UNIVERSIDADE FEDERAL DE ALAGOAS

DISSERTAÇÃO DE MESTRADO

UM SISTEMA TUTOR MÓVEL NO CONTEXTO DE UM FRAMEWORK DE SISTEMAS DE ENSINO ON-LINE

Luiz Cláudio Ferreira da Silva Júnior

Orientador:
Arturo Hernández-Domínguez

Maceió/AL

2009

LUIZ CLÁUDIO FERREIRA DA SILVA JUNIOR

UM SISTEMA TUTOR MÓVEL NO CONTEXTO DE UM FRAMEWORK DE SISTEMAS DE ENSINO ON-LINE

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Curso de Mestrado em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas.

Orientador: Arturo Hernández-Domínguez

Maceió/AL

2009

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária Responsável: Helena Cristina Pimentel do Vale

S586u Silva Júnior, Luiz Cláudio Ferreira da.
Um sistema tutor móvel no contexto de um Framework de sistema de ensino on-line / Luiz Cláudio Ferreira da Silva Júnior, 2009.
138 f. : il.

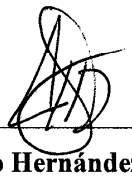
Orientador: Arturo Hernández-Domínguez.
Dissertação (mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2006.

Bibliografia: f. 118-127.
Apêndices: 128-134.
Anexos: f. 135-138.

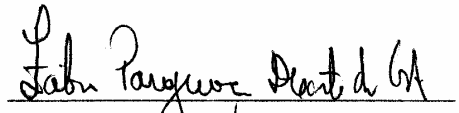
1. Framework (Arquitetura de Software). 2. Sistema de Agentes móveis. 3. Sistemas tutores inteligentes. 4. Ensino a distância. I. Título.

CDU: 004.78:37.018.43

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Modelagem Computacional de Conhecimento pelo Programa Multidisciplinar de Pós-Graduação em Modelagem Computacional de Conhecimento, da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina:



Prof. Dr. Arturo Hernández-Domínguez
UFAL – Instituto de Computação
Orientador



Prof. Dr. Fábio Paraguaçu Duarte da Costa
UFAL – Instituto de Computação
Examinador

Prof. Dr. Edilson Ferneda
UCB – Pró-Reitoria de Pós-Graduação e Pesquisa
Examinador

Maceió, setembro de 2009.

Dedicatória

Aos meus pais, Luiz Cláudio e Elisabethe, pela infinita dedicação e confiança.
À minha grande amiga, Márcia, pelo constante incentivo e apoio em todos os momentos.
À memória de meu avô, Liberalino, que, certamente, estaria compartilhando intensamente da felicidade por esta conquista.

Agradecimentos

Meus sinceros agradecimentos:

Aos meus pais, Luiz Cláudio e Elisabethe, pessoas especiais que sempre me guiaram e me apoiaram em todos os momentos.

Aos meus queridos irmãos, Teté e Lucas, pela confiança, companheirismo, amor e amizade.

Ao professor Arturo Hernández-Domínguez, por sua preciosa orientação e paciência.

À minha grande amiga, Márcia, por sempre me apoiar e incentivar nos momentos de dificuldades.

Aos meus familiares, que tanto me incentivaram, confiaram e torceram por mim ao longo destes anos.

Aos professores do curso de Pós Graduação em Modelagem Computacional de Conhecimento da Universidade Federal de Alagoas.

A todas as pessoas não citadas aqui, mas que estão implicitamente vinculadas a esta caminhada.

“A melhor maneira de predizer o futuro, é inventá-lo” (Alan Kay).

Resumo

Desde a sua concepção e desenvolvimento, o FA_PorT, um *framework* que permite a criação de sistemas de ensino *on-line* via *Internet*, tem mostrado ser uma ferramenta de grande potencial no desenvolvimento de aplicações no contexto de educação a distância através da *Internet*. Porém, notou-se que, nos sistemas instanciados pelo FA_PorT, seria necessário um tratamento diferenciado/personalizado junto aos alunos com desempenho não satisfatório em uma sessão de ensino *on-line*. Assim, a idéia foi a possibilidade de criarem-se sessões personalizadas de reforço a esses alunos. Dessa forma, foi proposto que um tutor móvel fosse incrementado ao FA_PorT, possibilitando a execução de uma sessão de ensino *off-line* na máquina do aluno. O sistema tutor móvel possibilita fornecer uma assistência ao aluno com desempenho insatisfatório, já que o conteúdo das sessões da assistência será configurado considerando o desempenho obtido pelo aluno durante uma sessão de ensino *on-line*. Um sistema tutor móvel é migrado, através do uso de um agente móvel, para o computador do aluno, proporcionando sessões de ensino sem a necessidade de manter uma conexão com o sistema *on-line*. Uma vez concluída a tarefa do tutor móvel, o agente móvel voltará ao computador de origem. A nova funcionalidade no FA_PorT foi projetada e o sistema tutor móvel foi implementado utilizando o a tecnologia Jade para agentes móveis.

Palavras-chave: agentes móveis, *framework*, sistemas tutores, educação a distância.

Abstract

Since its conception and development, the FA_PorT, a framework that allows the creation of systems education via the Internet, has been shown to be a tool of great potential in the development of applications in the context of distance education via the Internet. However, it was noted that in systems instantiated by FA_PorT, would require a different treatment/custom among the students with unsatisfactory performance in a session of teaching on-line. So the idea was the possibility to set up build custom sessions to these students. Thus, it was proposed that a mobile tutor be increased to FA_PorT, enabling the run of a training session off-line on the machine of the student. The mobile tutoring system allows providing to learners, particularly with unsatisfactory performance, on assistance, since the content of the sessions of the assistance will be set considering the performance obtained by the learner during an on-line teaching session. A mobile tutoring system is migrated through the use of a mobile agent for the learner's computer, providing teaching sessions without having to maintain a connection with the on-line system. Having completed the task of the mobile tutoring system, the mobile agent will come back to the origin computer. The mobile tutoring system was implemented using the Jade for mobile agents.

Keywords: mobile agents, framework, tutoring systems, distance education.

b

Lista de Figuras

Lista de Quadros

Sumário

1. Introdução

1.1. Motivações

A educação a distância é uma crescente modalidade de ensino no mundo contemporâneo; realidade esta, que também é observada no Brasil. Assim como no ensino tradicional, nos cursos a distância é necessário planejar o ensino para que, de fato, a aprendizagem ocorra.

Neste sentido, surge a necessidade de personalização do ensino em nível tal que chegue até a individualização, assim como propõe a técnica de ensino denominada de Sistema Personalizado de Instrução (PSI) que, posteriormente, por questões de tradução, passou a chamar-se de Sistema Personalizado de Ensino (PSI), que apresenta as seguintes características, conforme Keller(1968), citado por Teixeira (2002), a saber:

> O programa de ensino devia organizar-se numa seqüência ordenada de unidades, avançando dos conteúdos mais simples para os mais complexos. O ritmo próprio de aprendizagem dos alunos devia ser respeitado. A individualização do ensino estava, portanto, presente no PSI. Exigia-se um padrão de excelência de desempenho para avançar no programa. Isso correspondia à exigência do domínio pleno do aprendido ou à perfeição na unidade para avançar no programa. Os requisitos de individualização e de padrão de excelência de desempenho combinados, garantiam, como foi dito, a efetividade do ensino para todos. Palestras e demonstrações eram usadas apenas como veículos de motivação e eram freqüentadas apenas por alunos que tivessem um número especificado de unidades no curso. Funcionavam, portanto, como reforço positivo para esses alunos. As comunicações entre professor e alunos eram feitas sempre por escrito. Isso impunha clareza nas relações professor/aluno e impedia dúvidas e ambigüidades a respeito do curso. Propunha-se ainda, o uso de monitores (geralmente alunos mais avançados) na administração do curso. Com isso, preservava-se o aspecto sócio-pessoal do processo educacional, personalizando-o (TEIXEIRA, 2002).

Roca (apud SANCHO, 1998) reforça a tese ao dizer que na maioria dos profissionais da educação, já existe a consciência de que cada pessoa é diferente das outras, que cada uma tem as suas necessidades próprias, seus objetivos pessoais, um estilo cognitivo determinado,

que cada pessoa usa as estratégias de aprendizagem que lhe são mais positivas, possui um ritmo de aprendizagem específico etc.

Assim, as diferenças e individualidades dos alunos representam uma grande riqueza que é conveniente explorar na educação a distância. O apoio a um aluno com dificuldade na aprendizagem e a personalização do ensino é o *insight*[1] observado e a principal motivação para o atual trabalho.

1.2. Contexto da dissertação

O Mestrado em Modelagem Computacional de Conhecimento tem uma proposta interdisciplinar e organiza-se em linhas de pesquisa em torno de uma única área de concentração, em Modelagem Computacional de Conhecimento. Essas linhas de pesquisa são:

- Descoberta de Conhecimento e Otimização de Decisões;
- Modelagem Computacional em Educação;
- Modelos Quantitativos e de Simulação.

Este trabalho faz parte da linha de pesquisa em Modelagem Computacional em Educação, tendo como contexto a educação a distância.

Nesta dissertação, o sistema considerado para o desenvolvimento de sistemas de ensino *on-line* é o *framework*[2] FA_PorT (MEDEIROS, 2006). O FA_PorT permite a criação de sistemas de tipo portfólio-tutor (NASCIMENTO, 2002). Um portfólio-tutor é um sistema de ensino *on-line* via *Internet* que permite a realização de sessões de ensino *on-line*. As sessões são realizadas seguindo uma agenda de sessões definida pelo professor. No contexto de alunos que apresentam dificuldades durante uma sessão de ensino *on-line*, com desempenho não satisfatório, é necessário auxiliar esses alunos de tal forma que um aluno possa ter uma assistência/atendimento individualizado objetivando solucionar suas dificuldades de aprendizagem. Após a assistência individualizada, objetiva-se melhorar a compreensão do aluno sobre o conteúdo da sessão anterior, para que, sem dificuldades, este possa participar da próxima sessão *on-line*.

[1] Sinônimo de intuição/epifania, ou seja, percepção súbita de um assunto ou problema.

[2] Um framework permite o reuso de software e representa um esqueleto de aplicações que podem ser personalizadas. (FAYAD; SCHMIDT; JOHNSON, 1999).

Neste contexto, é apresentado um sistema tutor móvel objetivando fornecer ao aluno um suporte adequado para a realização da assistência individualizada através de sessões de ensino ditas *off-line*, desconectado do sistema *on-line*, que acontece quando uma sessão de ensino *on-line* é concluída e o aluno com desempenho não satisfatório recebe no seu computador um sistema tutor móvel que possui um conteúdo[1] diretamente relacionado aos conceitos ou dificuldades identificados que provocaram o desempenho não satisfatório. Assim, o objetivo da utilização do sistema tutor móvel, é melhorar a compreensão do aluno sobre o conteúdo que gerou suas dificuldades, de forma que não seja necessária conexão à *Internet*, ou seja, a sessão ocorre de forma *off-line* e é realizada pelo aluno no seu computador e no horário desejado.

1.3. Objetivos da dissertação

O objetivo geral desta dissertação é a concepção e desenvolvimento de um Sistema Tutor Móvel acoplado ao *framework* FA_PorT, possibilitando uma assistência personalizada e individualizada aos alunos com dificuldade de aprendizagem na sessão de ensino *on-line* em uma aplicação instanciada pelo FA_PorT. Os objetivos específicos que podem ser destacados para alcançar esta tarefa são os seguintes:

- Utilizar agentes móveis Jade na implementação do Tutor Móvel;
- Permitir que o componente desenvolvido seja plugado ao *framework* FA_PorT, possibilitando que o desenvolvedor de aplicações crie aplicações Portfólio-Tutor com um Tutor Móvel acoplado a partir do uso de componentes no *framework*;
- Possibilitar que as aplicações instanciadas pelo FA_PorT provenham uma assistência personalizada e individualizada através de sessões de ensino *off-line*;
- Possibilitar o envio, ao aluno, um tutor móvel com conteúdos relacionados aos conceitos vistos na sessão de ensino *on-line*;
- Tornar possível que a sessão de ensino *off-line* seja realizada em qualquer local e horário e independentemente de conexão com *Internet*.

[1] Nesta dissertação, podem ser considerados conteúdos didáticos: definições, exemplos, exercícios, estudo de caso, relatórios de projetos, links úteis etc..

1.4. Organização da dissertação

Esta dissertação está organizada em seis capítulos, além deste. Os demais o levantamento bibliográfico sobre Sistemas Tutores Inteligentes, Agentes Móveis e do framework utilizado; também a especificação e implementação do sistema tutor móvel e, por fim, as considerações finais; conforme segue no Quadro 1.1.

Quadro 1.1 - Organização da dissertação

Capítulo	Título	Descrição
2	**SISTEMAS TUTORES INTELIGENTES**	Apresentam-se definição, arquitetura básica e exemplos de sistemas tutores inteligentes.
3	**AGENTES MÓVEIS**	São abordados conceitos, características, ciclo de vida, plataformas, vantagens, desvantagens e aplicações relacionados à tecnologia de Agentes Móveis. Também é dado destaque às Aplicações de Agentes Móveis voltadas à educação.
4	**O FRAMEWORK FA_PORT**	São mostrados os requisitos do sistema, bem como, a parte arquitetural desse *framework.*
5	**MODELAGEM DO SISTEMA TUTOR MÓVEL PROPOSTO**	É apresentada a modelagem do Sistema Tutor Móvel proposto.
6	**IMPLEMENTAÇÃO**	O destaque é dado aos aspectos inerentes à implementação, estudo de caso e funcionamento do Sistema Tutor Móvel.
7	**CONSIDERAÇÕES FINAIS**	Apresentam-se os resultados obtidos nesta dissertação, as contribuições e as perspectivas para futuras pesquisas que podem ser realizadas a partir dos resultados obtidos.

2. Sistemas tutores inteligentes

2.1. Introdução

Sistemas de Tutoria automatizada, por exemplo, CAI - *Computer Aided Instruction*, ICAI - *Intelligent Computer Aided Instruction* e ILE - *Intelligent Learning Enviroment*, foi uma área do conhecimento bastante investigadas por pesquisadores, sobretudo, servindo para pesquisas na área de IA - Inteligência Artificial.

Para VICCARI e GIRAFFA (2003), os CAI surgiram no início de 1960, a partir de projetos na área de educação. Por este motivo, nem sempre este tipo de tutoria automatizada satisfaz os requisitos de uma teoria formal na área de Ciência da Computação. Os ICAI surgiram na década de 70 na tentativa de superar as limitações impostas pelos sistemas CAI. Para isso, passaram a incorporar recursos da IA e da Psicologia Cognitiva (COSTA, 1997). Sistemas ICAI passaram a ser conhecidos por muitos autores como sistemas tutores inteligentes (WENGER, 1987). ILE, também conhecido como Sistema Tutor Cooperativo ou Sistema de Aprendizagem Social, utiliza técnicas de IA em escopo. De acordo com Costa (1997), um ILE caracteriza-se como uma categoria de software educacional ao qual o aluno é inserido em uma situação de descoberta, englobando atividades de resolução de problemas. Durante esse processo, o tutor assiste o aluno em sua atividade e monitora o seu aprendizado. O foco da atenção no problema, por parte do aluno, passa a considerar mais o processo de solução, verificando passo a passo o seu conteúdo.

Por volta da segunda metade da década de 80, as pesquisas sobre sistemas tutores inteligentes tiveram concentração envolvendo questões pedagógicas e, no terceiro estágio na década de 90, o foco foi também na área pedagógica, sendo uma concentração específica em que existiam equipes interdisciplinares (COSTA; WERNECK, 1996). Os sistemas tutores inteligentes foram criados e têm potencial para serem utilizados pelos professores e alunos em diferentes domínios.

Ao se modelar um sistema tutor inteligente, deve-se considerar as características do domínio, o comportamento observável e mensurável do aluno e o conjunto de estratégias a serem adotadas pelo módulo tutor na busca de um ensino personalizado.

Novas tecnologias surgiram e permitiram aos pesquisadores o resgate de muitas questões em aberto na pesquisa de sistemas tutores inteligentes. A tecnologia de agentes tem se mostrado eficiente na modelagem e implementação de sistemas tutores inteligentes, conforme sugeriu Giraffa (1999).

Neste capítulo, serão abordados conceitos sobre sistemas tutores inteligentes, bem como, a arquitetura tradicional e alguns exemplos de sistemas tutores inteligentes.

2.2. O que é um sistema tutor inteligente

Para compreender novos conceitos, muitas vezes é necessário recorrer a analogias. A analogia imediata a um sistema tutor inteligente é o único outro agente capaz de realizar uma tarefa semelhante, o professor humano. A analogia entre o sistema tutor inteligente com o professor humano é unanimidade na literatura. Na verdade, ela é tão difundida que é difícil ter em mente que é apenas uma analogia.

Um dos motivos pelo qual há um campo tão grande e variado na área de sistemas tutores inteligentes, é que “sistemas tutores inteligentes” é um termo amplo. Conforme mencionado na sessão 2.1, sistema tutor inteligente é conseqüência natural dos primeiros sistemas CAI, que geralmente refere-se a um sistema baseado em frame com links projetados para tratar uma situação específica, isto é, hiperlinks com efeitos instrucionais.

Um sistema tutor inteligente é um sistema voltado ao ensino que busca modelar aspectos envolvidos na tutoria humana. São referenciados na literatura como sistemas que sabem o que ensinar (conteúdo), para quem ensinar (modelagem do aluno) e como ensinar (estratégias pedagógicas) (WENGER, 1987; SILVA, 2000; VICCARI; GIRAFFA, 2003; HATZILYGEROUDIS, 2004).

O ensino deve transcorrer de forma adaptada, ou seja, deve-se levar em conta o ritmo de aprendizado do aluno. A partir das interações do aluno com o sistema tutor durante uma sessão, o sistema tutor estabelece qual estratégia pedagógica será utilizada levando em conta o desempenho do aluno apresentado durante a realização da sessão de ensino.

2.3. Arquitetura clássica de sistemas tutores inteligentes

Há um notável consenso sobre a arquitetura padrão para sistemas tutores inteligentes: um sistema tutor inteligente consiste de componentes que envolve o aluno e a tutoria. Este é o *framework* básico apresentado na maioria dos recentes textos sobre sistemas tutores inteligentes. O modelo tradicional de sistemas tutores inteligentes propõe quatro componentes: o modelo do domínio especialista, o modelo do estudante, o modelo pedagógico e a interface do usuário. Uma extensão da discussão destes componentes pode ser encontrada em Polson e Richardson (1988). Seus projetos podem variar enormemente de acordo com o nível de inteligência envolvido entre os componentes. Entretanto, existem alguns componentes básicos que podem ser observados na literatura e, geralmente, sob diversas denominações, mas que possuem o mesmo propósito (WENGER, 1987; YAZDANI, 1987; BARR; FEIGENBAUM, 1982). A Figura 2.1 ilustra os módulos da arquitetura clássica de um sistema tutor inteligente (WENGER, 1987; GIRAFFA, 1999; SILVA 2000).

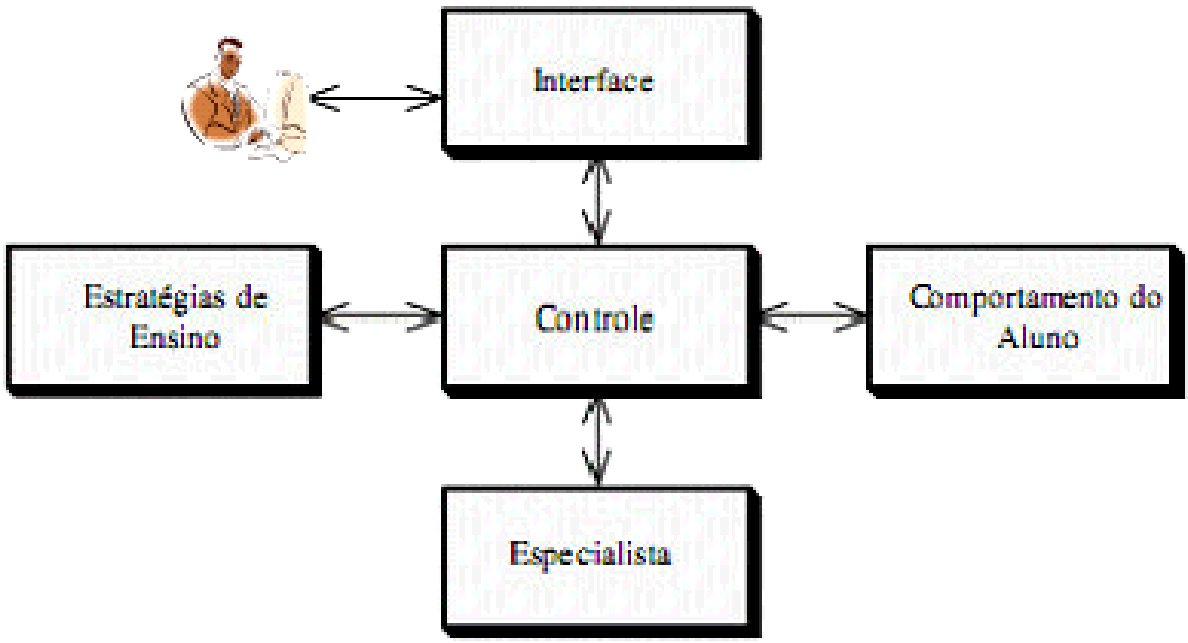


Figura 2.1 - Arquitetura clássica de um sistema tutor inteligente

2.3.1. Módulo domínio especialista

Como o próprio nome sugere, o módulo domínio especialista representa o conteúdo do conhecimento que o aluno irá adquirir. Este módulo é o coração de um sistema tutor inteligente e fornece a base para a interpretação das ações do estudante. Classicamente, o módulo do domínio especialista tem a forma de um sistema especialista que pode gerar soluções para os mesmos problemas que o aluno está resolvendo. É este módulo que manipula o conteúdo que vai ser ensinado pelo sistema tutor inteligente, onde podem ser encontrados: o material instrucional, mecanismos de geração de exemplos, formulação de diagnósticos etc.. Esse componente do sistema é muitas vezes referenciado na literatura com a denominação de "Base de Conhecimento do Domínio". Vários modelos de representação de conhecimento podem ser utilizados para o seu desenvolvimento, tais como: redes semânticas, regras de produção e *frames*.

Um dos mais importantes resultados na pesquisa sobre tutoria inteligente é a fidelidade cognitiva do módulo domínio especialista, possibilitando, ao tutor, "raciocinar" sobre um problema da mesma forma que os tutores humanos. Obviamente, o sistema especialista não aplica o raciocínio humano, mas se baseia nas suas decisões através de um conjunto de regras. Consequentemente, o tutor pode recomendar ótimas ações na resolução de um problema, inteiramente com base no que o estudante construiu. Esse é um dos fatores que diferencia um sistema tutor inteligente de um CAI convencional, já que o conteúdo é armazenado em uma base de conhecimento, com capacidade de "raciocínio" e não em uma base de dados convencional estática, contendo todos os fatos pertencentes ao domínio.

Para um sistema especialista obter sucesso em um cenário voltado à educação, é necessário desenvolver um verdadeiro modelo cognitivo do domínio que resolva exercícios da mesma forma que estudantes reais resolveriam. Esse seria um modelo ideal. Como resultado, um dos passos cruciais no desenvolvimento de um tutor inteligente, é uma cuidadosa análise da forma como os seres humanos, especialistas ou não, comportam-se na resolução de problemas.

2.3.2. Modelo do estudante

O modelo do estudante é um registro do estado do conhecimento de um aluno, devendo captar o estado do entendimento deste a respeito do assunto que está sendo apresentado (WENGER, 1987). Os dados que fazem parte deste módulo são de fundamental importância para que o sistema tutor inteligente possa comprovar hipóteses a respeito do aluno. É interessante que esse módulo seja atualizado dinamicamente, à medida que o sistema avalia o desempenho do aluno.

Classicamente, existem dois componentes no modelo do estudante: uma superposição do domínio de conhecimentos técnicos e um catálogo erros.

O primeiro componente é, essencialmente, uma cópia do modelo do domínio especialista, em que cada unidade de conhecimento é marcada com uma estimativa de quão bem o aluno aprendeu. O catálogo é erro é um conjunto de falsas ideias ou regras incorretas, cada um transportando uma indicação sobre uma possibilidade de equívoco do aluno.

Métodos estatísticos para estimar o estado do conhecimento de um estudante a partir de dados foi uma resposta fundamental na psicologia (GREEN; SWETS, 1973) e psicometria (CROCKER; ALGINA 1986). Em tutoria inteligente, o modelo do estudante acrescenta duas complicações. Em primeiro lugar, o estado do conhecimento do aluno não é fixo, e presume-se ser crescente. Segundo Corbett e Anderson (1995b), tem-se incorporado sucessivamente um modelo estatístico Bayesiano para resolver este problema. A segunda questão diz respeito à complexa granularidade. Tradicional, teorias da psicologia da aprendizagem poderiam assumir que cada resposta corresponde a um pedaço atômico cognitivo. No entanto, as ações de um estudante na resolução de problemas complexos podem refletir conhecimentos próprios que podem ser decompostos em componentes hipotéticos.

2.3.3. Módulo pedagógico

O módulo pedagógico, também chamado de módulo tutor, módulo instrutor, módulo de estratégias de ensino ou ainda módulo de estratégias didáticas, deve ser capaz de tomar as decisões sobre as estratégias de ensino a serem utilizadas e , determinando as informações que serão apresentadas a um aluno, semelhantemente como ocorre com um professor humano.

Pode-se dizer que o comportamento de um sistema tutor inteligente é determinado diretamente por este módulo.

Este é o módulo responsável pela estruturação das intervenções instrucionais e este pode funcionar em dois níveis, de acordo com Pirolli e Greeno (1988). No nível curricular, pode sequenciar tópicos para garantir uma adequada estrutura e individualizar a quantidade de conteúdo em cada nível, de forma que esse conteúdo seja assegurado aos alunos (CAPELL; DANNENBERG, 1993; CORBETT; ANDERSON, 1995b). No nível de apoio à resolução de problemas, pode-se intervir no sentido de recomendar aos alunos a resolução de problemas específicos. No Quadro 2.1, destacam-se os cinco tipos de intervenções instrucionais, segundo Towne e Munro (1992):

Quadro 2.1 - Tipos de intervenções instrucionais

Tipo	Intervenção instrucional
Demonstração de desempenho	O programa mostra uma seqüência de ações bem sucedidas em uma tarefa de resolução de problemas.
Passo a passo do desempenho	O programa prevê uma seqüência de ações para que o aluno siga até atingir a resolução de um problema.
Monitoramento de desempenho	O aluno resolve um problema fixado pelo programa. O programa intervém o aluno que comete algum erro.
Buscando o alvo	O aluno resolve um problema fixado pelo programa. O programa monitora o nível de abstração do problema, ao invés de ações individuais.
Exploração livre	A livre aprendizagem manipula o ambiente na resolução de problemas.

Fonte: (TOWNE; MUNRO, 1992)

Segundo Ohlsson (1987), um sistema tutor inteligente deve ter dois níveis de planejamento: estratégias e táticas pedagógicas, onde as estratégias pedagógicas possuem o conhecimento sobre como ensinar, ou seja, como gerar uma seqüência de táticas pedagógicas para apresentar com sucesso um determinado tópico a um determinado aluno e as táticas

pedagógicas são as ações necessárias para exteriorizar uma estratégia. O Tutor Móvel que está sendo proposto nesta dissertação utiliza o conceito sugerido por Ohlsson (1987). Desta forma, pode-se concluir que: uma estratégia particular é composta de um conjunto de táticas (ou ações) para realizá-la.

Existem diversos métodos de estratégias que podem ser utilizados em sistemas tutores inteligentes. Alguns deles são apresentados no Quadro 2.2, segundo Giraffa (1999):

Quadro 2.2 - Estratégias utilizadas em sistemas tutores inteligentes

Tipo	Intervenção instrucional
Método Socrático	Permite que sistema tutor inteligente ensine através de perguntas e diálogos, levando o aluno a tirar suas próprias conclusões.
Método Colaborativo (Assistente)	Faz com que o sistema tutor inteligente comporte-se como um colaborador na interação com o aluno, ajudando-o a esclarecer suas idéias.
Método *Coaching* (Treinamento)	Faz com que o sistema tutor inteligente observe o desempenho do aluno, a fim de aconselhá-lo nas realizações de suas atividades.

Fonte: (GIRAFFA, 1999)

Em outro nível, as táticas que podem ser utilizadas em estratégias de ensino são as seguintes: mostrar um exemplo usando uma situação similar; mostrar uma mensagem com a melhor opção; mostrar exemplos relacionados, porém sem nenhuma explicação; mostrar temas que são importantes, com o intuído de proporcionar maior atenção; mostrar o conteúdo de cada tópico; mostrar uma mensagem explicando as conseqüências de suas ações; mostrar ao aluno sucessivas questões para que ele possa analisar hipóteses, descobrir contradições e realizar inferências corretas (PEREIRA; D´AMICO; GEYER, 1998).

O método utilizado no sistema tutor inteligente, é que deve determinar quais táticas devem ser utilizadas em uma estratégia didática. Entretanto, caso o projetista do sistema tutor inteligente não esteja adotando nenhum método específico, ele deverá escolher o conjunto de

táticas para compor uma estratégia que melhor satisfazem os objetivos do sistema tutor inteligente.

É importante ressaltar, que muitos dos sistemas tutores inteligentes citados na literatura têm mais que uma estratégia didática, isto ocorre porque os sistemas geralmente têm princípios diferentes de instruir, tal qual um professor humano tem diferentes maneiras de apresentar o mesmo assunto a um aluno.

2.3.4. Interface com o usuário

O módulo de interface com o usuário, também chamado de ambiente de aprendizagem, é responsável pela comunicação entre o aluno e o sistema tutor inteligente. Através da interface, o sistema tutor inteligente pode apresentar o seu material instrucional e monitorar o progresso do aluno pela recepção de suas respostas.

Este componente define atividade para resolução de problemas em que o aluno é submetido. No mínimo, é constituído por um editor que representa ações do estudante. Por exemplo, no caso de tutores de programação, a interface pode ser um editor de texto (JOHNSON; SOLOWAY, 1984), editor estruturado (ANDERSON; REISER, 1985) ou editor gráfico (REISER; KIMBERG; LOVETT; RANNEY, 1992).

Existe um forte consenso sobre dois princípios sobre interface com usuário:

1. Deve se aproximar ao mundo real.
2. Deve facilitar o processo de aprendizagem.

Alguns autores consideram que uma boa interface é vital para o sucesso de qualquer sistema interativo. Em sistemas tutores inteligentes, o fator interação cresce de importância, já que se deve proporcionar ao aluno uma interação o mais amigável possível. Por isso, muitos desenvolvedores de sistemas tutores inteligentes optam pelo desenvolvimento de Interfaces Adaptativas.

2.3.5. Controle

O módulo de controle é responsável pela coordenação geral do sistema tutor inteligente e trata da comunicação entre os módulos, interface e eventuais chamadas a outros programas utilitários (GIRAFFA, 1999). Em alguns casos, esse módulo não é encontrado de maneira explícita nas arquiteturas e o controle fica distribuído entre os diversos módulos.

2.4. Arquitetura baseada na assistência ao aluno

Conforme Giraffa (2001), há algumas décadas, a arquitetura clássica, a restrição de hardware, de processamento de linguagem natural e de comunicação, acabaram tornando o tutor incapaz de tratar, em sua totalidade, o conjunto de informações que o professor possui em sala de aula.

Avanços e contribuições de outras áreas, como a de redes de computadores e sistemas distribuídos, trouxeram novas ferramentas para o ensino, como por exemplo, *e-mail*, fórum, *chat*, vídeo conferências, informações *on-line* etc., as quais colocaram relevância ao ensino-computadorizado, no que se refere a ensino personalizado e com a possibilidade de ser realizado em grande escala.

Para atuar de forma a complementar ao ensino em sala de aula, auxiliando o professor em sua tarefa, um sistema tutor inteligente necessita dar assistência com base em um processo contínuo, ao qual Giraffa (1999) chama de coreografia (Figura 2.2).

Figura 2.2 - Coreografia da assistência do STI ao aluno (GIRAFFA, 1999)

Para o autor, esta coreografia representa assistência personalizada do tutor, em função do modelo cognitivo atual do aluno. Esta assistência tem com finalidade organizar o domínio de acordo com os objetivos educacionais modelados no sistema tutor inteligente (como por exemplo, o paradigma construtivista). Esta organização do domínio busca alterar o estado cognitivo do aluno (modelo do aluno), finalizando um ciclo na coreografia desenvolvida pelo tutor.

Desta forma, os sistemas tutores inteligentes podem desempenhar suas tarefas, auxiliar o aluno e principalmente coletar informações e complementar as tarefas do professor.

2.5. Exemplos de sistemas tutores inteligentes

As pesquisas em sistemas inteligentes têm produzido e certamente continuarão proporcionando um campo de visão para os problemas relacionados com aprendizagem e instrução.

O desenvolvimento de um software educacional inteligente, atualmente requer uma grande quantidade de esforço e uma diversidade de conhecimentos e técnicas. Além disso, muitos destes programas têm sido indevidamente rotulados “inteligentes”. Portanto, um estudo comparativo de sistemas tutores inteligentes é importante para os profissionais das diversas áreas do conhecimento envolvidos na construção destes sistemas. Certamente, novas

pesquisas trarão avanços consideráveis em áreas como interação homem-máquina e aprendizagem humana e de máquina.

A seguir, serão descritos alguns dos esforços historicamente importantes no desenvolvimento de sistemas tutores inteligentes. Procurou-se dar destaque a sistemas tutores inteligentes mais recentes aos mais antigos. A maioria destes sistemas tutores inteligentes tem sido extensivamente documentados, como por exemplo, em Wenger (1987).

2.5.1. LEAP

O projeto LEAP - *Learn Explore And Practice* (SCOTTISH GOVERNMENT, 2007), é resultante do esforço cooperativo entre a *US WEST Advanced Technologies, Learning Systems* e *Mass Markets* para desenvolver uma plataforma de sistema tutor inteligente multimídia, com a capacidade de permitir uma abordagem inteligente relacionada ao contato entre o funcionário e o cliente. O LEAP proporciona atividades de aprendizagem bastante realistas através dos seus cenários e dos processos de simulação.

2.5.2. PROPAT

PROPAT (DELGADO; BARROS, 2004) é parte do projeto Eclipse (ECLIPSE) do Instituto de Matemática e Estatística da Universidade de São Paulo, financiado pela IBM© para a construção de um Ambiente Integrado de Desenvolvimento para cursos de introdução à programação. O Projeto foi desenvolvido inicialmente para a linguagem C. O plug-in PROPAT contém duas perspectivas: a Perspectiva do Aluno (Figura 2.3): em que os alunos podem escolher exercícios e programar por meio da seleção e inserção de padrões, ou ainda podem escrever livremente seu próprio código; e a Perspectiva do Professor: usada pelo professor para especificar novos exercícios e padrões para a Perspectiva do Aluno.

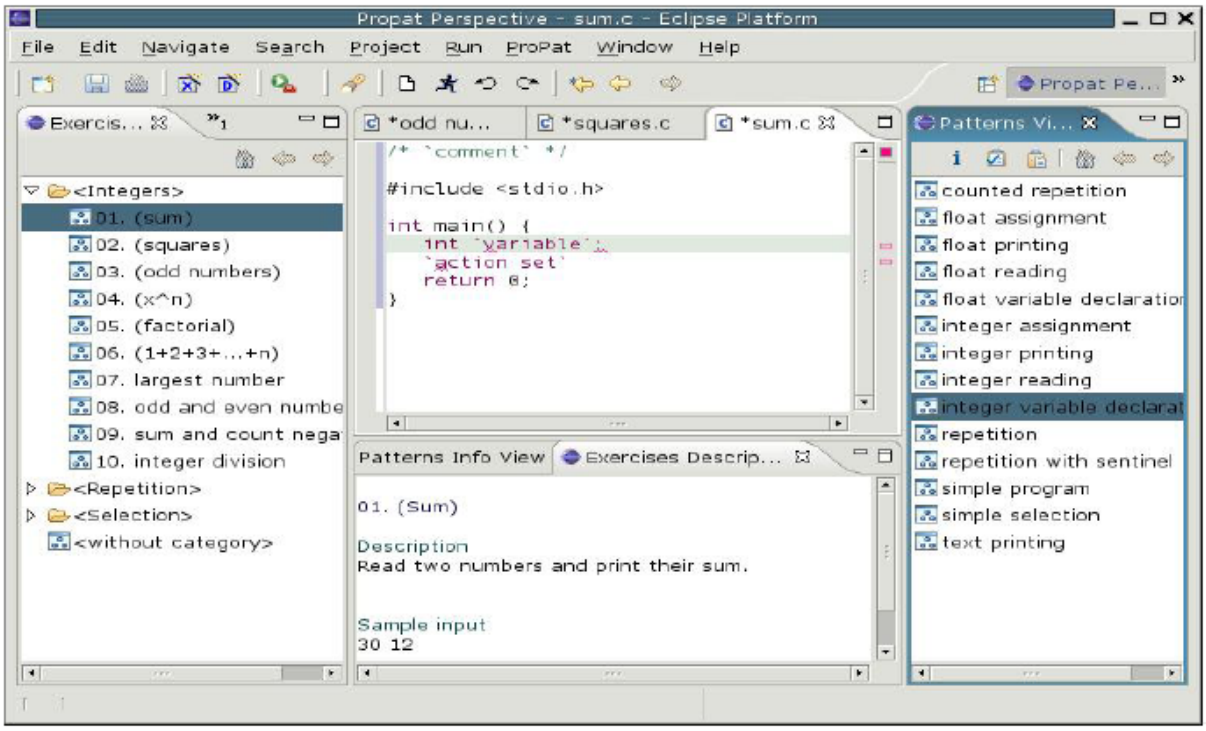


Figura 2.3 - Ambiente de programação PROPAT

2.5.3. AMANDA

O projeto AMANDA, de Eleuterio e Bortolozzi(2004), trata da concepção, implementação e validação de métodos computacionais destinados a apoiar as atividades de discussões de grupo em ambientes de aprendizagem a distância. O método AMANDA (ELEUTÉRIO, 2002), resultou no desenvolvimento de um ambiente computacional de mesmo nome. O ambiente AMANDA - Ambiente de Mediação e Análise de Discussões Argumentativas, é um sistema de comunicação mediada por computador destinado a promover interatividade entre participantes de discussões a distância, como as que ocorrem nos fóruns de discussão em ambientes virtuais de aprendizagem. O ambiente AMANDA difere dos sistemas de fórum convencionais por realizar a mediação da discussão de maneira inteiramente computacional, por meio de algoritmos inteligentes. O método AMANDA fundamenta-se na prática da argumentação e na produção de rodadas sucessivas de discussão entre os participantes. A cada rodada de discussão, o ambiente AMANDA detecta divergências de opinião e contra-argumentações e envolve progressivamente os participantes em uma interação coletiva de natureza argumentativa. Desde seus primeiros experimentos em

situações de aprendizagem a distância, o ambiente AMANDA tem demonstrado ser capaz de fomentar a interação em discussões de grupo e aliviar o trabalho de mediação humana, especialmente em discussões de grande escala. O projeto AMANDA possui ainda duas vertentes, além de seu propósito original de mediação das discussões. A primeira vertente refere-se à avaliação do comportamento argumentativo dos participantes, gerando indicadores de participação com base na análise das rodadas de discussão. A segunda vertente diz respeito ao processamento automático de texto sobre os resultados da discussão. Nesta última vertente, o projeto AMANDA explora a possibilidade de geração automática de sínteses da discussão em linguagem natural, bem como a produção de questões a partir de representações ontológicas.

2.5.4. ELM-ART

O projeto ELM-ART - ELM *Adaptive Remote Tutor* (WEBER; BRUSILOVSKY 2001), consiste na disponibilização de um sistema tutor inteligente na *www* para suportar o ensino da linguagem de programação LISP. O ELM-ART é considerado um livro texto inteligente e integrado com um ambiente resolvedor de problemas e sempre disponível para utilização. Apresenta os materiais através de recursos hipermídia e se difere dos demais hiperlivros da rede por dois aspectos: conhece o material que está sendo apresentado ao aluno e o assiste, permitindo oportunidades diferenciadas aos alunos. O segundo aspecto é que os exemplos e problemas se constituem em experiências reais, onde o aluno pode investigá-los.

A Figura 2.4 ilustra uma página com um problema de programação em LISP.

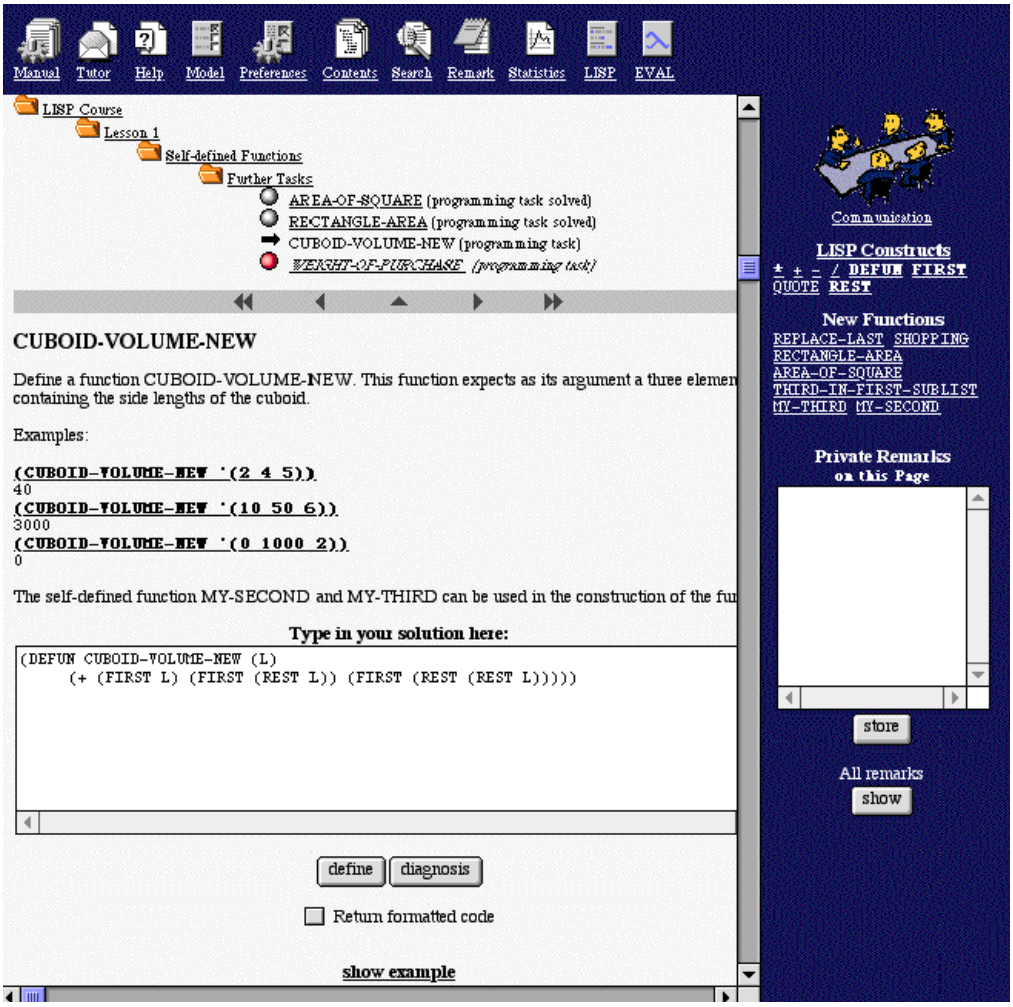


Figura 2.4 - Sistema tutor inteligente ELM-ART

2.5.5. Guidon

O sistema Guidon, de Clancey (1987), é um sistema tutor especialista para o ensino de diagnóstico de doenças infecciosas do sangue, que foi desenvolvido a partir de base de conhecimento já formada do MYCIN, talvez o mais antigo e conhecido sistema especialista, e cuja concepção original foi inspirada pelas capacidades de diálogo do sistema Scholar.

Os sistemas especialistas parecem oferecer uma base ideal para a construção de sistemas tutores. Além do fato óbvio de apresentarem grande quantidade de conhecimento especialista, outra vantagem, é a usual separação da base de conhecimento contendo as regras de produção do interpretador procedimental que as utiliza. Ainda que um sistema especialista tenha uma boa capacidade de explicação, ele pode somente justificar suas ações

passivamente. Para ser capaz de ativamente apresentar o conhecimento, um sistema tutor necessita de técnicas adicionais para selecionar o material educacional, ser sensitivo ao estudante e conduzir uma interação de modo bastante eficaz (WENGER, 1987). Por isso, os resultados da pesquisa de Clancey (1987) não foram os desejados, embora tenham deixado importantes contribuições tanto para as pesquisas em sistemas especialistas quanto para os sistemas educacionais.

A estratégia da apresentação pedagógica adotada pelo Guidon utiliza o método de caso: um diálogo de iniciativa mista concentra sobre casos específicos para transmitir o conhecimento do MYCIN aos estudantes em um contexto de resolução de problemas bastante realístico.

Os pesquisadores do Guidon perceberam que o importante conhecimento estrutural (hierarquias de dados e hipóteses de diagnóstico) e o conhecimento estratégico (pesquisa do espaço de problema através de refinamentos "*top-down*") estavam implícitos às regras. Isto é, o conhecimento procedimental que ocasionava um bom desempenho na resolução de problemas em uma consulta ao MYCIN não estava disponível para propósitos de ensino. Para tornar este conhecimento explícito, um novo sistema foi desenvolvido, o NEOMYCIN, que separa as estratégias de diagnóstico do conhecimento do domínio e faz bom uso da organização hierárquica de dados e hipóteses. Clancey (1987b) faz uma boa análise dos problemas enfrentados:

> "[...] ensino e explicação, nós reconhecemos, exigem diferentes demandas de um especialista que simplesmente resolve problemas. Um professor pode fazer analogias, ter múltiplas visões e níveis de explicação que são desconhecidos do MYCIN. Na construção do MYCIN, nós não tornamos explícito como um especialista organiza seu conhecimento, como ele recorda este conhecimento, e quais estratégias ele utiliza para problemas próximos [...]".

A principal contribuição (referencia) da primeira versão do Guidon é a identificação e o tratamento separado de diferentes tipos de conhecimento que devem ser disponibilizados para um tutor funcionar eficazmente.

2.5.6. Hydrive

Hydrive - HYDRaulics Interactive Video Experience, de Kaplan (1995), é um sistema tutor inteligente que incorpora multimídia para resolver problemas de sistema hidráulico de um avião F-15. Apesar do Hydrive utilizar um disco laser externo para suportar imagens de vídeo, o conteúdo do disco laser (seqüência de animações e imagens) é representado na base de conhecimento utilizada pelo sistema.

O propósito deste sistema tutor inteligente, é dar instrução aos técnicos de vôo para a solução de problemas complexos. O sistema, quando necessário, pode selecionar um conteúdo de vídeo, a partir de um disco laser, e então apresentá-lo. As pessoas que estão sendo treinadas podem ver os componentes da aeronave em operação. O sistema também pode apresentar instruções realísticas utilizando vídeo dos pilotos e mecânicos da aeronave.

No Hydrive, o material multimídia é disponível para o sistema através da codificação de uma representação do conteúdo na forma de regras na base de conhecimento. Quando uma seqüência em particular necessita ser mostrada, ela pode ser localizada através desta representação. Esta incorporação de vídeo no Hydrive satisfaz as exigências de um sistema tutor inteligente para proporcionar uma boa instrução aos funcionários que forem treinados. Os modelos definem o que o sistema deve fazer após a ação da pessoa em treinamento, a resposta do sistema pode então ser interpretada como uma determinada meta de apresentação.

O ambiente de implementação do Hydrive consiste de três elementos distintos: C, IL (Interface Language) e Arity Prolog. A linguagem C passa informação entre a interface e a máquina de inferência. IL é uma linguagem orientada a eventos utilizada para criar interfaces complexas consistindo de apresentações gráficas de alta qualidade e com recursos de controle de vídeo. O Arity Prolog foi utilizado para criar a máquina de inferência do Hydrive - os modelos do sistema, do estudante e educacional (KAPLAN; 1995).

Quando a implementação do sistema Hydrive começou, havia poucas alternativas para a implementação de um sistema tutor inteligente como este. Atualmente existem novas opções para a criação de sofisticadas interfaces, pois a ferramenta escolhida para a criação de um sistema tutor inteligente deve simplificar este processo ao máximo.

2.6. Conclusão

Neste capítulo, apresentaram-se os sistemas tutores inteligentes através dos ambientes de ensino-aprendizagem por computador, suas definições, os módulos da sua arquitetura e alguns exemplos de sistemas tutores inteligentes.

Assim, a partir do estudo realizado, buscou-se utilizar neste trabalho conceitos associados aos sistemas tutores inteligentes, permitindo que o professor possa definir diferentes estratégias de ensino que atenda o aluno de diversas formas ao ritmo de aprendizado do aluno.

Para que os sistemas tutores inteligentes possam ser incorporados de forma efetiva às aplicações educacionais, pesquisas com esse objetivo devem ser conduzidas envolvendo pelo menos três áreas de conhecimento: a Psicologia Cognitiva, a Inteligência Artificial e a Educação. Essas pesquisas fornecerão a base para a definição de modelos do domínio especialista, modelos do estudante e modelo pedagógico mais próximos dos processos cognitivos humanos.

No capítulo a seguir, será apresentado Agentes Móveis.

3. Agentes móveis

3.1. Introdução

Os benefícios relacionados aos agentes inteligentes não se limitam apenas aos aspectos pedagógicos. Outras soluções são igualmente aplicadas a diferentes aspectos do sistema de aprendizagem para melhorar o ensino personalizado. Juntamente à dificuldade de se implementar o ensino personalizado, há outro que está em evidência; que é a expansão da *Internet* de forma constante, onde a quantidade de informação disponível *on-line* expande proporcionalmente. A questão de como encontrar, reunir e obter eficientemente essa informação levou à investigação e desenvolvimento de sistemas e ferramentas que busquem fornecer uma solução para estes problemas. Os mais recentes temas relacionados a agentes apresentam uma nova "espécie" de agentes: os agentes móveis.

Agentes móveis são processos (por exemplo, programas em execução) que podem migrar de um *host* para outro (normalmente no mesmo sistema), a fim de satisfazer os pedidos feitos por seus clientes (ADNAN; DATUIN; YALAMANCHILI, 2000). Para isso, espera-se a execução em um *host* que provenha um recurso ou serviço ao qual se necessita para desempenhar a uma determinada tarefa. Se um *host* não contém os recursos/serviços necessários ou se o agente móvel requer um recurso/serviço diferente em outra máquina, o estado da informação desse agente móvel é preservado e, em seguida, este agente é transferido para um *host* que contenha o recurso/serviço necessário, assim, o agente móvel retoma a execução no novo *host*. Dessa forma, é necessária uma baixa largura de banda, ou seja, há a diminuição do tráfego na rede. Isso acontece porque não é necessária a troca de várias mensagens na rede para a realização da tarefa, haja vista que o agente migrado para o *host* destino continua sua execução sem a necessidade de comunicação com o *host* origem, assim, o tráfego pesado de mensagens ocorre localmente.

3.2. Conceitos

Nesta seção, serão apresentados alguns conceitos relacionados aos agentes de software, detalhando mais os conceitos voltados a agentes móveis. Há variadas definições

para estes conceitos encontradas na literatura, contudo, as apresentadas a seguir foram baseadas em Omg Masif (1998), sendo, assim, as utilizadas nesta dissertação.

- Agentes: são programas que podem agir de forma autônoma em nome de uma pessoa ou organização. Cada agente possui sua própria *thread* de execução, podendo, desta forma, executar sua tarefa de forma independente.
- Agente Estacionário: é aquele que executa no sistema onde foi gerado. Quando precisa de uma informação que esteja fora do sistema, ele utiliza mecanismos de comunicação tais como RPC (*Remote Procedure Call*) ou RMI (*Remote Method Invocation*).
- Agente Móvel: é aquele que não está restrito aos limites do sistema onde iniciou sua execução. Ele tem a habilidade de locomover-se de um sistema para outro através de uma rede. Esta habilidade permite que um agente se mova para um sistema que contenha um objeto com o qual ele precise interagir.
- Estado do Agente: pode ser tanto o seu estado de execução, quanto os valores de seus atributos, que determinam o que ele deve fazer quanto sua execução for iniciada no destino.
- Estado de Execução do Agente: é o estado do seu *runtime*, contador de programa e pilha.
- Proprietário do Agente: é a pessoa ou organização por quem o agente atua.
- Nome do Agente: nome pelo qual o agente é identificado e localizado via um serviço de nomes. A identidade de um agente é um valor único dentro do escopo de um proprietário e identifica uma instância particular de um agente. A combinação da identidade do agente e do proprietário do agente forma um nome único. Em ambientes com mais de um sistema de agentes, este nome único é formado pela identidade do agente, do proprietário do agente e no sistema de agentes. Desta forma, o nome pode ser usado como chave em operações que fazem referências à instância de um agente.
- Perfil do Agente: descreve as características deste agente, tais como: a linguagem em que ele é implementado e quais os métodos de serialização podem ser aplicados a ele.
- Sistemas de Agentes: são plataformas que podem criar, interpretar, executar, transferir e destruir um agente. Assim como um agente, um sistema de agentes é associado a uma pessoa ou organização por quem aquele sistema de agentes atua. Um sistema de

agentes é identificado por seu nome e endereço. Um *host* pode ser mais de um sistema de agentes.

- Agência: é um contexto dentro do sistema de agentes no qual o agente pode executar. Este contexto fornece ao agente funções, tais como, controle de acesso. As agências de origem e destino podem residir no mesmo sistema de agentes ou em sistemas distintos que suportem o perfil de agente.
- Região: é um conjunto de agentes que possuem o mesmo proprietário, mas não são necessariamente do mesmo tipo de sistema de agentes. O conceito de região permite que mais de um sistema de agentes represente o mesmo proprietário.

A ideia de agentes teve início com John McCarthy em 1950 e foi adotada por Oliver G. Selfridge quando os dois trabalhavam no MIT - *Massachusetts Institute of Technology*. Eles imaginaram um programa que pudesse executar determinada tarefa em nome no usuário de forma que este precisasse interagir o mínimo possível com o programa, ou seja, o sistema teria certa autonomia, bastando apenas apresentar ao usuário o resultado obtido (KAY, 1984). Consequentemente, foram surgindo várias ideias de agentes e uma delas foi a de agentes móveis, que como já mencionado, são programas que podem viajar através de uma ou mais redes, transportando tanto o código como o seu estado, realizando tarefas em máquinas que tenham capacidade de hospedar agentes. Isto permite que, além do aspecto de migração, também possam se replicar de tal forma que múltiplas instâncias de um mesmo agente executem em diferentes máquinas e retornem para o ponto de origem.

O paradigma de agentes móveis está apoiado na junção de dois conceitos: agentes e mobilidade de código. Já foi visto que um agente de software é qualquer programa que realize uma tarefa em nome de uma pessoa, da mesma forma que um agente, na vida real, representa seu cliente (agente de viagens, despachante, etc.) na execução de um serviço. Agentes móveis têm sido desenvolvidos como uma extensão para a abordagem de código móvel (por exemplo, *applet*) e poderá substituir o modelo cliente-servidor e as suas arquiteturas em um futuro próximo. Muitos pesquisadores desta área têm ampliado o código-móvel para o conceito de "objeto móvel" no qual um objeto (código + dados) é transferido de um *host* para outro. A abordagem de agentes móveis amplia ainda mais este conceito para a mobilidade de código, dados e estado (*thread*) de um *host* para outro, já que código móvel e objetos móveis

normalmente são migrados a partir de uma entidade externa, enquanto agentes móveis, usualmente, são migrados de forma autônoma.

Com relação ao conceito relacionado a mobilidade de código, pode-se compreender que é a capacidade de alterar, de forma dinâmica, as ligações entre fragmentos de código e a localização onde ele é executado (FUGGETA; PICCO; VIGNA, 1998). É um conceito bastante antigo, que já vem sendo utilizado desde a década de 70 em aplicações com entrada de *jobs* remotos e impressoras *PostScript*, onde o código era enviado para ser executado em uma máquina remota.

Antes do paradigma de agentes móveis, muitas abordagens têm sido propostas e desenvolvidas para a comunicação entre cliente e servidor, tais como: MP (*Message Passing*), RPC e REV (*Remote Evaluation*). Na RPC, o cliente envia os dados como parâmetros para um procedimento que reside no servidor. O processo será executado no servidor e os resultados serão enviados de volta para o cliente. O REV é uma arquitetura diferente da RPC; ao invés de chamar um procedimento remoto no lado do servidor, o próprio processo será enviado a partir do cliente para o servidor para ser executado e retorna o resultado. Resumidamente, em RPC os dados são transmitidos entre o cliente e o servidor, em ambos os sentidos. Em REV, o código é enviado do cliente para o servidor e os dados são devolvidos. Em contraste, um agente móvel é um programa enviado por um cliente para um servidor (DASGUPTA et al., 1999).

Para Chess, Harrison e Kershenbaum (1994), os agentes móveis são programas que podem ser enviados desde um computador cliente e transportados a um servidor remoto para a sua execução. Uma definição mais comumente aceita, no entanto, expressa os fatos de que:

- o agente móvel representa um usuário na rede;
- pode migrar autonomamente de um nó a outro para executar algum cálculo em nome do usuário. (Um nó é um *host* conectado à rede.)

Dessa forma, o usuário não precisa manter-se conectado à rede, uma consequência desejável em ambientes de desconexão frequente e de banda estreita como atualmente ocorre em computação móvel. Eles podem suspender a execução a qualquer momento, migrando para outro *host* para continuar a execução. Depois de criados, decidem quais locações visitar e quais instruções executar. Não precisa uma interação contínua com o *host* de origem. Em Barbosa (2007), é descrito de forma detalhada que a migração do agente móvel pode ser realizada por duas formas de mobilidade:

- Migração forte: é definida como a migração completa de um agente, ou seja, o agente migra carregando consigo seu código e seu estado de execução. Nesses termos, a migração forte implica total transparência para o desenvolvedor do agente. O agente pode ser interrompido em qualquer ponto de sua execução, sofrer uma migração, e retomar sua execução, de forma totalmente transparente;
- Migração fraca: somente o código é transferido entre *hosts* e, eventualmente, dados de inicialização. Como o estado de execução não é transportado, o *host* destino utiliza o código recebido para iniciar um novo agente ou então para ligá-lo dinamicamente a um agente já existente. Uma classificação mais detalhada de mobilidade fraca pode ser encontrada na tese de doutorado de Vigna (1997) e de Picco (1998).

Conforme ilustrado na Figura 3.1, a migração de toda uma unidade de execução (agente móvel) que encapsula o *know-how* para a realização de uma determinada tarefa. Este fato o diferencia dos demais paradigmas que simplesmente transferem código entre unidades de execução estáticas com relação a cada ambiente computacional. Para completar a tarefa programada, uma unidade de execução migra para os ambientes de execução que forneçam os recursos necessários. No ambiente remoto, o agente móvel pode acessar um recurso desejado, num estilo cliente-servidor.

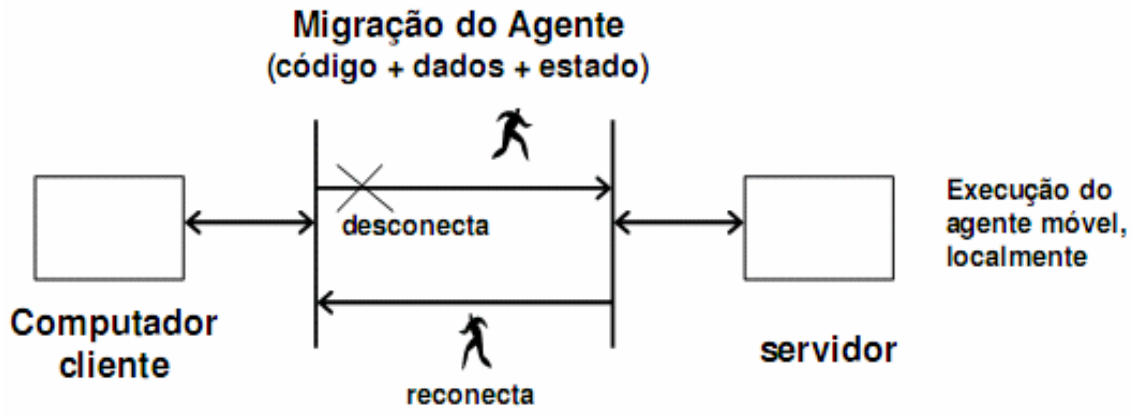


Figura 3.1 - Paradigma de agente móvel
(ANEIBA; REES, 2004)

O problema no modelo cliente-servidor é que o cliente limita-se às operações garantidas pelo servidor. Caso o cliente necessite de um serviço de que um determinado servidor não provê, é necessário encontrar um servidor que possa satisfazer o pedido, havendo

o envio de mais mensagens para outros servidores. Este é um claro uso ineficiente da largura de banda de rede. Além disso, este tipo de comunicação pode aumentar os atrasos na resposta devido ao tempo ocioso da máquina ou aos ruídos elétricos encontrados. Devido ao fator mobilidade dos dispositivos móveis (celulares, *PDAs*, *laptops*), esporádica desconexão é frequente em ambiente sem fios. Agentes móveis fornecem uma solução para o ambiente dinâmico dos dispositivos móveis (ANEIBA; REES, 2004). Kotz et al. (1996) reforçam essa tese mencionando que o agente móvel surge para resolver problemas significativos em comunicação com ou sem fios, como as desconexões, aumentou o tráfego de rede e outros.

Muitas questões já foram levantadas sobre as vantagens da utilização de determinado mecanismo de comunicação, ao invés de agentes móveis. Porém, como visto em Chess, Harrison e Kershenbaum (1994), a escolha correta do mecanismo de comunicação depende da aplicação a que se destina.

3.3. Características

De acordo com Etzioni e Weld (1995), um agente apresenta diversas propriedades algumas delas são listadas no Quadro 3.1. Destas, a única que normalmente é mencionada na literatura como necessária para caracterizar um agente, é a autonomia. Há outros autores que definem um agente como um programa que apresenta pelo menos as quatro primeiras propriedades do Quadro 3.1.

Quadro 3.1 - Atributos importantes que um agente deve possuir

Atributo	Significado
Reatividade	Capacidade de percepção e atuação
Autonomia	Cumprir seus objetivos sem a intervenção de terceiros
Personalidade	Manifestação de características especiais, tais como, "emoção"
Adaptabilidade	Aprender com sua experiência, ampliando seu conhecimento
Colaboração	Trabalhar em equipe com outros agentes para atingir um objetivo comum
Comunicação	Capacidade de comunicação com pessoas e outros agentes

Fonte: (Etzioni e Weld, 1995)

3.4. Ciclo de vida de um agente móvel

A seguir, o ciclo de vida de um agente móvel, segundo Schoeman e Loete (2003). No *host* local (Figura 3.2), uma entidade utiliza o *Authority API* para criar um ou mais agentes na *Agency*. Cada agente move-se da camada *Execution* às camadas *Mobility* e *Communication*, onde uma estrutura padrão é adicionada para assegurar a mobilidade e comunicação de forma eficaz. Mecanismos de tolerância a falhas são adicionados na camada *Persistence*. Na camada *Management Services*, o agente é serializado, antes de ser criptografado na camada *Agent Security*. Já na camada *Server*, o agente é nomeado e registrado para referência futura. O agente recebe credenciais de acesso a outros *hosts* na camada *Host Security*, antes de ser codificado por um protocolo de transporte apropriado na camada *Network* para o transporte através da rede. Ao chegar ao *host* remoto, a camada *Network* remove o cabeçalho referente ao protocolo de transporte e envia o agente à camada *Host Security*, onde o agente busca o acesso ao *host*, submetendo suas credenciais. Se for aceito, o *host server* registra o agente na camada *Server*, antes do agente ser decriptografado na camada *Agent Security*. A camada *Management Services* é responsável por deserializar o agente e realizar as conversões

necessárias. O agente move-se através das camadas *Mobility*, *Communications* e *Persistence* para ser executado em um *Place* na camada *Execution*. Durante a execução, o agente pode interagir com a camada *Persistence* para armazenar informações, com a camada Communication para a comunicação com outros agentes ou entidades e com a camada *Mobility*, para requisitar migrações futuras. Uma vez executado, o agente segue o trajeto de volta através da arquitetura de forma similar, até o *host* local.

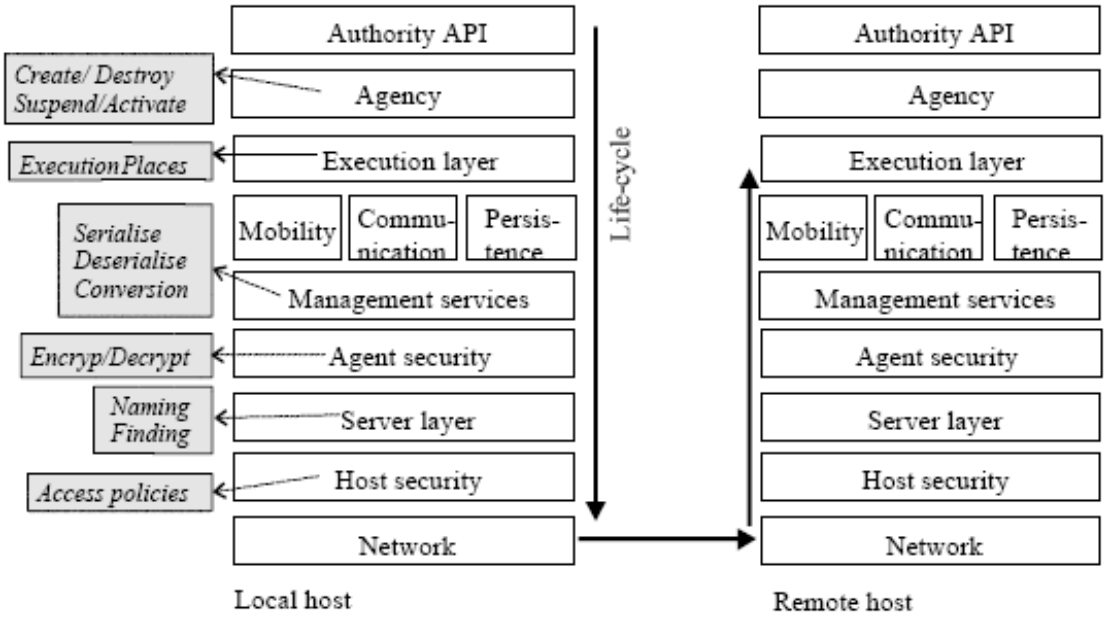


Figura 3.2 - Modelo de Arquitetura para Agente Móvel

(SCHOEMAN; LOETE, 2003)

3.5. Plataformas de agentes móveis

Como o FA_PorT (MEDEIROS, 2006), é uma ferramenta desenvolvida na tecnologia Java, obviamente, nos restringimos a investigar as plataformas de agentes móveis também baseadas em Java.

Altmann et al. (2001) relatam os resultados de uma comparação realizada entre vários ambientes de agentes móveis. Nesse relatório, onde as notas mais baixas representam melhor desempenho e os ambientes Grasshopper e Aglets (AGLETS, 2004) estão entre os melhores colocados, com notas 9,25 e 10,15 respectivamente.

Na época da realização dessa comparação, Jade (2009), outro sistema bastante popular na área de agentes - estava ainda em estágio muito inicial de seu desenvolvimento, obtendo uma nota muito alta - 15,75 - com relação ao Aglets e ao Grasshopper. No entanto, foi observado uma grande evolução dessa plataforma, a qual vem sendo cada vez mais utilizada por pesquisadores da área. De fato, atualmente, o projeto Jade parece ser o mais ativo de todos os projetos de desenvolvimento de ambientes para agentes móveis.

O restante desta seção irá destacar de forma breve as três tecnologias supracitadas, bem como, os seguintes sistemas de agentes móveis: Concordia, SOMA, Ajanta. Estas três últimas tecnologias foram analisadas com base em um *survey* sobre sistemas de agentes móveis obtidos em Uto (2003).

3.5.1. Grasshopper

O Grasshopper é uma plataforma de agentes móveis para Java, que segue o padrão Omg Masif (1998). Segundo Barbosa (2007), a ferramenta foi desenvolvida pela IKV++ Technologies - http://www.ikv.de/, o mesmo autor menciona que o tipo de migração oferecido pelo Grasshopper é a migração fraca.

Infelizmente, este projeto foi descontinuado. Dessa forma, não foi possível a instalação da ferramenta para realização de testes.

3.5.2. Jade

Jade (2009) - *Java Agent Development Framework*, é um sistema para agentes, inclusive os denominados agentes móveis, baseado em Java e desenvolvido pela Tilab (2009). Jade é regido pela licença de código aberto (GNU LGPL, 2007) e permite o desenvolvido de aplicações multi-agentes através do paradigma de comunicação *peer-to-peer*. Isso equivale a dizer que os agentes de Jade podem descobrir dinamicamente referências para outros agentes de modo a poder comunicar-se com eles. Os agentes podem estar hospedados em nós de uma rede com fio, ou mesmo em nós de rede sem fio. No Quadro 3.2 são destacadas algumas das características do Jade.

Quadro 3.2 - Características do Jade

Característica	Descrição
Interoperabilidade	Jade respeita as especificações Fipa (2009), uma iniciativa para a padronização de sistemas para agentes móveis. Sendo assim, Jade pode interoperar com todos os sistemas que respeitem esse padrão.
Uniformidade e portabilidade	Jade fornece o mesmo conjunto de *APIs* independentemente da versão de Java e tipo de rede. Em outras palavras, Jade fornece as mesmas *APIs* para J2SE, J2EE e J2ME, o que permite que uma mesma aplicação seja portada para essas diferentes arquiteturas
Transparência	a complexidade do sistema de agentes Jade é totalmente transparente para o programador, o que significa dizer que o programador não precisa ter conhecimento algum das especificidades de implementação de Jade

Fonte: (JADE, 2009)

O modelo de arquitetura de Jade é divido em duas partes principais:

1. Bibliotecas de desenvolvimento: representam a *API* das classes disponibilizada ao programador para a criação de um sistema multi-agentes;
2. Ambiente de execução: fornece os serviços necessários para que um nó possa executar os agentes.

Cada ambiente de execução que compõe o sistema é denominado contêiner. O conjunto de todos os contêineres é denominado plataforma. Sendo assim, a plataforma fornece uma camada de abstração homogênea que encobre as especificidades das camadas inferiores: *hardware*, sistema operacional, versão da *JVM* etc.

A Figura 3.3 esquematiza a arquitetura do JADE.

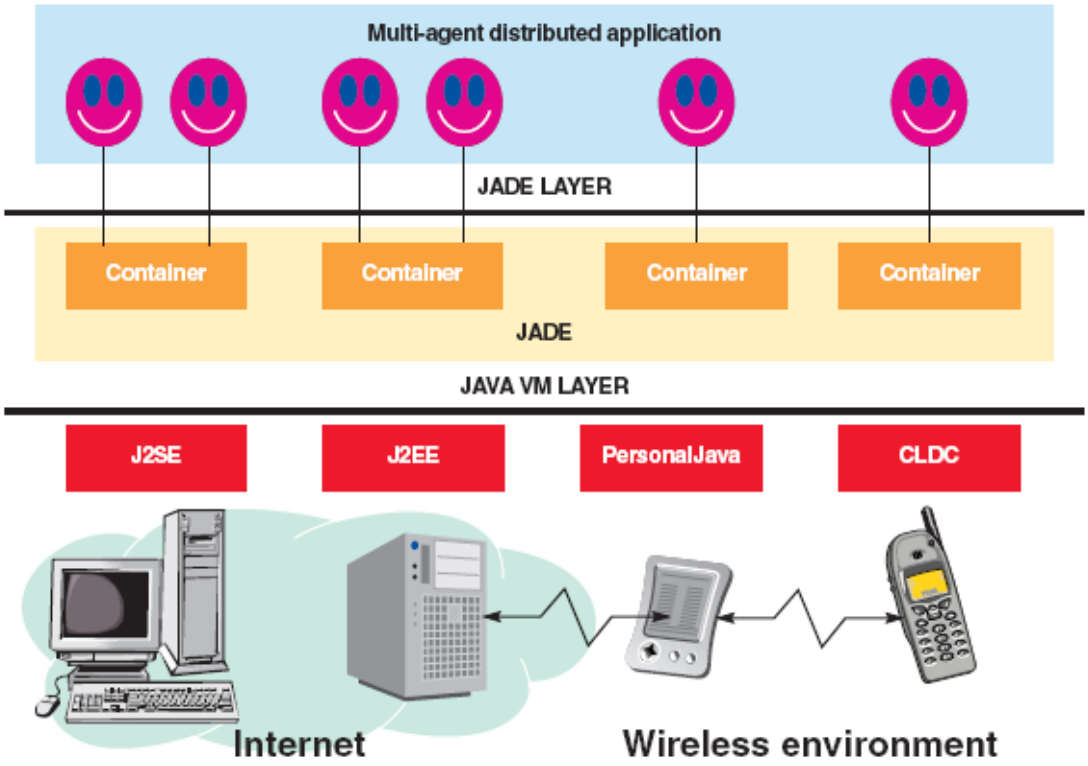


Figura 3.3 - Arquitetura JADE
(BELLIFEMINE et al., 2003)

Os ambientes de execução são disponibilizados para as diferentes versões de Java, tais como J2ME, J2SE, J2EE. Todavia, os mecanismos de migração são oferecidos somente para J2ME e J2SE. O tipo de migração oferecido por JADE é a migração fraca.

3.5.3. Aglets

Aglets Software Development Kit, antigamente denominado de *Aglets Workbench*, é um sistema de agentes móveis desenvolvido pela IBM *Tokyo Research Laboratory*. *Aglets* (AGLETS, 2004). O termo aglet é uma combinação das palavras *agent* e *applet*. Isso se deve ao fato de Aglets ter sido criado com base no modelo de *applets* de *Java*. O sistema é baseado na linguagem Java 1.1 e a primeira publicação data de 1996.

A IBM coordenou a maior parte das versões 1.x. Com o tempo, o projeto foi transformado em uma iniciativa de código aberto, regida pela licença da IBM para código aberto, que permite a utilização e modificação do código, além de liberar trabalhos e

comparações sobre o produto. O projeto passou a ser hospedado no SourceForge, (SOURCEFORGE, 2009), e, inicialmente, as primeiras versões lançadas pela comunidade visavam apenas a corrigir *bugs* do produto. Com o lançamento das versões 2.x, questões de gerenciamento de segurança foram melhoradas, além da introdução de mecanismos de registro (*log*) baseados no Log4J, (LOG4J, 2009). Depois do lançamento de algumas versões 2.x, o desenvolvimento parou. A partir de outubro de 2004, as atividades da comunidade foram reiniciadas comandadas por Luca Ferrari, um então aluno de PhD da Universidade de Modena e Reggio Emilia, na Itália. A documentação do produto, embora não muito completa, é relativamente organizada e oferecendo recursos básicos e mais avançados. Recursos para a criação, migração, clonagem, hospedagem e visualização de agentes móveis são oferecidos, além de recursos avançados de segurança, sincronização e troca de mensagens. O sistema segue parcialmente a especificação Omg Masif (1998).

Aglets, tal como Glasshopper e Jade, oferece somente o mecanismo de migração fraca, pois garante somente a migração de código e a migração de membros, sendo que a migração de recursos é oferecida apenas parcialmente (referências a certos recursos como *sockets* e arquivos locais não podem ser migradas).

A seguir, uma ilustração relacionada ao ciclo de vida de um Aglet (Figura 3.4), descrevendo e explicando cada processo envolvido nesse ciclo.

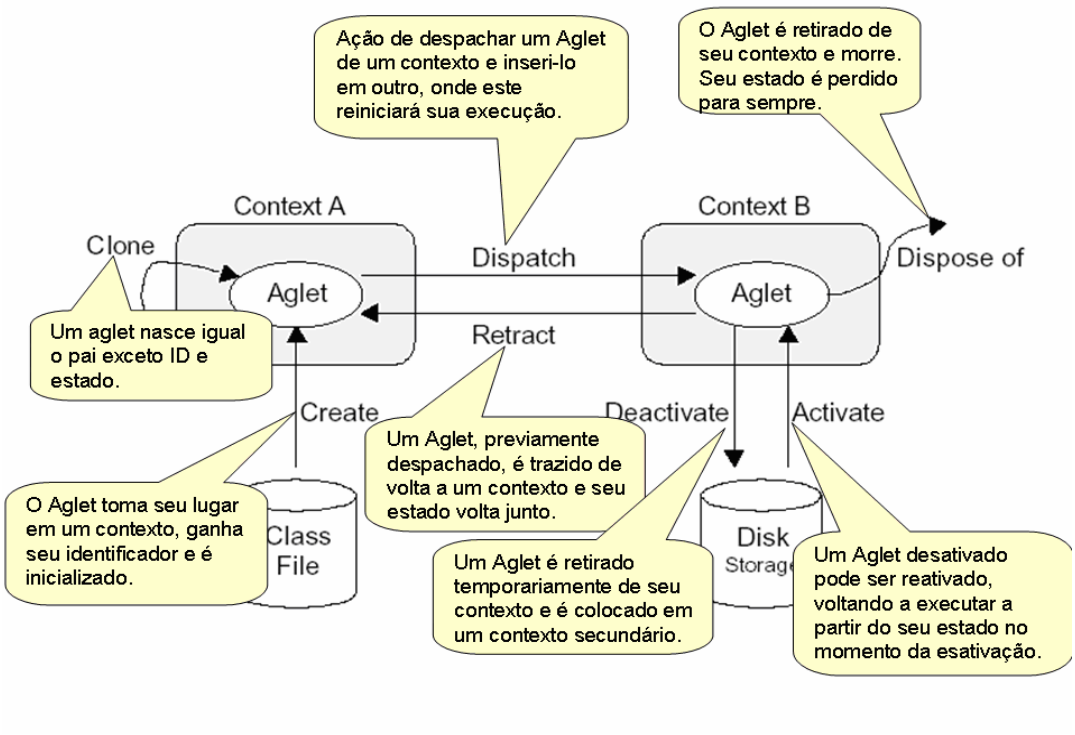


Figura 3.4 - Ciclo de vida de um Aglet

3.5.4. Tahiti

Tahiti é uma aplicação que é executada como um agente servidor (TAHITI, 2002). Pode-se executar múltiplos servidores (Tahiti) em um único computador atribuindo-lhes diferentes números de porta, assim como no Jade. Tahiti oferece uma interface ao usuário para o acompanhamento (Figura 3.5), que dá suporte para criar, enviar e eliminar agentes móveis, bem como, estabelece privilégios de acesso ao agente servidor.

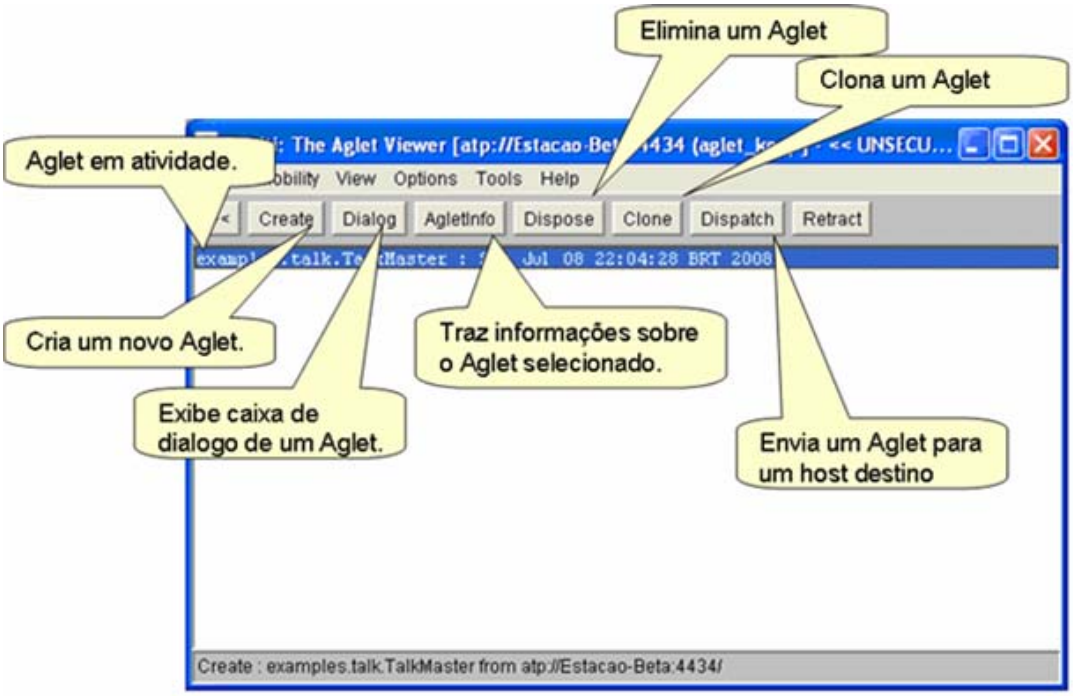


Figura 3.5 - Interface TAHITI

3.5.5. Concórdia

O Concórdia, é uma plataforma para o desenvolvimento e gerenciamento de aplicações de agentes móveis extensível a qualquer sistema que suporte *Java* (WONG et al., 1997). Foi desenvolvido pela *Mitsubishi Eletric Information Tecnology Center America* (MEITCA), com sede em Massachussets/EUA. Concordia consiste em múltiplos componentes, todos escritos em Java, que são combinados de forma a propiciar um ambiente completo para aplicações distribuídas.

Concordia é constituído de três partes: uma máquina virtual Java, um servidor Concordia e, pelo menos um agente. O servidor é composto de oito módulos que, instalados e em execução, lidam com todas as tarefas relacionadas ao manuseio dos agentes. Estes módulos são listados a seguir:

- Gerenciador de agentes – *Agent Mananger;*
- Gerenciador de segurança – *Security Mananger;*
- Gerenciador de persistência – *Persistence Mananger;*
- Gerenciador de comunicação entre agentes – *Inter-Agent Comunication;*
- Gerenciador de fila – *Queue Mananger;*

- Gerenciador de diretório – *Directory Mananger;*
- Gerenciador de administração – *Administration Mananger;*
- Biblioteca de ferramentas para agentes – *Agent Tool Library.*

3.5.6. Soma

O sistema de agentes móveis Soma - *Secure and Open Mobile Agent*) (SOMA, 2009) é um projeto em desenvolvimento no DEIS - Universidade de Bologna e iniciado em 1998. Os objetivos principais do projeto são segurança e interoperabilidade. Soma suporta a linguagem Java (JDK 2) e os sistemas operacionais Windows e Solaris. O binário e o código fonte do sistema estão disponíveis para uso não comercial e podem ser solicitados aos autores do projeto. É uma das poucas plataformas a ter uma real preocupação a nível de segurança do agente móvel contra servidores maliciosos.

Um servidor de agentes em Soma recebe o nome de *place*, como ocorre em outros sistemas de agentes móveis. Cada servidor é composto por diversos módulos:

- Gerenciador de agentes – *Agent Manager;*
- Gerenciador de recursos – *Local Resource Manager;*
- Serviço de informações distribuídas – *Distributed Information Service;*
- CORBABridge.

3.5.7. Ajanta

O sistema de agentes móveis Ajanta (AJANTA, 2009) é um trabalho desenvolvido na Universidade de Minnesota e iniciado em 1997. Suporta a linguagem Java 1.1.5 em plataformas Unix/Linux e necessita de Perl para ser instalado. O binário é disponibilizado gratuitamente para uso não comercial.

O principal foco do projeto Ajanta é nos mecanismos de segurança e execuções robustas de agentes móveis em sistemas abertos. Em Ajanta, o paradigma de agente móvel é baseado no conceito genérico de uma rede de objetos móveis. Ele também faz uso de várias outras facilidades de Java, como objeto serialização, reflexão e RMI.

A comunicação entre agentes sendo executados no mesmo servidor é realizada por meio da invocação de métodos ou do acesso compartilhado a algum recurso. No primeiro caso, um agente se registra no servidor como um recurso e fornece *proxies* aos agentes que desejem comunicar-se com ele. Além destes mecanismos de comunicação local, Ajanta permite que agentes interajam com pares localizados remotamente por meio de RMI autenticado.

3.6. Vantagens e Desvantagens

O uso de agentes móveis pode trazer várias vantagens para a aplicação. Algumas dessas vantagens, baseadas em Danny e Mitsuru (1999), são mostradas abaixo. Em seguida, algumas desvantagens trazidas pela aplicação dos agentes móveis são apresentadas.

3.6.1. Vantagens

- Redução do tráfego na rede: Sistemas distribuídos demandam um grande volume de comunicação (interação) para realizar tarefas, principalmente quando há restrições de segurança envolvidas. Agentes móveis podem reduzir o tráfego da rede, pois permitem despachar tarefas que podem executar suas interações localmente. Agentes móveis podem ainda reduzir o tráfego de dados da rede, pois permitem mover o processamento para o local onde os dados estão armazenados ao invés de transferir os dados para depois processá-los. O princípio é simples: "Mover o processamento para os dados ao invés de mover os dados para o local de processamento";
- Diminui a dependência da latência da rede: sistemas críticos necessitam de respostas em tempo real para mudanças no ambiente. O controle desses sistemas através de uma rede substancialmente grande ocasiona uma latência inaceitável. Agentes móveis oferecem uma solução, pois podem ser despachados pelo controlador central para realizarem suas tarefas localmente.
- Execução assíncrona e autônoma: tarefas podem ser embutidas em agentes móveis que podem ser despachados pela rede. Após serem despachados, os agentes são autônomos e independentes da criação de processo, podendo executar de forma assíncrona. Este

recurso é útil, por exemplo, para um dispositivo móvel (ex. um *laptop*) pode se re-conectar à rede para coletar o agente mais tarde.

- Adaptação dinâmica: agentes móveis possuem a habilidade de perceber mudanças no ambiente de execução e reagir autonomamente. Múltiplos agentes podem interagir entre si e se distribuir pela rede, de modo a manter uma configuração ótima para resolver um problema em particular.
- Naturalmente heterogêneos: redes de computadores, geralmente são heterogêneas, tanto na perspectiva de *hardware* como na de software. Agentes móveis são independentes da máquina e também da rede, sendo dependentes somente do seu ambiente de execução, não dificultando a integração de sistemas.
- Robustez e tolerância a falhas: a habilidade dos agentes móveis de reagirem dinamicamente a situações e eventos desfavoráveis torna fácil a construção de sistemas distribuídos robustos e tolerantes a falhas. Se uma máquina está para ser desligada, todos os agentes em execução na máquina podem ser advertidos para que possam ser despachados e continuar suas tarefas em outra máquina da rede.

Certamente, essas vantagens podem ser obtidas usando outros paradigmas que não os agentes móveis. Contudo, conforme destacado por Chess, Harrison e Kershenbaum (1994), se considerarmos a soma das vantagens, é que podemos perceber o grande benefício de usar agentes móveis, uma vez que o uso deste paradigma na aplicação traria todas essas vantagens juntas, sem a necessidade de combinação de vários paradigmas.

3.6.2. Desvantagens

Os agentes móveis também podem trazer algumas desvantagens. Primeiramente, existe a necessidade de instalação de um sistema extra (plataforma) em cada máquina que o agente poderá visitar. O código do agente é geralmente interpretado nestas plataformas, tendendo, assim, a ser mais lento. Deve-se evitar mover muito os agentes sem necessidade, pois isso pode aumentar o tráfego da rede, e como colocado em Peine (2002), pode-se ter desvantagens usando agentes móveis se não se atentar para dois pontos em relação ao tamanho do agente: o código do agente e os dados carregados. O código do agente deve ser o menor possível, portanto deve-se fatorar suas funcionalidades, deixando apenas aquelas que

são indispensáveis para migrar com o agente. Os dados carregados pelo agente também devem ser bem definidos, para que o agente não carregue dados que não serão mais usados ou que podem ser facilmente reconstruídos. Um último problema, que está presente em todas as tecnologias de aplicações distribuídas, é a segurança. A segurança é um dos problemas mais difíceis de resolver e vem sendo motivo de muita pesquisa como em Chess (1998) e em Tahara et al. (2000). Além disso, como mostrado em Tahara et al. (2001), quanto mais segurança, mais lento fica o sistema.

3.7. Aplicações

Existem diversas aplicações para o uso de agentes móveis e a maioria delas envolve procura de informações em nome de um usuário e, possivelmente, a execução de uma tarefa quando determinada informação é encontrada. No Quadro 3.3, são apresentadas algumas aplicações com base em Matos (2003) e outros autores:

Quadro 3.3 - Aplicações com agentes móveis

Aplicação	Descrição
Coleta de dados de vários lugares	uma das principais diferenças entre agente móvel e código móvel, é o itinerário. Enquanto o código móvel viaja de um *host* X para um Y, o agente móvel possui um itinerário que ele segue para ir de um lugar a outro. Uma aplicação natural para agentes móveis é a coleta de informação em vários lugares de uma rede.
Busca e filtragem	devido ao grande número de sites na *Internet*, a quantidade de informação disponível é imensa, acarretando também em aumento de quantidade de informação irrelevante. Em nome de um usuário, um agente móvel pode visitar diversos sites, procurar por informações disponíveis e criar um índice de *links* para trechos de informações que coincidem com um determinado critério ou preferências do usuário.
Monitoramento	novas informações são constantemente produzidas e disponibilizadas na rede. Agentes móveis podem ser enviados para monitorar quando determinada informação estará disponível, e quando estiver, executar determinada tarefa.
Processamento paralelo	dado que um agente pode mover-se através de uma rede e criar novos agentes, uma aplicação potencial para o seu uso é o processamento paralelo. Se o processo requer mais tempo de CPU do que o disponível localmente, a tarefa pode ser distribuída em diversos agentes para serem executados em outras máquinas.
Comércio eletrônico e comércio móvel	outro uso potencial de agentes móveis, é no comércio eletrônico. Um agente pode procurar em diversos sites de venda um determinado produto e comprá-lo naquele onde o preço for menor. Podendo interagir com outros agentes, com o intuito de negociar o preço de um produto, considerando que haja um agente de compra e um de venda.
Aprendizagem *on-line*	em um ambiente educativo, os agentes móveis podem ser utilizados para proporcionar uma aula personalizada, com base nas deficiências ou eficiências do aluno. Por ser móvel, o aluno pode ser assistido sem conexão com *Internet*, haja vista que todo o *knowhow* está embutido no agente móvel migrado.

Fonte: Baseado em Matos (2003) e outros autores

Uma extensão natural, é o uso de agentes móveis em comércio/aprendizagem eletrônico com mobilidade - *m-commerce* / *m-learning*. Assim, o uso de agentes torna-se ainda mais interessante, já que o agente age de forma autônoma e independente, não sendo necessário que o usuário permaneça conectado durante várias transações, o que é bastante útil devido às limitações dos dispositivos móveis atuais.

Os principais trabalhos sobre agentes móveis encontrados durante a pesquisa estão dentre este período os anos de 1995 a 2000. Essa foi a maior dificuldade encontrada durante as pesquisas. Particularmente, esta carência de trabalhos mais recentes se dá pela consolidação da área. Sendo que, as referências mais atuais envolvem a relação de agentes móveis com outros assuntos, como: segurança, paralelismo, padrões de projeto, grades computacionais etc.

Dentre as pesquisas encontradas, destacam-se:

- Computação móvel: onde Aneiba e Rees (2004) destacam que os agentes móveis têm sido usados em aplicações que vão desde a gestão das redes distribuídas, como a gestão das informações. Já Dasgupta et al. (1999) tem produzido várias aplicações com agentes móveis que podem ser aplicadas em comércio eletrônico. Por exemplo, um sistema que lida com recuperação de informação no campo de *m-commerce* tem sido desenvolvido. Matos (2003) apresenta um modelo de negociação automatizada para comércio móvel utilizando agentes móveis. Gialdi (2004) descreve o ICoMP (*I-centric Communication Mobile Portal*), um modelo para Portal Móvel que oferece serviços aos usuários baseado em um perfil (preferências, recursos disponíveis, localização e contexto).
- Segurança de Sistemas de Agentes Móveis: Em sua dissertação, Uto (2003) trata sobre um estudo detalhado dos aspectos de segurança de sistemas de agentes móveis, compreendendo as classes de ameaças existentes, os requisitos de segurança e os mecanismos propostos para solucionar ou amenizar tais ameaças.
- Grades computacionais: Barbosa (2007), em dissertação de mestrado, apresenta um projeto de implementação de um arcabouço de suporte a agentes móveis, o MobiGrid, dentro de um ambiente de grade denominado *InteGrade*.

- Segurança em sistemas distribuídos: Custódio (2002) apresenta um modelo para desenvolvimento de uma aplicação baseada em agentes móveis, baseando-se em uma política de segurança que reduz os riscos existentes na utilização de agentes móveis.
- Padrões de projeto para agentes móveis: Lima (2004) propõe uma classificação e formalização de padrões de projeto para agentes móveis, análise de desempenho de combinações destes padrões.
- Gerenciamento de redes: Uma pesquisa que trata de um sistema de agentes para o gerenciamento de rede (NASSIF; COSTA; RESENDE, 2004).
- Desenvolvimento baseado em componentes: É apresentada uma abordagem que combina Ontologias, Serviços Web Semânticos e Agentes Móveis para o Desenvolvimento Baseado em Componentes (SANTANA et al., 2007).
- Computação ubíqua: Cardoso (2005) estuda a utilização de agentes móveis com o objetivo em aplicações de multimídia e computação ubíqua, visando a garantia de qualidade de serviço.

Na seção a seguir, será dado destaque a aplicações educacionais que utilizam agentes móveis.

3.8. Agentes móveis na área educacional

De acordo com Sampson e Karagiannidis (2002), particularmente, em ambientes de aprendizagem na web, os agentes móveis oferecem benefícios específicos quando comparados a agentes estáticos, tais como:

- Podem ser usados para buscar previamente o conteúdo do domínio que, por ventura, o aluno possa necessitar futuramente, com base no acompanhamento do aluno em interações anteriores. Dependendo do estado da rede, uma solicitação imediata ou uma reserva pode ser feita com a ajuda de um agente móvel. Desta forma, há uma melhoria na qualidade do serviço para entrega de conteúdos educacionais distribuídos, especialmente quando estão envolvidos grandes arquivos multimídia. Assim, a tecnologia de agentes móveis pode evitar atrasos desnecessários, em face da limitação de largura de banda.
- Com o contínuo aumento do número de usuários móveis, o acesso à aprendizagem baseada em ambientes web tende a aumentar com o advento dos dispositivos portáteis de

computação, tais como, computadores portáteis, *palmtops*, celulares e de livros eletrônicos. Estes dispositivos podem ter pouco grau de confiança, no que se refere à baixa largura de banda, alta latência ou deficiências em conexões de rede sem fio. Agentes móveis surgem como um instrumento essencial para aumentar a eficácia de tal acesso.

- Agentes móveis oferecem um novo paradigma no desenvolvimento de sistemas de aprendizagem com um maior nível de abstração e unificação de processos e objetos. Em termos de escalabilidade do sistema e fácil autoria, essas características dos agentes móveis oferecem uma filosofia flexível e eficaz no desenvolvimento, no projeto e na escalabilidade do ambiente de aprendizagem.

- Os ambientes de aprendizagem baseados na web geralmente compartilham recursos em diferentes sistemas. Os computadores e as redes em que tais sistemas são desenvolvidos tendem a ter caráter heterogêneo. Como os sistemas de agentes móveis são geralmente independentes de computadores e de rede, eles podem fornecer um excelente suporte para sistemas distribuídos e compartilhamento de recursos.

Outra motivação aliada ao uso de agentes móveis em ambientes educacionais, é que esses agentes podem ser utilizados para proporcionar uma aula personalizada, com base nas deficiências ou eficiências do aluno. Devido à característica móvel desse tipo de agente, o aluno pode ser assistido sem conexão com *Internet*, haja vista que todo o *knowhow* está embutido no agente móvel.

Uma evolução do uso de agentes móveis em ambientes de aprendizagem está na adoção da aprendizagem móvel, isto é, *mobile learning* / *m-learning*. Assim, o uso de agentes torna-se ainda mais interessante, já que o agente age de forma autônoma e independente, não sendo necessário que o usuário permaneça conectado durante várias transações, o que é bastante útil devido às limitações dos dispositivos móveis atuais.

A seguir, serão exemplificados alguns investimentos em pesquisa e desenvolvimento voltados a aplicação da tecnologia de agentes móveis em ambientes de ensino-aprendizagem, a *e-learning* e, também, a um crescente conceito na literatura ligado à informática na educação, a *m-learning*.

3.8.1. e-Learning

Lin (2004), em um de seus trabalhos, trata da melhoria da *e-learning* através da aplicação da tecnologia de agentes móveis. Para o autor, a disponibilidade de infra-estrutura de banda larga, tais como: GPRS, 3G e redes UMTS, os avanços em tecnologias *wireless* (CHEN; NAHRSTEDT, 2000) e a popularidade dos dispositivos portáteis (MICROSOFT, 2001) proporcionam um novo caminho para a educação, estendendo a duração da aprendizagem e do espaço. Uma importante aplicação dos dispositivos móveis emergentes é a aprendizagem móvel (SHARPLES, 2000). Com o novo paradigma "*anytime, anywhere computing*", a *e-learning* tem sido estendida a *m-learning* (LEHNER; NÖSEKABEL 2002).

Nos últimos anos, há inúmeros esforços no sentido da utilização de dispositivos móveis para fins educacionais (MOBILEARN PROJECT, 2009; CHANG; SHEU, 2002; BECTA REPORT, 2009; MEGAN FOX, 2007). No entanto, embora os dispositivos móveis estejam aproximando-se da ubiquidade, a indústria *m-learning* ainda está na sua infância. O *m-learning* tem uma série de deficiências comuns, tais como: o acesso a materiais didáticos é lento; cursos não se adaptam ao perfil de cada estudante; a interação real time entre o estudante e sistema é difícil de ser alcançado devido à conexão não confiável e limitação de largura de banda. Nos últimos quinze anos, tem-se assistido a um potencial interesse na tecnologia orientada a agentes e uma distinta tendência tem evoluído para a pesquisa sobre agentes inteligentes. Esta tendência está relacionada com a diversificação dos tipos de agentes sendo investigados e tipos mais populares incluem agentes de interface do usuário, sistemas multi-agente, agente móveis e assim por diante. Agentes inteligentes e, em particular, agentes móveis têm um enorme potencial para resolver essas deficiências.

O seu trabalho, Lin (2004) destacou como a tecnologia de agentes móveis pode resolver os problemas que limitam a potencial aprendizagem em ambientes móveis. O mesmo autor utilizou o *framework* Bee-gent, que é um *framework* para implementar agentes móveis. A tecnologia Bee-gent foi lançada em 1999 pela Toshiba (2009), como um novo tipo de *framework* para desenvolvimento de agentes. O Bee-gent é composto por dois tipos de agentes: agentes *Wrappers* e agentes mediadores.

- Agentes *wrappers* são usados para gerenciar as aplicações existentes. Os agentes *wrappers* gerenciam os estados das aplicações e invocam as aplicações, quando necessário.

- Agente mediadores apoiam a coordenação inter-aplicação de todas as comunicações entre aplicações. Os agentes mediadores movem-se de uma aplicação para outra, onde interagem com os agentes *wrappers* remotamente.

3.8.2. Gestão de recursos de e-Learning

Yan (2006) trata de uma pesquisa dedicada a agentes móveis e sua utilização para a gestão dos recursos/conteúdo em *e-learning*, fornecendo um breve panorama do assunto e elaborando um estudo de caso. O autor descreve uma visão geral introduz o conceito de agente móvel, enumerando os seus benefícios. Há uma discussão do estado da arte da aplicação no domínio *e-Learning*. No estudo de caso, foram usados agentes móveis com objetos de aprendizagem. Há também um protótipo onde se compara o desempenho da utilização de agentes móveis em relação ao paradigma cliente/servidor.

3.8.3. ESBMA

Liu et al. (2007) também tratam de uma pesquisa direcionada aos agentes móveis em sistema e serviços *e-learning*. Mencionam que a importância do *e-learning* tem sido transferida para a construção de ambientes personalizados de aprendizagem, onde é oferecido um tipo de conhecimento e serviços personalizados baseados em teorias modernas de pedagogia e psicologia. O foco da pesquisa é voltado a um sistema e serviços *e-learning* baseados em agentes móveis, o ESBMA, o qual foi proposto e desenvolvido. A estrutura do sistema, o processo de trabalho, as tecnologias-chave de realização do ESBMA, o projeto de agentes móveis e o desenvolvimento desses agentes foram introduzidos. Os resultados manifestam que o ESBMA pode apoiar a *e-learning*, proporcionando um aprendizado individualizado e personalizado.

3.8.4. Um sistema *e-Learning* P2P

Na pesquisa, Kawamura e SugaHara (2005) apresentam um novo *framework* para ambiente de ensino-aprendizagem. O sistema proposto tem duas características importantes.

Em primeiro lugar, é baseada em arquitetura P2P, proporcionando escalabilidade e robustez. Em segundo lugar, o conteúdo do ambiente não são apenas dados, mas também, agentes que possam marcar respostas do usuário, informar as respostas corretas e mostrar alguma informação extra sem interferência humana. Também está presente um protótipo do sistema proposto em Maglog, que é baseado no Prolog, para a implementação do sistema multi-agentes móveis em desenvolvimento. Desempenho e simulações demonstram a eficácia do sistema proposto.

Geralmente, além de disponibilizar um serviço a um exercício, um servidor de ambientes de ensino-aprendizagem presta serviços para marcar as respostas do usuário, informar as respostas corretas e mostrar algumas informações adicionais sobre o exercício em andamento. Assim, para o sistema proposto ser considerado um sistema distribuído de ensino-aprendizagem, não basta que apenas que os exercícios sejam distribuídos entre nós de uma rede de computadores. Funções para prestar os serviços acima citados também devem ser distribuídas entre os nós da rede. Adotou-se a tecnologia de agentes móveis para atingir esse objetivo; ou seja, em um exercício não há somente dados, mas também agentes que possam marcar respostas do usuário, informe as respostas corretas e mostram algumas informações adicionais sobre o exercício. Além disso, a tecnologia de agentes móveis é aplicada para realizar a migração das categorias, ou seja, cada categoria também é um agente no sistema proposto. Para maiores detalhes dessa interessante proposta, sugere-se uma leitura em Kawamura e SugaHara (2005).

3.8.5. Avaliação de conhecimento em ambientes *e-Learning*

Para DINSOREANU et al. (2003) e ANGHEL; SALOMIE (2003), *e-learning* é hoje um dos mais interessantes domínios do "e-" disponíveis através da *Internet*. O principal problema em criar ambientes virtuais baseados na web, é a escolha do modelo tradicional e aplicá-lo usando a tecnologia mais adequada. Dessa forma, os autores analisaram o domínio ensino a distância investigando a possibilidade de realizar alguns serviços de *e-learning* utilizando a tecnologia de agentes móveis. O trabalho apresenta um modelo de serviço para

avaliação do estudante e o *framework* JADE como a tecnologia mais adequada para implementação de agentes móveis, atingindo os objetivos almejados pelos autores.

Analisando as principais entidades envolvidas na avaliação do estudante, foram identificadas as seguintes:

- Entidade de aprendizagem (o aluno)
- Ensino (Instrutor)
- Tipo de avaliação (exame obrigatório, auto-avaliação)
- Teste
- Tipo de pergunta
- Pergunta
- Resposta correta
- Engine de avaliação

As relações entre as principais entidades são representadas de forma simplificada na Figura 3.6:

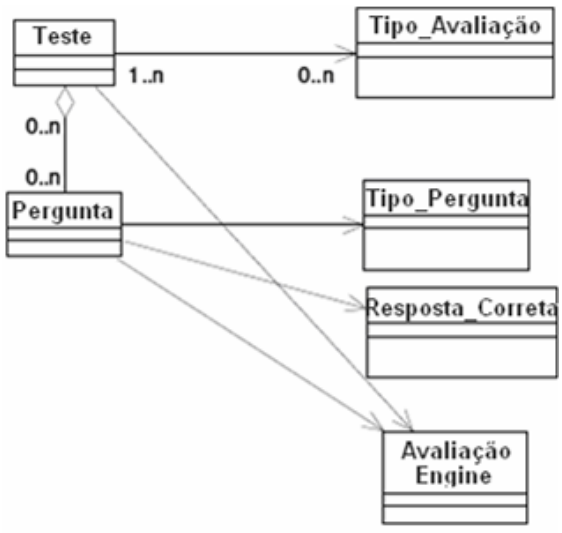


Figura 3.6 - Entidades e relacionamentos

Um ensino (instrutor) dispõe de testes. Um teste pode pertencer a diferentes avaliações (exames obrigatórios, auto-avaliações etc.) e contém um conjunto de perguntas. Uma pergunta está associada a um tipo de pergunta, para uma ou mais respostas corretas e também

para um *engine* de avaliação (implementado como um mecanismo de avaliação). O *engine* de avaliação fornece o conhecimento para avaliar a resposta do aluno em relação a(s) resposta(s) correta(s) associada a uma pergunta. Um aluno é capaz de acessar os testes disponíveis para executá-lo e fornecer suas respostas. Um aluno também deve receber *feedback* sobre o seu desempenho (o desempenho, as respostas corretas etc.)

Uma vez que se estar mencionando a um sistema altamente distribuído e considerando as suas limitações, investigou-se a possibilidade de fornecer uma solução baseada na tecnologia de agentes móveis. O objetivo foi conceber um sistema multi-agente móvel que preencha os requisitos funcionais e respeite as restrições. Nessa abordagem, o sistema multi-agente é considerado uma organização dos agentes. A organização de conhecimentos e capacidades são maiores do que a soma dos conhecimentos e capacidades dos diferentes agentes (WOOLDRIDGE; JENNINGS; KINNY, 2000; ZAMBONELLI et al., 2000; ZAMBONELLIet al., 2001).

3.9. Conclusão

Bastantes questões são constantemente relatadas na literatura sobre as vantagens da utilização de outros mecanismos de comunicação, como o RPC ou RMI, ao invés de agentes móveis, porém, conforme é destacado em Chess, Harrison e Kershenbaum (1994), a escolha certa do mecanismo de comunicação depende da aplicação a que se destina.

A tecnologia de agentes móveis é uma opção bastante interessante para a construção em ambientes distribuídos, principalmente em aplicações como computação móvel, pois permite que dispositivos móveis despachem agentes para a rede de modo que estes executem alguma tarefa em seu lugar, de forma mais eficiente, sem gastar os recursos dos dispositivos e independentemente das condições de conectividade. Dessa forma, agentes móveis contribuem de forma positiva para serviços e novas oportunidades de negócios através do melhor uso de recursos de comunicação (em termos de custo e desempenho) e suporte flexível à operação desconectada. Dessa forma, consideramos pertinente e viável a proposta de uma ferramenta de suporte a sessões de ensino *off-line*, baseada em agentes móveis e ligada ao *framework* FA_PorT. Assim, pretendemos adicionar uma nova tática que fornecerá apoio ao aluno de

forma *off-line*, ou seja, desconectada. A possibilidade de que esse reforço *off-line* ocorra através de dispositivos móveis, como celulares e *PDAs*, enriquece ainda mais o trabalho.

Quanto às diversas plataformas analisadas, observamos que o projeto Jade parece ser o mais ativo no que se refere a desenvolvimento de ambientes para agentes móveis. Outro fator que impulsiona o uso desta plataforma, é a ampla comunidade acadêmica que vem utilizando e enriquecendo o projeto.

Dos trabalhos encontrados na literatura, ficou clara a preocupação com a utilização da tecnologia de agentes móveis em detrimento ao modelo cliente/servidor, quando o assunto está relacionado ao domínio "ambientes educacionais". Essa preocupação fica mais intensa nos trabalhos onde o foco está voltado à aprendizagem móvel, pois é necessário lidar com um ambiente complexo e distribuído. Apesar dos avanços, esses dispositivos costumam ter conexões de rede poucos confiáveis, baixa largura de banda e alta latência. Dessa forma, uma interação *off-line* auxiliará tal dificuldade; a tecnologia de agentes móveis dá o suporte e disponibiliza os recursos essenciais, proporcionando uma alternativa atraente para implementar e melhorar a aprendizagem via ambientes móveis. Viu-se, também, que é aberto um novo nicho de pesquisa, que é a utilização de agentes móveis em ambientes de aprendizagem. Anghel e Salomie (2003) mostraram que a tecnologia Jade oferece o apoio oportuno no que se refere à utilização de agentes móveis em ambientes de aprendizagem.

No próximo capítulo, será apresentado o *framework* FA_PorT..

4. O *framework* FA_PorT

4.1. Introdução

O FA_PorT é um *framework* para sistemas portfólio-tutor baseado em agentes que permite a criação de aplicações portfólio-tutor (MEDEIROS, 2006). Um sistema portfólio-tutor é um sistema Web de ensino *on-line* para a aprendizagem de um grupo virtual de alunos. Para evoluir no desenvolvimento do *framework* FA_PorT com as novas funcionalidades relacionadas ao tutor móvel e sessões de ensino *off-line*, foi necessário compreender os requisitos, arquitetura, componentes, classes e interfaces associadas, e conhecer a camada tutor, que possibilita que uma sessão de ensino online seja definida.

4.2. Requisitos

A seguir, são descritos os principais requisitos do *framework* FA_PorT:

- *Framework* estruturado em camadas e componentes;
- Utilizar uma arquitetura de software reutilizável;
- Utilizar padrões de projetos;
- Trabalhar com agentes reativos, utilizando a ferramenta JADE;
- Permitir a criação de novas aplicações Portfólio-Tutor customizadas a partir do *framework* FA_PorT;
- Possuir um comportamento pró-ativo, com o objetivo de avisar sobre prazos de atividades e criação de sessões de ensino aos professores e aos alunos;
- Professores podem criar sessões de ensino contendo uma ou várias estratégias didáticas (que também é criada antecipadamente pelo professor, ou pode ser utilizada alguma estratégia didática existente), que são compostas por táticas.

4.3. Arquitetura do FA_PorT

Na Figura 4.1, é apresentada a arquitetura do FA_PorT e as camadas associadas a cada aplicação construída pelo mesmo.

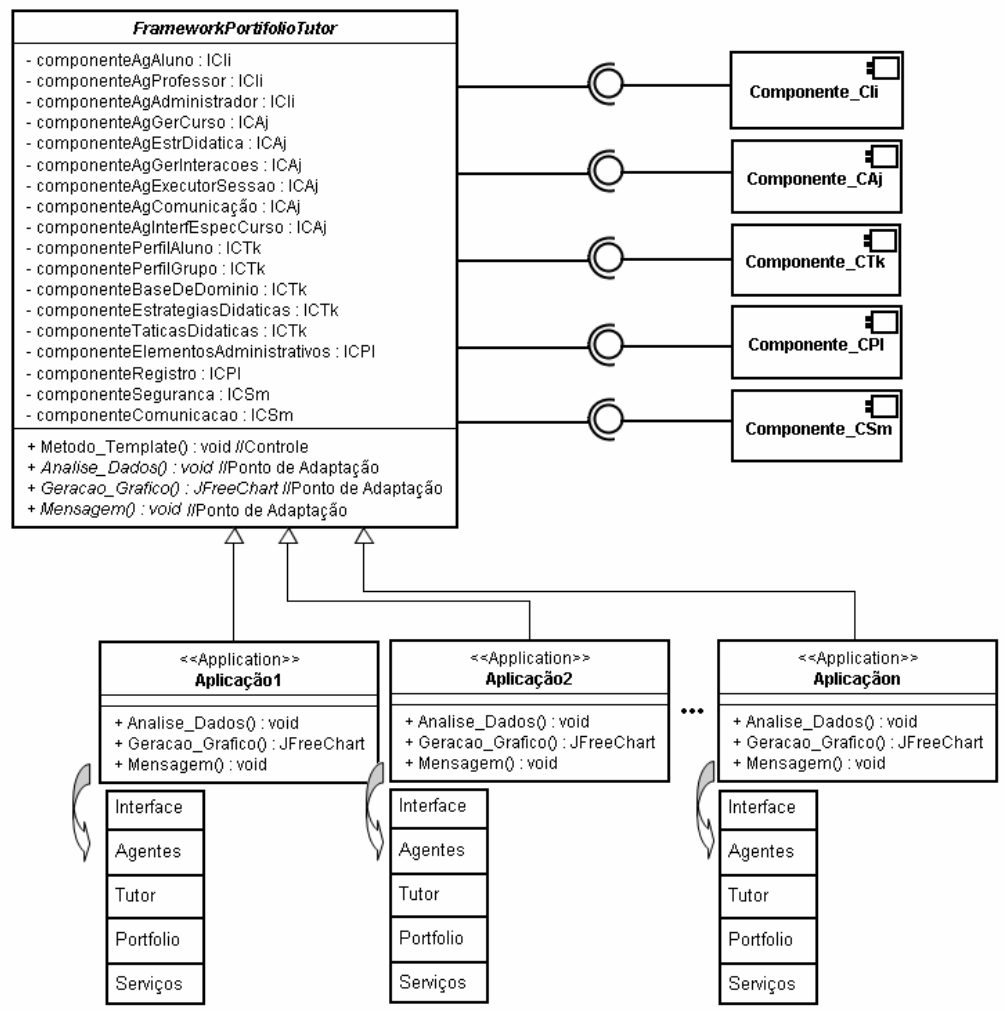


Figura 4.1 - Arquitetura do *framework* FA_PorT para sistemas portfólio-tutor.
(MEDEIROS, 2006)

Os elementos das camadas (Interface, Agentes, Tutor, Portfólio Eletrônico e Serviços) de uma nova aplicação portfólio-tutor são representados através de um conjunto de componentes organizados conforme apresentado no Quadro 4.1:

Quadro 4.1 - Conjunto de componentes

Componente	Descrição
Componente_CIi	representa o componente i da camada Interface.
Componente_CAj	representa o componente j da camada Agentes.
Componente_CTk	representa o componente k da camada Tutor.
Componente_CPl	representa o componente l da camada Portfólio Eletrônico.
Componente_CSm	representa o componente m da camada Serviços.

Fonte: (MEDEIROS, 2006)

Cada componente possui sua interface. A classe *FrameworkPortfolioTutor* possui atributos representando as interfaces dos componentes do *framework* e métodos (o método *template* que representa o controle e os pontos de adaptação de código).

O Quadro 4.2 detalha a arquitetura de um sistema portfólio-tutor e suas camadas:

Quadro 4.2 - Arquitetura de um sistema portifólio-tutor

Camada	Descrição
Camada Interface	gerencia as interações entre os alunos, professores e o sistema.
Camada Agentes	é representada por um conjunto de agentes, tais como, agentes de interface, agente de comunicação, agente executor de sessão e agentes associados aos módulos do tutor.
Camada Tutor	representa um sistema tutor que gerencia uma sessão de ensino *on-line*. São representados, nesta camada, os módulos da arquitetura básica de um sistema tutor (estratégia, domínio, o perfil do grupo e o perfil do aluno).
Camada Portfólio	permite o gerenciamento de atividades e o registro dos elementos (artefatos) de aprendizagem. Também, esta camada permite o envio de avisos de forma automática, via email, sobre datas de realização de sessões de ensino e atividades.
Camada Serviços	é representada por um conjunto de serviços sobre acesso ao banco de dados, segurança, comunicação e geração de relatórios.

Fonte: (MEDEIROS, 2006)

4.4. Diagrama de componentes

A seguir, a estrutura do *framework* FA_PorT é representada através do diagrama de componentes e as respectivas interações. De acordo com Medeiros (2006), os componentes foram definidos levando em consideração as entidades em comum de vários sistemas analisados (Figura 4.2).

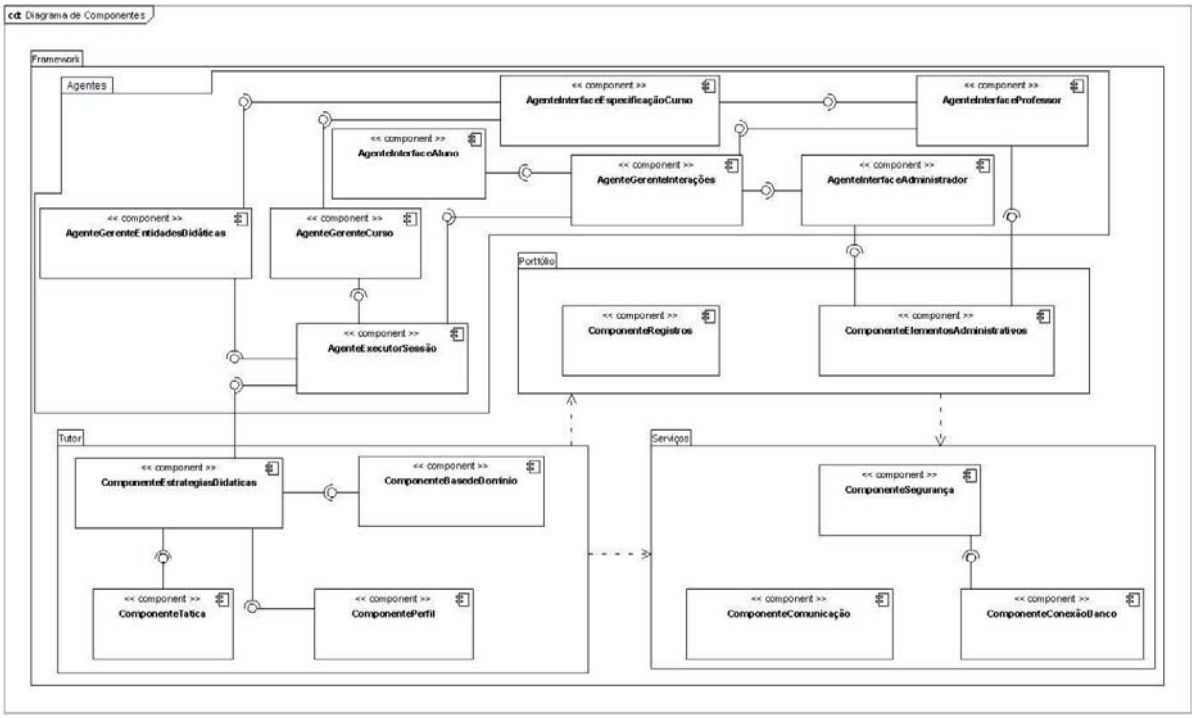


Figura 4.2 - Diagrama de componentes do *framework* FA_PorT

4.5. Especificação dos componentes e suas interfaces

Nesta sessão, são ilustradas e descritas as classes e interfaces referentes ao projeto de cada componente, segundo Medeiros (2006):

- Camada Agentes: os diagramas de classes dos Agentes, que representam os componentes Agente Gerente de Curso, Agente Gerente de Estratégias Didáticas, Agente Perfil de Grupo e Agente Perfil de Aluno, são representados respectivamente em Figura 4.3, Figura 4.4 e Figura 4.5.

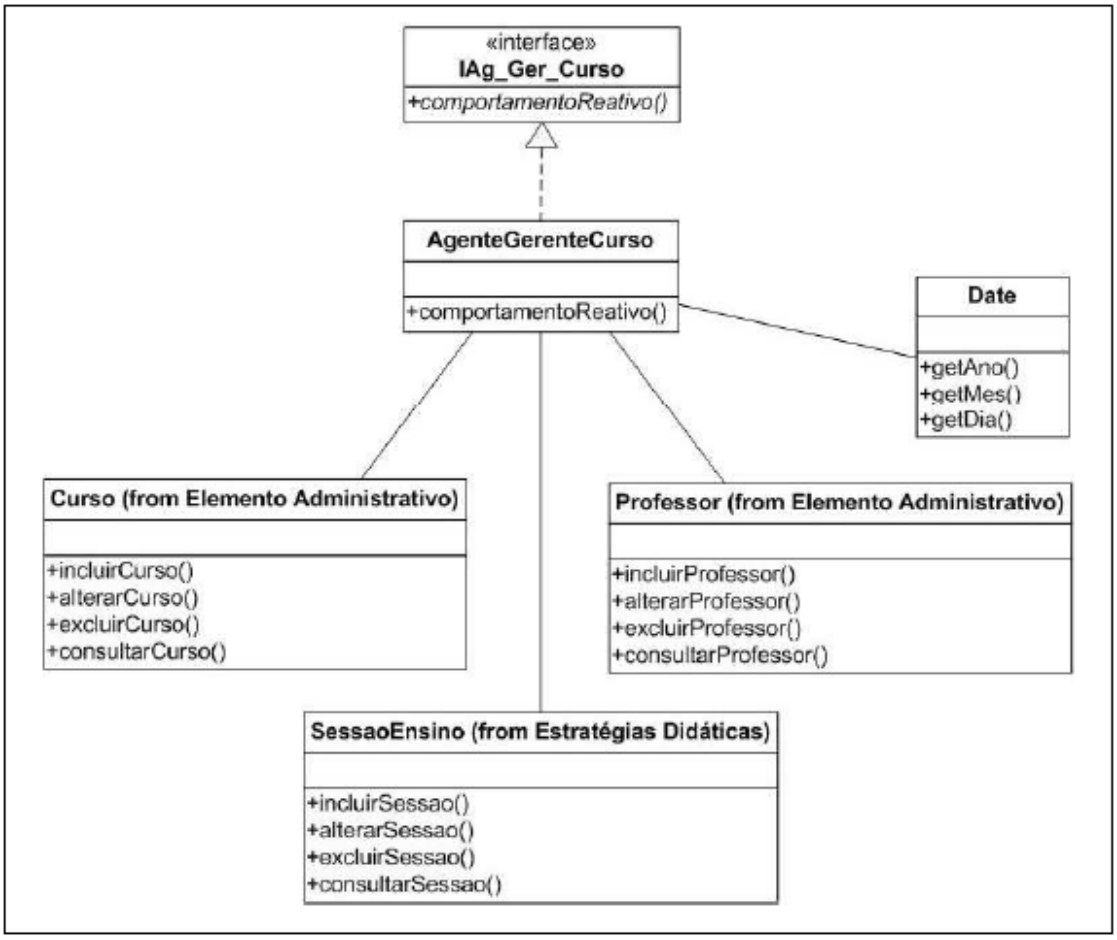


Figura 4.3 - Diagrama de classes do componente de Agente Gerente de Curso

Na Figura 4.3 é mostrado o diagrama de classes do componente agente gerente de curso, onde possui a classe AgenteGerenteCurso que implementa o serviço comportamentoReativo oferecido pela interface IAg_Ger_Curso. A classe AgenteGerenteCurso também utiliza outras classes, como: Date, Professor e Curso (do componente Elemento Administrativo) e SessaoEnsino (do componente Estratégia Didática).

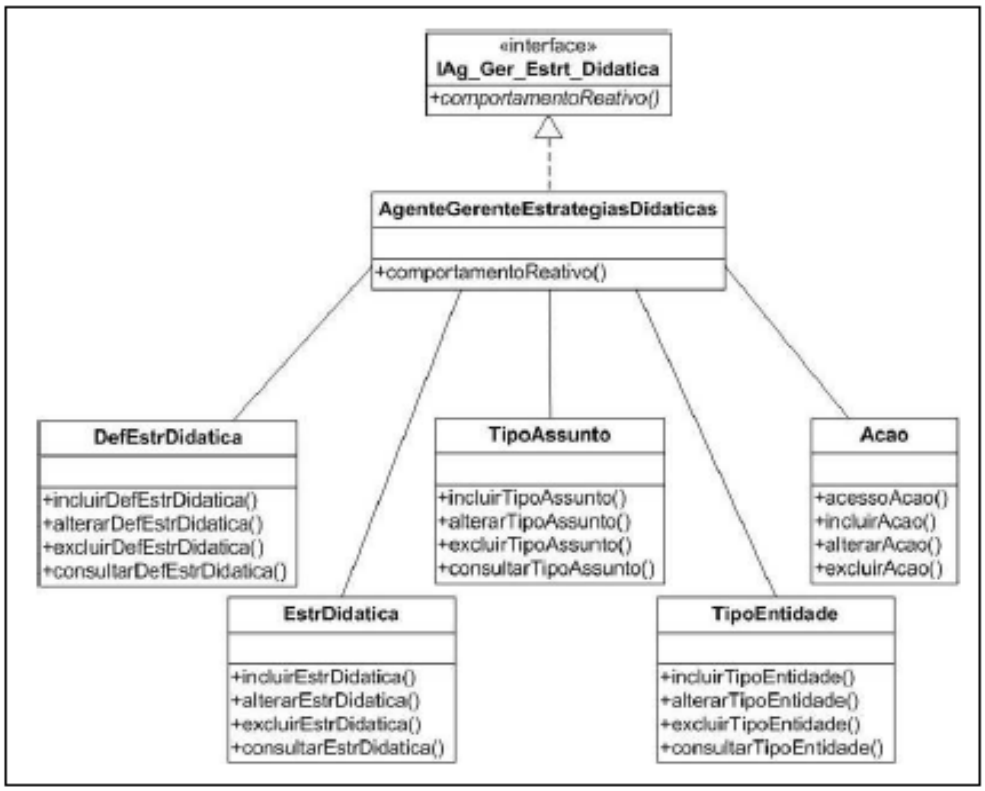


Figura 4.4 - Diagrama de classes do componente de Agente Gerente de Estratégias Didáticas

Na Figura 4.4 mostra-se o diagrama de classes do componente agente gerente de estratégias didáticas, responsável por gerenciar todas as estratégias do sistema, possui como classe principal a AgenteGerenteEstrategiaDidatica que implementa o serviço comportamentoReativo oferecido pela interface IAg_Ger_Estrt_Didatica. A classe principal também utiliza as classes EstrDidatica, TipoAssunto, dentre outras.

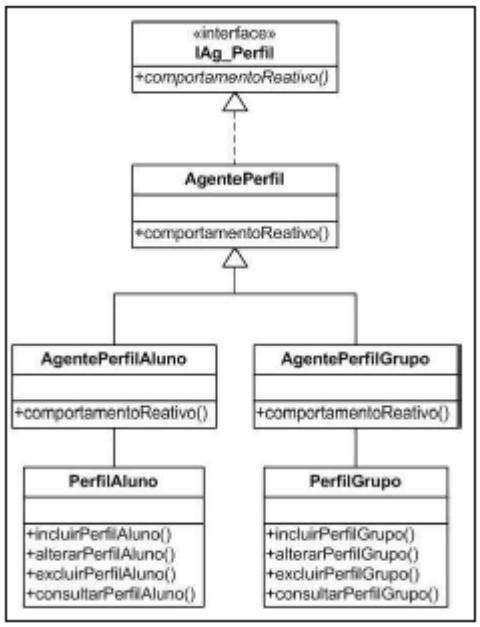


Figura 4.5 - Diagrama de classes do componente de Agente Perfil do Grupo e do Aluno

A Figura 4.5 mostra o diagrama de classes do componente agente perfil do grupo e do aluno, responsável por gerenciar todos os perfis de um sistema Portfólio-Tutor. A classe principal é a AgentePerfil que implementa o serviço comportamentoReativo oferecido pela interface IAg_Perfil. O diagrama também possui as classes AgentePerfilAluno e AgentePerfilGrupo.

- Camada Tutor: possui os componentes Perfil do Aluno e do Grupo (Figura 4.6), o componente Base de Domínio (Figura 4.7), o componente Estratégia Didática (Figura 4.8) e o componente Táticas de Ensino (Figura 4.9).

Figura 4.6 - Diagrama de classes do componente Perfil do Aluno e do Grupo

Na Figura 4.6, mostra-se o diagrama de classes do componente perfil do aluno e do grupo. A classe principal é ComponentePerfil, que relaciona-se com as classes PerfilAluno e Grupo, implementa os serviços da interface IComponentePerfil. A classe Grupo relaciona-se com PerfilGrupo, NivelConhecimento e ZonaComportamento.

Figura 4.7 - Diagrama de classes do componente Base de Domínio

A Figura 4.7 mostra o diagrama de classes do componente base de domínio, a classe principal é ComponenteBaseDominio, que relaciona-se com as classes RecursoDidatico (que possui a classe TipoRecursoDidatico como relacionamento) e Dominio e implementa os serviços da interface IComponenteBasedeDominio.

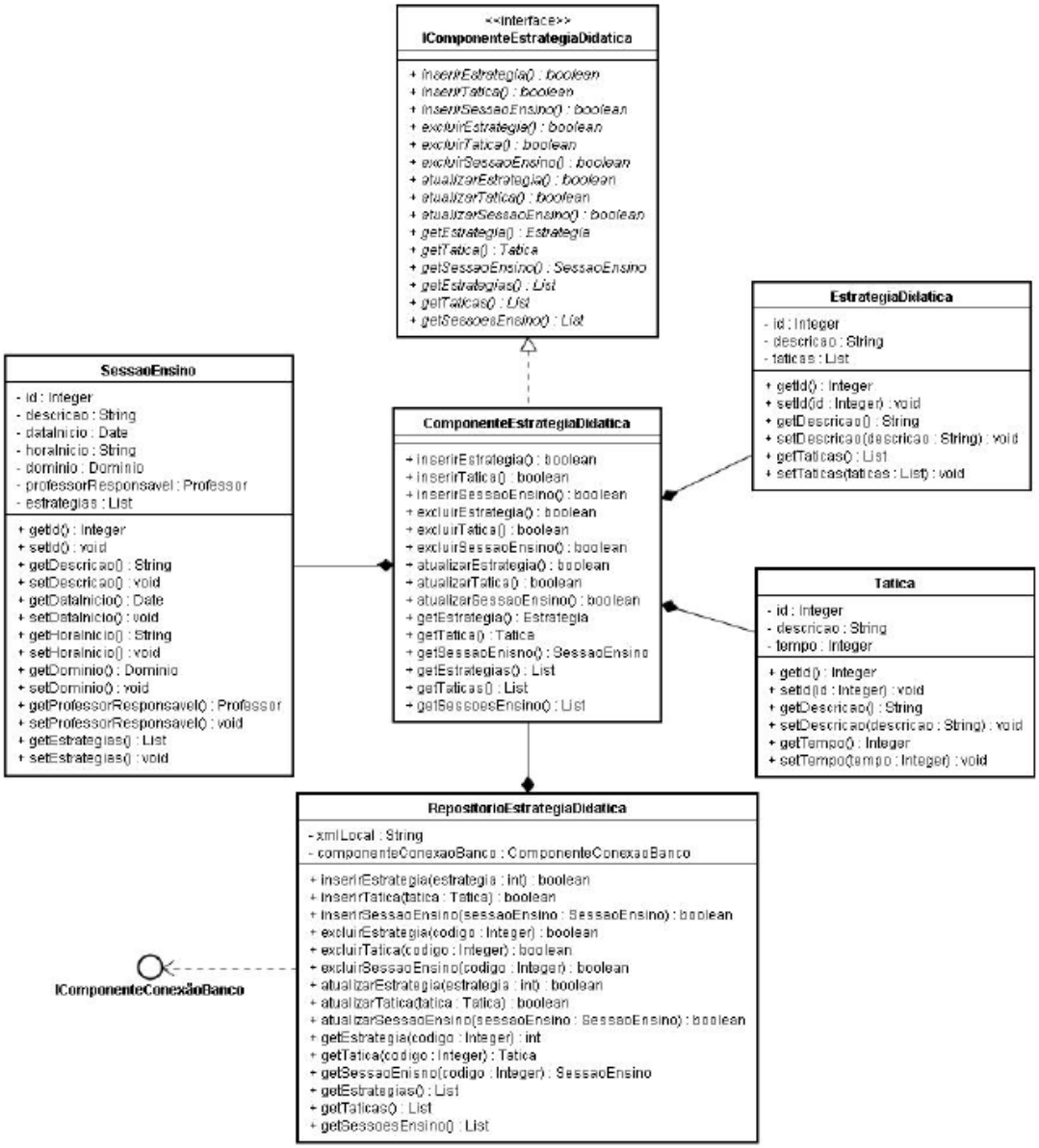


Figura 4.8 - Diagrama de classes do componente Estratégia Didática

Na Figura 4.8, tem-se o diagrama de classes do componente estratégia didática, a principal classe deste componente é ComponenteEstrategiaDidatica, que possui relacionamento com as classes SessaoEnsino, EstrategiaDidatica e Tatica e implementa os serviços da interface IComponenteEstrategiaDidatica.

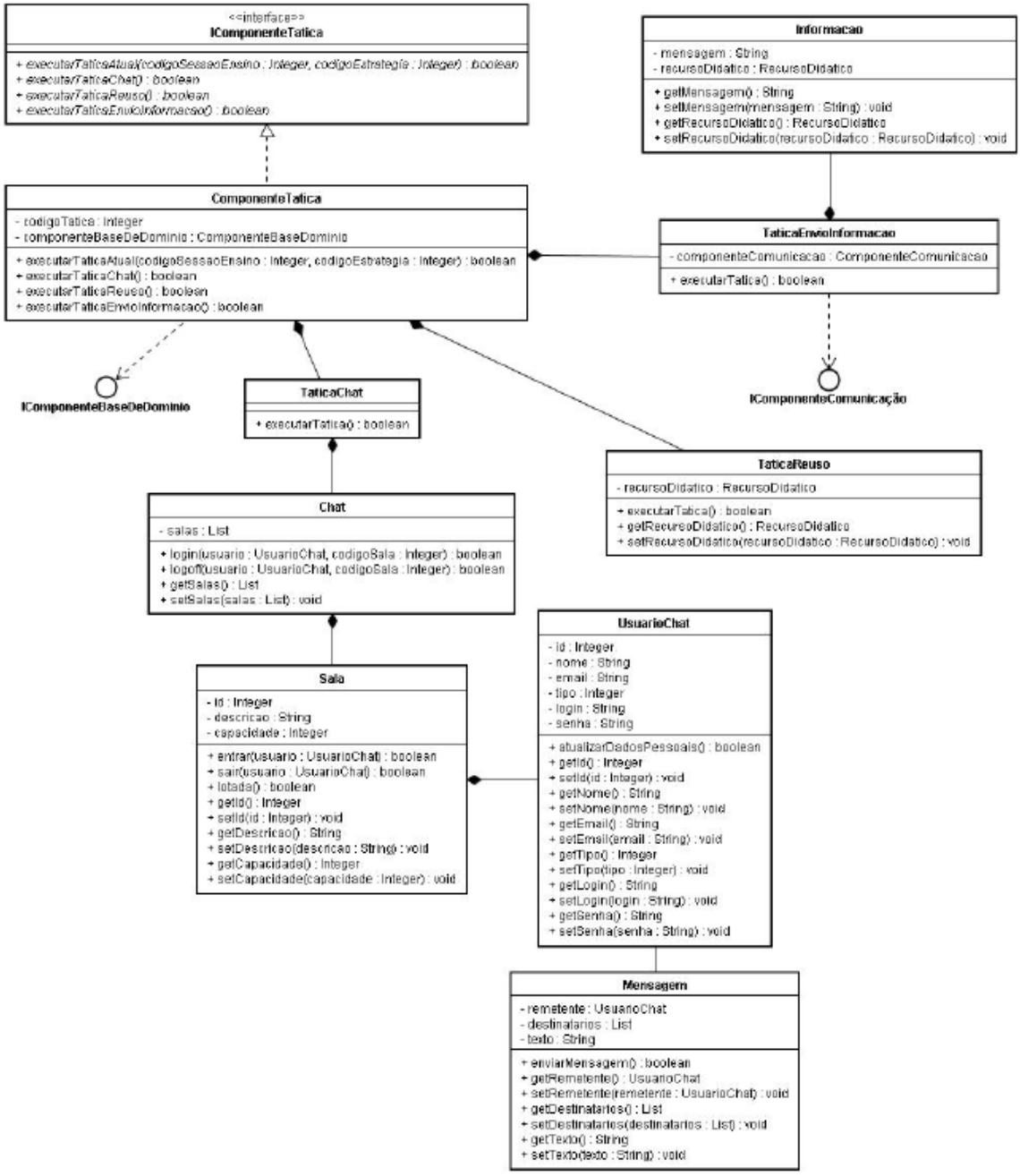


Figura 4.9 - Diagrama de classes do componente Táticas de Ensino

No diagrama de classes do componente táticas de ensino (Figura 4.9), tem-se como classe principal ComponenteTatica que relaciona-se com as classes TaticaReuso, TaticaChat e TaticaEnvioInformacoes, implementa os serviços da interface IComponenteTatica.

Camada Portfólio Eletrônico: possui os componentes Elementos Administrativos mostrado na Figura 4.10 e o Registros, na Figura 4.11.

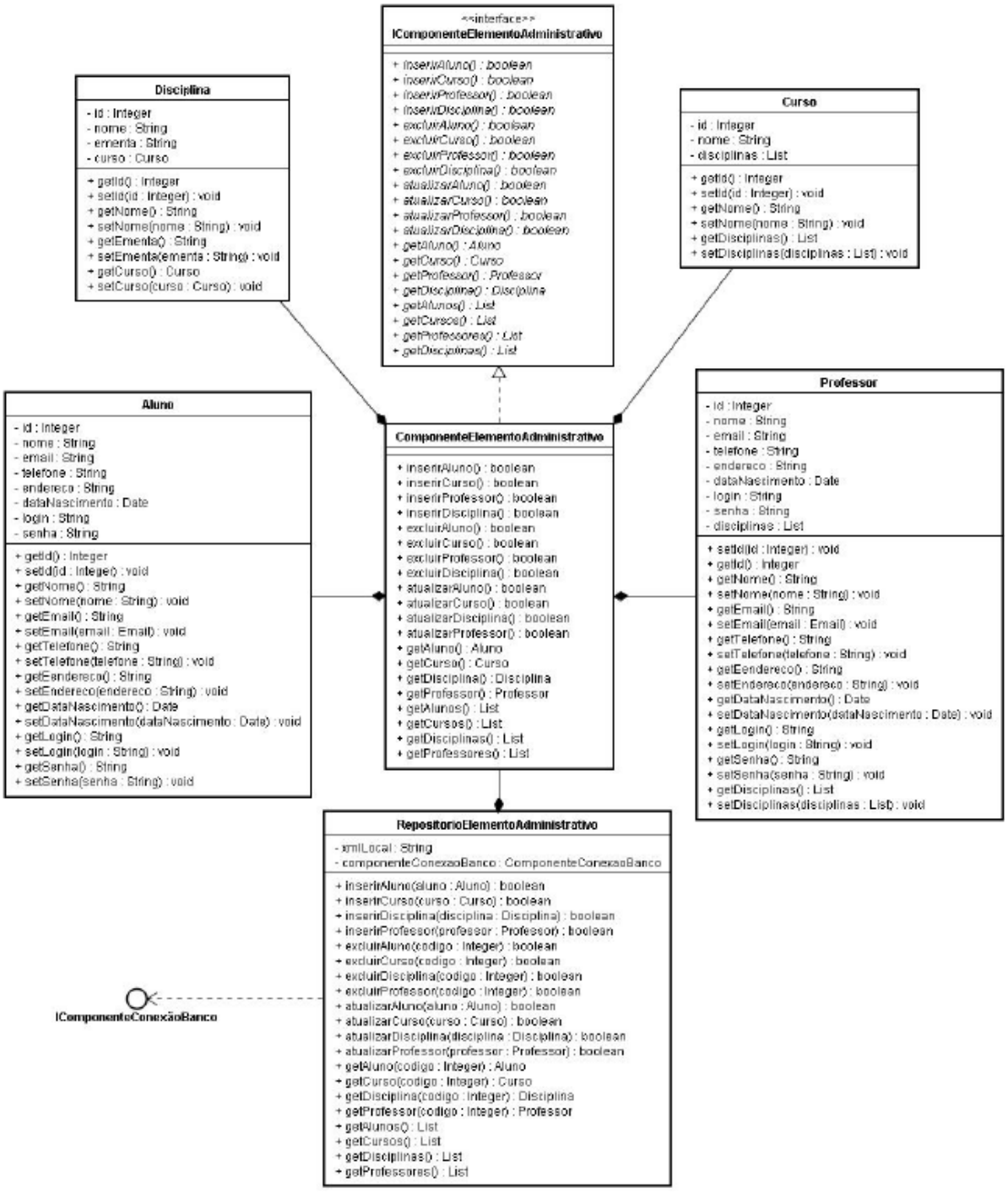


Figura 4.10 - Diagrama de classes do componente Elementos Administrativos

A Figura 4.10 mostra o diagrama de classes do componente Elementos Administrativos, possui como principal classe ComponenteElementoAdministrativo que implementa a interface IComponenteElementoAdministrativo e relaciona-se com as classes

Aluno, Professor, Disciplina e Curso. A classe RepositorioElementoAdministrativo possui relacionamento com a classe principal.

Figura 4.11 - Diagrama de classes do componente Registros

Na Figura 4.11, tem-se o diagrama de classes do componente registros, a principal classe deste componente é ComponenteRegistro, que implementa a interface IComponenteRegistro, e possui relacionamento com as classes Atividade, AtividadeAluno,

DesempenhoAluno e Unidade. A classe RepositorioRegistro possui relacionamento com a classe principal.

- Camada Serviços: a camada de serviços possui os componentes Acesso ao Banco de Dados (Figura 4.12), responsável pela conexão com o banco de dados e o componente Comunicação (Figura 4.13), que possuem os elementos que tratam do envio eletrônico de mensagens, transferência de arquivos e comunicação.

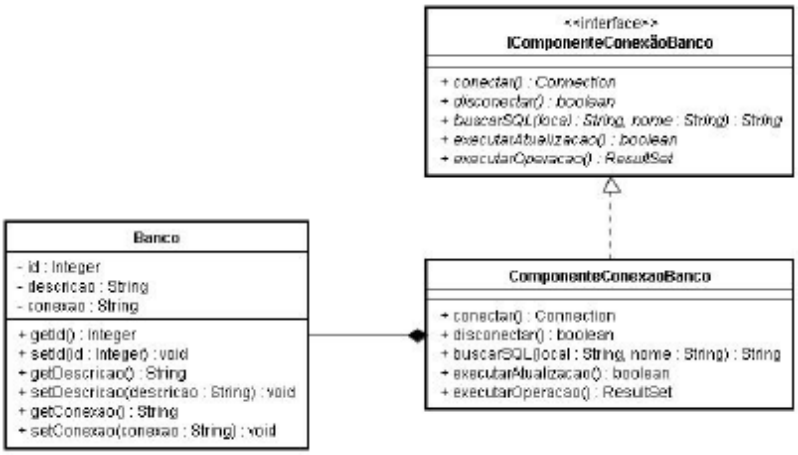


Figura 4.12 - Diagrama de classes do componente Acesso ao Banco de Dados

Na Figura 4.12, tem-se o diagrama de classes do componente acesso ao banco de dados, a classe ComponenteConexaoBanco é a principal classe que possui relacionamento com a classe Banco e implementa os serviços da interface IComponenteConexaoBanco.

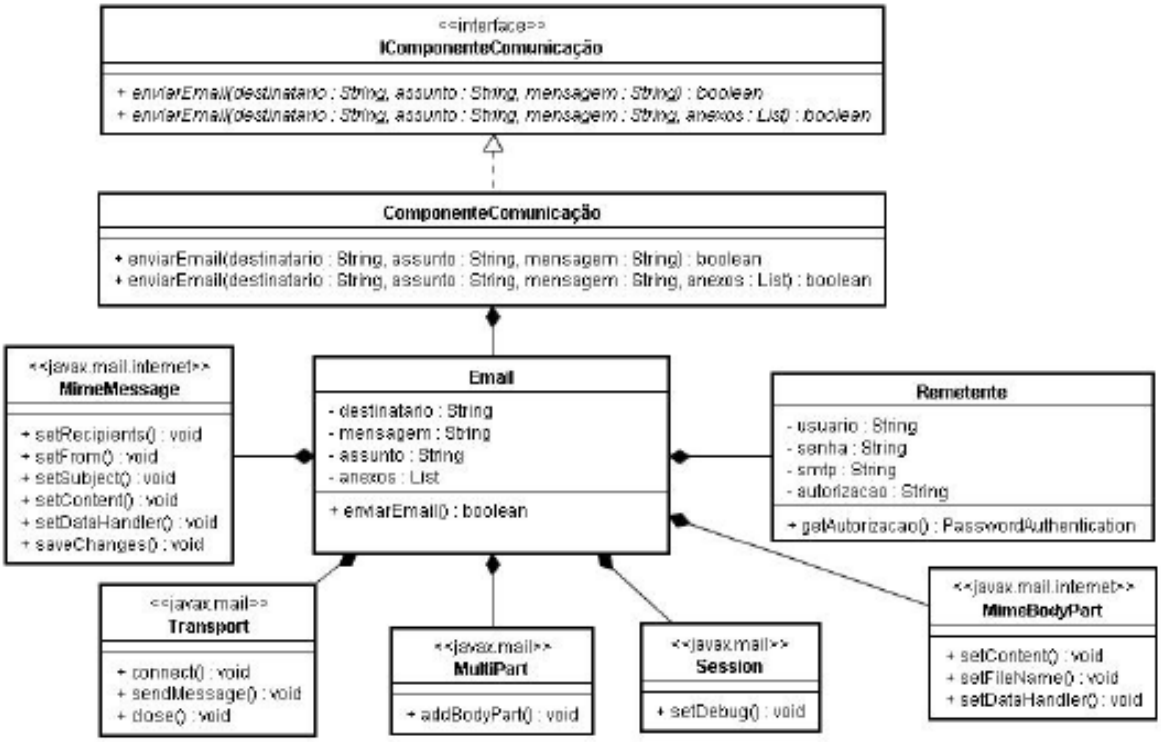


Figura 4.13 - Diagrama de classes do componente Comunicação

A Figura 4.13 mostra o diagrama de classes do componente comunicação, a classe principal deste componente é ComponenteConexao que possui relacionamento com a classe Email (possui como relacionamento as classes Transport, MimeMessage, MultiPart, Session, MimeBodyPart e Remetente) e implementa os serviços da interface IComponente-Comunicacao.

4.6. Sessões de ensino *on-line*

Especificamente na camada Tutor de um novo sistema portfólio-tutor, uma sessão de ensino *on-line* é definida para a aprendizagem de um grupo virtual de alunos e é iniciada quando a camada tutor inicia a execução de uma estratégia. Uma estratégia é representada por um conjunto de táticas, descritas no Quadro 4.3:

Quadro 4.3 - Conjunto de táticas para definição de uma estratégia

Camada	Descrição
Tática de Reutilização de Recurso	indica que será apresentado, durante x unidades de tempo, um recurso didático de ensino (definição, exemplo, exercício, estudo de caso, relatório de projeto etc.).
Tática de Debate Síncrono	representa um "chat" ou "bate-papo", onde os alunos do grupo podem interagir, durante x unidades de tempo, com o professor ou com outros alunos.
Tática de Envio de Informação	envio de informações, pelo sistema portfólio-tutor, para os alunos e professores.
Tática de Mudança de Estratégia	permite mudança da estratégia atual para uma outra. Essa tática possibilita o reuso de estratégias. Isto é importante no contexto do reuso de estratégias bem sucedidas em novas sessões de ensino.
Tática de Relatório	envio de relatório, gerado pelo sistema portfólio-tutor, sobre o desempenho dos alunos para o professor e para os próprios alunos.
Tática de Regra	permite que uma condição seja verificada e algumas ações (táticas) serão realizadas. As regras são criadas da seguinte forma: **Se** (condição) **então** ação. Por Exemplo: *Se (desempenho_aluno i < limite) então* *enviar e-mail com sugestões de leitura* *e ativar migração do tutor móvel*

Fonte: (MEDEIROS, 2006)

A execução de uma estratégia é mostrada na Figura 4.14.

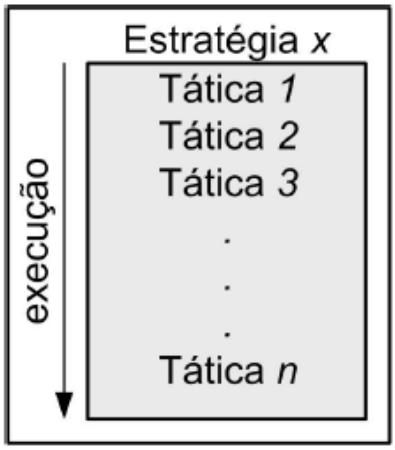


Figura 4.14 - Execução de uma Estratégia

4.7. Conclusão

Neste capítulo, apresentaram-se os requisitos, arquitetura, diagrama de componentes e as respectivas interações, especificação dos componentes e interfaces e detalhes sobre as sessões de ensino *on-line*. Tais informações, mostram como o *framework* pode ser utilizado para instanciar aplicações no contexto de ensino on-line. A análise e estudo detalhado da arquitetura do FA_PorT foi fundamental para o projeto e implementação da nova funcionalidade a ser adicionada ao *framework*, que é um tutor *off-line* baseado em agentes móveis, proposta do atual trabalho.

No próximo capítulo, será descrita a modelagem do sistema tutor móvel que está sendo proposto nesta dissertação.

5. Modelagem do sistema tutor móvel proposto

5.1. Introdução

Conforme já mencionado, o objetivo da criação de um Tutor Móvel acoplado ao FA_PorT está relacionado à necessidade de um tratamento diferenciado/personalizado aos alunos com desempenho não satisfatório em uma sessão de ensino *on-line*. Assim, a ideia é a possibilidade de criarem-se sessões personalizadas de reforço a esses alunos. Dessa forma, foi proposto um Tutor Móvel possibilitando a execução de uma sessão de ensino *off-line* na máquina do aluno. Um sistema Tutor Móvel é migrado, através do uso de um agente móvel, para o computador do aluno, proporcionando sessões de ensino sem a necessidade de manter uma conexão com o sistema Tutor *On-line*[1]. Uma vez concluída a tarefa do tutor móvel, o agente móvel voltará ao computador de origem, conforme pode ser observado na figura abaixo.

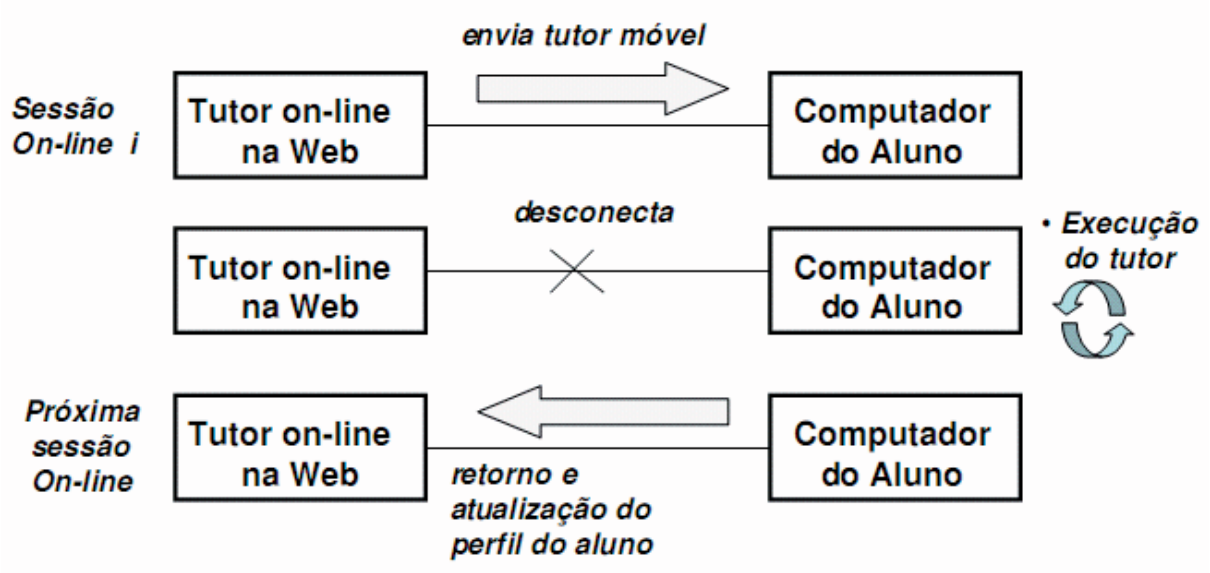


Figura 5.1 - Tutor móvel acionado em uma sessão de ensino *on-line*

[1] Neste trabalho, Tutor *On-line* representa um sistema, e não, uma pessoa que possui o papel de tutor.

Uma subcamada intermediária foi adicionada ao Tutor *on-line* do FA_PorT, dando suporte a um agente móvel relacionado ao Tutor Móvel (Figura 5.2).

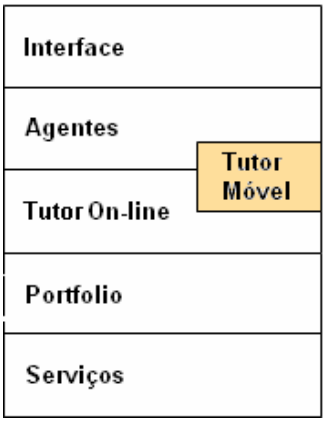


Figura 5.2 - Integração de um Tutor móvel no contexto de sessões *on-line* do FA_PorT.

Dessa forma, os alunos que têm dificuldade em uma sessão *on-line* A serão auxiliados pelo tutor móvel até que a próxima sessão *on-line* A+1 ocorra. O Tutor móvel (Figura 5.3) será executado na máquina do aluno e fornecerá uma aula para reforçar o conteúdo exposto na sessão *on-line*, visando uma melhoria no desempenho desse aluno. Assim, o aluno fica livre para executar sua aula no momento e local que apreciar. Reforça-se que o recomendado é que a sessão *off-line* seja finalizada antes do inicio da próxima aula *on-line*.

Figura 5.3 - Tutor móvel e algumas funcionalidades.

A seguir, é ilustrado e descrito um cenário da integração de sessões de ensino *off-line* no contexto de sessões *on-line* através do FA_PorT.

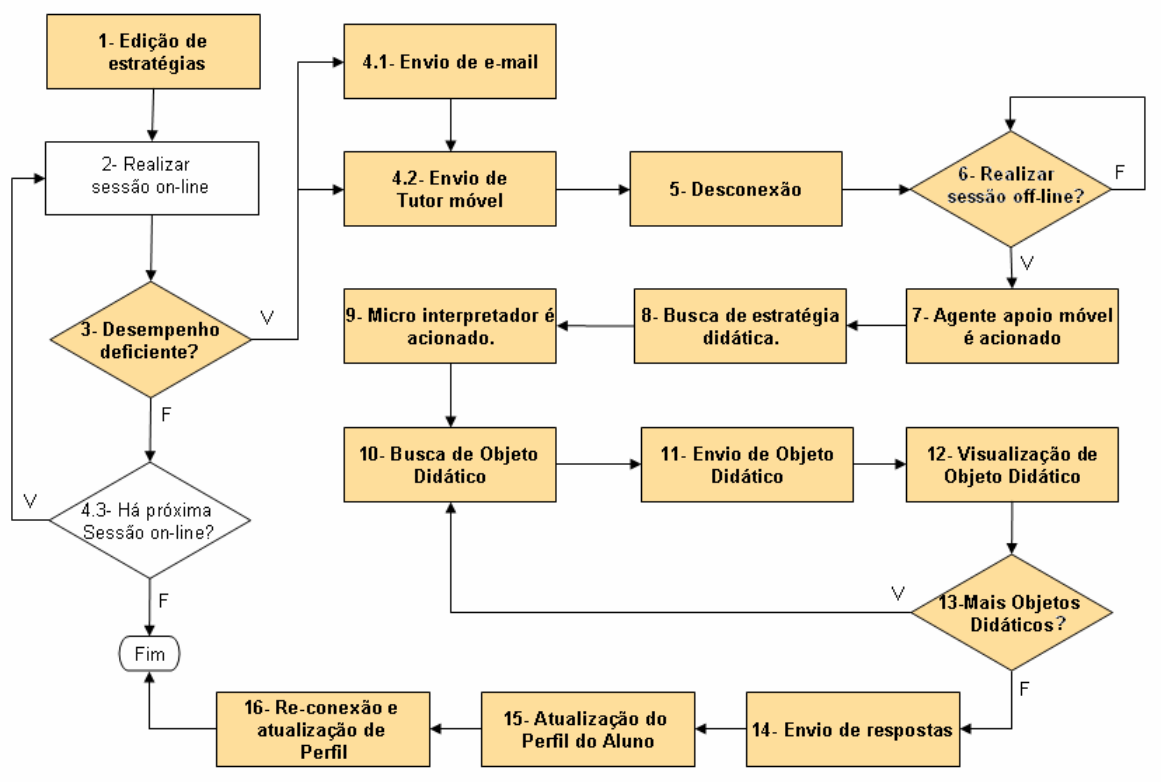


Figura 5.4 - Processos envolvidos em uma sessão de ensino *off-line*

1. O professor edita as estratégias que deverão ser aplicadas aos alunos nas sessões *on-line* e aos que tiverem alguma deficiência em algum conceito apresentado no curso.
2. Inicia-se a sessão *on-line*.
3. Há aluno com deficiência na sessão *on-line*?

4.1 O tutor envia um *e-mail* dando algumas sugestões de leitura.

4.2 Envio do agente tutor móvel para executar uma aula de reforço *off-line* (Sessão *off-line*), juntamente com as estratégias e Objetos didáticos necessários.

4.3 Iniciar próxima sessão *on-line*?

5. Tutor e aluno são desconectados.
6. A qualquer momento e lugar, o aluno pode iniciar a sessão *off-line*.
7. O aluno aciona o agente tutor móvel no seu computador e inicia-se a sessão *off-line*.
8. O tutor móvel busca a estratégia associada ao curso na máquina do aluno.

9. O tutor móvel aciona um ambiente pré-instalado na máquina do aluno, denominado micro-tutor, para interpretar a estratégia.
10. O tutor móvel busca um Objeto Didático na máquina do aluno.
11. O micro tutor envia o Objeto Didático para a interface do aluno.
12. O aluno visualiza o recurso didático.
13. O tutor móvel verifica se há outros Objetos Didáticos relacionados à estratégia criada pelo professor.
14. O micro tutor envia a resposta dos alunos para o Controlador de Comportamento.
15. O Controlador de Comportamento analisa as respostas e atualiza o Perfil do aluno, que é e enviado para o micro tutor.
16. O tutor móvel, na próxima sessão *on-line*, estabelecerá uma conexão com o sistema e atualiza o perfil do aluno no Banco.

Nas próximas sessões, serão abordados assuntos relacionados aos requisitos, o componente tutor móvel proposto, bem como, das alterações necessárias para adaptá-lo ao framework FA_PorT.

5.2. Requisitos funcionais

A seguir, serão descritos os requisitos funcionais do Tutor *On-line* e Tutor Móvel do framework FA_PorT. Dentre os requisitos, está a criação da nova tática denominada Assistência Personalizada *Off-line*.

5.2.1. Componentes da camada Tutor Móvel

Nesta sessão, serão descritos os requisitos funcionais para os componentes da camada Tutor Móvel.

Componente Estratégia *Off-line*:

a) Manter Tática de Ensino Personalizado: Cadastrar, remover e atualizar uma tática de assistência personalizada *off-line.*

b) Manter Estratégias *Off-line*: Cadastrar, remover e atualizar estratégias *off-line*, que são formadas a partir de táticas de assistência personalizada *off-line.*

c) **Consultar Desempenho dos Alunos:** Consultar o desempenho dos alunos com o intuito de enviar, ou não, o Tutor Móvel.

d) **Mapear Estratégia *Off-line*:** Possibilitar o mapeamento da estratégia *off-line* para o formato XML. Assim, a estratégia poderá ser interpretada na máquina do aluno.

Componente Sessão de Ensino *Off-line*:

a) **Manter Sessões de Ensino *Off-line*:** Cadastrar, remover e atualizar sessões de ensino *off-line*.

b) **Enviar Tutor Móvel:** Possibilitar o envio do Tutor Móvel para o computador do aluno através de um agente móvel. Para envio do Tutor Móvel são fornecidos: *id*, *login*, senha criptografada, curso, estratégia *off-line*, *ip* e porta do computador do aluno.

c) **Acionar Micro-Tutor:** Executar e interpretar a estratégia *off-line* no computador do aluno.

d) **Buscar Estratégia:** Buscar a estratégia entregue pelo Tutor Móvel para que este interprete e possibilite a execução da sessão *off-line*.

e) **Manipular Objeto Didático:** O componente fornece em sua interface a operações de buscar, exibir e encerrar os objetos didáticos necessários durante uma sessão de ensino *off-line.*

f) **Receber Tutor Móvel:** O componente possibilita o recebimento do Tutor Móvel devolvido pelo computador do aluno. Para recebimento do Tutor Móvel são fornecidos: *id*, *login*, perfil atualizado, *ip* e porta do computador do servidor.

Componente Agente Móvel:

a) **Manipulação na agência:** Criar, destruir, suspender e ativar agente móvel em uma agência(host origem).

b) **Migrar agente:** Despachar o agente móvel para o computador do aluno e retorno do mesmo agente móvel.

c) **Serializar/deserializar agente:** Serializar o agente para possibilitar migração do mesmo e deserializá-lo após a chegada no destino para ter acesso às funcionalidades e recursos didáticos.

d) **Criptografar/decriptografar agente:** Possibilitar criptografia antes da migração e decriptografia quanto o agente móvel chegar ao computador do aluno.

Componente Tutor Móvel:

a) **Atualizar Perfil do Aluno:** O componente possibilita a atualização do perfil do aluno após a execução de uma sessão de ensino *off-line*. Para a atualização do perfil do aluno, são fornecidos: *login*, perfil.

b) **Manter Domínio:** Cadastrar, remover e atualizar domínios dos assuntos a serem passados aos alunos. Domínios são formados por módulos conceituais.

c) **Manter Módulo:** Cadastrar, remover e atualizar módulos conceituais.

d) **Manter Recursos Didáticos:** Cadastrar, remover e atualizar recursos didáticos a serem utilizados pelo aluno de forma *off-line*.

e) **Consultar Perfil Individual**: Consultar perfis individuais dos alunos no momento em que o tutor móvel retorna do computador do aluno.

5.2.2. Componentes da camada Portfólio

Na atual sessão, serão delineados os requisitos funcionais para os componentes da camada Portfólio.

Componente Elemento Administrativo:

a) **Manter Alunos:** Cadastrar, remover e atualizar alunos que irão interagir com os tutores *on-line* e *off-line*.

b) **Manter Professores**: Cadastrar, remover e atualizar professores que irão mediar as sessões de ensino *on-line* e *off-line*.

c) **Manter Disciplinas:** Cadastrar, remover e atualizar disciplinas a serem utilizadas nas sessões de ensino *on-line* e *off-line*.

d) **Manter Cursos:** Cadastrar, remover e atualizar cursos associados às disciplinas de uma sessão de ensino *on-line* ou *off-line*.

Componente Registro:

a) **Manter Atividade:** Cadastrar, remover e atualizar atividades disponibilizadas aos alunos.

b) **Manter Itens de Avaliação**: Cadastrar, remover e atualizar itens de avaliação. Itens de Avaliação são os itens que vão ser levados em consideração para a avaliação das atividades.

c) **Manter Unidade:** Cadastrar, remover e atualizar unidades.
d) **Atribuir e Consultar Atividades para os Alunos**: Atribuir e consultar as atividades de um determinado aluno ou grupo de alunos.
e) **Consultar Desempenhos dos Alunos**: Consultar os desempenhos dos alunos.

5.2.3. Componentes da Camada Tutor On-line

Agora, serão detalhados os requisitos funcionais para os componentes da camada Tutor *On-line*. Os assuntos a serem passados aos alunos estão distribuídos em vários domínios diferentes. Cada domínio por sua vez está organizado em módulos e cada módulo contém vários conceitos, onde cada conceito é apresentado aos alunos através de recursos didáticos, como, por exemplo: definições, exemplo etc..

Componente Base de Domínio:

a) **Manter Domínio:** Cadastrar, remover e atualizar domínios de conhecimento.
b) **Manter Módulo:** Cadastrar, remover e atualizar módulos conceituais.
c) **Manter Recursos Didáticos:** Cadastrar, remover e atualizar recursos didáticos a serem apresentados ao aluno durante a sessão de ensino *on-line*.

Componente Perfil:

a) **Consultar Perfil Individual**: Consultar perfis individuais dos alunos.
b) **Consultar Perfil do Grupo**: Consultar perfis de grupos de alunos.
c) **Manter Nível de Conhecimento:** Cadastrar, remover e atualizar nível de conhecimento de um grupo de alunos. O nível de conhecimento pode ser: básico, intermediário ou avançado.
d) **Manter Zona de Comportamento:** Cadastrar, remover e atualizar zona de comportamento que classifica os alunos dentro de um mesmo grupo. As zonas de comportamento podem ser: inferior, normal ou superior.

Componente Estratégia Didática/Tática de Ensino:

a) **Manter Tática de Ensino:** Cadastrar, remover e atualizar táticas, que podem ser: chat, recurso, envio de informações etc..
b) **Manter Estratégias:** Cadastrar, remover e atualizar estratégias, que são compostas por táticas.

c) **Manter Sessões de Ensino *on-line*:** Cadastrar, remover e atualizar sessões de ensino *on-line*. As sessões de ensino *on-line* são construídas a partir de uma ou mais estratégias, podendo ser reutilizadas em várias sessões de ensino.

5.2.4. Componentes da Camada Serviços

Nesta sessão, serão descritos cinco requisitos funcionais referentes à camada Serviços. Tais requisitos são relacionados aos componentes **Interpretador, Segurança, Comunicação, Conexão com Banco de Dados, Geração de Gráficos** e **Geração de Relatório**, respectivamente.

Componente Interpretador:

Interpretar estratégia *off-line*: Interpretar uma estratégia em XML no computador do aluno. Ao interpretar uma estratégia, possibilita-se que os recursos didáticos sejam disponibilizados ao aluno.

Componente Segurança:

Verificar se o usuário está autenticado: Verificar se um determinado usuário está autenticado e se tem permissão para visualizar uma determinada tela do sistema. Caso o usuário esteja autenticado e tenha permissão à operação desejada, ele pode realizar a ação.

Componente Comunicação:

Envio de *e-mail*: Enviar *e-mail* com, ou sem, anexo. O envio de *e-mail* é utilizado para envio de informações a respeito de atividades, sessões de ensino on-line e envio de informações e instruções sobre a sessão de ensino *off-line*.

Componente Conexão com Banco de Dados:

Conectar ao banco de dados: Comunicação com o banco de dados. Através da conexão com o banco de dados é possível realizar consultas, inserções, atualizações e exclusões.

Componente Geração de Gráficos:

Gerar gráficos: Criar gráficos de barras e gráficos de pizza. São passadas as informações a respeito dos alunos e o componente gera gráficos para facilitar a análise das informações.

Componente Geração de Relatório:

Gerar relatório: Criar relatórios no formato PDF. É passado para o componente um arquivo "JASPER" com as informações do relatório e o local e nome do arquivo PDF a ser criado.

5.3. Requisitos não funcionais

A seguir, serão descritos os requisitos não funcionais relativos ao Sistema Tutor proposto no atual trabalho:

a) **FA_PorT**: O Sistema Tutor Móvel deve ser implementado como funcionalidade adicional do *framework* FA_PorT.
b) **Agentes Móveis**: O sistema proposto deve ser implementado utilizando-se a tecnologia de Agentes Móveis.
c) **Reuso:** O desenvolvimento deve ser baseado no reuso de software (componentes).

5.4. Diagrama de Casos de Uso

O diagrama de caso de uso para uma aplicação instanciada pelo *framework* será ilustrado a seguir, mostrando os casos de uso para os diferentes tipos de atores que interagem com a aplicação instanciada, no que se refere ao sistema Tutor Móvel.

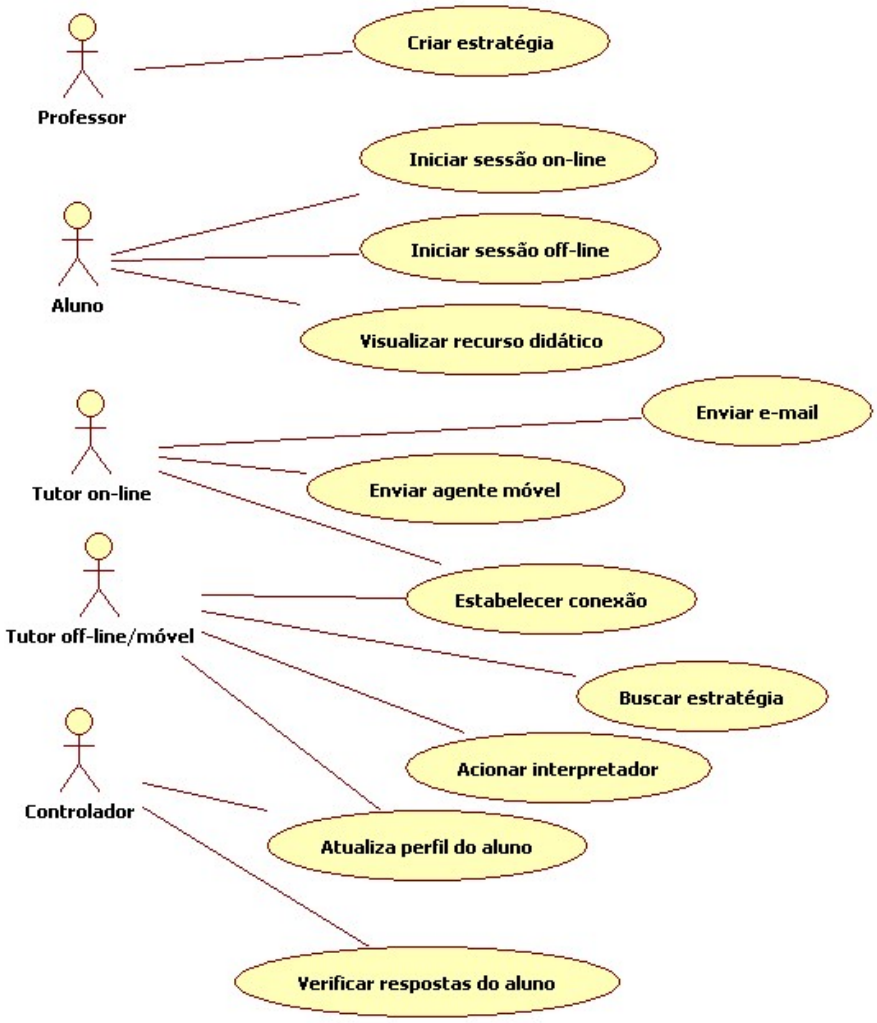


Figura 5.5 - Diagrama de casos de uso no contexto do Tutor Móvel

5.5. Identificação dos Atores

De acordo com o levantamento das funcionalidades que constituirão o sistema, são identificados os atores envolvidos por estas funcionalidades. Como os atores são representados pelo papel que desempenham, foram verificados os seguintes papéis:

- **Professor -** Corresponde a qualquer professor. É responsável pelo planejamento didático, atribuição de atividades, avaliação de atividades, criação de estratégias e criação e participação de sessões de ensino;
- **Aluno -** Corresponde a qualquer aluno que participa na realização das sessões de ensino. É responsável pela execução das atividades atribuídas, acompanhar o desenvolvimento de suas habilidades e, assim, assumir uma atitude mais ativa em

relação ao seu aprendizado. Este também pode acionar a execução de uma sessão de ensino *off-line*, caso haja dificuldade no aprendizado durante uma sessão de ensino *on-line*;

- **Tutor *off-line*/móvel** – Corresponde ao agente móvel migrado ao computador do aluno para executar a tarefa de estabelecer uma sessão de ensino *off-line*;
- **Tutor *on-line*** - Corresponde à camada que gerencia uma sessão de ensino *on-line*.
- **Controlador** - Corresponde à camada que gerencia uma sessão de ensino *off-line*.

Os Casos de Uso referentes aos tutores *on-line* e *off-line* estão especificados nos Apêndices desta dissertação.

5.6. Arquitetura em camadas dos Tutores *On-line* e Móvel do FA_PorT

Nesta sessão, apresenta-se a arquitetura do sistema em camadas ilustrada no diagrama de pacotes (Figura 5.6). Na camada Interface, que se preocupa em oferecer uma interface ao usuário, encontra-se o componente Layout, responsável pelo gerenciamento das interações entre os alunos, professores e o sistema. Este componente usufrui dos serviços oferecidos pelos componentes da camada Agentes. A camada Agentes, que é representada por um conjunto de agentes: AgenteGerenteCurso, AgentePerfilGrupo, AgentePerfilAluno e AgenteGerenteEstratDidatica, utiliza serviços disponibilizados pelos componentes das camadas TutorMovel e Tutor. A camada Tutor representa um sistema tutor que gerencia uma sessão de ensino *on-line*, e é nesta camada que estão os componentes PerfilAlunoGrupo, BaseDeDominio, EstrategiaDidatica e TaticasDeEnsino. Na camada TutorMovel, há os componentes TutorMovel, SessaoDeEnsinoOffline, EstrategiaOffline e AgenteMovel, que representam o sistema tutor que gere uma sessão de ensino *off-line* enviada ao aluno através de um agente móvel. Os componentes da camada Tutor e TutorMovel acessam as interfaces de componentes da camada Serviços, representada por um conjunto de serviços sobre acesso ao banco de dados, segurança, interpretador de estratégia em XML, comunicação e geração de relatórios.

Figura 5.6 - Arquitetura em camadas dos tutores online e móvel do FA_PorT

5.6.1. Especificação dos componentes

Na especificação dos componentes relacionados ao Tutor Móvel, serão utilizados os diagramas de classes da UML. Dessa forma, serão especificados os componentes TutorMovel e Interpretador.

Na Figura 5.7, é ilustrado o diagrama de classes do componente TutorMovel; que é formado pela classe TutorMovel e implementa as operações da interface InterfaceTutorMovel. Essas operações são: criaAgencia(), criaTutorMovel(), iniciaEstrategia(), destrio(), migra(), executaEstrategia(), addResultados() e cicloDeVida().

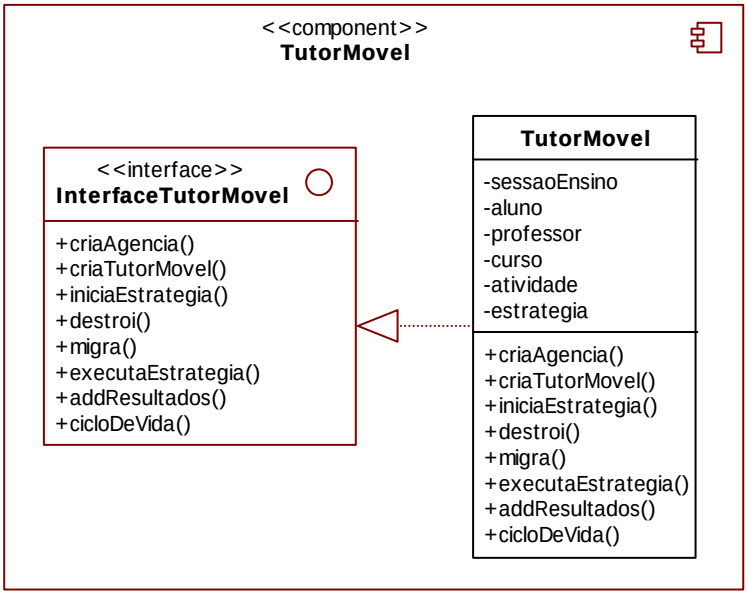


Figura 5.7 - Diagrama de classes para o componentes Tutor Móvel

Já o diagrama de classes do componente Interpretador é mostrado na Figura 5.8. O componente é constituído pela classe Interpretador, que implementa as operações da interface InterfaceInterpretador. As operações implementadas são: interpretaEstrategia(), mostraRecursosDidaticos(), getEstrategia() e parsingXML(). A Figura 5.9 ilustra o pseudo-código com o funcionamento da operação interpretaEstrategia(), que tem o objetivo de interpretar a(s) Estratégia(s) enviada(s) ao aluno, executando, assim, uma ação específica para cada tipo de Tática.

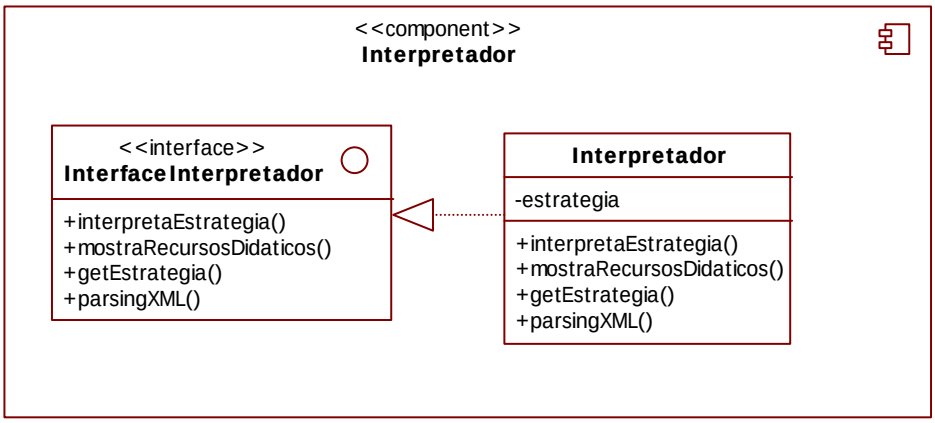


Figura 5.8 - Diagrama de classes para o componentes Interpretador

```
1   INTERPRETAR_ESTRATÉGIA (Estratégia)
2      LER (Estratégia)
3      LER (primeira Tática)
4      REPITA
5         IDENTIFICAR (tipo da Tática)
6            CASO (Tática de Reutilização de Recurso):
7               MOSTRAR (Recurso didático)
8            CASO (Tática de Mudança de Estratégia):
9               BUSCAR (nova Estratégia)
16              INTERPRETAR_ESTRATÉGIA (nova Estratégia) //Recursivo
17           CASO (Tática de Regra):
18              SE (condição) ENTÃO
19                 BUSCAR (Determinada Tática)
20                 EXECUTAR (Ação específica para tipo de Tática)
21              FIM-SE
22           CASO (Tática de Relatório):
23              MOSTRAR (Desempenho do aluno)
24           CASO (Tática de Chat):
25              INICIAR_CHAT (Aluno(s) e/ou Professor)
26        FIM-IDENTIFICAR
27     ATÉ QUE (Não haja mais Tática na Estratégia)
28  FIM_INTERPRETAR_ESTRATÉGIA
```

Figura 5.9 - Pseudo-código com o funcionamento da operação interpretaEstrategia()

5.6.2. Integração do Tutor Móvel ao FA_PorT

O diagrama que representa a integração entre os componente Tutor Móvel e outros componentes do *framework* FA_PorT: ComponentesEstrategiaDidatica, Registro, Comunicacao, ElementoAdministrativo e ConexaoBD é ilustrado na Figura 5.10. Para facilitar a visualização do diagrama, as operações das interfaces foram retiradas.

O Componente Tutor Móvel usa alguns serviços disponibilizados pelo Componente Estratégia Didática em funcionalidades inerentes a sessões de ensino, estratégias e táticas. O componente Registro oferece os serviços que o Componente Tutor Móvel necessitará para consultas e atualizações de atividades e desempenho dos alunos nas sessões de ensino. Para acessar serviços relacionados a envio de e-mail aos alunos que necessitarão iniciar uma sessão de ensino off-line, o Componente Tutor Móvel comunica-se com o Componente Comunicação. Já o Componente Elemento Administrativo oferecerá serviços ao Componente Tutor Móvel relacionados à administração de alunos, professores, disciplinas e cursos. Por fim, o Componente Tutor Móvel comunicar-se-á com o Componente Conexão BD para acessar serviços relacionados à conexão, desconexão e execução de *queries* em banco de dados relacional

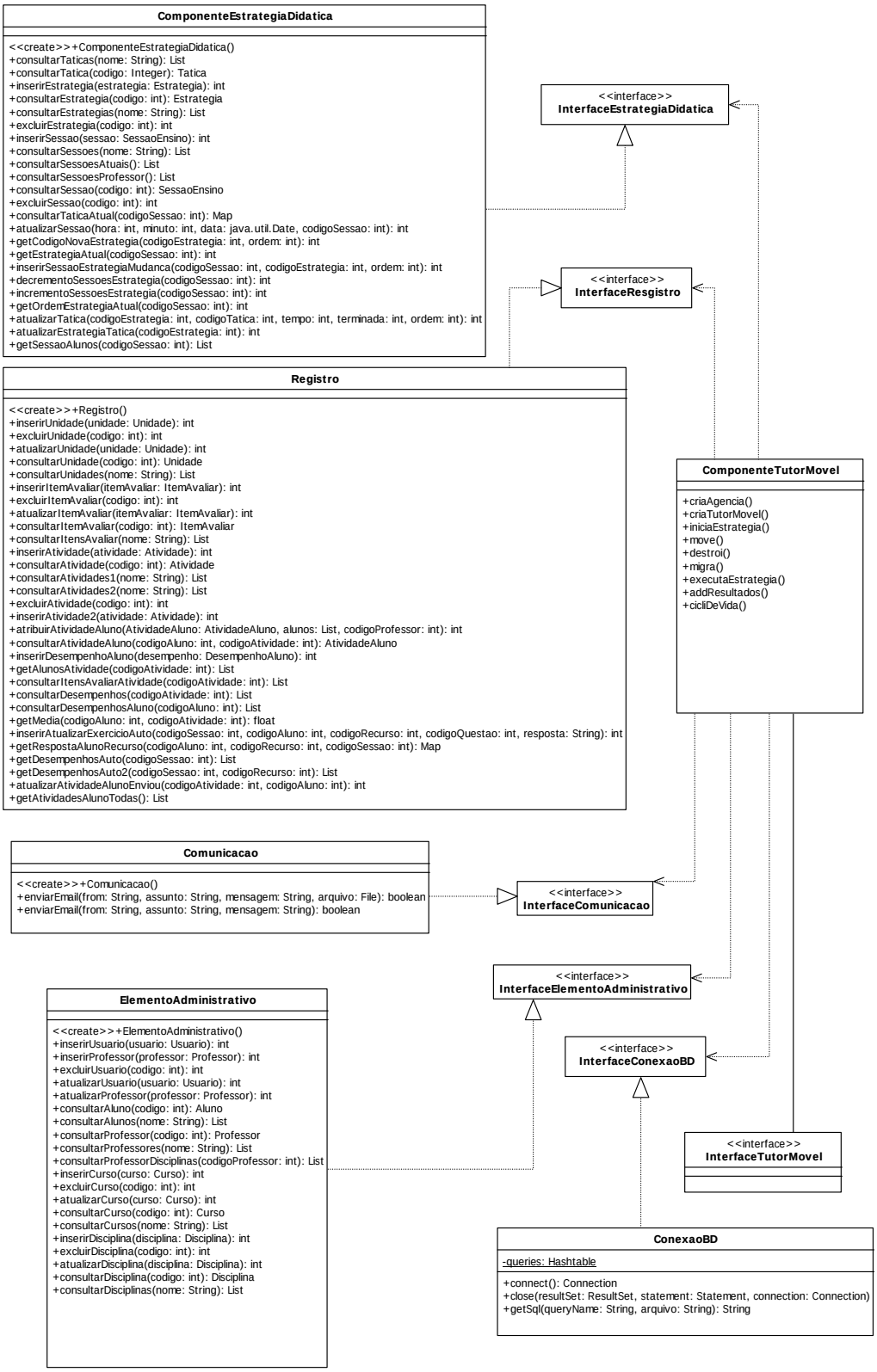


Figura 5.10 - Diagrama de componentes integrando o Tutor Móvel proposto ao FA_PorT

5.6.3. Diagrama de sequência

A especificação do tutor móvel foi baseada no padrão de projeto *Master-Slave* para agentes móveis (ARIDOR; LANGE, 1998). Basicamente, no funcionamento deste padrão, um agente mestre cria um escravo, que migra para execução remota de uma tarefa. Após a execução da tarefa, o agente escravo retorna para a origem, onde entrega os resultados para o mestre.

Com base no padrão *Master-Slave* para agentes móveis, especificou-se o diagrama de sequência relacionado à sessão de ensino *off-line* baseada em agentes móveis, vide Figura 5.11.

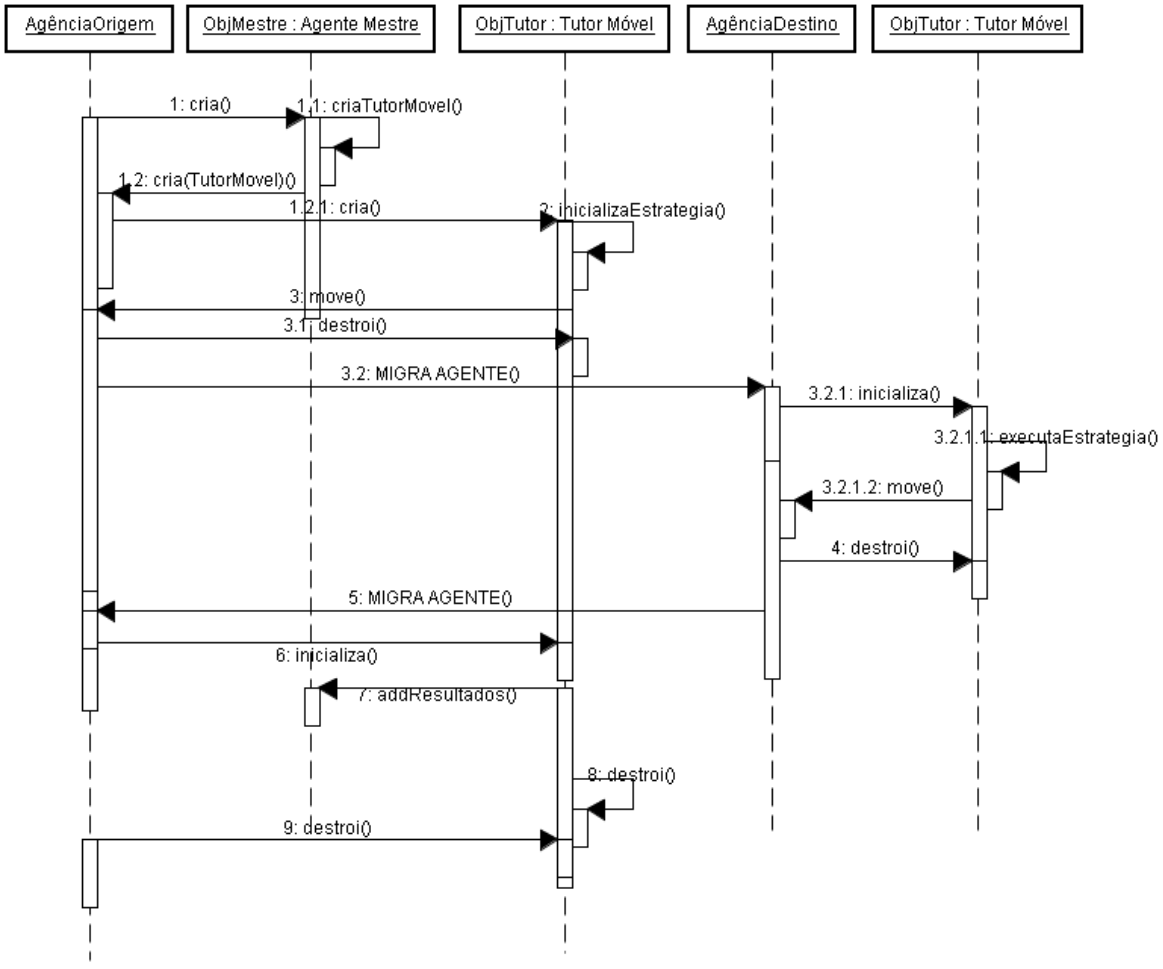


Figura 5.11 - Diagrama de sequência relacionado ao sistema tutor móvel

5.6.4. Diagrama de estados do Tutor Móvel

Na Figura 5.12, é ilustrado o diagrama de estados do Tutor Móvel. O diagrama integra estados dos componentes TutorMovel, AgenteMovel e Interprtetador. Assim, é representado a sequência de estados dos objetos durante o ciclo de vida de acordo com as várias transições, onde:

- Criar: Conjunto de dados enviados à agência.
- Inicializar: Obter os atributos básicos para inicializar o agente móvel.
- Ativar: Ativar o atente móvel, possibilitando as ações do atente móvel na tutoria.
- Clonar: Criar a cópia que é migrada.
- Acionar: O aluno aciona o agente móvel e inicia-se a sessão off-line.
- Carregar: O tutor móvel busca/carrega estratégia didática.
- Rodar: Acionar um ambiente no computador do aluno e interpretar estratégia.
- Buscar: Busca de objetos didáticos no computador do aluno.
- Reativar: A execução é reinicializada.
- Migrar: Despachar o agente móvel para o computador do aluno, bem com, enviá-lo de volta ao servidor.
- Remover: Atividades de fim de execução.
- Suspender: Tutor e aluno são desconectados e a atividade do agente móvel é temporariamente interrompida.

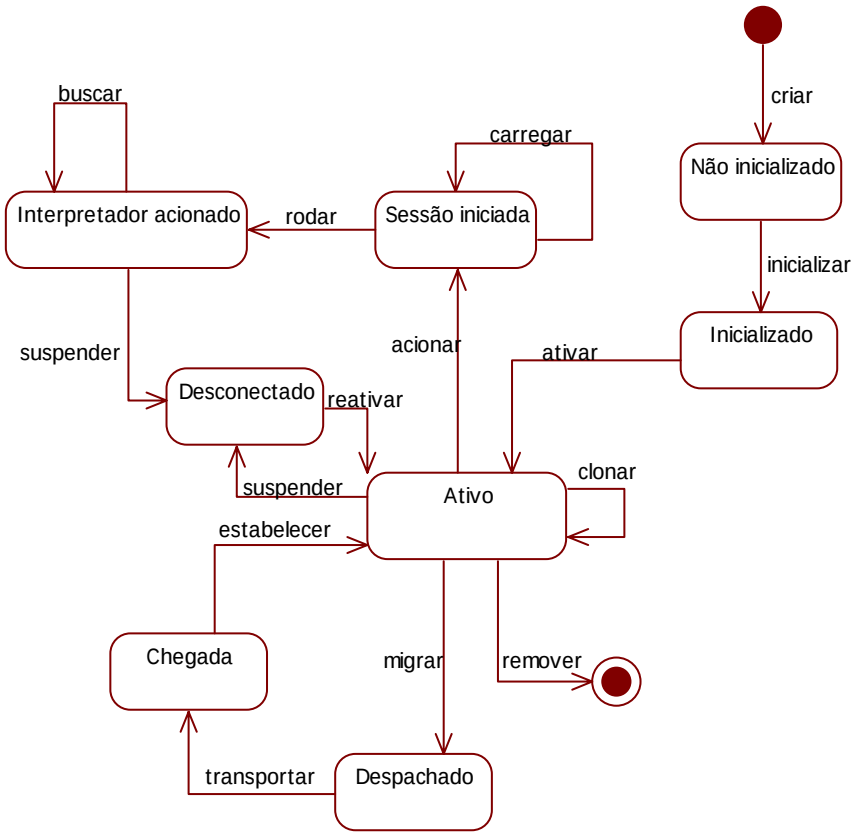


Figura 5.12 - Diagrama de estados do tutor móvel

5.7. Conclusão

Neste capítulo, foi dado destaque à modelagem do Sistema Tutor Móvel que está sendo proposto. Desta forma, foi realizada uma contextualização do sistema, de forma a inseri-lo na estrutura em camadas do *framework* FA_PorT. Foram discutidos funcionalidades e processos envolvidos em uma sessão de ensino *off-line* e, em seguida, descritos os requisitos funcionais e não funcionais do sistema, descrição dos casos de uso e identificação dos atores inseridos no escopo do sistema Tutor *on-line* e *off-line*. Por fim, foi mostrada a arquitetura em camadas e especificados os componentes, com diagrama de classes, sequência e estados, de cada camada.

No próximo capítulo, serão delineados aspectos relacionados à implementação do Sistema Tutor Móvel.

6. Implementação

Neste capítulo, é descrita a implementação das novas funcionalidades do *framework* FA_PorT considerando a integração do tutor móvel. A seção 6.1 apresenta a estrutura de uma estratégia a ser interpretada no computador do aluno sem a necessidade de conexão com *Internet*. Na seção 6.2, fala-se sobre o interpretador de estratégias s e na seção 6.3, mostra-se uma aplicação criada a partir do *framework*.

6.1. Estratégia

Para padronizar a estrutura da estratégia a ser interpretada no Tutor Móvel no computador do aluno, adotou-se uma estruturação em XML (*Extensible Markup Language*), onde cada nó representa uma tática e suas respectivas propriedades. Na Figura 6.1, é ilustrada uma estratégia que representa a utilização de três táticas de recurso no contexto do ensino de Proposições Lógicas, referente ao domínio Lógica Formal e à disciplina Lógica Matemática; estratégia utilizada do estudo de caso detalhado na sessão 6.3.

```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<estrategia>

	<tatica tipo="Reutilizacao-def">
		<tempo>5</tempo>
		<recurso>Definicao</recurso>
		<titulo>Objetivos</titulo>
		<descricao>Lógica para Computação</descricao>
		<conteudo tipo="imagem" >C:\\interpretador\\imagens\\principal.jpg</conteudo>
		<conteudo tipo="texto" >Fornecer ao aluno conhecimentos sobre a lógica matemática afim de
		<conteudo tipo="arquivo" >C:\\interpretador\\arquivos\\principal.pdf</conteudo>
	</tatica>

	<tatica tipo="Reutilizacao-def">
		<tempo>5</tempo>
		<recurso>Definicao</recurso>
		<titulo>Proposições</titulo>
		<descricao>O que é proposição?</descricao>
		<conteudo tipo="texto" >Uma proposição é uma afirmação passível de assumir valor lógico v
		<conteudo tipo="texto" >Toda proposição é verdadeira ou falsa (princípio do terceiro excl
		<conteudo tipo="texto" >Uma proposição não pode ser verdadeira E falsa (princípio da não-
		<conteudo tipo="arquivo" >C:\\interpretador\\arquivos\\proposicao.pdf</conteudo>
	</tatica>

	<tatica tipo="Reutilizacao-exe">
		<tempo>5</tempo>
		<recurso>Exemplo</recurso>
		<titulo>Proposições</titulo>
		<descricao>Exemplos de Proposições</descricao>
		<conteudo tipo="texto" >2 > 1 (V)</conteudo>
		<conteudo tipo="texto" >5 = 1 (F)</conteudo>
		<conteudo tipo="arquivo" >C:\\interpretador\\arquivos\\exemplo-proposicoes.pdf</conteudo>
	</tatica>

	<tatica tipo="Reutilizacao-def">
		<tempo>5</tempo>
		<recurso>Definicao</recurso>
		<titulo>Conectivos Lógicos</titulo>
		<descricao>Proposições podem ser conectadas através dos seguintes conectivos:</descricao>
		<conteudo tipo="texto" >"¬" ou "!" (negação)</conteudo>
		<conteudo tipo="texto" >"^" (conectivo "e")</conteudo>
		<conteudo tipo="texto" >"v" (conectivo "ou")</conteudo>
		<conteudo tipo="texto" >"->" (conectivo "implica")</conteudo>
		<conteudo tipo="texto" >"&lt;->" (conectivo "se, e somente se")</conteudo>
		<conteudo tipo="arquivo" >C:\\interpretador\\arquivos\\conectivos.pdf</conteudo>
	</tatica>

	<tatica tipo="Reutilizacao-exe">
		<tempo>5</tempo>
		<recurso>Exemplo</recurso>
		<titulo>Conectivos Lógicos</titulo>
		<descricao>Sejam "P" e "Q" proposições.</descricao>
		<conteudo tipo="texto" >"¬P" é verdadeira se "P" for falsa, e vice-versa;</conteudo>
		<conteudo tipo="texto" >"P e Q" é verdadeira se ambas forem verdadeiras, e falsa caso con
		<conteudo tipo="texto" >"P ou Q" é verdadeira se pelo menos uma delas for verdadeira, e f
		<conteudo tipo="texto" >"P!Q" é a mesma coisa que "(¬P) ou Q"; ou seja, é falsa se o lado
		<conteudo tipo="arquivo" >C:\\interpretador\\arquivos\\exemplo-conectivos.pdf</conteudo>
	</tatica>

	<tatica tipo="Reutilizacao-def">
		<tempo>5</tempo>
		<recurso>Definicao</recurso>
		<titulo>Referências</titulo>
		<descricao>Iniciação à Lógica Matemática</descricao>
		<conteudo tipo="texto" >Edgar de Alencar Filho. Editora Nobel.</conteudo>
		<conteudo tipo="imagem" >C:\interpretador\imagens\livro.jpg</conteudo>
	</tatica>

</estrategia>
```

Figura 6.1 - Representação de uma Estratégia em XML

6.2. Interpretador de Estratégias

Para que uma sessão de ensino *off-line* ocorra na máquina do aluno, é necessário que, encapsulado ao agente móvel, sejam migrados o tutor *off-line*, os recursos, a estratégia e o interpretador de estratégia. Dessa forma, independentemente de conexão com *Internet*, a estratégia enviada é interpretada e uma sessão de ensino personalizada pode ser realizada.

Como a estratégia trata-se de uma estrutura em XML, é necessário um mecanismo para realizar o *parsing* nesse XML e exibir o conteúdo referente às táticas envolvidas na estratégia. A Figura 6.2, Figura 6.3 e Figura 6.4 ilustram trechos do código responsável pelo carregamento da estratégia ao *parsing* na mesma.

```
1     function xmlCarregar(url){
2         if(window.XMLHttpRequest){
3             var Loader = new XMLHttpRequest();
4             Loader.open("GET", url ,false);
5             Loader.send(null);
6             return Loader.responseXML;
7         }else if(window.ActiveXObject){
8             var Loader = new ActiveXObject("Msxml2.DOMDocument.3.0");
9             Loader.async = false;
10            Loader.load(url);
11            return Loader;
12        }
13    }
```

Figura 6.2 - Trecho de código *javascript* responsável por carregar estratégia em XML

```
1     function exibeTatica(tatica){
2         if(tatica==1){
3             // Reutilização - Definição
4             // Reutilizacao-def.htm é gerado dinamicamente
5             //...
6             window.open ("Reutilizacao-def.htm", "Reutilização - Definição", "status=no, width=500, height=500");
7
8         }
9         if(tatica==2){
10            // Reutilização - Exemplo
11            // Reutilizacao-exe.htm é gerado dinamicamente
12            //...
13            window.open ("Reutilizacao-exe.htm", "Reutilização - Exemplo", "status=no, width=500, height=500");
14        }
15    }
```

Figura 6.3 - Trecho de código *javascript* responsável por exibir o conteúdo de uma tática

```
<script language="javascript" type="text/javascript">
function xmlTaticaArvore(xmlNode,identacao){
    var arvoreTxt=""; //esta var armazenara o conteudo
    var taticaNum=""; //esta var armazenara o tipo da tática

    for(var i=0;i<xmlNode.childNodes.length;i++){
    //percorrendo os filhos do nó if(xmlNode.childNodes[i].nodeType == 1){
    //ignorar espaço em branco
   if(xmlNode.childNodes[i].childNodes.length==0){
    //se não tiver filhos
    arvoreTxt = arvoreTxt + xmlNode.childNodes[i].nodeValue
    for(var z=0;z<xmlNode.childNodes[i].attributes.length;z++){
     var atrib = xmlNode.childNodes[i].attributes[z];
     //...
     // Monta página
     if(xmlNode.childNodes[i]=="tatica"){
        switch (xmlNode.childNodes[i].attributes[z]) {
          case "Reutilizacao-def":
            taticaNum = 1;
            break
          case "Reutilizacao-exe":
            taticaNum = 2;
            break
          //Para totod os tipos de tática
          default:
            alert("Erro...");
        }

     exibeTatica(taticaNum);
     }
     //...
    }
   }else if(xmlNode.childNodes[i].childNodes.length>0){
    //se tiver filhos eu tenho que pegar o valor pegando o valor do primeiro filho
    //...
    for(var z=0;z<xmlNode.childNodes[i].attributes.length;z++){
     var atrib = xmlNode.childNodes[i].attributes[z];
     //...
    }
    //recursividade para carregas os filhos dos filhos
    //...
   }
      }
    }
    return arvoreTxt;
  }
</script>
```

Figura 6.4 - Trecho de código javascript responsável pelo *parsing* no XML

A seguir, é ilustrada, na Figura 6.5, a estrutura do Tutor Móvel e os seguintes componentes: o Interpretador, a Estratégia e os Recursos.

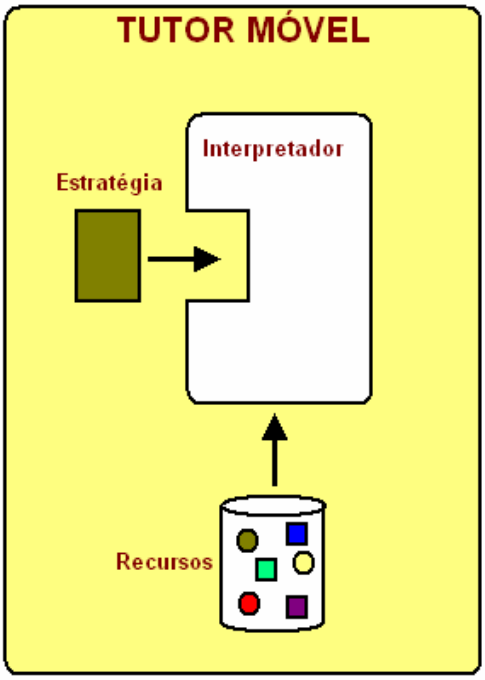


Figura 6.5 - Estrutura do Tutor Móvel

Depois de migrado o Tutor Móvel, a sessão *off-line* pode ser executada a qualquer momento, bastando que o aluno solicite o início. Assim, fica a cargo do Interpretador o mapeamento de cada tática com uma determinada ação do sistema. Vale salientar, que o Interpretador também utilizará uma base de Recursos compartilhados no momento da execução da sessão *off-line*. Um Recurso pode variar de um simples arquivo texto a arquivos multimídia. A escolha de qual Recurso utilizar fica a critério do professor no momento da definição da estratégia didática.

A seguir, na Figura 6.6 e Figura 6.7, são ilustrados trechos de código Java relacionados à classes que fazem parte da estrutura que é executada na forma de serviço no Sistema Operacional, onde este serviço fica ouvindo a porta em busca de agentes móveis com o conteúdo didático a ser exibido na sessão de ensino *off-line*.

```java
import jade.gui.GuiAgent;
import jade.gui.GuiEvent;
//...

public class Interpretador extends GuiAgent {
   private jade.wrapper.AgentContainer home;
   private jade.wrapper.AgentContainer[] container = null;
   private Map localizacao = new HashMap();
   private Vector agentes = new Vector();
   private int agenteCnt = 0;
   private int comando;
   transient protected InterpretadorGui minhaGui;

   public static final int SAIR = 0;
   public static final int NOVO_AGENTE = 1;
   public static final int MOVE_AGENTE = 2;
   public static final int CLONA_AGENTE = 3;
   public static final int DESTROI_AGENTE = 4;
   // Instância JADE
   jade.core.Runtime runtime = jade.core.Runtime.instance();

   protected void setup() {
      // Registro de linguagem e ontologia
      getContentManager().registerLanguage(new SLCodec());
      getContentManager().registerOntology(MobilityOntology.getInstance());

      try {
         // Cria os conteiners
         home = runtime.createAgentContainer(new ProfileImpl());
         container = new jade.wrapper.AgentContainer[3];
         // ...
         // Verifica localização via AMS
         sendRequest(new Action(getAMS(), new QueryPlatformLocationsAction()));
         //Resposta AMS
         MessageTemplate mt = MessageTemplate.and(
                              MessageTemplate.MatchSender(getAMS()),
                              MessageTemplate.MatchPerformative(ACLMessage.INFORM));
         ACLMessage resp = blockingReceive(mt);
         ContentElement ce = getContentManager().extractContent(resp);
         Result result = (Result) ce;
         jade.util.leap.Iterator it = result.getItems().iterator();

         while (it.hasNext()) {
             Location loc = (Location)it.next();
             String dest;
         //...
         }
      }
```

Figura 6.6 - Trecho de código Java referente à classe Interpretador

```
1  import jade.gui.GuiAgent;
2  import jade.domain.mobility.CloneAction;
3  import jade.domain.mobility.MobilityOntology;
4  //...
5
6  public class AgenteMovel extends GuiAgent {
7     private static final long serialVersionUID = 1L;
8     private AID interpretador;
9     private Location destino;
10    transient protected AgenteMovelGui minhaGui;
11
12    protected void setup() {
13       // Passagem de argumentos durante a criação do agente
14       //...
15
16       init();
17       // Comportamento do agente
18       addBehaviour(new ReceiveCommands(this));
19    }
20    void init() {
21       // Registro de linguagem e ontologia
22       getContentManager().registerLanguage(new SLCodec());
23       getContentManager().registerOntology(MobilityOntology.getInstance());
24       // ...
25    }
26    protected void onGuiEvent(GuiEvent e) {
27
28    }
29    protected void beforeMove() {
30       String dest;
31     //...
32    }
33    protected void afterMove() {
34     //...
35    }
36    protected void beforeClone() {
37     //...
38    }
39    protected void afterClone() {
40     //...
41    }
```

Figura 6.7 - Trecho de código Java referente à classe AgenteMovel

6.3. Estudo de caso

Utilizando o *framework* FA_PorT com as novas funcionalidades voltadas ao envio de um Tutor Móvel baseado em agentes móveis, foi instanciada uma aplicação. Nesta sessão, será visto o funcionamento da aplicação desde a criação das estratégias, pelo professor, até a

execução da sessão de ensino *on-line*, a execução e avaliação das atividades e, por fim, o envio do Tutor Móvel e execução, pelo aluno, da sessão de ensino *off-line*. A aplicação foi desenvolvida baseada na tecnologia *web*, mesmo a sessão *off-line*, porém, esta última não necessita de conexão com *Internet* para a correta execução. Na sessão de ensino *on-line*, a interface é basicamente a do navegador (*browser*), com as informações ocupando a área existente para a navegação. Já na sessão de ensino *off-line*, é necessário que um interpretador de estratégias e o ambiente de interação entre agentes móveis estejam devidamente instalados no computador do aluno.

A sessão de ensino, que faz parte da camada Tutor, é especificada pelo professor, ou seja, o professor cria as sessões de ensino *on-line* e *off-line*; e o sistema, através do comportamento pró-ativo, informa a criação desta sessão de ensino para os participantes (professores e alunos). Estas sessões de ensino podem conter uma ou várias estratégias (que também é criada pelo professor, ou pode ser utilizada alguma estratégia existente), que são compostas por táticas, conforme visto na seção 4.6.

A seguir, é apresentada com detalhes a aplicação Portfólio-Tutor com suporte a Tutoria Móvel desenvolvida.

A aplicação criada, objetiva auxiliar o professor da disciplina Lógica Matemática. Esta trabalha com a implementação *default* do *framework* FA_PorT, ou seja, é um sistema portfolio-tutor com apoio a Tutoria Móvel que possui funcionalidades específicas, sem a necessidade de redefinição dos pontos de adaptação de código.

Na Figura 6.8, é apresentada a tela principal da aplicação:

Figura 6.8 - Tela inicial da aplicação

As funcionalidades são disponibilizadas de acordo com o perfil do usuário da aplicação, sendo estes: professor, aluno e administrador.

O usuário, ao inicializar a utilização do sistema, precisará identificar-se através de um *login* e uma senha. O sistema, quando reconhecer o perfil do usuário, direcionará para sua respectiva interface. A Figura 6.9 ilustra a tela padrão para o perfil professor.

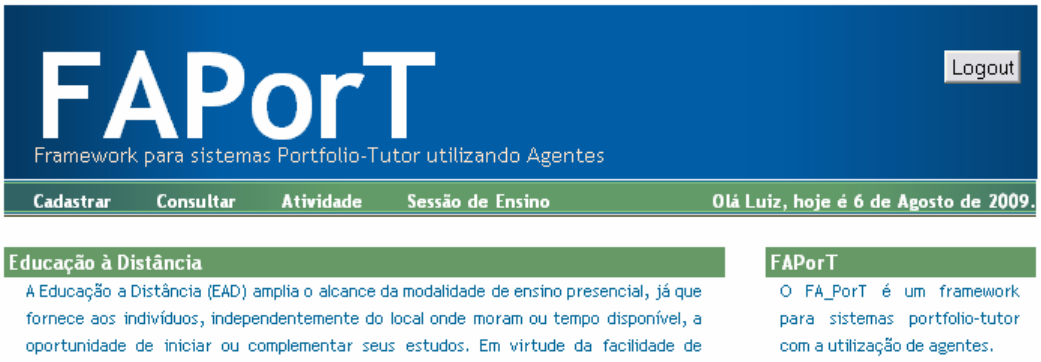


Figura 6.9 - Tela padrão para o perfil professor

A seguir, serão ilustradas algumas funcionalidades da aplicação.

A Figura 6.10 representa as opções para cadastro disponíveis ao professor, tais como, o cadastro de: aluno, disciplina, domínio, estratégia, unidade, item de avaliação, conceito, grupo, alunos de grupo e recurso didático, bem como, a criação de sessões de ensino e atividades.

Figura 6.10 - Tela padrão para o perfil professor

No presente estudo de caso, foi realizado o cadastro de 04 alunos, associando-os a um grupo, conforme pode ser visualizado na Figura 6.11.

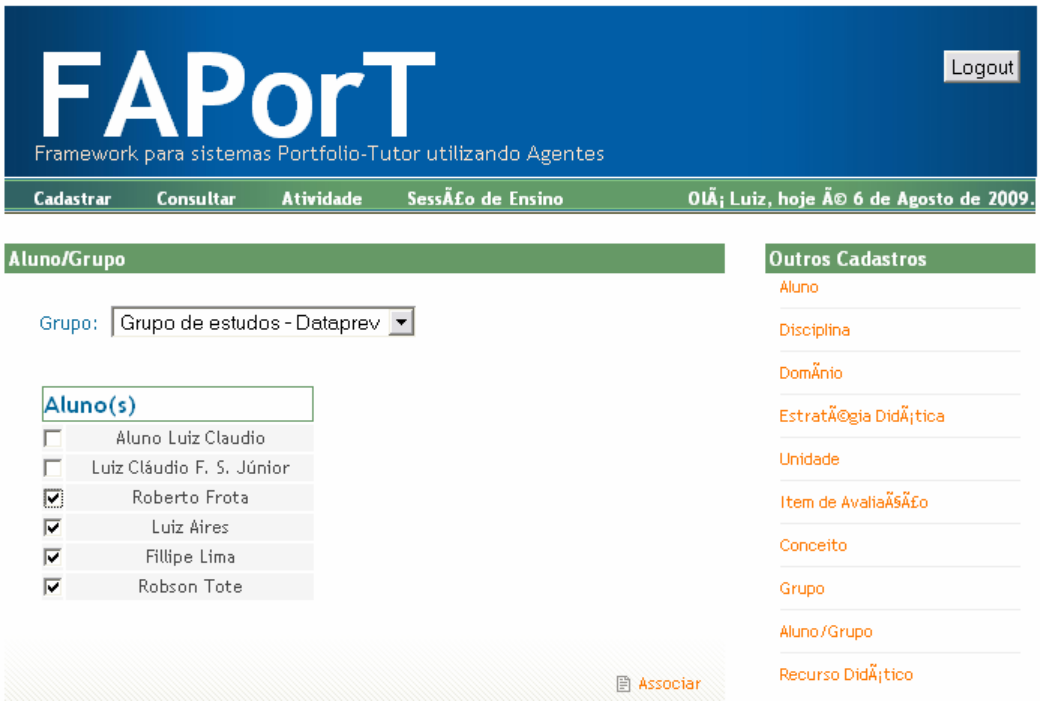


Figura 6.11 - Tela de associação de aluno x grupo

Após a criação dos alunos e associação a um grupo de alunos, foram criadas, pelo professor, as atividades, as quais foram atribuídas aos respectivos alunos. Quando uma atividade é associada a um aluno, este recebe um *e-mail* com o informe. A Figura 6.12, Figura 6.13 e Figura 6.14 ilustram a criação de atividades.

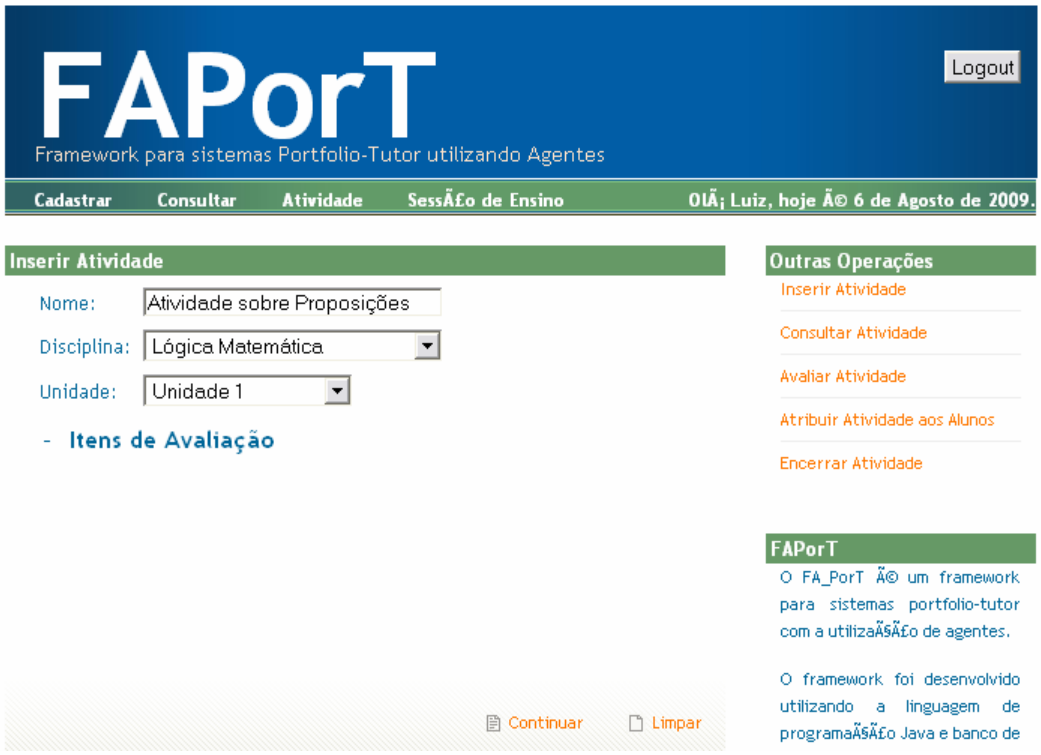


Figura 6.12 - Tela de inserção de atividades

Figura 6.13 - Tela de consulta de atividades

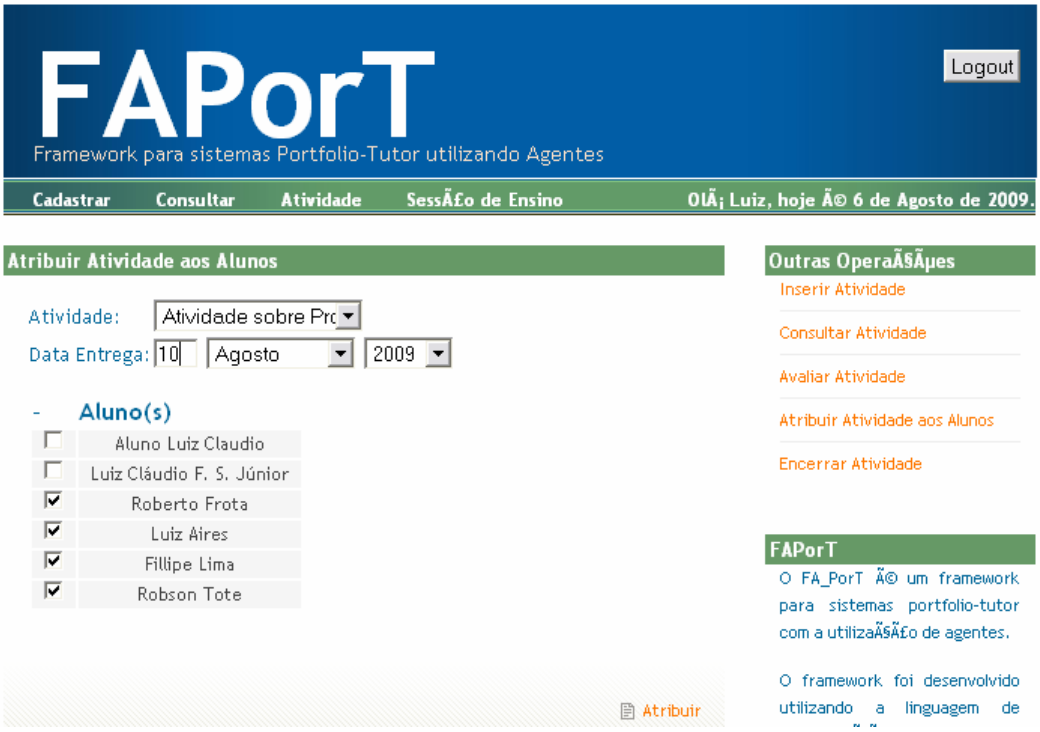


Figura 6.14 - Tela de atribuição de atividades aos alunos

Ao criar as atividades e atribuí-as aos alunos, o professor, no estudo de caso, criou as estratégias que serão utilizadas nas sessões de ensino *on-line* e *off-line*. Conforme pode ser visualizado na Figura 6.15 e Figura 6.16, foi especificada a estratégia de enviar recursos didáticos aos alunos, que em seguida passam por um *chat* e, por fim, recebem um Tutor Móvel, caso a nota na atividade seja inferior a 7.0.

Figura 6.15 - Tela de criação da estratégia a ser utilizada na sessão *on-line*

Figura 6.16 - Tela de criação da estratégia de reforço a ser utilizada na sessão *off-line*

Conforme mencionado, um interpretador de estratégias e um ambiente para dar apoio à interação entre os agentes móveis são previamente instalados no computador do aluno para que a sessão de ensino *off-line* ocorra. A ilustração a seguir, destaca o ambiente onde o agente móvel é clonado e migrado do servidor de aplicações, que será chamado de computador do professor, para o do computador aluno. A numeração informa a sequência dos passos desde a clonagem ainda no computador (*container*) do professor, até a finalização da sessão no computador (*container*) do aluno. Vale salientar, que o processo ilustrado é transparente para o aluno.

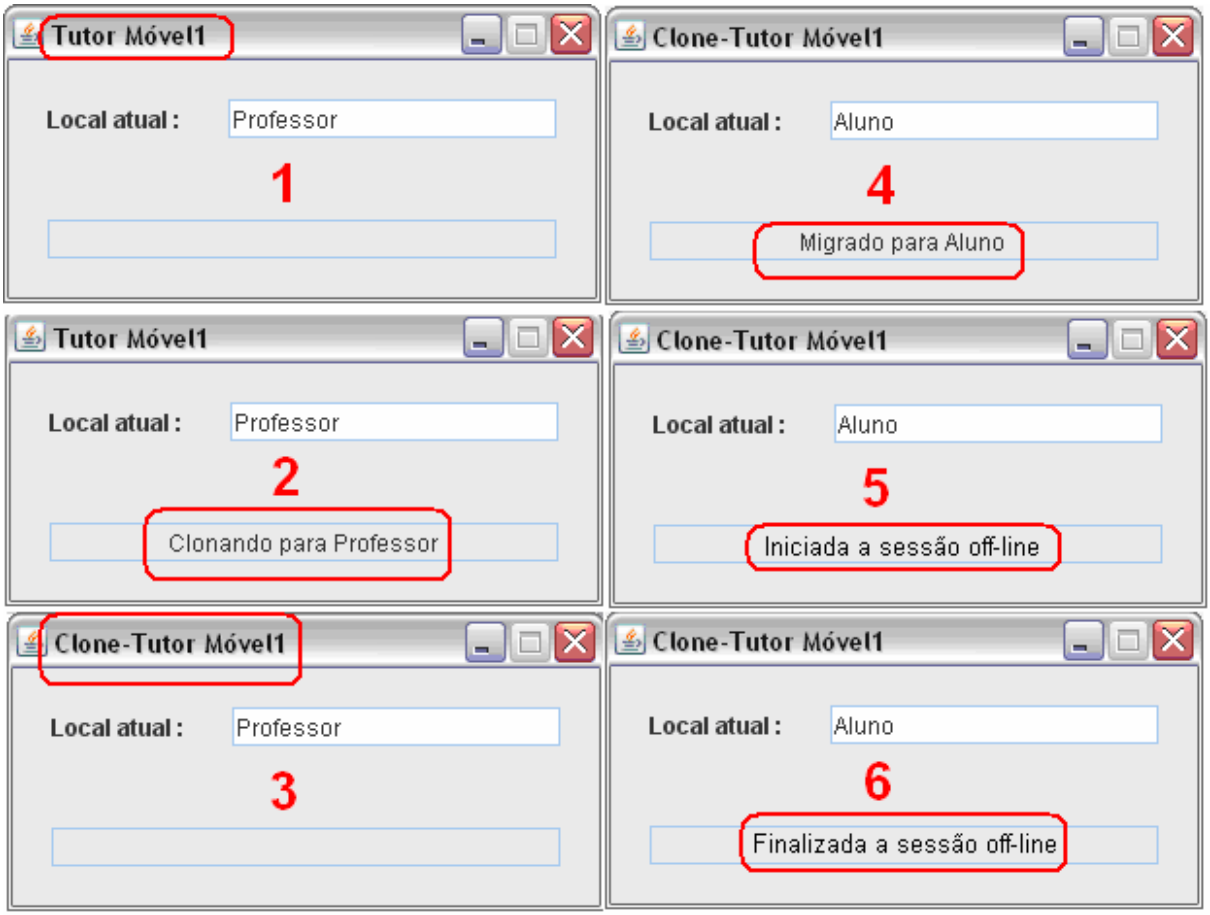


Figura 6.17 - Telas que mostram a sequência das interações entre o agente móvel e os *containeres*.

No passo 5 da figura acima, uma sessão de ensino é executada pelo aluno e após o passo 6, o agente móvel é migrado de volta ao container do professor, o perfil do aluno é atualizado e o agente móvel é destruído. Assim, o aluno poderá realizar novamente as atividades no contexto das sessões *on-line*.

Na Figura 6.18(a), Figura 6.18(c) e Figura 6.19(a), ilustram-se parte da sessão de ensino *off-line* exibida no computador do aluno. Trata-se da exibição das táticas Reutilização de Recursos - Definição nas duas primeiras ilustrações e Reutilização de Recursos – Exemplo na terceira, táticas essas, contidas do arquivo de estratégias *off-line*. Já a Figura 6.18(b), Figura 6.18(d) e Figura 6.19(b), mostram os recursos didáticos exibidos ao aluno no momento em que o aluno clica no *link* (Clique aqui para acessar o Recurso Didático) disponível na Figura 6.18(a), Figura 6.18(c) e Figura 6.19(a). Na Figura 6.18 e Figura 6.19 são exibidos

documentos em *pdf*[1] e um mapa conceitual na ferramenta *CMAPTools*[2]. Obviamente, tais ferramentas devem ser previamente instaladas no computador do aluno para efetiva visualização dos recursos didáticos. Outros tipos de táticas podem ser utilizados nas sessões de ensino *off-line*, tais como, Mudança de Estratégia e Regra. Porém, como se trata de uma execução em *background*, não há ilustração disponível para tais táticas.

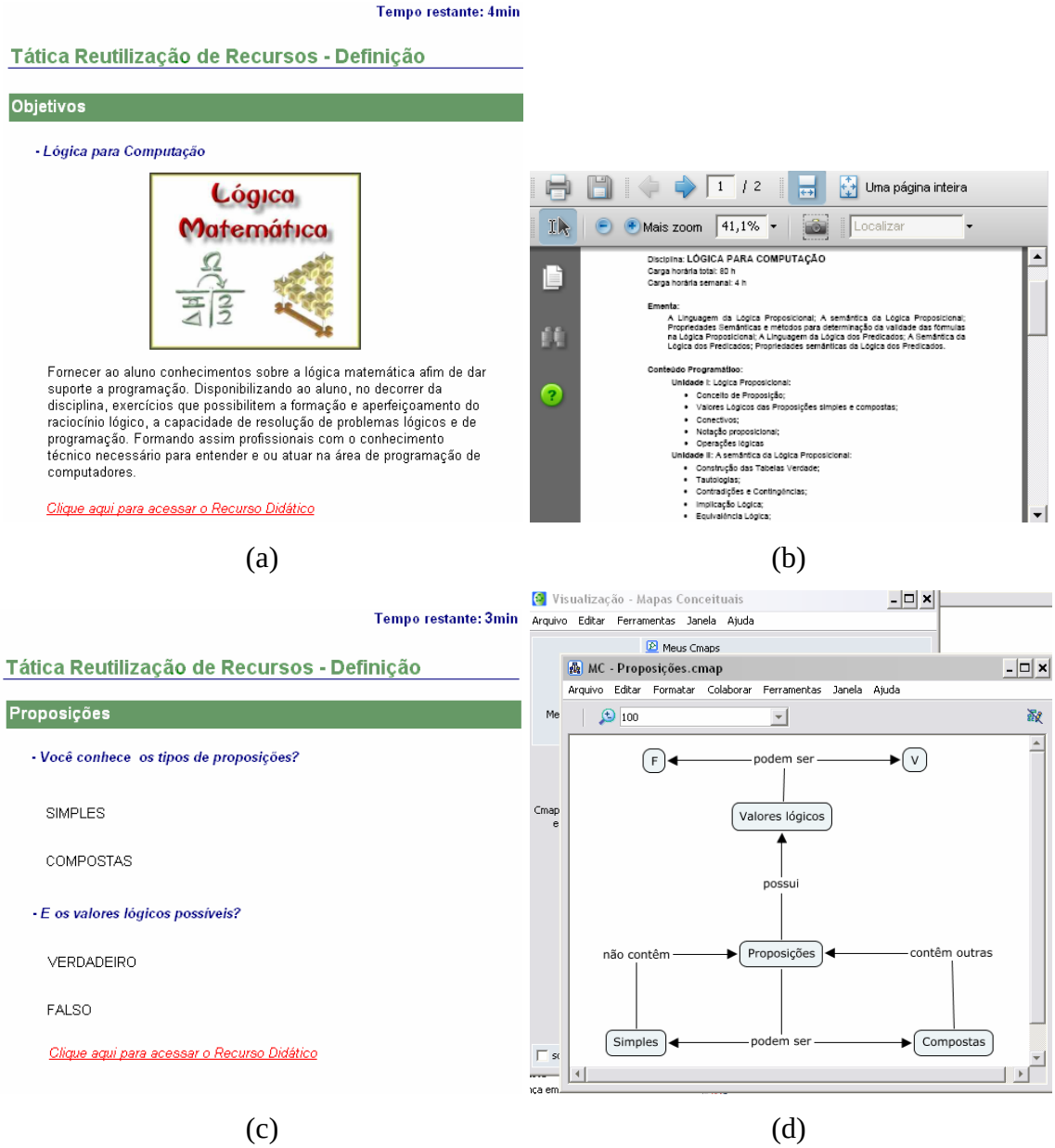
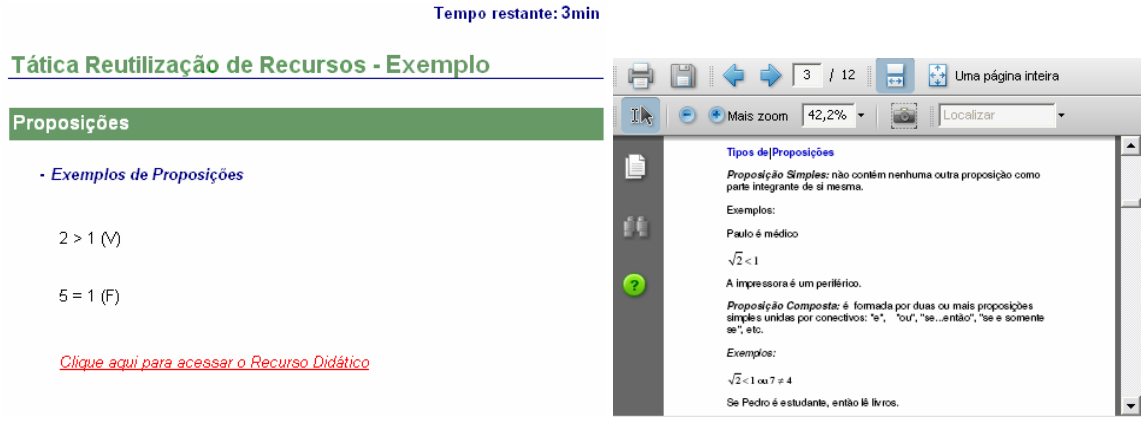


Figura 6.18 - Telas que mostram a tática Reutilização de Recursos *off-line* – Definição

[1] É um formato de arquivo denominado Portable Document Format, que permite capturar e visualizar documentos em praticamente todas as plataformas de sistemas operacionais.

[2] Ferramenta que auxilia na elaboração de mapas conceituais, facilitando a representação do conhecimento.



(a) (b)

Figura 6.19 - Tela que mostra a tática Reutilização de Recursos *off-line* - Exemplo

7. Considerações finais

Neste capítulo, apresentam-se as considerações finais, as contribuições e as perspectivas para trabalhos futuros.

O trabalho apresentado nesta dissertação objetivou a especificação e implementação (protótipo) de um sistema Tutor Móvel baseado em Agentes Móveis acoplado ao *framework* FA_PorT, bem como, a construção de uma aplicação que utiliza um tutor móvel gerada a partir do FA_PorT. A principal contribuição contextualiza-se a partir de pesquisas voltadas a: *framework* FA_PorT (MEDEIROS, 2006), uso de Agentes Móveis Jade (JADE, 2009; OMG MASIF, 1998) e aplicação do padrão de projeto *Master-Slave* para agentes móveis (ARIDOR; LANGE, 1998).

Aplicações educacionais são tipos de sistemas que vêm sendo utilizados de forma crescente. Porém, poucos têm sido desenvolvidos com a preocupação em apoiar, de forma *off-line*, os alunos com dificuldade na aprendizagem. O Tutor Móvel acoplado ao FA_PorT proposto neste trabalho, possibilita a personalização e individualização do ensino, especificamente aos alunos que apresentem dificuldades de aprendizagem durante uma sessão de ensino na *web*.

7.1. Contribuições

Nesta dissertação, foi apresentado uma ampliação do *framework* FA_PorT. Trata-se de um sistema tutor móvel baseado em agentes móveis, que auxilia o aluno com dificuldades na compreensão do conteúdo apresentado em sessões de ensino *on-line* para um grupo virtual de alunos. O sistema tutor móvel é ativado em sessões de ensino de um sistema tutor *on-line*, através de uma tática denominada Tática de Assistência Personalizada que, quando executada, aciona a migração do sistema tutor móvel para o computador aluno. Após a migração, o tutor móvel possibilita a realização de sessões de ensino individualizadas no computador do aluno, sem manter a conexão com o sistema tutor *on-line*. A arquitetura de um sistema tutor móvel é baseada na arquitetura de um sistema portfólio-tutor. O sistema tutor móvel proposto foi desenvolvido utilizando Java e Jade (*Java Agent Development Framework*). O sistema tutor

móvel poderá ser utilizado em diversos domínios, tais como: no ensino de projeto orientado a objetos, no ensino de componentes de software etc..

Durante a elaboração deste trabalho, publicou-se um artigo completo na área de pesquisa voltada a Informática na Educação (SILVA JUNIOR, 2008).

7.2. Trabalhos futuros

Alguns trabalhos futuros podem ser destacados. É interessante a pesquisa e desenvolvimento de tutores móveis para dispositivos móveis, o que poderia dar bastante suporte ao crescimento da *m-learning*. Há, também, o interesse em pesquisas voltadas a Usabilidade nas aplicações instanciadas pelo *framework* FA_PorT com o intuito de evitar a sobrecarga cognitiva, ou seja, deve-se evitar que os esforços despendidos pelos alunos para usar a ferramenta interfiram negativamente no processo de aprendizagem e, consequentemente, na construção de conhecimento sobre o conteúdo proposto na sessão de ensino *on-line* e *off-line*. Para isso, é importante a percepção de que a Usabilidade não é uma propriedade unidimensional de uma interface de usuário (DIAS, 2003), devendo, assim, estudos focados nos múltiplos componentes como: Aprendabilidade (fácil de aprender); Eficiência (eficiência no uso, maior produtividade); Memorabilidade (fácil de memorizar); Erros (baixa taxa de erros e permitir recuperação); Agradabilidade (usuário gostar do sistema). Pesquisa sobre Objetos de Aprendizagem (WILEY, 2000) é outra perspectiva futura no contexto do *framework* FA_PorT, cujo objetivo é possibilitar o reuso de recursos digitais que possam assistir a aprendizagem nas aplicações instanciadas pelo FA_PorT, podendo ser distribuídos pela rede, sob demanda.

Referências

ADNAN S.; DATUIN J.; YALAMANCHILI P. **A survey of mobile agent systems**, 2000. Disponível em: <http://www.agent.ai/doc/upload/200302/adna00.pdf>. Acesso em: agosto de 2009.

AGLETS. **Website do Projeto Aglets**, 2004. Disponível em: <http://aglets.sourceforge.net>. Acesso em: agosto de 2009.

AJANTA. **Website do projeto AJANTA,** 2009.Disponível em: <http://ajanta.cs.umn.edu/home.html>. Acesso em: agosto de 2009.

ALTMANN, J. et al. **Using Mobile Agents in Real World**: A Survey and Evaluation of Agents Platforms. In Proceedings of the 2nd International Workshop on Infrastructure for Agents, MAS, and Scalable MAS at the 5th International Conference on Autonomous Agents, 2001.

ANDERSON, J.R.; REISER, B.J. **The LISP Tutor**. Byte, 10, 1985. p. 159-175.

ANEIBA A.; REES S.J. **Mobile Agents Technology and Mobility**. Proc. of the 5th Annual Postgraduate Symposium on the Convergence of Telecommunications Networking, and Broadcasting, 2004.

ANGHEL, C. E SALOMIE, I. **JADE Based solutions for knowledge assessment in eLearning Environments**. TILAB & University of Limerick, 2003.

ARIDOR, Y; LANGE,D. **Agent design patterns**: Elements of agent application design. In Proceedings of the Second International Conference on Autonomous Agents. ACM Press, 1998. p. 108-115.

BARBOSA, R. M.. **MobiGrid**: arcabouço para agentes móveis em ambiente de grades computacionais. 2007. dissertação (Mestrado), Instituto de Matemática e Estatística. Universidade de São Paulo. São Paulo, 2007.

BARR, A.; FEIGENBAUM, E. A. **The handbook of Artificial Inteligence**. Los Altos, CA: Kaufmann, 1982.

BECTA REPORT. **Portable ICT devices**, 2009. Disponível em: <http://www.becta.org.uk/research/reports/portableict.cfm>. Acesso em: julho de 2009.

BELLIFEMINE, F et al. **JADE**: A White Paper, 2003. Disponível em: <http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf>. Acesso em: julho de 2009.

BERNERS-LEE, T. **Universal Resource Identifiers in WWW**, RFC1630, 1994.

BERNERS-LEE, T; MASINTER, L. **Uniform Resource Locators**, RFC1738, 1994.

ACM '74. ACM, New York, NY, 1974. 571-579 p. Disponível em: <http://doi.acm.org/10.1145/1408800.1408855>. Acesso em: agosto de 2009.

CAPELL, P.; DANNENBERG, R.B. **Instructional design and intelligent tutoring**: Theory and the precision of design. Journal of Artificial Intelligence in Education, 4, 1993. p. 95-121.

CARDOSO, R. S. **Adaptação Dinâmica Utilizando Agentes Móveis em Computação Ubíqua**. 2005. dissertação (Mestrado). Instituto de Matemática e Estatística, Universidade de São Paulo. São Paulo, 2005.

CHANG, C. Y.; SHEU, J. P. **Design and Implementation of Ad Hoc Classroom and eSchoolbag Systems for Ubiquitous Learning**. WMTE2002, 2002. p. 8-14.

CHEN, S.; NAHRSTEDT, K. **Distributed Quality-of-Service Routing in Ad Hoc Networks**. IEEE Journal on Selected Areas in Communications, 2000. p. 1594-1603.

CHESS, D. M. **Security issues in mobile code systems**. In Mobile Agents and Security, 1998. p. 1-144

CHESS, D; HARRISON, C.; KERSHENBAUM, A. **Mobile Agents**: Are They a Good Idea? Relatório Técnico RC 19887 (December 21, 1994 - Declassiéd March 16, 1995), IBM Research Division, 1994.

CLANCEY, W.J. **Knowledge-based tutoring**: the GUIDON program. The MIT Press, 1987.

_____. **Methodology for building an intelligent tutoring system**. In: KEARSLEY, G. Artificial intelligence and instruction: applications and methods, 1987b. p. 193-227.

CORBETT, A.T.; ANDERSON, J.R. **Knowledge racing**: Modeling the acquisition of procedural knowledge. User modeling and user-adapted interactions 4, 1995b. p. 253-278.

COSTA, E. B. **Um modelo de ambiente interativo de aprendizagem baseado numa arquitetura multiagente**. CPGEE, Processamento da Informação, Universidade Federal da Paraíba, João Pessoa, 1997.

COSTA, R. M.; WERNECK, V. **Tutores Inteligentes**. Rio de Janeiro: Coordenação de Programas de Pós-Graduação de Engenharia, Universidade Federal do Rio de Janeiro, 1996.

CROCKER, L.; ALGINA, J. **Introduction to classical and modern test theory.** New York, NY: Harcourt Brace Jovanovich College Publishers, 1986.

CUSTÓDIO, T. **Implementação de uma Aplicação de Agentes Móveis para Avaliar o Impacto da Mobilidade na Segurança.** 2002. dissertação (Mestrado). Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina. Florianópolis, 2002.

DANNY, B. L.; MITSURU, O. **Seven good reasons for mobile agents.** Communications of the ACM , 42(3): 1999. p. 88-89.

DASGUPTA, P. et al. **Mobile agents for networked**. Electronic Trading, IEEE, 1999.

DELGADO, K. V. ; BARROS, L. N. . **ProPAT**: A Programming ITS Based on Pedagogical Patterns. In LESTER, J. C.; VICARI, R. M.; PARAGUAÇU, F. (Eds.): The 7th International Conference, ITS (Intelligent Tutoring Systems), Maceió, AL: Springer-Verlag. v. 3220. 2004, p. 812-814.

DIAS, C. A. **Usabilidade na web**: criando portais mais acessíveis. Rio de Janeiro: Alta Books, 2003. 312 p.

DINSOREANU, M. et al. **Mobile Agent based Solutions for Knowledge Assessment in eLearning Environments**. Euromedia, 2003

ELEUTÉRIO, M. **AMANDA**: a computational method for medianting group asynchronous discussions. 2002. Tese (Doutorado). Programa de Pós-Graduação em Ciência da Computação Aplicada, Pontifícia Universidade Católica do Paraná. Departamento de Engenharia da Computação, Universidade de Tecnologia de Compiègne (UTC). Curitiba, 2002. 170 f.

ELEUTERIO, M. ; BORTOLOZZI, Flávio . **AMANDA**: An ITS for Mediating Asynchronous Group Discussions. In LESTER, J. C.; VICARI, R. M.; PARAGUAÇU, F. (Eds.): The 7th International Conference, ITS (Intelligent Tutoring Systems), Maceió, AL: Springer-Verlag. v. 3220. 2004, p. 815-817.

ETZIONI, O E; DANIEL S. **Intelligent Agents on the Internet**: Fact, Fiction, and Forecast. IEEE Expert: Intelligent Systems and Their Application, 10(4), 1995. p. 44-49.

FAYAD, M. E.; SCHMIDT, D. C. e JOHNSON, R. E. **Building Application Frameworks:** Object-Oriented Foundations of Framework Design. ed. John Wiley & Sons: Nova Iorque, 1999.

FIPA. **The Foundation for Intelligent Physical Agents**. Website FIPA, 2009. Disponível em: <http://www.fipa.org/>.Acesso em: julho de 2009.

FUGGETA, A.; PICCO, G.P.; VIGNA, G. **Understanding code mobility**. In IEEE Transactions on Software Engineering, volume 24 / 5, 1998.

GIALDI, M. V. **Um Modelo para Portais Móveis baseado em Middleware Reflexivo e Agentes Móveis**. 2004. dissertação (Mestrado). Instituto de Computação, Universidade Estadual de Campinas. Campinas, 2004.

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

GREEN, D.M.; SWETS, J.A. **Signal Detection theory and Psychophysics**. Huntington, NY: Robert E. Krieger Publishing, 1973.

HATZILYGEROUDIS, I.; PRENTZAS, J. **Knowledge representation requirements for intelligent tutoring systems**. In LESTER, J. C.; VICARI, R. M.; PARAGUAÇU, F. (Eds.): The 7th International Conference, ITS (Intelligent Tutoring Systems), Maceió, AL: Springer-Verlag. v. 3220. 2004, p. 87-97.

JADE. **Java Agent Development Framework**, 2009. Disponível em: <http://jade.tilab.com/>. Acesso em: agosto de 2009.

JOHNSON, W.L.; SOLOWAY, E.M. **PROUST**: Knowledge-based program debugging. Proceedings of the Seventh International Software Engineering Conference, Orlando, FL, 1984.

KAPLAN, R. New **directions for intelligent tutoring**. AI Expert, 1995. p. 30-40.

KAWAMURA, T.; SUGAHARA, K. **A Mobile Agent-Based P2P e-Learning System.** IPSJ Journal, Vol. 46, No. 1, 2005. p. 222-225.

KAY, A. **Computer software**. Scientific American, 3(251), 1984. p. 53-59.

KOTZ, D. et al. **Mobile Agents for Mobile Computing**. In Technical Report PCS-TR96-285, May 1996, Computer Science Department, Dartmouth College 1996 Disponível em: <http://citeseer.ist.psu.edu/gray96mobile.html>. Acesso em: agosto de 2009.

LEHNER, F.; NÖSEKABEL, H. **The Role of Mobile Devices In E-Learning**: First Experiences With A Wireless E-Learning Environment. WMTE, 2002. p. 103-106.

LIMA, E. F. A. **Formalização e Análise de Padrões de Projeto para Agentes Móveis.** 2004. dissertação (Mestrado). Pós-Graduação em Informática, Universidade Federal de Campina Grande. Campina Grande, 2004.

LIN, K. T. **Improving mobile learning environments by applying mobile agents technology**. Third Pan Commonwealth Forum on Open Learning, 2004. Disponível em: <http://www.col.org/pcf3/Papers/PDFs/Kinshuk_Lin_2.pdf>. Acesso em: julho de 2009.

LIU, Z. et al. **Research on Mobile Agents in E-learning Service System**. icnc, vol. 5, Third International Conference on Natural Computation (ICNC 2007), 2007. p. 801-804.

LOG4J. **Website do projeto Log4J**, 2009. Disponível em: <http://logging.apache.org/log4j/>. Acesso em: julho de 2009.

MATOS, F. M. **Um Modelo de Negociação Automatizada em Comércio Móvel Utilizando Agentes Móveis**. 2003. dissertação (Mestrado), Instituto de Computação. Universidade Estadual de Campinas. Campinas, 2003.

MEDEIROS, F. N. **Faport:** Um framework para sistemas portfólio-tutor baseado em agentes. 2006. dissertação (Mestrado) - Programa de Pós-Graduação em Modelagem Computacional do Conhecimento, Universidade Federal de Alagoas - Instituto de Computação, Maceió, 2006.

MEGAN FOX. **Mobile Technologies in Libraries**: How the academic library is using pda's, handhelds and other mobile technologies. Associate Director for Technology & Special Projects. Simmons College Library, 2007. Disponível em: <http://web.simmons.edu/~fox/PDA.html>. Acesso em: julho de 2009.

MICROSOFT. **Windows CE for Mobile Device.** White Paper, Microsoft Corp, 2001.

MOBILEARN PROJECT. **Website do projeto**, 2009. Disponível em: <http://www.mobilearn.org/> Acesso em: julho de 2009.

NASCIMENTO, D. M. C. **Um sistema tutor acoplado a um portfólio eletrônico no contexto da educação a distância - portfólio-tutor**. 2002. dissertação (Mestrado) - Coordenação de Pós-Graduação em Informática, Universidade Federal da Paraíba – Departamento de Sistemas e Computação, Campina Grande, 2002.

NASSIF, L. N.; COSTA, M. F.; RESENDE, L. H. A.. **AGA**: Sistema de agentes móveis no gerenciamento de redes orientado a aplicação. Anais do I WORKCOMP-SUL. I Workshop de Computação da Região Sul. Florianópolis, 2004.

OHLSSON, S. **Some principles of Intelligent Tutoring**. In: LAWLER, R., YAZDANI, M. (Eds.). Intelligent Tutoring Systems. USA: Ablex Publishing Corporation, 1987.

OMG MASIF. **MASIF Revision**. Technical report, Object Management Group, 1998. Disponível em: <http://www.omg.org/cgi-bin/doc?orbos/98-03-09.pdf>.Acesso em: julho de 2009.

PEINE, H. **Application and programming experience with the Ara mobile agent system**. Software-Practice and Experience, 2002.

PEREIRA, A. S.; D'AMICO. C. B.; GEYER. C. F. R. **Uma Aplicação de Ensino Orientada a Agentes**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE'98, 19., 1998, Fortaleza, CE. Anais... Fortaleza: UFC, 1998.

PICCO, G. P. **Understanding, Evaluating, Formalizing, and Exploiting Code Mobility.** 1998. Tese (Doutorado), Politecnico di Torino – Dipartimento di Automática e Informatica, 1998.

PIROLLI, P.L.; GREENO, J.G. **The problem space of instructional design**. In J. Psotka, L. Massey and S. Mutter (Eds.) Intelligent tutoring systems: Lesson learned. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

POLSON, M.C.; RICHARDSON, J.J. **Foundations of Intelligent Tutoring Systems**. Hillsdale, NJ: Lawrence Eribaum Associates, 1988.

REISER, B.J.; KIMBERG, D.Y.; LOVETT, M.C.; RANNEY, M. **Knowledge representation and explanation in GIL**: An intelligent tutor for programming. In J. Larkin and R. Chabay (Eds.) Computer-assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches. Hilisdale, NJ: Lawrence Eribaum Associates, 1992.

SAMPSON, D.; KARAGIANNIDIS, C. **Personalized learning**: educational, technological and standardization perspective. Interactive Educational Multimedia, Vol. 4. Universidade de Barcelona. Barcelona, 2002. p. 24-39.

SANCHO, J. M. **A Autoformação e a Formação à [sic] Distância** : as tecnologias da educação nos Processos de Aprendizagem. apud Para Uma Tecnologia Educacional, Porto Alegre, 1998, 185 p.

SANTANA, L. H. Z. et al. **Usando Ontologias, Serviços Web Semânticos e Agentes Móveis no Desenvolvimento Baseado em Componentes**. SBCARS 2007. Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software. Unicamp. Campinas, 2007.

SCHOEMAN, M; CLOETE, E. **Architectural components for the efficient design of mobile agent systems**. Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology, 2003. p. 48-58.

SCOTTISH GOVERNMENT. Leap: A manual for Learning Evaluation and Planning in Community Learning and Development. 2. ed. Edinburgh: RR Donnelley, 2007. Disponível em: <http://www.scotland.gov.uk/Resource/Doc/205982/0054748.pdf/> Acesso em: setembro de 2009.

SHARPLES, M. **The Design of Personal Mobile Technologies for Lifelong Learning, Computers and Education**, vol. 34, 2000. p. 177-193.

SILVA, A. S. **Tuta**: um tutor baseado em agentes no contexto do ensino à distância. 2000. dissertação (Mestrado) - Coordenação de Pós-Graduação em Informática, Universidade Federal da Paraíba – Departamento de Sistemas e Computação, Campina Grande, 2000.

SILVA JÚNIOR, L. C. F. ; HERNÁNDEZ-DOMÍNGUEZ, A. **Um Sistema Tutor Móvel no Contexto de um Framework de Sistemas de Ensino**. In: XIX Simpósio Brasileiro de Informática na Educação (SBIE 2008), 2008, Fortaleza-CE. Tecnologia e Educação para Todos. Porto Alegre RS : Sociedade Brasileira de Computação SBC, 2008. v. I. p. 696-705.

SOMA. **Website do projeto SOMA**, 2009. Disponível em: <http://lia.deis.unibo.it/Software/SOMA/>. Acesso: julho de 2009.

SOURCEFORGE. **SourceForge.net**. Website do SourceForge.net, 2009. Disponível em: <http://sourceforge.net/> Acesso em: julho de 2009.

TAHARA, Y. et al. **Safety and security in mobile agents**. In Draft Proceedings of AOSE2000, 2000. p. 107-126.

_____. **Secure and efficient mobile agent application reuse using patterns**. In Proceedings of the 2001 symposium on Software reusability, ACM Press, 2001. p. 78-85.

TAHITI. **Tahiti user's guide**, 2002. Disponível em: <http://www.trl.ibm.com/aglets/tahiti/tahiti.htm>. Acesso em: agosto de 2009.

TEIXEIRA, A. M. S. **Análise de Contingências em Programação de Ensino**: Legado de Carolina Martuscelli Bori. In: Hélio José Guilhardi; Noreen Campbell de Aguirre (Org.). Sobre Comportamento e Cognição, expondo a variabilidade, 1ª ed. Santo André: ESETec Editores Associados, v 15 (1), 2002. p.7-17.

TILAB. **Website do Telecom Italia Lab Home Page ITA**, 2009. Disponível em: <http://www.tilab.com>. Acesso em: julho de 2009.

TOSHIBA. **Bee-gent**: Bonding and Encapsulation Enhancement aGENT, 2009. Disponível em: <http://www.toshiba.co.jp/rdc/beegent/>. Acesso em: julho de 2009.

TOWNE, D.M.; MUNRO, A. **Supporting di verse instructional strategies in a simulationoriented training environment.** In J. Regian and V. Shute (Eds.) Cognitive Approaches to Automated Instruction. Hillsdale, NJ: Lawrence Erlbaum Associates, 1992.

UTO, N.. **Segurança de Sistemas de Agentes Móveis.** 2003. dissertação (Mestrado). Unicamp, 2003. Disponível em: <http://www.las.ic.unicamp.br/paulo/teses/trabalhos-relacionados/NelsonUto.pdf.>. Acesso em, julho de 2009.

VICCARI, R. M.; GIRAFFA, L. M. M. Fundamentos de Sistemas Tutores Inteligentes. In: BARONE, D. **Sociedades artificiais**: a nova fronteira da inteligência nas máquinas. Porto Alegre: Ed. Artmed, 2003. p. 155-208.

VIGNA, G. **Mobile Code Technologies, Paradigms, and Applications.** 1997. Tese (Doutorado). Politecnico di Milano, 1997.

WEBER, G., BRUSILOVSKY, P., **ELM-ART**: an adaptive versatile system for Web-based instruction, *International Journal of Artificial Intelligence in Education*, Vol. 12, 2001. pp.351-84. Disponível em: <http://www2.sis.pitt.edu/~peterb/papers/JAIEDFinal.pdf>. Acesso em: setembro de 2009.

WILEY, A. **Connecting learning objects to instructional design theory**: A definition, a metaphor, and a taxonomy. Versão on-line do livro, 2000. Disponível em: <http://reusability.org/read/ >. Acesso em: setembro de 2008.

WENGER, E. **Artificial intelligence and tutoring systems**: Computacional and cognitive approaches to the communication of knowledge. Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1987.

WONG, D. et al. **Concordia**: An Infrastructure for Collaborating Mobile Agents. In First International Workshop on Móbile Agents, Lecture Notes in Computer Science, Vol. 1219, Springer-Verlag, Berlin, 1997.

WOOLDRIDGE, M.; JENNINGS, N; KINNY, D. **The Gaia Methodology for Agent-Oriented Analysis and Design.** Autonomous Agents and Multi-Agent Systems, 3, 2000. p. 285-312.

YAN, W. C. **Mobile Agents in E-Learning Resource Management**, 36th ASEE/IEEE Frontiers in Education Conference, San Diego, CA, 2006.

YAZDANI, M. **Artificial Intelligence and Education**: learning environments and tutoring systems. Norwood, NJ: Ablex publishing corp,v.1, 1987. 439 p.

ZAMBONELLI, F. et al. **Organizational abstractions for the Analysis and Design of Multi-Agent Systems**. In: P. Ciancarini and M. Wooldridge, editors, Agent-Oriented Software Engineering. Springer- Verlag Lecture Notes in AI Volume 1957, 2001.

_____. Agent-Oriented Software Engineering for Internet Applications". In Coordination of Internet Agents: Models, Technologies and Applications. Springer-Verlag, 2000.

Apêndices

Especificação de Casos de Uso do Tutor *On-line* e *Off-line* do FA_PorT

A. Criar estratégia

Este caso de uso responsabiliza-se pela criação de uma estratégia didática, tento como autor, o professor. O ator monta a estratégia didática a partir da utilização de táticas e recursos didáticos. Há também a possibilidade de criar-se uma estratégia personalizada, para casos onde um aluno ou grupo de alunos necessite de um ensino personalizado através de uma sessão de ensino *off-line*.

As pré-condições deste caso de uso são:

- O professor deve ter efetuado *login* no sistema.
- Existir tática cadastrada.
- Existir recurso didático cadastrado.

Os fluxos de eventos básico e alternativos são descritos a seguir:

Fluxo Básico

1. O caso de uso inicia-se quando o professor deseja criar uma estratégia.
2. O sistema exibe as táticas disponíveis.
3. O professor escolhe a tática que deverá fazer parte da estratégia didática.
4. O professor escolhe o recurso didático relacionado à tática.
5. O professor escolhe o tempo de exibição da tática.
6. O professor insere a tática à estratégia.
7. Caso necessite adicionar outra tática à estratégia, voltar ao passo 3.
8. O professor insere o nome da estratégia.
9. O professor insere a estratégia.
10. O caso de uso é encerrado.

Fluxos Alternativos

a) Criar estratégia off-line

1. Este fluxo inicia-se quando o professor deseja criar uma estratégia que será executada de forma *off-line* no computador do aluno.
2. O professor marca a opção de listagem das táticas que podem ser executadas de forma *off-line*.
3. O sistema retorna ao **Passo 2 do Fluxo Básico.**

b) Criar estratégia com a utilização da Tática de Assistência Personalizada Off-line

1. Este fluxo inicia-se quando o professor deseja criar uma estratégia didática que utiliza a tática de assistência personalizada off-line para enviar um tutor móvel ao computador do aluno.
2. O sistema exibe as táticas disponíveis.
3. Preferencialmente, o professor escolhe uma tática de regra.
4. O professor escolhe a tática de assistência personalizada *off-line*.
5. O professor insere a estratégia didática.
6. O caso de uso é encerrado.

B. Iniciar sessão on-line

Este caso de uso responsabiliza-se pela execução de uma sessão de ensino *on-line*, onde o ator deve iniciar respeitando a data e hora definida pelo professor. O Aluno é o autos envolvido nesse caso de uso.

As pré-condições deste caso de uso são:

- O aluno deve ter efetuado *login* no sistema.
- Existir sessão de ensino criada pelo professor para um grupo ao qual o aluno faz parte.

O fluxo básico para o caso de uso em questão é descrito a seguir. Observa-se que não há fluxos alternativos.

Fluxo Básico

1. O caso de uso inicia-se quando o aluno deseja iniciar uma sessão de ensino *on-line*.
2. O sistema exibe as sessões de ensino disponíveis para o aluno para a data e hora atual.
3. O aluno inicia a sessão conforme estratégia(s) criada(s) pelo professor.

4. O caso de uso é encerrado.

C. Iniciar sessão *off-line*

Este caso de uso responsabiliza-se pela execução de uma sessão de ensino *off-line*, no computador do aluno, e tem o Aluno como autor, onde este deve executar a sessão de ensino *off-line* caso o desempenho na sessão de ensino *on-line* não tenha sido satisfatório.

As pré-condições para este caso de uso são:

- O aluno deve ter participado de uma sessão de ensino *on-line*.
- O aluno deve ter tido desempenho insatisfatório na sessão *on-line* ou o aluno almeje iniciar a sessão *off-line* independentemente do seu desempenho na sessão *on-line*.
- O Tutor Móvel, o interpretador de Estratégias, a Estratégia e os Recursos Didáticos já devem estar instalados no computador do aluno.

O fluxo básico é descrito a seguir. Observa-se que não há fluxos alternativos.

Fluxo Básico

1. O caso de uso inicia-se quando o aluno deseja iniciar uma sessão de ensino *off-line*.
2. Aciona-se o ícone referente ao início da sessão de ensino *off-line*. Deve-se seguir as instruções enviadas por *e-mail* ao aluno.
3. O aluno inicia a sessão conforme estratégia(s) criada(s) pelo professor.
4. O caso de uso é encerrado.

D. Visualizar recurso didático

Este caso de uso responsabiliza-se pela visualização do recurso didático vinculado à alguma(s) tática(s) existente(s) na estratégia à qual está sendo executada na sessão de ensino.

O autor envolvido neste caso de uso é o Aluno.

Abaixo, seguem as pré-condições do caso de uso Visualizar Recurso Didático:

- O aluno deve ter efetuado *login* no sistema.
- O aluno deve estar participando de uma sessão de ensino.
- Deve haver recurso didático cadastrado.
- Deve haver recurso vinculado à tática em execução.

Os fluxos de eventos básico e alternativo são descritos a seguir:

Fluxo Básico

1. O caso de uso inicia-se quando o aluno necessita visualizar um recurso didático.
2. A tática chama o recurso didático.
3. O recurso didático é exibido por um tempo definido na tática de ensino.
4. Caso ainda haja táticas na estratégia, voltar ao **Passo 2**.
5. O caso de uso é encerrado.

Fluxo Alternativo

a) Visualizar recurso didático de uma sessão de ensino off-line

1. Este fluxo inicia-se quando o aluno necessita visualizar um recurso didático a partir de uma sessão de ensino *off-line*.
2. Aciona-se o ícone referente ao início da Sessão de Ensino *Off-line*. Deve-se seguir as instruções enviadas ao *e-mail* do aluno.
3. Seguir **Passo 2 do Fluxo Básico.**

E. Enviar *e-mail*

Este caso de uso responsabiliza-se pelo envio de *e-mail* para o aluno com instruções sobre a sessão de ensino *off-line* a ser executada no computador do aluno; e tem o Tutor *on-line* como ator.

As pré-condições para este caso de uso são:

- O aluno deve ter participado de uma sessão de ensino *on-line*.
- O aluno deve ter tido um desempenho insatisfatório na sessão de ensino *on-line* ou ter solicitado a sessão de ensino *off-line* independentemente do seu desempenho na sessão *on-line*.

Quanto ao fluxo de eventos, este caso de uso possui apenas o fluxo básico, conforme segue:

Fluxo Básico

1. O caso de uso inicia-se quando o tutor *on-line* envia um *e-mail* ao aluno.
2. O caso de uso é encerrado.

F. Enviar agente móvel

Este caso de uso responsabiliza-se pelo envio de um agente móvel para o computador de um aluno, assim, a sessão de ensino *off-line* poderá ser executada. O Tutor *on-line* é o ator envolvido no caso de uso.

Seguem as pré-condições:

- O aluno deve ter participado de uma sessão de ensino *on-line*.
- O aluno deve ter tido um desempenho insatisfatório na sessão de ensino *on-line* ou ter solicitado a sessão de ensino *off-line* independentemente do seu desempenho na sessão *on-line*.
- O ip e porta devem estar devidamente cadastrados no perfil do aluno.

Este caso de uso possui apenas o fluxo básico, segue abaixo:

Fluxo Básico

1. O caso de uso inicia-se quando o tutor *on-line* envia um agente móvel com a estratégia, o interpretador e os recursos para o computador do aluno.
2. O caso de uso é encerrado.

G. Estabelecer conexão

Este caso de uso responsabiliza-se pelo estabelecimento de conexão entre o computador de origem e destino para o envio e retorno do agente móvel, tendo os Tutores *on-line* e *off-line*(móvel) como ator.

As pré-condições para este caso de uso são:

- O aluno deve ter participado de uma sessão de ensino *on-line*.
- O aluno deve ter tido um desempenho insatisfatório na sessão de ensino *on-line* ou ter solicitado a sessão de ensino *off-line* independentemente do seu desempenho na sessão *on-line*.
- O ip e porta devem estar devidamente cadastrados no perfil do aluno.

Este caso de uso possui apenas fluxo básico, conforme segue:

FB - Fluxo Básico

1. O caso de uso inicia-se quando o tutor *on-line* estabelece conexão com o computador do aluno para o envio um agente móvel com a estratégia, o interpretador e os recursos, bem como, na conexão entre o computador do aluno e o tutor *on-line* para o retorno do agente móvel.
2. São informados ip e porta de origem e destino.
3. É informado o status da conexão: Conexão estabelecida / Erro ao conectar.
4. O caso de uso é encerrado.

H. Buscar estratégia

Este caso de uso responsabiliza-se pela busca de estratégia (XML) associada à sessão de ensino *off-line* no computador do aluno e tem o Tutor off-line/móvel como ator.

As pré-condições para este caso de uso são:

- O aluno deve ter participado de uma sessão de ensino *on-line*.
- O aluno deve ter tido um desempenho insatisfatório na sessão de ensino *on-line* ou ter solicitado a sessão de ensino *off-line* independentemente do seu desempenho na sessão *on-line*.
- A estratégia (XML) deve estar devidamente instalada na máquina do aluno.

Abaixo, segue o fluxo básico do caso de uso Buscar Estratégia:

FB - Fluxo Básico

1. O caso de uso inicia-se quando o tutor *off-line* busca, no computador do aluno, o arquivo XML relacionado à estratégia *off-line*.
2. É informado o estado da busca: Arquivo encontrado / Arquivo não encontrado.
3. O caso de uso é encerrado.

I. Acionar interpretador

Este caso de uso responsabiliza-se pelo acionamento do interpretador de estratégias com o intuito de interpretar a estratégia (XML) associada à sessão de ensino *off-line* no

computador do aluno, mostrando os recursos didáticos relacionados a cada tática da estratégia *off-line*. O ator envolvido neste caso de uso é o Tutor *off-line*/móvel.

As pré-condições relacionadas a este caso de uso são:

- O aluno deve ter participado de uma sessão de ensino *on-line*.
- O aluno deve ter tido um desempenho insatisfatório na sessão de ensino *on-line* ou ter solicitado a sessão de ensino *off-line* independentemente do seu desempenho na sessão *on-line*.
- A estratégia (XML) deve estar devidamente instalada na máquina do aluno.
- O formato do XML deve está conforme a especificação.

O fluxo básico é descrito a seguir. Observa-se que não há fluxos alternativos.

Fluxo Básico

1. O caso de uso inicia-se quando o tutor *off-line* aciona, no computador do aluno, o interpretador para o arquivo XML relacionado à estratégia didática *off-line*.
2. O recurso didático de cada tática é exibido no tempo definido pelo professor.
3. O caso de uso é encerrado.

J. Atualizar perfil do aluno

Este caso de uso responsabiliza-se pela atualização do perfil do aluno com base no desempenho do aluno na sessão *off-line* e/ou *on-line* e tem o Tutor *off-line* e o Controlador como atores..

A única pré-condição para este caso de uso é que o aluno deve ter participado de uma sessão de ensino *on-line* ou *off-line*.

Este caso de uso também não possui fluxo de evento alternativo, tendo o seguinte fluxo básico:

Fluxo Básico

1. O caso de uso inicia-se quando o tutor *on-line* ou controlador atualiza o perfil de um aluno após a execução de uma sessão de ensino e atividades.
2. O caso de uso é encerrado.

Anexos

A especificação MASIF

A especificação MASIF consiste no primeiro documento de normalização de operações em sistemas de agentes móveis visando, assim, a interoperabilidade entre sistemas de diversos construtores. Esta, define um conjunto de conceitos e interfaces (em IDL – *Interface Definition Language*), procurando a simplicidade de forma a facilitar o desenvolvimento futuro de sistemas de agentes móveis.

A. Ações comuns em sistemas de agentes móveis

Independentemente da tecnologia que o suporta, um agente móvel é constituído por duas partes: o código – que descreve o comportamento do agente – e o estado – o valor das variáveis internas. Ambos são mantidos durante a migração, o que significa que o agente não só sabe o que fazer como sabe também o que fez.

Os agentes móveis possuem um ciclo de vida composto por estados e transições bem definidas (Figura A.1).

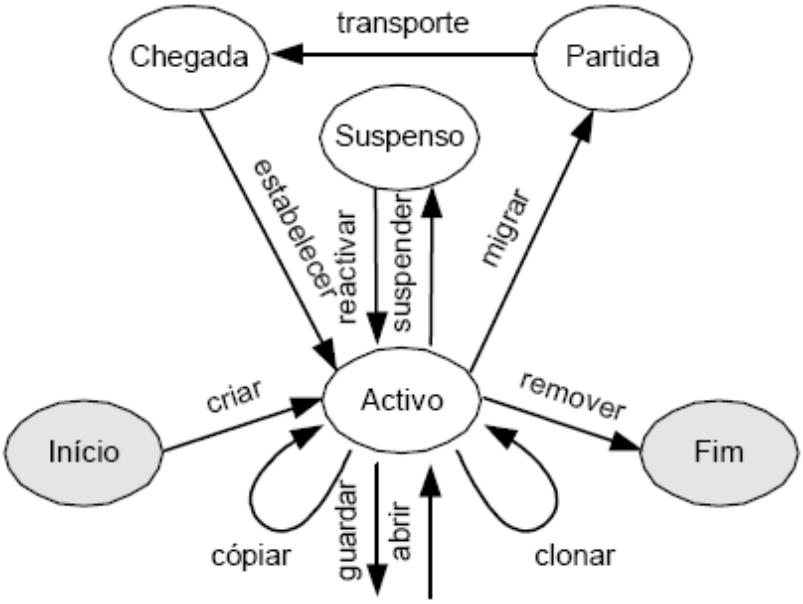


Figura A.1 - Ciclo de vida de um agente móvel

A gestão de agentes móveis atua sobre diversas fases do ciclo de vida:

- **Criar -** tarefas de inicialização tais como criar as estruturas de dados e iniciar a execução do agente. Cada agente é baseado num conjunto de classes que têm de ser enviadas à agência. Este conjunto de classes é localizado por uma cadeia de caracteres (string) - o *code base*.
- **Remover** - atividades de fim de execução.
- **Suspender** - interromper temporariamente a atividade do agente.
- **Reativar** - reiniciar a execução do agente.
- **Clonar -** criar outra instância do agente no mesmo lugar.
- **Copiar -** criar outra instância do agente em outra localização.
- **Migrar -** transferir o agente. Esta acção requer o conhecimento do destino e qual o protocolo de transporte (*sockets*, SSL, RPC ou outros).
- **Guardar -** armazenar permanentemente a informação interna do agente. Esta informação permite reiniciar um agente que tenha sido destruído inadvertidamente por alguma falha involuntária.
- **Abrir** - recupera o agente móvel do dispositivo de armazenamento permanente.
- **Invocação** - permite invocar acções definidas pelo programador.

Apesar de não estar directamente relacionadas com o ciclo de vida do agente, um sistema de agentes móveis necessita de um mecanismo de registo – a região – de forma a tornar possível a pesquisa de agentes, lugares e agências. Este mecanismo assegura a identificação unívoca no interior do domínio definido.

Por razões de segurança, nomeadamente, autenticação e controlo de acesso, uma agência identifica a autoridade que enviou o agente. É com base nesta que recursos como o sistema operativo, discos, processador, memória ou outros são protegidos contra actos ilícitos de agentes maliciosos.

B. Interfaces MASIF

A norma MASIF especifica duas interfaces em IDL que constituem a base para todas as operações sobre a agência e a região (Figura A.2).

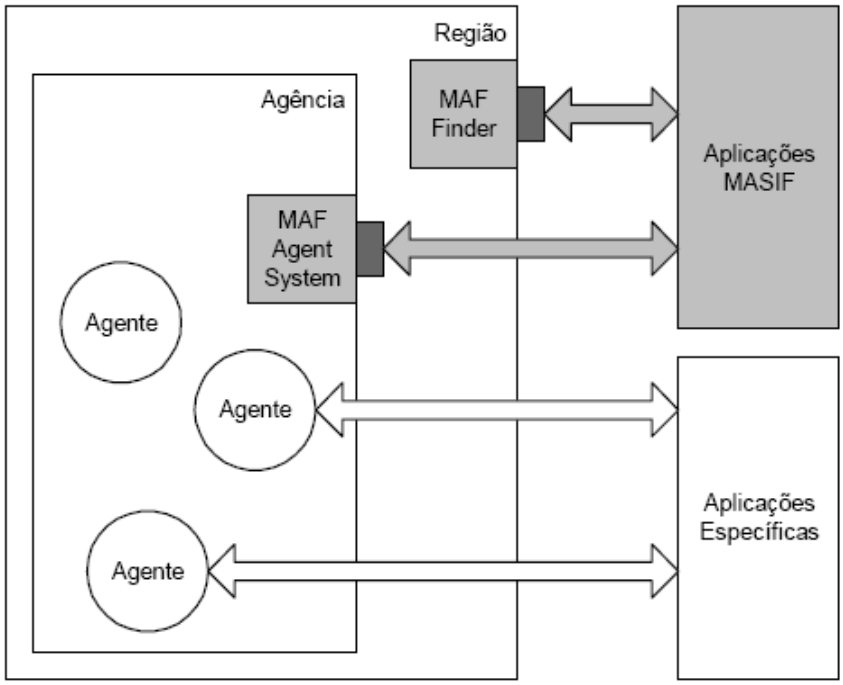


Figura A.2 - Arquitetura MASIF

A interface *MAFFinder*, ligada à região, consiste num ponto de acesso ao serviço de diretoria para agências, lugares e agentes. Os seus métodos permitem funções de pesquisa e catalogação. Já a interface *MAFAgentSystem* define métodos e objetos que suportam tarefas

como consultar o nome de uma agência ou receber um agente, entre outras. A seguir, Quadro A.1, as funcionalidades e métodos no MASIF:

Quadro A.1 - Funcionalidades e métodos no MASIF

Funcionalidade	Método

Registro	`void register_agent(…)`
	`void register_agent_system(…)`
	`void register_place(…)`
	`void unregister_agent(…)`
	`void unregister_agent_system(…)`
	`void unregister_place(…)`

Pesquisa	`Locations lookup_agent(…)`
	`Locations lookup_agent_system(…)`
	`Locations lookup_place(…)`

Ciclo de vida	`Name create_agent(…)`
	`void receive_agent(…)`
	`void resume_agent(…)`
	`void suspend_agent(…)`
	`void terminate_agent(…)`
	`OctetStrings fetch_class(…)`
	`void terminate_agent_system(…)`

Informação	`Location find_nearby_agent_system_of_profile(…)`
	`AgentStatus get_agent_status(…)`
	`AgentSystemInfo get_agent_system_info(…)`
	`AuthInfo get_authinfo(…)`
	`MAFFinder get_MAFFinder(…)`
	`NameList list_all_agents()`
	`NameList list_all_agents_of_authority(…)`
	`Locations list_all_places()`

Fonte: (OMG MASIF, 2009)

Associadas a estas interfaces encontram-se definidas várias estruturas de dados:

- ***Name*** **-** com três atributos (*authority, identity* e *agent_system_type*) que asseguram a individualidade do agente.
- ***ClassName*** - define a sintaxe para nomes das classes que, quando instanciadas, dão origem ao agente. Esta informação é utilizada nos métodos *create_agent*() e *receive_agent*(), por exemplo.
- **Location -** consiste numa sequência de caracteres contendo: a) um *URI* (*Universal Resource Identifier*) com um identificador CORBA (BERNERS-LEE, 1994) ou b) um *URL* (*Universal Resource Locator*) contendo um endereço da Internet (BERNERS-LEE; MASINTER). A vantagem do identificador CORBA é a independência do protocolo enquanto que os *URLs* são mais adequados à *Internet*.