UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Júlia Marques Carvalho da Silva

DESENVOLVIMENTO DE UM FRAMEWORK PARA OBJETOS INTELIGENTES DE APRENDIZAGEM ADERENTE A UM MODELO DE REFERÊNCIA PARA CONSTRUÇÃO DE CONTEÚDOS DE APRENDIZAGEM

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Ricardo Azambuja Silveira, Dr.

Florianópolis (SC), dezembro de 2007

DESENVOLVIMENTO DE UM FRAMEWORK PARA OBJETOS INTELIGENTES DE APRENDIZAGEM ADERENTE A UM MODELO DE REFERÊNCIA PARA CONSTRUÇÃO DE CONTEÚDOS DE APRENDIZAGEM

Júlia Marques Carvalho da Silva

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação – Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Mário Antônio Ribeiro Dantas, Dr.

Banca Examinadora

_____

Ricardo Azambuja Silveira, Dr.

_____

Sílvia Modesto Nassar, Dra.

_____

Jovelino Falqueto, Dr.

_____

Rosa Maria Viccari, Dra.

AGRADECIMENTOS

Essa seção de agradecimentos tem muito significado para mim, afinal essas pessoas que aqui cito, foram essenciais para a minha sobrevivência durante os dois anos de curso. Digo sobrevivência nas mais diversas esferas: intelectual, financeira, afetiva e profissional. Ao final, pareceu rápido, mas nem por isso pouca intensa.

Primeiramente, gostaria de agradecer aos meus pais, Paulo Roberto Caldas da Silva e Silva e Mara Solange Marques Carvalho da Silva, pela força. Seja nas conversas, na espera a cada retorno de Florianópolis já de madrugada, pela cama e comida esperando, etc. Agradeço também ao meu namorado, Edson Dalfovo, pela compreensão exaustiva, só você mesmo para me aturar. A esse trio, um obrigada em especial a cada vez que eu pedia: “pode ler para ver se ficou bom?” ☺.

Também gostaria de agradecer aos meus dois orientadores durante o Mestrado. Primeiro, ao professor Arthur Buschsbaum, por me permitir o ingresso e, em um segundo momento, ao professor Ricardo Azambuja Silveira por me aceitar como orientanda. Ricardo, obrigada por acreditar em mim e me guiar nessa pesquisa. Às vezes, quando me sentia perdida, você me ensinou que eu chegaria lá, e ao ver o trabalho pronto constatei que você tinha razão.

Obrigada à banca examinadora, Rosa Viccari, Sílvia Nassar, Jovelino Falqueto e Alice Cybis; pelo carinho em avaliar e fornecer considerações essenciais para o fechamento do trabalho. Também um agradecimento especial a toda UFSC, e em especial, a secretaria do PPGCC pelo excelente apoio dado aos alunos, e por me permitir defender ainda em dezembro ☺.

Não posso deixar de agradecer àqueles que acompanharam o dia-a-dia no Mestrado. Rafael Vendruscolo e Álvaro Altair pela convivência animada e guerreira nas aulas de lógica. Beatriz Wilges pela amizade e acolhimento, que espero que continue mesmo depois dessa jornada. Mathias Weber, é irmãozinho, continuamos juntos nessa jornada. Benjamin Grando, começamos como ex-alunos, viramos colegas de Mestrado e agora colegas de docência, obrigada pela convivência e filosofias nas idas e vindas de Vavatur. Natanael Bavaresco, obrigada pelo companheirismo nesse último ano e sabes que podes contar comigo.

Também queria deixar meu profundo agradecimento àqueles que torceram diariamente por mim e não me deixaram abater diante das pedras que foram aparecendo no caminho, os meus amigos da UNIVALI. Alessandra de Oliveira, obrigada pelos cafés da manhã e voltinhas para

espairecer que tanto me ajudaram e me fizeram crescer. Elis Maschio, obrigada por sempre ter confiado em mim e pela torcida que só você é capaz. Julio Quirino e Mateus Pelloso, vocês são meus anjos da guarda, aqueles que podemos contar para o que der e vier. Francielle Duarte, mesmo um pouquinho distante, sua energia e torcida só me fortaleceram.

Agradeço também aqueles que ora foram meus tutores e agora colegas. Obrigada aos professores Fabiane Vavassori Benitti e Rudimar Dazzi pelas palavras de incentivo. Ao Luca e à Anita Fernandes por acreditarem no meu potencial e confiança em me permitir lecionar na UNIVALI, minha gratidão a vocês será eterna, e sabem que podem contar comigo para tudo. Também agradeço ao meu eterno orientador, André Raabe, pela torcida e pelos ensinamentos.

Deixo aqui registrado também o meu agradecimento ao professor Sérgio Fantini pela confiança e oportunidade dada na UNIFEBE. Ao Sr. Rodrigo Cabral por tudo que fizeste para que eu freqüentasse as aulas do Mestrado. Ao Rafael de Santiago pela ajuda no momento onde nada mais funcionava com os agentes ☺. E à Fernanda Debortoli pela ajuda com o Inglês.

Agradeço aos meus aluninhos da UNIVALI - São José e Itajaí, e UNIFEBE - Brusque por todo conhecimento e amizade durante este último ano. Em especial aos alunos Reginaldo Gonçalves e Fábio Duarte Machado por acreditarem nessa idéia e aceitarem fazer parte dela.

Por fim, agradeço a Deus pela saúde, e a São Judas Tadeu por me amparar e nunca deixar que eu desistisse dos meus sonhos.

SUMÁRIO

LISTA DE ABREVIATURAS

ACC	Agente de Canal de Comunicação
ACL	*Agent Communication Language*
AMS	*Agent Manager System*
API	*Application Programming Interface*
CCL	*Constraint Choice Language*
CNIPS	*Contract Net Interaction Protocol Specification*
DF	*Directory Facilitator*
FIPA	*Foundation for Intelligent Physical Agents*
GUID	*Globally Unique Identifier*
ILO	*Intelligent Learning Object* (Objeto Inteligente de Aprendizagem)
ILOR	*Intelligent Learning Object Repository* (Repositório de Objetos de Aprendizagem)
JADE	*Java Agent Development Framework*
KIF	*Knowledge Interchange Format*
LMS	*Learning Management System* (Sistema Gerenciador de Aprendizagem)
MaSE	*Multi-agent Software Engineering*
RDF	*Resource Description Framework*
RIPS	*Request Interaction Protocol Specification*
RMA	*Remote Management Agent*
RMI	*Remote Method Invocation*
RTE	*Run-Time Environment*
SCO	*Sharable Content Object*
SCORM	*Sharable Content Object Reference Model*
SL	*Semantic Language*
SMA	Sistema Multi-Agente

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

SILVA, Júlia Marques Carvalho da. **Desenvolvimento de um Framework para Objetos Inteligentes de Aprendizagem Aderente a um Modelo de Referência para Construção de Conteúdos de Aprendizagem.** Florianópolis, 2007. no 108. Dissertação de Mestrado (Pós-Graduação em Ciência da Computação)–Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2007.

Em sistemas de aprendizagem existem dois aspectos que devem ser considerados: a adaptabilidade, que diz respeito às diferentes necessidades e estilos de aprendizagem dos alunos; e a reusabilidade, que visa apoiar a confecção dos cursos. Acredita-se que estas características podem ser alcançadas através da interligação entre Sistemas Multi-agentes e Objetos de Aprendizagem. O resultado consiste na abordagem denominada de Objetos Inteligentes de Aprendizagem, cujo objetivo é promover experiências educacionais mais completas, além de possibilitar uma maior reutilização e adaptabilidade do conteúdo instrucional. Este trabalho propõe um *framework* para aplicação de objetos inteligentes de aprendizagem construídos sob o modelo SCORM. Para o desenvolvimento deste projeto foi necessária a modelagem do sistema multiagente e, posteriormente, sua implementação utilizando uma plataforma de agentes como suporte. No caso da modelagem, utilizou-se a metodologia MaSE (*Multi-agent System Engineering* - Engenharia de Sistemas Multi-agentes), que fornece especificações e diagramas para a análise e modelagem. Na implementação, os agentes foram desenvolvidos através da plataforma JADE, enquanto a ontologia para comunicação foi construída na ferramenta Protegé. Para a realização dos testes, foi utilizada a ferramenta eXe Learning devido a sua facilidade de utilização e exportação para o formato SCORM.

Palavras-chave: Objetos Inteligentes de Aprendizagem. SCORM. Sistemas Multi-Agentes.

ABSTRACT

There are two aspects that must be considered in learning systems: adaptability and reusability. Adaptability consists in being flexible according to different contexts and learning styles of students. Reusability aims to support courses design. These goals can be reached by joining Multi-agent Systems and Learning Objects technologies. The result consists on Intelligent Learning Objects (ILO), whose goal intends to promote richer learning experiences and provide reusability and adaptability of these objects in a more effective way. This work proposes a framework to be applied in intelligent learning objects developed with SCORM standard, based in a previous group research. For the development of this project it was necessary design and implementing multi-agent system, using methodologies and tools to support the research. It was used MaSE (Multi-agent System Engineering) methodology for design model, which provides specifications and diagrams for the analysis and modeling. In the development phase, the agents were developed by JADE platform and the ontology was done with Protégé Tool. For the tests phase, it was used eXe Learning tool, because of your using facilities and the way to export to a SCORM format.

Keywords*: Intelligent Learning Objects. SCORM. Multi-agent systems.*

1 INTRODUÇÃO

A educação à distância (EaD) é uma modalidade de ensino-aprendizagem que vem sendo amplamente pesquisada e utilizada, especialmente após a popularização da Internet. A facilidade de comunicação e acesso proporcionada pela Internet, em conjunto com o uso de recursos multimídia para suplementação dos textos e materiais de apoio, tornaram-se diferenciais importantes. Além disso, houve uma preocupação em desenvolver ambientes virtuais onde os materiais didáticos pudessem ser disponibilizados.

Nesse sentido, os esforços foram reunidos para que algumas tarefas rotineiras do processo de ensino-aprendizagem fossem automatizadas. Ainda, peças inteligentes de software foram desenvolvidas de modo a identificar as necessidades individuais, tornando os ambientes virtuais de ensino-aprendizagem mais dinâmicos e interessantes ao aluno. Já sob o ponto de vista do professor, defende-se a construção de ambientes com assistentes inteligentes de ensino[1], os quais são responsáveis por fornecer informações relevantes sobre o processo de ensino de forma a apoiar o professor.

Pesquisadores e organizações têm se dedicado a formalizar a construção e disponibilização de conteúdos didáticos. Os estudos nessa área surgiram da percepção de que havia (e ainda há) um baixo reaproveitamento do material produzido. Normalmente, cada curso é construído por seu professor responsável e por ele utilizado. Quando um curso semelhante, se não com a mesma ementa, é promovido novamente, o processo se repete. Isto gera um alto custo tanto para o professor, que na ausência de materiais acaba produzindo seus próprios, quanto para a instituição promotora que os custeia.

Nesse sentido, surgiu o conceito de objetos de aprendizagem, o qual visa fornecer formalismos para o desenvolvimento de conteúdos didáticos digitais que sejam acessíveis em qualquer ambiente virtual de ensino-aprendizagem (interoperável) e aplicáveis em diferentes contextos (reusabilidade). Os objetos de aprendizagem consistem em recursos (hipertexto, figuras, som, vídeo) capazes de promover experiências de aprendizagem. Entretanto, para que haja um entendimento entre os objetos de aprendizagem, faz-se necessária a utilização de uma comunicação

---

[1] Assistentes Inteligentes de Ensino (*Intelligent Teaching Assistants* – ITA): propostos por YACEF (2002) tem por objetivo fornecer mecanismos que auxiliem os professores em suas atividades de tutoramento em ambiente virtuais de aprendizagem.

comum, promovida através de padrões para a sua construção, e mecanismos que permitam sua localização.

Um das finalidades do desenvolvimento dos objetos de aprendizagem é promover o reaproveitamento dos conteúdos, de forma a reduzir o tempo de criação dos mesmos, adaptando aqueles já existentes. Entretanto, o uso exclusivo de modelos de referência para a sua construção não garantem o sucesso de uma busca efetiva por objetos relacionados em um determinado contexto, nem a reutilização de objetos em contextos diferentes daqueles onde foram criados. Logo, é necessário a adição de elementos da Inteligência Artificial, a fim de disponibilizar um ambiente comum onde ocorra troca de informações entre tais objetos, de forma mais eficaz.

Trabalhos anteriores desenvolvidos pelo grupo de pesquisa (GOMES *et al.*, 2003; GOMES, 2005; SILVEIRA *et al.*, 2005; SILVEIRA *et al.*, 2006; SILVEIRA *et al.*, 2007; WILGES *et al.*, 2007) discutem como agentes inteligentes podem auxiliar alunos e tutores na elaboração de cursos cujos conteúdos sejam formados por objetos de aprendizagem. Os alunos realizam experiências de aprendizagem mais ricas, pois os agentes inteligentes são desenvolvidos com a possibilidade de identificar, em conjunto, quais conteúdos são mais eficazes ao seu perfil. Por outro lado, os professores podem ser auxiliados na construção de cursos, a partir da busca e adaptação de objetos de aprendizagem correlatos. A união de tais temáticas originou a pesquisa sobre Objetos Inteligentes de Aprendizagem. Como resultado, é proposto um *framework*[2] onde um conjunto de agentes representa o ambiente gerenciador de aprendizagem, os repositórios de objetos e os próprios objetos de aprendizagem. Além disso, foi introduzida a idéia do uso de uma ontologia contendo conceitos e ações os quais o sistema multi-agente pode a utilizar dentro de seu ciclo de vida.

A partir da análise das pesquisas desenvolvidas até então pelo grupo, percebeu-se a oportunidade de continuidade e avanço do trabalho, dando ênfase em aspectos que se julgou ainda em aberto, tais como: a necessidade de uma formalização mais explícita da arquitetura de sistemas multi-agentes desenvolvida, especialmente no que se refere aos protocolos de interação e à especificação da ontologia, utilizada através de recursos da Engenharia de Software. Além disso,

---

[2] *Framework* é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido. Ele pode incluir programas de suporte, bibliotecas de código, linguagens de script e outros softwares para ajudar a desenvolver e juntar diferentes componentes de um projeto de software. Ainda, são projetados com a intenção de facilitar o desenvolvimento de software, habilitando designers e programadores a gastarem mais tempo determinando as exigências do software do que com detalhes tediosos de baixo nível do sistema.

percebeu-se a necessidade de promover o reaproveitamento dos conteúdos de forma mais efetiva, adotando padrões consolidados para o desenvolvimento de objetos de aprendizagem, tal como o SCORM[3].

1.1 PROBLEMATIZAÇÃO

A partir da análise realizada sobre a proposta dos Objetos Inteligentes de Aprendizagem foram identificadas algumas limitações, às quais o presente trabalho propõe alguns avanços. O desenvolvimento de um Sistema Multi-agente, semelhantemente a um desenvolvimento adequado de um software qualquer, sem um projeto bem definido, pode acarretar em lacunas não trabalhadas. Neste sentido, a área de Engenharia de Software fornece suporte tecnológico através do uso de metodologias adequadas e da aplicação de seus artefatos.

Em seu estado original, o projeto desenvolvido por GOMES (2005) não havia tido até então seu modelo suficientemente testado e avaliado. O *framework* proposto foi desenvolvido a partir da percepção intuitiva sobre o problema identificado, sem a utilização de técnicas formais específicas para modelagem da arquitetura do sistema multi-agente. Isto também se aplica a ontologia proposta, a qual descreve os conceitos de forma geral, delimitando uma área de texto livre para especificação das características do objeto, dificultando a seleção efetiva de seu conteúdo. Logo, percebe-se a necessidade de revisão desta ontologia original, refinando o conceito de objetos de aprendizagem. Acredita-se que a formalização de suas informações poderá fornecer maior precisão na localização do objeto, seja para a promoção da reusabilidade, seja para uma experiência de aprendizagem mais efetiva.

Quanto à avaliação da aplicação do modelo, Wilges *et al*. (2007) apresentaram um estudo preliminar sobre a aplicação do modelo proposto, através da construção de um conjunto de objetos inteligentes de aprendizagem para apoio ao ensino de matemática.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

[3] Um melhor detalhamento encontra-se na seção 0.

Desenvolver um *framework* para a implementação de ambientes virtuais de aprendizagem baseados em Objetos Inteligentes de Aprendizagem, aderente a um modelo de referência para desenvolvimento de conteúdo educacional baseado em objetos de aprendizagem.

1.2.2 Objetivos Específicos

- Selecionar e aplicar uma metodologia de Engenharia de Software aplicada a Sistema Multi-Agente para modelagem dos Objetos Inteligentes de Aprendizagem;
- Integrar o Sistema Multi-Agente desenvolvido ao modelo de construção de objetos que utilizam modelo de referência para desenvolvimento de objetos de aprendizagem; e
- Validar o framework a partir do desenvolvimento de uma aplicação.

1.3 PROCEDIMENTOS METODOLÓGICOS

Os procedimentos metodológicos utilizados para o desenvolvimento deste trabalho iniciam-se pela revisão dos principais conceitos teóricos necessários para a compreensão do modelo e desenvolvimento do SMA, cuja proposta foi iniciada por Gomes (2005). Esta revisão concentra os principais conceitos relativos à pesquisa: agentes e objetos de aprendizagem. Sobre a tecnologia de agentes, apresentam-se os conceitos relacionados a sua estrutura individual e seu comportamento, quando necessitam trabalhar em grupo, a fim de atingir um objetivo. Para tanto, são necessários determinados formalismos para que os agentes se compreendam através de um mecanismo comum de comunicação. Neste caso, o padrão de comunicação entre agentes definidos pela organização FIPA (*Foundation for Intelligent Physical Agents*) que fornece a especificação necessária, incluindo as mensagens e protocolos de comunicação, foi adotado neste projeto.

Na seqüência são estudados alguns ambientes para desenvolvimento de sistemas multi-agentes. A necessidade deste estudo deve-se ao fato da escolha da plataforma FIPA-OS[4] no trabalho original de Gomes (2005). Esta plataforma encontra-se atualmente em processo de padronização, que como conseqüência, não foram lançadas atualizações de sua plataforma.

A verificação de outras soluções de plataformas para desenvolvimento de sistemas multi-agentes deveria promover a comunicação através do padrão FIPA, que assegura a comunicação dos

[4] É uma plataforma que disponibiliza um conjunto de componentes para a implementação de agentes compatíveis com os padrões FIPA (FIPA-OS, 2007).

agentes desenvolvidos originalmente e os propostos aqui, além de agentes especificados por outros projetos e sistemas.

Ainda na conceitualização da pesquisa, é abordado a importância de modelar e especificar sistemas multi-agentes, utilizando mecanismos formais de Engenharia de Software a fim de assegurar a robustez do projeto. Conforme será mostrado, existem algumas propostas de metodologias de Engenharia de Software adequadas ao paradigma de agentes, inclusive fornecendo ferramentas para a sua aplicabilidade. Logo, é apresentada uma seção com a descrição de algumas destas metodologias, com destaque para a metodologia empregada no desenvolvimento do trabalho.

Também foi verificada a necessidade de um estudo sobre ontologia, a qual dá suporte a troca de informações entre os agentes. A ontologia é para os agentes, como um contexto conceitual, através do qual eles se compreendem num processo de comunicação. A ontologia deve ser integrada ao desenvolvimento dos agentes, de acordo com o modelo de referência da FIPA.

Por fim, são discutidos os conceitos acerca dos objetos de aprendizagem e suas entidades correlatas. Uma das características fundamentais dos objetos de aprendizagem, diz respeito a sua reutilização. Para tanto, é necessário a utilização de um esquema de padronização que facilite o seu funcionamento. Atualmente, existem diversos padrões utilizados para o apoio no desenvolvimento dos conteúdos de aprendizagem, alguns dos quais já estão presentes em ferramentas de autoria de objetos e em ambientes virtuais de aprendizagem, responsáveis pela execução de tais objetos. No presente trabalho apresenta-se alguns padrões de construção de objetos de aprendizagem e a seleção de um destes para ser integrado ao SMA. Esta integração resulta nos objetos inteligentes de aprendizagem, conceito introduzido por Gomes (2005) e Silveira (2004), o qual é apresentado sua conceitualização e arquitetura proposta.

A partir da compreensão destes conceitos, é apresentada uma proposta de modelagem do SMA e da ontologia a ser compartilhada pelos agentes, utilizando uma metodologia selecionada durante a etapa de Fundamentação Teórica. Esta modelagem é desenvolvida a partir do trabalho desenvolvido por Gomes (2005), o qual foi refinado conforme a necessidade. A ontologia originalmente proposta também foi reformulada para adequá-la a um formalismo de dados estabelecidos pelo padrão de desenvolvimento de objetos de aprendizagem, alterando consequentemente os diálogos originalmente estabelecidos.

O desenvolvimento do SMA buscou apresentar uma avaliação da aplicabilidade dos objetos inteligentes de aprendizagem de acordo com o modelo proposto. Para que isto fosse possível, foi necessário a reformulação dos agentes desenvolvidos, o que inclui uma série de atualizações, tais como: a plataforma tecnológica de desenvolvimento dos agentes e a estrutura de comunicação entre os agentes.

A fim de validar o *framework* proposto para o desenvolvimento de Objetos Inteligentes de Aprendizagem, foi selecionado um domínio de conteúdos e construído alguns objetos com o auxílio de uma ferramenta de desenvolvimento. A ferramenta foi empregada para gerar objetos compatíveis com o padrão de metadados definido.

1.4 ESTRUTURA DO TRABALHO

O trabalho é organizado em seis capítulos: 1. Introdução, 2. Objetos de Aprendizagem, 3. Agentes, 4. Objetos Inteligentes de Aprendizagem; 5. Desenvolvimento do *Framework* e 6. Considerações Finais. Na Introdução são apresentadas as motivações e justificativas para a realização desta dissertação, bem como os objetivos a serem contemplados com a pesquisa.

Os capítulos 2, 3 e 4 abordam os conceitos necessários para a compreensão e desenvolvimento desta pesquisa. No capítulo 2 ocorre a conceitualização dos objetos de aprendizagem e seus formalismos para construção de conteúdos reutilizáveis, tais como a estrutura e conjunto de metadados. O Capítulo 3 introduz o conceito de agentes e sistemas multi-agentes, tendo como foco principal aspectos sobre a padronização para a comunicação de agentes. Além disto, são descritos alguns ambientes para desenvolvimento de sistemas multi-agentes. No Capítulo 4 apresenta-se a abordagem dos Objetos Inteligentes de Aprendizagem e a idéia de unir o paradigma dos objetos de aprendizagem ao dos agentes. Também é apresentado o modelo inicialmente proposto no trabalho de Gomes (2005) e propõe avanços a partir das percepções através da continuidade da pesquisa. A modelagem envolve o desenvolvimento de modelos (ex: diagramas) e da revisão da ontologia.

O desenvolvimento do *framework* proposto é descrito no Capítulo 5, que apresenta a arquitetura tecnológica desenvolvida e o estudo de caso de uma aplicação para avaliação da potencialidade do uso de Objetos Inteligentes de Aprendizagem.

No Capítulo 6 são expostas as considerações (percepções, dificuldades e sucessos) sobre cada uma das etapas da pesquisa realizada.

Após as Referências, ainda são apresentadas as publicações obtidas a partir dos resultados preliminares alcançados durante a pesquisa.

2 OBJETOS DE APRENDIZAGEM

Um Objeto de Aprendizagem é um pedaço de conteúdo de aprendizagem que pode ser aplicado ao aprendizado em diversos momentos, e em diferentes cursos e situações (DOWNES, 2001; MOHAN & BROOKS, 2003; e SOSTERIC & HESEMEIER, 2002). A IEEE (2006) define que um Objeto de Aprendizagem é qualquer entidade, digital ou não, que pode ser usado para aprendizagem ou treinamento.

Os objetos são geralmente armazenados em repositórios especialmente desenhados e podem ter seus métodos invocados por algum Sistema Gerenciador de Aprendizagem (*Learning Management System* - LMS) e, de um modo geral estão limitados a funcionar de uma forma específica. Estes conceitos são ilustrados na Figura 1.

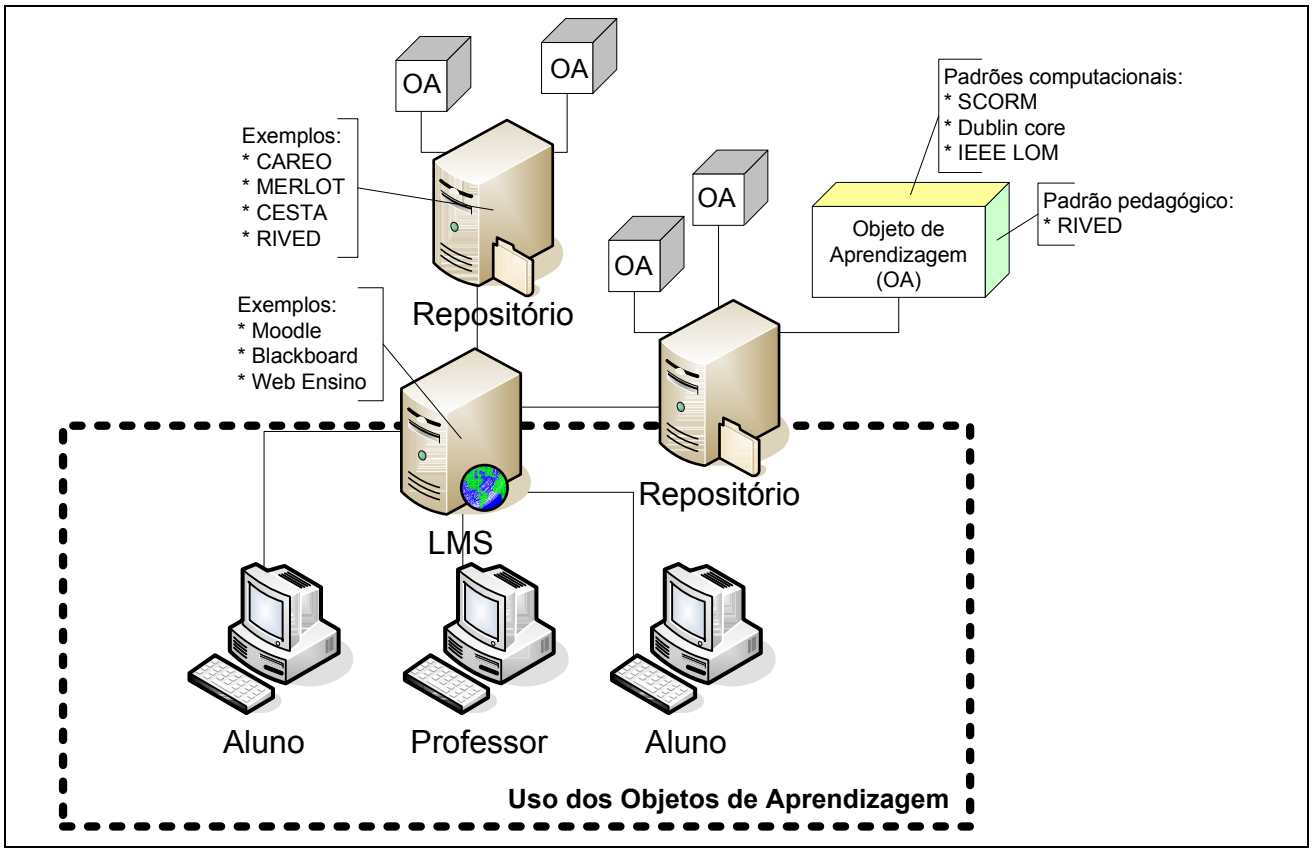

Figura 1. Entidades relacionadas aos objetos de aprendizagem

Conforme foi mencionado, os objetos de aprendizagem são catalogados e armazenados em um sistema denominado como repositório. A utilização de um repositório adequado implica na facilitação das buscas, tornando-as mais eficientes, em termos de recuperação dos dados (SANTOS,

2007). Entretanto, para que isto seja possível, os objetos devem ser associados a um conjunto de informações que especifiquem corretamente seu conteúdo. Esse conjunto costuma ser denominado de metadados (dados sobre os dados).

Os repositórios devem fornecer mecanismos para: (i) gerenciamento dos objetos de aprendizagem - tais como a inclusão, alteração e exclusão; e (ii) localização dos objetos – a partir de um conjunto de dados especificados e comuns aos objetos, ficando a critério de cada repositório como realizar esta localização. Podem ser citados alguns respositórios: Campus Alberta Repository of Educational Objects – CAREO (2007), Multimedia Educational Resource for Learning and Online Teaching – MERLOT (2007), eduSource, iLumina (2007), Coletânea de Entidades Superiores ao uso de Tecnologia de Aprendizagem – CESTA, Rede RIVED (2007). Tais repositórios foram desenvolvidos com o escopo geral (atendendo qualquer área de ensino) ou específico (ex: humanas, ciências, exatas, etc.).

Outro elemento importante é o *Learning Management System* (LMS). O LMS consiste em um sistema de gestão que possui um conjunto de funcionalidades para promover aprendizado. Essas funcionalidades relacionam-se com distribuição, acompanhamento, monitoramento e administração de conteúdo de aprendizagem e com o progresso e interações dos aprendizes, entre outros. O termo LMS pode ser aplicado a sistemas simples de gerenciamento de cursos ou a complexos ambientes distribuídos.

Um LMS tem como um dos objetivos, simplificar a administração dos programas de treinamento e ensino em uma organização. O sistema auxilia no planejamento dos processos de aprendizagem e ainda permite que os participantes colaborem entre si através da troca de informações e conhecimentos.

Existem LMS que são comercializados, tais como: Blackboard (2007), Web Ensino (2007) e WebAula (2007); e outros que são distribuídos com código-aberto: Moodle (2007), Claroline (2007), Dokeos (2007) e TelEduc (2007).

Esses sistemas auxiliam a análise, a disponibilidade das informações, o rastreamento de dados, e a geração de relatórios sobre o progresso dos aprendizes. A maioria dos LMS não possui recursos que permitem a rápida e simples criação de conteúdos instrucionais, e este é um dos principais motivos pelo qual a maioria das empresas fornecedoras tem procurado oferecer

ferramentas complementares, ou trabalhar com parceiros para desenvolvimento de ferramentas de autoria.

2.1 ESPECIFICAÇÕES PARA OBJETOS DE APRENDIZAGEM

Nos últimos anos, muitos pesquisadores têm dado atenção para a questão da reusabilidade dos objetos, gerando pesquisas com o foco na definição de padrões para especificação de metadados que facilitem a recuperação de recursos na Internet. Duval (2004) descreve que há uma hierarquia de organizações, que produzem e aprovam especificações. Tais especificações podem ser agrupadas em dois níveis:

- Nível 1: composto por organizações como a IEEE LTSC e CEN/ISSS WSLT, as quais são responsáveis pela definição das necessidades e requisitos do domínio, mantendo seu processo de especificação acessível a todos, permitindo uma contribuição constante da comunidade científica;
- Nível 2: representado por consórcios como AICC, IMS, ADL e ARIADNE. Fornecem documentos técnicos baseados em um processo interno, para que atinja as necessidades e os requisitos dos membros da organização. No entanto, tais especificações não são consideradas padrões oficiais, já que as mesmas não seguem os requisitos e as necessidades do domínio educacional, etc.

A autora acredita na existência de um terceiro nível, onde estão contempladas organizações como a RIVED (2007), que investe na promoção do desenvolvimento de objetos de aprendizagem com forte sustentação pedagógica. Ainda, sugerem como o processo de um objeto deve ocorrer, envolvendo equipe multidisciplinar, cujos papéis são bem definidos.

Os subitens a seguir apresentam alguns dos formalismos, relacionados à construção de objetos de aprendizagem, propostos em cada um dos níveis citados: (i) LOM, que consiste em um conjunto de metadados definidos pela IEEE; (ii) SCORM, que referencia a construção de objetos sob o ponto de vista tecnológico; e (iii) RIVED, que orienta as equipes desenvolvedoras quanto às perspectivas pedagógicas na definição da abordagem do conteúdo dentro dos objetos.

2.1.1 Padrão LOM IEEE

A utilização do conceito de metadados, ou dados sobre dados, funciona de forma semelhante a um catálogo de biblioteca. Eles fornecem informações sobre um determinado recurso, promovendo a interoperabilidade, identificação, compartilhamento, integração, utilização, reutilização, gerenciamento e recuperação dos mesmos de maneira mais eficiente. São dados descritivos que podem informar sobre o título, autor, data, publicação, palavras-chaves, descrição, localização do recurso, entre outros. Eles podem ser comparados a um sistema de rotulagem que descreve o recurso, seus objetivos e características, mostrando como, quando e por quem o recurso foi armazenado, e como está formatado. Os metadados são essenciais para entender o recurso armazenado, eles descrevem informações semânticas sobre o recurso (DeMARCHI & COSTA, 2004).

Os metadados podem ser diferenciados em objetivos ou subjetivos (HODGINS, 2002):

- Metadados objetivos: são gerados automaticamente, descrevendo atributos físicos, data, autor, requisitos operacionais, custos, número de identificação, proprietário, etc.;
- Metadados subjetivos: consistem em atributos variados e são determinados pela pessoa ou grupo que cria o metadado, são definições que dependem do conhecimento, contexto, perspectiva ou opinião.

Um exemplo prático seriam as informações de uma lata de molho de tomate, onde o rótulo provê metadados objetivos, no entanto, a opinião sobre o funcionamento desse molho como um bom ingrediente em uma determinada receita é um metadado subjetivo.

Existe um padrão aceito e definido pela IEEE que especifica os metadados que descrevem um objeto de aprendizagem: IEEE 1484.12.1-2002 *Draft Standard for Learning Object Metadata* (IEEE LOM, 2002). Os elementos que compõem um objeto de aprendizagem estão agrupados em nove categorias:

- Geral (*General*): agrupa informações gerais que descrevem os objetos de aprendizagem como um todo;
- Ciclo de vida (*Lifecycle*): agrupa as características relacionadas com a história e o estado atual do objeto de aprendizagem e como estas têm afetado o objeto durante a sua evolução;
- Meta-Metadado (*Meta-Metadata*): agrupa informações sobre a instância de metadados;

- Técnico (*Technical*): agrupa as características e os requisitos técnicos do objeto de aprendizagem;
- Educacional (*Educational*): agrupa as características educacionais e pedagógicas do objeto de aprendizagem;
- Direitos (*Rights*): agrupa as propriedades intelectuais e condições de uso para o objeto de aprendizagem;
- Relações (*Relation*): agrupa características que definem o relacionamento entre o objeto de aprendizagem e demais objetos de aprendizagem relacionados;
- Anotação (*Annotation*): provê os comentários sobre o uso educacional do objeto de aprendizagem, além de prover informações de quando e por quem os comentários foram criados;
- Classificação (*Classification*): descreve este objeto de aprendizagem em relação a um sistema de classificação em particular.

Baldoni *et al.* (2004) afirma que o uso de metadados potencializa a preparação de cursos porque pode unir dinamicamente os objetos de aprendizagem utilizados no curso, baseado nos objetivos a serem atingidos e se adequando às estratégias de aprendizagem ideal às necessidades do aluno. A arquitetura do Sistema de Gerenciamento de Aprendizagem por ser ampliado através da aplicação de técnicas de Inteligência Artificial e da introdução de novos componentes a fim de tornar os LMSs mais inteligentes, capazes de interagir com o usuário (ou com um agente que o requer) para atingir os objetivos de aprendizagem desejados conforme as condições, que em alguns casos vão ao encontro um ao outro.

2.1.2 SCORM

O SCORM (*Sharable Content Object Reference Model*) é um modelo de referência para objetos de conteúdo compartilháveis, que corresponde a uma coleção de especificações técnicas desenvolvidas por múltiplas organizações (AICC, IEEE, IMS, etc.) de forma a assegurar reusabilidade, acessibilidade, durabilidade e interoperabilidade em conteúdos de aprendizagem baseados na web. Além disso, estabelece normas para o desenvolvimento, empacotamento e entrega de materiais (ADL, 2006).

A especificação SCORM visa ainda atender os seguintes requisitos:

- Reusabilidade: Um objeto deve ser facilmente modificado e usado por diferentes ferramentas de desenvolvimento e plataformas, além de ser aplicável em múltiplos contextos;
- Acessibilidade: Pode ser encontrado e tornado disponível, se possível, por aprendizes ou por desenvolvedores de conteúdos, de qualquer local remoto;
- Interoperabilidade: Um objeto deve ser operável em diversos tipos de *hardware*, sistemas operacionais e navegadores web; e
- Durabilidade: Não deve ser necessário realizar modificações significativas (reconfigurar, reimplementação) para adequar a novas versões de software.

Durante o desenvolvimento deste trabalho, o SCORM encontra-se na versão 2004 3ª Edição, contemplando o seguinte conjunto de especificações (ADL, 2006):

- Modelo de Agregação de Conteúdo (*Content Aggregation Model*): que inclui o dicionário de metadados, empacotamento de conteúdo, estrutura de conteúdo, boas práticas e metadados obrigatórios em XML;
- Ambiente de Execução (*Run Time Environment*): que contempla o modelo de dados, API de comunicação e inicialização; e
- Seqüenciamento e Navegação (*Sequencing and Navigation*): é responsável pela apresentação dinâmica de conteúdos baseados nas necessidades do aprendiz.

Para a compreensão do presente trabalho, faz-se necessário um entendimento mais detalhado sobre os dois primeiros itens. O Modelo de Agregação de Conteúdo é responsável pela classificação dos objetos de aprendizagem, enquanto o Ambiente de Execução é onde ocorre a implementação de fato. As informações sobre o Seqüênciamento e Navegação foram incorporadas recentemente no SCORM, mais precisamente durante a realização deste trabalho, não sendo viável um estudo detalhado sobre sua adequação dentro do escopo proposto. Entretanto, não se descarta sua incorporação na arquitetura de Objetos Inteligentes de Aprendizagem em trabalhos futuros.

2.1.2.1 Modelo de Agregação de Conteúdo

A proposta do Modelo de Agregação de Conteúdo (*Content Aggregation Model*) é prover um conceito comum para a construção de conteúdos de aprendizagem, que possibilita descobrir,

reutilizar, compartilhar e interoperá-los. O modelo define que conteúdos de aprendizagem podem ser identificados e descritos, agregando-os em um curso ou em parte de um curso, possibilitando que o LMS possa recuperar objetos do repositório. O modelo inclui especificações para agregar conteúdo e definir metadados.

O Modelo de Agregação de Conteúdo é composto por (QU & NEJDL, 2001):

- Modelo de conteúdo (*Content Model*): Nomenclatura definida nos componentes de conteúdo de experiências de aprendizagem. Três tipos de componentes são:
  - Recursos (*Assets*): é a menor unidade física dentro de um material, possuindo a alta capacidade de reutilização devido a existência de um arquivo de metadados que o especifica;
  - Objeto de Conteúdo Compartilhado (*Sharable Content Object* - SCO): trata-se de uma coleção de SCOs capaz de originar um material instrucional;
  - Organização de Conteúdo (*Content Aggregation*): consiste em um mapa que orienta o uso pretendido de um conjunto de SCOs;
- Metadados (*Meta-data*): Um arquivo de manifesto que descreve as instâncias específicas de componentes do modelo de conteúdo;
- Empacotamento de conteúdo (*Content Packing*): Define como empacotar recursos de aprendizagem para trabalhar com diferentes LMSs.

Na terminologia SCORM as unidades de aprendizagem são chamadas de SCO, e a sua estrutura e as regras que regem as atividades de aprendizagem, são definidos em um arquivo de manifesto do SCO. Cada manifesto descreve tanto a estrutura do material de aprendizagem quanto à forma que este é apresentado. As regras são descritas utilizando três operadores: seqüencial, se-então, e um conjunto de itens de aprendizagem que podem ser explorados livremente pelos usuários. Estes operadores permitem a descrição de conjuntos de objetos de aprendizagem organizados como uma árvore cujos nodos (itens) representam sub-atividades. A decisão de qual o item deve ser mostrado na seqüência é realizado pelo LMS, baseado nas regras contidas no arquivo de manifesto e nas características que dependem do comportamento do usuário. Por exemplo, pode-se verificar se o usuário leu um item anterior, ou se ele não respondeu a questão corretamente (Baldoni *et al.*, 2004).

2.1.2.2 Ambiente de Execução

A proposta do Ambiente de Execução (*Run-Time Environment*) é formalizar os mecanismos para promoção da interoperabilidade entre o LMS e os objetos de aprendizagem no padrão SCORM. O SCORM requer que o conteúdo de aprendizagem seja acessível em múltiplos LMS. Logo, para que isso seja possível, eles precisam ter uma forma comum de ter seu conteúdo inicializado, de se comunicar com o LMS, e de se acessar durante sua execução. Os três componentes do SCORM Run-Time Environment são: inicialização, API e modelo de dados (ADL, 2006; QU & NEJDL, 2001).

- Inicialização (*Launch*): Define uma forma comum para o LMS iniciar recursos de aprendizagem baseados na Web. Isso define um procedimento e responsabilidades para o estabelecimento de comunicação entre recursos de aprendizagem e o LMS;
- API: Provê um mecanismo de comunicação para a informação do LMS sobre o estado do recurso de aprendizagem (ex: inicializado, finalizado e uma condição de erro). A API é, portanto, também utilizada para recuperar e registrar dados entre o LMS e o SCO;
- Modelo de dados: Fornece um padrão de conjunto de elementos de dados para definir a informação de comunicação, como o status de recurso de aprendizagem.

Para o presente trabalho há um interesse especial no estudo do conjunto de metadados, devido a sua aplicabilidade direta na especificação da ontologia dos agentes, e da API, responsável pela comunicação entre os objetos e o ambiente onde ele está disponível.

API SCORM

O objetivo do padrão SCORM é que os objetos de aprendizagem sejam reutilizáveis e interoperáveis através dos diversos LMS. Para que isto ocorra, eles precisam ter uma forma única de serem iniciados, de se comunicarem com o LMS, e uma linguagem pré-definida de comunicação. A Figura 2 ilustra como o ambiente de Run-Time deve funcionar, comunicando o objeto de aprendizagem ao LMS através do adaptador da API, chamado de APIAdapter (ADL – Run-Time, 2001).

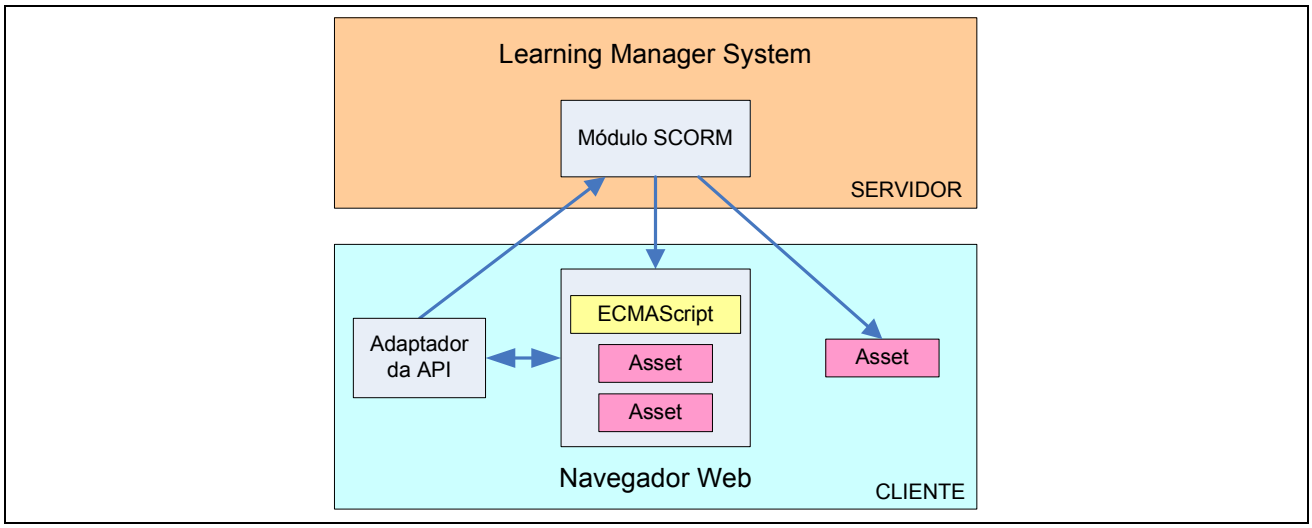

Figura 2. Ambiente de Execução do SCORM
Fonte: Adaptado de ADL (2006)

O uso de uma API comum provê uma forma de padronização para que os SCOs se comuniquem com o LMS, encapsulando detalhes de programação nem sempre interessantes ao conteudista[5]. Em termos gerais, uma API contém um conjunto de funções pré-definidas que o SCO pode acessar quando estiver ativo. O adaptador de API é um pedaço de software que somente expõe as funções da API. Logo, o LMS apenas precisa implementar este adaptador de forma a permitir que o objeto tenha acesso a informações que possa vir utilizar, por exemplo, o nome do aluno ou em que parte do objeto o aluno parou na última vez que o acessou. Tais dados são especificados pelo modelo de dados que o LMS deve implementar a fim de registrar as informações sobre a interação do aluno a um objeto de aprendizagem específico. A Figura 3 apresenta os conceitos presentes no modelo de dados que o objeto de aprendizagem pode solicitar para conhecer e se adequar ao aluno, bem como registrar os resultados obtidos pelo aluno.

[5] Profissional especializado na produção materiais didáticos.

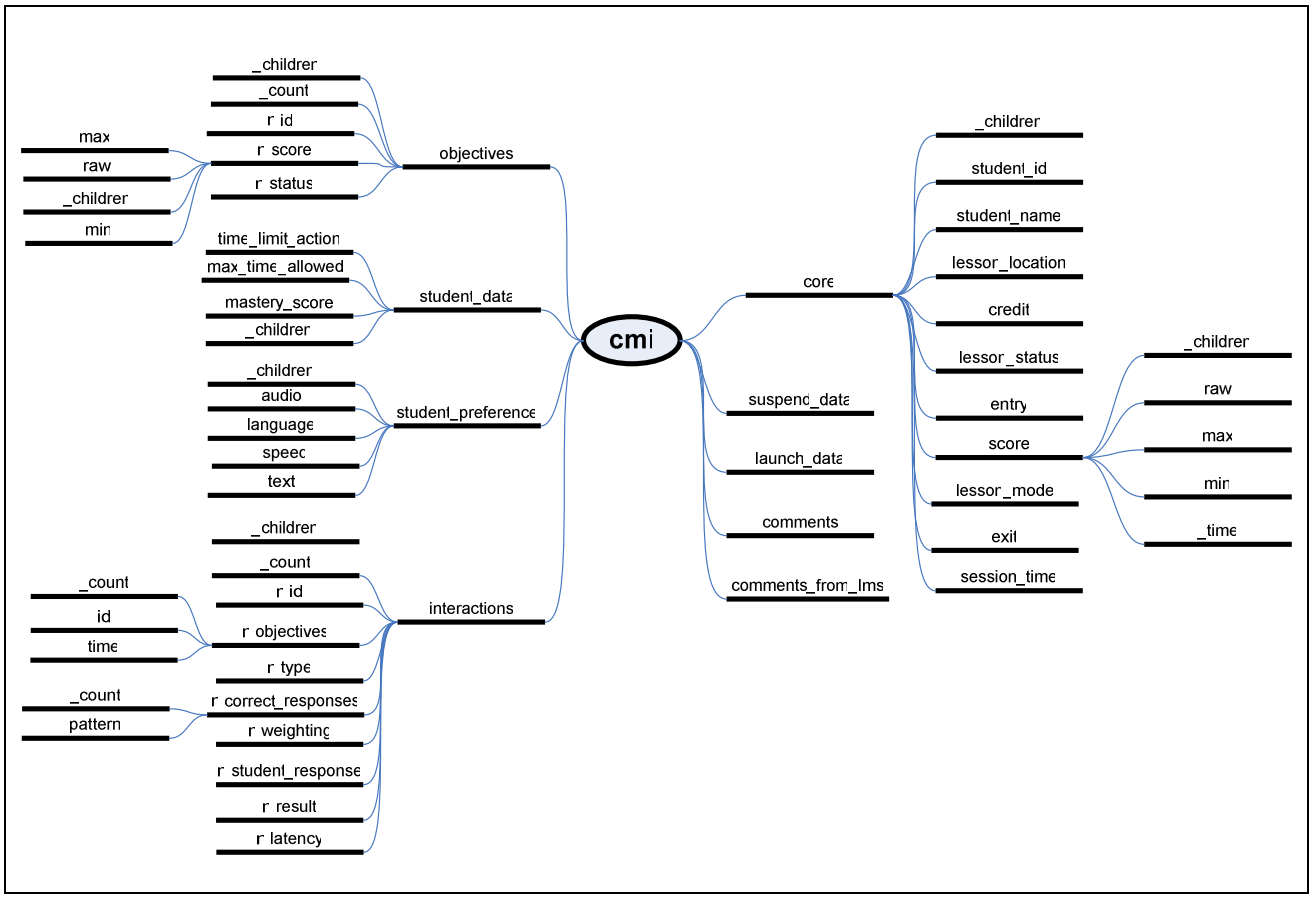

Figura 3. Modelo de dados SCORM

Fonte: Baseado em ADL (2001)

Do lado cliente localizam-se: o adaptador de API, o ECMAScript e os *Assets*. Geralmente, o adaptador de API é desenvolvido em uma linguagem de *script* executada no computador onde o objeto é visualizado, por exemplo, em JavaScript. O adaptador agrupa um conjunto de funções apresentado na Tabela 1.

Tabela 1. Classificação das funções da API do SCORM

Classificação	Função	Descrição
Estado de Execução	LMSInitialize	Indica ao adaptador da API que o SCO irá se comunicar com o LMS.
	LMSFinish	O SCO deve chamar esta função quando não precisará mais se comunicar com o LMS.
Transferência de Dados	LMSGetValue	Permite que o SCO obtenha uma informação do LMS.
	LMSSetValue	Permite que o SCO envie uma informação ao LMS.
	LMSCommit	Permite que o SCO solicite ao LMS que registre os valores enviados a ele de forma persistente.
Gerenciamento do Estado	LMSGetLastError	Possibilita ao SCO saber se alguma operação falhou ou não.
	LMSGetErrorString	Retorna uma mensagem textual sobre o erro obtido.
	LMSGetDiagnostic	Fornece informações adicionais sobre um determinado erro corrente.

Fonte: Adaptado de ADL (2006)

A comunicação do SCO com o LMS através de uma instância do adaptador da API, atravessa por diversos estados, conforme ilustra a Figura 4. Os estados do adaptador da API especificam as respostas fornecidas dada à ocorrência de um evento. Durante cada um destes estados há um conjunto de diferentes atividades que o SCO pode realizar. Os estados transpassados pela API são: "Não Iniciado", "Iniciado" e "Encerrado".

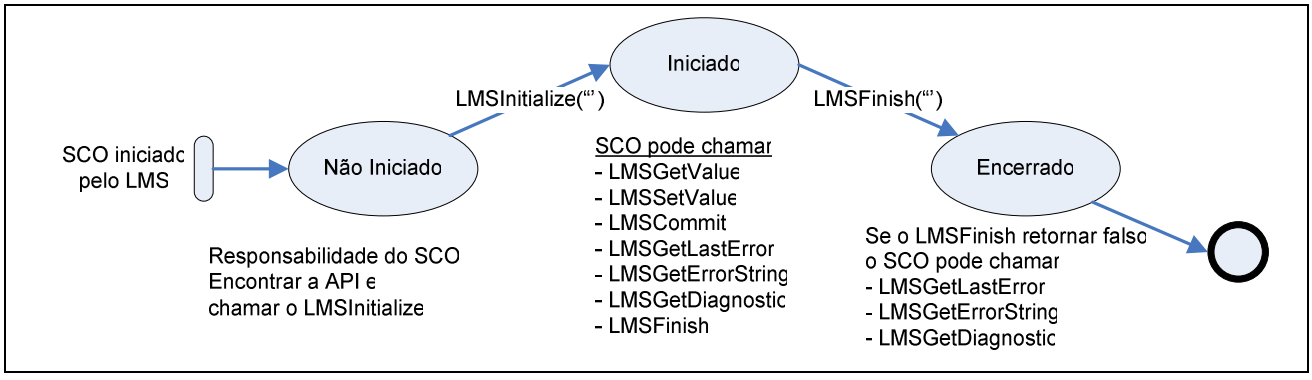

Figura 4. Estados de transições da API

Fonte: Adaptado de ADL (2006)

Ainda, algumas regras sobre a API devem ser observadas:

- Os nomes das funções são todos *case sensitive*, e precisam sempre ser expressos exatamente como são mostrados;
- Os parâmetros ou argumentos da função são *case sensitive*, ainda, todos os parâmetros são caixa baixa;
- Toda chamada para uma função da API, a exceção das funções de gerenciamento de estados, reinicia um código de erro.

Os recursos associados ao SCO (hipertexto, imagens, exercícios, etc.) são chamados de *Asset*. Eles podem ser disponibilizados individualmente ou agrupados e organizados através de uma estrutura de navegação, especificada por uma linguagem ECMA *Script*.

Nota-se que o SCORM fornece um modelo que descreve como implementar computacionalmente os objetos de aprendizagem e sua execução por um LMS. Entretanto, também existem modelos que sugerem como construir o conteúdo do objeto de aprendizagem sob o ponto de vista pedagógico. O projeto RIVED, além de prover um repositório para objetos de aprendizagem, descreve o processo de design instrucional, o qual será apresentado na seqüência.

2.1.3 RIVED

A RIVED (Rede Internacional Virtual de Educação) é um projeto que incentiva o desenvolvimento de objetos de aprendizagem que explorem as perspectivas pedagógicas e tecnológicas para complementar as atividades do ensino escolar, em especial. Para que isto ocorra, são incentivadas as práticas do design instrucional de atividades pedagógicas, da produção de material baseado na Web, do treinamento de professores, de uma rede de distribuição de objetos, e de um programa de avaliação.

O desenvolvimento de objetos segundo os padrões de qualidade da RIVED requer uma equipe multi-disciplinar, composta por especialistas em conteúdo, designer instrucional, pedagogo, programadores, ilustrador e designers de multimídia. É responsabilidade desse grupo o desenvolvimento e seqüência de atividades de ensino / aprendizagem , as quais são montadas em um formato de módulo, acompanhado de um manual com sugestões para a sua utilização em sala de aula.

A Figura 5 descreve cada uma das etapas de construção dos objetos de aprendizagem conforme as práticas orientadas pela RIVED, que vão desde a concepção do design instrucional até a confecção do objeto por especialistas em informática.

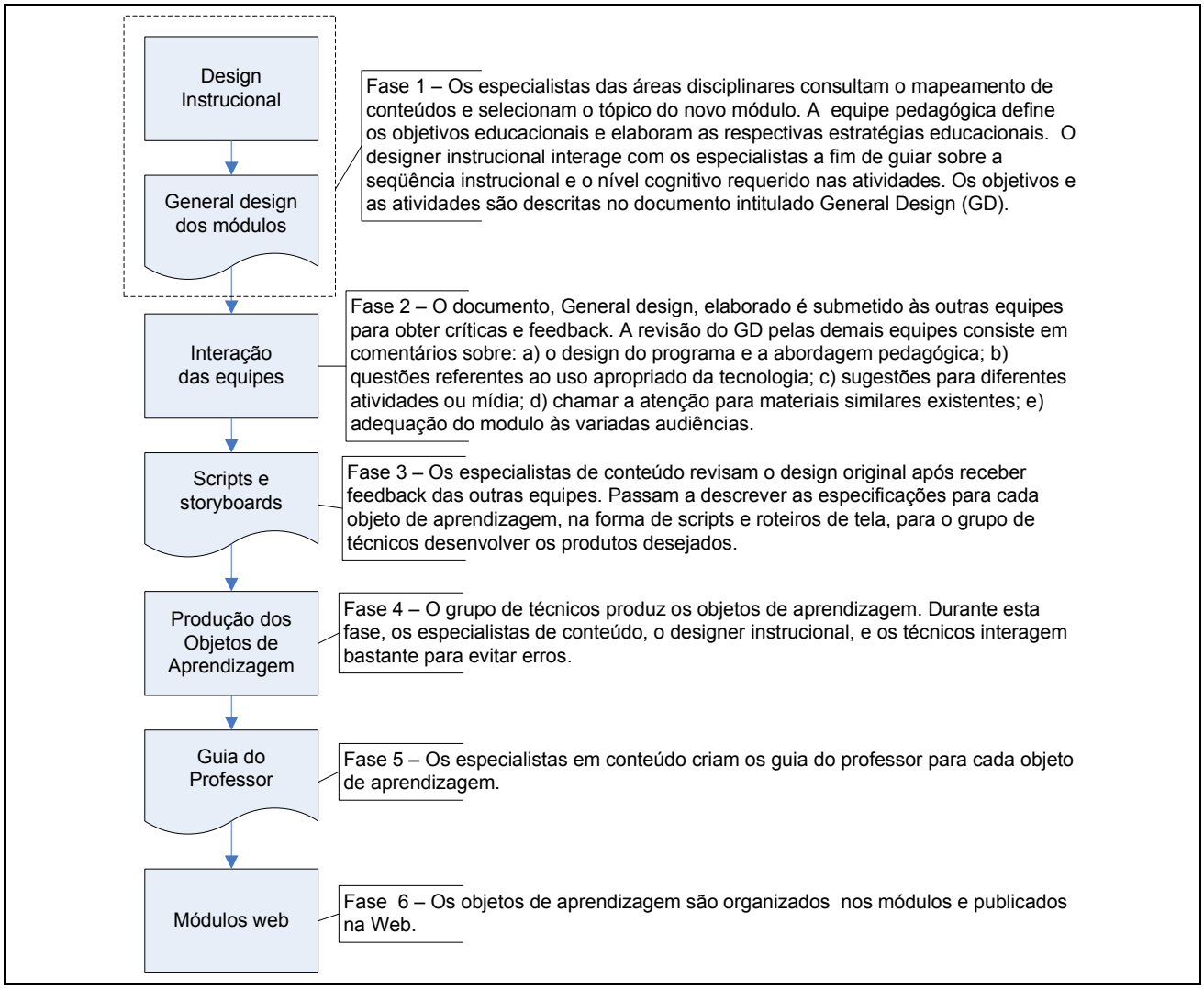

Figura 5. Processo de produção dos módulos e objetos de aprendizagem
Fonte: Adaptado de RIVED (2007)

Desde 2005, os objetos de aprendizagem desenvolvido seguindo o modelo proposto pela RIVED podem participar de um concurso anual promovido pela entidade e então, compor em seu repositório de objetos.

3 AGENTES

A área da Inteligência Artificial tem concentrado seus esforços na busca por desenvolver mecanismos baseados na experiência humana a fim de aplicá-los em sistemas computacionais. Possuir a capacidade de raciocinar, comunicar-se em linguagem natural e aprender devem ser adicionados aos softwares, de modo a permitir uma experiência mais rica e direcionada às necessidades individuais. Existem técnicas da Inteligência Artificial direcionadas a promover experiências semelhantes a ocorrências anteriores (Raciocínio Baseado em Casos ou Redes Neurais), através de um mapeamento das seleções realizadas consigam diagnosticar algo (Sistemas Especialistas) e, ainda, através do desenvolvimento de soluções a partir de um processo interativo de geração semi-aleatório de hipóteses (Evolucionista). Outra área visa desenvolver peças de software capazes de ter autonomia, colaborar e se comunicar, denominadas de agentes.

Um agente é um sistema computacional que está situado em alguns ambientes, e que é capaz de realizar ações autônomas neste ambiente de forma a atingir seus objetivos projetados (WOOLDRIDGE, 2002). Na Figura 6, o agente é ilustrado em seu ambiente. Este, por sua vez, captura informações através de sensores no ambiente, e produz uma ação de saída que irá afetá-lo. A interação geralmente ocorre de forma contínua.

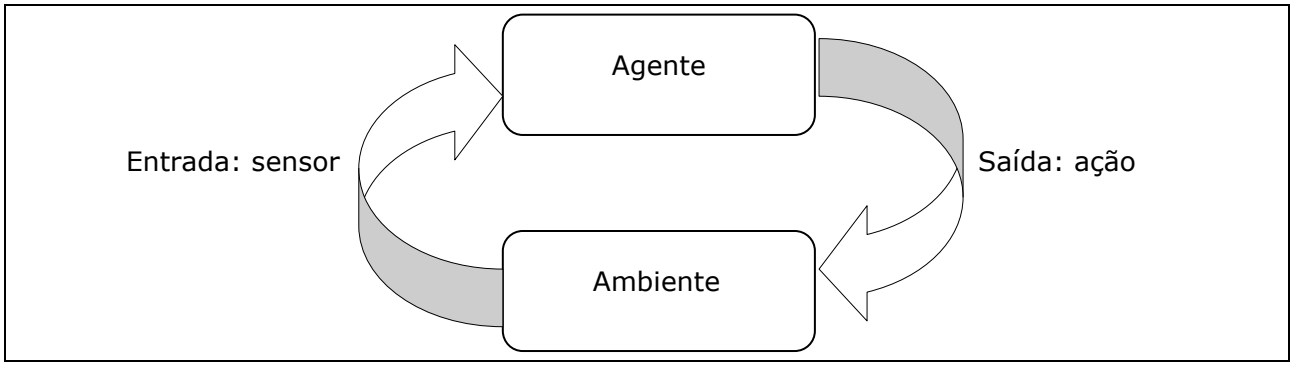

Figura 6. Conceito de agente
Fonte: Adaptado de Wooldridge (2002).

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

Na maioria dos casos, um agente não trabalha sozinho. Estabelece-se um ambiente onde a partir das características individuais, os agentes são capazes de interagir e atualizar o conhecimento mútuo e reconhecer as modificações do ambiente onde estão inseridos. Para que esta comunidade atinja seus objetivos, se estabelece a forma como irá interferir no ambiente para resolver um problema.

De forma prática, os agentes precisam notar e interpretar informações e mensagens recebidas, desenvolver raciocínio sobre suas crenças, realizar tomada de decisão e planejamentos que habilitarão novos procedimentos como, por exemplo, o envio de novas mensagens.

Wooldridge (2002) define que para um agente ser considerado inteligente, ele deve conter as seguintes habilidades de forma a satisfazer os objetivos projetados:

- Reatividade: os agentes devem perceber seu ambiente e responder, em tempo, as mudanças que ocorrerem;
- Pró-atividade: os agentes devem demonstrar comportamentos objetivos por iniciativa própria; e
- Habilidade social: os agentes devem ser capazes de interagir com outros agentes (incluindo humanos).

Sobre à arquitetura interna, os agentes podem ser classificados quanto à forma de resolução de seus problemas da seguinte maneira:

- Agentes reativos: suas ações são provocadas pelo resultado de uma outra ação já realizada. Seu comportamento baseia-se no modo estímulo-resposta, que inibe a recordação de atividades realizadas anteriormente e as que serão efetuadas no futuro; e
- Agentes cognitivos: a estes agentes estão envolvidas características como crenças, conhecimento, desejos, intenções e obrigações. Logo, agentes cognitivos diferenciam-se por possuir atitudes e estados mentais; tendo a capacidade de compreender o funcionamento do ambiente que o cerca, vivendo em sociedade com organização, cooperação e comunicação;
- Agentes híbridos: mesclam características dos agentes reativos e dos agentes cognitivos.

Demazeu e Müller (1989) descrevem a arquitetura interna de um agente através da posse de: (i) conhecimento sobre o mundo e sobre o problema que ele tem que resolver, adquirido através da comunicação com outros agentes ou através da percepção de mudanças no ambiente; e (ii) objetivos que devem ser executados por ele através de sua capacidade de raciocínio e de decisão (contidos nos algoritmos ou dinamicamente adquiridos através de comunicação com outros agentes, observação de mudanças no ambiente, raciocínio sobre comportamento dos outros agentes). Este modelo pode ser analisado na Figura 7.

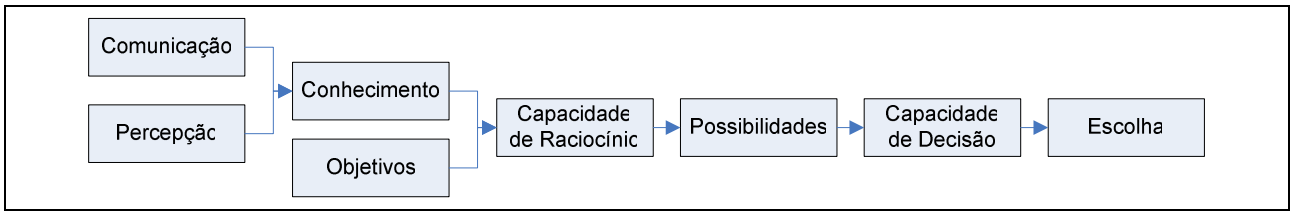

Figura 7. Modelo de agente
Fonte: Adaptado de Demazeu e Müller (1989).

3.1 SISTEMAS MULTI-AGENTES (SMA)

Quando a resolução de um objetivo necessita o esforço de dois ou mais agentes, denominamos de Sistema Multi-Agente. Embora os agentes possam habitar um ambiente em comum, deve-se garantir sua autonomia, ou seja, cada agente deve apresentar objetivo específico, de forma a manter sua característica. Através das habilidades de cooperação e coordenação, aliadas ao poder de comunicação bem definidos, o objetivo principal do sistema poderá ser atingido (JENNINGS, 1996).

Segundo Oliveira (1996), as sociedades de agentes podem ser classificadas quanto ao tipo de agentes imersos (homogêneos ou heterogêneos), quanto à migração de agentes (abertas ou fechadas) e quanto à presença de regras de comportamento (baseadas em leis ou não). Entretanto, para um funcionamento cooperativo numa sociedade, são necessários meios de comunicação. Estes podem ser diretos ou indiretos, e se diferem pelo fato dos agentes responsáveis pela troca de informações conhecerem-se ou não.

3.1.1 Comunicação entre Agentes

Para que haja um entendimento entre os agentes de um SMA, faz-se necessário a utilização de uma linguagem compreensível entre os agentes. Para tanto, é aplicada uma ACL (*Agent*

Communication Language), ou Linguagem de Comunicação entre Agentes, que é uma espécie de formalismo concebido a fim de codificar as mensagens trocadas entre agentes, independentemente da linguagem de programação utilizada pela aplicação. Uma ACL padronizada tem sua sintaxe e semântica, que devem ser reconhecidas e manipuladas pelos agentes. Usualmente uma ACL é definida publicamente de forma independente de implementação concreta, arquitetura e ambiente computacional onde está o Sistema Multi-Agente.

3.2 PADRÃO FIPA

A FIPA (*Foundation for Inteligent Physical Agents*) é uma fundação responsável pela criação de padrões para a implementação da comunicação entre agentes. Estabelece que o ciclo de vida de um padrão é dado pelas fases: preliminar, experimental, padrão, obsoleto e abandono. A versão existente no desenvolvimento deste trabalho, o FIPA-2000, é composta por 40 documentos distintos e sua organização pode ser observada na Figura 8 (FIPA, 2002).

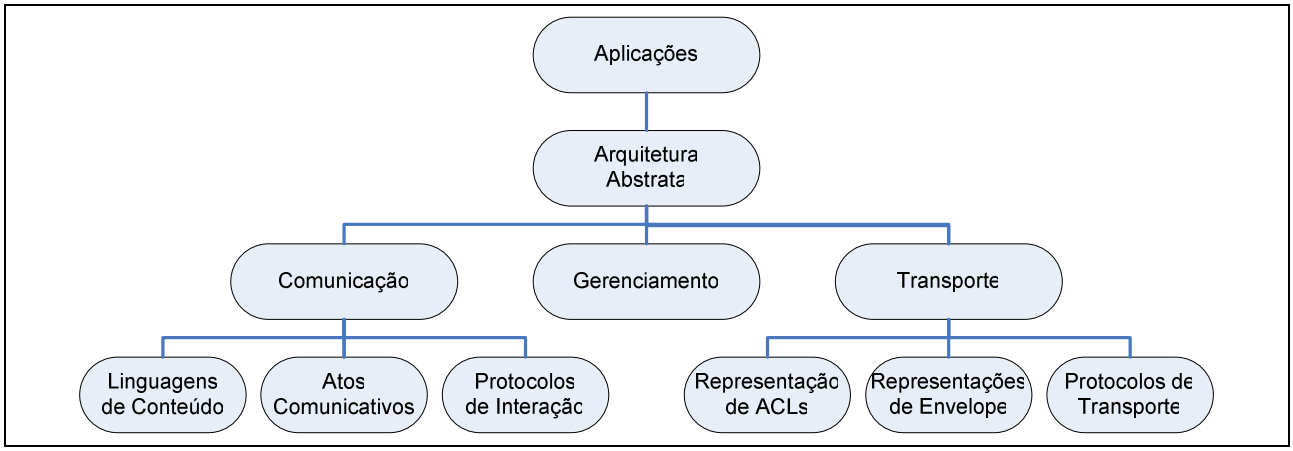

Figura 8. Organização geral do FIPA-2000
Fonte: Adaptado de FIPA (2002).

3.2.1 Linguagens de Conteúdo FIPA

O FIPA-2000 estabelece 4 linguagens padrão que representam o conteúdo das mensagens trocadas entre os agentes: SL (*Semantic Language*), KIF (*Knowledge Interchange Format*), CCL (*Constraint Choice Language*), e RDF (*Resource Description Framework*). Dentre estas se destaca a SL por representar estados mentais (crenças, desejos, incertezas e intenções) e ações (atos), através de uma lógica modal (FIPA, 2002).

A linguagem SL é dividida em subconjuntos que simplificam sua utilização quando uma aplicação não exige uma lógica de predicados modais expressivos:

- SL0: menor subconjunto de SL, usado como linguagem de conteúdo (ações, resultados de cálculos);
- SL1: extensão da SL0 incorporando conectivos lógicos (*and*, *or*, *not*); e
- SL2: permite a utilização de predicados de primeira ordem e operadores lógicos modais.

Conforme o documento *FIPA-ACL Message Structure Specification* - SC00061 (2002), cada mensagem FIPA possui uma série de parâmetros que o agente deve informar para que a mensagem seja enviada e recebida corretamente pelo destino. Para que isto ocorra, toda mensagem deve conter obrigatoriamente os campos *sender* (agente emissor), *receiver* (agente receptor) e o *performative* (ato performativo) preenchidos. Embora não impeçam o envio da mensagem, o preenchimento dos demais parâmetros é sugerido para que sua informação seja compreendida dentro do contexto em que os agentes conhecem. A citar, campos como: *ontology*, *protocol*, *conversation-id*, facilitam a identificação do conteúdo da mensagem transmitida, enquanto o campo *content* especifica o corpo da mensagem. A lista de parâmetros que compõe as mensagens do padrão FIPA é apresentada na Tabela 2.

Tabela 2. Parâmetros das mensagens FIPA-ACL

Classificação	**Parâmetro**	**Descrição**
Tipo de ato comunicativo	*Performative*	Denota o tipo de 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 identificado pelo o parâmetro *reply-to*.
Conteúdo de mensagem	*Content*	É o conteúdo ou conhecimento transportado pela mensagem.
Descrição do conteúdo	*Language*	É a linguagem na qual o conhecimento está expresso.
	Encode	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.

Fonte: Adaptado de Gluz e Viccari (2003).

3.2.2 Atos performativos FIPA

O parâmetro de ato performativo é responsável pela identificação do tipo de ação que está associado à mensagem. Os atos performativos reconhecidos pelo padrão FIPA-ACL, especificados no documento *FIPA-ACL Communicative Act Library Specification* – SC00037, são descritos abaixo:

- `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 executada 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`: Informa para um determinado agente que ele não necessita mais executar a ação que está informada no conteúdo da mensagem solicitada anteriormente.
- `cfp`: O ato cfp (*call for proposal*) solicita proposta para a execução de uma determinada ação. A mensagem deve conter qual a ação que deve ser realizada e qual a pré-condição para esta ação.
- `confirm`: O emissor informa ao receptor que uma dada proposição (informada no conteúdo) é verdadeira, se o receptor estava incerto disso.
- `disconfirm`: O emissor informa ao receptor que uma dada proposição (informada no conteúdo) é falsa, se o receptor estava certo de que ela era verdadeira.
- `failure`: O emissor informa ao receptor que tentou fazer uma ação e que essa tentativa falhou. O conteúdo 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 o emissor informar ao receptor se uma dada proposição (informada no conteúdo) é verdadeira ou não.
- `inform-ref`: É um ato composto que serve para o emissor informar ao receptor o objeto que corresponde a um dado descritor. O conteúdo da mensagem é uma expressão referencial, ou seja, 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 distribuição da mensagem encapsulada em anexo como se tivesse sido emitida diretamente por ele para os agentes que se encaixam na especificação também anexada, e reenvie a mensagem.
- `propose`: Serve para que o agente emissor envie ao receptor uma proposta para efetuar alguma ação, dada 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 encaixem 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 de 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 recusa da execução uma dada ação. Ao emitir uma mensagem deste tipo, também deve ser esclarecido o motivo da rejeição, utilizando o conteúdo da mensagem para fornecer esta explicação.
- `reject-proposal`: Representa a rejeição de uma proposta de execução de uma ação enviada previamente. O conteúdo é composto da ação rejeitada e da explicação do porquê da rejeição.
- `request`: O agente emissor solicita ao receptor que ele execute alguma ação (possivelmente outro ato comunicativo). O conteúdo da mensagem é composto da ação a ser feita e da proposição.
- `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 (descrição do valor a ser notificado).

A seqüência de atos performativos em um diálogo estabelecido consiste em um protocolo de interação. A FIPA disponibiliza, neste momento, especificação para onze protocolos de interação pré-definidos, os quais são apresentados na seção a seguir.

3.2.3 Protocolos de Interação FIPA

A especificação FIPA descreve um conjunto de protocolos de interação, isto é, padrões de interação de mensagens que podem ocorrer. Atualmente, são disponibilizados os protocolos apresentados na Tabela 3.

Tabela 3. Protocolos de interação FIPA

Identificador	Título
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

Fonte: Adaptado de FIPA (2002)

Na seqüência são detalhados como exemplo dois destes protocolos, os quais são utilizados nos diálogos entre os agentes do SMA proposto neste trabalho: FIPA *Request Interaction Protocol Specification* – SC00026 e FIPA *Contract Net Interaction Protocol Specification* – SC00029.

FIPA Request Interaction Protocol Specification – SC00026

O Protocolo de Requisição de Interação da FIPA (FIPA *Request Interaction Protocol*) permite a um agente requisitar, através do ato `request`, a outro agente a execução de alguma ação. O agente receptor processa a requisição e decide se aceita ou rejeita a solicitação, enviando uma mensagem de `accept` ou `refuse`, respectivamente.

No caso de aceite da solicitação, a informação processada pode ser devolvida através dos atos de: (i) `failure` – houve falha ao atender a requisição; (ii) `inform-done` – se ele conseguiu

completar a requisição com sucesso e precisa apenas informar ao emissor que a finalizou; (iii) `inform-result` – se ele conseguiu atender a requisição e qual foi o valor obtido como resultado.

A representação completa da troca de mensagens neste protocolo é ilustrada na Figura 9, detalhada na Seção 3.4.1.1.

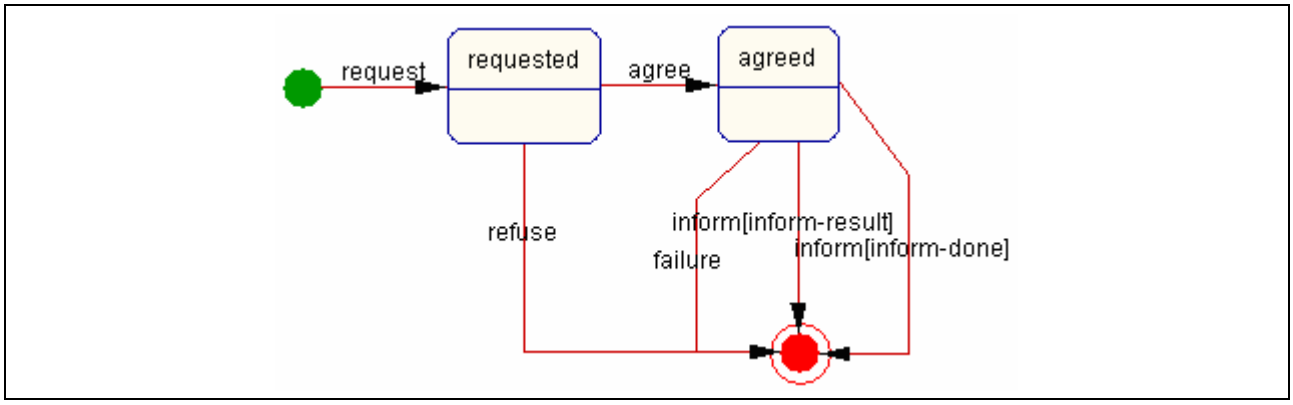

Figura 9. Protocolo *FIPA Request Interaction Protocol*
Fonte: FIPA Request Interaction Protocol (SC00026H)

FIPA Contract Net Interaction Protocol Specification – SC00029

O Protocolo de Interação de Rede de Contrato da FIPA (FIPA *Contract Net Interaction Protocol*) descreve que o agente emissor deseja obter uma informação ou que determinada tarefa seja executada e, para tanto, necessita conhecer quais agentes podem atender a esta solicitação. Para uma dada tarefa, qualquer participante pode responder a proposta, enviada através de uma mensagem ACL do tipo `cfp`; os demais precisam recusar (`refuse`). As negociações prosseguem com os participantes que enviaram propostas. A representação deste protocolo é apresentada na Figura 10.

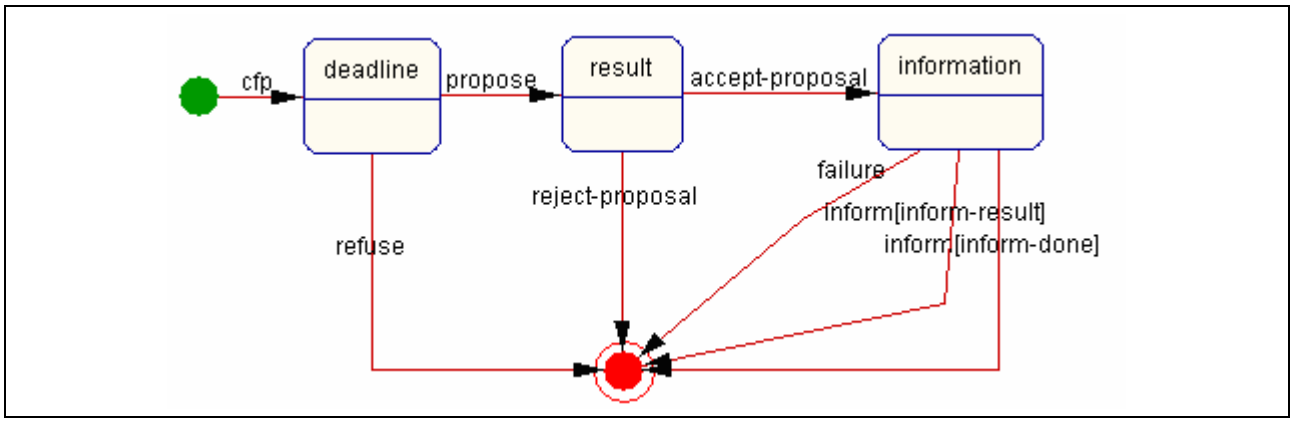

Figura 10. Protocolo *FIPA Contract Net Interaction Protocol*
Fonte: FIPA Contract Net Interaction Protocol (SC00029H)

O agente emissor solicita uma proposta, com as suas respectivas condições, através do ato performativo de "chamada de propostas" (`cfp`). Ao emitir a mensagem, é iniciada uma contagem de tempo, na qual delimita o prazo para envio das respostas dos outros agentes. Os receptores recebem a mensagem e, caso atendam os requisitos, emitem uma proposta (`propose`). Finalizado o *deadline*, o agente emissor avalia as propostas recebidas e seleciona o(s) agente(s) que irá(ão) executar a tarefa, informando-o(s) através de uma mensagem de proposta aceita (`accept-proposal`), os demais receberão uma mensagem de proposta rejeitada (`reject-proposal`). Os agentes responsáveis pela execução da tarefa, ao final, podem emitir uma das seguintes mensagens: (i) `failure` – houve falha ao atender a requisição; (ii) `inform-done` – se ele conseguiu completar a requisição com sucesso e precisa apenas informar ao emissor que a finalizou; (iii) `inform-result` – se ele conseguiu atender a requisição e qual foi o valor obtido como resultado.

3.3 AMBIENTES PARA DESENVOLVIMENTO DE SMA

A construção de Sistemas Multi-Agentes pode ser realizada em qualquer linguagem de programação e de forma livre por seus desenvolvedores. Entretanto, o uso de frameworks pode auxiliar nesta tarefa, permitindo aos projetistas se focarem mais no problema em que o SMA deve resolver. Detalhes internos como onde os agentes habitarão, atuarão e se comunicação, fica sob a responsabilidade do ambiente de execução.

Existem diversos ambientes que realizam esta camada intermediária de programação com o objetivo de facilitar, e até mesmo de utilizar padrões de comunicação, o desenvolvimento de

Sistemas Multi-Agentes. A seguir, é apresentada uma breve listagem contemplando alguns dos *frameworks* que permitem a construção de SMA:

- FIPA-OS: A plataforma FIPA-OS disponibiliza, além de um conjunto de classes, um ambiente de execução com todos os recursos necessários para a comunicação entre os agentes, ambos desenvolvidos na linguagem Java. O componente de comunicação utiliza o padrão FIPA-ACL. A construção de um agente é feita pela extensão da classe `FIPAOSAgent`, a qual provê os métodos que realizam a efetiva ligação do agente implementado ao ambiente propriamente dito. Embora alguns projetos tenham sido desenvolvidos sob esta plataforma, seu desenvolvimento foi descontinuado a partir de 2003 (FIPA-OS, 2007; GOMES, SILVEIRA & VICCARI, 2003).
- ZEUS: este ambiente é composto por uma biblioteca de componentes, ferramentas gráficas para construção de agentes e um conjunto de utilitários. Desta forma, permite sua integração com outras aplicações. As ações externas são desenvolvidas através de classes que estendem a classe `ZeusExternal`. Define-se que sua plataforma é composta por cinco camadas: Interface (contém os sensores e os efetores); Definição, Organização e Coordenação (camadas internas aos agentes) e Comunicação (permite a troca de mensagens entre os agentes). (NWANA , 1998; SILVA, 2004).
- Aglets: O ambiente Aglets, embora mantido pela empresa IBM, é disponibilizado gratuitamente. O foco destes agentes consiste na capacidade de se mover entre computadores, sem perder informações sobre seu funcionamento e estado interno. Seu desenvolvimento é realizado através da linguagem Java. (LANGE, 1997; SILVA, 2004).
- JATLite: este ambiente é composto por um pacote de programas desenvolvido em Java, com o intuito de permitir a criação de agentes de forma rápida. Ainda, provê uma infra-estrutura básica o qual os agentes devem se registrar, trocar mensagens e arquivos com outros agentes. (JEON, 2000).

Além dos *frameworks* descritos, encontra-se a plataforma JADE, a qual vem se destacando nos últimos anos. Isto se deve ao fato de um trabalho constante de seus pesquisadores em incorporar facilidades de modo a ser aplicados aos mais diversos contextos e sistemas. Para isto, conta com um grupo de pesquisadores ativos que freqüentemente lançam novas versões e exemplos de aplicações. Sendo assim, a seção a seguir apresenta os seguintes itens sobre o *framework* JADE: informações

gerais, as bibliotecas para desenvolvimento de agentes e suas ferramentas de apoio ao gerenciamento de Sistemas Multi-Agentes.

3.3.1 JADE

O JADE (*Java Agent Development Framework*) é um *framework* que apóia o desenvolvimento de Sistemas Multi-Agentes seguindo o padrão FIPA, realizando o papel de um *middleware*. Para tanto, disponibiliza um ambiente de execução, com apoio de ferramentas gráficas, onde os agentes podem ser criados e suas ações gerenciadas. Ainda, permite o desenvolvimento de agentes baseados em sua tecnologia através do uso de sua biblioteca de classes (JADE 2007).

Dentre as características de sistemas multi-agentes, o JADE fornece serviço de páginas amarelas para apoio no registro e busca por serviços oferecidos pelos agentes. Este sistema de gerenciamento de agentes provê o serviço de nomes (cada agente deve ter um nome exclusivo na plataforma em que habita). Ainda, permite criar e destruir agentes em contêineres remotos via requisição.

Quanto à troca de mensagens, o JADE utiliza uma linguagem de comunicação entre agentes com suporte ao padrão de protocolos de interação FIPA. Aliado a isto, pode-se utilizar ontologias como apoio a compreensão (e aceitação) das mensagens recebidas.

Cada instância do ambiente de execução JADE é chamada de contêiner (*Container*), o qual pode conter diversos agentes. Cada contêiner é um objeto RMI (*Remote Method Invocation*) que gerencia um conjunto de agentes localmente, controlando o ciclo de vida dos agentes (criação, suspensão, destruição).

O conjunto de contêineres ativos é vinculado em uma plataforma. Em especial, o contêiner *Main* precisa estar sempre ativo em uma plataforma e todos os contêineres devem ser registrados a este. Além disto, o contêiner *Main*, é responsável pelo sistema de gerenciamento de agentes e o serviço de páginas amarelas, através dos agentes MAS (*Agent Manager System*) e DF (*Directory Facilitator*), respectivamente. A Figura 11 ilustra a estrutura de contêineres e plataformas.

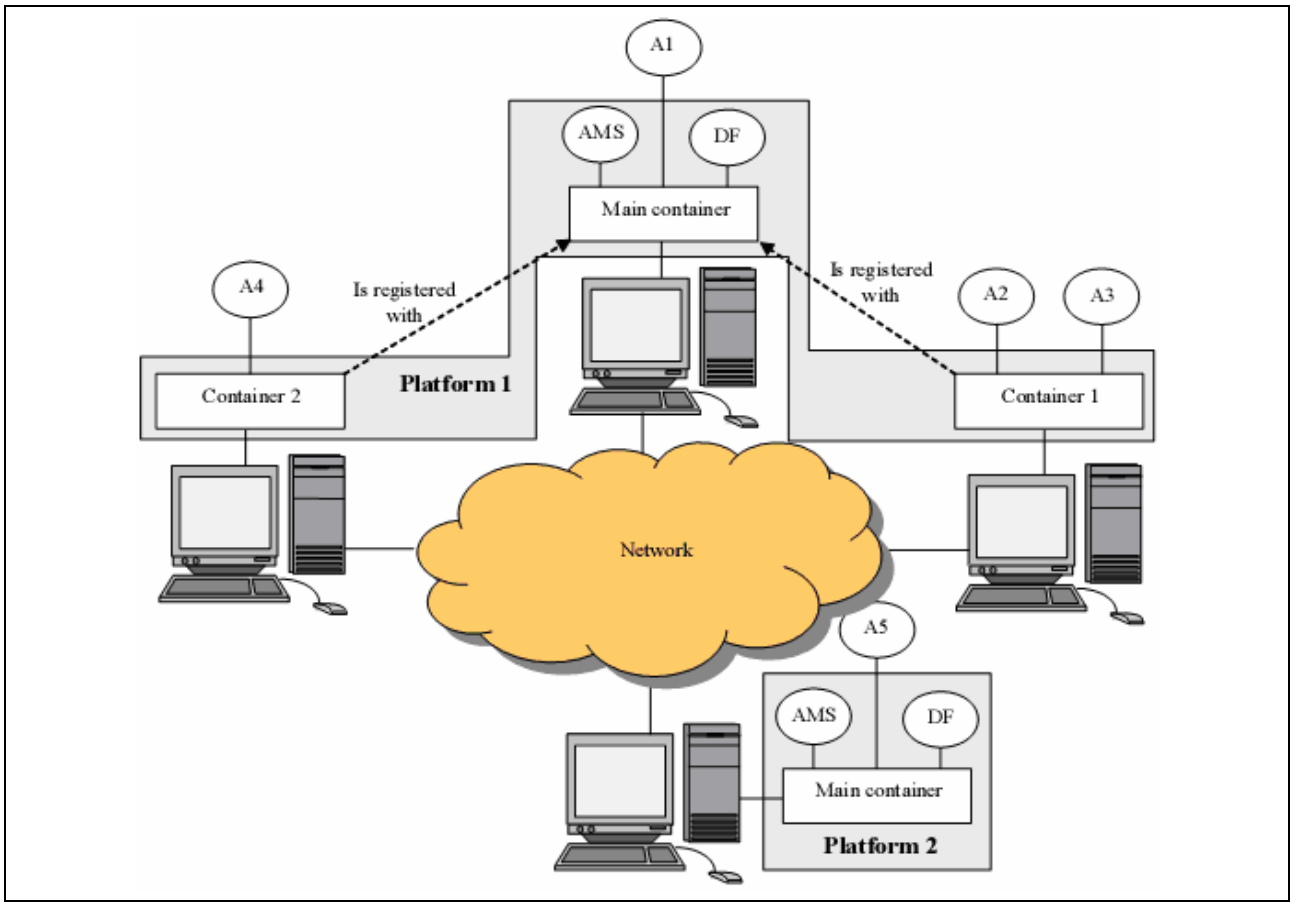

Figura 11. Contêineres e plataformas JADE
Fonte: Adaptado de JADE (2007).

Conforme a especificação FIPA, o ciclo de vida de um agente apresenta os seguintes estados, os quais são ilustrados na Figura 12:

- Iniciado (*Initied*): o agente é construído, porém ainda não foi registrado no AMS, tampouco apresenta nome ou endereço, impossibilitando-o de se comunicar com outros agentes;

- Ativo (*Active*): o agente está registrado no AMS, possui nome e endereço e, portanto, pode acessar todas as características do JADE;

- Suspenso (*Suspended*): as atividades do agente são suspensas, onde nenhum comportamento pode ser executado;

- Aguardando (*Waiting*): o agente é bloqueado, pois está aguardando alguma ação (geralmente o recebimento de alguma mensagem);

- Excluído (*Deleted*): o agente é finalizado e não possui mais registro com o AMS; e

- Em trânsito (*Transit*): um agente móvel entra neste estado quando está em processo de migração a outra localidade; o sistema continua a armazenar as mensagens que serão encaminhadas a sua nova localidade.

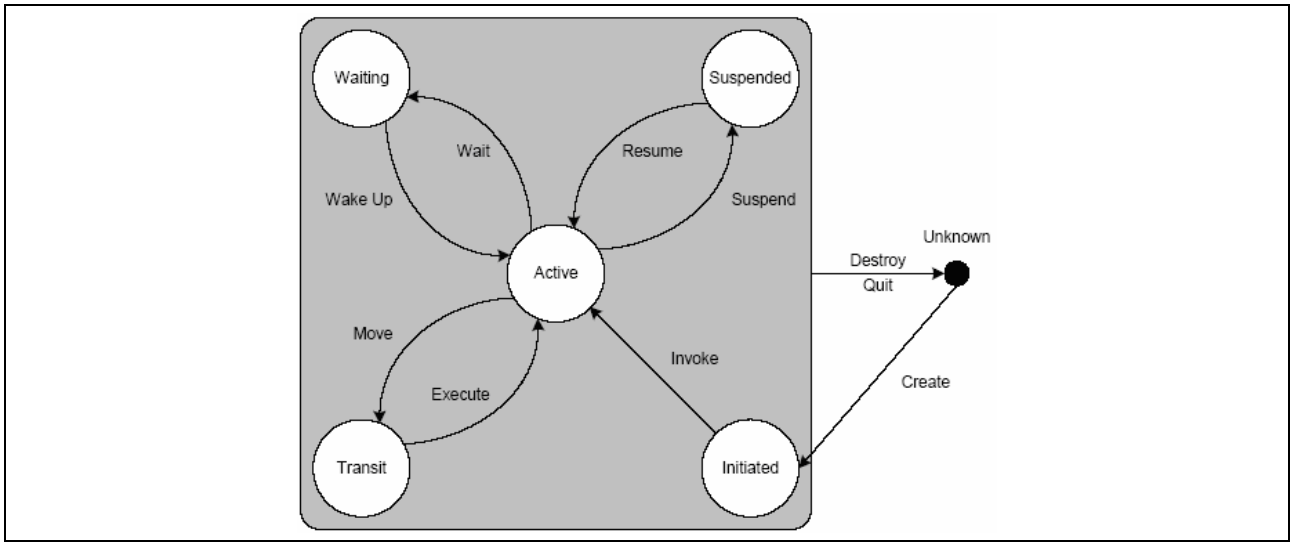

Figura 12. Ciclo de vida de agentes definido pela FIPA
Fonte: JADE (2007).

3.3.1.1 Biblioteca de Classes para Desenvolvimento de Agentes

A plataforma JADE permite que sejam desenvolvidos agentes baseados em sua tecnologia. Para isto, disponibiliza uma biblioteca com diversas funcionalidades, as quais permitem desde a criação de agentes simples até incluindo recursos remotos, troca de mensagens, e especificações de ontologias. Dentre os principais pacotes disponibilizados, constam:

- `jade.core`: implementa o *kernel* do sistema. Inclui a classe `Agent` que deve ser entendida para criar os agentes JADE; classe `Behaviour` implementa as tarefas, ou intenções de um agente.

- `jade.lang.acl`: realiza o processo de comunicação (ACL), de acordo com as especificações FIPA;

- `jade.content`: contém um conjunto de classes para suporte de ontologias;

- `jade.domain`: contém as classes que representam o gerenciamento de agentes definidos pelo FIPA, em particular os agentes AMS e DF, que provêm o ciclo de vida, e serviço de páginas brancas e amarelas.

- `jade.gui`: contém um conjunto de classes genéricas úteis para criar interfaces gráficas para exibir e editar os identificadores e descrições dos agentes, mensagens ACL, etc.;

- `jade.mtp`: contém uma interface que todo protocolo de transporte de mensagem deve implementar;

- `jade.proto`: contém as classes para modelar protocolos de interação padrão (*fipa-request, fipa-query, fipa-contract-net, fipa-subscribe*); e

- `jade.wrapper`: provê um invólucro de alto-nível do JADE.

3.3.1.2 Ferramentas de Apoio ao Gerenciamento de Agentes

O JADE disponibiliza um conjunto de ferramentas de suporte ao gerenciamento de sistemas multi-agentes, as quais simplificam a administração da plataforma e aplicação `jade.tools`. Cada ferramenta conta com um agente para apoio na atividade, a citar:

- Agente de Gerenciamento Remoto (RMA – *Remote Management Agent*) é uma interface gráfica para controle e gerenciamento de plataforma. Uma primeira instância de um RMA pode ser iniciado com um comando `-gui`. O JADE mantém coerência entre múltiplos RMAs. O console RMA permite iniciar outras ferramentas JADE;

- O Agente *Dummy* é uma ferramenta gráfica de monitoramento e *debugging*. Usando a interface gráfica é possível criar mensagens ACL (*Agent Communication Language*) e enviar a outros agentes; ainda é possível mostrar uma lista de mensagens ACL enviadas e recebidas;

- O *Sniffer* é um agente que pode interceptar mensagens ACL enquanto elas estão sendo transmitidas, e as mostram em uma interface gráfica usando uma notação

similar ao diagrama de seqüências UML. É útil para testar sociedades de agentes, observando como eles trocam mensagens;

- O *Introspector* é um agente que permite monitorar o ciclo de vida de outros agentes, tal como a troca de mensagens e comportamentos em execução;
- A interface de DF é usada pelo *Directory Facilitator* padrão do JADE, e pode também ser usado por qualquer outro DF necessário;
- O *LogManagerAgent* é um agente que permite registrar informação de log em execução;
- O *SocketProxyAgent* é um agente simples, que atua em modo bidirecional entre a plataforma JADE e a conexão TCP/IP.

3.3.1.3 Comunicação entre Agentes usando mensagens ACL

Uma das mais importantes características que os agentes JADE provêem é a habilidade de se comunicar. O paradigma de comunicação adotado é a troca de mensagens assíncronas. Cada agente tem uma caixa de correio (a fila de mensagens do agente) onde o JADE recebe as mensagens enviadas por outros agentes. Onde quer que uma mensagem seja enviada na fila de mensagens, o agente receptor é notificado.

Enviar uma mensagem para outro agente exige apenas o preenchimento dos campos do objeto `ACLMessage`, e então executar o método `send()`. Um agente pode capturar mensagem de sua fila de mensagens através do método `receive()`. Este método retorna a primeira mensagem na fila (removendo-a) ou NULL se a fila estiver vazia. Outra possibilidade é pré-selecionar as mensagens que serão analisadas pelo agente. Para isto, utiliza-se a classe `MessageTemplate`, onde um conjunto de critérios podem ser definidos.

3.4 MODELAGEM E ESPECIFICAÇÃO DE SMA

A presente seção discute algumas metodologias de Engenharia de Software aplicada a Sistemas Multi-Agentes, onde uma delas é melhor detalhada. Em seguida, apresenta-se o conceito de ontologias e sua aplicação perante o framework JADE.

3.4.1 Engenharia de Software para SMAs

A Engenharia de Software orientada a agentes concentra-se no uso de agentes para o desenvolvimento de sistemas distribuídos complexos em ambientes dinâmicos. Dentre os benefícios da aplicação de agentes, encontram-se a sua capacidade de alto nível de abstração das entidades ativas no sistema (SILVA *et al.*, 2003; TVEIT, 2001).

Dentre as metodologias utilizadas para modelagem de SMAs, destacam-se:

- AUML (*Agent Unified Modeling Language*): consiste em uma extensão da UML, expandindo os diagramas existentes de forma a representarem o comportamento dos sistemas multi-agentes (ODELL *et al.*, 2001)
- Gaia: permite modelar tanto a estrutura dos agentes quanto a sociedade e organização. Caracteriza-se por possibilitar a representação do SMA baseado nos requisitos do sistema (WOOLDRIDGE, 2002);
- AALAADIN: esta abordagem apresenta uma organização que define agrupamentos a partir de sua estrutura. Cada grupo descreve um conjunto de papéis que representam as funções que os agentes disponibilizam, e agentes, ambos necessários para seu pleno funcionamento. A metodologia não fornece qualquer tipo de impedimento quanto à estrutura interna de seus agentes (FERBER & GUTKNECH, 1998);
- MaSE (*Multi-agent Software Engineering*): possibilita descrever a análise e modelagem do SMA, a partir da especificação dos objetivos a serem alcançados. Em cada etapa da metodologia é obtido um artefato que pode ser relacionado aos demais, permitindo atrelar cada um dos agentes aos objetivos eles devem implementar.

Devido à potencialidade da metodologia MaSE, além de uma especificação bem definida, oferece uma ferramenta para construção de seus artefatos, esta foi selecionada para a especificação do SMA apresentado neste trabalho. Desta forma, uma descrição mais detalhada sobre seu funcionamento é apresentada na seção seguinte.

3.4.1.1 Multi-agent Software Engineering (MaSE)

A metodologia MaSE (*Multi-agent Software Engineering*) permite o desenvolvimento de sistemas multi-agentes com base nos princípios da Engenharia de Software (DeLOACH & WOOD, 2001). Conforme é ilustrada na Figura 13, o processo de desenvolvimento é dividido em duas fases principais: a análise e o projeto; onde cada um provê um conjunto de etapas a serem seguidas.

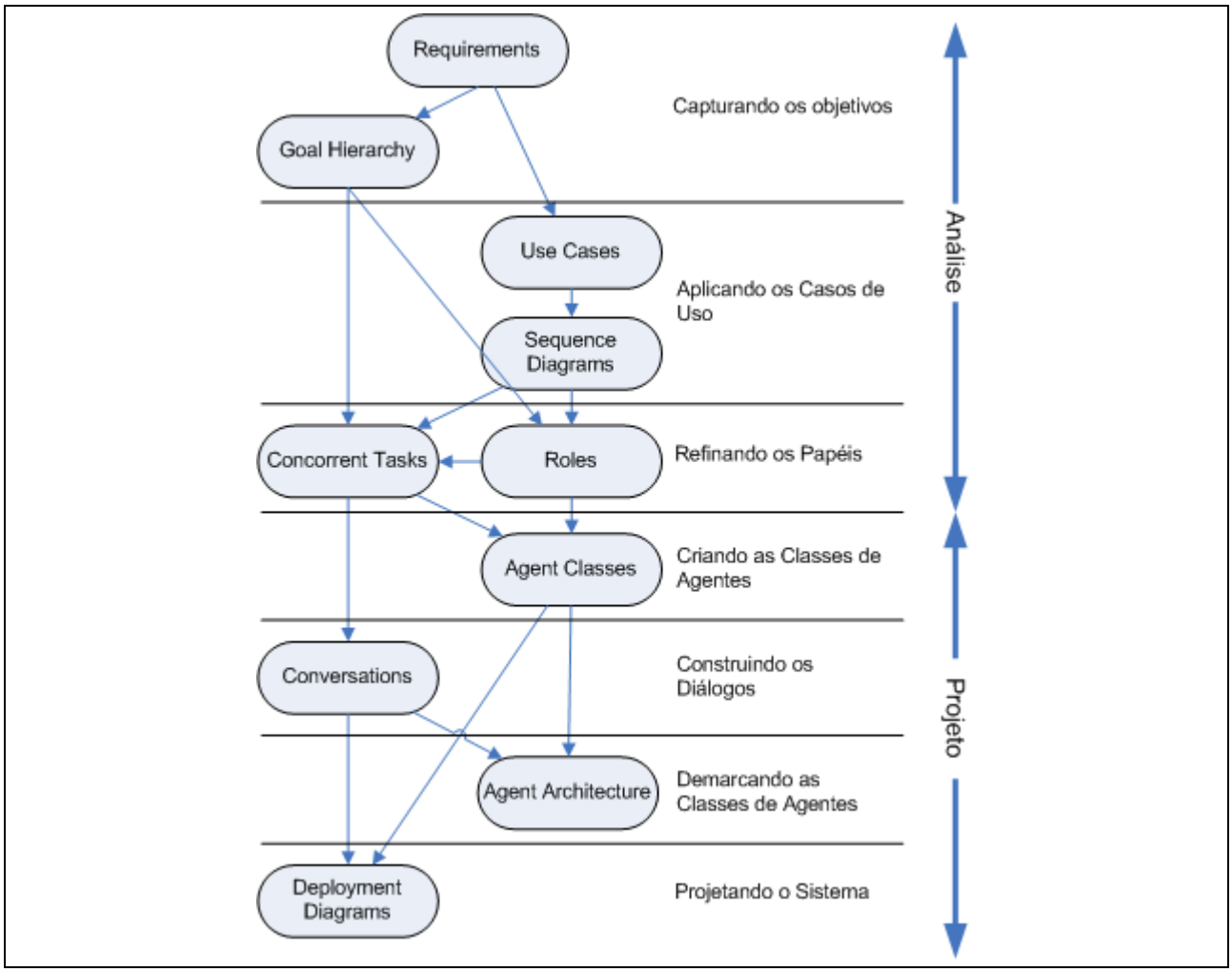

Figura 13. Etapas de modelagem da metodologia MaSE
Fonte: Adaptado de DeLoach e Wood (2001).

A fase de análise é composta pelas seguintes etapas:

- Capturarando os objetivos: Visa conhecer as especificações do sistema e transformá-las em uma estrutura de conjunto de objetivos. Conforme a metodologia, é necessário conhecer qual é o objetivo principal do SMA, para então expandir em sub-objetivos os quais permitirão atingir o primeiro.
- Aplicando os Casos de Uso: Traduz os objetivos em papéis e tarefas. Descreve a seqüência de eventos de forma visual e narrativa.
- Refinando os Papéis: Permite definir através de um diagrama os papéis (representados em retângulos), tarefas (representadas pelas elipses) e protocolos de comunicação

(representados pelas setas) que o SMA deverá conter. No diagrama, todos os objetivos estabelecidos inicialmente devem ser vinculados a um papel.

Já a fase de projeto contempla as etapas:

- Criando as Classes de Agentes: As classes de agentes são identificadas a partir dos papéis, representados por caixas, e os diálogos, em linhas, conectando as classes.
- Construindo os Diálogos: O diagrama de diálogo entre agentes é formado por um par de estados finitos que define a comunicação entre duas classes de agentes exclusivamente. O inicializador começa o diálogo enviando a primeira mensagem. O final do diálogo é representado pelo estado finito terminal.
- Demarcando as Classes de Agentes: Etapa onde o estado interno dos agentes é especificado. Este processo é simplificado devido o uso de uma linguagem de modelagem de arquitetura, que combina as linguagens tradicionais e linguagem natural.
- Projetando o Sistema: Define a configuração do sistema atual a ser implementada.

A metodologia MaSE foi desenvolvida para ser aplicada de forma interativa, onde é possível navegar entre as etapas em diversos momentos, acarretando em um modelo completo e consistente (DeLOACH & WOOD, 2001). Os modelos são transformados em diagramas que permitem descrever os agentes, seus diálogos e sua estrutura interna, nos mais diversos níveis de detalhamento. Ainda, o desenvolvedor pode selecionar quais diagramas irá utilizar e em que ordem eles serão descritos. Ao final, obtém-se uma consistência entre os modelos do MaSE de forma a satisfazer os objetivos iniciais do sistema.

Além de desenvolver a metodologia MaSE, DeLoach e Wood (2001) apresentam o agentTool, um sistema computacional que implementa os diagramas propostos. Os diagramas deste trabalho foram construídos através do agentTool.

3.4.2 Especificação de Ontologias

Confome Breitman (2005) as ontologias são especificações formais e explícitas de conceitualizações compartilhadas. Ainda, consistem em modelos conceituais que capturam e explicitam o vocabulário utilizado nas aplicações semânticas.

Elas devem possuir um significado e abrangência muito mais profundas do que as simples hierarquias de conceitos e palavras-chaves empregadas.

Apesar da palavra “ontologia” denotar uma teoria sobre a natureza do ser ou existir, em Inteligência Artificial ela pode ser interpretada como o conjunto de entidades com suas relações, restrições, axiomas e vocabulário. Uma ontologia define um domínio, ou mais formalmente, especifica uma conceitualização acerca dele. Normalmente, uma ontologia é organizada em hierarquia de conceitos (ou taxonomias).

Sob o ponto de vista de Sistemas Multi-Agentes, as ontologias se apóiam na definição e organização do conhecimento, fornecendo descrições não ambíguas das características e propriedades dos agentes e do ambiente. Sendo assim, as ontologias descrevem os conceitos e os relacionamentos que podem existir para uma comunidade de agentes, promovendo sua comunicação. De forma prática, elas especificam o domínio, isto é, o vocabulário usado em um diálogo entre agentes. Ainda, ao explicitar uma ontologia a organização do Sistema Multi-Agente torna-se mais clara. Uma ontologia é projetada com o propósito de permitir o compartilhamento e reuso do conhecimento. Ao se submeter a uma ontologia, o agente concorda sobre um determinado conceito especificado por ela (ZINI & STERLING, 1999).

O documento XC00086D - FIPA *Ontology Service Specification* estabelece que a comunicação entre agentes que utilizam sua especificação, compartilham uma ontologia comum, pois isto assegura que os agentes direcionam um mesmo significado para os termos utilizados nas mensagens. Logo, ele fornece um serviço de ontologia, o qual contempla a manutenção de ontologias, tradução de expressões entre ontologias ou linguagens distintas, responder a uma solicitação entre termos e ontologias, e facilitar a identificação de ontologias compartilhadas (FIPA, 2002).

Para o desenvolvimentos de ontologias, recomenda-se a utilização de editores gráficos devido à facilitação, tanto na criação quanto na manutenção e promoção do reuso. Um dos editores mais utilizados pela comunidade científica é o *Protégé*, o qual é detalhado na seqüência.

Gasevic *et al.* (2005) afirmam que as ontologias podem ser usadas para descrever conteúdos de objetos de aprendizagem, e desta maneira, promover os objetos de aprendizagem a uma nova dimensão de reusabilidade. A ontologia provê uma infra-estrutura para a recuperação dos objetos de

aprendizagem. Os autores defendem ainda que a ontologia possa ser utilizada tanto para a formalização dos metadados quanto do conteúdo instrucional do objeto.

3.4.2.1 Editor Protégé

O Protégé é um ambiente para desenvolvimento de sistemas baseados em conhecimento, descrito através de ontologias. No editor podem ser geradas ontologias compatíveis com tecnologias como CLIPS, JESS, PROLOG, XML, RDF, OIL; e em especial, ele também permite a geração do código-fonte de ontologia seguindo as especificações para o *framework* JADE através do componente Bean Generator (GENNARI *et al.*, 2003; FREITAS, 2003).

A Figura 14 ilustra o componente *Bean Generator* integrado no editor Protégé. A distribuição deste componente também inclui exemplos de projetos destinados ao *Bean Generator*, especificando conceitos e ações. Para a geração do código-fonte é necessário apenas informar o nome do pacote onde as classes serão criadas, o diretório onde ficará o pacote, e o nome da ontologia; por fim, confirma-se a geração dos arquivos.

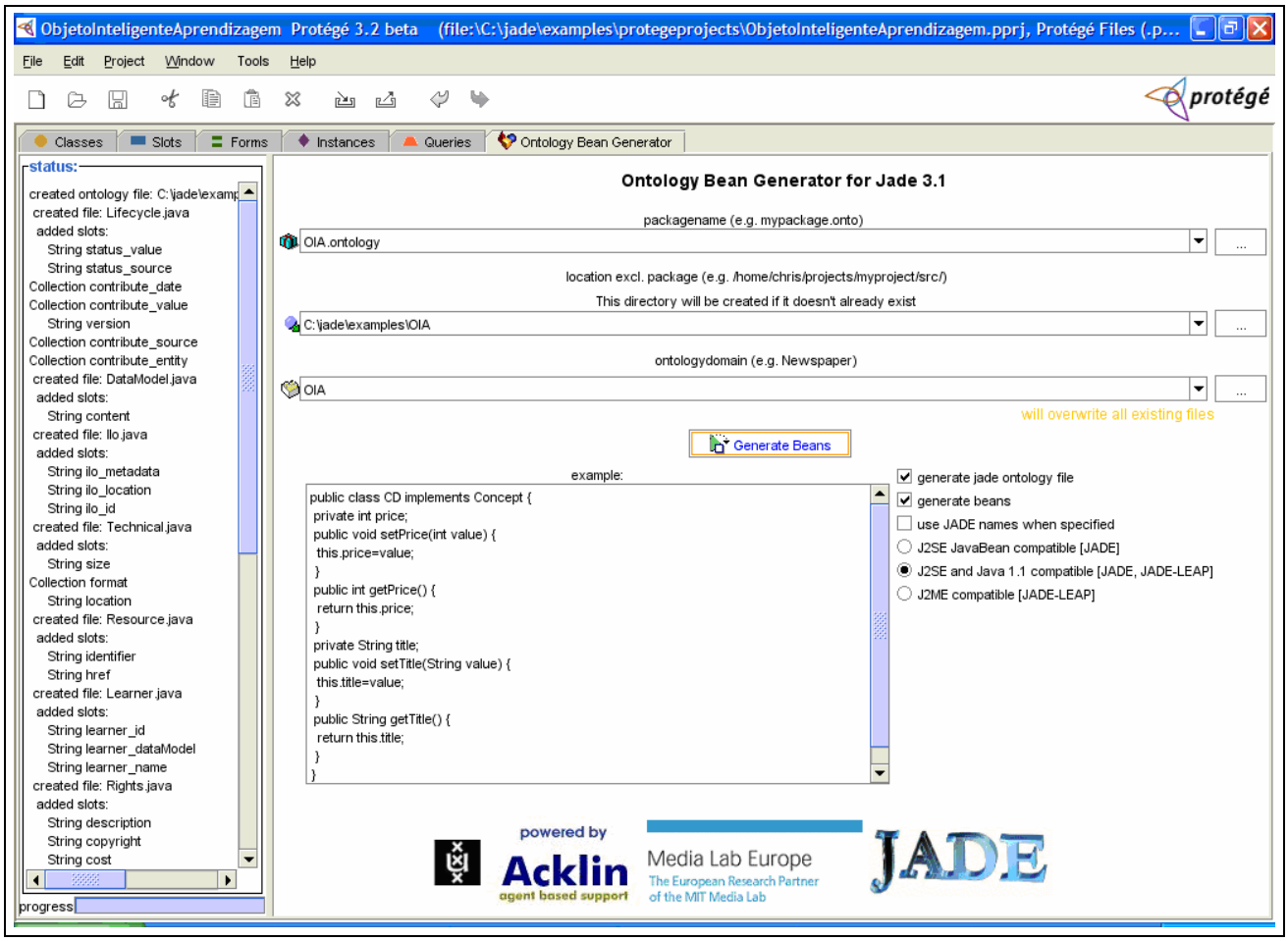

Figura 14. Componente Bean Generator do editor Protégé

4 OBJETOS INTELIGENTES DE APRENDIZAGEM

Trabalhos desenvolvidos pelo grupo de pesquisa anteriormente, apresentam o conceito de Objetos Inteligentes de Aprendizagem. De forma geral, um Objeto Inteligente de Aprendizagem agrega as características dos objetos de aprendizagem aliadas a tecnologia dos sistemas multi-agentes (GOMES, 2004; SILVEIRA *et al.*, 2005; 2006; 2007).

Outra definição aceita é que um objeto inteligente de aprendizagem consiste em um agente capaz de desempenhar o papel de um objeto de aprendizagem (SILVEIRA *et al.*, 2005). Wilges *et al*. (2007) considera que um objeto inteligente de aprendizagem é um objeto de aprendizagem que possui como base um agente que possa gerar experiências de aprendizagem, sem deixar de ser reutilizável.

A utilização de SMAs ampliam as possibilidades aos ambientes construídos com objetos de aprendizagem tradicionais, por estas apresentarem características de agentes, tais como (BRADSHAW, 1997):

- Um agente é um pedaço de software que trabalha de forma contínua e autônoma, dentro de um ambiente;
- Um agente pode inferir em um ambiente de forma flexível e inteligente, sem necessitar de intervenção humana; e,
- Um agente pode se comunicar com outros agentes através de troca de mensagens, usando Linguagens de Comunicação entre Agentes (*Agent Communication Language* – ACL).

As características acima citadas podem ser aplicadas ao contexto dos Objetos Inteligentes de Aprendizagem, da seguinte forma: um ILO é um pedaço de software, que funciona a partir de informações recebidas por outros agentes devido a sua autonomia e capacidade de inferência, sem necessitar de intervenção externa.

Outra possibilidade interessante refere-se à potencial capacidade de aprendizagem, na qual os agentes possuem. Logo, um objeto de aprendizagem com esta característica 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, tais como: acionar outros objetos de

aprendizagem que possam auxiliar o aluno e complementar o ensino, adquirir informações sobre os alunos, como as suas preferências e estilos cognitivos, a fim de se adaptar a estes e, inclusive, aprender como se adaptar a elas; adaptar seu conteúdo educacional no sentido de se adaptar ao estudante; entre outras. A Figura 15 ilustra esta idéia.

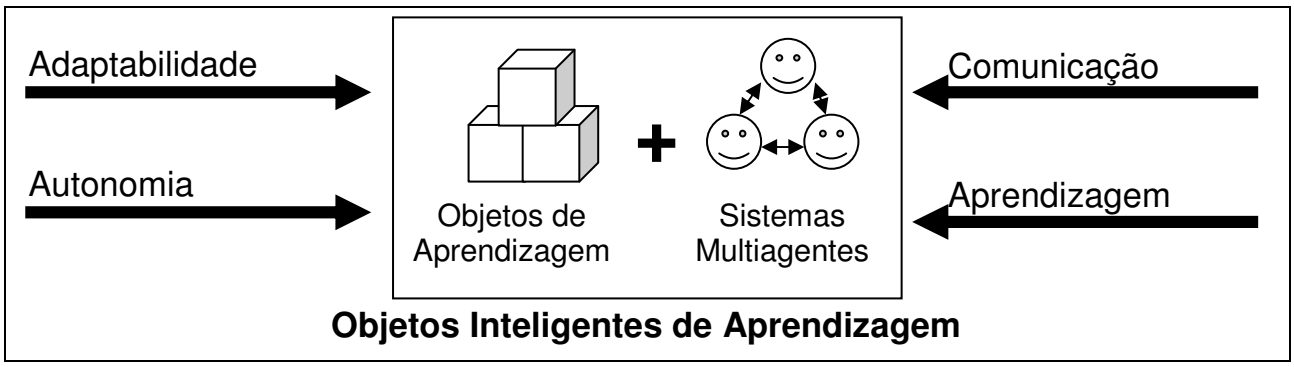

Figura 15. Objetos inteligentes de aprendizagem

4.1 ARQUITETURA PROPOSTA

Para o desenvolvimento dos objetos de aprendizagem sob a perspectiva de agentes, foi necessária a modelagem de uma sociedade de agentes, que contemplasse as entidades fundamentais no processo de ensino- aprendizagem: conteúdo – ambiente – aluno. No trabalho de Gomes (2005) foi apresentada uma arquitetura com agentes que representam os papéis de LMS, do repositório de objetos e dos próprios objetos.

A fim de compreender a arquitetura, optou-se pela utilização de uma metodologia de Engenharia de Software aplicada a sistemas multi-agentes. A aplicação de técnicas e artefatos permitem que se conheça e descreva cada etapa de um projeto.

Em Silva e Silveira (2006) foram apresentados os resultados iniciais quanto à aplicação da metodologia MaSE a fim de relatar a sociedade de agentes proposta por Gomes (2005). Constatou-se que seria necessária a revisão dos agentes e seus papéis perante a sociedade, tal como uma redefinição dos protocolos de comunicação dos diálogos.

Desta forma, além de aprimorar mais na definição dos papéis dos agentes, optou-se pela inclusão de um agente que representasse o aluno, conforme descrição a seguir:

- Agente LMS: representa o papel do sistema de gerenciamento de aprendizagem (LMS), lidando com os aspectos administrativos e gerenciais que envolvem os ambientes de aprendizagem; além de prover o acesso dos aprendizes aos ILOs;

- Agente ILO: encapsula o objeto de aprendizagem, sendo capaz de gerar experiências de aprendizagem aos alunos. Ainda, na necessidade de complemento da aprendizagem redireciona o aluno a outro objeto;

- Agente ILOR (*Intelligent Learning Object Repository*): é uma abstração do papel de repositório de objetos de aprendizagem, direcionando as informações sobre os ILOs; e,

- Agente Learner: contempla a representação do aluno perante o ambiente, além do resultado das interações com os objetos de aprendizagem. Também é responsável pelas operações de armazenamento e resgate de informações sobre o modelo de dados do aluno.

A Figura 16 ilusta a arquitetura de agentes proposta. Os alunos interagem com LMS, a partir do agente LMS que criará um agente Learner a fim de estabeler uma representação do aluno dentro do SMA. O agente LMS busca o ILO mais apropriado para a necessidade de aprendizagem do aprendiz e o invoca. O ILO é então responsável por gerar as experiências de aprendizagem ao aprendiz. Nesta tarefa, é possível comunicar-se com o agente LMS e com outros agentes ILO a fim de promover uma experiência de aprendizagem mais rica. Toda comunicação é promovida pela troca de mensagens no padrão FIPA-ACL. O ambiente onde tais agentes habitam estão no padrão FIPA, o qual provê todo o mecanismo necessário para troca de mensagens entre os agentes.

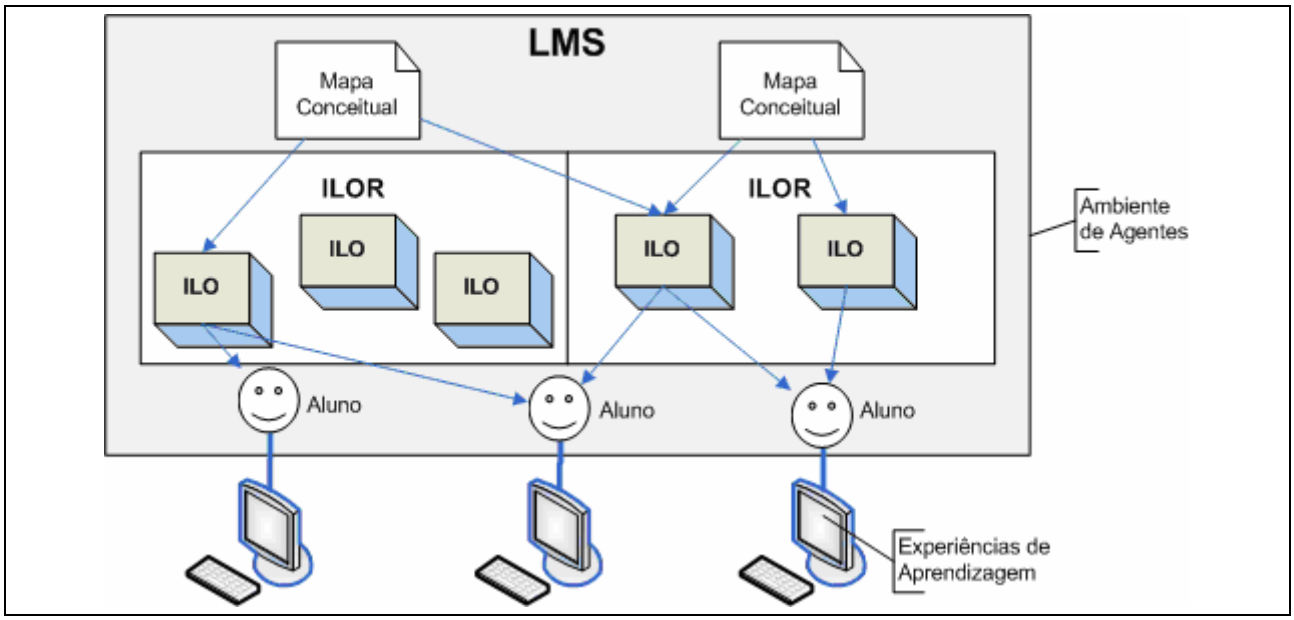

Figura 16. Arquitetura SMA para objetos inteligentes de aprendizagem

Além dos agentes citados serão consideradas as funcionalidades provenientes dos agentes presentes no *framework* JADE, tais como: o serviço de páginas amarelas (agente DF) e o sistema de gerenciamento de agentes (agente SMA). A inclusão desses agentes auxilia no processo de criação e remoção dos agentes durante a interação no SMA, além de conhecer as características de cada um dos agentes.

4.2 MODELAGEM DO FRAMEWORK

A modelagem apresentada neste trabalho consiste no resultado de uma verificação do modelo proposto para o trabalho apresentado por Silveira *et al.* (2005), a utilizando a metodologia MaSE. Conforme sugerido pelo próprio método, não é necessária a aplicação de todas as etapas (da Figura 13) e seus artefatos, buscando se concentrar apenas naqueles que forem adequados à situação.

Nas subseções seguintes, apresentam-se os resultados das etapas para a definição do modelo, contendo: a definição dos objetivos do SMA, o desenvolvimento dos casos de uso e respectivos diagramas de seqüência, especificação dos papéis e como devem ser desenvolvidos os agentes, e por fim os diálogos que os agentes devem estabelecer no SMA.

4.2.1 Capturando os Objetivos

A identificação dos objetivos do SMA visa conhecer qual é o objetivo principal do SMA, para então expandir em sub-objetivos os quais permitirão atingir o primeiro. No caso dos Objetos

Inteligentes de Aprendizagem, compreende-se que seu objetivo fundamental é proporcionar as experiências de aprendizagem. Para que isto seja possível, é necessário que se tenha controle sobre o ambiente, o objeto de aprendizagem e o aluno. O objetivo geral e os sub-objetivos identificados para este contexto são apresentados na Figura 17.

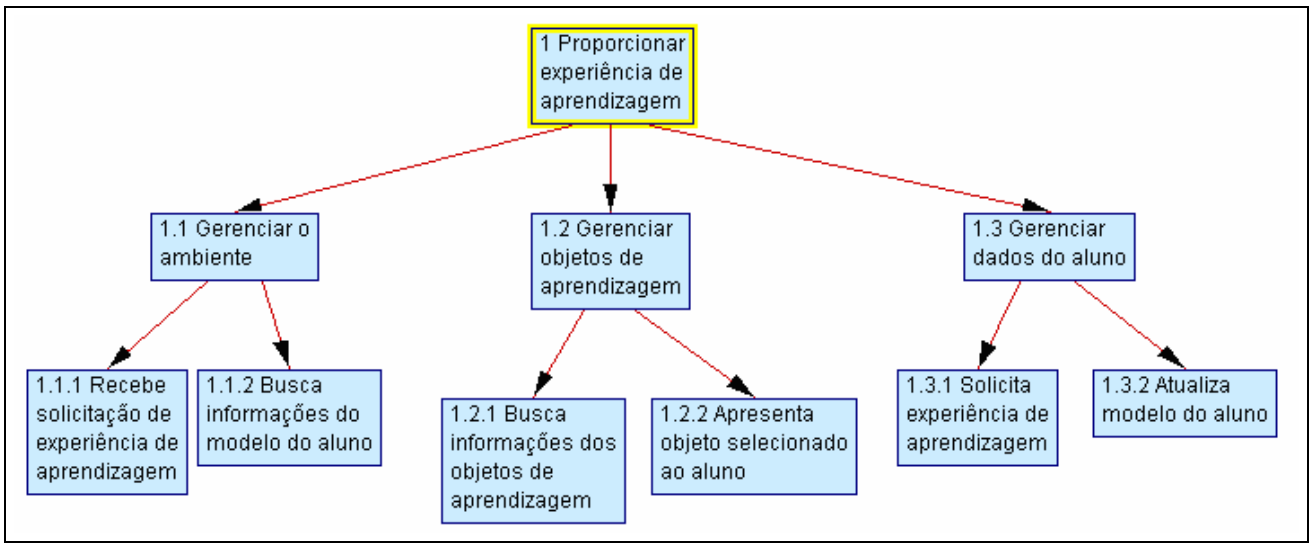

Figura 17. Diagrama de hierarquia de objetivos

Para que o objetivo geral do SMA seja realizado é necessário que ocorra o gerenciamento do ambiente (LMS) de forma a atender as solicitações de experiências de aprendizagem dos alunos. Ainda, por poder se adaptar às características individuais do aluno, o ambiente deve fornecer informações (de identificação e cognitiva) sobre o modelo do aluno.

Sobre as ações dos objetos de aprendizagem, devem ser contempladas a capacidade de localizar informações sobre outros objetos, de modo a tornar a seleção do conteúdo instrucional individualizada. Já da parte do aluno, o SMA deve possuir uma forma de solicitar novas experiências de aprendizagem, explicitamente ou não. Por fim, o objeto de aprendizagem deve ser capaz de atualizar o modelo do aluno.

4.2.2 Casos de Uso e Diagramas de Seqüência

Os casos de uso e diagramas de seqüência são apresentados como a segunda etapa na metodologia MaSE (Figura 13). A partir dessas definições, inicia-se uma compreensão e detalhamento das funcionalidades a serem contempladas pelo SMA. Cada caso de uso pode reunir uma ou mais ações que, opcionalmente, podem ser descritos pelos diagramas de seqüência.

Estas representações, embora sua construção seja sugerida no início da descrição do SMA, foram contempladas apenas durante o desenvolvimento do *framewok*. Isto ocorreu, pois apenas quando os papéis dos agentes e os diálogos foram estabelecidos, tal como iniciada sua implementação, obteve-se o conhecimento necessário para seu desenvolvimento.

Os casos de uso e diagramas de seqüência identificados e descritos a seguir são: a Preparação do Ambiente, Inicialização da Experiência de Aprendizagem, Finalização da Experiência de Aprendizagem, e Busca por Outros Objetos de Aprendizagem.

Preparação do Ambiente

O primeiro diagrama descreve o processo de Preparação do Ambiente realizada pelo agente que gerencia os processos do LMS. Ele é o responsável pela criação dos agentes de repositório e objetos de aprendizagem. Cada agente realizará seu próprio registro de páginas amarelas. A Figura 18 ilustra a interação entre os agentes.

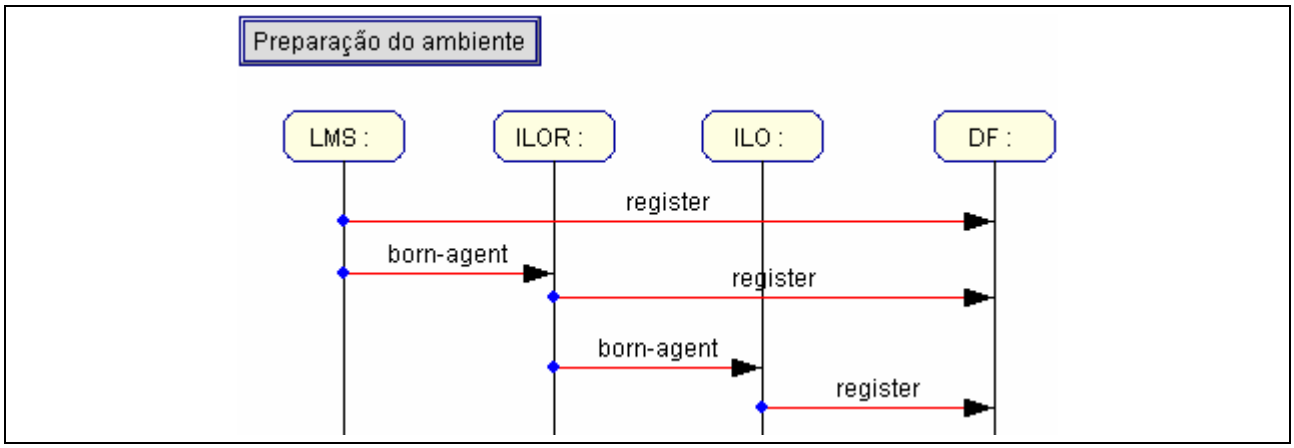

Figura 18. Diagrama de seqüência da Preparação do Ambiente

Cabe ressaltar que a mensagem `df-agent-registry` foi incluído no diagrama a fim de representar o processo de registro do agente no serviço de diretórios, embora essa tarefa seja realiza internamente pela plataforma de agentes.

Solicitação de Experiência de Aprendizagem

O processo de Solicitação de Experiência de Aprendizagem ocorre quando o aluno pede a ação de um determinado objeto de aprendizagem. A princípio, todos os objetos contidos nos repositórios já estão ativos e registrados no DF. Isto ocorre pois caso um objeto de aprendizagem solicite a complementação a outro objeto, o último já deve ser reconhecido pelo SMA

independentemente de seu uso por outro aluno em um mesmo momento. Logo, a solicitação de experiência de aprendizagem cria uma representação, do acesso do aluno ao objeto, dentro do SMA.

Conforme representado Figura 19, o diálogo `new-experience` é responsável pela criação da representação do aluno através do agente LMS. Nele, a interface de apresentação do objeto de aprendizagem solicita a criação de uma representação do aluno, criando um agente Learner, o qual deve se regisrar no serviço de páginas amarelas (DF).

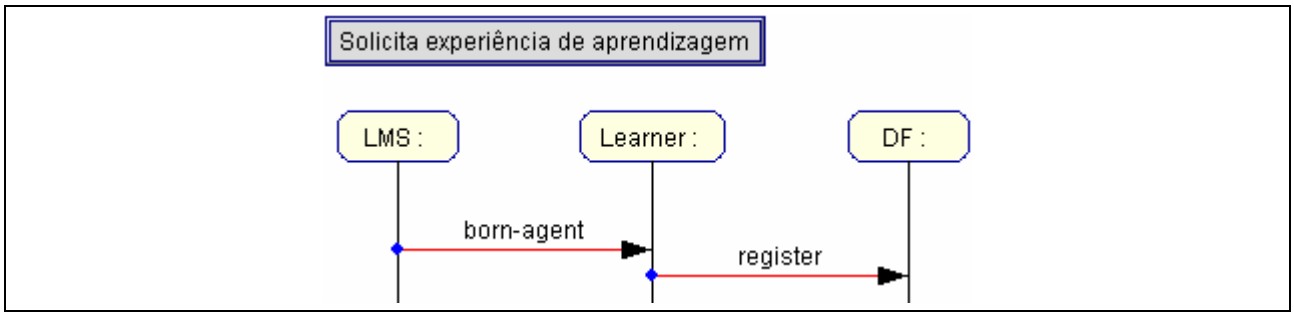

Figura 19. Diagrama de seqüência da Solicitação de Experiência de Aprendizagem

Finalização de Experiência de Aprendizagem

O diagrama de seqüência sobre a finalização de experiência de aprendizagem representa o ato do aluno sair do objeto de aprendizagem. Quando isso ocorre, a API informa ao SMA que deve encerrar a representação do aluno. A Figura 20 ilustra esse processo.

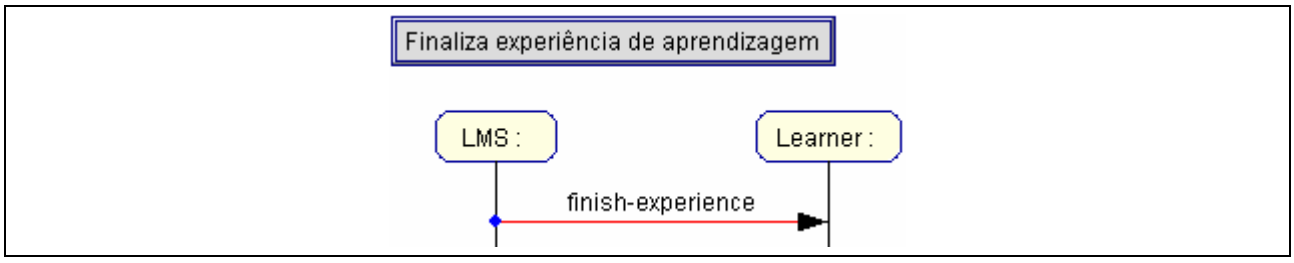

Figura 20. Diagrama de seqüência da Finalização de Experiência de Aprendizagem

Busca por Outro Objeto de Aprendizagem

O processo de busca por objetos que possam complementar ou dar continuidade a aprendizagem do aluno é descrita pelo diagrama de seqüência Busca por Objeto de Aprendizagem, ilustrado na Figura 21. Primeiramente, deve-se localizar quais agentes que representam os objetos de aprendizagem estão ativos no SMA. A partir desta localização, é enviada uma mensagem a eles para verificar se atendem as necessidades do objeto corrente.

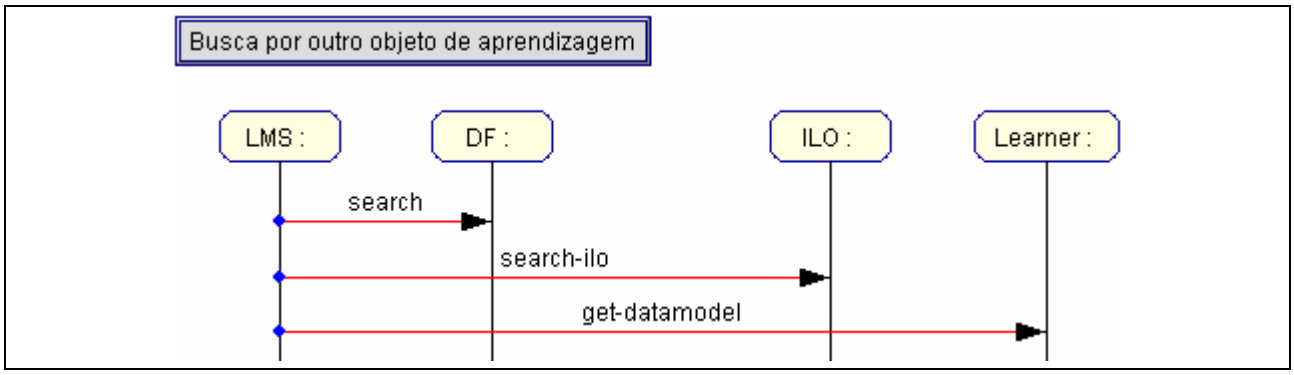

Figura 21. Diagrama de seqüência de Busca por Objeto de Aprendizagem

4.2.3 Estabelecendo os Papéis

O diagrama de papéis ilustra como cada objetivo pode ser alcançado através dos papéis estabelecidos, que podem ser implementados em um ou mais agentes. Um papel consiste em uma ação que um agente é capaz de executar, tal como fornecer ou processar uma informação. Na Figura 22, os papéis são representados em retângulos, as tarefas são representadas pelas elipses e os protocolos de comunicação são representados pelas setas.

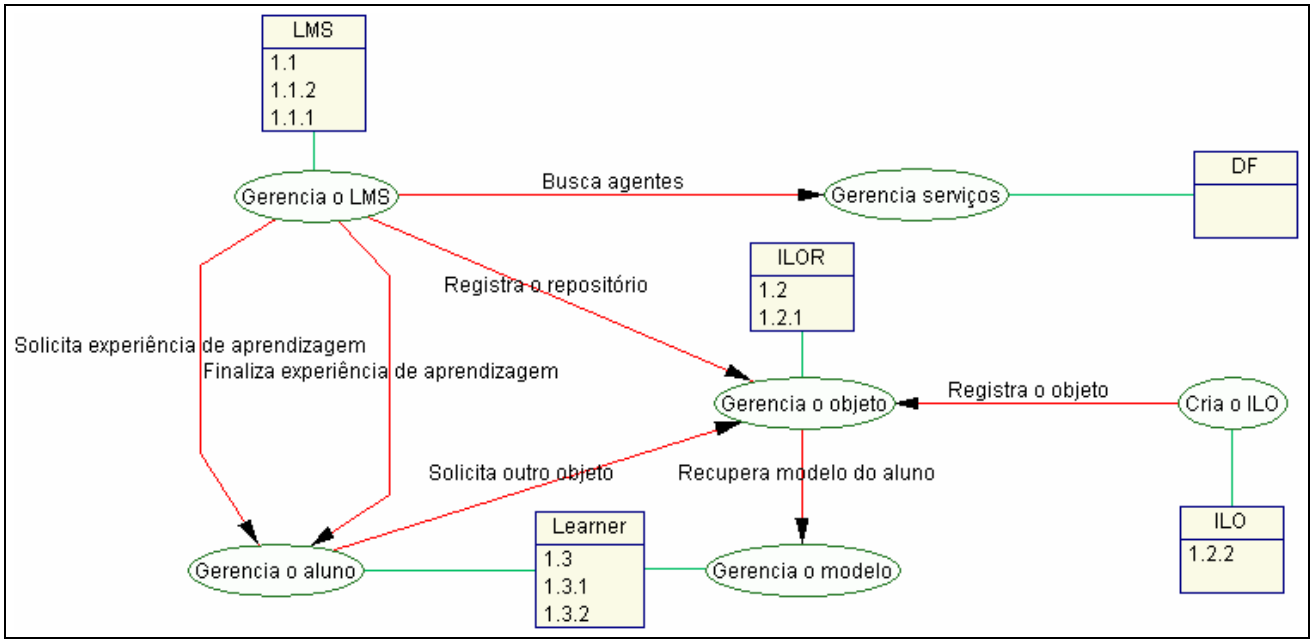

Figura 22. Diagrama de papéis

Cada um dos agentes informa qual objetivo estará contemplando, através dos números indicados. Por exemplo, o agente ILO implementa o objetivo 1.2.2, que conforme o diagrama de objetivos Apresenta objeto selecionado ao aluno. Durante o processo de modelagem do sistema, foram identificados quatro papéis necessários para a contemplação dos objetivos.:

- Aluno: solicitar nova experiência de aprendizagem e gerenciar o modelo do aluno;
- LMS: gerenciar o ambiente, receber solicitação de experiências de aprendizagem, buscar informações sobre o aluno;
- ILOR: gerenciar os objetos de aprendizagem (em seu repositório) e buscar um objeto de aprendizagem; e,
- ILO: conhecer os metadados e apresentar o conteúdo do objeto selecionado ao aluno.

Entretanto, a partir da implementação, notou-se a necessidade da inclusão de uma entidade: o DF - responsável pelo serviço de páginas amarelas da plataforma. Desta forma, o Diagrama de Papéis foi modificado a fim de contemplá-lo.

Tais papéis possibilitam que as seguintes tarefas sejam desenvolvidas através da utilização de um protocolo específico, conforme descrito na Tabela 4.

Tabela 4. Classificação das funções da API do SCORM

Papel	**Tarefa**	**Descrição**
Aluno	Gerencia o aluno	Realiza as solicitações de registro e cancelamento de alunos acessando um ILO
LMS	Gerencia o LMS	Processa as solicitações da API, convertendo-as em mensagens ACL
ILOR	Gerencia os objetos	solicita que um objeto seja registrado, e auxilia na seleção de objetos complementares a aprendizagem
ILO	Cria os objetos	apresenta-se ao repositório como um novo objeto
DF	Gerencia serviços	Provê informações sobre os serviços prestados pelos agentes que habitam o SMA

4.2.4 Desenvolvendo as Classes dos Agentes

As classes de agentes são identificadas a partir dos papéis (caixas), e os diálogos (linhas), que conectam as classes. O diagrama das classes de agentes para Objetos Inteligentes de Aprendizagem é mostrado na Figura 23.

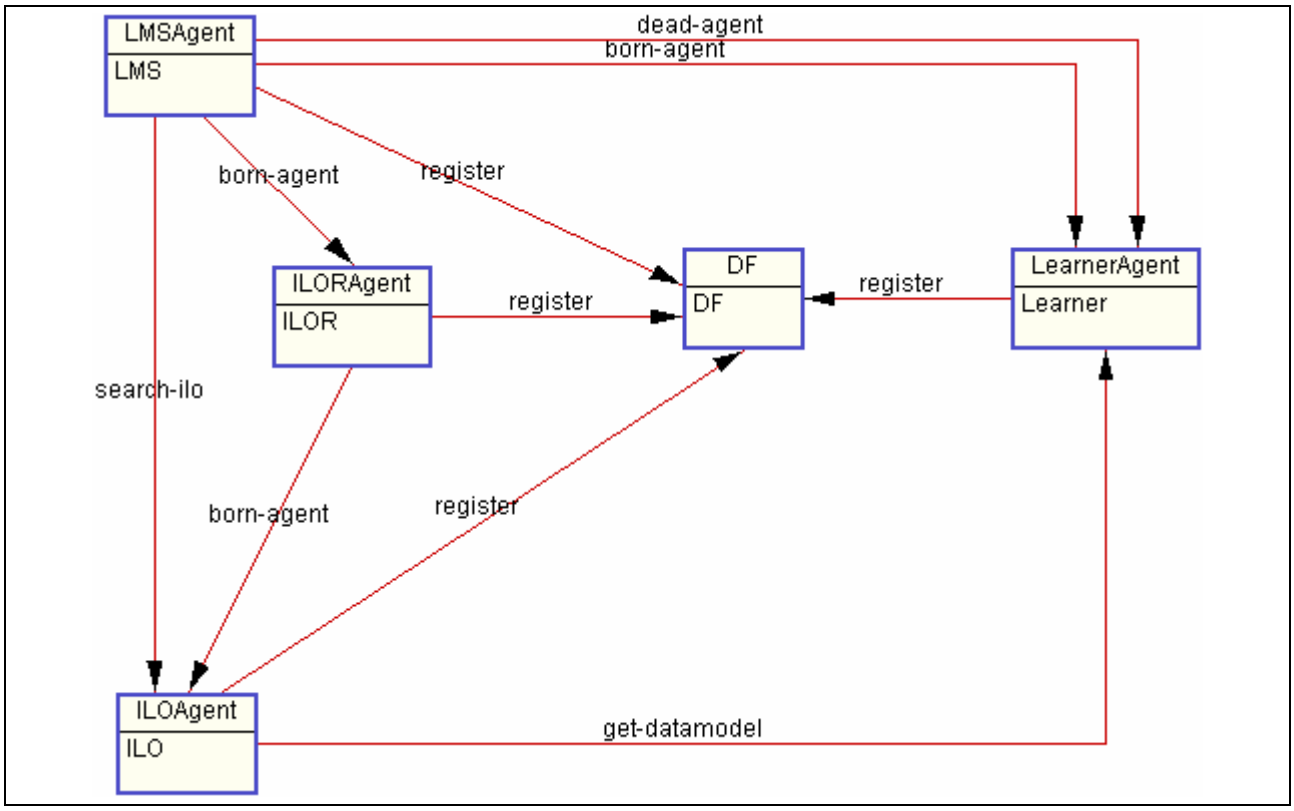

Figura 23. Diagrama de classes de agentes

Foram estabelecidas cinco classes de agentes os quais representam os papéis de agente LMS, agente ILOR, agente ILO e agente Learner no que diz respeito a retornar uma determinada informação solicitada; e o agente DF que é fornecido pela própria plataforma para gerenciamento de páginas amarelas.

4.2.5 Diálogos entre os Agentes

O diagrama de diálogo entre agentes é formado por um par de estados finitos que define a comunicação entre duas classes de agentes exclusivamente. O inicializador começa o diálogo enviando a primeira mensagem. O final do diálogo é representado pelo estado finito terminal.

Conforme o diagrama de classes de agentes, foram identificados os diálogos referentes a ontologia **ilo-ontology** (detalhada na seção 4.3 Ontologia Desenvolvida), FIPA-Agent-Management e JADE-Introspection. Ainda, os diálogos descritos pela ilo-ontology seguem os protocolos da especificação FIPA, já apresentados na seção 3.2.3 Protocolos de Interação FIPA.

4.3 ONTOLOGIA DESENVOLVIDA

De acordo com o modelo proposto pela FIPA, uma ontologia é definida pelo uso de predicados, ações e conceitos (FIPA, 2002). Logo, os conceitos do LMS, ILO, ILOR e do aluno são ilustrados na Figura 24. Nota-se que há uma interligação semântica entre os conceitos: o modelo de dados do aluno se refere a interligação do desempenho obtido pelo aluno em um determinado objeto. Já o ILOR implementa uma lista de objetos que ele possui. Enquanto o mapa conceitual define quais conceitos, presentes no objeto, devem ser abordados em um curso.

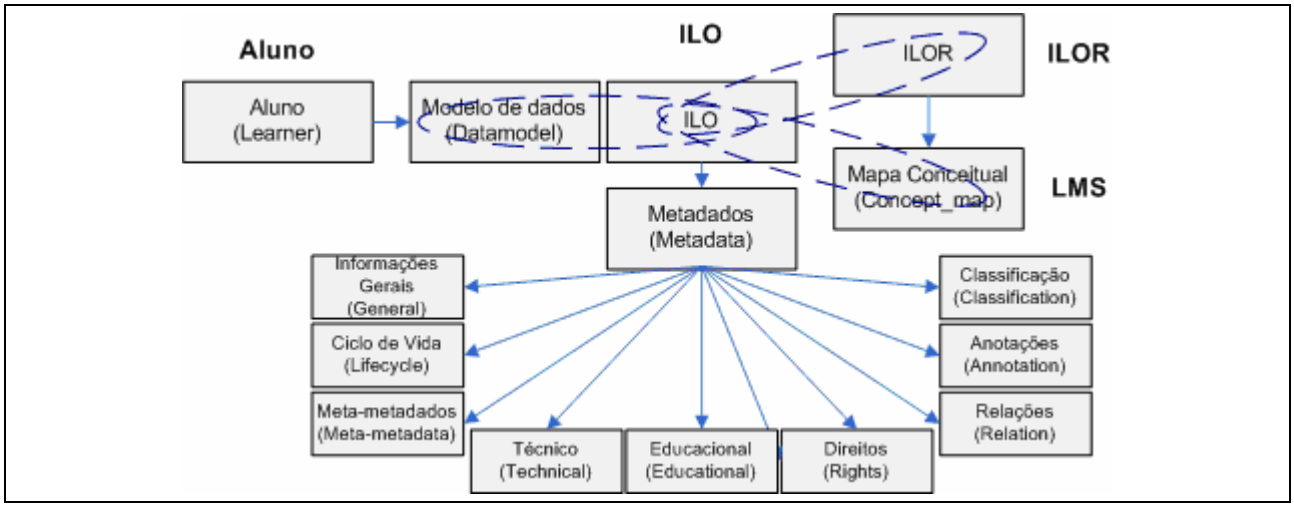

Figura 24. Conceitos da ontologia

A ontologia **ilo-ontology** contém informações sobre o LMS, o aluno, o objeto de aprendizagem e seu repositório. Os dados contidos referentes ao aluno incluem um identificador, seu nome e uma referência para o seu modelo de dados, o qual será responsável pelo armazenamento de sua aprendizagem ao utilizar cada objeto de aprendizagem. Já sobre os objetos de aprendizagem, tem-se um conjunto de informações que o caracterizam. A definição de quais variáveis seriam relevantes para se conhecer um objeto, é baseado no padrão IEEE 1484.12.1-2002 - *Learning Object Metadata* (2002), o qual inclui a relação entre o objeto e o aproveitamento do aluno..

O repositório dos objetos apresenta uma lista responsável por armazenar as referências dos objetos ativos, além de se conhecer o local onde tal repositório está localizado. Por fim, o LMS deve conhecer seus alunos, repositórios de objetos, e os mapas conceituais que definirão cada um dos cursos disponíveis no ambiente.

Outra entidade identificada é o mapa conceitual. Percebeu-se durante a pesquisa, a necessidade de uma estrutura que norteasse o desenvolvimento de um curso. Um mapa conceitual

permite que sejam identificados os conceitos a serem abordados, e relacioná-los através de pré-requisitos conceituais e temporais. Na arquitetura proposta, cada objeto inserido no SMA deverá ser relacionado a um ou mais conceitos, para que no momento de sua adaptação eles sejam identificados corretamente, evitando o problema de conceitos de mesmo nome, porém de semânticas diferentes. Por exemplo, o termo "janela" pode-se referir tanto a um elemento de uma obra / construção, quanto a uma interface de interação entre um software e um usuário. Ao inserir estes conceitos, dentro de um mapa, podem distinguir pelo seu contexto, oferecendo ao aluno um objeto semanticamente correto.

4.3.1 Conceitos

Os conceitos de uma ontologia, definem quais informações são necessárias para sua identificação e classificação dentro de um determinado domínio. Na ontologia **ilo-ontology** o conjunto de conceitos especificados têm suas propriedades e descrições detalhadas na seqüência.

concept-map

Propriedades: **:concept** *String* **:course-id** *Integer* **:ilo-id** *String* :**requisite** *set of instance of concept-map* **:requisit_temp** *set of instance of concept-map***.**

Descrição: Este conceito afirma que para cada termo presente em um mapa conceitual, pode-se relacioná-lo a um curso ou objeto de aprendizagem. Ainda, é possível indicar os termos que são requisitos conceituais e temporais.

course

Propriedades: **:course-id** *Integer* **:course-name** *String*.

Descrição: Este conceito afirma que podem ser estabelecidos cursos (ou disciplinas, conforme área de aplicação), os quais os objetos e mapas conceituais poderão ser estabelecidos.

data-model

Propriedades: **:content** *String* **:dm-ilo-id** *String*.

Descrição: Este conceito estabelece o desenvolvimento de modelo de dados que representem o conhecimento e desempenho do aluno para um determinado objeto de aprendizagem, independente do seu contexto de uso, isto é, os cursos que apresentaram o conteúdo.

ilo

Propriedades: **:ilo-id** *Integer* **:ilo-location** *String* **:ilo-metadata** *instance of metadata*.

Descrição: Este conceito descreve a estrutura dos objetos de aprendizagem, utilizando um identificador único, sua localização (em um LMS ou repositório) e seu conjunto de metadados.

ilor

Propriedades: **:ilor-id** *Integer* **:ilor-name** *String* **:ilor-location** *String*.

Descrição: Este conceito descreve a estrutura dos respositórios, utilizando um identificador único, nome e sua localização.

learner

Propriedades: **:course-id** *Integer* **:learner-dataModel** *set of instance of data-model* **:learner-id** *String* **:learner-name** *String*.

Descrição: Este conceito apresenta a estrutura de dados de um aluno no SMA, para tanto deve-se saber em qual curso ele está inserido em um momento, seu modelo de dados, identificador único e nome do aluno. Tais informações auxiliam o processo de adaptação da apresentação do objeto de aprendizagem ao aluno.

lom-annotation

Propriedades: **:date** *String* **:description** *String* **:entity** *String*.

Descrição: Este conceito segue o conjunto de metadados estabelecidos pelo padrão SCORM. Nele é possível informar anotações sobre o objeto, informando os seguintes dados relacionado a própria anotação: a data em que foi realizada, descrição e entidade relacionada.

lom-classification

Propriedades: **:description** *String* **:keyword** *set of String* **:purpose** *String* **:taxon-entity** *String* **:taxon-id** *String* **:taxon-source** *String*.

Descrição: Este conceito segue o conjunto de metadados estabelecidos pelo padrão SCORM. Nele é possível informar sobre a classificação do objeto, informando dados como sua descrição, palavras-chave, propósito, e sua taxonomia.

lom-educational

Propriedades: **:context** *String* **:description** *String* **:difficulty** *String* **:intended-user-role** *String* **:interacvity-level** *String* **: interacvity-type** *String* **:language** *String* **:resource-type** *String* **:semantic-density** *String* **:typical-age** *String* **:typical-learning-time**.

Descrição: Este conceito segue o conjunto de metadados estabelecidos pelo padrão SCORM. Nele é possível informar aspectos educationais sobre o objeto, tais como seu contexto de uso, dificuldade, nível e tipo de interatividade, idioma em que foi construído, tipo de recurso, densidade semântica, idade mais apropriada para aplciação e tempo de aprendizagem típico.

lom-general

Propriedades: **:aggregation-level** *String* **:catalog-entry** *set of String* **:catalogy** *set of String* **:coverage** *String* **:description** *String* **:keyword** *set of String* **:language** *String* **:structure** *String* **:title** *String*.

Descrição: Este conceito segue o conjunto de metadados estabelecidos pelo padrão SCORM. Nele é possível informar dados gerais sobre o objeto, tais como o nível de agregação, forma de catalogação, medidas, palavras-chave, idioma, estrutura e título.

lom-lifecycle

Propriedades: **:catalog-entry** *set of String* **:contribute-date** *set of String* **:contribute-entity** *set of String* **:contribute-role** *set of String* **:contribute-value** *set of String* **:status** *String* **:version** *String*.

Descrição: Este conceito segue o conjunto de metadados estabelecidos pelo padrão SCORM. Nele é possível inserir informações sobre o ciclo de vida do objeto, ou seja, manter um histórico ao longo do tempo através do registro de quem criou, revisou, modificou ou estabeleceu sua descontinuidade. O conceito permite marcar dados sobre sua catalogação; data, entrada, papel e descrição da contribuição realizada, status e versão atual do objeto.

lom-meta-metadata

Propriedades: **:contribute-date** *set of String* **:contribute-entity** *set of String* **:contribute-role** *set of String* **:identifier-catalog** *String* **:identifier-entity** *String* **:language** *String* **:metadata-schema** *String*.

Descrição: Este conceito segue o conjunto de metadados estabelecidos pelo padrão SCORM. Nele é possível inserir informações sobre os metadados do objeto, tais como contribuição, identificador no catálogo e de entrada, idioma, e esquema de metadados utilizado.

lom-relation

Propriedades: **:catalog-description** *String* **:catalog-entry** *set of String* **:kind** *String*.

Descrição: Este conceito segue o conjunto de metadados estabelecidos pelo padrão SCORM. Nele é possível inserir informações sobre a relação do objeto, exemplo: descrição e entrada do catálogo, além do tipo de relação.

lom-rights

Propriedades: **:copyright** *String* **:cost** *String* **:description** *String*.

Descrição: Este conceito segue o conjunto de metadados estabelecidos pelo padrão SCORM. Nele é possível inserir informações sobre os direitos autorais do objeto de aprendizagem, através da informação do tipo de licença, custo e descrição.

lom-technical

Propriedades: **:duration** *String* **:format** *set of String* **:installation-remarks** *String* **:location** *String* **:orComposite-maximum-version** *String* **:orComposite-minimum-version** *String* **:orComposite-name** *String* **:other-plataform** *String* **:requiriment** *String* **:size** *String*.

Descrição: Este conceito segue o conjunto de metadados estabelecidos pelo padrão SCORM. Nele é possível inserir informações técnicas sobre o objeto: tempo de duração, formato, observações de instação, localização, versão máxima-mínima e nome de composição, outras plataformas, requisitos técnicos para funcionamento e tamanho.

metadata

Propriedades: **:cAnnotation** *instance of lom-annotation* **:cClassification** *instance of lom-classification* **:cEducational** *instance of lom-educational* **:cGeneral** *instance of lom-general* **:Lifecycle** *instance of lom-lifecycle* **:cMetaMetadata** *instance of lom-meta-metadata* **:cRelation** *instance of lom-relation* **:cRights** *instance of lom-rights* **:cTechnical** *instance of lom-technical*.

Descrição: Este conceito reúne as instâncias das propriedades que contemplam o conjunto de metadados estabelecidos pelo padrão SCORM.

5 DESENVOLVIMENTO DO FRAMEWORK

A fim de compreender o funcionamento do SMA proposto, faz-se necessário o entendimento do processo da preparação do ambiente e como os objetos de aprendizagem se interligam através do SMA. Todo o desenvolvimento está diretamente relacionado às tecnologias necessárias para que a estrutura funcione. A Figura 25 apresenta a arquitetura geral desenvolvida.

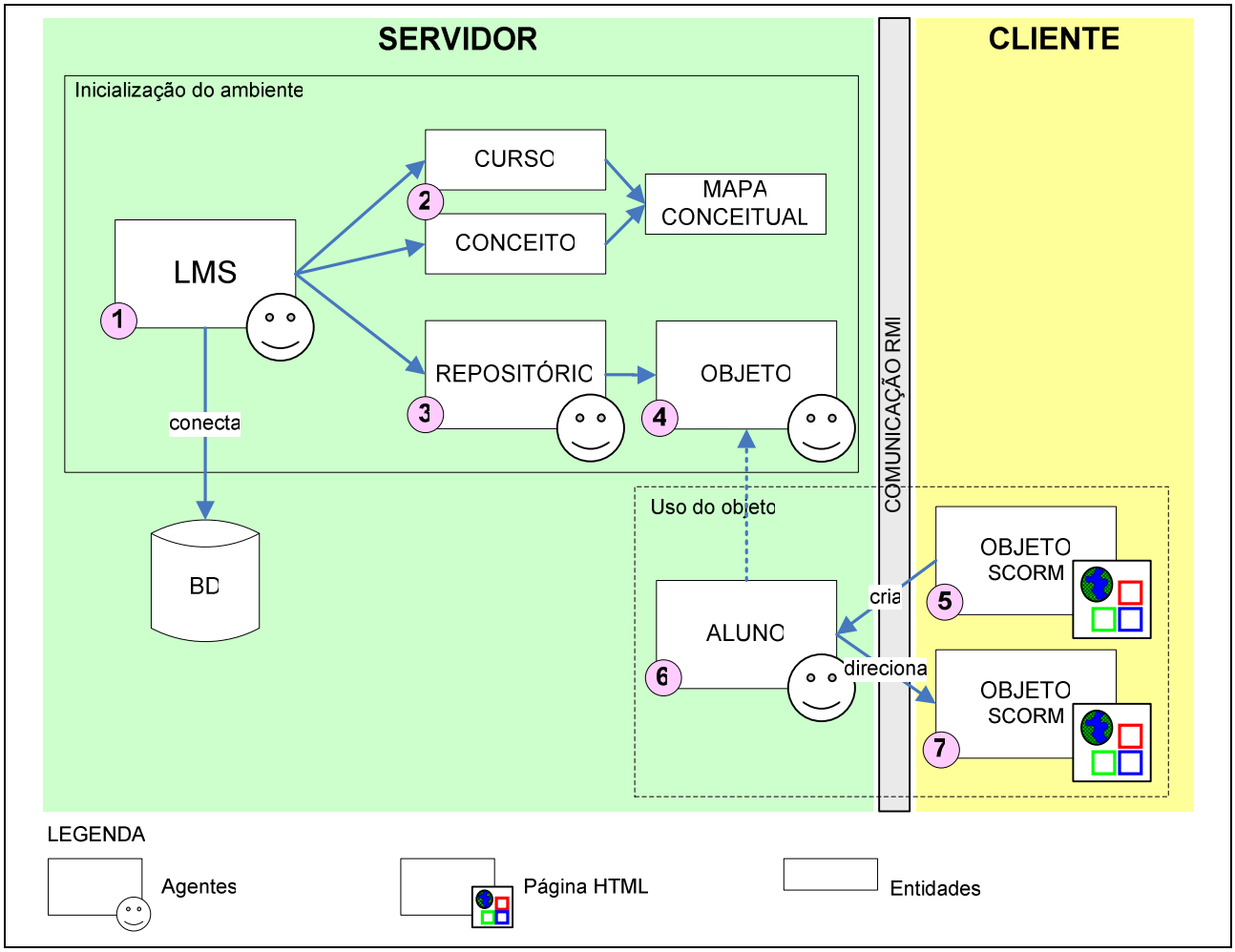

Figura 25. Arquitetura do *framework* proposto

A arquitetura representa três perspectivas que são detalhadas nas subseções a seguir. A seqüência foi delineada conforme o conhecimento necessário para a compreensão da arquitetura. Logo, a primeira relaciona o uso dos recursos tecnológicos para o desenvolvimento da aplicação. A segunda indica a seqüência do funcionamento da arquitetura. A terceira parte especifica a implementação de fato dos agentes que habitam o SMA.

5.1 TECNOLOGIAS RELACIONADAS

Sob o ponto de vista tecnológico, a arquitetura baseia-se no paradigma cliente-servidor. O lado servidor contém a plataforma JADE que gerencia os agentes, ontologias e regras de negócio; além de utilizar uma base de dados que realizará a persistência dos dados, facilitando as consultas para construção da estrutura da plataforma. Para a implementação do protótipo, foi utilizado o sistema gerenciador de banco de dados PostgreSQL e desenvolvido um modelo de dados com as mínimas informações utilizadas pelo SMA, a fim de simular a existência de um LMS.

Já do lado cliente consta o objeto de aprendizagem, o qual pode conter qualquer tipo de mídia que possa ser apresentada através de uma página HMTL, além da estrutura projetada pelo padrão o qual foi desenvolvido.

Para que estas duas estruturas se comuniquem foi utilizada a tecnologia RMI promovida pela linguagem Java. Isto permite que independentemente do LMS utilizado para execução dos objetos, é possível se conectar ao SMA, desde que implemente a API do modelo SCORM. Para que isto ocorra, é necessário apenas modificar: a página HTML de acesso ao objeto, incluindo um applet não visível ao aluno que se conectará ao agente, que por sua vez implementará a API do padrão SCORM, que irá invocar a conexão com o SMA.

5.1.1 Detalhes de implementação através do JADE

O desenvolvimento do SMA foi realizado através do *framework* JADE na versão 3.4.1. Devido às suas restrições de implementação nativas, até a versão utilizada, não é permitida a instanciação nem a comunicação direta[6] de seus agentes e serviços web, sejam aplicações cliente (*applet*) ou servidor (*jsp* ou *servlet*). Para tanto, foi necessário desenvolver um mecanismo de comunicação através de invocação remota de método (RMI – *Remote Method Invocation*), a qual especifica que deve haver um servidor de serviços e clientes que irão conectar-se a este servidor em busca de um determinado serviço. Logo, deve-se utilizar uma porta comum onde o serviço, também de nome conhecido, poderá ser localizado.

O servidor deve realizar a comunicação com o SMA através dos recursos de *Gateway* disponibilizados pelo *framework* do JADE; e o cliente, desenvolvido com recursos de applet,

[6] O JADE disponibiliza um mecanismo para aplicações móveis, JADE Web Services Integration Gateway (WSIG), o qual utiliza recursos do servidor web Axis, UDDI e a tecnologia SOAP.

conecta-se ao servidor. O serviço de *Gateway* permite a criação de um agente (através da classe `GatewayAgent`) que irá interagir com os demais agentes do SMA através da troca de mensagens, processo interpretado através da especificação de um comportamento.

Para o desenvolvimento dos agentes, eles devem herdar a classe `Agent` e então implementar o comportamento do agente através do método `Setup`. A classe `Agent` implementa mais de 40 operações (na versão utilizada), entretanto para o desenvolvimento do SMA foram utilizadas apenas as operações apresentadas na Figura 26.

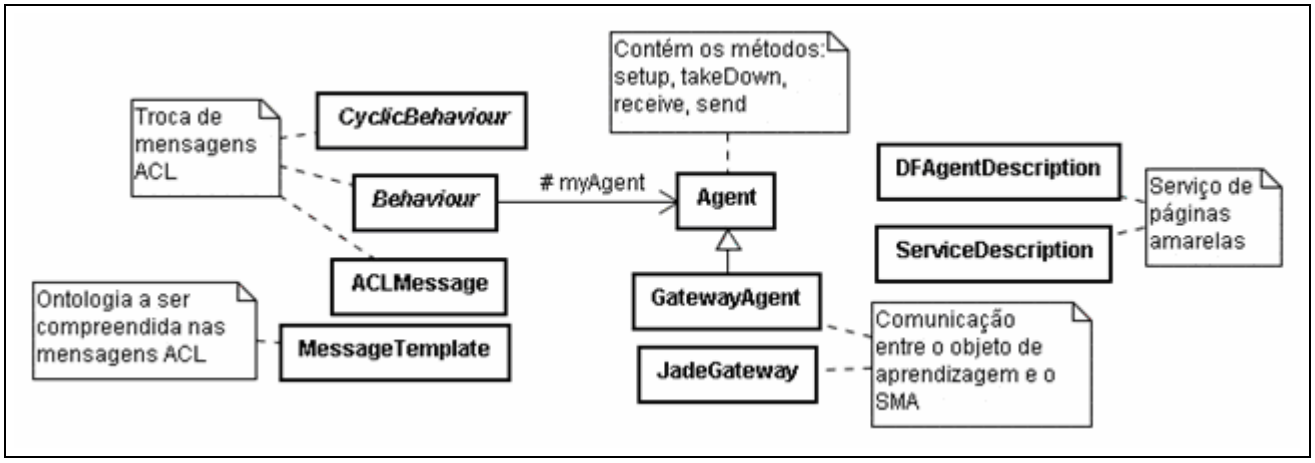

Figura 26. Principais classes do *framework* utilizadas para implementar o SMA

A estrutura interna de um agente da classe `Agent` é ilustrada pela Figura 27. O método `Setup` é o responsável pelo funcionamento do agente, equivalendo ao método `public static void main(String args[])` da linguagem Java. Quando o agente nasce no SMA este método é percorrido. Caso não implemente algum *listner*, ao terminar os comandos ele será também finalizado. Caso contrário poderá ficar aguardando o recebimento de mensagens até que o processo do SMA seja interrompido, e consequentemente, finalizado.

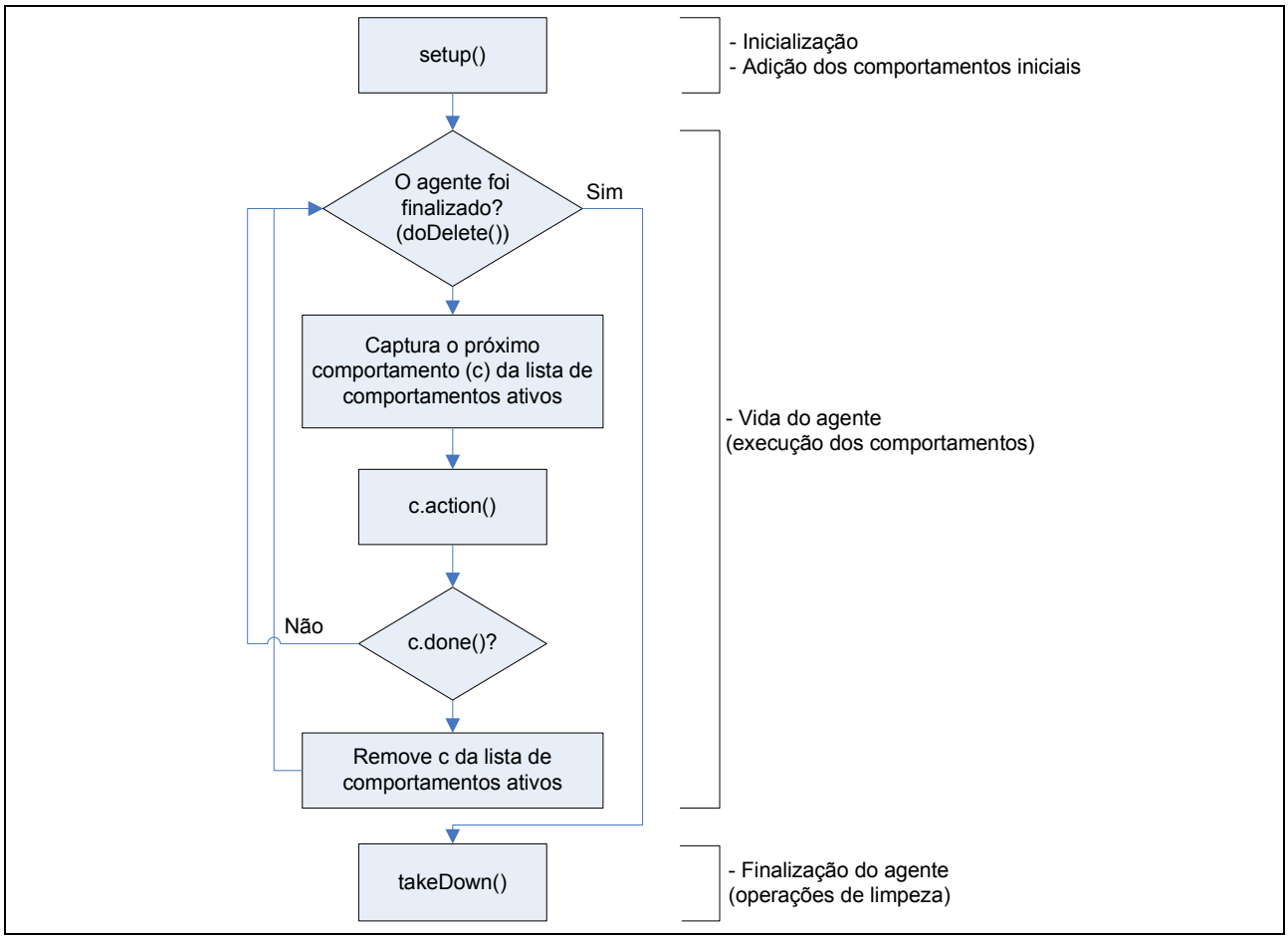

Figura 27. Estrutura interna do agente JADE

Fonte: Adaptado de JADE (2007)

Geralmente, os *listeners* implementam funcionalidades do comportamento do agente, isto é, das classes `Behaviour` e `CyclicBehaviour`. Estas classes permitem que seja verificado o recebimento de mensagens constantemente e então realizado o tratamento necessário, utilizando os métodos `receive` (para a verificação da chegada de mensagens ACL) e `send` (para o envio de mensagens ACL) da classe `Agent`. Já o método `takeDown` permite implementar determinada ação antes que o agente seja removido do SMA, enquanto o `getArguments` permite capturar argumentos informados para a especificação do agente a ser criado.

Os demais métodos complementam a implementação de um agente JADE possibilitando sua transição de status no ciclo de vida, resgatando informações do próprio agente, etc. Além destes métodos, foram utilizadas outras classes, tais como: `DFAgentDescription` e `ServiceDescription` a fim de inscrever os agentes no serviço de páginas amarelas; `MessageTemplate` para registrar a

ontologia na qual o agente está apto a compreender as mensagens trocadas; e `GatewayAgent` e `JadeGateway` para a comunicação entre o objeto de aprendizagem (HTML) e o SMA (Java).

5.2 CICLO DE VIDA DO SMA

Retomando a arquitetura ilustrada na Figura 25, percebe-se a existência de círculos com números. Eles representam a seqüência do ciclo de vida do SMA, ou seja, quais são os passos a serem seguidos para seu funcionamento pleno. Cada uma das etapas é especificada na seqüência.

1. Preparação do ambiente através do agente LMS: Para que a estrutura funcione, faz-se necessária a preparação do ambiente por meio da execução do servidor JADE, de registro e servidor serviços RMI. Juntamente a inicialização da plataforma JADE, é criada uma instância do agente LMS, responsável pelo gerenciamento da estrutura interna do SMA.
2. Criação das entidades de apoio (cursos, conceitos e mapas): O agente LMS verifica na base de dados do LMS os cursos existentes, estabelecendo os mapas e conceitos que são abordados. Estas entidades apóiam a tomada de decisão através da entidade de estratégia de ensino abordada.
3. Criação dos agentes que representam os repositórios de objetos de aprendizagem: Na seqüência, o agente LMS verifica quais são os repositórios de objetos de aprendizagem que o LMS trabalha e então cria sua representação através de agentes.
4. Criação dos agentes que representam objetos de aprendizagem: A partir da identificação dos repositórios, são verificados os objetos existentes e então criados seus agentes. Os agentes terão as informações carregadas através do arquivo de metadados que especificam o conteúdo dos objetos.

Os passos descritos até o momento dizem respeito a estrutura de serviços que devem ser estabelecidos antes da execução dos objetos. A seguir são apresentadas as etapas relacionadas ao fluxo de exibição dos objetos de aprendizagem, o qual pode ocorrer diversas vezes, conforme a solicitação de apresentação dos conteúdos.

5. Inicialização do objeto de aprendizagem através do LMS: Através de um LMS, o aluno solicita o acesso a um objeto de aprendizagem.

6. Criação do agente que representa o aluno utilizando o objeto: Juntamente com a etapa anterior, também é estabelecida conexão com a plataforma JADE, onde é criado um agente que representa o ato do aluno visualizar o conteúdo através do agente LMS.
7. Redirecionando a outro objeto: Durante a execução do objeto de aprendizagem selecionado pelo aluno, o SMA pode perceber a necessidade de complemento da aprendizagem. Nesse momento, pode ser encaminhado um outro objeto a esse aluno. Cabe ressaltar que esta ação não é obrigatória, ela ocorre conforme a necessidade individual do aluno.

Sobre a especificação do comportamento de cada um dos agentes desenvolvidos na arquitetura, tal como os diálogos estabelecidos entre eles, mostra-se uma descrição detalhada na seção seguinte.

5.3 AGENTES DESENVOLVIDOS

Conforme descrito na modelagem, foram desenvolvidos quatro agentes para atender as necessidades do SMA: `LMSAgent`, `ILORAgent`, `ILOAgent` e `LearnerAgent`. Conforme apresentado no diagrama de classes da Figura 28, cada um dos elementos retangulares representa uma classe Java com um determinado propósito. As classes `ILORAgent`, `LearnerAgent` e `ILOAgent` representam os agentes que habitam o SMA e, portanto, estendem as características de um agente padrão da plataforma JADE (através da classe `Agent`). Já a classe `LMSAgent` estende as características da classe `GatewayAgent`. Ainda, a `LMSAgent` conecta-se à base de dados do LMS a fim de obter informações sobre os cursos existentes; enquanto a `ILOAgent` implementa o tratamento das informações provenientes do arquivo de metadados associado ao objeto de aprendizagem através da classe `NodeLister`.

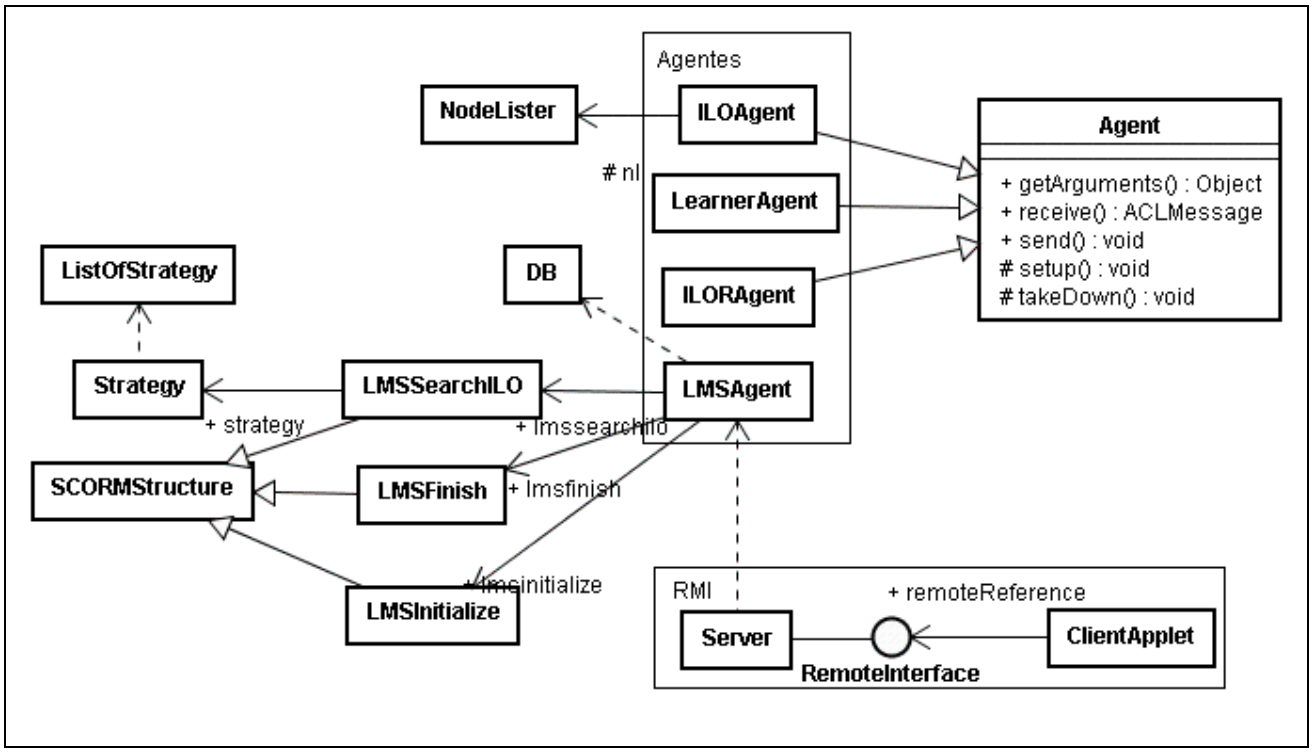

Figura 28. Diagrama de classes implementadas

As classes `ClientApplet`, `Server` e `RemoteInterface` são responsáveis pela estrutura de comunicação RMI, onde:

- ClientApplet: a página HTML irá instanciar esta classe para manipulação das informações através da chamada de seus métodos; também irá se comunicar via RMI com a classe `Server` (servidor);
- Server: o servidor deverá ser executado no equipamento que fornecerá o serviço web e o SMA; é uma extensão da classe de interface `RemoteInterface`;
- RemoteInterface: implementa a classe de interface que estende a interface `Remote`, possibilitando a comunicação RMI.

As classes `LMSInitialize`, `LMSSearchILO` e `LMSFinish` estendem a classe padrão denominada de `SCORMStrutcture`. Estas classes especificam cada um dos comportamentos processados pelo `LMSAgent`. A necessidade de implementar uma classe para cada comportamento desejado, consiste em uma especificação do *framework*, o qual não permite a simples chamada de método, mas em passagem de objetos processados por um método próprio dos agentes de tipo *Gateway*. A partir deste processamento prévio, é então possível a realização da comunicação entre os agentes, através da troca de mensagens. Por fim, as classes `Strategy` e `ListOfStrategy`

permitem a implementação da estratégia de seleção do objeto de aprendizagem mais adequado a necessidade do aluno.

Devido a complexidade de cada um dos agentes desenvolvidos nas classes descritas anteriormente, além da necessidade de descrever os diálogos por eles trabalhados, são apresentadas as subseções da seqüência.

5.3.1 LMSAgent

O agente LMS difere dos demais, que serão descritos na seqüência, pois precisa herdar as funcionalidades da classe `GatewayAgent`. Isto é necessário pois ele é responsável pelo recebimento dos eventos enviados pela API e tranformá-lo em mensagens ACL. O `GatewayAgent` descreve dois métodos para o funcionamento do agente: o `Setup` que implementa o funcionamento interno do agente, e o `processCommand` que é responsável pela interpretação das informações recebidas pela API. Isto ocorre, pois a API não tem condições de enviar mensagens ACL diretamente. Para isto, um applet Java que serve de cliente RMI, o qual trabalha em conjunto com a API implementada em JavaScript onde os métodos são invocados.

No outro lado, o servidor de RMI recebe a solicitação e as encaminha ao LMSAgent através de objetos implementados como comportamento, que então são interpretados através do método `processCommand`.

De forma geral, o fluxo do processamento das informações externas é descrito na Figura 29. Caso seja recebido o objeto LMSInitialize, o agente submete a mensagem `born-agent` ao agente LMS. Já, se o objeto recebido for o LMSSearchILO, localizam-se os agentes ILO que estiverem no SMA e submete a cada um deles a mensagem `search-ilo`. Por fim, ao receber o objeto LMSFinish, é enviado ao LearnerAgent a mensagem de `dead-agent`.

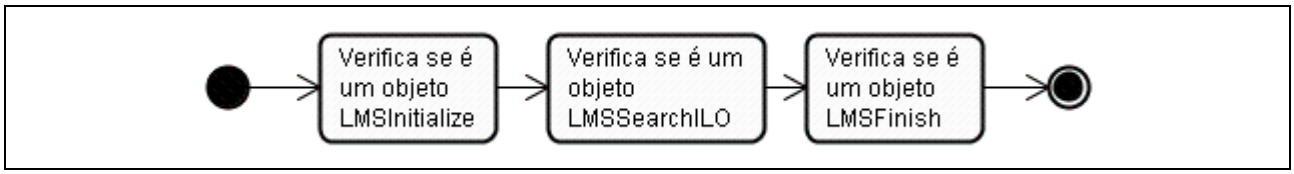

Figura 29. Comportamento interno do método de processamento de comandos do LearnerAgent

Já no fluxo do comportamento interno do `LearnerAgent`, o agente registra-se no serviço de páginas amarelas do JADE de modo a tornar disponível seus recursos a plataforma de SMA. Na seqüência, o agente LMS define e cria a estrutura necessária para seu funcionamento: (i)

implementa os agentes que realização o papel de repositório de objetos, (ii) carrega os cursos e conceitos a fim de estabelecer os mapas conceituais para então (iii) criar os agentes que representarão os objetos. Este processo inicial conta com o apoio de um mecanismo de persistência dos dados; neste caso foi desenvolvido um banco de dados cujo modelo é mostrado no Apêndice A. Diagrama Entidade-Relacionamento. O SMA proposto funciona independentemente da existência da camada de persistência; seu desenvolvimento foi realizado a fim de facilitar os testes e para compreender como se daria tal integração. Na seqüência, o agente fica aguardando o recebimento de mensagens de confirmação da realização das ações solicitadas através dos diálogos `born-agent` e `dead-agent`. Tais ações são representadas na Figura 30.

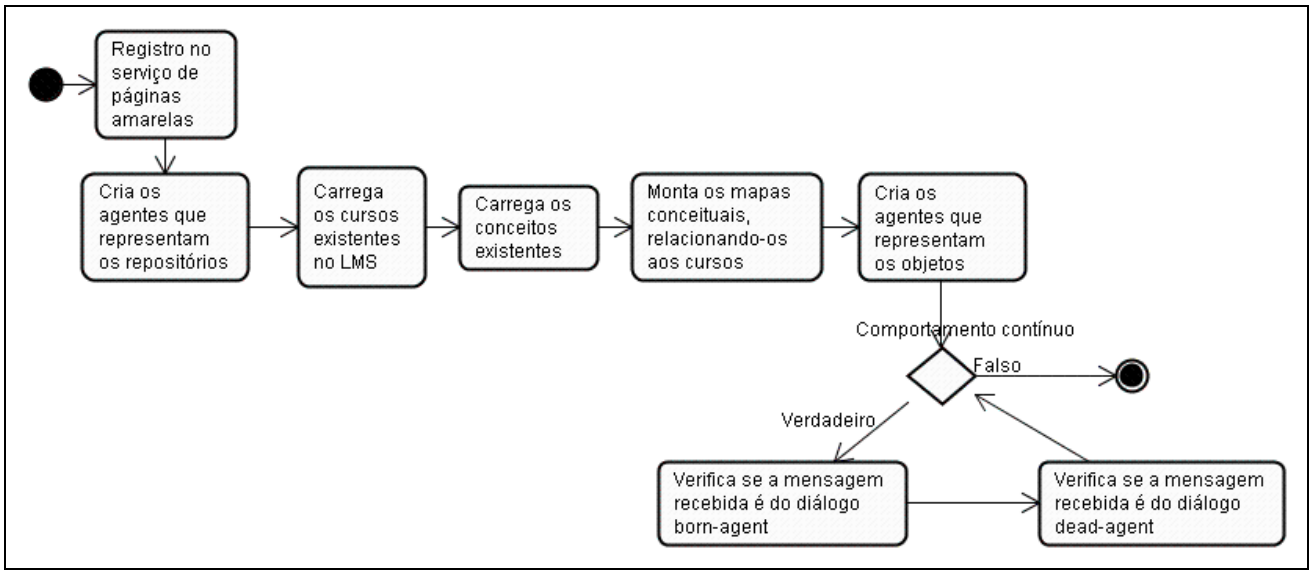

Figura 30. Comportamento interno do LMSAgent

5.3.2 ILORAgent

O ILORAgent representa os repositórios onde os objetos de aprendizagem são agrupados e organizados. Eles podem estar implantados dentro de um LMS ou distribuídos pela internet. No SMA, para cada repositório utilizado pelo LMS, é designado um agente para representá-lo.

Para o presente trabalho, não se identificou a necessidade de desenvolver diálogos específicos ao agente ILOR. Logo, seu ciclo de vida consiste em apenas se registrar no serviço de páginas amarelas, conforme é apresentado na Figura 31.

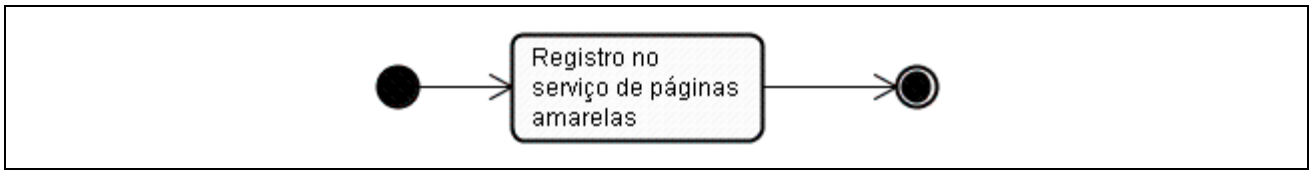

Figura 31. Comportamento interno do ILORAgent

5.3.3 ILOAgent

O ILOAgent é responsável por disponibilizar uma estrutura de informações e troca de mensagens sobre cada um dos objetos de aprendizagem que estejam cadastrados no LMS. A quantidade de objetos que existir fisicamente, representara o número de agentes deste tipo que habitarão o SMA. Não foram realizadas verificações quanto o desempenho do framework JADE diante um número expressivo de agentes, por não se tratar do escopo da pesquisa.

De forma semelhante aos demais agentes, na criação do agente ILO é realizado seu registro no serviço de páginas amarelas do JADE. Na seqüência, carrega-se a sua estrutura de informações a partir do arquivo de metadados do objeto. Para desenvolver esta funcionalidade, utilizou-se a biblioteca denominada JDOM (2006), o qual implementa uma forma de carregamento de informações contidas em um XML (*eXtensible Markup Language*). As informações carregadas são selecionadas e então preenchidas na estrutura definida pela ontologia desenvolvida. A Figura 32 ilustra tal processo.

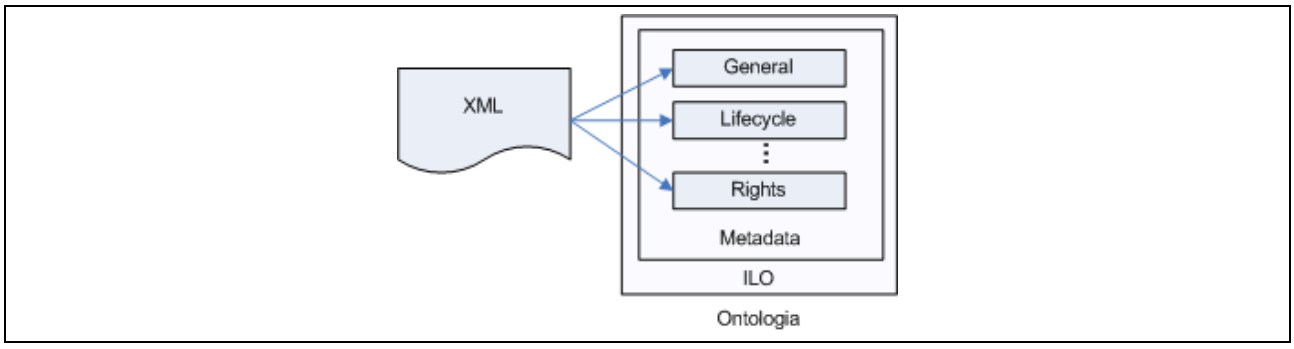

Figura 32. Conversão dos dados XML para a ontologia

Por fim, o agente ILO implementa a classe `CyclicBehavior` o qual interpreta mensagens do diálogo `search-ilo`. A estrutura interna do agente é apresentada na Figura 33.

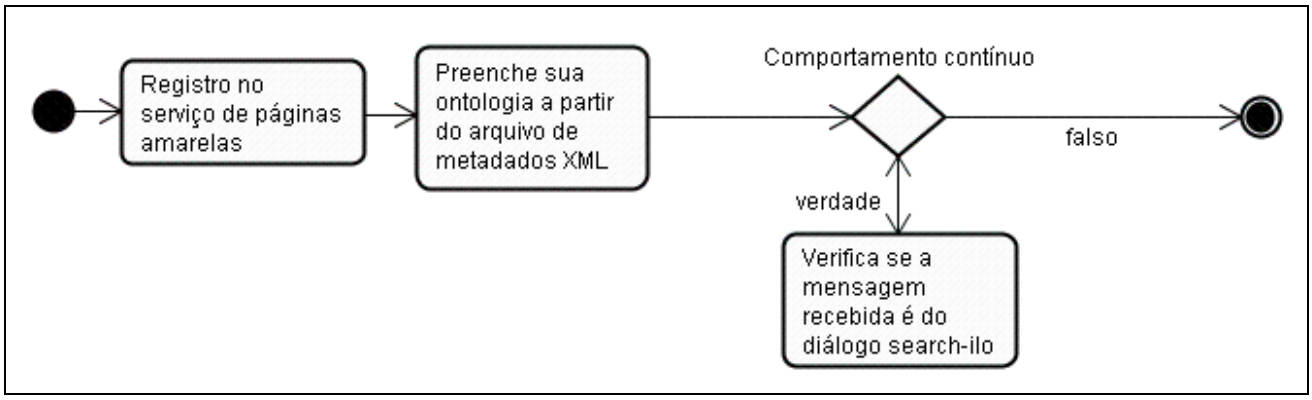

Figura 33. Comportamento interno do ILOAgent

Dentro da proposta realizada, compreendeu-se que este tipo de agente deve ser o mais presente no SMA, já que representa a existência de cada objeto.

5.3.4 LearnerAgent

O agente que representa o aluno no SMA, denominada de LearnerAgent, difere dos agentes anteriores pois não é criado a partir das informações da camada de persistência. Ele é iniciado a partir da ação de utilização de um objeto de aprendizagem, ou seja, sua criação ocorre no momento em que se invoca o método LMSInitialize e tem sua finalização determinada pelo LMSFinish. Logo, seu papel é de representar o uso momentâneo do objeto pelo aluno. Seu funcionamento está diretamente relacionado pelo agente APISCORMAgent, responsável pela mediação do objeto de aprendizagem com o SMA.

Seu funcionamento interno consiste em registrar-se no serviço de páginas amarelas e então implementar o `CyclicBehavior` a fim de compreender mensagens do tipo `finish-experience` e `get-datamodel`, conforme é mostrado na Figura 34. Ao receber o diálogo `finish-experience`, realiza a finalização do próprio agente, enquanto o `get-datamodel` recupera o modelo do aluno.

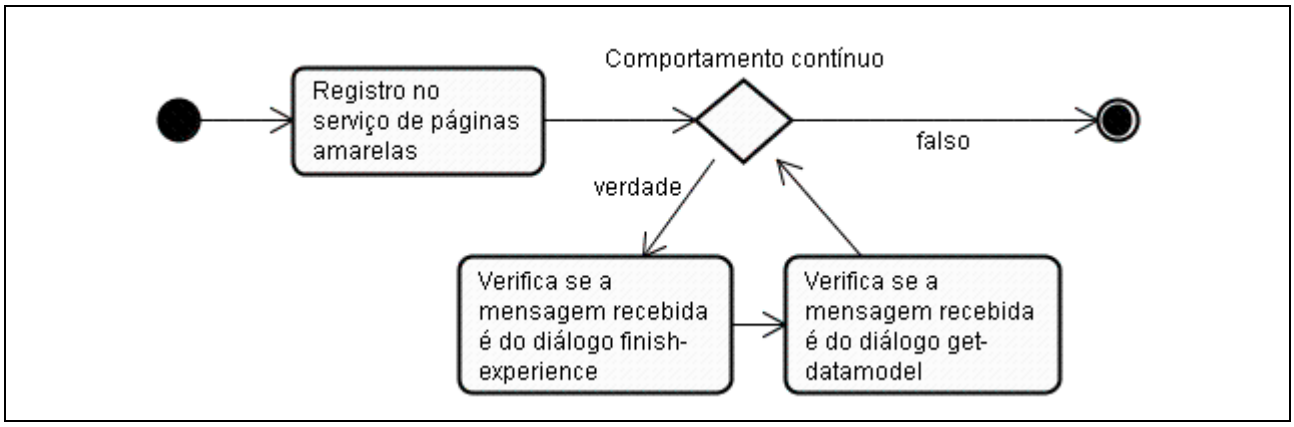

Figura 34. Comportamento interno do LearnerAgent

5.4 DIÁLOGOS DESENVOLVIDOS

Considerando os agentes desenvolvidos a partir da pesquisa, mais aqueles disponibilizados pelo framework do JADE, foram estabelecidos alguns diálogos para a comunicação do SMA. A listagem dos diálogos necessários para o funcionamento do SMA é apresentado na Tabela 5.

Tabela 5. Diálogos utilizados no *framework*

Emissor	**Receptor**	**Diálogo**	**Protocolo FIPA**	**Ontologia**	**Descrição**
LMSAgent	AMS	added-container	Nenhum	JADE-Introspection	Cria novo contêiner
LMSAgent	AMS	born-agent	Nenhum	JADE-Introspection	Cria um novo agente (LearnerAgent)
LMSAgent	AMS	dead-agent	Nenhum	JADE-Introspection	Destrói um agente (LearnerAgent)
LMSAgent	DF	search	RIPS	FIPA-Agent-Management	Busca agentes que atenda a um determinado serviço
Qualquer agente	DF	register	RIPS	FIPA-Agent-Management	Registra o agente no serviço de páginas amarelas
APISCORMAgent	ILOAgent	search-ilo	CNIPS	ILO-Ontology	Busca um ILO que atenda a um determinado critério
ILOAgent	LearnerAgent	get-datamodel	RIPS	ILO-Ontology	Obtém informações sobre o modelo de dados de um aluno

Legenda – Protocolos FIPA:
RIPS – Request Interaction Protocol Specification (SC00026H)
CNIPS – Contract Net Interaction Protocol Specification (SC00029H)

Em cada uma das subseções a seguir, é detalhado o funcionamento de cada um dos diálogos, a iniciar por aqueles que fazem parte da plataforma JADE, pois são necessários para o funcionamento dos diálogos desenvolvidos na pesquisa.

5.4.1.1 Cria novo contêiner

Quando o agente LMS é iniciado, existe a necessidade por parte do *framework* JADE, daí criação de um contêiner, pois é neste local que os agentes serão inseridos. Inicialmente, existirão dois contêineres: o primeiro é chamado Main-Container, padrão do JADE, que conterá os agentes MAS, RMA e DF; o segundo é o criado a partir da criação do agente LMS. Este último é necessário quando se trabalha com o agentes da classe `AgentGateway`. Um exemplo da mensagem enviada para o MAS é apresentada na Figura 35.

```
((occurred
     (event-record
       :what
         (added-container
           :container
             (container-ID
               :name Container-1
               :address localhost))
       :when 20071104T182536687Z
       :where
         (container-ID
           :name Main-Container
           :address localhost))))
```

Figura 35. Exemplo de diálogo para criação de contêiner

5.4.1.2 Cria novo agente

A criação de novos agentes pode ser feita através de uma linha de comando via *prompt* do DOS, através de chamadas remotas ou de programação do *framework* JADE. No caso do agente que representa o LMS, ele é criado via chamada remota; já os demais agentes são criados via programação, tal como o exemplo apresentado na Figura 36.

```
novoLearner = new AID(AIDName, AID.ISLOCALNAME);
try {
        AgentContainer container = (AgentContainer)getContainerController();
        AgentController novoAgente;
        novoAgente = container.createNewAgent(novoLearner.getName(), "LearnerAgent", null);
        novoAgente.start();
}
catch (Exception ioe) {
        System.err.println("Erro na criação: " + ioe.getMessage());
}
```

Figura 36. Exemplo de código-fonte para criação de agente

Independentemente da forma de solicitação de novo agente, o processo de criação é convertido em mensagem que é recebida pelo agente AMS, para que então seja criado o agente de

fato. Isto ocorre com os agentes de `ILORAgent` , `ILOAgent` e `LearnerAgent`. A Figura 37 ilustra um exemplo de mensagem `born-agent`, que cria um agente em um determinado contêiner.

```
((occurred
     (event-record
       :what
         (born-agent
           :agent
             (agent-identifier
               :name ControlContainer-1@localhost:1099/JADE)
           :where
             (container-ID
               :name Container-1
               :address localhost)
           :state active
           :ownership NONE)
       :when 20071104T182536906Z
       :where
         (container-ID
           :name Main-Container
           :address localhost))))
```

Figura 37. Exemplo de diálogo para criação de agente

5.4.1.3 Destrói agente

No caso do framework proposto, a finalização de um agente ocorre quando um aluno deixa de interagir com um objeto de aprendizagem. De forma prática, a destruição de um agente ocorre quando o método `doDelete()` é invocado. Imediatamente o agente acessa o método `takeDown()`, responsável pela finalização das informações e estados do agente. Ao realizar estas ações, o agente é finalmente encerrado a partir do envio de uma mensagem do tipo `dead-agent`, como ilustra a Figura 38.

```
((occurred
     (event-record
       :what
         (dead-agent
           :agent
             (agent-identifier
               :name Julia_1@localhost:1099/JADE
               :addresses (sequence http://localhost:7778/acc))
           :where
             (container-ID
               :name Main-Container
               :address localhost))
       :when 20071104T182912453Z
       :where
         (container-ID
           :name Main-Container
           :address localhost))))
```

Figura 38. Exemplo de diálogo para destruição de agente

5.4.1.4 Registra no serviço de páginas amarelas

Todos os agentes desenvolvidos para a presente pesquisa tem como ação inicial, o seu registro no serviço de páginas amarelas, o que facilita a identificação dos processos em que podem atuar dentro do SMA. Um exemplo de registro do `LearnerAgent` é apresentado na Figura 39.

```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sd = new ServiceDescription();
sd.setType("Learner");
sd.setName("Learner-service");
dfd.addServices(sd);
```

Figura 39. Exemplo de código-fonte para registro no serviço de páginas amarelas

O conjunto de comandos apresentados, são traduzidos em forma de mensagem que é encaminhada ao agente DF, no formato apresentado pela Figura 40.

```
((action
     (agent-identifier
       :name df@localhost:1099/JADE
       :addresses (sequence http://localhost:7778/acc))
     (register
       (df-agent-description
         :name
           (agent-identifier
             :name Main-Container@localhost:1099/JADE
             :addresses (sequence http://localhost:7778/acc)
             :X-JADE-agent-classname LearnerAgent)
         :services
           (set
             (service-description
               :name Learner-service
               :type Learner))))))
```

Figura 40. Exemplo de diálogo para registro no serviço de páginas amarelas

5.4.1.5 Busca no serviço de páginas amarelas

A partir do momento que os agentes estão registrados no DF, é possível realizar uma busca para conhecer quais agentes podem realizar um determinado processo. Para tanto, o agente deve implementar o trecho baseado no *framework* JADE conforme mostra a Figura 41.

```
DFAgentDescription template = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType("ilo");
template.addServices(sd);
DFAgentDescription[] result = DFService.search(this, template);
AID[] sellerAgents = new AID[result.length];
```

Figura 41. Exemplo de código-fonte para busca no serviço de páginas amarelas

Tal solicitação de busca é convertida em uma mensagem ACL enviada ao agente DF, que irá retornar em forma de um vetor de dados os agentes que se encaixarem nos requisitos da busca. O exemplo de mensagem buscando os agentes que representam os objetos de aprendizagem é ilustrada na Figura 42.

```
((action
     (agent-identifier
       :name df@localhost:1099/JADE
       :addresses (sequence http://localhost:7778/acc))
     (search
       (df-agent-description
         :services
           (set
             (service-description
               :type ilo)))
```

Figura 42. Exemplo de diálogo para busca no serviço de páginas amarelas

5.4.1.6 Busca por outro objeto (search-ilo)

Um dos pontos fundamentais para o desenvolvimento de objetos de aprendizagem com recursos de sistemas multi-agentes é permitir que o objeto perceba que o aluno não está aprendendo o conteúdo apresentado e então buscar objetos alternativos. É neste sentido que o diálogo `search-ilo` está inserido no SMA.

O diálogo `search-ilo` baseia-se no protocolo FIPA *Contract Net Interaction Protocol Specification* – SC00029, onde o `APIScormAgent` busca através do serviço de páginas amarelas, agentes ativos no SMA e que representam objetos de aprendizagem. Quando são identificados agentes com tais características, é enviada uma mensagem do tipo `CFP` (*call for proposes*) com conteúdo de busca desejado.

Os agentes ILO que apresentarem as características adequadas, irão responder uma mensagem de `agree` confirmando a solicitação, e na seqüência, enviando as informações para o `LMSAgent`. Aqueles que não atenderem às necessidades, não irão responder a fim de não sobrecarregar o SMA com mensagens desnecessárias.

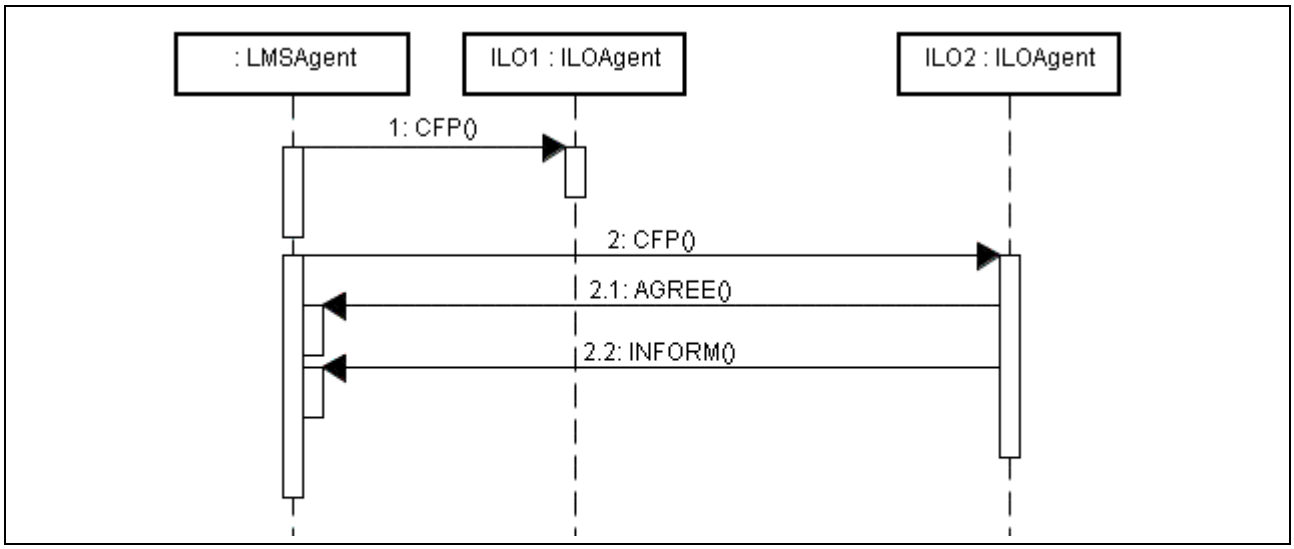

Figura 43. Diagrama de seqüência do diálogo *search-ilo*

No momento da chegada do `agree`, o objeto de aprendizagem é qualificado quanto a sua capacidade de aprendizagem e inserido em uma lista de objetos que responderam a solicitação. A qualificação fica a critério do especialista no momento da utilização do *framework* proposto: quais metadados são relevantes, quais podem ser descartados, que peso deve ser dado a cada critério avaliado, etc.

6 VALIDAÇÃO DO FRAMEWORK

A fim de verificar o *framework* proposto, foi desenvolvido um estudo de caso envolvendo a construção de objetos de aprendizagem. Primeiramente, foi selecionado um contexto e então desenvolvidos os conteúdos de aprendizagem.

Cada objeto de aprendizagem construído contém uma explicação textual seguido de um exercício de verificação do conteúdo aprendido. Em seguida, tiveram seus metadados informados e então empacotados sob o modelo de desenvolvimento SCORM. Cada uma das etapas da validação é apresentada na seqüência.

6.1 INVESTIGAÇÃO DO USO DA FERRAMENTA

Foram construídos alguns objetos de aprendizagem através da ferramenta eLearning XHTML Editor (eXe, 2007), versão 1.01. O editor permite o desenvolvimento de objetos para ambientes virtuais agregando metadados nos padrões SCORM e DublinCore, dando ênfase ao último. O eXe não realiza apenas o empacotamento do conteúdo, mas possibilita o desenvolvimento de hipertextos, exercícios com diversos tipos de questões, além da associação de animações em Flash e applets em Java. O objeto gerado pode ser registrado e interagir com qualquer LMS que suporte o SCORM ou DublinCore, sendo então exibido aos alunos.

Ao desenvolver os objetos, percebeu-se que devido a ênfase da ferramenta ao padrão DublinCore, nem todos os metadados do padrão SCORM podem ser preenchidos através da ferramenta. Este fator poderia limitar o potencial da localização de objetos, devido a ocultação de informações. Nesse momento, pensou-se em três possíveis soluções:

- Trocar de editor: que utilize e exporte os metadados do SCORM. Existem ferramentas que realizam esta função, como a Click2Learn (2007), contudo elas pecam na construção dos objetos no aspecto pedagógico, não disponibilizando mecanismos para confecção de conteúdo instrucional textual e de avaliação, servindo apenas como empacotador de arquivo;
- Trocar de padrão LOM: caso a ferramenta utilizada continuasse sendo a eXe, verificou-se qual seria a complexidade de adequação do SMA, desenvolvido até momento, para adaptá-lo ao padrão DublinCore. Foram realizados alguns testes e não foi notada

dificuldade alguma nesse sentido, embora fosse necessária uma revisão textual do presente documento. Outro fator limitador, deve-se ao fato do padrão DublinCore utilizar um conjunto reduzido de metadados, o que reduz a potencialidade de identificação de localização de objetos; ou

- Permanecer com a situação original: neste caso, seria realizada uma tentativa de trabalhar com os metadados disponíveis, contando com o suporte da construção do material instrucional existente.

A terceira opção pareceu ser a mais interessante por explorar a potencialidade do editor, já que o padrão de metadados pode ser substituído sem dificuldade, caso seja necessário. Vale ressaltar que a presente pesquisa não visa se deter em aspectos sobre o LMS, tampouco a execução dos objetos inteligentes e integração com o SMA. Logo, uma ferramenta de desenvolvimento de objetos que viabilize mecanismos de verificação da aprendizagem é válida. No caso da eXe (2007) é possível construir objetos com exercícios de verificação auto-corrigíveis, sem a necessidade de integração com o LMS. Isto indicaria o aproveitamento obtido pelo aluno e seu direcionamento a outros conteúdos.

6.2 CONSTRUÇÃO DOS OBJETOS DE APRENDIZAGEM

Para a realização dos testes com o SMA optou-se por confeccionar objetos de aprendizagem relacionados ao ensino de Algoritmos. O motivo da seleção deste tema deve-se ao fato da existência de um mapa conceitual e do material instrucional desenvolvido na monografia da autora (SILVA, 2005). A Figura 44 apresenta cada um dos conceitos agrupados em nível superior por unidades de ensino. As flechas contínuas representam os requisitos conceituais, enquanto as tracejadas indicam os requisitos temporais.

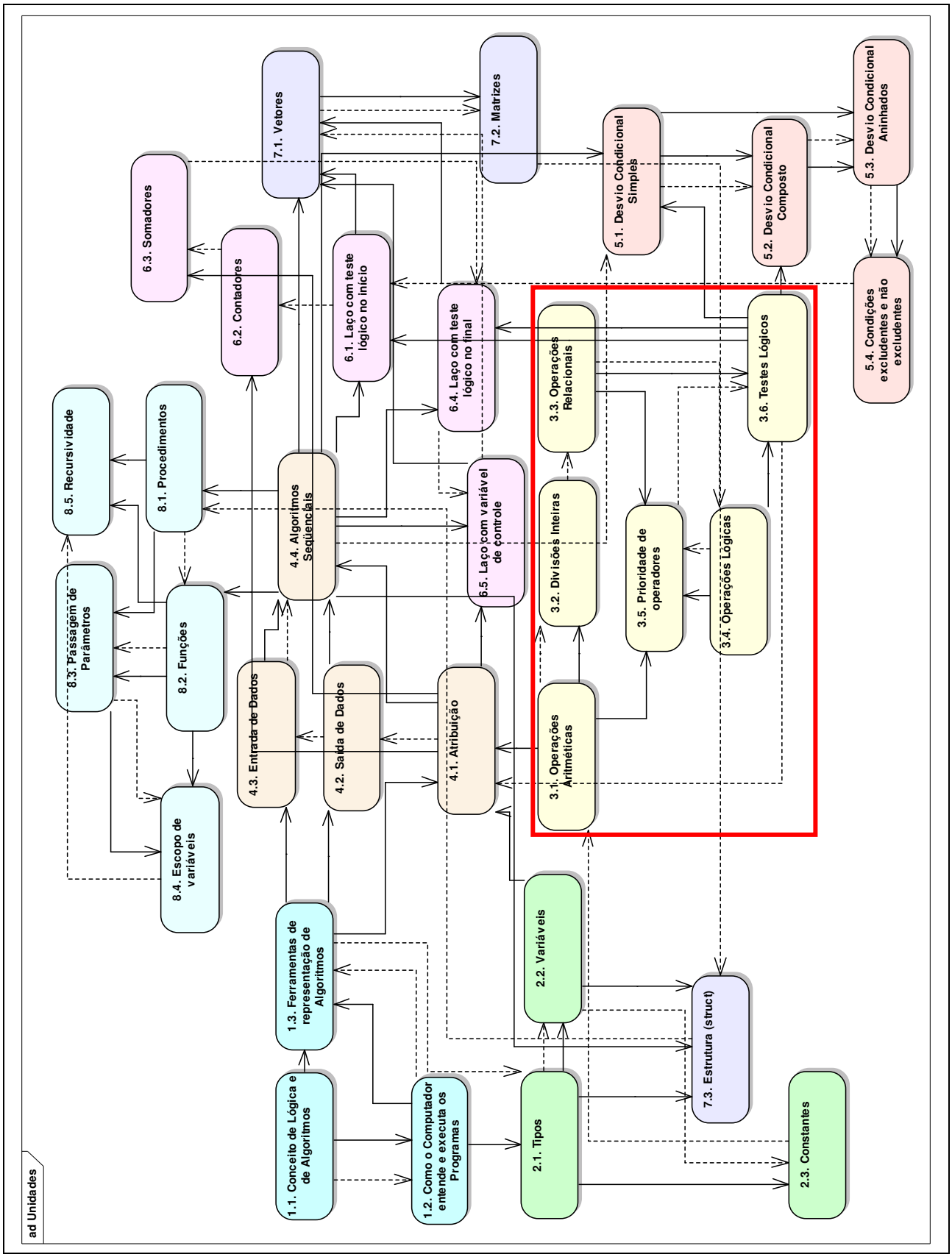

Figura 44. Mapa conceitual de pré-requisitos entre conceitos

Fonte: Silva (2005)

Dentre os temas abordados em uma disciplina de Algoritmos, selecionou-se a unidade de Operadores, destacada na Figura 44. Para cada conceito, foi desenvolvido um objeto composto por um hipertexto seguido por um exercício para verificação do conteúdo aprendido. Conforme o resultado obtido pela utilização do objeto, simulou-se situações de encaminhamento do aluno a outro objeto.

A construção de um objeto para cada conceito deve-se ao fato de manter a menor e mais adequada granularidade de conteúdo, embora o editor possibilite o desenvolvimento de cursos maiores, incluindo a possibilidade de seqüenciamento de materiais internos (*assets*). Um exemplo de objeto desenvolvido é ilustrado na Figura 45.

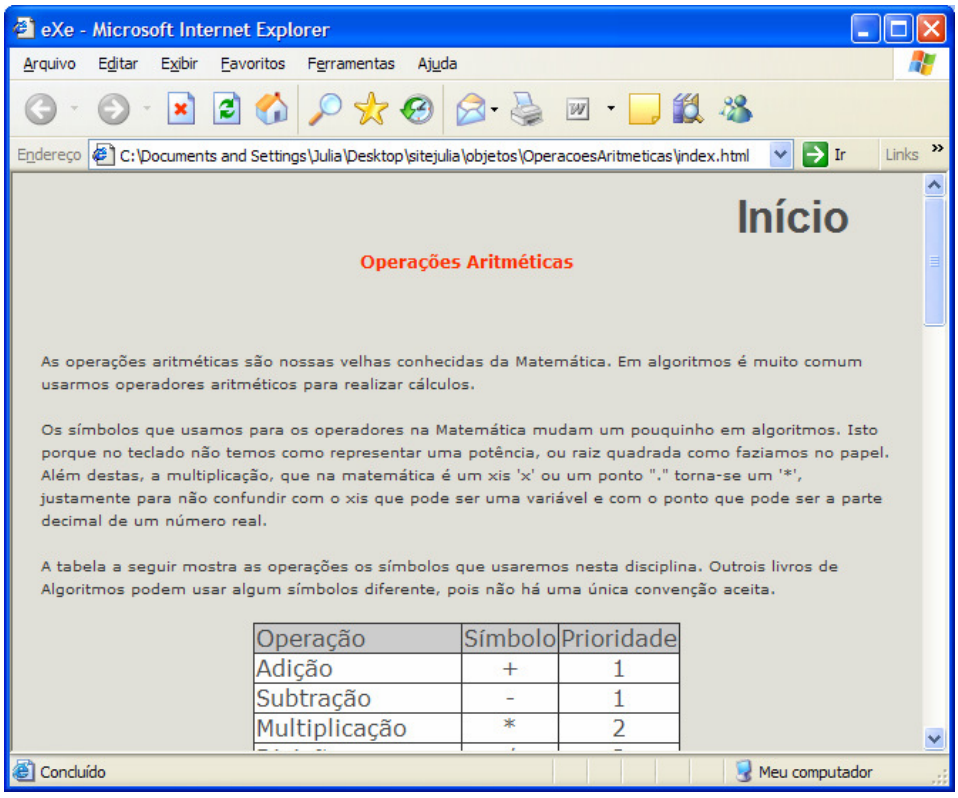

Figura 45. Exemplo de objeto de aprendizagem

O pacote gerado para cada conteúdo é composto por arquivos relacionados à exibição do material instrucional (HTML e imagens), especificação dos metadados (XML) e biblioteca de funções (JavaScript) para o funcionamento e comunicação do conteúdo. Já o LMS deve disponibilizar uma API para realizar a interação com o objeto. Para que o SMA conheça o objeto, ele carrega as informações do arquivo de metadados especificados no padrão SCORM.

Para a execução dos objetos e sua comunicação com o SMA houve a necessidade de adaptar as bibliotecas de funções, adicionando e removendo operações. Os métodos `doLMSInitialize` (internamente representado por `LMSInitialize`) e `doLMSFinalize` (internamente representado por `LMSFinish`) foram modificados de modo a estabelecer e interromper a comunicação com o SMA.

6.3 CONSIDERAÇÕES SOBRE O DESENVOLVIMENTO

A partir do desenvolvimento, pode-se verificar a aplicabilidade do *framework*, bem como testar de fato seu funcionamento. Para a realização dos testes, houve a necessidade de se buscar uma API já implementada, já que os objetos gerados no modelo SCORM dependiam dela. Como o trabalho não propunha o desenvolvimento do LMS, nem de uma API SCORM, tentou-se buscar implementações já existentes.

Após exaustivas buscas por API implementadas, verificou-se que embora a implementação seja bem discutida e incentivada pela ADL (2005), isso não ocorre com tanta freqüência. Foi notada a existência de LMS que implementam o módulo SCORM, porém a sua maioria possui o código fechado, exceto o ambiente Moodle (2007). O Moodle, implementado na linguagem PHP, permite que sejam inseridos objetos de aprendizagem SCORM dentro de seus cursos.

Ao estudar seu código-fonte, houve dificuldade na sua compreensão devido a uma mistura entre suas camadas de desenvolvimento. Sua API é implementada em PHP com JavaScript. Nesse sentido, foi necessário um estudo para então ocorrer a extração do código que seria realmente últil para os testes. Ao final, obteve-se apenas os arquivos JavaScript necessários.

6.4 RESULTADOS OBTIDOS

Devido à utilização de comunicação através de chamada remota de processos, a implementação realizada neste trabalho pode ser aplicada independentemente da linguagem de programação utilizada pelo LMS. Isto se deve ao fato da independência proporcionada através da utilização de comunicação via RMI, ou seja, as regras de negócio e estrutura do ambiente não são afetadas. Deve-se apenas conhecer a implementação do adaptador de API do LMS e adicionar a este, as chamadas aos métodos da classe `ClientApplet`.

Devido a utilização de nomes similares aos utilizados pela API, acredita-se que este procedimento não acarrete em um longo período de adaptação do sistema. Ainda, outras implementações já existentes na API, podem ser mantidas.

Ressalta-se que estas considerações são aplicáveis a APIs desenvolvidas em JavaScript. No caso de utilização da linguagem Java, podem ser utilizadas instâncias do agente `APISCORMAgent`, sem a necessidade de conexão RMI. Outra possibilidade seria manter a conexão RMI, porém alterando a natureza da classe `ClientApplet` para uma classe Java pura ou componente *desktop* (ex: biblioteca *Swing*). No caso de uso de recursos web, deve-se manter a conexão via applet, conforme considerações descritas no início deste capítulo.

Sobre o desenvolvimento dos objetos, não foram realizados testes quanto à continuidade de seu funcionamento em um LMS após as adaptações realizadas nas bibliotecas, já que este não é o escopo da atual pesquisa.

Ao realizar uma comparação sobre os resultados obtidos neste trabalho em relação a pesquisa iniciada por Gomes (2005), percebe-se que os objetos inteligentes de aprendizagem definidos originalmente são possíveis apenas se desenvolvidos por técnicos que conheçam a arquitetura interna do *framework*. Logo, as informações sobre as entidades relacionadas são de fácil acesso ao agente, podendo se adequar às necessidades de aprendizagem do aluno.

Entretanto, ao desenvolver os objetos sob uma especificação, a perspectiva do trabalho é alterada. O objetivo aqui delineado é de permitir que a aplicação e atuação dos agentes seja natural, fazendo com que qualquer conteudista possa utilizar e reutilizar seus objetos, indo ao encontro de proposta do padrão SCORM.

Em geral, conclui-se que os trabalhos desenvolvidos anteriormente pelo grupo de pesquisa, prima pela discussão e definição do que são os objetos inteligentes de aprendizagem, enquanto a presente pesquisa visa adequar o conceito na prática, de forma a verificar sua aplicabilidade. Tanto que questões gerenciais, tais como: “Como os objetos serão inseridos dinamicamente no SMA? Como os alunos serão representados no SMA? O que fazer quando um aluno abandona o LMS?” não são discutidos. Um quadro comparativo relacionando os resultados apresentados pelas pesquisas anteriores e o presente trabalho é apresentado na Tabela 6.

Tabela 6. Quadro comparativo dos resultados obtidos por trabalhos anteriores e a presente dissertação

Critérios	**Trabalhos anteriores**	**Dissertação atual**
Plataforma de SMA	FIPA-OS	JADE
Metodologia de Engenharia de Software	Nenhuma	MaSE
Padrões de Metadados	Nenhuma	SCORM
Diálogos estabelecidos	• get-metadata • get-learner • search-ilo • get-learner-lms • modify-status • put-learner • ... outros - *framework*	• added-container • born-agent • dead-agent • search • register • search-ilo • ... outros - *framework*

7 CONSIDERAÇÕES FINAIS

A abordagem de SMA reflete em um paradigma de desenvolvimento de sistemas computacionais, pois estes são orientados a troca de mensagens entre agentes. Desta forma, faz-se necessária a compreensão da forma de comunicação e o funcionamento da estrutura de mensagem e envelope.

O uso de protocolos padronizados permite que sistemas desenvolvidos independentemente possam conversar, sem que seja preciso conhecer sua estrutura interna (seja a lógica de programação ou *framework* de desenvolvimento) tampouco portas e protocolos de redes. Além disto, os protocolos facilitam a formalização do processo de troca de mensagens, por exemplo, solicitação → aceite / rejeição → execução → resposta.

Sob o ponto de vista computacional, o desenvolvimento de tais agentes através de um framework é facilitado devido a infra-estrutura já disponibilizada, permitindo que os esforços sejam concentrados no problema a ser resolvido. Atualmente o framework JADE encontra-se entre os mais consolidados, tendo projetos desenvolvidos em diversas áreas do conhecimento.

Ainda, possui uma comunidade de pesquisa ativa, colaborando na solução de dúvidas e ampliando as possibilidades de uso do *framework*. Durante esta pesquisa, foram encontradas certas dificuldades, em especial na execução de agentes através de Applets Java de forma simples e transparente. Através das listas de discussão foram debatidas tais dúvidas, além de encontrar desenvolvedores com problemas semelhantes.

Tal como o desenvolvimento de sistemas de informação tradicionais, a especificação de um SMA exige um tempo de análise expressivo, onde se deve identificar as entidades e a comunicação entre elas. Neste sentido, cabe utilizar metodologias de Engenharia de Software específica para SMA, como a MaSE. Sua utilização auxiliou na compreensão do problema e de como o mesmo poderia ser solucionado, evitando o desperdício de tempo de implementação (e re-implementação).

Os Objetos Inteligentes de Aprendizagem são entidades capazes de promover experiências de aprendizagem através de ambientes de aprendizagem, utilizando como base tecnológica o conceito de objetos de aprendizagem e sistemas multi-agentes. Isto permite a interação com outros agentes, buscando conhecimento sobre o ambiente, alunos e demais objetos que possam auxiliá-lo no ensino de um domínio.

A sociedade proposta por Silveira *et al.* (2005), foi modelada utilizando uma metodologia própria para sistemas multi-agentes, baseada em conceitos da Engenharia de Software. Esta metodologia facilita o entendimento e possibilita focar o real objetivo do sistema que se deseja construir. Devido à interligação entre os modelos, é possível estabelecer vínculos em todas as etapas, garantindo que todos os objetivos identificados sejam atendidos pelos agentes e diálogos entre eles.

A modelagem de um sistema já proposto permitiu identificar pontos que devem ser modificados ou ampliados. Foi necessária a redefinição dos agentes, e seus respectivos papéis. Por exemplo, anteriormente cabia ao agente de LMS solicitar a experiência de aprendizagem, o qual foi substituído pela ação de lms-initialize disparada pelo agente de APIAdapter. Outro agente modelado é o LearnerAgent, responsável pelas atividades de propriedade do aluno.

Ainda sobre a modelagem, a comunicação entre agentes foi revista, e propostos novos diálogos, a destacar aqueles que envolve a APIAdapter. Além disto, percebeu-se que os diálogos abordados utilizam apenas um processo definido de fases: requisição, confirmação, resposta ou rejeição. Cada um dos diálogos foram estudados quanto à sua forma de execução e definiu-se um protocolo de comunicação FIPA.

Acredita-se que dentre os benefícios em utilizar sistemas multi-agentes aliado ao paradigma de objetos de aprendizagem, refere-se a possibilidade de gerar seqüênciamentos dinâmicos. Isto é, conforme a interação do aluno com o objeto, que ele seja capaz de identificar o modelo do aluno e direcioná-lo a um conteúdo que possa ampliar seus conhecimentos. Entretanto, sabe-se que esta tarefa não é considerada trivial, nem solucionável em um único trabalho.

Sendo assim, o presente trabalho buscou formalizar a integração dos objetos a um SMA, de forma a promover um framework consistente. A partir de então, espera-se que seja possível a inclusão de tomadas de decisão sobre como e quando o seqüênciamento deve ocorrer.

7.1 TRABALHOS FUTUROS

A nova edição do padrão SCORM (ADL, 2006) incluiu especificações sobre seqüenciamento e navegação entre os conteúdos. A arquitetura aqui proposta não estabelece detalhes sobre sua aplicabilidade a este novo contexto. Portanto, um trabalho futuro pode ser desenvolvido neste sentido, a fim de adequar o SMA desenvolvido conforme as essas especificações.

Aliado a este, sugere-se um estudo sobre a seleção dos conteúdos de aprendizagem que o aluno deve ser orientado, seja aplicando uma técnica de inteligência artifical ou negociação entre agentes, por se tratar de um *framework* aplicado em um SMA.

REFERÊNCIAS

ADL. Sharable Content Object Reference Model: The SCORM Overview, Version 1.2, out. 2001. Disponível em: <http://xml.coverpages.org/SCORM-12-Overview.pdf>. Acesso em: 10 jan. 2007.

ADL. Advanced Distributed Learning Web Site. Disponível em: <http://www.adlnet.org>. Acesso em: 05 jul. 2006.

BALDONI, M.; BAROGLIO, C.; PATTI, V.; TORASSO, L. Reasoning about learning object metadata for adapting SCORM courseware. In: INTERNATIONAL WORKSHOP ON ENGINEERING THE ADAPTATIVE WEB (EAW'04), 2004, Eindhoven, Netherland. **Anais eletrônicos**... Eindhoven, 2004. Disponível em: <http://rewerse.net/publications/download/REWERSE-RP-2004-35.pdf>. Acesso em: 14 jan. 2007.

BLACKBOARD. Blackboard: Educate, Innovate, Everywhere. Disponível em: <http://www.blackboard.com/us/index.Bb>. Acesso em: 28 jun. 2007.

BRADSHAW, J. M. **An introduction to software agents**. Massachusetts: MIT Press, 1997.

BREITMAN, Karin. **Web semântica**: a internet do futuro. Rio de Janeiro: LTC, 2005.

CANCORE. CanCore Guidelines Version 2.0: Introduction. Cancore, 2003. Disponível em: <http://www.cancore.ca/guidelines/CanCore_Guidelines_Introduction_2.0.pdf>. Acesso em: 24 jul. 2007.

CAREO. Campus Alberta Repository of Educational Objects. Disponível em: <http://www.ucalgary.ca/commons/careo/CAREOrepo.htm>. Acesso em: 24 nov. 2007.

CESTA. Coletânea de Entidades Superiores ao uso de Tecnologia na Aprendizagem. Disponível em: < http://www.cinted.ufrgs.br/CESTA>. Acesso em: 24 nov. 2007.

CLAROLINE. Claroline.Net. Disponível em: <http://www.claroline.net>. Acesso em: 28 jun. 2007.

CLICK2LEARN. Disponível em: <http://academiaelearning.com/contenido/scorm/cooking/i_cookbook.htm>. Acesso em: 20 abr. 2007.

DeLOACH, S. A.; WOOD, M. Developing Multiagent Systems with agentTool. In: LECTURE NOTE INT ARTIFICIAL INTELLIGENCE, 2001, Verlag, Berlin. **Anais Eletrônicos**… Verlag. Berling, 2001. Disponível em: < http://www.springerlink.com/index/0pc24utnubkm51u1.pdf>. Acesso em: 26 ago. 2006.

DeMARCHI, Ana Carolina Bertoletti; COSTA, Antônio Carlos da Rocha. Uma proposta de padrão de metadados para objetos de aprendizagem de museus de ciência e tecnologia. **Revista de Novas Tecnologias na Educação**, v. 2, n. 1, mar. 2004. Disponível em: <http://www.cinted.ufrgs.br/renote/mar2004/artigos/02-umapropostadepadrao.pdf>. Acesso em: 19 set. 2006

DEMAZEU, Y.; MÜLLER, J. Decentralized Artificial Intelligence. In: EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTI-AGENT WORLD, 1, 1989, Cambridge. **Anais Eletrônicos**… North-Holland: Elselvier Science Publishers, 1990. p.3-13.

DOKEOS. Dokeos. Disponível em: <http://www.dokeos.com>. Acessado em: 30 jun. 2007.

DOWNES, S. Learning objects: resources for distance education worldwide. **International Review of Research in Open and Distance Learning**, v. 2, n. 1, 2001.

DUBLIN CORE. Dublin Core, 2007. Disponível em: < http://dublincore.org/documents/usageguide/elements.shtml>. Acesso em: 24 jul. 2007.

DUVAL, E. Learning Technology Standardization: Making Sense of it All. **International Journal Published by ComSIS Consortium**, v. 1, n. 1, fev. 2004. ISSN: 1820-0214. Disponível em: <http://www.comsis.fon.bg.ac.yu/ComSIS/Volume01/InvitedPapers/ErikDuval.pdf>. Acesso em: 14 out. 2007.

EDUSOURCE. General Information. EduSource, 2004. Disponível em: <http://edusource.licef.teluq.uquebec. ca/ese/en/project_goal.htm>. Acesso em: 11 set. 2006.

EXE. eLearning XHTML Editor. EXE, 2007. Disponível em: <http://exelearning.org>. Acesso em: 29 set. 2007.

FABRE, Marie-Christine J. M.; TAMUSIUNAS, F. R.; TAROUCO, L. Reusabilidade de objetos educacionais. **Revista Novas Tecnologias em Educação**, v.1, n.1, 2003. Disponível em: <http://www.cinted.ufrgs.br/renote/fev2003/artigos/marie_reusabilidade.pdf>. Acesso em: 28 jul. 2006.

FERBER, J.; GUTKNECH; O. Aalaadin: a meta-model for the analysis and design of organizations in multi-agent systems. In: INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS, jul. 1998, Paris, França. **Anais Eletrônicos**... Paris, 1998. Disponível em: <http://citeseer.ist.psu.edu/ferber97aalaadin.html>. Acesso em: 13 out. 2007.

FIPA. The foundation for Intelligent Physical Agents: Specifications. FIPA, 2002. Disponível em: <http://www.fipa.org>. Acesso em: 7 jul. 2005.

FIPA SC00061. FIPA-ACL Message Structure Specification. FIPA, 2002. Disponível em: <http://www.fipa.org/specs/fipa00061/SC00061G.pdf>. Acesso em: 14 abr. 2007.

FIPA-OS. FIPA-OS Agent Toolkit. Disponível em: <http://sourceforge.net/projects/fipa-os/> Acesso em: 13 out. 2007.

FREITAS, Frederico Luiz Gonçalves. Ontologia e a Web Semântica. In: JORNADA DE MINI-CURSOS DE INTELIGÊNCIA ARTIFICIAL, n. 3, ago. 2003, Campinas. **Anais do Congresso da Sociedade Brasileira de Computação**, n. 23, ago. 2003, Campinas.

FRIGO, L. B.; POZZEBON, E.; BITTENCOURT, G. O Papel dos Agentes Inteligentes nos Sistemas Tutores Inteligentes. In: WORLD CONGRESS ON ENGINEERING AND TECHNOLOGY EDUCATION, 2004, São Paulo. **Anais Eletrônicos**... São Paulo, 2004. Disponível em: <http://www.inf.ufsc.br/~l3c/artigos/frigo04a.pdf>. Acesso em: 10 abr. 2007.

GASEVIC, D.; DEVEDZIC, J. J.; BOSKOVIC, M. Ontologies for Reusing Learning Object Content. In: IEEE INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES, 2005, Surrey, UK. **Anais Eletrônicos**... Surrey, 2005. Disponível em: <http://www.win.tue.nl/SW-EL/2005/swel05-icalt05/final/W3-1.pdf>. Acesso em: 24 mar. 2007.

GENNARI, J. H.; MUSEN, M. A.; FERGENSON, R. W.; GROSSO, W. E.; CRUBÉZI, M.; ERIKSSON, H.; NOY, N. F.; SAMSON, T. W. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. **International Journal of Human-Computer Studies**, 2003. Disponível em: < http://smi.stanford.edu/smi-web/reports/SMI-2002-0943.pdf>. Acesso em: 7 fev. 2007.

GOMES, E. R.; SILVEIRA, R. A.; VICCARI, R. Utilização de agentes FIPA em ambientes para Ensino a Distância. In: CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA, nov. 2003, Montevideu, Uruguai. **Anais Eletrônicos**... Montevideu, 2003. Disponível em: <http://ifm.ufpel.edu.br/iate/downloads/clei2003.pdf>. Acesso em: 13 out. 2007.

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

GLUZ, João Carlos; VICCARI, Rosa Maria. Linguagens de Comunicação entre Agentes: Fundamentos Padrões e Perspectivas. In: JORNADA DE MINI-CURSOS DE INTELIGÊNCIA ARTIFICIAL, n. 3, ago. 2003, Campinas. **Anais do Congresso da Sociedade Brasileira de Computação**, n. 23, ago. 2003, Campinas.

HODGINS, H. W. The Future of Learning Objects. In: CONFERENCE ON E-TECHNOLOGIES IN ENGINEERING EDUCATION, aug. 2002, Davos, Switzerland. **Anais Eletrônicos**… Davos, 2002.

HÜBNER, J. F. Simple Agent Communication Infraestructure. In: CONGRESSO BRASILEIRO DE COMPUTAÇÃO, n. 3, 2003, Itajaí. **Anais**... Itajaí, 2003.

IEEE. IEEE Learning Technology Standards Committee: Specifications, 2004. Disponível em: <http://ltsc.ieee.org>. Acesso em: 7 jul. 2006.

IEEE LOM. Draft Standard for Learning Object Metadata: IEEE 1484.12.1-2002, 15 jul. 2002. Disponível em: <http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf>. Acesso em: 07 jan. 2007.

ILUMINA. About iLumina. Disponível em: <http://www.ilumina-dlib.org/contactPage.asp>. Acesso em: 24 nov. 2007.

JADE. Java Agent Development Framework. Disponível em: <http://jade.tilab.com>. Acesso em: 25 jun. 2007.

JDOM. Java representation of an XML document. Disponível em: <http://www.jdom.org>. Acesso em: 13 ago. 2006.

JENNINGS, N. R. Coordination Techniques for Distributed Artificial Intelligence. In: O'HARE, G.M.P.; Jennings, N.R. (Eds.). **Foundations of distributed artificial intelligence**. New York: John Wiley & Sons, 1996, pp.187-210.

JEON, H. JATLite Introductory FAQ, 2000. Disponível em: <http://www-cdr.stanford.edu/ProcessLink/papers/JATL.html#WhatIsJATLite>. Acesso em: 13 out. 2007.

LANGE, D. B. Java Aglet Application Programming Interface (J-AAPI) White Paper - Draft 2, 1997. Disponível em: <http://www.trl.ibm.com/aglets/JAAPI-whitepaper.htm>. Acesso em: 13 out. 2007.

MERLOT. Have You Discovered MERLOT?. Disponível em <http://taste.merlot.org/MERLOTBrochurev806.pdf>. Acesso em: 09 set. 2006.

MOHAN, P.; BROOKS, C. Engineering a Future for Web-based Learning Objects. In: INTERNATIONAL CONFERENCE ON WEB ENGINEERING, 2003, Asturias, Espanha. **Anais**… Espanha, 2003.

MOODLE. Modular Oriented-Object Dynamic Learning Environment. Disponível em: <http://www.moodle.org>. Acesso em: 28 jun. 2007.

NWANA, H. S.; NDUMU, D. T.; LEE, L. C. ZEUS: An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems. In: PRATICAL APPLICATION OF INTELLIGENT AGENTS AND MULTI-AGENT SYSTEMS, 1998, London, UK, pp. 377-391. **Anais Eletrônicos**… London, 1998. Disponível em: <http://citeseer.ist.psu.edu/nwana98zeus.html>. Acesso em: 19 abr. 2007.

NOY, N. F.; FERGENSON, R. W.; MUSEN, M. A. The knowledge model of Protégé-2000: combining interoperability and flexibility. In: EUROPEAN WORKSHOP ON KNOWLEGDE ACQUISITION, MODELING AND MANAGEMENT, n. 12, 2000. **Anais**… 2000. ISBN: 3-540-41119-4.

ODELL, J.; PARUNAK, H. V. D.; BAUER, B. Extending UML for Agent, 2001. Disponível em: <http://www.jamesodell.com/ExtendingUML.pdf>. Acesso em: 26 fev. 2007.

OLIVEIRA, F. M.; Inteligência Artificial Distribuída. In: ESCOLA REGIONAL DE INFORMÁTICA, n. 4, Canoas, 1996. **Anais**... Canoas, 1996.

QU, Changtao; NEJDL, Wolfgang. Towards Interoperability and Reusability of Learning Resource: a SCORM-conformant Courseware for Computer Science Education. **Technical report**, Learning Lab Lower Saxony, University of Hannover, Germany, 2001.

RIVED. Conheça o Projeto. Disponível em: <http://www.rived.mec.gov.br/site_objeto_lis.php>. Acesso em: 04 nov. 2007.

RUSSEL, Stuart; NORVIG, Peter. **Inteligência Artificial**. Ed. Elvesier, 2ª ed, 2004. ISBN: 8535211772.

SANTOS, Pedro Henrique Bandeira dos. **Um Repositório de Objetos de Aprendizagem para os Cursos de Ciência da Computação e Tecnologia de Sistemas para Internet**. 2007. Trabalho de

Conclusão de Curso(Bacharelado em Ciência da Computação)-Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí.

SILVA, Eli Lopes da. **Uma Experiência de Uso de Objetos de Aprendizagem na Educação Presencial**: Ação-Pesquisa num Curso de Sistemas de Informação. 2006. Dissertação(Mestrado em Educação)-Programa de Pós-Graduação em Educação, Universidade Católica de Minas Gerais, Belo Horizonte.

SILVA, Carla; PINTO, Rosa; CASTRO, Jaelson; TEDESCO, Patrícia. Requirements for Multi-Agent Systems. WORKSHOP EM ENGENHARIA DE REQUISITOS, 2003, Piracicaba. **Anais**... Piracicaba, 2003. ISBN: 8587926071.

SILVA, J. C. T. da. **Um modelo para avaliação de aprendizagem no uso de ferramentas síncronas em ensino mediado pela Web**. Tese (Doutorado em Informática) - Programa de Pós-Graduação em Informática, PUC-Rio, abr. 2004. Disponível em: <http://www2.dbd.puc-rio.br/pergamum/tesesabertas/9816122_04_cap_04.pdf>. Acesso em: 29 set. 2007.

SILVA, Júlia Marques Carvalho da. Análise da Influência de Parâmetros Afetivos em um Sistema Computacional de Apoio a Aprendizagem de Algoritmos. 2005. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí.

SILVEIRA, R. A.; GOMES, E. R; VICCARI, R. M. Inteligent Learning Objects: An Agent-Based Approach of Learning Objects. In: Weert, Tom Van, Tatnall, Arthur (Eds.) **Information and Communication Technologies and Real-Life Learning**. Boston Springer, 1103 – 110, 2005.

SILVEIRA, R. A.; GOMES, E. R; VICCARI, R. M. Using Intelligent Learning Objects in Adaptative Educational Portals. In: Tatnall, Arthur (Ed.) **Encyclopaedia of Portal Technology and Applications Idea Group Publishing**, Hershey, PA , EUA, 2007.

SILVEIRA, R. A.; GOMES, Eduardo Rodrigues; VICCARI, Rosa. Improving Interoperability Among Learning Objects Using FIPA Agent Communication Framework. In: IFIP WORLD COMPUTER CONFERENCE, Santiago. **Professional Practice in Artificial Intelligence**. Berlin: Springer, 2006. p. 51-60.

SOSTERIC, M., HESMEIER, 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**, 2002. ISSN: 1492-3831

TELEDUC. TelEduc: Ambiente de Ensino a Distância. Disponível em: <http://teleduc.nied.unicamp.br/pagina/>. Acesso em: 30 jun. 2007.

TVEIT, Amund. A survey of Agent-Oriented Software Engineering. In: NTNU COMPUTER SCIENCE GRADUATE STUDENT CONFERENCE, Norwegian University of Science and Technology, 2001, Norwegian. **Anais**… Norwegian, 2001.

WEBAULA. WebAula: Educação Sem Fronteiras. Disponível em: <http://www.webaula.com.br/>. Acesso em: 03 aug. 2007.

WEBENSINO. Web Ensino. Disponível em: <http://www.webensino.com.br/>. Acesso em: 03 aug. 2007.

WILGES, Beatriz ; Mateus, Gustavo P. ; SIVEIRA, Ricardo Azambuja ; NASSAR, S. M. . An Animated Pedagogical Agent as a Learning Management System manipulating Intelligent Learning Objects. In: The 7th IEEE International Conference on Advanced Learning Technologies - ICALT, 2007, Niigata. **Anais Eletrônicos**… IEEE Computer Society - Digital Library - Seventh IEEE International Conference on Advanced Learning Technologies, 2007. p. 186-188.

WOOLDRIDGE, Michel. **An Introduction to MultiAgent Systems**. John Wiley & Sons, Inc.: England, 2002.

YACEF, Kalina. Intelligent teaching assistant system. In: INTERNATIONAL CONFERENCE ON COMPUTERS IN EDUCATION, 2002, v. 1, pp 136 – 140. **Anais Eletrônicos**… 2002.

ZINI, F.; STERLING, L. Designing Ontologies for Agents. In: APPIA-GULP-PRODE 1999: Joint Conference on Declarative Programming. pp. 29-42. **Anais Eletrônicos**... Disponível em: <http://www.isys.dia.fi.upm.es/~phernan/AgentesInteligentes/referencias/zini99.pdf>. Acessado em: 14 out. 2007.

PRÊMIO E ARTIGOS ACEITOS PARA PUBLICAÇÃO

Prêmio

2º Melhor Artigo Completo do XVIII Simpósio Brasileiro de Informática na Educação.

Artigo Publicado em Periódico (completo)

SILVA, Júlia Marques Carvalho da, SILVEIRA, Ricardo Azambuja. Modelagem de Objetos Inteligentes de Aprendizagem utilizando a metodologia MaSE. RENOTE. Revista Novas Tecnologias na Educação. , v.1, p.1 - , 2006.

Artigos Publicados em Anais de Congressos ou Periódicos (completo)

SILVA, Júlia Marques Carvalho da, SILVEIRA, Ricardo Azambuja. The Development of Intelligent Learning Objects with an Ontology based on SCORM Standard In: Intelligent Systems Design and Applications, 2007, Rio de Janeiro. Intelligent Systems Design and Applications. , 2007

SILVA, Júlia Marques Carvalho da, BAVARESCO, Natanael, SILVEIRA, Ricardo Azambuja. Uma Modelagem Baseada em Sistemas Multiagentes para a Criação de Objetos de Aprendizagem Dinâmicos e Reutilizáveis In: Conferência Latinoamericana de Informática, 2007, San José, Costa Rica. Conferência Latinoamericana de Informática. , 2007. *Qualis Nacional nível B*

SILVA, Júlia Marques Carvalho da, BAVARESCO, Natanael., SILVEIRA, Ricardo Azambuja. Proposta de um Sistema Multi-agentes para a aplicação de Objetos Inteligentes de Aprendizagem seguindo o padrão SCORM In: XVIII Simpósio Brasileiro de Informática na Educação, 2007, São Paulo. XVIII Simpósio Brasileiro de Informática na Educação. , 2007. *Qualis Nacional nível B*

Artigo Publicado em Anais de Congresso (resumo expandido)

SILVA, Júlia Marques Carvalho da, SILVEIRA, Ricardo Azambuja. Modeling and Building Intelligent Learning Environments through Intelligent Learning Objects In: 7th IEEE International Conference on Advanced Learning Technologies, 2007, Niigata, Japão. 7th IEEE International Conference on Advanced Learning Technologies. , 2007. *Qualis Internacional nível A*

Artigo Aceito para Publicação (completo)

SILVA, Júlia Marques Carvalho da, SILVEIRA, Ricardo Azambuja. Designing a multi-agent systems to improve reusability of Learning Objects In: Intelligent Systems and Agents, 2007, Lisboa, Portugal. Intelligent Systems and Agents. , 2007.

SILVA, Júlia Marques Carvalho da, SILVEIRA, Ricardo Azambuja. Intelligent Learning Objects: Architecture, modeling and ontology In: WWW/Internet, 2007, Vila Real, Portugal. WWW/Internet. , 2007. *Qualis Internacional nível C*

APÊNDICE

A DIAGRAMA ENTIDADE-RELACIONAMENTO

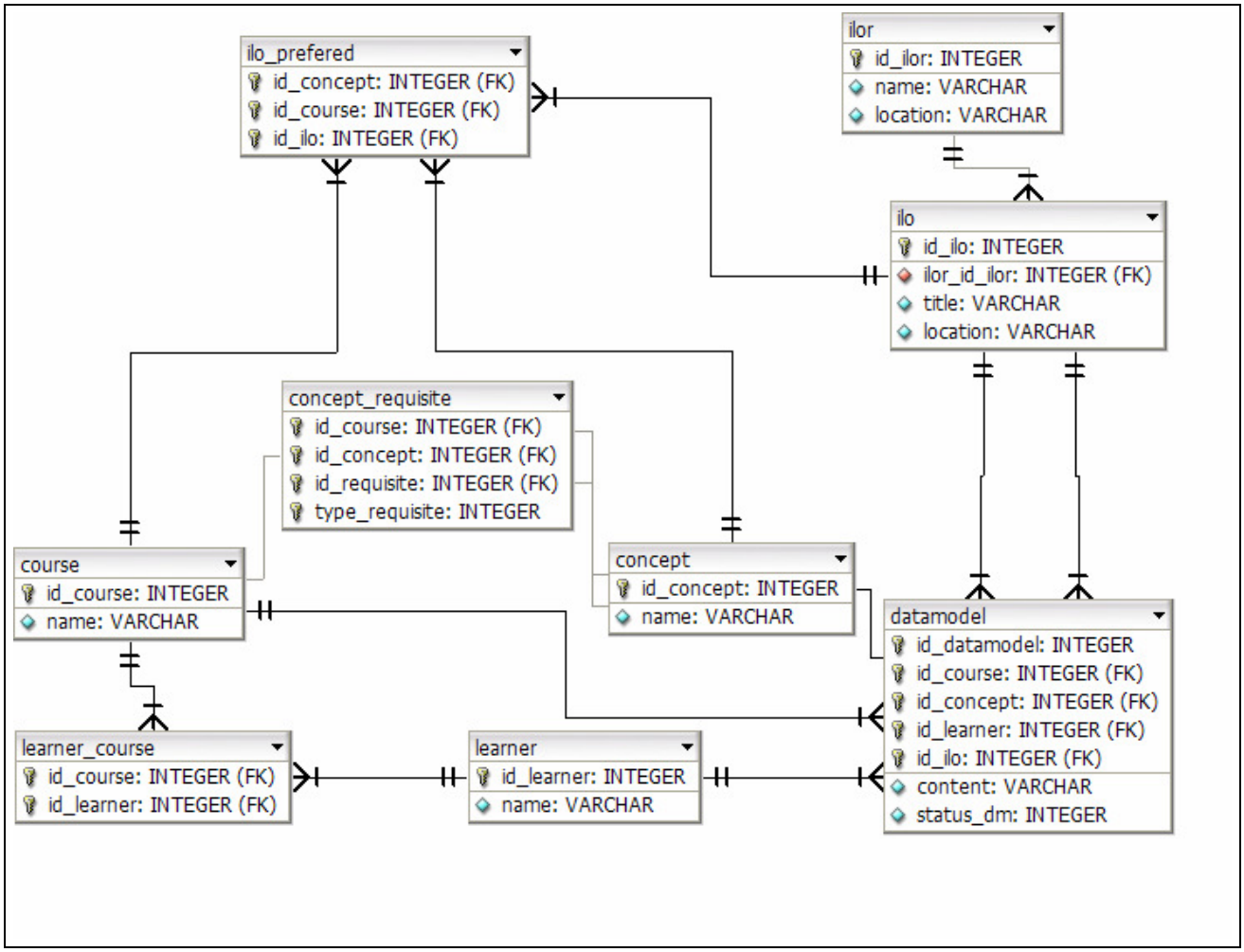

Figura 46. Diagrama de entidade-relacionamento do *framework*