContentManager().extractContent(request);
                   Associa e = (Associa) a.getAction();
                   Aluno s = e.getAluno();
                   Grupo g = e.getGrupo();

                   // Checa se o id. If > 0 --> ACEITA, else RECUSA and exit
                   int result = ((AssociaAgent) myAgent).doAssocia(s, g);

                   // Responda de acordo com o resultado obtido
                   if (result > 0){
                       // OK --> INFORMA a ação feita
                       Done d = new Done();
                       d.setAction(a);
                       myAgent.getContentManager().fillContent(msg, d);
                       msg.setPerformative(ACLMessage.INFORM);
                   }
                   else{
                       // NOT OK --> FALHA
                       ContentElementList l = new ContentElementList();
                       l.add(a);
                       l.add(new AssociaError());
                       myAgent.getContentManager().fillContent(msg, l);
                       msg.setPerformative(ACLMessage.FAILURE);
                   }

               }
               catch (Exception fe){
                   System.out.println(myAgent.getName() + ": Error handling the
engagement action.");
                   System.out.println(fe.getMessage().toString());
               }

               // System.out.println(msg);
               return msg;

        } // fim da classe prepareResultNotification

        public ACLMessage prepareResponse(ACLMessage request) {

           // Prepara uma dummy ACLMessage usada para criar
            //o conteudo de todas as mensagens_resposta
               ACLMessage temp = request.createReply();
               try{
                   // Obtem a ação de resquest.
                   Action a = (Action)getContentManager().extractContent(request);
                   Associa e = (Associa) a.getAction();
                   Aluno s = e.getAluno();
                   Grupo g = e.getGrupo();

                   // Checa o id do aluno. if id > 0 --> ACEITA, else REJEITA and exit
```

Figura 11.11: Classe AssociaAgent(cont2.)

```
                                AssociaAgent.java

                    if (a.getID().intValue() > 0){
                        //Aceita a associacao do aluno no grupo
                        ContentElementList l = new ContentElementList();
                        l.add(a);
                        l.add(new TrueProposition());
                        getContentManager().fillContent(temp, l);
                        temp.setPerformative(ACLMessage.AGREE);
                    }
                    else {
                        ContentElementList l = new ContentElementList();
                        l.add(a);
                        l.add(new AlunoCursista());
                        getContentManager().fillContent(temp, l);
                        temp.setPerformative(ACLMessage.REFUSE);
                    }

                } catch (Exception fe){
                    fe.printStackTrace();
                    System.out.println(getName() + ": Error handling the engagement
action.");
                    System.out.println(fe.getMessage().toString());
                }

                return temp;

            }// fim da classe prepareResponse

        }//Fim da classe HandleAssociaBehaviour


        // Variaveis locais do Agent
        private ontoCursistas.Grupo representedGrupo; // Grupo capaz de associar aluno
        private List alunos;    // alunos que jah estao associados ao grupo


        //CONSTRUCTOR do Agente
        public AssociaAgent(){
            super();

            representedGrupo = new Grupo();
            representedGrupo.setNome("Grupo 1");
            representedGrupo.setTipoGr("Matematica");
            representedGrupo.setID(new Long(1));

            alunos = new ArrayList();
        }

        // AGENT SETUP
        protected void setup() {
            System.out.println("Este eh o AssociaAgent: "+representedGrupo.getNome());
            System.out.println("Este eh o AssociaAgent: "+representedGrupo.getTipoGr());

            // Registrita o codec para SL0 language
            getContentManager().registerLanguage(new SLCodec(),
FIPANames.ContentLanguage.FIPA_SL0);

            // Registra a ontologia useda pela aplicação
            getContentManager().registerOntology(Modelo_Ontologia.getInstance());

            // Cria e adiciona o behaviour para manipular as QUERIES usando the Ontologia
```

Figura 11.12: Classe AssociaAgent (cont3.)

AssociaAgent.java

```
de aluno
        addBehaviour(new HandleAssociaQueriesBehaviour(this));

        // Cria e adiciona o behaviour para manipular REQUESTS usando Ontologia de
Aluno
        MessageTemplate mt = MessageTemplate.and(
                                                MessageTemplate.MatchProtocol(FIPANames.Int
eractionProtocol.FIPA_REQUEST),
                                                MessageTemplate.MatchOntology(Modelo_Ontolo
gia.NOME));
        HandleAssociaQueriesBehaviour b = new HandleAssociaQueriesBehaviour(this);
        HandleAssociaBehaviour c = new HandleAssociaBehaviour(this);
        addBehaviour(b);
        addBehaviour(c);
    }// do SETUP do AGENT


    //METODOS do AGENTE
    boolean pertence(Aluno a, Grupo g){
        boolean ehUmAluno = false;
        Iterator i = alunos.iterator();
        while (i.hasNext()){
            Aluno empl = (Aluno) i.next();
            if (g.equals(empl))
                ehUmAluno = true;
        }

        if (g.equals(representedGrupo))
            return ehUmAluno;
        else
            return !ehUmAluno;
    }

    int doAssocia(Aluno a, Grupo g){
        if (!g.equals(representedGrupo))
            return (-1);
        else
            alunos.add(a);
        return (1);
    }

}//Fim da classe AssociaAgent
```

Figura 11.13: Classe AssociaAgent (cont4.)

RequesterAgent.java

```
/**
* @author Ylana Figueiredo
*
* -Esta classe obtem as informações detalhadas dos aluno que que querem se associar ao
 grupo
*   e solicita ao AssociarAgent para associar alunos ao grupo
* - As informações dos alunos são obtidas pelo método onStart()
* -Uma consulata é feita em AssociarAgente para ver se o aluno já está ou não associado
 ao grupo
*   A consulta e feita atraves dos protocolos FIPA.
* - De acordo com o resultado da consulta, uma solicitação eh enviada para o
AssociarAgente
*
*/


import jade.lang.acl.ACLMessage;

/**
 * Este agente é capaz associar aluno através de requisição
 * feita um agente de associação que resolve essa requisição
 * de associação de aluno no grupo
 *
 * Recebe ciclicamente do próprio usuario as informações de aluno, afim de manipular o
agente aluno
 * com a ajuda da classe AssociaAgent
 *
 * Campos com mais de uma string devem estar entre ""
 *
 * */


public class RequesterAgent extends Agent{

    class HandleAssociaBehaviour extends SequentialBehaviour{

        //Variaveis locais
        Behaviour queryBehaviour = null;
        Behaviour requestBehaviour = null;

        // Construtor
        public HandleAssociaBehaviour(Agent myAgent){
            super(myAgent);
        }

        //Eh executdo no inicio do behaviour-comportamento do agente
        public void onStart(){

            // Obtem as informações do aluno associado ao grupo
            try{
                BufferedReader buff = new BufferedReader(new
InputStreamReader(System.in));

                Aluno a = new Aluno(); //cria um aluno
                System.out.println("Digite as informacoes do aluno associado");
                System.out.print("  Aluno nome --> ");
                a.setNome(buff.readLine());
                a.setNome(buff.readLine());
                System.out.print("  Aluno id ---> ");
```

Figura 11.14: Classe Requester

RequesterAgent.java

```
                s.setID(new Long(buff.readLine()));

                //cria um objeto representa que o aluno a pertence ou esta associado ao
grupo g
                PertenceAoGrupo pag = new PertenceAoGrupo();
                pag.setAluno(a);
                pag.setGrupo(((RequesterAgent) myAgent).g);

                Ontology o =
myAgent.getContentManager().lookupOntology(Modelo_Ontologia.NOME);
                //  Cria uma ACL message para saber se o agente
                //que associa eh verdadeiro ou falso
                ACLMessage queryMsg = new ACLMessage(ACLMessage.QUERY_IF);
                queryMsg.addReceiver(((RequesterAgent) myAgent).associar);
                queryMsg.setLanguage(FIPANames.ContentLanguage.FIPA_SL0);
                queryMsg.setOntology(Modelo_Ontologia.NOME);

                try {
                    myAgent.getContentManager().fillContent(queryMsg, pag);
                } catch (Exception e) {
                    e.printStackTrace();
                }

                // cria e adiciona um behaviou para consultar se Associa agent testou
se
                // o aluno a esta associado ao grupo g seguindo o portocolo FIPAQuery
                queryBehaviour = new CheckPertenceBehaviour(myAgent, queryMsg);
                addSubBehaviour(queryBehaviour);

            }//Fim do try
            catch (IOException ioe) {
                System.err.println("I/O error: " + ioe.getMessage());
            }


        }//Fim da classe OnStrat()

        // Eh executado ao final do behaviour do agente
        public int onEnd() {
            // Verifica se o usuário que continuar a aplicação
            try{
                BufferedReader buff = new BufferedReader(new
InputStreamReader(System.in));
                System.out.println("Gostaria de continuar a aplicacao?[s/n] ");
                String stop = buff.readLine();
                if (stop.equalsIgnoreCase("s"))
                    {
                        reset(); //  Executa um behaviour ciclico
                        myAgent.addBehaviour(this);
                    }
                else
                    myAgent.doDelete(); // Exit
            }
            catch (IOException ioe) {
                System.err.println("I/O error: " + ioe.getMessage());
            }
            return 0;
        }
        //Extende o método reset, a fim de remover sub-behaviours que sao adicionados
        //dinamicamente
```

Figura 11.15: Classe Requester (cont1.)

RequesterAgent.java

```
        public void reset(){
            if (queryBehaviour != null){
                removeSubBehaviour(queryBehaviour);
                queryBehaviour = null;
            }
            if (requestBehaviour != null){
                removeSubBehaviour(requestBehaviour);
                requestBehaviour = null;
            }
            super.reset();
        }

    }//Fim da classe HandleAssociaBehaviour

    /**
     * Este comportamento incorpora a verificação de que o aluno a ser adicionado
     * já esteja associado ao grupo, que eh feito com o porotocolo de interação FIPA
     * -Query
     * */

    class CheckPertenceBehaviour extends SimpleAchieveREInitiator {

        //Construtor
        public CheckPertenceBehaviour(Agent myAgent, ACLMessage queryMsg){
            super(myAgent, queryMsg);
            queryMsg.setProtocol(FIPANames.InteractionProtocol.FIPA_QUERY);
        }

        protected void handleInform(ACLMessage msg) {

            try{
                AbsPredicate cs =
(AbsPredicate)myAgent.getContentManager().extractAbsContent(msg);
                Ontology o =
myAgent.getContentManager().lookupOntology(Modelo_Ontologia.NOME);

                if (cs.getTypeName().equals(Modelo_Ontologia.PERTENCE_A)) {
                    // Indica que o aluno ja pertence ao grupo
                    PertenceAoGrupo pag = (PertenceAoGrupo)o.toObject((AbsObject)cs);
                    Aluno a = (Aluno) pag.getAluno();
                    Grupo g = (Grupo) pag.getGrupo();
                    System.out.println("Aluno " + a.getNome() + " jah pertence ao grupo
" + g.getNome());
                }

                else if (cs.getTypeName().equals(SLVocabulary.NOT)){

                    // Indica que o aluno ainda não pertence ao grupo
                    // Obtem os detalhes do grupo e o grupo e cria um
                    //objeto que a ação de associar

                    PertenceAoGrupo pag =
(PertenceAoGrupo)o.toObject(cs.getAbsObject(SLVocabulary.NOT_WHAT));
                    Aluno a = (Aluno) pag.getAluno();
                    Grupo c = (Grupo) pag.getGrupo();
                    Associa e = new Associa();
                    e.setAluno(a);
                    e.setGrupo(c);
                    Action ac = new Action();
                    ac.setActor(((RequesterAgent) myAgent).associar);
```

Figura 11.16: Classe Requester (cont3.)

RequesterAgent.java

```
                    ac.setAction(e);

                    // Cria uma ACL message que responde as acoes abaixo
                    ACLMessage requestMsg = new ACLMessage(ACLMessage.REQUEST);
                    requestMsg.addReceiver(((RequesterAgent) myAgent).associar);
                    requestMsg.setLanguage(FIPANames.ContentLanguage.FIPA_SLO);
                    requestMsg.setOntology(Modelo_Ontologia.NOME);

                    try {
                        myAgent.getContentManager().fillContent(requestMsg, a);
                    } catch (Exception pe) {
                    }

                    ((HandleAssociaBehaviour) parent).requestBehaviour = new
RequestAssociaBehaviour(myAgent, requestMsg);
                    ((SequentialBehaviour)
parent).addSubBehaviour(((HandleAssociaBehaviour) parent).requestBehaviour);

                }//fim do else if

                else{

                    System.out.println("Resposta inesperada do agente participativo");
                }
            }
            catch (Codec.CodecException fe) {
                System.err.println("FIPAException in fill/extract Msgcontent:" +
fe.getMessage());
            }
            catch (OntologyException fe) {
                System.err.println("OntologyException in getRoleName:" +
fe.getMessage());
            }
        }
    }//Fim da classe CheckPertenceBehaviour

    /**
     * Esse behaviour incorpora um aluno no grupo indicado
     * Seguindo o protocolo de interacao do FIPA-request
     * */

    class RequestAssociaBehaviour extends SimpleAchieveREInitiator {

        // Construtor
        public RequestAssociaBehaviour(Agent myAgent, ACLMessage requestMsg) {
            super(myAgent, requestMsg);
            requestMsg.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
        }

        protected void handleAgree(ACLMessage msg) {
            System.out.println("Aluno associado no grupo. Aguarde para completar a
notificacao...");
        }
        protected void handleInform(ACLMessage msg) {
            System.out.println("O aluno foi associado com sucesso");
        }
        protected void handleNotUnderstood(ACLMessage msg) {
            System.out.println("O pedido de associacao no grupo nao compreendido pelo
Agente!!! ");
        }
```

Figura 11.17: Classe Requester (cont4.)

RequesterAgent.java

```
        protected void handleFailure(ACLMessage msg) {
            System.out.println("Associacao do aluno Falhou!!! ");
            // Obtem a falha e comunica ao usuario
            try{
                AbsPredicate absPred =
(AbsPredicate)myAgent.getContentManager().extractContent(msg);

                System.out.println("The reason is: " + absPred.getTypeName());
            }
            catch (Codec.CodecException fe){
                System.err.println("FIPAException reading failure reason: " +
fe.getMessage());
            }
            catch (OntologyException oe){
                System.err.println("OntologyException reading failure reason: " +
oe.getMessage());
            }
        }

        protected void handleRefuse(ACLMessage msg) {
            System.out.println("Associacao recusada!!! ");

            try{
                AbsContentElementList list =
(AbsContentElementList)myAgent.getContentManager().extractAbsContent(msg);
                AbsPredicate absPred = (AbsPredicate)list.get(1);
                System.out.println("The reason is: " + absPred.getTypeName());
            }
            catch (Codec.CodecException fe){
                System.err.println("FIPAException reading refusal reason: " +
fe.getMessage());
            }
            catch (OntologyException oe){
                System.err.println("OntologyException reading refusal reason: " +
oe.getMessage());
            }
        }

    }//Fim da classe RequestAssociaBehaviour

    // Variaveis locais do agente
    AID associar; // As requisicoes do AID do agente deverao ser enviada
    Grupo g;   // ao grupo no qual os alunos pertencem

    // AGENT SETUP
    protected void setup(){

        // Registra o codec para o SL0 language
        getContentManager().registerLanguage(new SLCodec(),
FIPANames.ContentLanguage.FIPA_SL0);

        // Registra a ontologia usada para esta aplicação
        getContentManager().registerOntology(Modelo_Ontologia.getInstance());

        //Obtem do usuario o nome do agente que fez a requisição de associação
        // que deverá ser enviada a
        try {
            BufferedReader buff = new BufferedReader(new InputStreamReader(System.in));

            System.out.print("Digite o nomo do Agente Local de Grupo --> ");
```

Figura 11.18: Classe Requester (cont5.)

RequesterAgent.java

```
            String name = buff.readLine();
            associar = new AID(name, AID.ISLOCALNAME);

            g  = new Grupo();
        }
        catch (IOException ioe) {
            System.err.println("I/O error: " + ioe.getMessage());
        }

        // Cria e adiciona o behaviourprincipal do agente
        addBehaviour(new HandleAssociaBehaviour(this));

    }//Fim da classe setup()

}// fim da classe RequesterAgent
```

Figura 11.19: Classe Requester (cont6.)

Referências Bibliográficas

Allahverdi (1999).

Baker, A. (1997). *A java-based agent framework for multiagent systems.* PhD thesis, University of Cincinnati, Department of Electrical & Computer Engineering and Computer Science.

Bassani, P. B. S.; Flores, M. B. e Ritzel, M. (2007). Modelando acessibilidade na web: uma proposta para o desenvolvimento de. *Novas Tecnologias na Educação*, 5(1).

Bradshaw, J. M. (1997). An introduction to software agents. In *Software Agents*, Massachusetts - USA. MIT Press.

BRENNER, W.; RÜDIGERr, Z. e WITTIG, H. (1998). Intelligent software. *Springer-Verlag.*

computeRs (2010). Representação estruturada da informação.

Daniels, H. (2003). *Vygotsky e a Pedagogia.* Loyola Edições, São Paulo, SP, Brasil, 1 edição.

Falbo, R. d. A. (2005). Using ontologies to add semantics to a software engineering environment. *17th International Conference on*, pp. 151–156.

Fensel, D. (2001). Silver bullet for knowledge. *Springer Verlag - International Publisher Science.*

Finin, T.; Labrou, Y. e Mayfeld, J. (1997). Kqml as an agent communication language. *BRADSHAW*, pp. 291–316.

FIPA (2010).

Gómez-Peréz, A. (1999). Evaluation of taxonomic knowledge in ontologies and knowledge bases 12ª. *Workshop ON Knowledge Acquisittion, Modeling and Management.*

Lopes, J. G. R. C. (2005). Matching semântico de recursos de computacionais em ambientes de grades com múltiplas ontologias. Master thesis, Instituto de Ciências Exatas, Brasília, DF, Brasil.

Maedche, A. D. (2002). *Ontology Learning for the Semantic Web.* Editora Kluwer Academic.

Moreno, A. O. e Hernández, C. P. (2000). Reusing the mikrokosmos ontology for concept-based multilingual terminology data bases. In *Mikrokosmos*, Athens, Greece. Proceedings of 2ª Conference on Language Resources and Evaluation (LREC 2000).

Puc-Rio (2003). Ontologia. *Puc-Rio, Teses abertas.*

Revista Escola - Durkheim (2008). Émile durkheim - o criador da sociologia da educação. `http://revistaescola.abril.com.br/historia/pratica-pedagogica/criador-sociologia-educacao-423124.shtml`.

Revista Escola - Vygotsky (2008). Lev vygotsky, o teórico do ensino como processo social. `http://revistaescola.abril.com.br/historia/pratica-pedagogica/lev-vygotsky-teorico-423354.shtml`.

RUSSEL, S. e NORVIG, P. (1995). *Artificial Intelligence a Modern.* Prentice-Hall.

Severo, C. E. P.; Passerino, L. M.; Koch, S. H. S.; Maciel, M. e C., G. J. (2009). Uma ontologia para categorias de mediação segundo uma abordagem epistemológica baseada na interação social. *Novas Tecnologias na Educação*, 7(3).

SHOHAM, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60(1).

Telecom-Itália (2010).

Vygotsky, L. S. (2001). *A Formação Social da Mente.* Martins Editora, São Paulo, SP, Brasil.