UNIVERSIDADE FEDERAL DE PELOTAS
INSTITUTO DE FÍSICA E MATEMÁTICA
DEPARTAMENTO DE INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

UM AGENTE PEDAGÓGICO ANIMADO NO PAPEL DE LMS MANIPULANDO OBJETOS INTELIGENTES DE APRENDIZAGEM

BEATRIZ WILGES

PELOTAS, 2006

BEATRIZ WILGES

UM AGENTE PEDAGÓGICO ANIMADO NO PAPEL DE LMS MANIPULANDO OBJETOS INTELIGENTES DE APRENDIZAGEM

Trabalho acadêmico apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Ricardo Azambuja Silveira

PELOTAS, 2006

BANCA EXAMINADORA:

_____

Prof. Ricardo Azambuja Silveira, Dr. (Orientador)

_____

Prof. Anderson Priebe Ferrugem, MSc.

_____

Prof^{a}. Rosa Maria Vicari, Dr^{a}.

Agradecimentos

Em especial gostaria de agradecer a minha família, que me apoiou durante esses quatro anos de faculdade mostrando-se presente mesmo com toda distância que nos separou.

Gostaria também de agradecer aos meus colegas do grupo de pesquisa (IATE), Joel e Michele: obrigada pela dedicação e incentivo. Ao Gustavo, mais do que um amigo, alguém especial que esteve ao meu lado: obrigada pelo carinho e atenção. E aos meus amigos que de alguma forma contribuíram para o meu crescimento.

Agradeço a dedicação de todos os professores do curso, que sempre buscaram determinação em todas as suas atividades, mesmo com tantas barreiras impostas. De modo particular, ao professor José Luís Almada Güntzel que sempre se mostrou mais do que um grande amigo, um professor dedicado que sempre acreditou nos alunos e lutou por eles. Ajudando em tudo que estava ao seu alcance.

E por fim, gostaria de agradecer ao meu professor, amigo e orientador Ricardo Azambuja Silveira que me acompanhou durante um bom tempo no curso, me dando grandes oportunidades de crescimento tanto pessoal, como profissional.

Resumo

Este trabalho tem como objetivo propor avanços no uso dos métodos de comunicação e na capacidade de aprendizagem que os agentes possuem, com a finalidade de desenvolver um ambiente educacional, onde exista um Agente Pedagógico Animado (APA) manipulando Objetos Inteligentes de Aprendizagem (ILO). O objetivo de se desenvolver um ambiente de aprendizagem na forma proposta é possibilitar a construção de objetos de aprendizagem a partir do chamado paradigma de agentes, inseridos em uma arquitetura baseada em Sistemas Multiagentes (MAS), utilizando o modelo de referência proposto pela *Foundation for Intelligent Physical Agents* (FIPA). Também deseja-se possibilitar a construção de um Sistema de Gerenciamento de Aprendizagem - *Learning Management Systems* (LMS), sob a forma de um personagem animado com características de agente, também compatível com o padrão FIPA, capaz de interagir com Objetos Inteligentes de Aprendizagem construídos de acordo com a arquitetura proposta.

Em trabalhos anteriores do grupo de pesquisa foi proposto um avanço na idéia de objetos de aprendizagem. Considera-se que a utilização de tecnologias de agentes em objetos de aprendizagem retorne uma flexibilidade e adaptabilidade maiores aos mesmos, transformando-os em objetos inteligentes de aprendizagem, que são agentes desempenhando o mesmo papel de um objeto de aprendizagem.

Palavras-Chave: Agente Pedagógico Animado. SMA. Objetos Inteligentes de Aprendizagem. LMS. FIPA.

Abstract

This work highlights an approach which looks for using the agents' abilities on learning and communication methods that they supply. The aim of this work is to develop an educational environment containing an Animated Pedagogical Agent (APA) managing Intelligent Learning Objects (ILO). The purpose of building a learning environment with this approach is to allow the production of learning objects based on the principle of agents' technology, which are inserted in a Multiagent System (MAS) architecture. In order to achieve this aim we use the Foundation for Intelligent Physical Agents (FIPA) reference model to implement the agents and an animated pedagogical agent as well, playing the role of an Learning Management Systems (LMS), using agent's skills, which are also in accordance to FIPA standards.

Previous works of the research group proposed advances in the learning object idea. Considering the use of agents' technologies in learning objects, a higher flexibility and adaptability is expected, transforming them into intelligent learning objects, which are agents playing the role of a learning object.

Keywords: Animeted Pedagogical Agent. MAS. Intelligent Learning Object. LMS. FIPA.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACL - *Agent Communication Language* – Linguagem de Comunicação de Agentes

AMS - *Agent Management System* – Sistema de Gerenciamento de Agentes

AP - *Agent Platform* – Plataforma de Agentes

API - *Application Programming Interface*

CM - *Conversation Manager* – Gerenciador de Conversação

CL - *Content Language* – Linguagem de Conteúdo

DF - *Directory Facilitator* – Facilitador de Diretórios

EAD - Educação a Distância

FIPA - *Foundation for Intelligent Physical Agents*

FIPA-OS - FIPA – *Open Source*

IA - Inteligência Artificial

IEEE - *Institute of Electrical and Electronics Engineers*

ILO - *Intelligent Learning Object* – Objeto Inteligente de Aprendizagem

ILOR - *Intelligent Learning Object Repository* – Repositório de Objetos Inteligentes de Aprendizagem

LMS - *Learning Management System* – Sistema Gerenciador de Aprendizagem

LOM - *Learning Object Metadata* – Metadados para Objetos de Aprendizagem

LTSC - *Learning Technologies Standard Comitee*

MAIDE - Modelagem de Ambientes Inteligentes Distribuídos de Aprendizagem

MTS - *Message Transport System* – Sistema de Transporte de Mensagens

MAS - *MultiAgent Systems* - Sistema Multiagente

TM - *Task Manager* – Gerenciador de Tarefas

Sumário

1 Introdução

Esta proposta partiu da possibilidade de se utilizar a idéia de Objetos Inteligentes de Aprendizagem proposta por Silveira (2005) de modo que eles sejam reaproveitados, de modo mais eficaz, levando em conta a sua interoperabilidade, dentro de outros ambientes educacionais e, ao mesmo tempo, agregar um Agente Pedagógico Animado no papel de um Sistema Gerenciador de Aprendizagem (LMS - *Learning Management System).*

Johnson (2000) e Jaques (2003) afirmam que agentes pedagógicos animados são o novo paradigma para ambientes de ensino, pois estes agentes podem demonstrar tarefas complexas, empregar locomoção e gestos para focar a atenção de alunos nos aspectos mais importantes das tarefas e também expressar respostas emotivas. Eles são responsáveis por auxiliar o aluno em sua interação com o ambiente educacional, sendo capazes de guiá-lo em suas ações. Portanto, afirma-se que o processo de ensino-aprendizagem pode se tornar mais eficiente com a utilização de agentes pedagógicos animados.

Segundo Bradshaw (1997), um agente é uma entidade de software que trabalha continuamente e de forma autônoma em um ambiente educacional geralmente habitado por outros agentes. Um agente é capaz de interferir neste ambiente de forma flexível e inteligente sem requerer intervenção humana ou direcional. Idealmente um agente deve ter a capacidade de aprender através de suas experiências passadas e, se ele habita um ambiente com outros agentes, ele deve ser capaz de se comunicar e cooperar com eles.

De acordo com Ferber (1999), um SMA é uma aplicação de computação distribuída composta por um conjunto de processos autônomos, heterogêneos, assíncronos, e inteligentes, os quais são chamados de "agentes". Eles são capazes de cooperar uns com os outros para resolver problemas complexos que não seria possível resolvê-los individualmente.

Para estabelecer os padrões de comunicação, e ao mesmo tempo tornar os agentes compatíveis entre si, estão sendo utilizados os padrões da *Foundation for Intelligent Physical Agents* - FIPA (FIPA, 2004). Esta organização define todas as características que um sistema multiagente deve implementar e possuir para ser

considerado interoperável.

Dentro deste contexto, acredita-se que dotar um objeto de aprendizagem de características típicas de agentes, tais como autonomia, conhecimento sobre si próprio, e objetivos podem agregar valor ao mesmo e aos ambientes de educacionais produzidos com ele, deixando-os mais autônomos, dinâmicos e adaptáveis (SILVEIRA; GOMES; VICARI, 2004).

Neste trabalho o Agente Pedagógico Animado é responsável pelas tarefas administrativas e pedagógicas que envolvem um ambiente de aprendizagem, ou seja, ele representará os Sistemas Gerenciadores de Aprendizagem (LMS), proposto em (LTSC, 2004). De um modo geral, o Agente Pedagógico Animado representa a interface para o gerenciamento de objetos de aprendizagem através de seus metadados, e a interação destes objetos com o aluno.

1.1 Motivação

O fundamento desta proposta partiu da possibilidade de se utilizar objetos inteligentes de aprendizagem de modo que eles sejam reaproveitados, levando em conta a sua interoperabilidade, dentro de outros ambientes educacionais e, ao mesmo tempo, agregar um Agente Pedagógico Animado no papel de um LMS.

Um fator decisivo para o desenvolvimento deste trabalho estava ligado à possibilidade de validar o *framework* de agentes desenvolvido em trabalho de mestrado de membros do grupo de pesquisa. A idéia de agregar tecnologia de agentes em Objetos de Aprendizagem como forma de obter ganhos de produtividade dentro do ambiente foi um estímulo para avançar dentro desta proposta.

Para que um objeto de aprendizagem seja reusado, ele deve ser compatível entre diferentes ambientes educacionais, ou seja, ele deve ser interoperável. A interoperabilidade é alcançada através da definição de padrões. Um objeto de aprendizagem deve ser modular o bastante para que possa ser enquadrado em contextos distintos. Ele deve ter alguma estrutura que o descreva para que possa ser descoberto por algum projetista de curso, essas estruturas são chamadas de metadados. Ainda, é necessário que os objetos de aprendizagem sejam armazenados em algum local onde possam ser acessados e descobertos. Esses locais são denominados Repositórios de objetos de aprendizagem. Os ambientes educacionais nos quais os objetos de aprendizagem são entregues aos estudantes

são denominados Sistemas Gerenciadores de Aprendizagem (LMS – *Learning Management Systems*) (LTSC, 2004).

Considerando que cursos relacionados tendem a ter materiais educacionais similares, não seria necessário o desenvolvimento de um material novo toda vez que se quisesse disponibilizar um novo curso, desde que fosse possível compartilhar os que já estão desenvolvidos.

A tecnologia de objetos de aprendizagem fundamenta-se na hipótese de que é possível criar pequenos "pedaços" de material instrucional e organizá-los de forma a permitir que eles possam ser reutilizados, promovendo economia de tempo e de custo na produção de cursos on-line (DOWNES, 2001).

1.2 Objetivos

O objetivo deste trabalho é propor avanços na integração das tecnologias de agentes e de objetos de aprendizagem, através do projeto de um Agente Pedagógico Animado que funcione como um Sistema de Gerenciamento de Aprendizagem (*Learning Management Systems* – LMS), ou seja, um Agente Pedagógico Animado que irá gerenciar todas as atividades relacionadas aos objetos de aprendizagem.

Para alcançarmos o objetivo geral do trabalho foram definidos os seguintes objetivos específicos listados no próximo tópico.

1.2.1 Objetivos Específicos

- Estudar os conceitos da área de Sistemas Multiagente, bem como as características, métodos e linguagens de comunicação e propostas de padronização (em especial os padrões FIPA);
- Estudar os conceitos e tecnologias de suporte da área de Objetos de Aprendizagem;
- Propor avanços na arquitetura de agentes desenvolvidas no grupo de pesquisa, para a modelagem e implementação desta proposta. Esta arquitetura deve levar em consideração os seguintes requisitos:
  - Quantos aos agentes:
    - ✓ Propor avanços na arquitetura interna de agentes que manipula informações de metadados dos objetos de aprendizagem;

- ✓ Modelar a comunicação entre os agentes e o LMS, de acordo com os padrões FIPA;
- Quanto aos Objetos de Aprendizagem:
  - ✓ Modelar protocolos de comunicação para a troca de informações entre os Objetos Inteligentes de Aprendizagem e o APA;

- Implementar um grupo de objetos inteligentes de aprendizagem baseado no *framework* utilizado;
- Implementar um LMS na forma de um APA;
- Avaliar as propostas efetuadas;

1.3 Contribuição esperada

Espera-se com a conclusão deste trabalho atingir uma convergência ainda maior entre a abordagem de objetos de aprendizagem e a arquitetura de agentes, de modo que os objetos de aprendizagem apresentados neste trabalho possam ser controlados pelo Agente Pedagógico Animado.

Com isso, esperamos modelar e implementar uma família de agentes representativa de algum tipo de objeto de aprendizagem, além do agente LMS, modelar o processo de comunicação entre agentes que compõem essa família. Propondo uma arquitetura de agente que implemente os fundamentos apresentados.

Espera-se dar importantes contribuições, refinando a eficácia da tecnologia de objetos inteligentes de aprendizagem para a implementação de projetos de Educação a Distância enfatizando o uso de paradigmas de resolução cooperativa de problemas através de arquiteturas multiagentes.

Do desenvolvimento deste trabalho, espera-se possibilitar a construção de material educacional que possa ser utilizado em um largo espectro de domínios acadêmicos, já que reusáveis, e que possuem a habilidade de se auto-adaptar, através do uso de agentes, sempre que mudanças forem requeridas.

1.4 Organização do Trabalho

O próximo Capítulo apresenta aspectos da área de Sistemas Multiagentes, bem como conceitos sobre agentes pedagógicos. Destacando características da comunicação entre agentes

O Capítulo 3 exemplifica sugestões de padronização da FIPA. São destacados a linguagem FIPA-ACL e perspectivas gerenciais de uma plataforma padrão FIPA. Também será apresentada a plataforma FIPA-OS, que virá a ser utilizado na implementação deste trabalho.

O capítulo 4 apresenta uma abordagem sobre os Objetos Inteligentes de Aprendizagem, bem como suas características gerais. Nesse capítulo também é ilustrada a idéia básica de um objeto de Aprendizagem sob um paradigma voltado para agentes.

O capítulo 5 apresenta uma proposta de arquitetura de agentes que pode ser utilizada na implementação de uma sociedade de agentes. É apresentado também um mapeamento dessa arquitetura para uma arquitetura real e um *framework* desenvolvido a fim de facilitar a implementação de agentes que seguem essa arquitetura.

No capítulo 6 é mostrada a aplicação de um ambiente de aprendizagem que foi remodelado e incrementado a partir do *framework* apresentado no capítulo anterior, considerando seus casos de usos.

No capítulo 7 são apresentadas as conclusões geradas por este trabalho, bem como suas considerações finais e resultados encontrados.

2 Sistemas Multiagentes

Existem vários conceitos que definem um sistema multiagente, porém como forma de exemplificar o entendimento desta sociedade de agentes pode-se destacar que é um sistema composto por um conjunto de agentes trabalhando no sentido de alcançar um objetivo comum.

A principal hipótese para o desenvolvimento de um sistema multiagente é que 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, 1998). É neste sentido que sistemas multiagente são empregados, já que muitos problemas têm características distribuídas necessitando de entidades independentes que trabalhem a fim de garantir a solução de uma questão e que juntas possam resolver o problema, ou seja, um sistema composto por mais de um agente trabalhando de forma cooperativa para alcançar um resultado comum. Isto é um SMA que é baseado na interação social e de seus indivíduos.

2.1 Agentes

Atualmente ainda não temos nenhuma definição da palavra “agente”, e sim características básicas que ele deve possuir para ser considerado como tal.

De acordo com Brenner (1998), um agente deve possuir certo grau de inteligência para executar as suas tarefas. É esse nível de inteligência que permite a um agente agir de maneira autônoma. Um agente não inteligente, ou sem inteligência, pode ser considerado como qualquer *software* tradicional. Além disso, para este autor, um agente deve interagir com o seu ambiente.

Para Russell e Norvig (2004) um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores.

Essa idéia de como os agentes percebem o ambiente através de sensores e reagem através dos atuadores é ilustrada na Figura 2.1 abaixo.

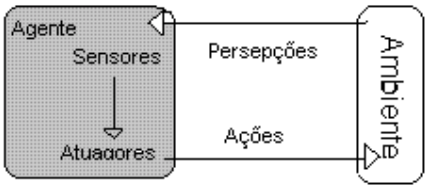

Figura 2.1 Agentes interagem com ambiente.

Além disso, podemos distinguir agentes racionais e agentes irracionais, dentro da Inteligência Artificial distribuída eles são denominados como agentes cognitivos e reativos, respectivamente. Os agentes reativos diferenciam-se dos agentes cognitivos pelo fato de não agirem deliberadamente e sim devido a eventos externos, ou seja, apenas reagem a situações, por isso o nome reativo. Por essa razão que um agente reativo é mais simples que um agente cognitivo e por si só não realiza modificações significativas no ambiente onde atua.

Vários atributos definem o comportamento de um agente, dependendo da forma de implementação do agente, estes atributos podem estar presentes em maior ou menor grau, mas sempre estão presentes. Entre eles destacam-se as seguintes propriedades:

- Adaptabilidade: Um agente pode mudar seu comportamento para melhor se adaptar a seu ambiente.
- Autonomia: Capacidade de o agente operar sem intervenção direta de um usuário, e também possuir um certo controle sobre suas ações.
- Interatividade: Interação entre agentes ou de agentes com os usuários através de uma linguagem de comunicação de agentes.
- Mobilidade: Um agente pode ser capaz de migrar de uma plataforma para outra.
- Pró-Atividade: Agentes devem ser capazes de planejar seus atos e não simplesmente reagir a estímulos externos.
- Racionalidade: Entre diversas opções de ações a disposição de um agente em um determinado momento, ele deve ser capaz de escolher a que julgar melhor.
- Reatividade: Agentes percebem os estímulos externos e respondem com uma ação respeitando seu comportamento.

- Sociabilidade: Define se um agente é cooperativo (benevolente, altruísta) ou competitivo (egoísta).

A maioria destes conceitos só tem validade quando analisados no contexto de sociedade de agentes, ou seja, um sistema multiagente. E também dependendo da aplicação envolvida algumas características são mais importantes que outras, ou seja, um agente não precisa obrigatoriamente ter todos esses atributos.

2.1.1 Agentes Pedagógicos

Os agentes podem ser considerados como pedagógicos quando fazem parte ou estão inseridos em ambientes de ensino-aprendizagem, baseados em uma sociedade de agentes. O termo agente pedagógico foi criado porque muitos sistemas desenvolvidos para a educação adotam um paradigma de agente.

De acordo com Pereira (1997), incorporar agentes dentro de software de educação é uma forma de atender o anseio dos aspectos pedagógicos nesses ambientes. Eles proporcionam vantagens além da inteligência ensinada em ambientes convencionais, pois garantem mais interações naturais que são a aproximação entre estudantes e sistemas.

Os agentes pedagógicos dividem-se em agentes *goal-driven* (guiados por objetivos) que são os tutores, mentores e assistentes e os agentes *utility-driven* (guiados pela sua utilidade no ambiente) que são os agentes que realizam tarefas auxiliares ligadas às atividades pedagógicas, executando tarefas para os estudantes ou para o professor como, por exemplo, o agendamento de encontro de grupos, lembrete de atividades a serem entregues (temas, exercícios), podendo atuar, tanto em ambientes de rede como ambientes *Web*. (GIRAFFA, 1999).

Os agentes pedagógicos têm algumas propriedades essenciais de agentes inteligentes como autonomia, habilidade social, incluindo interações e comunicações que se adaptam de acordo com o ambiente. Em acréscimo a essas características, eles também têm a capacidade de aprender, e na maioria dos casos eles podem ser representados por um personagem. São essencialmente cognitivos e de qualquer modo, podem ter características de agentes de reação, reagindo para mudanças de acordo com o ambientes onde eles são utilizados.

São responsáveis por acompanhar os estudantes na interação dele com o sistema educacional e, assim, são capazes de guiar estudantes em determinadas

situações, desta maneira os estudantes podem aprender de um modo mais eficiente, já que são atendidos com qualidade do ponto de vista pedagógico.

2.2 Métodos de comunicação entre agentes

As informações são transmitidas de um agente para outro através da troca de mensagens. A comunicação entre eles é muito importante, já que em uma sociedade de agentes onde cada um tem uma tarefa específica e o resultado final depende exclusivamente da junção de todas operações realizadas. Nesses casos é imprescindível uma forma para que possam transmitir suas informações e avisos.

Para Brenner (1998), os métodos de comunicação podem ser divididos em dois grupos: quadro de avisos e troca de mensagens.

2.2.1 Quadro de avisos

É também identificado como quadro negro, no qual existe uma área de trabalho comum onde os agentes podem trocas informações, dados e conhecimentos.

As ações de comunicação são iniciadas sempre que um agente escreve algo no quadro negro. Os agentes podem acessar o quadro negro a qualquer momento e verificar se tem alguma informação nova ou alterada. Um agente não lê todas as mensagens que estão no quadro de avisos, ele filtra e extrai as informações que afetam a sua área corrente de trabalho.

Os SMA podem ter vários quadros negros e estes podem ser administrados de forma a não autorizarem acessos de agentes não registrados.

Quando um sistema possui um grande número de agentes, a quantidade de informações escrita no quadro negro cresce exponencialmente e, portanto, a busca por informações relevantes se torna cada vez mais pesada (BRENNER, 1998). Como forma de solucionar isto, os quadros negros mais avançados têm sua área dividida em sub-áreas, cada qual privativa a um agente ou a um grupo deles. Nesse caso, um agente só precisa monitorar a região a ele associada.

Para Brenner (1998), a leitura das informações do quadro de avisos pode causar uma sobrecarga do sistema. Por isso, a importância dos sistemas baseados em troca de mensagens vem crescendo continuamente.

2.2.2 Troca de mensagens

A Figura 2.2 ilustra o conceito básico do sistema de troca de mensagens. Um agente, chamado emissor, envia uma mensagem a um outro agente, o receptor. Diferente do sistema de quadro negro, dessa forma as informações são passadas diretamente entre os agentes.

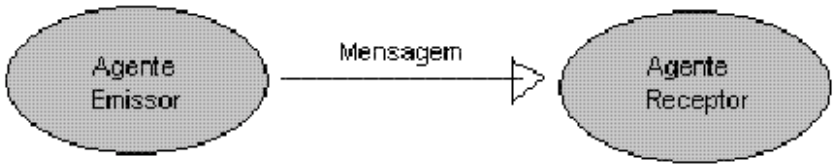

Figura 2.2 Troca de mensagens

Segundo Finin (1993) a troca de mensagens pode ser feita de três formas:

- Ponto-a-ponto: As mensagens são enviadas para um agente específico que deve ser conhecido pelo emissor. A vantagem dessa abordagem é que o agente emissor sempre sabe para onde uma mensagem está sendo enviada, permitindo controles de segurança.
- *Broadcast*: A mensagem é enviada para todos os agentes da sociedade. Nessa abordagem, um agente pode ser substituído por outro equivalente e o processamento do sistema será inalterado. A passagem de mensagens por broadcast não é segura, já que qualquer agente pode examinar o conteúdo de qualquer mensagem.
- *Multicast*: A sociedade de agente é dividida em grupos. Desta forma as mensagens podem ser enviadas somente aos agentes de determinados grupos.

A maioria dos SMAs do momento estão utilizando os sistemas baseados em troca de mensagens, pois estes proporcionam muito mais flexibilidade e dinamismo do que a arquitetura do quadro de avisos apresentada na sessão anterior.

2.3 Linguagens de Comunicação de Agentes

O mecanismo de troca de mensagens necessita da definição de um protocolo de comunicação que especifique 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 transporte de mensagens.

Logo, uma ACL determina um meio através do qual uma atitude (afirmação, requisição, consulta...) a respeito do conteúdo da troca de conhecimentos é comunicada (FININ, 1997).

Mayfield (1996) e Finin (1997) destacam que para uma linguagem de comunicação possa ser adequada à comunicação entre agentes, ela deve possuir as seguintes propriedades quanto a:

- **A 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 freqüentemente 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.

Uma das linguagens que procura implementar essas características é a KQML (*Knowledge Query and Manipulation Language*) que é uma linguagem e um protocolo para troca de informações e conhecimento entre agentes. Até o ano de 2000, quando foi lançada a primeira implementação da FIPA-ACL, ela era a única linguagem padronizada e com implementação disponível (GLUZ, 2002) e, portanto, dominava a área de comunicação entre agentes.

A KQML foi muito criticada na década de 1990. Segundo Labrou (1999), a maior parte das críticas era dirigida às imprecisões na semântica de sua gramática. Essas imprecisões, atribuídas também ao fato dela ser informal, mas principalmente ao fato da própria KQML ter sido desenvolvida sem as idéias de sistemas multiagente, teriam dificultado a construção deste tipo de sistemas de forma a serem compatíveis e interoperáveis (GLUZ, 2002).

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-OS, que ela passou a ter uma implementação disponível e a competir com a KQML.

No desenvolvimento deste trabalho será utilizada a linguagem FIPA-ACL, por ser a linguagem de comunicação de agentes sugerida pela FIPA. Tal linguagem será apresentada no próximo Capítulo.

3 FIPA e FIPA-OS

Neste capítulo será apresentado a *Foundation for Intelligent Physical Agents* (FIPA), fundação que cria padrões para o desenvolvimento de agentes e Sistemas Multiagentes, além do *Foundation for Intelligent Physical Agents Open Systems* – FIPA-OS, uma plataforma de desenvolvimento de Sistemas Multiagentes que segue totalmente aos padrões FIPA. Nas próximas secções será apresentada a linguagem FIPA-ACL, principal desenvolvimento da FIPA.

A FIPA é uma organização internacional sem fins lucrativos criada em dezembro de 1996 com o objetivo de estabelecer padrões e expandir o uso da tecnologia de agentes inteligentes, através de especificações para a interoperabilidade entre sistemas de agentes e aplicações baseadas em agentes. Várias empresas e universidades colaboram com a FIPA, e todos os resultados das atividades da fundação estão disponíveis para avaliação.

As atividades de padronização promovidas pela FIPA são centradas na criação e manutenção de especificações, trabalho gerenciado pelos Comitês Técnicos. As pesquisas que não contribuem necessariamente com a produção de especificações são feitas pelos Grupos de Trabalho. As tarefas auxiliares que são interesse de parcelas dos membros da FIPA são realizadas pelos Grupos de Interesses Especiais.

3.1 Especificações FIPA

São divididas em cinco áreas distintas, entre elas: Aplicações de Sistemas Multiagente, Arquiteturas Abstratas, Comunicação entre Agentes, Gerenciamento de Sistemas Multiagente e Transporte de Mensagens entre Agentes. A seguir será feita uma análise de cada uma dessas categorias.

3.1.1 Aplicações de Sistemas Multiagente

Nelas são encontrados exemplos de domínios onde podem ser utilizados agentes FIPA. Possui definições de ontologias e descrições de serviços para esses domínios. A Tabela 3.1 relaciona os documentos desta área.

Tabela 3.1 Especificações FIPA: Aplicações de Sistemas Multiagente

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

3.1.2 Arquitetura Abstrata para Sistemas Multiagente

A Arquitetura Abstrata define um nível abstrato como dois agentes se localizam e trocam mensagens. Um agente antes de enviar qualquer mensagem a outros agentes deve registrar-ser no sistema. Para que isso seja possível, deve existir um elevado grau de interoperabilidade entre estes sistemas.

Para aumentar o grau de interoperabilidade entre diferentes sistemas de agentes, a arquitetura FIPA especifica padrões para a implementação das características comuns a todos os sistemas de agentes, aumentando também o grau de abstração da arquitetura. A idéia básica é que cada arquitetura concreta baseada na arquitetura abstrata FIPA é livre para escolher com que linguagem de comunicação os agentes irão se comunicar ou qual será o protocolo de transporte de mensagens e, mesmo assim, os sistemas poderão interoperar porque foram baseados na mesma arquitetura.

Possui, também, uma especificação definindo como sistemas multiagente poderiam ser estruturados em domínios distintos e como poderiam ser estabelecidas políticas de manutenção. A *Tabela 3.2* relaciona os documentos que compõem esta área.

Tabela 3.2 Especificações FIPA: Arquitetura Abstrata

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

3.1.3 Comunicação entre Agentes

Os avanços mais significativos em termos de padronização ocorreram no tratamento das questões relativas à comunicação entre agentes. 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;

A Tabela 3.3 relaciona os documentos de acordo com a divisão mostrada acima.

Tabela 3.3 Especificações FIPA: Comunicação entre Agentes

Comunicação entre agentes	
SC00061	FIPA ACL Message Structure Specification
XC00086	FIPA Ontology Service Specification
Atos comunicativos	
SC00037	FIPA Communicative Act Library Specification
Protocolos de Interação	
SC00026	FIPA Request Interaction Protocol Specification
SC00027	FIPA Query Interaction Protocol Specification
SC00028	FIPA Request When Interaction Protocol Specification
SC00029	FIPA Contract Net Interaction Protocol Specification
SC00030	FIPA Iterated Contract Net Interaction Protocol Specification
XC00031	FIPA English Auction Interaction Protocol Specification
XC00032	FIPA Dutch Auction Interaction Protocol Specification
SC00033	FIPA Brokering Interaction Protocol Specification
SC00034	FIPA Recruiting Interaction Protocol Specification

Comunicação entre agentes	
SC00035	FIPA Subscribe Interaction Protocol Specification
SC00036	FIPA Propose Interaction Protocol Specification
Linguagens de Conteúdo	
SC00008	FIPA SL Content Language Specification
XC00009	FIPA CCL Content Language Specification
XC00010	FIPA KIF Content Language Specification
XC00011	FIPA RDF Content Language Specification

3.1.4 Gerenciamento de Agentes

Essa categoria 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 3.4 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.

Tabela 3.4 Especificações FIPA: Gerenciamento de Agentes

Identificador	Título
SC00023	FIPA Agent Management Specification

3.1.5 Transporte de Mensagens entre Agentes

A finalidade dessa categoria esta 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.

A *Tabela 3.5* relaciona os documentos de acordo com esta divisão.

Tabela 3.5 Especificações FIPA: Transporte de Mensagens entre Agentes

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
SC00084	FIPA Agent Message Transport Protocol for HTTP Specification

3.2 FIPA-ACL

A Linguagem de Comunicação de Agentes sugerida pela FIPA é a FIPA-ACL (FIPA – *Agent Communication Language*). Inicialmente apresentada em 1997, ela é baseada na linguagem ARCOL, que foi desenvolvida para ser a linguagem de comunicação do projeto ARTEMIS (SADEK, 1997). O projeto ARTEMIS foi patrocinado pela *France Telecom* e tinha por objetivo o desenvolvimento de uma plataforma de SMA que pudesse acessar as bases de conhecimento da própria *France Telecom*.

A semântica dessa linguagem foi totalmente formalizada sobre um modelo lógico-formal similar ao definido por Cohen & Levesque (COHEN, 1995). 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. Além disso, as implementações são livres para definirem novos parâmetros desde que seja utilizado o prefixo “x-“.

Os parâmetros nativos da FIPA-ACL são divididos em cinco categorias: tipo do ato comunicativo, participante da comunicação, conteúdo de mensagem,

descrição do conteúdo e controle de conversação. A lista completa dos parâmetros e suas respectivas categorias são mostradas na Tabela 3.6.

Tabela 3.6 Parâmetros das mensagens FIPA-ACL

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 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.

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

As mensagens FIPA-ACL carregam um ato comunicativo que representam o desejo de um agente sobre determinada informação carregada pela mensagem.

Os atos comunicativos das mensagens FIPA-ACL foram projetados para estarem de acordo e, dentro do possível, representar os atos da fala que foi estabelecido na Teoria dos Atos da Fala de Searle (1981). Eles possuem a denominação de atos comunicativos para deixar clara a vinculação com a comunicação entre agentes (computacionais) e não com a comunicação entre seres humanos.

A tabela abaixo mostra os atos comunicativos padrões da linguagem FIPA-ACL.

Tabela 3.7 Atos comunicativos em FIPA-ACL

Ato comunicativo	**Resumo**
Accept-Proposal	Informa a aceitação de uma proposta prévia para a execução de uma determinada ação. O conteúdo da mensagem deve conter a ação a ser feita e a condição aceita.
Agree	Informa a concordância em executar alguma ação, possivelmente no futuro. O conteúdo da mensagem deve conter a ação (futura) e a condição aceita.
Cancel	Informar para um determinado agente que ele não necessita mais executar a ação pedida anteriormente. No conteúdo deve estar expressa a ação que não é mais requerida.
Cfp	O ato *cfp* (*Call for Proposal*) solicita propostas para a execução de uma determinada ação. A mensagem deve conter qual a ação que deve ser feita e qual a pré-condição para esta ação.
Confirm	O emissor informa ao receptor que uma dada proposição é verdadeira, se o receptor estava (reconhecidamente) incerto disso. O conteúdo da mensagem é a proposição.
Disconfirm	O emissor informa ao receptor que uma dada proposição é falsa, se o receptor estava (reconhecidamente) certo de que ela era verdadeira. O conteúdo da mensagem é a proposição.
Failure	O emissor informa ao receptor que tentou fazer uma ação e que essa tentativa falhou. O conteúdo da mensagem é composto da ação que falhou e da razão da falha.
Inform	O emissor informa ao receptor que uma dada proposição é verdadeira. O conteúdo da mensagem é a própria proposição.
Inform-If	É um ato composto que serve para emissor informar ao receptor se uma dada proposição é verdadeira ou não. O conteúdo da mensagem é a própria proposição.

Ato comunicativo	Resumo
Inform-Ref	É um ato composto que serve para emissor informar ao receptor o objeto que corresponde a um dado descritor. O conteúdo da mensagem é uma expressão referencial, um descritor de objeto.
Not-Understood	O agente emissor informa ao agente receptor que não entendeu uma ação ou ato prévio do agente receptor. O conteúdo da mensagem é composto do ato ou ação não compreendida e de uma explicação do que não foi compreendido.
Propagate	Serve para que o agente emissor solicite a manipulação da mensagem encapsulada em anexo como se tivesse sido emitida diretamente por ele, mas que também busque outros agentes que se encaixam num descritor, também passado em anexo e reenvie a mensagem para os agentes selecionados pela busca. O conteúdo da mensagem é composto de dois elementos: um descritor dos outros agentes que deverão receber a mensagem sendo propagada e um ato comunicativo completo, contendo a mensagem encapsulada.
Propose	Serve para que agente emissor envie ao receptor uma proposta para efetuar alguma ação, dadas certas pré-condições. O conteúdo da mensagem é composto da descrição da ação sendo proposta e da pré-condição na execução dela.
Proxy	O agente emissor quer que o agente receptor busque outros agentes, que se encaixam na descrição passada em anexo, e envie a mensagem em anexo para esses agentes. O conteúdo da mensagem é composto de dois elementos: um descritor dos outros agentes que deverão receber a mensagem sendo passada por procuração e um ato comunicativo completo, contendo a mensagem encapsulada.
Query-If	Representa a ação de perguntar a um agente se uma determinada proposição é verdadeira ou não. O conteúdo da mensagem é a própria proposição.
Query-Ref	Representa a ação de perguntar a um agente qual o objeto que atende uma determinada expressão referencial. O conteúdo da mensagem é a própria expressão referencial (um descritor do objeto).
Refuse	Representa a ação de se recusar a executar uma dada ação e explicar a razão porque. O conteúdo da mensagem é composto da ação recusada e da explicação da recusa.
Reject-Proposal	Representa a ação de rejeitar a executar alguma ação durante uma negociação. O conteúdo da mensagem é composto da ação rejeitada e de explicação do porque para a rejeição.

Ato comunicativo	Resumo
Request	O agente emissor solicita ao receptor que ele execute alguma ação (possivelmente um outro ato comunicativo). O conteúdo da mensagem é uma ação a ser feita (expressão de ação).
Request-When	O agente emissor solicita ao receptor que ele execute alguma ação quando uma dada proposição for verdadeira. O conteúdo da mensagem é composto da ação a ser feita e da proposição.
Request-Whenever	O agente emissor solicita ao receptor que ele execute alguma ação assim que uma dada proposição for verdadeira e que a continue executando cada vez que ela se tornar verdadeira novamente. O conteúdo da mensagem é composto da ação a ser feita e da proposição.
Subscribe	Solicita a notificação do valor das atualizações no valor de uma dada referência. O conteúdo da mensagem é composto de uma expressão referencial (uma descrição do valor a ser notificado).

3.3 Plataforma FIPA-OS

As preocupações da FIPA giram em torno de definir e especificar os elementos necessários para a construção de SMA interoperáveis. Assim, nenhuma implementação é desenvolvida pela própria FIPA, essa tarefa é desempenhada por outras fundações ou empresas.

Atualmente temos várias implementações disponíveis das especificações FIPA. As plataformas de maior destaque são o JADE (BELLIFEMINE, 1999), o Zeus (NWANA, 1999) e o FIPA-OS (FIPA-OS, 2005), que será analisado nesta secção.

O FIPA-OS disponibiliza um conjunto de componentes para a implementação de agentes compatíveis com os padrões FIPA. De acordo com Laukkanen (2000) o principal objetivo da FIPA-OS é reduzir as barreiras na adoção da tecnologia FIPA através do suplemento de documentos e especificações técnicas com código fonte.

O FIPA-OS foi modelado para ser compatível com os padrões FIPA. Assim, existe um modelo de referência FIPA, que define a base da distribuição FIPA-OS: DF, AMS e MTS. A linguagem de comunicação de agentes é a FIPA-ACL. Este modelo é apresentado na seção seguinte.

3.3.1 Modelo de Gerenciamento de Agentes

Neste modelo é apresentado o ambiente onde o agente FIPA opera, bem como as atividades relacionadas ao mesmo. Nele se encontra um modelo lógico de referencia para criação, registro, localização, comunicação, migração e desativação de agentes na plataforma FIPA.

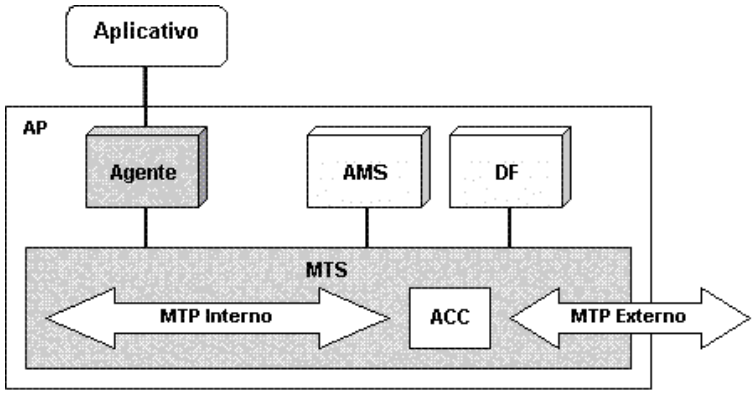

Figura 3.1 Modelo de Referência para Gerenciamento de Agentes FIPA

Um Agente é uma peça fundamental dentro de uma plataforma. É a combinação de uma ou mais capacidades de serviços e pode incluir acesso a aplicativos externos, usuários humanos e meios de comunicação. Um agente deve ter pelo menos um proprietário e pode suportar várias noções de identidade.

O Facilitador de Diretórios (DF) é um componente obrigatório da plataforma de agentes. Ele provê um serviço de páginas amarelas para os outros agentes. Os agentes podem registrar seus serviços com o DF ou requisitar ao DF que ele encontre serviços oferecidos pelos outros agentes. Múltiplos DF podem existir dentro de uma mesma plataforma de agentes e podem estar confederados. A interface do DF declara métodos de gerenciamento de registro de serviços.

O Sistema Gerenciador de Agentes (AMS) é um componente obrigatório da plataforma de agentes. O AMS exerce um controle superintendente de acesso e uso da plataforma de agentes. Só pode existir um AMS numa única plataforma de agentes. Ele apresenta um diretório de identificadores de agentes (AID - *Agent Identifier*) que contém endereços de transporte para agentes registrados com a plataforma. O AMS oferece páginas brancas para os outros agentes. Cada agente deve se registrar com o AMS de forma a receber uma AID válida.

O Serviço de Transporte de Mensagens (MTS) fornece o mecanismo de transporte de mensagens entre agentes de uma mesma plataforma ou de

plataformas diferentes. Todos os agentes FIPA têm acesso a no mínimo um MTS e somente mensagens endereçadas a agentes podem ser enviadas através do MTS. O MTS é disponibilizado pelo Canal de Comunicação de Agentes (ACC - *Agent Communication Channel*).

Uma Plataforma de Agentes (AP) provê a estrutura física no qual os agentes podem ser executados. Uma AP consiste na máquina, o sistema operacional, as aplicações de suporte aos agentes, os componentes de gerenciamento de agentes FIPA (DF, AMS e MTS) e os agentes.

Um Aplicativo descreve todo e qualquer conjunto de rotinas executáveis que não seja um agente e que possa ser acessível através dele. O desenho interno de cada um dos componentes descritos acima é tarefa do desenvolvedor da plataforma ou do SMA em questão e não é objeto de padronização da FIPA.

É interessante lembrar que estas especificações mostram o conjunto lógico para o desenvolvimento da plataforma de agentes. Assim, uma mesma plataforma pode estar residente em vários computadores.

Além de todos esse componentes obrigatórios definidos pela FIPA, o FIPA-OS adiciona suporte a:

- Diferentes tipos de *Agents Shells* para a implementação facilitada de agentes que possam se comunicar através da plataforma FIPA-OS;
- Suporte de múltiplas camadas para a comunicação de agentes;
- Configuração dinâmica da plataforma para suporte dos componentes permutáveis;
- Interfaces abstratas;
- Ferramenta de diagnóstico e visualização.

O FIPA-OS é totalmente construído em *Java*. Sua distribuição contém arquivos *class*, código fonte e documentação. Também são incluídos agentes de teste e aplicativos para inicialização e visualização da plataforma e dos agentes.

3.3.2 Componentes FIPA-OS

A arquitetura do FIPA-OS possui três tipos de componentes: obrigatórios, opcionais e permutáveis. Os componentes obrigatórios são necessários para a execução dos agentes. Já os componentes opcionais podem ou não ser utilizados. Os componentes permutáveis possuem mais de uma implementação, o que permite

a escolha da implementação que mais se adapte às necessidades do desenvolvedor.

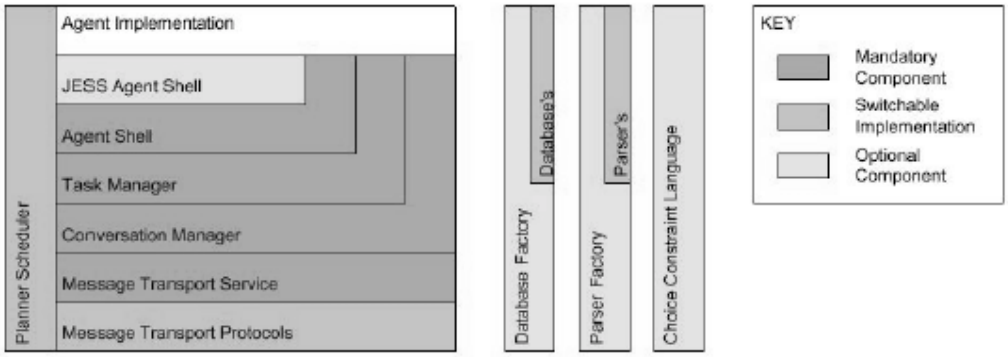

Figura 3.2 Componentes do FIPA-OS

O ***Agent Shell*** é um componente obrigatório que facilita a implementação de agentes FIPA-OS através de conjunto de classes bases extensíveis;

O ***Task Manager*** **(TM)** é um componente obrigatório que disponibiliza uma forma de dividir a funcionalidade de um agente em pequenas unidades de trabalhos conhecidos como *Task.* O objetivo é fazer com que uma *Task* seja uma peça de código que desempenhe alguma função e que opcionalmente retorne algum resultado. Além disso, ela deve ter a habilidade de enviar e receber mensagens e deve ter pouca ou nenhuma dependência com o agente em que está executando;

O ***Conversation Manager*** **(CM)** é um componente obrigatório que permite que seja mantido um histórico navegável das conversações entre os agentes, bem como mecanismos para o agrupamento de mensagens da mesma conversação. O CM certifica que o mesmo protocolo esta sendo utilizado pelos dois participantes da conversação;

O ***Message Transport Service*** **(MTS)** é um componente obrigatório que provê habilidades de envio e recebimento de mensagens entre os agentes. O FIPA-OS disponibiliza uma variedade de MTPs (*Message Transport Protocols*) para possibilitar que o MTS comunique utilizando vários protocolos.

O ***Message Transport Protocol*** **(MTP)** é um componente permutável de uso obrigatório que disponibiliza implementações de protocolos utilizados pelo MTS para o envio e recebimento de mensagens. O MTPs são divididos entre Internos, utilizados na comunicação entre agentes de uma mesma plataforma, e Externos, utilizados nas comunicações entre diferentes plataformas.

O ***Database Factory*** é um componente opcional que provê um mecanismo simples para interação com implementações de banco de dados através da definição de uma interface padrão a todos os agentes.

O ***Parser Factory*** é um componente opcional que permite que o conteúdo de uma mensagem seja expresso de forma a conter somente a sua informação semântica, sem preocupar o desenvolvedor com a real sintaxe da codificação. Esse trabalho está em progresso.

O ***Jess Agent Shell*** é um componente opcional e uma extensão do Agent Shell padrão e permite que sejam desenvolvidos agentes FIPA-OS que utilizem as vantagens oferecidas pelo sistema JESS (2004).

O ***Choice Constraint Language*** **(CCL)** é um componente opcional.

O ***Database*** é um componente permutável que disponibiliza implementações de interfaces utilizadas pelo componente *Database Factory* para a interação com implementações de Banco de Dados reais.O Componente ***Parser*** é um componente permutável que disponibiliza implementações de *parsers* utilizados pelo *Parser Factory;* Acompanham o FIPA os *parsers* para as linguagens SL, ACL, XML e RDF.

O ***Planner Scheduler*** é um componente permutável de uso opcional que provê mecanismos para o planejamento e cooperação de tarefas entre os agentes. O *Planner Scheduler* ainda não possui implementação disponível.

Algumas considerações importantes sobre o FIPA-OS são com relação a sua capacidade de dividir um agente em *Tasks*, as quais são gerenciadas pelo *Task Manager*. As execuções das *tasks* são baseadas em eventos, o que permite que mais de uma seja executada ao mesmo tempo, pois elas executam *threads* independentes. Além de poderem enviar e receber mensagens.

Outro fator importante está na presença do *Agent Shell.* A classe *FIPAOSAgent* facilita a implementação dos agentes através de sua extensão. Ela carrega automaticamente os componentes obrigatórios de um agente: o MTS, o *Task Manager* e o *Conversation manager*. O MTS permite através das classes *MessageSender* e *MessageReceiver*, o envio e o recebimento de mensagens.

O configurador do sistema FIPA-OS, o FIPA-OS *Configuration Tool,* é uma excelente ferramenta, pois para que o sistema funcione corretamente, uma série de variáveis devem ser configuradas de maneira correta. Se alguma dessas variáveis do sistema estiver mal configurada, todo o processo de instalação e uso do sistema fica prejudicado. Assim, o FIPA-OS tenta ser o mais amigável possível, para que a instalação e utilização não sejam uma barreira quanto ao uso do sistema.

4 Objetos Inteligentes de Aprendizagem

Atualmente tem sido crescente a utilização da informática em ambientes educacionais. As abordagens para esta utilização são bastante variadas, como por exemplo: sistemas que auxiliam o professor em tarefas diárias, tal como preparar avaliações; enciclopédias virtuais, nas quais os alunos podem fazer pesquisas escolares e aprimorar os seus conhecimentos; sistemas que auxiliam o professor no processo de ensino; e, até mesmo, sistemas de tutoria que pretendem prover alguma forma de aprendizagem autônoma, ou seja, sem a necessidade de um professor; entre outros.

Seguindo está linha de desenvolvimento, uma grande parcela de instituições está criando materiais educacionais para cursos *on-line.* Considerando que cursos relacionados tendem a apresentar materiais educacionais similares, não seria necessário o desenvolvimento de um material novo toda vez que se quisesse disponibilizar um novo curso, desde que fosse possível compartilhar os que já estão prontos. É com está finalidade que surgem os Objetos de Aprendizagem, uma abordagem que busca reduzir significativamente o tempo e o custo gastos com o desenvolvimento de materiais educacionais para diversos cursos. Um Objeto de Aprendizagem é caracterizado por ser uma entidade de material educacional reutilizável, ou seja, que pode ser reaproveitada em cursos diversos, diversas vezes.

Contudo, para ser reusado, um objeto de aprendizagem deve ser compatível entre diferentes ambientes educacionais, ou seja, ele deve ser interoperável. A interoperabilidade é alcançada através da definição de padrões. Um objeto de aprendizagem deve ser modular o bastante para que possa ser enquadrado em contextos distintos. Os ambientes educacionais nos quais os objetos de aprendizagem são entregues aos estudantes são denominados Sistemas Gerenciadores de Aprendizagem (LMS – *Learning Management Systems*).

Para se alcançar as características citadas acima, muitas organizações e grupos de pesquisa vêm trabalhando exaustivamente. A maioria dos esforços concentra-se justamente no desenvolvimento de tecnologias que permitam uma maior reusabilidade e interoperabilidade aos objetos de aprendizagem.

Na realidade não existe um conceito único do que seja um objeto de aprendizagem. O que existe são muitas explicações, onde cada pesquisador define o conceito que lhe é conveniente. Uma discussão bastante interessante é feita por Sosteric & Hesemeier (2002). Estes autores buscam em tal trabalho encontrar um conceito adequado e bem delimitado para objetos de aprendizagem.

O primeiro conceito apresentado por esses autores corresponde ao definido pelo LTSC IEEE (*Learning Technology Standard Comitee IEEE*):

Um objeto de aprendizagem é qualquer entidade, digital ou não, que possa ser usada, reusada, ou referenciada durante aprendizagem suportada por tecnologia (LTSC, 2004).

Sosteric & Hesemeier (2002) afirmam que todos os trabalho feitos na área de objetos de aprendizagem se referem a objetos digitais. Portanto, não existe a necessidade de incluir no conceito de objetos de aprendizagem o universo de objetos reais. Essa visão é bastante pragmática. Mesmo que uma flor, em sua manifestação física, possa ser utilizada em diversas disciplinas. No momento de se construir um repositório de objetos de aprendizagem, certamente não será armazenado o corpo físico da flor, mas uma foto desta, ou seja, um arquivo digital.

Outra questão levantada por Sosteric e Hesemeier (2002) é a necessidade de se ter associado aos objetos de aprendizagem um contexto. Essa necessidade pode ser ilustrada com as figuras que se encontram em livros, elas por si próprias não podem ser consideradas objetos de aprendizagem. Isso porque, a menos que esteja enquadrado em algum contexto, que neste caso é o título das figuras ou alguma explicação acerca delas, não produzem experiências de aprendizagem.

Em um outro artigo são citadas também algumas discussões sobre as definições que tentam associar os objetos de aprendizagem com os objetos de orientação a objetos. É o caso de Quinn (2000), Robson (1999) e Wiley (2000a), onde seguem as definições abaixo:

O modelo de objetos de aprendizagem é caracterizado pela crença de que podemos criar pedaços independentes de conteúdo educacionais que proveja experiências educacionais para algum propósito educacional. Projetados sobre o modelo de programação orientada a objetos, este modelo assume que esses pedaços são autocontidos, que podem conter referências a outros objetos, e que podem ser combinados ou seqüenciados para formar interações educacionais longas. Esses pedaços de conteúdo educacionais podem ser de qualquer tipo -

passivo, ativo - e pode ser de qualquer formato ou tipo de mídia. Um objeto de aprendizagem não é necessariamente um arquivo digital (QUINN, 2000).

Recursos de aprendizagem são objetos em um modelo de orientação a objetos. Eles possuem métodos e propriedades. Métodos típicos incluem renderização métodos de acompanhamento. Propriedades típicas incluem conteúdo e relacionamentos a outros recursos (ROBSON, 1999).

Um objeto de aprendizagem é qualquer recurso digital que possa ser usado para suportar a aprendizagem... A principal idéia dos objetos de aprendizagem é quebrar o conteúdo educacional em pequenos pedaços que possam ser reusados em vários ambientes de aprendizagem, no espírito da programação orientada a objetos (WILEY, 2000a).

Mohan e Greer (2003), citando Sosteric & Hesemeier (2002), afirmam que essa associação é inclusive contra-produtiva e causa muita confusão em se entender claramente o que é um objeto de aprendizagem. Isso porque a noção fundamental da orientação a objetos é um objeto que consiste em dados e em métodos, os quais definem os seus estado e comportamento, respectivamente. Os objetos de aprendizagem em suas manifestações correntes tal como arquivos HTML, arquivos de vídeo, apresentações *PowerPoint* são simplesmente coleções de bits que são interpretados de alguma forma por um *software* de visualização.

Essa mesma idéia também foi levantada por Friesen (2001), o qual minimiza a relação entre objetos e objetos de aprendizagem ao aspecto da reusabilidade. No entanto, não necessitamos da teoria de orientação a objetos para chamar um objeto de aprendizagem de reusável. Portanto, a associação entre objetos de aprendizagem e objetos deve ser esquecida (FRIESEN, 2001) (MOHAN, 2003) (SOSTERIC, 2002).

A conclusão que se tem sobre os objetos de aprendizagem a partir de todas essas definições é voltada para a idéia "entidade educacional reutilizável". O pragmatismo adotado por Sosteric & Hesemeier (2002) nos parece bastante plausível ao descartar do conceito os recursos educacionais físicos. Outra boa característica da definição desses autores é a inclusão de alguma estrutura que indique um contexto para o objeto de aprendizagem, que é o que acontece quando se tem metadados associados a eles. A falha dessa definição está ao considerar um objeto de aprendizagem como um arquivo digital apenas. Isso é até uma contradição dela própria, pois, se um objeto de aprendizagem contém via associação alguma informação de contexto, ele já não é mais formado por apenas um arquivo digital.

4.1 Propriedades

O alvo de toda a tecnologia de objetos de aprendizagem é permitir a reusabilidade do material instrucional, ou seja, buscar o reaproveitamento do material instrucional em cursos diversos, diversas vezes. Para obter-se essa característica é necessário que um objeto de aprendizagem apresente certas propriedades. A literatura provê um grande número delas, entre as mais citadas encontra-se: a modularidade; a interoperabilidade; e a capacidade de ser descoberto (FRIESEN, 2001).

4.1.1 Capacidade de ser descoberto

Para que o projetista de curso possa identificar quais os objetos de aprendizagem que mais se adaptam às suas necessidades, de alguma forma, um objeto de aprendizagem deve ter a capacidade de ser descoberto. Essa característica é dada pela utilização de metadados (SINGH, 2000) (ADL, 2004) (LONGMIRE, 2000), que são estruturas de informação que descrevem a informação carregada e a experiência gerada pelo objeto.

A palavra metadados significa “dados sobre dados”. Logo, os metadados são utilizados para descrever e categorizar os objetos de aprendizagem. A estrutura de um objeto de aprendizagem está ligada aos metadados que o descrevem. Alguns autores dizem que os metadados são um componente constituinte dos objetos de aprendizagem. Longmire (2000) diz que existem dois componentes obrigatórios em um objeto de aprendizagem: o objeto de conteúdo e seus metadados. Já Robson (1999) mantém que, embora os metadados possam ser associados com os recursos, eles não necessariamente estão ligados a eles.

4.1.2 Modularidade

A questão sobre a modularidade leva a pensar sobre o tamanho ideal de um objeto de aprendizagem, ou seja, a sua granularidade ideal (WILEY, 2000b). Como vimos anteriormente, muitos conceitos permitem que o programa de um curso inteiro seja abstraído como um objeto de aprendizagem. No entanto, quanto maior um objeto de aprendizagem, menor a sua chance de ser *reusado.*

Nesse caso, a situação ideal seria fazer os objetos de aprendizagem bastante pequenos. O problema advindo daí é que os objetos de aprendizagem estão sempre associados a informações de metadados. Logo, transformar cada

figura ou cada parágrafo em um objeto de aprendizagem pode tornar o trabalho de gerenciar esses objetos e seus metadados bastante custoso. A decisão sobre a granularidade de um objeto de aprendizagem deve ser feita de acordo com um balanceamento entre os possíveis benefícios do reuso e do gasto com a catalogação do objeto (WILEY, 2000b).

4.1.3 Interoperabilidade

A interoperabilidade é outra propriedade importante para os objetos de aprendizagem. Ser *interoperável* significa, para objetos de aprendizagem, operar em uma grande variedade de sistemas educacionais. Na realidade, o que torna as coisas *interoperáveis* é a utilização de padrões. A grande maioria das pesquisas que vem sendo levadas a cabo na área de objetos de aprendizagem trata justamente da definição de padrões que permitam a interoperabilidade entre objetos de aprendizagem e sistemas computacionais de educação. Nestas pesquisas, os padrões de metadados têm lugar de bastante destaque.

A utilização de protocolos abertos e de ambientes bastante difundidos contribui bastante com a interoperabilidade dos objetos de aprendizagem. Neste sentido, a construção de objetos de aprendizagem que rodem sobre plataforma WWW é encorajada. Por exemplo, uma simulação em *flash* pode ser visualizada por qualquer *browser* que tenha o *plug-in* apropriado, o qual a grande maioria dos *browsers* disponíveis hoje possui. A utilização de uma interface *Web* por si só já é bastante boa, no sentido da interoperabilidade.

4.2 Metadados e Padrões

Como foi apresentado anteriormente os metadados representam “dados sobre dados”, e são utilizados para descrever e categorizar os objetos de aprendizagem. Eles funcionam como um catálogo de biblioteca, que contém informações sobre os livros que a biblioteca possui.

Várias instituições vêm trabalhando na definição de padrões de metadados. Entre essas podemos citar o IMS *Project*, a *Dublin Core*, a *CanCore*, o LTSC IEEE, entre outras. Além de permitir a um objeto de aprendizagem ser descoberto, padrão de metadados são bastante importantes também para a interoperabilidade.

Neste trabalho optou por utilizar o Padrão LOM desenvolvido pelo LTSC da IEEE.

4.2.1 Padrão LOM da IEEE

O padrão *Learning Object Metadata* (LTSC, 2004) do *Learning Standard Technology Comitee* da IEEE define uma hierarquia de elementos de dados para descrição de metadados de objetos de aprendizagem. Para o LOM IEEE, um objeto de aprendizagem é definido como qualquer entidade digital ou não digital que pode ser utilizada para aprendizagem, educação ou treinamento.

Este padrão é definido em um conjunto de quatro documentos. O documento *1484.12.1 IEEE Standard for Learning Object Metadata* especifica o esquema conceitual de uma instância de metadados para um objeto de aprendizagem. O documento *1484.12.2 Standard for ISO/IEC 11404 binding for Learning Object Metadata data model* provê uma representação compatível com a notação 11404 para o LOM. O documento *1484.12.3 Standard for Learning Technology-Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata* provê uma representação XML para o LOM. Por fim, o documento *1484.12.4 Standard for Resource Description Framework (RDF) binding for Learning Object Metadata data model* provê uma representação RDF para o LOM.

No documento 1484.12.1, que define o esquema conceitual do padrão, os elementos de dados são divididos em 9 grandes categorias, das quais:

- ***General*:** que agrupa informações gerais que descrevem o objeto de aprendizagem como um todo;
- ***Lifecycle*:** que agrupa características relacionadas com o histórico e com o corrente estado do objeto de aprendizagem, assim como informações sobre aqueles que afetaram o desenvolvimento do objeto;
- ***Meta-metadata*:** que contém informações sobre os metadados, ao invés de informações sobre o objeto de aprendizagem que os metadados descrevem;
- ***Technical*:** que descreve os requisitos e características técnicas do objeto de aprendizagem;
- ***Educational*:** que agrupa informações sobre características educacionais e pedagógicas do objeto de aprendizagem;
- ***Rights*:** que contém informações sobre direitos de propriedade intelectual e condições de uso do objeto de aprendizagem;

- ***Relation*****:** na qual podem ser expressas características que definem o relacionamento entre o objeto de aprendizagem e os outros objetos.
- ***Annotation*****:** que provê comentários sobre o uso educacional do objeto de aprendizagem e informações sobre as entidades que fizeram tais comentários;
- ***Classification*****:** que descreve a classificação do objeto de aprendizagem em relação a um sistema de classificação particular.

Cada elemento de dado é explicado em detalhes pelo LOM. Informações como nome, tamanho, semântica, exemplos e tipo de dado são explicitados para cada um deles. Alguns elementos possuem um conjunto pré-definido de valores denominados vocabulários.

4.3 Sistemas Gerenciadores de Aprendizagem

Um Sistema Gerenciador de Aprendizagem (LMS) pode ser visto como um sistema o qual é responsável pelas tarefas administrativas que envolvem um ambiente de aprendizagem.

Segundo o LTSC IEEE (LTSC, 2004), um sistema gerenciador de aprendizagem é um sistema computacional que pode incluir as capacidades de registrar estudantes, controlar e guiar o processo de aprendizagem, analisar e reportar o desempenho dos alunos, planejar a apresentação de recursos de aprendizagem, planejar e rastrear os alunos.

Um aspecto interessante dos conceitos definidos pelo LTSC IEEE para esta área é a divisão que ele faz entre LMS e RTS (*Runtime Services*). Para o LTSC IEEE um RTS é um sistema que controla a execução e a entrega de recursos de aprendizagem. A relação entre LMS e RTS é a de que um LMS pode ter acoplado um RTS, ou seja, um LMS pode ser capaz de lançar e entregar recursos de aprendizagem.

Um exemplo de LMS é o WebCT (2005). O WebCT permite a criação e o gerenciamento de cursos *online*. Ele permite que o professor disponibilize conteúdos didáticos, gerencie o acesso, aplique testes de avaliação e se comunique com os alunos através de ferramentas como bate-papo e fórum. O WebCT suporta diversos formatos de conteúdos como: HTML, *Word*, *Power Point*, PDF, *Flash*, Sons, Imagens, Filmes.

Alguns LMSs necessitam trocar informações com os seus recursos de aprendizagem. O LTSC da IEEE desenvolveu um modelo de dados cujo objetivo é padronizar as informações que são passadas entre um LMS e objeto de conteúdo

4.3.1 O Padrão DMCOC

Este padrão é gerenciado pelo grupo de trabalho de número 11 (*Workgroup 11*) do LTSC IEEE e descreve um modelo de dados para a troca de informações entre um objeto de conteúdo e um RTS, geralmente acoplado a um LMS. Sua estrutura está projetada para suportar ambientes clientes-servidores, nos quais um sistema educacional utilizando um LMS entrega conteúdo digital, um objeto de conteúdo, aos estudantes. Um objeto de conteúdo é uma coleção de conteúdo digital que é apresentado ao aluno através de um sistema de tecnologia educacional, um RTS.

Este padrão está definido em dois documentos. O primeiro deles, o *1484.11.1 Standard for Learning Technology - Data Model for Content Object Communication* especifica o modelo de dados em si. Este modelo de dados suporta a representação de informações sobre alunos e suas preferências, interações, objetivos, saídas, informações de *status*, parâmetros de tempo e desempenho, no sentido de avaliação. Ele não especifica os meios através do qual a comunicação se dará entre estas duas entidades nem como elas devem reagir em resposta ao recebimento de dados na forma especificada.

O documento *1484.11.2 Standard for Learning Technology - ECMAScript Application Programming Interface for Content to Runtime Services Communication* descreve uma API (*Programming Application Interface*) ECMASScript para a comunicação entre o objeto de conteúdo e o LMS. Esta API descreve o meio pelo qual as informações que compõem o modelo de dados definido no documento 1484.11.1 devem ser trocadas entre as duas entidades.

4.4 Objetos Inteligentes de Aprendizagem

No contexto deste trabalho assumimos que um Objeto Inteligente de Aprendizagem (ILO) é um agente capaz de desempenhar o papel de um objeto de aprendizagem (GOMES, 2004) (SILVEIRA, 2004). Da mesma forma, um ILO pode ser considerado como um objeto de aprendizagem que possui como base um agente. As duas visões estão corretas.

Operacionalmente um ILO é um agente que pode gerar experiências de aprendizagem no mesmo sentido que os objetos de aprendizagem o fazem, ou seja, visando a reusabilidade.

A reusabilidade, no modelo de objetos de aprendizagem, é considerada como um produto de outras três características: a modularidade, a interoperabilidade e a capacidade de ser descoberto. Estas por sua vez são alcançadas através da utilização de certas tecnologias desenvolvidas especialmente para este fim.

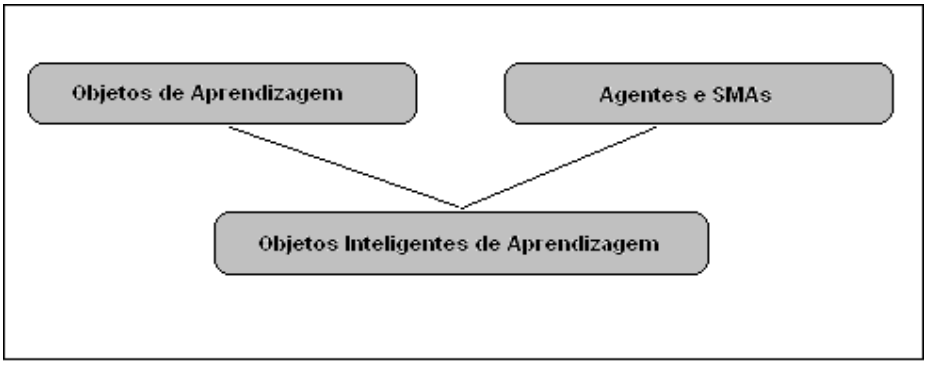

Figura 4.1 Bases tecnológicas dos Objetos Inteligentes de Aprendizagem

Assim, a base tecnológica que fundamenta este trabalho constitui-se em uma integração de tecnologias desenvolvidas para Objetos de Aprendizagem, para Agentes e para Sistemas Multiagentes. A figura abaixo apresenta esta integração.

O que justifica a utilização de Objetos de Aprendizagem em agentes são as inúmeras potencialidades que eles podem oferecer. Entre elas se destacam os métodos de comunicação, um agente é capaz de se comunicar através de troca de mensagens utilizando uma linguagem de comunicação de alto nível denominada Linguagem de Comunicação de Agentes (ACL). No modelo de objetos de aprendizagem mais completo, o SCORM (ADL, 2004), a comunicação é feita através de passagem de parâmetros e chamadas de métodos, no espírito da orientação a objetos. Isso resulta em uma comunicação muito estática, onde todas as possibilidades devem ser previstas em tempo de projeto. O uso de uma ACL extrapola esta estaticidade e dá uma dinâmica maior ao processo. Isso porque as ACL são baseadas em teorias capazes de dar mais semântica à comunicação. Além disso, o conteúdo das mensagens pode ser representado através de uma Linguagem de Conteúdo (CL), as quais são fortemente baseadas em formalismos lógicos. O resultado da comunicação através da união de CL e de ACL é

potencialmente melhor do que a comunicação através da abordagem de orientação a objetos, como os modelos de objetos de aprendizagem atuais fazem.

Outra característica interessante é a capacidade de aprendizagem que os agentes possuem. Um objeto de aprendizagem dotado desta capacidade pode adquirir novos conhecimentos e comportamentos no decorrer de sua existência através da interação com outros alunos e até mesmo com outros objetos de aprendizagem. Com essa idéia, é possível que o objeto de aprendizagem evolua, ele não é mais estático como nos modelos atuais. As possibilidades de aprendizagem são enormes, como por exemplo: adquirir novos materiais educacionais que podem auxiliar o aluno e complementar a sua tarefa; adquirir informações sobre os alunos, como as suas preferências e estilos cognitivos, para poder se adaptar a estas e até mesmo aprender como se adaptar a elas; mudar o seu conteúdo educacional no sentido de se adaptar ao estudante; entre outras possibilidades. Existem muitos trabalhos relacionados à implementação de aprendizagem em agentes e que podem ser utilizados.

Trabalhos enfocando mecanismos e métodos de coordenação e cooperação entre agentes podem dar à sociedade de ILOs a capacidade de se auto-organizar com vistas a disponibilizar experiências de aprendizagem mais ricas. Em conjunto com a capacidade de comunicação, a utilização de mecanismos de coordenação e cooperação possibilita a emergência de comportamentos complexos entre os ILOs e de experiências educacionais mais completas.

Alguns tipos de agentes deliberam e fazem planos baseados em conjuntos de estados mentais. Esse tipo de agentes, chamados Agentes BDI (*Belief, Desire and Intention*) (BRADSHAW, 1997), podem ser bastante úteis para a modelagem de comportamentos complexos. Um ILO concebido através de arquiteturas BDI pode implementar objetos de aprendizagem bastante avançados.

Algumas outras características típicas de agentes também são muito interessantes para a utilização em objetos de aprendizagem. A autonomia possibilita a um ILO a capacidade de atuar baseado no seu próprio conhecimento e comportamento, sem a necessidade de intervenção externa. A pró-atividade assegura que um ILO sempre atuará de forma a satisfazer os seus objetivos. As sociabilidade e benevolência referem-se às habilidades de ser social e de ser cooperativo com relação aos outros ILOs do ambiente.

Analisando todas essas características podemos destacar que as potencialidades do uso de “agentes objetos de aprendizagem”, os ILOs, são muito

grandes. Isso porque a flexibilidade e a dinamicidade que se pode alcançar com eles é maior do que a que se pode alcançar através do uso dos objetos de aprendizagem atuais. Em conseqüência, ambientes de aprendizagem baseados neles podem ser mais flexíveis e mais dinâmicos, também.

5 Arquitetura utilizada

No desenvolvimento deste trabalho foi empregada uma arquitetura interna de agentes, previamente definida (GOMES, 2005), como exemplo de estrutura para ser utilizada na construção dos agentes propostos. Na seqüência deste Capítulo será apresentada esta arquitetura, bem como o *framework* que instancia esta arquitetura de modo a facilitar a implementação de agentes compatíveis com ela.

Essa arquitetura possui uma descrição de quais são e como são os processos internos que possibilitam a interação do agente com o seu ambiente. Ela envolve três componentes básicos.

Existe um componente de sensoriamento que é responsável por perceber as modificações no ambiente como, por exemplo, o recebimento de mensagens e eventos em uma interface gráfica. Um componente de raciocínio que é responsável pela definição de qual comportamento o agente irá adotar em relação ao evento percebido. E um outro componente de atuação que é responsável por executar as ações do agente no ambiente. A Figura 5.1 abaixo ilustra esses três componentes.

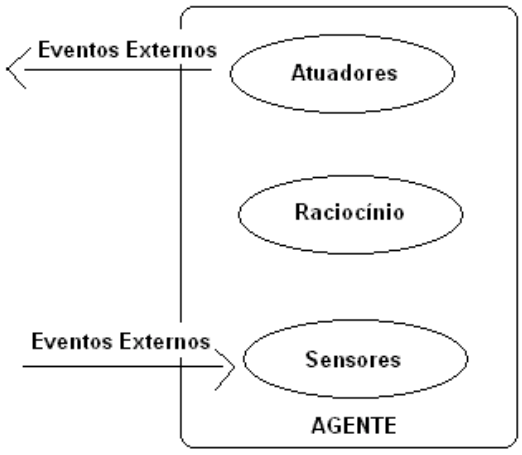

Figura 5.1 Componentes da arquitetura proposta

A arquitetura utilizada se caracteriza como uma arquitetura híbrida, pois contempla aspectos cognitivos e reativos. Esses aspectos foram modelados de

acordo com uma arquitetura em camadas, onde cada um dos componentes relacionados acima corresponde a uma das camadas do processamento.

A reatividade do sistema é dada devido ao fato de que o ciclo de agente é ativado quando o componente de sensoriamento percebe algum evento no ambiente FIPA ou no ambiente externo. A partir desta percepção, o componente de raciocínio é invocado e define o comportamento que o agente deve adotar. Se o comportamento a ser adotado precisa da execução de uma ação no ambiente, então o componente de atuação é ativado.

Os aspectos de cognição se refletem no conhecimento que o agente possui e que utiliza durante a sua existência. Este conhecimento é útil para tarefas como a identificação dos eventos, a definição do comportamento a ser adotado e a codificação das ações a serem executadas.

5.1 Camadas da Arquitetura

5.1.1 Primeira camada: sensores

Esta camada é responsável pela percepção do agente, implementando um dos seus aspectos reativos. Dessa forma, o agente reage aos eventos que percebe no ambiente.

Existem dois tipos de percepções de eventos: eventos de mensagens e eventos de interface. Os eventos de mensagem correspondem ao recebimento de mensagens enviadas pelos outros agentes presentes no ambiente. Nesta circunstância, o agente deve processar a mensagem e identificar qual a informação que a mensagem carrega. Efetuada esta identificação, o componente de raciocínio é invocado para definir o comportamento que será adotado mediante o evento percebido.

Esse componente de sensoriamento de mensagens está diretamente conectado ao ambiente FIPA no qual o agente está executando. As mensagens dos outros agentes necessariamente provêm deste ambiente.

O segundo tipo de evento previsto corresponde a eventos de interface. Um agente pode possuir uma interface, gráfica ou de qualquer outra natureza, onde agentes externos podem atuar. Esta atuação deve ser capturada pelo agente. Neste caso, as ações são identificadas e passadas ao componente de raciocínio, que define qual comportamento o agente deve adotar.

O componente de sensoriamento de interface está conectado ao ambiente do agente externo. Assim, se a interface do agente for uma interface gráfica WWW,

por exemplo, uma aplicação JSP, o componente de sensoriamento de interface terá que capturar as ações do agente externo através desta interface JSP.

5.1.2 Segunda camada: raciocínio

A segunda camada é responsável por definir o comportamento que o agente irá adotar em relação aos eventos percebidos pela primeira camada. Ela é a camada de ligação entre os componentes de percepção e os componentes de atuação.

Os eventos identificados pelo componente de sensoriamento passam por um conjunto de regras de comportamento. O resultado da aplicação destas regras pode gerar a necessidade de atuação no ambiente, o que é feito pelo componente de atuação, ou a modificação do estado interno do agente, atualização das regras e/ou do conhecimento do agente.

5.1.3 Terceira camada: atuadores

A terceira camada é responsável pela ação do agente no ambiente. Nela temos a execução de dois tipos de eventos: eventos de mensagens e eventos de interface. Os eventos de mensagem correspondem ao envio de mensagens aos outros agentes presentes no ambiente. Este tipo de evento é gerado a partir da necessidade do agente de se comunicar com os outros agentes. Tal necessidade é sempre gerada pela camada de raciocínio. Nesta circunstância, o agente codifica a mensagem requerida e a envia ao agente determinado.

O componente de atuação de mensagens está diretamente conectado ao ambiente FIPA no qual o agente está executando. As mensagens enviadas aos outros agentes necessariamente devem passar por este ambiente

O segundo tipo de evento previsto corresponde a eventos de interface. Se o agente precisar atuar na interface que liga o agente a um agente externo, caso essa possibilidade esteja prevista, tal ação deve ser executada. Neste caso, o comportamento adotado pelo agente é codificado de forma que a ação seja repercutida na dita interface, possibilitando que o agente externo seja capaz de perceber tal execução.

5.3 O *framework* utilizado na remodelagem do Ambiente de Aprendizagem

Este *framework* foi desenvolvido durante o trabalho de mestrado de Gomes (2005), baseado na arquitetura interna de agentes apresentada anteriormente. Este

framework foi implementado a partir de um conjunto de classes *Java*, com a finalidade de facilitar a construção de agentes que respeitam os requisitos definidos. A partir dele é que o Ambiente de Aprendizagem proposto neste trabalho foi remodelado.

5.3.1 Tratamento das conversações

O relacionamento entre os agentes que se envolvem em um diálogo, tal como os definidos no *framework*, são caracterizados por um modelo cliente-servidor. O agente que inicia um diálogo (requisita um serviço) é o cliente e o agente que responde ao diálogo (disponibiliza o serviço) é o servidor. Por essa razão é que existe um par de *tasks* para cada um dos diálogos. Uma delas deve ser usada pelo agente servidor e a outra pelo agente cliente.

O comportamento de cada um dos dois tipos de *tasks* é padronizado, porque o fluxo de mensagens é o mesmo entre todos os diálogos, já que eles utilizam sempre o mesmo protocolo, o *FIPA-Request*. Essa padronização gera uma classe abstrata para cada um dos tipos. Todas as *tasks* que tratam diálogos são, portanto, extensões de alguma destas classes.

Neste *framework* existe um relacionamento entre a classe abstrata que deve ser utilizada para a implementação da parte servidora de um diálogo e de uma das *tasks* que a estendem com este intuito, no caso a classe que implementa o diálogo *get-metadata*. Assim, são necessários apenas os métodos construtores, *startTask*() e *processa*().

Na classe *HandleTask*, os métodos *simpleResult*(), *composedResult*(), *simpleDone*() e *failure()* preparam o conteúdo das mensagens a serem enviadas. Elas codificam os resultados para a linguagem FIPA-SL, processo que é feito com o auxílio das classes que implementam a interface *Concept*. O método *simpleResult*() deve ser utilizado quando o resultado do diálogo é um único conceito. O método *composedResult*() deve ser utilizado quando o resultado é composto por um conjunto de conceitos. Por fim, o método *simpleDone*() deve ser utilizado quando não é necessário enviar resultados, mas apenas informar que a tarefa requisitada acabou com sucesso. Já o *failure*() monta o conteúdo da mensagem que indica a recusa ou falha no processamento da ação.

O método *startTask*() é chamado sempre que uma *task* é inicializada. Os métodos *sendAgree*(), *sendRefuse*(), *sendFailure*() e *sendInform*() devem ser

utilizados no envio de mensagens com os atos *Agree*, *Refuse*, *Failure* e *Inform*. Estes métodos utilizam como entrada o conteúdo codificado pelos métodos *simpleResult*(), *composedResult*(), *simpleDone*() e *failure*().

O método *processa*() é abstrato, ou seja, neste método deve ser inserido o comportamento da *Task* em si. Portanto, aqui entra grande parte do raciocínio da *task* em questão.

Existe um relacionamento entre a classe abstrata que deve ser utilizada para a implementação da parte cliente de um diálogo e de uma das classes que a estendem com este intuito, neste caso a classe que implementa o diálogo *get-metadata*.

Na classe *ReqTask*, o método *sendRequest*() é responsável por montar a mensagem inicial do diálogo. O argumento deste método corresponde a um objeto de uma classe que implementa a interface *Action*, que deve ser implementada por todas as ações previstas na ontologia deste *framework*. Ou seja, o argumento deste método é a ação a ser requisitada.

Os métodos *handleRefuse(), handleAgree(), handleFailure*() e *handleInform*() são métodos abstratos. Eles são chamados automaticamente pelo FIPA-OS quando da chegada de mensagens com os atos comunicativos *Refuse, Agree, Failure* e *Inform*, respectivamente.

Um método bastante útil disponível na superclasse *Task* do FIPA-OS é o método *done*(). Este método sinaliza a finalização de uma *task* e permite que seja feita a comunicação entre a *task* e a classe que a invocou. Todo método que possa ser um estado final em uma *task* deve chamar o método *done*(). Toda classe que invoque uma *task*, por sua vez, deve implementar um método *doneX*(), onde X é o nome da *task* em questão. Este método será chamado pelo FIPA-OS tão logo a *task* chame o seu método *done*().

Abaixo é apresentado o fluxo de troca de mensagens e chamadas de métodos para o tratamento de um diálogo.

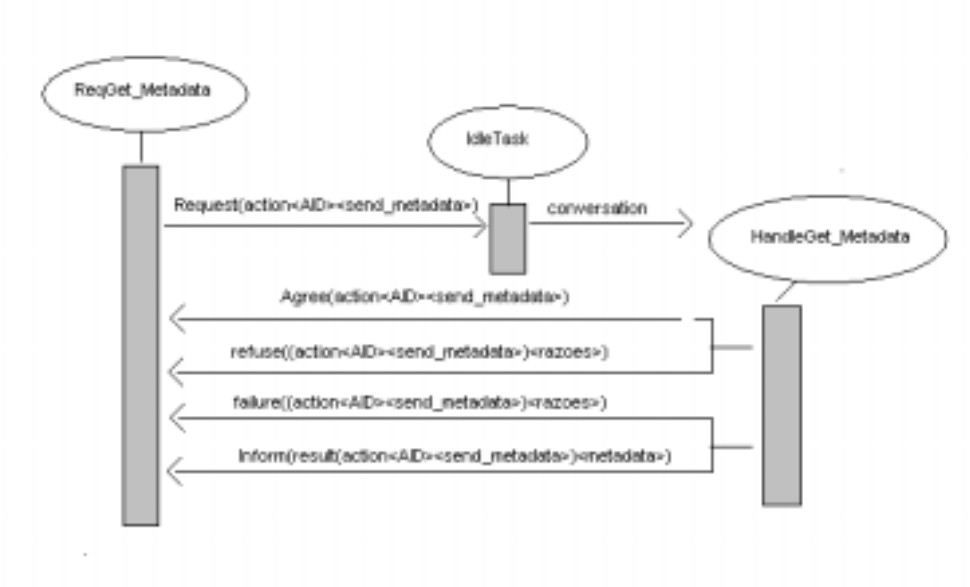

Figura 5.2 Relacionamento dinâmico entre as *tasks* cliente-servidora de um diálogo

Esta figura apresenta a seqüência de mensagens que começa em uma *task* de requisição de serviço. Esta *task* é uma sub-classe da classe *ReqTask*. Depois de iniciada, o seu método *sendRequest*(). Este método prepara a mensagem codificando a ação para o formato FIPA-SL. Depois de ter o seu conteúdo codificado, a mensagem é enviada. Uma vez que ela ainda não está ligada a nenhuma conversação, a *IdleTask* do agente servidor recebe a mensagem. Ela a identifica e invoca uma *task* capaz tratá-la, ou seja, ela invoca a classe cliente do diálogo em questão. A partir deste momento, todas as mensagens que forem recebidas ou enviadas, e que se relacionarem com esta conversação, serão automaticamente endereçadas às duas *Tasks* (a cliente e a servidor).

No seu método *startTask*(), a *task* servidor decide por aceitar ou não a requisição do serviço. Caso decida aceitar, o método *sendAgree*() é chamado para enviar a mensagem de aceite. Este método também chama o método processa(), que é responsável por executar a tarefa em si. Caso a tarefa tenha sido executada com sucesso, uma mensagem de informação é enviada através do método *sendIform*(). O conteúdo desta mensagem é codificado para FIPA-SL através de um dos métodos *simpleResult(), composedResult()* ou *simpleDone*(). Caso tenha ocorrido alguma falha no processamento da ação, o método *sendFailure*() é chamado para enviar a mensagem de falha. Tal método utiliza o método *failure*() para codificar a mensagem para FIPA-SL. Caso decida recusar a solicitação, o

mesmo método *failure*() é chamado para codificar o conteúdo da mensagem de recusa e *sendRefuse*() é chamado para enviá-la.

A chegada de uma mensagem de retorno, na *task* cliente, automaticamente chama um método capaz de tratá-la, conforme o ato comunicativo que ela carrega.

5.3.2 Tratamento das Ontologias

Um dos requisitos estabelecidos por este *framework* é a representação do conteúdo das mensagens através da linguagem FIPA-SL. Dessa forma, todos os elementos definidos na ontologia devem possuir também uma forma de representação FIPA-SL. Como forma de facilitar esta codificação, cada elemento da ontologia é responsável por produzir a sua própria representação em FIPA-SL. Esta responsabilidade foi inserida através da necessidade de que todos os elementos da ontologia fossem implementadores de uma interface que contém um método *toSL*(). Para cada tipo de elemento, ação, conceito ou predicado, existe uma interface base específica.

Dois conceitos são considerados chaves neste *framework*: o conceito *metadata* e o conceito *dataModel*. O primeiro corresponde às informações de metadados de um ILO e o segundo as informações acerca de um estudante. Estes dois conceitos tiveram a definição de uma interface especial para cada um deles. A definição destas interfaces permite que a implementação real de tais conceitos possa ser feita de acordo com a necessidade do projetista. Assim, se o projetista quiser que as informações de metadados de um ILO sejam armazenadas em uma base relacional, ele pode fazê-lo, desde que implemente a interface requerida.

5.3.3 As classes ILOs

Os ILOs podem ser divididos em classes de acordo com as suas possibilidades comunicativas. Tendo em vista esta divisão, é necessária uma classe *ILOAgent* e uma classe *IdleTask* para cada classe de ILO.

Como a única responsabilidade de um ILO de classe 1 é responder a solicitações de envio dos seus metadados, a classe *ILOIdleTaskC1* apenas identifica mensagens relacionadas a este tipo de diálogo. Quando uma mensagem assim chega, a *IdleTask* invoca uma *task HandleGet_Metadata*, responsável por implementar este serviço.

A classe ILOAgentC1 é a superclasse de ILOs de classe 1, ela possui um campo do tipo *IMetadata*, que corresponde à interface que deve ser implementada por elementos que desejem representar os metadados de um ILO. A indicação de qual implementação utilizar é feita no construtor da classe *ILOAgentC1*. Esta mesma característica ocorre com a *IdleTask,* outra *IdleTask* qualquer pode ser passada no construtor da ILOAgentC1 de forma que o projetista pode definir a sua própria implementação.

Na classe *ILOAgentC1* os métodos *registerAMS()* e *registerDF()* são responsáveis por registrar o agente no AMS e no DF da plataforma FIPA. O método *shutdown()* faz a finalização do agente. O métodos *getMetadata*() e *setMetadata*() são utilizados para acesso ao campo de informação de metadados. E, por fim, o método *interfaceEvent*() é responsável por fazer a ligação entre a interface externa do agente e o agente propriamente dito. Este método é abstrato e deve ser implementado de acordo com o tipo de interface externa que o agente terá. Caso o agente não possua uma interface externa, basta que este método seja implementado como vazio.

Um ILO de classe 2 é na verdade uma extensão de um ILO de classe 1, mas com a possibilidade de responder a solicitações sobre as informações do aluno. Assim, a classe *ILOAgentC2* estende a classe *ILOAgentC1*. É adicionado um campo *dataModel* que referencia um implementador da interface *IDataModel.* A possibilidade de o projetista escolher a maneira que mais lhe convier para manipular as informações acerca do estudante no ILO é mantida através deste mecanismo, basta que a interface seja seguida.

A *ILOIdleTaskC2, IdleTask* padrão para um ILO de classe 2, é uma extensão da *ILOIdleTaskC1*. É adicionada a funcionalidade de reconhecer a ação *send-learner*, que representa a solicitação das informações sobre o aluno.

Um ILO de classe 3 é uma extensão de um ILO de classe 2. A diferença aqui reside na possibilidade deste ILO acessar serviços de outros agentes. Assim, não é necessária a implementação de uma classe base específica para este tipo de ILO. Para implementá-lo basta que seja estendida a classe *ILOAgentC2* e que sejam utilizados as *tasks* clientes definidas para cada um dos diálogos.

5.3.4 Agente LMS

Na parte que corresponde a um Agente LMS são implementadas apenas as classes que desempenham os serviços básicos que este agente deve disponibilizar. Assim, está disponível a implementação de uma *IdleTask*, a *LMSIdleTask*, que identifica as ações relativas aos diálogos deste agente, e as classes que implementam a parte servidora dos diálogos. Para implementar um agente LMS mais sofisticado, basta que estas classes sejam estendidas.

6 Remodelagem do Ambiente de Aprendizagem

O ambiente de aprendizagem que será descrito neste capítulo foi remodelado com o intuito de realizar uma adaptação para que se enquadrasse nas especificações do *framework* apresentado no capítulo anterior.

O Ambiente de Aprendizagem utilizado foi desenvolvido em trabalhos anteriores (LUCAS; WILGES; SILVEIRA, 2005), ele é constituído de um sistema de auxílio para alunos de séries iniciais na aprendizagem das propriedades matemáticas básicas da multiplicação e adição. Esse ambiente é composto por dois agentes: um Agente Pedagógico Animado (APA) e um Agente Calculadora.

Nesse ambiente encontra-se um Agente Pedagógico Animado (APA), que no presente trabalho foi remodelado a fim de apresentar as características gerais de um agente *Learning Management Systems* - LMS (LTSC, 2004), o qual será responsável pelas tarefas administrativas e pedagógicas que envolvem um ambiente de aprendizagem. Ele será a interface para o gerenciamento de objetos de aprendizagem e seus metadados, e a interação destes objetos com o aluno. Tal agente foi desenvolvido utilizando a biblioteca *Swing*, a qual engloba classes e interfaces capazes de dar o aspecto de personagem animado ao agente pedagógico, onde o mesmo emprega expressões corporais realizadas através da animação de imagens e também exibe mensagens em uma caixa de diálogo.

O Agente Calculadora é o responsável pela interface gráfica da calculadora onde o aluno realizará os cálculos, e também pela comunicação dos resultados ao APA, o qual realiza a avaliação do que foi realizado pelo aluno. Com a finalidade de enquadrar o Agente Calculadora nas especificações do *framework* apresentado no capítulo anterior, fez-se necessário realizar uma adaptação no referido agente. Para que se encaixasse nestas especificações, ele foi remodelado como um ILO do *framework*, gerando experiências de aprendizagem do mesmo modo que um objeto de aprendizagem.

Tanto o Agente Pedagógico quanto o Agente Calculadora são compatíveis com plataforma FIPA-OS que foi desenvolvida na linguagem *Java*, portanto, a metodologia deste trabalho também baseou-se na sintaxe desta linguagem.

6.1 Remodelagem do APA

Nos códigos fontes do APA existem quatro classes: a *Apa* responsável por chamar o personagem no método *main*(); a *ApaGUI* responsável por sua interface gráfica; a classe Animador responsável pelos movimentos do personagem; a *Dialogo* que compõe a caixa de dialogo do personagem animado e mais uma classe que contém todas as constantes utilizadas pelo *Apa*. Tal agente possui três *tasks*: *ResultReasoning*, *InstructionReasoning* e *EvaluationReasoning*. A *ResultReasoning* é acionada a partir de uma mensagem enviada pela *task SendResult* pertencente a classe *CalcAgent* da Calculadora, sendo aquela responsável por fazer o raciocínio para analisar quais propriedades podem ser demonstradas, assim como instanciar outra *task*, a *InstructionReasoning*. Esta é acionada pela *ResultReasoining*, ela apresenta dois tipos de ações. A primeira corresponde a analisar o que foi feito pelo aluno (a partir dos parâmetros enviados pela *task ResultReasoning*) e imprimir na caixa de diálogo uma instrução ao aluno, especificando quais propriedades matemáticas podem ser provadas a partir do cálculo inicial. E a segunda ação corresponde a partir dos parâmetros enviados pela *task EvaluationReasoning*, imprimir na caixa de diálogo uma instrução ao aluno, dizendo quais foram as propriedades provadas corretamente, caso não forem, uma razão para o erro é especificada. Por fim, a *task EvaluationReasoning* apenas avalia se alguma propriedade foi provada. A Figura 6.1 apresenta um diagrama de funções, o qual ilustra os papéis padrões das *tasks* descritas acima.

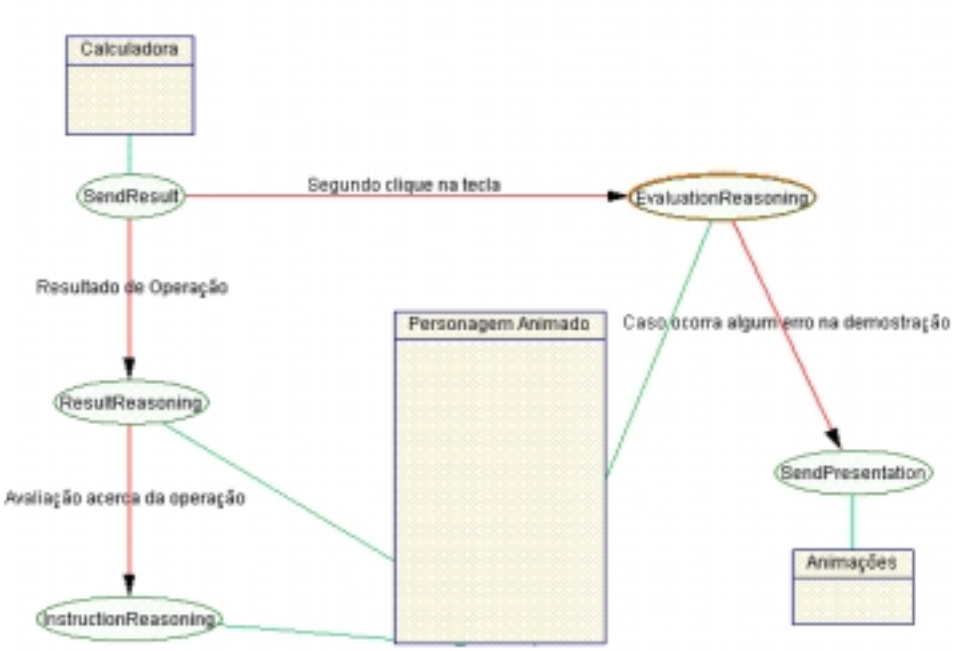

Figura 6.1 Diagrama de funções

Para transformar o Agente Pedagógico em um LMS foi necessário estender sua classe principal, ou seja, a classe *Apa* estendeu a classe *LMSAgent* do *framework*. Nesse caso, foi necessário adicionar duas *tasks* responsáveis por iniciar o servidor de diálogos: *get-learner-lms* e *put-learner-lms* e uma responsável por iniciar o cliente*: put-learner-ilo*. Após, foram adicionadas as *tasks* responsáveis pelo registro no AMS e DF da plataforma FIPA-OS. Com isso o agente pedagógico pode executar e operar com um LMS, oferecendo todos os serviços básicos de um agente desse tipo como, por exemplo, ações relativas aos diálogos deste agente.

6.2 Remodelagem da Agente Calculadora

O Agente Calculadora é composto pelas classes *CalcAgent* e *CalcAgentGUI*, onde esta é a responsável pela interface gráfica da calculadora. Ela possui apenas uma *task*, a *SendResult.* Esta *task* é acionada no momento em que o aluno pressiona a tecla referente a igualdade. A ação dele corresponde a enviar uma mensagem para o agente *Apa,* na mensagem esta contida o cálculo matemático realizado pelo aluno.

Neste caso a classe *CalcAgent* foi aprimorada, de modo que acabou por estender a classe *ILOAgentC1* do *framework*. Foram também acrescentadas a *task*

cliente do dialogo: *put-learner-lms* e a *task* servidora do dialogo: *put-learner-ilo*. A partir daí, a Calculadora passou a representar as características gerais de um ILO.

6.3 Implementação das animações

Na proposta deste trabalho implementou-se um conjunto de animações utilizando a tecnologia *flash*, já que está poderia garantir uma portabilidade em entre diferentes sistemas operacionais; O objetivo da inserção das animações no ambiente de aprendizagem, na forma de um ILO, foi de reunir informações necessárias para o facilitar o processo de ensino-aprendizagem. No sentido de ensinar a demonstrar uma determinada propriedade matemática, no caso em que um aluno errasse a demonstração. O APA invocaria um ILO que conteria as animações em *flash*.

As animações em *flash* tiveram uma adaptação semelhante a que ocorreu com o Agente Calculadora. Posteriormente as mesmas foram acrescidas de funções especificas de agentes. A partir daí elas foram adaptadas e inseridas dentro de uma arquitetura interna de agentes, mais especificamente o *framework* apresentado no capítulo 5. Para alcançar este resultado foi necessário estender a classe *ILOAgentC1* do *framework*. Por fim, foram adicionadas as mesmas *tasks* descritas na adaptação do Agente Calculadora, ou seja, passou se a ter dos ILO no ambiente de aprendizagem proposto.

6.4 Casos de usos do Ambiente de Aprendizagem

Com a finalidade de fazer a especificação do projeto, utilizou-se a ferramenta *AgentTool* (AGENTTOOL, 2005), a qual é baseada na *Unified Modeling Language* (UML), que é um modelo de linguagem. Com a ferramenta foi possível identificar de forma mais compreensiva os casos de usos deste ambiente de aprendizagem.

6.4.1 Primeiro caso de uso

O caso de uso que descreve o princípio da aplicação ocorre quando o aluno abre a aplicação através do *AgenteLoader* do FIPA-OS, o personagem animado exibe uma mensagem de boas vindas ao aluno e envia uma mensagem para o Agente Calculadora descongelar o teclado para que o aluno digite a primeira operação. A Figura 6.2 apresenta este caso de uso do sistema.

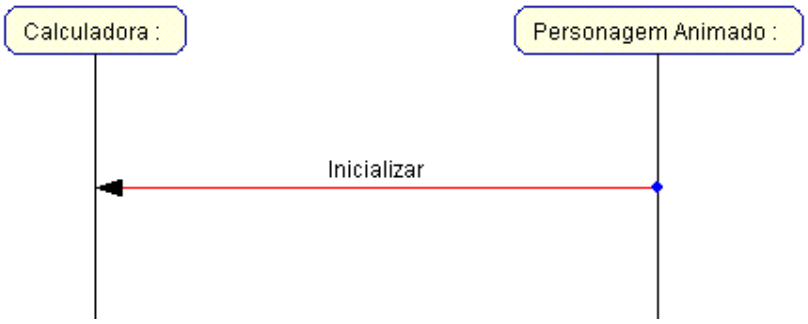

Figura 6.2 Inicio da Aplicação

A Figura 6.3 ilustra como se desenvolve este primeiro caso de uso diretamente no Ambiente de Aprendizagem.

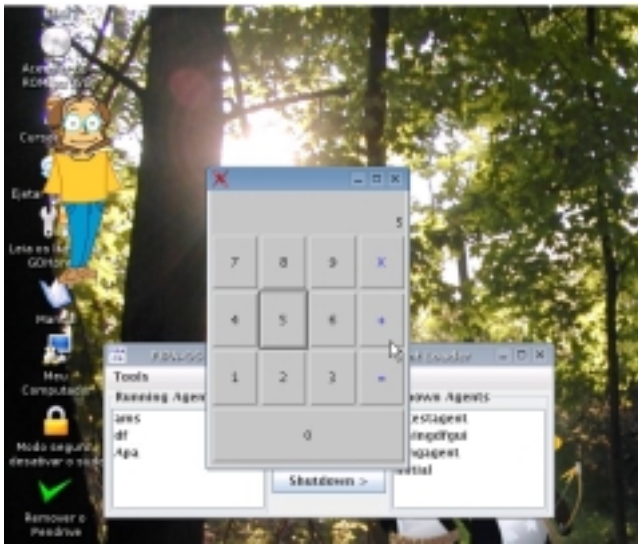

Figura 6.3 Inicio da aplicação no Ambiente de Aprendizagem

6.4.2 Segundo caso de uso

Este uso de caso corresponde a primeira operação matemática realizada, ele é iniciado a partir do momento em que o Agente Pedagógico Animado envia uma mensagem ao Agente Calculadora solicitando que o teclado presente na interface gráfica da calculadora seja descongelado, para que o aluno possa realizar a primeira operação. Após a realização da operação, o Agente Calculadora envia os resultados para o APA, que por sua vez envia uma requisição pedindo para que o Agente Calculadora trave o teclado novamente. Por fim, o APA faz uma avaliação acerca da operação realizada pelo aluno. A Figura 6.4 ilustra esse caso de uso.

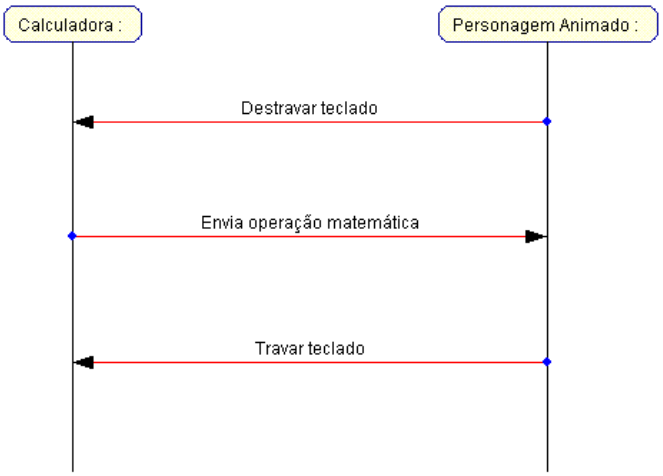

Figura 6.4 Realização da primeira operação matemática

A Figura 6.5 mostra o desenvolvimento deste segundo caso de uso realizado diretamente no Ambiente de Aprendizagem.

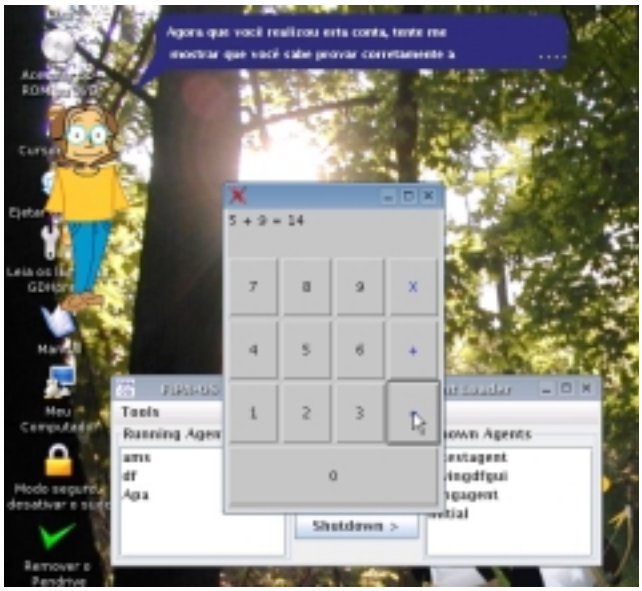

Figura 6.5 Realização da primeira operação no Ambiente de Aprendizagem

6.4.3 Terceiro caso de uso

Este uso de caso começa quando o Agente Pedagógico Animado envia uma mensagem ao Agente Calculadora para que ele descongele a calculadora, a fim de

que o aluno possa entrar com os dados da segunda operação matemática. Feito isto, o Agente Calculadora envia a operação ao agente animado, o qual avalia se as propriedades foram aplicadas corretamente e imprimi uma instrução na caixa de dialogo dizendo quais foram as propriedades matemáticas provadas corretamente, caso não forem provadas, é dada uma especificação para o erro. A Figura 6.6 ilustra esse caso de uso.

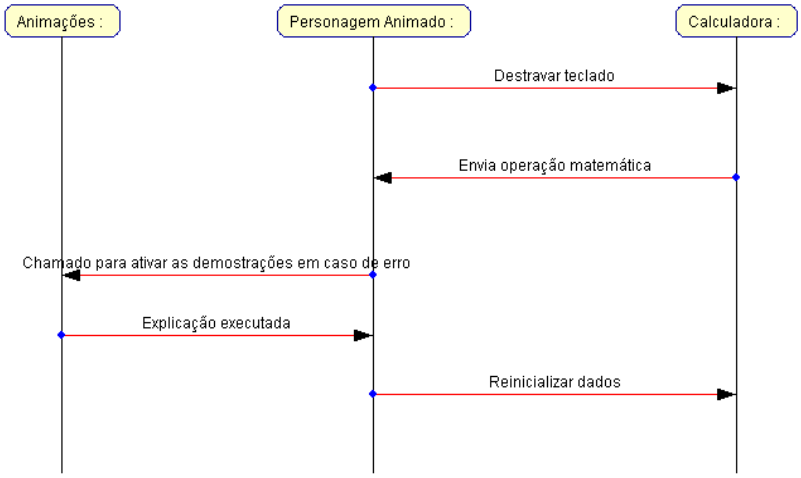

Figura 6.6 Realização da segunda operação matemática

A Figura 6.7 mostra o desenvolvimento deste terceiro caso de uso realizado diretamente no Ambiente de Aprendizagem.

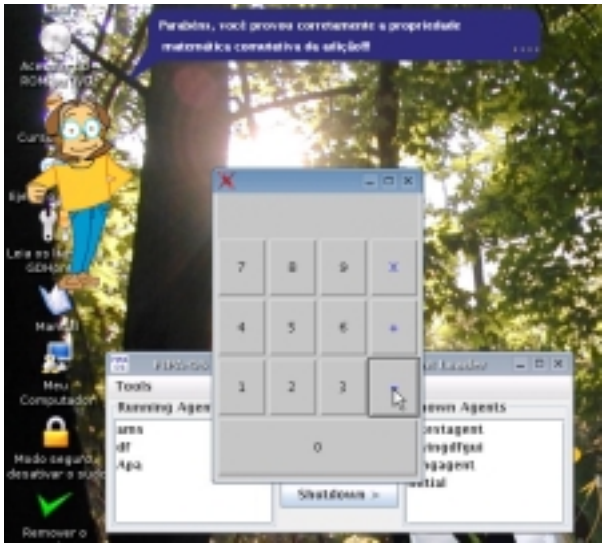

Figura 6.7 Realização da segunda operação

6.5.4 Quarto caso de uso

Outro caso de uso deste sistema envolve a apresentação das animações que ocorrem sempre depois de ter sido realizada a segunda operação e em caso de erro na demonstração de alguma propriedade. Neste caso o Agente Pedagógico Animado envia uma mensagem para o Agente que contém as animações, para que ele apresente a forma correta de demonstrar uma determinada propriedade. Este caso de uso é ilustrado na Figura 6.8.

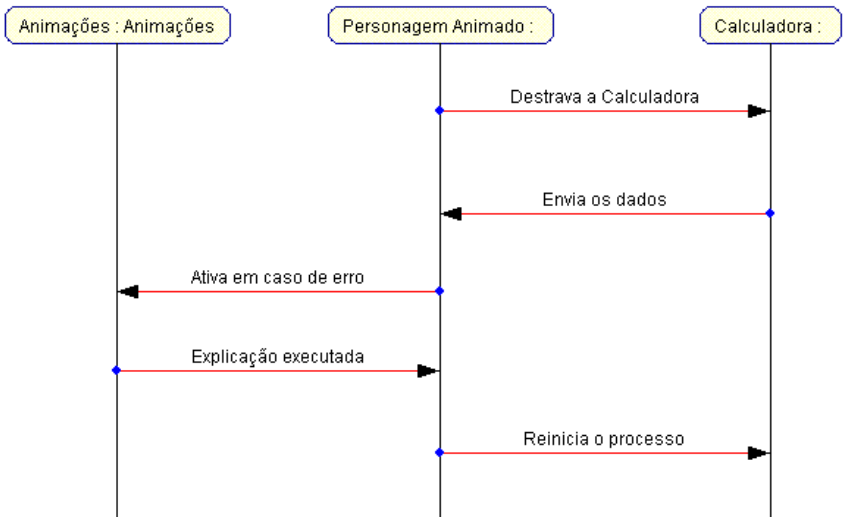

Figura 6.8 Apresentação das Animações

A Figura 6.9 mostra o desenvolvimento do quarto caso de uso realizado diretamente no Ambiente de Aprendizagem.

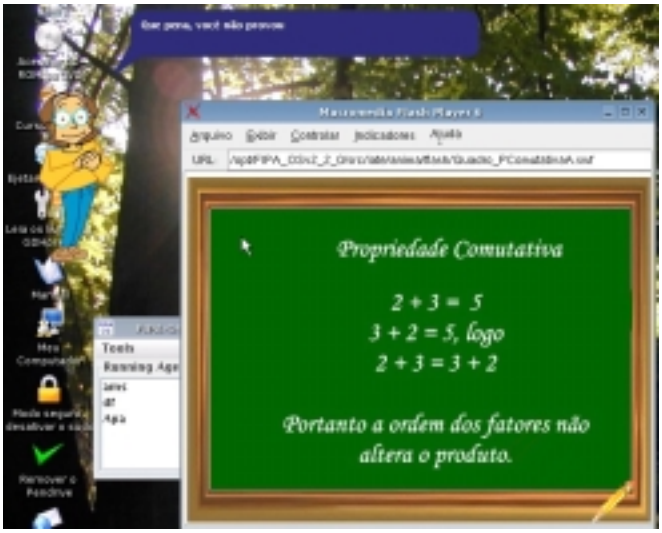

Figura 6.9 Apresentação das Animações no Ambiente de Aprendizagem

6.5.5 Quinto caso de uso

O último caso de uso deste sistema busca refletir como ocorre o reinicio do processo, ele ocorre logo após o APA fazer uma avaliação acerca dos resultados da segunda operação, e enviar uma mensagem ao aluno a respeito dos resultados obtidos. Para que o processo seja reinicializado, o Agente Pedagógico Animado envia uma mensagem ao Agente Calculadora para que ele inicialize todos os seus dados. A Figura 6.10 apresenta esse uso de caso.

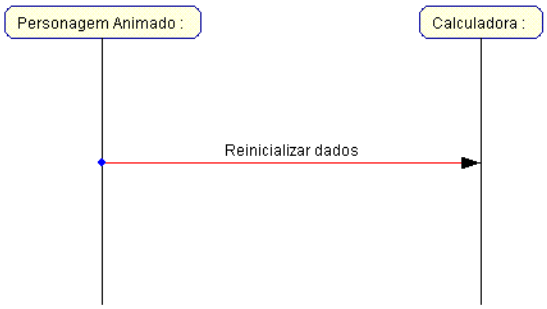

Figura 6.10 Reinicialização dos dados

A Figura 6.11 apresenta o desenvolvimento do quinto caso de uso realizado diretamente no Ambiente de Aprendizagem.

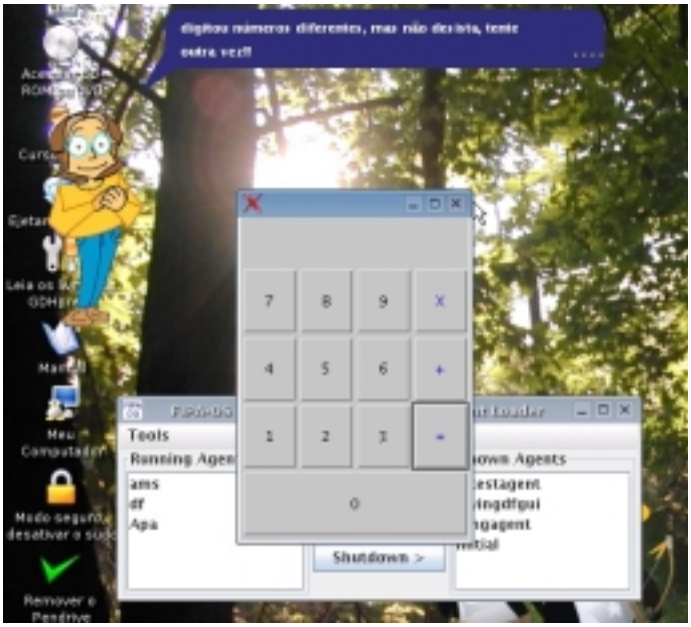

Figura 6.11 Reinicialização dos dados no Ambiente de Aprendizagem

7 Conclusão

Neste trabalho foi proposto como meta principal validar o *framework* desenvolvido no mestrado de Gomes (2005), ou seja, a idéia foi implementar um sistema que trabalhasse com a troca de mensagens, e ontologias definidas no *framework*. De maneira que a remodelagem do sistema trabalhasse com objetos de aprendizagem baseado em agentes.

O fundamento base escolhido foi o conceito de Objeto Inteligente de Aprendizagem, que é determinado como sendo um objeto de aprendizagem baseado em um paradigma de agente, capaz de gerar experiências de aprendizagem no mesmo sentido que os objetos de aprendizagem, ou seja, experiências de aprendizagem reutilizáveis.

Sabendo-se que no estado da arte da área de objetos de aprendizagem, possuímos algumas particularidades como a reusabilidade, característica principal de um objeto de aprendizagem, que é obtida com o produto de outras três características: modularidade, interoperabilidade e capacidade de ser descoberto. Desse modo, tivemos que considerar alguns requisitos para que um agente pudesse ser considerado um Objeto Inteligente de Aprendizagem. A idéia era baseada no fundamento de que um ILO deve manter as mesmas características de um objeto de aprendizagem.

Na área de objetos de aprendizagem, a interoperabilidade é obtida através da adoção de padrões. Um dos maiores grupos nesta área é o LTSC da IEEE. Portanto os padrões que um ILO deve adotar, sempre que possível, são os definidos pelo LTSC IEEE. Já na área de agentes, a interoperabilidade é obtida através de uma estrutura de comunicação comum. A organização de maior reputação nesta área é a FIPA. Portanto, um ILO deve seguir os padrões FIPA.

A capacidade de ser descoberto, na área de objetos de aprendizagem, é obtida através da incorporação de informações de metadados. Foi adotado o padrão de metadados sugerido pelo LTSC da IEEE. Na área de agentes esta capacidade pode ser obtida através da busca em serviços específicos, tal como o DF da plataforma FIPA. A modularidade, tanto na área de agentes quanto na área de objetos de aprendizagem, só é alcançada através de um projeto bem feito.

Outra característica importante definida é que um objeto inteligente de aprendizagem deve, além de ter que apresentar as características expostas acima, ter uma finalidade pedagógica, ou seja, ele deve ser utilizado no sentido de gerar experiências de aprendizagem para um estudante. Por essa razão é que foi adotada a aplicação apresentada, já que o Agente responsável pelas animações juntamente com o Agente Calculadora busca gerar experiências de aprendizagem em um sentido pedagógico.

Assim, trabalhamos com dois tipos de agentes: um agente LMS, abstração de um sistema de gerenciamento de aprendizagem, na forma de um personagem animado; e os próprios ILOs correspondentes ao Agente Calculadora e ao Agente responsável pelas animações.

Após, deu-se uma fase de remodelagem do processo de comunicação entre esses agentes, baseado no *framework* proposto por Gomes (2005). Para que os agentes se envolvam na sociedade proposta, era necessário seguir a estrutura de comunicação e os requisitos da arquitetura interna de agentes em questão.

Utilizamos essa arquitetura definida, pois consideramos que seria interessante empregar esse conjunto de recursos que facilitasse a adoção e a construção dos agentes propostos neste trabalho. Por isso que aplicamos no *framework* apresentado, como forma de auxiliar na remodelagem desse ambiente de aprendizagem.

A partir de todos os aspectos apresentados no *framework* é que foi modificado o ambiente de aprendizagem proposto neste trabalho como forma de avaliar o que havia sido declarado. Buscando uma arquitetura de agente que pudesse ser compatível com o que foi definido.

Um dos requisitos básicos deste trabalho foi a compatibilidade dos agentes aos padrões FIPA. Portanto, os agentes implementados através deste *framework* deveriam seguir estes padrões. Como forma de facilitar a adoção do padrão FIPA, optou-se por utilizar o *framework* FIPA-OS. Tal *framework* apresenta diversas funcionalidades interessantes e realmente facilita a adoção dos padrões FIPA em sistemas reais.

O *framework* utilizado é uma extensão do *framework* FIPA-OS, disponibilizando, portanto, todos os serviços e funcionalidades relativas ao processo de execução dos agentes e aos mecanismos de transporte de mensagens, entre eles: permitir que os agentes sejam implementados através da extensão de classes base, as quais contêm, para cada tipo de agente, as suas funcionalidades pré-

definidas já implementadas; e focar na possibilidade de facilitar a implementação da parte comunicativa dos agentes.

A proposta envolvida nesta pesquisa está longe de ser encerrada e ultrapassa as fronteiras deste trabalho. Com os resultados deste trabalho por si só, podemos inferir dos estudos feitos que uma abordagem baseada em agentes pode realmente trazer inúmeros benefícios à abordagem de objetos de aprendizagem, pois está sendo possível explorar pontos de convergência entre as duas tecnologias (de agentes e objetos de aprendizagem). De forma semelhante ocorre com os ambientes computacionais de educação que se utilizarem desta metodologia.

Uma das maiores dificuldades encontradas é relacionada ao desenvolvimento de um objeto de aprendizagem baseado em agente. Embora o *framework*, e até mesmo o próprio FIPA-OS, facilitem a construção destes agentes, não se pode negar que implementar um ILO é muito mais complexo do que se implementar um objeto de aprendizagem simples. No entanto, esta desvantagem é reparada pela possibilidade de se conseguir implementar objetos de aprendizagem muito mais avançados pedagogicamente.

Das metas propostas neste trabalho podemos destacar que foi validada parte da eficácia do *framework* proposto com um bom índice de satisfação. Isso porque acreditamos que a introdução do ILO responsável pela animação dentro do ambiente garantiu uma eficácia maior ao mesmo no sentido de adaptação. Pois agregando características diretas de agentes, este ILO teve uma capacidade de aprendizagem ampliada, pois só seria solicitado quando o aluno apresentasse alguma dificuldade na demonstração de alguma das propriedades matemáticas abordadas.

Buscamos com este trabalho atingir uma convergência ainda maior entre a abordagem de objetos de aprendizagem e a arquitetura de agentes, possibilitando a construção de uma material educacional que possa ser utilizado em largo espectro de domínios acadêmicos, já que reusáveis e possuindo a habilidade de auto-adaptação, através do uso de agentes.

Esperamos que este trabalho possa ainda dar contribuições importantes, refinando a eficácia da tecnologia de objetos de aprendizagem para a implementação de projetos de Educação a Distância.

Referências bibliográficas

ADL. Advanced Distributed Learning - About ADL, [S.l], ago. 2005. Disponível em: <http://www.adlnet.org/aboutadl/index.cfm>. Acesso em: ago. 2005.

AGENTTOOL - agentTool is a Java-based graphical development environment to help users analyze, design, and implement multiagent systems. Disponível em: <http://eprints.agentlink.org/5326/>. Acesso em: mai. 2005.

BELLIFEMINE, F.; POGGI, A.; RIMASSI, G. JADE: A FIPA-Compliant agent framework, In: PRACTICAL APPLICATIONS OF INTELLIGENT AGENTS AND MULTI-AGENTS, 1999, p. 97-108. **Proceedings…** [S.l:s.n] Abr. 1999.

BRENNER, W.; RÜDIGERr, Z.; WITTIG, H. **Intelligent Software Agents. Foundations and applications**. Berlin: Springer-Verlag, 1998.

DOWNES, S. Learning Objects: Resources For Distance Education Worldwide. International Review of Research in Open and Distance Learning, [S.l], v.2, n.1, jul. 2001. Disponível em: <http://www.irrodl.org/content/v2.1/downes.html>. Acesso em: 16 de mai 2005.

FERBER, J., 1999. **Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence.** Addison Wesley Professional, New York, EUA.

FININ, Tim; WEBER, Jay; WIDERHOLD, G. **DRAFT Specification of the KQML Agent-Communication Language: plus example agent policies and architectures**. [S.l.]: The DARPA Knowledge sharing Initiative External Interfaces Working Group, 1993.

FININ, T.; LABROU, Y.; MAYFELD, J. KQML as an agent communication language. IN: BRADSHAW, Jeffrey (Ed.). **Software Agents**. Menlo Park: AAAI Press/The MIT Press, 1997. p.291-316.

FOUNDATION FOR INTELLIGENT PHYSICAL – FIPA. Specifications Repository. Disponível em <http://www.fipa.org/specs/pesspecs.tar.gz> Acesso em: 16 de mai 2005.

FRIESEN, N. What are Learning Objects?. **Interactive Learning Environments**, [S.l], v.9, n.3, dez. 2001. Disponível em <http://www.careo.org/documents/objects.html>. Acesso em: dez. 2005.

GIRAFFA, Lúcia M. M. **Uma arquitetura de tutor utilizando estados mentais.** 1999. Tese (Doutorado em Ciências da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

GLUZ, J. C. **Linguagens de Comunicação entre Agentes: Fundamentos e Propostas de Padronização**. Trabalho Individual I (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre, 2002.

GOMES, Eduardo. R; SILVEIRA, Ricardo A; VICARI, Rosa M. Objetos Inteligentes de Aprendizagem: Uma abordagem baseada em agentes para objetos de aprendizagem. In: XV SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2004, Manaus. **Anais do**... Manaus: SBC, 2004. p. 408-417.

GOMES, Eduardo. **Objetos Inteligentes de Aprendizagem: uma abordagem baseada em agentes para objetos de aprendizagem**. 2005. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre.

JAQUES, P. A. et al. An Animated Pedagogical Agent that Interacts Affectively with the Student**.** ARTIFICIAL INTELLIGENCE IN EDUCATION, shaping the future of learning through intelligent technologies, 2003, Sydney, Australia. (Poster)

JAVADOC TOOL – Ferramenta de geração de documentação do código fonte em formato HTML. Disponível em <http://java.sun.com/j2se/javadoc/index.jsp> Acesso em: 18 de mai 2005.

JOHNSON**,** W. L. et al., 2000, Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. The International Journal of Artificial Intelligence in

Education, Vol. 11, pp. 47-78.

LABROU, Y.; FININ T.; PENG, Y. Agent Communication Languages: The Current Landscape. **IEEE Intelligent Systems**. [S.l], p. 45-52. Mar. 1999.

LAUKKANEN, M. **Sonera Evaluation: Evaluation of FIPA-OS 1.03**, Helsinki: [s.n], Fev. 2000. Disponível em <http://fipa-os.sourceforge.net/docs/papers/soneraevaluation.pdf> Acesso em: ago. 2005

LEARINING TECHNOLOGY STANDARD COMITEE - LTSC of the Institute of Electrical and Electronics Engineers (IEEE), [S.l]. Disponível em: http://ltsc.ieee.org Acesso em: 16 de mai 2005.

LONGMIRE, W. A Primer on Learning Objects. **Learning Circuits**, [S.l: s.n] Mar. 2000. Disponível em <http://www.learningcircuits.org/2000/mar2000/Longmire.htm> Acesso em: ago. 2005.

LTSC. Learning Technology Standard Comitee (LTSC) of the Institute of Electrical and Electronics Engineers (IEEE), [S.l], jan. 2004 Disponível em: <http://ltsc.ieee.org>. Acesso em: mai. 2005.

LUCAS, J. P.; WILGES, B.; SILVEIRA, R. A. (2005) Inserting Animated Pedagogical Agents Inside Distributed Learning Environment s by Means of FIPA Specifications. In: Fourth International Joint Conference on Autonomous Agents & Multiagents Systems (AAMAS) – Proceedings of Agent - Based Systems for Human learning Workshop (ABSHL) 2005, Utrecht.

MOHAN, P.; BROOKS, C. Engineering a Future for Web-Based Learning Objects. In: INTERNATION CONFERENCE ON WEB ENGINEERING, ICWE 2003, jul. 2003. **Proceedings...** Oviedo, Spain: Lectures Notes in Computer Science: Springer, 2003. p. 120-123.

NWANA, H. et al. ZEUS: a toolkit for building distributed multi-agent systems, In: **Artificial Intelligence Journal**, [S.l:s.n] v.13, n.1, 1999. p. 129-186.

PEREIRA, A.S. **Um Estudo de Aplicações de Ensino na Internet Orientada a Agentes.** Porto Alegre: UFRGS, 1997. (Trabalho Individual).

QUINN, C. Learning Objects and Instruction Components. **Educational Technology & Society**, [S.l], v.3, n.2, fev. 2000. Disponível em: <http://ifets.ieee.org/discussions/discuss_feb2000.html>. Acesso em: mai. 2005.

ROBSON, R. Object-oriented Instructional Design and Web-based Authoring. In: WORLD CONFERENCE ON EDUCATIONAL MULTIMEDIA, HYPERMEDIA AND TELECOMMUNICATIONS, ED-MEDIA, 1999, Seatle, USA. **Proceedings...** Seatle, USA: AACE, 1999. p. 698-702. Disponível em: <www.eduworks.com/robby/papers/objectoriented.pdf>. Acesso em: mar. 2004.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial.** 2. ed. Rio de Janeiro: Elsevier, 2004. 1021p.

SEARLE, J. **Os Actos de Fala - Um Ensaio de Filosofia da Linguagem**. Traduzido por: Carlos Vogt. Coimbra: Livraria Almedina, 1981. Tradução de: Speech Acts - An Essay in the Philosophy of Language.

SILVEIRA**,** R. A.; GOMES, E. R.; VICARI, R. M. Intelligent Learning Objects: An Agent-Based Approach of Learning Objects. In: WORKING CONFERENCE INFORMATION AND COMMUNICATION TECHNOLOGIES (ICT) AND REAL-LIFE LEARNING, IFIP WG 3.2 & 3.4, 2004, Proceeding: **Anais do...** Melbourne, 2004.

SINGH, H. An Intro to Metadata Tagging . **Learning Circuits**, [S.l: s.n] Dec. 2000. Disponível em <http://www.learningcircuits.org/2000/dec2000/singh.html> Acesso em: ago. 2005.

SOSTERIC, M.; HESEMEIER, S. When is a Learning Object not an Object: A first step towards a theory of learning objects. **International Review of Research in Open and Distance Learning**, [S.l:s.n], v.3, n.2, out. 2002. Disponível em <http://www.irrodl.org/content/v3.2/soc-hes.html>. Acesso em: dez. 2005.

WILEY, D. A. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In: WILEY, D. A. (Ed.) **The Instructional Use of Learning Objects**. [S.l:s.n], 2000a. Disponível em <http://reusability.org/read/chapters/wiley.doc>. Acesso em: dez. 2005.

WILEY, D. A.; GIBBONS, A.; RECKER, M. M. **A reformulation of learning object granularity**. [S.l:s.n] 2000b. Disponível em <http://reusability.org/granularity.pdf> Acesso em: Ago. 2005.