UNIVERSIDADE ESTADUAL PAULISTA

"Júlio de Mesquita Filho"

Pós-Graduação em Ciência da Computação

Luís Henrique De Rossi

Uma Arquitetura Multiagentes para Sistema Educacional Baseado na *Web* Semântica

Rio Claro / SP

2011

Luís Henrique De Rossi

Uma Arquitetura Multiagentes para Sistema Educacional Baseado na *Web* Semântica

Dissertação apresentada para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, Área de Concentração - Sistemas Inteligentes, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador: Prof. Dr. Ivan Rizzo Guilherme

Co-Orientadora: Profa. Dra. Simone das Graças Domingues Prado

Rio Claro / SP

2011

Luís Henrique De Rossi

Uma Arquitetura Multiagentes para Sistema Educacional Baseado na *Web* Semântica

Dissertação apresentada para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, Área de Concentração – Sistemas Inteligentes, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

MEMBROS DA BANCA EXAMINADORA

Presidente e Orientador: Prof. Dr. Ivan Rizzo Guilherme
UNESP – Rio Claro

Membro Titular: Profa. Dra. Hilda Carvalho de Oliveira
UNESP – Rio Claro

Membro Titular: Profa. Dra. Flávia Linhalis Arantes
UNICAMP

Rio Claro / SP
2011

AGRADECIMENTOS

Agradeço a Deus pela dádiva que é dom da vida.

Aos meus pais, que sempre me forneceram tudo que tinham ao seu alcance para que eu atingisse meus sonhos e objetivos. Também por todo esforço e incentivo dados durante todos esses anos, estando sempre ao meu lado.

A minha esposa Daniela e meus filhos Brenda e Leonardo, os quais entenderam minha ausência durante esses anos, na busca da realização do meu sonho de conseguir a pós-graduação. Obrigado pelo incentivo dado para meu crescimento. Amo vocês.

A minha irmã Luciane De Rossi, a quem sempre recorro para me amparar. Sinto que, às vezes, parece esquecer-se de si para me socorrer. Obrigado pela correção desse trabalho.

Aos professores Orlando de Andrade Figueiredo e Beatris Cristina Possato, que me incentivaram a continuar meus estudos, me fazendo acreditar que entrar no mestrado era possível.

Ao meu orientador, Ivan Rizzo Guilherme, por ter me dado essa oportunidade e aceito me orientar. Por todo seu esforço em me ensinar, não só transmitindo o seu conhecimento, mas também mostrando que o estudo é um amadurecimento do ser humano.

As professoras que fizeram parte da banca de qualificação, Hilda Carvalho de Oliveira e Flávia Linhalis Arantes, obrigado pelo seu tempo e pelas suas importantes sugestões.

Ao professor Ig Ilbert Bittencourt pelas sugestões dadas ao meu trabalho.

Aos meus amigos Gabriel Abdalla, Jonas Queiroz, Roberto Gueleri e Guilherme Belíssimo, por todo apoio que deram no desenvolvimento desse trabalho. Estando sempre dispostos a me ajudar.

A todos que, direta ou indiretamente, me ajudaram e torceram pelo meu sucesso, obrigado.

RESUMO

Nas últimas duas décadas, diversos sistemas de ensino para *Web* foram desenvolvidos e amplamente utilizados. Com o surgimento da *Web* Semântica, a qual integra os conceitos da *Web* e da área de Inteligência Artificial, foi possível conceber novos sistemas de ensino, os Sistemas de Ensino Baseados na *Web* Semântica (*Semantic Web-based Educational Systems* - SWBES).

Os SWBES são caracterizados por representar o conteúdo e o conhecimento na forma de ontologias e contar com agentes inteligentes que permitem às máquinas entenderem o conteúdo publicado na *Web*.

Neste trabalho são apresentados e discutidos os padrões arquiteturais utilizados para desenvolvimento dos sistemas baseados na *Web* Semântica, em especial os sistemas educacionais. Como resultado do estudo é proposta uma arquitetura para os SWBES.

As arquiteturas dos SWBES que utilizam a abordagem multiagente, normalmente, são concebidas de forma empírica, ou seja, apenas com a experiência do seu desenvolvedor. Neste trabalho, para a concepção do modelo multiagentes adotado na arquitetura do SWBES, foi utilizada uma metodologia de desenvolvimento de sistemas multiagentes, o *framework* Tropos. São apresentadas as fases do *framework* Tropos para o projeto de um modelo multiagentes utilizado na arquitetura de um SWBES.

Utilizando a arquitetura desenvolvida foi implementado um SWBES. Para estudo de caso dessa arquitetura, foi realizada uma implementação de um Sistema de Ensino de Rochas Carbonáticas. Para esse sistema foram definidas três ontologias: ontologia de Domínio de Rochas Carbonáticas, ontologia de Perfil e ontologia de Material de Aprendizagem. Os comportamentos dos agentes e as interações entre eles são detalhadas.

Finalmente, é apresentado um estudo de casos da utilização do sistema no ensino do curso de Rochas Carbonáticas.

Palavras-chave: agentes, ontologias, *web* semântica, educação a distância, Tropos.

ABSTRACT

In the past two decades, several Web educational systems were designed and widely used. With the appearing of the Semantic Web, which integrates Web and Artificial Intelligence area concepts, it was possible to conceive new educational concepts, the Educational Systems Based on the Semantic Web (Semantic Web-based Educational Systems - SWBES).

The SWBES represent the content and the knowledge in the form of ontology and count on intelligent agents that allow machines understand the content published on the web.

In this work, architectural standards used to develop new systems based on the Semantic Web will be presented and discussed, especially the educational ones. As a result of this study, it is proposed the architecture for the SWBES.

The architecture for the SWBES that use multi-agent approach generally is conceived empirically, that is, only with its developer experience. In this work, for the conception of the multi-agents model adopted in the architecture of the SWBES, a methodology of multi-agents systems development was used, the Tropos framework. The phases of Tropos framework for the project of a multi-agent model used in the architecture of SWBES are presented.

A SWBES was implemented by using the architecture that was developed. For the study case of this architecture, it was held an implementation of the Educational System of Carbonate Rock. For this system, three ontologies were defined: Domain Ontology of Carbonate Rocks, Profile Ontology and Learning Material Ontology. The behavior of agents and interaction among them are detailed.

Finally, a study of cases of using the educational system in Carbonate Rock is presented.

Key-words: *agents, ontologies, semantic web, e-learning, Tropos.*

LISTA DE ABREVIATURAS E SIGLAS

BDI	*Belief-Desire-Intention*
CAI	*Computer Aided Instruction*
CSS	*Cascading Style Sheets*
DAML	*DARPA Agent Markup Language*
DAO	*Data Access Object*
DF	*Directory Facilitator*
EAD	Educação a distância
HTML	*Hypertext Markup Language*
JSF	*Java Server Faces*
LMS	*Learning Management System*
LOM	*Learning Object Metadata*
OIL	*Ontology Inference Layer*
OWL	*Web Ontology Language*
PDF	*Portable Document File*
RDF	*Resource Description Framework*
SCO	*Sharable Content Objects*
SHOE	*Simple HTML Ontology Extension*
SLO	*Sharable Learning Objects*
SMA	Sistemas Multiagentes
STI	Sistemas Tutores Inteligentes
SWBES	*Semantic Web Based E-learning System*
SWS	*Semantic Web Services*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
WBES	*Web Based E-learning System*
WS	Web Semântica
XML	*Extensible Markup Language*

LISTA DE ILUSTRAÇÕES

Figura 1: Arquitetura em camadas da *Web* Semântica 11
Figura 2: Exemplo de XML..... 12
Figura 3: XML *Schema* correspondente a nota 13
Figura 4: Representação de um relacionamento Sujeito/Predicado/Objeto..... 14
Figura 5: Exemplo de um RDF que descreve uma lista de CDs de Música..... 15
Figura 6: Exemplo de RDF *Schema* para descrição de animais..... 16
Figura 7: Componentes utilizados nos diagramas do Tropos 21
Figura 8: Ligações de decomposição entre os componentes..... 22
Figura 9: Dependência entre atores..... 22
Figura 10: Arquitetura de referência para sistemas *Web* semântico..... 28
Figura 11: Modelo de Referência para SWBES 29
Figura 12: Arquitetura proposta para SWBES..... 31
Figura 13: Relacionamento entre ontologias..... 34
Figura 14: Diagrama de Atores (Requisitos Iniciais)..... 39
Figura 15: Diagrama de Metas do Ator Aluno (Requisitos Iniciais) 40
Figura 16: Diagrama de Metas do Ator Professor (Requisitos Iniciais)..... 40
Figura 17: Diagrama de metas do *Designer* Instrucional..... 41
Figura 18: Diagrama de atores - Requisitos Finais 44
Figura 19: Diagrama de metas Requisitos Finais - Ator SWBES..... 45
Figura 20: Estilo arquitetural *Joint-Venture* 47
Figura 21: Diagrama Arquitetural 50
Figura 22: Diagrama detalhado - Avaliador Individual 51
Figura 23: Diagrama detalhado - Buscador..... 52
Figura 24: Diagrama detalhado - Publicador..... 52
Figura 25: Diagrama detalhado – Tutor..... 53
Figura 26: Diagrama detalhado – Controlador 53
Figura 27: Diagrama detalhado - Avaliador Social..... 54
Figura 28: Tecnologias utilizadas na geração da interface *Web*..... 56
Figura 29: Representação da classe Diagenese da ontologia de rochas carbonáticas 60
Figura 30: Representação gráfica da ontologia de Materiais de Aprendizagem 61
Figura 31: Relacionamento da ontologia de materiais com a ontologia de domínio 62
Figura 32: Trecho do código da ontologia de perfil..... 63

Figura 33: Planos, metas, crenças e eventos referentes a atividade avaliar 65
Figura 34: Diagrama de Sequência - Agente Avaliador 66
Figura 35: Diagrama de Sequência - Agente Buscador 67
Figura 36: Diagrama de Sequência - Agente Buscador 68
Figura 37: Tópico Classificação das Rochas Carbonáticas 70
Figura 38: Código OWL para descrição de um tópico 71
Figura 39: Cadastro do tópico Classificação no Protégé 72
Figura 40: Código OWL para pergunta 73
Figura 41: Tela inicial do curso de Rochas Carbonáticas 74
Figura 42: SWBES para Rochas Carbonáticas - Material de Apoio 75
Figura 43: Exemplo da ferramenta Treinamento 76
Figura 44: Página de Treinamento - Resposta correta 77
Figura 45: Página de Treinamento - Resposta incorreta 77
Figura 46: Exemplo da ferramenta Treinamento – Exibição de Dica 78

LISTA DE TABELAS

Tabela 1: Relação dos interessados e suas intenções (Requisitos Iniciais) 37
Tabela 2: Definição dos planos e relacionamentos entre os atores (Requisitos Iniciais) 38
Tabela 3: Relação dos interessados e suas intenções (Requisitos Finais) 42
Tabela 4: Novos planos incluindo o ator sistema (Requisitos Finais) 43
Tabela 5: Catálogo de Correlação 46
Tabela 6: Especificação do papel Avaliador 48
Tabela 7: Especificação do papel Tutor 48
Tabela 8: Especificação do papel Buscador 48
Tabela 9: Especificação do papel Publicador 49

SUMÁRIO

1 Introdução 1
1.1 Motivação e escopo 2
1.2 Objetivos 3
1.3 Metodologia 3
1.4 Organização dos capítulos 4
2 Aspectos Conceituais 5
2.1 Educação a distância 5
2.1.1 A evolução do uso dos computadores na Educação 5
2.1.2 Padrões de metadados para educação baseada na *Web* 7
2.2 *Web* Semântica 7
2.2.1 Ontologias 8
2.2.1.1 Metodologias para a construção de ontologias 9
2.2.1.2 Codificação das Ontologias 11
2.2.1.3 URIs e *Namespaces* 12
2.2.1.4 XML e XML *SCHEMA* 12
2.2.1.5 RDF e RDF SCHEMA 13
2.2.1.6 OWL 16
2.3 O uso da *Web* Semântica na educação 17
2.4 Agentes 19
2.4.1 Sistemas Multiagentes 20
2.4.2 Metodologia de Desenvolvimento de Software Multiagente 20
2.4.2.1 Metodologia Tropos 21
2.4.2.1.1 Conceitos 21
2.4.2.1.2 Fases da modelagem 23
2.4.2.1.3 Fases de Desenvolvimento 24
3 Abordagens adotadas para sistemas educacionais baseados na *Web* Semântica 27
3.1 Arquitetura de referência 27

3.2 Arquitetura proposta para o SWBES 30
3.2.1 Componentes Arquiteturais 31
3.2.1.1 Sistema de ensino na *Web* 31
3.2.1.2 Modelo Multiagentes 32
3.2.1.3 Serviços de Integração 33
3.2.1.4 Interface de Dados 33
3.2.1.5 Camada de Persistência 34
3.2.1.6 Repositórios de objetos de aprendizagem 35
4 Modelagem Multiagente 36
4.1 Modelagem 36
4.1.1 Descrição do sistema 36
4.1.2 Requisitos Iniciais 37
4.1.3 Requisitos Finais 41
4.1.4 Projeto Arquitetural 45
4.1.4.1 Identificação do estilo arquitetural 46
4.1.4.2 Identificação dos papéis 47
4.1.4.3 Mapeamento dos papéis para o estilo arquitetural 49
4.1.5 Projeto Detalhado 51
5 Aspectos da Implementação do SWBES 55
5.1 Definições Tecnológicas para construção do SWBES 55
5.1.1 Sistema de Educação a Distância 55
5.1.2 Modelo Multiagente 56
5.1.3 Serviços de Integração 57
5.1.4 Interface de Dados 57
5.1.5 Camada de Persistência 58
5.2 Implementação do SWBES 59
5.2.1 Construção das Ontologias 60
5.2.1.1 Ontologia de Domínio de Rochas Carbonáticas 60
5.2.1.2 Ontologia de Materiais de Aprendizagem 61
5.2.1.3 Ontologia de Perfil 62

5.2.2 Implementação dos agentes 63
5.2.2.1 Agente Controlador 64
5.2.2.2 Agente Avaliador 64
5.2.2.3 Agente Buscador 66
5.2.2.4 Agente Publicador 67
5.3 Estudo de Caso 68
5.3.1 Construção do curso 69
5.3.1.1 Definição da ontologia de domínio 69
5.3.1.2 Construção do conteúdo do curso 69
5.3.1.3 Anotação dos Materiais de Aprendizagem 70
5.3.1.4 Definição da estrutura do curso 72
5.3.1.5 Cadastro de Perguntas 73
5.3.2 Exemplo da utilização do SWBES 74
5.3.2.1 Material de Apoio 75
5.3.2.2 Treinamento 76
6 Conclusão e Considerações Finais 79
REFERÊNCIAS 81
Apêndice A – Agente Controlador 85
Apêndice B – Agente Avaliador de Desempenho 91
Apêndice C – Agente Buscador 95
Apêndice D – Agente Publicador 99
Apêndice E – Diagrama de Sequência – Atividades dos Agentes 103
Apêndice F – Ontologia de Petrografia 104

1 Introdução

Os cursos de educação a distância existem há muito tempo e em diversos formatos. Entretanto, com o surgimento da *World Wide Web* (*Web*), os cursos *online* ganharam uma nova dimensão e têm sido alvo de diversas pesquisas na academia e no meio corporativo. Uma das grandes vantagens dos cursos na *Web* é permitir o desenvolvimento pessoal com liberdade para escolha do local e horário do estudo. Nesses cursos, os alunos têm acesso a materiais para estudo que auxiliam na realização das atividades de avaliação. As ferramentas mais comuns encontradas nesse tipo de sistema *Web* são: fóruns, questionários, bate-papo e listas de discussão.

Embora esses sistemas estejam sendo amplamente utilizados, existem limitações quanto à dificuldade de busca, integração e reuso dos conteúdos existentes. As atividades desenvolvidas por professores que utilizam os sistemas de ensino atuais são bastante custosas. O modelo da *Web* atual não permite a automatização dessas atividades.

Diferentes pesquisadores da área de Inteligência Artificial vêm desenvolvendo esforços com o objetivo de estudar as ferramentas de ensino atuais para definir formas de representar o conhecimento, identificar e automatizar os processos de ensino. Nesses estudos destacam-se as pesquisas de tutores inteligentes e sistemas baseados em agentes.

Recentemente, os conceitos da *Web* e da área de Inteligência Artificial foram integrados na proposição de uma nova *Web,* a *Web* Semântica. A proposição da *Web* Semântica (WS) visa aprimorar os aspectos das buscas, da interoperabilidade de dados entre sistemas *Web* e permitir que agentes possam processar automaticamente os dados e conhecimento. Neste contexto, a *Web* Semântica é caracterizada por representar o conteúdo e o conhecimento na forma de ontologias e contar com agentes inteligentes que permitem as máquinas entenderem o conteúdo publicado na *Web*.

Recentemente, os conceitos e técnicas da *Web* Semântica estão sendo incorporados nos sistemas de educação a distância, os quais são denominados Sistemas Educacionais Baseados na *Web* Semântica - SWBES (*Semantic Web Based Educational System*) (BITTENCOURT et al., 2009). Os SWBES, por adotar os conceitos da *Web* Semântica, permitem a criação de funcionalidades que auxiliam os alunos e professores a encontrarem materiais mais relevantes e reaproveitar conteúdos dos cursos entre diferentes sistemas de educação na *Web*. Outro importante aspecto é tornar o conteúdo didático processável pela máquina, o que torna possível a automação das diversas atividades de ensino utilizando os agentes inteligentes.

A maioria das arquiteturas de softwares dos SWBES são definidas empiricamente, levando em consideração apenas os módulos que a compõe sem a adoção de uma

metodologia da engenharia de software. Algumas arquiteturas não adotam a abordagem multiagentes e as que utilizam essa abordagem, o fazem de forma empírica. Existem trabalhos, como o de Bittencourt e colegas (2009), que apresentam a definição de uma arquitetura multiagentes para um SWBES, utilizando a metodologia Gaia (ZAMBONELLI; JENNINGS e WOOLDRIDGE, 2003).

A contribuição desse trabalho é a concepção e implementação de um SWBES utilizando uma metodologia de desenvolvimento orientado a agentes.

1.1 Motivação e escopo

Um dos principais aspectos da *Web* Semântica é prover uma infraestrutura de dados e conhecimento que possam ser processados por máquinas (agentes). Embora muitas das aplicações na *Web* Semântica sejam baseadas em agentes, os modelos de *softwares* utilizados nessas aplicações são construídos sem a adoção de metodologia orientada a agentes. Neste sentido, metodologias de desenvolvimento de *software* multiagentes vêm sendo criadas e podem ser utilizadas no desenvolvimento de sistema para a *Web* Semântica. Mas mesmo assim, muitos softwares ainda são desenvolvidos sem utilizar alguma dessas metodologias.

Em especial, os SWBES existentes adotam a abordagem baseada em agentes para a automação das tarefas, reaproveitamento dos materiais e no aprimoramento da busca de conteúdos. No entanto, muitos desses sistemas também são desenvolvidos sem a utilização de métodos. Neste cenário, utilizando uma metodologia orientada a agentes, foi feito o desenvolvimento de um modelo multiagentes para demonstrar como pode ser realizada a construção de uma arquitetura de SWBES. A metodologia de desenvolvimento de sistemas multiagentes utilizada segue os passos de um *framework* de desenvolvimento.

Portanto, o foco desse trabalho é a definição da arquitetura de um SWBES e o seu modelo multiagente. O desenvolvimento do modelo multiagentes tem intenção de investigar a comunicação entre os agentes. E não são alvo desse estudo a implementação dos melhores métodos de avaliação e busca, nem mesmo a discussão de métodos psico-pedagógicos.

A arquitetura definida foi desenvolvida visando ser flexível e aberta para ser adaptada as diversas formas de avaliação existentes e aos diversos mecanismos de buscas atuais.

1.2 Objetivos

O objetivo deste trabalho consiste na investigação de um modelo multiagentes e de uma arquitetura para sistema de ensino baseado na *Web* Semântica. No desenvolvimento desta tarefa será utilizada uma metodologia de desenvolvimento de software orientada a agentes. Posteriormente, para validar a arquitetura proposta foi implementado um sistema de ensino baseado na *Web* Semântica. Um estudo de caso foi feito sobre a construção de um curso introdutório de Rochas Carbonáticas.

Para chegar a esse objetivo, vários estudos foram feitos: estudo de sistema de ensino na *Web;* comparação da *Web* atual e a *Web* Semântica; os estudos dos conceitos e tecnologias usadas na *Web* Semântica, que envolvem ontologias, agentes, metodologias de desenvolvimento de sistemas multiagentes e arquiteturas específicas para esse tipo de sistema.

1.3 Metodologia

Este trabalho é de natureza aplicada, tendo como principal objetivo a investigação de um modelo multiagentes desenvolvido através de uma metodologia e a arquitetura de um sistema de ensino na *Web* Semântica.

O desenvolvimento do trabalho seguiu basicamente, as seguintes etapas:

- Levantamento bibliográfico para estudos da *Web* Semântica;
- Levantamento bibliográfico para estudos das metodologias para construção de ontologias;
- Investigação e estudo de sistemas de ensino na *Web*;
- Levantamento bibliográfico para estudos das metodologias de desenvolvimento de sistemas multiagentes;
- Levantamento bibliográfico para estudos das arquiteturas de sistemas desenvolvidos para a *Web* Semântica;
- Esforços para a definição da arquitetura do SWBES a ser desenvolvido;
- Esforços para a construção da modelagem multiagentes seguindo o *framework* Tropos;
- Esforços para definição e codificação das ontologias de domínio (Rochas Carbonáticas), material de aprendizagem e perfil;
- Implementação de um SWBES aplicado ao ensino de rochas carbonáticas

1.4 Organização dos capítulos

De acordo com a visão geral introduzida neste capítulo, esta dissertação foi organizada em mais cinco capítulos, conforme descrito a seguir.

No Capítulo 2 são abordados os aspectos conceituais necessários para este trabalho. Primeiramente é apresentado o tema educação a distância, passando pela evolução do uso dos computadores na educação e o uso de metadados. Depois, o tema *Web* Semântica é tratado, discorrendo sobre as tecnologias utilizadas para construção de ontologias, agentes e metodologia de desenvolvimento de sistemas multiagente, em especial a metodologia Tropos.

No Capítulo 3 são apresentadas as abordagens adotadas para sistemas educacionais baseados na *Web* Semântica. São descritas as arquiteturas de referência usadas para a definição de uma arquitetura para o SWBES. Essa arquitetura proposta é comentada em detalhes, definindo todos os seus componentes.

No Capítulo 4 é abordada a modelagem dos agentes utilizados pela arquitetura proposta no capítulo anterior. São passadas em detalhes as fases de Requisitos Iniciais, Requisitos Finais, Projeto Arquitetural e Projeto Detalhado definidas pelo Tropos.

No Capítulo 5 é apresentada uma implementação da arquitetura proposta para um SWBES aplicado ao ensino de Rochas Carbonáticas. Inicialmente são definidas as tecnologias adotadas para a construção de cada componente do sistema. E depois, o funcionamento o SWBES construído.

Por fim, as conclusões e considerações finais são apresentadas no Capítulo 6.

2 Aspectos Conceituais

Neste capítulo são apresentados os principais aspectos conceituais utilizados no desenvolvimento deste trabalho. Inicialmente, será descrita a evolução do uso de computadores na Educação a Distância e em seguida a *Web* Semântica e suas tecnologias. Ao final, é descrita a metodologia de desenvolvimento de sistemas multiagentes e a metodologia Tropos.

2.1 Educação a distância

Os trabalhos na área de Educação a Distância (EAD) tiveram seu início com o ensino por correspondência, com foco na educação profissional em áreas técnicas. Nas décadas de 70 e 80, começaram os cursos supletivos que usualmente usavam satélites para transmissão e material impresso para os alunos acompanharem as aulas. Outros recursos também eram usados, como rádio, televisão, fitas de áudio e de vídeo (BORBA, MALHEIROS e ZULATTO, 2007).

A EAD ganhou uma nova dinâmica com a popularização e a evolução dos computadores e das tecnologias da informação e comunicação.

2.1.1 A evolução do uso dos computadores na Educação

Segundo Costa (2000), o emprego de computadores no suporte à educação e treinamento ocorreu pouco tempo depois da construção do computador digital. Essa evolução motivou a criação de variadas propostas do emprego de computadores na educação, as quais refletem concepções diferentes do processo ensino-aprendizagem.

Valente (1999) acrescenta ainda que naquela época os computadores eram usados apenas para armazenar a informação em uma dada sequência para transmiti-la ao aprendiz. No entanto, hoje a informática na Educação é muito mais diversificada e o computador é utilizado em ambientes de ensino-aprendizagem para auxiliar o aprendiz no processo de construção do seu conhecimento.

De acordo com Costa (2000), a evolução dos programas educacionais pode ser demonstrada em quatro momentos distintos.

No primeiro momento existiam três categorias de softwares educacionais: Sistemas de Instrução Assistida por Computador (CAI), Micromundos e Simuladores de Jogos Educacionais. Os CAIs foram os primeiros softwares a surgirem nessa área, apoiando-se

numa abordagem comportamental. Neste tipo de sistema o aluno caminha por uma estrutura pré-estabelecida de unidades de ensino que visa conduzi-lo a um dado comportamento. A principal crítica a esse sistema é que ele não leva em conta o conhecimento prévio do aluno e nem suas características cognitivas.

No segundo momento, com a adoção das técnicas da Inteligência Artificial, os sistemas CAIs deram origem aos Sistemas Tutores Inteligentes (STI). Os STIs, de acordo com Freedman (2000), são sistemas que utilizam as técnicas de representação de conhecimento para implementar sistemas que abordam questões como: o que ensinar, a quem ensinar e como ensinar. O modelo tradicional de um STI é composto de quatro componentes: o modelo de domínio, o modelo do estudante, o modelo do professor e o ambiente de aprendizado ou interface com o usuário.

Nas duas últimas décadas, com a vertiginosa expansão da Internet, foi possível criar novas ferramentas que auxiliam o processo educacional. Ferramentas de autoria que disponibilizam conteúdo em tempo real ajudaram a superar os meios convencionais de educação a distância, possibilitando a troca de informações de forma mais rápida, independentemente das distâncias geográficas (ARAÚJO, 2003). Essas ferramentas fazem parte do terceiro momento, em que a computação distribuída permite a criação de salas de aulas virtuais e os STIs evoluem para ambientes interativos de aprendizagem, fazendo uso da abordagem de Sistemas Multiagentes (SMA).

Em decorrência da evolução da Internet, foi desenvolvida a primeira geração de Sistemas Educacionais Baseados na *Web* (*Web Based Educational System* - WBES), representados pelos Sistemas de Gerenciamento de Ensino (*Learning Management System* - LMS), nos quais destacam-se o Blackboard (BLACKBOARD, 2009) e o Moodle (MOODLE, 2009).

No quarto momento, os ambientes de ensino buscam tirar proveito das redes de alta velocidade, em que se destaca o projeto TIDIA-AE (TIDIA-AE, 2009). Esse projeto desenvolve um sistema de educação para redes de alta velocidade. Houve também uma contribuição significativa da Inteligência Artificial Distribuída, por meio da abordagem de Sistemas Multiagentes, contribuindo para tornar mais ricas as interações intra e inter indivíduos, bem como nos sistemas de hipermídia adaptativos.

Mesmo após todos esses avanços, ainda existem limitações no modelo de educação baseada na *Web*. Algumas das limitações identificadas por Devedzic (2000 *apud* ARAÚJO, 2003) são apresentadas a seguir:

- Não possui compartilhamento e reutilização dos materiais de aprendizagem entre aplicações, pois cada ferramenta trabalha com uma estrutura de armazenamento independente e não cambiável;
- Não existem pontos comuns de referência entre materiais de aprendizagem, em

que conceitos comuns possam ser compartilhados entre diversos materiais;

- Não existe uma anotação (metadados) nos materiais existentes;
- Falta de suporte na interação entre aplicações e usuários. Os usuários necessitam aprender a ferramenta antes de aprender o conteúdo. Não existe uma facilidade de o sistema conhecer a necessidade do aluno e apresentar informações pré-selecionadas de forma automatizada.

Alguns problemas citados estão relacionados à reutilização de conteúdo. Neste sentido, a solução adotada tem sido o uso dos padrões de metadados para a descrição de objetos de aprendizagem.

Segundo Wiley (2001 *apud* ARAUJO, 2003), objeto de aprendizagem é qualquer recurso digital que possa ser reutilizado para suporte ao ensino. Os objetos de aprendizagem são pequenos pedaços de um material de aprendizagem.

2.1.2 Padrões de metadados para educação baseada na *Web*

Uma das formas para resolver alguns problemas relacionados à reutilização de conteúdo é a adoção de padrões de metadados para objetos de aprendizagem. Várias organizações procuraram criar padrões para metadados educacionais, dentre elas destacam-se: *Learning Object Metadata* (LOM) do IEEE (LOM, 2011), *Technology Standards Committee* (LTSC) do *Global Learning Consortium* (IMS) (LTSC, 2011), o Ariadne da *Alliance of Remote Intructional Authoring and Distribution Networks for Europe* (ARIADNE, 2011) e o *Sharable Content Object Reference Model (*SCORM) da ADL (SCORM, 2011).

Por meio dos padrões é possível definir como os materiais de aprendizagem podem ser descritos para que possam ser encontrados mais facilmente e também para facilitar a interoperabilidade entre as aplicações (ARAUJO, 2003).

2.2 *Web* Semântica

A *World Wide Web* (*Web*) mudou o modo como as pessoas se comunicam, disponibilizam e acessam recursos. Atualmente, a *Web* é usada como o principal meio no processamento de informações, como plataforma de aplicações envolvendo bases de dados, processamento de texto e jogos.

A evolução das tecnologias da *Web* tem permitido o acesso ao conteúdo por meio de diversos dispositivos, porém cabe aos seres humanos o processo de interpretação do conteúdo. O processo de interpretação é uma tarefa muito mais difícil e requer o conhecimento para avaliar, classificar e selecionar a informação de interesse. O grande

desafio é conceder aos computadores a capacidade de executar essa tarefa para os humanos (BREITMAN, 2005). Por exemplo, quando acessamos um site de busca e procuramos por um produto ou um serviço específico, muitos resultados irrelevantes são apresentados, o que demanda certo tempo e esforço até filtrarmos as informações de interesse.

A *Web* foi desenvolvida como um meio para a troca e compartilhamento de documentos entre pessoas, em vez de um meio que fomentasse a troca de dados e informações que pudessem ser processadas automaticamente. Entretanto, o conteúdo semântico das páginas, ou seja, seu significado é codificado de uma maneira acessível apenas para os seres humanos (BREITMAN, 2005).

Segundo Lacy (2005), a *Web* não provê a estrutura suficiente para suportar o processamento avançado de conteúdo por computador. Dependemos da *Web* para administrar nossas vidas e negócios. No entanto, necessitamos de melhores representações da informação na *Web* para que possamos ter aplicações mais avançadas.

Para que o conteúdo disponível possa ser processado por máquinas, uma outra abordagem para representar os dados disponíveis na *Web* foi idealizada. Essa nova *Web* é denominada *Web* Semântica (ANTONIOU; VAN HARMELEN, 2004).

2.2.1 Ontologias

Na implementação da *Web* Semântica é necessário definir um modelo de representação das informações e do conhecimento. Nesse contexto as ontologias são utilizadas para capturar o conhecimento do domínio de interesse, descrever seus conceitos e também as relações (HORRIDGE, KNUBLAUCH; STEVENS, 2007).

A palavra ontologia vem do grego *ontos* (ser) + *logos* (palavra). O termo ontologia possui diversas definições em diferentes áreas do conhecimento. A definição encontrada mais frequentemente na literatura da *Web* Semântica é a proposta por Gruber (1993), onde “Ontologia é uma especificação formal e explícita de uma conceitualização compartilhada”. No contexto da *Web* Semântica, as ontologias são utilizadas na representação de alguma área do conhecimento, codificadas segundo padrões estabelecidos.

A definição de ontologia de acordo com o W3C inclui que as ontologias devem prover descrições das classes nos vários domínios de interesse, relacionamentos entre as classes e propriedades que estas devem possuir (BREITMAN, 2005).

Utilizando como critério de classificação da generalidade de uma ontologia, Guarino (1998) dividiu as ontologias em 4 tipos de acordo com sua finalidade:

- Ontologia de alto nível: esse tipo de ontologia descreve os conceitos mais

genéricos, tais como espaço, tempo e eventos. Conceitos esses que são independentes do domínio e podem ser reutilizadas na confecção de novas ontologias.

- Ontologia de domínio: descrevem o vocabulário relativo a um domínio específico por meio da especialização de conceitos presentes na ontologia de alto nível.
- Ontologia de tarefas: descrevem o vocabulário relativo a uma tarefa genérica ou atividade por meio da especialização de conceitos presentes na ontologia de alto nível.
- Ontologias de aplicação: ontologias mais específicas. Os conceitos presentes nesse tipo de ontologia correspondem, de maneira geral, a papéis desempenhados por entidades do domínio no desenvolvimento de alguma tarefa.

2.2.1.1 Metodologias para a construção de ontologias

No desenvolvimento de aplicações na *Web* Semântica, um dos principais aspectos é dispor de ontologias. A construção de ontologias é uma tarefa complexa que requer metodologias adequadas. Há várias pesquisas visando propor uma metodologia mais adequada para o desenvolvimento de ontologias para *Web* Semântica. Em decorrência da complexidade, ainda não há uma única metodologia adequada para todas as situações. A melhor solução frequentemente adotada é a escolha entre algumas possibilidades ou a composição de várias metodologias (BREITMAN, 2005).

Um dos grandes desafios na construção de ontologias é captar as reais necessidades dos profissionais da área de interesse. Esse processo é marcado pelo constante *feedback* e participação dos especialistas do domínio estudado, por meio de diversas reuniões.

Há diferentes metodologias propostas para a construção de ontologias, as mais conhecidas são:

- A metodologia Enterprise foi proposta por Mike Uschold, da Universidade de Edimburgo, em cooperação com outras empresas, para a construção da ontologia *Enterprise*, onde foram modelados os processos das empresas (USCHOLD et al. 1995).
- Gruninger e Fox propuseram a metodologia Toronto Virtual Enterprise (TOVE), que foi derivada da experiência dos autores no desenvolvimento de ontologias para os domínios de processos de negócios e corporativo (GRUNINGER e FOX, 1995).

- O *Methontology*, desenvolvida no laboratório de Inteligência Artificial do Politécnico de Madri, é uma metodologia que adota aspectos de Engenharia de Software e conta com um *framework* para apoiar as atividades na construção de ontologias (GÓMEZ-PÉREZ, 2005).

No desenvolvimento das ontologias utilizadas no estudo de caso deste trabalho foi utilizada uma metodologia para construção de ontologias desenvolvida pelo Grupo de Engenharia de Requisitos da PUC-Rio. Esse método foi escolhido em razão do enfoque linguístico adotado pelas ferramentas de aquisição de conhecimento a partir de textos, utilizadas nas pesquisas do grupo de Inteligência Artificial da UNESP – Rio Claro.

Essa metodologia consiste em três fases: Construção do Léxico, Mapeamento Léxico-Ontologia e Construção da Hierarquia de Classes (BREITMAN, 2005).

Na Construção do Léxico (Fase I), é feito um levantamento do Léxico, identificando palavras ou frases peculiares ao meio social da aplicação sob estudo. Após a identificação dessas frases e palavras, procuram-se seus significados (BREITMAN, 2005). O resultado dessa etapa é uma lista de termos candidatos a fazerem parte da ontologia.

Após isso, parte-se para a modelagem do Léxico Ampliado da Linguagem (LAL). A representação do LAL requer que para cada termo sejam descritos Noções e Impactos. Noção é o significado do termo, enquanto Impacto descreve os efeitos do uso ou ocorrência do termo na aplicação ou o efeito de algo na aplicação sobre o termo.

Os símbolos ou termos do Léxico são classificados em quatro categorias: objeto, sujeito, estado e verbo. Símbolos ou termos do tipo objeto definem um objeto em questão e os relacionamentos que mantêm com outros termos do Léxico, sejam eles outros objetos, sujeito, estado ou verbos. Os Impactos de um termo do tipo objeto descrevem ações que podem ser aplicadas ao objeto. Termos do tipo sujeito descrevem uma pessoa ou grupo e quais ações executam. Termos do tipo estado definem o significado de um estado ou situação e ações que precedem o mesmo. Os Impactos de um termo do tipo estado devem descrever outros estados e ações que podem ocorrer a partir do estado inicial.

Com o Léxico construído, parte-se para a Fase II da modelagem: o Mapeamento Léxico-Ontologia. Os temos do Léxico classificados como do tipo objeto e sujeito serão mapeados em classes da ontologia; os termos do Léxico classificados em verbo serão mapeados para propriedades; os termos classificados como do tipo estado serão mapeados para classes ou propriedades, dependendo de sua importância relativa para a ontologia. A noção de cada termo é mapeada na descrição da respectiva classe; e por meio da lista de Impactos de cada termo do Léxico mapeia-se o verbo em propriedades e o predicado em restrições das classes.

A última fase desse processo é a construção da Hierarquia de Classes, que consiste na análise da Ontologia de modo a identificar conceitos que possam estar

relacionados hierarquicamente (BREITMAN, 2005).

2.2.1.2 Codificação das Ontologias

Após terem sido modeladas seguindo alguma metodologia, as ontologias necessitam ser codificadas em linguagens que possam ser processadas por máquinas. Uma linguagem para codificar ontologias na *Web* Semântica deve atender a alguns requisitos, como, por exemplo: as ontologias poderem ser explicitamente estendidas, possuírem suporte a tipos complexos de dados, poderem ser referenciadas de forma não ambígua e disporem de métodos de inferência.

Algumas linguagens foram propostas para a construção de ontologias para a *Web* Semântica, dentre elas estão a SHOE (*Simple HTML Ontology Extension*)[1], da Universidade de Maryland, a OIL (*Ontology Inference Layer*)[2], patrocinada por um consórcio da Comunidade Européia através do projeto *On-to-Knowledge*, a DAML (*DARPA Agent Markup Language*)[3], do *Defense Advanced Research Projects Agency* (DARPA), a DAML + OIL[4], uma combinação da DAML e da OIL, e a OWL (*Web Ontology Language*)[5], uma revisão da linguagem DAML + OIL (BREITMAN, 2005). Atualmente, a linguagem mais difundida para a construção de ontologias é a OWL.

Neste trabalho, as ontologias criadas foram construídas em OWL. A OWL utiliza uma arquitetura em camadas (Figura 1).

Aplicações		Camada de Implementação
Ontology Languages (OWL Full, OWL DL and OWL Lite)		Camada Lógica
RDF Schema	***Individuals***	Camada Ontológica Primitiva
RDF* e *RDF/XML		Camada Básica da Linguagem Relacional
XML* e *XMLS Datatypes		Camada de Transporte/Sintática
URIs* e *Namespaces		Camada de Referência/Simbólica

Figura 1: Arquitetura em camadas da *Web* Semântica

Fonte: Extraído e adaptado de LACY (2005)

1 http://www.cs.umd.edu/projects/plus/SHOE/

2 http://www.ontoknowledge.org/oil/

3 http://www.daml.org/

4 http://www.daml.org/2001/03/daml+oil-index.html

5 http://www.w3.org/TR/owl-features/

2.2.1.3 URIs e *Namespaces*

Os elementos básicos em uma aplicação da *Web* Semântica são: *Uniform Resource Identifier* (URI) e os *namespaces*.

URI é uma cadeia de caracteres que identifica um recurso da Internet. O URI mais comum é o *Uniform Resource Locator* (URL), que identifica um endereço de domínio na Internet (W3SCHOOLS, 2009).

Os *Namespaces* provêm um método de evitar conflitos com nomes de elementos XML. Estes são um atributo de documentos XML.

2.2.1.4 XML e XML *SCHEMA*

Outro aspecto conceitual importante na *Web* Semântica é a linguagem de marcação de dados chamada e*Xtensible Markup Language (*XML) (BRAY et al., 2008), que foi desenvolvida para ser uma metalinguagem e a partir dela, vários padrões foram criados. A XML é uma recomendação do W3C.

A XML não é apenas mais uma linguagem de marcação como a HTML (*Hyper Text Markup Language*), pois ela possibilita a utilização de vários recursos importantes. Por exemplo, possibilita ao desenvolvedor definir marcadores personalizados, dando significado ao texto armazenado entre os marcadores. O objetivo maior do desenvolvimento da linguagem XML foi possibilitar que os criadores de páginas *Web* descrevessem suas próprias *tags*, superando as limitações impostas pela HTML (FURGERI, 2001). A estrutura criada pelo documento XML permite que ferramentas possam consultar e processar seu conteúdo.

Na Figura 2 é apresentado um exemplo de um recado marcado por *tags* no padrão do XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nota>
      <para>João</para>
      <de>Maria</de>
      <cabeçalho>Lembrete</cabeçalho>
      <corpo>Não se esqueça de pegar o trem das onze!</corpo>
</nota>
```

Figura 2: Exemplo de XML

O exemplo na Figura 2 é auto-descritivo, sendo fácil perceber o remetente e o destinatário, além do um cabeçalho e do corpo de mensagem. Na primeira linha é feita a declaração XML: define a versão XML (1.0) e a codificação utilizada (ISO-8859-1 = *Latin-1/West European character set*). Na próxima linha há o marcador referente ao elemento "nota", correspondente a raiz do documento. Nas próximas quatro linhas são descritos os quatro elementos filhos da raiz (*de, para, cabeçalho e corpo*). Na última linha há o marcador que define o final do elemento raiz.

Os documentos XML devem obrigatoriamente conter um elemento raiz, que é o "pai" de todos os outros elementos. Além disso, todos os elementos podem ter "filhos". Os termos pai, filho e irmão descrevem os relacionamentos entre os elementos. Filhos no mesmo nível são chamados de irmãos. Todos os elementos podem conter texto e atributos (como em HTML).

O XML *Schema* (XMLS) (FALLSIDE; WALMSLEY, 2004) provê importantes características para documentar a estrutura de documentos XML. O XMLS também permite definir as características dos tipos de dados. Na Figura 3 é apresentado o *schema* correspondente ao exemplo de XML descrito na Figura 2. O XMLS adiciona suporte a tipos de dados para a OWL com tipos internos de dados primitivos e derivados (LACY, 2005).

```
<xs:element name="nota">
<xs:complexType>
      <xs:sequence>
          <xs:element name="de" type="xs:string"/>
         <xs:element name="para" type="xs:string"/>
         <xs:element name="cabeçalho" type="xs:string"/>
         <xs:element name="corpo" type="xs:string"/>
      </xs:sequence>
</xs:complexType
</xs:element>
```

Figura 3: XML *Schema* correspondente a nota

2.2.1.5 RDF e RDF SCHEMA

A XML provê características para a representação e intercâmbio de informações, porém são insuficientes para suportar os requisitos da *Web* Semântica. A principal razão é que a XML define somente a sintaxe, e não os aspectos relacionados à semântica. Embora a XML seja especificada para descrever recursos, estas especificações não permitem

descrever o relacionamento entre recursos (LACY, 2005).

Neste contexto, foi proposto o *Resource Description Framework* (RDF) (MANOLA; MILLER, 2004), que tem como finalidade formar a camada relacional básica da arquitetura da *Web* Semântica. O RDF é usado para especificar instâncias da *Web Ontology Language* (OWL), que será vista na próxima seção. Este *framework* é utilizado para descrição de recursos, sendo um mecanismo padrão para construção de declarações sobre recursos. Inicialmente, o RDF foi projetado para codificar metadados. Muitos textos descrevem o RDF como uma forma padrão para representar metadados.

O fundamento da RDF consiste de um modelo básico para descrição de dados que consiste em três tipos de objetos:

- Recursos: um recurso é especificado por uma URI e pode ser uma página da *Web*, um site na *Web* ou parte deste.
- Propriedades: uma propriedade é uma característica, um atributo ou uma relação utilizada para descrever um recurso.
- Declarações: são recursos acrescidos das propriedades desse recurso e do valor dessas propriedades.

Essas três partes individuais são chamadas de sujeito, predicado e objeto, respectivamente. Por exemplo, a seguinte sentença: José da Silva é o Gerente do site http://www.rapidos.com.br, pode ser representado através de grafos rotulados como na Figura 4. Este grafo pode ser lido como: http://www.rapidos.com.br tem como Gerente José da Silva, ou de maneira mais geral, "<sujeito> TEM <predicado><objeto>".

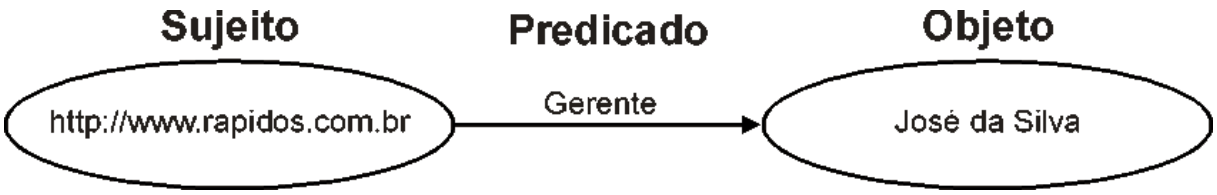


Figura 4: Representação de um relacionamento Sujeito/Predicado/Objeto

Um exemplo de um documento RDF para representar uma lista de CDs (W3SCHOOLS [B], 2009) é apresentado na Figura 5.

Um documento RDF é uma declaração XML que é seguida pelo elemento raiz de documentos RDF: <rdf:RDF>. Os *namespace rdf* e cd são definidos como referência às URI nas quais estão especificados os esquemas. Assim, os elementos com o prefixo “rdf” estão especificados no *namespace* "http://www.w3.org/1999/02/22-rdf-syntax-ns#". Os elementos com o prefixo “cd” estão especificados no *namespace* "http://www.recshop.fake/cd#". O elemento <rdf:Description> contém a descrição do recurso identificado pelo atributo

rdf:about. Os elementos: <cd:artist>, <cd:country>, <cd:company> e demais, são propriedades do recurso.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">
<rdf:Description
 rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
</rdf:Description>
<rdf:Description
 rdf:about="http://www.recshop.fake/cd/Hide your heart">
  <cd:artist>Bonnie Tyler</cd:artist>
  <cd:country>UK</cd:country>
  <cd:company>CBS Records</cd:company>
  <cd:price>9.90</cd:price>
  <cd:year>1988</cd:year>
</rdf:Description>
.
.
.
</rdf:RDF>
```

Figura 5: Exemplo de um RDF que descreve uma lista de CDs de Música

O RDF descreve os recursos utilizando classes, propriedades e declarações. Assim como a XML, o RDF também necessita de uma maneira para definir classes e propriedades, que devem ser definidas utilizando-se extensões para o RDF. Uma extensão é o RDF *Schema* (RDFS) (BRICKLEY; GRUHA, 2004). O RDF *Schema* provê a estrutura para descrever classes e propriedades. Classes em RDF *Schema* são muito parecidas com classes em linguagens de programação orientadas a objeto. Isto permite que os recursos sejam definidos como instâncias de classes, e subclasses de classes (W3SCHOOLS [C], 2009). Na Figura 6 é apresentado um exemplo de RDF *Schema*. No exemplo, o recurso "*horse*" é uma subclasse da classe "animal".

O RDF *Schema* provê mecanismos para declaração de propriedades. Um esquema não define somente as propriedades dos recursos, mas também os tipos de recursos que estão sendo descritos. Pode ser entendido como uma espécie de dicionário onde são

definidos os termos que serão utilizados em declarações RDF. A especificação do RDF *Schema* fornece os mecanismos necessários à definição de elementos, de classes de recursos, de possíveis restrições de classes e relacionamentos e detecção de violação de restrições.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base=  "http://www.animals.fake/animals#">
<rdf:Description rdf:ID="animal">
  <rdf:type
   rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:ID="horse">
  <rdf:type
   rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>
</rdf:RDF>
```

Figura 6: Exemplo de RDF *Schema* para descrição de animais

O RDF e o RDF *Schema* são recomendações do W3C.

2.2.1.6 OWL

O RDFS adiciona primitivas ontológicas que são críticas para a descrição de dados para *Web* Semântica. Isso provê características para suportar especificações de ontologia. No entanto, há outras características que ainda são necessárias para suportar os requisitos das linguagens para construção de ontologias. O RDFS é útil para especificar a filiação de classes. Porém, este não restringe a cardinalidade nas propriedades. Além disso, o RDFS dispõe de poucos descritores que suportam inferência. Regras adicionais são necessárias para raciocinadores inferirem novos fatos. Por fim, o RDFS não disponibiliza expressividade suficiente para prover as descrições de ontologia requeridas para a *Web* Semântica. Para tal, conceitos mais avançados são necessários. Estes são providos pela OWL (*Web Ontology Language*) (MCGUIINNESS; VAN HARMELEN, 2004).

Segundo Breitman (2005), a OWL foi projetada de modo a atender as necessidades das aplicações para a *Web* Semântica, que podem ser resumidas em:

- Construção de ontologias;
- Explicitar fatos sobre um determinado domínio;
- Raciocínio, por meio de inferência, sobre ontologias e fatos.

De maneira similar à DAML + OIL, a intenção da OWL é representar conceitos e seus relacionamentos na forma de uma ontologia. OWL está dividida em três linguagens, que descrita em ordem crescente de expressividade são: OWL Lite, OWL DL e OWL FULL (BREITMAN, 2005).

A OWL-Lite é sintaticamente a sub-linguagem mais simples. Ela tem o propósito de ser usada em situações onde apenas uma simples hierarquia de classes e simples restrições são necessárias.

A OWL-DL é muito mais expressiva do que a OWL-Lite e é baseada em Lógicas Descritivas (DL - *Descripitions Logics*). A Lógica Descritiva é um fragmento da Lógica de Primeira Ordem, portanto, propícia ao raciocínio automático. É possível então computar automaticamente a hierarquia de classificação e verificar inconsistências na ontologia (HORRIDGE et al., 2007).

A OWL-Full é a mais expressiva sub-linguagem da OWL. Ela tem o propósito de ser usada em situações em que alta expressividade é mais importante do que garantir decidibilidade ou exaustividade computacional da linguagem. Não é, portanto, possível realizar raciocínio automático em ontologias codificadas em OWL-Full.

A OWL possui alguns elementos essenciais, como os *namespaces* (vocabulários que estão sendo utilizados), os cabeçalhos (coleções de sentenças sobre a ontologia agrupadas), as classes, que representam um conjunto ou coleção de indivíduos, bem como suas subclasses, os indivíduos, que são objetos do mundo, pertencem a uma classe e estão relacionados a outros indivíduos e classes por meio de propriedades (BREITMAN, 2005).

As propriedades servem para descrever fatos em geral. São relações binárias, ligando dois indivíduos (HORRIDGE et al., 2007). Estas são utilizadas para criar restrições, que definem alguns limites para indivíduos que pertencem a uma classe (podem ser quantificadores, restrições de cardinalidade e restrições “tem valor de”) (BREITMAN, 2005). Restrições podem ainda ser existenciais. Para um conjunto de indivíduos, uma restrição existencial especifica a existência de pelo menos um relacionamento que utiliza uma dada propriedade com um indivíduo de uma classe específica.

2.3 O uso da *Web* Semântica na educação

Com o surgimento da *Web* Semântica, a qual integra os conceitos da *Web* e da área de Inteligência Artificial, foi possível conceber novos sistemas de ensino, os Sistemas

de Ensino Baseado na *Web* Semântica (BITTENCOURT et. al., 2009).

A *Web* Semântica permite aos computadores interpretar e compreender o conteúdo dos materiais disponíveis, possibilitando novas formas de navegação. A *Web* Semântica estende a *Web* Clássica provendo uma estrutura semântica para que as páginas *Web* possam ser entendidas por humanos e agentes inteligentes. Segundo Stojanovic, Staab e Studer (2001), a propriedade chave da *Web* Semântica é o compartilhamento de significados e o processamento automático de metadados. Com isso, um material de aprendizagem pode ser semanticamente anotado. Isso torna possível relacionar tópicos semelhantes criando novos cursos, pesquisar conteúdos de outras plataformas e personalizar o material de estudo de forma fácil. Um exemplo disso seria o aluno realizar uma pesquisa do tipo "Quais são os materiais sobre ontologia disponibilizados na UNESP?". O sistema irá retornar não só os materiais disponibilizados no curso do aluno, como outros materiais de outros cursos que podem ser utilizados por ele.

Nos sistemas convencionais, do ponto de vista do aluno, pode-se destacar a dificuldade que o mesmo tem para encontrar um curso com o conteúdo adequado às suas necessidades. Além disto, os alunos podem ser desmotivados para continuar seus estudos devido à forma com que a informação é apresentada e as atividades são propostas e desenvolvidas. Sob o ponto de vista de quem oferece o serviço, encontra-se a reutilização de conteúdos e a vinculação com outros sistemas de WBES, como sendo as maiores dificuldades enfrentadas (BRANCO, 2006).

Llop (2007) define reutilização como a possibilidade de estruturar um curso utilizando uma ou mais peças de outras fontes já produzidas. Essas peças são chamadas de Objetos de Aprendizagem Compartilhados (*Sharable Learning Objects* - SLOs) ou Objetos de Conteúdo Compartilhados (*Sharable Content Objects* - SCOs). Tais elementos devem possuir características como independência, qualidade reconhecida, durabilidade, interoperabilidade e passível de uso por agentes inteligentes.

Segundo Goñi (2003), os SLOs são um dos avanços tecnológicos mais promissores na área da educação na *Web*. Cada SLO corresponde ao menor bloco de instrução ou informação, elaborado de forma independente capaz de transmitir um significado. Um SLO pode ser um arquivo de som, uma imagem, um filme ou até mesmo um programa. Usando ontologias para descrever os SLOs, é possível fazer uma descrição do seu conteúdo de uma forma mais refinada.

Prado (2004) apresenta um estudo da utilização de ontologias e abordagem multiagentes na implementação de sistema de ensino na *Web*. Como estudo de casos, foi feito um experimento voltado para o reforço no ensino de Linguagem de Programação.

2.4 Agentes

Na literatura existem diversos conceitos para agentes, um deles é o apresentado por Wooldridge (2009): “Um agente é um sistema de computador encapsulado que está situado em algum ambiente e que é capaz de realizar ações flexíveis e autônomas neste ambiente, a fim de cumprir os seus objetivos de projeto.”

Assim, temos os agentes que são programas e possuem interfaces definidas (sensores e atuadores). Através dessas interfaces eles têm a percepção e atuam sobre esse ambiente de forma autônoma, ou seja, possuem controle sobre seus estados e comportamentos, a fim de realizar tarefas para alcançar seus objetivos. Eles podem agir em respostas a mudanças do ambiente e são independentes da intervenção humana ou de outros sistemas para tomar decisões.

Um dos mais utilizados modelo de software para a programação de agentes inteligentes é o *Belief-Desire-Intention* (BDI) (RAO e GEORGEFF, 1991). Nesse modelo, o estado interno de processamento de um agente é descrito por meio de um conjunto de “estados mentais” correspondentes a: crenças (*Beliefs*), desejos (*Desires*) e intenções (*Intentions*).

Os estados mentais, crenças, desejos e intenções, representam, respectivamente, os seus estados: informacional, motivacional e deliberativo. Em uma arquitetura BDI um agente pode ser completamente especificado pelos eventos que podem ser percebidos, as ações que podem realizar, as crenças que podem manter, os objetivos que pode tomar, e os planos que dão origem a suas intenções (KINNY; GEORGEFF e RAO, 1996). Dessa forma um agente BDI é capaz de raciocinar continuamente sobre suas crenças, objetivos e intenções e reagir de acordo.

Em geral, uma arquitetura construída no modelo BDI é especificada em termos das estruturas a seguir (BRAUBACH; POKAHR e LAMERSDORF, 2005):

- Crenças: representam o estado informacional do agente, em outras palavras, suas crenças sobre o mundo (incluindo a si próprio e aos outros agentes, o seu conhecimento sobre o ambiente). Isto introduz uma visão de mundo pessoal dentro do agente podendo influenciar na maneira como o agente percebe e pensa sobre o mundo.
- Desejos: representam o estado motivacional do agente. Constituem objetivos ou situações que o agente gostaria de realizar ou produzir. São eles que determinam o curso de suas ações.
- Intenções: representam o estado deliberativo do agente - o que o agente tenha optado por fazer. As intenções são uma forma que os agentes têm para alcançar seus desejos.

- Planos: Os planos são sequências de ações que um agente pode realizar para atingir uma ou mais das suas intenções e reagir à ocorrência de eventos.

O modelo BDI é utilizado por *frameworks* e ferramentas que implementam agentes, como um mecanismo de raciocínio dos agentes, agindo principalmente para gerenciar a ocorrência de eventos e posterior escolha dos planos que irão tratar os mesmos.

Neste trabalho os agentes definidos seguem o modelo BDI.

2.4.1 Sistemas Multiagentes

Existem sistemas em que um único agente é suficiente para a realização de algumas tarefas, mas na maioria das vezes são necessários mais de um agente. Dessa forma, temos então um sistema multiagentes. Em sistemas multiagentes os agentes têm que cooperar, negociar e coordenar ações dentro de um ambiente comum.

Um sistema multiagentes, de acordo com Wooldridge (2009), pode ser definido como: um conjunto de agentes, que interagem uns com os outros. No caso mais geral, os agentes agem em favor de usuários com diferentes objetivos e motivações. Para terem sucesso nas interações, eles requererem habilidades de cooperação, coordenação e negociação uns com os outros.

Nesses sistemas os agentes exibem um comportamento autônomo e ao mesmo tempo interagem ou trabalham em conjunto de forma a desempenhar determinadas tarefas ou satisfazer um conjunto de objetivos.

2.4.2 Metodologia de Desenvolvimento de Software Multiagente

As metodologias de software orientadas a agentes são utilizadas para auxiliar os desenvolvedores a projetar sistemas multiagentes (SMA), isto é, determinar uma arquitetura que atenda os requisitos do sistema, identificar seus componentes, suas funcionalidades, como gerenciar o acesso aos recursos e também como será a coordenação e comunicação entre os agentes. Muitas metodologias que suportam a análise, projeto e implementação de SMA foram propostas no contexto da Engenharia de Software Orientada a Agentes (*Agent Oriented Software Engineering* - AOSE). Algumas dessas metodologias são Gaia, MaSE, Prometheus, Tropos, Message, Passi e Adelfe. A definição dessas e de outras metodologias podem ser encontradas em Henderson-Sellers et al. (2005), juntamente com uma comparação entre elas.

2.4.2.1 Metodologia Tropos

A metodologia adotada para o desenvolvimento do SMA proposto neste trabalho é a Tropos. A adoção dessa metodologia é decorrente do fato da mesma atender todas as fases do desenvolvimento de Software, enquanto outras metodologias costumam especificar apenas parte do processo. Neste sentido, Tropos oferece um *framework* que engloba todas as fases de desenvolvimento de *software*, dos requisitos iniciais até a implementação.

2.4.2.1.1 Conceitos

Os modelos na metodologia Tropos são descritos utilizando alguns componentes (BRESCIANI et al., 2004), cujas representações gráficas são apresentadas na Figura 7 e suas definições são descritas a seguir:

- Ator: entidade que possui objetivos estratégicos e intenções dentro do sistema;
- Objetivo ou meta: representam os interesses estratégicos dos atores;
- Meta-soft: representam os requisitos não-funcionais do sistema;
- Plano: representa de forma abstrata a maneira de se fazer alguma coisa. A execução de um plano pode ser considerada um “meio” de satisfazer um objetivo.
- Recurso: representa uma entidade física ou informacional.
- Dependência: indica que um ator (*depender*) depende de outro ator (*dependee*) por algum motivo (*dependum*), que pode ser para alcançar um objetivo, executar um plano, ou fornecer um recurso (Figura 9).

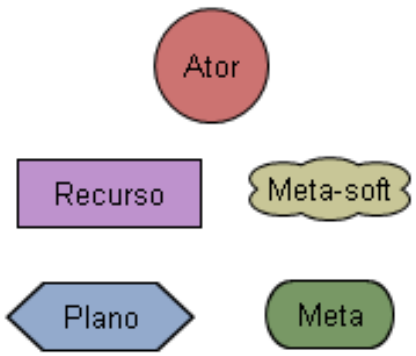


Figura 7: Componentes utilizados nos diagramas do Tropos

A diferença entre metas e metas-soft consiste em que as metas descrevem os serviços ou funcionalidades que o sistema deve possuir, ao passo que as metas-soft

referem-se a propriedades não-funcionais, tais como as qualidades do sistema esperado, como desempenho ou problemas de excelência (BRAUBACH et al., 2004).

Os componentes metas e planos utilizados na modelagem podem ser decompostos como *OR*, ou decomposição meio e fim (Figura 8). A decomposição *OR*, pode ser aplicada tanto para metas como para planos, significando que para alcançar a meta raiz uma das sub-metas tem que ser atingida. A decomposição *AND*, também é aplicada a metas e planos, significando que todas as sub-metas ou planos devem ser alcançadas para que a meta ou plano raiz seja alcançada.

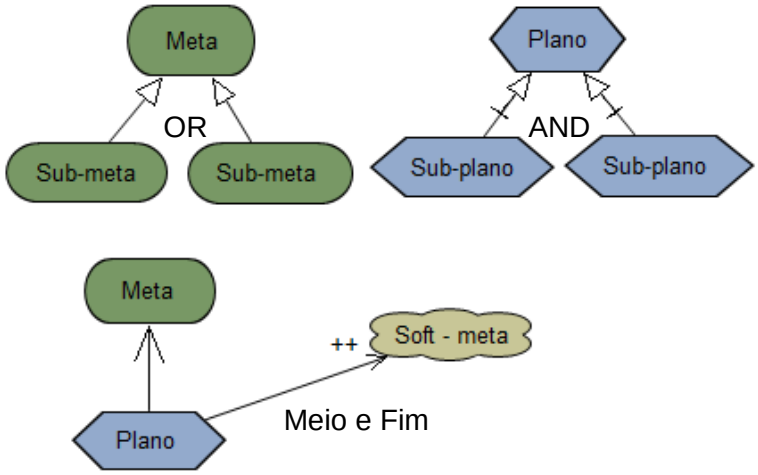


Figura 8: Ligações de decomposição entre os componentes

A decomposição meio fim é utilizada para a decomposição de um plano que é um meio para alcançar uma meta que é um fim. Essa ligação só é permitida de planos para metas e metas para metas. Há também uma forma de explicitar o nível de contribuição representado na ponta da seta por um sinal (+,++, -, --), significando que um plano ou meta contribuem positiva ou negativamente para meta-soft, dependendo do sinal.

Na metodologia há forma de explicitar um relacionamento de dependência entre dois atores (Figura 9). Um ator (*depender*) depende de outro ator (*dependee*) por algum motivo (*dependum*). Esse motivo pode ser para alcançar um objetivo (utiliza uma meta como *dependum*), executar um plano (utiliza um plano como *dependum*), ou fornecer um recurso (utiliza um recurso como *dependum*).



Figura 9: Dependência entre atores

2.4.2.1.2 Fases da modelagem

A metodologia Tropos possui um conjunto de fases de desenvolvimento e ao final de cada fase é produzido um modelo ou diagrama, que será refinado nas próximas fases. Para a produção desses modelos são utilizadas algumas atividades de modelagem. A seguir são apresentadas as atividades que contribuem para a criação, refinamento e evolução dos modelos (BRESCIANI et al., 2004):

- Modelagem de Atores *(Actor Modeling)* – consiste em identificar e analisar os atores do sistema (*stakeholders* - representam as partes interessadas, como usuários, organizações, sistemas), e representá-los nos diagramas como atores sociais e suas intenções;
- Modelagem de Dependências *(Dependency modeling)* – identificar os atores que dependem um do outro para alcançar objetivos, executar planos e fornecer recursos. Consiste no relacionamento dos atores utilizando para isso as relações de dependência;
- Modelagem de Metas *(Goal Modeling)* – baseia-se na analise de objetivos do ponto de vista do ator. É utilizada para representar as intenções dos atores. Utiliza 3 técnicas básicas:
    - Decomposição AND/OR (*AND/OR decomposition)* – decompõe um objetivo raiz em sub-objetivos;
    - Análise Meio-Fim (*Means-end analisis)* – identificar planos recursos e metas-soft que providenciam um meio para alcançar um objetivo;
    - Análise de Contribuição (*Contribution analisis)* – identifica uma meta-soft que contribui positiva ou negativamente para o funcionamento do objetivo analisado;
- Modelagem de Planos *(Plan Modeling)* – similar a *goal modeling* e utiliza as mesmas técnicas, para planos ao invés de metas.

Na atividade de *Goal Modeling* deve ser realizado o seguinte procedimento: para cada ator do diagrama de metas são aplicadas as três técnicas. Primeiro cada meta em relação ao ator responsável por seu cumprimento deve ser analisada e se possível decomposta em sub-metas. Depois é realizada a decomposição das metas através de uma análise de “meios-fins” que visa identificar os planos, recursos, metas-soft que forneçam os meios para alcançar as metas. Uma análise de contribuição também é realizada para identificar os objetivos que contribuem positiva ou negativamente para as metas-soft.

2.4.2.1.3 Fases de Desenvolvimento

Segundo Bresciani et al. (2004), a metodologia Tropos é dividida em cinco fases de desenvolvimento:

a. Requisitos Iniciais

O primeiro passo dessa fase é, a partir da descrição do sistema, encontrar as partes interessadas e suas intenções. As partes interessadas podem ser organizações, pessoas, subsistemas, etc. Cada parte interessada é representada por um ator social e suas intenções correspondem às metas desse ator. Essa fase foca na representação da estrutura organizacional existente para a compreensão do problema onde o sistema será introduzido.

Inicialmente é feita a análise que visa a Modelagem dos Atores (*Actor Modeling*) em que os engenheiros de requisitos modelam *stakeholders* como atores e suas intenções como metas.

Em seguida é realizada a análise que visa a Modelagem de Dependência (*Dependency Modeling*), em que cada meta é analisada do ponto de vista do ator resultando num conjunto de dependências entre pares de atores.

A partir dessas duas análises é produzido o Diagrama de Atores que representa os atores e seus relacionamentos.

Após, é realizada a análise que visa a Modelagem de Metas (*Goal Modeling*), onde as metas dos atores são decompostas e também encontradas suas contribuições. E em seguida é aplicada a análise visando a Modelagens dos Planos (*Plan modeling*) onde os planos são definidos e decompostos de modo a estabelecer como alcançar as metas.

Ao final dessa fase são obtidos como resultados dois diagramas: o Diagrama de Atores – representa os atores e seus relacionamentos; e o Diagrama de Metas – representa a forma como os atores alcançaram suas metas.

b. Requisitos Finais

Esta fase é a continuação da análise dos Requisitos Iniciais com o acréscimo do sistema que será desenvolvido chamado de ator sistema (traduzido da expressão *system-to-be*). No decorrer da análise o ator sistema pode se decomposto em sub-atores que vão assumir algumas de suas responsabilidades.

Nessa fase as questões a serem analisadas são as seguintes (MORANDINI et al., 2007): quais são as metas que podem ser designadas ao sistema e quais dependências podem ser redirecionadas dos atores para o sistema – de forma que o sistema possa

realizar tarefas e alcançar metas que antes eram dos atores.

Ao responder essas questões um novo Diagrama de Atores é produzido com o ator sistema, as dependências, metas, tarefas e recursos antigos são revisados e novos podem ser adicionados.

Em seguida, os mesmos procedimentos são realizados nos Requisitos Iniciais, a Modelagem de Metas e Planos, também é realizada nessa fase para o ator sistema e com isso um novo Diagrama de Metas é gerado.

c. Projeto Arquitetural

Define a arquitetura global do sistema em termos de subsistemas (atores) interconectados através de suas dependências para troca de recursos e realização de tarefas. Esta fase consiste de três passos (BRESCIANI et al., 2004):

1. Selecionar o estilo arquitetural, geralmente para isso é utilizado o *NFR Framework* (CHUNG et al., 1999). É realizada a inclusão de novos atores no sistema de acordo com o estilo arquitetural escolhido e feita a redistribuição das sub-metas para esses atores;
2. Identificar as capacidades (*capabilities*) necessárias para os atores desempenharem suas metas e planos. As capacidades podem ser identificadas a partir do diagrama de atores estendido, na qual cada relação de dependência pode dar lugar a uma ou mais capacidades desencadeadas por eventos externos;
3. Definir um conjunto de agentes e atribuir a eles uma ou mais das diferentes capacidades encontradas. A definição dos agentes e quais capacidades ele vai possuir dependerá da arquitetura escolhida e de como o projetista pensa no sistema em termos de agentes.

No final dessa fase, tem-se como resultado o Diagrama Arquitetural do sistema.

d. Projeto Detalhado

O funcionamento dos agentes é especificado. As metas, crenças e capacidades, assim como a comunicação entre os agentes são especificadas em detalhes. Na parte prática dessa fase, é escolhida a plataforma e a linguagem de programação para os agentes. Pode ser utilizada uma extensão da UML, a AUML, para a representação dos diagramas de classe. Ocorre também a elaboração de diagramas de atividades, sequência e colaboração.

Ao fim dessa fase, é produzido o diagrama detalhado do sistema com a

representação dos agentes; e também os diagramas de atividades, sequência e colaboração que representam a comunicação e as capacidades dos agentes.

e. Implementação

Consiste na geração de código, de acordo com as especificações do Projeto Detalhado.

Basicamente, é realizado um mapeamento entre os conceitos do Tropos e os elementos da plataforma de implementação escolhida. Esse mapeamento gera uma estrutura para a implementação, onde o código é adicionado utilizando a linguagem de programação suportada pela plataforma.

3 Abordagens adotadas para sistemas educacionais baseados na Web Semântica

Neste capítulo são apresentados os trabalhos correlacionados e às abordagens adotadas nos Sistemas Educacionais Baseados na *Web* Semântica.

3.1 Arquitetura de referência

Heitmann e colegas (2009) realizaram um estudo o qual analisaram diversos sistemas desenvolvidos para a *Web* Semântica. Nesse estudo foi observado que a maioria dos sistemas para a Web semântica são compostos pelos mesmos componentes arquiteturais, os quais serão descritos a seguir (Figura 10):

- Interface com o usuário: conhecida como interface do portal ou *view*. Proporciona uma interface que permite ao usuário humano acessar o sistema e visualizar os dados;
- Interface com os dados: conhecida também como “adaptador de dados” ou “provedor de acesso”. Proporciona uma interface necessária para que a lógica da aplicação acesse fontes de dados locais ou remotas, tanto remota fisicamente como administrativa e organizacional;
- Camada de persistência: proporciona para a aplicação a persistência dos dados. O acesso a esta camada é feita através da Interface de dados;
- Serviços de Integração: conhecido também como camada de integração, agregação ou mediação. Proporciona meios para endereçamento estrutural, sintático ou semântico heterogêneo dos dados, causados por acessar dados de múltiplas fontes utilizando diversos tipos de formato, esquema ou estrutura. O resultado desejado é a visão homogênea dos dados para a aplicação;
- Mecanismo de Busca: conhecido também como mecanismo ou interface de consulta. Proporciona meios para executar busca por conteúdos, estrutura ou características especifica de dados em um domínio. Podem ser providas interfaces para humanos, agentes ou ambos;
- *Crawler*: implementa a descoberta automática e recuperação de dados. Requerido se dados necessitam ser encontrados e acessados em domínio específico antes de serem integrados;
- Interface de Anotação: permite ao usuário inserir novos dados, editar os existentes e importar ou exportar dados.

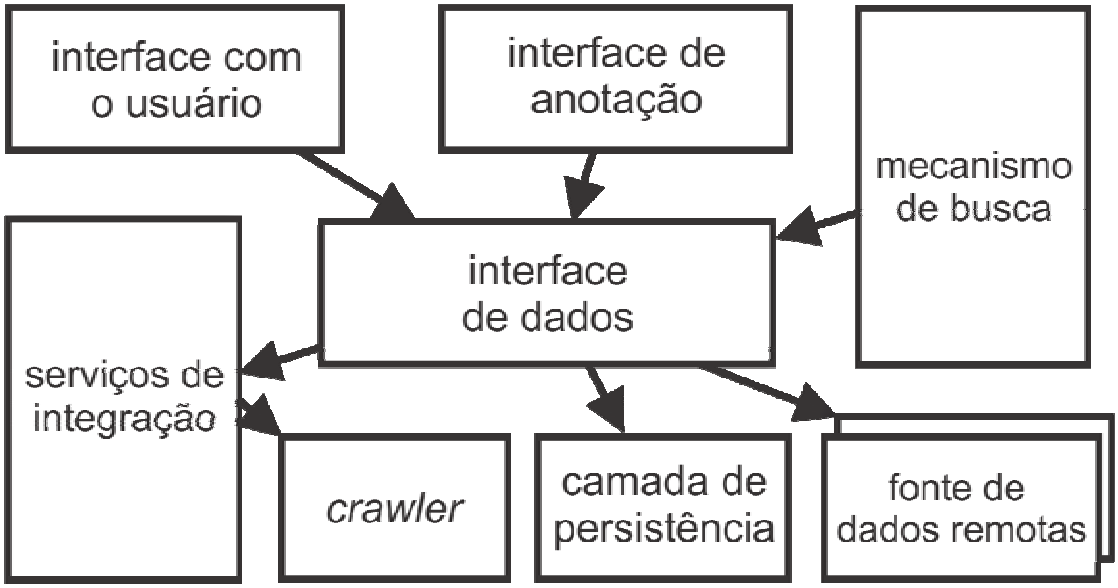


Figura 10: Arquitetura de referência para sistemas *Web* semântico

Fonte: HEITMANN et al (2009)

De acordo com Anderson (2004) um Sistema Educacional *Web* Semântico deve dispor de três aspectos fundamentais: mecanismos que permitam a efetiva capacidade de armazenamento e recuperação dos conteúdos; agentes inteligentes que incorporem a capacidade humana para efetuarem as tarefas que facilitem as atividades dos diversos atores envolvidos no processo de ensino, promovendo a melhor capacidade de ensino, aprendizado e recuperação dos conteúdos; e capacidade de recursos na *Web* para suportar, estender e expandir as comunicações dos atores envolvidos no processo de ensino.

Bittencourt e colegas (2009) propõem um modelo de referência, apresentado na Figura 11, para SWBES. No modelo, os sistemas SWBES estão envolvidos por *software* e seres humanos da seguinte maneira: o *software* é o sistema educacional, responsável por prover informações de acordo com o contexto do ensino e do outro lado o ser humano, que pode assumir diversos papéis, como professor, colaborador, autor, aprendiz etc. Uma visão mais ampla pode ser apresentada considerando os seguintes componentes: os Papéis (*Roles*) que representam os humanos; Ambiente de Interfaces (*Interface Environment*); as fontes educacionais; os componentes correspondentes ao SWBES; e o ambiente externo (*Semantic Web Environment*) concebido utilizando os conceitos da *Web* Semântica (Ambiente *Web* Semântico). Exceto o componente correspondente aos papéis, os demais estão de acordo com as arquiteturas de referência apresentadas, em especial com a proposta de Anderson (2004).

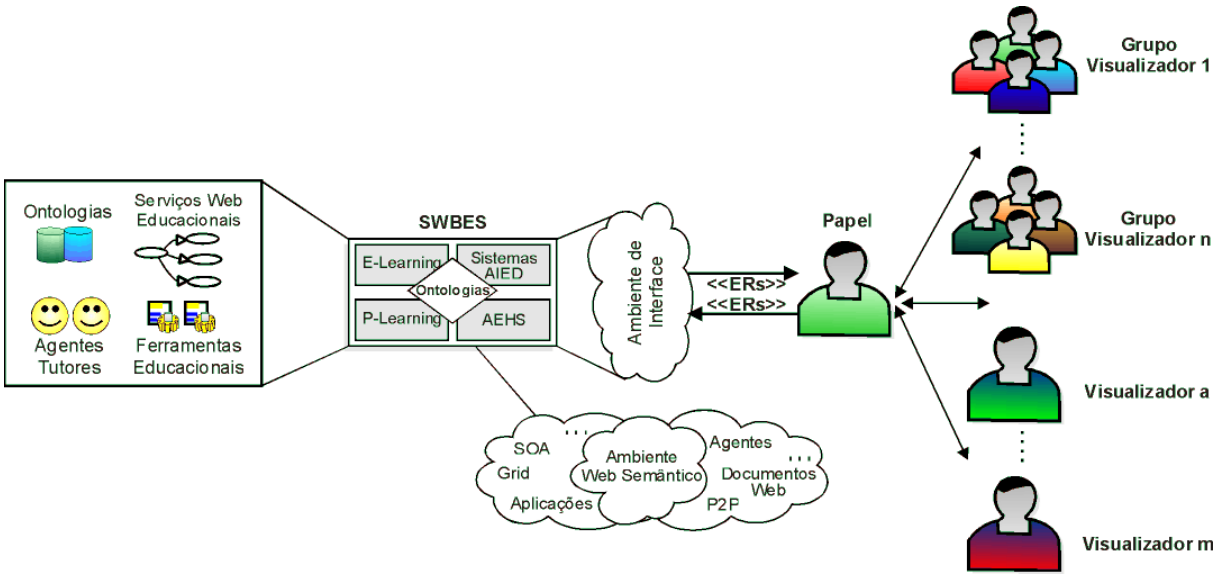


Figura 11: Modelo de Referência para SWBES

Fonte: BITTENCOURT et. al. (2009)

Nesse modelo, os componentes correspondentes ao SWBES fornecem suporte para vários usuários, guiando-os para atingir suas metas. Assim, ele representa a nova geração de WBES, que objetiva prover diversos serviços através do uso das tecnologias da *Web* Semântica. Bittencourt et al. (2009), define que esse modelo é formado pelos seguintes componentes:

a. Ontologias: garantem a interatividade entre diferentes sistemas e definem os dados de forma semântica. No contexto da WS e Educação, ontologias podem ser aplicadas para resolver diversos tipos de problemas complexos, como representação de conhecimento em sistemas inteligentes, compartilhamento e reuso entre diferentes tipos de aplicações, anotação e busca de objetos de aprendizagem, personalização de conteúdo de ensino, entre outros muitos desafios;
b. Agentes Tutores: têm papel importante nos SWBES, pois ajudam no processo de aprendizagem de diversas formas. Os agentes tutores podem procurar similaridades entre perfis e recomendar materiais de aprendizagem. Outra funcionalidade desses agentes é garantir a interação entre as ferramentas do sistema e o usuário (principalmente alunos), como por exemplo, a descoberta de características de aprendizagem do aluno e a personalização de conteúdo entre diferentes tipos de usuários;
c. Serviços *Web* Educacionais (*Educational SWS*): são usados para prover aos usuários, agentes e ferramentas, acesso aos serviços orientados para cada um deles. Os SWS podem prover várias atividades educacionais diferentes, como

personalizar conteúdo, interfaces, avaliações, colaboração e recomendação;

d. Ferramentas Educacionais (*Educational Tools*): os sistemas de educação a distância podem oferecer diversas ferramentas educacionais para interação. Um SWBES deve prover ferramentas de acordo com o tipo de usuário, o propósito e o dispositivo (celular, PDAs, navegador etc.). Essas ferramentas podem ser classificadas em ferramentas síncronas e assíncronas de comunicação. Também é importante salientar que nos SWBES devem existir ferramentas de administração, as quais permitem configurar o ambiente, associar papéis aos usuários etc;
e. Ambiente de Interface: representa a interface de comunicação entre o SWBES e o papel específico do usuário. O ambiente de interface pode ser um navegador, uma TV digital, celular, PDA, desktops etc;
f. SWBES: representa os vários tipos de sistemas de educação baseados na *Web* Semântica;
g. Ambiente *Web* Semântico: representa o ambiente de interação disponível usado para descobrir, selecionar e chamar recursos da *Web* de acordo com a tecnologia e a descrição da arquitetura semântica.

Considerando os aspectos destacados nas arquiteturas de referência apresentadas, na seção seguinte é apresentada arquitetura adotada neste trabalho.

3.2 Arquitetura proposta para o SWBES

Considerando os componentes apresentados nas arquiteturas de referência, uma proposta de arquitetura é apresentada.

A arquitetura proposta, apresentada na Figura 12, é composta dos seguintes componentes:

- uma interface formada por funcionalidades disponíveis em ferramentas de Ensino Baseado na *Web*;
- um modelo baseado em agentes inteligentes que incorporem a capacidade humana para efetuarem as tarefas que facilitem as atividades dos diversos atores envolvidos no processo de ensino promovendo a melhor capacidade de ensino, aprendizado e recuperação dos conteúdos;
- mecanismos que permitam a efetiva capacidade de armazenamento e recuperação dos conteúdos;
- capacidade de obtenção dos recursos externos na *Web* para suportar, estender e expandir as comunicações entre os atores envolvidos no processo de ensino.

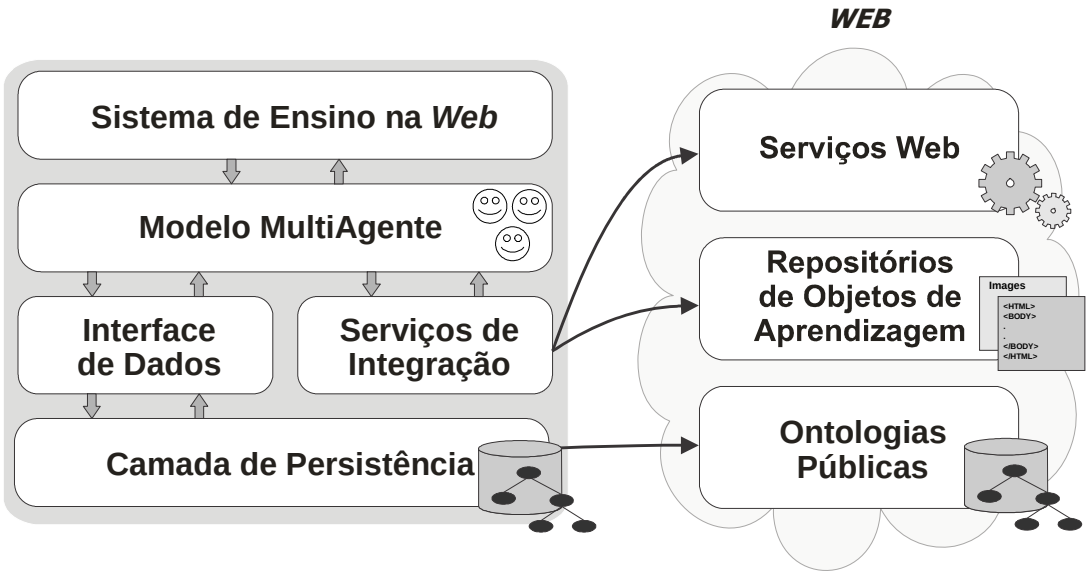


Figura 12: Arquitetura proposta para SWBES

3.2.1 Componentes Arquiteturais

Os componentes arquiteturais definidos para o SWBES proposto foram baseados na análise de sistemas educacionais baseados na *Web* e nos modelos de referência apresentados. Foi definido que deve existir um componente de interação com o usuário, um componente que terá as regras dessas interações com o usuário (regras de negócio), um modelo de dados para armazenar as informações e um componente responsável por acessar informações externas a ferramenta.

3.2.1.1 Sistema de ensino na *Web*

O componente de interação com o aluno deve ter ferramentas de ensino que interajam com ele. Assim, qualquer ferramenta atual de educação baseado na *Web* pode ser adaptada para fazer o papel desse componente. É necessário que o sistema de educação baseado na *Web* escolhido para a interação com o aluno disponha de ferramentas que possam apresentar materiais de estudo. Esses materiais podem estar em qualquer formato que aluno possa ler, ver ou ouvir; ou seja, o sistema de educação baseado na *Web* escolhido deve suportar algum tipo de mídia para exibir informações. Pode ser ferramenta de materiais disponíveis para *download* ou textos, imagens e *link*s; pode ser uma ferramenta de vídeo-aula, um *podcast* ou qualquer outro tipo de mídia que alimente o aluno com informações.

Também deve existir, nesse sistema de educação, alguma ferramenta que possa

avaliar o aluno. Essa ferramenta é de extrema importância, pois é a partir dela que os agentes irão recuperar informações sobre o desempenho do aluno e decidir se o aluno está acompanhando bem o curso a distância ou se precisa de alguma intervenção para ajudá-lo.

Então, desde que a ferramenta de educação escolhida tenha essas duas formas de interação com o aluno (oferecer informações e avaliar o aluno), é possível utilizá-la na arquitetura proposta para o SWBES. A informação gerada através da ferramenta de avaliação será lida por um sistema multiagentes que poderá alterar a forma de apresentar os conteúdos na ferramenta.

3.2.1.2 Modelo Multiagentes

O componente correspondente ao modelo multiagentes da arquitetura proposta implementa as regras dessas interações com o aluno (regras de negócio). Para isso o componente deve ter acesso a *Web* para realizar buscas de novos objetos de aprendizagem, acesso às ontologias que fazem parte da arquitetura proposta e acesso às informações que o sistema de educação baseado na *Web* irá gerar.

Considerando os requisitos deste componente, a partir da modelagem, descrita no capítulo 4 , chegou-se a definição de um modelo multiagentes composto por cinco papéis principais que farão parte dessa arquitetura. Os papéis definidos são: Controlador, Tutor, Avaliador, Buscador e Publicador.

O papel Controlador tem a função de coordenar qual agente deve ser iniciado e quando deve ser iniciado. Para isso ele deve conhecer cada agente e qual serviço cada um oferece. Também deve possuir um mecanismo que determina quando iniciar um processo de avaliação do aluno.

O papel Tutor é o responsável pela interação com o aluno. Ele deve apresentar ao aluno qual conteúdo ele pode acessar em cada momento do curso. Para isso ele deve dispor das informações do aluno, sabendo em qual nível de conhecimento esse aluno se encontra e quais materiais estão disponíveis para serem exibidos para esse aluno. O Tutor tem que conseguir atuar na ferramenta de ensino modificando a sequência dos materiais de aprendizagem que serão exibidos no curso para aquele aluno específico.

O papel Avaliador deve conter uma ou mais formas de avaliar o aluno. Ele deve conseguir acompanhar o andamento do aluno através de acesso aos resultados das avaliações realizadas por esse aluno no sistema de educação baseado na *Web*. A partir dessa avaliação, o Avaliador deve conseguir acessar e atualizar a representação do conhecimento do aluno, que nessa arquitetura será mantida em uma ontologia chamada ontologia de Perfil.

Com o resultado da avaliação realizada pelo Avaliador, o papel Buscador deve ter

capacidade de acessar materiais de aprendizagem por meio da camada de serviços. Também foi definido o uso do padrão LOM para que exista interoperabilidade entre os sistemas.

Os objetos de aprendizagem obtidos pelo Buscador estarão disponíveis para o Publicador que deve conhecer a estrutura de materiais do curso para identificar em qual ferramenta esse novo material será introduzido. Por exemplo, se o Buscador encontrar um texto, o Publicador deve inserir uma entrada na ferramenta de texto, se o Buscador encontrar um vídeo, uma nova entrada na ferramenta de exibição de vídeo deve ser realizada pelo Publicador.

3.2.1.3 Serviços de Integração

O componente Serviços de Integração tem a responsabilidade de prover acesso dos agentes a repositórios externos e informações disponíveis na *Web,* que possam ajudar o sistema a entregar informações diversas para o aluno. Toda a complexidade de acesso, protocolos, chamadas a serviços, devem estar implementadas nessa camada, objetivando manter os agentes focados nas suas funções principais.

Outra grande vantagem de manter uma camada de serviços é a manutenibilidade do sistema. As tecnologias evoluem rapidamente, novos protocolos são implementados, novos tipos de serviços são oferecidos e mantendo a complexidade desse acesso isolado em apenas uma camada, o trabalho de atualização será pontual e não interferirá nos demais mecanismos da arquitetura.

3.2.1.4 Interface de Dados

Com finalidade semelhante ao componente Serviço de Integração, o componente Interface de Dados tem a função de isolar a complexidade de acesso aos dados, tanto da ontologia como do banco de dados.

Assim caso seja necessário a troca do gerenciador de banco de dados ou mudar a tecnologia de acesso às ontologias, os demais componentes da arquitetura não sofrerão nenhuma alteração.

O componente Interface de Dados deve conter toda a implementação dos métodos de busca e atualizações das informações, bem como os métodos de conexão. Deve ainda responder para a aplicação somente com estruturas de dados definidas internamente.

3.2.1.5 Camada de Persistência

O componente Camada de Persistência é o local onde estão armazenadas todas as informações que alimentam o sistema. Nesse componente ficam as ontologias e os dados do banco de dados. As ontologias são usadas para anotar os materiais de aprendizagem e codificar os conceitos do domínio. No banco de dados ficarão armazenadas as informações pertencentes aos usuários do sistema: dados pessoais, senha, perfil, pontuação e nível.

Em relação às ontologias, são requeridas para essa arquitetura, a definição de três ontologias (Figura 13), sendo duas ontologias internas da aplicação e uma ontologia de conceitos (Ontologia de Domínio), que deve ser pública. A ontologia pública deve tratar do assunto que será trabalhado no curso, essa ontologia é chamada de ontologia de domínio. As ontologias internas são usadas para armazenar os materiais do curso (Ontologia de Materiais de Aprendizagem) e armazenar o desempenho do aluno (Ontologia de Perfil).

Na Ontologia de Domínio estão codificados os conceitos pertencentes à área específica de conhecimento a que o sistema é destinado. Essa ontologia serve para mapear o perfil de cada aluno de acordo com sua evolução no aprendizado. Por meio da relação entre os Objetos de Aprendizagem e os conceitos desta ontologia é possível verificar quais conceitos estão fixados e quais conceitos necessitam de mais estudo.

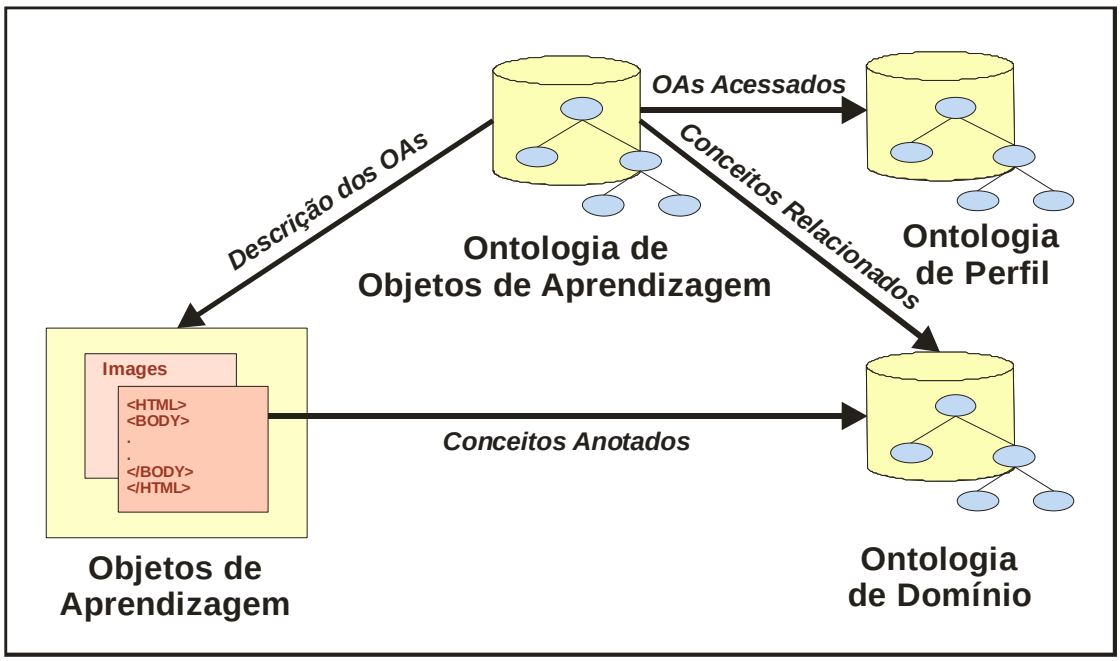


Figura 13: Relacionamento entre ontologias

Na ontologia de materiais de aprendizagem estão as descrições e anotações dos materiais de aprendizagem utilizados no sistema. Os objetos de aprendizagem descritos nessa ontologia estão relacionados a um ou mais conceitos da ontologia de domínio. Assim,

é possível saber quantos e quais objetos de aprendizagem explicam ou utilizam um determinado conceito da área de interesse (Ontologia de Domínio). Isso se dá por meio de relacionamentos entre recursos das ontologias e anotações dos materiais de aprendizagem.

As ontologias de perfil armazenam as informações de cada usuário no que se refere à sua evolução em um determinado curso (área de interesse). Com essas ontologias é possível obter as informações necessárias para mapear o perfil de cada aluno, conhecendo seu desempenho em cada curso realizado. Por exemplo, no perfil constarão os materiais de aprendizagem acessados pelo aluno e seu grau de sucesso em cada tópico, construindo assim um caminho percorrido (trilha).

3.2.1.6 Repositórios de objetos de aprendizagem

A integração com repositórios externos é o ponto de extensão da arquitetura. Esses repositórios podem estar em outras universidades, publicados em servidores remotos, que os agentes consultarão para obter novos materiais de aprendizagem.

Além disso, o acesso a serviços *Web* podem servir de base para os agentes buscarem outros materiais de aprendizagem ou textos que não estão em repositórios oficiais, mas que possam ajudar o aluno, como por exemplo, conceitos na Wikipedia[6].

Outros serviços na Internet também podem ser utilizados no ambiente de ensino, como a utilização de um serviço de mapas para identificar alunos que moram perto e sugerir estudos presenciais para esses alunos. Esses serviços devem ser acessados através da camada de integração.

---

[6] Wikipedia - http://pt.wikipedia.org

4 Modelagem Multiagente

Neste capítulo são apresentados os passos executados na modelagem da arquitetura multiagentes. Na modelagem é apresentado o desenvolvimento das fases da metodologia Tropos com o objetivo de definir uma organização arquitetural multiagentes que atenda os requisitos do contexto educacional apresentado no Capítulo 3 . Ao final de cada fase da metodologia Tropos é produzido um modelo ou diagrama que será refinado nas próximas fases até a conclusão do sistema. Os diagramas gerados durante o desenvolvimento da metodologia Tropos foram construídos utilizando a ferramenta TAOM4E (TAOM4E, 2010). As fases da modelagem são apresentadas em detalhes a seguir.

4.1 Modelagem

O objetivo da definição da arquitetura é identificar as tarefas que possam ser executadas por agentes, automatizando o máximo possível o apoio ao aluno no processo de aprendizagem. A adoção da metodologia Tropos permite identificar quais agentes devem ser definidos para que os requisitos sejam alcançados. Neste sentido, os passos da modelagem Tropos, são os seguintes: Descrição do sistema, levantamento dos Requisitos Iniciais, Requisitos finais, Projeto Arquitetural e Projeto Detalhado.

4.1.1 Descrição do sistema

Em um curso *online* os alunos têm acesso a diversos materiais para leitura que podem auxiliar na realização das atividades de avaliação. Cada ferramenta de educação na *Web* possui um conjunto de ferramentas para avaliar e interagir com o aluno, as ferramentas mais comuns são fóruns, questionários, bate-papo e listas de discussão. Os materiais disponibilizados para os alunos podem ser compostos por textos, figuras, vídeos, áudio e *link*s; organizados em módulos.

Os materiais que compõem os cursos são criados por pesquisadores, professores ou especialistas. Posteriormente, disponibilizados na *Web* ou por editoras em forma de livros, apostilas e artigos. Os alunos e professores podem acessar todos os materiais disponíveis nas livrarias e Internet a qualquer tempo fora da ferramenta educacional. Na construção de um curso, o professor é o responsável por selecionar quais serão os materiais que farão parte do curso e serão publicados na ferramenta para o aluno.

A publicação do material didático é realizada pelo *Designer* Instrucional, profissional

treinado para formatar o conteúdo selecionado pelo professor para que seja aproveitado de forma efetiva pelo aluno. O *Designer* faz uso de uma ferramenta utilizada para a publicação.

O professor é também o responsável pela criação das atividades que serão utilizadas para avaliar os alunos. A partir da definição dos materiais que farão parte do curso, o professor deve desenvolver atividades que ajudem os alunos a fixar o conteúdo estudado. Essas atividades serão a fonte de avaliação dos alunos, indicando ao professor, em quais assuntos abordados o aluno apresenta dificuldade.

Os cursos só podem ser acessados por pessoas autorizadas através de um cadastro prévio. Neste contexto, é possível identificar os alunos que acessaram os materiais e atividades, visando acompanhar o desenvolvimento.

4.1.2 Requisitos Iniciais

A fase de Requisitos Iniciais consiste em definir o ambiente onde o sistema será inserido e identificar as partes interessadas (*stakeholders*), suas intenções (*goal*s) e seus relacionamentos. Esse ambiente é obtido por meio da análise da definição do escopo, definido na seção anterior. Durante a análise são identificados e listados todos os atores juntamente com suas respectivas intenções. Na Tabela 1 é apresentado o resultado desta análise, com os atores Aluno, Professor, Secretaria, *Designer* Instrucional e Autor, e suas respectivas intenções.

O ator Aluno possui como intenções principais concluir o curso e obter máximo aproveitamento sobre o que está sendo ensinado. O ator Professor tem a responsabilidade de selecionar materiais e preparar as atividades com a finalidade de disponibilizar um curso. Outra intenção é adaptar constantemente os aspectos pedagógicos visando sempre promover o aprendizado do aluno.

Tabela 1: Relação dos interessados e suas intenções (Requisitos Iniciais)

Ator	**Intenções**
Aluno	Concluir o curso Obter máximo aproveitamento do curso
Professor	Disponibilizar Curso Promover o aprendizado do aluno Adaptação dos aspectos pedagógicos
***Designer* Instrucional**	Publicar Materiais
Autor	Produzir Materiais
Secretaria	Controlar acesso

O ator *Designer* Instrucional é o responsável por receber o conteúdo selecionado pelo professor, formatá-lo para facilitar a leitura e tornar disponível na ferramenta educacional, tarefa representada pela intenção “Publicar Materiais”. A intenção “Produzir Materiais” engloba todas as atividades do ator Autor, desde a escrita até a publicação e disponibilização. E por último, o ator Secretaria é responsável pela intenção de controlar o acesso à ferramenta e manter os cadastros atualizados.

Após definidos os interessados e suas intenções, uma nova análise foi realizada para identificar quais planos os atores deverão realizar para atingir suas intenções (Tabela 1). Também foram identificados os relacionamentos entre os atores para a realização dos planos. Na Tabela 2 são apresentadas apenas aquelas atividades que possuem relacionamento entre atores.

Tabela 2: Definição dos planos e relacionamentos entre os atores (Requisitos Iniciais)

Ator	**Planos**	**Relacionamentos**
Aluno	Ler Materiais Responder Atividades Enviar Dúvidas Consultar Materiais	*Designer* Instrucional Professor Professor Autor
Professor	Pesquisar Materiais	Autor
***Designer* Instrucional**	Receber material	Professor
Secretaria	Solicitar dados do Professor Solicitar dados do Aluno	Professor Aluno

Por exemplo, o ator Aluno para alcançar suas intenções, realiza os seguintes planos: “Ler material de estudo”, “Responder Atividades”, “Enviar Dúvidas” e “Consultar Materiais”. Na realização desses planos há dependência do Aluno com os demais atores, o Professor, Autor e *Designer* Instrucional.

O Professor deve “Pesquisar Materiais” para disponibilizar e manter o curso. O material selecionado pelo professor é encaminho ao *Designer* Instrucional que deve receber o material e formatá-lo. Para manter os cadastros atualizados, a Secretaria deve executar os planos de solicitar os dados dos usuários.

Utilizando a ferramenta TAOM4E (TAOM4E, 2010), as intenções da Tabela 1 foram transcritas como metas no diagrama de atores (Figura 14). Nesse diagrama são demonstradas graficamente as metas, metas-soft, planos e as dependências entre os atores. Também foram identificados os recursos gerados pelos mesmos.

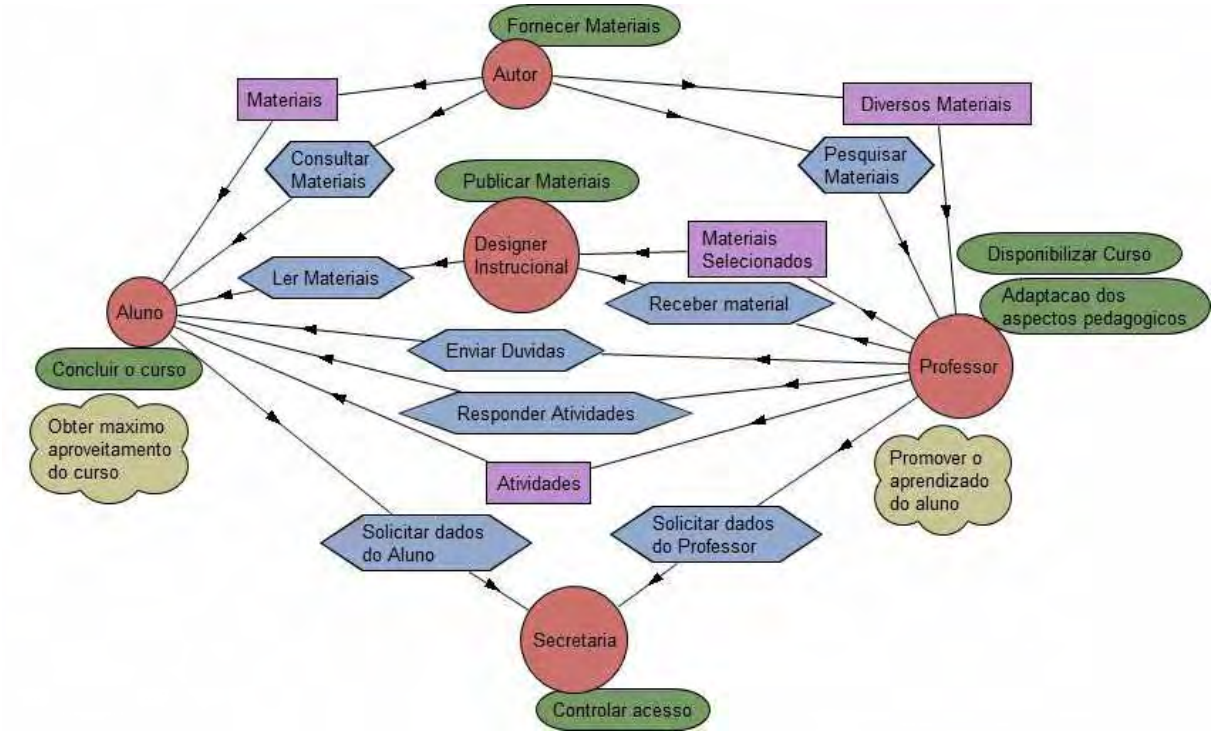


Figura 14: Diagrama de Atores (Requisitos Iniciais)

A partir da modelagem do Diagrama de Atores, o próximo passo é a construção do diagrama de metas, que consiste em decompor cada meta que o ator possui em sub-metas, se possível. Nesta fase podem ser encontradas novas metas e também planos. Depois é realizada a análise meio-fim para definir quais planos serão os meios para alcançar as metas e em seguida uma análise de contribuição, na qual são identificados os planos e metas que contribuem para os requisitos não funcionais do sistema (metas-soft).

No diagrama de metas do ator Aluno (Figura 15) pode ser observado que a principal meta do Aluno é “Concluir o curso”. Para atingir essa meta, o aluno deve executar o plano “Responder Atividades”, atividades essas que foram criadas pelo professor e estão representadas no diagrama pelo recurso “Atividades”.

O plano “Responder Atividades” é suficiente para satisfazer a meta “Concluir o curso”, desde que o aluno responda corretamente todas as atividades. Entretanto, o aluno possui outra meta chamada “Obter conhecimento”, a qual contribui para a meta principal e para a meta-soft “Obter máximo aproveitamento do curso”. Essa meta deve realizar os planos “Ler materiais”, “Consultar Materiais” ou “Enviar Dúvidas”.

Os materiais disponibilizados pelo professor, através do *Designer* Instrucional, são acessados pelo Aluno no plano “Ler materiais”. O acesso do aluno a outros materiais está representado no plano “Consultar Materiais”, o qual representa a ação do aluno em procurar qualquer material de referência fora da ferramenta educacional.

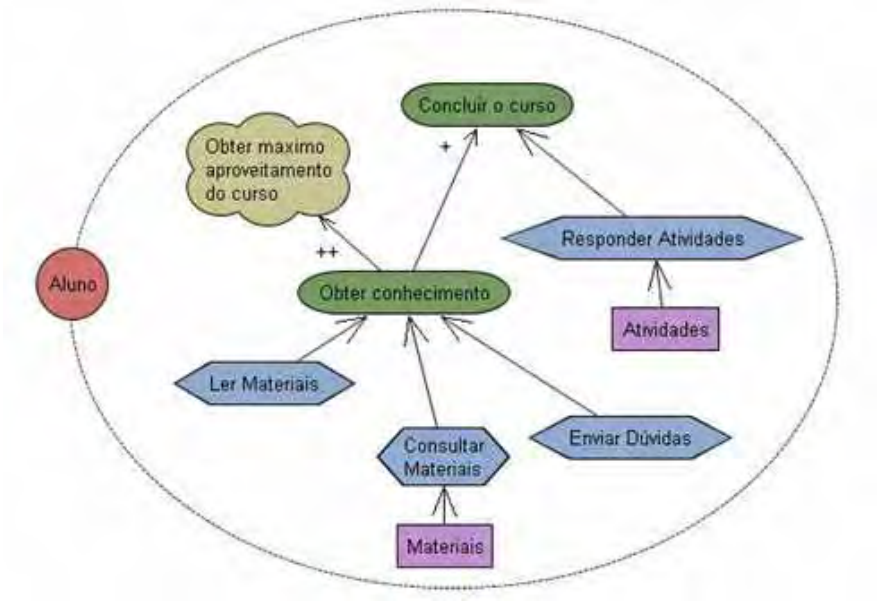


Figura 15: Diagrama de Metas do Ator Aluno (Requisitos Iniciais)

No diagrama de metas do Professor (Figura 16) é apresentada a decomposição das metas principais: “Disponibilizar Curso” e “Adaptação dos aspectos pedagógicos”. Também foi identificada outra meta denominada “Esclarecer Dúvidas”, que será executada quando o aluno envia uma dúvida para o professor.

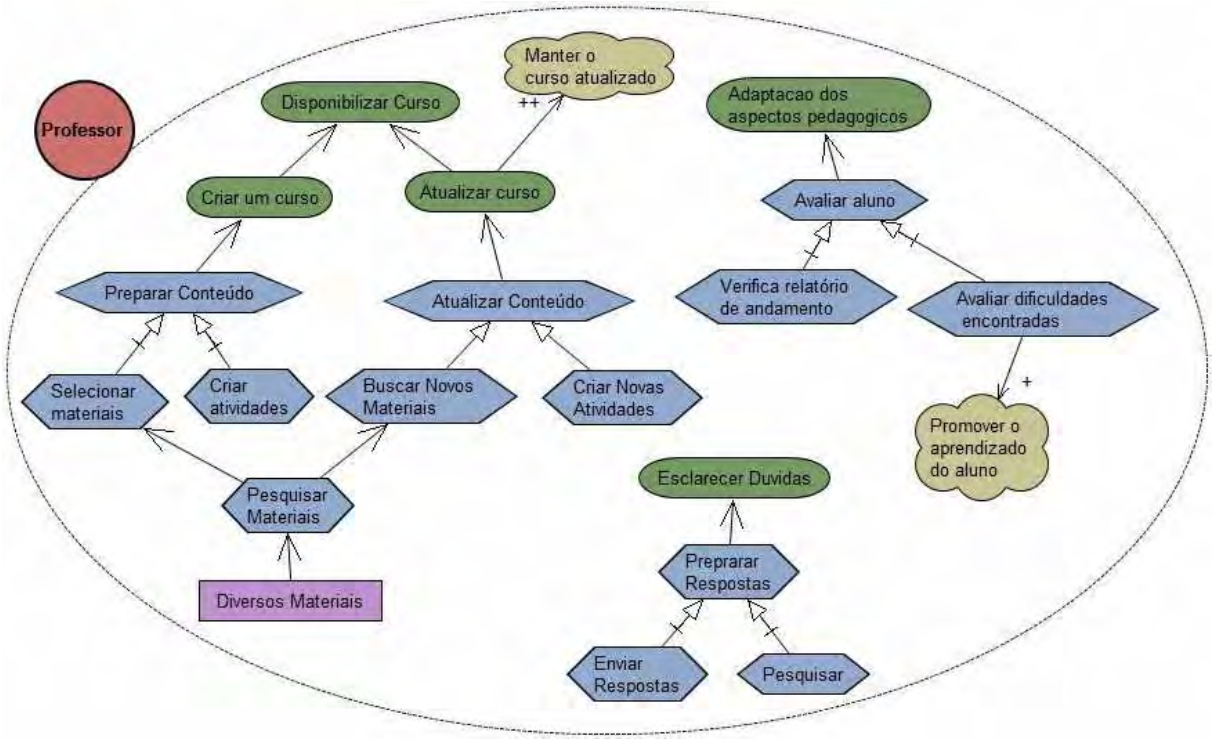


Figura 16: Diagrama de Metas do Ator Professor (Requisitos Iniciais)

A meta “Disponibilizar um curso” foi decomposta em duas outras metas: “Criar um curso” e “Atualizar curso”. Na meta “Criar um curso”, o Professor executa planos visando pesquisar materiais existentes, selecionar dentre esses materiais quais farão parte do curso e criar as atividades. Com o material selecionado e as atividades criadas, o curso pode ser disponibilizado.

A meta “Atualizar curso” pode ser realizada constantemente para contribuir com a meta-*soft* “Manter o curso atualizado”. Essa meta consiste na realização dos planos: “Buscar novos Materiais” e “Criar novas Atividades”.

A meta “Adaptação dos aspectos pedagógicos” é atingida através do plano “Avaliar aluno”. O professor deve avaliar como está sendo o aprendizado do aluno e identificar se é necessário mudar a abordagem de ensino adotada.

A meta “Esclarecer Dúvidas” representa a intenção do professor em responder as dúvidas enviadas pelos alunos. O meio para atingir essa meta é realizar o plano “Preparar Resposta” que consiste em “Pesquisar” e “Enviar respostas”.

Na Figura 17 é apresentado o Diagrama de Metas do *Designer* Instrucional, onde a meta principal é “Publicar Materiais”, a qual é alcançada através dos planos “Receber Materiais” do professor e “Formatar” para disponibilizá-los.

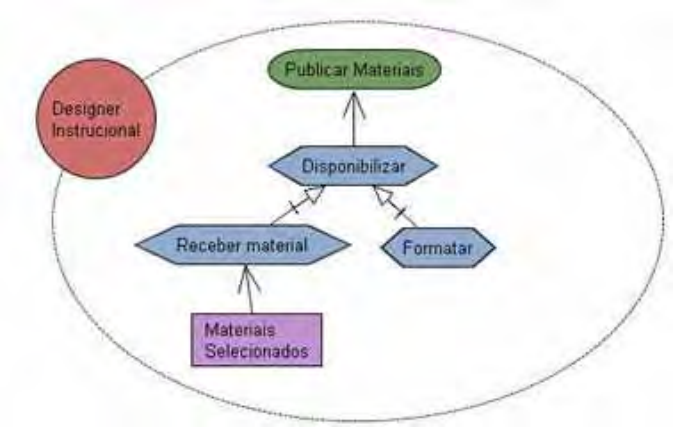


Figura 17: Diagrama de metas do *Designer* Instrucional

4.1.3 Requisitos Finais

Após concluir a definição do ambiente na fase anterior, é realizada a identificação dos requisitos do sistema. O primeiro passo da fase de Requisitos Finais é inserir um ator para representar o sistema, esse ator foi denominado SWBES. Com o objetivo de automatizar as atividades realizadas pelos atores identificados na fase anterior, é realizada uma análise das intenções e relacionamentos desses atores considerando agora o ator

SWBES, em que os planos e metas antes descritos podem ser atribuídos para esse novo ator.

Na Tabela 3 é apresentado o resultado da redefinição das intenções dos atores previamente definidos com o novo ator SWBES. O novo ator SWBES recebeu as intenções que antes foram associadas à Secretaria, ao Professor e ao *Designer* Instrucional e algumas novas intenções identificadas para atender os requisitos. As intenções atribuídas para o ator SWBES são voltadas para a automação do processo de aprendizagem, visando o monitoramento constante do aluno e suas atividades para prover uma maior eficiência no seu aprendizado.

O ator sistema SWBES adquiriu as responsabilidades da Secretaria no sentido de controlar os acessos ao sistema, por isso foram atribuídas as intenções “Controle de Acesso” e “Segurança”, no intuito de controlar os acessos dos usuários para fins de personalização. A intenção “Promover o Aprendizado” tem com objetivo, monitorar os alunos no sentido de avaliá-los e analisar suas dificuldades para que possam receber ajudas e dicas. O sistema também pode iniciar fóruns de discussão, publicar sugestões de conteúdos para apoiá-los, ou até mesmo no caso de um bom desempenho dos alunos, indicá-los para outros alunos como possíveis amigos de forma a auxiliar os que têm dificuldades.

Tabela 3: Relação dos interessados e suas intenções (Requisitos Finais)

Ator	**Intenções**
Aluno	Concluir o curso Máximo aproveitamento do curso
Professor	Criar Cursos
Autor	Fornecer Materiais
SWBES	Promover o aprendizado Qualidade Controle de Acesso Segurança

A intenção do professor de criar o curso e definir quais serão os materiais a serem apresentados para o aluno passou a ser de responsabilidade do ator SWBES. A relação do sistema com o ator Autor foi definida como sendo o acesso aos repositórios externos de materiais, nos quais o sistema busca por novos conteúdos para sugerir aos alunos com dificuldades.

A partir dessas novas intenções foram identificados os principais planos dos atores envolvidos com o sistema (Tabela 4).

Tabela 4: Novos planos incluindo o ator sistema (Requisitos Finais)

Ator	Planos	Relacionamentos
Aluno	Acessar Materiais Responder Atividades	SWBES SWBES
Professor	Cadastrar cursos	SWBES
SWBES	Propor atividades Pesquisar novos materiais Oferecer conteúdo Monitorar aluno	Aluno Autor Aluno Aluno

Os principais planos dos alunos são “Acessar os Materiais” para adquirir os conceitos que serão usados posteriormente para “Responder as Atividades” e consequentemente concluir o curso.

O ator Professor faz o cadastro dos cursos de modo a permitir o acesso aos alunos e possibilitar que os mesmos possam fazer a inscrição.

Os planos “Propor atividades”, “Monitorar aluno”, “Oferecer conteúdo” e “Pesquisar novos materiais” foram criados para atingir a intenção “Promover o aprendizado”. O plano “Propor atividades” consiste em disponibilizar atividades de avaliações individuais ou em grupo. O plano “Monitorar aluno” envolve observar as interações do aluno com o sistema para avaliá-lo. Os conteúdos disponibilizados pelo sistema serão executados através do plano “Oferecer conteúdo”. Com a intenção de manter o curso atualizado e oferecer conteúdo adicional, o SWBES faz buscas por conteúdos, representado pelo plano “Pesquisar novos Materiais”.

Novamente, utilizando a ferramenta TAOM4E (TAOM4E, 2010) foi gerado o Diagrama de Atores da fase de Requisitos Finais (Figura 18). Nesse diagrama foram redefinidos os relacionamentos entre os atores definidos no Diagrama de Atores (Figura 14). No diagrama foram também inseridas as novas metas, planos e recursos foram identificados para atender as novas necessidades do Sistema.

Assim como realizado na fase de Requisitos Iniciais, o próximo passo dessa fase é realizar o refinamento das metas do ator correspondente ao sistema, o SWBES (Figura 19).

A meta principal do ator SWBES é “Promover o Aprendizado”. Essa meta foi decomposta em vários planos e uma sub-meta. A sub-meta “Personalizar Conteúdo” tem por objetivo escolher qual atividade e/ou conteúdo deve ser exibido para o aluno durante o seu progresso no curso. O conteúdo personalizado contribui para as metas-*soft* “Qualidade”, “Controle de Acesso” e “Máximo aproveitamento do curso”.

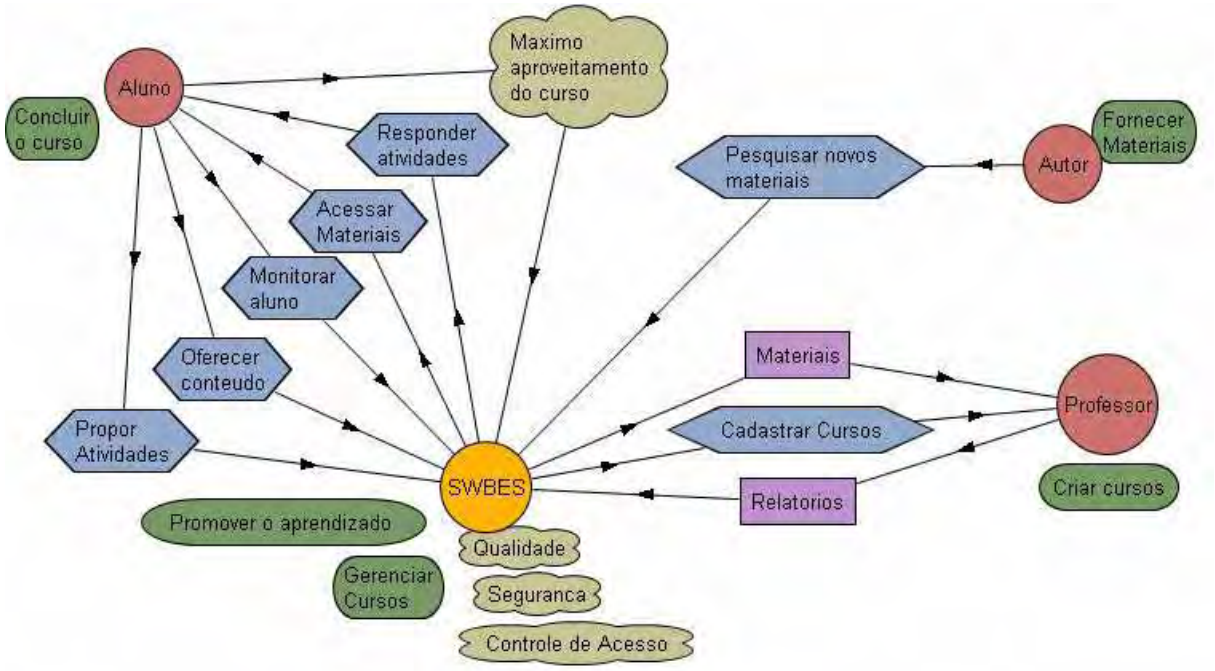


Figura 18: Diagrama de atores - Requisitos Finais

Ao acessar o curso, o aluno é identificado pela ferramenta, seus dados são recuperados e ficam disponíveis para utilização. Durante o andamento do curso, o aluno realiza atividades e através da correção dessas atividades é feito o acompanhando de suas dificuldades. O sistema consegue, a partir dessas avaliações, determinar se o aluno está progredindo e tornando-se apto para subir de nível. Essas avaliações correspondem ao plano “Monitorar Aluno”.

A partir das dificuldades encontradas pelo “Monitorar Aluno”, é executado o plano “Pesquisar novos materiais”. Esse plano foi encontrado a partir da dependência entre Autor e Sistema, e decomposto para automatização do processo de busca. Esse plano foi decomposto em outros planos para buscar repositórios públicos de objetos de aprendizagem, encontrar novos objetos de aprendizagem, anotá-los de acordo com os conceitos existentes no curso e gravá-los.

Os conteúdos são disponibilizados ao aluno através do plano “Oferecer conteúdo”, o qual deve escolher qual o melhor conteúdo será disponibilizado ao aluno, a partir de suas dificuldades. Esse plano também deve escolher entre as atividades cadastradas, aquelas que auxiliem o aluno a fixar seu conhecimento ou a trabalhar o assunto em grupo com outros alunos.

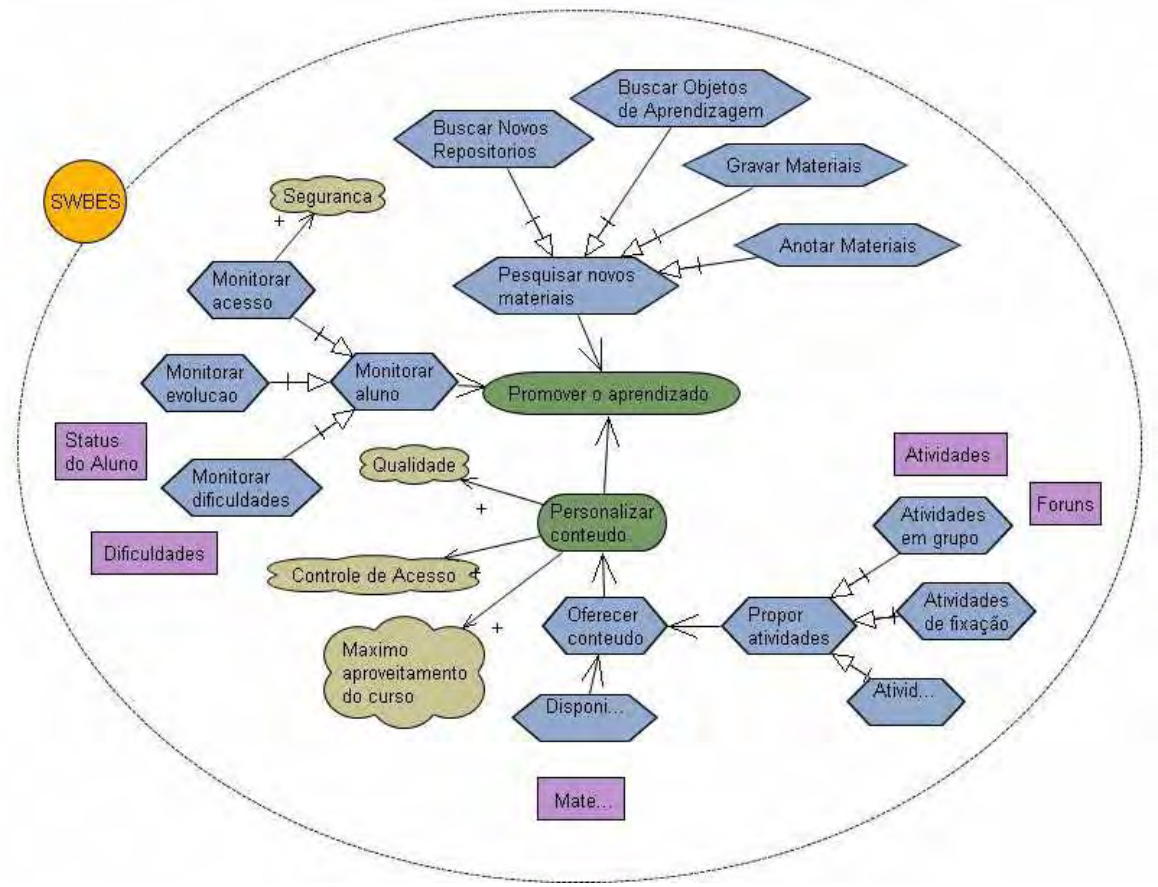


Figura 19: Diagrama de metas Requisitos Finais - Ator SWBES

4.1.4 Projeto Arquitetural

Na fase anterior foram encontradas as funcionalidades do sistema. Agora essas funcionalidades devem ser agrupadas em subsistemas representadas por atores e interconectadas através de suas dependências para troca de recursos e realização de tarefas. A finalidade desta fase é a definição da arquitetura global do sistema. Ela consiste na realização de 3 atividades:

1 - escolher um estilo arquitetural que satisfaça as necessidades do sistema;

2- a partir dos requisitos encontrados nas fases anteriores, agrupá-los em papéis com funcionalidades (ou responsabilidades) específicas;

3 - a partir do estilo escolhido, fazer o mapeamento dos papéis encontrados para os componentes do estilo arquitetural.

4.1.4.1 Identificação do estilo arquitetural

O primeiro passo consiste na escolha do estilo arquitetural e para isso os estilos arquiteturais são analisados em relação aos requisitos não-funcionais (metas-soft) identificados nas fases de requisitos. Existe um conjunto de estilos arquiteturais já definidos baseados em conceitos organizacionais (KOLP, MYLOPOULOS e GIORGINI 2006).

Na Tabela 5 são apresentadas as correlações dos estilos organizacionais com os atributos de qualidade (Previsibilidade, Segurança, Adaptabilidade, Coordenação, Cooperação, Disponibilidade, Integridade, Modularidade, Agregação). As relações são identificadas como contribuições do tipo parcial/positivo(+), suficiente/positivo(++), parcial/negativo(-) e suficiente/negativo(--).

Tabela 5: Catálogo de Correlação

	Prev.	Seg.	Adapt.	Coord.	Coop.	Disp.	Integr.	Mod.	Agreg.
Flat structure	--	--	-			+	+	++	-
Structure-in-5	+	+		+	-	+	++	++	++
Pyramid	++	++	+	++	-	+	--	-	
Join venture	+	+	++	+	-	++		+	++
Bidding	--	--	++	-	++	-	--	++	
Take over	++	++	-	++	--	+		+	+
Arms's length	-	--	+	-	++	--	++	+	
Hierarchical Contracting			+	+	+	+		+	+
Vertical integration	+	+	-	+	-	+	--	--	--
Cooptation	-	-	++	++	+	--	-	--	

Os requisitos não funcionais identificados na fase anterior (Figura 19) são: Qualidade, Segurança, Controle de Acesso e Máximo Aproveitamento do Curso. Esses requisitos foram comparados com os atributos de qualidade associados aos estilos arquiteturais do catálogo (Tabela 5) (KOLP, MYLOPOULOS e GIORGINI 2006) e feita uma avaliação de qual estilo arquitetural possui mais contribuições positivas (valores + e ++ na Tabela 5) para os atributos do sistema. Os atributos de qualidades mais relevantes para o sistema são os atributos de Segurança, Adaptabilidade, Coordenação, Disponibilidade e Integridade. Assim, o estilo arquitetural que atendeu aos requisitos não funcionais identificados foi o *Joint-Venture*[7] (Figura 20).

[7] O estilo *Joint-Venture* envolve um acordo entre dois ou mais parceiros para operar em larga escala e reusar a experiência e conhecimento dos outros parceiros. Estes parceiros se dividem em: um coordenador, cuja finalidade é gerenciar o compartilhamento do conhecimento e dos recursos; parceiros principais que interagem para trocar, prover e receber serviços, dados e conhecimento; e parceiros secundários que suprem serviços ou suportam tarefas para o núcleo da organização. (SILVA, 2008)

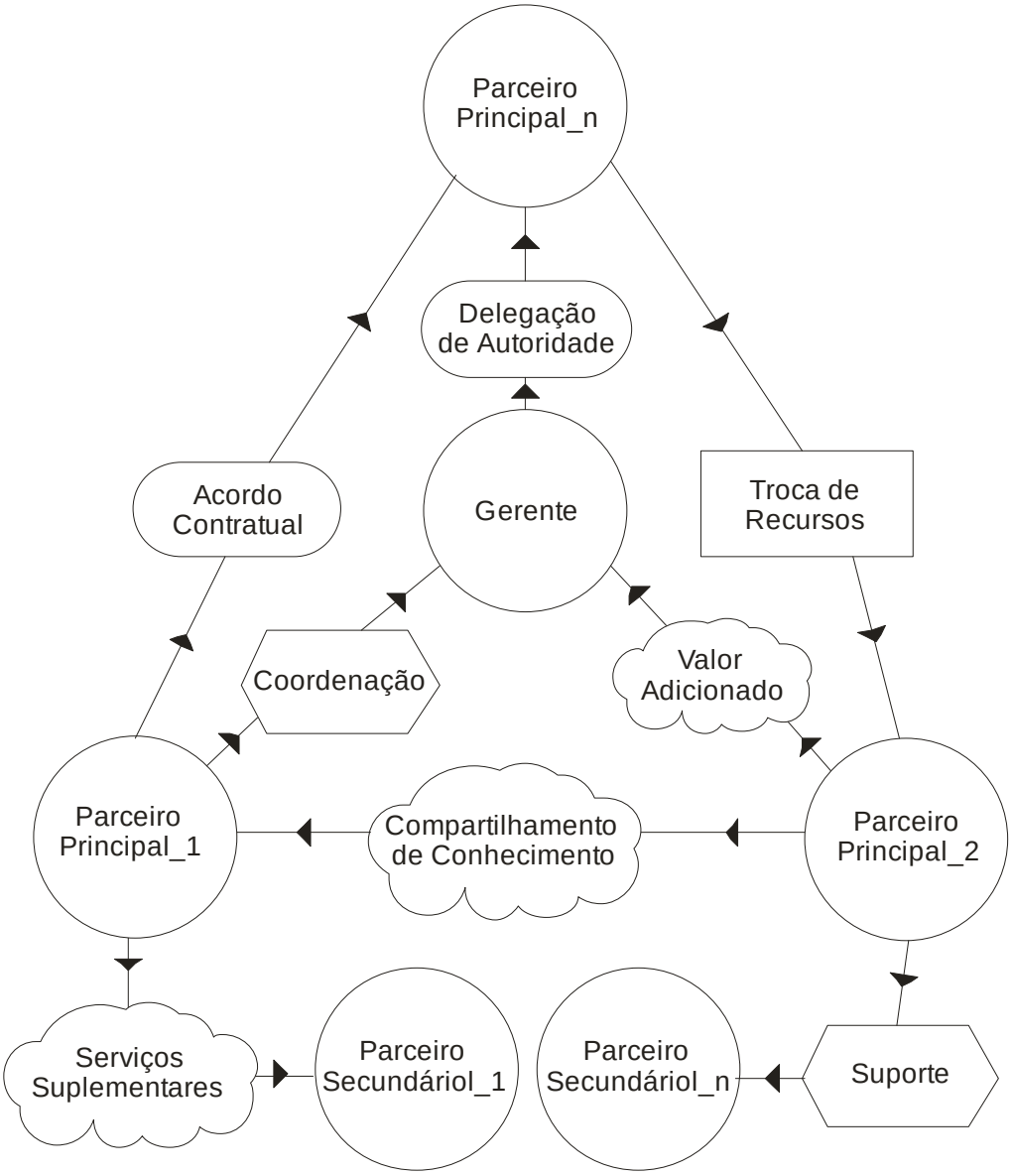


Figura 20: Estilo arquitetural *Joint-Venture*

Fonte: Traduzido de KOLP, MYLOPOULOS e Giorgini (2006)

4.1.4.2 Identificação dos papéis

Analisando o diagrama de metas do SWBES obtido na fase de Requisitos Finais (Figura 19), foi realizado um agrupamento, por responsabilidades, das metas e planos do ator SWBES, resultando na identificação de quatro papéis: Avaliador, Tutor, Buscador e Publicador. Esse agrupamento tem como objetivo a divisão de trabalho do ator SWBES em papéis, onde cada papel fica responsável pela realização de uma ou mais metas e planos. Como por exemplo, o Avaliador ficou responsável pelas metas e planos de avaliação do aluno, enquanto o Buscador pelos planos de busca de materiais.

Cada papel foi especificado segundo seus objetivos, responsabilidades, colaboradores (outros papéis com quem são trocadas informações), habilidades e normas, que são apresentados na Tabela 6, Tabela 7, Tabela 8 e Tabela 9.

Tabela 6: Especificação do papel Avaliador

Avaliador	
Objetivos:	Monitorar e avaliar o aluno
Responsabilidades:	Identificar o aluno e o seu nível de conhecimento; Identificar dificuldades encontradas pelo aluno.
Colaboradores:	Buscador e Tutor
Habilidades:	Conhecer as respostas das atividades do aluno e os conceitos que são tratados nas atividades
Normas:	Acessar a base de respostas do aluno e conhecer o critério de avaliação

Tabela 7: Especificação do papel Tutor

Tutor	
Objetivos:	Propor atividades e conteúdos
Responsabilidades:	Enviar atividades e conteúdos para o aluno de acordo com a solicitação e o nível de conhecimento do aluno
Colaboradores:	Avaliador
Habilidades:	Conhecer as dificuldades dos alunos para buscar atividades compatíveis; Personalizar o curso de acordo com nível de conhecimento do aluno
Normas:	Acesso as atividades e conteúdos do aluno.

Tabela 8: Especificação do papel Buscador

Buscador	
Objetivos:	Buscar materiais que não existem no sistema
Responsabilidades:	Buscar novos materiais em repositórios ou na Internet que possam ajudar o aluno
Colaboradores:	Avaliador e Publicador
Habilidades:	Conhecer repositórios e conseguir acessar a Internet para pesquisar materiais
Normas:	Ter acesso a materiais com o assunto solicitado; Ter acesso a Internet

Tabela 9: Especificação do papel Publicador

Publicador	
Objetivos:	Publicar atividades e materiais no sistema
Responsabilidades:	Disponibilizar cada atividade e material no sistema
Colaboradores:	Tutor e Buscador
Habilidades:	Gravar atividade ou material no sistema.
Normas:	As atividades e materiais devem ser disponibilizados nas ferramentas corretas.

4.1.4.3 Mapeamento dos papéis para o estilo arquitetural

Nessa etapa é feito o mapeamento dos papéis especificados na etapa anterior para o estilo arquitetural *Joint-Venture* escolhido na primeira etapa. Analisando as especificações dos papéis identificados e os componentes do *Joint-Venture* cada papel é mapeado assumindo o lugar do componente do estilo arquitetural que ele possui maior similaridade.

Na Figura 21, é apresentado o Diagrama Arquitetural do sistema SWBES resultante desse mapeamento em conformidade com o estilo arquitetural *Joint-Venture*, onde círculos alaranjados representam os papéis desempenhados pelos agentes na arquitetura e as setas identificam os relacionamentos. Em conformidade com o estilo arquitetural *Joint-Venture* que requer um componente administrador (Figura 20), foi introduzido um novo papel responsável por controlar e coordenar a interação entre os demais papéis, denominado Controlador.

O papel do Tutor é propor para os alunos as atividades requeridas para a conclusão do curso. O Tutor, quando necessário, irá propor novas atividades para suprir as dificuldades identificadas pelo Avaliador. Essas atividades podem ser individuais ou em grupo, caso haja mais alunos com a mesma dificuldade. Um exemplo de atividade em grupo que pode ser proposta é a criação de fóruns de discussão. Os temas a serem abordados nos fóruns de discussão ficam armazenados e o Tutor é o responsável por saber qual é a melhor hora de apresentá-lo para o aluno e quais alunos devem participar dessa discussão. Assim os alunos que precisam de reforço podem ter uma chance de trocar informações com outros participantes através da infraestrutura da rede social.

O papel do Avaliador foi dividido em: Avaliador de Desempenho e Avaliador Social. O papel do Avaliador de Desempenho é monitorar e verificar se o aluno apresenta dificuldades em algum conceito estudado. Como resultado da avaliação do desempenho é produzido o recurso “Dificuldades”. O Avaliador de Desempenho também tem a função de analisar o recurso “Dificuldades” e identificar, quando possível, grupos de alunos que possuem a mesma dificuldade, gerando um segundo recurso chamado “Grupos”. O

Avaliador Social deve acompanhar o aluno fora da ferramenta em redes sociais, identificando assim quais assuntos do curso são mais comentados pelo aluno.

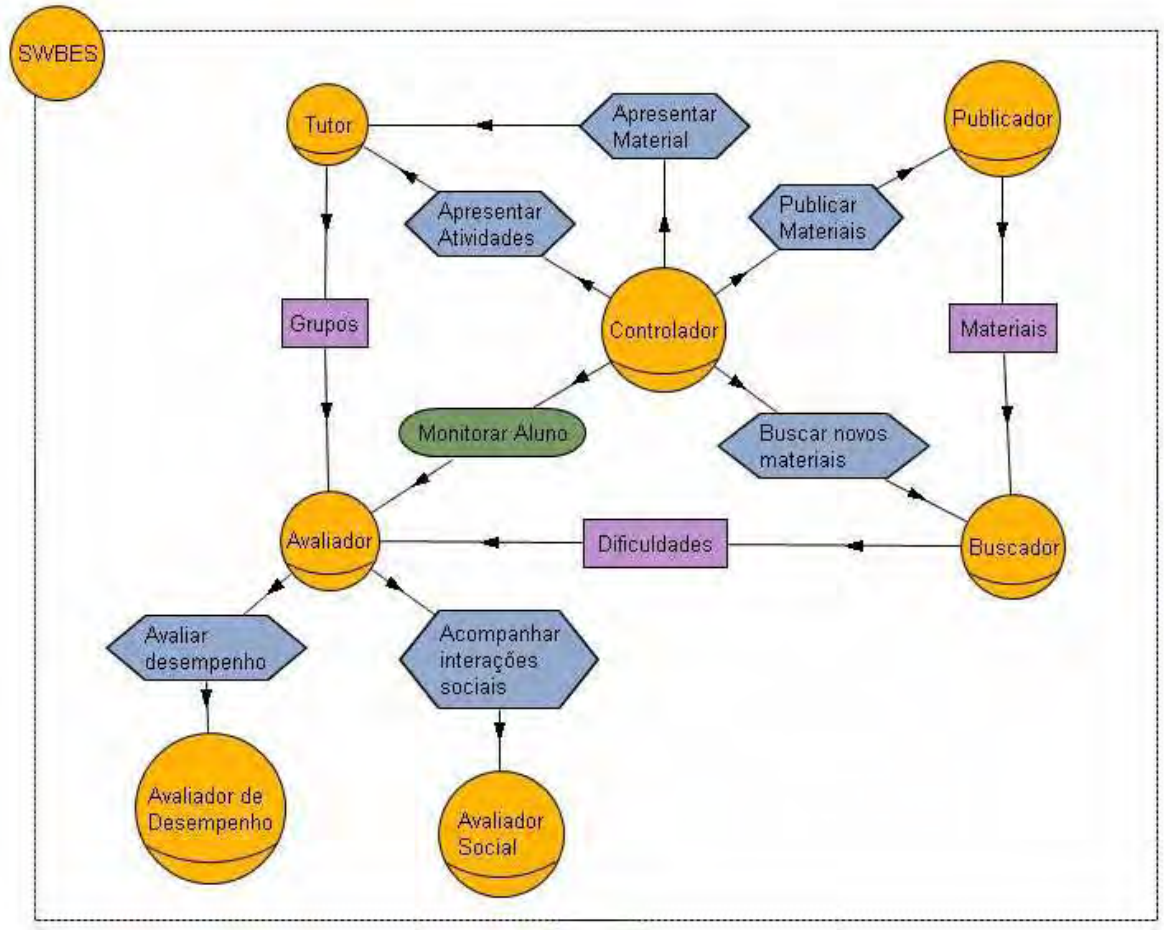


Figura 21: Diagrama Arquitetural

O papel do Buscador é procurar novos materiais sobre assuntos em que os alunos estão com dificuldades. Ele deve procurar em repositórios externos por objetos de aprendizagem que possam contribuir com a formação do aluno, fornecendo outros tipos de mídias e/ou de outras fontes. O Buscador prioriza a busca de materiais em repositórios mais confiáveis, como por exemplo, os de universidades e somente depois busca conteúdos na *Web*.

O papel do Publicador é receber materiais e atividades que serão inseridas no ambiente de ensino. Os materiais que o Publicador recebe, podem vir do professor ou do Buscador. O Publicador deve analisar o conteúdo do material ou atividade recebida e anotá-lo de acordo com os conceitos encontrados e o domínio de conhecimento do curso.

4.1.5 Projeto Detalhado

A fase de Projeto Arquitetural foi finalizada com a geração do diagrama arquitetural e a definição dos papéis. Na fase de Projeto Detalhado será descrita a estrutura interna de cada papel.

O funcionamento interno do Avaliador de Desempenho (Figura 22) foi dividido em três metas: “Monitorar dificuldades encontradas pelo aluno”, “Monitorar acertos do aluno” e “Agrupar alunos”. A primeira meta tem por objetivo analisar as respostas do aluno nos questionários. A partir dessa análise, é gerado um cadastro de dificuldades. Caso mais alunos tenham a mesma dificuldade, a meta “Agrupar alunos” analisará e gerará um cadastro de grupos de alunos. A última meta “Monitorar acertos do aluno” tem a função parecida com a primeira meta, a diferença é que serão analisados acertos, ao invés de erros.

Na Figura 23 é detalhado o funcionamento do Buscador, o qual deve acessar o cadastro de dificuldades gerado pelo Avaliador para identificar quais materiais devem ser buscados. A partir de identificado o tema que deve ser buscado, o Buscador tenta encontrar materiais em universidades, que satisfaçam a meta de qualidade, aqui descrita como “Buscar Material Confiável”. Caso não encontre nenhum material, a última opção é procurar material na Internet. O aluno terá a opção de responder se o material encontrado ajudou ou não seu desenvolvimento.

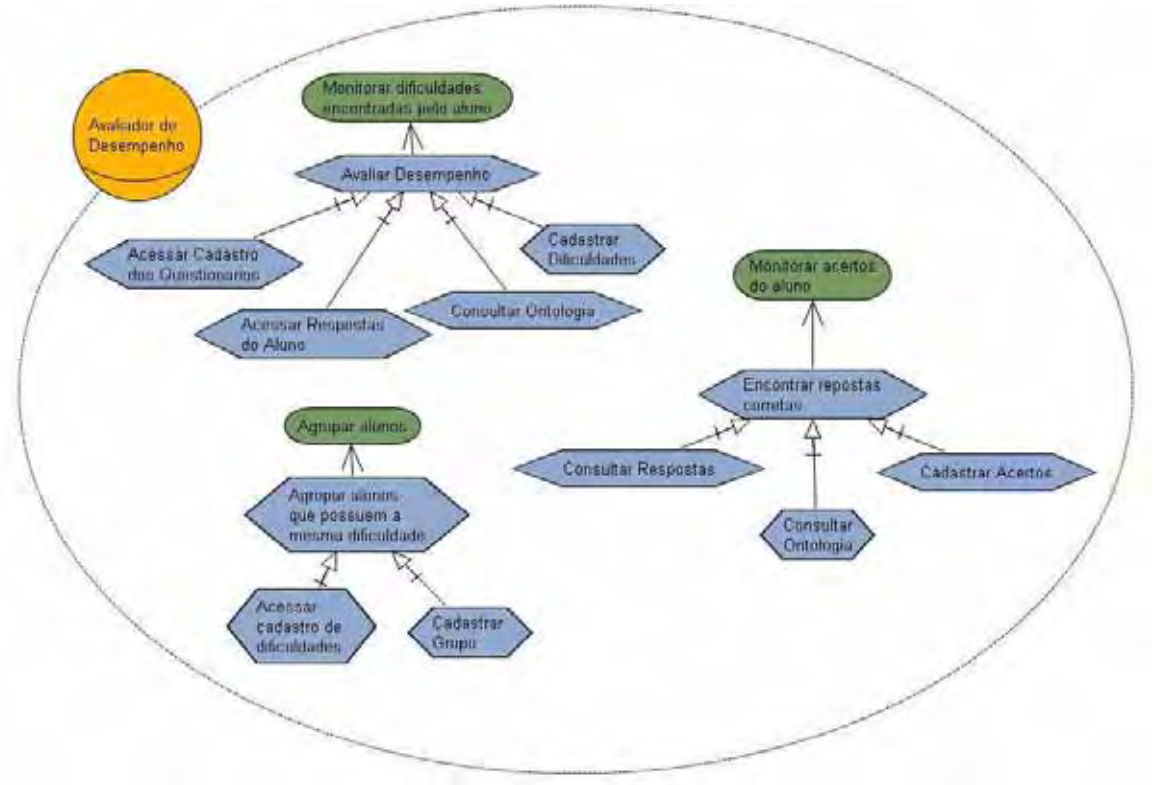


Figura 22: Diagrama detalhado - Avaliador Individual

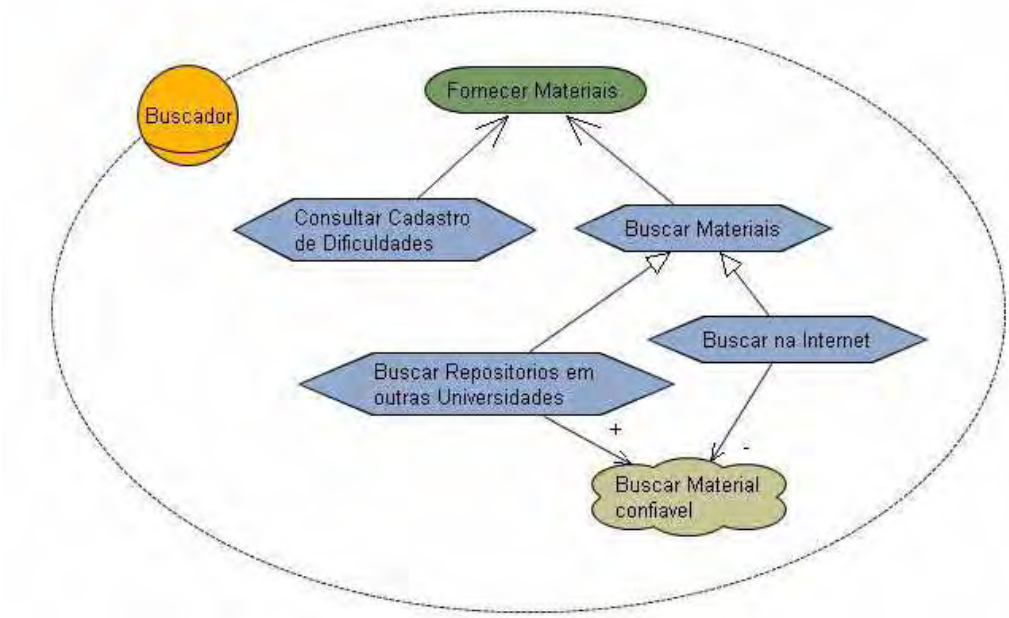


Figura 23: Diagrama detalhado - Buscador

A Figura 24 ilustra o funcionamento interno do Publicador, o qual é o responsável por receber os materiais do professor ou do Buscador e associá-lo com o conteúdo do curso podendo anotar seu conteúdo para facilitar a busca posterior.

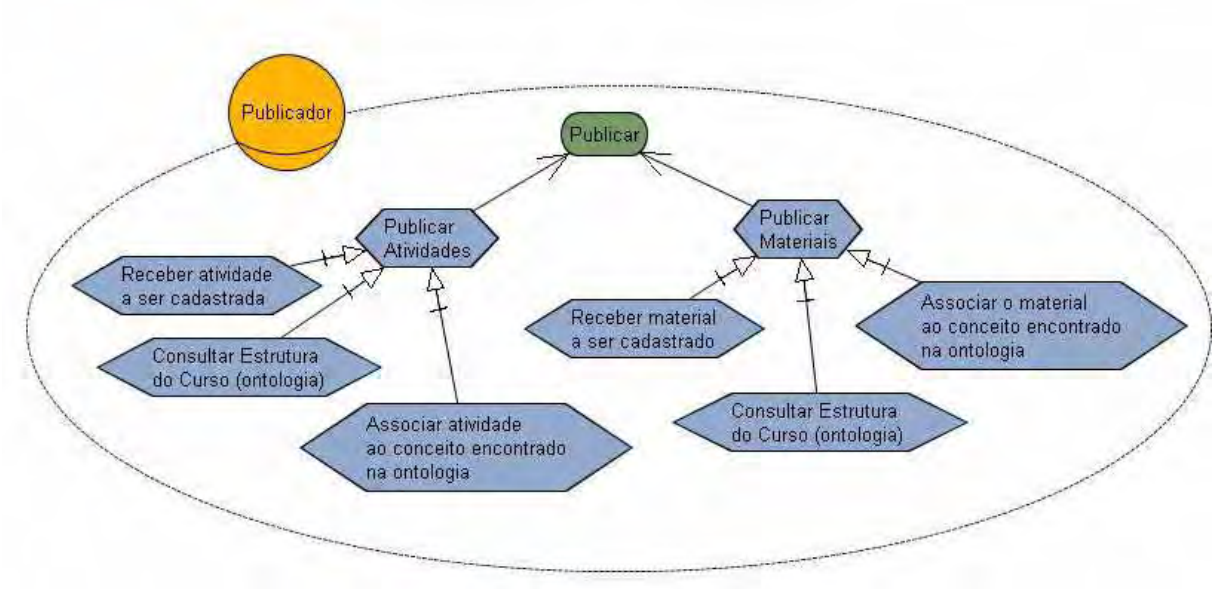


Figura 24: Diagrama detalhado - Publicador

O Tutor (Figura 25) tem como meta apresentar o conteúdo e atividade para o aluno. Para isso ele deve acessar os dados do aluno para identificar qual o nível de estudo que o aluno está e depois acessar a lista de materiais cadastrados de acordo com o nível do aluno. A apresentação das atividades segue o mesmo passo, com a diferença que as

atividades em grupo já foram filtradas para o nível do aluno.

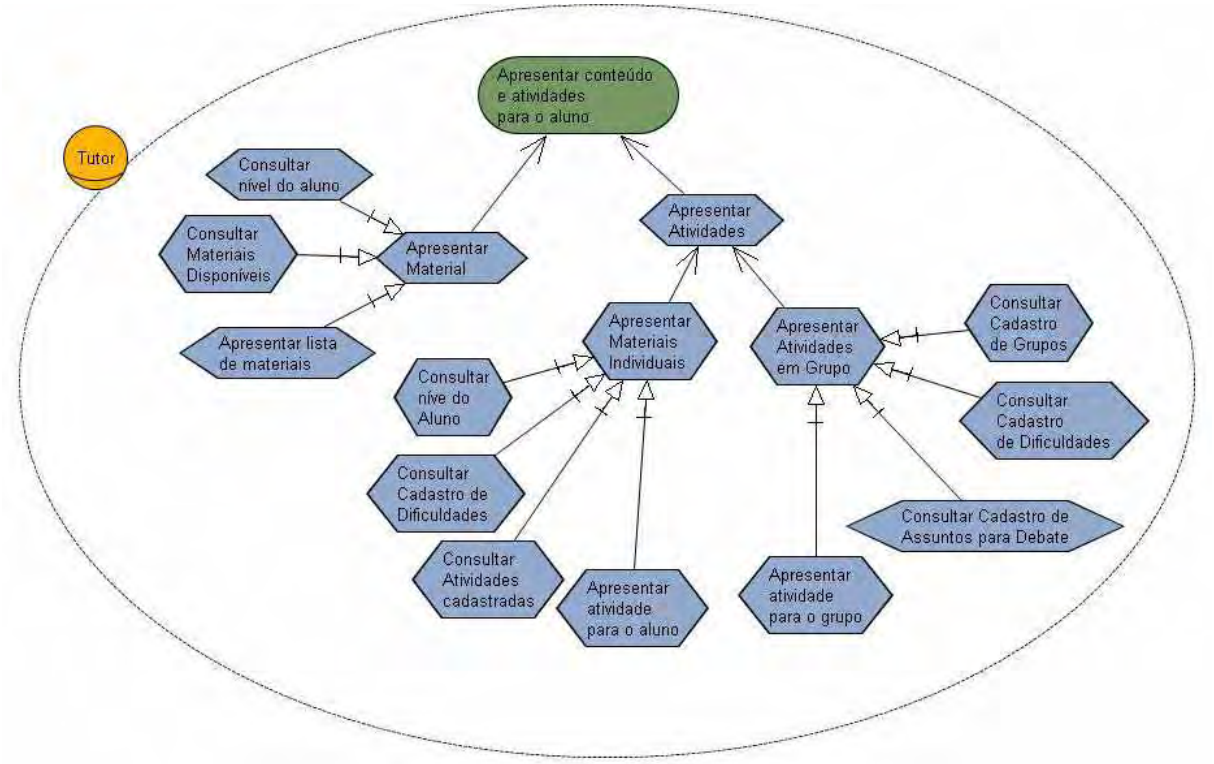


Figura 25: Diagrama detalhado – Tutor

O Controlador (Figura 26) tem a meta de coordenar os agentes e para atingi-la, dois planos são necessários: “Atender Requisições” e “Monitorar Agentes”. O primeiro plano foi criado com o objetivo de atender as requisições da aplicação e a partir dessas requisições iniciar os agentes conforme necessário.

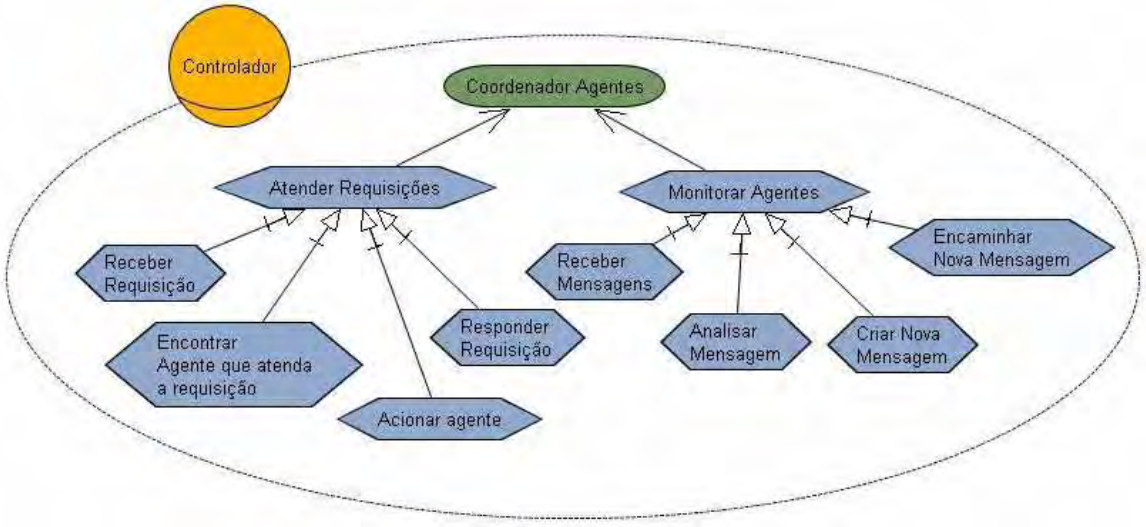


Figura 26: Diagrama detalhado – Controlador

O plano “Monitorar Agentes” tem por objetivo trocar mensagens com os agentes realizando o papel de coordenador, recebendo mensagens e enviando mensagens para iniciar novos processos.

O Avaliador Social (Figura 27) tem a meta “Monitorar Redes Sociais”, para isso ele deve acompanhar as interações sociais, acessando as redes, monitorando os assuntos mais comentados e as interações entre os alunos.

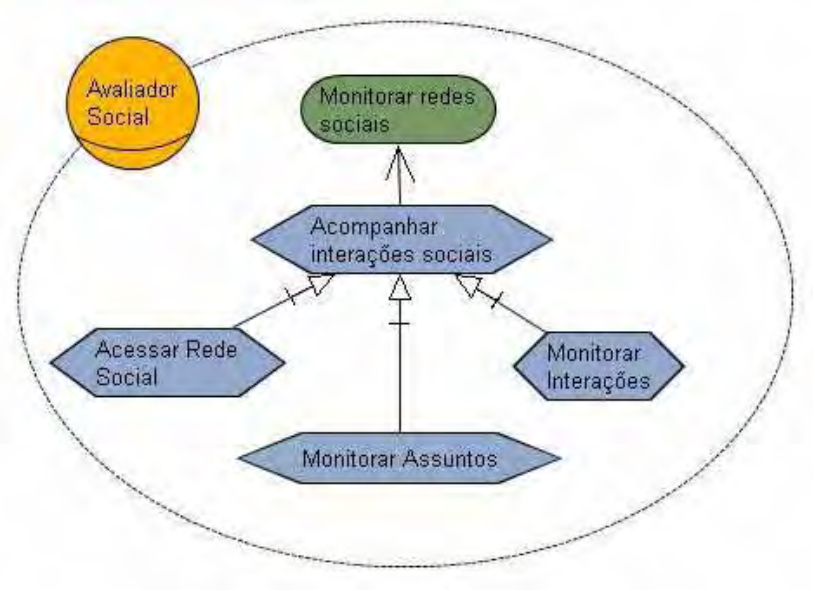


Figura 27: Diagrama detalhado - Avaliador Social

A próxima fase do Tropos é a Implementação dos agentes. Essa fase será tratada no capítulo seguinte juntamente com a integração dos agentes ao SWBES. Como um estudo de caso, utilizando a arquitetura será implementado um SWBES e criado um curso voltado ao ensino de rochas carbonáticas.

5 Aspectos da Implementação do SWBES

Neste capítulo é apresentado a implementação de um SWBES de acordo com a arquitetura proposta nos capítulos anteriores. Inicialmente, são apresentados os detalhes da implementação de cada componente da arquitetura juntamente com as tecnologias utilizadas.

Na validação do modelo arquitetural implementado, um SWBES para ensino de Rochas Carbonáticas foi criado, que consistiu da implementação de uma ferramenta de educação a distância simplificada, da criação do material de aprendizagem, dos questionários, das ontologias e dos agentes. Finalmente, é apresentado um exemplo com as principais interações de um aluno no SWBES de Rochas Carbonáticas.

5.1 Definições Tecnológicas para construção do SWBES

Na arquitetura proposta para o SWBES foram definidos seus componentes. Cada componente foi analisado e, em seguida foram definidas as tecnologias utilizadas para a sua implementação. Para a construção do SWBES aplicado ao ensino de Rochas Carbonáticas foram realizadas as definições das tecnologias de cada componente, as quais são descritas abaixo.

5.1.1 Sistema de Educação a Distância

A arquitetura prevê trabalhar com qualquer tipo de Sistema de Educação a Distância. A decisão de criar um sistema novo em uma versão simplificada foi tomada em razão da experiência do autor no desenvolvimento de ferramentas de ensino na *Web* e do tempo necessário para adquirir os conhecimentos de outras ferramentas de ensino.

O sistema de educação a distância foi implementado com algumas das funcionalidades existentes nos ambientes tradicionais. Há funcionalidades que permitem ao aluno efetuar a autenticação no SWBES (Figura 28), o acesso ao material didático e treinamento. As funcionalidades do sistema são utilizadas para a comunicação com os agentes e são adaptadas de acordo com o perfil e a interação com o usuário.

Na implementação do Sistema de Educação a Distância foi utilizada a linguagem JAVA, utilizando o *framework* JSF com Richfaces. Para o estilo das páginas foi utilizado o CSS (*Cascading Style Sheets*).

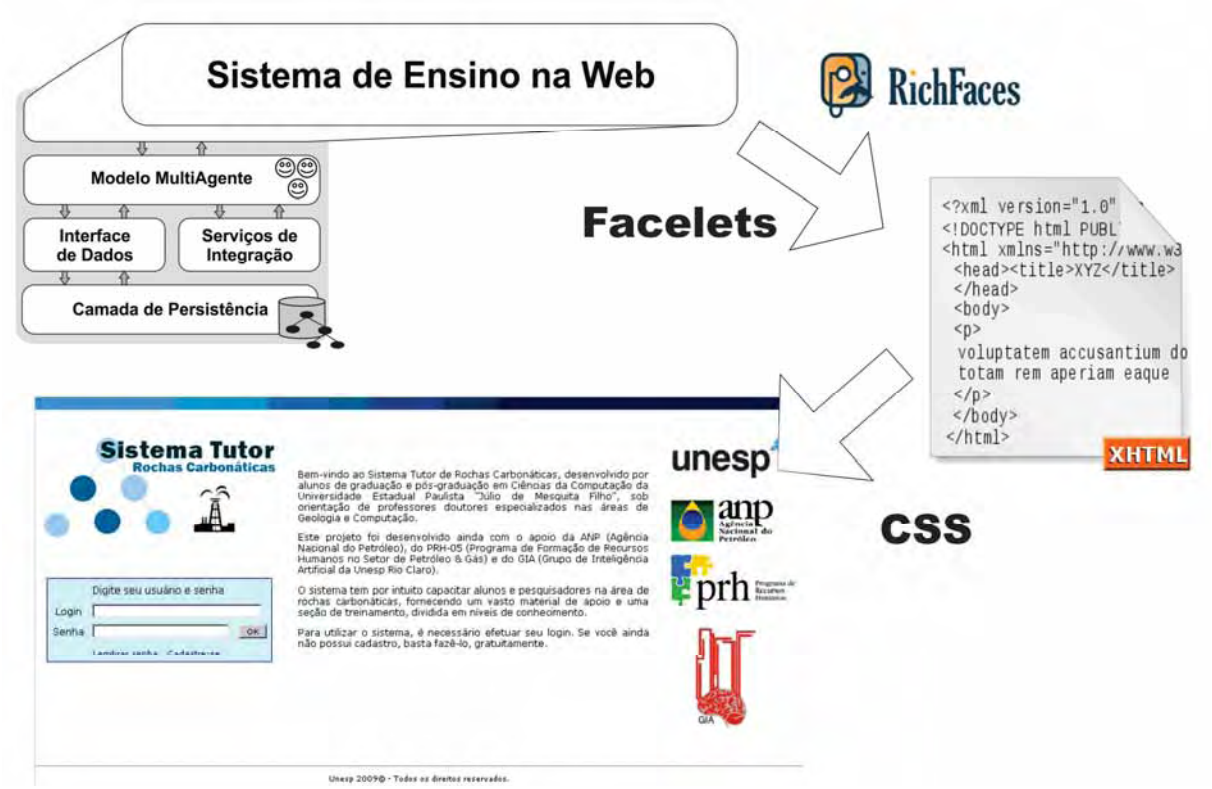


Figura 28: Tecnologias utilizadas na geração da interface *Web*

5.1.2 Modelo Multiagente

O componente Modelo Multiagentes implementa a arquitetura definida na fase de Projeto Detalhado (Capítulo 4). O sistema multiagentes foi desenvolvido para ser executado com a plataforma JADEX. Os agentes modelados no TAOM4E foram exportados utilizando o *software t2x Tool*, gerando a estrutura em JAVA reconhecida pelo JADEX e os arquivos de definição dos agentes (*Agent Definition File* – ADF) no formato XML.

No arquivo ADF estão representadas as crenças, os objetivos dos agentes, também são descritas todas as mensagens (requisições) que os agentes podem receber. Nele os planos dos agentes são ligados as metas por meio de um mecanismo de gatilho. A ferramenta também gera o esqueleto dos planos, que são representados por classes Java com um método *body* onde deve ser implementado o que o plano irá executar.

O *t2xTool* faz o mapeamento dos componentes gerando o esqueleto do código seguindo o modelo de agentes BDI (*Beliefs Desire Intentions*). Esse modelo define um mecanismo de deliberação, utilizado para a escolha dos planos a serem executados para alcançar determinadas metas. A plataforma JADEX, que suporta o modelo de agente BDI, foi utilizada para a implantação e teste dos agentes gerados.

A comunicação entre o Sistema de Educação e o componente multiagentes foi

realizada por meio de *socket*, de forma que o sistema de educação estabelece uma conexão com o agente Controlador e dessa forma passa a enviar requisições.

Os agentes depois de implementados são executados na plataforma JADEX para serem utilizados junto com a aplicação. Todos os agentes são registrados como serviços na plataforma através do DF (*Directory Facilitator*), que funciona como um serviço de páginas amarelas, onde o requisitante procura pelo serviço requerido e encontra quem o fornece.

5.1.3 Serviços de Integração

A arquitetura proposta oferece um ponto de extensão que ficou sob responsabilidade do componente de Serviços de Integração. O objetivo desse componente é ocultar todo o trabalho de conexão com outros serviços deixando o código da aplicação limpo e de fácil manutenção.

Para a construção desse componente, foi utilizado um *framework* chamado *Ont-Space*[8] que permite consultar Objetos de Aprendizagem de forma simples seguindo o padrão LOM. A consulta é realizada através de objetos JAVA criados como exemplo do que se deseja consultar. Como por exemplo, para consultar por objetos de aprendizagem que tenham como título “Rochas”, deve ser instanciado um objeto com o valor “Rochas” no atributo *title*, tornando a consulta em repositórios remotos muito simples.

5.1.4 Interface de Dados

O componente de Interface de Dados foi implementado de acordo com o padrão de projeto (*design pattern) Data Access Object (DAO)*. Esse padrão foi especificado com a função de tradutor entre o formato de armazenamento dos dados e formato em que os dados estão nos objetos JAVA.

Uma vantagem de trabalhar utilizando essa abordagem é a abstração da origem e do modo de acesso às informações. Na implementação do SWBES essa abstração traz ganho para o sistema, pois a aplicação pode consultar dados de ontologias locais, remotas ou banco de dados sem precisar conhecer onde esses dados estão armazenados.

Outra vantagem no uso do DAO é a unificação de acesso aos dados, ou seja, a responsabilidade de acesso aos dados fica exclusiva do componente de Interface de Dados.

Neste componente ficam implementados os métodos de conexão com o banco de dados, bem como seus métodos de consulta, atualização, inserção e exclusão. Também

[8] http://code.google.com/p/ont-space/

neste mesmo componente ficam os métodos de leitura e escrita nas ontologias.

Neste componente, os seguintes DAOs foram construídos:

- ConceptDAO: possui os métodos para busca de conceitos na ontologia de domínio;
- ObjectiveQuestionDAO: possui métodos para busca de questões do tipo objetivas. A especialização desse método em relação ao QuestionDAO é devido as alternativas existentes nesse tipo de questão, alternativas quais os demais tipos de perguntas não possui;
- ProfileDAO: possui os métodos para manipular a ontologia de perfil do aluno. Nesse DAO ficam os métodos para atualizar a pontuação do aluno no conceito correto que ele está respondendo e também a busca de conceitos que ele está com dificuldades;
- QuestionDAO: é responsável pela atualização e pesquisa das questões. As questões cadastradas podem ser pesquisadas de maneiras diferentes, como por exemplo, pelo nível de dificuldade (*findQuestionByLevel*), pelo código (*findQuestionById*), pelo conceito que ela trata (*findQuestionsByConcept*) ou buscar todas as questões que já foram respondidas pelo aluno (*findAnsweredQuestions*).
- TopicDAO: possui os métodos de busca e ordenação dos tópicos cadastrados.
- UserDAO: é responsável pela criação, atualização e busca dos dados dos usuários.

5.1.5 Camada de Persistência

Na camada de persistência estão as ontologias que codificam os materiais de aprendizagem, o perfil dos alunos e os conceitos do domínio, além do Banco de Dados de informações dos usuários.

O banco de dados foi criado em *PostgreSQL* e nele estão armazenadas informações gerais pertencentes aos usuários do sistema: dados pessoais, senha e nível de conhecimento atual (*status*).

As ontologias foram criadas utilizando OWL. Na ontologia de domínio estão codificados os conceitos da área de conhecimento que o sistema de ensino está destinado. Os conceitos descritos nessa ontologia servem para anotar os textos que estão na ontologia de materiais de aprendizagem. Essa ontologia é utilizada também em conjunto com a ontologia de perfil para mapear o perfil de cada aluno de acordo com sua evolução no aprendizado. Assim, se torna possível verificar quais conceitos estão fixados e quais

conceitos necessitam de mais treinamento.

A ontologia de materiais de aprendizagem foi concebida de acordo com a ontologia proposta por Araujo (2003). Nesta ontologia estão as descrições, as referências e anotações dos materiais de aprendizagem utilizados no curso. Os objetos de aprendizagem descritos nessa ontologia estão relacionados a um ou mais conceitos da ontologia de domínio. Assim, é possível saber quantos e quais objetos de aprendizagem explicam ou utilizam um determinado conceito da área de interesse (ontologia de domínio). Isso se dá por meio de relacionamentos entre recursos das ontologias e anotações dos materiais de aprendizagem. Na ontologia também estão definidos na sequência correta estabelecida pelo professor. As perguntas e dicas utilizadas no treinamento do aluno também estão descritas nessa ontologia.

As perguntas que estão na ontologia de materiais de aprendizagem se relacionam com os conceitos da ontologia de domínio. Assim, conforme a evolução do aluno é possível mapear quais conceitos o mesmo já tem conhecimento e quais ainda necessitam mais treinamento.

Segundo o modelo de Araújo (2003), os materiais de aprendizagem podem ter relações entre si, como um material ser pré-requisito de outro, um material ser similar a outro e relações com diversos objetos de aprendizagem, que podem ser exercícios, simulados ou questionários. Para cada tema diferente, uma ontologia de domínio é necessária para anotar os materiais utilizados.

As ontologias de perfil armazenam as informações de cada usuário no que se refere à sua evolução no curso. Com essas ontologias é possível obter as informações necessárias para mapear um perfil de cada aluno, conhecendo seu desempenho em cada curso realizado. Por exemplo, no perfil constam os materiais de aprendizagem acessados pelo aluno e seu grau de sucesso em cada tópico, construindo assim um caminho percorrido (trilha).

5.2 Implementação do SWBES

Definidos os componentes arquiteturais, um SWBES foi criado para avaliar seu comportamento na prática. Para o desenvolvimento dos testes foram necessárias: ferramenta de educação a distância, criação de materiais de estudo, questionários, ontologias e agentes.

O conteúdo de um curso abordando o tema rochas carbonáticas foi escolhido para ser utilizado no SWBES, pois está sendo alvo de diversas pesquisas no campus da UNESP em Rio Claro - SP. As rochas carbonáticas são de grande significado estratégico e

econômico, pois são importantes reservatórios de petróleo. Atualmente, os estudos em rochas carbonáticas são extremamente relevantes para o país em decorrência das novas reservas descobertas no pré-sal.

5.2.1 Construção das Ontologias

Na construção do curso de Rochas Carbonáticas foi requerido o desenvolvido de três ontologias: a ontologia de Materiais de Aprendizagem, a ontologia de Perfil e a ontologia de Domínio. No domínio de Rochas Carbonáticas foi constatado que não existia uma ontologia de Rochas Carbonáticas, por isso a construção dessa ontologia se fez necessária.

As ontologias são detalhadas a seguir.

5.2.1.1 Ontologia de Domínio de Rochas Carbonáticas

A ontologia de Rochas Carbonáticas utilizada no SWBES está descrita detalhadamente no Relatório Técnico do Grupo de Inteligência Artificial/DEMAC/UNESP, elaborado por Abdalla e colegas (2009), em que são apresentados todos os conceitos da área de rochas carbonáticas e todas as etapas do processo de modelagem detalhados.

Na ontologia de Rochas Carbonáticas (Apêndice F) foram definidas a hierarquia de classes e subclasses, e as propriedades que relacionam as classes da ontologia, além da inserção das restrições que definem as classes.

Na Figura 29 é apresentada a classe Diagenese dessa ontologia, a imagem foi construída utilizando o *plugin* OWLViz do Protégé, no qual é possível visualizar a classe pai chamada Petrografia e que dela derivam outras classes referentes às características da rocha e suas classificações.

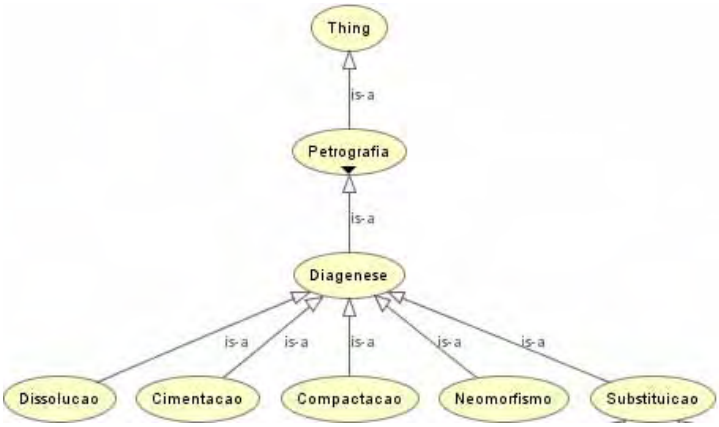


Figura 29: Representação da classe Diagenese da ontologia de rochas carbonáticas

5.2.1.2 Ontologia de Materiais de Aprendizagem

A Ontologia de Materiais de Aprendizagem foi construída tomando como base a ontologia de Materiais de Aprendizagem proposta no trabalho de Araújo (2003). É nessa ontologia que estão as questões e imagens da seção de treinamento, além das apostilas virtuais disponibilizadas no sistema.

Um Curso pode ter Disciplinas que são formadas por um ou mais tópicos. Cada tópico pode conter um ou mais Materiais de Aprendizagem. Por sua vez, cada Material de Aprendizagem pode conter um ou mais Objetos de Aprendizagem.

Um Objeto de Aprendizagem pode ser do tipo Perguntas, Apostilas ou Imagem. Em cada pergunta foi associado um grau de Dificuldade. Nesse sentido, na classe Pergunta foi criada a propriedade “perguntaTemDificuldade”. Utilizando essa propriedade é possível obter todas as perguntas que estão no nível de dificuldade em que o aluno se encontra atualmente.

Essa estrutura de curso foi mapeada na ontologia de materiais de aprendizagem.

Na Figura 30 é apresentada uma visão gráfica da ontologia de Materiais de Aprendizagem gerada pelo *plugin* OWLViz do Protégé (PROTEGE, 2009).

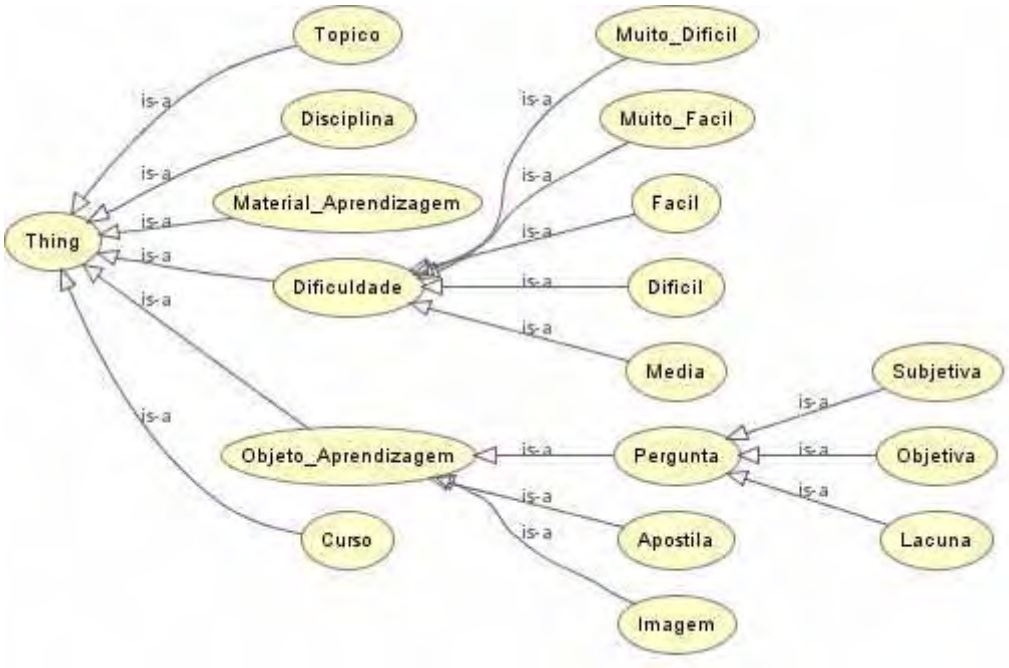


Figura 30: Representação gráfica da ontologia de Materiais de Aprendizagem

A ontologia de material de aprendizagem tem a finalidade de organizar e descrever o material de aprendizagem. No entanto, é necessário associar cada material a um domínio referente ao curso. Isso é feito relacionando a ontologia de material de aprendizagem com a

ontologia de domínio.

Nesse sentido, no contexto de construção dos materiais de rochas carbonáticas, a ontologia de domínio foi importada na ontologia de materiais de aprendizagem. Para representar essa associação, uma propriedade denominada “objetoTemConceito” foi criada tendo como domínio os objetos de aprendizagem e como contra-domínio todos os conceitos de Petrografia. Por meio dessa propriedade então é possível identificar qual objeto de aprendizagem trata de qual conceito do assunto lecionado.

Utilizando a propriedade descrita, podem ser definidas associações entre as perguntas e os conceitos da ontologia de domínio (Figura 31). Como por exemplo, a pergunta que tem como enunciado: “Quais os fenômenos diagenéticos observados na imagem?” deve estar relacionada ao conceito “Diagenese” da ontologia de domínio.

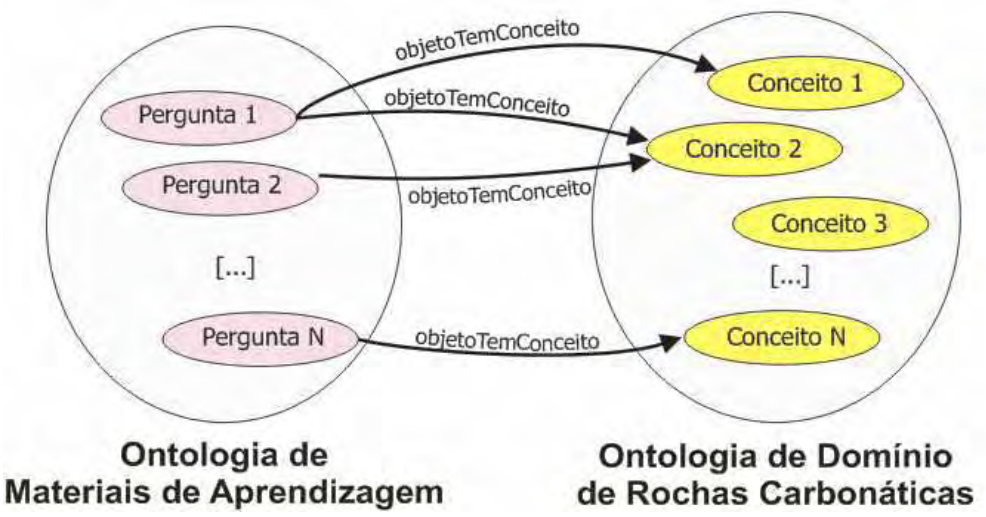


Figura 31: Relacionamento da ontologia de materiais com a ontologia de domínio

Existem três tipos de perguntas que podem ser criadas: Lacuna, Objetiva e Subjetiva. As propriedades em comum entre elas são: “perguntaTemImagem”, “perguntaTemCodigo”, “perguntaTemRepostaCorreta”, “perguntaTemEnunciado”, “perguntaTemDificuldade”, “perguntaTemTipo” e “objetoTemConceito”.

5.2.1.3 Ontologia de Perfil

A ontologia de perfil foi criada para acompanhar o andamento do aluno. Essa ontologia é composta da importação da ontologia de domínio e inclui atributos para gravar o andamento do aluno em relação a cada conceito.

A ontologia de perfil é criada com a importação da ontologia de domínio e as

propriedades “erros” e “acertos” são incluídas. Todos os conceitos da ontologia de domínio são instanciados na ontologia de perfil com essas duas propriedades citadas. Essas propriedades foram criadas tendo como domínio a classe raiz, assim todas as suas subclasses terão também essas propriedades.

Uma ontologia de perfil é única para cada aluno, ou seja, cada aluno é tratado individualmente em cada ontologia de perfil.

A cada pergunta da ontologia de materiais de aprendizagem que o aluno responde, o sistema verifica quais conceitos da ontologia de domínio são tratados na pergunta. Tendo encontrado os conceitos, o sistema atualiza na ontologia de perfil a pontuação de acerto ou erro em cada conceito abordado na pergunta.

Como por exemplo, se é apresentada uma pergunta ao aluno que trata do conceito “Arcabouço”. No caso em que o aluno acerta a resposta, o sistema procura o conceito “Arcabouço” na ontologia de perfil e acrescenta um ponto na propriedade “acertos”. Do contrário, se o aluno erra a resposta, o sistema acrescenta um ponto na propriedade “erros” da ontologia de perfil (Figura 32).

```
<domainOntology:Arcabouco rdf:ID="Arcabouco_21">
  <acertos rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
      2
  </acertos>
  <erros rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
      4
  </erros>
</domainOntology:Arcabouco>
```

Figura 32: Trecho do código da ontologia de perfil

5.2.2 Implementação dos agentes

Considerando a complexidade do modelo multiagentes proposto, foram escolhidos para implementação no Sistema de Rochas Carbonáticas os seguinte agentes: Avaliador, Buscador, Controlador e Publicador.

Utilizando a ferramenta *t2x Tool*, a partir da modelagem (Seção 5.1.2), são gerados os esqueletos dos agentes. Em razão dos agentes definidos adotarem o modelo BDI, a implementação dos agentes consiste na definição das crenças (*Beliefs*), metas (*Desires*) e planos (*Intentions*).

A seguir serão descritas as implementações de acordo com a modelagem realizada na seção 4.1.5 .

5.2.2.1 Agente Controlador

O agente Controlador tem por objetivo iniciar os agentes de acordo com as necessidades do sistema. Nesse agente, foi implementado um evento chamado “CoordenarAgentes”. Esse evento inicial é responsável por iniciar o plano “realPlan_IniciarAgente”.

Como seu nome sugere, o plano “realPlan_IniciarAgente” tem a função de iniciar os agentes. Para isso, ele instancia uma classe JAVA chamada “RealPlan_IniciarAgente”.

A implementação do agente Controlador foi realizada de acordo com o seguinte fluxo. Uma mensagem é enviada para o Controlador pelo SWBES, essa mensagem contém a identificação do aluno. Assim, o plano “realPlan_IniciarAgente” é chamado.

O agente Avaliador é então iniciado para avaliar o aluno, após realizada a avaliação, uma resposta é enviada para o controlador informando quantas dificuldades foram encontradas para aquele aluno. Caso não tenham sido encontradas dificuldades, o Controlador é encerrado. Do contrário, o agente Buscador é iniciado.

O agente Buscador realiza as buscas e responde ao Controlador com uma mensagem informando quantos materiais foram encontrados. Se não forem encontrados materiais, o Controlador é encerrado. Do contrário, o agente Publicador é iniciado.

O agente Publicador é executado e o processo do Controlador é encerrado. Restando ao Controlador aguardar uma nova mensagem para avaliar outro aluno.

O ADF completo desse agente está apresentado no Apêndice A e o Diagrama de Sequência completo dos agentes está no Apêndice E.

5.2.2.2 Agente Avaliador

O agente Avaliador é o responsável por avaliar o aluno e gravar as dificuldades identificadas no repositório de dificuldades.

Nesse agente estão implementados os planos “request_MonitorarDificuldades” e “realPlan_AvaliarDesempenho”. O plano “request_MonitorarDificuldades” tem a função de tratar o evento que é disparado pela mensagem “avaliar” enviada pelo agente Controlador ao agente Avaliador. Quando iniciado, esse plano lança a meta chamada “MonitorarDificuldades” (Figura 33).

A meta “MonitorarDificuldades” quando lançada tem a função de iniciar o segundo plano desse agente, denominado “realPlan_AvaliarDesempenho”. Esse plano tem a função de avaliar o aluno e gravar uma lista dos conceitos identificados como sendo dificuldades para o mesmo.

Metas:

```
<goals>
    <achievegoal name="MonitorarDificuldades" exclude="when_failed">
</goals>
```

Planos:

```
<plans>
    <plan name="requestPlan_MonitorarDificuldades">
      <body>new GoalRequestPlan("MonitorarDificuldades")</body>
      <trigger>
          <messageevent ref="request_MonitorarDificuldades"/>
      </trigger>
    </plan>

    <plan name="realPlan_AvaliarDesempenho">
        <parameter name="param" class="String">
          <goalmapping ref="MonitorarDificuldades.param"/>
        </parameter>
        <parameter name="result" class="String">
          <goalmapping ref="MonitorarDificuldades.result"/>
        </parameter>
            <body>new RealPlan_AvaliarDesempenho()</body>
            <trigger>
                <goal ref="MonitorarDificuldades"/>
            </trigger>
     </plan>
 </plans>
```

Eventos:

```
<events>
    <messageevent name="request_MonitorarDificuldades">
      <parameter name="content-start" class="String" direction="fixed">
            <value>"avaliar"</value>
      </parameter>
    </messageevent>
</events>
```

Figura 33: Planos, metas, crenças e eventos referentes a atividade avaliar

O plano “realPlan_AvaliarDesempenho” irá instanciar uma classe “RealPlan_AvaliarDesempenho”, na qual estão implementados os métodos responsáveis por fazer a avaliação dos alunos. O plano que avalia o desempenho do aluno realiza uma consulta no DAO passando a identificação do aluno como parâmetro (Figura 34). Essa consulta busca na ontologia de perfil todos os conceitos que possui a propriedade “erros” maior que zero, ou seja, todos os conceitos que foram feitas perguntas para o aluno e ele respondeu de forma incorreta.

Para cada conceito é realizada uma nova consulta, agora na ontologia de materiais de aprendizagem. Essa consulta recebe como parâmetro os conceitos (resultado da primeira consulta) e traz como resultado a quantidade de perguntas cadastradas para cada conceito.

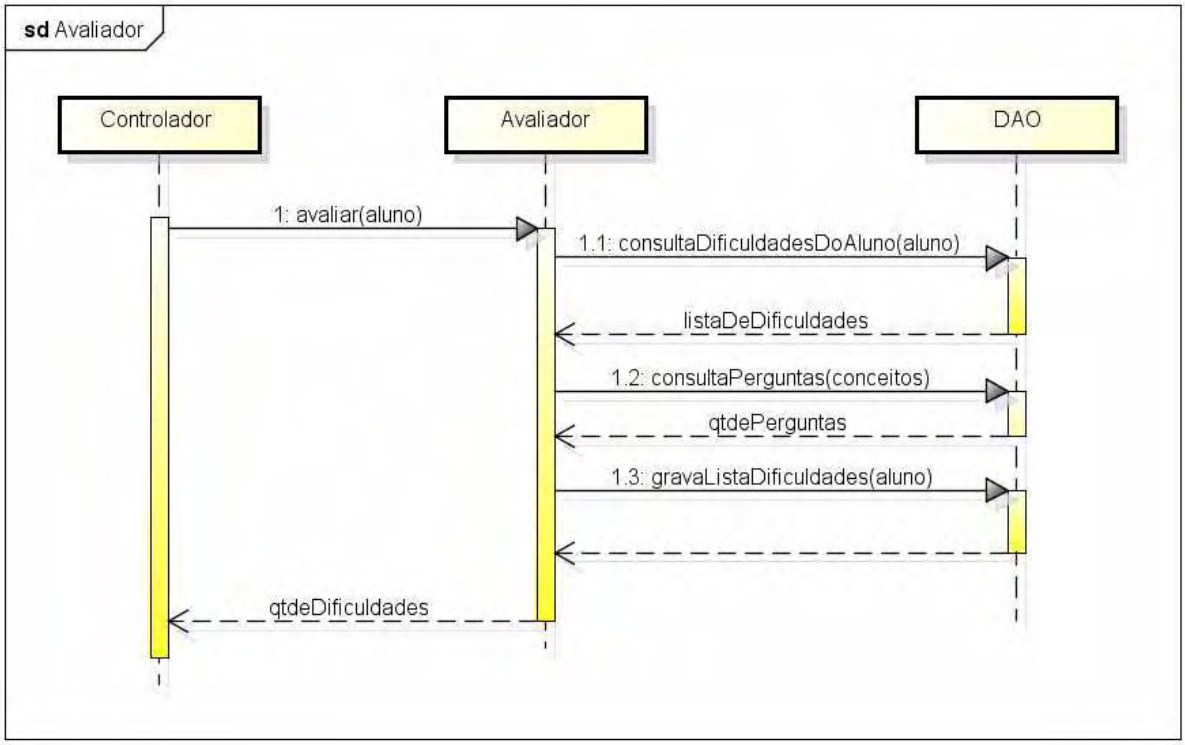


Figura 34: Diagrama de Sequência - Agente Avaliador

Assim, de posse dessas duas informações, o agente Avaliador verifica dentre as perguntas cadastradas, se o aluno errou mais que a metade desse total. Esse processo é realizado para todos os conceitos e a lista dos conceitos identificados é gravada no repositório “Dificuldades” junto com a identificação do aluno.

O agente Avaliador responde ao agente Controlador quantos são conceitos identificados como dificuldade para o aluno. Caso não sejam encontradas dificuldades, o Controlador encerra o processamento.

O ADF completo desse agente está apresentado no Apêndice B.

5.2.2.3 Agente Buscador

O agente Buscador é o responsável por buscar novos materiais em repositórios externos e gravá-los para serem utilizados pelo sistema.

Esse agente possui dois planos “requestPlan_FornecerMateriais” e “realPlan_BuscarMateriais”. O plano “requestPlan_FornecerMateriais” tem a função de receber a mensagem “buscar” do Controlador e lançar a meta “FornecerMateriais”.

A meta “FornecerMateriais” quando lançada tem a função de iniciar o segundo

plano desse agente, denominado “realPlan_BuscarMateriais”. Esse plano tem a função de buscar materiais na *Web*.

O plano “realPlan_BuscarMateriais” irá instanciar uma classe “RealPlan_BuscarMateriais”, na qual estão implementados os métodos responsáveis por buscar novos materiais.

Inicialmente o Buscador deve conhecer quais são as dificuldades que o aluno possui. Para isso é realizada uma consulta na base de Dificuldades gerada pelo agente Avaliador. Como resposta, uma lista de conceitos é devolvida (Figura 35).

Para cada conceito retornado, o Buscador realiza uma pesquisa nos repositórios cadastrados, enviando o conceito da ontologia de domínio. Se o repositório possuir algum material que trata do mesmo conceito, o Buscador grava no repositório de Materiais.

Essa é uma busca semântica, pois os materiais que são buscados devem estar relacionados ao conceito da ontologia de domínio para serem incluídos nos resultados da busca.

O ADF desse agente está apresentado no Apêndice C.

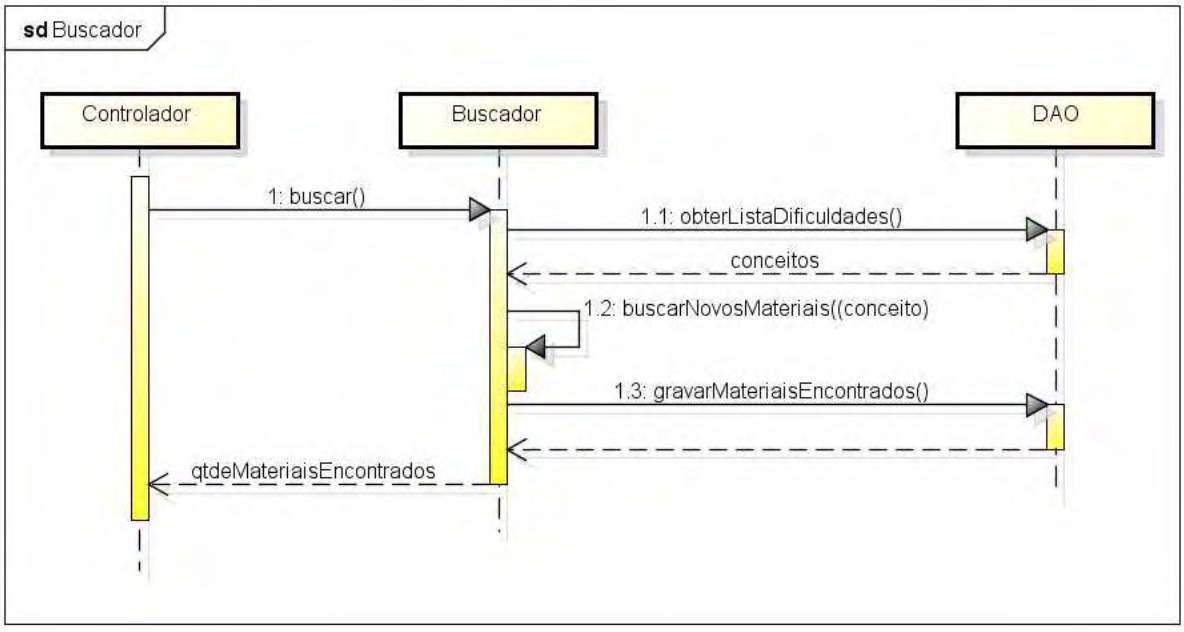


Figura 35: Diagrama de Sequência - Agente Buscador

5.2.2.4 Agente Publicador

O agente Publicador tem a função de publicar os materiais encontrados pelo agente Buscador.

Esse agente possui dois planos "requestPlan_Publicar" e "realPlan_PublicarMateriais". O plano “requestPlan_Publicar” tem a função de receber a mensagem “publicar” do Controlador e lançar a meta “Publicar”.

A meta “Publicar” quando lançada tem a função de iniciar o segundo plano desse agente, denominado “realPlan_PublicarMateriais”. Esse plano tem a função de gravar na ontologia de materiais de aprendizagem, o novo material encontrado.

O plano “realPlan_PublicarMateriais” irá instanciar uma classe “RealPlan_PublicarMateriais”, na qual estão implementados os métodos responsáveis por publicar os novos materiais.

O agente Publicador faz uma requisição ao DAO para obter a lista de materiais que foram encontrados pelo Buscador (Figura 36). Para cada material encontrado, o publicador identifica quais são os materiais existentes que já tratam desse conceito e inclui em seguida os novos materiais na ontologia de materiais de aprendizagem.

O ADF desse agente está apresentado no Apêndice D.

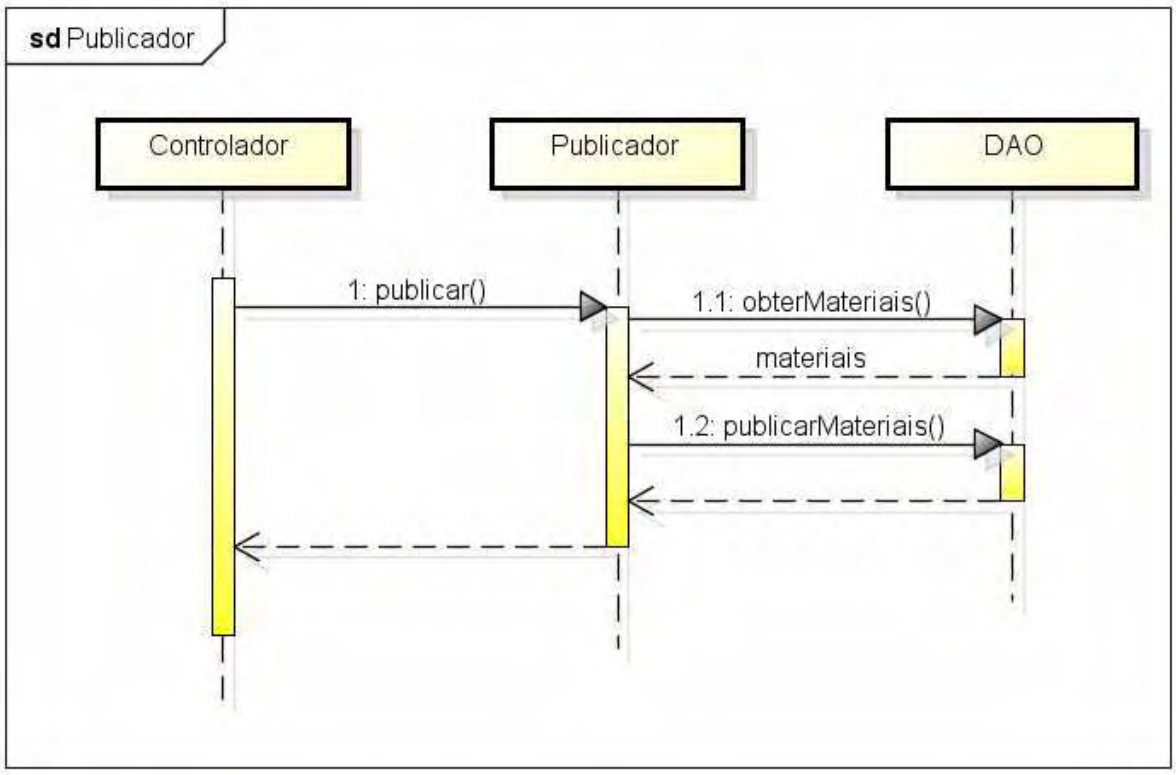


Figura 36: Diagrama de Sequência - Agente Buscador

5.3 Estudo de Caso

Para validar a arquitetura do SWBES proposta um estudo de caso foi realizado. Este estudo consistiu na criação de um curso introdutório para o ensino dos conceitos

básico da área de Rochas Carbonáticas.

5.3.1 Construção do curso

A construção do curso segue uma abordagem *bottom-up*, ou seja, primeiro é realizada a definição dos materiais, posteriormente a estruturação do curso e por último as questões. Os passos para a criação do curso são os seguintes:

- definição da ontologia de domínio: construção ou re-utilização de uma ontologia de domínio existente;
- construção do conteúdo do curso: os conteúdos que irão formar os materiais devem ser criados e disponibilizados na Web;
- anotação dos materiais de aprendizagem: os materiais criados são descritos na ontologia de material de aprendizagem;
- definição da estrutura do curso: a estrutura divide o curso em disciplinas, tópicos e define qual a sequência que os materiais de aprendizagem devem aparecer para o aluno;
- cadastro das perguntas: as perguntas devem ser formuladas e gravadas na ontologia de materiais de aprendizagem e associadas aos conceitos que são abordados em cada uma.

Cada passo será descrito detalhadamente a seguir.

5.3.1.1 Definição da ontologia de domínio

O primeiro passo para a construção do curso foi a definição da ontologia de domínio. Com a ajuda de um especialista foi definida a Ontologia de Rochas Carbonáticas, descrita na seção 5.2.1.1 e detalhada em Abdalla e colegas (2009).

Os conceitos descritos nessa ontologia foram utilizados para anotação semântica dos materiais e das perguntas, definindo todo o conteúdo abordado no curso.

5.3.1.2 Construção do conteúdo do curso

Para a preparação do curso é necessário a definição do material a ser utilizado. O professor deve definir quais são os materiais de aprendizagem que farão parte do seu curso e quais são os objetos de aprendizagem que irão compor os materiais.

No curso introdutório foram criados seis tópicos para serem utilizados (ABDALLA et.

al., 2009): Introdução às Rochas Carbonáticas, Componentes Texturais das Rochas Carbonáticas, Classificação das Rochas Carbonáticas, Diagênese das Rochas Carbonáticas, Fácies Deposicionais e Modelos de Fácies e Carbonatos no Tempo e Espaço.

Os textos podem estar em qualquer formato que o aluno tenha condição de acessá-los, como por exemplo, HTML, PDF, DOC etc. No curso de Rochas Carbonáticas, esses textos foram criados em HTML, usando imagens para ilustrá-los e CSS para a formatação.

Como um exemplo do material construído, na Figura 37 é apresenta o texto do tópico de Classificação das Rochas Carbonáticas.



Figura 37: Tópico Classificação das Rochas Carbonáticas

5.3.1.3 Anotação dos Materiais de Aprendizagem

Nesse passo é realizada a anotação do conteúdo do curso utilizando a ontologia de material de aprendizagem e de domínio.

A anotação do conteúdo consiste na criação de uma instância na ontologia de material de aprendizagem, como demonstrado na Figura 38. Por exemplo, para o conteúdo

que aborda a “Classificação das Rochas Carbonáticas”, é criada na classe “Apostila” da ontologia de material de aprendizagem uma instância chamada “Apostila_Classificacao”, que é do tipo apostila. Após, é realizada a associação da propriedade “apostilaTemURL” com a URL que contém o endereço físico para o tópico da classificação (http://localhost:8084/Tutor/pages/material/topico3.html).

```
<Apostila rdf:ID="Apostila_Classificacao">
    <rdfs:comment xml:lang="pt">
        Apostila para o Tópico Classificação das Rochas Carbonáticas
    </rdfs:comment>
    <apostilaTemURL rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        http://localhost:8084/Tutor/pages/material/topico3.html
    </apostilaTemURL>
    <rdfs:label xml:lang="pt">
        Apostila para o Tópico Classificação das Rochas Carbonáticas
    </rdfs:label>
    <objetoTemConceito rdf:resource="http://www.owl-
ontologies.com/Ontology1242432309.owl#Classificacao"/>
    <objetoTemConceito
rdf:resource="http://www.owl-ontologies.com/Ontology1242432309.owl#Grabau"/>
    <objetoTemConceito
rdf:resource="http://www.owl-ontologies.com/Ontology1242432309.owl#Folk"/>
    <objetoTemConceito
rdf:resource="http://www.owl-ontologies.com/Ontology1242432309.owl#Dunham"/>
</Apostila>
```

Figura 38: Código OWL para descrição de um tópico

A classe apostila possui também a propriedade comentário. O comentário deve ser criado para explicar resumidamente ao que se refere o texto da apostila.

Finalmente, é realizada a associação do conteúdo, já descrito na ontologia de materiais de aprendizagem, com o conceito na ontologia de domínio. A associação é realizada com a propriedade “objetoTemConceito” da classe apostila. Essa associação pode ser repetida quantas vezes necessária, como por exemplo, o conteúdo na apostila “Classificação de Rochas Carbonáticas” (Figura 38) é associada a quatro conceitos: Classificação, Grabau, Folk e Dunham. Os conceitos na ontologia de domínio que descrevem os conceitos (Classificação, Grabau, Folk e Dunham) são relacionados através da propriedade “objetoTemConceito” na descrição do conteúdo da apostila.

Como a apostila é um tipo objeto de aprendizagem, essa apostila já pode ser utilizada para compor um material de aprendizagem. O material de aprendizagem possui

somente uma propriedade chamada “materialTemObjetos”, que é utilizada para associação de um ou mais objetos de aprendizagem. Assim, um material de aprendizagem é uma agregação de objetos de aprendizagem. Para o exemplo citado, uma instância de material de aprendizagem foi criada com o nome de “Material_Classificacao” e sua propriedade “materialTemObjetos” foi associada a “Apostila_Classificacao”.

5.3.1.4 Definição da estrutura do curso

Nessa fase é feita na ontologia de materiais de aprendizagem a definição do curso e sua estrutura, composta por disciplinas e tópicos de cada disciplina.

Inicialmente é feita a definição dos tópicos. Os tópicos são criados na ontologia como instâncias da classe Tópico.

No curso de Rochas Carbonáticas, foi criada uma instância da classe Tópico chamada “Classificacao” (Figura 39). Os tópicos definem a sequência que os materiais de aprendizagem deverão ser apresentados. Para definir a sequência de tópicos são utilizadas as propriedades “topicoTemAnterior” e “topicoTemProximo” da classe “Topico”. Por exemplo, o tópico “Classificacao” é posterior ao tópico “Componentes_Texturais” e anterior ao tópico “Diagenese”. Na Figura 39, é apresentada a instância do tópico “Classificacao” e suas propriedades “topicoTemAnterior” preenchida com “Componentes_Texturais e a propriedade “topicoTemProximo” preenchida com “Diagenese”.

O tópico “Classificacao” foi associado ao material de aprendizagem “Material_Classificacao” utilizando a propriedade “topicoTemMaterialAprendizagem”.



Figura 39: Cadastro do tópico Classificação no Protégé

Para finalizar a definição do curso, é criada uma instância da classe “Disciplina” chamada “Rochas_Carbonaticas” e o um instância da classe “Curso” chamada “Geologia”. A disciplina é associada ao curso através da propriedade “cursoTemDisciplina” da classe “Curso” e os tópicos são associados a disciplina pela propriedade “disciplinaTemTopicos” da classe “Disciplina”.

5.3.1.5 Cadastro de Perguntas

As perguntas utilizadas no curso devem ser cadastradas na ontologia de materiais de aprendizagem e os conceitos tratados em cada pergunta devem estar associados aos conceitos definidos na ontologia de domínio.

Em cada pergunta pode ser inserida uma imagem. A propriedade que associa a imagem a pergunta é “perguntaTemImagem” (Figura 40).

```
<Subjetiva rdf:ID="Subjetiva_159">
    <perguntaTemImagem>
      <Imagem rdf:ID="Imagem_76">
        <imagemTemURL rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
         http://localhost:8084/Tutor/imagens/img0076.jpg
        </imagemTemURL>
      </Imagem>
    </perguntaTemImagem>
    <objetoTemConceito
rdf:resource="http://www.owl-ontologies.com/Ontology1242432309.owl#Dunham"/>
    <perguntaTemEnunciado xml:lang="pt">
      De acordo com Dunham, qual é o tipo de rocha da amostra?
    </perguntaTemEnunciado>
    <perguntaTemCodigo rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
      6
    </perguntaTemCodigo>
    <perguntaTemDificuldade rdf:resource="#Facil"/>
    <perguntaTemTipo rdf:resource="#Subjetiva"/>
    <perguntaTemDica xml:lang="pt">
      Observe o componente entre os grãos
    </perguntaTemDica>
    <perguntaTemRespostaCorreta
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      Packstone
    </perguntaTemRespostaCorreta>
</Subjetiva>
```

Figura 40: Código OWL para pergunta

No exemplo, é definida uma pergunta subjetiva, representada pela instância “Subjetiva_159”, que possui o seguinte enunciado: “De acordo com Dunham, qual é o tipo de rocha da amostra?”. Esse enunciado se refere a uma imagem contendo uma amostra de rocha, que está no endereço “http://localhost:8084/Tutor/imagens/img0076.jpg”.

Outra propriedade da classe “Pergunta” é “perguntaTemDica”, que é utilizada para fornecer dicas para o aluno. No exemplo existe apenas uma dica, mas podem ser associadas várias dicas para cada pergunta.

Por fim, associada a cada pergunta deve ter uma resposta correta na propriedade “perguntaTemRespostaCorreta”, no exemplo (Figura 40) a resposta é “Packstone”.

5.3.2 Exemplo da utilização do SWBES

Após finalizada a construção do curso de Rochas Carbonáticas, o sistema pode ser acessado utilizando um navegador. A página de autenticação é a primeira página a ser exibida. Após autenticar o usuário, o sistema apresenta a página principal do curso (Figura 41).

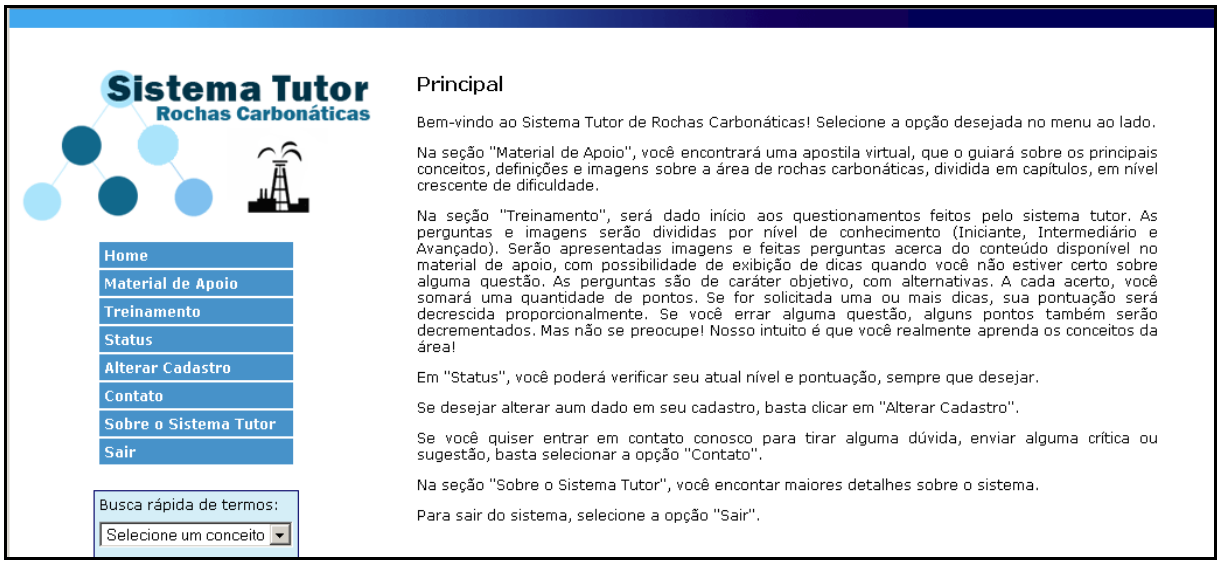


Figura 41: Tela inicial do curso de Rochas Carbonáticas

Na Figura 41, são apresentadas as funcionalidades da ferramenta. No *menu* lateral estão localizadas as ferramentas para acesso aos materiais de aprendizagem (Material de Apoio), as perguntas (Treinamento), a pontuação do aluno (Status), alteração de dados cadastrais e o envio de mensagens para o administrador.

Abaixo do *menu*, está disponível um glossário onde estão listados todos os conceitos da ontologia de domínio. Os alunos podem acessar esse glossário para visualizar

a descrição de cada conceito cadastrado.

5.3.2.1 Material de Apoio

O *link* “Material de Apoio” permite o acesso do aluno aos materiais de estudo (apostila virtual) sobre a área de Rochas Carbonáticas (Figura 42), esses materiais foram construídos e anotados (sessões 5.3.1.2 e 5.3.1.3) e a sequência de exibição definida (seção 5.3.1.4).

O *link* fornece acesso ao material utilizando a URL cadastrada na Ontologia de Materiais de Aprendizagem. Como por exemplo, se o aluno clicar no *link* “Classificação de Rochas Carbonáticas”, a apostila de classificação (Figura 37) será exibida para o aluno.

Caso o sistema identifique que o aluno apresenta alguma dificuldade, novos materiais serão buscados e publicados. O *link* dos novos materiais serão exibidos para o aluno nessa página.

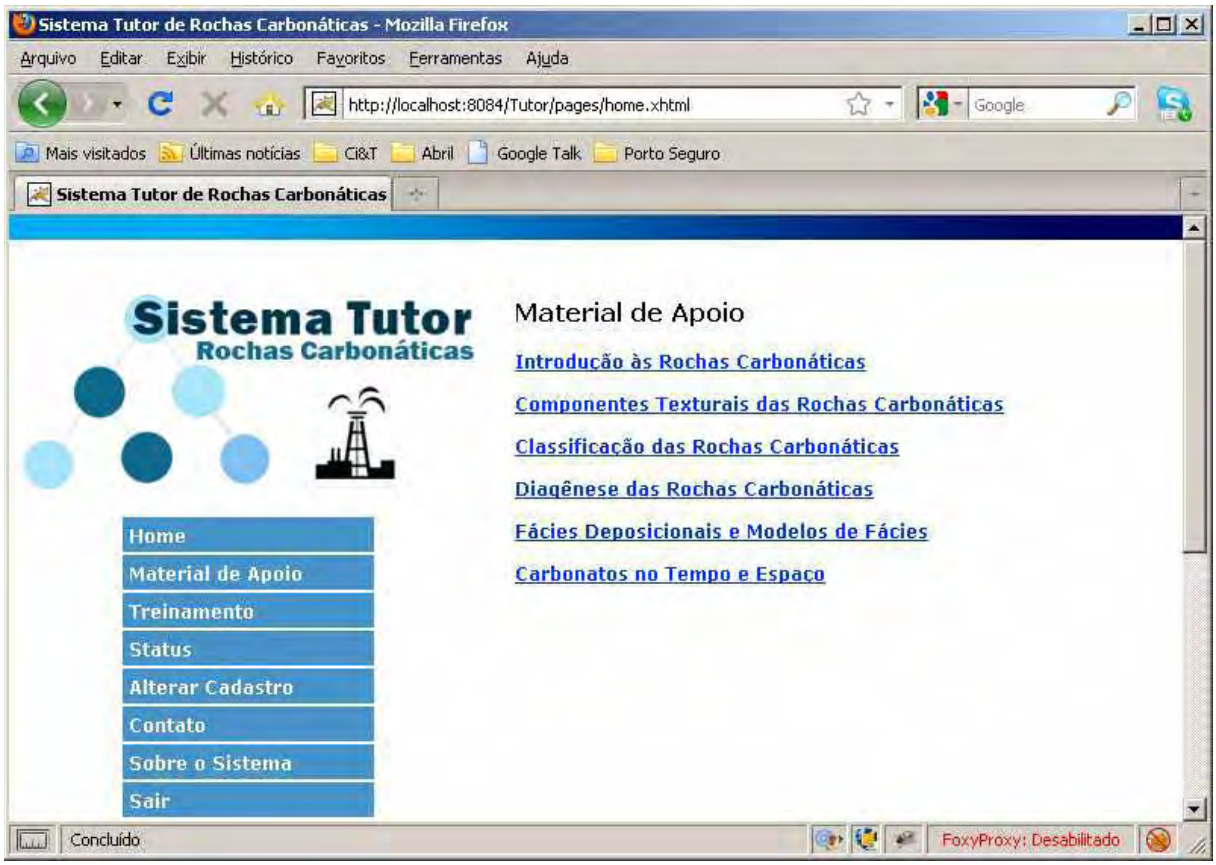


Figura 42: SWBES para Rochas Carbonáticas - Material de Apoio

5.3.2.2 Treinamento

No *link* “Treinamento” (Figura 43), o sistema apresenta as perguntas cadastradas na Ontologia de Materiais De Aprendizagem (Seção 5.3.1.5). As perguntas serão selecionadas de acordo com o nível do aluno, como por exemplo, se o aluno estiver no nível iniciante, apenas perguntas com nível de dificuldade “fácil” serão exibidas.

O SWBES seleciona uma pergunta aleatória, excluindo as perguntas que o aluno já respondeu corretamente. Se uma pergunta é do tipo objetiva, serão exibidas opções para o aluno selecionar a resposta que considera correta. Se a pergunta for subjetiva, uma caixa para entrada de texto será exibida para o aluno digitar sua resposta, nesse caso a resposta deve ser exata. Como por exemplo, seguindo o cadastro realizado na seção 5.3.1.5 , a pergunta subjetiva será exibida conforme mostrado na Figura 43.

Nesse primeiro momento, o aluno responde perguntas mais básicas (por exemplo, sobre classificação da rocha), respondendo posteriormente a perguntas de maior dificuldade. As respostas dessas perguntas são comparadas com os dados codificados na Ontologia de Materiais de Aprendizagem.

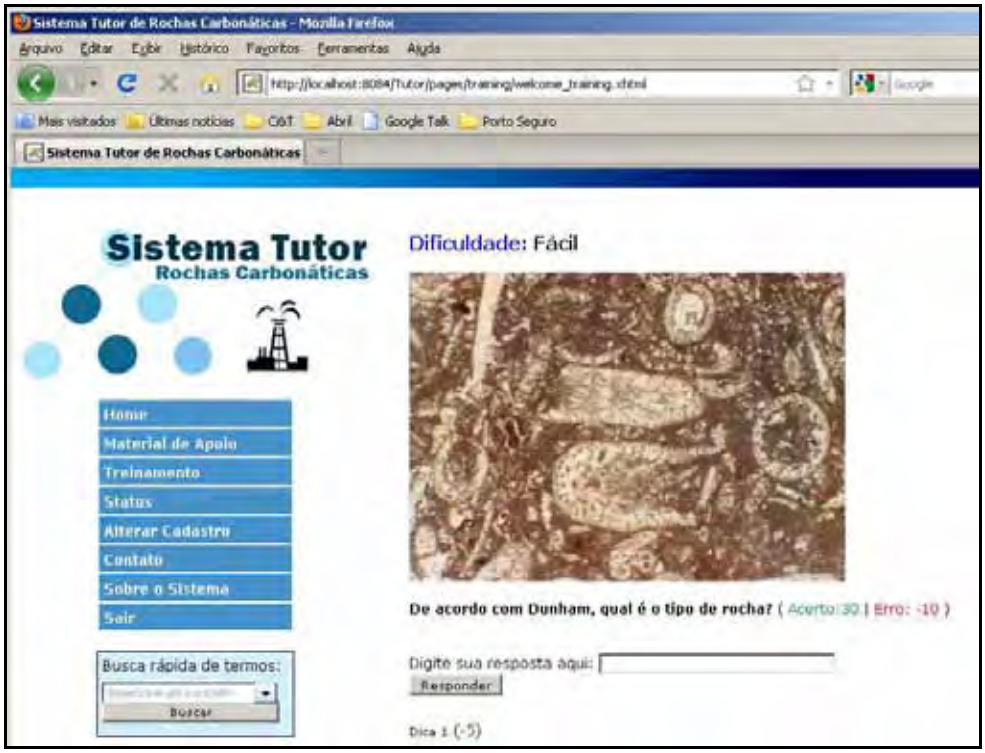


Figura 43: Exemplo da ferramenta Treinamento

A cada questão que o aluno responde corretamente será acrescido 30 pontos na pontuação atual do aluno (Figura 44).

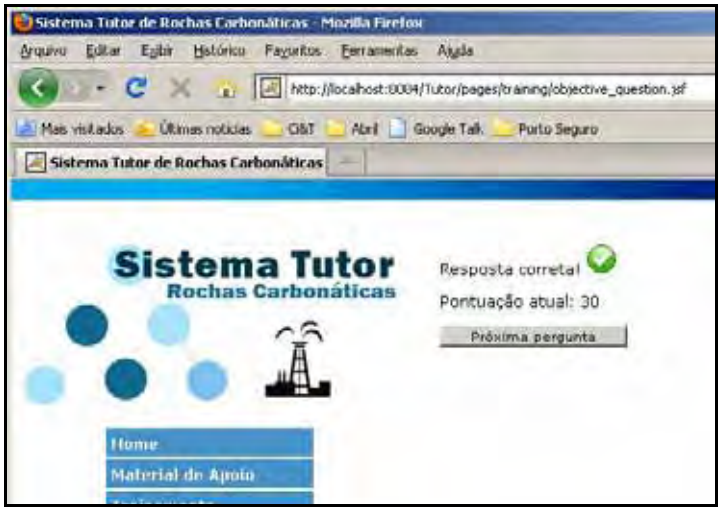


Figura 44: Página de Treinamento - Resposta correta

A cada resposta errada fornecida pelo aluno, sua pontuação é decrementada em 10 pontos. Caso o aluno não tenha pontos, sua pontuação é mantida zerada (Figura 45).

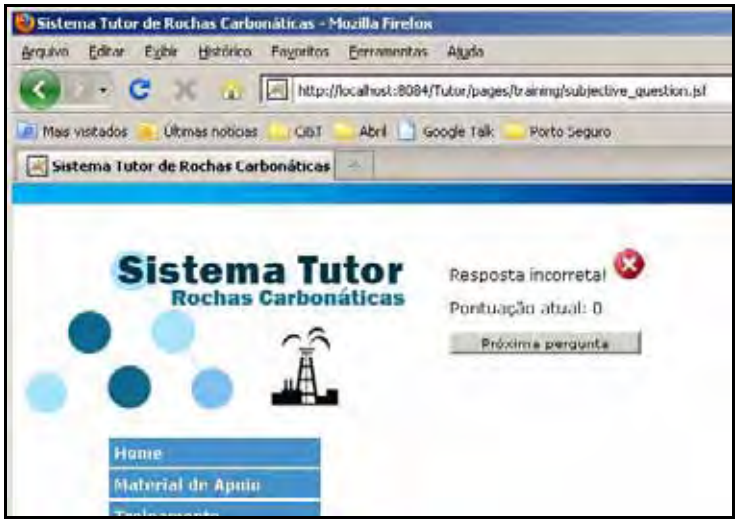


Figura 45: Página de Treinamento - Resposta incorreta

O sistema pode auxiliar o aluno, quando este não souber a resposta de uma pergunta, apresentando dicas. As dicas gravadas na Ontologia de Materiais de Aprendizagem ficam ocultas para o aluno, um *link* chamado “Dica” é exibido. O texto contendo a dica só aparece se o aluno clicar nesse *link*. Ao solicitar uma dica, o aluno não terá a mesma pontuação que teria se tivesse respondido corretamente, sem ajuda. A cada dica visualizada, o valor do acerto da questão é decrementado em 5 pontos (Figura 46).

Para o sistema mudar o nível de dificuldade das perguntas, é necessário que o

aluno obtenha uma pontuação correspondente a 50% do total de pontos do seu nível. Esse total de pontos é calculado de acordo com o número de questões cadastradas em cada nível. O aluno também muda de nível quando responde corretamente todas as questões do nível em que se encontra. Isso pode acontecer quando o aluno contabilizar muitos erros ou solicitar muitas dicas.

As respostas desse questionário servirão de base para os agentes avaliarem o andamento do aluno (Apêndice E). Cada questão é associada a um conceito da ontologia de domínio, assim os agentes conseguirão identificar qual conceito o aluno está tendo mais dificuldade e saberão como auxiliá-lo sem intervenção humana.

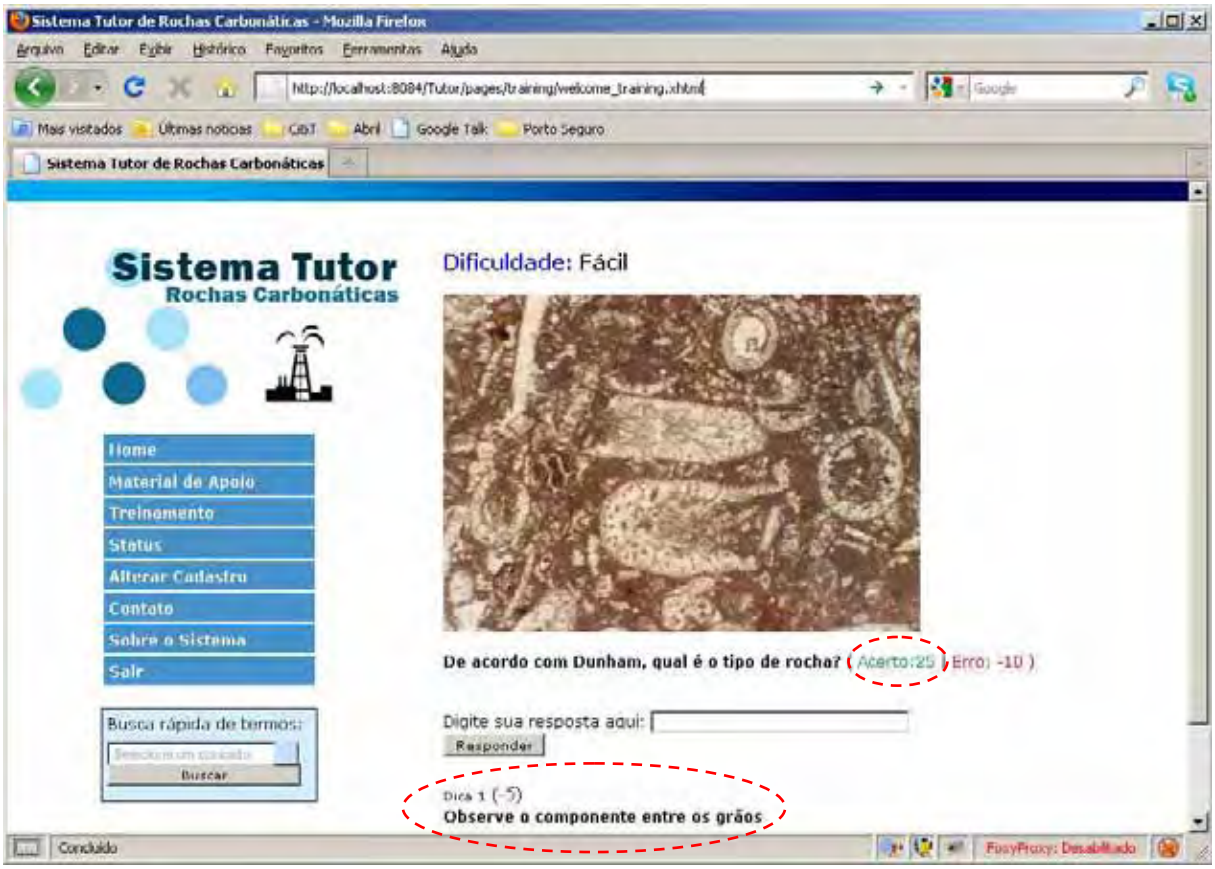


Figura 46: Exemplo da ferramenta Treinamento – Exibição de Dica

6 Conclusão e Considerações Finais

O objetivo inicialmente proposto desse trabalho consistia na definição e implementação de uma arquitetura para um SWBES utilizando a metodologia de desenvolvimento multiagente, o qual foi plenamente atingido.

O uso de metodologias para o desenvolvimento de sistemas multiagentes se mostrou de extrema importância para a criação de um *software* aderente aos requisitos definidos.

No estudo das arquiteturas de SWBES existentes foi constatada a existência de poucas arquiteturas baseadas na *Web* Semântica nas literaturas atuais. Em especial arquiteturas para sistemas voltados para a área da educação.

Na modelagem do sistema multiagentes ficou evidente a importância da utilização de uma metodologia de desenvolvimento orientada a agentes. A riqueza de detalhes levantados durante o processo de modelagem evitaram grandes retrabalhos que poderiam ocorrer na fase de implementação. Em referência a metodologia Tropos, considero que sua abordagem é bastante abrangente, mas ainda falta uma definição na forma de documentar os conteúdos dos repositórios de dados.

No desenvolvimento e implementação da arquitetura do SWBES, embora tenha sido uma tarefa complexa, com a utilização do *framework* JADEX e da linguagem JAVA foi possível implementar os principais componentes.

Na implementação do estudo de caso, um curso introdutório de Rochas Carbonáticas foi construído. O conteúdo do curso e as ontologias requeridas foram implementadas. Alguns testes foram realizados, demonstrando ser uma abordagem interessante para a automatização das tarefas realizadas pelos professores. Um aspecto importante foi constatado, que para essa automatização ocorra de maneira eficaz, é requerido um esforço inicial por parte dos professores para o correto cadastramento e anotação dos conteúdos de ensino. No entanto, quando utilizada em larga escala, esse esforço é compensado.

Uma contribuição deste trabalho foi a definição de uma arquitetura orientada a agentes para um SWBES, utilizando uma metodologia para seu desenvolvimento.

Em continuidade desse trabalho, diversos trabalhos futuros podem ser desenvolvidos:

- finalizar a implementação dos componentes da arquitetura;
- implementar os agentes Tutor e Avaliador Social com o objetivo de integrar o SWBES a uma Redes Social Semântica;
- utilizar a arquitetura proposta para o desenvolvimento de novos SWBES;

- realizar a integração da arquitetura com sistemas de ensino *web* disponíveis;
- desenvolver novas formas de automatizar a avaliação do aluno;
- desenvolver novas formas de busca semântica para pesquisa de materiais;
- integrar a arquitetura a um anotador semântico com o objetivo de automatizar a tarefa de anotação do materiais.

REFERÊNCIAS

ABDALLA, Gabriel; PEGORARO, Stefhan Fugagnoli; DE ROSSI, Luís Henrique; TIBANA, Paulo; GUILHERME, Ivan Rizzo; **Modelagem de uma Ontologia de Rochas Carbonáticas**. Rio Claro: Relatório Técnico do Grupo de Inteligência Artificial / DEMAC / UNESP, 2009. Disponível em <http://gia01.rc.unesp.br/>. Acesso em 02 dez. 2009.

ANDERSON, Terry; WHITELOCK, Denise. **The educational semantic web: visioning and practicing the future of education**, Journal of Interactive Media in Education – JIME 7 2004.

ANTONIOU, Grigoris; VAN HARMELEN, Frank. **A semantic *Web* primer.** Massachusetts: MIT Press, 2004. 288 p.

ARIADNE - **Alliance of Remote Intructional Authoring and Distribution Networks for Europe** [S.l.], [200-]. Disponível em: <http://www.ariadne-eu.org/>. Acesso em 7 jan. 2011.

ARAUJO, Moysés de. **Educação a distância e a WEB semântica:** modelagem ontológica de materiais e objetos de aprendizagem para a plataforma COL. 2003. 173f. Tese (Doutorado em Engenharia) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2003.

BITTENCOURT, Ig Ibert; Costa, Evandro; Silva, Marlos; Soares, Elvys; **A computational model for developing semantic web-based educational systems**. Knowledge Based Systems, v. 22, n. 4, p. 302-315, 2009

BLACKBOARD. **Blackboard**. Disponível em: <http://www.blackboard.com>. Acesso em: 10 fev. 2009.

BORBA, Marcelo Carvalho; MALHEIROS, Ana Paula do Santos; ZULATTO, Rúbia Barcelos Amaral, **Educação a Distância *online***. Coleção Tendências em Educação Matemática; Ed. Belo Horizonte: Autêntica, 2007.

BRANCO NETO, Wilson Castelo. **Web semântica na construção de sistemas de aprendizagem adaptativos**. 2006. 220f. Tese (Doutorado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis, 2006.

BRAUBACH, Lars; POKAHR, Alexander; MOLDT, Daniel; LAMERSDORF, Winfried. **Goal Representation for BDI Agent Systems**, in: Proceedings of the Second International Workshop on Programming Multiagent Systems: Languages, frameworks, techniques, and tools (ProMAS04). Springer Verlag, Berlin, pp 9-20, 2004.

BRAUBACH, Lars; POKAHR Alexander; LAMERSDORF, Winfried. **Jadex: A BDI-Agent System Combining Middleware and Reasoning**. In Software Agent-Based Applications, Platforms and Development Kits. p. 143-168. 2005.

BRAY, Tim; PAOLI, Jean; SPERBERG-MCQUEEN, Michael; MALER, Eve; YEARGAL, François. **Extensible Markup Language (XML) 1.0 (Fifth Edition)**, W3C Recommendation. 2008. Disponível em: <http://www.w3.org/TR/xml/>. Acesso em: 10 fev. 2009.

BRESCIANI, Paolo; PERINI, Anna; GIORGINI, Paolo; GIUNCHIGLIA, Fausto. **Tropos: an agent-oriented software development methodology. Autonomous Agents and Multi-**

Agent Systems, Norwell, v. 8, n. 3, p. 203-236, 2004.

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

BRICKLEY, Dan; GUHA, Ramanathan.**RDF Vocabulary Description Language 1.0: RDF Schema.** W3C Recommendation, 2004. Disponível em: <http://www.w3.org/TR/rdf-schema/>. Acesso em: 10 fev. 2009.

CHUNG, Lawrence, Nixon, Brian A., Yu, Eric, Mylopoulos, J. **Non-Functional Requirements in Software Engineering**. Publisher: Springer; 1st edition, 1999.

COSTA, Evandro B. **Inteligência artificial em educação presencial e à distância:** modelos, tecnologias, aplicações e tendências. Tutorial apresentado no Simpósio Brasileiro de Inteligencia Artificial, [S.l.: s.n.], 2000.

FALLSIDE, David; WALMSLEY, Priscilla. **XML Schema Part 0: Primer Second Edition.** W3C Recommendation. 2004. Disponível em: <http://www.w3.org/TR/xmlschema-0/>. Acesso em: 10 jan. 2010.

FREEDMAN, Reva; ALI, Syed. S.; MCROY, Susan. **What is an Intelligent Tutoring System?** Intelligence, v.11, n.3:, 2000. p. 15–16, ACM.

FURGERI, Sérgio. **Ensino didático da linguagem XML:** aprenda a criar padrões e documentos inteligentes com a XML. São Paulo: Érica, 2001.

GÓMEZ-PÉREZ, Asunción; FERNÁNDEZ-LÓPEZ, Mariano; CORCHO, Oscar. **Ontological Engineering**: with Examples from the Areas of Knowledge Management, e-commerce and the Semantic Web. London: Springer-Verlag, 2005. 403 p.

GOÑI, Jorge Luis; FERNANDES, Maria Cristina Pfeiffer; LUCENA, Carlos José Pereira de. **E-learning e a *Web*Semântica**. Rio de Janeiro: Pontifícia Universidade Católica do Rio de Janeiro, 2003. Disponível em: <http://www.dbd.puc-rio.br/depto_informatica/02_12_goni.pdf>. Acesso em: 10 jan. 2010.

GRUBER, Tom Robert. **A translation approach to portable ontologies.** Knowledge Acquisition, Elsevier, v.5, n. 2 : p.199-220, 1993.

GRUNINGER, Michael; FOX, Mark. **Methodology for the Design and Evaluation of Ontologies**. Proceedings of IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing. 1995

GUARINO, Nicola. Formal **Ontology and Information Systems**. In N. Guarino (ed.) Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, June 6-8, 1998. IOS Press, Amsterdam, pp. 3-15

HEITMANN, Benjamin; HAYES, Conor; OREN, Eyal. **Towards a reference architecture for Semantic Web applications,** in: Proceedings of the 1st International Web Science Conference, 2009.

HENDERSON-SELLERS, Brian; GIORGINI, Paolo. **Agent Oriented Methodologies.** Publisher: Idea Group Publishing.--- B. Henderson-Sellers and P. Giorgini, editors. Agent-Oriented Methodologies. Idea Group Inc., 2005.

HORRIDGE, Matthew. KNUBLAUCH, Holger; RECTOR, Alan; STEVENS, Robert; WROE, Chris. **Practical guide to building owl ontologies using the protégé-owl plugin and Co-ode tools**. Manchester: The university of Manchester, 2007. Disponível em: <http://www.co-ode.org/resources/tutorials/ProtegeOWL/Tutorial.pdf>. Acesso em: 11 mar. 2009.

KINNY, David; GEORGEFF, Michael; RAO, Anand. **A Methodology and Modelling Technique for Systems of BDI Agents**. In proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away. p. 56 – 71. 1996.

KOLP, Manoel; MYLOPOULOS, John; GIORGINI; Paolo. **Multi-Agent Architectures as Organizational Structures.** Hingham, v. 13, n. 1, p. 3-25, 2006.

LACY, Lee. **OWL:** representing information using the *Web* ontology language. Victoria, Canada: Trafford, 2005. 285p.

LLOP, Roberto Romero. **Especificación OLW de una ontología para teleeducación en la *Web* semântica**. Valencia: Departamento de Comunicaciones – Universidad Politecnica de Valencia, 2007.

LOM - **Learning Object Metadata** [S.l.], [200-]. Disponível em: <http://ltsc.ieee.org/wg12/> . Acesso em 7 jan. 2011.

LTSC - **Technology Standards Committee** [S.l.], [200-]. Disponível em: <http://www.imsglobal.org/>. Acesso em 7 jan. 2011.

MANOLA, Frank; MILLER, Eric. **RDF Primer.** W3C Recommendation. 2004. Disponível em: <http://www.w3.org/TR/rdf-primer/>. Acesso em 8 jan. 2010.

MCGUINNESS, Deborah Louise; VAN HARMELEN, Frank. **OWL Web Ontology Language.** W3C Recommendation, 2004. Disponível em: <http://www.w3.org/TR/owl-features/>. Acesso em 7 jan. 2010.

MORANDINI, Mirko; NGUYEN, Duy Cu; PERINI, Anna; SIENA, Alberto; SUSI, Angelo. **Tool-supported Development with Tropos: The Conference Management System Case Study**. 8th International Workshop on Agent-Oriented Software Engineering, AOSE 2007, Honolulu, HI, USA, Maio 2007.

MOODLE. **Standard Moodle Package**. Disponível em: <http://moodle.org>. Acesso em: 10 fev. 2009.

PRADO, Simone das Graças Domingues. **Um Experimento do uso de Ontologias para Reforço de Aprendizagem em Educação a distância**. 2004. 177f. Tese (Doutorado em Engenharia) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2004.

PROTEGE - **The Protégé Ontology Editor and Knowledge Acquisition System**. 2009. Sítio do programa. Disponível em: <http://protege.stanford.edu/>. Acesso em: 30 nov. 2009.

RAO, Anand.; GEORGEFF, Michael. **Modeling rational agents within a BDI-Architecture**. In Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, p. 473-484, 1991.

SCORM - **Sharable Content Object Reference Model**. [S.l.], [200-]. Disponível em: <http://www.adlnet.gov> . Acesso em 7 jan. 2011.

SILVA, Maria Jocélia; **U-TROPOS: uma proposta de processo unificado para apoiar o desenvolvimento de software orientado a agentes**. 2008. 151f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Pernanbuco, Recife, 2008.

STOJANOVIC, Ljiljana; STAAB, Steffen; STUDER, Rudi. Elearning based on the Semantic *Web*. In: WORLD CONFERENCE ON THE WWW AND INTERNET, 2001, Orlando. **Anais eletrônicos...** Orlando: *Web*Net, 2001. Disponível em: <http://www.aifb.uni-karlsruhe.de /WBS/Publ/2001/*Web*Net_lstsstrst_2001.pdf>. Acesso em: 18 dez. 2008.

TAOM4E. ***Tool* for Agent Oriented Modeling**. A Project of the SE Division at FBK - fondazione Bruno Kessler. Trento, Italy. Disponível em: <http://sra.itc.it/*Tool*s/taom4e/>. Acessado em: 27 de junho de 2010.

TIDIA-AE. **TIDIA-AE**. Disponível em: <http://tidia-ae.usp.br/portal>. Acesso em: 10 fev. 2009.

USCHOLD, Mike; KING, Martin. **Towards a Methodology for Building Ontologies**. Proceedings of IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing, 1995.

VALENTE, José Armando. Informática na educação no Brasil: análise e contextualização histórica. In: VALENTE, José Armando (Org.). **O computador na sociedade do conhecimento.** Campinas: NIED-UNICAMP, 1999.

W3SCHOOLS. **URIs e XML *namespaces***. Disponível em: <http://www.w3sch oools.com/xml/xml_*namespaces*.asp>. Acesso em 11 mar. 2009.

W3SCHOOLS [B]. **RDF**. Disponível em: <http://www.w3schools.com/rdf/rdf _rules.asp>. Acesso em 12 mar. 2009.

W3SCHOOLS [C]. **RDF Schema**. Disponível em: <http://www.w3schools.com /rdf/rdf_schema.asp>. Acesso em 12 mar. 2009.

WOOLDRIDGE, Michael. **An Introduction to MultiAgent System** , Wiley; 2nd edition, 2009.

ZAMBONELLI, Franco; JENNINGS, Nicholas R.; WOOLDRIDGEZ, Michael. Developing multiagent systems: the gaia methodology. **ACM Transactions on Software Engineering and Methodology**, New York, v. 12, n. 3, p. 417-470, 2003.

Apêndice A – Agente Controlador

```
<!--
       Taom4e "t2x Tool" auto-generated Jadex ADF file.
       Automatic implementation of Tropos goal diagrams to Jadex.
       Author: Mirko Morandini, FBK-irsT / University of Trento (I), 2006-2008
       Please change it to your needs.
-->
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-0.96.xsd"
       name="Controlador"
       package="AgentRole_Controlador">

       <imports>
              <import>jadex.util.*</import>
              <import>jadex.adapter.fipa.*</import>
              <import>jadex.runtime.*</import>
              <import>java.util.logging.*</import>
              <import>java.net.Socket</import>
              <import>jadex.planlib.*</import>
              <import>AgentRole_Controlador.plans.*</import>
              <import>AgentRole_Controlador.util.kl.plans.*</import>
              <import>AgentRole_Controlador.util.kl.components.*</import>
              <!--import the package of the environmental entity classes. Adapt if
you have a package structure.-->
              <!--empty if no environment defined-->
       </imports>

       <capabilities>
         <capability name="procap" file="jadex.planlib.Protocols" />
         <capability name="dfcap" file="jadex.planlib.DF" />
       </capabilities>


       <beliefs>
              <!-- The belief contains the tropos hierarchy as facts. -->
              <beliefset name="goals" class="TGoal">
                     <fact>Components.createGoal("CoordenarAgentes","ME")</fact>
              </beliefset>
              <beliefset name="softgoals" class="TSoftgoal">
              </beliefset>
              <beliefset name="decomp" class="TLink">
              </beliefset>
              <beliefset name="meansend" class="TLink">
```

```
                <fact>new TLink("CoordenarAgentes","IniciarAgente")</fact>
            </beliefset>
            <beliefset name="contributions" class="TContrib">
            </beliefset>
            <beliefset name="dependencies" class="TDependency">
                <!--TDependency:  'why'-Goal(must  be  AND-decomposed),  Dependum,
Dependee -->
            </beliefset>
            <!--$ENTITIES-->

            <!--$RESOURCES-->

            <!--To force reevaluation of all conditions-->
            <belief name="counter" class="Integer">
            <fact evaluationmode="dynamic"> 0 </fact>
      </belief>
      </beliefs>

      <goals>
            <!-- default values: metalevelreasoning enabled, exclude when tried. -
->
            <achievegoal name="CoordenarAgentes" exclude="when_failed">
                <parameter name="param" class="String">
                    <value>""</value>
                </parameter>
                <parameter name="result" class="String" direction="out"/>
            </achievegoal>

            <achievegoalref name="rp_initiate">
                <concrete ref="procap.rp_initiate" />
            </achievegoalref>

            <!--  procura por agentes que realizem o servico desejado-->
            <achievegoalref name="df_search">
                <concrete ref="dfcap.df_search" />
            </achievegoalref>
      </goals>

      <plans>

            <plan name="informChangePlan">
                <body>new InformChangePlan()</body>
                <trigger>
                    <messageevent ref="inform_ChangeEnv"/>
                </trigger>
            </plan>
```

```
            <!-- Real Plans that hold the activity part of a capability and "do"
the requested things. -->
            <plan name="realPlan_IniciarAgente">
                  <parameter name="param" class="String">
                        <goalmapping ref="CoordenarAgentes.param"/>
                  </parameter>
                  <parameter name="result" class="String">
                        <goalmapping ref="CoordenarAgentes.result"/>
                  </parameter>
                  <body>new RealPlan_IniciarAgente()</body>
                  <trigger>
                        <goal ref="CoordenarAgentes"/>
                  </trigger>
            </plan>

            <!-- Initial plan for starting a server thread waiting for client
connections.
                  Adds new sockets with the new connection to the client belief
set. -->
            <plan name="server">
                  <body>new ServerPlan(9099)</body>
            </plan>

      </plans>


      <events>
            <!-- Specifies a request to achieve a goal (one per goal). -->
            <messageevent    name="request_CoordenarAgentes"    direction="receive"
type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                  </parameter>
                  <parameter          name="content-start"          class="String"
direction="fixed">
                        <value>"CoordenarAgentes"</value>
                  </parameter>
            </messageevent>


            <!--messages for informs on environment changes-->
            <messageevent name="inform_ChangeEnv" direction="receive"
                  type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
```

```
                        <value>SFipa.INFORM</value>
                    </parameter>
                    <parameter          name="content-start"          class="String"
direction="fixed">
                        <value>"change "</value>
                    </parameter>
                </messageevent>

                <!--FIPA-messages needed to communicate with other agents-->

                <messageevent name="request" direction="send" type="fipa">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                    </parameter>
                    <parameter name="reply-with" class="String">

        <value>SFipa.createUniqueId($scope.getAgentName())</value>
                    </parameter>
                </messageevent>

                <!--default FIPA-messages needed to communicate with other agents-->
                <messageevent name="inform" direction="send_receive" type="fipa">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.INFORM</value>
                    </parameter>
                </messageevent>
                <messageevent name="agree" direction="send_receive" type="fipa">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.AGREE</value>
                    </parameter>
                </messageevent>
                <messageevent name="failure" direction="send_receive" type="fipa">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.FAILURE</value>
                    </parameter>
                </messageevent>
                <messageevent name="n_u" direction="send_receive" type="fipa">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.NOT_UNDERSTOOD</value>
                    </parameter>
                </messageevent>
                <messageevent name="refuse" direction="send_receive" type="fipa">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REFUSE</value>
                    </parameter>
                </messageevent>
```

```
	</events>

	<expressions>
		<!--All expressions are not changed during the automatic code generation-->
		<!-- This query selects the first matching entry from the dictionary,
			whereby the parameter $eword is compared to the first element of
			a belief set tuple. -->
		<expression name="query_link">
			select $link.getGoal(1)
			from TLink $link in $beliefbase.decomp
			where $link.get(0).equals($component)
			order by $link.getPriority() desc
			<parameter name="$component" class="String"/>
		</expression>
		<expression name="query_ME_link">
			select $link.get(1)
			from TLink $link in $beliefbase.meansend
			where $link.get(0).equals($component)
			<parameter name="$component" class="String"/>
		</expression>
		<expression name="query_contributions">
			select $link
			from TContrib $link in $beliefbase.contributions
			where $link.get(0).equals($component)
			<parameter name="$component" class="String"/>
		</expression>
		<expression name="query_dependencies">
			select $link
			from TDependency $link in $beliefbase.dependencies
			where $link.getWhyGoal().equals($component)
			<parameter name="$component" class="String"/>
		</expression>
	</expressions>

	<properties>
		<!-- Only log outputs >= level are printed. -->
	 <property name="logging.level">Level.INFO</property>
		<!--  The default parent handler prints out log messages on the console. -->
	 <property name="logging.useParentHandlers">true</property>
	   <!--<property name="debugging">true</property>-->
    </properties>

	<!-- initialstates changed to configurations in Jadex 0.96 -->
```

```
        <configurations>
                <configuration name="default">
                        <plans>
                                <initialplan ref="server"/>
                        </plans>
                </configuration>
        </configurations>
</agent>
```

Apêndice B – Agente Avaliador de Desempenho

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-0.96.xsd"
       name="AvaliadorDesempenho"
       package="AgentRole_AvaliadorDesempenho">

       <imports>
              <import>jadex.util.*</import>
              <import>jadex.adapter.fipa.*</import>
              <import>jadex.runtime.*</import>
              <import>java.util.logging.*</import>
              <import>jadex.planlib.*</import>
              <import>AgentRole_AvaliadorDesempenho.plans.*</import>
              <import>AgentRole_AvaliadorDesempenho.util.kl.plans.*</import>
              <import>AgentRole_AvaliadorDesempenho.util.kl.components.*</import>
       </imports>

       <capabilities>
              <capability name="procap" file="jadex.planlib.Protocols" />
              <capability name="dfcap" file="jadex.planlib.DF" />
       </capabilities>

       <beliefs>
              <beliefset name="goals" class="TGoal">
                     <fact>Components.createGoal("MonitorarDificuldades","ME")</fact>
              </beliefset>
              <beliefset name="meansend" class="TLink">
                     <fact>new
TLink("MonitorarDificuldades","AvaliarDesempenho")</fact>
              </beliefset>
              <beliefset name="dependencies" class="TDependency">
              </beliefset>
       </beliefs>

       <goals>
              <achievegoal name="MonitorarDificuldades" exclude="when_failed">
                     <parameter name="param" class="String">
                     </parameter>
                     <parameter name="result" class="String" direction="out"/>
              </achievegoal>

              <!--  manter o agente registrado como um servico -->
```

```
            <maintaingoalref name="df_keep_registered">
                  <concrete ref="dfcap.df_keep_registered" />
            </maintaingoalref>
      </goals>

      <plans>
            <plan name="requestPlan_MonitorarDificuldades">
                  <body>new GoalRequestPlan("MonitorarDificuldades")</body>
                  <trigger>
                        <messageevent ref="request_MonitorarDificuldades"/>
                  </trigger>
            </plan>
            <plan name="realPlan_AvaliarDesempenho">
                  <parameter name="param" class="String">
                        <goalmapping ref="MonitorarDificuldades.param"/>
                  </parameter>
                  <parameter name="result" class="String">
                        <goalmapping ref="MonitorarDificuldades.result"/>
                  </parameter>
                  <body>new RealPlan_AvaliarDesempenho()</body>
                  <trigger>
                        <goal ref="MonitorarDificuldades"/>
                  </trigger>
            </plan>
      </plans>


      <events>
            <messageevent  name="request_MonitorarDificuldades"  direction="receive"
type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                  </parameter>
            </messageevent>
            <messageevent name="request" direction="send" type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                  </parameter>
                  <parameter name="reply-with" class="String">

      <value>SFipa.createUniqueId($scope.getAgentName())</value>
                  </parameter>
            </messageevent>
            <messageevent name="inform" direction="send_receive" type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.INFORM</value>
```

```
            </parameter>
        </messageevent>
        <messageevent name="agree" direction="send_receive" type="fipa">
            <parameter name="performative" class="String" direction="fixed">
                <value>SFipa.AGREE</value>
            </parameter>
        </messageevent>
        <messageevent name="failure" direction="send_receive" type="fipa">
            <parameter name="performative" class="String" direction="fixed">
                <value>SFipa.FAILURE</value>
            </parameter>
        </messageevent>
        <messageevent name="n_u" direction="send_receive" type="fipa">
            <parameter name="performative" class="String" direction="fixed">
                <value>SFipa.NOT_UNDERSTOOD</value>
            </parameter>
        </messageevent>
        <messageevent name="refuse" direction="send_receive" type="fipa">
            <parameter name="performative" class="String" direction="fixed">
                <value>SFipa.REFUSE</value>
            </parameter>
        </messageevent>
    </events>

    <expressions>
        <expression name="query_link">
            select $link.getGoal(1)
            from TLink $link in $beliefbase.decomp
            where $link.get(0).equals($component)
            order by $link.getPriority() desc
            <parameter name="$component" class="String"/>
        </expression>
        <expression name="query_ME_link">
            select $link.get(1)
            from TLink $link in $beliefbase.meansend
            where $link.get(0).equals($component)
            <parameter name="$component" class="String"/>
        </expression>
        <expression name="query_contributions">
            select $link
            from TContrib $link in $beliefbase.contributions
            where $link.get(0).equals($component)
            <parameter name="$component" class="String"/>
        </expression>
        <expression name="query_dependencies">
            select $link
```

```
                    from TDependency $link in $beliefbase.dependencies
                    where $link.getWhyGoal().equals($component)
                    <parameter name="$component" class="String"/>
              </expression>
       </expressions>

       <properties>
              <property name="logging.level">Level.INFO</property>
              <property name="logging.useParentHandlers">true</property>
       </properties>

       <configurations>
              <configuration name="default">
                    <goals>
                          <initialgoal ref="df_keep_registered">
                                <parameter ref="description">

       <value>SFipa.createAgentDescription(null,SFipa.createServiceDescription("serv
ice_avaliar","avaliar", "br.unesp.gia"))</value>
                                </parameter>
                                <parameter ref="leasetime">
                                      <value>20000</value>
                                </parameter>
                          </initialgoal>
                    </goals>
              </configuration>
       </configurations>
</agent>
```

Apêndice C – Agente Buscador

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-0.96.xsd"
       name="Buscador"
       package="AgentRole_Buscador">

       <imports>
              <import>jadex.util.*</import>
              <import>jadex.adapter.fipa.*</import>
              <import>jadex.runtime.*</import>
              <import>java.util.logging.*</import>
              <import>jadex.planlib.*</import>
              <import>AgentRole_Buscador.plans.*</import>
              <import>AgentRole_Buscador.util.kl.plans.*</import>
              <import>AgentRole_Buscador.util.kl.components.*</import>
       </imports>

       <capabilities>
              <capability name="procap" file="jadex.planlib.Protocols" />
              <capability name="dfcap" file="jadex.planlib.DF" />
       </capabilities>

       <beliefs>
              <beliefset name="goals" class="TGoal">
                     <fact>Components.createGoal("FornecerMateriais","ME")</fact>
              </beliefset>
              <beliefset name="meansend" class="TLink">
                     <fact>new TLink("FornecerMateriais","BuscarMateriais")</fact>
              </beliefset>
       </beliefs>

       <goals>
              <achievegoal name="FornecerMateriais" exclude="when_failed">
                     <parameter name="param" class="String">
                     </parameter>
                     <parameter name="result" class="String" direction="out"/>
              </achievegoal>

              <!--  manter o agente registrado como um servico -->
              <maintaingoalref name="df_keep_registered">
                     <concrete ref="dfcap.df_keep_registered" />
              </maintaingoalref>
```

```
	</goals>

	<plans>
		<plan name="requestPlan_FornecerMateriais">
			<body>new GoalRequestPlan("FornecerMateriais")</body>
			<trigger>
				<messageevent ref="request_FornecerMateriais"/>
			</trigger>
		</plan>

		<!-- Real Plans that hold the activity part of a capability and "do"
the requested things. -->
		<plan name="realPlan_BuscarMateriais">
			<parameter name="param" class="String">
				<goalmapping ref="FornecerMateriais.param"/>
			</parameter>
			<parameter name="result" class="String">
				<goalmapping ref="FornecerMateriais.result"/>
			</parameter>
			<body>new RealPlan_BuscarMateriais()</body>
			<trigger>
				<goal ref="FornecerMateriais"/>
			</trigger>
		</plan>
	</plans>


	<events>
		<messageevent	name="request_FornecerMateriais"	direction="receive"
type="fipa">
			<parameter name="performative" class="String" direction="fixed">
				<value>SFipa.REQUEST</value>
			</parameter>
		</messageevent>

		<!--FIPA-messages needed to communicate with other agents-->

		<messageevent name="request" direction="send" type="fipa">
			<parameter name="performative" class="String" direction="fixed">
				<value>SFipa.REQUEST</value>
			</parameter>
			<parameter name="reply-with" class="String">

	<value>SFipa.createUniqueId($scope.getAgentName())</value>
			</parameter>
		</messageevent>
```

```
            <!--default FIPA-messages needed to communicate with other agents-->
            <messageevent name="inform" direction="send_receive" type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.INFORM</value>
                  </parameter>
            </messageevent>
            <messageevent name="agree" direction="send_receive" type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.AGREE</value>
                  </parameter>
            </messageevent>
            <messageevent name="failure" direction="send_receive" type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.FAILURE</value>
                  </parameter>
            </messageevent>
            <messageevent name="n_u" direction="send_receive" type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.NOT_UNDERSTOOD</value>
                  </parameter>
            </messageevent>
            <messageevent name="refuse" direction="send_receive" type="fipa">
                  <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REFUSE</value>
                  </parameter>
            </messageevent>

      </events>

      <expressions>
            <!--All expressions are not changed during the automatic code
generation-->
            <!-- This query selects the first matching entry from the dictionary,
                  whereby the parameter $eword is compared to the first element of
                  a belief set tuple. -->
            <expression name="query_link">
                  select $link.getGoal(1)
                  from TLink $link in $beliefbase.decomp
                  where $link.get(0).equals($component)
                  order by $link.getPriority() desc
                  <parameter name="$component" class="String"/>
            </expression>
            <expression name="query_ME_link">
                  select $link.get(1)
                  from TLink $link in $beliefbase.meansend
```

```
                where $link.get(0).equals($component)
                <parameter name="$component" class="String"/>
            </expression>
            <expression name="query_contributions">
                select $link
                from TContrib $link in $beliefbase.contributions
                where $link.get(0).equals($component)
                <parameter name="$component" class="String"/>
            </expression>
            <expression name="query_dependencies">
                select $link
                from TDependency $link in $beliefbase.dependencies
                where $link.getWhyGoal().equals($component)
                <parameter name="$component" class="String"/>
            </expression>
        </expressions>

        <properties>
         <property name="logging.level">Level.INFO</property>
         <property name="logging.useParentHandlers">true</property>
    </properties>

        <configurations>
            <configuration name="default">
                <goals>
                    <initialgoal ref="df_keep_registered">
                        <parameter ref="description">

        <value>SFipa.createAgentDescription(null,SFipa.createServiceDescription("serv
ice_buscar","buscar", "br.unesp.gia"))</value>
                        </parameter>
                        <parameter ref="leasetime">
                            <value>20000</value>
                        </parameter>
                    </initialgoal>
                </goals>
            </configuration>
        </configurations>
</agent>
```

Apêndice D – Agente Publicador

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-0.96.xsd"
       name="Publicador"
       package="AgentRole_Publicador">

       <imports>
              <import>jadex.util.*</import>
              <import>jadex.adapter.fipa.*</import>
              <import>jadex.runtime.*</import>
              <import>java.util.logging.*</import>
              <import>jadex.planlib.*</import>
              <import>AgentRole_Publicador.plans.*</import>
              <import>AgentRole_Publicador.util.kl.plans.*</import>
              <import>AgentRole_Publicador.util.kl.components.*</import>
       </imports>

       <capabilities>
              <capability name="procap" file="jadex.planlib.Protocols" />
              <capability name="dfcap" file="jadex.planlib.DF" />
       </capabilities>


       <beliefs>
              <beliefset name="goals" class="TGoal">
                     <fact>Components.createGoal("Publicar","ME")</fact>
              </beliefset>
              <beliefset name="meansend" class="TLink">
                     <fact>new TLink("Publicar","PublicarMateriais")</fact>
              </beliefset>
       </beliefs>

       <goals>
              <achievegoal name="Publicar" exclude="when_failed">
                     <parameter name="param" class="String">
                     </parameter>
                     <parameter name="result" class="String" direction="out"/>
              </achievegoal>

              <!--  manter o agente registrado como um servico -->
              <maintaingoalref name="df_keep_registered">
                     <concrete ref="dfcap.df_keep_registered" />
              </maintaingoalref>
```

```
</goals>

<plans>
	<plan name="requestPlan_Publicar">
		<body>new GoalRequestPlan("Publicar")</body>
		<trigger>
			<messageevent ref="request_Publicar"/>
		</trigger>
	</plan>
	<plan name="realPlan_PublicarMateriais">
		<parameter name="param" class="String">
			<goalmapping ref="Publicar.param"/>
		</parameter>
		<parameter name="result" class="String">
			<goalmapping ref="Publicar.result"/>
		</parameter>
		<body>new RealPlan_PublicarMateriais()</body>
		<trigger>
			<goal ref="Publicar"/>
		</trigger>
	</plan>
</plans>


<events>
	<messageevent name="request_Publicar" direction="receive" type="fipa">
		<parameter		name="performative"		class="String"
		direction="fixed">
			<value>SFipa.REQUEST</value>
		</parameter>
	</messageevent>

	<messageevent name="request" direction="send" type="fipa">
		<parameter		name="performative"		class="String"
		direction="fixed">
			<value>SFipa.REQUEST</value>
		</parameter>
		<parameter name="reply-with" class="String">

<value>SFipa.createUniqueId($scope.getAgentName())</value>
		</parameter>
	</messageevent>
	<messageevent name="inform" direction="send_receive" type="fipa">
		<parameter name="performative" class="String" direction="fixed">
			<value>SFipa.INFORM</value>
```

```
                </parameter>
            </messageevent>
            <messageevent name="agree" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                    <value>SFipa.AGREE</value>
                </parameter>
            </messageevent>
            <messageevent name="failure" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                    <value>SFipa.FAILURE</value>
                </parameter>
            </messageevent>
            <messageevent name="n_u" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                    <value>SFipa.NOT_UNDERSTOOD</value>
                </parameter>
            </messageevent>
            <messageevent name="refuse" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                    <value>SFipa.REFUSE</value>
                </parameter>
            </messageevent>

        </events>

        <expressions>
            <!--All expressions are not changed during the automatic code
generation-->
            <!-- This query selects the first matching entry from the dictionary,
                whereby the parameter $eword is compared to the first element of
                a belief set tuple. -->
            <expression name="query_link">
                select $link.getGoal(1)
                from TLink $link in $beliefbase.decomp
                where $link.get(0).equals($component)
                order by $link.getPriority() desc
                <parameter name="$component" class="String"/>
            </expression>
            <expression name="query_ME_link">
                select $link.get(1)
                from TLink $link in $beliefbase.meansend
                where $link.get(0).equals($component)
                <parameter name="$component" class="String"/>
            </expression>
            <expression name="query_contributions">
                select $link
```

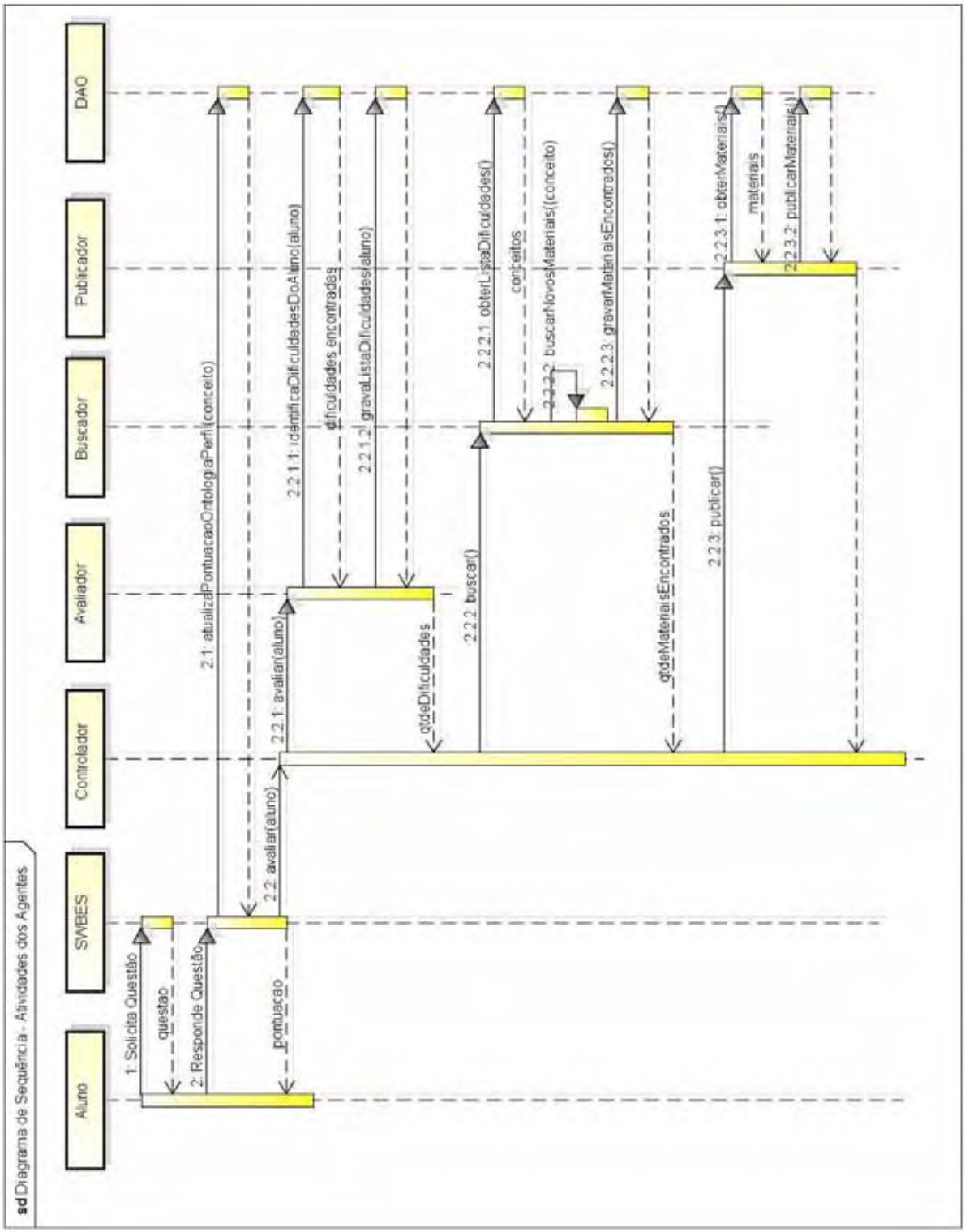
```
                from TContrib $link in $beliefbase.contributions
                where $link.get(0).equals($component)
                <parameter name="$component" class="String"/>
            </expression>
            <expression name="query_dependencies">
                select $link
                from TDependency $link in $beliefbase.dependencies
                where $link.getWhyGoal().equals($component)
                <parameter name="$component" class="String"/>
            </expression>
        </expressions>

        <properties>
         <property name="logging.level">Level.INFO</property>
         <property name="logging.useParentHandlers">true</property>
    </properties>

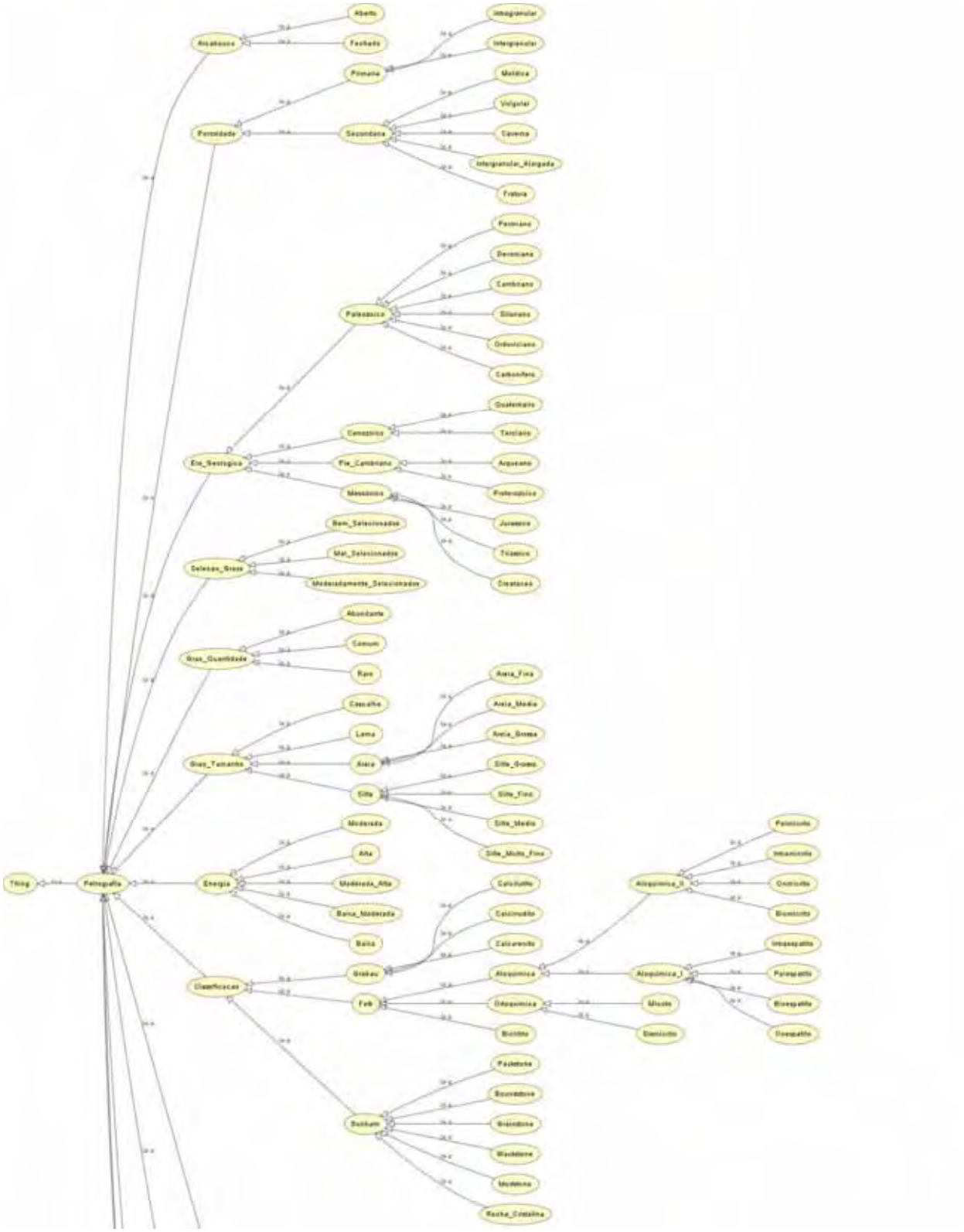
        <configurations>
            <configuration name="default">
                <goals>
                    <initialgoal ref="df_keep_registered">
                        <parameter ref="description">

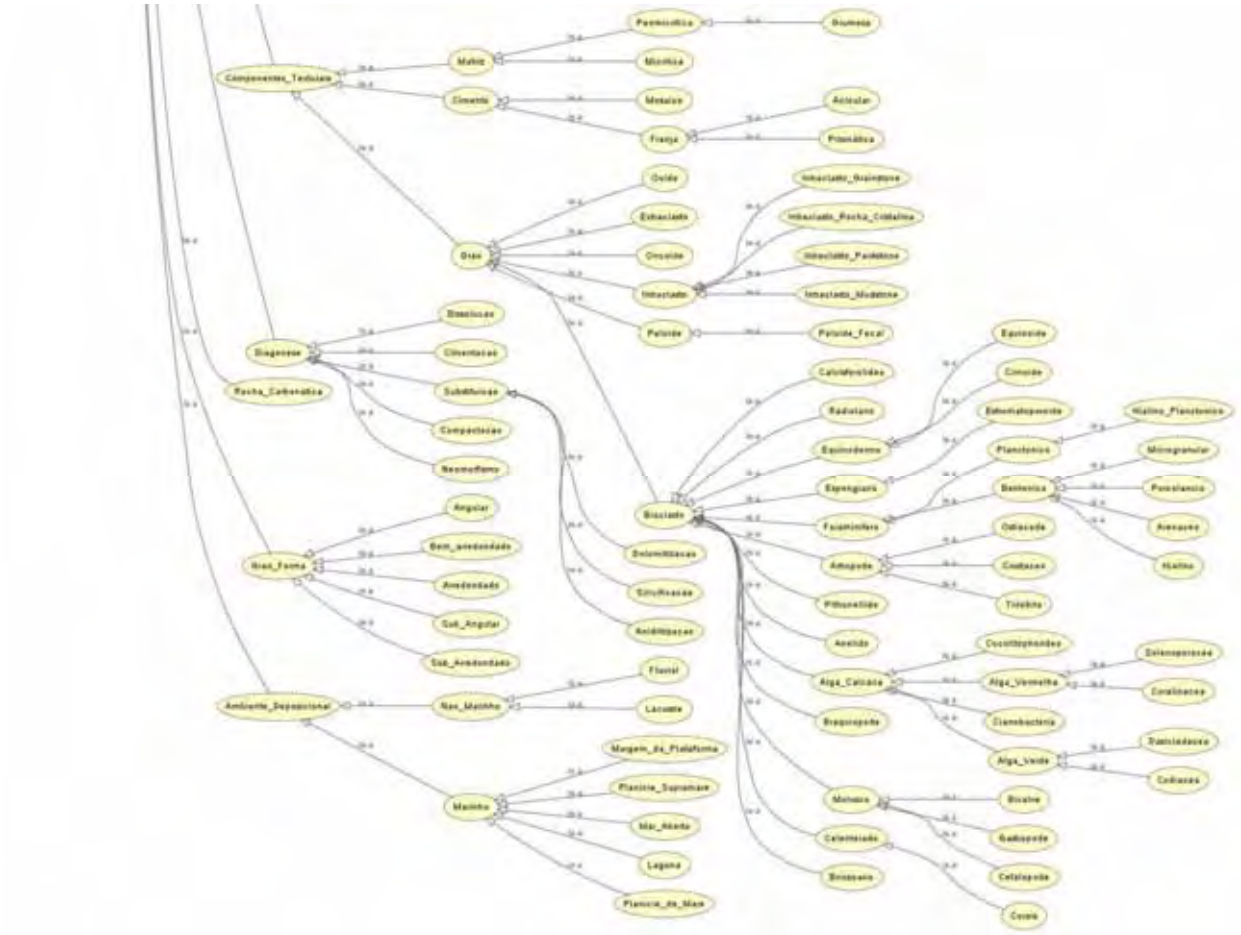
        <value>SFipa.createAgentDescription(null,SFipa.createServiceDescription("serv
ice_publicar","publicar", "br.unesp.gia"))</value>
                        </parameter>
                        <parameter ref="leasetime">
                            <value>20000</value>
                        </parameter>
                    </initialgoal>
                </goals>
            </configuration>
        </configurations>
</agent>
```

Apêndice E – Diagrama de Sequência – Atividades dos Agentes



Apêndice F – Ontologia de Petrografia




Componentes_Texturais
Matriz
Cimento
Poros
Fratura
Dissolução
Orientado
Fecho_Carbonático
Subtilhitos
Grao_Forma
Angular
Sub_Angular
Ambiente_Deposicional
Mar_Marinho
Lacustre
Fluvial
Margem_da_Plataforma
Planicie_Supramare
Mar_Aberto
Laguna
Planicie_de_Mare
Bioclasto
Peloide
Peloide_Fecal
Intraclasto
Intraclasto_Grainstone
Intraclasto_Rocha_Coralina
Intraclasto_Packstone
Intraclasto_Mudstone
Oolito
Oncoide
Equinodermo
Espongiario
Foraminifero
Ostracode
Briozoario
Alga_Calcaria
Alga_Vermelha
Alga_Verde
Molusco
Bivalve
Gastropode
Cefalopode
Corais