UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LAYSA MABEL DE OLIVEIRA FONTES

UMA ARQUITETURA MULTIAGENTE DE APOIO À APRENDIZAGEM BASEADA EM PROBLEMA

MOSSORÓ – RN

2013

LAYSA MABEL DE OLIVEIRA FONTES

UMA ARQUITETURA MULTIAGENTE DE APOIO À APRENDIZAGEM BASEADA EM PROBLEMA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação – associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Francisco Milton Mendes Neto – UFERSA.

MOSSORÓ – RN

2013

LAYSA MABEL DE OLIVEIRA FONTES

UMA ARQUITETURA MULTIAGENTE DE APOIO À APRENDIZAGEM BASEADA EM PROBLEMA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: ___ / ___ / _____.

BANCA EXAMINADORA

_____

Prof. Dr. Francisco Milton Mendes Neto – UFERSA
Presidente

_____

Prof. Dr. Rommel Wladimir de Lima – UERN
Membro Interno

_____

Prof. Dr. Ricardo Alexsandro de Medeiros Valentim – UFRN
Membro Externo

Dedico este trabalho especialmente à minha mãe, Lenica, pelo exemplo de vida, aos meus irmãos, Pedro e Larissa, e ao meu namorado, Éderson.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me guiar à conclusão de mais uma preciosa etapa de minha vida.

Aos meus irmãos Pedro e Larissa, pela torcida e incentivo, e em especial a minha querida mãe, Lenica, que, mesmo com poucos recursos, sempre nos incentivou a lutar por nossos ideais. Amo vocês!

Agradeço imensamente ao meu orientador, Milton Mendes, pela orientação, conselhos, ensinamentos, amizade, pelo apoio e confiança, pra você o meu respeito e profunda admiração. Muito obrigada por tudo!

Ao meu namorado, Éderson, por todo o amor, incentivo, amizade, compreensão e por estar sempre ao meu lado, apoiando as minhas decisões.

Aos meus amigos, Ferdinandy e Sairo, pelo companheirismo, ajuda, e, principalmente, pela amizade.

Aos colegas do Laboratório de Engenharia de Software (LES), pela excelente convivência. Agradeço a Alexandre, Fábio, Luiz Júnior e Luiz Cláudio, pela parceria nos trabalhos de pesquisa. E em especial a Danilo, pela valiosa contribuição na realização deste trabalho.

À Universidade Federal Rural do Semi-Árido (UFERSA) e Universidade do Estado do Rio Grande do Norte (UERN), em particular ao Programa de Pós-Graduação em Ciência da Computação (PPGCC), pela oportunidade de realização desse curso.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pela concessão da bolsa de estudo.

Aos professores do mestrado, por toda a dedicação e aprendizado.

À professora Flávia Coelho, pela ajuda concedida.

Aos professores Rommel Lima e Ricardo Valentim, pelo aceite de participar da minha banca e por todas as valiosas contribuições fornecidas para o aperfeiçoamento deste trabalho.

Enfim, agradeço a todos aqueles que, direta ou indiretamente, me ajudaram.

“Se a educação sozinha não transforma a sociedade, sem ela tampouco a sociedade muda”.

Paulo Freire

RESUMO

A Educação a Distância (EaD) é uma modalidade de ensino e aprendizagem que tem crescido e apresentado bons resultados. A evolução das tecnologias de redes de computadores, a melhoria na capacidade de processamento dos computadores pessoais e o avanço das tecnologias multimídia, dentre outros fatores, contribuíram para a criação deste cenário. A aprendizagem baseada em problema (*Problem-Based Learning* - PBL) é um método no qual os estudantes aprendem através da resolução de um problema que, em geral, não possui uma solução trivial e uma única solução correta. A PBL destaca o trabalho em equipe como um dos principais requisitos para o sucesso do processo de aprendizagem, ou seja, a colaboração é essencial. No entanto, a implantação de um método de ensino com base na PBL não é uma tarefa trivial. Em Ambientes Virtuais de Aprendizagem (AVAs), a complexidade de implantação deste método é ainda maior, pois o facilitador nem sempre pode detectar possíveis problemas na colaboração, nem possui todas as informações necessárias para aplicar as técnicas de aprendizagem deste método. Desta forma, este trabalho apresenta uma arquitetura multiagente de apoio à PBL, com o objetivo de detectar e corrigir problemas inerentes à implantação desta teoria de aprendizagem. Esse trabalho também apresenta como o AVA Moodle foi adaptado para suportar o processo de ensino e aprendizagem segundo a PBL e como o sistema multiagente pode apoiar os estudantes e os facilitadores durante esse processo.

Palavras-chave: Aprendizagem baseada em problema. Agente pedagógico animado. Sistema multiagente.

ABSTRACT

Distance Education (DE) is a teaching and learning approach that has grown and presented good results. The evolution of network technology, the improvement of personal computers processing capabilities and multimedia technology, among other factors, have contributed to this scenario. Problem-Based Learning (PBL) is a teaching method through which students are able to learn while solving problems that, in general, don't have trivial solutions or just one solution. PBL favors teamwork as one of the main requirements for a successful learning process. Therefore, collaboration is essential. However, applying a teaching approach like PBL is not a trivial task. The complexity of deploying such an approach on a Virtual Learning Environment (VLE) is even higher, as the facilitator may not always be able to detect possible collaboration issues, or acquire all the information they may need in order to properly apply the method's techniques. Thus, this work presents a multi-agent architecture that aims to support PBL by detecting and solving problems related to its deployment. The work also presents how popular VLE Moodle was adapted in order to support teaching and learning process based on PBL and how the multi-agent system can support both students and facilitators during the process.

Keywords: Problem-based learning. Animated pedagogical agent. Multi-agent system.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SIGLAS

ABED – Associação Brasileira de Educação a Distância;

ACC – *Agent Communication Channel*;

ACL – *Agent Communication Language*;

AG – Algoritmo Genético;

AID – *Agent Identifier*;

AML – *Agent Modeling Language*;

AMS – *Agent Management System*;

AP – *Agent Platform*;

AUML – *Agent UML*;

AVA – Ambiente Virtual de Aprendizagem;

BDI – *Belief Desire Intention*;

BL – *Blender License*;

CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior;

COHM – Centro de Oncologia e Hematologia de Mossoró;

CRC – *Class-Responsibility-Collaborator*;

DE – *Distance Education*;

DF – *Directory Facilitator*;

EaD – Educação a Distância;

ESOA – Engenharia de Software Orientada a Agentes;

FIPA – *Foundation for Intelligent Physical Agents*;

GPL – *General Public License*;

HINTS – *Health Information Network Teaching System*;

IA – Inteligência Artificial;

IEEE – *Institute of Electrical and Electronics Engineers*;

JADE – *Java Agent Development Framework*;

LES – Laboratório de Engenharia de Software;

MaSE – *Multiagent Systems Engineering*;

MEC – Ministério da Educação;

Moodle – *Modular Object-Oriented Dynamic Learning Environment*;

MSC – *Message Sequence Charts*;

MTP – *Message Transport Protocol*;

MTS – *Message Transport Service*;

OA – Objeto de Aprendizagem;

OWL – *Web Ontology Language*;

PASSI – *Process for Agent Societies Specification and Implementation*;

PBL – *Problem-Based Learning*;

POA – Programação Orientada a Agentes;

PPGCC – Programa de Pós-Graduação em Ciência da Computação;

RUP – *Rational Unified Process*;

SCORM – *Sharable Content Object Reference Model*;

SDL – *Specification and Description Language*;

SMA – Sistema Multiagente;

UERN – Universidade do Estado do Rio Grande do Norte;

UFERSA – Universidade Federal Rural do Semi-Árido;

UML – *Unified Modeling Language*;

VLE – *Virtual Learning Environment*;

ZDP – Zona de Desenvolvimento Proximal.

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A Educação a Distância (EaD) é uma modalidade de ensino e aprendizagem que tem crescido e apresentado bons resultados (COUTINHO; BOTTENTUIT JUNIOR, 2007; PONTES, 2010). De acordo com pesquisa realizada pela Associação Brasileira de Educação a Distância (ABED) e pelo Ministério da Educação (MEC), a demanda por cursos de especialização a distância aumentou 60% de 2008 a 2010 (MAIA, 2011). A evolução das tecnologias de redes de computadores, a melhoria na capacidade de processamento dos computadores pessoais e o avanço das tecnologias multimídia, dentre outros fatores, contribuíram para a criação deste cenário (PONTES, 2010). Entretanto, apesar de consistir em uma modalidade de ensino eficiente, a EaD ainda apresenta alguns desafios, dentre os quais se destaca a necessidade de um suporte informatizado adequado às características de cada indivíduo (SILVA, 2012). Além disso, as ferramentas disponíveis nesses sistemas podem não oferecer um suporte suficiente para aquisição de conhecimento no processo de ensino e aprendizagem (JAQUES *et al.*, 2002).

Com isso, a utilização de técnicas de Inteligência Artificial (IA), no projeto e desenvolvimento de ambientes de ensino e aprendizagem computadorizados, tem se constituído em objeto de maior investigação por parte dos pesquisadores da área de informática aplicada à educação, devido às suas potencialidades (SANTOS *et al.*, 2001).

O conceito de agentes de software tem se mantido como um importante tema de pesquisa no âmbito educacional. Esta abordagem tem se mostrado bastante promissora como auxílio em ambientes colaborativos de aprendizagem, devido a sua capacidade de dinamizar o processo. Eles podem ser usados, por exemplo, para auxiliar no cumprimento de uma dada teoria de aprendizagem em um Ambiente Virtual de Aprendizagem (AVA) (PONTES, 2010). Sendo assim, a combinação de agentes e AVAs consiste em uma abordagem promissora para o aprendizado eficaz auxiliado por computador (SOLIMAN; GUETL, 2010).

A aprendizagem baseada em problema (*Problem-Based Learning - PBL*) é um método no qual os estudantes aprendem através da resolução de um problema que, em geral, não possui uma solução trivial e uma única solução correta. A aprendizagem é centrada no estudante e o conhecimento é adquirido de forma autodirigida. Os estudantes trabalham em

pequenos grupos colaborativos para identificar o que eles necessitam aprender para resolução do problema. O professor atua como facilitador[1] do processo de aprendizagem ao invés de apenas transmitir conhecimentos (HMELO-SILVER, 2004).

1.2 PROBLEMÁTICA

A implantação de um método de ensino com base na PBL não é uma tarefa trivial. Em AVAs, a complexidade de implantação deste método é ainda maior, pois o facilitador nem sempre pode detectar possíveis problemas na colaboração, nem possui todas as informações necessárias para aplicar as técnicas de aprendizagem deste método, como, por exemplo, saber quando os estudantes estão saindo do foco da discussão e tomar medidas de correção (HMELO-SILVER; BARROWS, 2006; PONTES, 2010).

A PBL enfatiza o trabalho em equipe como a chave para o sucesso do processo de aprendizagem. Em outras palavras, a colaboração é essencial (SAVERY, 2006). No entanto, a PBL possui algumas dificuldades para ser colocada em prática. Isto se deve ao fato de que esta teoria de aprendizagem possui várias características únicas, além de alguns fatores que devem existir para o sucesso de sua aplicação. Dentre as principais dificuldades de implementação da PBL, podemos citar (HMELO-SILVER, 2004; HMELO-SILVER; BARROWS, 2006, SAVERY, 2006):

1. Não é trivial cumprir com o ciclo de desenvolvimento da PBL;
2. A construção do cenário do problema não é fácil, pois os problemas devem ser complexos e mal estruturados;
3. O processo de aprendizado é complexo, pois deve ser multidisciplinar;
4. É difícil para o facilitador garantir uma colaboração efetiva;
5. É difícil para o facilitador monitorar o processo de aprendizagem, pois este é autodirigido pelo estudante, de forma que o facilitador não pode oferecer as respostas ao estudante, apenas guiá-lo neste processo;
6. O método de avaliação é complexo, pois o facilitador deve avaliar o progresso do estudante seguindo os objetivos específicos da PBL;

[1] Responsável, no sistema de EaD, pelo acompanhamento da evolução da aprendizagem do estudante. É semelhante ao professor no ensino presencial, mas com outro papel, pois enquanto o professor é responsável pela formação do estudante, o facilitador tem a função de facilitar sua aprendizagem (MENDES NETO, 2000).

7. O facilitador deve possuir competências específicas para guiar o processo de aprendizagem, como, por exemplo, conhecimentos adequados no que diz respeito ao tema em estudo, bem como a habilidade de expressar-se em uma linguagem compreendida pelos estudantes;
8. É difícil para o facilitador ter conhecimento profundo das técnicas e estratégias usadas na PBL.

Não foi encontrado na literatura um trabalho que propusesse uma arquitetura multiagente para apoiar AVAs que implementasse a teoria de aprendizagem PBL. Também não foram encontrados trabalhos que propusessem agentes pedagógicos animados para auxiliar os estudantes no processo de ensino e aprendizagem utilizando a PBL.

1.3 OBJETIVO GERAL

Tendo em vista a problemática apresentada, o presente trabalho tem como objetivo geral desenvolver uma arquitetura multiagente para atender as principais metas relacionadas ao auxílio no cumprimento da PBL, auxiliando o facilitador na detecção de possíveis problemas na colaboração e promovendo a aprendizagem dos estudantes durante o processo de aplicação da PBL.

1.3.1 Objetivos Específicos

- Pesquisar e selecionar possíveis metas relacionadas ao auxílio no cumprimento da PBL, que sejam viáveis computacionalmente e que sejam adaptáveis à tecnologia de agentes;
- Projetar e modelar a arquitetura multiagente, utilizando a metodologia de modelagem de Sistemas Multiagente (SMAs) MAS-CommonKADS+;
- Implementar a proposta de solução e integrar ao AVA Moodle.

1.4 METODOLOGIA

Para atingir o objetivo geral deste trabalho, as seguintes atividades são necessárias:

1. Realizar uma pesquisa bibliográfica sobre agentes de software e agentes pedagógicos, e seu uso no contexto educacional, pesquisando sistemas que utilizaram com êxito a tecnologia de agentes;
2. Realizar uma pesquisa bibliográfica sobre as principais teorias de aprendizagem que apoiam a aprendizagem colaborativa, principalmente a PBL, que servirá de base para implementação do SMA proposto;
3. Realizar uma pesquisa bibliográfica sobre as metodologias de modelagem de SMAs;
4. Realizar uma pesquisa bibliográfica sobre o funcionamento dos Algoritmos Genéticos (AGs);
5. Construir duas ontologias para modelar os domínios de conhecimento do problema e da PBL;
6. Realizar a modelagem do agente pedagógico animado proposto, utilizando a ferramenta Blender (BLENDER, 2013);
7. Modelar o SMA proposto, utilizando a metodologia de modelagem selecionada, ou seja, a MAS-CommonKADS+;
8. Implementar a proposta de solução e integrar ao AVA Moodle.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação está organizada da seguinte forma: o Capítulo 2 traz conceitos de agentes e SMAs. Ao longo deste capítulo também são mostradas algumas experiências de implantação de sistemas de suporte à aprendizagem que utilizam agentes. O Capítulo 3 descreve aspectos relacionados à aprendizagem colaborativa e às teorias de aprendizagem que apoiam esta abordagem. O Capítulo 4 apresenta conceitos sobre as metodologias utilizadas para modelagem de SMAs. O Capítulo 5 apresenta a fundamentação teórica necessária para que seja possível entender o funcionamento do AG proposto neste trabalho. O Capítulo 6 descreve a arquitetura multiagente de apoio à PBL desenvolvida neste trabalho, detalhando os

agentes e como estes foram modelados e implementados. Este capítulo também exemplifica alguns cenários de uso do Moodle (*Modular Object-Oriented Dynamic Learning Environment*) (MOODLE, 2011). O Capítulo 7 apresenta detalhes do desenvolvimento das ontologias propostas neste trabalho. E, por fim, o Capítulo 8 apresenta as considerações finais e os trabalhos futuros.

2 AGENTES E SISTEMAS MULTIAGENTE

Este capítulo descreve aspectos relacionados a agentes e SMAs. A Seção 2.1 traz uma discussão sobre definições de agentes de software apresentadas por diversos autores. A Seção 2.2 traz um resumo dos principais tipos de agentes existentes. A Seção 2.3 mostra conceitos relacionados aos SMAs. Já a Seção 2.4 apresenta conceitos relacionados aos agentes pedagógicos.

2.1 DEFINIÇÕES DE AGENTES

Segundo Artero (2009) e Pontes (2010), apesar das diversas definições de agentes que podem ser encontradas na literatura, ainda não existe um consenso sobre o assunto. Entretanto, é possível construir um conceito a partir de definições dadas por pesquisadores da área (SILVA, 2012).

De acordo com Russel e Norvig (2003), um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de atuar sobre esse ambiente através de atuadores.

Já para Wooldrigde (2002), um agente é um sistema computacional que está situado em algum ambiente e que é capaz de efetuar ações autônomas neste ambiente com o intuito de cumprir os objetivos para os quais ele foi projetado.

Henderson-Sellers e Giorgini (2005), por sua vez, defendem a ideia de que agentes são entidades de software ou não que são caracterizadas por serem: autônomas, proativas e direcionadas a objetivos.

De acordo com Artero (2009), agentes inteligentes são programas que executam um conjunto de operações no lugar de um usuário, utilizando uma representação do conhecimento que contém os objetivos do usuário. De acordo com o autor, existe outra definição que afirma que agentes são programas que realizam diálogos para negociar e coordenar transferências de informação.

Segundo Russel e Norvig (2003), outra característica a ser levada em consideração é a racionalidade. Quatro fatores influenciam a racionalidade (RUSSEL e NORVIG, 2003; SILVA, 2012):

- Medida de desempenho que define o critério de sucesso do agente;
- Conhecimento prévio que o agente possui;
- Ações que o agente pode realizar;
- Sequência de percepções captadas pelo agente até o momento.

Diante desses elementos, é possível conceituar um agente racional como sendo aquele que, para cada sequência de percepções possível, seleciona uma ação que venha a maximizar sua medida de desempenho, dada a evidência fornecida pela sequência de percepções e por qualquer conhecimento interno do agente (RUSSEL e NORVIG, 2003). Levando em consideração essa definição, é possível perceber que nem sempre um agente racional tomará a melhor decisão possível, mas sim aquela que maximize sua medida de desempenho (SILVA, 2012).

Dessa forma, neste trabalho, um agente será considerado como uma entidade de software que percebe, de forma autônoma, o seu ambiente e atua sobre o mesmo. Além disso, essas entidades poderão se comunicar entre si, trocando informações para alcançar um objetivo comum.

2.2 TIPOS DE AGENTES

Resumidamente, podem ser citados cinco tipos básicos de agentes (RUSSEL e NORVIG, 2003; ARTERO, 2009; SILVA, 2012):

- Agente tabela: essa é a estrutura mais simples de um agente, na qual todas as percepções e ações possíveis estão relacionadas em uma tabela. O grande problema dessa abordagem reside na necessidade de incluir todas as percepções e ações possíveis.
- Agente reativo simples: seleciona ações a serem executadas com base exclusivamente na percepção atual, não levando em consideração o histórico de percepções. Ele possui um conjunto de regras do tipo condição-ação que substitui de forma satisfatória a estrutura do agente tabela, o qual deve conter em sua tabela

todas as possíveis percepções e ações. Como não possuem uma memória, esses agentes são incapazes de planejar ações futuras.

- Agente reativo baseado em modelo: também conhecido como agente reativo com estado interno, esse tipo de agente controla o estado atual do mundo usando um modelo interno do ambiente. Esse modelo depende do seu histórico de percepções e, dessa forma, reflete, no mínimo, alguns aspectos não observados no estado atual. Este agente combina as informações da percepção atual com as provenientes do modelo para gerar a descrição atualizada do estado atual. De posse dessas informações, ele escolhe uma ação da mesma forma que o agente reativo simples.
- Agente baseado em objetivos: também chamado de agente cognitivo, ele pondera suas ações levando em consideração a descrição do estado atual e os objetivos a serem alcançados. Esse tipo de agente combina essas informações com aquelas sobre os resultados das ações possíveis, podendo escolher, dentre estas, a que lhe permita atingir mais rapidamente seus objetivos.
- Agente baseado em utilidade: escolhe suas ações tentando sempre maximizar uma função de utilidade. Essa função mapeia um estado (ou uma sequência de estados) em um número real, que descreve o grau de "felicidade" do agente caso aquele estado seja alcançado.

Além disso, esses tipos básicos de agentes podem ser convertidos em agentes com aprendizado, sendo necessário, para tal, o uso de algoritmos de aprendizagem, melhorando, consequentemente, o seu desempenho (RUSSEL e NORVIG, 2003). Os agentes com aprendizado possuem uma grande adaptabilidade às mudanças do ambiente, pois vão aprendendo com ele (ARTERO, 2009; SILVA, 2012).

2.3 SISTEMAS MULTIAGENTE

Uma vez tendo o conceito de um agente de forma isolada, é possível definir o conceito de Sistema Multiagente (SMA). De acordo com Henderson-Sellers e Giorgini (2005), um SMA é um sistema composto de agentes cooperativos ou competitivos que interagem entre si para atingir um objetivo individual ou comum.

É comum utilizar-se de SMAs para transformar grandes problemas em problemas menores e possibilitar, dessa forma, que cada agente possa utilizar sua habilidade para tratar

esses pequenos problemas, modularizando o sistema e tornando mais fácil, por consequência, adicionar novas funcionalidades através da inclusão de novos agentes (ARTERO, 2009; SILVA, 2012). Artero (2009) cita ainda outras vantagens na utilização de SMAs:

> É bastante comum a necessidade de usar as habilidades de diferentes agentes para resolver problemas. De fato, tratar problemas complexos como um aglomerado de problemas menores tem sido uma estratégia bastante usada na prática. Além disso, sistemas multiagente podem apresentar maior rapidez na resolução dos problemas, por causa do paralelismo que pode ser obtido. Também se observa uma maior flexibilidade do sistema, combinando-se, de diversas maneiras, as diferentes habilidades dos agentes para resolver os problemas. [...] Por fim, a modularidade obtida com os sistemas multiagente é outra característica muito relevante, porque quando o sistema não consegue resolver uma determinada tarefa, geralmente, é mais simples projetar e incluir um novo agente ao sistema do que substituir o sistema inteiro.

Uma arquitetura bastante simples e que é muito utilizada no controle dos agentes é a arquitetura “Quadro Negro”. Nessa arquitetura não é necessária uma comunicação direta entre os agentes, pois todas as comunicações são feitas por meio de uma estrutura de dados central (quadro negro) que é compartilhada por todos os agentes, sendo também responsável por controlar o acesso dos agentes. Quando se utiliza essa arquitetura, agentes podem escrever informações no quadro, assim como podem ler informações deixadas por outros agentes (ARTERO, 2009; SILVA, 2012).

Pode-se adotar também uma estratégia oposta ao quadro negro, na qual os agentes se comuniquem diretamente. Porém, esta abordagem acaba retirando um pouco da flexibilidade da arquitetura multiagente, pois obriga os agentes a possuírem uma identificação precisa, como um nome único no sistema, além da adoção de algum protocolo de comunicação entre eles (ARTERO, 2009; SILVA, 2012).

2.3.1 Organização dos Sistemas Multiagente

Os SMAs podem ser organizados de diversas maneiras, sendo três os tipos mais comuns (ARTERO, 2009; SILVA, 2012):

- Hierárquica: existe um agente superior que controla e toma as decisões, comunicando sua decisão para os demais agentes, que pertencem a um nível inferior na hierarquia;
- Comunidade de especialistas: todos os agentes estão em um mesmo nível, tendo cada agente uma especialidade em certo domínio. Nesse tipo de organização, a interação entre os agentes ocorre de acordo com regras previamente estabelecidas;
- Comunidade científica: os problemas são solucionados localmente e, em seguida, essas soluções são testadas e refinadas por agentes solucionadores de problemas.

A organização do SMA proposto neste trabalho é do tipo comunidade de especialistas.

2.3.2 Aplicações dos Sistemas Multiagente

O conceito de gentes de software tem sido muito utilizado no âmbito educacional. Esta abordagem tem se mostrado promissora como auxílio em ambientes colaborativos de aprendizagem, tornando estes ambientes mais proativos e autônomos. Agentes de software podem ser usados, por exemplo, para auxiliar na implementação de uma dada teoria de aprendizagem em um ambiente colaborativo. Além disso, agentes de software podem realizar diversas tarefas em ambientes de aprendizagem, tais como monitorar as atividades do estudante, capturar dinamicamente informações do estudante, recomendar conteúdos de interesse deste, dentre outras atividades (SILVA, 2012).

Vizcaíno (2005) descreve uma arquitetura, baseada em um estudante simulado, que foi projetada para detectar e evitar três situações que podem dificultar o processo de aprendizagem em AVAs: conversações que fogem ao conteúdo ministrado, estudantes com comportamento passivo e problemas relacionados ao aprendizado do estudante.

Azevedo e Scalabrin (2005) descrevem um ambiente de apoio à aprendizagem colaborativa, chamado COLE, que enfoca aspectos sociais na interação entre os participantes do processo de aprendizagem. Este ambiente foi construído para apoiar a aprendizagem baseada em projeto (*Project Based Learning*) (SONG *et al.*, 2012), uma variação da PBL. O processo de aprendizagem no ambiente é efetuado com auxílio de portfólios.

Moisil *et al.* (2006) descrevem um modelo para AVAs que emprega agentes inteligentes para implementar a teoria sócio-cultural de Vygotsky (VYGOTSKI, 1998), enfocando o aspecto social de interação. O modelo proposto possui diversos agentes, dentre

os quais destacam-se: (i) o agente social, que tem como principal objetivo a construção de modelos para os grupos de estudantes, além de identificar grupos de estudantes que podem cooperar em boas condições; (ii) o agente tutor, que avalia os objetivos educacionais do estudante e recomenda algum tipo de atividade; e (iii) os agentes de assistência ao estudante, que monitoram suas atividades e se comunicam com os outros agentes.

Lima *et al.* (2005) propõem uma abordagem baseada em algoritmos genéticos para a formação dos grupos, na qual são aplicados pelo professor fatores de aceitação de um grupo. Esta abordagem leva em consideração o perfil dos estudantes e a coesão do grupo, utilizando técnicas sócio-métricas. Esta abordagem é usada no ambiente NetClass (LABIDI; SOUZA; NASCIMENTO, 2003) de ensino e aprendizagem cooperativa.

Faria, Virela e Coello (2005) descrevem o *Learn in Group*, um sistema baseado na Web para suporte ao aprendizado colaborativo de programação. O sistema conta com dois agentes mediadores, um para a constituição de grupos de trabalho colaborativos e outro para acompanhar e estimular a participação dos membros dos grupos.

Silveira e Barone (2006) apresentam uma aplicação de técnicas de IA, mais especificamente SMAs, para a formação de grupos colaborativos em um ambiente multiagente interativo de aprendizagem na Web. Os autores apresentam a definição e a implementação de uma arquitetura de agentes, modelados com algoritmos genéticos, bem como sua integração com o ambiente TelEduc (FERREIRA; OTSUKA; ROCHA, 2003).

Felix e Tedesco (2008) apresentam a ferramenta *Smart Chat Group*, que usa uma sociedade de agentes inteligentes para fazer acompanhamento, sugestão e formação automática de pequenos grupos de aprendizagem com base em informações de contexto dos aprendizes.

Bremgartner e Netto (2011) apresentam um SMA para adaptação de AVAs a fim de auxiliar estudantes que apresentem dúvidas ou erros ao executarem as atividades propostas pelo professor. O auxílio é baseado na recomendação personalizada de estudantes com perfis adequados, baseados em suas habilidades e competências específicas, que irão ajudar os estudantes com dúvidas ou erros em atividades. Os autores também relatam que os resultados parciais aplicados em uma disciplina de Cálculo Numérico mostram a viabilidade da proposta.

Como diferencial do nosso trabalho, podemos destacar que, diferentemente dos outros trabalhos discutidos nessa seção, o presente trabalho é voltado especificamente para aplicação da PBL, ou seja, apresenta uma arquitetura multiagente para auxiliar na aplicação correta da PBL, uma teoria de aprendizagem comprovadamente eficaz (STROBEL; VAN BARNEVELD, 2009). Vale ressaltar também que a solução apresentada neste trabalho está

integrada ao Moodle, uma ferramenta difundida tanto na comunidade acadêmica como na indústria (KUMAR; GANKOTIYA; DUTTA, 2011).

2.4 AGENTES PEDAGÓGICOS

Ao apoiar uma atividade educacional, os agentes são ditos pedagógicos. Os agentes pedagógicos são entidades cujo propósito fundamental é a comunicação com o estudante. Os agentes pedagógicos oferecem instrução personalizada, aumentam a motivação dos estudantes e agem pedagogicamente, por conta própria ou com o auxílio dos professores. Por outro lado, AVAs agregam valor ao processo educativo, dando novas possibilidades de educação. Sendo assim, a combinação de agentes pedagógicos e AVAs gera uma abordagem promissora para o aprendizado eficaz auxiliado por computador (SOLIMAN; GUETL, 2010).

Agentes pedagógicos são agentes inseridos em ambientes interativos de aprendizagem, sendo suas principais funções: acompanhar o trabalho dos estudantes, monitorar o desenvolvimento das tarefas, identificar dificuldades, oferecer dicas e auxiliar na resolução de problemas (REATEGUI; MORAES, 2006). Além disso, os agentes pedagógicos devem motivar o estudante, despertando o interesse dele em interagir cada vez mais com o ambiente de aprendizagem (SILVA; BERNARDI, 2009).

De acordo com Giraffa (1999), os agentes pedagógicos podem ser classificados como orientados pela utilidade e dirigidos por objetivos. Agentes pedagógicos orientados pela utilidade são aqueles que auxiliam o estudante na execução de diversas tarefas, como, por exemplo, busca de arquivos ou programas ou seleção de grupos de estudantes para troca de informações em uma sessão de trabalhos de um ambiente de cooperação. Os agentes pedagógicos dirigidos por objetivos podem desenvolver atividades em colaboração e/ou em cooperação com estudantes, dependendo de seus próprios objetivos. Esses agentes podem ser do tipo tutores, mentores ou assistentes, ou seja, são responsáveis pelo planejamento, decisão e execução das ações pedagógicas no ambiente (GIRAFFA, 1999).

2.4.1 Agentes Pedagógicos Animados

Na literatura, existem vários trabalhos que propõem o desenvolvimento de agentes que simulam o comportamento de seres vivos, denominados agentes pedagógicos animados (BERCHT, 2001). Agentes pedagógicos são considerados animados quando são implementados com recursos de animação (SILVA; BERNARDI, 2009).

Os agentes pedagógicos animados são representados por personagens animados que interagem com os estudantes. Estes agentes usam recursos de multimídia para fornecer ao usuário um personagem com características semelhantes àquelas de seres humanos. Estas características, tais como expressões faciais e entendimento das emoções humanas, juntamente com uma boa interface de diálogo com o usuário, tornam esses agentes mais atraentes ao estudante (JAQUES; VICARI, 2005). Essa técnica apresenta vantagens quando comparada com ambientes de ensino baseados na Web convencionais, uma vez que possibilita interações mais naturais e mais próximas entre estudante e sistema (SANTOS *et al.*, 2001).

2.4.2 Aplicações dos Agentes Pedagógicos

Vários esforços têm sido realizados para utilizar agentes pedagógicos para apoiar os estudantes de forma personalizada, buscando melhorar a interatividade e motivação e tentando compensar a falta dos aspectos humanos no ambiente de aprendizagem utilizado (SOLIMAN e GUETL, 2010).

Arroyo, Woolf e Cooper (2011) apresentam os resultados de uma avaliação, realizada com estudantes de ensino médio, do impacto da utilização de agentes pedagógicos integrados a um sistema de tutoria inteligente de matemática. Os resultados apresentados pelos autores indicaram que agentes pedagógicos melhoraram os aspectos afetivos dos estudantes em geral, mas tendo um impacto maior com os estudantes do sexo feminino.

Frozza *et al.* (2011) apresentam o desenvolvimento e a atuação de dois agentes pedagógicos animados (agente tutor e agente companheiro), que expressam emoções e estão integrados em um AVA, a fim de interagir com estudantes. O agente tutor Dóris tem o papel semelhante ao de um professor, identificando as características de aprendizagem do estudante.

O agente companheiro Dimi atua juntamente com o estudante na realização de atividades propostas pelo ambiente virtual, fornecendo dicas e desafios.

Silva e Bernardi (2009) apresentam um agente pedagógico animado, chamado Cal, que foi desenvolvido com o objetivo de interagir afetivamente com o estudante, de modo a facilitar a relação ensino e aprendizagem, além de auxiliar o estudante na utilização do Objeto de Aprendizagem (OA) no qual o agente está inserido.

Cheng *et al.* (2009) propõem uma arquitetura baseada em um agente pedagógico. A arquitetura foi implementada juntamente com um sistema de aprendizagem baseado na Web, chamado HINTS (*Health Information Network Teaching System*). Esta arquitetura foi desenvolvida com o intuito de facilitar a aprendizagem dos estudantes e, assim, tornar o HINTS mais eficaz e eficiente no processo de aprendizagem. Os autores também relatam os resultados preliminares da avaliação do desempenho do sistema, concluindo que o agente pedagógico, de fato, ajuda os estudantes no processo de aprendizagem.

O presente trabalho também apresenta um agente pedagógico animado com o intuito de apoiar os estudantes na resolução de problemas, através da PBL. Esse agente consiste em um modelo humanoide tridimensional animado responsável por acompanhar os estudantes durante o processo de aplicação da PBL, além de manter os estudantes sempre motivados, principalmente quando algum problema de colaboração for detectado. Para obter sucesso nesse caso, o agente pedagógico animado expressa emoções similares às dos seres humanos.

3 APRENDIZAGEM COLABORATIVA E AS TEORIAS DE APRENDIZAGEM

Este capítulo descreve aspectos relacionados à aprendizagem colaborativa e às teorias de aprendizagem que apoiam esta abordagem. A Seção 3.1 apresenta conceitos relacionados à aprendizagem colaborativa. A Seção 3.2, por sua vez, apresenta diversas teorias de aprendizagem, dando maior ênfase a PBL.

3.1 APRENDIZAGEM COLABORATIVA

A aprendizagem colaborativa é definida como aprendizado por colaboração (DILLENBOURG, 1999). Está explícito no termo que este tipo de aprendizagem surge da colaboração entre os estudantes para resolução de problemas comuns. Esta perspectiva nos mostra a importância da formação de grupos colaborativos por estudantes que possuem um objetivo comum.

Neste modelo de aprendizagem, há uma contribuição de cada membro do grupo, existindo, portanto, uma manutenção da contribuição individual. O indivíduo, posteriormente, poderá ser avaliado de forma individual. A aprendizagem se dá em um ambiente onde todos os membros têm o mesmo grau, ou seja, não há hierarquias. Cada membro participa da resolução do problema motivado por seu sucesso individual, pelo menos inicialmente, e o professor surge como um estimulador e orientador deste processo de aprendizagem (GONZÁLEZ, 2005).

O conhecimento é adquirido por meio da interação entre os participantes. O processo de ensino deixa de seguir um modelo tradicional, onde há apenas uma transmissão de informações unidirecional (do professor para o estudante), e desenvolve a discussão e o pensamento crítico no estudante. Assim, o professor tem um papel de facilitador deste processo, deixando de ser um mero transmissor de informações para os estudantes. A aprendizagem é autodirigida, ou seja, o estudante é responsável pelo seu próprio aprendizado. Além disso, o estudante possui a responsabilidade perante o grupo de cumprir as tarefas para a solução do problema proposto (PONTES, 2010).

3.2 TEORIAS DE APRENDIZAGEM

Muitas teorias apoiam a aprendizagem colaborativa. Esta seção tem por objetivo apresentar as principais teorias de aprendizagem que oferecem apoio a esta modalidade de aprendizagem.

3.2.1 Construtivismo Piagetiano

Um dos principais estudiosos desta corrente, Jean Piaget, mostra em seus estudos como o conhecimento é adquirido pelo indivíduo. Ele tenta entender como o indivíduo passa de uma fase de menor conhecimento para outra de maior, sendo sua principal obra a *epistemologia genética* (PIAGET, 1973).

Segundo esta teoria, o conhecimento não é algo inerente ao indivíduo, mas parte da relação entre o indivíduo e o meio em que ele está inserido. Neste caso, o indivíduo é um ser ativo que obtém conhecimento através da relação entre este e os objetos do meio em que ele está inserido (PIAGET, 1973).

Segundo Piaget (1973), existem estruturas cognitivas no ser humano que são capazes de armazenar novas informações extraídas do meio, as quais ele chama de esquemas. Assim, no processo de construção do conhecimento, dois conceitos são definidos: assimilação e acomodação.

Assimilação se refere ao processo dinâmico e contínuo de aquisição do conhecimento. Já a acomodação se refere ao fato de como os esquemas irão se reorganizar para acomodar este novo conhecimento que foi assimilado. Em outras palavras, quando uma criança ou qualquer pessoa tenta adquirir conhecimento ou passa por uma situação (experiência) nova, ela primeiramente tenta assimilar essa experiência em seus esquemas existentes (CASTAÑON, 2005; CASTAÑON, 2007). A tendência é que o esquema se modifique de modo a acomodar-se a esta nova informação. Ainda em seus estudos, Piaget trata das fases de evolução desses esquemas, do nascimento até a idade adulta. Ele define quatro estágios nesta evolução: sensório-motor, que constitui a fase exploratória da criança através de movimentos motores para alcançar um objeto, por exemplo, respondendo a estímulos do ambiente; pré-operatório, onde a criança começa a criar símbolos mentais e realizar associações, podendo

realizar associações mesmo na ausência do objeto concreto; operatório concreto, onde a criança começa a pensar de maneira lógica e começa a entender melhor o meio e suas ações sobre ele; formal, que possui estruturas que possibilitam raciocínio complexo e é capaz de discutir teorias e problemas reais (PONTES, 2010).

3.2.2 Teoria Sócio-Cultural de Vygotsky

A teoria sócio-cultural de Vygotsky (VYGOTSKI, 1998; RIBAS; MOURA, 2006) enfatiza que a inteligência do indivíduo se origina da sociedade e da cultura, ou seja, de uma interação entre o indivíduo e o ambiente no qual ele está inserido. Esta teoria de aprendizagem enfatiza que o conhecimento humano advém do meio, através de relações interpessoais, e não de aspectos internos (intrapessoais) do indivíduo. Esta perspectiva remete à importância da sociabilidade e interação das pessoas no desenvolvimento mental dos indivíduos. Este é o elemento fundamental da teoria de Vygotsky sobre interação social, ou seja, o fato de que esta possui um papel formador e construtor. Isto significa que muitas funções mentais do indivíduo não são possíveis de serem construídas sem as interações sociais.

Vygotski (1998) apresenta um dos principais conceitos desta teoria de aprendizagem, a Zona de Desenvolvimento Proximal (ZDP). Segundo este conceito, os indivíduos possuem uma capacidade de resolução de problemas elementar e outra capacidade que só pode ser alcançada com a ajuda de um tutor (facilitador). Este limite corresponde à ZDP. Assim, cada um é capaz de resolver alguns problemas sem ajuda e isto representa seu conhecimento em uma dada idade. Contudo, o indivíduo pode atingir um limite (ZDP) caso obtenha ajuda do facilitador no processo de aprendizagem, onde este limite seria sua idade mental real. Vygotsky analisou, em sua obra, exemplos de crianças em seu processo de aprendizagem inicial.

É importante salientar que, neste aspecto, o estudante possui uma grande importância no processo de aprendizagem. O tutor exerce um papel de facilitador da aprendizagem, podendo guiar o estudante na resolução de problemas mais complexos, os quais o estudante por si só não poderia solucionar. Esta teoria pode ser facilmente aplicada em ambientes de aprendizagem colaborativa, pois além das interações com os demais colegas no processo de

aprendizagem, o facilitador desempenha o papel do tutor neste processo, dinamizando o processo de aprendizagem (PONTES, 2010).

3.2.3 Cognição Distribuída

A Cognição Distribuída foi desenvolvida por Ed Hutchins (HUTCHINS, 1995), sendo considerada uma mudança radical na forma de pensar sobre o fenômeno da cognição. A visão tradicional enfoca que o fenômeno de cognição é intrínseco ao indivíduo, enquanto Hutchins afirma que este processo se dá de forma distribuída no ambiente, englobando fatores externos, sobretudo artefatos do meio (ROGERS, 2006).

Assim, a cognição distribuída está relacionada com a interação do indivíduo com os artefatos do meio e não só com uma perspectiva interna. Hutchins (1995) mostra alguns exemplos do uso desta teoria em problemas reais, como, por exemplo, a análise do processo de navegação de um navio e a análise de um piloto de avião em sua cabine. Em outro exemplo, Rogers (1997) mostra um estudo de como controladores aéreos interagem com um sistema de radar ao controlar o tráfico. Estes exemplos mostram que a cognição está relacionada com o indivíduo e sua relação com o meio (PONTES, 2010).

3.2.4 Teoria da Flexibilidade Cognitiva

Esta teoria de aprendizagem foi proposta por Spiro na década de 80 (SPIRO *et al.*, 2003). Esta teoria afirma que as pessoas adquirem conhecimento em ambientes não estruturados (complexos) pela associação entre unidades conhecidas. Este conceito pode ser melhor ilustrado (implementado) em ambientes que utilizam documentos hipermídia, onde os documentos estão inter-relacionados, ou seja, os conceitos similares estão dispostos através de referências cruzadas.

A teoria da flexibilidade cognitiva afirma que o aprendizado é dependente do contexto do aprendiz. A teoria está embasada em teorias construtivistas e também enfatiza a construção do conhecimento pelo indivíduo. Assim esta teoria é bem aplicada em ambientes interativos

de aprendizagem, onde o usuário aprende pela interação com o ambiente de aprendizado (PONTES, 2010).

Greene e Rogers (2006) descrevem um *framework* para avaliação de estudantes da área de saúde, baseado na teoria da Flexibilidade Cognitiva, com o objetivo de verificar se os estudantes conseguem atingir as habilidades necessárias.

3.2.5 Teoria da Aprendizagem Significativa

Teoria proposta por Ausubel (AUSUBEL, 1963) que defende que a aprendizagem se dá através de um conhecimento prévio do indivíduo, sendo que o novo conhecimento é adquirido através da conexão com este conhecimento prévio e estruturas cognitivas presentes no estudante. Assim, um conhecimento significativo é obtido da interconexão de conhecimentos prévios que já estão estruturados na mente do aprendiz. Esta forma de aquisição de conhecimento parte de duas hipóteses fundamentais: a primeira é que as pessoas, em geral, aprendem partindo de um conceito mais geral e, a partir daí, assimilam melhor as partes específicas; a segunda é que a organização do conhecimento na mente ocorre de uma forma hierárquica, ou seja, conceitos mais gerais estão no topo e os mais específicos são incorporados posteriormente a estes conceitos prévios (RISSOLI; GIRAFFA; MARTINS, 2006).

Ausubel denomina os aspectos relevantes, nos quais serão agregados os demais conhecimentos, de conceito subsunçor (AUSUBEL, 1963). O conceito subsunçor é a base para agregação dos demais conhecimentos, sendo que a mente é formada por uma estrutura hierárquica com diversos conceitos subsunçores. Existe, no entanto, uma situação onde o aprendiz se depara com um conhecimento totalmente novo, sendo necessário formar um novo conceito subsunçor. Este novo conceito é formado a partir de estruturas denominadas organizadores prévios, que manipulam as estruturas cognitivas para estruturar o novo conceito na mente (PONTES, 2010).

3.2.6 Aprendizagem Baseada em Problema

Segundo Hmelo-Silver (2004), a PBL é um método no qual os estudantes aprendem através da resolução de um problema. Na PBL, o facilitador tem o papel de guiar os estudantes no processo de aprendizagem, identificando possíveis deficiências de conhecimento e habilidades necessárias à solução do problema proposto. Assim, nesta teoria de aprendizagem, ao invés de termos o professor simplesmente repassando os conhecimentos e depois testando-os através de avaliações, ele faz com que os estudantes apliquem o seu conhecimento em situações novas. Os estudantes se deparam com problemas muitas vezes mal estruturados e tentam descobrir, através da investigação e pesquisa, soluções úteis.

Na abordagem tradicional de ensino, primeiramente são apresentados os materiais de apoio e, em seguida, são feitas avaliações em cima destes materiais e aulas que foram lecionadas. Na PBL, primeiro é apresentado o problema e os estudantes necessitam procurar materiais e formas de resolvê-lo. O método de avaliação também é diferente, pois não consiste em provas, mas na avaliação do desenvolvimento das habilidades e conhecimento (PONTES, 2010).

3.2.6.1 Ciclo de Desenvolvimento da PBL

Para o sucesso da aplicação da PBL como estratégia pedagógica, os seguintes estágios devem ser cumpridos (HMELO-SILVER, 2004; PONTES, 2010):

- O facilitador propõe um problema mal estruturado para o grupo de estudantes, chamado cenário do problema;
- Os estudantes analisam o problema e extraem fatos relevantes ao problema em questão, através de um *brainstorming* inicial;
- Os estudantes têm um melhor entendimento do problema e formulam hipóteses para uma possível solução;
- Os estudantes, auxiliados pelo facilitador, identificam deficiências de conhecimento para solução do problema;
- Os estudantes procuram por novos conhecimentos relacionados ao domínio e tentam gerar fatos sobre este novo conhecimento;

- Ao final de cada problema, os estudantes refletem sobre os conhecimentos adquiridos.

A Figura 1 ilustra o ciclo de desenvolvimento da PBL.

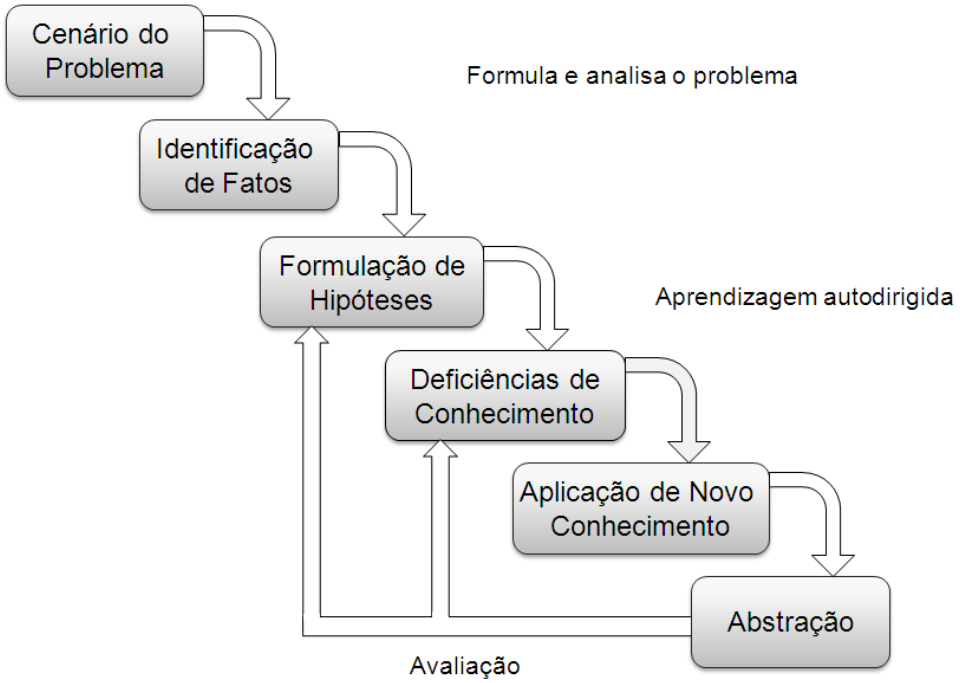

Figura 1 – Ciclo da PBL
Fonte: Adaptação de (HMELO-SILVER, 2004)

Mandal (2011) apresenta um *plug-in*, para o Moodle, de apoio à PBL. Esse *plug-in* consiste em um módulo de atividades e sua criação tem por intuito apoiar todos os passos que regem a PBL. Nesse trabalho, são apresentadas todas as adaptações realizadas no Moodle para atender à PBL. Esse trabalho, apesar de apresentar uma solução voltada para PBL, não faz uso de técnicas de IA, diferentemente do presente trabalho, que apresenta uma arquitetura multiagente para auxiliar essa teoria de aprendizagem.

3.2.7 Comparativo entre as Teorias de Aprendizagem

A Tabela 1 mostra um comparativo entre as teorias de aprendizagem apresentadas anteriormente. Os parâmetros avaliados foram (i) a aquisição de conhecimento, ou seja, a

forma como a teoria explica a origem do conhecimento no indivíduo; (ii) o foco da teoria, ou seja, o que ela enfoca como princípio da aprendizagem; e (iii) a aplicação da teoria, ou seja, onde ela é melhor aplicada.

Tabela 1 – Comparativo entre as teorias de aprendizagem

Teria	**Aquisição de Conhecimento**	**Foco**	**Aplicação**
Construtivismo Piagetiano	Interação com o ambiente (pessoas e objetos)	Na interação	Estudos de desenvolvimento inicial
Teoria Socio-Cultural de Vygotsky	Interação social	Colaboração e sociabilidade	Ambientes colaborativos
Cognição Distribuída	Interação com artefatos (ex. sistemas de navegação de aeronaves)	Indivíduos e artefatos	Estudos de interação entre indivíduo e artefatos
Flexibilidade Cognitiva	Interação com ambientes hipermídia	Ambientes hipermídia	Ambientes interativos de aprendizagem
Aprendizagem Significativa	Com base em conhecimentos prévios do indivíduo	Conhecimentos prévios	Ambientes tutoriais
Aprendizagem Baseada em Problema	Resolução de problemas	Problema	Centros de formação e ambientes colaborativos

Fonte: Adaptação de (PONTES, 2010)

As diversas teorias de aprendizagem apresentadas anteriormente oferecem variados métodos e estratégias para um aprendizado eficaz. A PBL, quando aplicada corretamente, oferece alguns benefícios, dentre os quais se destacam (HMELO-SILVER, 2004):

- Desenvolve pensamento crítico e criatividade no estudante;
- Aumenta sua capacidade de resolução de problemas;
- Aumenta a motivação; e
- Ajuda os estudantes a aplicarem os conhecimentos adquiridos em novas situações.

3.2.8 Uso de Ontologias em Ambientes de Aprendizagem

Na literatura são encontradas diversas definições sobre as ontologias. Uma das definições mais conhecidas é proposta por Gruber (1993):

> Uma ontologia é uma especificação explícita de uma conceitualização. [...] Em tal ontologia, definições associam nomes de entidades no universo do discurso (por exemplo, classes, relações, funções etc.) com textos que descrevem o que os nomes significam e os axiomas formais que restringem a interpretação e o uso desses termos [...].

A partir desse conceito, é possível perceber que uma ontologia é uma "especificação de um conceito", ou seja, ela é utilizada para especificar um conhecimento a respeito de um determinado domínio de conhecimento. Isso significa que uma ontologia permite a um projetista especificar, de uma forma aberta e significativa, os conceitos e relacionamentos que caracterizam de modo coletivo algum domínio. A vantagem de se utilizar uma ontologia é que, mesmo sendo desenvolvida com uma finalidade específica, ela pode ser publicada e reutilizada para outros propósitos (DICKINSON, 2009; SILVA, 2012).

Há muitas formas de escrever uma ontologia, e uma variedade de opiniões sobre que tipos de definição devem englobar. Porém, na prática, os conceitos de uma ontologia são largamente dirigidos pelos tipos de aplicação que elas terão de suportar (DICKINSON, 2009; SILVA, 2012).

Visto que são formas de representar o conhecimento de um dado domínio, as ontologias podem ser usadas em ambientes de aprendizagem para representação de conceitos e modelos inerentes ao ambiente em questão (PONTES, 2010). Por exemplo, Bittencourt *et al.* (2006) descrevem uma ontologia para apoiar a construção de ambientes interativos de aprendizagem. Utilizando-se de uma ontologia, são modelados diversos conceitos de um ambiente de aprendizagem, como: modelo do estudante, que representa o estudante que será ensinado; modelo pedagógico, que diz respeito às estratégias para aprendizagem; modelo de colaboração, que define a forma de colaboração no ambiente; e modelo de domínio, que refere-se ao que será ensinado.

As ontologias podem ser utilizadas com diversas finalidades em ambientes de aprendizagem, sendo uma das aplicações desta abordagem a personalização de ambientes de aprendizagem. Nozawa *et al.* (2010) propõem o desenvolvimento de um ambiente hipermídia

de aprendizagem adaptativo, auxiliado por ontologias, para o ensino de um novo idioma, que melhora o processo de ensino e aprendizagem.

Gava e Menezes (2003) propõem uma ontologia de domínio para a aprendizagem cooperativa, com o objetivo de fornecer uma conceituação explícita sobre estes elementos, ajudando outras pessoas a compreenderem melhor esta área de conhecimento e contribuindo para a construção de ambientes cooperativos mais fundamentados.

Duez-Rodriguez, Morales-Luna e Olmedo-Aguirre (2008) apresentam uma ontologia para buscar, descobrir e publicar materiais de aprendizagem relevantes, como OAs, para ajudar estudantes no cumprimento das fases da PBL.

Véras *et al.* (2008) propõem a criação de ontologias que representem formalmente ambientes educacionais de hipermídia, dada a complexidade da construção desses ambientes.

Isotani e Mizoguchi (2007) propõem um modelo baseado em ontologias que auxilia na análise das interações entre indivíduos e no planejamento de sessões efetivas de aprendizagem colaborativa, oferecendo recomendações baseadas nas teorias de aprendizagem.

Oliveira *et al.* (2010) apresentam a arquitetura, desenvolvimento e avaliação do Dr. Pierre, um *chatterbot* educacional com intenção e personalidade, que faz uso de ontologias, como base de conhecimento, e tem como objetivo apoiar o ensino e aprendizagem de Psiquiatria e Psicologia.

Diante dos trabalhos apresentados, é possível perceber a variedade de finalidades com as quais as ontologias podem ser utilizadas em ambientes colaborativos.

4 METODOLOGIAS DE MODELAGEM DE SISTEMAS MULTIAGENTE

Este capítulo descreve aspectos relacionados às metodologias de modelagem de SMAs. A Seção 4.1 traz uma breve discussão a respeito das principais metodologias de modelagem de SMAs. A Seção 4.2 apresenta os conceitos inerentes à metodologia MAS-CommonKADS. A Seção 4.3 apresenta detalhes da metodologia que será adotada neste trabalho, a MAS-CommonKADS+. Por fim, a Seção 4.4 apresenta um padrão utilizado para estabelecer a comunicação e o gerenciamento dos agentes.

4.1 PRINCIPAIS METODOLOGIAS DE MODELAGEM DE SMAs

Com o surgimento do paradigma de Programação Orientada a Agentes (POA), várias metodologias para a modelagem de SMAs foram propostas nos últimos anos, como, por exemplo, MAS-CommonKADS (IGLESIAS e GARIJO, 2005), MAS-CommonKADS+ (MORAIS II, 2010), MaSE (*Multiagent Systems Engineering*) (DELOACH e KUMAR, 2005), Tropos (CASTRO *et al.*, 2005), PASSI (*Process for Agent Societies Specification and Implementation*) (CONSSENTINO, 2005), Prometheus (PADGHAM e WINIKOFF, 2005), Gaia (ZAMBONELLI, JENNINGS e WOOLDRIDGE, 2005), ADELFE (ROUGEMAILLE, *et al.*, 2009), MESSAGE (GARIJO, GÓMEZ-SANZ e MASSONET, 2005) e INGENIAS (PAVÓN, GOMEZ-SANZ e FUENTES, 2005). A Tabela 2 mostra uma breve descrição de cada uma dessas metodologias.

Tabela 2 – Metodologias de Modelagem de SMAs

Metodologia	**Descrição**
MAS-CommonKADS	É uma extensão da metodologia CommonKADs (SCHREIBER, *et al.*, 2000) com abordagem voltada para modelagem de sistemas orientados a agentes. É dividida nas fases de conceituação, análise, projeto, codificação, integração, operação e manutenção.
MAS-CommonKADS+	Extensão da metodologia MAS-CommonKADS que adiciona novos conceitos e modelos (ex. modelo de requisitos e modelo de recursos e objetos) e estende a AML (*Agent Modeling Language*)

	(CERVENKA e TRENCANSKY, 2007) para suportar os recursos inseridos na nova metodologia.
MASE	Metodologia para análise e projeto de SMAs desenvolvida inicialmente para projetos fechados, heterogêneos e de propósito geral. Trata os agentes como especializações dos modelos de objetos, aplicando várias técnicas do paradigma orientado a objetos para especificação e projeto do SMA.
Tropos	Metodologia orientada a agentes e baseada no *framework i** proposto por (YU, 1995). Propõe a modelagem baseada em conceitos como atores, metas e dependências sociais entre atores para representar os requisitos, a arquitetura e o projeto detalhado do sistema.
PASSI	Metodologia passo a passo que vai desde os requisitos até o código. Utilizada no desenvolvimento de SMAs usando conceitos extraídos da orientação a objetos usando UML (*Unified Modeling Language*)[2]. Possui cinco modelos: de requisitos, de sociedade de agentes, de implementação de agentes, de código e de implantação.
Prometheus	Proporciona mecanismos para análise e projeto de SMAs baseados em arquiteturas BDI (*Belief Desire Intention*) (PADGHAM e WINIKOFF, 2005). Pode ser dividida, de forma macro, em três atividades: especificação de sistema, projeto da arquitetura e projeto detalhado.
Gaia	Primeira metodologia proposta para guiar o processo de desenvolvimento de SMAs, sendo aplicável a uma grande quantidade destes, lidando com características macro (sociedade) e micro (agente) do sistema. Baseia-se em um conjunto de abstrações, a saber: ambiente, papéis, interações, papéis organizacionais e estruturas organizacionais.
ADELFE	Metodologia especializada no desenvolvimento de SMAs adaptativos. É baseada no RUP (*Rational Unified Process*)[3], englobando desde os requisitos até o projeto de agentes adaptativos. Usa as notações da UML e, para modelagem dos protocolos de interação entre os agentes, a AUML (*Agent UML*)[4].
MESSAGE	Metodologia também baseada no RUP e que abrange as fases de análise e projeto no ciclo de vida do desenvolvimento de software. Estende a UML adicionando novas notações para a modelagem de conceitos relacionados a agentes, como papéis, organização e serviços.
INGENIAS	Metodologia criada a partir da MESSAGE, com diversas modificações. É baseada no processo de desenvolvimento do RUP e é composta das fases de análise, projeto e implementação, contendo cerca de setenta passos que guiam o processo de desenvolvimento.

Fonte: Adaptação de (SILVA, 2012)

[2] Maiores informações em: *www.uml.org*.
[3] Maiores informações em: *http://www-01.ibm.com/software/awdtools/rup/*.
[4] Maiores informações em: *http://www.auml.org/*.

Vale ressaltar que não faz parte do escopo deste trabalho apresentar um estudo comparativo detalhado entre as várias metodologias citadas. Caso seja necessário entender melhor os principais pontos de cada uma das metodologias, podem ser consultadas as análises comparativas realizadas em (PONTES, 2010) e (MORAIS II, 2010). Além disso, os próprios trabalhos citados em cada uma das metodologias também podem ser consultados.

Analisando-se os principais pontos fortes e fracos de cada metodologia, principalmente através dos trabalhos apresentados em (PONTES, 2010) e em (MORAIS II, 2010), a metodologia escolhida para modelagem dos agentes deste trabalho foi a metodologia MAS-CommonKADS+, proposta em (MORAIS II, 2010), que consiste em uma extensão à metodologia MAS-CommonKADS. Para uma melhor compreensão da metodologia MAS-CommonKADS+, é interessante, inicialmente, entender em que consiste a metodologia MAS-CommonKADS.

4.2 METODOLOGIA MAS-CommonKADS

A MAS-CommonKADS é uma metodologia de Engenharia de Software Orientada a Agentes (ESOA) (SILVA, 2012) que estende, com uma abordagem que guia o processo de análise e projeto de SMAs (IGLESIAS e GARIJO, 2005), a metodologia CommonKADS, que é a principal metodologia estruturada de suporte à engenharia do conhecimento (MORAIS II, 2010). A MAS-CommonKADS utiliza técnicas de modelagem bem conhecidas, tais como cartões CRC (*Class-Responsibility-Collaborator*), diagramas de caso de uso, diagramas de sequência de mensagens - MSC (*Message Sequence Charts*) e SDL (*Specification and Description Language*) com novas perspectivas dirigidas pela metáfora dos agentes (IGLESIAS e GARIJO, 2005; SILVA, 2012).

O ciclo de vida do desenvolvimento de software na MAS-CommonKADS segue as seguintes fases (IGLESIAS e GARIJO, 2005; SILVA, 2012):

- Contextualização: tarefa de elicitação com o intuito de obter uma primeira descrição do problema através da definição de um conjunto de casos de uso que auxiliam no entendimento do sistema e de como testá-lo.
- Análise: nessa fase são determinados os requisitos funcionais do sistema. O sistema é descrito através de um conjunto de modelos.

- Projeto: essa fase combina uma abordagem *top-down* e uma *bottom-up*, reutilizando componentes já desenvolvidos e desenvolvendo novos, dependendo da plataforma de agentes almejada. Recebe os modelos de análise como entrada e transforma-os em especificações (modelo de projeto) prontas para serem implementadas. A arquitetura interna de cada agente e a "arquitetura de rede" do sistema são determinadas.
- Desenvolvimento e testes: são realizadas as tarefas de codificação e testes dos agentes que foram previamente definidos.
- Operação: o sistema é colocado em operação e são realizadas tarefas de manutenção.

A metodologia define, para descrever as características de um agente e seus comportamentos sociais no SMA, sete modelos, os quais estão relacionados conforme a Figura 2 (IGLESIAS e GARIJO, 2005; MORAIS II, 2010; SILVA, 2012).

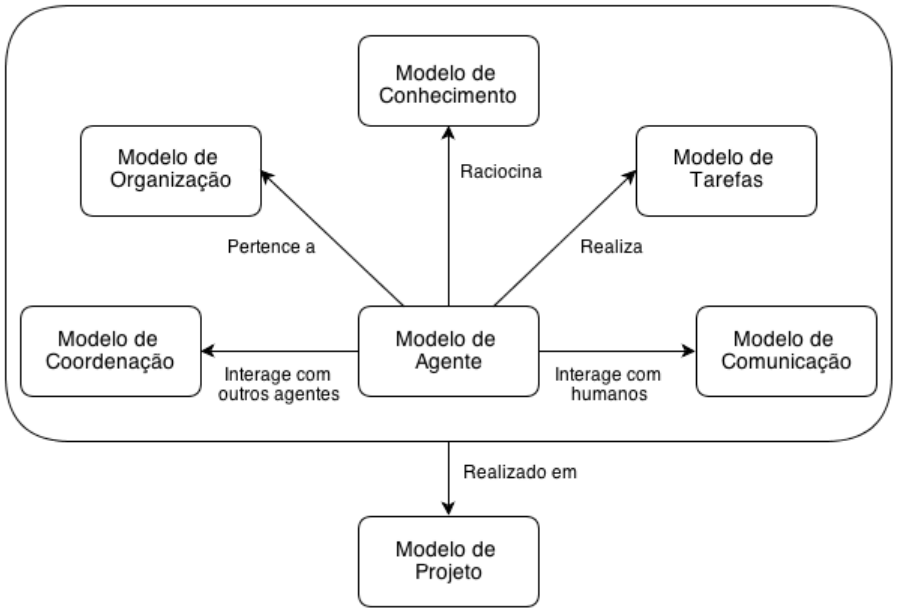

Figura 2 – Modelos da metodologia MAS-CommonKADS
Fonte: Adaptado de (IGLESIAS e GARIJO, 2005)

O modelo de agente especifica, através de *templates* textuais, as principais características dos agentes, tais como nome, tipo, papel, serviços oferecidos, metas e habilidades. É considerado o principal modelo da metodologia, visto que ele funciona como uma ligação entre o restante dos modelos, coletando as capacidades e restrições dos agentes.

O modelo de tarefas descreve todas as atividades (chamadas de tarefas) que devem ser realizadas com o intuito de alcançar determinada meta. As tarefas e subtarefas são demonstradas em um diagrama de tarefas, seguindo uma abordagem *top-down*. Também são utilizados *templates* textuais para descrever as entradas, saídas, pré-condições, pós-condições e meta de cada uma das tarefas.

O modelo de conhecimento, que é o foco da CommonKADS, é usado para modelar as capacidades de raciocínio dos agentes para realizar suas tarefas e atingir suas metas, ou seja, o conhecimento que cada agente possui para atingir seu objetivo.

O modelo de organização mostra os relacionamentos estáticos ou estruturais entre os agentes. A notação gráfica desse modelo é baseada no diagrama de classes da UML, adicionando um estereótipo para distinguir entre agentes e objetos.

O modelo de coordenação, diferentemente do modelo de organização, mostra os relacionamentos dinâmicos entre os agentes, ou seja, as conversações entre os agentes, suas interações e protocolos de comunicação. Ele utiliza várias técnicas orientadas a objetos para demonstrar a interação entre os agentes, dentre as quais diagramas de sequência de mensagem ou de comunicação, para modelar a comunicação entre os agentes, e diagramas de transição de estados, para modelar o processamento de transações.

O modelo de comunicação descreve as interações entre um agente humano e um agente de software.

No modelo de projeto, todos os modelos previamente criados são coletados de modo a contribuírem com a criação do projeto, que consiste em três submodelos: (i) o projeto de rede, para projetar os aspectos relevantes da infraestrutura de redes dos agentes, como coordenação dos agentes (facilidades para gerência de grupos) e a rede de comunicação (páginas amarelas/brancas ou *agent name service* (BELIFEMINE, CAIRE e GREENWOOD, 2007)); (ii) o projeto dos agentes, para determinar a arquitetura mais adequada para cada agente; e (iii) o projeto da plataforma, para selecionar a plataforma de desenvolvimento de agentes que suporte as arquiteturas de cada agente (MORAIS II, 2010; SILVA, 2012).

4.3 METODOLOGIA MAS-CommonKADS+

A metodologia MAS-CommonKADS+ mantém muitos dos modelos já propostos na metodologia MAS-CommonKADS, porém realiza algumas modificações e adiciona novos

conceitos. Ao invés de definir sete modelos, a MAS-CommonKADS+ contém nove modelos. Esses modelos são representados através de notações da AML, que consiste em uma linguagem de modelagem especificada como uma extensão da UML 2.0, com o intuito de especificar, modelar e documentar SMAs (SILVA, 2012).

Foram adicionados à metodologia os modelos de requisitos, de papéis e de recursos, enquanto que os modelos de organização, de interação e de projeto foram alterados, com o intuito de complementar a especificação dos diagramas da AML. O modelo de agentes foi o que sofreu mais alterações, possibilitando agora demonstrar como o agente irá perceber e atuar no ambiente de acordo com seus comportamentos e planos. A Figura 3 mostra a arquitetura da MAS-CommonKADS+ (SILVA, 2012).

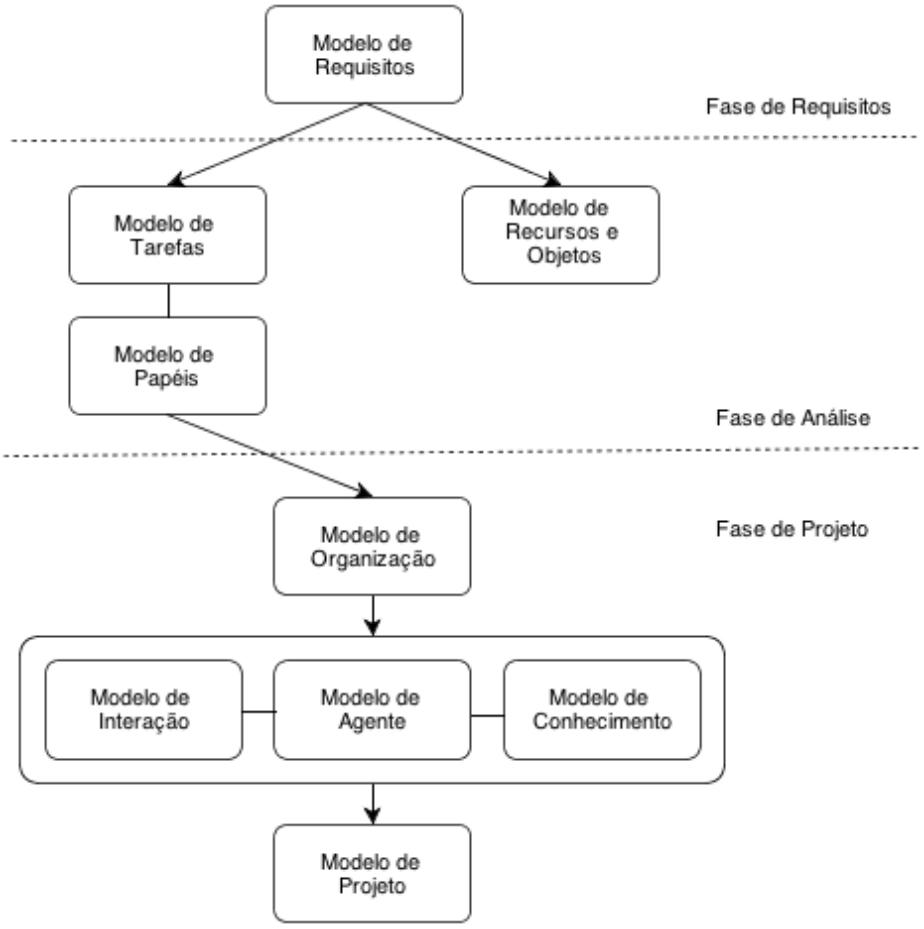

Figura 3 – Arquitetura da MAS-CommonKADS+
Fonte: Adaptado de (MORAIS II, 2010)

O modelo de requisitos é utilizado para descrever os requisitos do sistema, sendo dividido em análise de casos de uso e cenários, análise por objetivos e análise do ambiente.

O modelo de papéis tem como objetivo a identificação dos papéis do sistema e a representação de papéis que realizam as tarefas descritas no modelo de tarefas. Um papel é uma abstração que define as tarefas que um agente deve realizar dentro de uma organização. Um único agente pode ser responsável por vários papéis em um sistema. Logo, a inclusão deste modelo é de extrema relevância, pois permite modelar os vários papéis que um agente pode exercer, o que não é possível ser realizado na MAS-CommonKADS.

O modelo de recursos e objetos foi adicionado com o intuito de possibilitar a modelagem de objetos e recursos, permitindo assim uma melhor definição do sistema, visto que não era algo possível de ser modelado na MAS-CommonKADS.

O modelo de organização descreve a estrutura organizacional de papéis do sistema, e não mais a organização de agentes, como acontecia na MAS-CommonKADS.

O modelo de interação consiste na junção dos modelos de coordenação e de comunicação da MAS-CommonKADS. Nele são descritas, através da AML, todas as interações entre agentes. Cada interação deve obedecer a um protocolo de interação, o qual estabelece como os agentes podem se comunicar.

O modelo de agentes especifica os agentes, e por quais papéis eles são responsáveis, as percepções, os atuadores, as condições de ativação e de parada e a arquitetura do agente. O primeiro passo para construção desse modelo é descrever os agentes do sistema, identificando-os e definido os seus papéis. Em seguida, é realizada a identificação dos comportamentos internos do agente de acordo com sua arquitetura.

O modelo de projeto descreve as características do local onde o sistema será instalado, os diagramas de implantação e informações a respeito da mobilidade dos agentes. Este modelo tem o intuito de facilitar o entendimento da infraestrutura, além de trazer informações sobre como componentes do SMA podem ser localizados, distribuídos e conectados.

Os modelos de tarefas e de conhecimento continuam sendo utilizados conforme especificado na MAS-CommonKADS (MORAIS II, 2010; SILVA, 2012).

4.4 COMUNICAÇÃO E GERENCIAMENTO DE AGENTES

Um dos componentes chaves de um SMA é a comunicação entre os agentes. Na realidade, agentes precisam se comunicar para que sejam capazes de cooperar, colaborar e negociar entre si. De uma forma geral, os agentes interagem entre si através de algumas

linguagens de comunicação especiais, chamadas linguagens de comunicação entre agentes. Atualmente, a linguagem de comunicação entre agentes mais difundida e utilizada é a FIPA (*Foundation for Intelligent Physical Agents*) ACL (*Agent Communication Language*). As principais características da FIPA ACL são a possibilidade de utilizar linguagens de conteúdos diferentes e o gerenciamento de conversações através de protocolos de interação predefinidos (BELIFEMINE, CAIRE e GREENWOOD, 2007; SILVA, 2012).

FIPA é um conjunto de padrões da IEEE (*Institute of Electrical and Electronics Engineers*) cujo objetivo é promover (i) as tecnologias baseadas em agentes, (ii) a interoperabilidade desses padrões com outras tecnologias, (iii) a interoperação de agentes heterogêneos e (iv) os serviços que eles podem representar (FIPA, 2011; SILVA, 2012).

Além de prover um padrão e uma linguagem comum através da qual os agentes podem se comunicar, FIPA define um modelo lógico de referência para gerenciamento dos agentes. Desta forma, as plataformas que atendem à sua especificação devem implementar esse modelo, possibilitando assim criação, registro, localização, comunicação, migração e operação dos agentes (BELIFEMINE, CAIRE e GREENWOOD, 2007; SILVA, 2012). A Figura 4 mostra os componentes desse modelo.

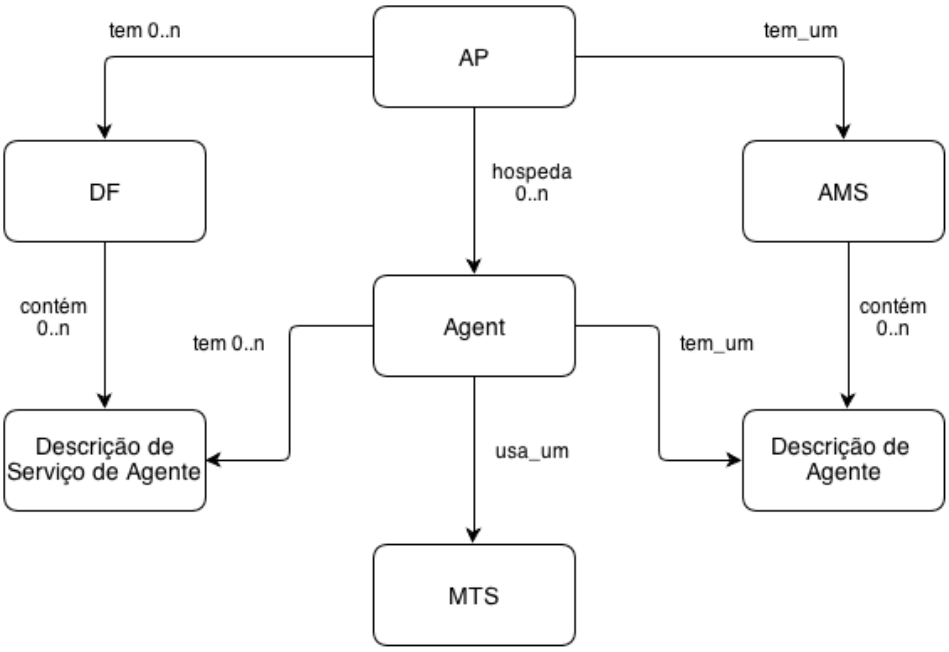

Figura 4 – Modelo de referência FIPA para gerenciamento de agentes
Fonte: Adaptado de (BELIFEMINE, CAIRE e GREENWOOD, 2007)

Como pode ser visto na Figura 4, os seguintes componentes podem ser visualizados nesse modelo (SILVA, 2012):

- *Agent Platform* (AP): provê a infraestrutura física mínima na qual os agentes são executados. Consiste de máquinas, sistemas operacionais, componentes para gerenciamento de agentes FIPA (descritos a seguir), os agentes em si e qualquer software de suporte adicional.
- *Agent Management System* (AMS): componente obrigatório de uma AP responsável por gerenciar as operações desta, tais como criação e destruição de agentes, e por fiscalizar as migrações dos agentes entre APs. Em cada AP existe apenas um AMS e, caso a AP seja composta de várias máquinas, o AMS será a autoridade máxima entre todas elas. Cada agente deve se registrar com o AMS para que possa obter um AID (*Agent Identifier*), identificador que o identifica de forma única dentro da AP. O AMS mantém, então, um diretório contendo todos os AIDs e os respectivos estados dos agentes (ex. ativo, suspenso ou em espera), serviço este chamado de páginas brancas.
- *Agent*: processo computacional que reside em uma AP e oferece um ou mais serviços computacionais, que podem ser publicados como uma descrição de serviço. Cada agente deve ter, obrigatoriamente, um AID.
- *Directory Facilitator* (DF): componente opcional de uma AP que provê, para outros agentes, o serviço de páginas amarelas. Esse serviço consiste em uma lista de todos os agentes e os respectivos serviços oferecidos por cada um destes. Todo e qualquer agente que deseje publicar seus serviços para outros agentes devem encontrar um DF apropriado e requisitar o registro da sua descrição de serviço. Além disso, os agentes podem retirar e modificar o seu próprio registro de um DF e buscar neste, de acordo com um critério de busca, por um registro de serviço fornecido por outro agente. Esse componente é de extrema importância para o SMA proposto neste trabalho.
- *Message Transport Service* (MTS): também conhecido como ACC (*Agent Communication Channel*), o MTS é o serviço provido por uma AP para transportar mensagens FIPA-ACL entre os agentes em uma determinada AP e entre agentes localizados em APs distintas.

5 ALGORITMOS GENÉTICOS

Este capítulo traz a fundamentação teórica necessária para que seja possível entender o funcionamento do AG proposto neste trabalho. A Seção 5.1 mostra os conceitos básicos dos AGs. A Seção 5.2 apresenta a estrutura básica de um AG.

5.1 CONCEITOS BÁSICOS DOS ALGORITMOS GENÉTICOS

Os AGs consistem em um ramo dos algoritmos evolucionários e, portanto, seu funcionamento é bastante similar a este tipo de algoritmo (SILVA, 2012). Os algoritmos evolucionários são aqueles que usam modelos computacionais dos processos naturais de evolução como uma ferramenta para resolver problemas (LINDEN, 2008). Mesmo existindo uma grande variedade de modelos computacionais propostos, todos eles possuem o mesmo princípio básico: todos simulam a evolução das espécies baseando-se na teoria da evolução humana e utilizando os denominados operadores genéticos (LINDEN, 2008; PETROLI NETO, 2011; SILVA, 2012).

De uma forma sucinta, os algoritmos genéticos tentam resolver problemas para os quais não existe um algoritmo conhecido, gerando-se uma população inicial e, de acordo com critérios de avaliação, selecionando os melhores indivíduos dessa população, que servirão como solução para o problema ou, caso contrário, serão combinados para obter uma nova geração. Esse processo é repetido até que se encontre uma solução ou até que se perceba que não serão alcançadas melhores soluções nas novas gerações (ARTERO, 2009; SILVA, 2012).

Porém para entender melhor o funcionamento de um AG é necessário ter em mente alguns conceitos.

5.1.1 Definições

5.1.1.1 Genes

Na genética, constituem o material genético que poderá ser trocado entre os cromossomos durante a reprodução (PETROLI NETO, 2011). Na abordagem dos AGs, correspondem a uma representação de algum parâmetro de interesse, seguindo algum alfabeto pré-estabelecido, podendo ser representados por valores inteiros, reais e cadeias de caracteres (ARTERO, 2009). A utilização mais comum é usar apenas os valores 0 e 1 de um alfabeto binário para representar cada gene, consistindo apenas em uma divisão do cromossomo (ARTERO, 2009; PETROLI NETO, 2011; SILVA, 2012).

5.1.1.2 Cromossomos ou Indivíduos

Os cromossomos são compostos por uma cadeia de genes e representam os indivíduos da população, os quais, no caso dos AGs, representam as soluções encontradas em um problema de otimização (ARTERO, 2009). Vale ressaltar que, nos sistemas naturais, um ou mais cromossomos se combinam para formar um indivíduo, porém, no caso dos AGs, os termos cromossomo e indivíduo são intercambiáveis, sendo utilizados como sinônimos (LINDEN, 2008; SILVA, 2012).

5.1.1.3 População

É formada por um conjunto de indivíduos (soluções) que irão competir pela sobrevivência e pela reprodução, com o intuito de perpetuar suas características para as próximas gerações (ARTERO, 2009; SILVA, 2012).

5.1.1.4 Geração

Uma geração corresponde a uma população em certo período. No caso dos AGs, representa os valores dos indivíduos obtidos após um determinado número de execuções (iterações) (ARTERO, 2009; SILVA, 2012).

5.1.1.5 Função de Aptidão

Também chamada de *Função Objetivo* ou *Função de Fitness* (ZINI, 2009) ou mesmo *Função de Avaliação* (LINDEN, 2008), é usada para medir a habilidade do indivíduo para sobreviver e se reproduzir (ARTERO, 2009). O funcionamento correto de um AG depende fortemente desta função, pois ela faz a ligação do algoritmo com o problema real (PETROLI NETO, 2011), motivo pelo qual é destinada uma seção mais adiante neste trabalho especificamente para discussão dos aspectos relacionados a essa função.

5.2 ESTRUTURA BÁSICA DE UM ALGORITMO GENÉTICO

Tendo em mente os conceitos apresentados na seção anterior, é possível definir uma estrutura básica de AG. Algoritmicamente, um AG pode ser descrito através dos seguintes passos (LINDEN, 2008; PETROLI NETO, 2011; SILVA, 2012):

1. Inicializa-se a população de cromossomos inicial;
2. Avalia-se cada um dos cromossomos na população através de uma função de aptidão, de modo a encontrar uma classificação dos indivíduos mais adaptados ao problema;
3. Seleciona-se os indivíduos que servirão como pais para criação de uma nova população de acordo com uma estratégia de seleção previamente definida;
4. Aplica-se os operadores genéticos (cruzamento e mutação) aos indivíduos selecionados no passo anterior para gerar os indivíduos da nova população;

5. Elimina-se os cromossomos da população antiga, de forma que os novos cromossomos gerados possam ser inseridos sem alterar o tamanho da população inicial;
6. Aplica-se a função de aptidão a todos os novos cromossomos e insere-se os melhores selecionados na população anterior, gerando uma nova população;
7. Se a população de cromossomos atual representar o resultado esperado ou se a quantidade máxima de gerações foi atingida ou ainda se o algoritmo não conseguir mais mostrar evolução, a execução deve parar, do contrário, o passo 3 deve ser reiniciado.

A estrutura básica de um AG pode ser representada conforme a ilustração da Figura 5.

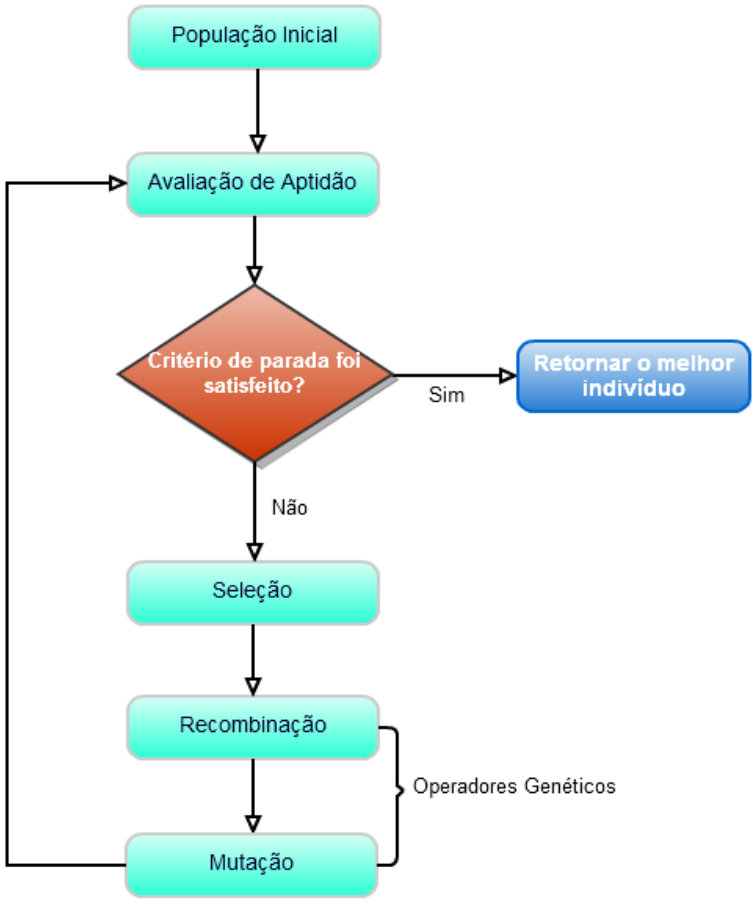

Figura 5 – Estrutura básica de um AG
Fonte: Adaptado de (ZINI, 2009)

Ao final da execução do algoritmo apresentado, espera-se que a população de cromossomos gerada seja a melhor adaptada à função e, por conseguinte, a que melhor

represente o resultado do problema. Porém, apesar do algoritmo apresentado na Figura 5 fornecer uma representação geral e esclarecedora a respeito do funcionamento de um AG, é apenas uma visão de alto nível do problema. Ou seja, ele esconde aspectos mais complexos que devem ser tratados, tais como (LINDEN, 2008; PETROLI NETO, 2011; SILVA, 2012):

- Escolha de uma codificação dos cromossomos adequada ao problema;
- Definição do tamanho da população necessária;
- Definição da forma como será realizada a mutação; e
- Seleção de uma função de aptidão que avalie satisfatoriamente o grau de adequação de cada indivíduo como solução do problema em questão.

Esses são os principais pontos a serem tratados quanto ao AG em questão e, portanto, serão detalhados na seção que trata da implementação do AG. Porém, outros detalhes podem ser de grande valia quando da construção de um AG. Caso seja necessário obter mais detalhes a respeito de tais informações, recomenda-se a consulta ao trabalho proposto por (SILVA 2012).

6 ARQUITETURA MULTIAGENTE DE APOIO À PBL

Este capítulo descreve a arquitetura multiagente de apoio à PBL desenvolvida neste trabalho. A Seção 6.1 traz uma descrição das principais ferramentas, bibliotecas e *frameworks* que foram utilizados no desenvolvimento desse trabalho. A Seção 6.2 mostra a arquitetura completa e apresenta uma descrição dos principais componentes da mesma. A Seção 6.3 mostra detalhes da arquitetura multiagente proposta no presente trabalho e da modelagem dos agentes, além de descrever como é realizada a comunicação entre os agentes.

6.1 FERRAMENTAS UTILIZADAS

Para o desenvolvimento da arquitetura apresentada nesse trabalho foram utilizadas várias bibliotecas, *frameworks* e ferramentas provenientes de projetos *Open Source*. Nas próximas subseções, são fornecidas informações técnicas sobre as principais ferramentas utilizadas, de forma a auxiliar no entendimento da arquitetura proposta neste trabalho.

6.1.1 Moodle

O Moodle é um AVA desenvolvido com a linguagem PHP (PHP, 2011) e que utiliza um banco de dados MySQL (MYSQL, 2011). É distribuído livremente sobre a licença GNU GPL (*General Public License*) (GPL, 2007) e pode ser utilizado por aqueles que desejam disponibilizar cursos a distância através da Web (MOODLE, 2011; SILVA, 2012).

6.1.2 *Framework* JADE

O JADE (*Java Agent Development Framework*) consiste em uma plataforma completa para desenvolvimento e execução de SMAs. Este *framework* Java permite desenvolver SMAs

e aplicações de acordo com os padrões FIPA para agentes inteligentes. Ele inclui dois componentes principais: uma plataforma de agentes compatível com os padrões FIPA e um pacote para desenvolvimento de agentes. Além disso, ele também possui uma interface gráfica que pode ser utilizada durante as fases de desenvolvimento e de teste dos agentes (BELIFEMINE, CAIRE e GREENWOOD, 2007; SILVA, 2012).

Em termos da cobertura dos padrões FIPA, o JADE implementa a especificação completa de gerenciamento de agentes, incluindo os serviços chave de AMS, DF, MTS e ACC (ver Seção 4.4). O JADE estende esses serviços com características adicionais, mas a conformidade com o padrão FIPA é mantida. O JADE também implementa, completamente, a pilha de comunicação de agentes FIPA, englobando desde a FIPA-ACL, para estrutura de mensagens, até o suporte para muitos dos protocolos de transporte e de interação FIPA (BELIFEMINE, CAIRE e GREENWOOD, 2007; SILVA, 2012).

6.1.3 StarUML

O StarUML (STARUML, 2011) é um projeto *Open Source* cujo intuito é disponibilizar uma ferramenta de modelagem de software e uma plataforma que possa substituir completamente ferramentas UML comerciais. O StarUML permite que sejam utilizadas várias linguagens para desenvolvimento de módulos para a ferramenta (SILVA, 2012).

Em (MORAIS II, 2010), foi proposta uma extensão para o StarUML para auxiliar o projeto de SMAs utilizando a metodologia MAS-CommonKADS+. Toda a modelagem realizada no presente trabalho foi criada através dessa extensão.

6.1.4 Blender

Para a modelagem e animação do agente pedagógico animado proposto neste trabalho foi utilizada a ferramenta Blender (BLENDER, 2013; BRITO, 2010), desenvolvida e mantida pela *Blender Foundation* (BLENDER, 2013). Sua escolha se deve ao fato de ser uma ferramenta robusta, de código aberto, disponível sob licença GNU BL (*Blender License*)

(BLENDER, 2013), e por possuir uma comunidade ativa que mantém a ferramenta em constante atualização (FROZZA *et al.*, 2009).

6.2 ARQUITETURA PROPOSTA

A arquitetura multiagente proposta neste trabalho é representada na Figura 6.

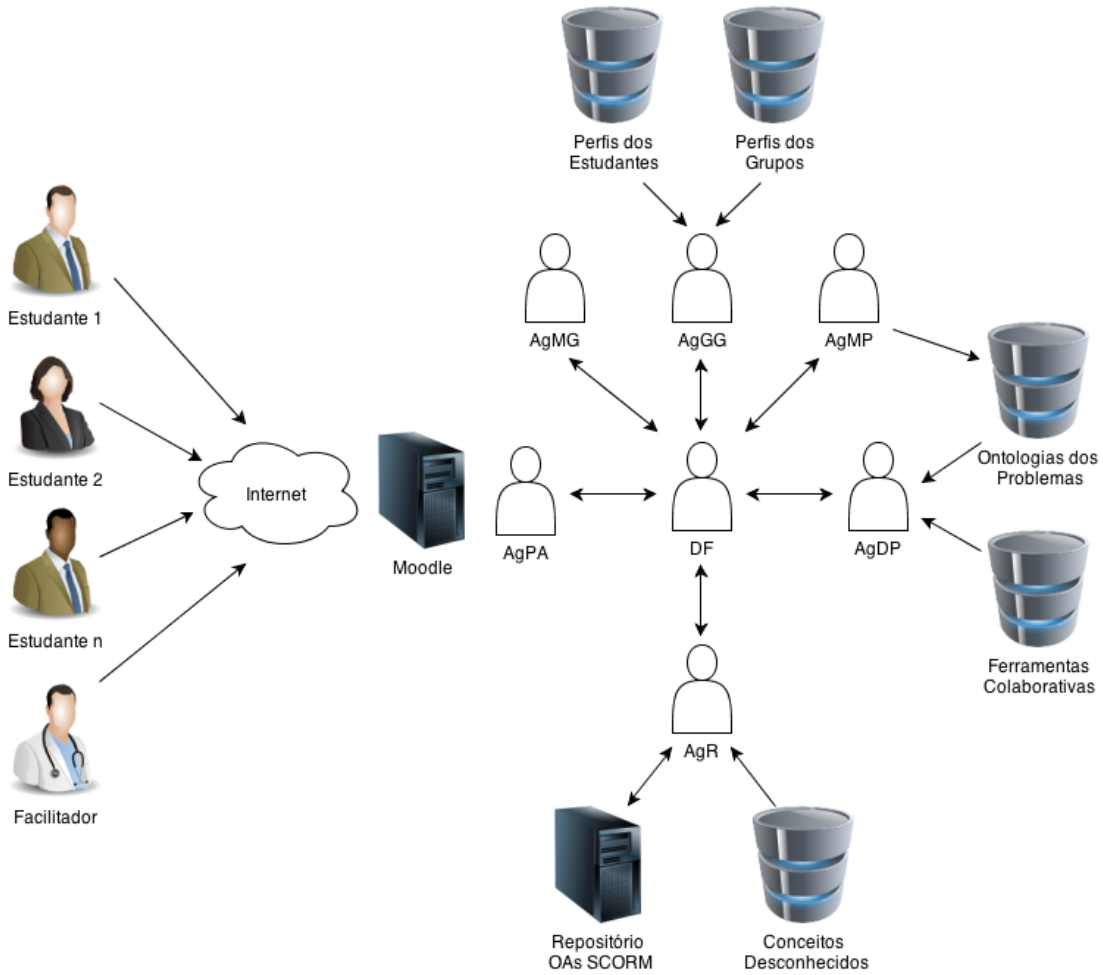

Figura 6 – Arquitetura de apoio à PBL
Fonte: Dados produzidos pelo autor

Como pode ser visto na Figura 6, os estudantes devem, inicialmente, se autenticar no Moodle e acessar algum dos cursos nos quais esteja matriculado. Os principais componentes dessa arquitetura são:

- SMA: arquitetura multiagente criada com o objetivo de auxiliar no cumprimento da implantação da PBL. Esse SMA será detalhado na próxima seção.
- AVA: o AVA utilizado neste trabalho é o Moodle, que é um ambiente de código-fonte aberto utilizado para disponibilização e realização de cursos a distância (MOODLE, 2011).
- Perfis dos Estudantes: referem-se à base de perfis dos estudantes. Os estudantes preenchem seus respectivos perfis, via interface do Moodle, no início do processo. O perfil dos estudantes é composto por habilidades, conhecimentos e deficiências, onde cada um possui um nível, que pode ser baixo, médio ou alto, podendo um estudante ter uma ou mais habilidades, deficiências e conhecimentos. Os perfis dos estudantes são consultados no processo de formação de grupos.
- Perfis dos Grupos: referem-se aos perfis dos grupos desejados para cada problema a ser solucionado. Esses perfis são criados pelo facilitador, via interface do Moodle, da mesma forma que o estudante. Um perfil de grupo é composto por habilidades, conhecimentos e deficiências, cada um possuindo um nível, que pode ser baixo, médio ou alto, bem como um valor *fuzzy*, que varia de 0.1 a 1.0 e está vinculado aos valores baixo, médio e alto. Os perfis dos grupos são consultados no processo de formação de grupos.
- Ontologias dos Problemas: são as ontologias utilizadas para armazenamento e consulta das informações referentes aos problemas propostos durante a aplicação da PBL. O facilitador é responsável por criar esses problemas. Sendo assim, para cada novo problema criado, uma nova ontologia é instanciada com as informações fornecidas manualmente pelo facilitador, como, por exemplo, o título, a definição, as palavras relacionadas e não relacionadas ao problema, dentre outras.
- Ferramentas Colaborativas: são as ferramentas colaborativas usadas pelos estudantes para colaboração e comunicação durante a resolução do problema. As informações referentes a essas interações são armazenadas e consultadas durante a detecção de problemas.
- Conceitos Desconhecidos: referem-se aos conceitos que os estudantes desconhecem na definição do problema proposto. Esses conceitos desconhecidos são fornecidos pelo próprio estudante, via interface do Moodle, durante a aplicação da PBL. Essas informações, dentre outras, são consideradas na recomendação de OAs.

- Repositório de OAs SCORM: é o repositório utilizado para armazenamento dos OAs, os quais devem ser desenvolvidos seguindo o padrão SCORM (*Sharable Content Object Reference Model*) (SCORM, 2013). Através das informações fornecidas no SCORM o agente é capaz de comparar as informações dos OAs com os perfis dos estudantes.

6.3 SISTEMA MULTIAGENTE

A organização do SMA é do tipo comunidade de especialistas, pois cada um dos sete tipos de agentes criados para este trabalho encontra-se no mesmo nível, sendo cada um deles especialista em determinada tarefa. Os agentes interagem entre si através de um protocolo de comunicação previamente estabelecido. Para o desenvolvimento dos agentes, foram utilizadas as bibliotecas do JADE e os mesmos executam sob esta plataforma.

Como é possível perceber na Figura 6, o SMA é composto de sete tipos de agentes: Agente Pedagógico Animado (AgPA), Agente Monitorador de Problemas (AgMP), Agente Detector de Problemas (AgDP), Agente Monitorador de Grupos (AgMG), Agente Gerenciador de Grupos (AgGG), Agente Recomendador (AgR) e Agente DF (*Directory Facilitator*).

As subseções a seguir fornecem uma noção de como foi criada a arquitetura do presente trabalho. Porém, são necessárias maiores informações de como essa arquitetura foi modelada em detalhes. A Subseção 6.3.8 apresenta a modelagem do SMA proposto nesse trabalho, seguindo os passos estabelecidos pela metodologia MAS-CommonKADS+.

6.3.1 Agente Pedagógico Animado – AgPA

O AgPA foi implementado com o intuito de apoiar os estudantes na resolução de problemas, através da teoria de aprendizagem PBL. O AgPA consiste em um modelo humanoide tridimensional animado responsável por acompanhar os estudantes durante o processo de aplicação da PBL, além de manter os estudantes sempre motivados. Para obter

sucesso nesse último caso, o AgPA expressa emoções similares às dos seres humanos, conforme ilustrado na Figura 7.

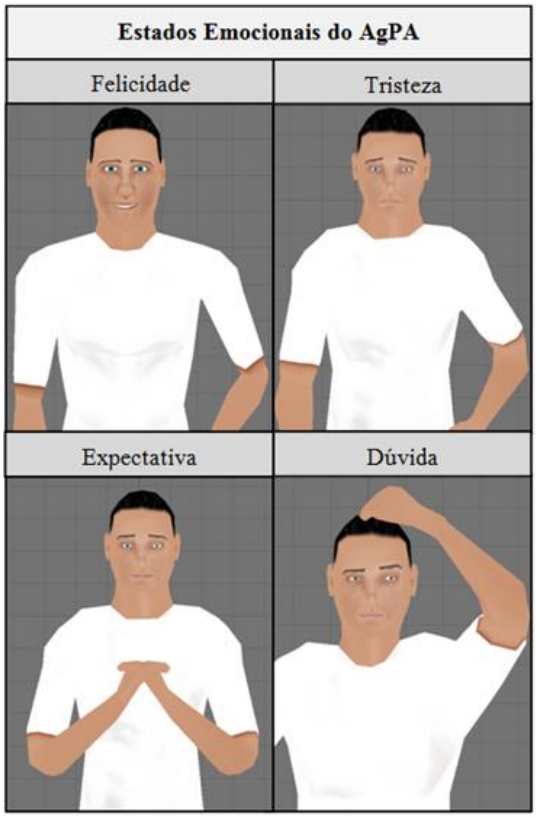

Figura 7 – Animações referentes aos estados emocionais do AgPA
Fonte: Dados produzidos pelo autor

Até a escrita deste trabalho, foram modeladas e implementadas quatro animações para expressar as emoções do AgPA, conforme ilustrado na Figura 7: felicidade, que remete, por exemplo, momentos em que o estudante esteja interagindo com o ambiente; tristeza, quando algum problema de colaboração for detectado, como, por exemplo, a detecção de estudantes passivos; expectativa, durante os questionamentos do AgPA para o estudante; e dúvida, quando o estudante permanecer muito tempo sem interagir com o ambiente.

O AgPA se mantém ativo durante toda a permanência do estudante no ambiente. Ele se comunica com os outros agentes, de forma colaborativa, e atua de acordo com o que for constatado no ambiente.

6.3.1.1 Modelagem do AgPA

Na modelagem da forma do AgPA foi utilizada a técnica de malha poligonal (FOLEY *et al.*, 1996), conforme ilustrada na Figura 8. Essa malha poligonal foi produzida utilizando uma *Blueprint* (FUNK; AYMONE, 2010). Durante a modelagem da malha, foi empregada a técnica de modelagem denominada *Low Poly* (TOTTEN, 2012). Essa técnica constitui na redução de polígonos da malha, necessária para que o AgPA tenha um melhor desempenho ao ser inserido no ambiente Web.

A confecção da textura foi feita utilizando a ferramenta *Gimp* (KYLANDER; KYLANDER, 1999) (ilustrada na Figura 8), tendo como referência a representação 2D do modelo gerado a partir da técnica de mapeamento UV (BRITO, 2010; FUNK; AYMONE, 2010).

O controle das poses do AgPA foi feito utilizando o modificador *Armature* (BRITO, 2010) (ilustrado na Figura 8), que permite aplicar movimentos articulados à malha e cinemática inversa. As animações foram realizadas utilizando as técnicas *Key Frames* (AZEVEDO; CONCI, 2003).

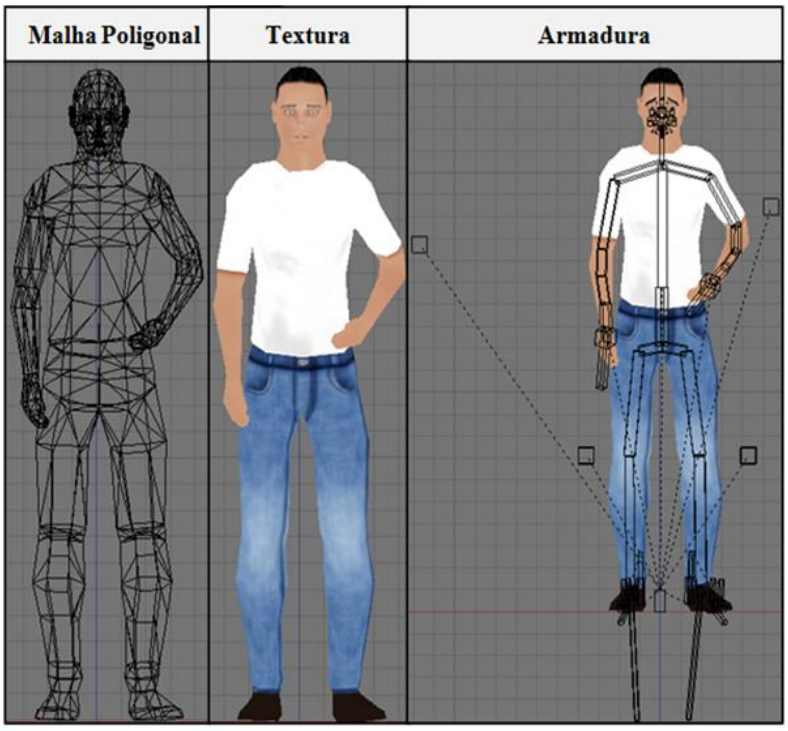

Figura 8 – Malha poligonal, textura e armadura do AgPA
Fonte: Dados produzidos pelo autor

Para renderizar o AgPA, foi utilizado o motor de jogos *jMonkey* (DONGLAI; GOUXI; QIUXIANG, 2010). Para isso, foi necessário adaptar e exportar o modelo do AgPA do Blender para o formato do *Ogre3D* (JUNKER, 2006), um dos tipos de arquivos suportados pelo motor.

6.3.2 Agente Monitorador de Problemas – AgMP

O AgMP é responsável pelo monitoramento da criação de novos problemas. Este agente possui dois comportamentos: o primeiro tem o objetivo de instanciar uma ontologia do problema, com todas as informações do problema inseridas pelo facilitador; já o segundo, tem o objetivo de enviar uma mensagem para o AgDP, com o intuito de acionar os seus comportamentos de detecção de problemas.

6.3.3 Agente Detector de Problemas - AgDP

O AgDP é responsável pela detecção de estudantes passivos e detecção de conversações fora do contexto do problema. O AgDP executa estes comportamentos uma vez por dia, durante toda a realização de um curso. Esse agente foi criado com o intuito de auxiliar o facilitador na avaliação do comportamento dos estudantes durante o processo de aplicação da PBL. Desta forma, uma vez sendo detectado um comportamento indesejado, o AgDP irá notificar o facilitador, e esse, por sua vez, poderá tomar uma providência cabível. As subseções a seguir descrevem esses comportamentos em mais detalhes.

6.3.3.1 Detecção de Estudantes Passivos

O AgDP tem a função de detectar os estudantes passivos. Para que o AgDP consiga cumprir com esta meta, é necessário que ele atualize os perfis dos estudantes conforme o uso das ferramentas colaborativas disponíveis no ambiente. Nesta abordagem, para cada ação executada pelo estudante no ambiente, este é pontuado com base em uma tabela cujos valores

são previamente definidos. Apenas para facilitar a compreensão, definimos as pontuações apresentadas na Tabela 3.

Tabela 3 – Tabela de pontuação de participação

Interação	**Pontuação**
Visualizar o chat	5
Escrever uma mensagem no chat	5
Visualizar o fórum	5
Adicionar uma postagem no fórum	10
Adicionar uma nova discussão no fórum	30

Fonte: Dados produzidos pelo autor

Esta informação será usada posteriormente pelo AgDP para realizar a detecção dos estudantes passivos propriamente dita. Os passos do processo de detecção de estudantes passivos executados pelo AgDP são descritos no Algoritmo 1:

Algoritmo 1: Detecção de estudantes passivos

Considerando:
estudante: estudante resolvendo um problema na PBL
grupo_estudantes: conjunto de estudantes
serie: conjunto de pontos de um grupo de estudantes
e: elemento da série
c1; c2: elementos centrais da série
ordenar_serie(serie): ordena valores em ordem crescente
quantidade_de_valores: número de elementos da série
lim_inf: limite inferior para valores discrepantes
lim_sup: limite superior para valores discrepantes
limiar_deteccao_passivos: limite de participação definido
lista_passivos : armazena resultado do algoritmo

```
1:  para todo grupo_estudantes faça
2:      ordenar_serie(serie)
3:      se mod(quantidade_de_valores / 2) = 0 então
4:          mediana = (c1 + c2) / 2
5:      senão
6:          posicao_mediana = (quantidade_de_valores + 1) / 2
7:          mediana = serie(posicao_mediana)
8:    fim se
9:      para todo e ∈ serie faça
10:         se e / mediana > lim_inf ^ e / mediana < lim_sup então
```

```
11:             somatorio = somatorio + e
12:             numero_elementos = numero_elementos + 1
13:         fim se
14:     fim para
15:     media = somatorio / numero_elementos
16:     para todo estudante Є grupo_estudantes faça
17:         se e / media < limiar_deteccao_estudantes então
18:             lista_passivos = estudante
19:         fim se
20:     fim para
21:     retorne lista_passivos
22: fim para
```

Como pode ser visto no Algoritmo 1, inicialmente é calculada a média de participação dos estudantes (conforme o uso das ferramentas colaborativas), eliminando os valores discrepantes (*outliers*). A detecção de valores discrepantes é realizada calculando-se a mediana dos valores da série (conjunto de pontos de um grupo de estudantes). A discrepância é eliminada de acordo com um limiar pré-estabelecido em relação à mediana dos valores. Em seguida, é calculada a média aritmética dos valores restantes, refletindo melhor a tendência da série. Com base na média dos valores restantes, é possível detectar um estudante passivo que diste do limiar pré-estabelecido pelo facilitador.

Após detectar um estudante passivo, o AgDP notifica o facilitador, via e-mail, informando todas as informações inerentes ao referente estudante. Em seguida, todas as informações relativas ao estudante passivo são enviadas para o AgPA.

6.3.3.2 Detecção de Conversações Fora do Contexto

O AgDP também é responsável por detectar as conversações fora de contexto baseado no uso das ferramentas colaborativas disponíveis no ambiente e em uma ontologia do problema que está sendo resolvido. O Algoritmo 2 apresenta os passos para alcançar esta meta.

Algoritmo 2: Detecção de conversações fora do contexto

Considerando:
estudante: estudante resolvendo um mesmo problema na PBL
ferramenta_colaborativa: ferramentas colaborativas disponíveis no ambiente
mensagem: mensagem enviada por um estudante nas ferramentas colaborativas
palavra: uma palavra em uma mensagem
PR: palavras relacionadas ao contexto extraídas da ontologia do problema
QPR: quantidade de palavras relacionadas ao contexto
PNR: palavras não relacionadas ao contexto extraídas da ontologia do problema
QPNR: quantidade de palavras não relacionadas ao contexto
Δ: fator de balanceamento usado pelo facilitador
NF: nível de fuga do contexto (guarda o resultado do algoritmo)

```
1:  para todo estudante faça
2:      para todo mensagem ∈ ferramenta_colaborativa faça
3:          para todo palavra ∈ mensagem faça
4:              se palavra ∈ PR então
5:                  QPR = QPR + 1
6:              senão
7:                  se palavra ∈ PNR então
8:                      QPNR = QPNR + 1
9:                  fim se
10:             fim se
11:         fim para
12:     fim para
13:     NF = QPNR / (QPR - Δ)
14:     retorne NF
15: fim para
```

Como pode ser visto no Algoritmo 2, o AgDP monitora as ferramentas usadas pelos estudantes para cooperação e comunicação durante a resolução do problema. Então, o AgDP compara as palavras usadas pelos estudantes nas suas interações com um conjunto de palavras previamente instanciadas na ontologia do problema.

Em seguida, o AgDP calcula a porcentagem de palavras fora do contexto do problema usadas pelo estudante nas ferramentas disponíveis no ambiente. Isso é útil para identificar o nível de fuga do contexto do estudante em relação aos assuntos relacionados ao problema em discussão. Este nível pode ser obtido pela expressão $NF = PNR / (PR - \Delta)$, onde NF = nível de fuga; PNR = quantidade de palavras não relacionadas; PR = quantidade de palavras relacionadas e Δ é um fator que o facilitador pode gerenciar para aumentar ou diminuir o impacto de palavras não relacionadas.

Uma vez detectada uma conversação fora do contexto, caso o *NF* possua um valor superior ao definido pelo facilitador, o AgDP envia uma mensagem automaticamente para o facilitador, via e-mail, notificando todas as informações inerentes aos estudantes dispersos. Em seguida, todas as informações relacionadas aos estudantes dispersos são enviadas para o AgPA.

6.3.4 Agente Monitorador de Grupos – AgMG

O AgMG é responsável pelo monitoramento da criação de novos perfis dos grupos. Ele é responsável por solicitar, ao AgGG, a criação automática dos grupos.

6.3.5 Agente Gerenciador de Grupos – AgGG

O processo de formação de grupos é efetuado da seguinte forma: os estudantes preenchem seus respectivos perfis, via interface do Moodle, que alimentam uma base de perfis que será usada no processo de formação de grupos. O perfil dos estudantes é composto por habilidades, conhecimentos e deficiências, onde cada um possui um nível, que pode ser baixo, médio ou alto, podendo um estudante ter uma ou mais habilidades, deficiências e conhecimentos.

Por outro lado, o facilitador preenche os perfis dos grupos desejados para cada problema a ser solucionado através de uma interface Web, da mesma forma que o estudante. Um perfil de grupo é composto por habilidades, conhecimentos e deficiências, cada um possuindo um nível, que pode ser baixo, médio ou alto, bem como um valor *fuzzy*, que varia de 0.1 a 1.0 e está vinculado aos valores baixo, médio e alto. Após o facilitador construir os perfis dos grupos, haverá uma base de perfis de grupos que será consultada pelo AgGG no processo de formação de grupos. É importante salientar que o perfil desejado construído pelo facilitador é o que melhor se adéqua à resolução do problema; assim, um estudante que tenha um perfil aproximado ao desejado terá as competências necessárias à resolução do problema proposto.

O AgGG é o responsável pela formação automática dos grupos e possui dois comportamentos principais, um implementado em Java (VERONESE *et al.*, 2002) e outro em Prolog (GOMES *et al.*, 2002). O comportamento do AgGG implementado em Java é responsável pela geração de candidatos que estão aptos a participar de determinado grupo. Esse processo é feito analisando os perfis dos estudantes e os perfis dos grupos. Após essa análise, ele gera um arquivo, que será o arquivo de entrada para o comportamento do AgGG implementado em Prolog.

O comportamento do AgGG em Prolog é responsável pela alocação dos estudantes aos grupos propriamente dita. No final desse processo é gerado um arquivo que contém os *rankings* para formação de grupos. O facilitador analisará este resultado, que é exibido através de uma interface do Moodle, e decidirá se acata ou não a sugestão do AgGG. O AgGG é acionado a cada novo perfil do grupo criado. O código fonte do comportamento do AgGG em Prolog pode ser visto no Apêndice A.

6.3.6 Agente Recomendador – AgR

O AgR tem o intuito de detectar OAs adequados ao contexto do estudante, levando em consideração (i) as informações providas pelos seus respectivos perfis, (ii) os assuntos que os estudantes desconhecem e (iii) as informações dos OAs disponíveis no repositório de OAs SCORM.

O AgR utiliza um AG para identificar os OAs a serem recomendados. O AG considera para recomendação de OAs (i) a incidência das palavras-chave na descrição do curso do qual o estudante está participando, (ii) a incidência das áreas de interesse do estudante nas palavras-chave do OA, (iii) o horário de estudo preferido do estudante e (iv) a incidência dos conceitos desconhecidos na descrição, no título e nas palavras-chave do OA, conforme descrito na Subseção 6.3.6.1. Tal recomendação poderá auxiliar os estudantes a sanar as possíveis dúvidas durante a fase de identificação dos fatos.

A utilização do AG é justificada pela complexidade do problema de recomendação, o que pode depreciar o desempenho do AgR quando o número de OAs for suficientemente grande. Portanto, a aplicação de um algoritmo exato seria inviável. Neste caso, um algoritmo aproximativo (AG) viabiliza a busca por uma solução próxima da ótima.

6.3.6.1 Aspectos Considerados no Algoritmo Genético

Inicialmente, temos uma população composta de certa quantidade de OAs. Cada OA possui uma série de características relacionadas a ele. Para o problema do presente trabalho, foram consideradas as características detalhadas a seguir.

Incidência das Palavras-Chave:

Os OAs disponibilizados no padrão SCORM possuem metadados que informam as palavras-chave relacionadas ao assunto daquele OA (*tag keywords*), além do título deste. Em posse dessas informações, é feita uma verificação da incidência dessas palavras na descrição do curso do qual o estudante está participando. Quanto maior a quantidade de palavras-chave relacionadas, a probabilidade de que o OA possua assuntos relacionados ao curso será maior, sendo pontuado, portanto, com um peso maior (no máximo 5). Por outro lado, quanto menor a quantidade de palavras-chave relacionadas, o peso atribuído ao OA nessa variável torna-se proporcionalmente menor.

Áreas de Interesse:

As áreas de interesse são fornecidas pelo próprio estudante, durante o preenchimento de seu perfil (a Subseção 6.3.9.2 traz mais detalhes sobre esse assunto). Em posse dessas informações, é feita uma verificação da incidência dessas áreas de interesse nas palavras-chave do OA. Quanto maior a quantidade de áreas de interesse relacionadas, a probabilidade de que o OA possua assuntos de interesse do estudante, sendo pontuado, portanto, com um peso maior (no máximo 5). Por outro lado, quanto menor a quantidade de áreas de interesse relacionadas, o peso atribuído ao OA nessa variável torna-se proporcionalmente menor.

Horário de Estudo:

Alguns estudantes possuem hábitos de estudar em determinados horários, seja por vontade própria ou por restrições de disponibilidade de horários. Dessa forma, é interessante definir se o horário capturado dinamicamente influencia positiva ou negativamente a recomendação de OAs.

Os horários de estudo são definidos em faixa de horários. O estudante define, inicialmente, qual o horário de estudo preferido dele (a Subseção 6.3.9.2 traz mais detalhes sobre esse assunto). Assim, quando o estudante acessar o ambiente de aprendizagem, o sistema se encarregará de verificar se o horário corrente está incluso na faixa de estudo preferida do estudante. Caso esteja nessa faixa, o sistema atribui o maior valor possível a essa variável (valor 5). Do contrário, será atribuído um valor cada vez menor, sendo o mínimo 1, à medida que o horário corrente se distancie daquele definido como preferido pelo estudante.

Incidência dos Conceitos Desconhecidos:

Os conceitos desconhecidos são fornecidos pelo próprio estudante, após a análise do problema (a Subseção 6.3.9.2 traz mais detalhes sobre esse assunto). Esses conceitos desconhecidos são termos que os estudantes desconhecem na definição do problema. Essa ação caracteriza a segunda fase da PBL, chamada de identificação dos fatos.

Em posse dessas informações, é feita uma verificação da incidência desses conceitos desconhecidos na descrição, no título e nas palavras-chave do OA. Quanto maior a quantidade de conceitos desconhecidos relacionados, maior a probabilidade de que o OA possua assuntos que auxiliarão os estudantes a sanar suas dúvidas, sendo pontuado, portanto, com um peso maior (no máximo 5). Por outro lado, quanto menor a quantidade de conceitos desconhecidos relacionados, o peso atribuído ao OA nessa variável torna-se proporcionalmente menor.

6.3.6.2 Aspectos de Codificação do Algoritmo Genético

Para conseguir uma melhor compreensão do AG proposto, é necessário entender alguns aspectos mais detalhados relacionados à sua codificação. A seguir são discutidos alguns aspectos importantes relacionados a esse assunto.

Representação Cromossomial:

O AG é alimentado de acordo com o resultado alcançado avaliando-se os aspectos descritos na Subseção 6.3.6.1. Para atender a essa abordagem, cada cromossomo (ou indivíduo da população) é composto conforme o exemplo de cromossomo mostrado na Figura 9.

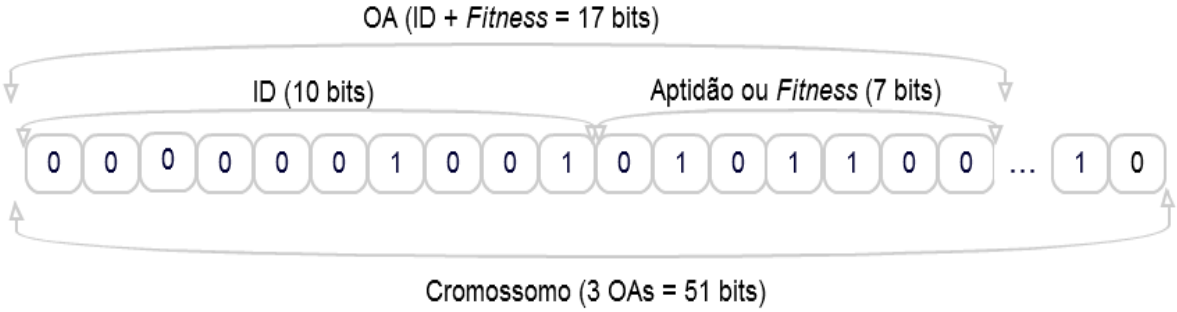

Figura 9 – Representação cromossomial utilizada
Fonte: Dados produzidos pelo autor

A população é composta de cromossomos que possuem, cada um, cinquenta e um bits em representação binária (0 ou 1). Cada OA é representado por dezessete bits, sendo dez utilizados para a sua identificação (ID) e sete para a representação do resultado de sua aptidão, considerando os aspectos relatados na Subseção 6.3.6.1. Assim, com cada OA utilizando dezessete bits, e com uma representação de cromossomo de cinquenta e um bits, é possível representar até três OAs por cromossomo (17 x 3 = 51 bits).

A utilização da representação em codificação binária foi escolhida, pois percebeu-se que, em termos computacionais, essa representação utilizava melhor os recursos de processamento, visto que na execução do AG são criadas várias gerações.

Com o ID do OA sendo representado por dez bits, é possível endereçar todo um repositório contendo até mil e vinte e quatro OAs ($2^{10} = 1024$). Isso permite que o repositório de OAs possa crescer sem causar problemas no AG. O ID do OA é capturado de forma automática por classes auxiliares do AG. Essa captura automatizada é possível porque, cada vez que é inserido um novo OA através da interface do Moodle, é criado um diretório no sistema de arquivos para abrigar os arquivos do OA. O nome do diretório é exatamente o mesmo do campo *id* (chave primária) utilizado na tabela *mdl_scorm* do banco de dados do Moodle. Assim, é possível percorrer os arquivos de determinado OA no sistema de arquivos e, no momento da recomendação resultante do AG, saber qual OA está sendo recomendado.

Outra forma de representar o ID do OA seria capturar a quantidade de OAs no repositório em tempo de execução e, a partir dessa informação, definir quantos bits seriam necessários para representação do OA. Porém, essa forma de representação mostrou-se com um custo de processamento muito elevado, pois, mesmo antes da execução do AG, seria necessário percorrer todos os arquivos do repositório apenas para definir o tamanho do ID do OA.

A aptidão de cada OA é representada por sete bits, podendo resultar em um valor de até cento e vinte e oito na representação decimal ($2^7 = 128$). Quando da execução do AG, é considerado como aptidão do cromossomo a soma das aptidões dos três OAs do cromossomo. Ou seja, para efeito da escolha do indivíduo com maior possibilidade de reprodução (maior aptidão), é considerada a aptidão do cromossomo como um todo.

Essa abordagem de um cromossomo contendo três OAs foi utilizada porque, no final da execução do AG, será indicado um cromossomo como o indivíduo mais apto, ou seja, como a melhor solução levando-se em consideração o contexto em questão. Portanto, não seria interessante, recomendar apenas um OA, mas sim um conjunto dentre o qual o estudante pudesse escolher aquele que lhe fosse mais conveniente. Dessa forma, o estudante tem a opção de escolher, dentre os OAs recomendados, aquele que lhe pareça realmente útil.

Tamanho da População:

O tamanho da população utilizada depende da quantidade de OAs no repositório. A quantidade de cromossomos será correspondente à quantidade de OAs dividida por três.

Para a composição da população durante os testes, foram utilizados tanto OAs no padrão disponíveis em repositórios de OAs encontrados na Internet, como também OAs que possuíam apenas metadados. Nesse último caso, eram construídos manualmente apenas os metadados relativos ao conteúdo fictício dos OAs, como se possuíssem o conteúdo propriamente dito. Essa foi uma alternativa para realização de testes com uma maior quantidade de OAs, e foi possível devido à utilização do padrão SCORM.

Realização da Mutação:

Quando ocorre uma mutação, muta-se uma característica do cromossomo em questão. No caso do AG implementado, uma característica representa o próprio OA em si. Podem ocorrer duas situações distintas quando da mutação: caso a quantidade de OAs no repositório possua uma divisão exata por três e caso a divisão não seja exata.

No caso da divisão ser exata, haverá um número *x* de cromossomos e cada um deles conterá exatamente três OAs. Nesse caso, a mutação ocorre trocando-se, de forma aleatória, um OA em um cromossomo por outro OA que já pertença a outro cromossomo. Para tanto, deve-se sortear: (i) a posição do cromossomo que sofrerá a mutação, ou seja, em que posição esta ocorrerá; (ii) qual o cromossomo do qual será retirado o OA; e (iii) qual a posição do OA neste mesmo cromossomo.

Por outro lado, caso não seja uma divisão exata, haverá uma quantidade *x* de cromossomos completos, mais um ou dois OAs (não pode ser três, pois a divisão seria exata e entraria no caso citado no parágrafo anterior) em "reserva", ou seja, OAs que não pertencem a nenhum cromossomo. Nesse caso, quando da ocorrência da mutação, será escolhida uma posição aleatória do cromossomo que sofrerá a mutação e o OA pertencente a essa posição será trocado por um daqueles da reserva. Dessa forma, o OA que está em reserva terá a chance de participar do AG, podendo melhorar ou piorar a aptidão de determinado cromossomo.

Função de Aptidão:

Uma vez gerada a população inicial, deve ser utilizada uma função de aptidão, que consiste em uma maneira através da qual os AGs determinam a qualidade de um indivíduo como solução do problema em questão, ou seja, ela é praticamente um modo de avaliar o quão distante um determinado indivíduo está da solução ótima (LINDEN, 2008; PETROLI NETO, 2011). A função de aptidão usa todos os parâmetros armazenados no cromossomo e calcula, então, um valor numérico que representa uma métrica da qualidade da solução obtida utilizando-se aqueles parâmetros (LINDEN, 2008). É através desses valores que são selecionados os pais para reprodução (SILVA, 2012).

No AG tratado no presente trabalho, cada cromossomo é formado por um conjunto de três OAs. A representação do OA (gene) é formada pelo ID e sua respectiva aptidão. No presente trabalho, o cálculo da aptidão do OA consiste no somatório dos valores atribuídos a cada característica citada anteriormente na Subseção 6.3.6.1. Ou seja, a função de aptidão, desenvolvida para este caso, busca retornar o maior valor possível, identificando assim o indivíduo com maior probabilidade de ser o mais adequado para a recomendação. Já a aptidão do indivíduo é o somatório das aptidões dos OAs que compõem os cromossomos.

6.3.7 *Directory Facilitator (DF)*

O papel de mediador da comunicação entre os agentes é realizado pelo agente DF, o qual é provido pela própria plataforma JADE, conforme exigência da especificação FIPA (FIPA, 2011). Na arquitetura multiagente proposta, foi necessário apenas codificar a forma como os agentes criados se comunicam com o DF.

6.3.8 Modelagem do SMA

Seguindo os passos da metodologia MAS-CommonKADS+, essa subseção apresenta a modelagem do SMA desenvolvido neste trabalho. Nem todos os diagramas dos respectivos modelos são necessários para a compreensão da arquitetura multiagente proposta, visto que alguns deles acabam fornecendo informações que se tornam redundantes. Logo, decidiu-se representar cada modelo através de um diagrama, contribuindo assim para a compreensão do SMA. A escolha dos modelos a serem inseridos na modelagem depende do tamanho e da complexidade do SMA sendo modelado (SILVA, 2012).

Baseando-se nas necessidades da arquitetura proposta, o SMA deve realizar uma série de tarefas, as quais podem ser observadas no Modelo de Tarefas exibido na Figura 10.

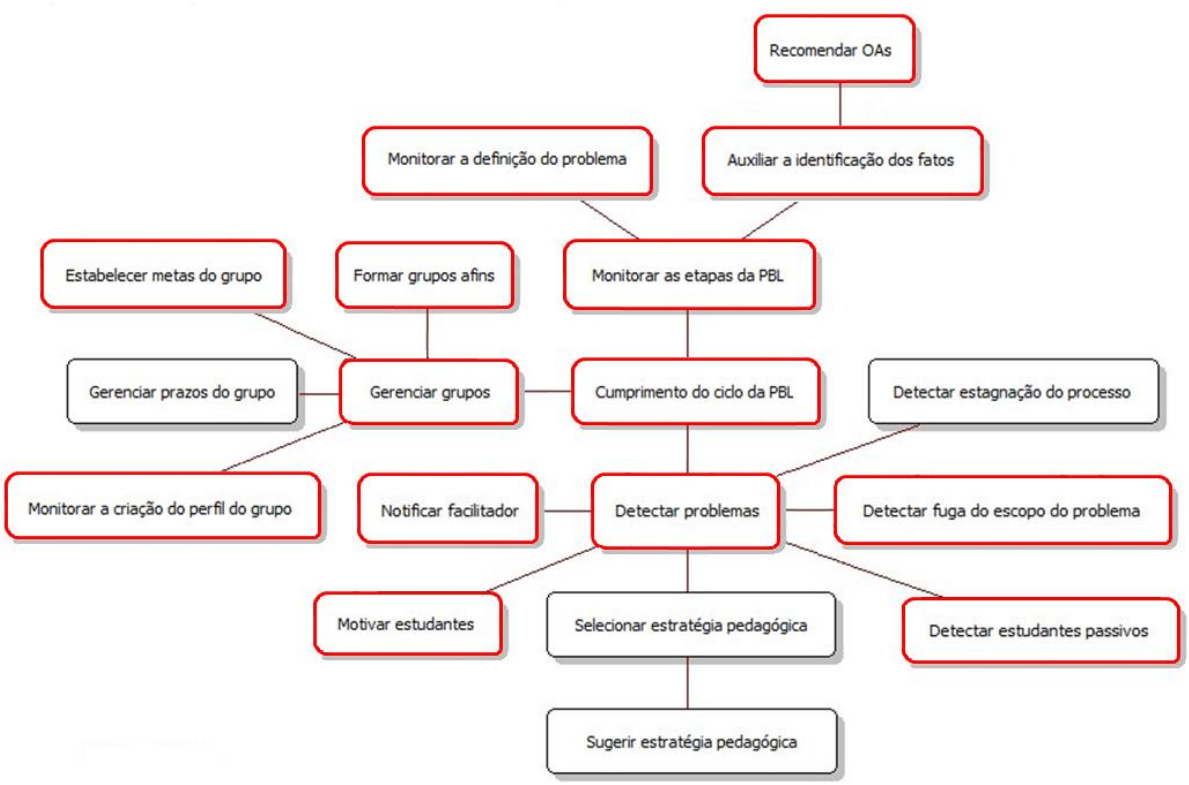

Figura 10 – Modelo de Tarefas
Fonte: Dados produzidos pelo autor

As tarefas destacadas em vermelho foram contempladas neste trabalho. A tarefa principal desse SMA é o cumprimento do ciclo da PBL. Essa tarefa, por sua vez, foi

decomposta em outras subtarefas, que têm como intuito final auxiliar no atendimento da tarefa principal. Seguindo os passos propostos para a metodologia, deve-se realizar a captura dos objetos e recursos do sistema (MORAIS II, 2010). O diagrama de classe encontrado para esse modelo está ilustrado na Figura 11.

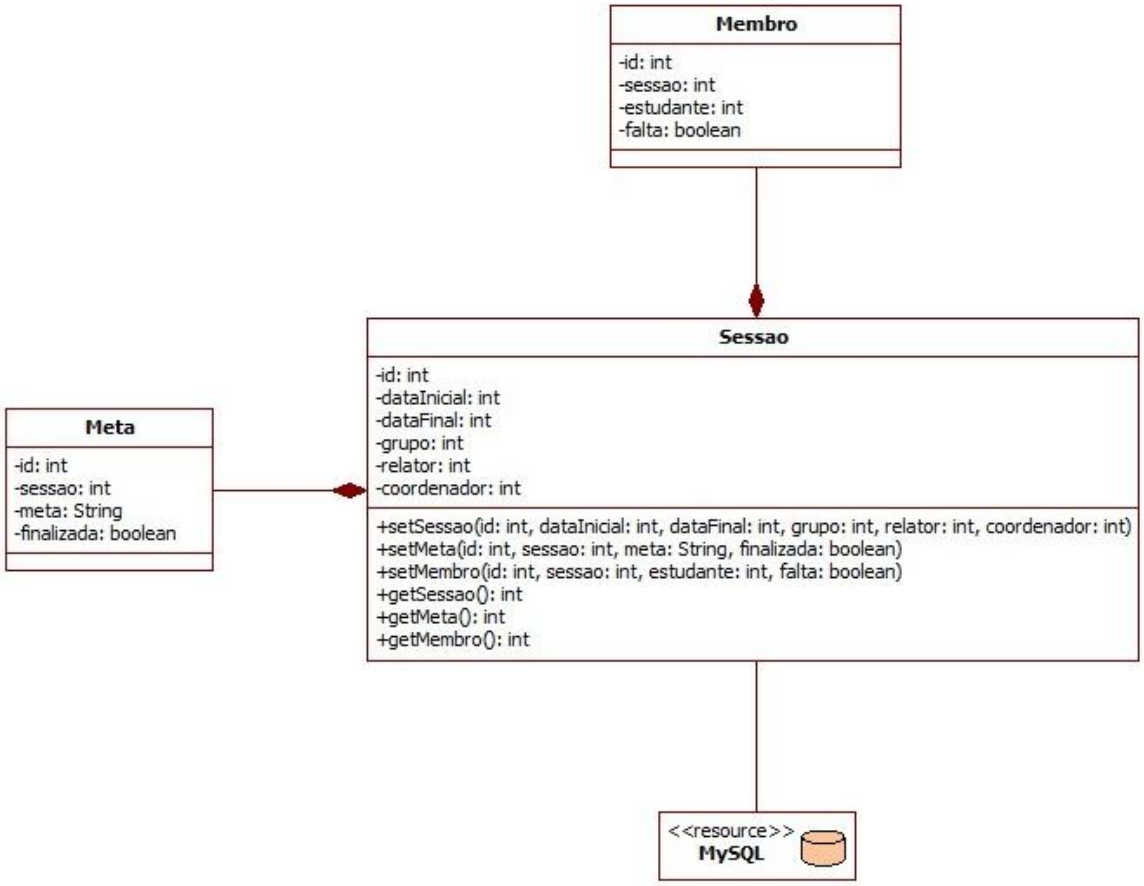

Figura 11 – Modelo de Recursos e Objetos
Fonte: Dados produzidos pelo autor

O próximo passo consiste em definir os papéis que cada agente irá realizar no sistema. Vale ressaltar que um único agente pode ser responsável por vários papéis em um SMA. Cada papel pode realizar várias das tarefas exibidas em um Modelo de Tarefas. O Modelo de Papéis serve para identificar quais papéis irão realizar quais tarefas. Baseando-se nas tarefas elencadas no Modelo de Tarefas da Figura 10, foi definido o Modelo de Papéis apresentado na Figura 12.

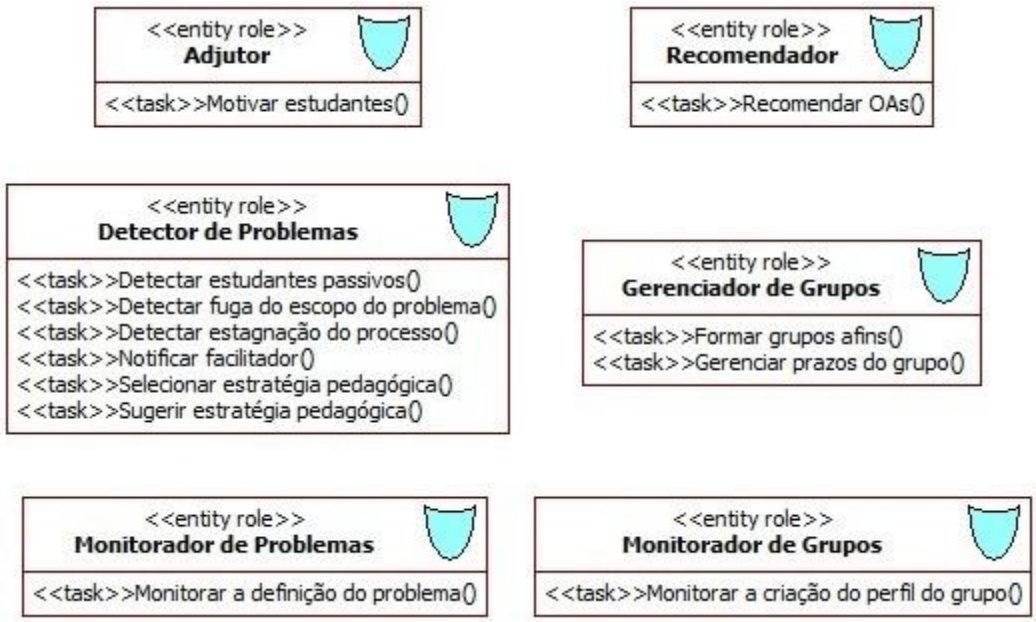

Figura 12 – Modelo de Papéis
Fonte: Dados produzidos pelo autor

Na Figura 12, é possível perceber que foram definidos seis tipos de papéis para atender às tarefas inerentes ao Modelo de Tarefas: adjutor, recomendador, detector de problemas, gerenciador de grupos, monitorador de problemas e monitorador de grupos. Apesar de ser possível que cada agente possa realizar mais de um papel, decidiu-se que, no SMA proposto, cada um desses papéis será realizado por um agente. Esses agentes são, respectivamente, o Agente Pedagógico Animado - AgPA, o Agente Recomendador - AgR, o Agente Detector de Problemas - AgDP, o Agente Gerenciador de Grupos - AgGG, o Agente Monitorador de Problemas - AgMP e o Agente Monitorador de Grupos - AgMG. Desta forma, a partir de agora, os papéis não serão mais citados, mas sim os agentes que realizam esses papéis. Esses agentes serão detalhados no Modelo de Agentes, em passos posteriores da metodologia. Vale ressaltar que as tarefas *detectar estagnação do processo*, *selecionar estratégia pedagógica* e *sugerir estratégia pedagógica*, do papel detector de problemas, e *gerenciar prazos do grupo*, referente ao papel gerenciador de grupos, não foram contempladas neste trabalho.

Uma vez construído o Modelo de Papéis, deve ser desenvolvido o Modelo de Organização, que serve para descrever a estrutura organizacional dos papéis no sistema, ou seja, como os papéis estão relacionados entre si. A Figura 13 mostra o Modelo de Organização correspondente aos papéis exibidos na Figura 12.

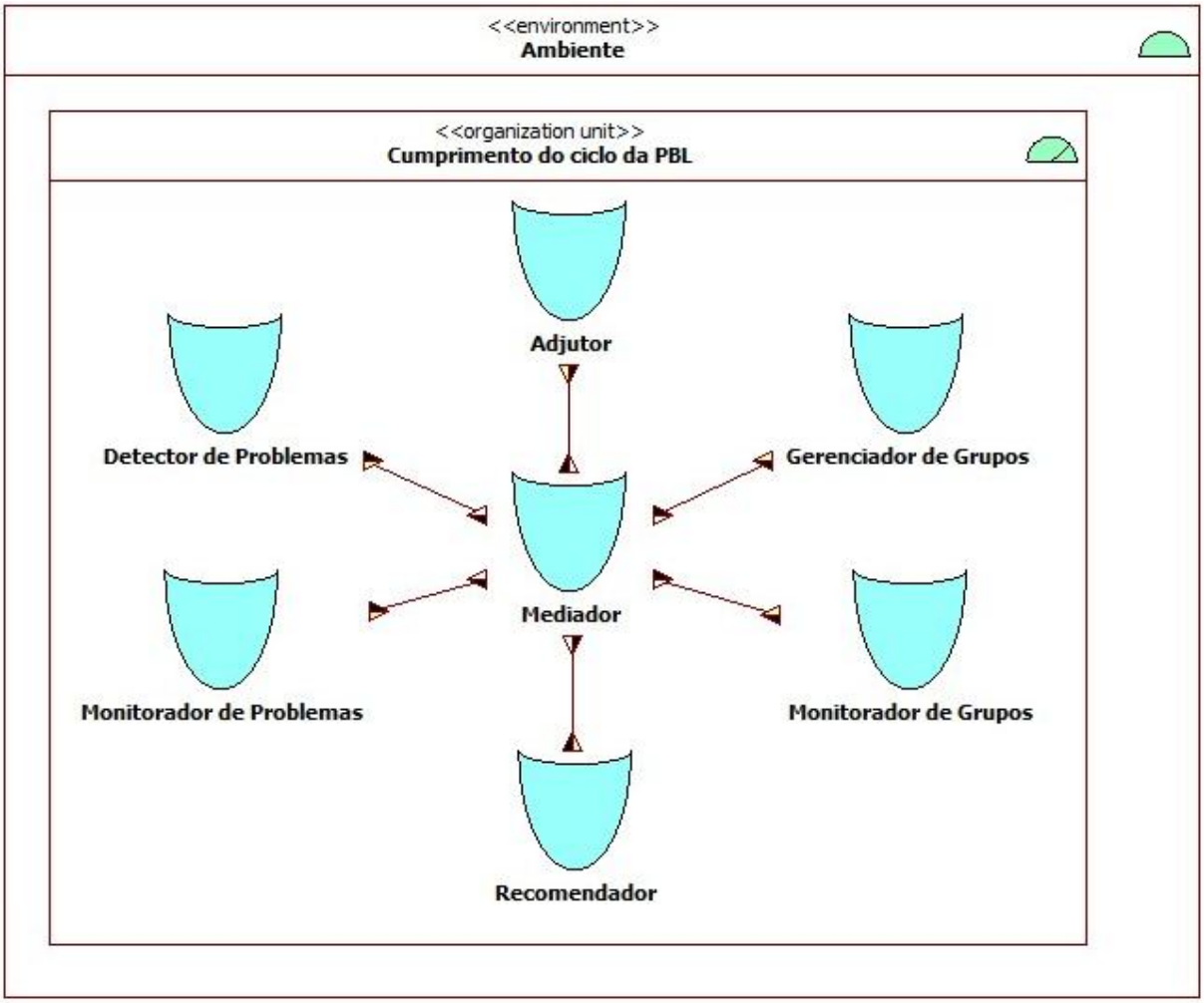

Figura 13 – Modelo de Organização
Fonte: Dados produzidos pelo autor

Nesse modelo é possível perceber a presença de um outro papel, o mediador. O mediador permitirá que os agentes se comuniquem sem que, inicialmente, eles se conheçam. O motivo dele não fazer parte do Modelo de Papéis é que o mesmo não realiza tarefas que compõem a finalidade principal do SMA, ou seja, não realiza uma das tarefas que compõem o Modelo de Tarefas. Mesmo assim, essa tarefa é de extrema relevância, pois é através dela que os agentes conseguem informações a respeito dos serviços de outros agentes e conseguem se comunicar uns com os outros. Ou seja, ele funciona como um intermediador, permitindo que agentes cadastrem informações e pesquisem por informações de outros agentes. Conforme já mencionado, o papel do mediador da comunicação entre os agentes é realizado pelo agente DF. O funcionamento desse agente será melhor explicado no Modelo de Agentes.

O Modelo de Organização apresenta a organização interna e externa dos papéis. A primeira apresenta os relacionamentos entre os papéis do SMA, enquanto que a segunda demonstra os relacionamentos entre as organizações em si. O SMA deste trabalho possui apenas uma unidade organizacional (*organization unit*), logo não existe uma representação externa. Como pode ser visto no Modelo de Organização da Figura 13, na representação interna da unidade "Cumprimento do ciclo da PBL", os sete papéis estão relacionados através de um relacionamento do tipo *peer*, que é usado para papéis com a mesma autoridade e é representado por uma seta parcialmente cheia (SILVA, 2012).

Uma vez sabendo como os diferentes papéis em um SMA estão organizados entre si, é necessário definir quais agentes serão responsáveis por cada papel. É necessário também definir a arquitetura dos agentes, seus objetivos e suas características, como entrada de dados, condições de ativação do agente e tipos de informação disponíveis. Para uma melhor visualização das informações, os *templates* textuais, com os objetivos e características dos agentes, serão descritos em tabelas em subseções específicas para cada agente. Nessas subseções, serão fornecidas informações da modelagem, assim como informações da codificação relativa ao que foi modelado.

Na fase inicial da modelagem, é necessário que se tenha, no mínimo, uma noção estática do SMA, o que pode ser alcançado com o Modelo de Organização. Porém, é interessante também ter uma visão do comportamento dinâmico do SMA, o que é possível conseguir através do Modelo de Interação (SILVA, 2012). A Figura 14 mostra o modelo de interação entre os agentes AgMP, AgDP, DF e AgPA.

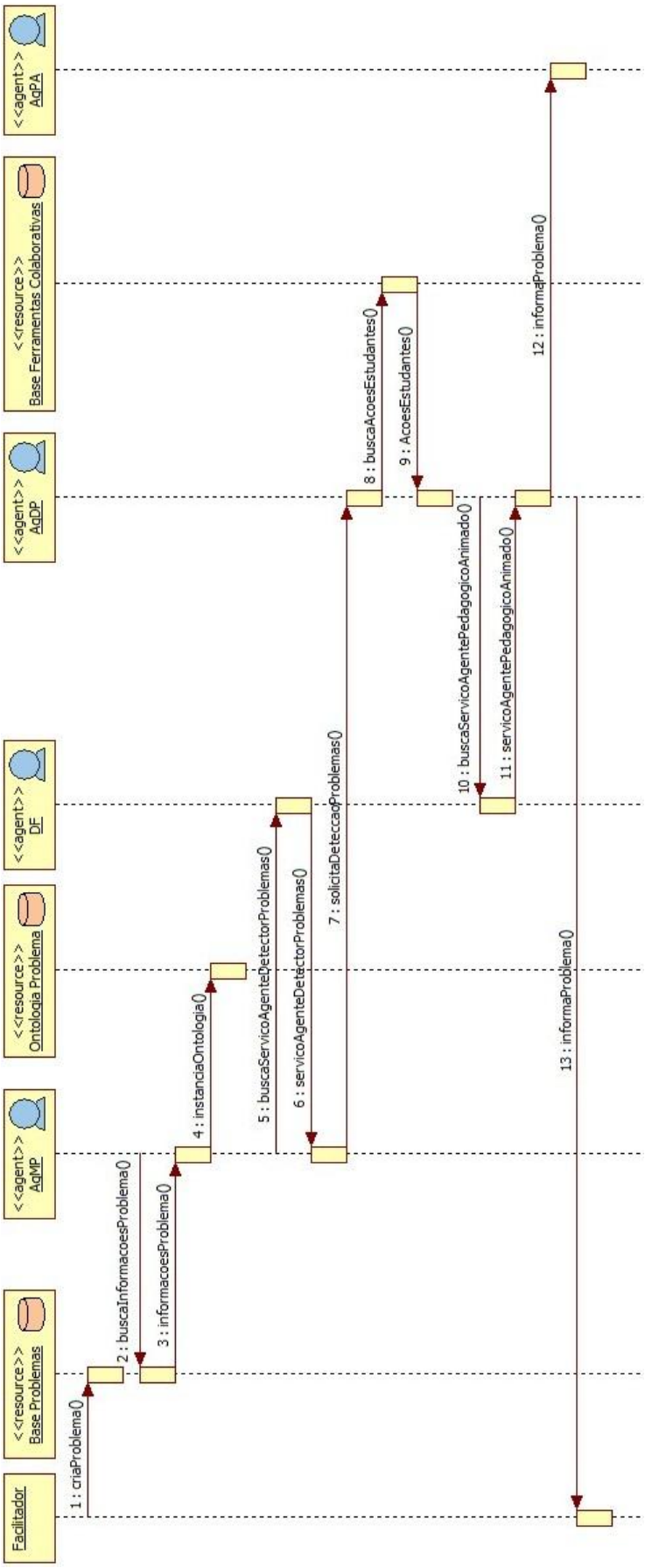

Figura 14 – Modelo de interação entre os agentes AgMP, AgDP, DF e AgPA
Fonte: Dados produzidos pelo autor

No modelo apresentado na Figura 14, é possível visualizar toda a interação que ocorre entre os agentes AgMP, AgDP, DF e AgPA e os recursos que são consultados por tais agentes. O diagrama mostra desde o comportamento do AgMP no momento que o facilitador cria um novo problema até a informação chegar no AgPA, que irá tentar motivar os estudantes envolvidos no problema detectado, e o próprio facilitador, que poderá tomar uma providência cabível. É possível perceber ainda que toda a interação ocorre com o auxílio do agente DF, através do seu serviço de páginas amarelas.

Também é possível perceber que as informações dos problemas criados pelo facilitador são instanciadas em ontologias. Outra forma possível de se armazenar essas informações seria com a utilização de um banco de dados. Porém, o uso de uma ontologia para representação desse conhecimento mostra-se como uma alternativa mais flexível, visto que esse conhecimento pode ser utilizado por outras partes da aplicação e, caso seja necessário modificar o comportamento desta, pode-se criar uma nova forma de representar a ontologia, sem modificar a codificação do sistema em si. Diante disso, está sendo utilizada uma ontologia genérica do problema, sendo instanciada uma ontologia para cada novo problema criado pelo facilitador.

Na Figura 15, é possível observar o modelo de interação entre os agentes AgMG, AgGG e DF.

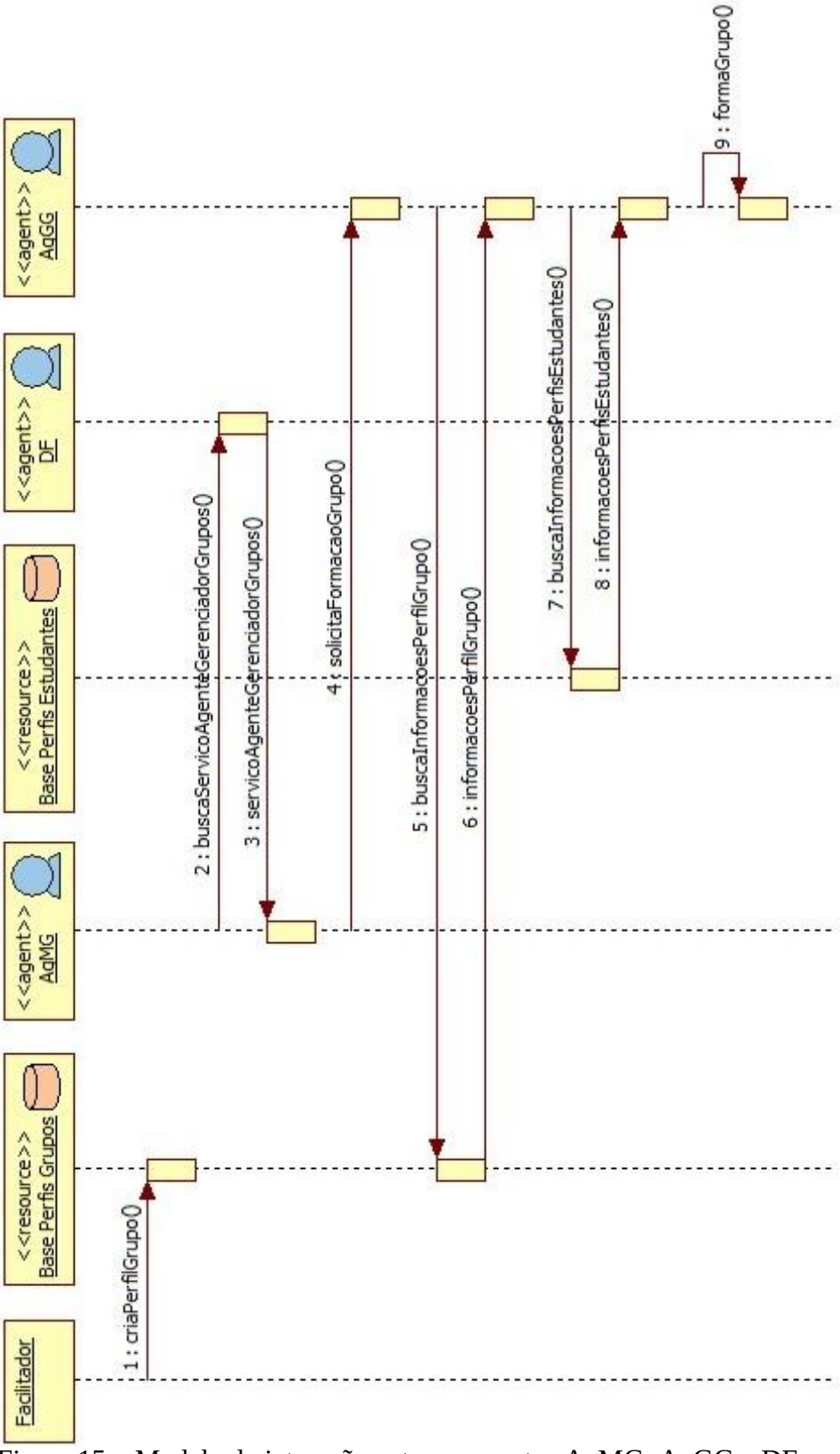

Figura 15 – Modelo de interação entre os agentes AgMG, AgGG e DF
Fonte: Dados produzidos pelo autor

No modelo apresentado na Figura 15, é possível visualizar toda a interação que ocorre entre os agentes AgMG, AgGG e DF e os recursos que são consultados por tais agentes. O diagrama mostra desde o comportamento do AgMG no momento que o facilitador cria um novo perfil do grupo até a informação chegar no AgGG, que irá realizar a formação do grupo propriamente dita.

Percebe-se que o AgR não participa das interações apresentadas anteriormente. De fato, o AgR não se comunica com nenhum dos agentes presentes nessa arquitetura. Quando acionado, o AgR executa suas tarefas sem precisar de nenhum serviço oferecido pelos outros agentes. No entanto, ele faz a recomendação com base, dentre outras coisas, nas informações obtidas nas bases de dados, que são alimentadas, na maioria das vezes, pelos demais agentes.

6.3.8.1 Modelo do Agente Pedagógico Animado – AgPA

O AgPA realiza o papel "Adjutor". A Tabela 4 apresenta o seu *template* textual.

Tabela 4 – *Template* textual do AgPA

Agente: AgPA
Objetivos Disponibilizar um serviço do tipo "*agente-pedagogico-animado*" no DF de modo que o AgDP possa encontrar esse serviço para se comunicar com ele. Uma vez em posse das informações enviadas pelo AgDP, o AgPA tenta motivar os estudantes, exibindo animações com expressões de emoções e mensagens textuais condizentes com o problema de colaboração detectado.
Parâmetros de Entrada Informações sobre os estudantes envolvidos no problema detectado enviadas pelo AgDP.
Parâmetros de Saída Animações com expressões de emoções e mensagens textuais.
Condição de Ativação Quando o agente percebe que um estudante se autenticou no ambiente.
Condição de Finalização Quando o agente percebe que o estudante se desconectou do ambiente.
Informação Associada Possui um comportamento que disponibiliza um serviço do tipo "*agente-pedagogico-animado*" no DF de modo que o AgDP possa encontrar esse serviço e se comunicar com ele. Também possui um comportamento cíclico (tipo *Cyclic Behaviour* (BELIFEMINE;

CAIRE; GREENWOOD, 2007)) para ficar monitorando por requisições do AgDP. Uma vez em posse dessas informações, o AgPA tenta motivar os estudantes a participarem mais das discussões e a usarem as ferramentas disponíveis no AVA, exibindo animações com expressões de emoções e mensagens textuais.
Descrição Este agente possui o comportamento voltado para o acompanhamento dos estudantes durante o processo de aplicação da PBL. Ele permanece ativo durante todo o período em que o estudante estiver conectado ao ambiente, motivando-o quando algum problema de colaboração for detectado.

Fonte: Dados produzidos pelo autor

A Figura 16 mostra a parte do código-fonte do AgPA responsável por cadastrar um serviço do tipo "*agente-pedagogico-animado*" no DF.

```
24      @Override
25      protected void setup(){
26          System.out.println("Agente Pedagógico Animado " + getAID().getName() + " executando.");
27
28          //Criando uma entrada no DF
29          DFAgentDescription template = new DFAgentDescription();
30          //Informando a AID do agente
31          template.setName(getAID());
32
33          //Criando um serviço do tipo "agente-pedagogico-animado" no DF
34          ServiceDescription sd = new ServiceDescription();
35          sd.setType("agente-pedagogico-animado"); //Tipo do Serviço
36          sd.setName("agente-pedagogico-animado-moodle"); //Nome do Serviço
37          template.addServices(sd); //Adicionando o Serviço
38
39          //Registrando o agente no DF
40          try{
41              DFService.register(this, template); //register(agente que oferece, descrição)
42          }catch(FIPAException e){
43              System.out.println(e.getACLMessage());
44          }
45
46          addBehaviour(new GetMessage());
47      }
```

Figura 16 – Parte do código-fonte do AgPA para cadastrar serviço no DF
Fonte: Dados produzidos pelo autor

Nesse código, o AgPA cria, primeiramente, uma entrada (*template*), na qual é possível registrar mais de um serviço. Em seguida, adiciona um serviço (*sd*) a essa entrada. Por fim, registra essa entrada no DF. Ao final do cadastro do serviço, ele adiciona um comportamento *GetMessage*. Esse é um comportamento cíclico responsável por receber do AgDP as informações dos estudantes envolvidos em possíveis problemas de colaboração. Nesse comportamento, o AgPA procura continuamente em sua pilha por uma mensagem que utilize

o identificador de conversação *estudantes-passivos* ou *conversacao-fora-contexto*. A Figura 17 mostra um trecho de código desse comportamento.

```
@Override
public void action(){

    jade.lang.acl.ACLMessage msg = myAgent.receive();

    if(msg != null){

        if(msg.getConversationId().equals("estudantes-passivos")){
            try {
                list = (ArrayList<Student>) msg.getContentObject();
            } catch (UnreadableException ex) {
                Logger.getLogger(AgPA.class.getName()).log(Level.SEVERE, null, ex);
            }
            System.out.println("O agente "+msg.getSender().getLocalName()+
                    " detectou estudantes com comportamento passivo.");
            setMessageBD(list, "dep");
        }else if(msg.getConversationId().equals("conversacao-fora-contexto")){
            try {
                list = (ArrayList<Student>) msg.getContentObject();
            } catch (UnreadableException ex) {
                Logger.getLogger(AgPA.class.getName()).log(Level.SEVERE, null, ex);
            }
            System.out.println("O agente "+msg.getSender().getLocalName()+
                    " detectou conversas fora do contexto do problema."),
            setMessageBD(list, "dcfc");
        }else{
            block();
        }
    }
}
```

Figura 17 – Trecho do código-fonte do AgPA para tratar requisições do AgDP
Fonte: Dados produzidos pelo autor

Nesse comportamento, o AgPA insere no banco de dados, através do método *setMessageBD*, as informações dos estudantes envolvidos no problema detectado, bem como as informações sobre qual animação ele deve executar e sobre a mensagem que ele deve exibir para o estudante.

6.3.8.2 Modelo do Agente Monitorador de Problemas – AgMP

O AgMP realiza o papel “Monitorador de Problemas”. A Tabela 5 apresenta o seu *template* textual.

Tabela 5 – *Template* textual do AgMP

Agente: AgMP
Objetivos Monitorar a criação de um novo problema, capturando o identificador do problema; instanciar uma ontologia com as informações do novo problema; e enviar uma mensagem para o AgDP, acionando os seus comportamentos de detecção de problemas.
Parâmetros de Entrada Identificador do novo problema.
Parâmetros de Saída Instância da ontologia do problema.
Condição de Ativação Quando o agente percebe que o facilitador criou um novo problema.
Condição de Finalização Quando o agente cria uma instância da ontologia do problema e se comunica com o AgDP, acionando os seus comportamentos de detecção de problemas.
Informação Associada Possui dois comportamentos de execução única (tipo *One Shot Behaviour* (BELIFEMINE, CAIRE e GREENWOOD, 2007)). O primeiro tem o objetivo de instanciar uma ontologia do problema, com todas as informações do problema inseridas pelo facilitador. O segundo tem o objetivo de enviar uma mensagem para o AgDP, com o intuito de acionar os seus comportamentos de detecção de problemas.
Descrição Este agente possui o comportamento voltado para o monitoramento da criação de novos problemas. Ele é responsável por instanciar ontologias para cada novo problema criado e acionar os comportamentos de detecção de problemas do AgDP.

Fonte: Dados produzidos pelo autor

Inicialmente o AgMP aciona o comportamento para instanciar uma ontologia com as informações do novo problema inseridas pelo facilitador. A Figura 18 ilustra o trecho de código desse comportamento.

```
34      private class CreateOntology extends OneShotBehaviour{
35          private String title;
36          private String definition;
37          private String keywords;
38          private String knowledge;
39          private String firstdate;
40          private String lastdate;
41          private String related_words;
42          private String not_related_words;
43          private OntModel ontologia;
44
45          public CreateOntology(){...}
57
59          @Override
60          public void action(){
61              String problemid = "";
62              Object[] args = getArguments();
63
64              if(args != null && args.length > 0){
65                  problemid = (String) args[0];
66              }
67              getDatasDB(problemid); //Recupera valores do BD
68              setDatasOntology(); //Insere valores na nova ontologia
69              saveOntology(); //Salva a nova ontologia
70              addBehaviour(new SendMessageToAgDP());
71          }
72
97          public void getDatasDB(String problemid){...}
98
130         public void setDatasOntology(){...}
131
139         public void saveOntology(){...}
        }
```

Figura 18 – Trecho do código-fonte do AgMP para instanciar a ontologia do problema
Fonte: Dados produzidos pelo autor

Conforme ilustrado na Figura 18, o AgMP faz uma chamada ao método *getDatasDB* para recuperar as informações do problema, passando como parâmetro a identificação do problema (linha 66). Uma vez em posse dessas informações, o AgMP faz uma chamada ao método *setDatasOntology* para inserir essas informações na nova ontologia (linha 67). Por fim, o AgMP faz uma chamada ao método *saveOntology* para salvar a nova ontologia (linha 68). Em seguida, o AgMP aciona o comportamento *SendMessageToAgDP*. A Figura 19 apresenta um trecho de código do comportamento *SendMessageToAgDP*.

```
149         @Override
150         public void action() {
151             //Criando uma entrada no DF
152             DFAgentDescription template = new DFAgentDescription();
153             //Criando um objeto contendo a descrição do serviço
154             ServiceDescription sd = new ServiceDescription();
155             //Definindo o tipo de serviço
156             sd.setType("agente-detector-problemas");
157 
158             //Adicionando o serviço na entrada
159             template.addServices(sd);
160             try{
161                 //Realiza busca pelos agentes
162                 //A busca retorna um array DFAgentDescription
163                 //O parâmetro myAgent indica o agente que está realizando a busca
164                 DFAgentDescription[] result = DFService.search(myAgent, template);
165 
166                 //AgMP envia mensagem para o AgDP
167                 for(int i = 0; i < result.length; i++){
168                     ACLMessage message = new ACLMessage(ACLMessage.INFORM);
169                     message.addReceiver(new AID(result[i].getName().getLocalName(), AID.ISLOCALNAME));
170                     message.setConversationId("deteccao-problemas");
171                     send(message);
172                  }
173             } catch (FIPAException ex) {
174                 Logger.getLogger(AgMG.class.getName()).log(Level.SEVERE, null, ex);
175             }
176 
177             //Finalizou suas tarefas, então finaliza a instância do agente
178             doDelete();
179         }
    }
```

Figura 19 – Parte do código-fonte do AgMP para enviar mensagem para o AgDP
Fonte: Dados produzidos pelo autor

Conforme ilustrado na Figura 19, o AgMP cria, inicialmente, um *template* que serve para informar qual tipo de serviço ele está buscando, no caso o serviço do tipo "*agente-detector-problemas*" (linhas 151-158). Ele faz uma busca no DF para verificar se um serviço do tipo citado foi adicionado e, caso exista, a descrição desse serviço é adicionada a um vetor (linha 163). Em seguida, ele envia a mensagem para o AgDP, usando o identificador de conversação *deteccao-problemas* (linhas 166-171).

6.3.8.3 Modelo do Agente Detector de Problemas – AgDP

O AgDP realiza o papel "Detector de Problemas". A Tabela 6 apresenta o seu *template* textual.

Tabela 6 – *Template* textual do AgDP

Agente: AgDP
Objetivos Disponibilizar um serviço do tipo "*agente-detector-problemas*" no DF de modo que o AgMP possa encontrar esse serviço para se comunicar com ele. Uma vez em posse das informações enviadas pelo AgMP, o AgDP aciona os seus dois comportamentos: detecção de estudantes passivos e detecção de conversações fora do contexto do problema. Uma vez que o AgDP tenha detectado um dos dois problemas, este irá notificar o facilitador, via e-mail, informando todas as informações inerentes aos estudantes envolvidos com o problema detectado. Em seguida, todas as informações relativas a esses estudantes, assim como as informações sobre o problema detectado, são enviadas para o AgPA.
Parâmetros de Entrada Informações sobre quais atividades os estudantes estão executando nas ferramentas colaborativas; informações sobre o conteúdo das discussões nas ferramentas colaborativas; e as ontologias dos problemas.
Parâmetros de Saída E-mail com informações sobre o problema detectado e sobre os estudantes envolvidos.
Condição de Ativação Quando o agente recebe do AgMP uma mensagem com identificador de conversação do tipo *deteccao-problemas*.
Condição de Finalização Quando o agente percebe que o ciclo da PBL foi encerrado.
Informação Associada Possui um comportamento que disponibiliza um serviço do tipo "*agente-detector-problemas*" no DF de modo que o AgMP possa encontrar esse serviço e se comunicar com ele. Também possui um comportamento cíclico (tipo *Cyclic Behaviour*) para ficar monitorando por requisições do AgMP. Uma vez em posse dessas informações, o AgDP aciona os seus dois comportamentos: detecção de estudantes passivos e detecção de conversações fora do contexto do problema. Esses dois comportamentos são executados a cada determinado período de tempo (tipo *Ticker Behaviour* (BELIFEMINE, CAIRE e GREENWOOD, 2007)). Uma vez detectado um dos problemas citados anteriormente, o AgDP notifica o facilitador, via e-mail, informando todas as informações inerentes aos estudantes envolvidos com o problema detectado. Ele também envia todas as informações relativas a esses estudantes, assim como as informações sobre o problema detectado para o AgPA.
Descrição Este agente possui o comportamento voltado para o auxílio ao facilitador durante todo o ciclo da PBL. Ele é responsável por detectar problemas de colaboração que podem vir a acontecer durante o processo de aplicação da PBL.

Fonte: Dados produzidos pelo autor

A Figura 20 mostra a parte do código-fonte do AgDP responsável por cadastrar um serviço do tipo "*agente-detector-problemas*" no DF.

```
43      @Override
44      protected void setup(){
45          System.out.println("Agente Detector de Problemas " + getName() + " executando.");
46
47          //Criando uma entrada no DF
48          DFAgentDescription template = new DFAgentDescription();
49          //Informando a AID do agente
50          template.setName(getAID());
51
52          //Criando um serviço de "agente-detector-problemas" no DF
53          ServiceDescription sd = new ServiceDescription();
54          sd.setType("agente-detector-problemas"); //Tipo do Serviço
55          sd.setName("agente-detector-problemas-moodle"); //Nome do Serviço
56          template.addServices(sd); //Adicionando o Serviço
57
58          //Registrando o agente no DF
59          try{
60              DFService.register(this, template); //register(agente que oferece, descrição)
61          }catch(FIPAException e){
62              System.out.println(e.getACLMessage());
63          }
64
65          addBehaviour(new GetMessage());
        }
```

Figura 20 – Parte do código-fonte do AgDP para cadastrar serviço no DF
Fonte: Dados produzidos pelo autor

Similarmente ao código de cadastro do AgPA (ilustrado na Figura 16), no trecho de código do AgDP, ao final do cadastro do serviço, ele adiciona um comportamento *GetMessage*. Esse é um comportamento cíclico para ficar monitorando por requisições do AgMP. Nesse comportamento, o AgDP procura continuamente em sua pilha por uma mensagem que utilize o identificador de conversação *deteccao-problemas*. Uma vez em posse dessas informações, o AgDP aciona os seus dois comportamentos: detecção de estudantes passivos e detecção de conversações fora do contexto do problema. Esses dois comportamentos são executados a cada determinado período de tempo (tipo *Ticker Behaviour*), no caso 86400000 milissegundos ou 24 horas. A Figura 21 mostra um trecho de código do comportamento *GetMessage*.

```
80      private class GetMessage extends CyclicBehaviour{
81
83          @Override
            public void action(){
84
85              jade.lang.acl.ACLMessage msg = myAgent.receive();
86
87              if(msg != null){
88                  if(msg.getConversationId().equals("deteccao-problemas")){
89                      addBehaviour(new DetectaEstudantesPassivos(agdp, 86400000));
90                      addBehaviour(new DetectaConversacoesForaDoContexto(agdp, 86400000));
91                      System.out.println("O agente "+msg.getSender().getLocalName()+
92                              " acionou os comportamentos do agente AgDP.");
93                  }else{
94                      block();
95                  }
96              }
97          }
98      }
99
100     //Classe que detecta estudantes com comportamento passivo
101     private class DetectaEstudantesPassivos extends TickerBehaviour{...}
273
274     //Classe que detecta conversações fora do contexto do problema
275     private class DetectaConversacoesForaDoContexto extends TickerBehaviour{...}
488 }
```

Figura 21 – Trecho do código-fonte do AgDP para tratar requisições do AgMP
Fonte: Dados produzidos pelo autor

Uma vez detectado um dos problemas citados anteriormente, o AgDP notifica o facilitador, via e-mail, informando todas as informações inerentes aos estudantes envolvidos com o problema detectado. Ele também envia todas as informações relativas a esses estudantes, assim como as informações sobre o problema detectado para o AgPA. Caso o AgDP tenha detectado estudantes com comportamentos passivos, ele envia uma mensagem para o AgPA, conforme ilustrado na Figura 22.

```
244         public void sendMessageToAgPA() throws FIPAException{
245             //Criando uma entrada no DF
246             DFAgentDescription template = new DFAgentDescription ( ) ;
247             //Criando um objeto contendo a descrição do serviço
248             ServiceDescription sd = new ServiceDescription();
249             //Definindo o tipo de serviço
250             sd.setType("agente-pedagogico-animado");
251
252             //Adicionando o serviço na entrada
253             template.addServices(sd);
254             try{
255                 //Realiza busca pelos agentes
256                 //A busca retorna um array DFAgentDescription
257                 //O parâmetro myAgent indica o agente que está realizando a busca
258                 DFAgentDescription[] result = DFService.search(myAgent, template);
259
260                 //AgDP envia mensagem para o AgPA informando o problema detectado
261                 for(int i = 0; i < result.length; i++){
262                     ACLMessage message = new ACLMessage(ACLMessage.INFORM);
263                     message.addReceiver(new AID(result[i].getName().getLocalName(), AID.ISLOCALNAME));
264                     message.setContentObject(lista);
265                     message.setConversationId("estudantes-passivos");
266                     send(message);
267                 }
268             } catch (IOException ex) {
269                 Logger.getLogger(AgDP.class.getName()).log(Level.SEVERE, null, ex);
270             }
271         }
272     }
```

Figura 22 – Mensagem de detecção de estudantes passivos enviada para o AgPA
Fonte: Dados produzidos pelo autor

Conforme ilustrado na Figura 22, o AgDP cria, inicialmente, um *template* que serve para informar qual tipo de serviço ele está buscando, no caso o serviço do tipo "*agente-pedagogico-animado*" (linhas 246-253). Ele faz uma busca no DF para verificar se um serviço do tipo citado foi adicionado e, caso exista, a descrição desse serviço é adicionada a um vetor (linha 258). Em seguida, ele envia a mensagem para o AgPA, passando como parâmetro uma lista com todas as informações dos estudantes detectados com o comportamento passivo, usando o identificador de conversação *estudantes-passivos* (linhas 261-267).

Da mesma forma, caso o AgDP tenha detectado conversações fora do contexto do problema, ele envia uma mensagem para o AgPA, conforme ilustrado na Figura 23.

```
459         public void sendMessageToAgPA() throws FIPAException{
460             //Criando uma entrada no DF
461             DFAgentDescription template = new DFAgentDescription ( ) ;
462             //Criando um objeto contendo a descrição do serviço
463             ServiceDescription sd = new ServiceDescription();
464             //Definindo o tipo de serviço
465             sd.setType("agente-pedagogico-animado");
466
467             //Adicionando o serviço na entrada
468             template.addServices(sd);
469             try{
470                 //Realiza busca pelos agentes
471                 //A busca retorna um array DFAgentDescription
472                 //O parâmetro myAgent indica o agente que está realizando a busca
473                 DFAgentDescription[] result = DFService.search(myAgent, template);
474
475                 //AgDP envia mensagem para o AgPA informando o problema detectado
476                 for(int i = 0; i < result.length; i++){
477                     ACLMessage message = new ACLMessage(ACLMessage.INFORM);
478                     message.addReceiver(new AID(result[i].getName().getLocalName(), AID.ISLOCALNAME));
479                     message.setContentObject(lista);
480                     message.setConversationId("conversacao-fora-contexto");
481                     send(message);
482                 }
483             } catch (IOException ex) {
484                 Logger.getLogger(AgDP.class.getName()).log(Level.SEVERE, null, ex);
485             }
486         }
487     }
488 }
```

Figura 23 – Mensagem de detecção de conversação fora do contexto enviada para o AgPA
Fonte: Dados produzidos pelo autor

O código é similar àquele mostrado na Figura 22. A diferença é que, no trecho de código da Figura 23, o AgDP envia a mensagem para o AgPA, passando como parâmetro uma lista com todas as informações dos estudantes que estavam conversando sobre assuntos fora do contexto do problema, usando o identificador de conversação *conversacao-fora-contexto* (linhas 476-482).

6.3.8.4 Modelo do Agente Monitorador de Grupos – AgMG

O AgMG realiza o papel “Monitorador de Grupos”. A Tabela 7 apresenta o seu *template* textual.

Tabela 7 – *Template* textual do AgMG

Agente: AgMG
Objetivos Monitorar a criação de um novo perfil do grupo, capturando o identificador do grupo a ser criado, e enviar uma mensagem para o AgGG, solicitando a criação automática do grupo.
Parâmetros de Entrada Identificador do grupo a ser criado.
Parâmetros de Saída Mensagem enviada para o AgGG, com a solicitação de criação automática do grupo.
Condição de Ativação Quando o agente percebe que o facilitador criou um novo perfil do grupo.
Condição de Finalização Quando o agente se comunica com o AgGG e envia para este a solicitação de criação automática do grupo.
Informação Associada Possui um comportamento de execução única (tipo *One Shot Behaviour*) para enviar, para o AgGG, a solicitação de criação automática do grupo, passando como parâmetro o identificador do grupo a ser criado.
Descrição Este agente possui o comportamento totalmente voltado para o monitoramento da criação de novos perfis dos grupos. Ele é responsável por solicitar, ao AgGG, a criação automática dos grupos.

Fonte: Dados produzidos pelo autor

A Figura 24 mostra o comportamento do AgMG responsável por verificar se existe um serviço do tipo "*agente-gerenciador-grupos*" cadastrado no DF.

```
39              //Criando uma entrada no DF
40              DFAgentDescription template = new DFAgentDescription();
41              //Criando um objeto contendo a descrição do serviço
42              ServiceDescription sd = new ServiceDescription();
43              //Definindo o tipo de serviço
44              sd.setType("agente-gerenciador-grupos");
45
46              //Adicionando o serviço na entrada
47              template.addServices(sd);
48              try{
49                  //Realiza busca pelos agentes
50                  //A busca retorna um array DFAgentDescription
51                  //O parâmetro myAgent indica o agente que está realizando a busca
52                  DFAgentDescription[] result = DFService.search(myAgent, template);
53
54                  //AgMG envia mensagem para o AgGG informando o id do grupo
55                  for(int i = 0; i < result.length; i++){
56                      ACLMessage message = new ACLMessage(ACLMessage.INFORM);
57                      message.addReceiver(new AID(result[i].getName().getLocalName(), AID.ISLOCALNAME));
58                      message.setContentObject(groupid);
59                      message.setConversationId("formacao-grupos");
60                      send(message);
61                   }
62              } catch (FIPAException ex) {
63                  Logger.getLogger(AgMG.class.getName()).log(Level.SEVERE, null, ex);
64              } catch (IOException ex) {
65                  Logger.getLogger(AgMG.class.getName()).log(Level.SEVERE, null, ex);
66              }
67
68              //Finalizou suas tarefas, então finaliza a instância do agente
69              doDelete();
```

Figura 24 – Parte do código-fonte do AgMG para enviar mensagem para o AgGG
Fonte: Dados produzidos pelo autor

Como pode ser visto na Figura 24, o AgMG cria, inicialmente, um *template* que serve para informar qual tipo de serviço ele está buscando, no caso um serviço do tipo "*agente-gerenciador-grupos*" (linhas 40-47). Esse comportamento é de execução única. Ele faz uma busca no DF para verificar se um serviço do tipo citado foi adicionado e, caso exista, a descrição desse serviço é adicionada a um vetor (linha 52). Em seguida, ele envia uma mensagem para o AgGG com o resultado da busca, passando como parâmetro o identificador do grupo a ser criado (linhas 55-61). Por fim, após encerrar suas tarefas, a instância do agente é finalizada.

6.3.8.5 Modelo do Agente Gerenciador de Grupos – AgGG

O AgGG realiza o papel "Gerenciador de Grupos". A Tabela 8 apresenta o seu *template* textual.

Tabela 8 – *Template* textual do AgGG

Agente: AgGG
Objetivos Disponibilizar um serviço do tipo "*agente-gerenciador-grupos*" no DF de modo que o AgMG possa encontrar esse serviço para se comunicar com ele. Uma vez em posse das informações enviadas pelo AgMG, o AgGG forma, automaticamente, grupos afins com base nos perfis dos estudantes e dos grupos.
Parâmetros de Entrada Identificador do grupo a ser criado e informações sobre os perfis dos estudantes e dos grupos.
Parâmetros de Saída Arquivo com um *ranking* dos candidatos aptos a formarem o grupo.
Condição de Ativação Quando o agente recebe do AgMG uma mensagem com identificador de conversação do tipo *formacao-grupos*.
Condição de Finalização Quando o agente gera um *ranking* com os candidatos aptos a formarem o grupo.
Informação Associada Possui um comportamento que disponibiliza um serviço do tipo "*agente-gerenciador-grupos*" no DF de modo que o AgMG possa encontrar esse serviço e se comunicar com ele. Também possui um comportamento cíclico (tipo *Cyclic Behaviour*) para ficar monitorando por requisições do AgMG. Uma vez em posse dessas informações, um comportamento de execução única (tipo *One Shot Behaviour*) é acionado para gerar um *ranking* com os candidatos aptos a formarem o grupo.
Descrição Este agente possui o comportamento voltado para o auxílio ao facilitador na tarefa de formação de grupos. Ele é responsável pela alocação automatizada de estudantes a grupos de trabalho para solução de um problema.

Fonte: Dados produzidos pelo autor

A Figura 25 mostra a parte do código-fonte do AgGG responsável por cadastrar um serviço do tipo "*agente-gerenciador-grupos*" no DF.

```
@Override
protected void setup(){
    System.out.println("Agente Gerenciador de Grupos " + getAID().getName() + " executando.");

    //Criando uma entrada no DF
    DFAgentDescription template = new DFAgentDescription();
    //Informando a AID do agente
    template.setName(getAID());

    //Criando um serviço de "agente-gerenciador-grupos" no DF
    ServiceDescription sd = new ServiceDescription();
    sd.setType("agente-gerenciador-grupos"); //Tipo do Serviço
    sd.setName("agente-gerenciador-grupos-moodle"); //Nome do Serviço
    template.addServices(sd); //Adicionando o Serviço

    //Registrando o agente no DF
    try{
        DFService.register(this, template); //register(agente que oferece, descrição)
    }catch(FIPAException e){
        System.out.println(e.getACLMessage());
    }

    addBehaviour(new GetMessage());
}
```

Figura 25 – Parte do código-fonte do AgGG para cadastrar serviço no DF
Fonte: Dados produzidos pelo autor

Similarmente aos códigos de cadastro dos agentes AgPA e AgDP (ilustrados nas Figuras 16 e 20, respectivamente), no trecho de código do AgGG, ao final do cadastro do serviço, é adicionado um comportamento *GetMessage*. Esse é um comportamento cíclico para ficar monitorando por requisições do AgMG. Nesse comportamento, o AgGG procura continuamente em sua pilha por uma mensagem que utilize o identificador de conversação *formacao-grupos*. Uma vez em posse dessas informações, o AgGG aciona o comportamento para a formação automática de grupos. Esse comportamento é de execução única. A Figura 26 mostra um trecho de código do comportamento *GetMessage*.

```
63      private class GetMessage extends CyclicBehaviour{
64          private String groupid = "";
65
66          @Override
67          public void action(){
68
69              jade.lang.acl.ACLMessage msg = myAgent.receive();
70
71              if(msg != null){
72
73                  if(msg.getConversationId().equals("formacao-grupos")){
74                      try {
75                          groupid = (String) msg.getContentObject();
76                      } catch (UnreadableException ex) {
77                          Logger.getLogger(AgGG.class.getName()).log(Level.SEVERE, null, ex);
78                      }
79                      addBehaviour(new FormacaoDeGrupos(groupid));
80                      System.out.println("O agente "+msg.getSender().getLocalName()+
81                              " solicitou uma formação de grupos.");
82                  }else{
83                      block();
84                  }
85              }
86          }
87      }
88
89      private class FormacaoDeGrupos extends OneShotBehaviour{...}
61  }
```

Figura 26 – Trecho do código-fonte do AgGG para tratar requisições do AgMG
Fonte: Dados produzidos pelo autor

O comportamento *FormacaoDeGrupos*, ilustrado na Figura 26 é responsável pela formação automática dos grupos. Ele carrega um programa em Prolog que gera um *ranking* com os candidatos aptos a formarem o grupo, a partir de um arquivo Percepcoes.txt que contém os candidatos. Esse *ranking* é armazenado no arquivo Acoes.txt. O trecho de código desse comportamento está ilustrado na Figura 27.

```
@Override
public void action() {

    try {

        JPL.init(); // Inicialização do JPL

        Term consult_arg[] = {
            //Especificação do caminho absoluto do programa prolog (.pl)
            new Atom("C:\\Users\\Mabel Fontes\\Documents\\NetBeansProjects\\MAS_PBL\\Prolog\\Forma_Grupos2.pl")
        };

        Query query = new Query("consult", consult_arg);

        boolean consulted = query.query();

        if (!consulted) {
            System.exit(1);
        }

        if (query.hasSolution()) {
            Query queryAgenteFormaGrupos = new Query("agente_forma_grupo", new Term[]{
                new Atom("C:\\Users\\Mabel Fontes\\Documents\\NetBeansProjects\\MAS_PBL\\Prolog\\Percepcoes.txt"),
                new Atom("C:\\Users\\Mabel Fontes\\Documents\\NetBeansProjects\\MAS_PBL\\Prolog\\Acoes.txt")});


            if (queryAgenteFormaGrupos.hasSolution()) {

                BufferedReader reader = new BufferedReader(new FileReader
                        ("C:\\Users\\Mabel Fontes\\Documents\\NetBeansProjects\\MAS_PBL\\Prolog\\Acoes.txt"));
                String line;
```

Figura 27 – Trecho do código-fonte do AgGG para formar grupos automaticamente
Fonte: Dados produzidos pelo autor

6.3.8.6 Modelo do Agente Recomendador – AgR

O AgR realiza o papel “Recomendador”. A Tabela 9 apresenta o seu *template* textual.

Tabela 9 – *Template* textual do AgR

Agente: AgR
Objetivos Detectar OAs adequados ao contexto do estudante, de acordo com (i) as informações providas pelos seus respectivos perfis, (ii) os assuntos que os estudantes desconhecem e (iii) as informações obtidas dos OAs disponíveis no repositório.
Parâmetros de Entrada Informações sobre os perfis dos estudantes, os conceitos que os estudantes desconhecem do problema proposto e informações dos OAs disponíveis no repositório de OAs SCORM.
Parâmetros de Saída Possíveis OAs adequados ao contexto do estudante.
Condição de Ativação Quando o estudante submete os conceitos desconhecidos, via interface do Moodle, durante a fase de identificação dos fatos do ciclo da PBL.
Condição de Finalização Quando o agente envia para o estudante, via e-mail, as informações do OA a ser

recomendado.
Informação Associada Toma as decisões de acordo com o resultado da execução de um AG, descrito na Subseção 6.3.6. Essa recomendação auxilia os estudantes a sanar suas dúvidas na fase de identificação dos fatos, durante a resolução do problema.
Descrição Este agente possui o comportamento voltado para a recomendação de OAs. Ele é responsável por detectar e realizar a recomendação de OAs.

Fonte: Dados produzidos pelo autor

6.3.8.7 Modelo do Agente DF

O Agente DF realiza o papel "Mediador". A Tabela 10 apresenta o seu *template* textual.

Tabela 10 – *Template* textual do Agente DF

Agente: DF
Objetivos Possibilitar que os agentes possam cadastrar e procurar por serviços oferecidos por outros agentes (serviço de páginas amarelas).
Parâmetros de Entrada Registros de serviços por parte dos agentes.
Parâmetros de Saída Lista de serviços e os respectivos agentes responsáveis pelo mesmo.
Condição de Ativação Quando a plataforma JADE é inicializada.
Condição de Finalização Caso a plataforma JADE seja finalizada.
Informação Associada É um agente inicializado pela própria plataforma JADE como um dos componentes integrantes do padrão FIPA.
Descrição Este agente possui o comportamento voltado para a mediação entre os outros agentes. Sua função principal é fornecer uma arquitetura do tipo "quadro-negro", onde agentes escrevem informações, procuram por informações escritas por outros agentes e conseguem, através do serviço provido pelo DF, se comunicar com o agente que escreveu aquela informação.

Fonte: Dados produzidos pelo autor

A principal vantagem em utilizar o JADE reside no fato de que é possível utilizar todos os componentes da especificação FIPA, visto que o JADE atende todas as especificações deste padrão e ainda o estende, sendo que um desses componentes é o agente DF (SILVA, 2012).

Outra vantagem dessa plataforma consiste na possibilidade de consultar e utilizar, através do MTS (*Message Transport Service*), DFs que estejam sendo executados em outras plataformas de agentes (APs). Assim, o DF utilizado na plataforma JADE deste trabalho pode, por exemplo, ser consultado, por meio da Internet, por um agente que esteja sendo executado em outro local, através do protocolo MTP (*Message Transport Protocol*). Essa característica permite a criação de uma rede de colaboração entre os agentes, mesmo que estejam localizados, por exemplo, em universidades distintas (SILVA, 2012). No presente trabalho, este protocolo foi utilizado para comunicação entre agentes que se encontravam na mesma plataforma, visto que os seis agentes criados e o DF se encontram no mesmo servidor.

A Figura 28 mostra, através da interface gráfica do JADE, como ocorre a comunicação entre os agentes AgMP, AgDP, DF e AgPA descrita através dos *templates* textuais dos agentes apresentados anteriormente.

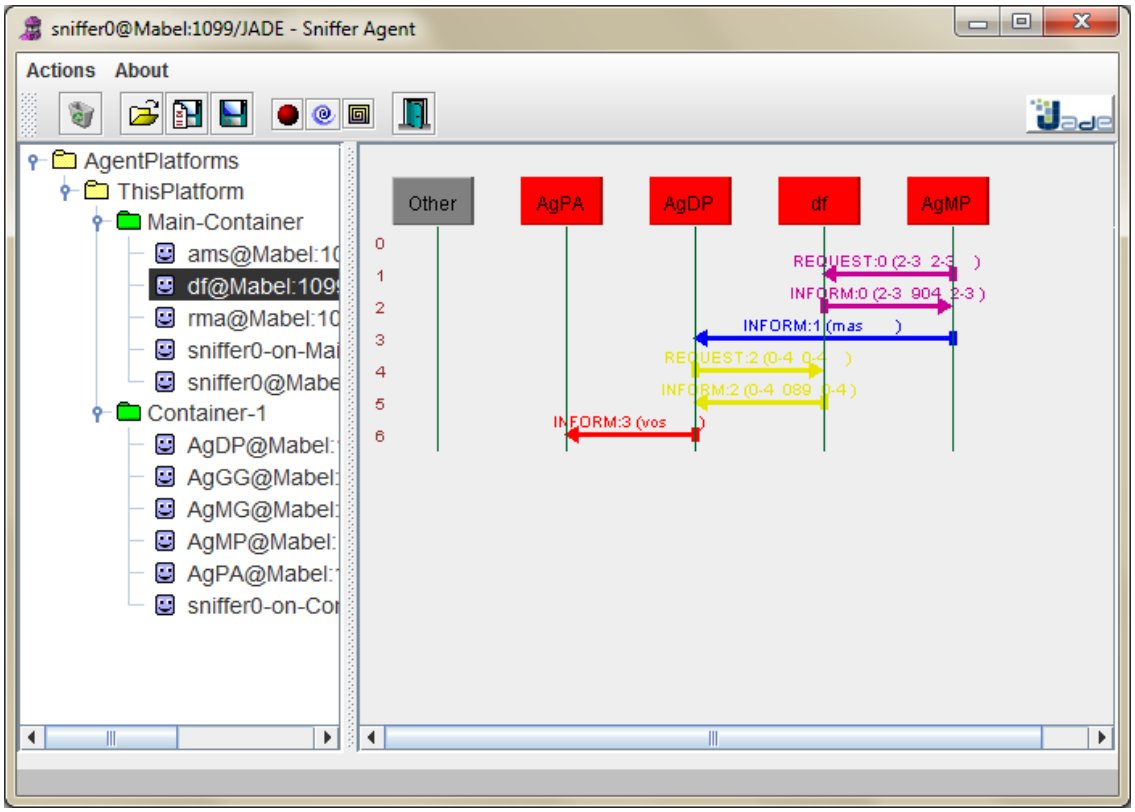

Figura 28 – Comunicação entre os agentes AgMP, AgDP, DF e AgPA através do JADE
Fonte: Dados produzidos pelo autor

Como pode ser visto na Figura 28, inicialmente, o AgMP solicita (REQUEST:0) por informações de serviço do tipo "*agente-detector-problemas*" ao DF. Em seguida, o DF responde (INFORM:0) com a informação de serviço disponível e com o AID do agente responsável pelo serviço, ou seja, o AgDP. No próximo passo, o AgMP envia uma mensagem para o AgDP (INFORM:1), acionando os seus comportamentos de detecção de problemas. Uma vez que o AgDP tenha detectado algum problema de colaboração, ele solicita ao DF (REQUEST:2) por informações de serviço do tipo "*agente-pedagogico-animado*". O DF, por sua vez, responde (INFORM:2) com a informação de serviço disponível e com o AID do AgPA. Por fim, quando recebe a resposta do DF, o AgDP envia uma lista com todas as informações dos estudantes envolvidos no problema detectado para o AgPA (INFORM:3), configurando, para tanto, o identificador de conversação *estudantes-passivos* ou *conversacao-fora-contexto*, dependendo do problema detectado.

A Figura 29 apresenta como ocorre a comunicação entre os agentes AgMG, AgGG e DF.

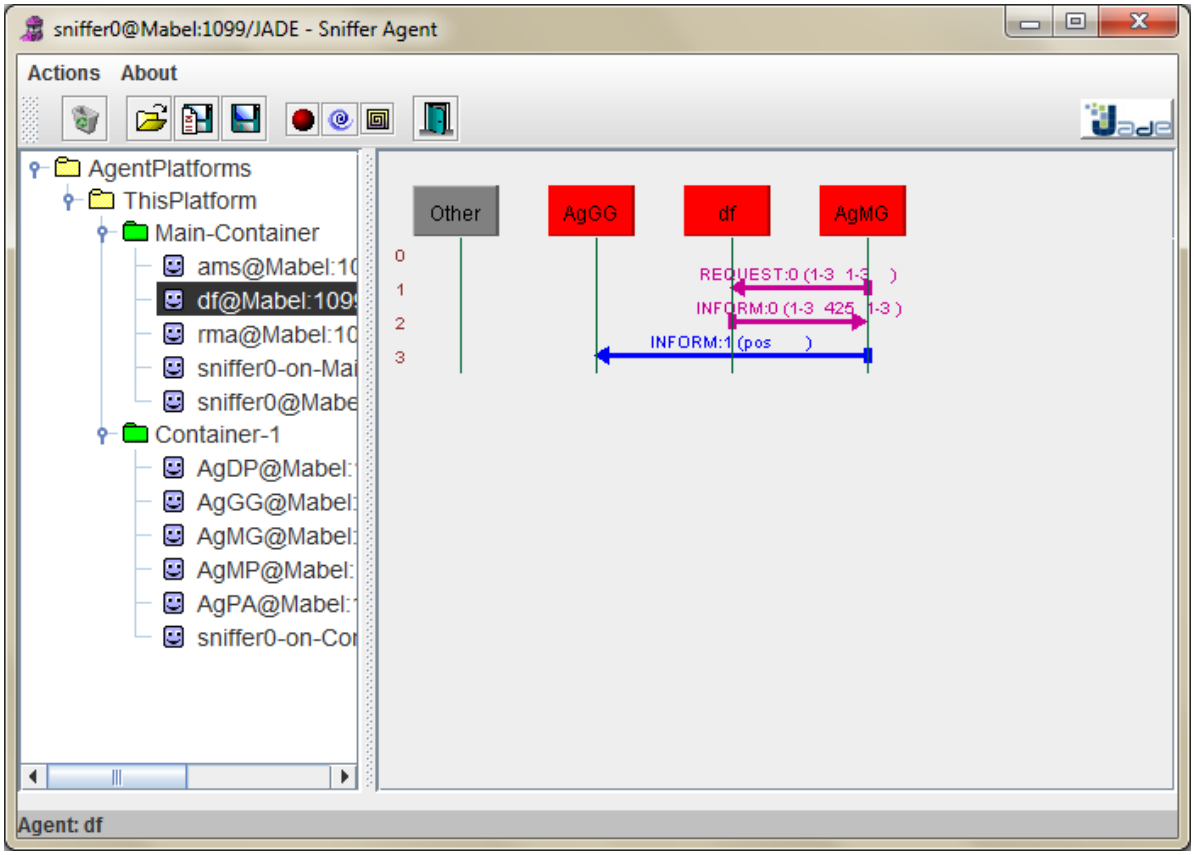

Figura 29 – Comunicação entre os agentes AgMG, AgGG e DF através do JADE
Fonte: Dados produzidos pelo autor

Conforme pode ser visto na Figura 29, inicialmente, o AgMG solicita (REQUEST:0) por informações de serviço do tipo "*agente-gerenciador-grupos*" ao DF. Em seguida, o DF responde (INFORM:0) com a informação de serviço disponível e com o AID do agente responsável pelo serviço, ou seja, o AgGG. Por fim, quando recebe a resposta do DF, o AgMG envia uma solicitação de criação automática do grupo para o AgGG (INFORM:1), configurando, para tanto, o identificador de conversação *formacao-grupos*.

Como já foi dito, o AgR não participa das comunicações apresentadas anteriormente. De fato, o AgR não se comunica com nenhum dos agentes presentes nessa arquitetura. Quando acionado, o AgR executa suas tarefas sem precisar, diretamente, de nenhum serviço oferecido pelos outros agentes, embora necessite de informações gravadas por estes nas bases de dados.

A Figura 28 retrata a interação que ocorre entre os agentes AgMP, AgDP, DF e AgPA ilustrada no Modelo de Interação da Figura 14. Já a Figura 29 retrata a interação que ocorre entre os agentes AgMG, AgGG e DF ilustrada no Modelo de Interação da Figura 15. Como o DF é inicializado com a própria plataforma, não foi necessário programar o seu comportamento, mas sim programar o comportamento dos agentes que se comunicam com o DF, fosse para consultar ou para cadastrar um novo serviço. Os trechos de códigos mostrados nos agentes das subseções anteriores mostraram como é possível se comunicar com o DF.

6.3.9 Cenários de Uso do Moodle

Nas subseções a seguir, são apresentados alguns cenários típicos de uso do facilitador e do estudante no Moodle.

6.3.9.1 Cenários de Uso do Facilitador

A Figura 30 ilustra a tela inicial do Moodle após a autenticação do facilitador no sistema.

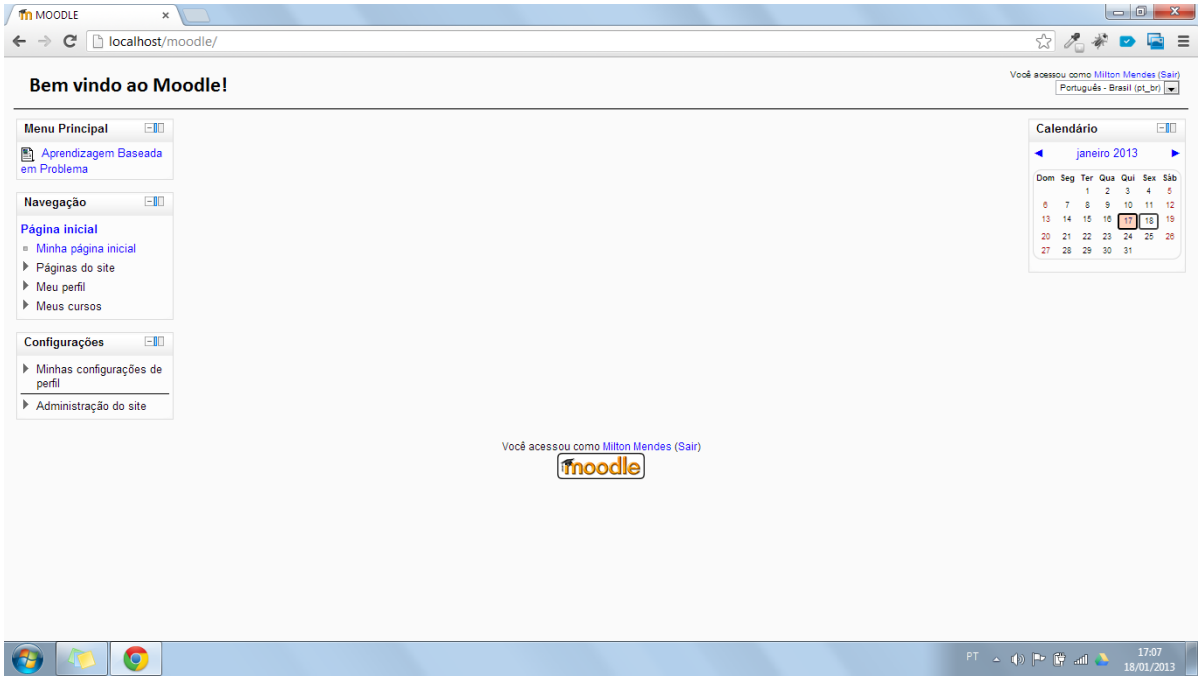

Figura 30 – Tela inicial do Moodle após a autenticação do facilitador
Fonte: Dados produzidos pelo autor

É interessante ressaltar que a solução apresentada neste trabalho permite ao facilitador decidir se este irá realizar o curso com base na PBL ou não. Caso o facilitador não tenha interesse em adotar essa teoria de aprendizagem, ele tem a liberdade de aplicar seu curso com base nas ferramentas já disponíveis no Moodle. Agora, caso o facilitador queira implantar um método de ensino com base na PBL, ele deverá iniciar este processo clicando no link "Aprendizagem Baseada em Problema", no menu principal do Moodle, conforme ilustrado na Figura 30.

A partir desse link é dado início ao processo de aplicação da PBL. A primeira tela exibida ao facilitador após o início do processo de aplicação da PBL apresenta uma listagem com todos os cursos, vinculados ao referido facilitador, existentes até o momento. A partir dessa interface, o facilitador poderá selecionar e acessar um curso de seu interesse. A Figura 31 ilustra a interface com todos os cursos existentes.

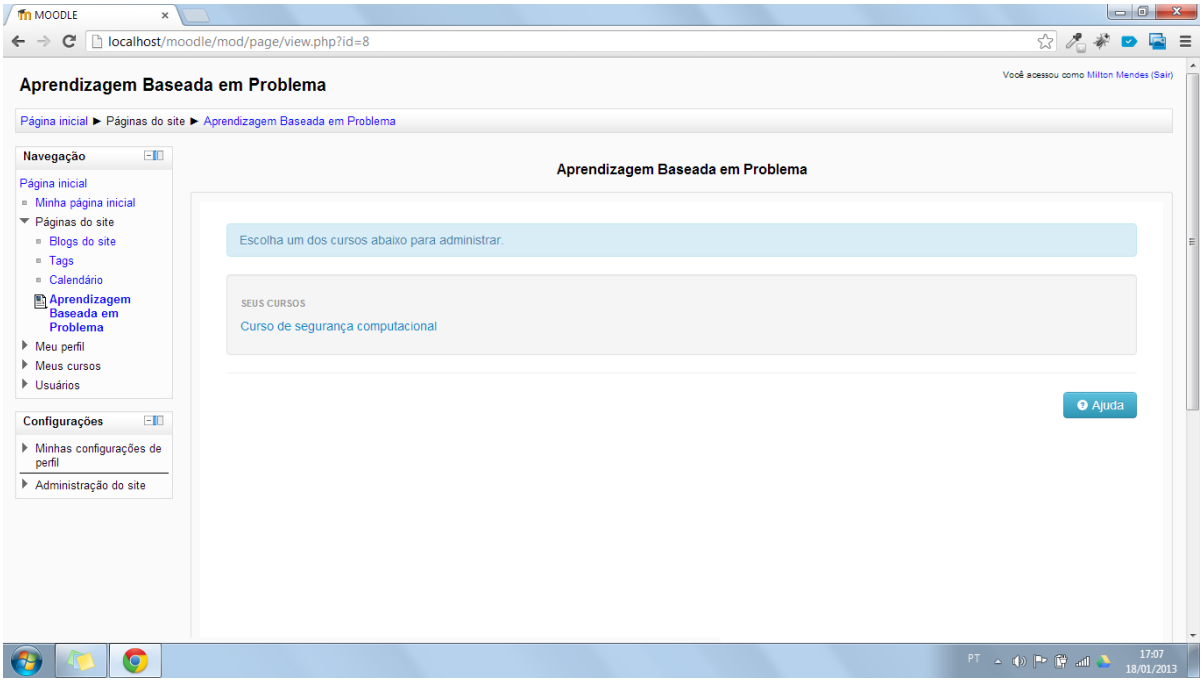

Figura 31 – Tela de listagem dos cursos
Fonte: Dados produzidos pelo autor

Nessa mesma tela, ilustrada na Figura 31, é apresentado um botão de ajuda, no qual o facilitador poderá consultar e obter informações a respeito da PBL. Esse menu de ajuda foi criado com o intuito de auxiliar o facilitador, não familiarizado com a PBL, a entender seu funcionamento, além de apresentar dicas de como se portar em determinadas situações. A Figura 32 ilustra o menu de ajuda.

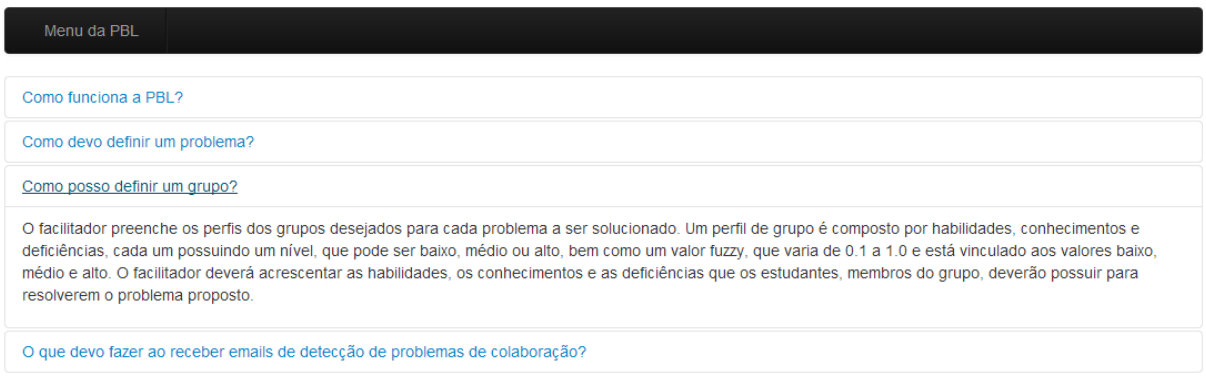

Figura 32 – Menu de ajuda da PBL
Fonte: Dados produzidos pelo autor

Como já foi dito, no processo da PBL, o facilitador é responsável por elaborar e apresentar um problema a um grupo de estudantes. Isso caracteriza a primeira fase do ciclo da PBL, chamada de cenário do problema. O facilitador deverá inserir todas as informações, como, por exemplo, título, definição, palavras-chave, dentre outras, inerentes ao problema a ser solucionado por um grupo de estudantes. A Figura 33 ilustra a interface da definição do problema.

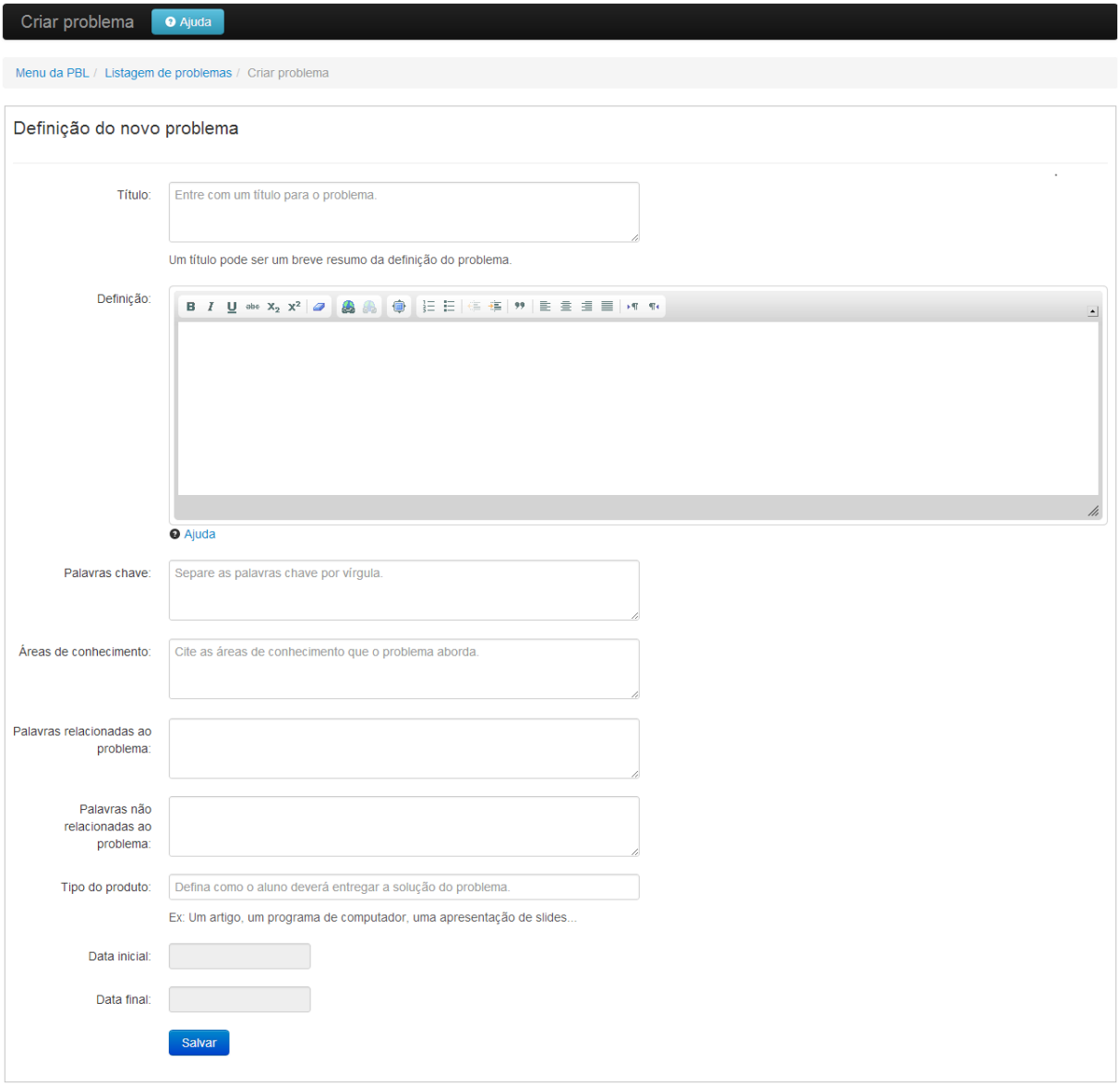

Figura 33 – Interface da definição do problema
Fonte: Dados produzidos pelo autor

Após inserir todas as informações necessárias à definição do problema, ilustradas na Figura 33, o facilitador irá submeter esses dados através do botão “Salvar”. Ao clicar neste botão, os dados serão salvos e o Agente Monitorador de Problemas (AgMP) será acionado. Uma vez acionado, o AgMP irá instanciar uma nova ontologia com todas essas informações inseridas pelo facilitador. Após instanciar e salvar a ontologia do referido problema, o AgMP irá acionar os comportamentos de detecção de problemas do Agente Detector de Problemas (AgDP). As ontologias dos problemas, instanciadas pelo AgMP, serão utilizadas pelo AgDP durante a execução do comportamento de detecção de conversações fora do contexto. Na verdade, o AgDP consulta, na ontologia, as palavras relacionadas e não relacionadas ao problema, necessárias ao algoritmo de detecção de conversações fora do contexto.

Uma vez tendo criado o problema, é necessário que o facilitador crie um grupo e atribua o problema a este grupo. Conforme mencionado, este trabalho apresenta um mecanismo de formação automática de grupos. Este mecanismo é realizado baseando-se nos perfis dos estudantes, descrito na próxima subseção, e perfis dos grupos. O facilitador é responsável pela criação do perfil do grupo. Um perfil de grupo é composto por habilidades, conhecimentos e deficiências, cada um possuindo um nível, que pode ser baixo, médio ou alto, bem como um valor *fuzzy*, que varia de 0.1 a 1.0 e que está vinculado aos valores baixo, médio e alto, conforme ilustrado na Figura 34. É importante salientar que o perfil desejado construído pelo facilitador é o que melhor se adéqua à resolução do problema; assim, um estudante que tenha um perfil aproximado ao desejado terá as competências necessárias à resolução do problema proposto.

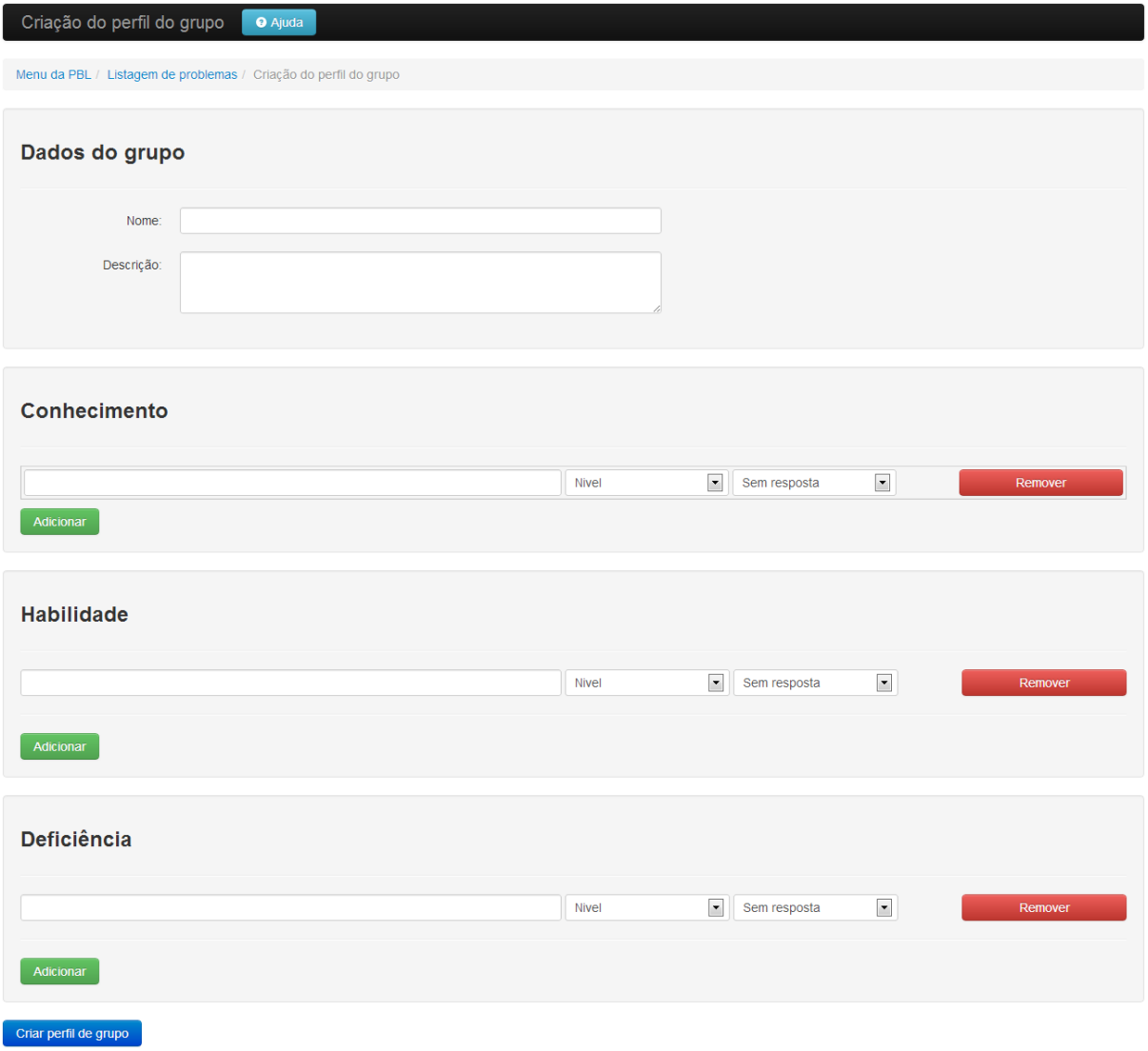

Figura 34 – Interface de criação de perfil do grupo
Fonte: Dados produzidos pelo autor

Após inserir todas as informações necessárias ao perfil do grupo, ilustradas na Figura 34, o facilitador irá submeter esses dados através do botão “Criar perfil de grupo”. Ao clicar neste botão, o perfil do grupo será criado e o Agente Monitorador de Grupos (AgMG) será acionado. Uma vez acionado, o AgMG irá solicitar, ao Agente Gerenciador de Grupos (AgGG), a formação automática do grupo. O AgGG irá analisar os perfis dos estudantes e o perfil do grupo e, no final desse processo, o AgGG irá gerar um arquivo com todos os candidatos aptos a formarem o grupo. O facilitador analisará este resultado, que é exibido através de uma interface do Moodle, ilustrada na Figura 35, e decidirá se acata ou não a sugestão do AgGG.

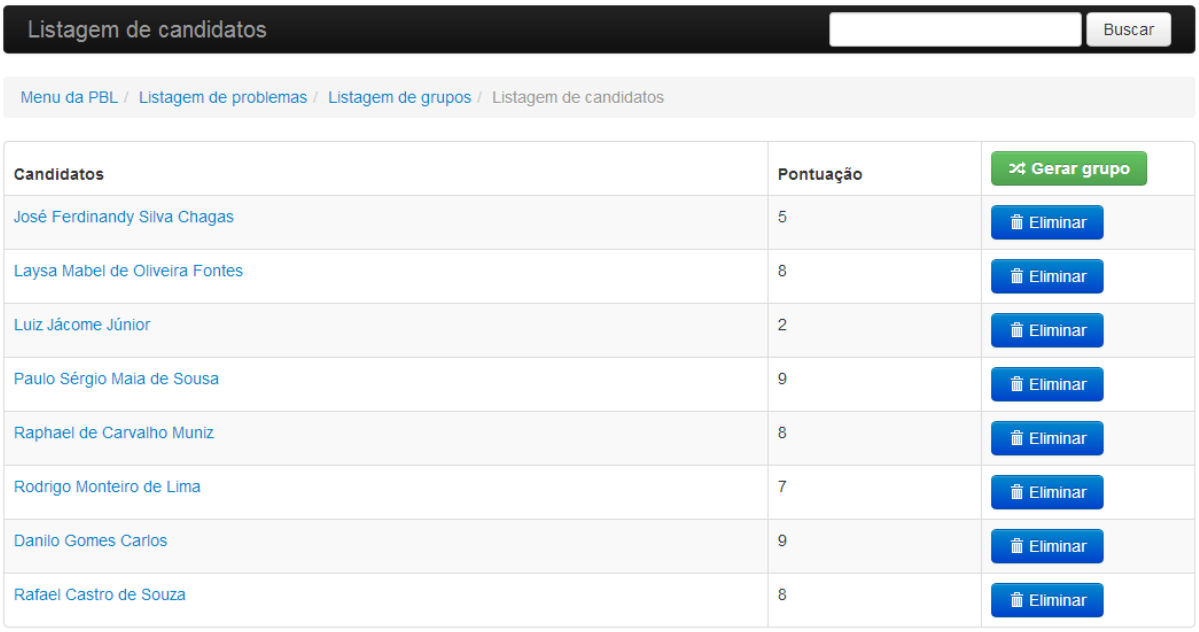

Figura 35 – Interface de listagem de candidatos
Fonte: Dados produzidos pelo autor

O facilitador irá analisar os possíveis candidatos, considerando suas respectivas pontuações, podendo excluir algum candidato com baixa pontuação. Finalmente, após esta análise, o facilitador poderá criar o grupo através do botão “Gerar grupo”, ilustrado na Figura 35. Outra interface interessante é a de listagem de grupos, ilustrada na Figura 36.

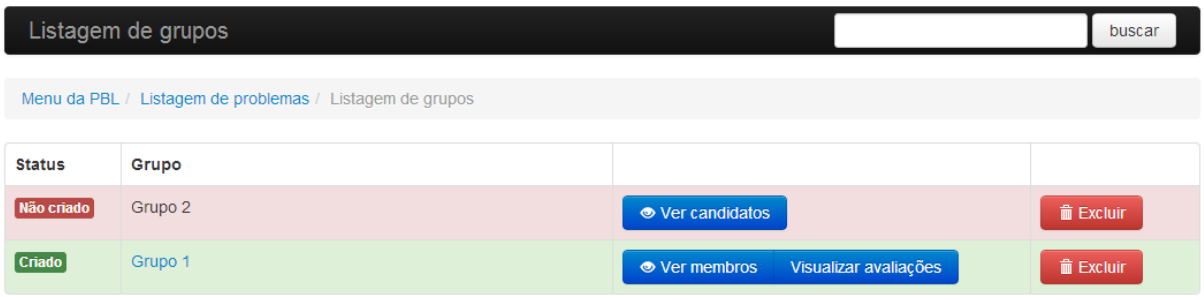

Figura 36 – Interface de listagem de grupos
Fonte: Dados produzidos pelo autor

O facilitador poderá visualizar todos os grupos existentes, conforme ilustrado na Figura 36. A coluna “Status”, exibida na Figura 36, mostra a situação atual dos grupos, ou seja, os grupos que já foram criados e os grupos que ainda estão aguardando a análise do facilitador para sua criação efetiva. Neste último caso, é necessário que o facilitador clique no

botão “Ver candidatos”, para realizar a análise dos candidatos e a criação do grupo, similarmente ao processo explicado anteriormente, ilustrado na Figura 35.

A Figura 37 ilustra a interface de visualização e gerenciamento das sessões[5].

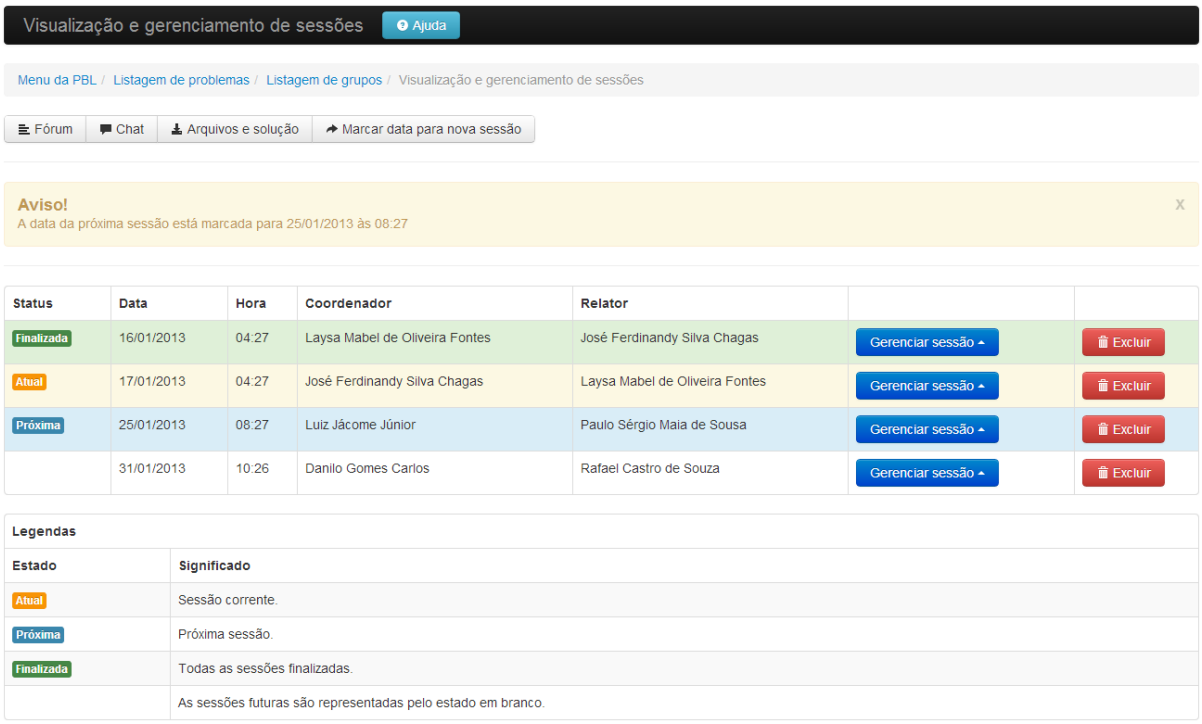

Figura 37 – Interface de visualização e gerenciamento das sessões
Fonte: Dados produzidos pelo autor

Através da interface ilustrada na Figura 37, é possível visualizar todas as informações, como, por exemplo, status, data, hora, coordenador e relator, referentes às sessões existentes. O facilitador também pode realizar o gerenciamento das sessões através do botão “Gerenciar sessão”. Por meio deste botão, é possível visualizar o relatório da sessão, as faltas dos estudantes, as metas de aprendizagem, além de realizar a avaliação do grupo para as sessões finalizadas.

Este trabalho também apresenta um módulo de avaliação dos estudantes. Este módulo é composto por duas interfaces: uma para o facilitador e outra para o estudante.

A interface do facilitador consiste na avaliação do grupo, citada anteriormente. A Figura 38 ilustra a interface de avaliação do grupo. A interface do estudante será detalhada na próxima subseção.

[5] Na PBL, uma sessão é uma reunião, com data e hora previamente definida, onde os membros de um grupo utilizam uma ferramenta síncrona para discutir e trocar informações inerentes ao problema proposto.

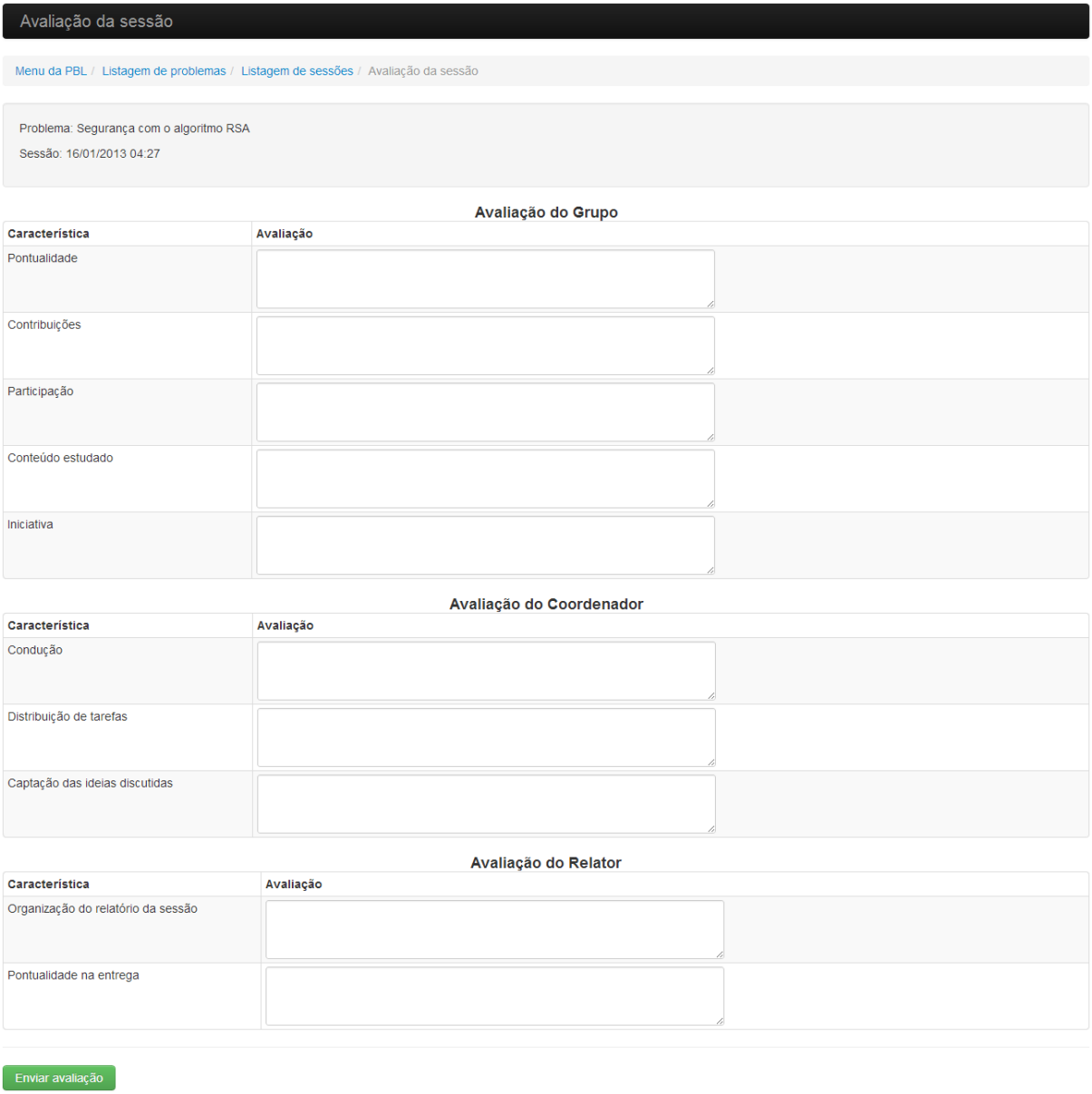

Figura 38 – Interface de avaliação da sessão
Fonte: Dados produzidos pelo autor

Conforme ilustrado na Figura 38, a interface do facilitador consiste em um formulário com vários critérios de avaliação do grupo, como, por exemplo, pontualidade, contribuições, participação, conteúdo estudado e iniciativa. Nesse mesmo formulário também existem alguns critérios de avaliação direcionados especificamente para o coordenador[6] e o relator[7] da sessão, como, por exemplo, condução, distribuição de tarefas e captação das ideias discutidas para o

[6] Na PBL, o coordenador é responsável por gerir os demais membros do grupo, no qual ele pertence, a atender as fases da PBL.
[7] Na PBL, o relator é um membro do grupo responsável pela criação do relatório da sessão e pelo gerenciamento dos estudantes faltosos.

coordenador, e organização do relatório da sessão e pontualidade na entrega para o relator. Essa avaliação é realizada a cada nova sessão concluída.

Vale ressaltar que, ao final de cada problema, os estudantes realizam uma autoavaliação e uma avaliação dos outros membros do grupo, chamada também de avaliação de pares. Essas avaliações são enviadas e visualizadas pelo facilitador de forma que o auxiliará na avaliação dos membros do grupo. A Figura 39 ilustra a interface, onde o facilitador poderá visualizar a avaliação de pares.

Figura 39 – Interface de visualização de avaliações
Fonte: Dados produzidos pelo autor

6.3.9.2 Cenários de Uso do Estudante

A Figura 40 ilustra a tela inicial do Moodle após a autenticação do estudante no sistema.

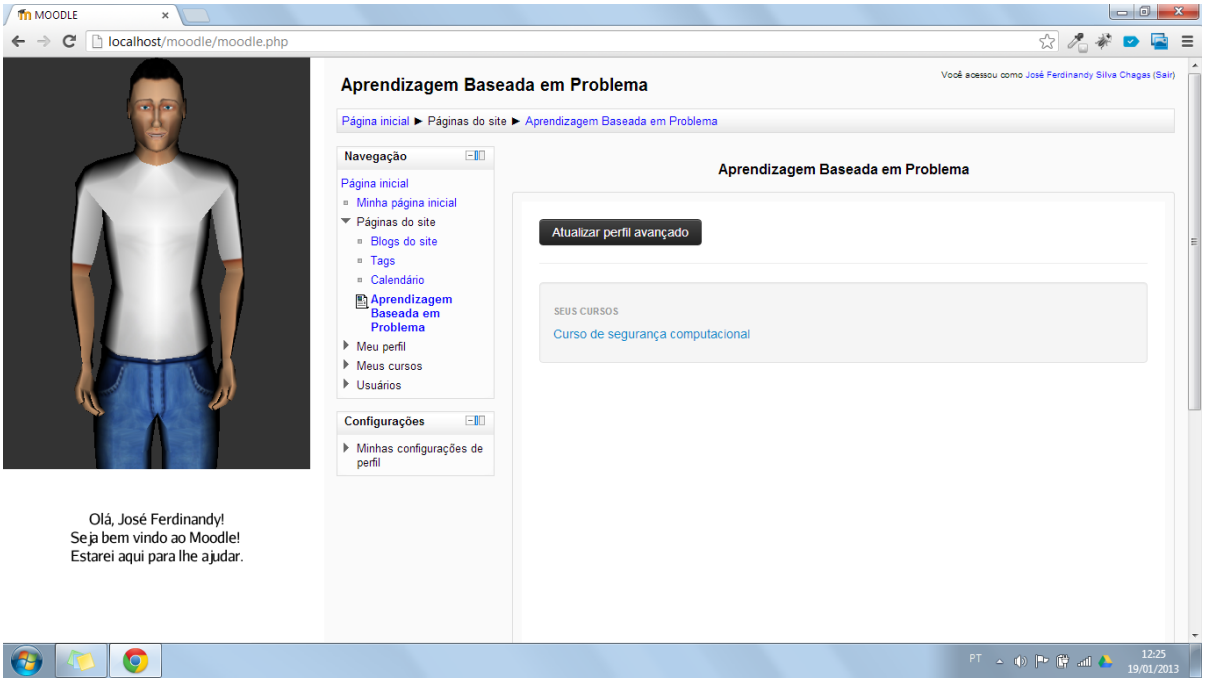

Figura 40 – Interface de boas vindas ao estudante
Fonte: Dados produzidos pelo autor

Conforme ilustrado na Figura 40, o Agente Pedagógico Animado (AgPA) exibe uma mensagem de boas vindas ao estudante. A Figura 40 também mostra todos os cursos no qual o estudante está matriculado. Nessa interface, é possível atualizar o perfil avançado, através do botão "Atualizar perfil avançado". Este botão irá direcionar para uma página de perfil do usuário. O perfil do usuário é formado pelo perfil avançado, utilizado para a recomendação de OAs, e o perfil do grupo, utilizado para a formação automática de grupos. É necessário que o estudante preencha esses perfis logo no primeiro acesso ao Moodle e que ele seja atualizado periodicamente, já que eles serão utilizados pelos agentes em momentos específicos desse processo. Por esse motivo, esse botão foi inserido logo no início da interação do estudante com o ambiente, como forma de "forçá-lo" a realizar o preenchimento de seus respectivos perfis. A Figura 41 ilustra a interface do perfil do usuário.

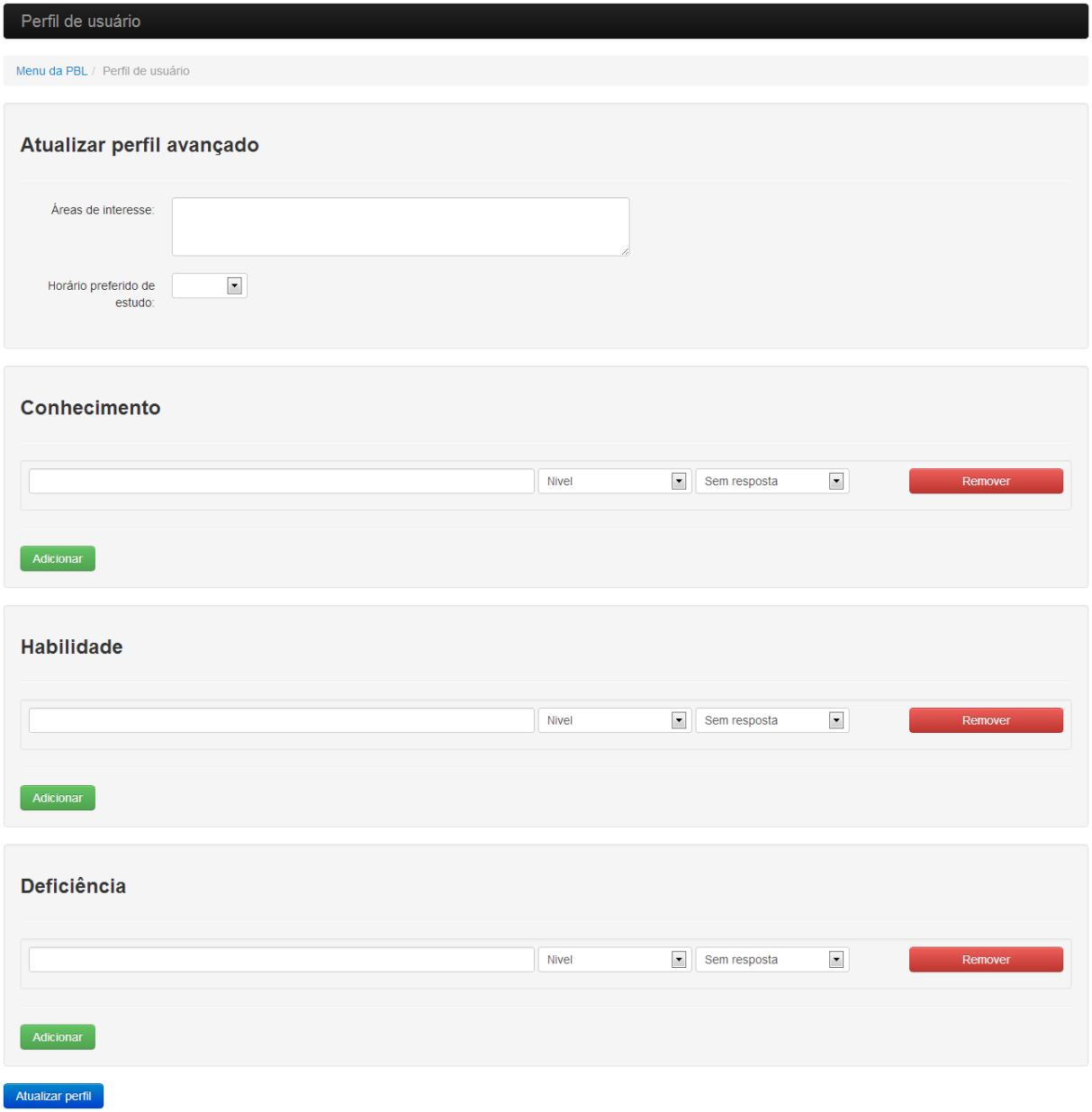

Figura 41 – Interface de perfil de usuário
Fonte: Dados produzidos pelo autor

A Figura 41 ilustra o perfil avançado do estudante que alimenta o seu respectivo perfil. Nesse perfil avançado, os estudantes deverão informar as áreas de interesse e o seu horário preferido de estudo. Essas informações serão úteis para identificar OAs adequados às necessidades do estudante. Outro critério analisado na detecção de OAs são os conceitos desconhecidos. Esses conceitos desconhecidos serão detalhados mais adiante.

A Figura 41 ainda apresenta outras três informações importantes: conhecimento, habilidade e deficiência, onde cada um possui um nível, que pode ser baixo, médio ou alto, podendo um estudante ter uma ou mais habilidades, deficiências e conhecimentos. Os

estudantes, através da interface ilustrada na Figura 41, preenchem seu perfil, que alimenta uma base de perfis que será usada no processo de formação de grupos, conforme explicado na subseção anterior.

Uma vez que o facilitador tenha definido e lançado o problema para um grupo, os estudantes, membros desse grupo, conseguem visualizar este problema. A Figura 42 ilustra a interface de visualização do problema.

Figura 42 – Interface de visualização de problema
Fonte: Dados produzidos pelo autor

Conforme ilustrado na Figura 42, os estudantes visualizam o problema proposto, a forma de entrega e a data da entrega do produto, ou seja, a solução do problema no formato solicitado pelo facilitador. Após a análise do problema, os estudantes deverão inserir os conceitos desconhecidos, ou seja, os conceitos que eles desconhecem na definição do problema. Essa ação caracteriza a segunda fase da PBL, chamada de identificação dos fatos. Após inserir os conceitos desconhecidos, ilustrados na Figura 42, o estudante irá submeter esses dados através do botão “Alterar”. Ao clicar neste botão, os dados serão salvos e o Agente Recomendador (AgR) será acionado. Uma vez acionado, o AgR irá tentar detectar OAs adequados ao contexto do estudante, de acordo com as informações providas pelos seus respectivos perfis, ou seja, as áreas de interesse, o seu horário preferido de estudo, os conceitos desconhecidos e as informações obtidas dos OAs disponíveis no repositório. Essa recomendação irá auxiliar os estudantes a sanar possíveis dúvidas durante a fase de identificação dos fatos da PBL.

Como explicado na subseção anterior, uma vez que o facilitador tenha criado o problema, o agente AgMP é acionado e, após realizar suas tarefas, ele aciona os comportamentos de detecção de problemas do AgDP. O AgDP é responsável por detectar estudantes com comportamento passivo, ou seja, aqueles estudantes que têm uma baixa interação nas ferramentas colaborativas; e detectar conversações fora do contexto, que acontecem quando os estudantes ficam dispersos e acabam utilizando as ferramentas colaborativas para conversarem sobre assuntos que fogem do contexto do problema. Uma vez que o AgDP tenha detectado um dos problemas citados, este irá comunicar o AgPA. O AgPA, por sua vez, irá exibir uma animação e uma mensagem condizentes com esta situação. A Figura 43 ilustra um momento no qual um estudante é detectado com comportamento passivo.

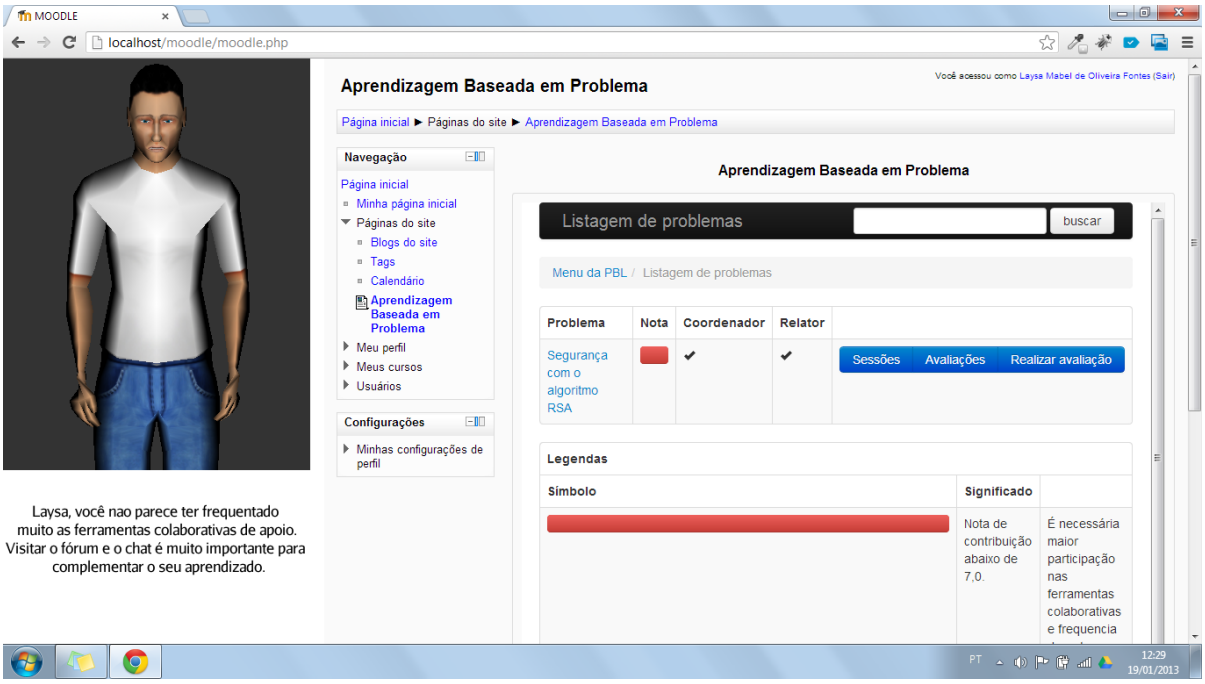

Figura 43 – Detecção de estudante passivo
Fonte: Dados produzidos pelo autor

Conforme ilustrado na Figura 43, o estudante pode também visualizar e gerenciar todos os problemas, no qual faz parte. Além da animação e mensagem exibidas pelo AgPA, os estudantes também conseguem visualizar seus desempenhos, através de suas notas, ilustradas também na Figura 43. Essas notas são representadas pelas cores vermelha, amarela e verde, condizentes aos seus desempenhos durante as sessões. A Figura 44 mostra a legenda da Figura 43 na íntegra.

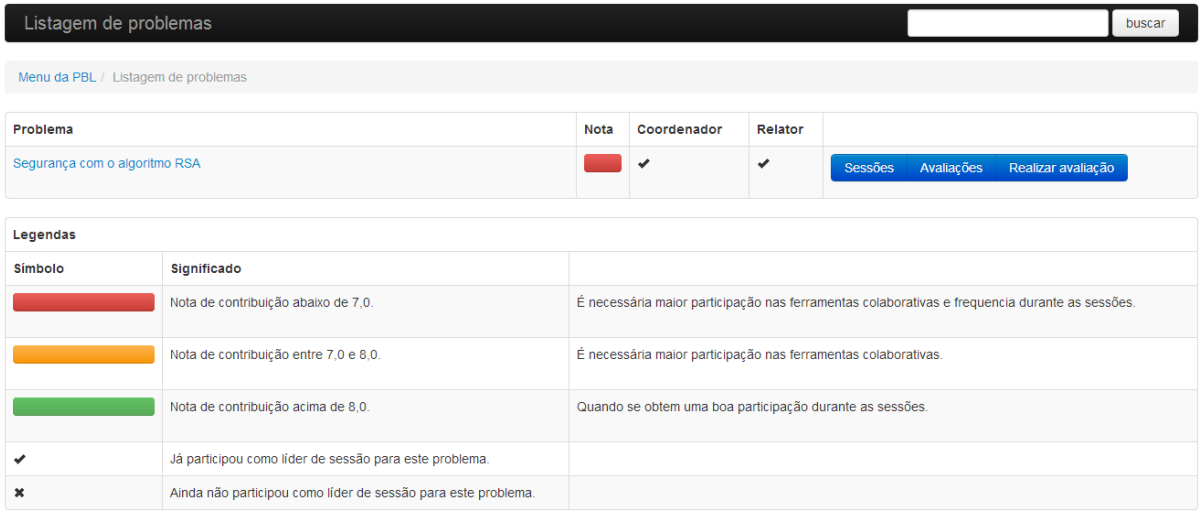

Figura 44 – Interface de listagem de problemas
Fonte: Dados produzidos pelo autor

Os estudantes também podem visualizar e gerenciar todas as sessões, através da interface ilustrada na Figura 45.

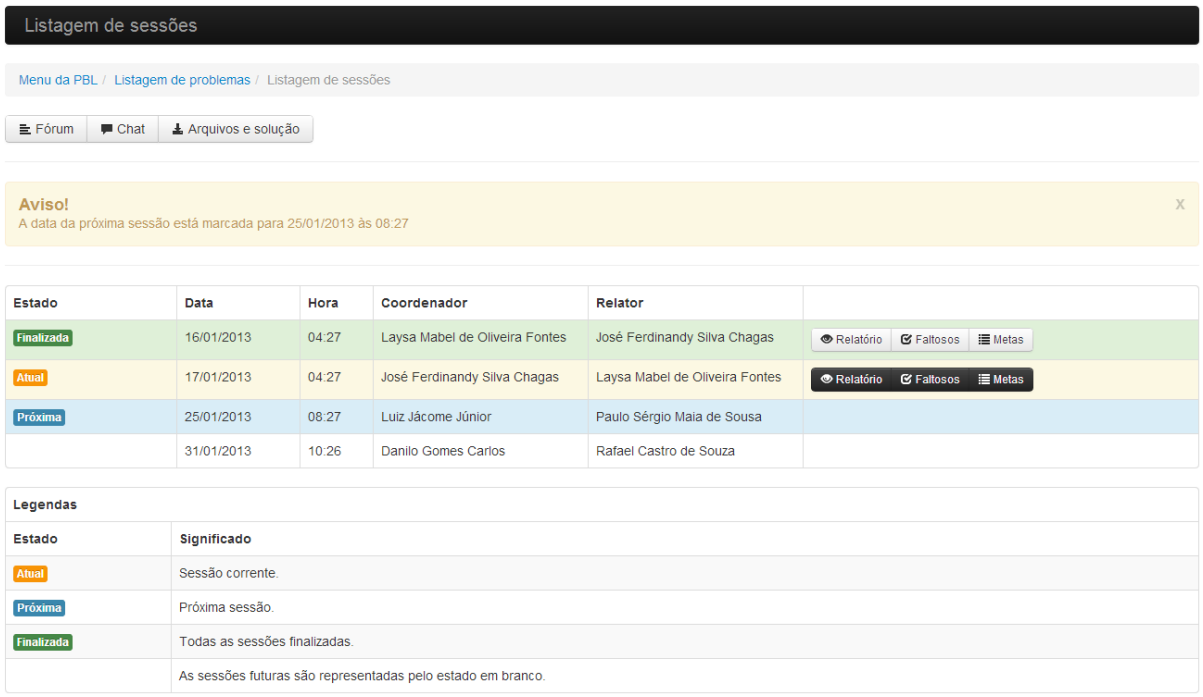

Figura 45 – Interface de listagem de sessões
Fonte: Dados produzidos pelo autor

Conforme pode ser visto na Figura 45, também é possível visualizar os relatórios, os estudantes faltosos e as metas de aprendizagem das sessões. O relator é responsável por criar o relatório da sessão. Esse relatório é criado e enviado para todos os membros do grupo, inclusive o facilitador, ao final de cada sessão. A Figura 46 ilustra a interface de relatório da sessão.

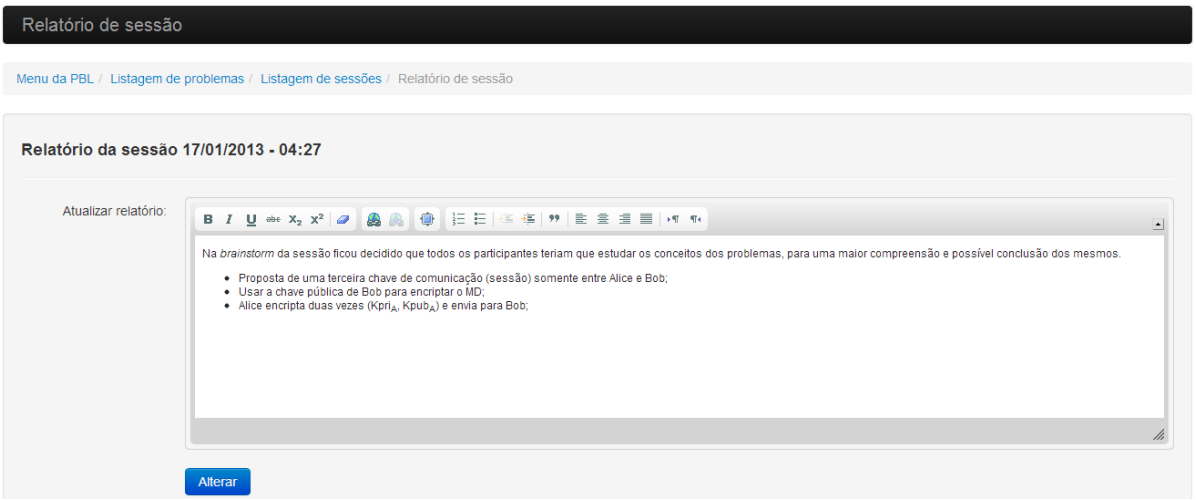

Figura 46 – Interface de relatório da sessão
Fonte: Dados produzidos pelo autor

O relator também é responsável por gerenciar os estudantes faltosos. A Figura 47 ilustra a interface de gerenciamento dos estudantes faltosos.

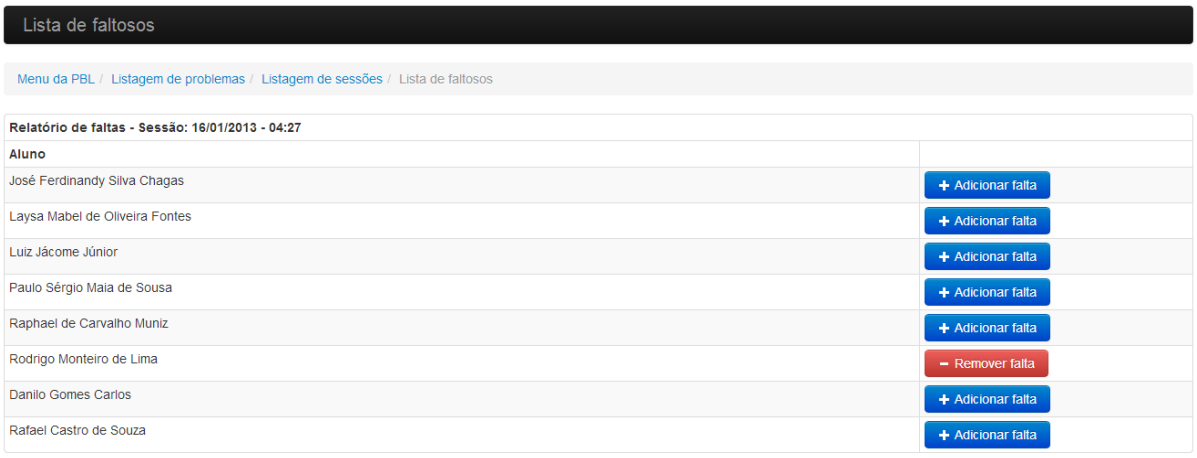

Figura 47 – Interface de gerenciamento dos estudantes faltosos
Fonte: Dados produzidos pelo autor

Já o coordenador é responsável pelo gerenciamento das metas para a próxima sessão. A Figura 48 ilustra a interface de gerenciamento de metas.

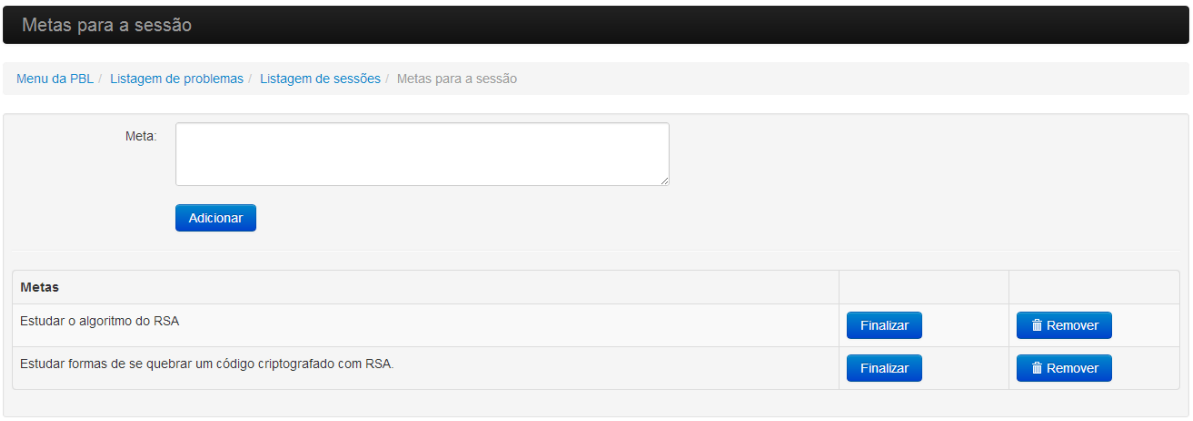

Figura 48 – Interface de gerenciamento de metas
Fonte: Dados produzidos pelo autor

Vale ressaltar que na PBL os grupos são compostos de 8 a 10 estudantes. Dentre os estudantes, um será o coordenador e outro será o relator, alternando de sessão em sessão, para que todos exerçam essas funções (BERBEL, 1998).

Uma vez que os estudantes tenham solucionado o problema, é necessário que eles enviem essa solução para o facilitador. Qualquer membro do grupo poderá ficar responsável por este envio. Esta pessoa é escolhida através de um acordo entre todos os membros do grupo. A Figura 49 ilustra a interface de envio de arquivos.

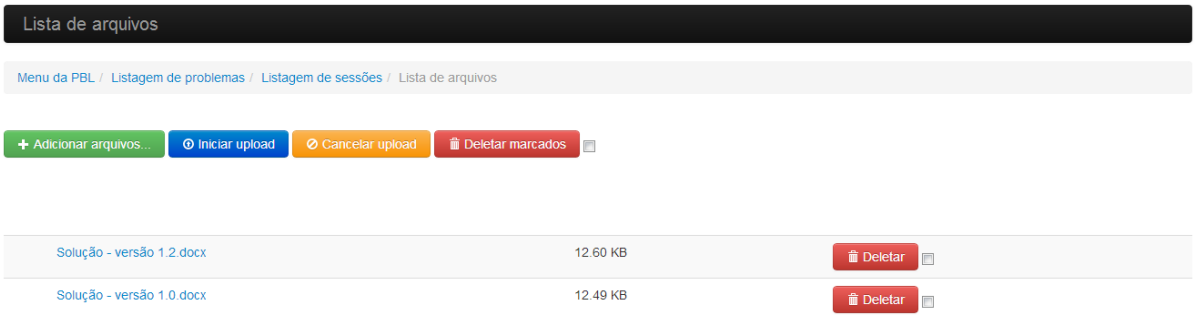

Figura 49 – Interface de envio de arquivos
Fonte: Dados produzidos pelo autor

Conforme mencionado na subseção anterior, ao final de cada problema, os estudantes realizam uma autoavaliação e uma avaliação dos outros membros do grupo, chamada também de avaliação de pares. Essas avaliações são enviadas e visualizadas pelo facilitador de forma que o auxiliará na avaliação dos membros do grupo. A Figura 50 ilustra a interface onde os estudantes realizam a autoavaliação e a avaliação de pares.

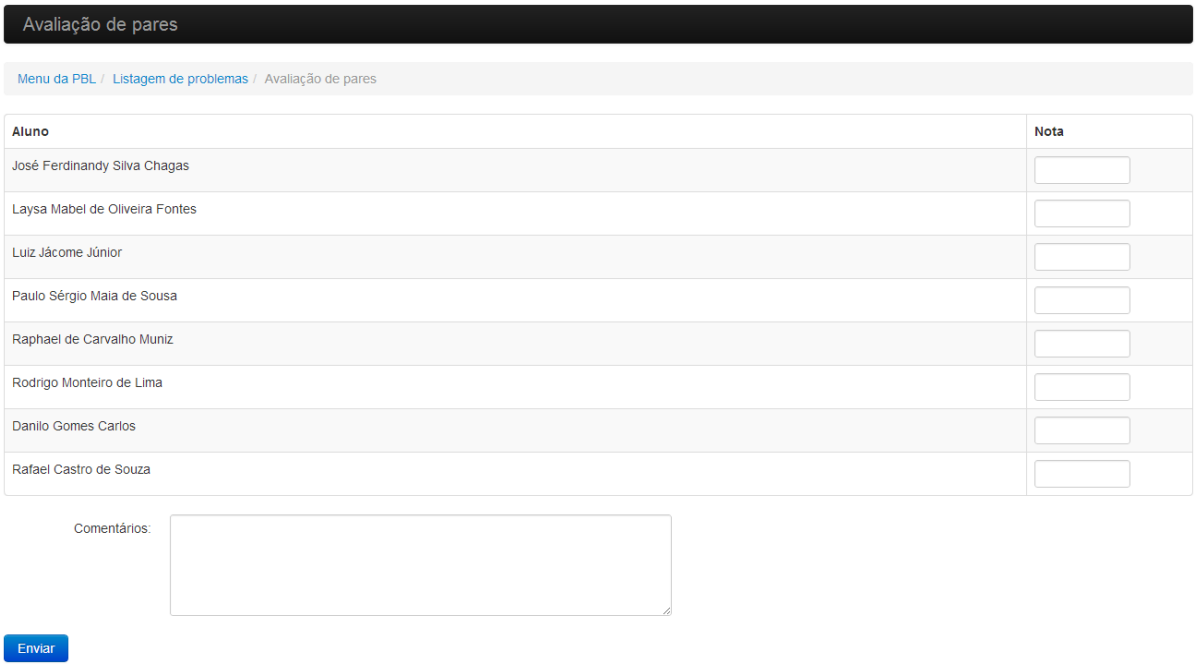

Figura 50 – Interface de avaliação de pares
Fonte: Dados produzidos pelo autor

7 PROCESSO DE DESENVOLVIMENTO DAS ONTOLOGIAS PROPOSTAS

Este capítulo apresenta detalhes do desenvolvimento das ontologias propostas neste trabalho. A Seção 7.1 descreve as tecnologias utilizadas no desenvolvimento das ontologias propostas. A Seção 7.2 apresenta a estrutura das ontologias. A Seção 7.3 descreve o processo de desenvolvimento da ontologia do problema. Por sim, a Seção 7.4 descreve o processo de desenvolvimento da ontologia da aprendizagem baseada em problema.

7.1 DESCRIÇÃO DAS TECNOLOGIAS DE DESENVOLVIMENTO

No desenvolvimento das ontologias propostas neste trabalho, foi utilizada a linguagem OWL (*Web Ontology Language*) (OWL, 2013). Esta linguagem de descrição de ontologia foi escolhida por apresentar características que tornam as ontologias mais robustas. A linguagem OWL apresenta também recursos adicionais não suportados por outras linguagens, como, por exemplo, sublinguagens incrementais, projetadas para serem usadas por diferentes comunidades de implementadores e usuários. A ferramenta selecionada para a modelagem da ontologia foi a Protégé (PROTÉGÉ, 2013). Para realizar a manipulação das ontologias, foi utilizado o *framework* Jena (DICKINSON, 2009).

7.2 ESTRUTURAÇÃO DAS ONTOLOGIAS

As ontologias não apresentam sempre a mesma estrutura, mas existem características e componentes básicos comuns presentes em grande parte delas (BRÓLIO *et al.*, 2006). As ontologias propostas neste trabalho possuem a seguinte estrutura:

- Classes e/ou subclasses: abrangem um conjunto de classes e uma hierarquia entre essas classes, ou seja, uma taxonomia;
- Propriedades: nas ontologias propostas, foram definidos dois tipos de propriedades: *Object Property* e *Datatype Property* (OWL, 2013). Propriedades do tipo *Object Property* têm o papel de qualificar ou relacionar classes. Já as

propriedades do tipo *Datatype Property* constituem campos que podem ser instanciados;

- Axiomas: modelam sentenças que são sempre verdadeiras. A criação de axiomas é feita através de definições formais;
- Instâncias: representam elementos específicos, ou seja, os próprios dados (instâncias das classes).

7.3 ONTOLOGIA DO PROBLEMA

A ontologia do problema está sendo utilizada por dois agentes: o AgMP e o AgDP. O AgMP instancia uma ontologia do problema para cada novo problema criado pelo facilitador durante a aplicação da PBL. O AgDP, por sua vez, consulta as ontologias dos problemas, instanciadas pelo AgMP, extraindo informações utilizadas em seu algoritmo de detecção de conversações fora do contexto. A seguir, será apresentado o processo de desenvolvimento da ontologia do problema.

7.3.1 Principais Classes Identificadas

Foram definidas 4 (quatro) classes (*Curso*, *Estudante*, *Grupo* e *Problema*), com o intuito de representar os conceitos gerais, conforme ilustrado na Figura 51. A seguir é apresentada uma breve explanação das principais classes que compõem o domínio da referida ontologia.

7.3.1.1 Problema

O problema na PBL desempenha um papel fundamental no processo de aprendizagem. Este problema deve possuir algumas características particulares, entre as quais destacam-se (PONTES, 2010):

- Precisa ser multidisciplinar, ou seja, necessita envolver várias áreas de conhecimento;
- Precisa ser mal estruturado e complexo, com o objetivo de instigar os estudantes a levantar questionamentos que posteriormente irão fazer com que eles busquem conhecimentos que em um primeiro momento eles não possuíam;
- Deve exigir um trabalho em equipe e motivar a colaboração entre os membros do grupo, ou seja, ele precisa ter um formato tal que motive esta socialização e divisão de tarefas;
- Precisa instigar a investigação e elaboração de hipóteses através das ideias;
- Precisa motivar cada estudante em particular ao aprendizado e resolução de problemas.

Essa classe foi criada com o intuito de representar todas as informações inerentes ao problema, inseridas pelo facilitador. A Figura 51 ilustra o desenvolvimento da classe *Problema* na ferramenta Protégé, como forma de exemplificar o desenvolvimento de um conceito presente na ontologia proposta.

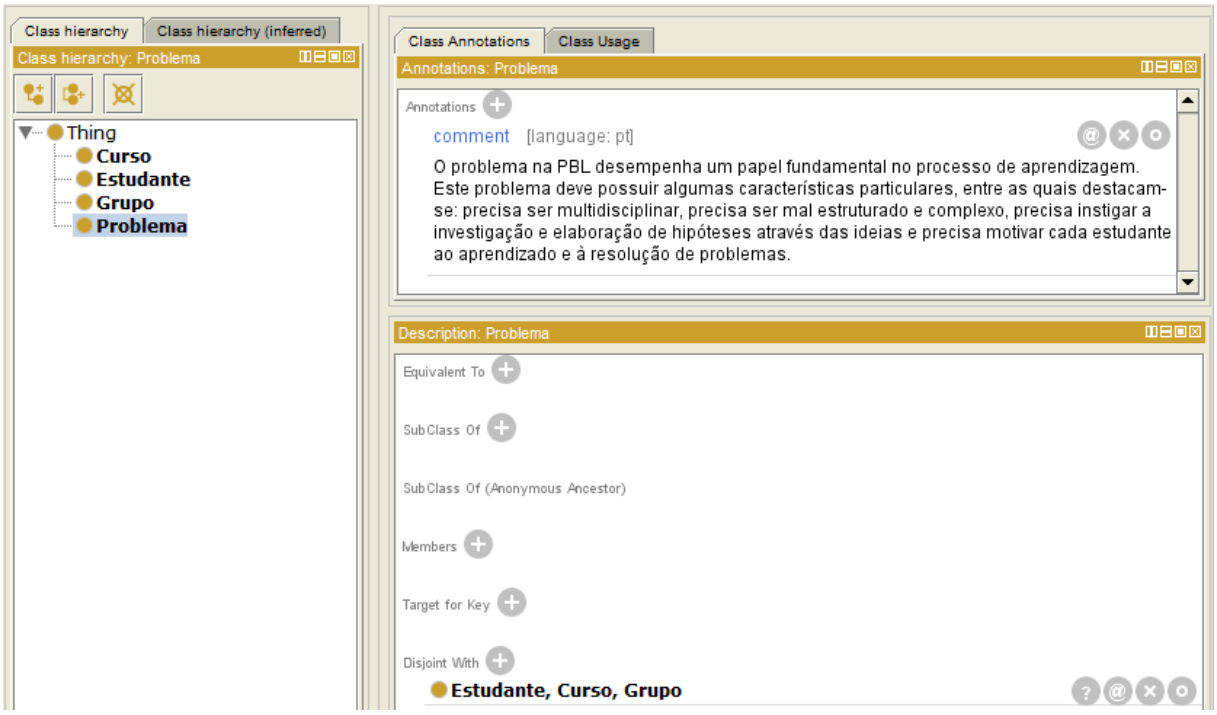

Figura 51 – Representação da classe *Problema* na ferramenta Protégé
Fonte: Dados produzidos pelo autor

O campo *Disjoint With*, ilustrado na Figura 51, é usado para desconectar um grupo de classes, ou seja, torná-las disjuntas. Isto garante que uma instância que tenha sido declarada como sendo membro de uma das classes do grupo não pode ser um membro de nenhuma outra classe naquele mesmo grupo. Na Figura 51, temos todas as classes disjuntas à classe *Problema*. A Figura 52 ilustra o código OWL referente à propriedade *Disjoint*.

```
<owl:Class rdf:ID="Problema">
  <owl:disjointWith>
    <owl:Class rdf:ID="Grupo"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Curso"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Estudante"/>
  </owl:disjointWith>
</owl:Class>
```

Figura 52 – Representação das classes disjuntas à classe *Problema* em OWL
Fonte: Dados produzidos pelo autor

7.3.1.2 Curso

Esta classe foi criada com o intuito de armazenar informações sobre o curso no qual o problema está vinculado.

7.3.1.3 Grupo

Na PBL, os membros de um grupo são responsáveis pela resolução de um problema, e, para isso, eles precisam ter competências complementares em relação ao problema em questão. Os grupos são compostos de um facilitador e entre 8 a 10 estudantes. Dentre os estudantes, um será o coordenador e outro será o relator, alternando de sessão em sessão, para que todos exerçam essas funções (BERBEL, 1998).

Esta classe foi criada com o intuito de armazenar informações sobre o grupo no qual o problema foi proposto.

7.3.1.4 Estudante

Esta classe foi criada com o intuito de armazenar informações sobre os estudantes, membros do grupo no qual o problema foi proposto.

7.3.2 Especificação das Propriedades

As propriedades de cada classe da ontologia foram especificadas à medida que as classes foram sendo definidas ou reutilizadas. A ontologia resultante possui 4 (quatro) propriedades do tipo *Object Property* e 8 (oito) propriedades do tipo *Datatype Property*.

Para representar uma propriedade do tipo *Object Property* ou *Datatype Property*, é necessário a definição do domínio da propriedade (*domain*) e da classe a qual se aplica a propriedade (*range*). Uma propriedade deve ser declarada como *Object Property* quando essa tem o papel de qualificar ou relacionar classes. Podemos citar, como exemplo do tipo *Object Property*, a propriedade *eMembroDoGrupo*, que tem como domínio a classe *Estudante* e como *range* a classe *Grupo*. Esta propriedade foi criada para associar todos os estudantes membros de um grupo. O trecho de código OWL referente à propriedade *eMembroDoGrupo* é mostrado na Figura 53.

```
<owl:ObjectProperty rdf:ID="eMembroDoGrupo">
  <rdfs:domain rdf:resource="#Estudante"/>
  <rdfs:range rdf:resource="#Grupo"/>
</owl:ObjectProperty>
```

Figura 53 – Representação da propriedade *eMembroDoGrupo* em OWL
Fonte: Dados produzidos pelo autor

Propriedades do tipo *Datatype Property* constituem os atributos de uma classe que podem ser instanciados através, por exemplo, do preenchimento de campos em um formulário de entrada de dados. Neste tipo de propriedade, é necessário definir o domínio ao qual ela pertence e a *range* que, diferentemente do tipo *Object Property*, não mais será uma classe, mas um tipo de dado (ex. *date*, *string*, *boolean*, *int*, *double*, entre outros). Podemos citar,

como exemplo de propriedade do tipo *Datatype Property*, a propriedade *palavras_relacionadas*, que tem como domínio a classe *Problema* e como *range* o tipo *string*. Essa propriedade foi criada com o intuito de armazenar as palavras relacionadas ao problema, inseridas pelo facilitador e que são utilizadas pelo agente AgDP durante a detecção de conversações fora do contexto. O trecho de código OWL referente à propriedade *palavras_relacionadas* é mostrado na Figura 54.

```
<owl:DatatypeProperty rdf:ID="palavras_relacionadas">
  <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Problema"/>
</owl:DatatypeProperty>
```

Figura 54 – Representação da propriedade *palavras_relacionadas* em OWL
Fonte: Dados produzidos pelo autor

O desenvolvimento de todas as outras classes e propriedades que compõem a ontologia do problema seguiu o mesmo procedimento apresentado nas Subseções 7.3.1 e 7.3.2, respectivamente.

7.4 ONTOLOGIA DA APRENDIZAGEM BASEADA EM PROBLEMA

Essa ontologia não está sendo utilizada neste trabalho. Ela foi criada com o intuito de torná-la base de conhecimento de um agente. Pretende-se, futuramente, utilizá-la como base de conhecimento de um agente, permitindo ao mesmo extrair informações da ontologia para monitorar o cumprimento do ciclo da PBL. A seguir, será apresentado o processo de desenvolvimento da ontologia sobre a PBL.

7.4.1 Principais Classes Identificadas

Fazendo uso da análise do domínio da PBL, foram definidas 7 (sete) classes (*Ciclo*, *Estrategias*, *Habilidades*, *Metas*, *Usuario*, *Grupo* e *Problema*), com o intuito de representar

os conceitos gerais, conforme ilustrado na Figura 55. A partir dos conceitos gerais foram definidas e agrupadas suas especificidades, ou seja, suas 20 (vinte) subclasses. Por exemplo, para representar os termos mais específicos da classe *Ciclo* foram definidas 6 (seis) subclasses. Do mesmo modo, para as classes *Estrategias*, *Habilidades*, *Metas* e *Usuario* foram definidas 2, 4, 5 e 4 subclasses, respectivamente. A seguir é apresentada uma breve explanação das principais classes que compõem o domínio da referida ontologia.

7.4.1.1 Ciclo

Como visto na Subseção 3.2.6.1, o ciclo da PBL é composto de 6 (seis) fases: cenário do problema, identificação de fatos, formulação de hipóteses, deficiências de conhecimento, aplicação de novo conhecimento e abstração (HMELO-SILVER, 2004). A Figura 55 ilustra o desenvolvimento da classe *Cenario_do_Problema* na ferramenta Protégé, como forma de exemplificar o desenvolvimento de um conceito presente na ontologia proposta.

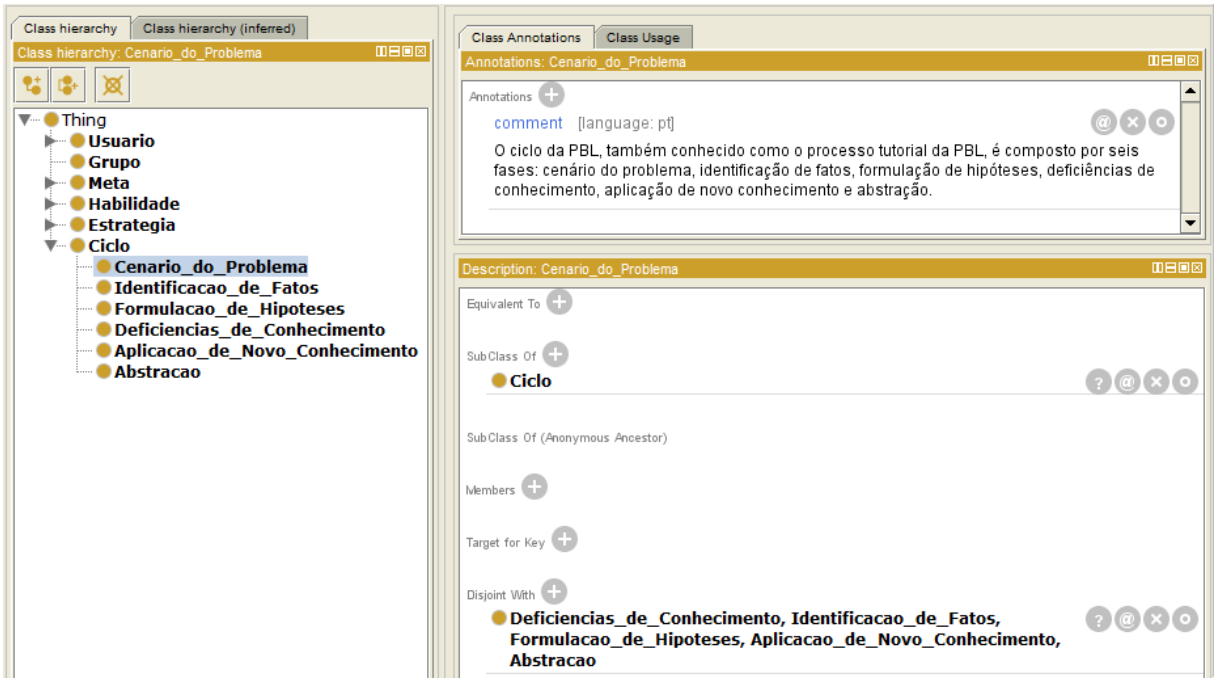

Figura 55 – Representação da classe *Cenario_do_Problema* na ferramenta Protégé
Fonte: Dados produzidos pelo autor

Na Figura 55, temos a definição da classe *Cenario_do_Problema* como sendo subclasse de *Ciclo*. O trecho de código OWL referente a essa definição é exibido na Figura 56.

```
<owl:Class rdf:ID="Cenario_do_Problema">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Ciclo"/>
  </rdfs:subClassOf>
</owl:Class>
```

Figura 56 – Representação da classe *Ciclo* como superclasse de *Cenario_do_Problema*
Fonte: Dados produzidos pelo autor

Na Figura 55, temos todas as classes disjuntas à classe *Cenario_do_Problema*. A Figura 57 ilustra o código OWL referente à propriedade *Disjoint*.

```
<owl:Class rdf:about="#Cenario_do_Problema">
  <owl:disjointWith>
    <owl:Class rdf:ID="Identificacao_de_Fatos"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Formulacao_de_Hipoteses"/>
  </owl:disjointWith>
  <owl:disjointWith
    <owl:Class rdf:ID="Deficiencias_de_Conhecimento"/>
  </owl:disjointWith>
  <owl:disjointWith
    <owl:Class rdf:ID="Aplicacao_de_Novo_Conhecimento"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Abstracao"/>
  </owl:disjointWith>
</owl:Class>
```

Figura 57 – Representação das classes disjuntas à classe *Cenario_do_Problema*
Fonte: Dados produzidos pelo autor

7.4.1.2 Estratégia

Para que processo de aplicação da PBL obtenha sucesso, é necessário também, que o facilitador desempenhe estratégias de ensino específicas, que deverão ser aplicadas aos estudantes, para que esses possam desenvolver competências e/ou superar algumas deficiências, detectadas pelo facilitador durante a resolução de um problema (HMELO-SILVER e BARROWS, 2006). Na ontologia proposta sobre o domínio em questão, foram

abordadas apenas duas estratégias, por suas representações serem possíveis e por essas apresentarem bons resultados quando aplicadas, são elas:

- Questionário: estratégia aplicada com o intuito de incentivar explicações e reconhecimento das limitações do conhecimento;
- Resumo: estratégia aplicada para obter resultados como garantir a representação conjunta do problema, envolver os estudantes menos participativos, ajudar os alunos a sintetizar os dados, revelar fatos que os estudantes consideram importantes, entre outros.

7.4.1.3 Habilidade

Na PBL, o facilitador ajuda os estudantes a desenvolverem habilidades cognitivas necessárias para a resolução de problemas. Portanto, é necessário que os estudantes estabeleçam as suas metas de aprendizagem e estratégias para solucionar os problemas (HMELO-SILVER, 2004). Na ontologia proposta, abordamos quatro habilidades essenciais para o sucesso do processo como um todo, são elas:

- Autoaprendizado: é a maneira como os estudantes adquirem os conhecimentos necessários para a resolução dos problemas;
- Capacidade de resolução de problema: os estudantes devem desenvolver as habilidades necessárias para serem capazes de resolverem os problemas propostos;
- Colaboração: no processo de aplicação da PBL, os estudantes, membros de um grupo, devem trabalhar em conjunto, colaborando uns com os outros na resolução dos problemas;
- Motivação: é de suma importância que os estudantes se mantenham motivados durante o processo da PBL, principalmente pelo fato de serem responsáveis pela sua própria aprendizagem.

7.4.1.4 Meta

A PBL possui várias metas em relação ao estudante (HMELO-SILVER, 2004). Na ontologia proposta, foram abordadas as seguintes metas:

- Construir uma base de conhecimento flexível e extensível: na PBL, é interessante que os alunos construam uma base de conhecimento a partir da resolução dos problemas para auxiliar no entendimento de problemas futuros;
- Desenvolver habilidades efetivas de resolução de problemas: um indicador da efetiva capacidade de resolver problemas é a capacidade de transferência de estratégias de raciocínio para novos problemas;
- Desenvolver habilidades de aprendizagem autodirigida: um dos benefícios da PBL é a sua pretensão de preparar os estudantes ao longo da vida através da sua ênfase na aprendizagem autodirigida;
- Tornar-se colaboradores eficazes: outro objetivo da PBL é ajudar os estudantes a adquirirem habilidades de colaboração;
- Motivar intrinsecamente o estudante a aprender: melhorar a motivação dos estudantes é uma meta da PBL, e, portanto, a motivação intrínseca deve ser reforçada.

7.4.1.5 Grupo

Esta classe foi criada com mesmo intuito da classe mencionada na Subseção 7.3.1.3.

7.4.1.6 Usuário

Na PBL, usuário é qualquer pessoa que participa do processo de aplicação desta teoria de aprendizagem. Na ontologia proposta, definimos as classes *Facilitador*, *Estudante*, *Coordenador* e *Relator* como subclasses da classe *Usuario*.

7.4.2 Especificação das Propriedades

As propriedades de cada classe da ontologia foram especificadas à medida que as classes foram sendo definidas ou reutilizadas. A ontologia resultante possui 32 (trinta e duas) propriedades do tipo *Object Property* e 15 (quinze) propriedades do tipo *Datatype Property*.

Podemos citar, como exemplo do tipo *Object Property*, a propriedade *temEstrategia*, que tem como domínio a classe *Facilitador* e como *range* a classe *Estrategia*. Esta propriedade foi criada para associar todas as estratégias que um dado facilitador pode utilizar. O trecho de código OWL referente à propriedade *temEstrategia* é mostrado na Figura 58.

```
<owl:ObjectProperty rdf:ID="temEstrategia">
  <rdfs:range rdf:resource="#Estrategia"/>
  <rdfs:domain rdf:resource="#Facilitador"/>
</owl:ObjectProperty>
```

Figura 58 – Representação da propriedade *temEstrategia* em OWL
Fonte: Dados produzidos pelo autor

Podemos citar, como exemplo de propriedade do tipo *Datatype Property*, a propriedade *quantidade*, que tem como domínio a classe *Grupo* e como *range* o tipo *int*. Essa propriedade foi criada com o intuito de especificar a quantidade de membros que um dado grupo possui. O trecho de código OWL referente à propriedade quantidade é mostrado na Figura 59.

```
  <owl:DatatypeProperty rdf:ID="quantidade">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Grupo"/>
  </owl:DatatypeProperty>
```

Figura 59 – Representação da propriedade *quantidade* em OWL
Fonte: Dados produzidos pelo autor

O desenvolvimento de todas as outras classes e propriedades que compõem a ontologia sobre a PBL seguiu o mesmo procedimento apresentado nas Subseções 7.4.1 e 7.4.2, respectivamente.

8 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Nos últimos anos, a EaD tem crescido e apresentado bons resultados, embora ainda apresente grandes desafios. Um destes desafios é a aplicação eficaz de teorias de aprendizagem no processo de ensino e aprendizagem nos ambientes que oferecem suporte informatizado a esta atividade. Muitos AVAs enfocam aspectos relacionados à sua funcionalidade, esquecendo a função pedagógica do ambiente, no que tange à aplicação de uma teoria de aprendizagem para o sucesso do processo de ensino e aprendizagem.

A PBL, como visto ao longo deste trabalho, tem apresentado bons resultados com relação ao processo de ensino e aprendizagem, tais como: desenvolvimento do pensamento crítico e criatividade do estudante; aumento de sua capacidade de resolução de problemas; aumento de sua motivação entre outros. Contudo, foi visto também que sua implantação é complexa, pois necessita de uma série de requisitos para seu sucesso, como, por exemplo: o desenvolvimento de todo o seu ciclo, que não é um processo trivial; a definição de um problema de natureza complexa e mal estruturada; a responsabilidade de autoaprendizado do estudante; a necessidade de o facilitador conhecer as técnicas da PBL; entre outros. A sua implantação na EaD é uma tarefa ainda mais complicada, pois o facilitador não está presente fisicamente nas sessões de interação e nem sempre possui informações sobre como o processo está sendo realizado ou se há algum problema a ser solucionado.

O uso de agentes de software tem se destacado como auxílio neste problema, facilitando o gerenciamento do ambiente e, ainda mais importante, o cumprimento de uma teoria de aprendizagem eficaz para o sucesso do processo de ensino e aprendizagem no ambiente. O uso de ontologias em ambientes de aprendizagem também tem se tornado comum, pois elas representam conceitos e modelos inerentes ao ambiente no qual são usadas, podendo modelar o conhecimento do domínio e também auxiliar no processo de aprendizagem.

Neste trabalho, foi descrita uma arquitetura baseada em um agente pedagógico animado e outros seis agentes de software para auxiliar na aplicação correta da teoria de aprendizagem PBL. Nesta abordagem, o agente pedagógico animado tem o papel de tutor, acompanhando os estudantes durante o processo de aquisição de conhecimento, além do papel motivacional, que é de suma importância ao se tratar de AVAs. Os outros agentes de software foram criados com o intuito de contornar possíveis problemas de colaboração que possam vir a acontecer durante a aplicação da PBL, provendo suporte ao facilitador para resolver esses

problemas. Também foi mostrado como o AVA Moodle foi adaptado para suportar o processo de aplicação da PBL.

Como contribuição deste trabalho podemos destacar o desenvolvimento de uma arquitetura multiagente para apoiar o AVA Moodle, seguindo os requisitos de uma teoria comprovadamente eficaz, ou seja, a PBL. Essa arquitetura multiagente foi criada com o intuito de aperfeiçoar a aplicação da PBL principalmente nos seguintes aspectos: detecção de estudantes passivos, detecção de conversações fora do contexto, formação de grupos e recomendação de OAs sensível ao contexto do estudante. A arquitetura apresentada neste trabalho irá facilitar a tarefa de ensino e aprendizagem quando a PBL for utilizada como teoria de aprendizagem. Ela irá também oferecer mais subsídio para que o facilitador possa acompanhar o processo de aplicação da PBL, possibilitando uma intervenção mais eficaz quando problemas de colaboração forem detectados.

Como trabalhos futuros, pretende-se: (i) abordar outras metas relacionadas ao auxílio no cumprimento da aplicação da PBL, conforme apresentado em (PONTES, 2010), principalmente o monitoramento do cumprimento das etapas, ou seja, incluir outro agente de software para verificar se o ciclo da PBL está sendo cumprimento corretamente. Neste caso, será utilizada a ontologia sobre a PBL, proposta neste trabalho, como base de conhecimento do agente responsável por essa tarefa; (ii) abordar as tarefas que não foram contempladas neste trabalho, como, por exemplo, gerenciar prazos do grupo, selecionar e sugerir estratégias pedagógicas e detectar estagnação do processo, que é um problema que ocorre quando os estudantes não possuem conhecimentos suficientes para continuar o processo, ou seja, o grupo fica “perdido”; e (iii) realizar um estudo de caso como forma de validar a eficácia da solução apresentada neste trabalho. Esse estudo de caso está integrado ao projeto intitulado “Um Agente Pedagógico 3D de Apoio a Estudantes de Medicina na Resolução de Problemas na Área da Oncologia Utilizando a Aprendizagem Baseada em Problema” aprovado pelo COHM (Centro de Oncologia e Hematologia de Mossoró). Pretende-se realizar o estudo de caso com estudantes do curso de medicina, sendo essa validação apoiada pelo COHM. Nessa ocasião, os estudantes utilizarão a ferramenta, com a solução integrada, para resolverem problemas relacionados à área de Oncologia. Dessa forma, será possível avaliar a solução proposta e verificar se a ferramenta provê um auxílio eficaz aos estudantes do curso de medicina na resolução de problemas da área de Oncologia.

8.1 PUBLICAÇÕES

Os resultados das pesquisas desenvolvidas durante a realização deste trabalho, até o momento, resultaram nas seguintes publicações:

- Em periódicos:
  - ✓ (Aceito para publicação) FONTES, L. M. O.; MENDES NETO, F. M.; DINIZ, F. A.; CARLOS, D. G.; JÁCOME JÚNIOR, L.; SILVA, L. C. N. An Animated Pedagogical Agent to Support Problem-Based Learning. IEEE-RITA, 2013.
  - ✓ DINIZ, F. A.; MENDES NETO, F. M.; LIMA JÚNIOR, F. C.; FONTES, L. M. O. RedFace: Um Sistema de Reconhecimento Facial para Identificação de Estudantes em um Ambiente Virtual de Aprendizagem. RENOTE. Revista Novas Tecnologias na Educação, v. 10, p. 1-11, 2012.
  - ✓ FONTES, L. M. O.; MENDES NETO, F. M.; DINIZ, F. A.; CARLOS, D. G.; JACOME JUNIOR, L.; SILVA, L. C. N. Um Agente Pedagógico Animado de Apoio à Aprendizagem Baseada em Problema. IEEE-RITA, v. 7, p. 181-188, 2012.
  - ✓ FONTES, L. M. O.; MENDES NETO, F. M.; PONTES, A. A. A. A. Multiagent System to Support Problem-Based Learning. Creative Education, v. 02, p. 452-457, 2011.
  - ✓ FONTES, L. M. O.; MENDES NETO, F. M.; PONTES, A. A. A. Um Sistema Multiagente de Apoio à Aprendizagem Baseada em Problema. Revista Brasileira de Computação Aplicada, v. 3, p. 103-117, 2011.
- Em conferências:
  - ✓ FONTES, L. M. O.; MENDES NETO, F. M.; DINIZ, F. A.; CARLOS, D. G.; JÁCOME JÚNIOR, L.; SILVA, L. C. N. . UMA ARQUITETURA MULTIAGENTE DE APOIO AO PROCESSO DE ENSINO E APRENDIZAGEM NO MOODLE BASEADO NA PBL. In: VIII International Conference on Engineering and Computer Education - ICECE 2013, 2012, Luanda. Proceedings of VIII International Conference on Engineering and Computer Education - ICECE 2013. Luanda: COPEC, 2013.

- ✓ DINIZ, F. A.; MENDES NETO, F. M.; LIMA JÚNIOR, F. C.; FONTES, L. M. O. A Facial Recognition System for Students Identification in a Virtual Learning Environment. In: International Conference on Social Science and Health (ICSSH 2013), 2013, Los Angeles, CA, USA. Advances in Education Research. Los Angeles: CPCI-SSH (ISSHP), 2013.
- ✓ DINIZ, F. A.; MENDES NETO, F. M.; LIMA JÚNIOR, F. C.; FONTES, L. M. O. A Facial Recognition System for Students Identification in a Virtual Learning Environment Composed by Pedagogical Agents. In: World Conference on Information Systems and Technologies (WorldCIST'13), 2013, Algarve. Proceedings of 2013 World Conference on Information Systems and Technologies, 2013.
- ✓ DINIZ, F. A.; MENDES NETO, F. M.; LIMA JÚNIOR, F. C.; FONTES, L. M. O. A Facial Recognition System Online for Students Identification in a Virtual Learning Environment. In: 5th International Conference on Computer Supported Education - CSED 2013, 2013, Aachen, Germany. Proceedings of 5th International Conference on Computer Supported Education. Aachen, Germany: CSEDU, 2013.
- ✓ FONTES, L. M. O.; MENDES NETO, F. M.; DINIZ, F. A.; CARLOS, D. G.; JÁCOME JÚNIOR, L.; SILVA, L. C. N. Um Agente Pedagógico Animado de Apoio à Aprendizagem Baseada em Problema Utilizando o Moodle. In: Workshop sobre Avaliação e Acompanhamento da Aprendizagem em Ambientes Virtuais, XXIII Simpósio Brasileiro de Informática na Educação, 2012, Rio de Janeiro. Anais do XXIII Simpósio Brasileiro de Informática na Educação. Rio de Janeiro: UFRJ, 2012.
- ✓ DINIZ, F. A.; MENDES NETO, F. M.; LIMA JÚNIOR, F. C.; FONTES, L. M. O. Um Sistema de Reconhecimento Facial Aplicado a um Ambiente Virtual de Aprendizagem Compostos por Agentes Pedagógicos. In: VIII International Conference on Engineering and Computer Education - ICECE 2013, 2012, Luanda. Proceedings of VIII International Conference on Engineering and Computer Education - ICECE 2013. Luanda: UniPiaget, 2012.
- ✓ DINIZ, F. A.; MENDES NETO, F. M.; LIMA JÚNIOR, F. C.; FONTES, L. M. O. RedFace: Um Sistema de Reconhecimento Facial para Identificação de Estudantes em um Ambiente Virtual de Aprendizagem. In:

XX Ciclo de Palestras Novas Tecnologias na Educação, 2012, Porto Alegre. Anais do XX Ciclo de Palestras Novas Tecnologias na Educação. Porto Alegre: CINTED, 2012.

- ✓ FONTES, L. M. O.; MENDES NETO, F. M.; PONTES, A. A. A.; CAMPOS, G. A. L. de. An Agent-Based Architecture for Supporting the Workgroups Creation and the Detection of Out-of-Context Conversation on Problem-Based Learning in Virtual Learning Environments. In: ACM Symposium on Applied Computing, Track on Intelligent, Interactive and Innovative Learning Environments, 2011, TaiChung, Taiwan. Proceedings of the 2011 ACM Symposium on Applied Computing. New York: ACM, 2011.
- ✓ FONTES, L. M. O.; MENDES NETO, F. M.; PONTES, A. A. A. OntoPBL: Uma Ontologia de Domínio sobre Aprendizagem Baseada em Problema. In: XXII Simpósio Brasileiro de Informática na Educação - SBIE 2011, 2011, Aracaju. Anais do XXII Simpósio Brasileiro de Informática na Educação. Aracaju: UFS, 2011.
- ✓ FONTES, L. M. O.; MENDES NETO, F. M.; PONTES, A. A. A. Uma Ontologia de Domínio para Aprendizagem Baseada em Problema. In: II Workshop Técnico-Científico de Computação (II WTCC), 2011, Mossoró. Anais do II Workshop Técnico-Científico de Computação. Mossoró: UFERSA, 2011.
- ✓ PONTES, A. A. A.; MENDES NETO, F. M.; FONTES, L. M. O.; CAMPOS, G. A. L. de. Uma Arquitetura de Agentes para Auxílio à Formação de Grupos e Detecção de Conversações Fora do Contexto na Aprendizagem Baseada em Problemas em Ambientes Virtuais de Aprendizagem. In: V Workshop de Arquiteturas Pedagógicas para Suporte à Educação a Distância Mediada pela Internet, XXI Simpósio Brasileiro de Informática na Educação, 2010, João Pessoa. Anais do XXI Simpósio Brasileiro de Informática na Educação. João Pessoa: UFPB, 2010.
- ✓ FONTES, L. M. O.; PONTES, A. A. A.; MENDES NETO, F. M. Uma Arquitetura de Agentes para Detecção de Conversações Fora do Contexto na Aprendizagem Baseada em Problemas em Ambientes Virtuais de Aprendizagem. In: Escola Potiguar de Computação e suas Aplicações -

EPOCA 2010, 2010, Mossoró. Anais da Escola Potiguar de Computação e suas Aplicações - EPOCA 2010. Mossoró: UERN, 2010.

- ✓ FONTES, L. M. O.; PONTES, A. A. A.; MENDES NETO, F. M.; CAMPOS, G. A. L. de. Uma Arquitetura de Agentes para Auxílio à Formação de Grupos na Aprendizagem Baseada em Problemas em Ambientes Virtuais de Aprendizagem. In: Escola Potiguar de Computação e Suas Aplicações - EPOCA 2010, 2010, Mossoró. Anais da Escola Potiguar de Computação e Suas Aplicações - EPOCA 2010. Mossoró: UERN, 2010.

REFERÊNCIAS

ARROYO, I.; WOOLF, B. P.; COOPER, D. G. The Impact of Animated Pedagogical Agents on Girls' and Boys' Emotions, Attitudes, Behaviors and Learning. In: INERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES. Athens: IEEE, p. 506-510, 2011.

ARTERO, A. O. **Inteligência Artificial - Teoria e Prática**. 1ª. ed. São Paulo: Livraria da Física, 2009.

AUSUBEL, D. P. **The psychology of meaningful verbal learning:** An introduction to school learning. New York: Grune & Stratton, 1963.

AZEVEDO, E.; CONCI, A. **Computação Gráfica: teoria e prática**. Rio de Janeiro, Elsevier, 2003.

AZEVEDO, H. J. S.; SCALABRIN, E. E. A Human Collaborative Learning Environment Using Intelligent Agents. In: LIN, F. O. (Ed.). DESIGNING DISTRIBUTED LEARNING ENVIRONMENTS WITH INTELLIGENT SOFTWARE AGENTS. United States of America: Information Science Publishing, p. 1-32, 2005.

BELIFEMINE, F.; CAIRE, G.; GREENWOOD, D. **Developing multi-agent systems with JADE**. Liverpool, Inglaterra: John Wiley & Sons, Ltd, 2007.

BERBEL, N. A problematização e a aprendizagem baseada em problemas: diferentes termos ou diferentes caminhos? **Interface**, SciELO Brasil, v. 2, n. 2, 1998.

BERCHT, M. **Em Direção a Agentes Pedagógicos com Dimensões Afetivas**. 2001. 152 f. Tese (Doutorado em Ciência da Computação) - Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2001.

BITTENCOURT, I.; BEZERRA, C.; NUNES, C.; COSTA, E.; TADEU, M.; NUNES, R.; COSTA, M. SILVA, A. **Ontologia para Construção de Ambientes Interativos de Aprendizagem**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO - SBIE, 17., Brasília: [s.n.], p. 559-568, 2006.

BLENDER. Blender Foundation. **Site oficial do Blender**, 2013. Disponível em: <http://www.blender.org/>. Acesso em: 08 jan. 2013.

BREMGARTNER, V.; NETTO, J. F. M. Auxílio Personalizado a Estudantes em Ambientes Virtuais de Aprendizagem Utilizando Agentes e Competências. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE, 22. Aracaju, 2011.

BRITO, A. **Blender 3D: Guia do Usuário**. 4. ed. Novatec, 2010.

BRÓLIO, M.; OMAR, N.; FRANGO, I.; PIMENTEL, E. Modelagem de um Ambiente de Apoio à Avaliação Continuada Construído sob Ontologia. In: SÁNCHEZ, J. (Ed.). **Nuevas Ideas en Informática Educativa**, v. 2, p. 64-71, 2006.

CASTAÑON, G. Construtivismo e Ciências Humanas. **Ciências & Cognição**, v. 5, p. 36-49, 2005.

CASTAÑON, G. Construtivismo, Inatismo e Realismo: compatíveis e complementares. **Ciências & Cognição**, v. 10, p. 115-131, 2007.

CASTRO, J.; GIORGINI, P.; KOLP, M.; MYLOPOULOS, J. Tropos: A requirements-driven methodology for agent-oriented software. In: HENDERSON-SELLERS, B.; GIORGINI, P. Agent-Oriented Methodologies. Hershey, PA: IDEA Group Publishing, 2005. Cap. II, p. 20-45.

CERVENKA, R.; TRENCANSKY, I. **AML, The Agent Modeling Language:** A Comprehensive Approach to Modeling Multi-Agent Systems. Basel, Suíca: Birkhauser Verlag, v. Whitestein Series in Software Agent Technologies and Automatic Computing, 2007.

CHENG, Y. M.; CHEM, L. S.; HUANG, H. C.; WENG, S. F.; CHEN, Y. G.; LIN, C. H. Building a General Purpose Pedagogical Agent in a Web-Based Multimedia Clinical Simulation System for Medical Education. **IEEE Transactions on Learning Technologies**, v. 3, n. 1, 2009.

CONSSENTINO, M. From requirements to code with the passi methodology. In: HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Hershey, PA: IDEA Group Publishing, 2005. Cap. IV, p. 79-106.

COUTINHO, C. P.; BOTTENTUIT JUNIOR, J. B. Collaborative learning using Wiki: a pilot study with master students in educational technology in Portugal. In: PROCEEDINGS OF WORLD CONFERENCE ON EDUCATIONAL MULTIMEDIA, HYPERMEDIA AND TELECOMMUNICATIONS. Vancouver, Canadá: [s.n.], p. 1786-1791, 2007.

DELOACH, S.; KUMAR, M. Multiagent systems engineering: An overview and case study. In: HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Hershey, PA: IDEA Group Publishing, 2005. Cap. XI, p. 317-340.

DICKINSON, I. Jena Ontology API. **Site do *Framework* Jena**, 2009. Disponível em: <http://jena.sourceforge.net/ontology/index.html>. Acesso em: 08 jan. 2013.

DILLENBOURG, P. What do you mean by collaborative learning? In: DILLENBOURG, P. (Ed.). Collaborative-learning: Cognitive and Computational Approaches. Oxford: Elsevier, p. 1-19, 1999.

DONGLAI, F.; GOUXI, C.; QIUXIANG, Y. A Robust Software Watermarking for jMonkey Engine Programs. In: INTERNATIONAL FORUM ON INFORMATION TECHNOLOGY AND APPLICATIONS – IFITA. Taiyuan, 2010.

DUEZ-RODRIGUEZ, H.; MORALES-LUNA, G.; OLMEDO-AGUIRRE, J. O. Ontology Based Knowledge Retrieval. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE. Mexico: IEEE, 2008.

FARIA, E. S. J.; VILELA, J. M.; COELLO, J. M. A. Um Sistema de Aprendizado Colaborativo de Programação Baseado em Agentes chamado Learn In Group. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO – WEI, 13. [s.l.]: p. 2278-2290, 2005.

FELIX, Z. C.; TEDESCO, P. A. Smart Chat Group: Ferramenta Ciente de Contexto para Formação de Grupos-Versão Final. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE, 19. Fortaleza, 2008.

FERREIRA, T. B.; OTSUKA, J. L.; ROCHA, H. V. Interface para Auxílio à Avaliação Formativa no Ambiente TelEduc. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE, 15. Rio de Janeiro, 2003.

FIPA. Welcome to the Foundation for Intelligent Physical Agents. **Site Oficial do Padrão FIPA**, 2011. Disponível em: <http://www.fipa.org/>. Acesso em: 08 jan. 2013.

FOLEY, J. D.; DAM, A. V.; FEINER, S. K.; HUGHES, J. F.; PHILLIPS, R. L. **Introduction to computer graphics**. New York: Addison-Wesley, 1996.

FROZZA, R.; SILVA, A. A. K.; LUX, B.; CRUZ, M. E. J. K.; BORIN, M. Dóris 3D: Agente Pedagógico baseado em Emoções. In: ANAIS DO SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE, 20. Florianópolis, 2009.

FROZZA, R.; SILVA, A. A. K.; SCHREIBER, J. N. C.; LUX, B.; MOLZ, K. W.; KIPPER, L. M.; BORIN, M. P.; CARVALHO, A. B.; BAIERLE, J. L.; SAMPAIO, L. Agentes Pedagógicos Emocionais atuando em um Ambiente Virtual de Aprendizagem. **Renote - Novas Tecnologias na Educação**, v. 9, n. 1, 2011.

FUNK, S.; AYMONE, J. L. F. Proposta de Diretrizes para o Processo Criativo do Design Virtual de Embalagens. **Design & Tecnologia**, v. 1, n. 2, p. 55-68, 2010.

GARIJO, F. J.; GÓMEZ-SANZ, J. J.; MASSONET, P. The MESSAGE Methodology for Agent-Oriented Analysis and Design. In: HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Hershey, PA: IDEA Group Publishing, 2005. Cap. VIII, p. 203-234.

GAVA, T. B. S.; MENEZES, C. S. Uma Ontologia de Domínio para a Aprendizagem Cooperativa. In: ANAIS DO SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO - SBIE, 14. Rio de Janeiro, 2003.

GIRAFFA, L. M. M. **Uma arquitetura de Tutor Utilizando Estados Mentais**. 1999. 151 f. Tese (Doutorado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 1999.

GOMES, A. K.; BERNADINI, F. C.; MONARD, M. C.; BATISTA, G. E. A. P. A. **Uma Sintaxe Padrão Prolog para Classificadores Simbólicos**. São Carlos: n. 154, 2002.

GONZÁLEZ, L. A. G. **Um Modelo Conceitual para Aprendizagem Colaborativa Baseada na Execução de Projetos pela Web**. 2005. 254 f. Tese (Doutorado em Engenharia da Computação) – Departamento de Engenharia de Computação e Sistemas Digitais. Escola Politécnica, Universidade de São Paulo, 2005.

GPL. The GNU General Public License v3.0 - GNU Project - Free Software Foundation (FSF). **GNU General Public License**, 2007. Disponível em: <http://www.gnu.org/copyleft/gpl.html>. Acesso em: 08 jan. 2013.

GREENE, R.; ROGERS, G. L. Justifying Core Faculty Assessment of Students' Clinical Performance Using Cognitive Flexibility Theory: A Case Example. **The Internet Journal of Allied Health Sciences and Practice**, v. 4, n. 3, 2006.

GRUBER, T. R. A translation approach to portable ontologies. **Knowledge Acquisition**, v. 5, n. 2, p. 199-220, 1993.

HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Guernsey, Reino Unido: IGI Global, 2005.

HMELO-SILVER, C. E.; BARROWS, H. S. Goals and strategies of a problem-based learning facilitator. **The Interdisciplinary Journal of Problem-based Learning**, v. 1, n. 1, p. 21-39, 2006. Disponível em: <http://docs.lib.purdue.edu/ijpbl/vol1/iss1/4/>. Acesso em: 8 abr. 2010.

HMELO-SILVER, C. E. Problem-Based Learning: What and How Do Students Learn? **Educational Psychology Review**, v. 16, n. 3, 2004.

HUTCHINS, E. **Cognition in the Wild**. Cambridge, MA: MIT Press, 1995.

IGLESIAS, C. A.; GARIJO, M. The Agent-Oriented Methodology MAS-CommonKADS. In: HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Hershey, PA: IDEA Group Publishing, 2005. Cap. III, p. 46-78.

ISOTANI, S.; MIZOGUCHI, R. Planejamento e Análise de Sessões Colaborativas Utilizando Teorias de Aprendizagem e Ontologias. Revista Brasileira de Informática na Educação, v. 15, n. 2, 2007.

JAQUES, P. A.; ANDRADE, A.; JUNG, J., BORDINI, R.; VICARI, R. Using pedagogical agents to support collaborative distance learning. In: PROCEEDINGS OF THE CONFERENCE ON COMPUTER SUPPORT FOR COLLABORATIVE LEARNING: FOUNDATIONS FOR A CSCL COMMUNITY, p. 546–547. International Society of the Learning Sciences, 2002.

JAQUES, P. A.; VICARI, R. Estado da Arte em Ambientes Inteligentes de Aprendizagem que Consideram a Afetividade do Aluno. **Informática na Educação: Teoria e Prática**, v. 8, n. 1, p. 15-38, 2005.

JUNKER, G. **Pro OGRE 3D Programming**. Berkely, Apress, 2006.

KUMAR, S.; GANKOTIYA, A. K.; DUTTA, K. A Comparative Study of Moodle with other e-Learning Systems. In: INTERNATIONAL CONFERENCE ON ELECTRONICS COMPUTER TECHNOLOGY – ICECT, 3. Kanyakumari: IEEE, 2011.

KYLANDER, K.; KYLANDER, O. S. **GIMP User's Manual: The Complete Guide to Gimp**. Scottsdale, Coriolis Group, 1999.

LABIDI, S.; SOUZA, C. M. NASCIMENTO, E. NetClass: Cooperative Learner Modeling in a Web-Based Environment. In: INTERNATIONAL CONFERENCE ON COMPUTER BASED LEARNING IN SCIENCE, 6. Nicosia, Cyprus: University of Cyprus, 2003.

LIMA, M. R. C.; LABIDI, S.; BASTOS FILHO, O. C.; FONSECA, L. C. C. Aprendizagem cooperativa e o problema de formação de grupos. **Renote - Novas Tecnologias na Educação**, v. 3, n. 1, 2005.

LINDEN, R. **Algoritmos Genéticos - Uma importante ferramenta da Inteligência Computacional**. 2ª. ed. Rio de Janeiro, RJ: Brasport, 2008.

MAIA, N. Ensino a distância cresce no País. **O POVO Online**, Fortaleza, CE, 2011. Disponível em: <http://www.opovo.com.br/app/opovo/empregos/2011/10/01/noticiaempregosjornal,2307499/ensino-a-distancia-cresce-no-pais.shtml>. Acesso em: 08 jan. 2013.

MANDAL, S. **Problem Based Learning Tool as a Plug-in for Moodle**. 2011. 111 f. Dissertação (Mestrado em Ciência da Computação e Engenharia) - Indian Institute of Technology, Bombay, 2011.

MENDES NETO, F. M. **E-grupo: Um ambiente para suporte à aprendizagem colaborativa baseada na web**. 2000. 147 f. Dissertação (Mestrado em Informática) - Universidade Federal da Paraíba, Campina Grande, 2000.

MOISIL, I.; PAH, I.; BARBAT, B. POPA, E. M. Socio-cultural modelling of the student as the main actor of a virtual learning environment. In: PROCEEDINGS OF THE WSEAS, INTERNATIONAL CONFERENCE ON MATHEMATICAL METHODS AND COMPUTATIONAL TECHNIQUES IN ELECTRICAL ENGINEERING, 8. Bucharest: [s.n.], 2006.

MOODLE. About Moodle. **Site oficial do Moodle**, 2011. Disponível em: <http://docs.moodle.org/21/en/About_Moodle>. Acesso em: 02 dez. 2011.

MORAIS II, M. J. D. O. **MAS-CommonKADS+: Uma Extensão à Metodologia Mas-CommonKADS para Suporte ao Processo Detalhado de Sistemas Multiagentes Racionais**. Dissertação de Mestrado. Universidade Estadual do Ceará - UECE. Fortaleza, CE. 2010.

MYSQL. MySQL: The world's most popular open source database. **Site oficial do MySQL**, 2011. Disponível em: <http://www.mysql.com/>. Acesso em: 08 jan. 2013.

NOZAWA, E. H.; OLIVEIRA, E. H. T.; COSTA, M. L. F.; SANTOS, E. R.; ISOTANI, S.; CASTRO JÚNIOR, A. N.; VICARI, R. M. Modelagem de um Ambiente de Aprendizagem Adaptativo Baseado em Ontologias. In: ANAIS DO II ESCOLA REGIONAL DE INFORMÁTICA - ERIN. Manaus, 2010.

OLIVEIRA, H. T. A.; GADELHA, R. N. S.; AZEVEDO, R. R.; DELFINO JÚNIOR, J. B.; DIAS, G. A.; FREITAS, F. Dr. Pierre: Um Chatterbot com Intenção e Personalidade Baseado em Ontologias para Apoiar o Ensino de Psiquiatria. In: ANAIS DO SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE, 21. João Pessoa, 2010.

OWL. OWL: Web Ontology Language. **Site Oficial do OWL**, 2013. Disponível em: < http://www.w3.org/TR/owl-features/>. Acesso em: 08 jan. 2013.

PADGHAM, L.; WINIKOFF, M. Prometheus: A practical agent-oriented methodology. In: HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Hershey, PA: IDEA Group Publishing, 2005. Cap. V, p. 107-135.

PAVÓN, J.; GOMEZ-SANZ, J. J.; FUENTES, R. The INGENIAS Methodology and Tools. In: HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Hershey, PA: IDEA Group Publishing, 2005. Cap. IX, p. 236-276.

PETROLI NETO, S. Computação Evolutiva: desvendando os algoritmos genéticos. **Ubiquidade - Revista de Estudos de Tecnologia da Informação e Comunicações**, v. 1, n. 1, p. 34-45, 2011.

PHP. PHP: Hypertext Preprocessor. **Site Oficial do PHP**, 2011. Disponível em: <http://www.php.net/>. Acesso em: 08 jan. 2013.

PIAGET, J. **Psicologia e epistemologia: por uma teoria do conhecimento**. Rio de Janeiro: Forense Universitária, 1973.

PONTES, A. A. A. **Uma Arquitetura de Agentes para Suporte à Colaboração na Aprendizagem Baseada em Problemas em Ambientes Virtuais de Aprendizagem**. 2010. 114 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual do Rio Grande do Norte, Universidade Federal Rural do Semi-Árido, Mossoró, 2010.

PROTÉGÉ. **Site Oficial do Protégé**, 2013. Disponível em: < http://protege.stanford.edu/>. Acesso em: 08 jan. 2013.

REATEGUI, E.; MORAES, M. Agentes Pedagógicos Animados: Concepção, Desenvolvimento e Aplicação. In: ANAIS DO SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE, 17. Brasília: SBC, 2006.

RIBAS, A.; MOURA, M. de. Abordagem sociocultural: algumas vertentes e autores. **Psicologia em Estudo,** SciELO Brasil, v. 11, n. 1, p. 129-138, 2006.

RISSOLI, V. R. V; GIRAFFA, L. M. M.; MARTINS, J. P. Sistema Tutor Inteligente baseado na Teoria da Aprendizagem Significativa com acompanhamento Fuzzy. **Informática na educação: teoria & prática**, v. 9, n. 2, 2006.

ROGERS, Y. **A Brief Introduction to Distributed Cognition**. Brighton, UK, 1997. Disponível em: <http://mcs.open.ac.uk/yr258/papers/dcog/dcog-brief-intro.pdf>. Acesso em: 08 jan. 2013.

ROGERS, Y. **Distributed Cognition and Communication**. Encyclopedia of Language and Linguistics. 2. ed. Elsevier: Oxford, p. 181-202, 2006.

ROUGEMAILLE, S.; ARCANGELI, J. P.; GLEIZES, M. P.; MIGEON, F. ADELFE Design, AMAS-ML in Action. In: ARTIKIS, A. A. P. G. A. V. L. **Engineering Societies in the Agents World IX**. Berlin, Heidelberg: Springer-Verlag, 2009. p. 105-120.

RUSSEL, S. J.; NORVIG, P. **Artificial Intelligence**: A Modern Approach. 2ª Edição. ed. Upper Saddle River, Nova Jersey: Prentice Hall, 2003.

SANTOS, C. T.; DAHMER, A.; FROZZA, R.; GASPARY, L. P. DÓRIS - Um Agente de Acompanhamento Pedagógico em Sistemas Tutores Inteligentes. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE, pp. 97-105, 2001.

SAVERY, J. R. Overview of Problem-based Learning: Definitions and Distinctions. **The Interdisciplinary Journal of Problem-based Learning**, v. 1, n. 1, p. 9-20, 2006.

SCHREIBER, G.; AKKERMANS, H.; ANJEWIERDEN, A.; HOOG, R.; SHADBOLT, N.; VELDE, W. V.; WIELINGA, B. **Knowledge Engineering and Management - The CommonKADS Methodology**. Cambridge, MA: MIT Press, 2000.

SCORM. ADL SCORM. **Site Oficial do SCORM**, 2013. Disponível em <http://www.adlnet.gov/capabilities/scorm>. Acesso em: 08 jan. 2013.

SILVA, L. C. N. **MobiLE - Um Ambiente Multiagente de Aprendizagem Móvel para Apoiar a Remomendação Ubíqua de Objetos de Aprendizagem**. 2012. 108 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual do Rio Grande do Norte, Universidade Federal Rural do Semi-Árido, Mossoró, 2012.

SILVA, T. G.; BERNARDI, G. Cal: um Agente Pedagógico Animado para Apoio em um Objeto de Aprendizagem para o Ensino de Matemática. In: ANAIS DO SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE, 20. Florianópolis: SBIE, 2009.

SILVEIRA, S. R.; BARONE, D. A. C. Formação de grupos colaborativos em cursos a distância via web: um estudo de caso utilizando técnicas de inteligência artificial. **Revista Brasileira de Informática na Educação**, v. 14, n. 2, 2006.

SOLIMAN, M.; GUETL, C. Intelligent Pedagogical Agents in Immersive Virtual Learning Environments: A Review. In: PROCEEDINGS OF THE INTERNATIONAL CONVENTION - MIPRO. Opatija, Croatia, p. 827-832, 2010.

SONG, H.; SI, G.; YANG, L.; LIANG, H.; ZHANG, L. Using Project-Based Learning and Collaborative Learning in Software Engineering Talent Cultivation. In: INTERNATIONAL CONFERENCE ON TRUST, SECURITY AND PRIVACY IN COMPUTING AND COMMUNICATIONS, 10. Changsha, China: IEEE, p. 1288 - 1293, 2012.

SPIRO, R. J.; COLLINS, B. P.; THOTA, J. J.; FELTOVICH, P. J. Cognitive Flexibility Theory: Hypermedia for Complex Learning, Adaptive Knowledge Application, and Experience Acceleration. **Educational Technology**, v. 43, n. 5, p. 5-10, 2003.

STARUML. StarUML - The Open Source UML/MDA Platform. **Site oficial do StarUML**, 2011. Disponível em: <http://staruml.sourceforge.net/en/>. Acesso em: 08 jan. 2013.

STROBEL, J.; VAN BARNEVELD, A. When is PBL More Effective? A Meta-synthesis of Meta-analyses Comparing PBL to Conventional Classrooms. **Interdisciplinary Journal of Problem-based Learning**, v. 3, n. 4, 2009.

TOTTEN, C. **Game Character Creation with Blender and Unity**. Sybex, 2012.

VÉRAS, D.; BRAZ, L. M.; BITTENCOURT, I.; TOLEDO, A.; COSTA, E. Representando Ambientes Educacionais de Hipermídia Adaptativa Através de Ontologias. In: ANAIS DO WORKSHOP DE INFORMÁTICA NA ESCOLA – WIE. Belém, 2008.

VERONESE, G.; CORREA, A.; WERNER, C.; JEZINI NETTO, F. **ARES: Uma Ferramenta de Engenharia Reversa Java-UML**. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE – SBES, 16. Gramado: p. 347-352, 2002.

VIZCAÍNO, A. A Simulated Student Can Improve Collaborative Learning. **International Journal of Artificial Intelligence in Education**, v. 15, n. 1, p. 3-40, 2005.

VYGOTSKI, L. S. **Formação social da mente:** o desenvolvimento dos processos psicológicos superiores. 6. ed. São Paulo: Martins Fontes, 1998.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. 1ª. ed. Chihester, Inglaterra: John Wiley & Sons, 2002.

YU, E. **Modelling strategic relationships for process reengineering**. Tese de Doutorado. University of Toronto. Toronto, Canadá. 1995.

ZAMBONELLI, F.; JENNINGS, N.; WOOLDRIDGE, M. Multi-agent systems as computational organizations: The gaia methodology. In: HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Hershey, PA: IDEA Group Publishing, 2005. Cap. VI, p. 136-171.

ZINI, E. D. O. C. **Algoritmo Genético Especializado na Resolução de Problemas com Variáveis Contínuas e Altamente Restritos**. 2009. 149 f. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Estadual Paulista, Ilha Solteira, SP, 2009.

APÊNDICE A

A seguir é apresentado o código fonte do comportamento do AgGG em Prolog.

```
% Para realizar uma consulta:
% ?- agente_forma_grupo('Percepcoes.txt','Acoes.txt').

   agente_forma_grupo(Percepcoes,Acoes):-
       ver(Percepcoes),
       ação(Acoes).

/*****************************************************************************
***************************/

   ver(Percepcoes):-
   see(Percepcoes),
   vê_quantidade_alunos,
        vê_parâmetros_similaridade,
   vê_situação_desejada_e_Importâncias,
   vê_situações_correntes,
   seen.

   :- dynamic quantidade_alunos/1.

   vê_quantidade_alunos:-
   read(Q),
   asserta(quantidade_alunos(Q)).

   :- dynamic parâmetros_similaridade/2.


   vê_parâmetros_similaridade:-
   read(P),
   read(N),
   asserta(parâmetros_similaridade(P,N)).

   :- dynamic situação_desejada/1.
   :- dynamic Importâncias/1.

   vê_situação_desejada_e_Importâncias:-
        read(SD),
   read(ValoresImportância),
   quantizar(SD,NiveisDesejados),
   asserta(situação_desejada(NiveisDesejados)),
   asserta(Importâncias(ValoresImportância)).

   :- dynamic aluno_candidato/3.

   vê_situações_correntes:-
   read(Aluno),
        Aluno = [ID_Aluno, Nome, SC],!,
   quantizar(SC,NiveisCorrentes),
```

```
   asserta(aluno_candidato(ID_Aluno,Nome,NiveisCorrentes)),
   vê_situações_correntes.
   vê_situações_correntes.

/*************************************************************************
**************************/

   quantizar([Char],[LReais]):-
   !,converter(Char,LReais).
   quantizar([Char|RLChar],[LReais|RLReais]):-
   converter(Char,LReais),
   quantizar(RLChar,RLReais).

   converter(b,[1.0, 1.0, 0.9, 0.7, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0]).
   converter(m,[0.0, 0.0, 0.0, 0.8, 0.9, 1.0, 0.7, 0.0, 0.0, 0.0, 0.0]).
   converter(a,[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.3, 0.5, 1.0, 1.0, 1.0]).

/*************************************************************************
**************************/

   ação(Acoes):-
   situação_desejada(NiveisDesejados),
   Importâncias(ValoresImportância),
   parâmetros_similaridade(P,N),

aval(NiveisDesejados,ValoresImportância,0,AlunosCandidatosAvaliados,P,N),
   ordenar(AlunosCandidatosAvaliados,AlunosCandidatosOrdem),
   salvar_limparBaseconhecimento(AlunosCandidatosOrdem,Acoes).

/*************************************************************************
**************************/

   aval(_,_,Q,_,_,_):-
        quantidade_alunos(Q),!.

aval(NiveisDesejados,ValoresImportancia,X,[candidato(ID_Aluno,Nome,Aval)|R]
,P,N):-
   ID_Aluno is X+1,
   aluno_candidato(ID_Aluno,Nome,NiveisCorrentes),
   Diferençass(NiveisDesejados,NiveisCorrentes,Diferençass,P,N),
   f_avalia(Diferençass,ValoresImportancia,Aval),
   aval(NiveisDesejados,ValoresImportancia,ID_Aluno,R,P,N).

/*************************************************************************
**************************/

   Diferençass([NivelDesejado],[NivelCorrente],[Diferenças],P,N):- !,
   Diferenças(NivelDesejado,NivelCorrente,Diferenças,P,N).

Diferençass([NivelDesejado|NiveisDesejados],[NivelCorrente|NiveisCorrentes]
,[Diferenças|Diferençass],P,N):-
   Diferenças(NivelDesejado,NivelCorrente,Diferenças,P,N),
   Diferençass(NiveisDesejados,NiveisCorrentes,Diferençass,P,N).

/*************************************************************************
**************************/

   Diferenças(NivelDesejado,NivelCorrente,Diferenças,P,N):-
   pares_valores_participação(NivelDesejado,NivelCorrente,L1),
   subtrações_pares(L1,L2),
   modulos(L2,L3),
```

```
    elevados(L3,L4,P),
    somatoria(L4,S),
    E is 1/P,
    E1 is S**E,
    E2 is N**E,
    Diferenças is E1/E2.

    pares_valores_participação([],[],[]):-!.
    pares_valores_participação([X|Y],[Z|W],[par(X,Z)|L]):-
    pares_valores_participação(Y,W,L).

    subtrações_pares([],[]):-!.
    subtrações_pares([par(A,B)|Y],[Z|W]):-
    Z is A-B,
    subtrações_pares(Y,W).

    modulos([],[]):-!.
    modulos([X|Y],[Z|W]):-
    modulo(X,Z),
    modulos(Y,W).

    modulo(N,N):-
    N>=0.0,!.
    modulo(N,M):-
    N < 0.0,
    M is (-1.0)*N.

    elevados([],[],_):-!.
    elevados([X|Y],[Z|W],P):-
    Z is X**P,
    elevados(Y,W,P).

    somatoria(L,S):-
    somat(L,0.0,S).

    somat([],Acum,Acum):-!.
    somat([X|Y],Acum,S):-
    Acum_aux is X + Acum,
    somat(Y,Acum_aux,S).

/*****************************************************************************
**************************/

    f_avalia(Diferenças,ValoresImportancia,Aval):-
    um_menos(Diferenças,Similaridades),
    soma_ponderada(ValoresImportancia,Similaridades,Avalaux),
    somatoria(ValoresImportancia,Somaimp),
    Aval is Avalaux/Somaimp.

    um_menos([X],[Y]):- !, Y is 1-X.
    um_menos([X|Y],[Z|W]):-
    Z is 1-X,
    um_menos(Y,W).

    soma_ponderada(Imp,Sim,Aval):- s_p(Imp,Sim,0,Aval).

    s_p([],[],Acum,Aval):- !, Aval = Acum.
    s_p([Imp|RImp],[Sim|RSim],Acum,Aval):-
    Acumaux is Acum + Imp*Sim,
    s_p(RImp,RSim,Acumaux,Aval).
```

```
/*************************************************************************
**************************/

   ordenar([],[]):-!.
   ordenar(Lava,[candidato(ID_Aluno,Nome,Aval)|RLava_ordenada]):-
   buscar_maximo(Lava,candidato(ID_Aluno,Nome,Aval)),
   eliminar_aluno(candidato(ID_Aluno,Nome,Aval),Lava,L1),
   ordenar(L1,RLava_ordenada).


buscar_maximo([candidato(ID_Aluno,Nome,Aval)],candidato(ID_Aluno,Nome,Aval)
):-!.

buscar_maximo([candidato(ID_Aluno,Nome,Aval),candidato(_,A)|RLava],Min):-
   A =< Aval,!,
   buscar_maximo([candidato(ID_Aluno,Nome,Aval)|RLava],Min).
   buscar_maximo([_,candidato(ID_Aluno,Nome,Aval)|RLava],Min):-
   buscar_maximo([candidato(ID_Aluno,Nome,Aval)|RLava],Min).

   eliminar_aluno(O,[O|N],N):-!.
   eliminar_aluno(O,[X|Y],[X|Z]):-
   eliminar_aluno(O,Y,Z).

/*************************************************************************
**************************/

   salvar_limparBaseconhecimento(AlunosCandidatosOrdem,Acoes):-
   tell(Acoes),
   imprime(AlunosCandidatosOrdem),
   told,
   retract(quantidade_alunos(_)),
   retract(parâmetros_similaridade(_,_)),
   retract(situação_desejada(_)),
   retract(Importâncias(_)),
   abolish(aluno_candidato/3).

   imprime([]):-!.
   imprime([A|RA]):-
        write(A),nl,
        imprime(RA).
```