UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDUARDO RODRIGUES GOMES

Objetos Inteligentes de Aprendizagem: uma abordagem baseada em agentes para objetos de aprendizagem.

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dra. Rosa Maria Vicari
Orientadora

Prof. Dr. Ricardo Azambuja Silveira
Co-orientador

Porto Alegre, junho de 2005.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Gomes, Eduardo Rodrigues

Objetos Inteligentes de Aprendizagem: uma abordagem baseada em agentes para objetos de aprendizagem / Eduardo Rodrigues Gomes – Porto Alegre: Programa de Pós-Graduação em Computação, 2005.

98 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2005. Orientador: Rosa Maria Vicari; Co-orientador: Ricardo Azambuja Silveira.

1.Objetos Inteligentes de Aprendizagem. 2.Objetos de Aprendizagem 3.Agentes. 4.FIPA 5.LTSC IEEE I. Vicari, Rosa Maria. II. Silveira, Ricardo Azambuja. III. Título.

AGRADECIMENTOS

Gostaria de agradecer a todos aqueles que de alguma forma contribuem com o meu crescimento.

Agradeço aos meus irmãos Márcia, Evandro e Sílvia: obrigado pelo carinho e amizade. Agradeço aos meus sobrinhos José, João e Marco: obrigado pelas alegrias e saudades. Agradeço à minha namorada Larissa: valeu. Em especial, agradeço aos meus pais José e Enice: obrigado pela imensa atenção dedicada aos filhos, e netos, mas principalmente pelo incentivo permanente em cada um de nossos projetos.

Agradeço também aos meus orientadores Rosa e Ricardo: obrigado pela amizade e pela inestimável orientação.

Por fim, agradeço à Fundação Coordenação de Pessoal de Nível Superior (CAPES) pelo apoio através do Programa de Apoio à Pesquisa em Educação à Distância (PAPED).

SUMÁRIO

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
DMCOC	*Data Model for Content Object Communication* – Modelo de Dados para Comunicação de Objetos de Conteúdo
EAD	Educação a Distância
FIPA	*Foundation for Intelligent Physical Agents*
FIPA-OS	FIPA – *Open Source*
IA	Inteligência Artificial
ILE	*Intelligent Learning Environment* – Ambiente Inteligente de Aprendizagem
ITS	*Intelligent Tutoring System* – Sistema Tutor Inteligente
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

SMA	Sistema Multiagente
TM	*Task Manager* – Gerenciador de Tarefas

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Esta pesquisa propõe uma abordagem na qual objetos de aprendizagem são construídos com base no paradigma de agentes. A fundamentação tecnológica desta abordagem é constituída por uma integração entre tecnologias desenvolvidas para Objetos de Aprendizagem e para Sistemas Multiagentes.

O conceito central apresentado é o de Objeto Inteligente de Aprendizagem, entidade que corresponde a um agente que é capaz de gerar experiências de aprendizagem reutilizáveis, no mesmo sentido que os objetos de aprendizagem.

É apresentada uma sociedade multiagente concebida com a finalidade de dar suporte a abordagem proposta, bem como a modelagem do processo de comunicação entre os agentes desta sociedade.

Como forma de validar as propostas feitas, são apresentados uma arquitetura de agentes que implementa os conceitos definidos e um conjunto de recursos para a construção de agentes compatíveis com esta arquitetura. Através destes recursos é possível a implementação das entidades propostas neste trabalho.

Palavras-Chave: Objetos Inteligentes de Aprendizagem, Agentes, Objetos de aprendizagem, FIPA, LTSC IEEE.

Intelligent Learning Objects: an agent-based approach for learning objects.

ABSTRACT

This research proposes an approach in which learning objects are built based on the Agent Paradigm. The technologic fundamentals of this approach is composed of an integration between technologies developed for both Multi-agent Systems and Learning Objects

The central concept presented herein is the Intelligent Learning Object's concept, an entity that corresponds to an agent that can generate reusable learning experiences, in the same sense that learning objects can do.

This work presents an agent society that supports the proposed approach. It also presents the communication's modeling for the agents of this society.

To validate the proposed approach, it presents both an agent architecture that implements the defined concepts and a framework to construct agents that are compliant with this architecture. This toolkit may be used to implement the entities that are proposed in this work.

Keywords: Intelligent Learning Objects, Agents, Learning Objects, FIPA, LTSC IEEE.

1 INTRODUÇÃO

Este trabalho busca por pontos de convergência entre as tecnologias de Objetos de Aprendizagem e de Sistemas Multiagente como forma de dar mais flexibilidade, adaptabilidade e interatividade a ambientes de aprendizagem. O produto desta busca é uma abordagem na qual objetos de aprendizagem são construídos com base em agentes.

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). Não existe um conceito único do que seja um objeto de aprendizagem. Para o Learning Technology Standard Comitee (LTSC) do Institute of Electrical and Electronics Engineers (IEEE) (LTSC, 2004), um objeto de aprendizagem é qualquer entidade, digital ou não, que possa ser usada, reusada e referenciada durante alguma forma de aprendizagem suportada por tecnologia. Para Sosteric & Hesemeier (2002), um objeto de aprendizagem é um arquivo digital (imagem, filme, etc...) que pode ser usado com algum propósito educacional e que inclui, internamente ou via associação, sugestões de contextos nos quais ele deve ser utilizado.

A principal propriedade de um objeto de aprendizagem é a *reusabilidade*. Tal característica pode ser alcançada através da modularidade, da interoperabilidade e da capacidade de ser descoberto (FRIESEN, 2001). Ser modular significa ser pequeno o suficiente para que possa ser reutilizável. Ser interoperável significa poder funcionar em sistemas diversos. Ter a capacidade de ser descoberto diz respeito à capacidade de poder ser encontrado em função da descrição de suas propriedades e funcionalidades.

Muitos grupos de pesquisa vêm trabalhando bastante no sentido de alcançar estas características. Entre eles podemos destacar o LTSC IEEE (2004), a *Alliance of Remote Instructional Authoring and Distribution Networks for Europe* (ARIADNE) (2004), o *IMS Global Learning Consortium* (IMS) (2004), e a *Advanced Distributed Learning initiative* (ADL) (2004).

Alguns autores associam os objetos de aprendizagem ao Modelo de Orientação a Objetos (QUINN, 2000) (ROBSON, 1999) (WILEY, 2000a). Wiley (2000a) diz que 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 do Paradigma de Programação Orientada a Objetos. Esta associação, embora criticada por outros autores (FRIESEN, 2001) (SOSTERIC, 2002), pode ser verificada na prática através do modelo SCORM (ADL, 2004) (ver secção 5.2)

1.1 Motivação

Apesar de todo o esforço que vem sendo despendido na área, muito trabalho precisa ser feito para o uso de um objeto de aprendizagem (DOWNES, 2002): é necessário a construção de um ambiente educacional no qual este objeto funcione; os possíveis usuários precisam localizar os objetos de aprendizagem e então arranjá-los de acordo com algum fim pedagógico; em alguns casos são necessárias a instalação e a configuração de softwares específicos para que estes objetos possam ser visualizados (um vídeo ou uma animação flash, por exemplo); e, os objetos precisam ser entregues em algum tipo de contexto educacional.

P Mohan e C Brooks (MOHAN, 2003) apontam mais algumas limitações: a tarefa de encontrar o objeto de aprendizagem correto é muito custosa, porque o especialista em conteúdo precisa examinar cuidadosamente cada um deles. Para Downes (2002), o tanto de trabalho necessário para o uso de um objeto de aprendizagem leva-nos a acreditar que necessitamos na verdade é de objetos de aprendizagem mais “espertos”.

Por outro lado, uma família de ambientes computacionais voltados à educação tem utilizado técnicas de Inteligência Artificial como forma de se adequar às necessidades de aprendizagens ricas e complexas que os estudantes realmente necessitam. Isso é feito através da incorporação de mecanismos que levam em consideração as características dos estudantes e que tentam prover alguma adaptação a estas características. Tais sistemas são conhecidos como Sistemas Tutores Inteligentes (ITS – *Intelligent Tutoring Systems*) ou Ambientes Inteligentes de Aprendizagem (ILE – *Intelligent Learning Environments*).

Diversos trabalhos nessa área têm proposto o uso de arquiteturas baseadas em sistemas multiagente (GIRAFFA, 1998) (JOHNSON, 1997). Um sistema multiagente (SMA) é composto por uma comunidade de entidades individuais denominadas agentes. A principal característica de um agente é a autonomia, ou seja, a capacidade de agir e de interagir com outros agentes sem a necessidade de intervenção humana e com vistas à satisfação dos seus próprios objetivos.

O uso de agentes na concepção de sistemas educacionais traz algumas vantagens, tais como: o conhecimento pode ser distribuído entre vários “tutores” (agentes), cada um com suas crenças, desejos, objetivos, emoções e planos de ação; essa distribuição cria maiores oportunidades de variar técnicas pedagógicas; o aprendiz interage com um tutor de forma mais flexível; o aprendiz pode passar conhecimentos ao tutor que serão repassados a outros aprendizes. Além de tornar mais fácil a adição de novos agentes ao sistema, pois cada agente é um módulo único e independente do outro. Ambientes de ensino baseados em arquiteturas multiagentes possibilitam suportar o desenvolvimento de sistemas de forma mais robusta, mais rápida e com menores custos.

Os agentes, assim como os objetos de aprendizagem, não possuem um conceito aceito por toda a comunidade de pesquisadores. Para Bradshaw (1997) um agente é uma entidade de software que trabalha continuamente e de forma autônoma em um ambiente particular geralmente habitado por outros agentes. Um agente é capaz de interferir neste ambiente de forma flexível e inteligente, sem requerer intervenção humana ou direcionamento. 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.

Ao longo dos anos, instituições e grupos de pesquisas vêm trabalhando no sentido de promover padrões que possibilitem a construção de agentes compatíveis entre si. Uma destas instituições é a FIPA. A Foundation for Intelligent Physical Agents (FIPA, 2004) é uma fundação internacional sem fins lucrativos voltada exclusivamente para a criação de padrões que tornem possível a implementação de agentes interoperáveis. Entre os desenvolvimentos promovidos por esta fundação estão uma linguagem de comunicação exclusivamente voltada para a comunicação entre agentes, a FIPA-ACL, e uma linguagem para a descrição do conteúdo das comunicações, a FIPA-SL.

Alguns autores acreditam que os agentes podem ser considerados como uma evolução do paradigma de orientação a objetos. Para Bauer (2001) um agente é um objeto ativo dotado de autonomia dinâmica, capacidade de decidir quando deve agir e sem a necessidade de invocação externa, e de autonomia determinística, capacidade de recusar ou de modificar uma requisição externa. Para Fernadéz (1998), um agente pode ser visto como um elemento integrador de técnicas de orientação a objetos e sistemas baseados em conhecimento. Dessa forma, os agentes se caracterizam como uma extensão dos objetos, porque proporcionam uma abstração maior do que estes, e como sistemas baseados em conhecimento, pois trabalham com informações, com exceção dos puramente reativos.

Com base no que foi explicitado acima, cremos que dotar um objeto de aprendizagem de características típicas de agentes, tais como autonomia, conhecimento sobre si próprio, e objetivos, pode agregar valor ao mesmo e aos ambientes de educacionais produzidos com ele, deixando-os mais autônomos, dinâmicos e adaptáveis (GOMES, 2004) (SILVEIRA, 2004).

1.2 Objetivos

O objetivo geral deste trabalho é investigar pontos de convergência entre as tecnologias de agentes e de objetos de aprendizagem, propondo uma abordagem que permita a construção de objetos de aprendizagens baseados em agentes.

Aos objetos de aprendizagem construídos através desta abordagem foi dado o nome de Objetos Inteligentes de Aprendizagem (ILO – *Intelligent Learning Object*).

A fim de alcançar o objetivo geral foram definidos os seguintes objetivos específicos:

- Estudar aspectos da área de Sistemas Multiagente: foram estudados conceitos, características, métodos e linguagens de comunicação e propostas de padronização (em especial os padrões FIPA);
- Estudar os principais aspectos da área de Objetos de Aprendizagem: estudou-se os seus conceitos e suas tecnologias de suporte;
- Propor uma sociedade multiagente que suporte a abordagem apresentada;
- Propor uma arquitetura de agentes para a modelagem e implementação desta proposta. Esta arquitetura leva em consideração os seguintes requisitos:
    - Quantos aos agentes:
        - Uma arquitetura interna de agentes que manipule informações de metadados;

        - A modelagem de comunicação entre os agentes de acordo com os padrões FIPA, sempre que possível;
    - Quanto aos Objetos de Aprendizagem:
        - Utilização de padrões definidos pelo LTSC IEEE, sempre que possível;
- Implementar uma infra-estrutura básica de classes JAVA que permita a construção de Objetos Inteligentes de Aprendizagem;
- Avaliar as propostas efetuadas;

1.3 Contribuições Esperadas

Ao final deste trabalho esperamos termos constituído a formulação de uma abordagem na qual objetos de aprendizagem são construídos através de arquiteturas de agentes.

O diferencial entre as abordagens atuais para objetos de aprendizagem e a abordagem proposta nesta pesquisa é justamente a utilização de tecnologias desenvolvidas para agentes e sistemas multiagente. Os benefícios desta utilização são discutidos na secção 6.1. O Capítulo 5 apresenta alguns trabalhos relacionados e discute as diferenças encontradas entre eles e a abordagem proposta.

Ao final deste trabalho teremos modelado uma sociedade multiagente que suporta o funcionamento da abordagem proposta, assim como teremos modelado do processo de comunicação entre os agentes que compõem esta sociedade. Teremos também um exemplo de arquitetura de agente que implementa os fundamentos propostos e um conjunto de recursos para a construção de agentes compatíveis com esta arquitetura.

Através destes desenvolvimentos esperamos potencializar a implementação de ambientes de aprendizagem mais flexíveis e adaptáveis a partir da possibilidade de se construir 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 Capítulo seguinte apresenta aspectos da área de Sistemas Multiagente. Serão enfatizados os aspectos que envolvem a comunicação em uma comunidade de agentes.

O Capítulo 3 apresenta as propostas de padronização da FIPA. São tratados os aspectos gerenciais de uma plataforma padrão FIPA e a linguagem FIPA-ACL. É ainda apresentado o *framework* FIPA-OS (FIPA-OS, 2004), utilizado na implementação dos recursos previstos neste trabalho.

O Capítulo 4 apresenta a tecnologia de objetos de aprendizagem.

No Capítulo 5 são analisados alguns trabalhos relacionados com o que foi definido nesta pesquisa.

O Capítulo 6 apresenta a abordagem proposta neste trabalho. São discutidos seus fundamentos, potencialidades e requisitos. Na seqüência do Capítulo é apresentada uma sociedade multiagente que implementa os conceitos propostos e apresentada a modelagem do processo de troca de informações entre os agentes desta sociedade.

O Capítulo 7 apresenta uma arquitetura de agentes que pode ser utilizada na implementação dos agentes propostos. É apresentado também um mapeamento desta arquitetura para uma arquitetura real e um *framework* desenvolvido com o objetivo de facilitar a implementação de agentes que seguem esta arquitetura.

Por fim, no Capítulo 8 são apresentadas as considerações finais, tais como conclusões, trabalhos futuros e dificuldades encontradas.

2 CONCEITOS DE SISTEMAS MULTIAGENTE

Atualmente, os sistemas multiagente constituem uma das áreas de maior atividade em pesquisa e desenvolvimento dentro da Ciência da Computação. Um sistema multiagente é composto por uma comunidade de entidades individuais denominadas agentes. A principal característica de um agente é a autonomia, ou seja, a capacidade de agir e de interagir com outros agentes sem a necessidade de intervenção humana e com vistas à satisfação dos seus próprios objetivos.

O objetivo deste Capítulo é dar ao leitor uma visão geral acerca desta abordagem, de forma que ao final dela, o leitor consiga estar familiarizado com os conceitos utilizados neste trabalho.

2.1 IA e IAD

A Inteligência Artificial (IA) é tradicionalmente apresentada como a parte da Ciência da Computação cuja ênfase está no estudo de sistemas computacionais inteligentes, ou seja, sistemas que exibam características associadas à inteligência do comportamento humano – compreensão de linguagem, aprendizado, raciocínio, resolução de problemas, e assim por diante (BARR, 1981).

Na Inteligência Artificial Distribuída, um ramo da IA, a busca pela solução de um problema parte da individualidade para a coletividade, ou seja, a inteligência é vista como emergente da ação e interação de entidades (agentes) autônomas.

Segundo Torsun (1995), a IAD divide-se em três áreas:

- **Resolução Distribuída de Problemas (RDP):** Interessa-se pela decomposição de problemas em um número de entidades que cooperam entre si e dividem conhecimento. As entidades são desenhadas especificamente para um problema em particular;
- **Sistemas Multiagente (SMA):** Caracteriza-se pela existência de um certo número de entidades autônomas, heterogêneas e potencialmente independentes, trabalhando juntas para resolver um problema. A sociedade existe independentemente da existência de um problema particular;
- **Inteligência Artificial Paralela (IAP):** Preocupa-se mais por problemas de *performance* do que por avanços conceituais, interessando-se principalmente em desenvolver linguagens e algoritmos de computação paralela.

Atualmente, o termo Sistema Multiagente é utilizado para denominar qualquer sistema composto por um conjunto de agentes. O termo Resolução Distribuída de Problemas está em desuso.

2.2 Agentes

Ainda não existe um conceito unânime e uniforme para o termo “agente”. O que existe na realidade é uma convergência para características fundamentais que uma “peça de software” deve possuir para ser considerada um agente.

Russell & Norvig (1995) definem um agente como um sistema capaz de perceber, através de sensores, e agir, através de atuadores, em um dado ambiente.

Um agente também pode ser uma entidade à qual se atribuem estados, denominados estados mentais, tais como crenças, decisões, capacidades, compromissos e objetivos (conceitos análogos ou similares aos humanos) (SHOHAM, 1993).

Segundo 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.

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

Alguns autores acreditam que os agentes podem ser considerados como uma evolução do paradigma orientado a objetos. Para Bauer (2001), um agente é um objeto ativo dotado de autonomia dinâmica, capacidade de decidir quando devem agir e sem a necessidade de invocação externa, e de autonomia determinística, capacidade de recusar ou de modificar uma requisição externa. Para Fernadéz (1998), um agente pode ser visto como um elemento integrador de técnicas de Orientação a Objetos e Sistemas Baseados em Conhecimento. Dessa forma, os agentes se caracterizam como uma extensão dos objetos, porque proporcionam uma abstração maior do que estes, e como sistemas baseados em conhecimento, pois trabalham com informações, com exceção dos puramente reativos.

Tendo em vista as várias visões apresentadas, podemos destacar as seguintes propriedades para agentes:

- **Autonomia:** Um agente deve agir sem intervenção humana direta, portanto deve possuir algum tipo de controle sobre suas ações e seu estado interno;
- **Reatividade:** Um agente deve ser capaz de reagir aos estímulos externos produzidos pelo seu ambiente ou por outros agentes;

- **Pró-atividade:** Um agente não somente reage ao seu ambiente, mas também deve exibir um comportamento orientado à satisfação de seus objetivos (Orientação a Objetivos).
- **Intencionalidade:** Capacidade de representação explícita dos seus objetivos;
- **Racionalidade:** Habilidade de agir de forma a atingir seus objetivos e não contra eles;
- **Continuidade temporal:** Persistência de identidade por longos períodos de tempo;
- **Sociabilidade:** Habilidade de interação com outros agentes através de mecanismos de comunicação;
- **Benevolência:** Capacidade de cooperação com outros agentes;
- **Adaptabilidade:** Capacidade de aprender através da experiência;
- **Mobilidade:** Capacidade de mover-se de um ambiente para outro.

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

A Figura 2.1 ilustra de forma esquemática a noção elementar de um agente. Pode-se notar os atributos de persistência (estado interno), reatividade (percepções e ações) e pró-atividade (estado interno e ações). Os atributos de sociabilidade não estão esquematizados, pois nos conduzem a conceitos de Sistemas Multiagente que serão discutidos mais adiante.

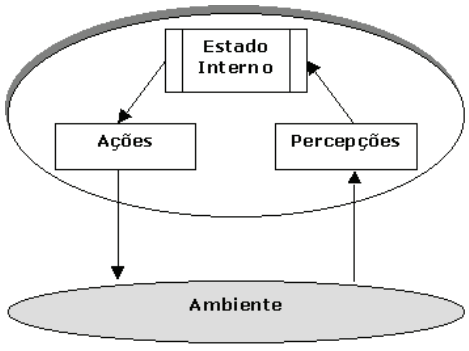

Figura 2.1: Esquema genérico de um agente

2.3 Arquiteturas de Agentes

A arquitetura de um agente compreende a descrição de quais são e como são os processos internos que possibilitam a interação do agente com o seu ambiente. A comunidade de IA costuma aglutinar as arquiteturas de agentes em dois grandes grupos: Arquiteturas deliberativas (cognitivas) e não-deliberativas (reativas). Essa divisão funciona mais para fins didáticos do que práticos. Pois, na maioria das vezes, o que

realmente se utiliza é uma arquitetura híbrida, ou seja, que contempla aspectos reativos e cognitivos.

A arquitetura Interrap, descrita por Müller (1996), é um exemplo típico de um sistema de agentes híbrido.

2.3.1 Arquiteturas reativas

Um agente reativo, ou não deliberativo, não possui um modelo simbólico interno de seu ambiente. Portanto, não tem a capacidade de executar raciocínios complexos. É da interação com o ambiente que ele obtém a sua inteligência. A escolha das ações está relacionada com a ocorrência de um conjunto de eventos que acontecem no ambiente do agente.

São propriedades dos agentes reativos:

- Não haver representação explícita do conhecimento – o conhecimento é implícito e se manifesta através do comportamento;
- Não haver representação do ambiente – o comportamento é baseado no que é percebido a cada instante;
- Não haver memória das ações – o resultado de uma ação passada não exerce nenhuma influência sobre uma ação futura.

A descrição feita por Brooks (1986) denominada arquitetura de subsunção é um excelente exemplo de modelo não-deliberativo.

2.3.2 Arquiteturas cognitivas

Agentes deliberativos, ou cognitivos, possuem uma representação simbólica do seu ambiente e possuem capacidade de raciocínio lógico acerca desse conhecimento. A escolha das ações é feita através de uma deliberação sobre determinados fatores, por exemplo, algum plano.

São características dos agentes deliberativos:

- Possuir uma representação explícita do ambiente e de outros agentes;
- Possibilidade de manter um histórico de ações ou interações passadas;
- Têm um mecanismo de controle deliberativo – os agentes raciocinam e decidem sobre quais objetivos devem alcançar, que planos seguir e quais as ações a serem tomadas em determinado instante.

Shoham (1993) descreve um modelo de agentes baseados em estados mentais. As arquiteturas que seguem este paradigma são conhecidas como **arquiteturas BDI** (Belief, Desire and Intention) e podem ser consideradas como um subconjunto das arquiteturas deliberativas.

As arquiteturas BDI descrevem o processamento interno dos estados de um agente através de um conjunto de estados mentais: **crenças**, **desejos** e **intenções**. Novas visões adicionam as idéias de **planos** e **objetivos** ao modelo BDI clássico (MÜLLER, 1996).

As crenças representam o possível conhecimento do agente. Um agente pode ter crenças sobre si próprio, sobre o mundo e sobre outros agentes.

Os desejos são o conjunto de metas a serem realizados num período de tempo. Os desejos motivam o agente a agir de forma a realizar as metas. Os desejos podem ser conflitantes.

Os objetivos representam um subconjunto dos desejos. Em contraste com os desejos, os objetivos não podem ser conflitantes.

As intenções representam um subconjunto dos objetivos. Se um agente decide atingir determinado objetivo esse objetivo passa a ser uma intenção.

Os planos combinam as intenções de um agente em unidades consistentes.

Um exemplo de arquitetura deliberativa é descrito por Rao & Georgeff (1995). Eles desenvolveram um modelo que pode ser usado para implementar agentes BDI.

2.4 Sistemas Multiagente

Um Sistema Multiagente (SMA) é conceituado como um sistema composto por um conjunto de agentes trabalhando no sentido de alcançar um resultado 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). Além disso, muitos problemas têm características distribuídas e, portanto, necessitam de unidades distribuídas de conhecimento para a sua resolução. Portanto, é interessante a existência de unidades independentes que resolvesse cada questão individualmente e que juntos resolvessem o problema todo, ou seja, um sistema composto por mais de um agente que trabalham juntos para alcançarem um objetivo comum. Este é um SMA.

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

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

2.5 Comunicação entre Agentes

A comunicação é o processo pelo qual são trocadas informações entre os agentes. Esta troca de informação é de suma importância para sistemas multiagente complexos, onde cada agente é especializado em alguma tarefa e os resultados desta tarefa precisam ser utilizados por outros agentes.

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

Os sistemas baseados em quadro de avisos, também conhecidos como sistemas de quadro negro, mantêm uma área de trabalho comum na qual os agentes podem trocar informações, dados e conhecimentos. Uma ação de comunicação é iniciada quando um agente escreve alguma coisa no quadro negro. Os agentes podem acessar o quadro negro a qualquer momento e verificar se alguma informação foi modificada ou adicionada.

Segundo Brenner (1998), os sistemas de quadro negro provêm um conceito muito flexível para a cooperação e comunicação entre agentes, pois são independentes da estratégia de cooperação selecionada. Entretanto, a leitura das informações do quadro

negro pode causar uma sobrecarga do sistema. Por isso, a importância dos sistemas baseados em troca de mensagens vem crescendo continuamente.

A Figura 2.2 ilustra o conceito básico do sistema de troca de mensagens. Um agente, denominado emissor, envia uma mensagem a um outro agente, o receptor. Em contraste com os sistemas de quadro negro, as informações são trocadas diretamente entre os agentes.

Figura 2.2: Comunicação por troca de mensagens

Finin (1993) diz que 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. Uma grande vantagem da abordagem ponto-a-ponto é que o agente emissor sempre sabe para onde uma mensagem está sendo enviada, o que permite a fácil introdução de controles de segurança;
- ***Broadcast:*** A mensagem é enviada para todos os agentes da sociedade. Na abordagem *broadcast*, um agente pode ser substituído por outro equivalente e o processamento do sistema será inalterado. A passagem de mensagem por *broadcast* não é segura, já que qualquer agente pode examinar o conteúdo de qualquer mensagem;
- ***Multicast:*** A sociedade de agentes é dividida em grupos. Desta forma as mensagens podem ser enviadas somente aos agentes de determinados grupos.

Atualmente, os sistemas multiagente são praticamente todos desenvolvidos com comunicação por troca de mensagens. Isso porque esta técnica proporciona muito mais flexibilidade e dinamismo do que a arquitetura de quadro de avisos apresentada na secção anterior.

2.5.1 Linguagens de Comunicação de Agentes

A forma atual mais usual de interação entre agentes é a comunicação através de troca de mensagens. Este tipo de comunicação foi fortemente influenciado pelo desenvolvimento da Teoria dos Atos da Fala (SEARLE, 1981) e de seus desdobramentos. Estes estudos serviram como base para a análise do fenômeno da comunicação entre agentes e fundamentaram o desenvolvimento de linguagens para uso na comunicação entre agentes: as Linguagens de Comunicação de Agentes (ACL – *Agent Communication Language*).

A Teoria dos Atos da Fala trata a linguagem como um veículo de comunicação, ou seja, como um mecanismo para efetivar a troca de informações entre os falantes e ouvintes. Nesta teoria, o ato de falar é dividido em três elementos ou atos: *Ato Locucionário, Ato Ilocucionário e Ato Perlocucionário.* O *Ato Locucionário* corresponde à enunciação dos fonemas, sílabas e palavras. O *Ato Ilocucionário* compreende o significado ou a semântica que o falante tentou associar ao seu ato

locucionário e que está tentando transmitir ao ouvinte. E o *Ato Perlocucionário* corresponde aos efeitos que o ato de falar em si causaram no ouvinte.

O trabalho de Searle focou principalmente os atos ilocucionários, ou simplesmente atos da fala. Um dos seus avanços importantes foi afirmar que eles possuem significado, independentemente do contexto da sua enunciação. Esta característica foi herdada pelas linguagens de comunicação de agentes desenvolvidas com base em sua teoria.

Assim, uma ACL deve determinar um meio através do qual uma atitude (afirmação, requisição, consulta...) a respeito do conteúdo da troca de conhecimentos é comunicada (FINNIN, 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 dever 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. Na prática, 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 *framework* FIPA-OS, que ela passou a ter uma implementação disponível e a competir com a KQML.

Neste 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.

2.5.2 Protocolos de Interação entre Agentes

Imagine o seguinte cenário: ao chegar em uma padaria, o atendente lhe dá as boas vindas e pergunta o que você quer comprar; você informa que quer dez pães franceses; logo após, ele lhe entrega os pães e pergunta se você quer algo mais; você pode dizer que sim, e fazer um novo pedido, ou dizer que não, e encerrar a conversa entregando a ele o pagamento.

Partindo do exemplo acima, podemos inferir que o simples uso de um ato da fala, definido em forma e semântica em uma linguagem de comunicação de agentes, nem sempre será suficiente para modelar as interações complexas que podem ocorrer entre os agentes de um sistema multiagente. E, por conseqüência, estes agentes também precisam utilizar diálogos ou conversações.

É claro que não existe um único arranjo possível de enunciações que pode ser utilizado para se alcançar um objetivo. Existe na verdade um sem número deles. No nosso exemplo, nem sempre compradores e vendedores irão seguir o mesmo protocolo, o que dá flexibilidade à comunicação. Porém, essa característica é bastante limitada quando tratamos de agentes. Ainda não é possível calcular todas as possibilidades de interações e dar flexibilidade suficiente ao agente para que ele "compreenda" todas elas. Considerando essa limitação, na prática, os diálogos entre agentes acabam caindo em padrões típicos: os Protocolos de Interação.

Assim, um **Protocolo de Interação**, ou Protocolo de Conversação, ou simplesmente Protocolo, é um padrão típico de comunicação entre agentes, onde uma certa seqüência de mensagens é esperada, e, em qualquer ponto da comunicação, outras mensagens são esperadas para seguir (FIPA, 2004). Nowostawski (2001) define um protocolo de interação como um modelo de uma seqüência de atos comunicativos. As duas definições apresentadas acima são compatíveis entre si.

Uma **conversação**, ou diálogo, corresponde à uma instância de um ou de um conjunto de protocolos de interação (NOWOSTAWSKI, 2001). Note a diferença: um protocolo é um modelo, enquanto que uma conversação é uma instância deste modelo. Essa definição é também compatível com a da FIPA, onde um diálogo é visto como "uma seqüência em andamento de trocas de atos comunicativos entre agentes e relacionada a algum tópico do discurso" (FIPA, 2004).

A organização FIPA define uma série de Protocolos de Interação entre Agentes. Na definição destes protocolos foram utilizados diagramas da AUML (BAUER, 2001).

2.5.3 Ontologias para comunicação entre agentes

O modelo de comunicação da FIPA é baseado na hipótese de que dois agentes, que pretendam conversar, saibam e apliquem o mesmo significado para todos os símbolos utilizados no conteúdo das mensagens. Isso é feito através da asseguração de que os dois agentes conheçam a mesma ontologia.

No sentido filosófico, dado por Aristóteles, uma ontologia é vista como um sistema particular de categorias relacionado a uma certa visão do mundo. Segundo este conceito, uma ontologia independe de uma linguagem particular usada para descrevê-la, ou seja, ela é sempre a mesma.

No entanto, esta definição não é a única e nem a mais usada na área da IA. Para a IA uma ontologia refere-se a um artefato constituído por um vocabulário específico usado para descrever certa realidade, somado a um conjunto de assunções explícitas concernindo o significado pretendido das palavras do vocabulário. Este conjunto de assunções possui usualmente a forma de uma teoria lógica de primeira ordem, onde as palavras do vocabulário aparecem como nomes de predicados unários (conceitos) ou binários (relações).

Para a FIPA, as duas visões de ontologias apresentadas acima são relacionadas. No entanto, a FIPA adota o termo *conceitualização* para se referir à visão filosófica. Assim, duas ontologias podem ser diferentes no vocabulário usado (Inglês e Português, por exemplo) enquanto compartilham a mesma conceitualização.

A partir desta divisão, uma ontologia pode ser definida como uma especificação de uma conceitualização. Dois agentes podem compartilhar a mesma conceitualização e utilizar ontologias diferentes.

Ontologias possuem papel fundamental na troca de informações entre agentes. Como exemplo, tenha em mente a palavra “time”. Se esta palavra for passada como conteúdo de uma mensagem para um agente que conhece uma ontologia acerca da língua inglesa, é possível que este agente entenda o seu conceito como o conceito de “tempo”. No entanto, a intenção do agente emissor, que aplicou uma ontologia sobre a língua portuguesa ao construir a mensagem, era discorrer sobre o conceito de time (de futebol, de vôlei...). O resultado desta comunicação, sem que seja aplicada a mesma ontologia, será no mínimo confuso.

3 O PADRÃO FIPA

O objetivo deste Capítulo é apresentar os desenvolvimentos da organização FIPA. Estes desenvolvimentos adquiriram extrema importância no atual contexto da construção de sistemas multiagente.

A FIPA (2004) é uma fundação internacional sem fins lucrativos voltada exclusivamente para a criação de padrões que tornem possível a implementação de agentes abertos e interoperáveis. As atividades de padronização promovidas por ela são centradas na criação e manutenção de especificações.

A secção seguinte apresenta as especificações FIPA. Na seqüência, a linguagem FIPA-ACL, principal desenvolvimento da FIPA, é apresentada, assim como o ambiente multiagente proposto por esta organização. Por fim, é apresentado o *framework* FIPA-OS (2004), que através de mecanismos de extensão permite a construção de agentes compatíveis com os padrões FIPA.

3.1 Especificações FIPA

As especificações produzidas pela FIPA são divididas em cinco áreas distintas: 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

As Aplicações de Sistemas Multiagente são 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

Fonte: FIPA, 2004.

3.1.2 Arquitetura Abstrata para Sistemas Multiagente

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

É incluída, 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

Fonte: FIPA, 2004.

3.1.3 Comunicação entre Agentes

O maior avanço e detalhamento 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 (ver secção 2.5.2).
- **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
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

Fonte: FIPA, 2004.

3.1.4 Gerenciamento de Agentes

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

A plataforma de Gerenciamento de Agentes descrita no documento mostrado pela Tabela 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

Fonte: FIPA, 2004.

3.1.5 Transporte de Mensagens entre Agentes

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

Esta categoria está divida em:

- **Representações da ACL:** define diferentes formas de representar uma mensagem ACL;
- **Representações de Envelope:** define diferentes formas de representar um envelope;

- **Protocolos de Transporte:** define formas de transporte de mensagens ACL para diferentes protocolos de redes.

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

Tabela 3.5: Especificações FIPA: Transporte de Mensagens entre Agente

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

Fonte: FIPA, 2004.

3.2 FIPA-ACL

A Linguagem de Comunicação de Agentes sugerida pela FIPA é a FIPA-ACL (*FIPA – Agent Communication Language*). Apresentada inicialmente 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 da linguagem FIPA-ACL 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-".

Como exemplo, o agente i pergunta se o agente j está registrado no domínio d1:

```
(query-if
      :sender (agent-identifier :name i)
      :receiver (set (agent-identitfier :name j))
      :content "((registered (server d1) (agent j)))"
      :reply-with r09
      )
```

Figura 3.1: Mensagem FIPA-ACL (query-if)

O agente j responde que não:

```
(inform
      :sender (agent-identifier :name j)
      :receiver (set (agent-identifier :name i))
      :content "((not (registered (server d1) (agent j))))"
      :in-reply-to r09
      )
```

Figura 3.2: Mensagem FIPA-ACL (inform)

Os parâmetros nativos da FIPA-ACL são divididos em cinco categorias: tipo do ato comunicativo, participantes 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, indicado pelo parâmetro *content*, referencia a informação sobre a qual o ato comunicativo se aplica. Em geral, o conteúdo pode ser expresso em qualquer linguagem. A linguagem utilizada na representação do conteúdo pode ser declarada no parâmetro *language*. O projeto FIPA define e sugere algumas linguagens de representação de conteúdo padrões, tal como a linguagem SL.

Cada mensagem FIPA-ACL carrega um ato comunicativo, o qual representa a vontade 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 definidos na Teoria dos Atos da Fala de Searle (1981). São, entretanto, denominados de atos comunicativos para deixar clara a vinculação com a comunicação entre agentes (computacionais) e não com a comunicação entre seres humanos.

A Tabela 3.7 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.
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.
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 Modelo de Gerenciamento de Agentes

O modelo de gerenciamento de agentes provê o ambiente onde um agente FIPA existe e opera. Ele estabelece um modelo lógico de referência (Figura 3.3) para criação, registro, localização, comunicação, migração e desativação de agentes na plataforma FIPA.

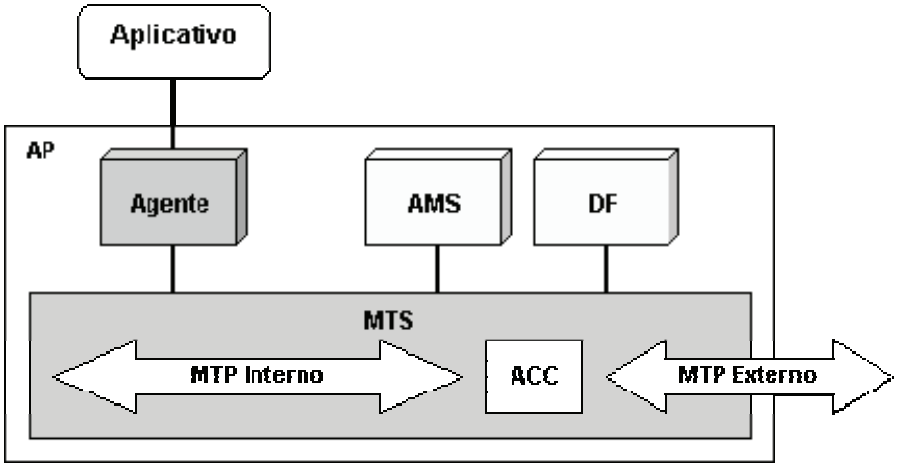

Figura 3.3: Modelo de Referência para Gerenciamento de Agentes FIPA (FIPA, 2004)

Um **Agente** é o fator fundamental numa plataforma de agentes. Ele combina 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 único AMS em uma plataforma de agentes. O

AMS 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 sistema multiagente em questão e não é objeto de padronização da FIPA.

É importante lembrar que estas especificações mostram o conjunto lógico para desenvolvimento da plataforma de agentes, mas não implicam na configuração física da plataforma de agentes. Assim, uma mesma plataforma pode estar residente em vários computadores.

3.4 O *Framework* FIPA-OS

A FIPA preocupa-se apenas em definir e especificar os elementos necessários para a construção de sistemas multiagente *interoperáveis*. Assim, nenhuma implementação é desenvolvida pela própria FIPA, essa tarefa é desempenhada por outras fundações ou empresas.

Existem várias implementações disponíveis das especificações FIPA. As plataformas de maior destaque atualmente são o JADE (BELLIFEMINE, 1999), o Zeus (NWANA, 1999) e o FIPA-OS (FIPA-OS, 2004), 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.

3.4.1 Visão geral

O FIPA-OS (2004) foi desenhado para ser compatível com os padrões FIPA. Assim, o modelo de referência FIPA, discutido anteriormente na secção 3.3, define a base da distribuição FIPA-OS: DF, AMS e MTS. A linguagem de comunicação de agentes é a FIPA-ACL.

Além dos 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 (ver secção 3.4.2);
- Interfaces abstratas;
- Ferramentas de diagnóstico e visualização.

O FIPA-OS é totalmente construído em Java. A 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.

3.4.2 Componentes FIPA-OS

Segundo (FIPA-OS, 2004), a arquitetura do FIPA-OS, mostrada na Figura 3.4, possui componentes de três tipos: 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.

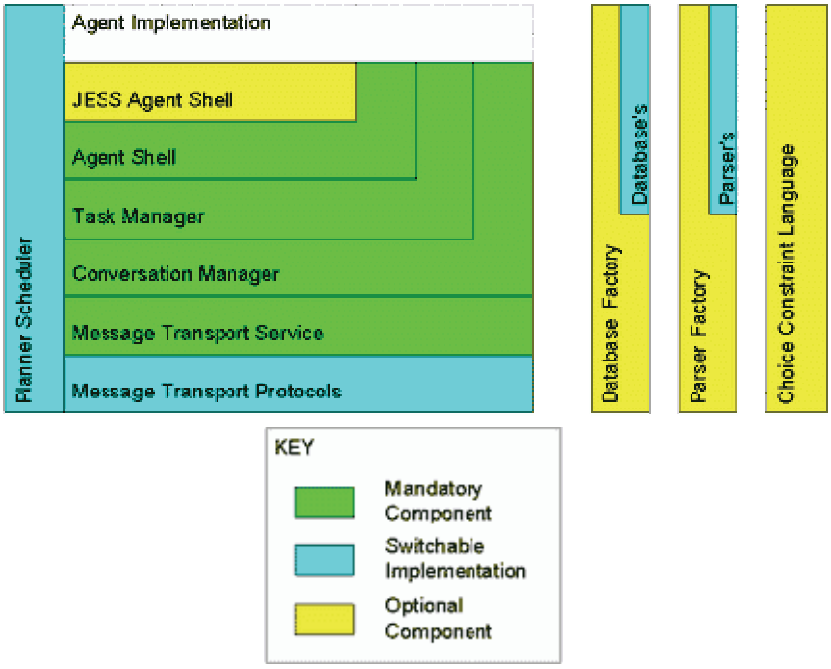

Figura 3.4: Componentes do FIPA-OS (FIPA-OS, 2004)

O ***Agent Shell*** é um componente obrigatório que disponibiliza uma forma facilitada para a implementação de agentes FIPA-OS através de conjunto de classes bases extensíveis;

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

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

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. Acompanham o FIPA-OS as implementações:

- **RMI:** O transporte através de RMI é baseado na tecnologia RMI da Sun Microsytems disponível como parte do Java 1.1 e do J2SE. O RMI utiliza a tecnologia de serialização de objetos para codificar as mensagens e, portanto, não é compatível com agentes escritos em outras linguagens de programação diferentes da linguagem Java. Entretanto, isso também significa que é um meio muito eficiente para a comunicação entre agentes escritos em Java (atualmente todos os agentes da plataforma FIPA-OS). Por isso, ele é utilizado como o MTP Interno padrão;
- **IIOP:** O transporte através de IIOP, geralmente utilizado como MTP Externo, é baseado na tecnologia CORBA da Sun MicroSystems disponível como parte do J2SE. Ele é potencialmente compatível com agentes escritos em qualquer linguagem que suporte a tecnologia CORBA e está de acordo com a especificação FIPA IIOP MTP.

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. Acompanham o FIPA-OS as implementações:

- ***MemoryDatabase:*** Implementação de um banco de dados na memória; Toda informação é perdida quando a máquina virtual Java no qual o agente está executando é desligada;
- ***SerialisationDatabase:*** Provê um mecanismo simples de banco de dados que utiliza a tecnologia de serialização de objetos, da linguagem Java.

O Componente ***Parser*** é um componente permutável que disponibiliza implementações de *parsers* utilizados pelo *Parser Factory;* Acompanham o FIPA os p*arsers* para as linguagens SL, ACL, XML e RDF.

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

3.4.3 Especificações FIPA suportadas pelo FIPA-OS

A Tabela 3.8 mostra a relação completa das especificações FIPA suportadas pelo FIPA-OS.

Tabela 3.8: Especificações FIPA suportadas pelo FIPA-OS

Comunicação entre Agentes	
XC00061D	FIPA ACL Message Structure Specification;
Comunicação entre Agentes – Protocolos de Interação	
XC00025D	FIPA Interaction Protocol Library Specification;
XC00026E	FIPA Request Interaction Protocol Specification;
XC00027E	FIPA Query Interaction Protocol Specification;
XC00028E	FIPA Request When Interaction Protocol Specification;
XC00029E	FIPA Contract Net Interaction Protocol Specification;
XC00030E	FIPA Iterated Contract Net Interaction Protocol pecification;
XC00031E	FIPA English Auction Interaction Protocol Specification;
XC00032E	FIPA Dutch Auction Interaction Protocol Specification;
XC00033E	FIPA Brokering Interaction Protocol Specification;
XC00034E	FIPA Recruiting Interaction Protocol Specification;
PC00035D	FIPA Subscribe Interaction Protocol Specification;
XC00036E	FIPA Propose Interaction Protocol Specification;
Comunicação entre Agentes – Atos comunicativos	
XC00037G	FIPA Communicative Act Library Specification;
Comunicação entre Agentes – Linguagens de Conteúdo	
XC00007A	FIPA Content Languages Specification;
XC00008D	FIPA SL Content Language Specification;
XC00009A	FIPA CCL Content Language Specification;
XC00011A	FIPA RDF Content Language Specification;

Gerenciamento de Agentes	
XC00023G	FIPA Agent Management Specification;
Transporte de Mensagem entre Agentes	
XC00067C	FIPA Message Transport Service Specification;
Transporte de Mensagem entre Agentes - Representações da ACL	
XC00069C	FIPA ACL Message Representation in Bit-Efficient Specification;
XC00070F	FIPA ACL Message Representation in String Specification;
Transporte de Mensagem entre Agentes – Protocolos de Transporte	
XC00075D	FIPA Agent Message Transport Protocol for IIOP Specification;
XC00084C	FIPA Agent Message Transport Protocol for HTTP Specification;
Transporte de Mensagem entre Agentes – Representações de Envelope	
XC00085G	FIPA Agent Message Transport Envelope Representation in XML Specification;
Aplicações de Sistemas Multiagente	
XC00079A	FIPA Agent Software Integration Specification.

Fonte: FIPA-OS, 2004.

3.4.4 Requisitos de sistema

O *Hardware* mínimo necessário para a execução da plataforma FIPA-OS é:

- Processador tipo Pentium 166Mhz;
- Memória de 64 MB;
- Espaço de 4 MB no disco rígido.

Quanto aos requisitos de *software*, o FIPA-OS necessita de vários softwares de terceiros obrigatórios e alguns opcionais. A versão do FIPA-OS para Java 2 exige que esteja instalada um ambiente de execução Java 2 (J2RE) para a sua execução e o JDK1.2.2 ou superior para a compilação dos agentes. Já versão para Java 1 exige que esteja instalada uma máquina virtual Java compatível com o JDK1.1 e, com objetivo de fornecer as mesmas funcionalidades entre as duas versões, necessita das classes do JDK1.1 *Collection Classes* (disponível em http://java.sun.com/products/javabeans/infobus/) e do JDK1.1 *Swing Classes* (disponível em http://java.sun.com/products/jfc/).

Os outros *softwares* obrigatórios para as duas versões são distribuídos juntamente com o FIPA-OS. São eles: SiRPAC & SAX 1.0 XML Parser, Enhydra XML Databinding Sub-System e JDOM Parser.

Opcionalmente, pode ser utilizado um servidor *Web* (como o Apache HTTP Server, disponível em http://www.apache.org) para suportar a distribuição inicial de endereços de transporte para plataformas remotas de agentes, ele habilita a interoperabilidade entre as plataformas. Não é necessário um servidor dedicado, pois ele somente é requerido no processo de carga da plataforma de agentes.

O Java Expert System Shell 5.0 (JESS, disponível em http://herzberg.ca.sandia.gov/jess/) é um sistema que disponibiliza aos agentes FIPA-OS a capacidade de raciocínio baseado em regras. Ele é opcional, mas obrigatório para o uso do *agent Shell* JessAgent.

Para aumentar a segurança, opcionalmente pode ser instalado o Java Secure Sockets Extension (disponível em http://java.sun.com/products/jsse/), que é uma extensão da API padrão Java que adiciona suporte SSL a qualquer plataforma Java 2.

3.4.5 Considerações sobre o FIPA-OS

Uma característica muito interessante do FIPA-OS é a sua capacidade de dividir um agente em *Tasks*, as quais são gerenciadas pelo *Task Manager*. O fluxo de execução de uma *task* é baseado em eventos. Várias *tasks* são podem ser executadas ao mesmo tempo, pois elas executam em *threads* independentes. Além disso, elas podem enviar e receber mensagens.

Outra característica importante é a presença do *Agent Shell*. A classe FIPAOSAgent facilita a implementação dos agentes através de sua extensão. Ela carrega automaticamente os componentes obrigatórios que compõem 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 CONCEITOS DE OBJETOS DE APRENDIZAGEM

Este Capítulo tem por objetivo analisar a abordagem de objetos de aprendizagem e com suas tecnologias de apoio.

O Capítulo começa com uma discussão acerca das motivações que levaram ao desenvolvimento desta abordagem. Logo após, serão apresentados os seus fundamentos teóricos. Por fim, serão apresentadas algumas tecnologias que dão suporte a ela.

4.1 Justificativa para objetos de aprendizagem

Ao longo dos anos, vários pesquisadores têm proposto a utilização da informática em ambientes educacionais. As abordagens para esta utilização são bastante variadas, como por exemplo: sistemas que auxiliam o professor nas suas tarefas diárias, tal como confecção de avaliações; enciclopédias virtuais, nas quais os alunos podem fazer pesquisas escolares e aprimorar os seus conhecimentos; sistemas de tutoria 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.

O grande desenvolvimento da área de Redes de Computadores, em especial da *internet*, fez que com aparecessem sistemas computacionais de ensino e aprendizagem também com o propósito de serem utilizados para a Educação a Distância[1] (EAD).

Nesse mesmo sentido, um grande número de instituições de ensino está desenvolvendo material educacional para cursos *on-line*. Tendo em vista 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. Surgem os Objetos de Aprendizagem, uma abordagem que pretende reduzir significativamente o tempo e o custo gastos com o desenvolvimento de material educacional para cursos *on-line*.

Um Objeto de Aprendizagem caracteriza-se por ser uma entidade de material educacional reutilizável, ou seja, que pode ser reaproveitada em cursos diversos, diversas vezes. Entre os benefícios dessa abordagem está a economia. Essa economia é

[1] Segundo Keegan (1991), a EAD visa o desenvolvimento de ambientes e metodologias que propiciem o aprendizado remoto, isto é, que um ou mais alunos possam vivenciar experiências de aprendizagem em local fisicamente diferente do qual o ambiente e os recursos instrucionais se encontram.

justificada através da reuso, pois, não é necessário gastar tempo e dinheiro no desenvolvimento de um novo material instrucional se ele já estiver disponível.

No entanto, para ser reusado, um objeto de aprendizagem deve ser compatível entre diferentes ambientes educacionais, ou seja, ele deve 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, a essas estruturas chama-se 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*).

No sentido de 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.

4.2 Fundamentos

Podemos analisar os objetos de aprendizagem de 3 modos distintos: através de uma discussão sobre o conceito de objeto de aprendizagem; através de metáforas que permitam visualizar concretamente o que é um objeto de aprendizagem; e, através de propriedades que são desejáveis e que capacitam um objeto de aprendizagem. Esta secção apresentará cada uma dessas visões.

4.2.1 Discussão Conceitual

Não existe um conceito único do que seja um objeto de aprendizagem. Vários pesquisadores definem de forma diferente. 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:

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

De acordo com essa definição qualquer coisa que possa estar relacionada com o processo de aprendizagem pode ser considerada como sendo um objeto de aprendizagem. Por exemplo, uma carteira escolar, um teclado de um computador, entre outros objetos mundanos que apenas servem de suportes para a aprendizagem. Segundo os autores, o problema desta definição é que tudo o que se pode inferir dela é que um objeto de aprendizagem é “alguma coisa” usada em algum tipo de ambiente para promover a aprendizagem.

Sosteric & Hesemeier (2002) prosseguem, afirmando que todos os trabalho feitos na área de objetos de aprendizagem se referem a objetos digitais. Logo, 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 preocupação 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 estejam enquadradas em algum contexto, que neste caso é o título das figuras ou alguma explicação acerca delas, não produzem experiências de aprendizagem.

Assim, nem todo arquivo digital, ou objeto digital, é um objeto de aprendizagem, mesmo que todo objeto de aprendizagem seja um objeto digital. Um objeto digital só pode ser considerado um objeto de aprendizagem se tiver associado a ele um contexto. Por exemplo, um software utilitário que lista arquivos pode ser um objeto de aprendizagem. Mas, somente o será se alguém o desejar utilizar desta forma e der um contexto educacional a ele.

Tendo isso em mente, Sosteric & Hesemeier (2002) definem objetos de aprendizagem como:

> Um objeto de aprendizagem é um arquivo digital (imagem, filme, etc.,) que pretende ser utilizado para propósitos educacionais e que inclui, internamente ou via associação, sugestões sobre o contexto apropriado no qual deve ser utilizado. (SOSTERIC, 2002)

No artigo citado anteriormente, ainda é levantada uma discussão a respeito das definições que associam os objetos de aprendizagem com os objetos do Paradigma de Orientação a Objetos. São os casos de Quinn (2000), Robson (1999) e Wiley (2000a), cujas definições seguem 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 auto-contidos, que podem conter referências a outros objetos, e que podem ser combinados ou seqüenciados para formar interações educacionais longas. Esses pedaços de conteúdo educacional pode 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 discussão 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).

Nos artigos que levantam essa discussão da relação entre os objetos e os objetos de aprendizagem, citados acima, em nenhum momento foi mencionado o modelo SCORM (ADL 2004), o qual será tratado mais adiante e que, podemos dizer, conserva muitas características da orientação a objetos.

Segundo Mohan e Greer (2003), alguns autores afirmam que um objeto de aprendizagem deve facilitar um objetivo de aprendizagem simples.Com isso em mente, esses autores definiram os objetos de aprendizagem da seguinte forma:

> Um objeto de aprendizagem é um recurso digital que facilita um objetivo de aprendizagem simples e que pode ser reutilizado em diferentes contextos (MOHAN, 2003).

O que se pode concluir de todas as definições apresentadas acima é que o coração da abordagem de objetos de aprendizagem, ou melhor, o aspecto que todas elas possuem em comum, é justamente a idéia de “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. Neste caso, a locução “recurso digital” utilizada por Mohan e Greer (2003) parece ser mais pertinente. Ainda, a definição de Mohan e Greer inclui que um objeto de aprendizagem deve satisfazer um objetivo de aprendizagem simples. Essa inclusão é boa em parte, pois achamos que um objeto de aprendizagem pode satisfazer também objetivos de aprendizagem complexos.

Com base nesta discussão, para efeitos deste trabalho definimos um objeto de aprendizagem da seguinte forma: Um objeto de aprendizagem é um recurso digital (imagem, filme, *website*, simulação, etc.,) utilizável para a satisfação de algum objetivo educacional e que possui, internamente ou via associação, sugestões sobre o seu contexto de utilização.

4.2.2 Metáforas

Afora a definição conceitual, algumas metáforas foram propostas para tentar explicar melhor o que seria um objeto de aprendizagem. A primeira analogia que surgiu foi com o jogo Lego, aquele brinquedo onde pequenos blocos podem ser combinados para formar algum objeto maior. As principais características deste jogo é que: qualquer bloco pode ser combinado com qualquer outro; os blocos podem ser associados da maneira que se quiser; é tão simples que qualquer pessoa pode jogar.

A analogia entre objetos de aprendizagem e blocos Lego, feita por Hodgins e Conner (2000), apesar de bastante interessante apresenta uma visão bastante simplista e não deve retratar a realidade com os objetos de aprendizagem (WILEY, 2000a). Se essas fossem as propriedades de um objeto de aprendizagem não se conseguiria certificar a qualidade educacional dessa abordagem. Isso porque, educacionalmente, nem todos os blocos são realmente combináveis com qualquer outro, Ainda, nem todo mundo pode montar um curso simplesmente agrupando objetos de aprendizagem. Um curso deve ser construído cuidadosamente de acordo com algum planejamento educacional e por pessoas capazes.

Pensando nisso, Wiley (2000a) introduziu a metáfora do átomo. Essa metáfora parece ser bastante convincente. Segundo o autor, um objeto de aprendizagem é análogo a um átomo porque: nem todo átomo é combinável com qualquer outro átomo; átomos só podem ser agregados em certas estruturas prescritas por sua própria estrutura interna; é necessário algum treinamento para agregar átomos.

4.2.3 Propriedades

A principal característica, e o alvo de toda a tecnologia de objetos de aprendizagem, é permitir a reusabilidade do material instrucional, ou seja, permitir que uma entidade de material instrucional seja reaproveitada em cursos diversos, diversas vezes. Para alcançar essa característica, um objeto de aprendizagem deve apresentar algumas propriedades. A literatura provê um grande número delas. As mais citadas são: modularidade; interoperabilidade; e, capacidade de ser descoberto (FRIESEN, 2001).

4.2.3.1 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. Eles funcionam como um catálogo de biblioteca, quem contém informações sobre os livros que a biblioteca possui.

A estrutura de um objeto de aprendizagem está intimamente ligada aos metadados que o descrevem. Alguns autores afirmam 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.

Além de permitir a um objeto de aprendizagem ser descoberto, padrões de metadados são bastante importantes também para a interoperabilidade, como será visto na secção 4.2.3.3. O padrão de metadados a ser utilizados neste trabalho será apresentado na secção 4.3.

4.2.3.2 Modularidade

Tendo em vista a *reusabilidade*, um objeto de aprendizagem deve também ser modular o suficiente para que possa se adaptar em cursos diversos. De acordo com Longmire (2000), um objeto de aprendizagem deve ser: "independente de posição, não seqüencial, coerente e unitário". Outros autores descrevem a mesma idéia utilizando termos um pouco diferentes. Roschelle et al (1998) diz que um objeto de aprendizagem deve ser adaptável para necessidades não previstas, sem a ajuda do desenvolvedor original.

A questão sobre a modularidade leva a uma discussão 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.*

Assim, 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, 200b).

Wiley (2000b) diz que de um ponto de vista instrucional, a decisão sobre a granularidade pode ser vista como um problema de "escopo". O autor prossegue, afirmando que, visto desse ponto de vista, as maiores questões sobre o emprego de objetos de aprendizagem, suas granularidades e suas combinações caem em duas considerações bastante conhecidas pelos projetistas instrucionais: escopo e sequência.

4.2.3.3 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 prática, 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.

4.3 Metadados para Objetos de Aprendizagem

Os metadados, cujo significado literal corresponde a “dados sobre dados”, são utilizados para descrever e categorizar os objetos de aprendizagem. Eles funcionam como um catálogo de biblioteca, quem contém informações sobre os livros que a biblioteca possui.

Muitas instituições vêm trabalhando na definição de padrões de metadados. Entre essas podemos citar o *IMS Global Learning Consortium* (IMS) (2004), a Dublin Core Metadata Initiative (DCMI, 2004), a CanCore Metadata Initiative (CANCORE, 2004), LTSC IEEE (2004), entre outras.

Neste trabalho utilizaremos o Padrão LOM desenvolvido pelo LTSC da IEEE. Este padrão será apresentado na secção seguinte.

4.3.1 O Padrão LOM 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 4 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.

4.4 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 Barron (2000), existem muitas características que são únicas a sistemas individuais de LMS. No entanto, o "coração" de um sistema deste tipo são as capacidades de coordenar registros em cursos, planejar, rastrear e avaliar os alunos.

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.

Outro exemplo de LMS é o AulaNet (2005). A AulaNet foi desenvolvido no Laboratório de Engenharia de Software da PUC-Rio. A ferramente agrega serviços de comunicação por grupos de interesse, grupos de discussão, para contato com o professor e para debate. Ainda permite que sejam feitas avaliações, que sejam armazenados materiais didáticos e provê outras ferramentas mais gerais, tais como: tutorial sobre Internet, página pessoal dos alunos e serviços de busca.

Outros exemplos de LMSs são Blackboard (2005), o TopClass (2005), o VirtualU (2005).

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. Este padrão será analisado na próxima secção.

4.4.1 O Padrão DMCOC

O padrão *Data Model for Content Object Communication* (DMCOC) é gerenciado pelo grupo de trabalho de número 11 (*Workgroup 11*) do LTSC IEEE (LTSC, 2004) 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, um LMS, entrega conteúdo digital, um objeto de conteúdo, aos estudantes. Um objeto de conteúdo é uma coleção de conteúdo digital que é apresentada a 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.

4.5 Repositórios de Objetos de Aprendizagem

Repositórios de Objetos de Aprendizagem é o nome dado a sistemas que objetivam disponibilizar de forma centralizada a busca, o acesso e a recuperação de objetos de aprendizagem.

Campos (2003) afirma que esses repositórios podem ser vistos como facilitadores na montagem de novos cursos baseados em objetos de aprendizagem. Segundo esta autora, as funcionalidades de um repositório de objetos de aprendizagem podem incluir: armazenamento dos metadados do LOs, armazenamento dos próprios LOs, suporte à modelagem conceitual de cursos; integração com LMSs; interface para gerenciamento de objetos de aprendizagem e seus metadados; mecanismos de segurança; e serviços gerais como *backup* e *restore.*

Um dos principais repositórios de objetos de aprendizagem é o CAREO (2005). O CAREO é um projeto suportado pela Alberta Learning e pela CANARIE, cujo objetivo é criar uma coleção multidisciplinar de material educacional, desenvolver mecanismos de busca sobre esta coleção e disponibilizá-la para professores no contexto educacional canadense.

Outro bom exemplo de um repositório de objetos de aprendizagem é o MERLOT (2005). Ele é um recurso livre e aberto projetado primariamente para professores e alunos de educação superior.

O CESTA (Coletânea de Entidades de Suporte ao uso de Tecnologia na Aprendizagem) (2005) é um projeto desenvolvido pela equipe do Programa Pós-Graduação Informática na Educação e do Centro Interdisciplinar de Novas Tecnologias na Educação da UFRGS. Ele foi idealizado com vistas a sistematizar e organizar o registro dos objetos educacionais que vinham sendo desenvolvidos nestes dois grupos.

5 TRABALHOS RELACIONADOS

Neste Capítulo serão apresentados alguns trabalhos que de alguma forma se relacionam ao que foi desenvolvido nesta pesquisa. Começaremos por analisar o MAIDE, um modelo de sistema multiagente desenvolvido com o objetivo de permitir a construção de cursos on-line através da reutilização de técnicas educacionais. Logo após, é apresentado o SCORM, que é o modelo mais utilizado atualmente para o compartilhamento de material educacional. Após, será apresentada uma abordagem baseada em objetos para a construção de objetos de aprendizagem. Por fim, dois trabalhos são comentados por utilizar a mesma denominação “Objetos Inteligentes de Aprendizagem”.

5.1 MAIDE

O projeto MAIDE (Modelagem de Ambientes Inteligentes de Aprendizagem) visa o estudo e a implementação de modelos de Ambientes Inteligentes Distribuídos de Aprendizagem baseados na abordagem de Arquiteturas Multiagente voltado para a implementação de programas de capacitação de recursos humanos baseados em Treinamento Virtual.

A arquitetura multiagente adotada por esse projeto (SILVEIRA 2003) é composta por duas famílias de agentes: um agente responsável pelo modelo do aluno (Agentes do Modelo de Aluno) e um conjunto de agentes responsáveis por tarefas relacionadas com táticas de ensino (Agentes Pedagógicos).

Os agentes pedagógicos desempenham atividades de ensino como exemplos, exercícios, demonstrações, simulações, entre outras. Eles podem ser reaproveitados de um curso para outro. Para tal, é suficiente que se construa material educacional para eles e que a base de conhecimento do sistema seja alimentada adequadamente. Baseado no material educacional disponível, um agente pedagógico é capaz de gerar experiências de aprendizagens para os alunos. A formação completa do aluno é obtida através da cooperação entre o conjunto de agentes pedagógicos definidos para um curso.

A principal diferença entre os objetos inteligentes de aprendizagem, componentes centrais na nossa abordagem, e os agentes pedagógicos do MAIDE está na filosofia de desenvolvimento de um e de outro. No MAIDE os agentes são especializados em táticas de ensino, sem se importar com o conteúdo que será exibido e sem terem, portanto, capacidade de ser descoberto em função deste conteúdo. A filosofia de desenvolvimento de um ILO, por sua vez, deve primar por esta capacidade. Assim, um ILO deve ser

desenvolvido em função do seu conteúdo educacional, podendo utilizar-se de uma ou mais táticas de ensino.

A arquitetura do MAIDE influenciou bastante a arquitetura deste trabalho, tanto que consideramos os objetos inteligentes de aprendizagem como uma evolução dos agentes pedagógicos do MAIDE, como apresentado em Gomes (2004) e em Silveira (2004).

5.2 SCORM

O *Sharable Content Object Reference Model* (SCORM) (ADL, 2004) é o modelo atual mais completo para o compartilhamento de conteúdo educacional. Nele, recursos de aprendizagem são agrupados em pacotes. Cada pacote possui um arquivo que descreve o conteúdo do pacote e como este deve ser usado. A comunicação entre um pacote SCORM e os ambientes de aprendizagem é feita através de uma API (*Application Program Interface*). As chamadas à API são efetuadas em *JavaScript* e podem envolver informações sobre os estudantes. Essas informações são armazenadas em um Modelo de Dados (*Data Model)*, o qual permite a comunicação padronizada entre os pacotes e os ambientes de aprendizagem. Somente os pacotes são capazes de chamar os métodos da API.

Um dos diferencias entre os pacotes de conteúdo do SCORM e os ILOs se concentra na forma como é feita a comunicação entre os objetos e os ambientes de aprendizagem. No SCORM a comunicação é feita no espírito da orientação a objetos, com chamadas de métodos e passagem de parâmetros. Nos ILOs a comunicação é feita via troca de mensagens codificadas através de uma Linguagem de Comunicação de Agentes e de uma Linguagem de Conteúdo, como comentado na secção 6.1.

Outro diferencial corresponde à iniciativa de comunicação, que nos ILOs pode ser tomada tanto pelo próprio ILO como pelo ambiente de aprendizagem. Já no SCORM, devido a limitações tecnológicas, somente o objeto é capaz de tomar a iniciativa na comunicação.

No SCORM, um objeto não é capaz de se comunicar com outros objetos no sentido de coordenar ações. No modelo ILO esta possibilidade pode ser contemplada.

E, por fim, o comportamento do objeto no modelo SCORM é estático, ditado totalmente pelo arquivo de manifesto. O comportamento de um ILO é dinâmico, guiado pelo seu projeto e ditado pelas suas regras de comportamento e pelo seu conhecimento, que podem ser atualizados.

5.3 *Object Learning Object*

O trabalho de Mohan e Brooks (MOHAN, 2003) propõe um modelo de objetos de aprendizagem baseado em objetos. Nesta arquitetura existe uma classe LearningObject, que é a superclasse para todos os objetos de aprendizagem. Os LearningObject tem propriedades como coleções de referências a instâncias de metadados (por exemplo, Dublin Core, CanCore e LOM). Eles também contêm um conjunto de diferentes objetos Version, cada qual correspondendo a uma versão diferente dos recursos de aprendizagem que formam o objeto.

Uma propriedade importante é a existência de um objeto Context, que contém informações contextuais, tais como tipos de alunos para o qual o objeto é apropriado e as estratégias de aprendizagem empregadas. Outra propriedade é a existência de um

objeto Combination, o qual contém informações sobre as características do objeto de aprendizagem e que o permite ser combinado com outros objetos de aprendizagem. LearningObject também contém mapas conceituais, que descrevem como o objeto de aprendizagem é posicionado no estrutura conceitual do domínio, e os objetivos instrucionais que devem ser alcançados por um aluno que o estudou com sucesso.

A diferença deste trabalho em relação à nossa abordagem reside na abordagem de desenvolvimento utilizada. Segundo alguns autores, o paradigma de agentes é visto como uma evolução do paradigma de orientação a objetos. Seguindo esta idéia, podemos dizer que um objeto de aprendizagem baseado em agentes pode ser mais avançado do que um baseado em objetos. Além disso, os objetos inteligentes de aprendizagem são capazes de implementar as funcionalidades que a abordagem de por P. Mohan & C. Brooks apresenta.

5.4 Outros trabalhos

A empresa Command Technologies (COMMAND, 2005) utiliza o termo “Objeto Inteligente de Aprendizagem” para a idéia de desmembrar um Sistema Tutor Inteligente em vários sistemas menores. Ela afirma que estes sistemas menores serão reusáveis em ambientes de aprendizagem projetados para ensinar habilidades relacionadas. Esta idéia está sendo implementada baseada no modelo SCORM, cujas diferenças em relação à nossa abordagem foram pautadas acima.

A empresa BrainX (BRAINX, 2005) também está utilizando o termo “Objetos Inteligentes de Aprendizagem”. Porém, enquanto escrevíamos esta dissertação, não obtivemos informações sobre qual o contexto desta utilização.

Em 18 de maio de 2005, uma pesquisa no mecanismo de busca Google (http://www.google.com) retornou 50 resultados quando inserido o termo “intelligent learning objects”. Nenhum destes resultados referenciava alguma abordagem semelhante ao que está sendo desenvolvido nesta pesquisa, com exceção dos que apontavam para a própria. O mesmo aconteceu quando pesquisado o termo “intelligent learning object”, para o qual foram retornados 15 resultados. Para os termos Objeto “Inteligente de Aprendizagem” e “Objetos Inteligentes de Aprendizagem”, foram retornados apenas resultados que apontavam esta pesquisa.

6 OBJETOS INTELIGENTES DE APRENDIZAGEM

Para a abordagem assumida neste trabalho, 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.

O importante é entender que, 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 (ver Capítulo 4).

Assim, a base tecnológica que fundamenta esta abordagem constitui-se em uma integração de tecnologias desenvolvidas para Objetos de Aprendizagem, para Agentes e para Sistemas Multiagente. A Figura 6.1 apresenta uma diagramação desta integração.

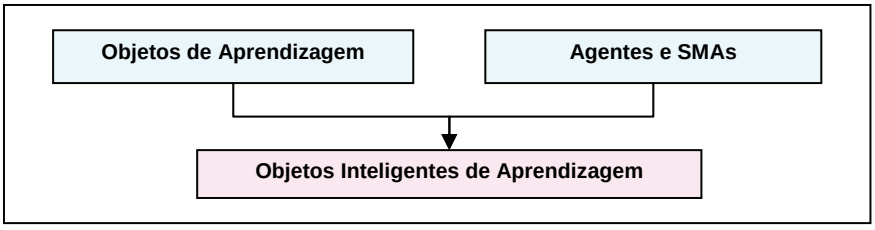

Figura 6.1: Bases tecnológicas dos Objetos Inteligentes de Aprendizagem

O objetivo deste Capítulo é formular a base conceitual desta abordagem. Serão levantadas as suas justificativas, potencialidades e requisitos. Ainda, será apresentada uma sociedade multiagente proposta com o objetivo de prover suporte a tal abordagem, bem como a modelagem do processo de comunicação entre os agentes desta sociedade.

6.1 Por que transformar objetos de aprendizagem em agentes?

A utilização do paradigma de agentes para a construção de objetos de aprendizagem apresenta inúmeras potencialidades. Uma delas diz respeito aos métodos de comunicação adotados pelos agentes. 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 atualmente, 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 bastante 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. Ainda, 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 possibilidade interessante está ligada à 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. Assim, é possível que o objeto de aprendizagem evolua, ele não é mais estático como nos modelos atuais. As possibilidades de aprendizagem são enormes, tais como: 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. 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 bastante 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.

Em suma, as potencialidades do uso de “agentes objetos de aprendizagem”, os ILOs, são bastante 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.

6.2 Requisitos de um Objeto Inteligente de Aprendizagem

A definição para Objeto Inteligente de Aprendizagem adotada neste trabalho permite que surjam diversos tipos de ILOs, e com as mais variadas arquiteturas. Não é o objetivo deste trabalho restringir tais existências. No entanto, deve ser assegurado que elas não firam as características básicas tanto de agentes quanto de objetos de aprendizagem.

Esta secção apresenta os requisitos que devem ser considerados no projeto de um Objeto Inteligente de Aprendizagem. Tais requisitos foram separados em dois grupos, que correspondem às duas tecnologias base desta abordagem: requisitos quanto aos conceitos de objetos de aprendizagem e quanto aos conceitos de agentes.

6.2.1 Quanto aos conceitos de objetos de aprendizagem

Sob o ponto de vista de objetos de aprendizagem, deve-se sempre buscar as características que podem tornar um objeto de aprendizagem *reusável*.

Primeiramente, quanto menor e mais simples a tarefa pedagógica executada pelo ILO, mais adaptável e flexível será a experiência de aprendizagem gerada por ele. Este cuidado visa habilitar a sua modularidade.

A “capacidade de ser descoberto”, sob o ponto de vista de objetos de aprendizagem, significa poder ser descoberto em função do seu conteúdo educacional. Ou seja, os outros agentes do sistema e usuários devem poder descobrir qual o conteúdo que o objeto carrega e como ele trabalha este conteúdo. Isso nos leva à utilização de informações de metadados. Para tal, adotaremos o padrão IEEE *1484.12.1 Standard for Learning Object Metadata* (LTSC, 2004).

Alguns objetos de aprendizagem são capazes de manter e manipular informações sobre os estudantes, tal como preferências e aspectos da interação dele com o objeto. Caso um Objeto Inteligente de Aprendizagem almeje manipular este tipo de informação, deve utilizar o padrão *IEEE 1484.11.1 Standard for Learning Techology – Data Model for Content Object Communication*. Este padrão foi definido para que objetos de conteúdo possam trocar informações com ambientes de aprendizagem baseados neles e se adapta perfeitamente aos nossos planos. Maiores detalhes podem ser encontrados na secção 4.4.1.

A *interoperabilidade* também é alcançada através da utilização de padrões. Neste trabalho, como pode ser visto acima, foram adotados dois padrões bastante aceitos pela comunidade da área. O padrão *IEEE 1484.12.1 Standard for Learning Object Metadata* e o padrão *IEEE 1484.11.1 Standard for Learning Techology – Data Model for Content Object Communication*.

Afora essas características, é imprescindível que um ILO não perca o seu fim pedagógico. Um ILO deve ser criado e utilizado no sentido de executar tarefas específicas que sejam capazes de gerar experiências de aprendizagem significativas através de interações com o estudante. Desta forma, o seu projeto deve envolver, sempre que possível, um especialista em conteúdo, um especialista em educação e um especialista em computação.

Em suma, enquanto objeto de aprendizagem, um ILO deve buscar a *reusabilidade*, que pode ser alcançada através das modularidade, capacidade de ser descoberto e *interoperabilidade*. Estas características, por sua vez, podem ser alcançadas através da utilização de padrões e da elaboração de um projeto pedagógico bem feito.

6.2.2 Quanto aos conceitos de agentes

O componente central de um ILO é o seu agente. Quanto a este quesito, adotamos as concepções de Wooldridge (1999). Tais autores vêem um agente como um sistema computacional granular que faz uso de recursos e que maximiza alguma medida global. Esta definição prima pela idéia de que um agente deve compor um sistema maior, realizando sub-tarefas de uma tarefa mais geral.

Assim, o agente de um ILO não deve tentar buscar a formação completa do aluno por si só. Deve-se ter em mente que esta formação será alcançada através da interação do aluno com a sociedade de ILOs. Esta idéia está diretamente relacionada com a propriedade de modularidade.

Para assegurar a *interoperabilidade* devem ser utilizados padrões que promovam esta propriedade. Neste trabalho optamos por trabalhar com o padrão FIPA, descrito em detalhe no Capítulo 3. Assim, o agente de um ILO deve ser compatível com o padrão FIPA, utilizando a linguagem FIPA-ACL para a troca de mensagens e a linguagem FIPA-SL para descrição do conteúdo das mensagens.

Ainda com vistas à *interoperabilidade*, o agente de um ILO deve utilizar, sempre que requerido, os diálogos comunicativos definidos neste trabalho.

A “capacidade de ser descoberto”, sob o ponto de vista de agentes, significa poder ser descoberto pelos outros agentes da sociedade em função da sua atividade, que no caso de um ILO é gerar experiências de aprendizagem reusáveis. A secção 6.4.3.5 trata deste ponto em especial.

A “capacidade de ser descoberto” em função do seu conteúdo educacional está mais relacionada com a área de objetos de aprendizagem e foi discutida na secção anterior. No entanto, cabe ressaltar aqui que os agentes devem conhecer os diálogos utilizados para a troca de informações de metadados, as quais habilitam esta propriedade. Eles estão descritos na secção 6.4.3.1. Os métodos utilizados pelo agente para representar internamente as informações de metadados não estão especificados, ficando a cargo do projetista do mesmo.

Caso um ILO manipule informações sobre os alunos e esteja interessado em trocar este tipo de informações com outros agentes (vide secção anterior), devem ser utilizados os diálogos definidos para este fim. Eles estão descritos na secção 6.4.3.2. Ressalta-se, mais uma vez, que o método de representação de conhecimento utilizado pelo agente para manipular estas informações não está especificado, ficando a cargo do seu projetista.

Em suma, a principal preocupação sob o ponto de vista de agentes é definir uma comunicação padronizada nos moldes dos padrões FIPA. Na secção 6.4, que trata da modelagem da comunicação, são apresentados os aspectos que devem ser obrigatoriamente observados pelos agentes definidos neste trabalho. A modelagem dos processos internos que regem o comportamento do agente e a escolha dos métodos de representação de conhecimento utilizados ficam a cargo do projetista do mesmo.

6.3 Sociedade Multiagente

Esta secção define uma sociedade multiagente que é capaz de fornecer apoio ao funcionamento de ILOs. Tal sociedade está composta por três tipos de agentes (Figura 6.2): ILOs, Agentes LMS e Agentes ILOR.

Um **Agente LMS** é uma abstração de um Sistema Gerenciador de Aprendizagem (LMS – *Learning Management System*) (ver secção 4.4). Sua responsabilidade é preocupar-se com as questões administrativas que envolvem um ambiente de aprendizagem que tenha nos ILOs as suas entidades geradoras de experiências de aprendizagem.

Um **Agente ILOR**, de *Intelligent Learning Object Repository*, é uma abstração de um Repositório de Objetos de Aprendizagem (LOR – *Learning Object Repository*) (ver secção 4.5). Sua função é manter dados que possibilitem a algum usuário encontrar ILOs que adaptem-se a alguma demanda específica.

As entidades **DF** e **AMS** fazem parte do Modelo FIPA. Elas disponibilizam respectivamente um serviço de diretórios e um conjunto de serviços para o gerenciamento do ambiente. Embora não sejam necessariamente definidas como agentes, a maioria das implementações reais dos padrões FIPA as implementam como tal.

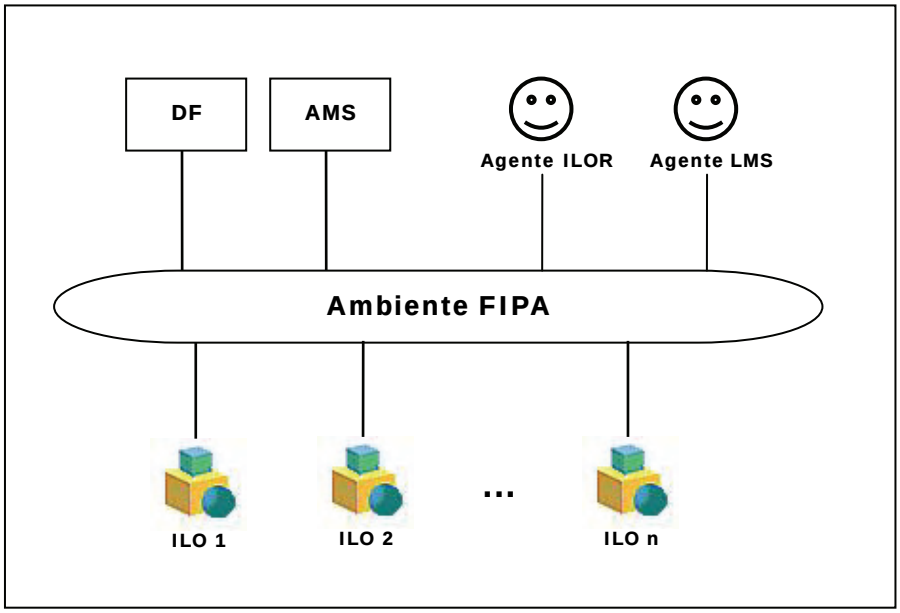

Figura 6.2: Sociedade multiagente proposta

O ambiente no qual os agentes devem estar inseridos é necessariamente um ambiente compatível com os padrões FIPA. Isso assegura que este ambiente seja capaz de disponibilizar toda a infra-estrutura necessária para a comunicação entre os agentes, bem como todas as funcionalidades administrativas para a plataforma.

Um resumo da dinâmica da interação entre os agentes da sociedade pode começar com um aluno acessando um Agente LMS em busca de experiências de aprendizagem. Baseado no seu processamento, o Agente LMS invoca determinado ILO, que passa a interagir com o aluno. Durante este processo, o ILO pode interagir com:

- o Agente LMS: para passar ou pedir maiores informações sobre o aluno e assim poder adotar alguma sistemática de adaptação, por exemplo;
- outros ILOs: para obter algum conhecimento que ainda não possui e desta forma completar a experiência gerada, por exemplo;
- o Agente ILOR: para encontrar algum ILO com o qual possa trocar experiências e conhecimentos tendo em vista algum processo de adaptação ao estudante, por exemplo.

Ainda quanto às possibilidades de interação entre os agentes, é possível que o Agente LMS execute uma pesquisa no Agente ILOR.

Uma análise prévia da sociedade remete a implementação dos agentes através de arquitetura híbrida, ou seja, que contemple aspectos cognitivos, já que eles devem possuir determinado conhecimento acerca do ambiente e dos outros agentes, e aspectos reativos, uma vez que devem reagir a estímulos externos.

As secções seguintes detalham as características e funcionalidades dos agentes desta sociedade.

6.3.1 Objetos Inteligentes de Aprendizagem

Os Objetos Inteligentes de Aprendizagem são as entidades centrais de estudo nesta pesquisa, sobretudo no que corresponde às suas características de agentes. Como já comentado, a principal tarefa de um ILO é gerar experiências de aprendizagem para os alunos.

Um ILO deve ser capaz de disponibilizar as suas informações de metadados aos outros agentes da plataforma. Como capacidade adicional, ele pode ser capaz de obter informações sobre os aspectos e características dos estudantes e de trocar estas informações com os outros agentes da plataforma, contribuindo com o aumento da qualidade geral do processo de aprendizagem gerado pela sociedade.

Tendo em vista as capacidades de manipular essas informações, os ILOs podem ser divididos em 3 níveis ou classes. Esta divisão se dá sob um aspecto incremental, ou seja, uma classe superior incorpora todas as capacidades da sua classe ligeiramente inferior.

A **classe 1** compreende os ILOs que manipulam e respondem a consultas sobre as suas informações de metadados. A **classe 2** é composta pelos ILOs que manipulam e respondem a consultas sobre os seus metadados e também sobre o estudante com o qual está interagindo. A **classe 3** é formada pelos ILOs que, além de manipular e responder a consultas sobre os seus metadados e sobre o estudante com o qual está interagindo, são capazes de obter de outros ILOs e agentes este mesmo tipo de informações.

A seguir é apresentada uma análise detalhada das características de cada uma dessas classes.

6.3.1.1 Classe 1

É a classe composta pelos ILOs mais simples. As suas capacidades de comunicação são limitadas a responder consultas sobre os seus metadados. Isso implica que uma arquitetura mínima para a construção de ILOs deve prever uma base de conhecimento que guarde as suas informações de metadados, as quais devem ser acessíveis através de consultas feitas por troca de mensagens. A Figura 6.3 apresenta esta concepção. O diálogo comunicativo a ser utilizado neste processo está descrito na secção 6.4.3.1. A

forma como as informações estão representadas na base de conhecimento do ILO foge ao escopo deste trabalho, ficando a cargo do seu projetista.

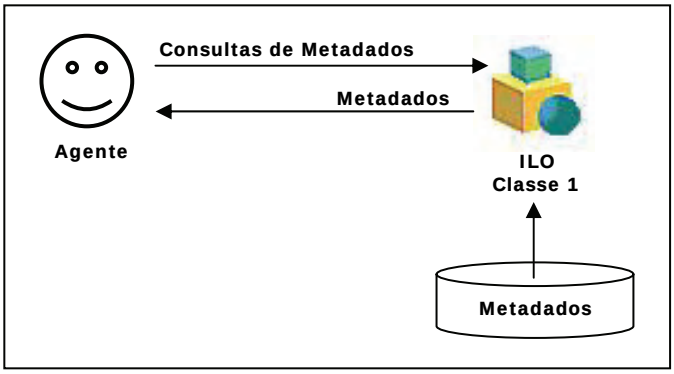

Figura 6.3: Esquema de uma arquitetura para ILOs de classe 1.

Esta classe é considerada a menos avançada devido ao fato de que a simples manipulação de metadados não torna possível que o ILO apresente algum comportamento de adaptação ao contexto de aprendizagem do estudante.

Essa mesma limitação é também observada ao nível da sociedade se considerarmos que todos os ILOs que a compõem pertencem à esta classe. No entanto, se a sociedade possuir também ILOs de classes mais avançadas, é possível se obter alguma adaptação. Como exemplo, um ILO de classe 3 pode ser capaz de buscar informações de ILOs de classe 1 com o objetivo de invocá-los e assim completar a experiência de aprendizagem de que o aluno necessita. O mesmo pode ser feito por um Agente LMS que atue como um Ambiente Inteligente de Aprendizagem (ILE), o qual pode utilizar ILOs de classe 1 no seu processo de ensino e obter informações sobre os estudantes, que são essenciais para o processo de adaptação de um ILE, a partir dos ILOs de outras classes.

6.3.1.2 Classe 2

Os ILOs que pertencem a esta classe são capazes de responder a consultas sobre os seus metadados, característica herdada da classe 1, e são capazes de rastrear e de responder a consultas sobre informações acerca dos estudantes. Isso implica que além de uma base de conhecimento sobre metadados, um ILO de classe 2 deve armazenar também informações sobre o estudante. A Figura 6.4 apresenta uma diagramação desta concepção. Os diálogos comunicativos que devem ser utilizados, representados na figura pelas setas que unem o ILO ao Agente, estão apresentados na secção 6.4.3.2.

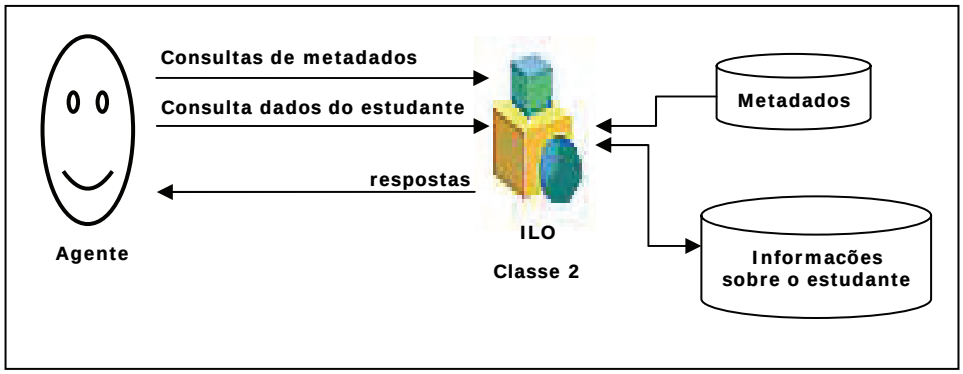

Figura 6.4: Esquema de uma arquitetura para ILOs de classe 2.

Esta classe de ILOs é considerada como uma classe intermediária devido ao fato de que, rastreando dados acerca do estudante, é possível que o ILO seja capaz de apresentar algum comportamento de adaptação ao mesmo. No entanto, esta classe ainda não possui capacidade de acessar os dados levantados pelos outros ILOs, o que poderia aumentar consideravelmente a eficácia do seu processo de adaptação.

Sendo capaz de responder a consultas com relação aos dados dos estudantes, este tipo de ILO contribui com que a sociedade aumente a sua capacidade de adaptação. Como exemplo, um ILO de classe 3 (ver próxima secção) poderia acessar os dados levantados por vários ILOs de classe 2, e a partir da análise destes dados tomar as suas decisões mais acertadamente. O mesmo pode ser aplicado ao Agente LMS.

6.3.1.3 Classe 3

Os ILOs que pertencem a esta classe, além de apresentarem as mesmas funcionalidades das duas classes apresentadas acima, são capazes obter informações mantidas por outros agentes. Primariamente, estas informações podem ser de metadados ou sobre os estudantes. A obtenção delas acarreta na possibilidade de implementar adaptação em um grau bastante alto tanto ao nível da sociedade quanto ao nível do próprio ILO.

A Figura 6.5 apresenta uma diagramação desta classe.

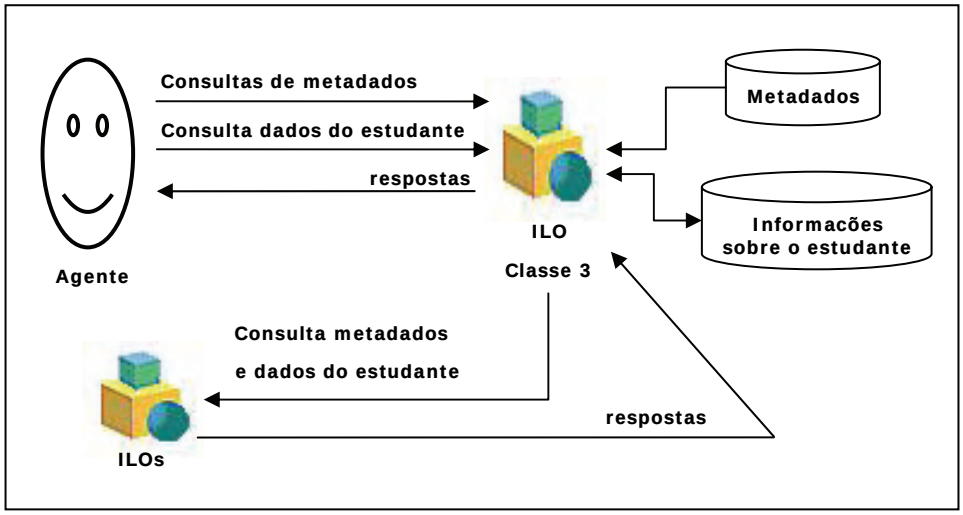

Figura 6.5: Esquema de uma arquitetura para ILOs de classe 3.

Para consultar as informações dos outros ILOs, um ILO de classe 3 deve utilizar os diálogos definidos para interação com os ILOs de classes 1 e 2. A particularidade inserida aqui é a de que o iniciante do diálogo agora é um ILO de classe 3.

6.3.2 Agente LMS

Um Agente LMS é uma abstração de um Sistema Gerenciador de Aprendizagem (LMS – *Learning Management System*) (ver secção 4.4). Suas responsabilidades envolvem as questões administrativas de um ambiente de aprendizagem que tenha nos ILOs as suas entidades geradoras de experiências de aprendizagem.

A condução da comunicação deste agente com os outros agentes da sociedade deve contemplar minimamente os itens definidos na secção 6.4.

O escopo deste trabalho não contempla a definição de uma arquitetura para a construção de agentes deste tipo. Também não são definidos os serviços auxiliares que este agente deve prover à sociedade e aos estudantes. No entanto, a título de exemplo, este agente pode disponibilizar:

- mecanismos de autenticação de usuários;
- controle sobre o processo de aprendizagem do aluno; tornando-se um Sistema de Apoio a Aprendizagem, um Ambiente Inteligente de Aprendizagem ou um Sistema Tutor Inteligente, conforme o seu nível de sofisticação;
- ferramentas diversas e auxiliares à aprendizagem, como calculadoras, mecanismos de interação entre os alunos, e etc;
- mecanismos de armazenamento de informações sobre os estudantes, tornando-se um repositório no qual os ILOs podem buscar informações relevantes que possam guiá-lo durante a interação com o aluno.
- outras funcionalidades.

6.3.3 Agente ILOR

Um Agente ILOR corresponde a uma abstração de um repositório de objetos de aprendizagem que, ao invés de armazenar informações sobre objetos de aprendizagem, armazena informações sobre ILOs.

A principal função de um Agente ILOR é responder a consultas de outros agentes que estejam em busca de ILOs. Deste modo ele permite que outros agentes encontrem ILOs de forma facilitada e centralizada. Sob esta mesma função, um ILOR deve manter uma lista dos ILOs que estão ativos na plataforma, de forma que possa informar aos agentes que o consultam se o ILO retornado está ativo ou não.

O diálogo a ser utilizado para acesso ao serviço de busca de um ILOR está definido na secção 6.4.3.3.

O escopo deste trabalho não contempla a definição de uma arquitetura para a construção de agentes deste tipo. Também não são definidos os serviços auxiliares que este agente deve prover à sociedade e aos estudantes. No entanto, a título de exemplo, este agente pode disponibilizar serviços alternativos de busca como procurar conteúdo na *Internet* quando não conseguir satisfazer à demanda de um agente.

6.4 Modelagem da Comunicação

Todo o processo de comunicação entre os agentes apresentados na secção anterior deve se dar por troca de mensagens FIPA-ACL através do ambiente FIPA.

Na modelagem da comunicação foi utilizada uma técnica extraída da metodologia MAS-CommonKADS (FERNANDEZ, 1998). Tal técnica corresponde a observar os serviços que os agentes devem oferecer aos outros agentes da plataforma e relacionar cada um destes serviços com uma conversação, ou diálogo como também são conhecidas. As conversações são, portanto, o meio através do qual os agentes podem acessar os serviços de outros agentes.

Esse conceito de conversação pode ser utilizado de forma complementar ao conceito de conversação apresentado no Capítulo 2. Assim, neste trabalho, uma conversação é caracterizada por ser uma instanciação de um modelo de seqüência de troca de mensagens (um protocolo de interação) cujo objetivo é acessar um serviço disponibilizado por outro agente.

Nas secções seguintes serão apresentados detalhadamente o protocolo de interação, a ontologia, e os diálogos que devem ser utilizados pelos agentes.

6.4.1 Protocolo de Interação de Agentes utilizado

Tendo em mente a idéia de que a interação entre os agentes está baseada em serviços, cada diálogo constituiu-se na verdade em uma requisição de um serviço. Esta idéia permitiu que os diálogos identificados utilizassem o protocolo FIPA-Request. Este protocolo deve ser empregado por agentes que desejam solicitar a execução de alguma ação por parte de outro agente. Como todos os diálogos identificados correspondem a solicitações de serviços, logo o protocolo FIPA-Request pode ser perfeitamente utilizado.

Uma das possibilidades, alternativa à modelagem com FIPA-Request, seria a utilização do protocolo FIPA-Query em algumas das conversações. Este protocolo deve ser utilizado quando um agente deseja obter informações que estão na base de conhecimento de outro agente. A tarefa de obter as informações de metadados de um ILO é um grande candidato para a sua utilização. No entanto, a primeira alternativa parece ser mais viável tendo em vista a abordagem baseada em serviços que adotamos.

O protocolo FIPA-Request exige que o conteúdo da sua mensagem inicial, a mensagem de requisição (ato comunicativo *request*), seja uma **ação.** Uma ação é introduzida pela palavra-chave `action`. Tal palavra-chave introduz na verdade um predicado com dois argumentos. O primeiro argumento é corresponde ao agente da ação e é um identificador do agente que executará a ação. O segundo argumento é um termo indicando a ação a ser executada. Uma ação corresponde a uma abstração do conceito real de uma ação que um agente pode executar. Por exemplo, a ação **send-metadata** será utilizada por um agente que deseja obter as informações de metadados de um ILO.

A mensagem final do protocolo FIPA-Request, caso tudo ocorra com sucesso durante o processamento da ação, é uma mensagem de informação (ato comunicativo *inform*). Uma mensagem de informação requer um elemento do tipo **predicado**. Um predicado diz alguma coisa acerca do estado atual do mundo e pode ser verdadeiro ou falso. Neste trabalho utilizaremos dois predicados definidos pela própria FIPA. O primeiro deles corresponde ao predicado `result`, cujo significado é o de que o resultado da avaliação da ação codificada no primeiro argumento está contido no segundo

argumento. O segundo deles corresponde ao predicado `done`, cujo significado é o de que a ação codificada no primeiro argumento chegou ao final.

6.4.2 Ontologia para a comunicação entre os agentes

A ontologia apresentada nesta secção é baseada na abordagem adotada pela FIPA, que descreve suas ontologias em termos de predicados, ações e conceitos. Esta visão pragmática satisfaz todas as necessidades que os nossos agentes possuem em relação às suas comunicações.

Como linguagem de conteúdo, decidimos adotar um sub-conjunto da linguagem FIPA-SL: a linguagem FIPA-SL0. Tal sub-conjunto permite a construção de ações, conceitos e predicados binários simples. Não incorporando construções mais complexas nem instruções booleanas, ela facilita o processamento do conteúdo das mensagens.

Uma particularidade interessante da linguagem FIPA-SL, pelo menos neste momento do estado da arte, é a não existência de uma máquina de inferência. Isso têm limitado a sua utilização em sistemas mais complexos, pois todas as construções devem ser processadas de maneira um pouco estática e pelo próprio agente. Esta limitação foi sensível também a este trabalho.

Assim, juntamente com a apresentação dos elementos que compõem a ontologia utilizada pelos agentes deste trabalho, será apresentada também a forma como cada elemento deverá estar representado no conteúdo das mensagens. Para tal, utilizaremos a seguinte convenção: símbolos terminais estarão representados em **negrito** e deverão ser transcritos literalmente no conteúdo das mensagens; símbolos não-terminais que correspondem a conceitos ou ações definidos na própria ontologia estarão em representados em *itálico* e entre os sinais “<” e “>”. Símbolos não-terminais que não correspondam a elementos definidos na ontologia e que necessitam de uma explicação em linguagem natural ao invés de uma indicação estarão entre os sinais “<” e “>”. Os demais símbolos não-terminais que correspondem a tipos básicos de dados (*string, inteiro*) estarão em *itálico.* Símbolos não-terminais devem ser substituídos pelos seus conteúdos reais no momento da instanciação da conversação.

Além da ontologia apresentada abaixo, os agentes devem conhecer a ontologia definida no documento de referência SC0000023 (FIPA, 2004). Tal documento trata da definição do Modelo de Referência de Agentes da FIPA.

A ontologia apresentada abaixo deve ser referenciada como **ilo-ontology**.

6.4.2.1 Conceitos

a) Conceito `metadata`

Formato:

```
(metadata :content string)
```

Significado:

Este conceito afirma que existe um modelo de dados contendo um padrão de informações de metadados. Tais informações estão contidas em **:content**. Elas devem estar conforme o padrão *IEEE 1484.12.1 Standard for Learning Object Metadata* e ser representadas em XML conforme o padrão *IEEE 1484.12.3*

Standard for Learning Technology – Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata.

b) Conceito `dataModel`

Formato:

`(dataModel :content` *`string`***`)`**

Significado:

Este conceito afirma que existe um modelo de dados que corresponde a informações entre a interação entre um estudante e um ILO. Tais informações estão contidas em **:content**. Elas devem estar conforme o padrão *IEEE 1484.11.1 Standard for Learning Techology – Data Model for Content Object Communication*.

c) Conceito `learner`

Formato:

`(learner :name` *`string`* **`:id`** *`string`* **`:data-model`** *`string`***`)`**

Significado:

Este conceito afirma que existe um estudante de nome definido em **:name**, que tem como identificador único o conteúdo de **:id**, e cujas informações acerca da sua interação com um determinado ILO estão relacionadas em **:data-model**. O conteúdo de :data-model deve estar conforme o padrão *IEEE 1484.11.1 Standard for Learning Techology – Data Model for Content Object Communication*.

d) Conceito `ilo`

Formato:

`(ilo :agent-id` *`string`* **`:metadata`** *`string`* **`:location`** *`string`***`)`**

Significado:

Este conceito afirma que existe um ILO cujas informações de metadados estão definidas em **:metadata**. Caso o ILO esteja ativo na plataforma, **:agent-id** contém o seu identificador único**.** Caso contrário, **:location** contém a referência para o local onde o ILO pode ser encontrado (por exemplo, uma classe a partir do qual o ILO pode ser invocado). **:metadata** deve ser descrito de acordo com o padrão *IEEE 1484.12.1 Standard for Learning Object Metadata* e ser representadas em XML conforme o padrão *IEEE 1484.12.3 Standard for Learning Technology – Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata.*

6.4.2.2 Ações

a) Ação `send-metadata`

Formato:
`(send-metadata)`

Descrição:
Este termo corresponde à ação de requisitar que lhes sejam enviadas as informações de metadados de um ILO qualquer. Esta ação não possui argumentos.

b) Ação `send-learner`

Formato:
`(send-learner)`

Descrição:
Este termo corresponde à ação de requisitar que lhes sejam enviadas as informações acerca do estudante com o qual um ILO qualquer está trabalhando. Esta ação não possui argumentos.

c) Ação `search-ilo`

Formato:
`(search-ilo :metadata` *`string`*`)`

Descrição:
Este termo corresponde à ação de requisitar que o agente ILOR lhe envie informações sobre ILOs que satisfaçam os critérios definidos em **:metadata.** O argumento desta ação deve conter o conjunto mínimo de elementos de metadados que o ILO deve possuir para que possa ser considerado membro do conjunto de resultado da ação.

d) Ação `get-learner-lms`

Formato:
`(get-learner-lms :learner` *`string`* `:ilo` *`string`*`)`

Descrição:
Este termo corresponde à ação de solicitar que o agente LMS envie as informações sobre o estudante **:learner** relacionado ao ILO **:ilo**. Os argumentos **:learner** e **:ilo** são identificadores únicos para o estudante e para o ILO.

e) Ação `put-learner`

Formato:

(put-learner :learner <*learner*> **:ilo** <*ilo*>**)**

Descrição:

Este termo corresponde à ação de solicitar que o agente LMS armazene as informações sobre um determinado estudante. Os argumentos **:learner** e **:ilo** contêm as informações sobre o estudante e sobre o ILO.

f) Ação `activate`

Formato:
(activate :ilo <*ilo*>**)**

Descrição:

Este termo corresponde à ação de solicitar que o agente ILOR ajuste o status do ILO contido em **:ilo** para ativo.

g) Ação `deactivate`

Formato:
(deactivate :ilo <*ilo*>**)**

Descrição:

Este termo corresponde à ação de solicitar que o agente ILOR ajuste o status do ILO contido em **:ilo** para inativo.

6.4.3 Diálogos

Esta secção apresenta os diálogos que devem ser utilizados pelos agentes da sociedade proposta.

6.4.3.1 Requisitando informações de metadados de um ILO

A requisição das informações de metadados de um ILO é feita através do diálogo **get-metadata.** A mensagem de requisição tem como conteúdo a ação **send-metadata**. Caso tudo ocorra com sucesso, a mensagem final, de informação, conterá um conceito **metadata**. A seguir são apresentados maiores detalhes sobre este diálogo.

a) Definição

Diálogo get-metadata
Propriedades

Objetivo	Obter as informações de metadados de um ILO
Agentes	Um agente qualquer e um ILO
Iniciador	Um agente qualquer
Protocolo	FIPA-Request
Ontologia	**ilo-ontology**
Descrição	Um agente qualquer deseja obter as informações de metadados de um determinado ILO. Para tal, ele envia uma

mensagem de requisição tendo como conteúdo a ação **send-metadata**. Caso não entenda esta mensagem, o ILO responde com um **not-understood** repetindo o conteúdo da mensagem inicial. Caso a ação seja recusada ou tenha ocorrido alguma falha durante o seu processamento, o ILO envia respectivamente um **refuse** ou um **failure,** ambos contendo o conteúdo da mensagem inicial e as razões da recusa ou da falha. Caso a ação seja processada com sucesso, a mensagem de informação (**inform**) contém um conceito **metadata** contendo as informações de metadados do ILO.

b) Fases do diálogo

Fase	Emissor	Receptor	Ato	Conteúdo
1	Agente	ILO	**Request**	**(action** <*AID*> <*send-metadata*> **)**
2	ILO	Agente	**Agree**	**(** <conteúdo do ato da fase 1> **)**
	ILO	Agente	**not-understood**	**(** <conteúdo do ato da fase 1> **)**
	ILO	Agente	**Refuse**	**(** <conteúdo do ato da fase 1> <razões> **)**
3	ILO	Agente	**Inform**	**(result** <conteúdo do ato da fase 1> <*metadata*> **)**
	ILO	Agente	**Failure**	**(** <conteúdo do ato da fase 1> <razões> **)**

6.4.3.2 Requisitando informações sobre o aluno

A requisição das informações sobre o aluno com o qual um ILO está trabalhando é feita através do diálogo **get-learner.** A mensagem de requisição tem como conteúdo a ação **send-learner**. Caso tudo ocorra com sucesso, a mensagem final, de informação, conterá um conceito **learner**. A seguir são apresentados maiores detalhes sobre este diálogo.

a) Definição

Diálogo get-learner
Propriedades

Objetivo	Obter as informações sobre o estudante com o qual o ILO está trabalhando
Agentes	Um agente qualquer e um ILO
Iniciador	Um agente qualquer
Protocolo	FIPA-Request
Ontologia	**ilo-ontology**
Descrição	Um agente qualquer deseja obter as informações sobre o estudante com o qual um ILO está trabalhando. Para tal, ele uma mensagem de requisição para o ILO tendo como conteúdo a ação **send-learner**. Em caso de não entendimento da mensagem, o ILO responde com um **not-understood**

contendo o conteúdo da mensagem inicial. Caso a ação seja recusada ou tenha ocorrido alguma falha durante o seu processamento, o ILO envia respectivamente um **refuse** ou um **failure,** ambos contendo o conteúdo da mensagem inicial e as razões da recusa ou da falha. Caso a ação seja processada com sucesso, a mensagem de informação (**inform**) contém um conceito **learner** contendo as informações que o agente iniciante requisitou.

b) Fases do diálogo

Fase	Emissor	Receptor	Ato	Conteúdo
1	Agente	ILO	**Request**	**(action** <*AID*> <*send-learner*> **)**
2	ILO	Agente	**Agree**	**(** <conteúdo do ato da fase 1> **)**
	ILO	Agente	**not-understood**	**(** <conteúdo do ato da fase 1> **)**
	ILO	Agente	**Refuse**	**(** <conteúdo do ato da fase 1> <razões> **)**
3	ILO	Agente	**Inform**	**(result** <conteúdo do ato da fase 1> <*learner*> **)**
	ILO	Agente	**Failure**	**(** <conteúdo do ato da fase 1> <razões> **)**

6.4.3.3 Buscando ILOs na base de conhecimento do ILOR

A busca por ILOs em um ILOR é feita através do diálogo **search-ilo**. A mensagem de requisição deste diálogo contém a ação **search-ilo**. A ação search-ilo possui um argumento :metadata onde devem estar representados os critérios da busca. Estes critérios são na verdade uma representação dos elementos mínimos que as informações de metadados de um ILO deve possuir para que ele possa ser retornado como resultado da busca. A resposta da solicitação, caso tudo ocorra com sucesso, será um conjunto de conceitos **ilo** com informações sobre os ILOs que satisfazem os critérios da busca.

A seguir serão apresentadas mais informações sobre o diálogo para acessar este serviço.

a) Definição

Diálogo search-ilo
Propriedades

Objetivo	Obter informações sobre ILOs que atendem a um determinado critério
Agentes	Um agente qualquer e um ILOR
Iniciador	Um agente qualquer
Protocolo	FIPA-Request
Ontologia	**ilo-ontology**
Descrição	Um agente qualquer deseja solicitar que um agente ILOR lhe envie uma lista com os ILOs que satisfazem um determinado critério. Então, uma mensagem de requisição é enviada para o

ILOR tendo como conteúdo a ação **search-ilo**. Em caso de não entendimento desta mensagem, o ILOR responde com um **not-understood** repetindo o conteúdo da mensagem inicial. Caso a ação seja recusada ou tenha ocorrido alguma falha durante o seu processamento, o ILOR envia respectivamente um **refuse** ou um **failure,** ambos contendo o conteúdo da mensagem inicial e as razões da recusa ou da falha. Caso a ação seja processada com sucesso, a mensagem de informação (**inform**) contém um conjunto de conceitos **ilo**.

b) Fases do diálogo

Fase	Emissor	Receptor	Ato	Conteúdo
1	Agente	ILOR	**request**	**(** **action** <*AID*> <*search-ilo*> **)**
2	ILOR	Agente	**agree**	**(** <conteúdo do ato da fase 1> **)**
	ILOR	Agente	**not-understood**	**(** <conteúdo do ato da fase 1> **)**
	ILOR	Agente	**refuse**	**(** <conteúdo do ato da fase 1> <razões> **)**
3	ILOR	Agente	**inform**	**(** **result** <conteúdo do ato da fase 1> **(set <***ilo***>*)** **)**
	ILOR	Agente	**failure**	**(** <conteúdo do ato da fase 1> <razões> **)**

6.4.3.4 Solicitando informação sobre um estudante a um Agente LMS

Um ILO de classe 3 pode solicitar dois tipos de informações a um agente LMS. O primeiro corresponde a uma espécie de histórico da interação entre o próprio ILO e o estudante. O outro tipo corresponde às informações de um aluno em geral, independente de um ILO específico.

A solicitação dessas informações é feita através do diálogo **get-learner-lms.** A mensagem de requisição deste diálogo contém a ação **get-learner-lms**. Caso o agente queira obter as informações sobre um ILO específico, ele deve completar o argumento ILO desta ação com o identificador do ILO em questão. Caso ele queira as informações de todos os ILOs, tal campo deve ser deixado vazio. A resposta da solicitação, caso tudo ocorra com sucesso, será um conjunto de conceitos **dataModel** com as informações solicitadas.

A seguir serão apresentadas mais informações sobre o diálogo para acessar este serviço.

a) Definição

Diálogo get-learner-lms
Propriedades

Objetivo	Obter informações sobre um aluno
Agentes	Um agente qualquer (geralmente um ILO) e um Agente LMS
Iniciador	Um agente qualquer (geralmente um ILO)
Protocolo	FIPA-Request

Ontologia **ilo-ontology**
Descrição Um agente qualquer, geralmente um ILO, deseja solicitar que um agente LMS lhe envie as informações sobre um estudante qualquer. Então, uma mensagem de requisição é enviada para o LMS tendo como conteúdo a ação **get-learner-lms**. De acordo com a presença ou ausência do parâmetro **:ilo**, o agente LMS deve enviar respectivamente informações sobre o estudante e o ILO em questão, ou informações sobre o estudante e todos os ILOs. Em caso de não entendimento da mensagem, o ILO envia um **not-understood** com o conteúdo da mensagem inicial. Caso a ação seja recusada ou tenha ocorrido alguma falha durante o seu processamento, o ILO envia respectivamente um **refuse** ou um **failure,** ambos contendo o conteúdo da mensagem inicial e as razões da recusa ou da falha. Caso a ação seja processada com sucesso, a mensagem de informação (**inform**) contém um conjunto de conceitos **dataModel** contendo as informações que o agente iniciante requisitou.

b) Fases do diálogo

Fase	Emissor	Receptor	Ato	Conteúdo
1	Agente	LMS	**request**	**(** **action** <*AID*> <*get-learner-lms*> **)**
2	LMS	Agente	**agree**	**(** <conteúdo do ato da fase 1> **)**
	LMS	Agente	**not-understood**	**(** <conteúdo do ato da fase 1> **)**
	LMS	Agente	**Refuse**	**(** <conteúdo do ato da fase 1> <razões> **)**
3	LMS	Agente	**Inform**	**(** **result** <conteúdo do ato da fase 1> **(set <***dataModel***>*** **)** **)**
	LMS	Agente	**Failure**	**(** <conteúdo do ato da fase 1> <razões> **)**

6.4.3.5 Se registrando no DF

O agente DF da plataforma FIPA trabalha como um serviço de páginas amarelas, onde os agentes ativos na plataforma podem divulgar os serviços que são capazes de disponibilizar. O documento SC0000023 (FIPA, 2004) especifica como tal registro deve ser procedido.

Como forma de padronizar o registro dos serviços que os nossos agentes disponibilizam, convencionaremos que os agentes devem, ao entrar na plataforma, registrar todos os diálogos que são capazes de engajar.

A Figura 6.6 apresenta o conteúdo da mensagem que deve enviada ao DF caso um ILO queira registrar que conhece os diálogos get-learner e get-metadata.

```
(action
   (agent-identifier :name ilo@platform.com)
   (register
      (df-agent-description
         :name (agent-identifier :name ilo@platform.com)
         :languages (set fipa-sl0)
         :services (set
               (service-description
                 :name dialogue
                 :type get-metadata)
               (service-description
                 :name dialogue
                 :type get-learner))
          )
    )
)
```

Figura 6.6: Exemplo de registro junto ao DF.

6.4.3.6 Se registrando no AMS

Todo agente que entre em atividade numa plataforma FIPA deve obrigatoriamente efetuar o seu registro no AMS desta plataforma. O documento SC0000023 (FIPA, 2004) especifica como tal registro deve ser procedido.

6.4.3.7 Solicitando ao ILOR mudança de status para ativo/inativo

Para modificar o seu status de atividade junto a um ILOR, um ILO deve utilizar o diálogo **modify-status**. Na mensagem de requisição são passadas as ações **activate** ou **deactivate**. A resposta da solicitação, se tudo ocorrer como desejado, não conterá conceito algum, apenas a indicação de que a ação foi efetuada.

A seguir serão apresentadas mais informações sobre o diálogo para acessar este serviço.

a) Definição

Diálogo modify-status
Propriedades

Objetivo	Alterar o status de um ILO atividade no ILOR
Agentes	Um ILO e um ILOR
Iniciador	ILO
Protocolo	FIPA-Request
Ontologia	**ilo-ontology**
Descrição	Um ILO deseja mudar o seu status de atividade em um ILOR. Para tal ele envia para o ILOR uma mensagem de requisição contendo como conteúdo a ação **activate** ou a ação **deactivate**. Para mudar para ativo a ação deve ser **activate**. Para mudar para não ativo a ação deve ser **deactivate.** Em caso de não entendimento desta mensagem, o ILOR responde com um **not-understood** repetindo o conteúdo da mensagem inicial. Caso a ação seja recusada ou tenha ocorrido alguma falha durante o seu processamento, o ILOR envia respectivamente um **refuse** ou um **failure,** ambos contendo o

conteúdo da mensagem inicial e as razões da recusa ou da falha. Caso a ação seja processada com sucesso, a mensagem de informação (**inform**) contém apenas a indicação de que a ação solicitada foi concluída com sucesso.

b) Fases do diálogo

Fase	Emissor	Receptor	Ato	Conteúdo
1	ILO	ILOR	**request**	**(action** <*AID*> <*activate* | *deactivate*> **)**
2	ILOR	ILO	**agree**	**(** <conteúdo do ato da fase 1> **)**
	ILOR	ILO	**Not-understood**	**(** <conteúdo do ato da fase 1> **)**
	ILOR	ILO	**refuse**	**(** <conteúdo do ato da fase 1> <razões> **)**
3	ILOR	ILO	**inform**	**(done** <conteúdo do ato da fase 1> **)**
	ILOR	ILO	**failure**	**(** <conteúdo do ato da fase 1> <razões> **)**

6.4.3.8 Solicitando ao LMS que este armazene o dataModel de um aluno

A solicitação para que um Agente LMS salve as informações de um determinado estudante deve ser feita através do diálogo **put-learner.** Tal diálogo tem como argumentos **:learner e :ilo** onde são passadas as informações que se quer armazenar. A mensagem final do diálogo contém apenas uma indicação de que tudo correu com sucesso.

A seguir serão apresentadas mais informações sobre o diálogo para acessar este serviço.

a) Definição

Diálogo put-learner
Propriedades

Objetivo Solicitar armazenamento das informações de um aluno
Agentes ILO e Agente LMS
Iniciador ILO
Protocolo FIPA-Request
Ontologia **ilo-ontology**
Descrição Um ILO deseja armazenar as informações que recolheu acerca do aluno**.** A mensagem de requisição contém a ação **put-learner**, a qual possui um conceito **learner** e um conceito **ilo**. Em caso de não entendimento da mensagem, o LMS envia um **not-understood** com o conteúdo da mensagem inicial. Caso a ação seja recusada ou tenha ocorrido alguma falha durante o seu processamento, o LMS envia respectivamente um **refuse** ou um **failure,** ambos contendo o conteúdo da mensagem inicial e as razões da recusa ou falha. Caso a ação seja processada com sucesso, a mensagem de informação (**inform**) contém apenas a indicação de que a ação solicitada foi concluída.

b) Fases do diálogo

Fase	Emissor	Receptor	Ato	Conteúdo
1	Agente	ILO	**request**	**(action** <*AID*> <*put-learner*> **)**
2	ILO	Agente	**agree**	**(** <conteúdo do ato da fase 1> **)**
	ILO	Agente	**Not-understood**	**(** <conteúdo do ato da fase 1> **)**
	ILO	Agente	**refuse**	**(** <conteúdo do ato da fase 1> <razões> **)**
3	ILO	Agente	**inform**	**(done** <conteúdo do ato da fase 1> **)**
	ILO	Agente	**failure**	**(** <conteúdo do ato da fase 1> <razões> **)**

6.5 Dinâmica da Sociedade

Os agentes da sociedade proposta possuem três estados distintos: não-iniciado, ativo e finalizado.

O estado não-iniciado simboliza aquele estado em que o agente não foi invocado ou ainda não se registrou na plataforma. O registro na plataforma corresponde ao ato do agente notificar ao DF e ao AMS acerca da sua instanciação. Todo agente que é invocado em uma plataforma FIPA deve efetuar o seu registro no DF e no AMS. Esta é uma exigência do esquema de gerenciamento de agentes da FIPA. Neste processo devem ser utilizadas as definições apresentadas no documento SC0000023 (FIPA, 2004) e as constantes definidas na secção 6.4.3.5.

Um agente do tipo ILO deve, além de registrar-se no DF e no AMS, efetuar a modificação do seu *status* junto ao agente ILOR. Isso deve ser feito através da utilização do diálogo modify-status.

Depois de ser invocado e de registrar-se na plataforma, o agente passa ao estado de ativo. Neste estado o agente está hábil a desempenhar as funções para o qual foi designado.

Para passar ao estado de finalizado, o agente necessita passar por um processo de finalização. A finalização é processo no qual o agente notifica o DF e o AMS de que está sendo finalizado. Esta é uma exigência do esquema de gerenciamento de agentes da FIPA. Todo agente que esteja deixando a plataforma deve ter seus registros apagados junto ao DF e ao AMS. Neste processo devem ser utilizadas as definições apresentadas no documento SC0000023 (FIPA, 2004).

No caso de um agente do tipo ILO, o seu *status* no agente ILOR deve ser modificado para inativo. Isso é feito através do diálogo modify-status e da ação deactivate.

Tão logo o agente seja finalizado, ele retorna ao estado de não-iniciado.

7 PROPOSTA DE UMA ARQUITETURA DE AGENTE

Como forma de validar a proposta feita no Capítulo anterior, este Capítulo apresenta uma arquitetura interna de agentes como exemplo de arquitetura para ser utilizada na construção dos agentes propostos. Na seqüência do Capítulo será apresentado um *framework* que instancia a arquitetura proposta e facilita a implementação de agentes compatíveis com ela.

7.1 Arquitetura proposta

Uma arquitetura de agente compreende a descrição de quais são e como são os processos internos que possibilitam a interação do agente com o seu ambiente (ver Capítulo 2). A arquitetura proposta (Figura 7.1) envolve três componentes básicos.

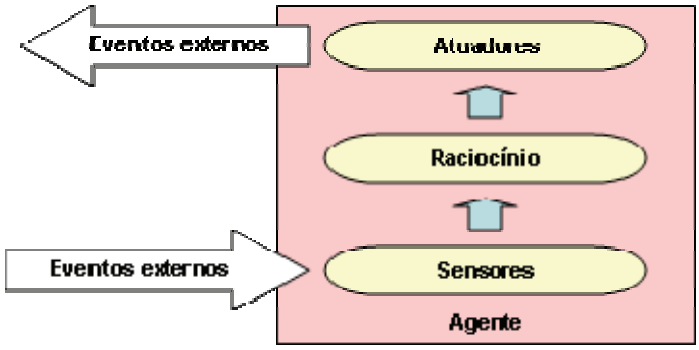

Figura 7.1: Componentes da arquitetura proposta

Um componente de sensoriamento é responsável por perceber as modificações no ambiente, tal como recebimento de mensagens e eventos em uma interface gráfica, caso o agente possua uma. Um componente de raciocínio é responsável pela definição de qual comportamento o agente irá adotar em relação ao evento percebido. E um componente de atuação é responsável por executar as ações do agente no ambiente.

7.1.1 Arquitetura híbrida em camadas

A arquitetura apresentada se caracteriza como uma arquitetura híbrida, na qual são contemplados aspectos cognitivos e reativos. Tais 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 necessita 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.

A Figura 7.2 ilustra o fluxo de como os eventos são tratados através das diversas camadas desta arquitetura.

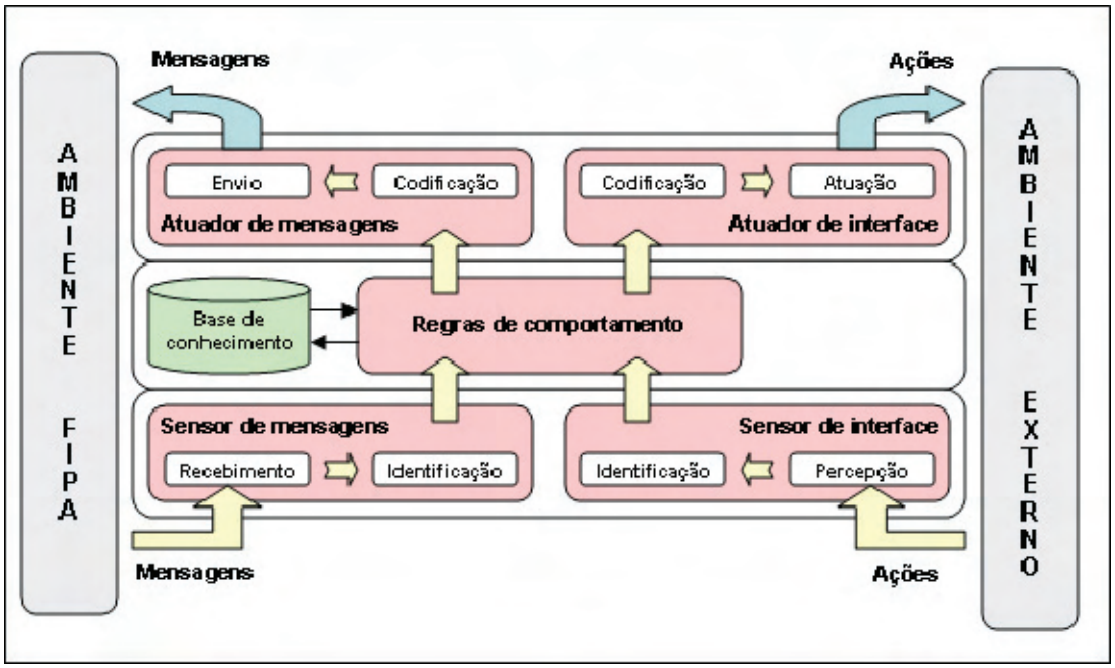

Figura 7.2: As três camadas da arquitetura proposta.

As secções 7.1.1.1, 7.1.1.2 e 7.1.1.3 apresentam as camadas de sensores, de raciocínio e de atuadores representadas na Figura acima pelos três retângulos centrais.

7.1.1.1 Primeira camada: sensores

A primeira camada é a camada de sensoriamento. Encapsulada no componente de sensoriamento, 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.

Está prevista a percepção de dois tipos 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.

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[2] 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.

Tendo em vista os dois tipos de eventos previstos, o componente de sensoriamento foi dividido em dois sub-componentes: componente de sensoriamento de mensagens e componente de sensoriamento de interface.

O 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.

Como exemplo mais específico do comportamento deste sub-componente, considere o recebimento de uma mensagem requisitando que o agente execute alguma ação. Esta mensagem foi enviada por um agente através do ambiente FIPA. O sub-componente de sensoriamento de mensagens recebe a mensagem e identifica a ação que foi solicitada pelo outro agente. A partir disso, o componente de raciocínio entra em ação e prossegue o processamento.

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.

Como exemplo mais específico deste sub-componente, considere uma ação de um usuário na interface JSP exemplificada acima. Por exemplo, o pressionamento de um botão. O componente de sensoriamento de interface percebe tal evento e o identifica. A partir daí, o componente de raciocínio entra em ação e assume o processamento.

7.1.1.2 Segunda camada: raciocínio

A segunda camada, camada de raciocínio, 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.

Esta camada é a que mais caracteriza o aspecto deliberativo da arquitetura em geral. É aqui que o conhecimento do agente entra mais profundamente em ação para permitir que o agente tome as suas decisões.

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.

---

[2] Neste trabalho, consideramos agentes externos como sendo agentes de sotware ou não que atuam fora do ambiente FIPA. Por exemplo, um agente externo pode corresponder a um aluno ou a um usuário qualquer de um Objeto Inteligente de Aprendizagem.

7.1.1.3 Terceira camada: atuadores

A terceira camada é a camada de atuadores. Encapsulada no componente de atuação, esta camada é responsável pela ação do agente no ambiente.

Está prevista a execução de 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 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.

Tendo em vista os dois tipos de eventos previstos, o componente de atuação foi dividido em dois sub-componentes: componente de atuação de mensagens e componente de atuação de interface.

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.

Como exemplo do processamento deste sub-componente, considere que, de acordo com o comportamento adotado pelo componente de raciocínio, seja necessário solicitar que um outro agente execute uma ação. Tal solicitação deve ser feita através do envio de uma mensagem ao agente determinado. O sub-componente de atuação de mensagens é responsável por codificar a solicitação em uma mensagem e por enviar a mensagem ao agente específico.

O componente de atuação 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 atuação de interface terá que executar as suas ações através desta interface JSP.

Como exemplo do processamento deste sub-componente, considere que, de acordo com o comportamento adotado pelo componente de raciocínio, seja necessário atuar na interface do agente externo. Por exemplo, é necessário atualizar algum item na aplicação JSP exemplificada acima. O componente de atuação de interface é responsável por codificar o comportamento adotado pelo agente de raciocínio e, neste caso, atualizar a página JSP.

7.2 Mapeamento da arquitetura proposta para uma arquitetura real

Tendo em vista as facilidades oferecidas pelo *framework* FIPA-OS (FIPA-OS, 2004) para a implementação de agentes compatíveis com o padrão FIPA, ele foi escolhido para servir de base para a implementação da arquitetura proposta na secção anterior.

No FIPA-OS, agentes são naturalmente implementados através de extensões da classe FIPAOSAgent. Esta classe é responsável por carregar automaticamente os componentes obrigatórios que compõem um agente FIPA-OS: o *Message Transport System (MTS),* o *Task Manager (TM)* e o *Conversation Manager (CM)*. O MTS

corresponde ao componente necessário para o transporte das mensagens entre os agentes. O CM possibilita que se mantenha um histórico das mensagens recebidas. E o TM habilita a capacidade de dividir as funcionalidades do agente em pequenas partes denominadas *tasks*.

O objetivo de uma *task* é desempenhar uma pequena função de um agente. Dessa forma, um agente FIPA-OS pode ser implementado como uma composição de *tasks*, sendo que várias delas podem ser executadas ao mesmo tempo, pois são *threads* independentes. Uma *task* possui também a capacidade de enviar e receber mensagens.

O fluxo de execução de um *task* é totalmente baseado em eventos. Assim, ao receber uma mensagem, por exemplo, um evento é levantado e o método implementado para tratá-lo é invocado automaticamente.

Ainda, uma *task* especial, geralmente denominada *IdleTask*, pode ser indicada como a *task* ouvinte do agente. Todas as mensagens que chegarem ao agente, e que ainda não tiverem sido ligadas à uma conversação serão desviadas para esta *task*.

Tendo em vista estas características, foi necessário um mapeamento entre o estilo arquitetural dos agentes implementados com o FIPA-OS e a arquitetura de agentes proposta na secção anterior. A seguir serão apresentados alguns aspectos deste mapeamento.

7.2.1 Componente de sensoriamento de mensagens e de atuação de mensagens

O comportamento do evento de receber uma mensagem em um agente FIPA-OS é o seguinte:

- Se receber uma mensagem que ainda não tenha sido ligada a nenhuma conversação ativa, esta mensagem será endereçada à task definida como a *IdleTask* do agente. Isso acontece toda vez é recebida a primeira mensagem de um diálogo. Nesta situação, a *IdleTask* é responsável por identificar o objetivo da mensagem e invocar uma *task* capaz de tratá-la, ligando assim a task à conversação;
- Se receber uma mensagem que já estiver ligada a uma conversação, tal mensagem é endereçada à *task* responsável pela conversação. Nesta situação, a *task* é responsável por identificar o objetivo da mensagem e por processá-la. Tendo sido ligada a uma conversação, uma *task* irá receber automaticamente toda e qualquer mensagem que se relacione à conversação em questão, até que o fluxo de mensagens definido no protocolo que a guia termine.

Caso o agente necessite empregar uma nova conversação, ele deve instanciar uma nova *task*, a qual será responsável pelo envio e pelo recebimento de todas as mensagens relativas à esta conversação.

Tendo em vista este comportamento, os sub-componentes de sensoriamento e de atuação que se relacionam ao envio e recebimento de mensagens estão distribuídos em alguns métodos da *IdleTask* e das *tasks* do agente. Num nível mais baixo nível, alguns componentes do FIPA-OS, como o MTS por exemplo, também estão envolvidos nestes componentes.

7.2.2 Componente de raciocínio

Dividir as funcionalidades de um agente em *tasks* significa dividir a inteligência do agente em pequenas unidades. Assim, o componente de raciocínio do agente está distribuído entre as diversas *tasks* que o formam.

7.2.3 Componente de atuação de interface e componente de sensoriamento de interface

Os sub-componentes de atuação e de sensoriamento que se relacionam à interface não serão modelados na implementação deste trabalho. Esta escolha foi feita devido ao fato de que a forma de como esta interface deve ser feita não é definida nesta pesquisa. Os projetistas de um Objeto Inteligente de Aprendizagem, e também dos outros agentes que compõem a sociedade proposta, devem ser livres para escolher a melhor forma de implementar a interface com os agentes externos. É possível, inclusive, que os agentes não tenham nenhuma interface externa.

Por estes motivos, está prevista apenas a presença de um método de entrada para eventos de interface. Os eventos de interface podem ser ligados ao agente através da implementação deste método, que será abstrato. Tal método não deve estar em nenhuma *task*, mas sim na classe base do agente, ou seja, aquela que é subclasse da classe FIPAOSAgent.

7.3 Um *framework* para o desenvolvimento dos agentes propostos

Baseado no mapeamento apresentado na secção anterior, foi construído um conjunto de classes Java que visa facilitar a construção de agentes que respeitam os requisitos definidos neste trabalho.

Este *framework* é na verdade uma extensão do *framework* FIPA-OS. Nesta secção serão apresentados alguns de seus aspectos.

7.3.1 Tratamento das conversações

O relacionamento entre os agentes que se envolvem em um diálogo tal como os definidos neste trabalho é caracterizado por um modelo cliente-servidor, onde 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.

Essa análise permitiu que fosse implementado um par de *tasks* para cada um dos diálogos definidos na secção 6.4. Uma delas deve ser usada pelo agente servidor e a outra pelo agente cliente.

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

A Figura 7.3 apresenta uma diagramação UML do 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.

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 IConcept (ver secção 7.3.2). simpleResult() deve ser utilizado quando o resultado do diálogo é um único conceito. composedResult() deve ser utilizado quando o resultado é composto por um conjunto de conceitos. simpleDone() deve ser utilizado quando não é necessário enviar resultados, apenas informar que a tarefa requisitada acabou com sucesso. 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. Neste método deve ser inserido o comportamento da Task em si. Portanto, aqui entra grande parte do raciocínio da *task* em questão.

A Figura 7.3 apresenta ainda um exemplo de classe que extende a classe HandleTask: a classe HandleGet_Metadata. Esta classe implementa a parte servidora do diálogo get-metadata. Ao implementá-la, foi necessário implementar apenas os métodos construtor, startTask() e processa().

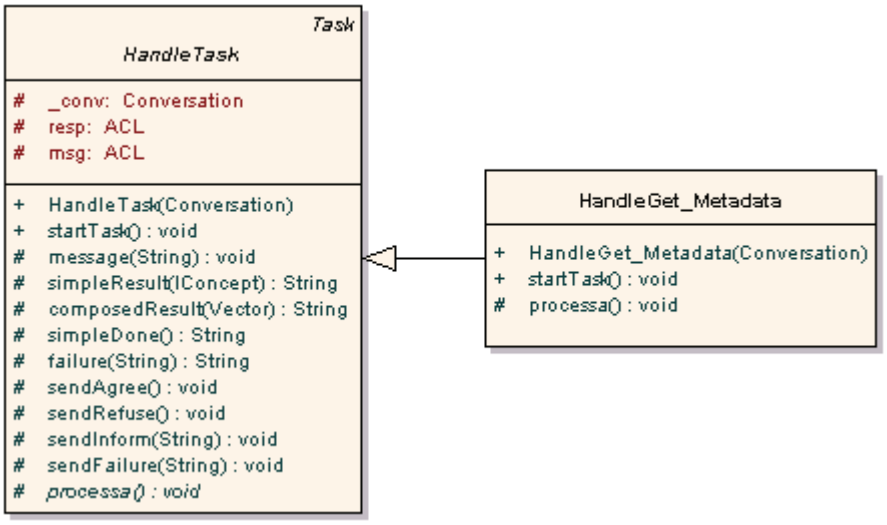

Figura 7.3: Diagramação UML do relacionamento entre uma classe servidora de diálogos e sua superclasse.

A Figura 7.4 apresenta uma diagramação UML do 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 IAction (ver secção 7.3.2), que deve ser implementada por

todas as ações previstas na ontologia definida neste trabalho. 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.

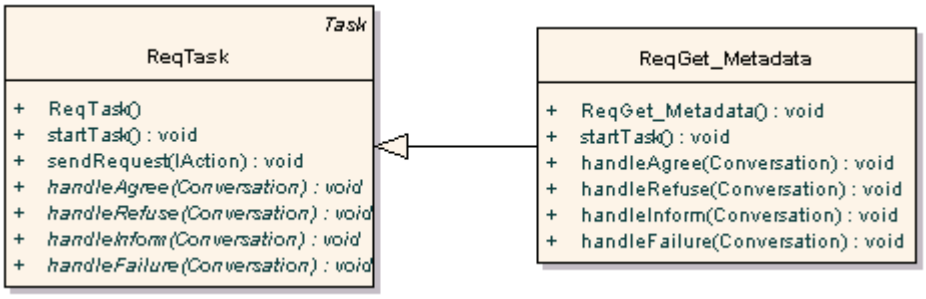

Figura 7.4: Diagramação UML do relacionamento entre uma classe cliente de diálogos e sua superclasse.

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 done*X*(), 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().

A Figura 7.5 ilustra o fluxo de troca de mensagens e chamadas de métodos para o tratamento de um diálogo. Tal ilustração possui sintaxe parecida com a sintaxe de um diagrama de seqüência da UML.

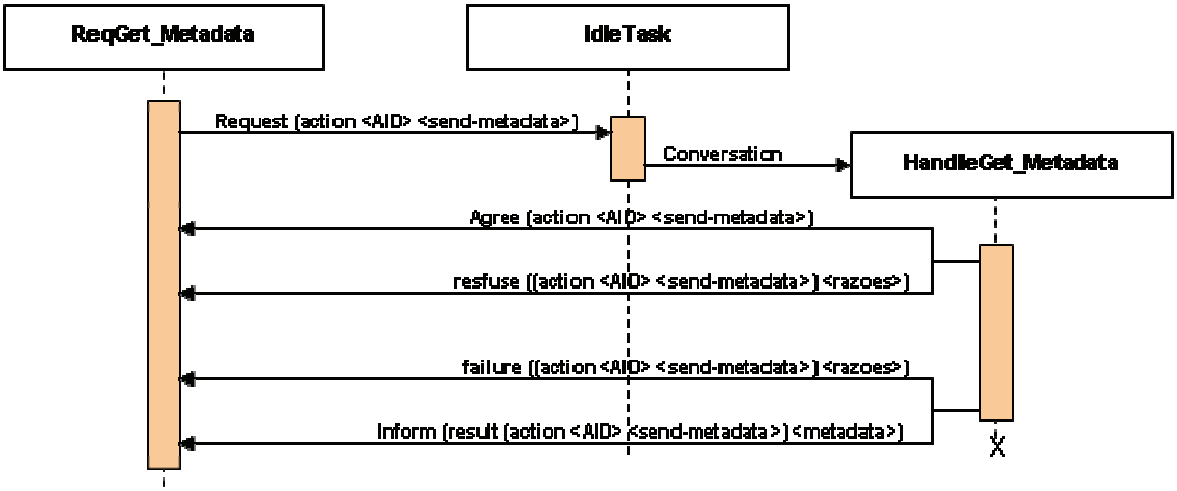

Figura 7.5: Relacionamento dinâmico entre as tasks cliente-servidora de um diálogo

Na figura, a seqüência de mensagens começa numa *task* de requisição de serviço. Esta *task* é uma sub-classe da classe ReqTask. Após 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 servidora).

No seu método startTask(), a task servidora decide 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 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 esta 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.

7.3.2 Tratamento da ontologia

Um dos requisitos estabelecidos pela modelagem da comunicação deste trabalho 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 e que entre em acordo com o que foi definido na secção 6.4.2. 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 de ontologia fossem implementadores de uma interface que contém um método toSL(). Para cada tipo de elemento, ação, conceito ou predicado, foi definida uma interface base específica.

A Figura 7.6 apresenta o diagrama de classes do pacote que modela os predicados da ontologia. A interface IPredicate define o método toSL() e o getName(). O predicado PAction contém um AID de um agente, representado aqui por uma string, e um objeto implementador da interface IAction, que representa a ação. Os predicados Done e Result contêm uma referência a um predicado PAction. Result contém um vetor de IConcept.

A Figura 7.7 apresenta o diagrama de classes que modela os conceitos da ontologia. Dois conceitos foram considerados chaves neste trabalho: o conceito metadata e o conceito dataModel. O primeiro corresponde às informações de metadados de um ILO e o segundo às 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 por tal em uma base relacional, ele pode

fazê-lo, desde que implemente a interface requerida. Na figura é apresentada uma implementação de exemplo.

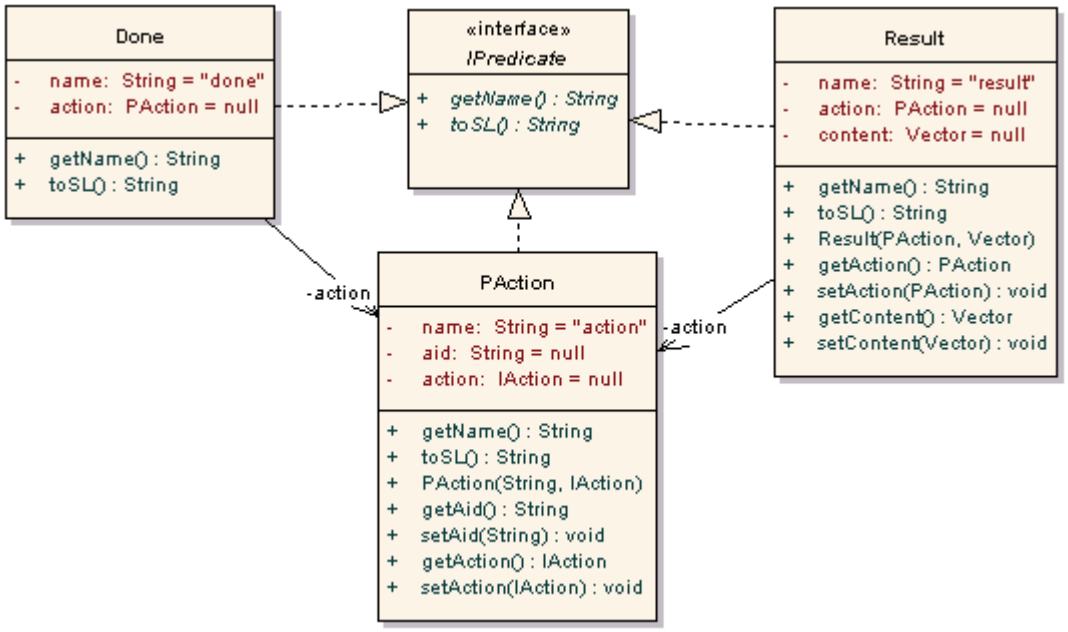

Figura 7.6: Diagrama de classe dos predicados que formam a ontologia.

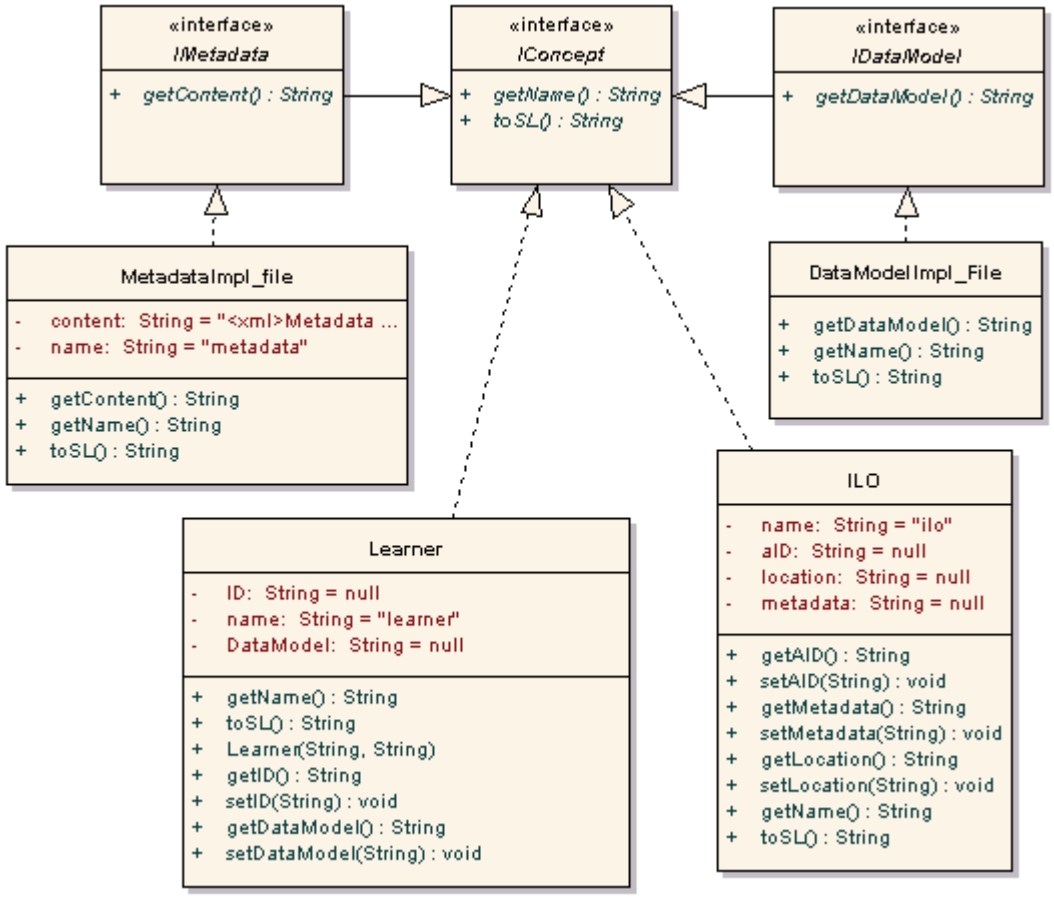

Figura 7.7: Diagrama de classes dos conceitos que formam a ontologia

A Figura 7.8 apresenta a modelagem das ações que formam a ontologia. Algumas das ações contêm referências a conceitos. Este é o caso das classes Put_learner, Activate, Deactivate e Get_learner_lms.

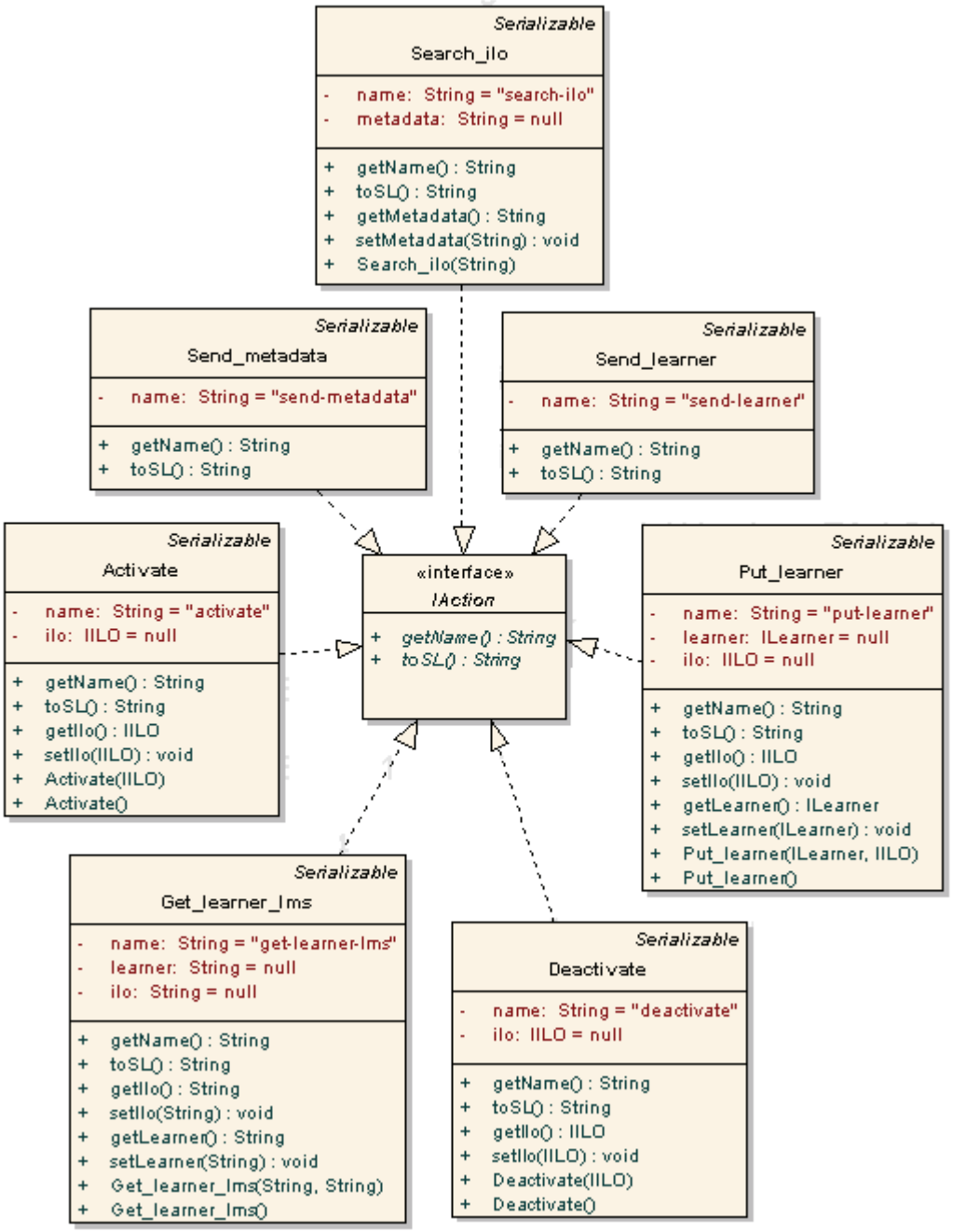

Figura 7.8: Diagrama de classes das ações que formam a ontologia.

7.3.3 As várias classes de ILOs

Conforme definido no Capítulo 6, os ILOs podem ser divididos em classes de acordo com as suas possibilidades comunicativas. Tendo em vista esta divisão, foi implementada uma classe ILOAgent e uma clase 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. Característica marcante nesta implementação é a possibilidade de que cada projetista escolha a maneira que mais lhe convier para inserir no ILO as suas informações de metadados, basta que a interface seja seguida. A indicação de qual a 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 de IdleTask.

A Figura 7.9 apresenta o diagrama de classe que representa a parte do *framework* utilizada para implementar um ILO de classe 1.

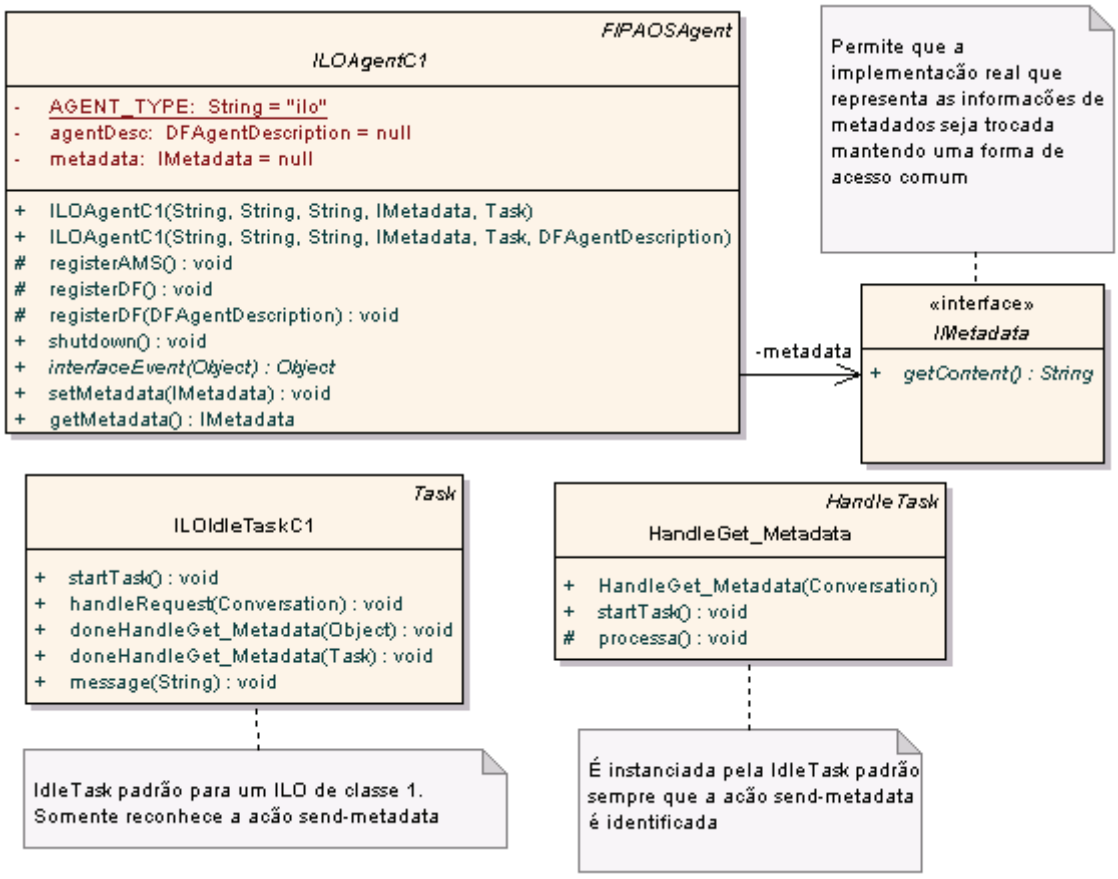

Figura 7.9: Diagrama de classes de um ILO de classe 1.

Na classe ILOAgentC1 apresentado na Figura 7.9, 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 extende 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 foi necessária a implementação de uma classe base específica para este tipo de ILO. Para implementá-lo basta que seja extendida a classe ILOAgentC2 e que sejam utilizados as tasks clientes definidas para cada um dos diálogos.

7.3.4 Agente ILOR

Para o agente ILOR foi definida uma IdleTask, a ILORIdleTask, que é capaz de identificar as ações relativas aos diálogos que este agente deve ser capaz de compreender. Sempre que uma solicitação de um serviço previsto é identificada, esta IdleTask instancia a task sevidora relacionada ao serviço identificado.

Assim, além da IdleTask, foram implementadas também as tasks servidoras para cada um dos diálogos relacionados a este agente.

Para implementar um agente ILOR mais sofisticado, basta que estas classes sejam extendidas.

7.3.5 Agente LMS

A implementação da parte que corresponde a um Agente LMS também seguiu os moldes utilizados para a implementação do Agente ILOR. Foram 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 extendidas.

8 CONCLUSÃO

O objetivo geral deste trabalho foi investigar pontos de convergência entre as duas tecnologias envolvidas, agentes e objetos de aprendizagem, desenvolvendo uma abordagem que permitisse a construção de objetos de aprendizagem baseados em agentes.

O fundamento base adotado foi a definição do conceito de Objeto Inteligente de Aprendizagem. Definimos um objeto inteligente de aprendizagem como sendo um agente que é capaz de gerar experiências de aprendizagem no mesmo sentido que os objetos de aprendizagem, ou seja, experiências de aprendizagem reutilizáveis.

Investigando o estado da arte da área de objetos de aprendizagem, verificou-se que a reusabilidade, característica principal de um objeto de aprendizagem, é tida como produto de outras três características: modularidade, *interoperabilidade* e capacidade de ser descoberto.

Assim, surgiu a necessidade de se definir alguns requisitos para que um agente pudesse ser considerado um Objeto Inteligente de Aprendizagem. A proposta destes requisitos foi baseada no fundamento de que um Objeto Inteligente de Aprendizagem deve manter as mesmas características de um objeto de aprendizagem. Ou seja, ele deve ser *interoperável*, modular e ter a capacidade de ser descoberto.

Na área de objetos de aprendizagem, a *interoperabilidade* é obtida através da adoção de padrões. Um dos grupos de maior reputação nesta área é o LTSC da IEEE. Então definimos que um ILO deve adotar, sempre que possível, os padrões 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. Para promover esta característica foi definido um agente responsável por parte deste trabalho, como será visto mais adiante.

A modularidade, tanto na área de agentes quanto na área de objetos de aprendizagem, só é alcançada através de um projeto bem feito.

Foi definido ainda 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.

O passo seguinte da pesquisa foi a definição de uma sociedade multiagente na qual os objetos inteligentes de aprendizagem pudessem ser executados. A definição dos membros desta sociedade se deu baseada nos sistemas que dão suporte aos objetos de aprendizagem. Assim, foram definidos três tipos de agentes: agente LMS, abstração de um sistema de gerenciamento de aprendizagem; agente ILOR, abstração de um repositório de objetos de aprendizagem; e os próprios ILOs.

Depois de definida a sociedade, deu-se uma fase de modelagem do processo de comunicação entre os agentes desta sociedade. Uma estrutura de comunicação bem definida permite que não seja necessário fazer restrição quanto ao tipo de arquitetura de agente que um desenvolvedor irá utilizar ao conceber o seu objeto inteligente de aprendizagem. Para que o agente se envolva na sociedade, basta que sejam seguidas tal estrutura de comunicação e os requisitos do ambiente multiagente em questão.

A modelagem da comunicação partiu do pressuposto de que toda a comunicação entre dois agentes está ligada a um diálogo e que um diálogo está sempre relacionado a um objetivo. A tarefa então era definir quais os objetivos de cada comunicação. Verificamos que existiam dois grandes grupos de objetivos: solicitações de serviços e consultas à base de conhecimento dos outros agentes. Nesta fase foi encontrada uma das grandes dificuldades enfrentadas neste trabalho. A linguagem de conteúdo sugerida pela FIPA, a FIPA-SL, é bastante poderosa. Porém, ela ainda não possui uma máquina de inferência, o que limita a utilização de todo o seu poder, sobretudo na formulação de consultas. Outras linguagens foram analisadas com o intuito de serem utilizadas no conteúdo das mensagens, tal como a linguagem Prolog, mas foram abandonadas devido às suas não concordâncias com o padrão FIPA.

A saída encontrada foi modelar a comunicação através de uma das técnicas previstas na metodologia MAS-CommonKADS. Tal técnica consiste em definir diálogos em função dos serviços que os agentes disponibilizam aos outros agentes. Assim, todos os diálogos da nossa sociedade são, na verdade, solicitações de serviço.

Quanto a isso, podemos concluir que a FIPA-SL é uma linguagem extremamente poderosa para a troca de informações entre agentes. No entanto, a não existência de uma máquina de inferência limita a utilização de todo o seu poder. Essa não existência também limita a utilização da própria linguagem FIPA-ACL, uma vez que mensagens que poderiam ser modeladas naturalmente através do ato comunicativo *query*, tiveram que ser modeladas através de um *request*.

O processo de modelagem da comunicação envolveu também a definição de uma ontologia. Esta ontologia refletiu informações que podem ser úteis durante um processo pedagógico utilizando os ILOs. Essa preocupação delimita bem o escopo de toda a modelagem de comunicação proposta neste trabalho: definir a comunicação de forma que os agentes da sociedade possam obter os dados de que necessitam durante o seu processo pedagógico.

Inicialmente, propomos a construção de uma ontologia de metadados. Esta ontologia teria um caráter bastante interno ao processamento dos agentes, permitindo que os agentes efetuassem algum raciocínio sobre ela. Porém, esta modelagem não se mostrou interessante nesta fase do trabalho. Isso porque a ontologia que foi definida, respeitando

o escopo de comunicação comentado acima, se caracteriza muito mais como uma ontologia de comunicação do que como uma ontologia de domínio. O objetivo da ontologia modelada é de que os agentes saibam qual deve ser o formato esperado para cada termo do conteúdo de uma mensagem. Esta é uma reflexão da visão pragmática que FIPA aplica na modelagem das suas ontologias.

Para permitir que os agentes trocassem informações de metadados, foram adicionados alguns conceitos na ontologia definida. Tais conceitos possuem argumentos nos quais devem ser inseridas as informações de metadados. Assim, ao invés de troca informações de metadados seguindo uma ontologia específica para metadados, os agentes trocam este tipo de informações através de alguns conceitos definidos na ontologia modelada.

Neste ponto, já estávamos com toda a proposta pronta. Faltava desenvolver algum artefato para avaliar o que havia sido declarado. Assim, foi proposta uma arquitetura de agente que pode ser utilizada na construção de agentes compatíveis com o que foi definido. Ressalta-se aqui que esta arquitetura serve de exemplo, os projetistas não são obrigados a seguí-la.

A arquitetura proposta se caracteriza por ser uma arquitetura híbrida que possui um fluxo de controle em camadas. A arquitetura é híbrida porque contempla ao mesmo tempo aspectos cognitivos e reativos. O fluxo do controle possui três camadas. A primeira camada é composta pelos sensores, responsáveis por identificar os eventos ocorridos no ambiente. A segunda camada é a camada de raciocínio, responsável por definir que comportamento o agente adotará com relação aos eventos percebidos pelos sensores. E, a terceira camada incorpora os atuadores, que são responsáveis pela atuação do agente no ambiente, caso a camada de raciocínio defina que alguma ação é necessária.

Tendo esta arquitetura definida, resolveu-se que seria interessante a implementação de um conjunto de recursos que facilitasse a sua adoção e a construção dos agentes propostos neste trabalho. Assim, partiu-se para a implementação de um pequeno *framework* que tem por objetivo o que foi descrito acima.

Um dos requisitos impostos neste 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 problema enfrentado aqui é que o estilo arquitetural de um agente FIPA-OS, à primeira vista, é um pouco diferente do estilo arquitetural dos agentes que seriam desenvolvidos através da arquitetura proposta. Assim, foi necessária uma tarefa de mapeamento entre um estilo e outro.

Concluído o mapeamento, construiu-se um *framework* com as seguintes características:

- 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;
- Permite 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;

- Foca a facilitação da implementação da parte comunicativa dos agentes;
- Permite que o projetista defina quais os diálogos, dos definidos neste trabalho, que o agente irá tratar. Sendo que todos os diálogos definidos neste trabalho estão disponíveis para utilização;
- Nos casos específicos dos ILOs, para os quais a incorporação de informações de metadados é obrigatória, o *framework* permite que o projetista defina a melhor maneira de implementar como estas informações são tratadas internamente pelos ILOs; o mesmo é aplicado para ILOs que manipulam informações sobre os estudantes.

A proposta envolvida nesta pesquisa está longe de ser encerrada. Como resultados deste trabalho em si, podemos inferir dos estudos feitos que uma abordagem baseada em agentes pode realmente trazer inúmeros benefícios à abordagem de objetos de aprendizagem. O mesmo ocorre com os ambientes computacionais de educação que dela se utilizarem.

A grande desvantagem percebida, pelo menos neste momento, é a complexidade que é incorporada ao desenvolvimento de um objeto de aprendizagem baseado em agente. Embora o *framework* desenvolvido, e até mesmo o próprio FIPA-OS, facilitem a construção destes agentes, não se pode negar que implementar um "agente objeto de aprendizagem" é muito mais complexo do que se implementar um objeto de aprendizagem simples. No entanto, esta desvantagem é compensada pela possibilidade de se conseguir implementar objetos de aprendizagem muito mais avançados pedagogicamente.

Assim, embora esta pesquisa, neste primeiro momento, não minimize as dificuldades encontradas pelo especialista em conteúdo no momento de construir um objeto de aprendizagem, através da adoção de uma arquitetura orientada a agentes pode-se desenvolver objetos de aprendizagem capazes de interagir de forma mais autônoma, promovendo uma maior adaptabilidade e interatividade aos ambientes computacionais de educação.

Dos objetivos propostos, todos eles foram satisfeitos com um bom nível. Com exceção da modelagem da ontologia de metadados, a qual verificou-se não ser necessária neste momento, tal como explicitado acima.

Como sugestão de trabalhos futuros podemos indicar a implementação de um conjunto de objetos inteligentes de aprendizagem utilizando o *framework* desenvolvido e a aplicação destes objetos em ambientes reais de aprendizagem como forma de avaliar parte da eficácia desta nova abordagem. Este trabalho envolveria também a implementação dos outros agentes da sociedade, ou seja, a implementação de um Agente LMS e de um Agente ILOR.

Trabalhos futuros poderiam envolver também a definição de novas arquiteturas para a implementação dos agentes. Estes estudos em particular poderiam avaliar a eficácia do modelo de comunicação definido neste trabalho.

No ramo específico de educação, um trabalho interessante seria utilizar esta abordagem para aplicar e testar teorias educacionais no sentido de verificar como a sociedade se comporta, qual a eficácia destas teorias na aprendizagem do aluno e como esta abordagem suportou as suas aplicações.

Na área específica de agentes, poderiam ser utilizados diferentes métodos de raciocínio para incorporar inteligência aos objetos. Existem diversos estudos enfocando este aspecto de um agente e que podem ser utilizadas.

O próximo passo no desenvolvimento desta pesquisa será a aplicação de mecanismos de coordenação no sentido de fazer a sociedade de objetos inteligentes de aprendizagem ser capaz de se auto-organizar, criando cursos inteiros e gerando experiências de aprendizagem mais avançadas. Um mecanismo eficiente de coordenação pode aumentar a qualidade dos resultados obtidos pelo sistema e reduzir o tempo gasto para que ele resolva as suas tarefas. (JENNINGS, 1996). Acreditamos que através disso podemos reduzir o tempo gasto para a construção de cursos e aumentar a eficácia da abordagem proposta nesta pesquisa.

Esta nova fase está prevista para um estudo de doutorado, no qual propomos estudar os métodos de coordenação de agentes existentes e aplicar alguns deles numa comunidade de objetos inteligentes de aprendizagem para verificar os seus benefícios e viabilidades. Se for preciso, iremos propor adaptações e criar variantes dos métodos estudados.

No final desta nova fase, esperamos responder a algumas questões, tais como:

(a) Pode o uso de tecnologias de agentes construir objetos de aprendizagem auto-adaptáveis e mais reusáveis?

(b) Pelo uso de métodos de coordenação apropriados, será a sociedade de objetos inteligentes de aprendizagem capaz de se auto-organizar, criando cursos inteiros e experiências de aprendizagem mais avançadas?

(c) Se sim, isso pode aumentar a eficácia e a efetividade desta nova abordagem?

(d) E, isso pode reduzir o tempo e custos envolvidos no desenvolvimento de cursos *on-line*?

Este trabalho apresentou uma abordagem na qual objetos de aprendizagem são construídos através de arquiteturas de agentes. Foi apresentada a base conceitual de tal abordagem, a modelagem de uma sociedade multiagente que a suporta e a modelagem do processo de comunicação entre os agentes desta sociedade. Foi apresentada uma arquitetura de agente que implementa os conceitos propostos e foi construído um conjunto de recursos para a implementação dos agentes da sociedade através da arquitetura proposta.

Esperamos ter dado contribuições importantes, refinando a eficácia da tecnologia de objetos de aprendizagem para a implementação de projetos de Educação a Distância.

REFERÊNCIAS

ADL. Advanced Distributed Learning - About ADL, [S.l.], Mar. 2004. Disponível em: <http://www.adlnet.org/aboutadl/index.cfm>. Acesso em: mar. 2004.

ARIADNE. Alliance of Remote Instructional Authoring & Distribution Networks for Europe, [S.l.], Mar. 2004. Disponível em: <http://www.ariadne-eu.org>. Acesso em: mar. 2004.

AULANET. Ambiente AulaNet. Disponível em: <http://aulanet.les.inf.puc-rio.br/aulanet>. Acesso em: jan. 2005.

BARR, A.; FEIGENBAUM, E. A. (Ed.). **The Handbock of Artificial Intelligence**. Los Altos: Morgan Kaufmann, 1981.

BAUER, B.; MULLER, J. P.; ODELL, J. Agent UML: A Formalism for Specifying Multiagent Interaction. In: CIANCARINI, P.; WOOLDRIDGE. M. (Ed.). **Agent-Oriented Software Engineering**. [S.l.]: Springer, 2001. p. 91-103. Disponível em: <http://www.auml.org/auml/supplements/Bauer-AOSE2000.pdf> Acesso em: maio 2004.

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

BLACKBOARD. **Blackboard Overview**. Disponível em: <http://blackboard.com/about>. Acesso em: fev. 2005.

BRADSHAW, J. M. An introduction to software agents. In: BRADSHAW, J. M. (Ed.). **Software Agents**. Massachusetts: MIT Press, 1997.

BrainX. **About BrainX**. Disponível em: <http://www.brainx.com/about.htm> Acesso em: jan. 2005.

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

BROOKS, R. A. A Robust Layered Control System for a Mobile Robot. **IEEE Journal Of Robotics And Automation**, [S.l.], p. 14-23, Apr. 1986.

CAMPOS, G. H. B. Relato de uma Experiência – Parte II. **TI Master**, [S.l.], dez. 2003. Disponível em: <http://www.timaster.com.br/revista/artigos/main_artigo.asp?codigo=879&pag=1>. Acesso em: jun. 2004.

CanCore: About CanCore Metadata Initiative. Disponível em: <http://www.cancore.ca./en/about.html>. Acesso em: jul. 2004.

CAREO: Campus Alberta Repository of Educational Objects. Disponível em: <http://www.careo.org/docs.html>. Acesso em: jan. 2005.

CESTA: Coletânea de Entidades de Suporte ao uso de Tecnologia na Aprendizagem. Disponível em: <http://www.cinted.ufrgs.br/CESTA/cestadescr.html>. Acesso em: mar. 2005.

COHEN, P. R.; LEVESQUE, H. J. Communicative Actions for Artifical Agents. In: INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS, ICMAS, 1., 1995. **Proceedings...** Cambridge, MA, USA: The MIT Press, 1995.

Command Technologies: Technology Application to Training. Disponível em: <http://www.commtechinc.com/productsandservices/scorm.htm> Acesso em: jan. 2005.

DCMI: About Dublin Core Metadata Initiative. Disponível em: <http://www.dublincore.org/about/>. Acesso em: jul. 2004.

DOWNES, S. Learning Objects: Resources For Distance Education Worldwide. **International Review of Research in Open and Distance Learning**, [S.l.], v.2, n.1, July 2001. Disponível em: <http://www.irrodl.org/content/v2.1/downes.html>. Acesso em: jan. 2004.

DOWNES, S. Smart Learning Objects. **The Learning Place**. [S.l.:s.n.], 2002. Disponível em: <http://education.qld.gov.au/learningplace/onlinelearning/courses/sdownesapril.html>. Acesso em: jan. 2004.

FERNÁNDEZ , C. Á. I. **Definición de una Metodología para el Desarrollo de Sistemas Multiagentes**. 1998. Tese (Doutorado) - Universidad Politécnica de Madrid, Madri, Espanha.

FININ, T.; WEBER, J.; 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.

FIPA - FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS. **Specifications Repository.** Disponível em: <http://www.fipa.org/specs/pesspecs.tar.gz> Acesso em: nov. 2004.

FIPA-OS: FIPA - Open Source. Disponível em: <http://fipa-os.sourgeforge.net>. Acesso em: ago. 2004.

FRIESEN, N. What are Learning Objects? **Interactive Learning Environments**, [S.l.], v.9, n.3, Dec. 2001. Disponível em: <http://www.careo.org/documents/objects.html>. Acesso em: mar. 2004.

GIRAFFA L. M.; VICCARI R. M.; SELF, J. Multi-Agent based pedagogical games. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TUTORING SYSTEMS, ITS 4., 1998. **Intelligent Tutoring Systems:** Proceedings. Berlin: Springer-Verlag, 1998.

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

GOMES, E. R.; SILVEIRA, R. A.; VICARI, R. M. Objetos Inteligentes de Aprendizagem: Uma abordagem baseada em agentes para objetos de aprendizagem. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 15., 2004, Manaus. **Anais...** Manaus : SBC, 2004. p. 408-417.

HODGINS, W.; CONNER, M. L. Everything You Ever Wanted to Know about Learning Standards but Were Afraid to Ask. **Learning in the New Economy Magazine**, [S.l.] 2000. Disponível em: <http://linezine.com/2.1/features/wheyewtkls.htm>. Acesso em: jun. 2004.

IMS Global Learning Consortium: About IMS. Disponível em: <http://www.imsglobal.org/aboutims.html>. Acesso em: mar. 2004.

JENNINGS, N. R. Coordination Techniques for Distributed Artificial Intelligence. In: JENNINGS, N. R.; O'HARE, G. M. P. (Ed.). **Foundations of Distributed Artificial Intelligence.** [S.l.:s.n.], 1996.

JESS. Java Expert System Shell. Disponível em: <http://herzberg.ca.sandia.gov/jess/>. Acesso em: jan. 2004.

JOHNSON, W. L.; SHAW, E. Using Agents to Overcome Deficiencies in Web-Based Courseware. In: WORLD CONFERENCE ON ARTIFICIAL INTELLIGENCE IN EDUCATION, AI-ED, 8., 1997. **Proceedings…** [S.l.: s.n.], 1997.

KEEGAN, D. **Foundations of distance education.** 2nd ed. London: Routledge, 1991.

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.], 2000. Disponível em: <http://fipa-os.sourceforge.net/docs/papers/soneraevaluation.pdf>. Acesso em: jan. 2003.

LONGMIRE, W. A Primer on Learning Objects. **Learning Circuits**, [S.l.], Mar. 2000. Disponível em: <http://www.learningcircuits.org/2000/mar2000/Longmire.htm>. Acesso em: abr. 2004.

LTSC: Learning Technology Standard Comitee of the Institute of Electrical and Electronics Engineers. Disponível em: <http://ltsc.ieee.org>. Acesso em: jan. 2004.

MAYFIELD, J.; LABROU, Y.; FININ, T. Evaluation of KQML as an Agent Communication Language. In: WORKSHOP AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, ECAI, 1995, Montreal, Canadá. **Proceedings...** Berlin: Springer-Verlag, 1996.

MERLOT: Multimedia Educational Resource for Learning and OnLine Teaching Home Page. Disponível em: <http://www.merlot.org/Home.po>. Acesso em: fev. 2005.

MOHAN, P.; BROOKS, C. Engineering a Future for Web-Based Learning Objects. In: INTERNATION CONFERENCE ON WEB ENGINEERING, ICWE, 2003, **Proceedings...** [S.l.]: Springer, 2003. p. 120-123.

MÜLLER, J. P. **The design of intelligent agents**: a layered approach. Heidelberg: Springer-Verlag, 1996. (Lecture Notes in Computer Science, v. 1177)

NOWOSTAWSKI, M.; PURVIS, M.; CRANEFIELD, S. A Layered Approach for Modelling Agent Conversations. In: INTERNATIONAL WORKSHOP ON INFRASTRUCTURE FOR AGENTS, MAS; SCALABLE MAS, 2001, Canadá. **Proceedings…** Montreal, Canada: [s.n.], 2001. p. 163-170.

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

QUINN, C. Learning Objects and Instruction Components. **Educational Technology & Society**, [S.l.], v.3, n.2, Feb. 2000. Disponível em: <http://ifets.ieee.org/discussions/discuss_feb2000.html>. Acesso em: mar. 2004.

RAO, A.; GEORGEFF, M. BDI agents: From theory to practice. In: INTERNATIONAL CONFERENCE ON MULTIAGENT SYSTEMS, ICMAS, 1., 1995. **Proceedings**... San Francisco: [s.n.], 1995.

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.

ROSCHELLE, J.; KAPUT, J.; STROUP, W.; KAHAN, T. M. Scaleable Integration of Educational Software: Exploring the Promise of Component Architectures. **Journal of Interactive Media in Education,** [S.l.], v.98, n.6, 1998. Disponível em: <http://www-jime.open.ac.uk/98/6/> Acesso em: jul. 2004.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence a Modern Approach**. [S.l.]: Prentice-Hall, 1995.

SADEK, M. D.; BRETIER, P.; PENAGET, F. **Submission for standardisation of components of France Télécoms ARTIMIS technology.** Turin: [s.n.], 1997.

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

SHOHAM, Y. Agent-oriented programming. **Artificial Intelligence**, Amsterdam, v.60, n.1, 1993.

SILVEIRA, R. A.; GOMES, E. R.; VICARI, R. M. Modelagem de ambientes de aprendizagem baseado na utilização de agentes FIPA. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, SBIE, 2003, Rio de Janeiro. **Anais...** Rio de Janeiro: UFRJ, NCE, 2003. p. 503-512.

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

SINGH, H. An Intro to Metadata Tagging . **Learning Circuits**, [S.l.], Dec. 2000. Disponível em: <http://www.learningcircuits.org/2000/dec2000/singh.html> Acesso em: abr. 2004.

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.], v.3, n.2, Oct. 2002. Disponível em: <http://www.irrodl.org/content/v3.2/soc-hes.html>. Acesso em: mar. 2004.

TOPCLASS. **TopClass e-Learning Suite**. Disponível em: <http://www.wbtsystems.com/products>. Acesso em: fev. 2005.

TORSUN, I.S. **Foundations of Intelligent Knowledge-Based Systems**. London: Academic Press, 1995.

VIRTUALU. eLearningSolutions Inc. Disponível em: <http://elearningsolutionsinc.com/productsvu.htm>.Acesso em: fev. 2005.

WEBCT. About Us. Disponível em: <http://www.webct.com/company>. Acesso em: jan. 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: mar . 2004.

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. 2004.

WOOLDRIDGE, M.; JENNINGS, N. R.; KINNY, D. A methodology for agent-oriented analysis and design. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 3., 1999. **Proceedings….** New York: ACM, 2002.