UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

LUIZ CLÁUDIO NOGUEIRA DA SILVA

MobiLE – Um Ambiente Multiagente de Aprendizagem Móvel para Apoiar a Recomendação Ubíqua de Objetos de Aprendizagem.

MOSSORÓ – RN

2012

LUIZ CLÁUDIO NOGUEIRA DA SILVA

MobiLE – Um Ambiente Multiagente de Aprendizagem Móvel para Apoiar a Recomendação Ubíqua de Objetos de Aprendizagem.

Dissertação apresentada ao Mestrado em Ciência da Computação – associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semiárido, para a obtenção do título de Mestre em Ciência da Computação.

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

MOSSORÓ – RN

2012

LUIZ CLÁUDIO NOGUEIRA DA SILVA

MobiLE – Um Ambiente Multiagente de Aprendizagem Móvel para Apoiar a Recomendação Ubíqua de Objetos de Aprendizagem.

Dissertação apresentada ao Mestrado 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. Marcelino Pereira dos Santos Silva – UERN
Membro Interno

_____

Profa. Dra. Cecília Dias Flores – UFCSPA
Membro Externo

Dedico este trabalho especialmente à minha mãe, Adelte Antonia Nogueira, ao meu irmão, Luiz Carlos Nogueira, e à minha noiva, Jamila Vasconcelos.

AGRADECIMENTOS

Aproveito a oportunidade para externar minha gratidão a pessoas que foram essenciais em minha vida e nessa caminhada árdua do mestrado, sem as quais eu não teria alcançado todos os meus objetivos.

Inicialmente, agradeço à minha mãe, Adelte Antonia Nogueira, e ao meu irmão, Luiz Carlos Nogueira, sem os quais eu não estaria hoje aqui. Esse obrigado não se resume aos anos na graduação e na pós-graduação, mas sim a uma vida inteira de cuidados, que fizeram de mim a pessoa que sou hoje. Minha mãe, especialmente, por sempre ter lutado para me educar sozinha, sacrificando interesses pessoais em prol da minha educação e, apesar das diversas dificuldades encontradas, ter me incentivado a continuar e nunca desistir. Ao meu irmão, que, na ausência de um pai, assumiu tal papel e me mostrou, por diversas vezes, qual rumo trilhar dentre tantos que se abriram à minha frente.

À minha noiva, Jamila Vasconcelos, e à sua família. A Jamila por estar ao meu lado há dez anos, dividindo, durante todo esse tempo, todos os momentos bons e ruins da nossa vida e por me encorajar para que sigamos em frente em prol de um futuro projetado em pequenos detalhes. Agradeço também por ter compreendido que a minha ausência era por um bem maior. A sua família, por sempre ter me apoiado e confiado em mim ao longo de todos esses anos.

Aos colegas do LES, dentre os quais Marcos Tullyo, Luiz Júnior, Mabel, Ferdinand, Rodrigo e Miguel, pela compreensão durante os momentos de estresse e pelo companheirismo de todos, o que fez com que os integrantes desse laboratório se tornassem uma família. Agradeço em especial ao amigo Luiz Júnior, pois, mesmo sem o auxílio esperado, se dispôs a trabalhar comigo até a finalização do mestrado, conforme combinado inicialmente.

Ao orientador e amigo Francisco Milton Mendes Neto, o qual é conhecido pelos mais distantes como o professor exigente, e pelos mais próximos como a pessoa humana, dedicada, também exigente, mas que sabe dividir bem a vida profissional da pessoal. Como já disse em outras oportunidades, Milton se mostrou um amigo, realizando atribuições que estavam longe de ser de um orientador. Muito obrigado Milton, e desejo que você seja retribuído em dobro por tudo o que fazes.

Aos colegas e professores do mestrado, com os quais adquiri não apenas conhecimentos técnicos necessários para realização deste trabalho, mas também valores que irão me nortear durante toda a vida.

À CAPES, pelo apoio financeiro que viabilizou a realização deste trabalho.

Enfim, agradeço a todos aqueles que, direta ou indiretamente, contribuíram para a minha formação. Para finalizar, aproveito para parafrasear Fernando Pessoa que, certa vez, disse: “Pedras no caminho? Guardo todas, um dia irei construir um castelo”. Vou além e digo: Pedras no caminho? Venho guardando todas, estou terminando de construir o meu próprio castelo!

"*Não é o mais forte da espécie que sobrevive, nem o mais inteligente, mas sim o que melhor se adapta à mudança" (Charles Darwin)*.

RESUMO

A Educação a Distância (EaD) é uma modalidade de ensino que vem crescendo há alguns anos. Uma das formas de se prover a EaD é através de ambientes de aprendizagem móveis, que provêem aos estudantes um método de ensino que não poderia ser realizado em um curso convencional baseado na web. Isto se deve, principalmente, ao aumento da mobilidade e da disponibilidade do ambiente de aprendizagem provida aos estudantes, que podem acessá-lo de qualquer lugar e sem a necessidade de ter sempre um computador pessoal disponível. O uso de objetos de aprendizagem (OA) em um formato padrão, por sua vez, consiste em uma forma efetiva de permitir, entre outras características, o reuso e a interoperabilidade de conteúdo entre diferentes sistemas de gestão de aprendizagem. No entanto, um problema que ocorre com freqüência é a inadequação do conteúdo ao contexto do estudante. Um ambiente de suporte à aprendizagem ubíqua permite que estudantes participando neste tipo de curso recebam conteúdos educacionais que atendam melhor a suas necessidades. É fundamental que as características do contexto desse ambiente sejam capturadas de forma dinâmica e autônoma para que o mesmo seja transparente para o estudante, o que pode ser conseguido através do uso de um sistema multiagente. Desta forma, o presente trabalho apresenta um ambiente multiagente de aprendizagem móvel para apoiar a recomendação ubíqua de OAs a fim de aperfeiçoar o processo de aprendizagem.

Palavras-Chave: Educação a Distância. Aprendizagem Móvel. Objetos de Aprendizagem. Aprendizagem Ubíqua. Agentes. Sistema Multiagente.

ABSTRACT

Distance Education (DE) is a teaching method that has been growing for some years. One way of providing distance education is through mobile learning environments, which provide students with a teaching method that would be not possible to be performed in a conventional web-based course. This is mainly due to increased mobility and the availability of the learning environment provided to students who can access it from anywhere and without the need to always have a personal computer available. The use of learning objects (LO) in a standard format, in turn, consists of an effective way to allow, among other features, content reuse and interoperability among different learning management systems. However, a problem that occurs frequently is the unsuitability of the content to the context in which the student is inserted. An ubiquitous learning environment allows students participating in this type of course to receive educational content that meet their needs. It is essential that the characteristics of the context of this environment be captured dynamically and autonomously so that it is transparent to the student, which can be achieved through the use of a multiagent system. Thus, this work presents a multiagent mobile learning environment to support ubiquitous recommendation of LOs in order to improve the learning process.

Keywords: Distance Education. Mobile Learning. Learning Objects. Ubiquitous Learning. Agents. Multiagent System.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABED	Associação Brasileira de Educação a Distância
ACC	*Agent Communication Channel*
ACL	*Agent Communication Language*
ADL	*Advanced Distributed Learning*
AG	Algoritmo Genético
AID	*Agent Identifier*
AML	*Agent Modeling Language*
AMS	*Agent Management System*
AP	*Agent Platform*
API	*Application Programming Interface*
AUML	*Agent UML*
BDI	*Belief Desire Intention*
CMS	*Course Management System*
DF	*Directory Facilitator*
EaD	Educação a Distância
E-learning	*Electronic Learning*
ESOA	Engenharia de Software Orientada a Agentes
GPL	*GNU Public License*
HTTP	*Hypertext Transfer Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
JADE	*Java Agent Development Framework*
LES	Laboratório de Engenharia de Software.
LMS	*Learning Management System*

LOM	*Learning Object Metadata*
LTSC	*Learning Technology Standard Committee*
MaSE	*Multiagent Systems Engineering*
MEC	Ministério da Educação
M-learning	*Mobile Learning*
Moodle	*Modular Object-Oriented Dynamic Learning Environment*
MSC	*Message Sequence Charts*
MTP	*Message Transport Protocol*
MTS	*Message Transport Service*
OA	Objeto de Aprendizagem
OAM	Objeto de Aprendizagem Móvel
PBL	*Problem-Based Learning*
POA	Programação Orientada a Agentes
RFID	*Radio Frequency Identification*
RUP	*Rational Unified Process*
SCORM	*Sharable Content Object Reference Model*
SDL	*Specification and Description Language*
SEED	Secretaria de Educação a Distância
SMA	Sistema Multiagente
U-Learning	*Ubiquitous Learning*
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A Educação a Distância (EaD) é uma modalidade de ensino e aprendizagem que vem crescendo há alguns anos. De acordo com pesquisa realizada pela Associação Brasileira de Educação a Distância (ABED) e pelo Ministério da Educação (MEC), a demanda em 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. Através deste suporte é possível automatizar cada vez mais o processo de ensino, tornando o professor um facilitador, deixando de ser a fonte principal de informações e passando a direcionar o processo de aprendizagem (PONTES, 2010).

Um das formas de se prover a EaD é através do uso de dispositivos móveis, modalidade esta conhecida como aprendizagem móvel (ou *mobile learning*, ou ainda *M-learning*). Este meio de oferecer ensino permite que estudantes e professores possam tirar vantagens dos recursos oferecidos pelas tecnologias móveis. Um destes benefícios é a possibilidade de acessar, visualizar e prover conteúdo independentemente do horário e a partir de qualquer localidade (CASTILHO e AYALA, 2008; YAU e JOY, 2010).

Contudo, para melhorar a eficácia da absorção de conhecimento pelos estudantes na aprendizagem móvel, deve-se levar em consideração as características particulares de cada estudante. Isso é necessário não apenas para fornecer um conteúdo que atenda às características cognitivas dos estudantes, mas também para prover conteúdos de forma adequada às restrições dos seus dispositivos móveis, uma vez que os mesmos possuem recursos distintos e limitados (com restrições). Surge então o conceito de ambientes sensíveis ao contexto (*context-aware environments*). Esse tipo de ambiente se adéqua ao perfil do usuário, levando em consideração informações fornecidas pelo próprio usuário, além daquelas captadas dinamicamente a partir de sua interação com os dispositivos computacionais (MOORE, *et al.*, 2009).

A fim de desenvolver ambientes de aprendizagem que sejam sensíveis ao contexto do estudante, é fundamental que os conteúdos educacionais sejam criados de maneira padronizada. Desta forma, é possível que um ambiente de suporte à aprendizagem exiba o conteúdo de forma adequada e reutilize conteúdos em diferentes contextos e a partir de diversos repositórios. Além disso, é possível combinar conteúdos distintos, o que, por sua vez, melhora o processo de produção de conteúdo e, como conseqüência, reduz os seus custos (RODOLPHO, 2009). Um modo eficaz de padronizar conteúdos educacionais é através do uso de Objetos de Aprendizagem (OAs), os quais consistem em pequenas unidades de conteúdo que podem ser usadas, reutilizadas e referenciadas durante um processo de aprendizagem (LTSC , 2002).

1.2 PROBLEMÁTICA

Tecnologias inteligentes e baseadas na Web trouxeram grande comodidade na aquisição de conteúdos de aprendizagem, assim como liberdade de tempo, lugar e ritmo de estudo. Entretanto, mais comodidade e liberdade levam a mais responsabilidades, o que tende a deixar os estudantes desorientados e sobrecarregados (HAN, GAO e WANG, 2010).

Além disso, a aprendizagem móvel requer que o fornecimento de conteúdos seja adequado às necessidades dos estudantes, visto que cada um destes possui necessidades específicas. Outro fator que influencia a aprendizagem móvel é o fato dos dispositivos móveis possuírem recursos limitados, que devem ser levados em consideração para uma aprendizagem eficaz (MOORE e HU, 2007). Deve-se levar em consideração também as condições nas quais o estudante se encontra em determinado momento, ou seja, o seu contexto. A adequação ao contexto do estudante pode mitigar as limitações encontradas, permitindo o fornecimento do serviço condizente com as suas necessidades e que se adéqüe aos dispositivos disponíveis (MOORE e HU, 2007).

Entretanto, para que seja possível o fornecimento adequado de conteúdos e para que o mesmo OA possa ser reutilizado em diversos ambientes computacionais, bem como possa ser visualizado em dispositivos com diferentes recursos, é necessária a utilização de um padrão para desenvolvimento desses OAs (DIAS, *et al.*, 2009). Uma vez seguindo um padrão para o desenvolvimento desses OAs, é possível que diferentes aplicações apresentem o mesmo conteúdo de forma apropriada, levando em consideração suas próprias restrições.

Nesse contexto, surge um novo problema: os dispositivos móveis possuem características que devem ser consideradas quando do fornecimento de OAs que, normalmente, não são consideradas no momento da criação destes. Com base nesta limitação, alguns trabalhos sugerem a extensão de padrões de OAs de forma que possam atender aos requisitos necessários em uma aprendizagem móvel.

Yau & Joy (2010), em seu trabalho, consideram características do contexto do usuário para recomendação de OAs. Entretanto, apenas levam em consideração o local de estudo, o nível de barulho e a hora do dia preferidos. Já o trabalho de Han, Gao e Wang (2010) recomenda OAs levando em consideração características cognitivas dos estudantes e a estrutura de conteúdo e curricular dos OAs. É necessário dar atenção, também, a características que indiquem possíveis limitações do dispositivo móvel em si, possibilitando a definição do contexto do usuário de maneira mais apropriada.

Não foi encontrado na literatura um trabalho que propusesse um ambiente de apoio à aprendizagem móvel que considerasse conjuntamente (i) características de preferências do estudante, (ii) localização geográfica do mesmo e (iii) limitações nos recursos físicos disponíveis no dispositivo móvel para a definição do contexto do estudante. Além disso, também não foi encontrada uma abordagem baseada em algoritmos genéticos para realizar a recomendação de conteúdos educacionais.

1.3 OBJETIVO GERAL

Tendo em vista a problemática apresentada, o presente trabalho tem como objetivo geral prover um ambiente de aprendizagem, através do uso de dispositivos móveis, que auxilie a aprendizagem dos estudantes e se adéqüe às necessidades dos mesmos, de acordo com características do contexto no qual os mesmos se encontram, das características dos perfis dos mesmos e dos conteúdos educacionais disponíveis na forma de OAs.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação está organizada da seguinte forma: o Capítulo 2 traz conceitos de agentes e sistemas multiagente e sobre as metodologias utilizadas para modelagem de agentes. Ao final deste capítulo também são mostradas algumas experiências de implantação de sistemas de suporte à aprendizagem que utilizam agentes. O Capítulo 3 traz uma fundamentação sobre algoritmos evolucionários com foco em algoritmos genéticos, por ser uma técnica de inteligência artificial (IA) utilizada pelos agentes no sistema multiagente proposto neste trabalho. No Capítulo 4 são apresentados conceitos relacionados à aprendizagem ubíqua, englobando tanto conceitos gerais sobre EaD, como também sobre aprendizagem móvel. No Capítulo 5 são apresentadas definições de objetos de aprendizagem, elencando suas características e descrevendo os padrões utilizados durante sua construção. O Capítulo 6 apresenta a modelagem do ambiente proposto e o mecanismo de recomendação de OAs. O Capítulo 7 descreve a arquitetura do ambiente de aprendizagem proposto, detalhando os agentes e como estes foram implementados. E, por fim, o Capítulo 8 apresenta as considerações finais e os trabalhos futuros.

2 AGENTES INTELIGENTES E SISTEMAS MULTIAGENTE

Este capítulo descreve aspectos relacionados a agentes e sistemas multiagente. 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 tipos de agentes existentes. A Seção 2.3 mostra conceitos relacionados a Sistemas Multiagente (SMAs). Já a Seção 2.4 apresenta vários ambientes de aprendizagem que se utilizaram de agentes para atingir seus objetivos. A Seção 2.5 traz o conceito de Engenharia de Software Orientada a Agentes (ESOA) e uma breve discussão a respeito de metodologias de modelagem de agentes, mostrando um pouco mais de detalhes da metodologia que será adotada neste trabalho, além de um padrão utilizado para estabelecer a comunicação e o gerenciamento dos agentes.

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.

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 idéia de que agentes são entidades de software ou não que são caracterizadas por serem: autônomas, proativas e direcionadas a objetivos.

Por fim, 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 uma outra

definição que afirma que agentes são programas que realizam diálogos para negociar e coordenar transferências de informação.

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, de acordo com o conhecimento interno que possui (que pode advir, inclusive, de percepções anteriores do agente) e a percepção atual do ambiente. Além disso, essas entidades poderão se comunicar entre si, trocando informações para possibilitar alcançar um objetivo comum. A Figura 1 ilustra esta definição:

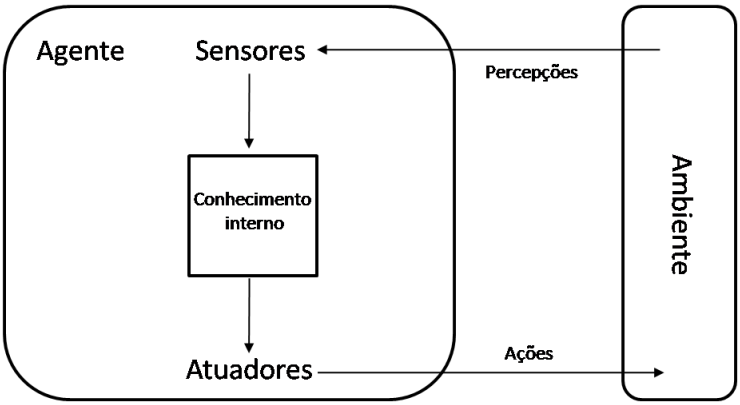

Figura 1 – Visão de um agente padrão. Fonte: Adaptado de (RUSSEL e NORVIG, 2003).

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

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

Diante desses elementos, é possível conceituar um agente racional como sendo aquele que, para cada seqüência de percepções possível, seleciona uma ação que venha a maximizar sua medida de desempenho, dada a evidência fornecida pela seqüência 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.

2.2 TIPOS DE AGENTES

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

- 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 seqüência 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, conseqüentemente, 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).

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 conseqüência, adicionar novas funcionalidades através da inclusão de novos agentes (ARTERO, 2009). 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 (ARTERO, 2009, p. 212-213).

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). Essa será a abordagem adotada para a arquitetura do ambiente multiagente proposto neste trabalho.

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).

2.3.1 Organização dos Sistemas Multiagente

Os sistemas multiagente podem ser organizados de diversas maneiras, sendo três os tipos mais comuns (ARTERO, 2009):

- 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 sistema multiagente proposto neste trabalho é do tipo comunidade de especialistas.

2.4 APLICAÇÕES DOS SISTEMAS MULTIAGENTE

De um modo geral, os agentes inteligentes são aplicados na resolução de problemas para os quais não se conhece um algoritmo exato. As aplicações que fazem o uso de agentes inteligentes têm crescido bastante nos últimos anos, sendo utilizados principalmente em: aplicações na Web, desenvolvimento de jogos, construção de simuladores e aplicações da área médica (para monitoramento de pacientes e desenvolvimento de sistemas tolerantes a falhas) e doméstica, como é o caso do exemplo do aspirador de pó, apresentado em (RUSSEL e NORVIG, 2003), que já se tornou realidade (ARTERO, 2009).

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.

Em (AL-SAKRAN, MUHAYA e SERGUIEVSKAIA, 2010), os autores propõem uma arquitetura de um sistema de aprendizagem móvel multiagente que é capaz de ajustar o conteúdo de acordo com o progresso dos estudantes, melhorando assim os processos de ensino e aprendizagem. Essa arquitetura também permite que os estudantes possam coletar, analisar, distribuir e usar conteúdos de aprendizagem de múltiplas fontes de conhecimento. A arquitetura proposta possui nove componentes, sendo um servidor de agentes e oito agentes em si.

Ching-bang (2010) apresenta o PNULKA (*Personalized Navigation and Ubiquitous Learning with Knowledge Agents*), que consiste em uma biblioteca inteligente para aprendizagem ubíqua através do uso de um sistema multiagente. O sistema criado combina a tecnologia RFID (*Radio Frequency Identification*) com agentes móveis para o sistema de aprendizagem móvel da biblioteca, a fim de prover a informação apropriada e no momento correto aos estudantes. Com esse intuito, são utilizados três tipos de agentes no sistema: um agente sensível ao ambiente, um agente de análise de preferência pessoal e, por fim, um agente de análise de livro ou serviço.

Pontes (2010) apresenta uma arquitetura de agentes para suporte à colaboração na aprendizagem baseada em problemas – PBL (*Problem-Based Learning*) – em ambientes virtuais de aprendizagem, com o intuito de detectar e corrigir problemas inerentes à implantação desta teoria de aprendizagem, melhorando o processo de aprendizagem. A abordagem proposta no trabalho utiliza-se de três tipos de agentes: um agente detector de problemas, um agente estudante e um agente animado de interface.

2.5 ENGENHARIA DE SOFTWARE ORIENTADA A AGENTES

Para se construir software de alta qualidade, é necessário seguir uma metodologia de desenvolvimento de software, a qual define, de forma organizada, como devem ser realizadas as atividades, ações e tarefas necessárias para desenvolvimento do software (PRESSMAN, 2011).

Nesse sentido, para gerenciar com sucesso a complexidade associada ao desenvolvimento, manutenção e distribuição de SMAs, são necessárias técnicas e ferramentas de engenharia de software ao longo do ciclo de vida do software. A construção de um SMA não é uma tarefa trivial, devido ao fato de que devem ser tratados todos os problemas dos sistemas distribuídos e concorrentes tradicionais, além das dificuldades que surgem dos requisitos de flexibilidade e interações sofisticadas (CASTRO, ALENCAR e SILVA, 2006).

A Engenharia de Software Orientada a Agentes (ESOA) vem justamente para suprir essa necessidade. De acordo com Castro, Alencar e Silva (2006), a ESOA representa uma nova maneira de analisar, projetar, construir, gerenciar e manter software complexo envolvendo o paradigma de desenvolvimento orientado a agentes.

Qualquer metodologia para a ESOA deve prover abstrações adequadas e ferramentas para modelar não apenas as tarefas individuais, mas também as tarefas sociais dos agentes, além do comportamento inteligente de cada agente. Pensando nisso, pesquisadores têm proposto várias metodologias para a modelagem de SMAs (MORAIS II, 2010).

2.5.1 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 (GIORGINI, *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 1 mostra uma breve descrição de cada uma dessas metodologias.

Tabela 1 – 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 sistemas multiagente 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 sistema multiagente.
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 sistemas multiagente usando conceitos extraídos da orientação a objetos usando UML[1]. 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 sistemas multiagente adaptativos. É baseada no RUP (*Rational Unified Process*)[2], 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*)[3].
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

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

	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.

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.

2.5.2 Metodologia MAS-CommonKADS

A MAS-CommonKADS é uma metodologia de ESOA 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 seqüência de mensagens - MSC (*Message Sequence Charts*) e SDL (*Specification and Description Language*) com novas perspectivas dirigidas pela metáfora dos agentes (IGLESIAS e GARIJO, 2005).

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

- 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).

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, precondições, 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 seqüência 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).

2.5.3 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.

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+.

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.

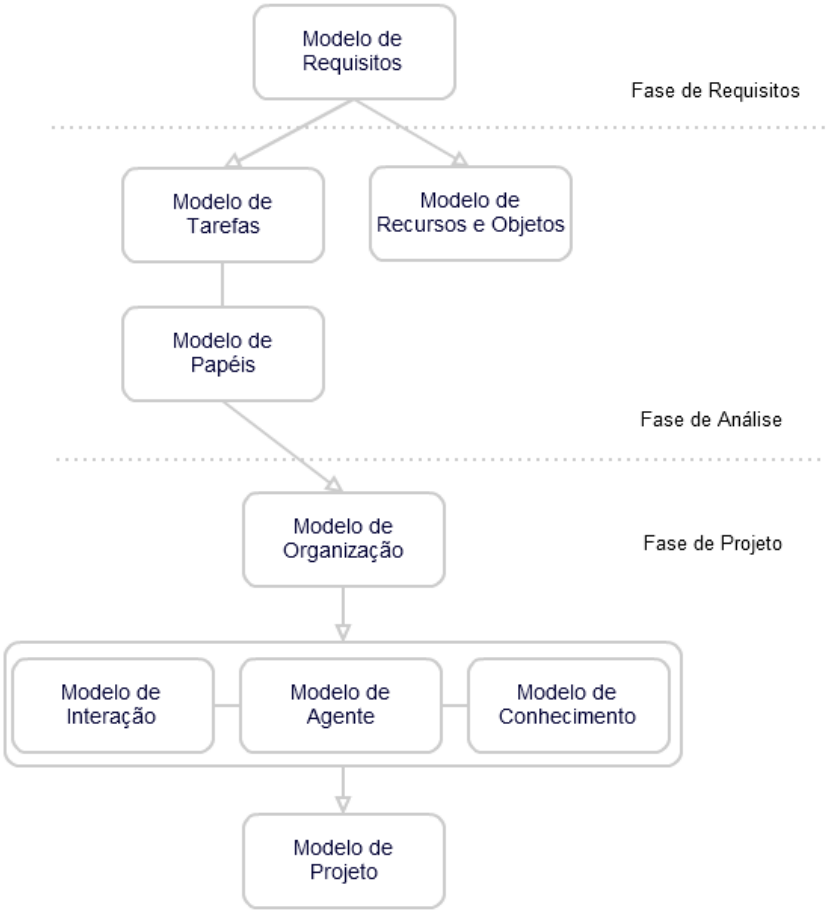

Figura 3 – Arquitetura da MAS-CommonKADS+. Fonte: Adaptado de (MORAIS II, 2010).

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).

2.5.4 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 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).

FIPA (*Foundation for Intelligent Physical Agents*) é um conjunto de padrões da IEEE *Computer Society* 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).

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). A Figura 4 mostra os componentes desse modelo.

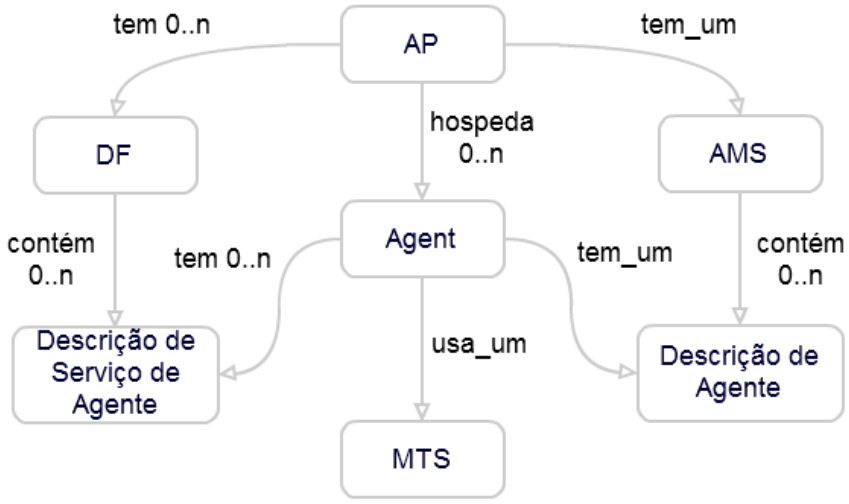

Figura 4 – Modelo de referência FIPA para gerenciamento de agentes. Fonte: Adaptado de (BELIFEMINE, CAIRE e GREENWOOD, 2007).

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

- *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.

3 ALGORITMOS EVOLUCIONÁRIOS GENÉTICOS

Este capítulo traz a fundamentação teórica necessária para que seja possível entender o funcionamento do algoritmo genético criado no presente trabalho. A Seção 3.1 mostra brevemente uma introdução a respeito da teoria da evolução humana de modo a auxiliar no entendimento dos algoritmos evolucionários. A Seção 3.2 mostra os conceitos básicos dos algoritmos evolucionários. A Seção 3.3 aborda os conceitos inerentes aos algoritmos genéticos, tipo de algoritmo evolucionário utilizado no presente trabalho.

3.1 TEORIA DA EVOLUÇÃO

Uma das teorias que tentam explicar a evolução das espécies é a teoria da evolução. Essa teoria foi proposta por Charles Robert Darwin (1809-1882) por não concordar com outras teorias propostas a época (PETROLI NETO, 2011). De acordo com essa teoria, na natureza todos os indivíduos dentro de um ecossistema competem entre si por recursos limitados, tais como comida e água. Os indivíduos de uma mesma espécie que não obtêm êxito nessa competição tendem a ter sua sucessão reduzida, tornando menor a probabilidade de ter seus genes propagados ao longo de sucessivas gerações (LINDEN, 2008). É o que ocorre, por exemplo, com os machos de determinadas espécies, que competem entre si para garantir o direito às fêmeas ou, em outras situações, de espécies que participam de concursos de canto ou mesmo de exibições de plumagens no intuito de impressionar as fêmeas de suas espécies para, conseqüentemente, conseguirem se reproduzir (ARTERO, 2009). Esse é o denominado processo de seleção natural (LINDEN, 2008).

Os sucessores de um casal de reprodutores tendem a possuir combinações das características de seus pais. A combinação entre as características dos indivíduos que sobrevivem pode produzir um novo indivíduo muito mais bem adaptado às características de seu meio ambiente, ao mesclar características positivas de cada um dos reprodutores. Isso implica que os descendentes são variações de seus pais (LINDEN, 2008; PETROLI NETO, 2011). Essa idéia também é muito utilizada, por exemplo, no aprimoramento genético de animais, como é o caso dos bovinos, onde são cruzados animais com maior capacidade de

produzir leite, com o intuito que as novas gerações apresentem uma melhoria nessa característica (ARTERO, 2009).

Porém, vale ressaltar que a variação acima referida pode ser positiva ou negativa. Esta evolução natural não tem, necessariamente, o objetivo de maximizar alguma característica das espécies. Assim, não existe nenhum garantia de que descendentes de pais bem adaptados também o sejam. Por exemplo, existem casos de descendência nos quais os pais são fortes e saudáveis, mas os filhos possuem doenças ou outras fraquezas (LINDEN, 2008). Essa é uma conseqüência natural do mecanismo de transmissão da informação, cuja discussão não convém ao presente trabalho.

Assim, de acordo com a teoria da evolução humana, pode-se afirmar, resumidamente, que a evolução é um processo no qual os seres vivos são alterados por um conjunto de modificações genéticas com o passar dos tempos. Outras informações mais detalhadas sobre genética serão fornecidas na seção que trata dos algoritmos genéticos.

3.2 ALGORITMOS EVOLUCIONÁRIOS

Para entender o que são algoritmos genéticos é preciso inicialmente definir o que são algoritmos evolucionários. Estes são algoritmos 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).

Operadores genéticos podem ser definidos, de uma forma concisa, como aproximações computacionais de fenômenos vistos na natureza, tais como reprodução sexuada (exige a presença de dois organismos, na maioria das vezes de sexos opostos, que trocam material genético), mutação genética, seleção de indivíduos, além de outros processos que podem ser criados pelos próprios programadores. A ocorrência de tais processos depende do “desempenho” dos indivíduos desta espécie dentro do “ambiente” (LINDEN, 2008). Ou seja, é necessário que um indivíduo mostre características que lhe atribuam destaque dentro do grupo para que ele possa, conseqüentemente, sobreviver para uma próxima geração.

Algoritmos evolucionários funcionam mantendo uma população de estruturas denominadas indivíduos ou cromossomos, que se comportam de forma semelhante às espécies. A qualidade de cada indivíduo é então avaliada como solução do problema sendo tratado. Com base nessa avaliação são aplicados os operadores genéticos (LINDEN, 2008). O pseudocódigo da Figura 5 mostra um resumo do comportamento padrão dos algoritmos evolucionários. No pseudocódigo é possível ver que os algoritmos evolucionários dependem fortemente de fatores estocásticos (ou probabilísticos), tanto na etapa de inicialização de sua população quanto na etapa de seleção dos pais para evolução. Assim, os algoritmos evolucionários não podem garantir que será obtida sempre a solução ótima em cada uma de suas execuções.

```
T:=0                                    // Inicializa contador de tempo
Inicializa_População P(0)               // Inicializa a população aleatoriamente
Enquanto não terminar faça              // condição de término (por tempo, avaliação etc.)
        Avalie_População P(t)           // Avalia a população neste instante
        P´:= Selecione_Pais P(t)        // Seleciona sub-população para gerar nova geração
        P´:= Recombinacao_mutacao P´    // Aplica operadores genéticos
        Avalie_População P´             //Avalia a nova população
        P (t+1) = Selecione_sobreviventes P (t), P´  //Seleciona sobreviventes desta geração.
        T:= t+1                         //Incrementa o contador de tempo
```

Figura 5 – Pseudocódigo básico de um algoritmo evolucionário. Fonte: Adaptado de (LINDEN, 2008).

Diante do que foi exposto, os algoritmos evolucionários são considerados heurísticas. Essas técnicas não garantem que a melhor solução possível será encontrada, pois, para alguns problemas, é impossível pesquisar, em tempo hábil, todo o universo de soluções existentes. No entanto, as heurísticas garantem encontrar uma solução relativamente boa dentro de um tempo considerável, o que pode ser suficiente para o problema sendo tratado. O objetivo dessas técnicas é, então, reduzir o espaço de busca do problema, de modo a se obter um bom resultado em tempo aceitável (BRASIL, 2011).

Entretanto, diferentemente de outras técnicas, os algoritmos evolucionários não são puramente aleatórios, pois utilizam também informações do estado corrente para guiar a pesquisa, isto é, a informação conhecida direciona a busca. Devido a esse fato, eles são posicionados como um ramo específico das técnicas de busca: as “Técnicas Aleatórias-Guiadas” (LINDEN, 2008; ZINI, 2009), conforme mostra a Figura 6.

Figura 6 – Subdivisão das técnicas de busca. Fonte: Adaptado de (LINDEN, 2008).

3.3 ALGORITMOS GENÉTICOS

Os Algoritmos Genéticos (AG) constituem um ramo dos algoritmos evolucionários e, portanto, também podem ser definidos como uma metáfora do processo biológico de evolução natural (LINDEN, 2008). 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).

AGs são técnicas heurísticas de otimização global, diferentemente de métodos de subida de encosta (ou *hill climbing*), os quais seguem a derivada de uma função de forma a encontrar o máximo de uma função, podendo ficar, facilmente, retidos em um máximo local (LINDEN, 2008), como mostra a Figura 7. Conforme se pode perceber na imagem, uma técnica de *hill climbing* se inicia em qualquer um dos pontos que estão marcados e segue a direção de maior crescimento, o que acaba deixando-a presa em um ponto de máximo local.

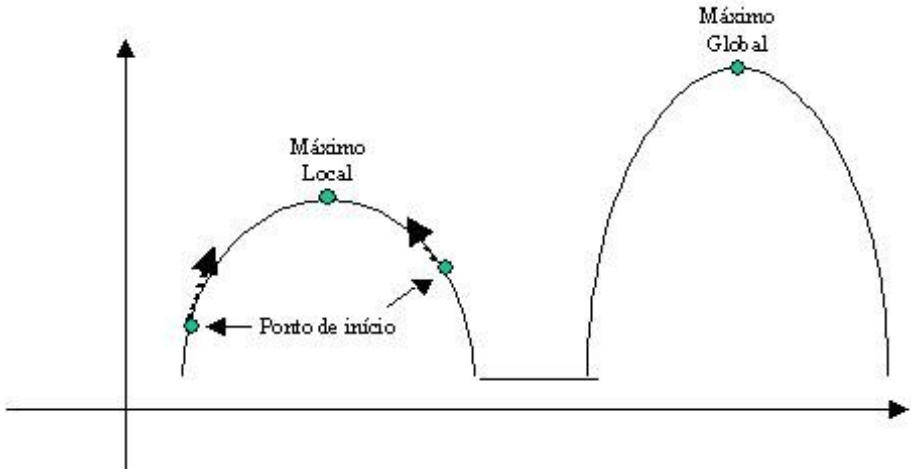

Figura 7 – Função hipotética com um máximo global e outro local. Fonte: (LINDEN, 2008).

Diferentemente do que acontece com esse tipo de técnica, AGs não dependem tão fortemente dos valores iniciais, até mesmo porque uma das formas de se gerar sua população inicial é através de métodos aleatórios. Dessa forma, eles não ficam estagnados simplesmente pelo fato de terem encontrado um máximo local. Nesse ponto, fica visível o porquê do AG se basear na evolução natural, pois, mesmo encontrando um indivíduo que é instantaneamente o melhor de certo grupo, ele não pára de criar novas gerações em busca de indivíduos ainda melhores (LINDEN, 2008).

Os AGs, assim como os demais tipos de algoritmos evolucionários, não podem garantir que uma geração será melhor do que a sua geração antecessora, mesmo sendo pouco provável não ocorrer tal situação. Isso acontece devido ao fato desses algoritmos serem baseados na evolução natural que, por sua vez, não consiste em um processo dirigido à obtenção da solução ótima. De fato, o processo simplesmente consiste em fazer competir um conjunto de indivíduos, fazendo com que sobrevivam aqueles indivíduos que são mais aptos (LINDEN, 2008; PETROLI NETO, 2011).

Antes de realizar um melhor detalhamento das etapas integrantes de um AG, é necessário ter em mente algumas definições básicas provenientes da genética, visto que aquele se inspira nos conceitos desta. Alguns desses termos são introduzidos na próxima seção, enquanto outros termos serão discutidos no decorrer das seções seguintes.

3.3.1 Definições

3.3.1.1 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).

3.3.1.2 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).

3.3.1.3 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).

3.3.1.4 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).

3.3.1.5 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.

3.3.3 Estrutura Básica de um Algoritmo Genético

Como já visto, os AGs consistem em um ramo dos algoritmos evolucionários e, portanto, seu funcionamento é bastante similar àquele mostrado na Figura 5. Algoritmicamente, um AG pode ser descrito através dos seguintes passos (LINDEN, 2008; PETROLI NETO, 2011):

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 8.

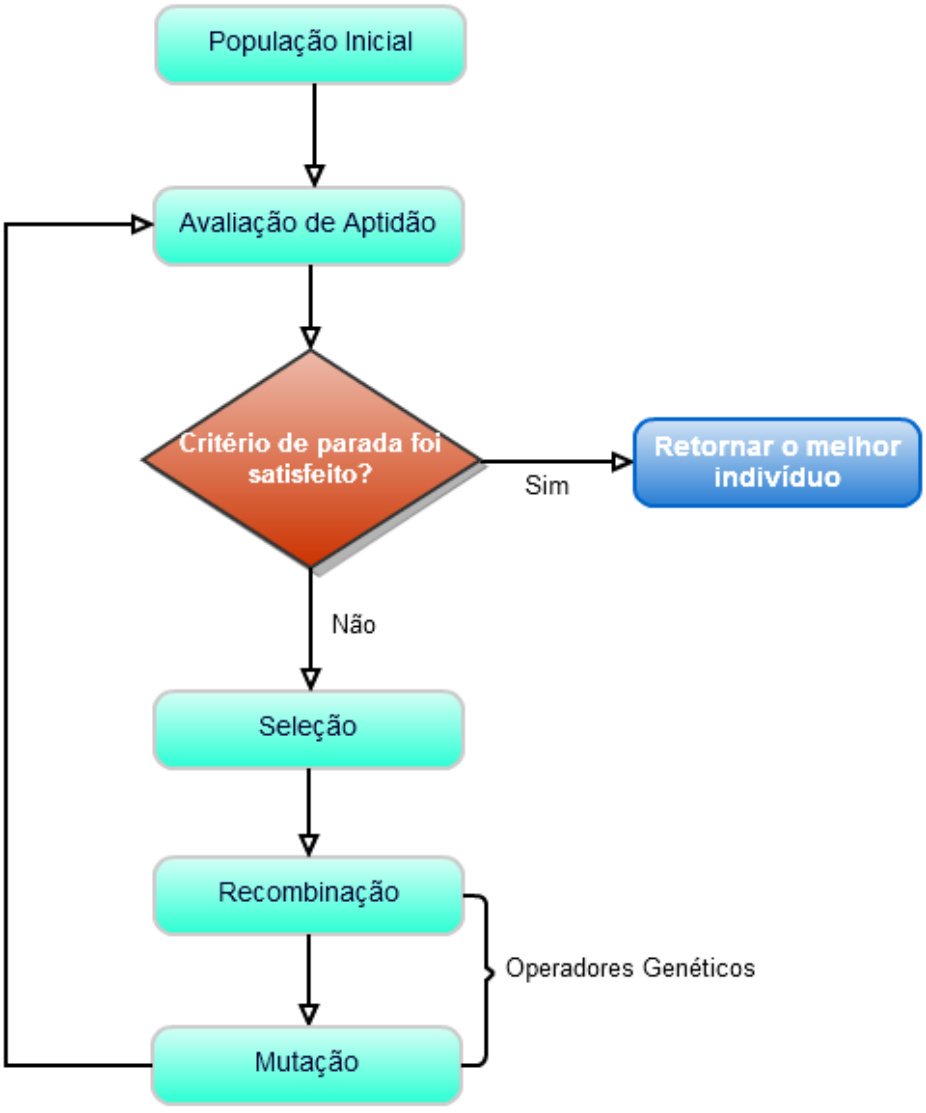

Figura 8 – Estrutura Básica de um Algoritmo Genético. Fonte: Adaptado de (ZINI, 2009).

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 8 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 a escolha de uma codificação dos cromossomos adequada ao problema e de uma função de aptidão que avalie satisfatoriamente o grau de adequação de cada indivíduo como solução do problema em questão (LINDEN, 2008; PETROLI NETO, 2011). Esses e outros aspectos implícitos no algoritmo apresentado serão tratados de forma mais detalhada a partir das próximas seções.

3.3.4 Codificação dos Cromossomos

Juntamente com a função de avaliação, a codificação dos cromossomos é o item mais importante para o perfeito funcionamento de um AG (PETROLI NETO, 2011). Os cromossomos podem ter seus genes representados de várias maneiras, porém, como os genes podem usar valores binários, inteiros ou reais, os cromossomos mais comuns são formados com estes tipos (ARTERO, 2009). Mais precisamente, pode-se dizer que a codificação binária, onde cada indivíduo é representado por um vetor composto por 0s e 1s, é a mais utilizada (LINDEN, 2008). Essa é uma das principais diferenças entre os AGs e as técnicas de otimização convencionais: os AGs trabalham, na maioria das vezes, com uma codificação do conjunto de parâmetros e não com os próprios parâmetros (ARTERO, 2009).

Caso se adote uma abordagem binária e esteja sendo tratado um problema cujas variáveis sejam inteiras ou reais, estas podem ser, de alguma forma, transformadas para valores binários (ZINI, 2009). Pode-se imaginar, por exemplo, uma situação na qual seja necessário representar números inteiros entre -15 e 15. Nesse caso, estes números podem ser descritos como um conjunto de *5 bits*, sendo 1 bit para sinal e os 4 restantes para representar amplitude (2^4=16). Nesse caso, o número -10 seria representado conforme a Figura 9, onde o bit em vermelho representa o sinal negativo. Em determinados casos, quantos mais bits forem usados para representar um indivíduo, melhor será a precisão numérica. Porém, caso os cromossomos sejam muito longos, pode se tornar difícil a sua manipulação (ARTERO, 2009).

1	1	0	1	0

Figura 9 – Representação do inteiro -10 na codificação com 5 bits.

Uma das vantagens em se utilizar a representação binária é a possibilidade de manipular de maneira fácil e eficiente os operadores genéticos sobre esse cromossomo

(PETROLI NETO, 2011). Isso se deve ao fato de que a implementação dos operadores genéticos e as características de convergência tiveram como base a codificação binária (ZINI, 2009).

Outras codificações podem ser utilizadas para representar os cromossomos, dependendo da estrutura e da complexidade que se deseja resolver, desde que, independentemente de qual seja a codificação escolhida, ela consiga retratar o problema real de forma clara e eficiente (PETROLI NETO, 2011).

3.3.5 Geração da População Inicial

Na maioria das vezes, a população inicial é gerada de forma aleatória. Porém, existem situações nas quais a geração da população inicial se dá utilizando-se uma heurística. Dessa forma, é possível introduzir na população inicial indivíduos que representem soluções aproximadas conhecidas contendo algum tipo de informação prévia (ZINI, 2009). De fato, existem diversos trabalhos nos quais é comprovada a teoria de que a geração da população inicial não é uma fase crítica em algoritmos genéticos. Porém, é de extrema importância que a população contenha indivíduos que sejam diversificados (ZINI, 2009).

3.3.6 Escolha da 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.

Como os AGs são, geralmente, bastante genéricos, em muitos casos, a função de aptidão consiste no único componente que faz a ligação entre o AG e o problema real

(LINDEN, 2008; PETROLI NETO, 2011). Isso acontece porque a função de aptidão apenas avalia a qualidade da solução que está sendo representada por aquele indivíduo, sem armazenar qualquer tipo de informação sobre as técnicas de resolução utilizadas no problema (LINDEN, 2008). Dessa forma, é possível concluir que o mesmo AG pode ser utilizado para descobrir o máximo de toda e qualquer função de *n* variáveis sem nenhuma alteração das estruturas de dados e procedimentos adotados, alterando-se, apenas, a função de aptidão (LINDEN, 2008).

Sem a função de aptidão, é impossível determinar se um problema está evoluindo para a solução ótima ou não (PETROLI NETO, 2011). Deve-se tomar bastante cuidado no momento da escolha de uma função de aptidão, pois ela deve embutir todo o conhecimento que se possui sobre o problema a ser resolvido, englobando tanto suas restrições quanto os seus requisitos de qualidade. Quanto mais conhecimento for embutido, menos genérico será o AG e, conseqüentemente, melhor desempenho demonstrará (LINDEN, 2008). Se houver pouca precisão na avaliação, uma solução ótima pode ser deixada de lado durante a execução do algoritmo e pode ser gasto um maior tempo computacional explorando soluções pouco promissoras (ZINI, 2009).

3.3.7 Seleção de Pais

Uma vez que os indivíduos de uma população foram avaliados de acordo com uma função de aptidão, tem que se criar um mecanismo capaz de transmitir a hereditariedade deles às populações seguintes, de modo que suas boas características possam passar para as gerações sucessoras. O mecanismo de seleção se baseia no princípio da sobrevivência dos melhores indivíduos, de acordo com o qual aqueles que são mais aptos recebem uma maior probabilidade de serem copiados para fazer parte da geração da nova população (ZINI, 2009).

Vale destacar que, apesar de ser possível, não é uma boa prática simplesmente descartar, da população reprodutora, os indivíduos menos aptos. Isto poderia causar uma convergência genética da população para um conjunto de soluções que possuiriam características (genes) semelhantes. Desta forma, faltaria diversidade a esta população e ela não conseguiria evoluir, a não ser através de mutações aleatórias que fossem positivas. Quanto maior for esta perda de diversidade, mais rápida será a convergência genética de um AG. Selecionando de forma justa os indivíduos menos aptos da população, pode-se evitar, ou

pelos minimizar, a convergência genética e representar de forma verídica o que ocorre na natureza, onde os indivíduos mais fracos também se reproduzem, apesar de fazê-lo com menos freqüência do que os indivíduos mais aptos. Além disso, até mesmo indivíduos cujo valor resultante da função de aptidão foi extremamente baixo podem possuir características genéticas favoráveis à criação de um indivíduo que represente a melhor solução para o problema (LINDEN, 2008).

Existem vários métodos de seleção de pais, os quais podem variar significativamente, dentre os quais merecem destaque os seguintes (ARTERO, 2009; LINDEN, 2008):

- Seleção aleatória: são escolhidos, ao acaso, dois indivíduos da população para participarem como pais no processo de reprodução.
- Seleção por torneio: a idéia é selecionar uma quantidade aleatória de indivíduos da população para competirem entre si pelo direito de ser pai, levando-se em consideração a sua avaliação. A quantidade de indivíduos escolhida para participar do torneio é chamada de tamanho do torneio (k). Nesse caso, k deve ser, no mínimo, igual 2, pois, do contrário, não haverá competição. Por outro lado, se k for igual ao tamanho da população, o melhor indivíduo será sempre o mesmo. Nesse método, a única vantagem que os melhores indivíduos possuem é que, se selecionados, vencerão o torneio.
- Seleção por giro de roleta: é criada uma roleta (de maneira virtual) na qual cada indivíduo recebe um pedaço proporcional à sua avaliação de aptidão. Ou seja, para aqueles com alta aptidão é dada uma porção maior da roleta, enquanto que aos indivíduos com aptidão mais baixa é dada uma porção proporcionalmente menor. Em seguida, a roleta é "girada" e o selecionado será o indivíduo sobre o qual ela parar. Logo, os maios aptos possuem mais chances de serem escolhidos.
- Seleção por truncamento: nesse método apenas os melhores x% da população podem ser escolhidos como pais da próxima geração. O valor do parâmetro x pode variar entre 0 e 100%, porém geralmente é utilizado um valor inserido na faixa [10%; 50%] (LINDEN, 2008). Para implementar este método, os indivíduos são ordenados em uma lista decrescente de acordo com o resultado da função de aptidão, podendo participar da seleção somente aqueles que estiverem entre a posição 1 e a posição de corte. Este método é o que causa maior perda de diversidade e, como conseqüência, é o que tende a fazer com

que o AG apresente o pior desempenho. Pode ser melhor utilizado no começo da evolução de um AG em uma estratégia híbrida, quando a diversidade ainda é muito grande.

O método mais utilizado pela maioria dos pesquisadores de AGs é o método da seleção por giro de roleta (LINDEN, 2008), até mesmo porque, assim como ocorre na natureza, esse método fornece uma probabilidade maior de que indivíduos mais aptos sejam escolhidos para reprodução.

3.3.8 Operadores Genéticos

Após a seleção dos indivíduos em uma população, são aplicados os operadores genéticos, obtendo-se assim uma população com melhores indivíduos ou não. Esse procedimento é repetido através de sucessivas gerações até que se atinja um resultado satisfatório (ZINI, 2009). Os AGs básicos são geralmente constituídos de dois operadores: cruzamento e mutação (ZINI, 2009).

3.3.8.1 Cruzamento (*crossover*)

Usando-se este operador, ocorre o cruzamento entre o material genético de dois indivíduos pais, combinando informações de modo que exista uma probabilidade razoável dos novos indivíduos produzidos serem melhores que seus pais (ZINI, 2009).

Depois de selecionados dois pais de acordo com o módulo de seleção escolhido, um ou mais pontos de corte são selecionados aleatoriamente (ARTERO, 2009), de acordo com a estratégia adotada. Um ponto de corte consiste em uma posição entre dois genes de um cromossomo. Assim, cada indivíduo possui *n* genes e *n-1* pontos de corte (LINDEN, 2008). As estratégias mais conhecidas de utilização de ponto de corte são as seguintes (ZINI, 2009):

- Operadores de um ponto: cada par de indivíduos é particionado em um único ponto de corte. Depois de sorteado aleatoriamente o ponto de corte, os pais são separados em duas partes, sendo uma à esquerda do ponto de corte e outra à direita. Nesse caso, serão gerados dois filhos a partir dos pais. O $filho_1$ será

composto da concatenação da parte esquerda do ponto de corte do pai_1 com a parte direita do pai_2. Já o $filho_2$ é composto das partes que sobraram, ou seja, a parte esquerda do pai_2 com a parte direita do ponto de corte do pai_1. A Figura 10 mostra um exemplo de cruzamento com um ponto. Normalmente este operador apresenta desempenho pior que o multiponto.

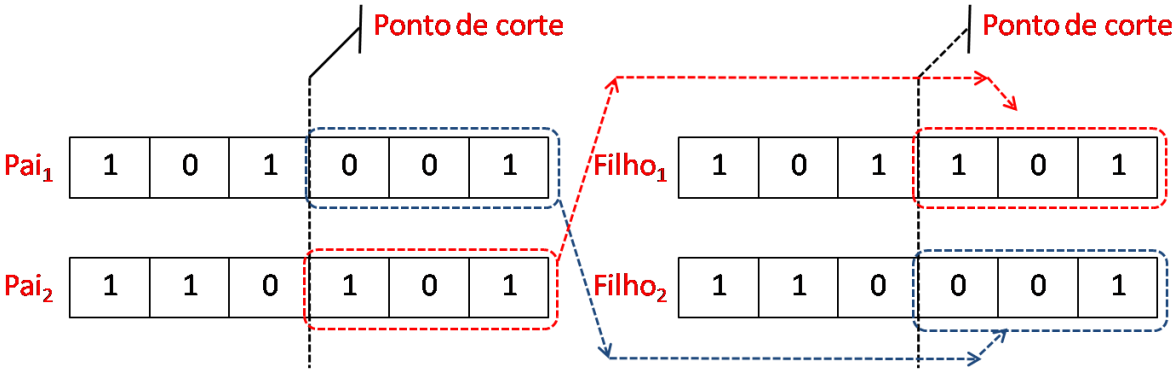

Figura 10 – Cruzamento utilizando um ponto de corte.

- Operadores multiponto: nesse tipo de operador, são utilizados dois ou mais pontos de corte. Porém, apesar de ser possível, segundo Linden (2008), não existe (ou pelo menos não são encontrados facilmente) na literatura, exemplos de caso de uso de *crossover* com mais de dois pontos de corte. Com dois pontos de corte, os bits à esquerda do $corte_1$ do pai_1 são enviados para o $filho_1$ e os bits entre o $corte_1$ e o $corte_2$ do mesmo pai são enviados para o $filho_2$. Os bits após o $corte_2$ permanecem no $filho_1$. O $filho_2$ é formado com os bits restantes. A Figura 11 mostra um exemplo que ajuda a esclarecer o uso deste tipo de operador.

A taxa de cruzamento (ρ_c) é um parâmetro de controle que deve ser levado em consideração quando da construção de um AG. Este parâmetro indica a probabilidade de ocorrência do cruzamento entre os indivíduos selecionados na população. É consenso entre os estudiosos que a probabilidade de ocorrência do cruzamento deve ser alta, fazendo com que ρ_c pertença geralmente ao intervalo [0,6; 0,9] (ARTERO, 2009). Quanto maior a taxa, mais rapidamente novas estruturas serão introduzidas na população. Porém, deve-se tomar certo cuidado, pois uma taxa muito alta poderá eliminar rapidamente indivíduos de boa qualidade da população (ZINI, 2009). Apesar de ser pouco provável, pode acontecer, em último caso, de não ocorrer o cruzamento entre os pais selecionados, caso no qual os filhos serão cópias exatas dos pais (ARTERO, 2009).

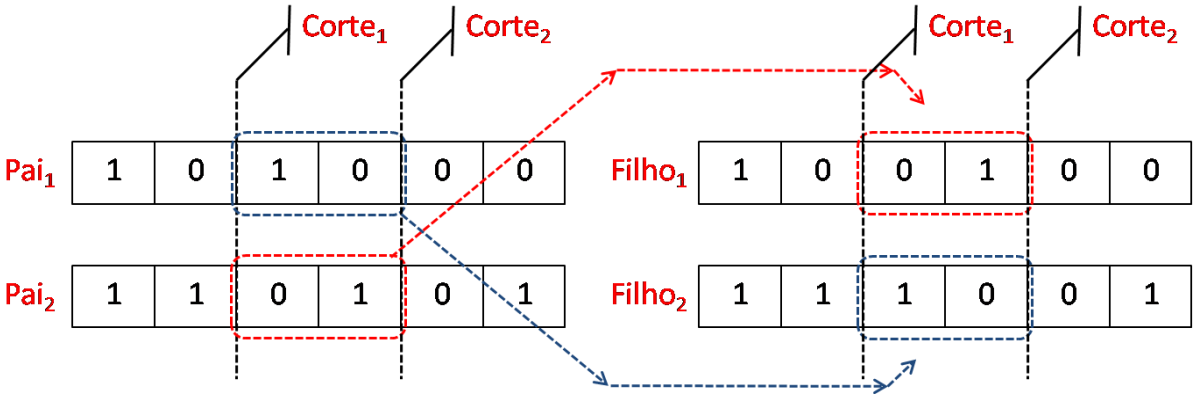

Figura 11 – Cruzamento utilizando dois pontos de corte.

3.3.8.2 Mutação

Depois de compostos os filhos, pode ser utilizado o operador de mutação, o qual modifica aleatoriamente o código genético destes, atribuindo a eles características que não são pertencentes aos pais e, conseqüentemente, cria uma variabilidade maior entre os descendentes. Caso esteja sendo utilizada uma codificação binária, alguns bits podem ser invertidos. Caso seja uma codificação real ou inteira, pequenos valores podem ser somados aleatoriamente (ARTERO, 2009).

Um parâmetro que deve ser considerado é a taxa de mutação (ρ_m), que corresponde à probabilidade de ocorrência da mutação nos indivíduos da população no decorrer do processo de evolução. Uma alta taxa de busca possibilita uma maior varredura do espaço de busca. Por outro lado, caso ρ_m seja muito alta, pode tornar a busca aleatória, como uma técnica de *random walk* (LINDEN, 2008). É consenso que a probabilidade de ocorrência da mutação é extremamente baixa, devendo o valor de ρ_m ser da ordem de 0,5% (LINDEN, 2008) a 1% (ARTERO, 2009).

3.3.9 Elitismo

O elitismo consiste em uma pequena modificação no módulo de população que tem o objetivo de aumentar a velocidade de convergência do algoritmo. A idéia básica consiste em

copiar os *n* melhores indivíduos de cada geração para a geração seguinte, garantindo assim que estas soluções não serão destruídas nas etapas de cruzamento e mutação, visto que não participarão destas etapas. Os melhores indivíduos, assim, não apenas serão passados de uma geração para outra, mas também participarão da criação de novos indivíduos. A manutenção do melhor indivíduo da geração *t* na população da geração *t*+1 garante que o melhor indivíduo da geração *t*+1 é, no pior caso (caso nenhum indivíduo melhor seja criado), igual ao melhor indivíduo da geração *t* (LINDEN, 2008).

3.3.10 Vantagens e Desvantagens do uso de Algoritmos Genéticos

Pelo exposto anteriormente, os AGs consistem em uma técnica de busca que apresenta uma série de características positivas, dentre as quais pode-se destacar (LINDEN, 2008; ZINI, 2009):

- São capazes de lidar com funções reais, discretas, booleanas, categóricas (não-numéricas) ou até mesmo formas híbridas dessas representações;
- Não utilizam apenas informação local, logo, não necessariamente ficam presos em máximos locais como certos métodos de busca. Esta característica torna os AGs técnicas extremamente adequadas para funções de comportamento complexo;
- Não são totalmente aleatórias, diferentemente de alguns métodos de pesquisa. Os AGs têm componentes aleatórios, mas utilizam também a informação da população corrente para determinar o próximo estado da busca;
- Tendo em vista que são buscas direcionadas e inteligentes, são adequados para tratar problemas com espaços de busca muito grandes, que não podem ser resolvidos com o uso de técnicas tradicionais;
- Apresentam um bom desempenho para uma grande escala de problemas;
- São de fácil implementação e proporcionam maior flexibilidade no tratamento do problema a ser resolvido.

Por outro lado, é possível citar também algumas desvantagens do uso dos AGs (LINDEN, 2008; ZINI, 2009):

- Necessidade de um grande número de avaliações de aptidão, o que acaba tornando o tempo de processamento maior que outros métodos;
- Dificuldade de achar o ótimo global exato;
- Grandes possibilidades de configurações, o que pode acabar complicando a resolução do problema tratado.

Apesar das dificuldades apresentadas, as mesmas podem ser superadas através de um estudo mais aprofundado dessa técnica e do uso de computadores com maior poder de processamento.

4 APRENDIZAGEM UBÍQUA

Neste capítulo serão mostrados alguns conceitos necessários para o entendimento de um ambiente de aprendizagem ubíqua. Serão mostrados conceitos relacionados à EaD e algumas formas de se prover este tipo de ensino. A Seção 4.1 traz conceitos básicos relacionados à EaD. A Seção 4.2 mostra conceitos e aspectos relacionados à aprendizagem móvel, uma das formas de se prover a EaD. A Seção 4.3 mostra conceitos inerentes a ambientes sensíveis ao contexto. A Seção 4.4 explica o que vem a ser a aprendizagem ubíqua propriamente dita.

4.1 EDUCAÇÃO A DISTÂNCIA

Diferentes conceitos de EaD podem ser encontrados na literatura. Entretanto, para o presente trabalho, será adotada uma definição elaborada pela Secretaria de Educação a Distância (SEED) do MEC, a qual pode ser encontrada no Decreto 5.622/2005 (BRASIL, 2005). Com o objetivo de instruir os sistemas de ensino, esse decreto define, em seu artigo 1º, a EaD como sendo uma

> [...] modalidade educacional na qual a mediação didático-pedagógica nos processos de ensino e aprendizagem ocorre com a utilização de meios e tecnologias de informação e comunicação, com estudantes e professores desenvolvendo atividades educativas em lugares ou tempos diversos (BRASIL, 2005).

De fato, conforme percebe-se através da definição apresentada, não há um modelo único de EaD, podendo ser consideradas como tal várias formas de ensino e aprendizagem. A natureza do curso, as condições reais do cotidiano e as necessidades dos estudantes são os elementos que irão definir a tecnologia e a metodologia mais adequadas, assim como a definição dos momentos presenciais necessários e obrigatórios (previstos em lei), estágios supervisionados, práticas em laboratórios de ensino, tutorias presenciais nos pólos descentralizados de apoio presencial e outras estratégias (SEED/MEC, 2007). Porém, dentre as várias formas de EaD, pode-se destacar a aprendizagem realizada através do uso de dispositivos móveis.

4.2 APRENDIZAGEM MÓVEL

A aprendizagem móvel refere-se ao uso de dispositivos móveis, tais como computadores de bolso, PDAs ou telefones celulares, em atividades de aprendizagem realizadas a partir de qualquer lugar e a qualquer hora, trazendo informação e conhecimento para uma situação e lugar onde a atividade de aprendizagem acontece (CASTILHO e AYALA, 2008).

Já Yau e Joy (2010) definem a aprendizagem móvel de uma perspectiva pedagógica, sendo considerada como tal qualquer tipo de aprendizagem que acontece quando o estudante tira vantagem de oportunidades de aprendizagem oferecidas por tecnologias móveis. De acordo com a primeira parte da definição, aprendizagem móvel não necessariamente implica o uso de dispositivos móveis, sendo suficiente apenas que a aprendizagem não aconteça em um local fixo. Nesse sentido, os autores consideram a mobilidade do estudante, e não a tecnologia móvel em si.

O presente trabalho considera a abordagem apresentada por Castilho e Ayala (2008). De acordo com estes autores, um ambiente de aprendizagem móvel implica o uso de dispositivos móveis computacionais para suportar atividades de aprendizagem, independentemente de local e horário. Nestes ambientes, os dispositivos móveis devem (i) desempenhar o papel de mediadores entre os estudantes, (ii) suportar essa interação e (iii) prover informação e recursos de conhecimento com base nas capacidades e nos interesses dos estudantes.

4.3 AMBIENTES SENSÍVEIS AO CONTEXTO

Sensibilidade ao contexto descreve um paradigma no qual o contexto de um usuário é levado em consideração para definir o seu perfil (MOORE, HU e WAN, 2008; MOORE, *et al.*, 2009). Não existe um consenso a respeito da definição de “contexto”, sendo este específico da aplicação e da intenção desejadas, requerendo a identificação das funções e propriedades dos domínios dos indivíduos (MOORE, *et al.*, 2009). Não há como definir de

forma precisa o que seria o contexto. Dessa forma, as características que serão consideradas para construção do contexto dependem da situação específica.

O contexto pode ser definido de acordo com informações relativas a propriedades que se combinam para descrever e caracterizar uma entidade e seu papel de uma forma legível pelo computador (MOORE, HU e WAN, 2008; MOORE, *et al.*, 2009). A localização do estudante, por exemplo, é uma característica importante para a definição do seu contexto em um ambiente para aprendizagem móvel. Entretanto, o contexto inclui mais do que apenas a localização. De fato, quase todas as informações disponíveis no momento da interação podem ser vistas como informações contextuais, dentre as quais destacam-se (MOORE, *et al.*, 2009):

- As diversas tarefas realizadas pelos usuários no sistema;
- A variada gama de dispositivos que se combinam para criar sistemas móveis, com a infraestrutura de serviços associada;
- Os recursos disponíveis (ex.: condição da bateria, tamanho de tela etc.);
- A situação física (ex.: nível de ruído, temperatura, nível de luminosidade etc.);
- A informação espacial (ex. localização, velocidade, orientação etc.);
- A informação temporal (ex. hora do dia, data etc.).

A lista acima, embora não contenha exatamente todas as informações que podem ser consideradas, serve para demonstrar a complexidade inerente ao contexto, sua natureza de domínio específico e a dificuldade em defini-lo e medi-lo (MOORE, *et al.*, 2009). Na tentativa de diminuir essa complexidade, Moore, Hu e Wan (2008) definem dois tipos gerais de contexto:

- Contexto estático (denominado *customização*): diz respeito à situação na qual um perfil do usuário é criado manualmente, estando o usuário ativamente envolvido no processo e tendo um elemento de controle.
- Contexto dinâmico (denominado *personalização*): refere-se à condição na qual o usuário é visto como sendo passivo, ou pelo menos com um pouco menos de controle. Nesse caso, o sistema monitora, analisa e reage dinamicamente ao comportamento do usuário e ao papel identificado.

Muitas aplicações de aprendizagem móvel sensíveis ao contexto utilizam contextos de aprendizagem a fim de adaptar ou sugerir apropriadamente atividades e conteúdos para os estudantes (YAU e JOY, 2010). Entretanto, os trabalhos encontrados na literatura não levam em consideração os recursos físicos dos dispositivos móveis, o que compromete uma definição mais precisa do contexto dos estudantes e, conseqüentemente, o acesso e a

navegação adequados nos conteúdos recomendados, uma vez que isto é diretamente influenciado pelas características dos dispositivos móveis utilizados pelos estudantes.

4.3.1 Uso de ontologias em ambientes sensíveis ao contexto

Formalmente, uma ontologia pode ser considerada como sendo:

> uma especificação formal explícita conceitualizando um domínio de interesse específico com estrutura e detalhe semânticos para representar conceitos, entidades, atributos, relacionamento, restrições, axiomas e suas propriedades de uma forma legível para o humano e para o computador (MOORE, HU e WAN, 2008 apud MOORE, *et al.*, 2009, p. 248).

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). Por exemplo, uma vinícola pode usar a ontologia de vinho para ligar sua programação de produção ao sistema de estoque no armazém do vinho.

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).

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, em Bittencourt *et al.* (2006) é descrita 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 tecnologia a personalização de ambientes de aprendizagem. Isso é alcançado levando-se em consideração características específicas do perfil de cada estudante. Em (MIN., WEI e LEI, 2008), é proposto um *framework* para adaptação em ambientes de aprendizagem, no qual dois modelos ontológicos são construídos: um modelo de conhecimento, no qual são guardadas as informações relacionadas ao objetivo do estudante, e um modelo do estudante, o qual armazena, além do seu objetivo, outras informações, tais como histórico de aprendizagem, conceitos dominados e grau de desempenho. Dessa forma, com base no modelo do estudante armazenado em uma ontologia, o sistema irá se adaptar de forma a cumprir os requisitos do estudante.

Em (RANI, ASHOK e PALANIVEL, 2009), é apresentada uma abordagem para personalização de aprendizagem eletrônica (ou *electronic learning* ou ainda *e-learning*) com base em ontologia, na qual a troca de informações é mantida por serviços Web *(web services*), sob uma arquitetura orientada a serviços. A idéia principal do sistema é identificar requisitos do usuário, ou seja, as preferências dos estudantes e suas características, e criar o modelo do estudante. Nesse modelo constará o seu conhecimento expresso através de um conjunto de palavras-chave pertencentes a uma ontologia comum para, de acordo com o modelo de cada estudante, adaptar o curso de forma individual.

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

4.4 APRENDIZAGEM UBÍQUA

Como é possível perceber pela definição de aprendizagem móvel, a principal vantagem dessa forma de EaD é o fato de permitir que a aprendizagem ocorra a qualquer hora e em qualquer localidade. Porém, apesar de prover mobilidade, ela não fornece uma aprendizagem que seja sensível ao contexto do estudante (MANDULA, MEDA e KAMBHAM, 2011).

Reunindo as definições de aprendizagem móvel e de sensibilidade ao contexto, surge o conceito de Aprendizagem Ubíqua (também conhecida como *u-learning*, de *ubiquitous learning*). Ou seja, a aprendizagem ubíqua pode ser considerada como sendo a aprendizagem móvel que é realizada levando-se em consideração as características do contexto dos

estudantes, provendo a estes conteúdos adaptados às suas necessidades (MANDULA, MEDA e KAMBHAM, 2011). Assim, a aprendizagem ubíqua provê um novo paradigma, através do uso de dispositivos móveis, que provê um serviço de forma transparente aos estudantes (LOUREIRO, *et al.*, 2009). A Figura 12 mostra um diagrama representando a aprendizagem ubíqua como a intersecção entre a aprendizagem móvel e a sensibilidade ao contexto.

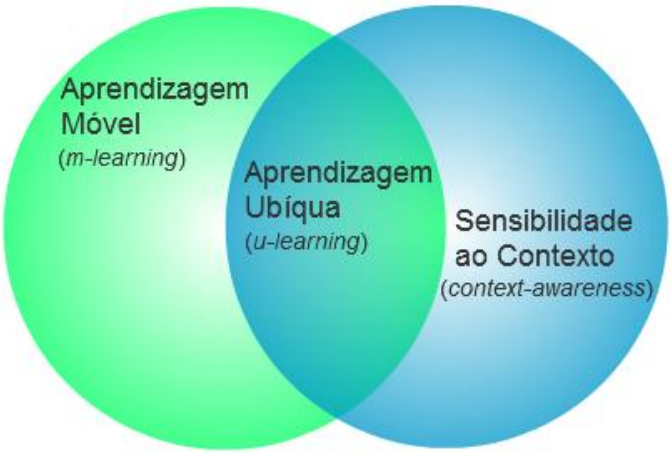

Figura 12 – Representação da Aprendizagem Ubíqua.

A mobilidade permitida ao usuário devido ao uso de dispositivos móveis em aplicações ubíquas torna ainda mais importante a consideração do contexto de um estudante, visto que as características do estudante podem se modificar a qualquer momento. Essas modificações podem ocorrer em diversos aspectos (ex. condições físicas, recursos físicos disponíveis, recursos computacionais etc.) (LOUREIRO, *et al.*, 2009), conforme mencionado na Seção 4.3.

Vale ressaltar que não existe, na literatura, um consenso a respeito da definição precisa de computação ubíqua. Em (MANDULA, MEDA e KAMBHAM, 2011), a definição de aprendizagem ubíqua envolve a aprendizagem móvel e a sensibilidade ao contexto. Já para outros autores, a aprendizagem ubíqua envolve outras características, como, por exemplo, a computação pervasiva (ZHAO e WANG, 2011). Não faz parte do objetivo deste trabalho levantar uma discussão mais profunda a respeito dessa definição. No presente trabalho será considerada a definição provida em (MANDULA, MEDA e KAMBHAM, 2011), conforme explicado anteriormente.

5 OBJETOS DE APRENDIZAGEM

Este capítulo descreve aspectos relacionados aos objetos de aprendizagem (OAs) e aos padrões utilizados para o seu desenvolvimento. A Seção 5.1 traz uma discussão sobre definições de OAs apresentadas por diversos autores. A Seção 5.2 traz uma breve discussão sobre o conceito de Objetos de Aprendizagem Móvel (OAMs). A Seção 5.3 mostra os componentes principais dos padrões de OAs mais conhecidos.

5.1 DEFINIÇÃO DE OBJETOS DE APRENDIZAGEM

Um conceito relevante quanto ao conteúdo didático utilizado em EaD é o de OA. Segundo o Comitê de Padronização de Tecnologias de Aprendizagem (LTSC - *Learning Technology Standard Committee*), do *Institute of Electrical and Electronics Engineers* (IEEE), um OA é uma entidade material educacional, digital ou não, que pode ser usada, reutilizada e referenciada durante alguma manifestação de ensino-aprendizagem apoiada por recursos tecnológicos (LTSC, 2002; RODOLPHO, 2009).

Já Wiley (2002) apresenta uma definição que aproxima os OAs da informática, argumentando que estes objetos são recursos digitais que podem ser utilizados para dar suporte ao ensino, e que são construídos de forma a dividir o conteúdo em pequenos módulos reutilizáveis em diversos ambientes, seguindo os princípios da orientação a objetos.

Segundo Rodolpho (2009), para ambientes de EaD, os OAs são restritos aos formatos digitais. Outra característica dos OAs é que eles podem ser reutilizados em diferentes contextos de aprendizagem e combinados entre si, compondo outros OAs. Essa reutilização aperfeiçoa a produção do conteúdo e, conseqüentemente, diminui o seu custo, além de oferecer maior flexibilidade na elaboração e organização das atividades (RODOLPHO, 2009).

De acordo com Mendes, Souza e Caregnato (2004), são sete as características dos OAs, enquanto que para ADL (2008), são quatro as características apresentadas pelos OAs. A Tabela 2 mostra um comparativo entre as propriedades dos OAs apresentadas por estes autores.

Um OA, além do conteúdo propriamente dito, possui uma estrutura contendo metadados que permitem, através dos seus elementos e atributos, descrever o seu conteúdo,

sua formatação, como esse conteúdo é apresentado, além de outras informações (ex. autor do conteúdo, data da criação, dados pedagógicos etc.) (RODOLPHO, 2009). Esses metadados são criados baseando-se em padrões, os quais serão apresentados posteriormente.

Tabela 2 – Propriedades de um Objeto de Aprendizagem.

Propriedade dos OAs	**Descrição**	(MENDES, SOUZA e CAREGNATO, 2004)	(ADL, 2008)
Reusabilidade (*Reusability*)	Um OA pode ser reutilizado diversas vezes, em diversos ambientes de aprendizagem.	X	X
Acessibilidade (*Acessibility*)	Um OA pode ser facilmente acessado via Internet para ser usado em diversos locais.	X	X
Interoperabilidade (*Interoperability*)	Um OA pode ser utilizado e gerenciado através de diferentes plataformas de hardware, sistemas operacionais e navegadores Web, possibilitando intercâmbio efetivo entre diferentes sistemas.	X	X
Durabilidade (*Durability*)	Um OA pode ser utilizado e reutilizado independentemente da tecnologia.	X	X
Adaptabilidade (*Adaptability*)	Um OA é adaptável a qualquer ambiente de ensino.	X	---
Granularidade (*Granularity*)	O conteúdo de um OA deve ser fragmentado para facilitar sua reutilização.	X	---
Metadados (*Metadata*)	O conteúdo de um OA pode ser descrito com propriedades de um objeto, como título, autor, data, assunto, etc.	X	---

Em resumo, a idéia central do conceito de OAs é permitir que os educadores que elaboram os conteúdos educacionais construam componentes educativos relativamente pequenos que possam ser usados em diferentes contextos de aprendizagem. Ou seja, são conteúdos digitais que permitem ou facilitam alcançar um objetivo educacional e sua reusabilidade (CASTILHO e AYALA, 2008).

5.2 OBJETOS DE APRENDIZAGEM MÓVEL

Além do conceito de OA, outro conceito relevante é o de Objeto de Aprendizagem Móvel (OAM), do inglês *Mobile Learning Object*. Um OAM pode ser considerado como sendo uma entidade de informação i) digital, ii) interativa, iii) adaptável e reusável em diferentes contextos, iv) destinada a alcançar um objetivo educacional, v) criada para ser usada em um ambiente de aprendizagem móvel, e vi) capaz de suportar diferentes abordagens de aprendizagem e perspectivas de interação (CASTILHO e AYALA, 2008).

Segundo Ayala e Castilo (2008), OAMs não devem apenas ser projetados e desenvolvidos como OAs para computadores pessoais com uma adaptação para a tela dos dispositivos móveis. Eles devem ser projetados de acordo com as características dos dispositivos móveis, como artefatos pessoais que suportam a aprendizagem em situações e contextos que são diferentes daqueles dos computadores de mesa e de laptops. Os mesmos autores defendem ainda que o desenvolvimento, o uso e reuso de OAMs devem ser baseados em uma arquitetura que suporte abordagens de aprendizagem apropriadas, aliadas ao contexto correspondente necessário.

Apesar da definição de Ayala e Castilo (2008) deixarem explícita a necessidade de um desenvolvimento apropriado de OAs para dispositivos móveis, o que se evidencia na prática é que, de um modo geral, os OAs projetados para computadores pessoais são utilizados nos dispositivos móveis. Dessa forma, quando em um ambiente de ensino, deve ser considerado se o dispositivo móvel de um estudante suporta, de fato, os recursos providos por determinado OA.

5.3 PADRÕES DE OBJETOS DE APRENDIZAGEM

Apesar dos benefícios em se utilizar OAs, há de se considerar os problemas enfrentados na criação de OAs digitais. Rodolpho (2009) descreve um conjunto de dificuldades que são enfrentadas durante a criação de OAs digitais, dentre as quais destacam-se:

- Definição da estrutura de navegação;
- Adequação do conteúdo de uma mídia escrita para uma mídia eletrônica;
- Atendimento aos aspectos pedagógicos de ensino; e

- Integração do OA com diferentes tipos de ambientes de EaD.

Para solucionar tais problemas, são utilizados padrões de OAs. Esses padrões constituem um meio de organizar os dados de um OA para prover comunicação entre diferentes ambientes computacionais, bem como garantir o seu acesso e usabilidade, além de prover interoperabilidade (DIAS, *et al.*, 2009). Os mesmos autores relatam que esses padrões são divididos, de acordo com suas funcionalidades, em: padrões de metadados, de empacotamento, de interface e comunicação e de integração. Dentre estes, os padrões de metadados e de integração se adéquam melhor à realização deste trabalho. Nas Subseções 5.3.1 e 5.3.2 são explicados os motivos pelos quais esses tipos de padrões são os mais adequados para o presente trabalho.

5.3.1 Padrões de Metadados

Segundo Li,Yang e Liu (2008), a definição mais simples e objetiva que pode ser dada a metadados é a de dados sobre dados. Ou seja, os metadados são dados de objetos e descrevem informações sobre o conteúdo destes. Os padrões de metadados são utilizados na identificação de recursos, auxiliando na filtragem de uma busca e na recuperação de um registro ou de um OA (DIAS, *et al.*, 2009).

Um exemplo de padrão de metadados é o LOM (*Learning Object Metadata*) (LTSC , 2002), o qual foi desenvolvido pelo LTSC. O propósito desse padrão é facilitar a busca, a avaliação, a aquisição e o uso de OAs, tanto por estudantes como por instrutores, ou até mesmo por processos de software automatizados. Além disso, esse padrão também facilita a distribuição e a troca de OAs, permitindo o desenvolvimento de catálogos e inventários que levam em consideração a diversidade de contextos lingüísticos e culturais nos quais os OAs e seus metadados são reusados. Através da especificação de um esquema de dados comum, este padrão garante que as linguagens de OAs tenham um alto grau de interoperabilidade semântica (LTSC , 2002).

Os elementos de dados descrevem um OA e são agrupados em categorias. A estrutura de metadados básica do LOM é definida em nove categorias, conforme mostra a Tabela 3 (LTSC , 2002).

Tabela 3 – Categorias do padrão LOM e suas descrições.

Categoria	Descrição
Geral (*General*)	Agrupa informações gerais que descrevem o OA como um todo.
Ciclo de vida (*Lifecycle*)	Agrupa as características relacionadas à história e ao estado atual do OA, além dos aspectos que afetaram o OA durante sua evolução.
Meta-Metadado (*Meta-Metadata*)	Reúne informações sobre as instâncias dos metadados em si, ao invés do OA que a instância do metadado descreve.
Técnico (*Technical*)	Agrupa as características e os requisitos técnicos do OA.
Educacional (*Educational*)	Reúne as características educacionais e pedagógicas do OA.
Direitos (*Rights*)	Descreve os direitos de propriedade intelectual e as condições de uso para o OA.
Relação (*Relation*)	Agrupa as características que definem o relacionamento entre o OA e outros OAs relacionados.
Anotação (*Annotation*)	Provê comentários sobre o uso educacional do OA e informações sobre quando e por quem os comentários foram criados.
Classificação (*Classification*)	Descreve o OA no que diz respeito a um sistema de classificação particular.

4.3.2 Padrões de Integração

Um padrão de integração, como o próprio nome já indica, unifica em um modelo de referência diferentes tipos de padrões, tais como padrões de metadados, empacotamento, interface e comunicação, desenvolvidos por outras organizações (DIAS, *et al.*, 2009).

O padrão de integração SCORM (*Sharable Content Object Reference Model* – Modelo de Referência de Objetos de Conteúdos Compartilháveis), desenvolvido pela ADL (*Advanced Distributed Learning*), integra um conjunto de padrões técnicos, especificações e orientações designadas a atender os requisitos de alto nível do SCORM, que são: sistemas e conteúdo acessíveis, interoperáveis, duráveis e reutilizáveis (ADL, 2008).

O conteúdo no padrão SCORM pode ser compartilhado com os estudantes através de qualquer Sistema de Gestão de Aprendizagem, também chamado de LMS (*Learning Management System*), que seja compatível com o SCORM e que use a mesma versão deste (ADL, 2008). Um LMS é um pacote de software que é tipicamente acessível através da

internet e é usado para administrar um ou mais cursos para um ou mais estudantes. De forma simplificada, um LMS permite que estudantes se autentiquem, se registrem em cursos, acessem conteúdos educacionais, realizem tarefas e então armazenem os registros de seus desempenhos (ADL, 2008).

A versão atual do padrão (SCORM 2004 4ª edição, versão 1.1) está especificada em quatro livros (ADL, 2008; SCORM, 2009):

- Visão Geral (*The SCORM Overview*): mostra uma visão geral do SCORM 2004, ou seja, sua história, *status* da versão atual e próximos passos da ADL e do SCORM, além de uma introdução à funcionalidade arquitetural do SCORM.
- Modelo de Agregação de Conteúdo (*The SCORM Content Agregation Model*): mostra (i) como empacotar conteúdos para permitir o intercâmbio entre sistemas, (ii) como descrever o conteúdo de forma a facilitar a sua busca e descoberta e (iii) como realizar o seqüenciamento do conteúdo no *manifesto* (arquivo padrão que contém os recursos do OA).
- Ambiente de Execução (*The SCORM Runtime Environment*): mostra os elementos do modelo de dados do SCORM que permitem recolher e armazenar dados sobre o desempenho dos estudantes e interação com conteúdo instrucional.
- Seqüenciamento e Navegação (*The SCORM Sequencing & Navigation*): explica como o seqüenciamento pode ser aplicado ao conteúdo para recomendar o modo através do qual os estudantes recebem conteúdo do LMS.

A Figura 13 ilustra a organização destes livros como um conjunto de especificações de outras organizações contidas ou referenciadas no modelo.

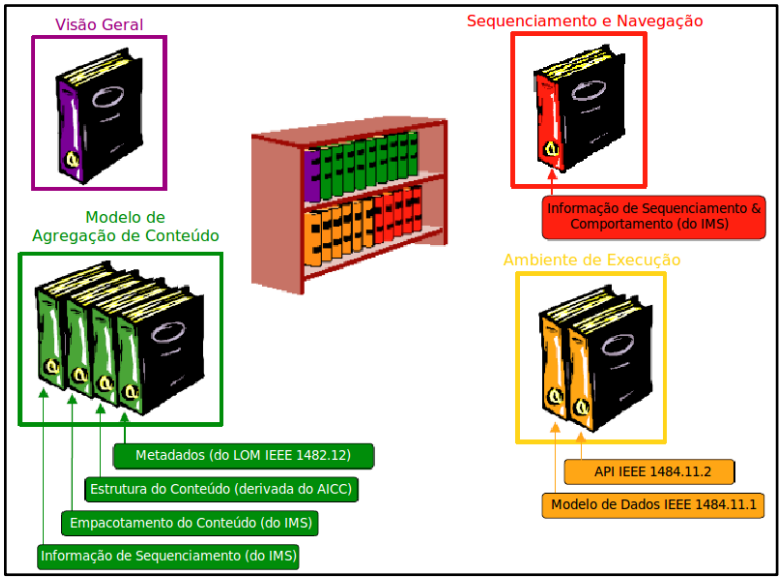

Figura 13 – Padrão SCORM como um conjunto de especificações. Fonte: Adaptado de **(**DUTRA e TAROUCO, 2006).

No presente trabalho, utilizaremos o padrão SCORM como padrão dos OAs, tanto por ser um padrão amplamente difundido na literatura, como também por ser composto de uma série de outros padrões. Utilizando o SCORM, é possível utilizar, inclusive, os recursos providos pelo padrão LOM. Assim, é possível se beneficiar dos melhores recursos oferecidos por cada um dos padrões.

6 MODELAGEM DO PROBLEMA DE RECOMENDAÇÃO DE OBJETOS DE APRENDIZAGEM

Para realizar uma recomendação de OAs adequada ao contexto do estudante, é essencial levar em consideração características que podem ser limitantes durante a aprendizagem do mesmo. Este capítulo apresenta uma descrição detalhada de quais características serão consideradas para a realização da recomendação de OAs implementada no ambiente multiagente deste trabalho. A Seção 6.1 traz uma descrição textual dos aspectos considerados para a recomendação de OAs. A Seção 6.2 mostra detalhes de como foi projetado o AG. A Seção 6.3 traz informações detalhadas de como foi codificado o AG.

6.1 ASPECTOS CONSIDERADOS

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, serão consideradas as características detalhadas nas subseções seguintes.

6.1.1 Afinidade com o curso

Em um repositório de OAs pode haver muitos destes que não possuam o mínimo relacionamento com alguns cursos que estão sendo ofertados. Desta forma, é interessante definir quão fortemente relacionado a um curso é determinado OA. Assim, este aspecto representa justamente o grau de relacionamento do conteúdo de um OA a determinado curso. Esse grau de afinidade é representado em uma escala que varia de 0 (nenhuma afinidade) a 5 (alta afinidade).

Determinado OA pode estar relacionado a um ou vários cursos, sendo o administrador de cada curso responsável por determinar, manualmente, quais OAs são afins ao curso e qual o seu grau de afinidade.

6.1.2 Afinidade com o dispositivo móvel

Assim como determinados OAs estão mais relacionados a alguns cursos do que a outros, os mesmos são mais adequados a alguns tipos de dispositivos móveis do que a outros. Isso acontece devido ao fato de que cada OA pode conter uma série de diferentes tipos de recursos, tais como vídeo, imagem, texto, entre outros. Logo, há de se considerar que nem todos os dispositivos serão capazes de suportar todos os recursos.

O grau de afinidade com o dispositivo é definido de acordo com uma faixa que varia de 0 (nenhum recurso suportado) a 5 (todos os recursos suportados). A informação de qual dispositivo está sendo utilizado para acessar o curso é capturada a partir da requisição HTTP enviada do dispositivo móvel para o servidor no qual se encontra o LMS. Em seguida, em posse dessas informações, é possível verificar em uma base de dados, chamada WURFL (WURFL, 2011), quais recursos aquele dispositivo móvel suporta.

6.1.3 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. 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.

6.1.4 Localização corrente

Outra informação que pode ser levada em consideração para a definição do contexto do estudante é a localidade na qual ele se encontra. Em posse dessa informação, é possível verificar, nos metadados do padrão de OAs LOM, a incidência de palavras relacionadas à localização do estudante. Desta forma, é possível recomendar OAs relacionados à localização corrente do estudante.

Com base nisso, são capturadas e consideradas informações do país, estado e cidade na qual o estudante se encontra. Em seguida, no momento da recomendação, os metadados dos OAs são verificados em busca dessas informações. Quanto mais palavras relacionadas à localização do estudante forem encontradas, maior será o valor atribuído a essa variável (máximo 5). Por outro lado, caso não seja encontrada nenhuma informação relacionada, será atribuída a pontuação mínima (valor 0).

6.1.5 Escolhas de outros estudantes

Outro ponto de interesse a ser levado em consideração é a escolha realizada por outros estudantes. Esse é um recurso muito utilizado, principalmente, em comércio eletrônico (*e-commerce*). O OA mais acessado de determinado curso recebe a maior pontuação para essa variável (valor 5). Os demais OAs dentro do mesmo curso recebem uma pontuação proporcional a esse valor, de acordo com a quantidade de acessos que receberam. O OA com a menor quantidade de visualizações recebe a pontuação mínima (valor 0).

A informação de quais OAs são mais acessados é levada em consideração em conjunto com a informação de afinidade do OA com o curso, pois de nada adiantaria para a recomendação um OA ser o mais acessado e não possuir nenhuma afinidade com o curso em questão.

6.1.5 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.

Essas informações estão localizadas, mais especificamente, no padrão LOM, o qual é parte integrante do padrão SCORM.

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.

6.2 PROJETO DO ALGORITMO GENÉTICO

O projeto do AG é composto de classes auxiliares, responsáveis por capturar as informações, e do núcleo do AG, ou seja, das classes que efetivamente realizam a abordagem genética. O núcleo da abordagem baseada em AG criada para o presente trabalho foi codificada basicamente em duas classes:

- Classe *GA*[4], a qual consiste na representação do AG propriamente dito, contemplando todos os parâmetros necessários ao AG.
- Classe *ChromossomeGA*, que representa as informações de cada um dos OAs no repositório e as operações que podem ser aplicadas sobre estes.

A Figura 14 mostra de forma simplificada os principais componentes dessas classes e como elas se relacionam entre si.

[4] A sigla *GA* é uma abreviação do termo, em inglês, *Genetic Algorithm*, para facilitar a publicação em eventos e periódicos internacionais.

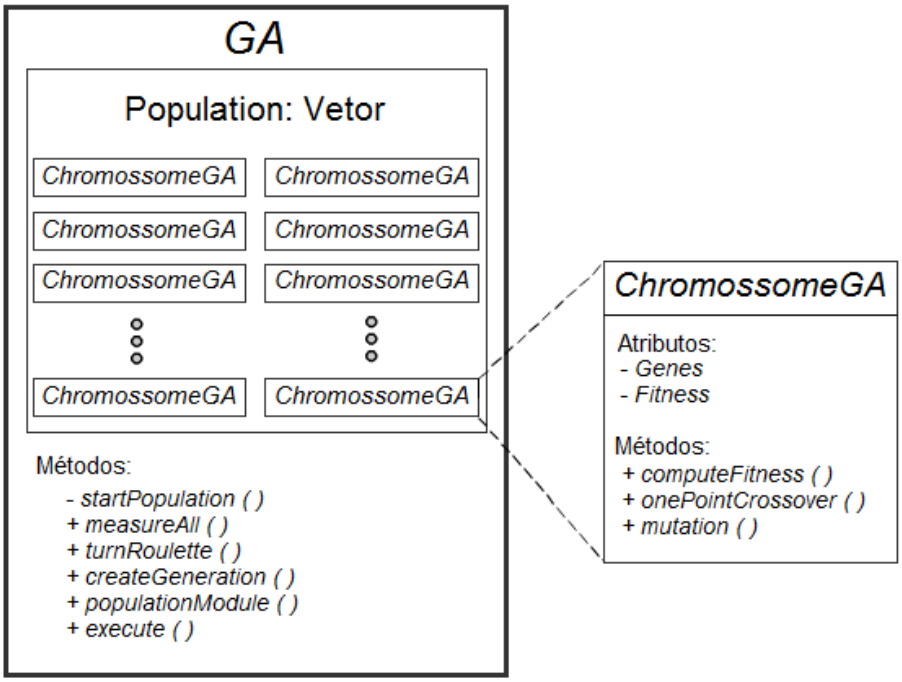

Figura 14 – Estrutura e relacionamento entre as classes *GA* e *ChromossomeGA*.

Como é possível perceber na Figura 14, várias instâncias da classe *ChromossomeGA* são utilizadas pela classe *GA*. As Subseções a seguir trazem uma descrição mais detalhada dessas classes.

6.2.1 Classe *GA*

A Classe *GA* possui a representação dos parâmetros necessários para a execução do AG. Alguns desses parâmetros são definidos como constantes, enquanto outros são passados através do construtor da classe no momento em que esta é instanciada pelo agente de software responsável (a descrição desse agente será vista no Capítulo 7). Dentre os principais parâmetros, destacam-se os seguintes:

- Quantidade de gerações do AG: foi definido um total de 50 gerações para o AG criado.
- Informações do estudante: atributo que é uma instância da entidade *Student*, criada especificamente para guardar as informações do estudante. Esse objeto é manipulado por um dos agentes de software (a descrição desse agente será vista no Capítulo 7).
- Taxa de *crossover* (ρ_c): o valor de ρ_c foi definido como 60% (ou 0,6).

- Taxa de mutação (ρ_m): foi definido um valor de 5% (ou 0,05) para ρ_m.
- Tamanho do cromossomo: essa informação será melhor detalhada na próxima subseção.

Além dos parâmetros citados, essa classe também engloba os métodos necessários para execução do AG, dentre os quais deve-se destacar:

- *startPopulation*: inicializa a população de cromossomos do AG.
- *measureAll*: calcula a aptidão de todos os cromossomos e, ao final de sua execução, salva o resultado de todas as avaliações para auxiliar no método do giro da roleta.
- *turnRoulette*: implementa a técnica do giro da roleta, também conhecida como "Roleta Viciada" (LINDEN, 2008), para escolher os pais que participarão do próximo cruzamento.
- *createGeneration*: cria uma nova geração da população.
- *populationModule*: controla o tamanho da população, mantendo sempre o tamanho da população existente como sendo igual ao número de filhos gerados.
- *execute*: executa o AG de acordo com os parâmetros já instanciados e o número de gerações definido.

6.2.2 Classe *ChromossomeGA*

A classe *ChromossomeGA* possui as informações de cada um dos OAs pertencentes ao repositório e as operações que podem ser aplicadas sobre estes. As instâncias dessa classe são manipuladas pela classe *GA*. Merecem destaque os seguintes atributos da classe *ChromossomeGA*:

- Genes: cadeia de caracteres contendo todos os genes que formam um cromossomo.
- *Fitness*: valor que representa a aptidão de um OA em determinado contexto.

Dentre os métodos dessa classe, merecem destaque os seguintes:

- *computeFitness*: calcula a aptidão de cada um dos cromossomos individualmente.

- *onePointCrossover*: realiza o crossover de um ponto entre dois pais escolhidos através do método do giro da roleta.
- *mutation*: realiza a mutação dos filhos.

6.3 ASPECTOS DE CODIFICAÇÃO DO ALGORITMO GENÉTICO

As descrições das classes *GA* e *ChromossomeGA* fornecem uma visão de projeto para auxiliar no entendimento de como o AG foi criado. Entretanto, para conseguir uma melhor compreensão do AG, é necessário entender alguns aspectos mais detalhados relacionados à sua codificação. As próximas subseções discutem alguns aspectos importantes relacionados a esse assunto.

6.3.1 Representação Cromossomial

O AG é alimentado de acordo com o resultado alcançado avaliando-se os aspectos descritos na seção 6.1. Para atender a essa abordagem, cada cromossomo (ou indivíduo da população) é composto conforme o exemplo de cromossomo mostrado na Figura 15.

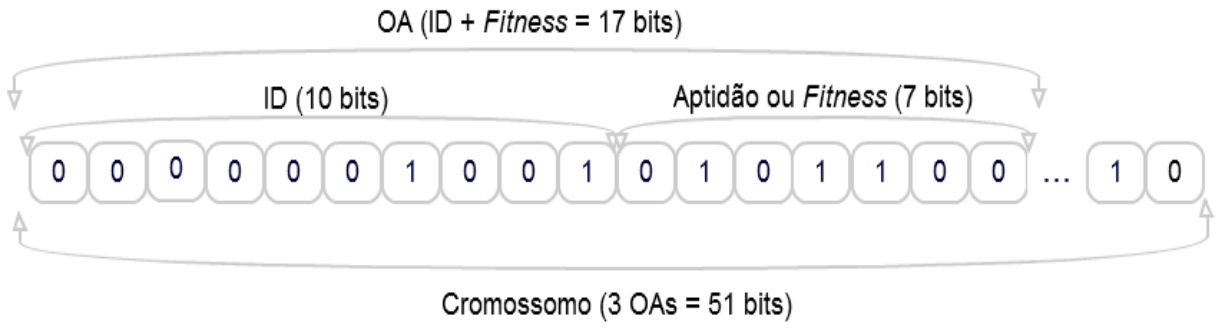

Figura 15 – Representação cromossomial utilizada.

A população é composta de cromossomos que possuem, cada um, cinqüenta 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 seção anterior. Assim, com cada OA utilizando

dezessete bits, e com uma representação de cromossomo de cinqüenta 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 (MOODLE, 2011), é criado um diretório no sistema de arquivos para abrigar os arquivos desse 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 algoritmo genético, 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 seja pareça realmente útil.

6.3.2 Tamanho da população

O tamanho da população utilizada depende da quantidade de OAs no repositório. Devido ao que já foi explicado anteriormente, 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.

6.3.3 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: uma caso a quantidade de OAs no repositório possua uma divisão exata por três e outra 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.

7 *MobiLE* - AMBIENTE BASEADO EM AGENTES DE SUPORTE À APRENDIZAGEM UBÍQUA

Este capítulo descreve o ambiente de aprendizagem ubíqua desenvolvido neste trabalho, o qual recebe o nome de *MobiLE*, do inglês *Mobile Learning Environment*. A Seção 7.1 traz uma descrição das principais ferramentas, bibliotecas e frameworks que foram utilizados para o desenvolvimento do *MobiLE*. A Seção 7.2 traz uma descrição do ambiente de suporte à aprendizagem ubíqua desenvolvido. A Seção 7.3 mostra a arquitetura completa do ambiente e apresenta uma descrição dos principais componentes do mesmo. A Seção 7.4 mostra detalhes da arquitetura multiagente do ambiente proposto no presente trabalho e da modelagem dos agentes, além de descrever como é realizada a comunicação entre os agentes.

7.1 FERRAMENTAS UTILIZADAS

Para o desenvolvimento da infraestrutura de software do ambiente *MobiLE* 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.

7.1.1 LMS Moodle

O Moodle (*Modular Object-Oriented Dynamic Learning Environment*) é um LMS 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).

7.1.2 Framework MLE

A interface gráfica do ambiente foi desenvolvida a partir de um framework Java de código-fonte aberto chamado MLE (*Mobile Learning Engine*) (MLE, 2011). O framework MLE pode ser utilizado para a construção da interface gráfica para o dispositivo móvel do usuário. Ele funciona como um complemento para o LMS Moodle, podendo ser acessado, também, através de uma interface Web adaptada para dispositivo móvel. No presente trabalho, utilizamos este framework para desenvolver a interface do aplicativo, que pode ser instalada no dispositivo móvel de cada estudante.

Visto que o aplicativo foi desenvolvido utilizando um framework Java, é necessário, logicamente, que o dispositivo móvel do estudante suporte a execução de aplicações criadas sobre a arquitetura J2ME (JME, 2011). Além disso, para que o ambiente desenvolvido funcione corretamente, se faz necessário instalar um servidor de mensagens[5] e um servidor de gateway[6], de forma a permitir uma comunicação adequada entre a interface gráfica instalada no dispositivo e o servidor no qual se encontra o Moodle. Esses servidores são disponibilizados pelo próprio MLE.

7.1.3 Framework JADE

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

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

[5] Servidor escrito em J2SE (Java padrão) usado pelo cliente MLE para recebimento instantâneo de mensagens.
[6] Servidor de *proxy* escrito em J2SE usado pelo MLE para se conectar à Internet de uma forma mais eficiente, permitindo que dispositivos móveis que possuem baixa largura de banda recebam apenas os dados necessários.

ACC (ver Subseção 2.5.4). 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).

7.1.4 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.

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

7.2 DESCRIÇÃO DO MOBILE

O *MobiLE* consiste em uma arquitetura multiagente, integrada a um LMS, com interface gráfica acessível através de um dispositivo móvel, permitindo que o estudante possa receber conteúdos adequados às suas necessidades e possa acessá-los a partir de qualquer localidade e em qualquer horário. A Figura 16 mostra a tela principal do *MobiLE* e uma recomendação de OA executando em um simulador.

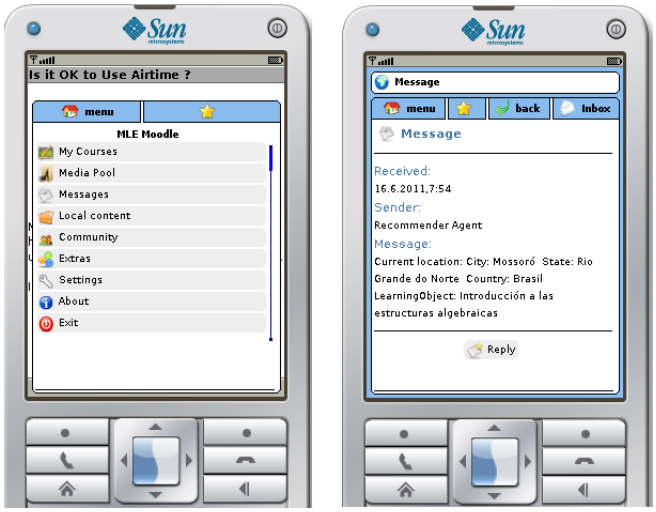

Figura 16 – Tela inicial do ambiente e sugestão de OAs.

O *MobiLE* deve ser utilizado juntamente com o Moodle (MOODLE, 2011), pois ele funciona como um componente que deve ser integrado à infraestrutura deste. No momento de inscrição no sistema, o estudante deve fornecer algumas informações adicionais, úteis para a definição do seu contexto, dentre as quais pode-se citar: a área de interesse do estudante, o horário preferido de estudo, sua localização atual, entre outras. Essas informações servirão para definir o contexto do estudante. Maiores detalhes a respeito do propósito dessas informações serão fornecidos na Seção 7.3. A Figura 17 mostra a interface do Moodle adaptada para capturar as informações mencionadas.

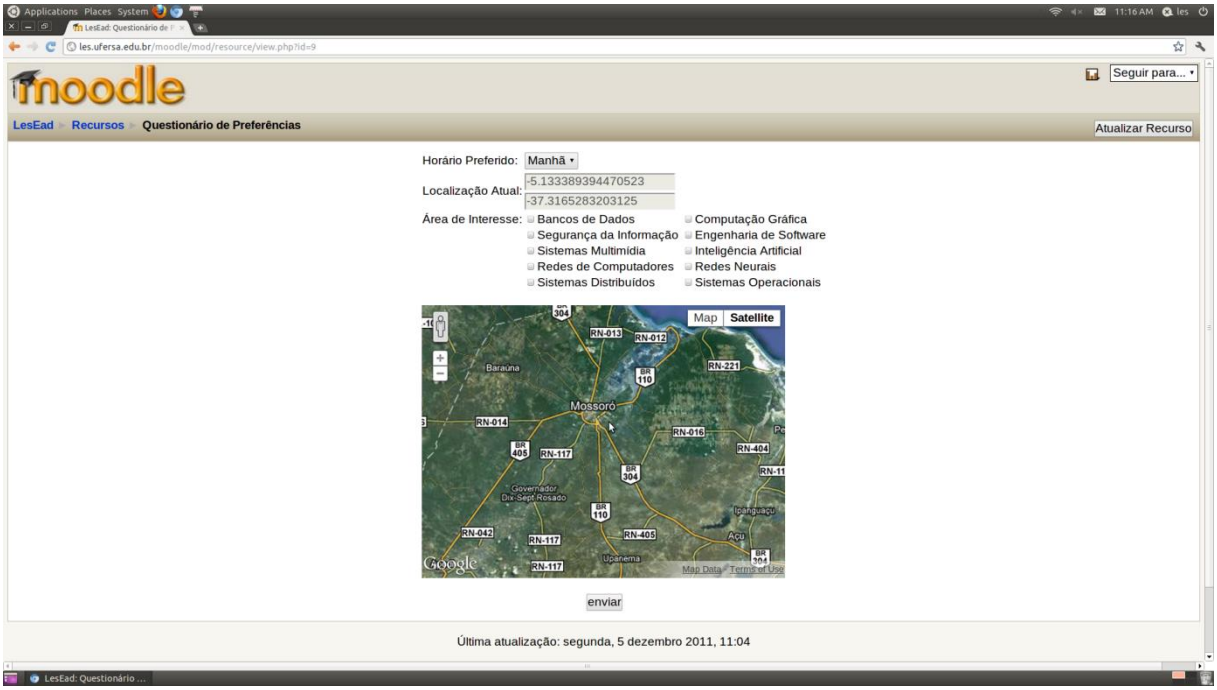

Figura 17 – Informações de preferências no Moodle.

Vale ressaltar que a idéia principal de utilização do *MobiLE* é que o mesmo funcione como uma ferramenta de suporte à aprendizagem do estudante, não apenas por prover um

conteúdo que seja sensível ao contexto deste, mas também devido à flexibilidade que lhe é fornecida. Porém, não faz nenhum sentido utilizar o *MobiLE* em um local no qual esteja disponível o acesso à interface tradicional do Moodle através de um computador pessoal ou notebook, visto que uma interface móvel apresenta limitações que, geralmente, não são apresentadas nestes (Ex.: tamanho de tela reduzido, menor capacidade de processamento, ausência de alguns recursos de processamento etc.).

Além disso, também vale destacar que o objetivo principal do presente trabalho consiste em prover a infraestrutura necessária para recomendação adequada de OAs. Logo, não faz parte da contribuição principal deste trabalho a construção de uma interface gráfica inovadora, visto que o MLE fornece recursos suficientes para construção de uma interface gráfica amigável e que atende às necessidades de aprendizagem dos alunos.

Para utilização do *MobiLE*, o estudante deve, inicialmente, estar matriculado em algum dos cursos disponibilizados através do Moodle. Uma vez que o estudante se autentique no sistema, ele tem a possibilidade de fazer o download do aplicativo para ser instalado no seu dispositivo móvel, de acordo com o modelo deste dispositivo. Após a instalação do aplicativo no dispositivo, o estudante pode acessar os cursos nos quais está matriculado e receber OAs adequados às condições do contexto no qual está inserido.

7.3 ARQUITETURA DO AMBIENTE

A arquitetura do ambiente proposto neste trabalho é representada na Figura 18.

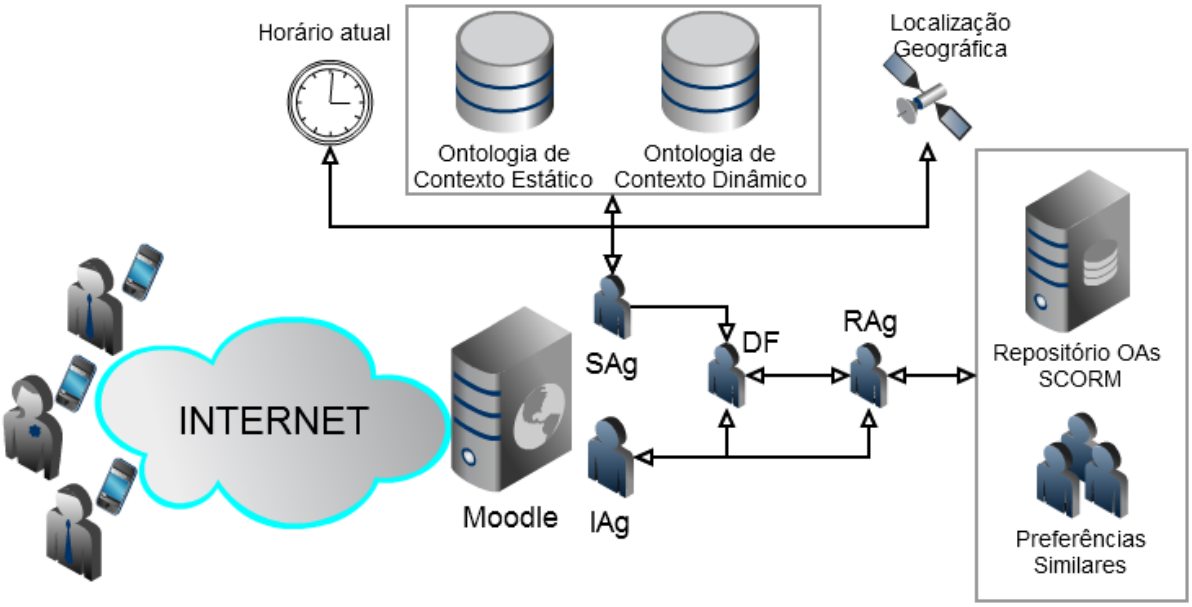

Figura 18 – Arquitetura do ambiente MobiLE.

Os principais componentes dessa arquitetura são:

- SMA: Arquitetura de agentes criada para dar maior autonomia ao sistema e possibilitar a recomendação de OAs de forma automática. Esse SMA será detalhado na próxima seção.
- LMS: O LMS 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). Sem a utilização de um LMS se tornaria inviável a disponibilização de um curso a distância.
- Ontologia de Contexto Estático: É a ontologia utilizada para armazenamento e consulta das informações que são fornecidas manualmente pelos estudantes, como, por exemplo, a área de interesse do estudante, o horário preferido de estudo e a localização atual. Essas informações são fornecidas ao sistema através da interface mostrada na Figura 17.
- Ontologia de Contexto Dinâmico: É a ontologia utilizada para armazenamento e consulta das informações capturadas em tempo de execução através de um dos agentes da arquitetura multiagente. São informações como horário corrente e localização geográfica do estudante (caso o dispositivo móvel do estudante possua um GPS integrado), além de informações de preferências capturadas dinamicamente pelo SMA, as quais podem divergir das informações do contexto estático do estudante.

- Repositório de OAs SCORM: É o repositório utilizado para armazenamento dos OAs, os quais devem ter sido desenvolvidos seguindo o padrão SCORM. Através das informações fornecidas no SCORM e dos metadados do padrão LOM, que é parte integrante do SCORM, os agentes serão capazes de comparar as informações dos OAs com os perfis dos estudantes.
- Preferências similares: Diz respeito a informações de estudantes que possuem características similares e que escolheram o mesmo OA, fazendo com que a recomendação não leve em consideração apenas o conteúdo dos OAs em relação ao perfil do estudante.

7.4 SISTEMA MULTIAGENTE

A organização do SMA é do tipo comunidade de especialistas, pois cada um dos 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 18, o SMA é composto de quatro tipos de agentes: Agente Estudante (*Student Agent* - SAg), Agente Recomendador (*Recommender Agent* - RAg), Agente de Interface (*Interface Agent* - IAg)[7] e Agente DF (*Directory Facilitator*).

Os SAgs são responsáveis por monitorar as atividades dos estudantes, atualizar e recuperar, das ontologias de contexto estático e dinâmico, as preferências de aprendizagem que compõem os perfis dos estudantes e seus respectivos históricos de escolhas de OAs. Os SAgs também capturam as informações de localização geográfica e do horário corrente. Ou seja, o comportamento do SAg é voltado para monitoramento das informações relativas ao estudante. Em seguida, todas essas informações são cadastradas no agente DF.

O RAg tem o intuito de detectar OAs adequados ao contexto do estudante, levando em consideração (i) as informações consultadas no agente DF, (ii) as informações dos OAs disponíveis no repositório de OAs SCORM e (iii) as preferências similares de outros

[7] As siglas SAg, RAg e IAg são abreviações dos nomes dos agentes em inglês (*Student Agent*, *Recommender Agent* e *Interface Agent*, respectivamente) para facilitar a publicação em eventos e periódicos internacionais.

estudantes. Após identificar os OAs adequados, o RAg verifica, no agente DF, se há algum serviço disponibilizado por um agente de interface (IAg). Caso exista, o RAg envia para ele as informações de recomendação.

O IAg é responsável, principalmente, por verificar a adequação visual do OA recomendado pelo RAg às características do dispositivo móvel do estudante. Caso o OA não seja adequado ao dispositivo, o IAg rejeitará a recomendação. Além disso, o IAg tem a responsabilidade de informar ao instrutor o conteúdo que foi sugerido.

Essas informações 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 próxima subseção apresenta a modelagem do SMA proposto nesse trabalho, seguindo os passos estabelecidos pela metodologia MAS-CommonKADS+.

7.4.1 Modelagem do SMA

Seguindo os passos da metodologia MAS-CommonKADS+, essa seção 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. Além disso, percebeu-se que alguns modelos não eram necessários para a compreensão do SMA proposto e, conseqüentemente, não serão mostrados aqui. A escolha dos modelos a serem inseridos na modelagem depende do tamanho e da complexidade do SMA sendo modelado.

Baseando-se nas necessidades do *MobiLE*, o SMA deve realizar uma série de tarefas, as quais podem ser observadas no Modelo de Tarefas exibido na Figura 19.

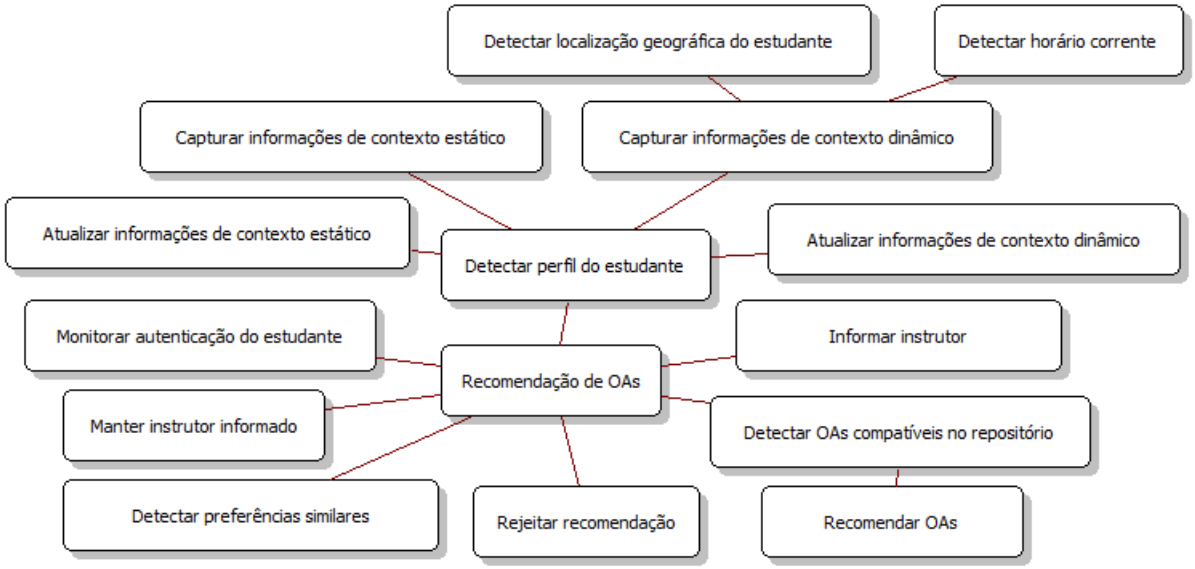

Figura 19 – Modelo de Tarefas do SMA proposto.

A tarefa principal desse SMA é a recomendação de OAs. Essa tarefa, por sua vez, foi decomposta em outras subtarefas, que têm como intuito final auxiliar no atendimento da tarefa principal. O próximo passo consiste em definir os papéis que cada agente irá realizar no sistema, visto 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 19, foi definido o Modelo de Papéis apresentado na Figura 20.

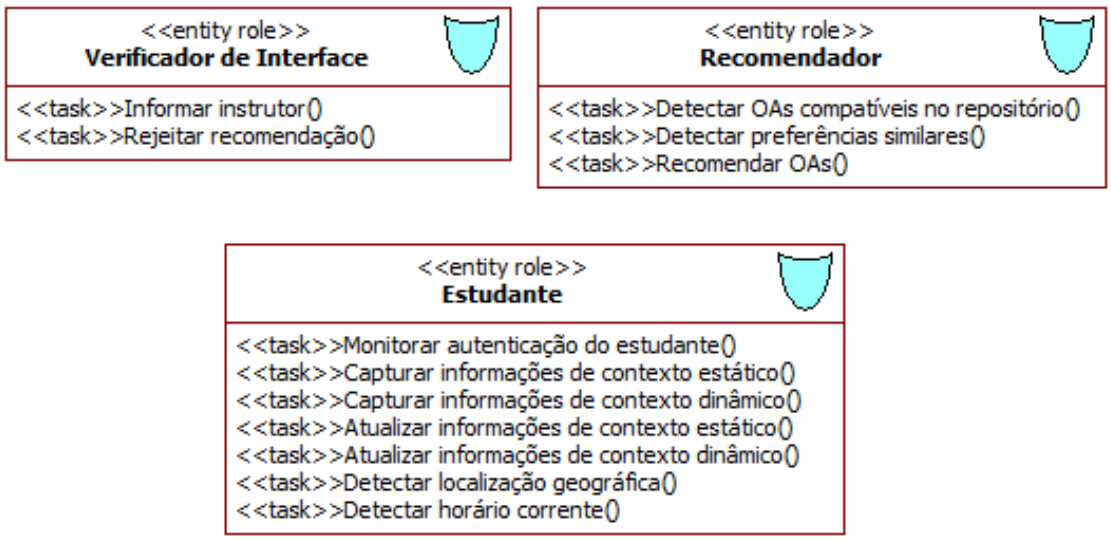

Figura 20 – Modelo de Papéis.

Na Figura 20, é possível perceber que foram definidos três tipos de papéis para atender às tarefas inerentes ao Modelo de Tarefas: estudante, recomendador e verificador de interface.

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 Estudante, o Agente Recomendador e o Agente de Interface. Desta forma, a partir de agora, os papéis não serão mais citados, mas sim os agentes que realizam esses papéis. Esses agentes serão melhor detalhados no Modelo de Agentes, em passos posteriores da metodologia.

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 21 mostra o Modelo de Organização correspondente aos papéis exibidos na Figura 20.

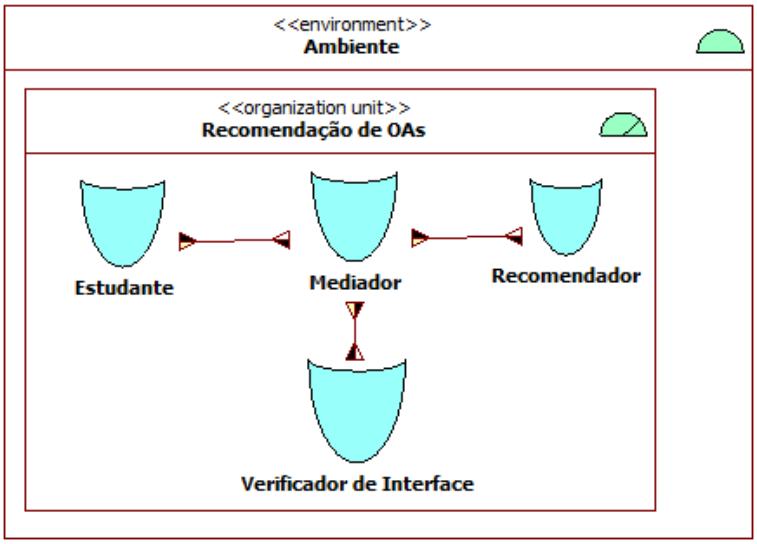

Figura 21 – Modelo de Organização.

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 que pesquisem por informações de outros agentes.

O papel do mediador será realizado pelo agente *Directory Facilitator* (DF), o qual é provido pela própria plataforma JADE, conforme exigência da especificação FIPA. É necessário apenas codificar a forma como os agentes criados no SMA irão se comunicar com o 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 21, na representação interna da unidade "Recomendação de OAs", os quatro 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.

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. A Figura 22 mostra o modelo de interação entre os agentes que compõem o SMA proposto.

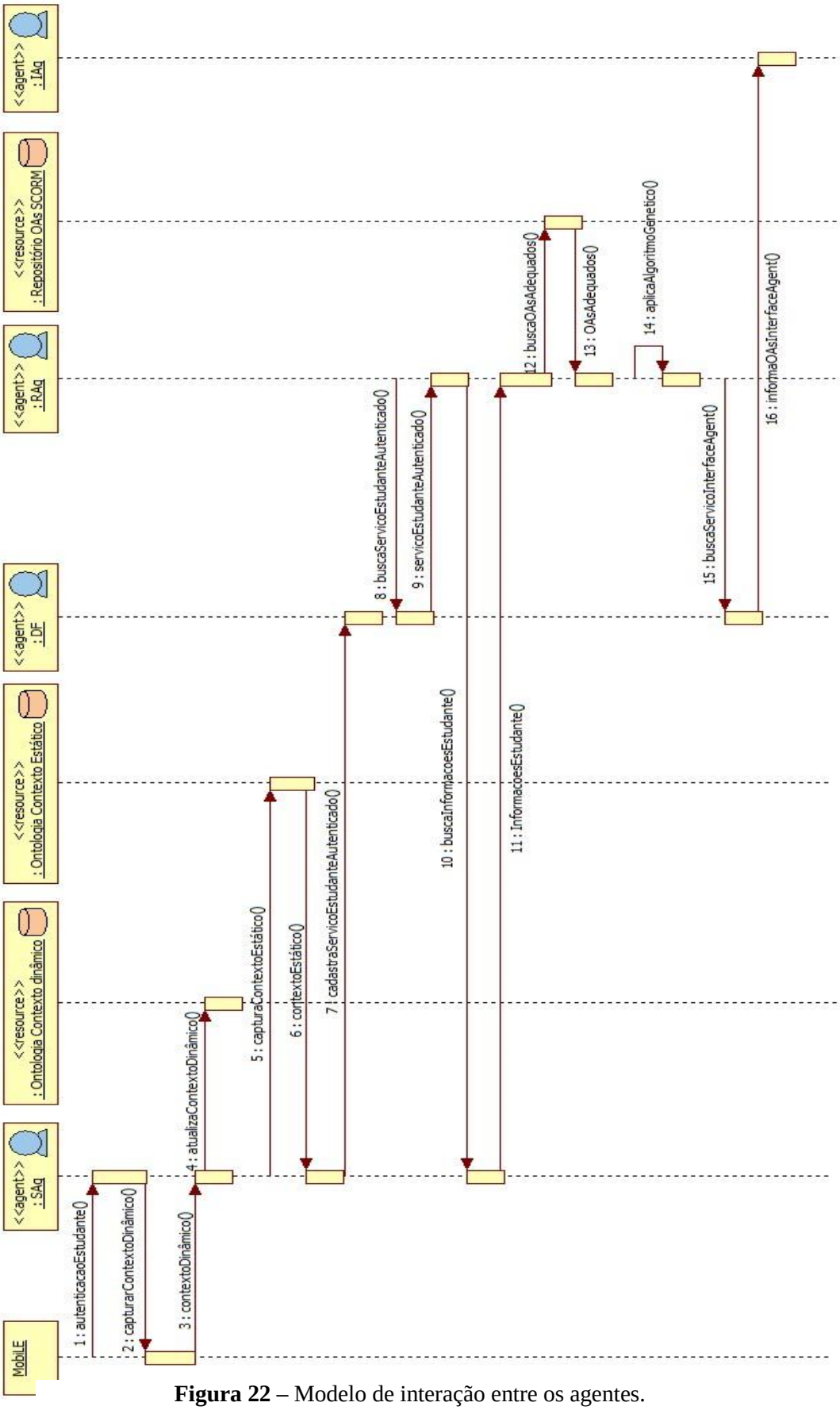

Figura 22 – Modelo de interação entre os agentes.

No modelo apresentado na Figura 22, é possível visualizar toda a interação que ocorre entre os agentes e os recursos que são consultados por cada um dos quatro tipos de agentes. O diagrama mostra desde o comportamento do SAg no momento que o estudante se autentica no *MobiLE* até a informação chegar no IAg, que irá realizar ou rejeitar a recomendação. 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 de contexto estático e dinâmico do estudante precisarão ser armazenadas. Uma forma possível de se armazenar essa informação 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 útil, 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, estão sendo utilizadas duas ontologias, sendo uma para representação do contexto estático e outra para representação do contexto dinâmico do estudante.

7.4.2 Modelo do Agente Estudante (SAg)

Os SAgs realizam, como dito anteriormente, o papel "Estudante". A Tabela 4 mostra o seu *template* textual.

Tabela 4 – *Template* textual do Agente Estudante

Agente: SAg
Objetivo Monitorar as informações do estudante, capturando o contexto dinâmico do mesmo, e cadastrar um novo serviço do tipo "estudante-autenticado" no DF para que o Agente Recomendador possa utilizar essas informações.
Parâmetros de Entrada Identificador do estudante no sistema.
Parâmetros de Saída Dados relativos ao contexto do estudante.
Condição de Ativação Quando o agente percebe que um estudante se autenticou no LMS.
Condição de Finalização

Quando o agente se comunica com o Agente Recomendador e envia para este as informações do contexto do estudante.
Informação Associada Possui um comportamento de execução única (tipo *One Shot Behaviour* (BELIFEMINE, CAIRE e GREENWOOD, 2007)) para cadastro das informações no DF, sendo criado, portanto, um SAg para cada estudante que se autentica no sistema. Cada SAg também possui um comportamento cíclico (tipo *Cyclic Behaviour* (BELIFEMINE, CAIRE e GREENWOOD, 2007)) para ficar monitorando por requisições do RAg.
Descrição Este agente possui o comportamento totalmente voltado para monitoramento das atividades dos estudantes, recuperando e atualizando, nas respectivas ontologias, as informações de preferências de aprendizagem que compõem os perfis dos estudantes, seus históricos de escolhas de OAs, a localização geográfica e o horário corrente.

A Figura 23 mostra o trecho do código-fonte do SAg responsável por recuperar informações do estudante.

```
30      private Student student = new Student();
31      private static final String ONTOLOGY_PATH = "file:"
32              +System.getProperty("user.dir")+"/ontologies/StaticContext.owl";
33
34      //[...]
43      student = new OntologyParser(ONTOLOGY_PATH).getStudent(id);
44      Location location = new ManagerLocation().reverseGeocoding(latitude, longitude);
45      student.setCurrentLocation(location);
46      student.setCurrentTime(new Date().getTime());
```

Figura 23 – Trecho do código-fonte do SAg para recuperar contexto do estudante.

Inicialmente, o SAg instancia as informações da ontologia de contexto estático do estudante (linha 43), baseando-se em uma constante que define onde está localizada a ontologia (linhas 31 e 32).

Em seguida, o SAg captura as informações de latitude e longitude do estudante e realiza a geocodificação reversa[8] dessas informações (linha 44). Essa informação, na realidade, já foi fornecida ao sistema através da interface adaptada do Moodle, conforme mostrado na Figura 17. Porém, o estudante pode estar em outro local. Pensando nisso, o SAg captura essa informação através de uma extensão ao framework MLE criada neste trabalho. Essa extensão consiste em uma classe adicionada ao MLE que utiliza uma API (*Application Programming Interface*), disponibilizada pela Google (GOOGLE, 2010), que permite recuperar informações do GPS do dispositivo móvel do estudante. Caso o dispositivo móvel do estudante não possua um GPS integrado, a informação utilizada será aquela fornecida através da interface do Moodle. A extensão criada neste trabalho pode ser vista no Apêndice A.

[8] Conversão de latitude e longitude em endereço (cidade, estado e país).

Por fim, o SAg captura a informação do horário atual do estudante (linha 46) e trata essa informação, transformando-a para manhã, tarde ou noite. Essa informação, juntamente, com a informação de localização geográfica influenciará a recomendação do OA adequado para o estudante.

A Figura 24, por sua vez, mostra a parte do código-fonte do SAg responsável por cadastrar um serviço do tipo "estudante-autenticado" no DF.

```
//[...]
//Registra o serviço de "estudante-autenticado" no DF
DFAgentDescription myDescriptionTemplate = new DFAgentDescription();
myDescriptionTemplate.setName(getAID());
ServiceDescription myServiceDescription = new ServiceDescription();
myServiceDescription.setType("estudante-autenticado");
myServiceDescription.setName("autenticacao-estudante-moodle");
myDescriptionTemplate.addServices(myServiceDescription);
try {
    DFService.register(this, myDescriptionTemplate);
} catch (FIPAException fe) {
    fe.printStackTrace();
}
```

Figura 24 – Parte do código-fonte do SAg para cadastrar serviço no DF.

Nesse código, o SAg cria, primeiramente, um template (*myDescriptionTemplate*), no qual é possível registrar mais de um serviço. Em seguida, adiciona um serviço (*myServiceDescription*) a esse template. Por fim, registra esse template no DF.

7.4.3 Modelo do Agente de Interface (IAg)

O IAg realiza o papel "Verificador de Interface". A Tabela 5 mostra o seu *template* textual.

Tabela 5 – *Template* textual do Agente de Interface.

Agente: IAg
Objetivo Disponibilizar um serviço do tipo "interface-agent" no DF de modo que o RAg possa encontrar esse serviço para se comunicar com ele. Uma vez em posse das informações enviadas pelo RAg, o IAg realiza a sua recomendação ou rejeição. Em seguida, deve informar sua decisão ao instrutor.
Parâmetros de Entrada Informações de recomendação de OA enviadas pelo RAg.
Parâmetros de Saída Uma mensagem de texto contendo o título do OA a ser recomendado para o estudante e

uma mensagem para o instrutor.
Condição de Ativação Quando o agente recebe do RAg uma mensagem com identificador de conversação do tipo “informacao-objeto-aprendizagem”.
Condição de Finalização Quando o agente envia a mensagem de texto para o estudante ou quando decide rejeitar a recomendação.
Informação Associada Possui um comportamento que disponibiliza um serviço do tipo “interface-agent” no DF de modo que o RAg possa encontrar esse serviço e se comunicar com o IAg. Também possui um comportamento cíclico (tipo *Cyclic Behaviour*) para ficar monitorando informações de OAs enviadas pelo RAg. Uma vez em posse dessas informações, ele realiza a recomendação ou rejeição do OA.
Descrição Este agente possui o comportamento voltado para a verificação da adequação de OAs ao dispositivo móvel do estudante, recomendando o conteúdo ou rejeitando o mesmo.

A Figura 25 mostra um trecho de código do IAg no qual ele cadastra um serviço do tipo “interface-agent” no DF.

```
37      DFAgentDescription myDescriptionTemplate = new DFAgentDescription();
38      myDescriptionTemplate.setName(getAID());
39      ServiceDescription myServiceDescription = new ServiceDescription();
40      myServiceDescription.setType("interface-agent");
41      myServiceDescription.setName("recomendacao-objeto");
42      myDescriptionTemplate.addServices(myServiceDescription);
43      try {
44          DFService.register(this, myDescriptionTemplate);
45      } catch (FIPAException fe) {
            fe.printStackTrace();
47      }
48      //Comportamento para receber recomendação(ões) do RAg
49      addBehaviour(new GetRecommenderAgentLO());
```

Figura 25 - Parte do código-fonte do IAg para cadastrar serviço no DF.

O código é similar àquele mostrado na Figura 24. A diferença é que, no trecho de código do IAg, ao final do cadastro do serviço, ele adiciona um comportamento *GetRecommenderAgentLO*. Esse é um comportamento cíclico responsável por receber do RAg as informações de possíveis OAs a serem recomendados. Nesse comportamento, o IAg procura continuamente em sua pilha por uma mensagem que utilize o identificador de conversação “informacao-objeto-aprendizagem”.

7.4.4 Modelo do Agente Recomendador (RAg)

A recomendação de OAs efetuada pelo RAg ocorre levando-se em consideração o resultado da execução do algoritmo genético (AG) discutido no Capítulo 6. Essa divisão foi feita para facilitar a visualização e o entendimento da modelagem, visto que inserir todas as informações do problema de recomendação de OAs em apenas uma seção dificultaria o seu entendimento.

O RAg realiza o papel "Recomendador". A Tabela 6 mostra o seu *template* textual.

Tabela 6 – *Template* textual do Agente Recomendador

Agente: RAg
Objetivo Monitorar os serviços do tipo "estudante-autenticado" cadastrados pelo SAg no DF e, em seguida, enviar informações de recomendação ao IAg.
Parâmetros de Entrada Informações cadastradas pelo RAg no DF, informações dos OAs disponíveis no repositório de OAs SCORM e informações de preferências similares de outros estudantes.
Parâmetros de Saída Possíveis OAs adequados ao contexto do estudante.
Condição de Ativação Quando o agente percebe no DF um serviço cadastrado do tipo "estudante-autenticado".
Condição de Finalização Quando o agente se comunica com o Agente de Interface e envia para este as informações do OA a ser recomendado e para qual estudante, através do identificador de conversação "informação-objeto-aprendizagem".
Informação Associada Possui um comportamento executado a cada determinado período de tempo (tipo *Ticker Behaviour* (BELIFEMINE, CAIRE e GREENWOOD, 2007)) para monitoramento dos serviços do tipo "estudante-autenticado" no DF. Desta forma, apenas um agente RAg é criado no sistema. Também possui um comportamento do tipo execução única (*One Shot* Behaviour) para indicar ao Agente de Interface (IAg) o possível OA a ser recomendado. Este agente toma as decisões de acordo com o resultado da execução do AG apresentado no Capítulo 6.
Descrição Este agente possui o comportamento voltado para a recomendação de OAs. É responsável por detectar e realizar a recomendação de OAs.

A Figura 26 mostra o comportamento do RAg responsável por verificar se existe um novo serviço cadastrado pelo SAg.

```
48      addBehaviour(new TickerBehaviour(this, 10000) {
            @Override
50          protected void onTick() {
51              //Atualiza a lista de serviços dos AgentesEstudantes cadastrados no DF
52              DFAgentDescription studentTemplate = new DFAgentDescription();
53              ServiceDescription studentServiceDescription = new ServiceDescription();
54              studentServiceDescription.setType("estudante-autenticado");
55              studentTemplate.addServices(studentServiceDescription);
56              try {
57                  DFAgentDescription[] result = DFService.search(myAgent, studentTemplate);
58                  //Se tiver algum serviço cadastrado no DF
59                  if (result.length > 0) {
60                      System.out.println("Foram encontrados " + result.length
61                              + " serviços do tipo " + studentServiceDescription.getType());
62                      //Cria o array de serviços dos agentes estudantes
63                      studentAgents = new AID[result.length];
64                      for (int i = 0; i < result.length; i++) {
65                          studentAgents[i] = result[i].getName();
66                      }
67                      //Adiciona o comportamento para solicitar os dados do estudante
68                      addBehaviour(new LearningObjectRecommendation());
69                  }
70              } catch (FIPAException fe) {
                    fe.printStackTrace();
72              }
73          }
74      });//Fim do comportamento TickerBehaviour
```

Figura 26 – Parte do código-fonte do RAg para verificação de serviços cadastrados pelo SAg

Como pode ser visto na Figura 26, o RAg cria, inicialmente, um *template* que serve para informar qual tipo de serviço ele está buscando, no caso o serviço do tipo "estudante-autenticado" (linhas 52-55). A cada 10 segundos, ele faz uma busca no DF para verificar se novos serviços do tipo citado foram adicionados e, caso existam, as descrições desses serviços são adicionadas a um vetor (linha 57). Em seguida, ele cria um vetor contendo os identificadores (AIDs) de todos os SAgs responsáveis por esses serviços (linha 63) e adiciona um novo comportamento (linha 68) para iniciar a comunicação com o respectivo SAg.

A Figura 27 mostra o comportamento do RAg responsável por verificar se existe um serviço do tipo "interface-agent" cadastrado no DF.

```
146     DFAgentDescription recommenderTemplate = new DFAgentDescription();
147     ServiceDescription recommenderServiceDescription = new ServiceDescription();
148     recommenderServiceDescription.setType("interface-agent");
149     recommenderTemplate.addServices(recommenderServiceDescription);
150     try {
151         DFAgentDescription[] result = DFService.search(myAgent, recommenderTemplate);
152         //Se tiver algum agente oferecendo um serviço típico de InterfaceAgente no DF
153         if (result.length > 0) {
154             System.out.println("Foram encontrados " + result.length +
155                     " serviços do tipo " + recommenderServiceDescription.getType());
156             //Cria o array de serviços do(s) agente(s) de interface
157             interfaceAgents = new AID[result.length];
158             for (int i = 0; i < result.length; i++) {
159                 interfaceAgents[i] = result[i].getName();
160             }
161             //Adiciona o comportamento para enviar os dados do Objeto de Aprendizagem
162             addBehaviour(new InterfaceAgentInform());
163             step++;
164         }
165     } catch (FIPAException fe) {
            fe.printStackTrace();
167     }
```

Figura 27 - Parte do código-fonte do RAg para verificação de serviço cadastrado pelo IAg.

É possível perceber que esse código é bastante parecido com aquele mostrado na Figura 26, visto que ambos possuem o objetivo de procurar por serviços cadastrados no DF. Porém há algumas diferenças. A primeira delas consiste nos tipos de serviços pesquisados e nos tipos de comportamentos, visto que aquele mostrado na Figura 26 é do tipo *Ticker Behaviour*, enquanto que o apresentado na Figura 27 é de execução única. Outra diferença é que, ao final do comportamento, é adicionado o comportamento *InterfaceAgentInform* (linha 162), que serve para informar o IAg das informações do OA a ser recomendado. Esse comportamento pode ser visualizado na Figura 28.

```
186     private class InterfaceAgentInform extends OneShotBehaviour {
187
            @Override
189         public void action() {
190             ACLMessage LORecommendantion = new ACLMessage(ACLMessage.INFORM);
191             for (int i = 0; i < interfaceAgents.length; i++) {
192                 LORecommendantion.addReceiver(interfaceAgents[i]);
193             }
194             LORecommendantion.setConversationId("informacao-objeto-aprendizagem");
195             LORecommendantion.setReplyWith("inform" + System.currentTimeMillis());
196             try {
197                 LORecommendantion.setContentObject(learningObject);
198             } catch (IOException ex) {
199                 Logger.getLogger(RecommenderAgent.class.getName()).log(Level.SEVERE, null, ex);
200             }
201             myAgent.send(LORecommendantion);
202         }
203     }
```

Figura 28 – Código-fonte do RAg para envio das informações de OA ao IAg.

A Figura 28 mostra o comportamento de execução única do RAg utilizado para informar ao IAg os possíveis OAs a serem recomendados. Para que o IAg possa recuperar e

identificar com precisão a mensagem enviada pelo RAg, é utilizado um identificador de conversação do tipo “informação-objeto-aprendizagem” (linha 194).

7.4.5 Modelo do Agente DF

O Agente DF realiza o papel “Mediador”. A Tabela 7 mostra o seu *template* textual.

Tabela 7 – *Template* textual do Agente DF

Agente: DF
Objetivo 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.

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. Um desses componentes é o agente DF.

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. No presente trabalho, este protocolo foi utilizado para comunicação entre agentes que se encontravam na mesma plataforma, visto que os três agentes criados e o DF se encontravam no mesmo servidor.

A Figura 29 mostra, através da interface gráfica do JADE, como ocorre a comunicação entre os agentes descrita através dos *templates* textuais dos agentes apresentados anteriormente.

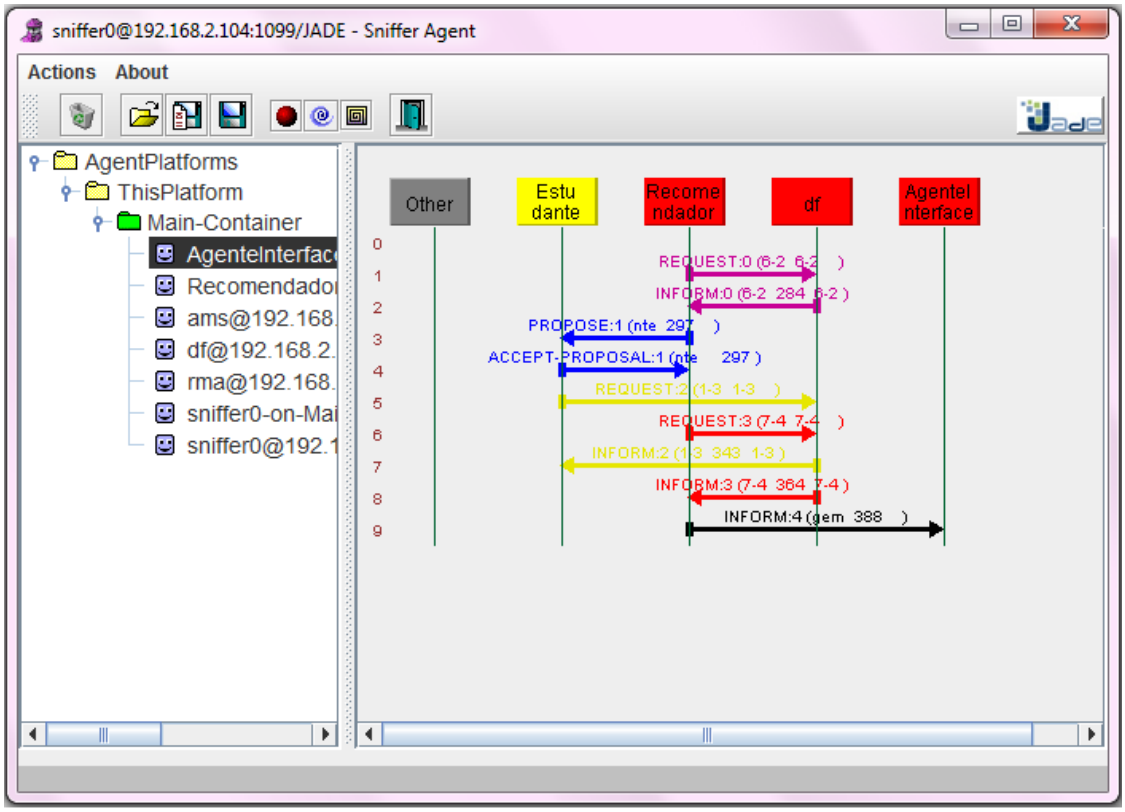

Figura 29 – Comunicação entre os agentes através do JADE.

Como pode ser visto na Figura 29, inicialmente, o RAg solicita (REQUEST:0) por informações de serviço do tipo "estudante-autenticado" ao DF. Em seguida, o DF responde (INFORM:0) com as informações de serviços disponíveis e com os AIDs dos respectivos SAgs responsáveis pelos serviços. No próximo passo, o RAg solicita aos SAgs (PROPOSE:1) pelas informações dos estudantes. Uma vez que os SAgs respondam ao RAg (ACCEPT-PROPOSAL:1), o SAg solicita ao DF (REQUEST:2) para retirar o registro do seu serviço, de modo que este fique indisponível. Isto acontece porque é instanciado um SAg para cada estudante. No passo seguinte, o RAg procura por serviços do tipo "interface-agent" no DF (REQUEST:3). Por fim, quando recebe a resposta do DF (INFORM:3) com o AID do agente responsável por este serviço, o RAg envia os dados dos possíveis OAs a serem recomendados

para o IAg (INFORM:4), configurando, para tanto, o identificador de conversação “informacao-objeto-aprendizagem”. A Figura 29 retrata a interação que ocorre entre os agentes ilustrada no Modelo de Interação da Figura 22.

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.

8 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

No âmbito da educação, muito se tem pesquisado a respeito de novas formas de prover aprendizagem, de forma a estimular cada vez mais os estudantes. Dessa forma, novas metodologias e abordagens educacionais têm sido criadas, com o intuito de alcançar melhores resultados pedagógicos em um público cada vez mais dinâmico e diversificado. Ao instituir essas mudanças, muitos paradigmas foram quebrados, dentre os quais o de que só é possível ocorrer um ensino eficaz em sala de aula.

Com isso, surgiu a EaD, que tem crescido e apresentado bons resultados, embora ainda apresente grandes desafios. Um destes desafios é a adequação do conteúdo ao estudante de acordo com as características cognitivas deste. No ensino presencial, esta já é uma tarefa difícil de ser realizada, visto que é necessário que o professor possua certo conhecimento das necessidades de cada um dos alunos. Tal tarefa torna-se ainda mais complexa quando o ensino é realizado a distância, visto que os ambientes virtuais de aprendizagem enfocam, geralmente, aspectos relacionados à sua funcionalidade, esquecendo assim a função pedagógica do ambiente.

Outro fator que deve ser levado em consideração é que, nem sempre, os recursos tecnológicos utilizados para o ensino permitem realizar o ensino de forma simples e completa, sendo necessárias algumas adaptações, como é o caso da aprendizagem móvel, a qual pode apresentar limitações de acordo com as características do dispositivo móvel do estudante.

Pensando nisso, o presente trabalho apresentou uma abordagem para recomendação de conteúdos educacionais (OAs) que sejam adequados às características do contexto do estudante. Para tanto, a abordagem leva em consideração uma série de características, conforme mostrado ao longo deste trabalho. O intuito desta abordagem é aperfeiçoar o processo de ensino-aprendizagem de forma transparente aos estudantes, provocando assim um maior interesse no estudo por parte destes.

Outra contribuição deste trabalho que merece destaque é o fato do ambiente ter sido desenvolvido utilizando-se de um SMA que pode ser utilizado em outros contextos, ou seja, a arquitetura proposta pode ser utilizada em qualquer área de aplicação.

Por fim, a extensão adicionada ao MLE pode ser utilizada por outras pessoas que utilizem esse framework para desenvolvimento de suas aplicações, podendo assim utilizar a informação de localização geográfica do estudante como acharem necessário.

Apesar do trabalho realizado, há algumas melhorias que podem ser realizadas. Assim, como trabalho futuro, pretende-se:

- Testar o ambiente desenvolvido em um contexto real, utilizando os principais repositórios de conteúdos educacionais (OAs). Isso possibilitará mensurar, estatisticamente, o quanto este mecanismo de recomendação contribui para o processo de ensino-aprendizagem.
- Disponibilizar a infraestrutura do *MobiLE* como um componente do Moodle sob uma licença GPL, de forma a facilitar a instalação e utilização do ambiente por instituições que utilizem esse LMS e desejem disponibilizá-lo através de dispositivos móveis.
- Testar o ambiente desenvolvido em á*reas e*specíficas (área médica, petrolífera, etc.) para verificar a adequabilidade da recomendação de OAs em contextos específicos.
- Aperfeiçoar a visualização da recomendação, visto que no momento a recomendação é feita de forma textual, informando o nome do OA.
- Levar em consideração outras características do dispositivo móvel do estudante antes de decidir se um OA é adequado ao contexto do estudante, aperfeiçoando, conseqüentemente, o mecanismo de recomendação.
- Modificar o Agente de Interface (IAg), permitindo que ele adéqüe a apresentação do conteúdo do OA ao dispositivo móvel do estudante, quando estes não forem compatíveis entre si.
- Identificar e classificar os estilos cognitivos dos estudantes de acordo com uma teoria já consolidada na literatura, como, por exemplo, a Teoria da Carga Cognitiva (SANTOS, REATEGUI e TAROUCO, 2009), para que seja possível aperfeiçoar ainda mais o processo de ensino-aprendizagem do estudante. O presente trabalho não se baseou em uma teoria de aprendizagem específica.

Além dessas contribuições, que serão desenvolvidas posteriormente, há ainda outras possibilidades de contribuição, dentre as quais pode-se destacar: (i) a consideração, na recomendação baseada no conteúdo, das palavras que sejam sinônimas daquelas que estão na ontologia de contexto estático, considerando a relação semântica entre as palavras no momento da recomendação; e (ii) a criação de uma adaptação do *MobiLE* para que seja possível utilizá-lo com outros LMS.

REFERÊNCIAS

ADL. **ADL Guidelines for Creating Reusable Content with SCORM 2004**. Advanced Distributed Learning. [S.l.]. 2008. Disponível em: http://www.adlnet.gov/wp-content/uploads/2011/07/ADL_Guidelines_Creating_Reusable_Content.pdf.

AL-SAKRAN, H. O.; MUHAYA, B. F.; SERGUIEVSKAIA, I. **IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering (SIBIRCON)**. Multi agent-based m-learning system architecture. Riyadh, Arábia Saudita: IEEE Computer Society. 2010. p. 870 -875.

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

AYALA, G.; CASTILO, S. **Towards Computational Models for Mobile Learning Objects**. V IEEE International Conference on Wireless, Mobile, and Ubiquitous Technology in Education (WMUTE). Los Alamitos, CA, USA: IEEE Computer Society. 2008. p. 153-157.

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

BITTENCOURT, I. et al. **Ontologia para Construção de Ambientes Interativos de Aprendizagem**. Simpósio Brasileiro de Informática na Educação (SBIE). Brasília, DF: SBIE. 2006. p. 567-576.

BRASIL. **Decreto nº 5622, de 19 de dezembro de 2005. Regulamenta o art. 80, da Lei n° 9.394, de 20 de dezembro de 1996, que estabelece as diretrizes e bases da educação nacional**. Brasília, DF: Diário Oficial da União, 2005.

BRASIL, M. M. A. **Planejamento de Releases de Software Através da Aplicação de Técnicas de Busca Multiobjetivas**. Dissertação de Mestrado. Universidade Estadual do Ceará (UECE). Fortaleza, CE. 2011.

CASTILHO, S.; AYALA, G. **ARMOLEO - An Architecture for Mobile Learning Objects**. 18th International Conference on Electronics, Communications and Computers (CONIELECOMP). Puebla: IEEE Computer Society. 2008. p. 53-58.

CASTRO, J.; ALENCAR, F.; SILVA, C. T. L. L. Engenharia de Software Orientada a Agentes. In: BREITMAN, K.; ANIDO, R. **Atualizações em Informática**. Rio de Janeiro: PUC-Rio, 2006. Cap. V, p. 245-282.

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.

CHING-BANG, Y. **Personalized guidance and ubiquitous learning in intelligent library with multi-agent**. The 2nd International Conference on Computer and Automation Engineering (ICCAE). Taipei, Taiwan: IEEE Computer Society. 2010. p. 578 -582.

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.

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.

DIAS, C. C. L. et al. **Padrões abertos:** aplicabilidade em Objetos de Aprendizagem (OAs). Simpósio Brasileiro de Informática na Educação (SBIE). Florianópolis, SC: SBIE. 2009.

DICKINSON, I. Jena Ontology API. **Site do Framework Jena**, 2009. Disponivel em: <http://jena.sourceforge.net/ontology/index.html>. Acesso em: 24 out. 2011.

DUTRA, R. L. D. S.; TAROUCO, L. M. R. Objetos de aprendizagem: uma comparação entre SCORM e IMS Learning Design. **RENOTE : revista novas tecnologias na educação**, Porto Alegre, RS, v. IV, p. 7, 2006.

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

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 Publhising, 2005. Cap. VIII, p. 203-234.

GIORGINI, P. et al. 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.

GOOGLE. Google Maps API Web Service - Serviços da web da Google Maps API - Google Code. **Google Code**, 2010. Disponivel em: <http://code.google.com/intl/pt-BR/apis/maps/documentation/geocoding/>. Acesso em: 05 dez. 2011.

GPL. The GNU General Public License v3.0 - GNU Project - Free Software Foundation (FSF). **GNU General Public License**, 2007. Disponivel em: <http://www.gnu.org/copyleft/gpl.html>. Acesso em: 15 dez. 2011.

HAN, Q.; GAO, F.; WANG, H. **Ontology-based learning object recommendation for cognitive considerations**. 8th World Congress on Intelligent Control and Automation (WCICA). Jinan, China: IEEE Computer Society. 2010. p. 2746-2750.

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

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.

JME. Java ME Landing Page. **Site da Oracle**, 2011. Disponivel em: <http://www.oracle.com/technetwork/java/javame/index.html>. Acesso em: 02 dez. 2011.

LI, S.; YANG, Z.; LIU, Q. **Research of Metadata Based Digital Educational Resource Sharing**. International Conference on Computer Science and Software Engineering. Los Alamitos, CA, USA: IEEE Computer Society. 2008. p. 828-831.

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

LOUREIRO, A. A. F. et al. **Computação Ubíqua Ciente de Contexto:** Desafios e Tendências. XXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SRBC´09). Recife, PE: SBIE. 2009.

LTSC. **Learning Technology Standards Committee. Draft Standard for Learning Object Metadata. IEE Standard 1484.12.1**. Institute of Electrical and Electronics Engineers. Nova Yorque. 2002. Disponível em:< http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf>.

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

MANDULA, K.; MEDA, S. R.; KAMBHAM, R. **Implementation of Ubiquitous Learning System Using Sensor Technologies**. IEEE International Conference on Technology for Education. Hyderabad, India: IEEE Computer Society. 2011. p. 142-148.

MENDES, R. M.; SOUZA, V. I.; CAREGNATO, S. E. **A propriedade intelectual na elaboração de objetos de aprendizagem**. V Encontro Nacional de Ciência da Informação (CINFORM). Salvador, BA: Universidade Federal da Bahia. 2004.

MIN., W. X.; WEI, C.; LEI, C. **Research of Ontology-based Adaptive Learning System**. International Symposium on Computational Intelligence and Design (ISCID). Wuhan, China: IEEE Computer Society. 2008. p. 366 -370.

MLE. MLE - Mobile Learning Engine. **Página Sourceforge do MLE**, 2011. Disponivel em: <http://mle.sourceforge.net/mle/index.php>. Acesso em: 02 dez. 2011.

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

MOORE, P. et al. **Intelligent Context for Personalised M-Learning**. International Conference on Complex, Intelligent and Software Intensive Systems (CISIS). Birmingham: Birmingham City University. 2009. p. 247-254.

MOORE, P.; HU, B. **A context framework with ontology for pesonalized and cooperative mobile learning**. 10th international conference on Computer Supported Cooperative Work in Design III (CSCWD). Berlin, Heidelberg: Springer-Verlag. 2007. p. 727-738.

MOORE, P.; HU, B.; WAN, J. Smart-Context: A Context Ontology for Pervasive Mobile Computing. **The Computer Journal**, 53, n. 2, 2008. 191-207.

MORAIS II, M. J. D. O. **MAS-CommonKADS+: Uma Extensão à Metodologia Mas-CommonKADS para Suporte ao Processo Detalhado de Sitemas 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. Disponivel em: <http://www.mysql.com/>. Acesso em: 15 dez. 2011.

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. Disponivel em: <http://www.php.net/>. Acesso em: 15 dez. 2011.

PONTES, A. A. **Uma Arquitetura de Agentes para Suporte à Colaboração na Aprendizagem Baseada em Problemas em Ambientes Virtuais de Aprendizagem**. Dissertação de Mestrado. Universidade Federal Rural do Semiarido (UFERSA). Mossoró, RN. 2010.

PRESSMAN, R. S. **Engenharia de Software:** uma abordagem profissional. Tradução de Ariovaldo Griesi e Mario Moro Fecchio. 7ª. ed. Porto Alegre: Mac Graw Hill, 2011.

RANI, S. J.; ASHOK, M. S.; PALANIVEL, K. **Adaptive content for personalized E-learning using web service and semantic web**. International Conference on Intelligent Agent Multi-Agent Systems (IAMA). Pondicherry, India: IEEE Computer Society. 2009. p. 1-4.

RODOLPHO, E. R. **Convergência Digital de Objetos de Aprendizagem SCORM**. Dissertação de Mestrado. Universidade Estadual Paulista Júlio Mesquita Filho. São José do Rio Preto, SP. 2009.

ROUGEMAILLE, S. et al. 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, L. M. A.; REATEGUI, E.; TAROUCO, L. **A Inserção de um Agente Conversacional Animado em um Ambiente Virtual de Aprendizagem a partir da Teoria da Carga Cognitiva**. Tese de Doutorado. Universidade Federal do Rio Grande do Sul. Porto Alegre, RS. 2009.

SCHREIBER, G. et al. **Knowledge Engineering and Management - The CommonKADS Methodology**. Cambridge, MA: MIT Press, 2000.

SCORM. **SCORM 2004 4th Edition**. Advanced Distributed Learning. [S.l.]. 2009. Disponível em: http://www.adlnet.gov/capabilities/scorm/scorm-2004-4th.

SEED/MEC. **Referencias de Qualidade para Educação Superior a Distância**. Secretaria de Educação a Distância/Ministério da Educação. Brasília, DF. 2007. Disponível em: http://portal.mec.gov.br/seed/arquivos/pdf/legislacao/refead1.pdf.

STARUML. StarUML - The Open Source UML/MDA Platform. **Página Sourcefourge do STARUML**, 2011. Disponivel em: <http://staruml.sourceforge.net/en/>. Acesso em: 04 dez. 2011.

WILEY, D. A. Connecting learning objects to instructional design theory: a definition, a metaphor, and a taxonomy. In: _____ **The Instructional Use of Learning Objects**. Bloomington, EUA: The Agency for Instructional Technology, 2002. p. 3-23.

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

WURFL. WURFL - Mobile Device Detection By ScientiaMobile. **Site da WURFL**, 2011. Disponivel em: <http://wurfl.sourceforge.net/>. Acesso em: 30 Nov. 2011.

YAU, J. Y.-K.; JOY, M. **A Context-Aware Personalized M-learning Application Based on M-learning Preferences**. 6th IEEE International Conference on Wireless, Mobile and Ubiquitous Technologies in Education (WMUTE). Reino Unido: IEEE Computer Society. 2010. p. 11-18.

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 computacional organizations: The gaia methodology. In: HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Hershey, PA: IDEA Group Publishing, 2005. Cap. VI, p. 136-171.

ZHAO, R.; WANG, J. Visualizing the research on pervasive and ubiquitous computing. **Scientometrics**, Budapeste, Hungria, v. 86, n. 3, p. 593-612, 2011.

ZINI, É. D. O. C. **Algoritmo Genético Especializado na Resolução de Problemas com Variáveis Contínuas e Altamente Restritos**. Dissertação de Mestrado. Universidade Estadual Paulista (UNESP). Ilha Solteira, SP. 2009.

APÊNDICE A – Extensão ao MLE

A classe *LocationManager*, cujo código é mostrado abaixo, foi criada como uma extensão ao MLE, com o objetivo de capturar a localização geográfica do estudante a partir do GPS do dispositivo móvel do mesmo.

```
package model;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import model.entity.Location;
import org.json.JSONArray;
import org.json.JSONObject;

/**
 *
 * @author luizclaudio
 */
public class LocationManager {
    public Location reverseGeocoding(String latitude, String longitude){
        Location location = null;
        try{
            location = new Location();
            JSONObject json = new JSONObject(send("http://maps.google.com/maps/api/geocode/json?latlng="+latitude+","+longitude+"&sensor=true&language=pt-BR"));
            JSONArray array1 = json.getJSONArray("results");
            JSONArray array2 = array1.getJSONObject(0).getJSONArray("address_components");
            JSONObject city = (JSONObject) array2.get(1);
            JSONObject state = (JSONObject) array2.get(2);
            JSONObject contry = (JSONObject) array2.get(3);

            location.setCity(city.getString("long_name"));
            location.setState(state.getString("long_name"));
            location.setCountry(contry.getString("long_name"));

            System.out.println(location.toString());

        }catch(Exception e){
            System.out.println("Exception reverseGeocoding: "+e.getMessage());
        }finally{
            return location;
        }
```

```
        }

        private String send(String url){
           String response=null;
           try {
              //String url = "http://api.photobucket.com/login/request";
              URL endereco = new URL(url);
              HttpURLConnection connection = (HttpURLConnection)
endereco.openConnection();
              connection.setRequestMethod("GET");
              connection.setDoInput(true);
              connection.setDoOutput(false);
              connection.connect();

              // abre a conexão pra input
              BufferedReader br = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
              // lê até o final
              StringBuffer newData = new StringBuffer(10000);
              String s = "";
              while (null != ((s = br.readLine()))) {
               newData.append(s.trim());
              }
              br.close();
              response = new String(newData);

           } catch (Exception e) {
              System.out.println("Exception Class Server -> send(): "+e.getMessage());
           }  finally{
              return response;
           }
        }
}
```