UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

TUTA – Um Tutor Baseado em Agentes no Contexto do Ensino a Distância

Aleksandra do Socorro da Silva

Campina Grande – PB

Fevereiro de 2000

Aleksandra do Socorro da Silva

TUTA – Um Tutor Baseado em Agentes no Contexto do Ensino a Distância

Dissertação de mestrado submetida à Coordenação do Curso de Pós-Graduação em Informática - COPIN - da Universidade Federal da Paraíba - Campus II, como parte dos requisitos necessários à obtenção do grau de Mestre em Informática.

Orientador: ***Prof. Dr. Arturo Hernández-Domínguez***

Área de Concentração: ***Ciência da Computação***

Linha de Pesquisa: ***Inteligência Artificial***

Campina Grande – PB

Fevereiro de 2000

Ficha Catalográfica

SILVA, Aleksandra do Socorro da

S586T

TUTA – Um Tutor Baseado em Agentes no Contexto do Ensino a Distância

Dissertação (mestrado) – Universidade Federal da Paraiba, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande – 2000.

141 p. Il.

Orientador: Arturo Hernández-Domínguez

1. Inteligência Artificial
2. Sistemas Tutores Inteligentes
3. Educação a Distância
4. Metodologia Orientada a Agentes
5. Metodologia Orientada a Objetos

CDU – 007.52

TUTA – UM TUTOR BASEADO EM AGENTES NO CONTEXTO DO ENSINO A DISTÂNCIA

ALEKSANDRA DO SOCORRO DA SILVA

DISSERTAÇÃO APROVADA EM 28.02.2000

PROF. ARTURO HERNÁNDEZ DOMÍNGUEZ, Dr.
Orientador

PROF. EDILSON FERNEDA, Dr.
Examinador

PROF. MARCELO ALVES DE BARROS, Dr.
Examinador

PROF. EVANDRO DE BARROS COSTA, Dr.
Examinador

CAMPINA GRANDE – PB

A meus pais Maria do Carmo e Francisco, fontes da minha vida.
A minha querida tia Conceição.
A minha madrinha e mestra Srª Maria Paraense.

Agradecimentos

Sem dúvida, esta é uma parte muito importante deste trabalho, pois este só foi realizado devido ao enorme carinho e confiança que recebi dos meus amigos nos últimos tempos. Irei registrar aqui alguns agradecimentos especiais.

Em primeiro lugar, agradeço a Deus, por tudo que ele quis me ensinar, através das dificuldades e alegrias que passei e também por ter colocado pessoas tão maravilhosas no meu caminho.

A minha mãe, minha eterna incentivadora e fã, sem a qual eu não teria chegado ao fim desta etapa e ao meu pai, que ao longo de toda minha vida, tem sido o meu maior orgulho e ídolo. A minha segunda mãe: Tia Coquinha, pela dedicação e cuidados que sempre teve comigo.

Agradeço especialmente ao professor Arturo Hernández, pela paciência, pela amizade e que mesmo distante, sempre buscou estar presente. Ao professor Edilson Ferneda, sempre companheiro, amigo e disposto a ajudar. Obrigada Ferneda ! Com certeza, as universidades seriam grandiosas se contassem sempre com pessoas como você. Ao professor Marcus Sampaio, pela compreensão no início do curso e pelo voto de confiança depositado em mim.

A minhas irmãzinhas queridas: Norma, Antônia, Edna e aos meus irmãozinhos: Nego, Zuza e Val. Nego obrigada pelas correções. A todos os meus sobrinhos, ao meu tio Jucá e meu primo Roberto.

À Drª Rejane, pela amizade, pelas conversas que me fizeram acreditar que o final deste trabalho não era impossível, por seu carinho, companheirismo e presença incondicional.

À amiga Silvana, por sua presença constante em minha vida e por todo o apoio, e amizade dedicados nos últimos meses, que me permitiram terminar este trabalho.

Aos amigos que fiz na UFPB, por terem tornado Campina Grande uma cidade colorida e feliz. Em especial aos amigos Cristiany e Ricardo, por tudo que passamos juntos.

A uma grande amiga colombiana: Ivette Kafure, por suas palavras e gestos, que me permitiram ter forças.

Aos amigos e companheiros, Deni e Márcio.

Aos amigos paraenses: Lúcio Ricardo e Lucinéia Teixeira, que mesmo longe, nunca saíram do meu coração.

A minha tia (por laços afetivos) Janete, que cuidou tão bem de mim em Campina Grande. Obrigada tia !

À Adriana, pelas palavras de conforto e apoio durante as madrugadas difíceis. À Patrícia, por nossas conversas filosóficas e por sua amizade.

À amiga Lamarka, que mesmo longe, está presente neste trabalho.

À amiga Diana, pelo apoio e pelos momentos de descontração.

Especialmente, sem exagero nenhum, às melhores secretárias de pós-graduação do Brasil: Aninha e Vera.

Aos funcionários e sobretudo, amigos da MINIBLIO: Josirene, Emanuela, Zeneide e a Arnaldo.

À CAPES, pelo apoio financeiro.

Sempre que um trabalho científico apresenta alguma informação, ele vem acompanhado de uma margem de erro – um silencioso, mas insistente lembrete que conhecimento algum é completo ou perfeito (Carl Sagan).

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Abreviaturas

ACVA - Arquitetura de uma Classe Virtual Adaptativa

AME-A - Ambiente Multiagente de Ensino-Aprendizagem

CAI - Instrução Auxiliada por Computador (do inglês: *Computer Assisted Instruction*)

CSCL - Aprendizagem Cooperativa Apoiada por Computador (do inglês: *Computer Supported Cooperative Learning*)

CSCW - Trabalho Cooperativo Apoiado por Computador (do inglês: *Computer Supported Cooperative Work*)

DLE - Ambiente de Ensino Distribuído (do inglês: *Distributed Learning Enviroment*)

IA - Inteligência Artificial

IAD - Inteligência Artificial Distribuída

ICAI - Instrução Inteligente Auxiliada por Computador (do inglês: *Computer Assisted Instruction*)

IE - Informática na Educação

ILE - Ambientes Interativos/Inteligentes de Aprendizagem (do inglês: *Interactive/Intelligent Learning Enviroment*)

LOO - Linguagens Orientada a Objetos

MIT - *Masachusets Institute of Technology*

MCOE - *Multiagent Cooperative Enviroment*

MOA - Modelagem Orientada a Agentes

MOO - Modelagem orientada a Objetos

AO - Orientação a Agentes

OO - Orientação a Objetos

RDP - Resolução Distribuída de Problemas

SBC - Sistemas Baseados em Conhecimento

SEI - Sistema de Ensino Inteligente

SF_Básicos - Serviços de Formação Básicos

SF_CV - Serviços de Formação da Classe Virtual

SF_Grupo - Serviços de Formação de Grupo

SMA - Sistemas Multiagentes

STI - Sistema Tutor Inteligente

TUTA - Tutor Baseado em Agentes

www - *World Wide Web*

Resumo

Este trabalho consiste no desenvolvimento de um Sistema Tutor Baseado em Agentes (TUTA) no contexto de uma Classe Virtual Adaptativa. O TUTA atende um grupo de alunos geograficamente distantes e permite o aprendizado de conceitos (exemplo: o domínio da orientação a objetos). O funcionamento do tutor é determinado pelo módulo de estratégias (multi-estratégias) que permite a interpretação de qualquer estratégia definida pelo professor (armazenada em arquivo). Uma estratégia é representada via um conjunto de primitivas básicas que permitem a definição de um algoritmo-estratégia e o Módulo Didático é representado por um Servidor de Objetos Didáticos, que está associado aos objetos do domínio.

Abstract

This work consists of the development of a Based Tutorial System in Agents (TUTA) in the context of a Virtual Classroom Adaptable. TUTA attend a group of geographically distant students and allows to the learning of concepts (example: the domain of the object-oriented). The functioning of the tutor is determined by the module of strategies (multi-strategies) that it allows the interpretation of any strategy defined through teacher (stored in archive). A strategy is represented by a set of basic primitive that allow to the definition of a algorithm-strategy and Didactic Module is represented by Didactic Object Server, who is associated with objects of the domain.

Capítulo 1
Introdução

Este capítulo apresenta as motivações gerais deste trabalho; o contexto em que ele está inserido; seus objetivos; e sua organização.

1.1 MOTIVAÇÕES

A dificuldade de manter-se bem qualificado e atualizado profissionalmente é uma realidade nas mais diversas áreas de atuação do mercado profissional. Atualmente, existem muitas formas de atualização, como: livros, revistas, artigos, *sites*, palestras, congressos, aulas, entre outros, em que o objetivo básico é informar e atualizar os profissionais. Em alguns desses casos, há necessidade do deslocamento físico até as fontes de informações necessárias, a fim de se obter a formação desejada. Mas, em algumas situações, devido às limitações de tempo e distância, torna-se inviável que ocorra tal deslocamento.

Neste contexto, aparece o ensino a distância, como uma necessidade de flexibilizar o ensino convencional: flexibilidades estas, relativas ao tempo e à distância. Assim, o ensino a distância pode representar um complemento (e não apenas uma alternativa) ao ensino presencial. Nos últimos tempos novas perspectivas de aplicações de ensino a distância surgiram através da Internet, em oposição aos sistemas de ensino a distância tradicionais (que utilizam tecnologias unidirecionais, como: os correios e canais de televisão/rádio). A Internet, ao contrário, representa um meio bidirecional de comunicação, capaz de disponibilizar um conjunto de ferramentas de apoio ao ensino-aprendizagem (lista de discussão, *"chats"*, correio eletrônico, hipertexto, entre outros).

A necessidade de troca de informações e integração entre os indivíduos ao realizar suas atividades computacionais foi de vital importância para o surgimento de ambientes que dão suporte ao trabalho cooperativo. Tais ambientes passaram a fazer parte de uma área de pesquisa intitulada "***CSCW"*** (do inglês: *Computer-Supported Cooperative Work* - ***CSCW***) (Kling, 1991), que significa Trabalho Cooperativo Suportado por Computador ou Suporte por Computador ao Trabalho Cooperativo. Essa área de pesquisa é divulgada entre a comunidade de Informática na Educação (**IE**) como: Aprendizagem Cooperativa Apoiada por Computador (do inglês: *Computer-Supported Cooperative Learning* – ***CSCL***) e tem a finalidade de apoiar os processos de aprendizagem que necessitam de ações colaborativas/cooperativas[1] entre os

[1] Conforme (Giraffa, 1999) existe uma significativa diferença entre o termo colaboração e cooperação: "Colaboração é compartilhar informação sem modificar a informação recebida. Cooperação é compartilhar informação e poder interferir e modificar a informação recebida, e atuar de forma conjunta para construir algo em comum. Toda a cooperação envolve colaboração".

indivíduos. Novamente neste âmbito a Internet torna-se cada vez mais familiar, oferecendo ferramentas de trabalho cooperativo a distância. Conforme (Hernández-Domínguez, 1995), os ambientes providos de suporte para que as pessoas possam interagir cooperativamente podem reunir pessoas (alunos e professores) em atividades de aprendizagem cooperativa em uma espécie de *Classe*[2] *Virtual*.

Além da importância do cooperativismo, outro objetivo perseguido pelos ambientes de aprendizagem por computador está relacionado à necessidade de facilitar a aprendizagem dos alunos. Assim pode-se dizer que, uma boa resposta para alcançar esse objetivo consiste na adaptação de um tutor artificial a um dado aluno humano em uma situação de aprendizagem. Mas alcançar esta adaptabilidade através do computador não é uma tarefa fácil, que é perseguida por alguns tipos de sistemas denominados *"Sistemas Tutores Inteligentes"* (Wenger, 1987*)*. Existem duas abordagens básicas para o desenvolvimento de um Sistema Tutor Inteligente (**STI**) (Giraffa, 1999): a abordagem tradicional e a abordagem baseada em agentes. Agentes representam um novo paradigma na área de desenvolvimento de aplicações de *software* (Wooldridge & Jennings, 1998) e têm representado um forte aliado no desenvolvimento de STI´s, inclusive facilitando a adaptabilidade de tais sistemas aos alunos, dentre outros fatores vantajosos na sua utilização[3].

1.2 CONTEXTO DA DISSERTAÇÃO

Neste trabalho apresenta-se o TUTA: um **Tut**or Baseado em **A**gentes, que leva em conta os princípios e módulos de um Sistema Tutor Inteligente[4] (Módulo Especialista, Módulo do Comportamento do Aluno, Módulo de Estratégias de Ensino, Módulo de Interface e Módulo de Controle) (Wenger, 1987), considerando que tais módulos são representados por Agentes.

O TUTA, situa-se no contexto do ensino a distância e pode ser utilizado por um professor para auxiliá-lo no ensino de conceitos (exemplo usado neste trabalho: o domínio da orientação a objetos) para um grupo de alunos geograficamente distantes (via Internet). Esse tutor está inserido no contexto da Arquitetura de uma Classe Virtual Adaptativa (**ACVA**) (Hernández, 1997), (Hernández, 1998). A ACVA considera que uma Classe Virtual é composta de diversos grupos heterogêneos e ainda que dentro de cada grupo existem diferentes alunos com diversos níveis de conhecimento. Esses níveis de conhecimento em um grupo são tratados como *zonas de comportamento*. Exemplo: Grupo 1 e Grupo 2, com as seguintes zonas de comportamento: zona de comportamento 1, zona de comportamento 2, zona de comportamento 3, zona de comportamento 4, zona de comportamento 5. Assim, o objetivo da

[2] Usa-se o termo *Classe* equivalente ao termo Aula.
[3] Aspectos relacionados ao uso de agentes em STI´s serão discutidos no Capítulo 3 desta dissertação.
[4] Os módulos de um STI serão apresentados no Capítulo 2. Cf. seção 2.3.1

ACVA é permitir a adaptabilidade dos alunos aos diferentes grupos, bem como a adaptabilidade deles às diferentes zonas de comportamento dentro dos grupos. Nesse contexto, o TUTA permite a formação de um determinado grupo de alunos, realizando a adaptação deste a suas diferentes zonas de comportamento.

1.3 OBJETIVOS DA DISSERTAÇÃO

O objetivo geral desta dissertação é apresentar o sistema tutor artificial denominado TUTA. Alguns objetivos específicos que podem ser destacados para alcançar esta tarefa são os seguintes:

- Definir os requisitos do tutor (TUTA), no contexto da ACVA;
- Realizar a modelagem, considerando o enfoque baseado em agentes do tutor, ou seja, utilizar técnicas específicas para a modelagem de agentes, acopladas à técnicas de modelagem orientadas a objetos; e
- Propor a implementação de uma aplicação, utilizando o TUTA.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação está estruturada em 7 capítulos, sendo esse o primeiro capítulo. Os demais estão organizados como segue abaixo:

No Capítulo 2 (Sistemas Tutores Inteligentes), discute-se inicialmente o uso de computadores na educação e treinamento. Apresentam-se alguns tipos de ambientes de ensino-aprendizagem por computador. A seguir são feitas considerações a respeito de sistemas tutores inteligentes (arquitetura, aspectos pedagógicos envolvidos, bem como algumas de suas tendências e dificuldades).

No Capítulo 3 (Sistemas Tutores Inteligentes baseados em Agentes), apresentam-se inicialmente aspectos relacionados à tecnologia de agentes e Inteligência Artificial Distribuída (IAD). Posteriormente são feitas considerações acerca do uso de agentes no desenvolvimento de sistemas tutores inteligentes e são apresentados alguns exemplos.

No Capítulo 4 (O TUTA), apresenta-se inicialmente a ACVA (arquitetura em que este trabalho faz parte), contextualizando o TUTA em tal arquitetura. Posteriormente apresentam-se: os requisitos do TUTA, sua arquitetura, seu funcionamento geral e possíveis aplicações.

No Capítulo 5 (Modelagem do TUTA), apresenta-se o processo de desenvolvimento utilizado neste trabalho; discute a necessidade de uma metodologia orientada a agentes. Poste-

riormente, apresenta-se a modelagem do TUTA orientada a agentes; a sua arquitetura e finalmente realiza-se outra etapa de modelagem: a modelagem orientada a objetos

No Capítulo 6 (Implementação do TUTA), apresentam-se os principais aspectos de implementação do TUTA, em que são mostrados os aspectos relevantes em relação a linguagem utilizada, bem como os aspectos dos agentes do sistema, suas características, formas de distribuição e comunicação. Posteriormente, apresenta-se o funcionamento dos agentes no TUTA e os resultados da implementação de uma aplicação.

No Capítulo 7 (Conclusões), apresentam-se os resultados obtidos nesta dissertação e as perspectivas para futuras pesquisas que podem ser realizadas a partir dos resultados obtidos.

Capítulo 2
Sistemas Tutores Inteligentes

Este capítulo discute o uso dos computadores na educação; apresenta alguns ambientes de ensino-aprendizagem por computador considerados relevantes no desenvolvimento deste trabalho e enfoca os Sistemas Tutores Inteligentes: sua arquitetura, aspectos pedagógicos envolvidos, bem como as tendências e dificuldades encontradas no seu desenvolvimento.

2.1 COMPUTADORES NA EDUCAÇÃO

A possibilidade de uso de computadores na educação é uma idéia bastante antiga e tem se tornado cada vez mais presente. Entretanto, antes de utilizar o computador no processo educativo, é necessário definir o que se pretende com o seu uso, a fim de associar isto a uma proposta pedagógica, para então seguir um ou vários caminhos possíveis. Existem vários papéis exercidos pelo computador no processo educativo e várias justificativas. Entretanto, um fator muito importante, senão um dos mais importantes a considerar, é quanto ao papel exercido pelo computador como meio de ensino-aprendizagem.

A introdução do computador na educação não deve ser considerada apenas como uma simples mudança tecnológica, mas deve estar relacionada a fatores mais importantes, como a mudança da maneira de aprender, a possibilidade de interações entre os envolvidos no processo de ensino-aprendizagem, entre outros motivos.

2.2 AMBIENTES DE ENSINO-APRENDIZAGEM POR COMPUTADOR

Observa-se, na literatura da área, que existem várias taxionomias para os ambientes educacionais. Desta forma, torna-se difícil conseguir um perfeito levantamento deles. Uma das taxionomias mais tradicionais e citadas na área de Informática na Educação é a de Taylor (Taylor, 1980), que classifica os ambientes de aprendizagem em: tutor, tutelado e ferramenta, conforme a sua forma de utilização. Outras taxionomias consideradas neste trabalho foram as propostas por Costa (Costa, E. B., 1997) e por Giraffa (Giraffa, 1999). Costa realiza uma abordagem histórico-taxonômica dos ambientes de ensino-aprendizagem por computador, procurando enquadrá-los em três momentos cronologicamente distintos: 1º momento (que abrange os ambientes surgidos até meados da década de 70), 2º momento (abrange os ambientes surgidos aproximadamente a partir da segunda metade da década de 70 até o final da década de 80) e 3º momento (abrange os ambientes iniciados na década de 90). Outra taxionomia proposta por (Giraffa, 1999), consiste em classificar os ambientes conforme o tipo de aprendizagem proporcionado por eles, classificando-os em dois grandes grupos: ambientes onde a aprendiza

gem do aluno está centrada na aquisição de habilidades específicas, como os Sistemas do tipo ***CAI*** (do inglês: *Computer Assisted Instruction*) e ***ICAI*** (do inglês: *Intelligent Computer Assisted Instruction*); e ambientes que visam a aprendizagem de habilidades cognitivas amplas, como os Micromundos e Sistemas de autoria, entre outros.

Neste capítulo, procura-se apresentar alguns tipos de Ambientes de Ensino-Aprendizagem considerados relevantes no contexto deste trabalho.

2.2.1 SISTEMAS DE INSTRUÇÃO AUXILIADA POR COMPUTADOR

Uma das primeiras categorias de *software* educacionais que surgiram no início de 1960 foram os sistemas de Instrução Auxiliada por Computador, que utilizam a teoria comportamentalista (*behaviorista*) proposta por Skinner (Skinner, 1958) como modelo de ensino e aprendizagem. Tais sistemas são citados por muitos autores como *sistemas de instrução programada*.

Os sistemas do tipo *CAI* representam o conhecimento de um professor especialista em um certo domínio e possuem um conjunto de estratégias pedagógicas[5] fixo para aplicar ao aluno, ou seja, todas as decisões de como o sistema irá funcionar são tomadas antecipadamente, durante o projeto do sistema, não havendo possibilidades de que o sistema se adapte ao aluno em tempo de execução.

Algumas modalidades de sistemas deste tipo são (Giraffa, 1999): *Software* do tipo "exercício e prática" e "tutoriais", entre outros.

2.2.2 *MICROMUNDOS*

Os *Micromundos* (Papert, 1987) surgiram na década de 60 e foram propostos inicialmente por *Seymour Papert* e seu grupo de pesquisa no *Masachusets Institute of Technology* ***(MIT)***. O objetivo dos *Micromundos* é auxiliar a aprendizagem através de descoberta e exploração. Para isso possui um conjunto de ferramentas de exploração, com as quais o aluno pode manipular o ambiente e assim construir o seu conhecimento. Os *Micromundos* representam uma proposta pedagógica contrária aos *CAI's*, já que a ênfase está na aprendizagem cognitiva do aluno. Um exemplo de *Micromundo* é o ambiente LOGO, desenvolvido por Papert (Papert, 1985), que possui vários recursos como um *Micromundo* gráfico, composto por um objeto representado por uma tartaruga, que é controlado através de primitivas de uma linguagem de programação que fazem parte do ambiente.

[5] Estratégias Didáticas determinam como o assunto será apresentado ao aluno. Isso, será discutido na seção 2.3.1.3.

2.2.3 JOGOS EDUCACIONAIS

Jogos Educacionais representam uma importante modalidade de *software* educacional, permitindo que os alunos desenvolvam atividades perceptivas, sensório-motoras, verbais, entre outras (Giraffa, 1999).

2.2.4 SIMULADORES

Os simuladores permitem que o aprendiz possa simular situações como se estivesse em um mundo real. Assim, é possível que o aluno desenvolva hipóteses, teste-as e analise os seus resultados. E isso pode se repetir até que o aluno decida parar. Um exemplo comum é um simulador de vôos composto por vários recursos, no qual os aprendizes de pilotos podem treinar várias acrobacias e realizar vôos sob os diversos tipos de situações sem envolver grandes danos (Costa, E. B., 1997).

2.2.5 SISTEMAS INTELIGENTES DE INSTRUÇÃO AUXILIADA POR COMPUTADOR

Os sistemas *ICAI's* surgiram nos meados da década de 70, com a tentativa de superar as limitações impostas pelos sistemas *CAI's*. Para isso, passaram a incorporar recursos da Inteligência Artificial (IA) e da Psicologia Cognitiva (Costa, E. B., 1997). Sistemas *ICAI´s* passaram a ser conhecidos por muitos autores como sinônimo de Sistemas Tutores Inteligentes (do inglês: *Intelligent Tutoring Systems - ITS*) (Wenger, 1987). Neste trabalho, sistemas do tipo *ICAI* serão utilizados também como sinônimo de STI´s. Posteriormente, serão discutidos alguns aspectos relacionados aos STI´s[6].

2.2.6 TELEMÁTICA EDUCACIONAL

A Telemática Educacional refere-se à utilização do computador em educação através de redes de computadores.

Um projeto bastante conhecido é o *Kidlink*[7]. *Kidlink* é uma organização cujo objetivo é envolver o maior número possível de jovens até a idade de 15 (quinze) anos em um diálogo global. Seu principal meio de comunicação é o eletrônico (*e-mail*), mas permite também interações em tempo-real (como "*chats*"). Outros meios utilizados, são por exemplo: fax, vídeo conferência e rádio.

Outro exemplo de aplicação da área (Telemática Educacional) é o projeto ACVA (um dos contextos em que este trabalho está inserido).

---

[6] cf. seção 2.3.

[7] As informações citadas a respeito do projeto *Kidlink* estão disponíveis em *http://www.kidlink.org*.

2.2.7 ROBÓTICA EDUCACIONAL

A Robótica Educacional ou Pedagógica proporciona ambientes de aprendizagem baseados em dispositivos robóticos que permitem a construção do conhecimento em diferentes áreas de conhecimento (D´Abreu, 1991). Um ambiente bastante difundido é o LEGO-Logo[8]. O LEGO-Logo consiste de um conjunto de peças LEGO que permitem a montagem de objetos (máquinas e animais) e de um conjunto de comandos da linguagem de programação Logo. Através do Logo, é possível elaborar programas para controlar os objetos LEGO. Uma das formas de comunicação entre o objeto LEGO e o computador é por intermédio de uma interface eletrônica.

Uma possível aplicação do LEGO-logo é na disciplina de Matemática (Cristóvão, 1997), onde os alunos constróem um carro e fazem com que ele suba uma rampa apoiada numa escada. Em seguida, mudam a inclinação desta rampa até que o carro não consiga mais subir. Feito isso, os alunos medem os lados deste triângulo, formado pela rampa, chão e altura da escada e aplicam então o *Teorema de Pitágoras* para verificar se o triângulo é retângulo. Caso ele não seja retângulo, então usam a Trigonometria para calcular o ângulo formado pela rampa com o chão e este é o ângulo máximo que o carro conseguiu suportar na subida.

2.2.8 AMBIENTES INTERATIVOS/INTELIGENTES DE APRENDIZAGEM

Os Ambientes Interativos/Inteligentes de Aprendizagem (do inglês: *Interactive/Intelligent Learning Environment* - ***ILE***) podem ser considerados uma evolução dos STI´s tradicionais e também são conhecidos como Sistemas Tutores Cooperativos. Ambientes interativos/inteligentes de aprendizagem caracterizam-se por considerar mais de um aluno ou mais de um tutor trabalhando no mesmo ambiente (Giraffa, 1999). Esses ambientes combinam aspectos das categorias de STI´s e *Micromundo*, onde a aprendizagem ocorre através de descoberta.

2.2.9 AMBIENTES DE ENSINO-APRENDIZAGEM A DISTÂNCIA VIA INTERNET

A utilização da Internet como meio de comunicação para a Educação a Distância é uma realidade que tem produzido diferentes modalidades de Ambientes de Ensino-Aprendizagem. Conforme (Santos, 1999), tais ambientes podem ser reunidos em seis modalidades:

- Aplicações Hipermídia para fornecer instrução distribuída: nesta modalidade, encontram-se os cursos multimídias com objetivos educacionais definidos, tarefas a serem realizadas pelos alunos, formas de avaliação e suporte para comunicação com os pares

[8] LEGO-Logo: É resultado do trabalho conjunto de um grupo de pesquisadores do *"Epistemology and Learning Group"* do *Massachusetts Institute of Technology* e da indústria dinamarquesa LEGO.

e com o professor; cursos no formato hipertexto, compostos de páginas na *www*, seguindo o modelo de livro-texto.

- *Sites* educacionais: nesta modalidade estão reunidas diferentes formas de apoio ao trabalho docente e ao aprendizado autônomo dos alunos. Alguns exemplos são (Santos, 1999): *Study Web*, *The Internet Public Library*, *The World Lecture Hall*, *Escolanet* e *Projeto Aprendiz*.

- Sistemas de Autoria para cursos a distância: esta modalidade permite que professores possam criar os seus próprios cursos na *www*, utilizando os diversos recursos disponibilizadas por tais sistemas. Alguns sistemas bastantes difundidos são (Santos, 1999): *LearningSpace*, *TopClass*, *Virtual-U* e *WebCT*.

- Salas de Aulas Virtuais: esta modalidade surgiu como uma forma de facilitar a passagem gradual de professores e alunos da sala de aula presencial para a sala de aula virtual, e para isso são dotadas de espaços de comunicação e cooperação para seus participantes. Alguns exemplos de Salas de Aula Virtuais são (Santos, 1999): *Classe Virtual* e *AulaNet*.

- *Frameworks* para aprendizagem cooperativa: *frameworks* permitem o desenvolvimento de ambientes customizáveis, integrando ferramentas disponíveis. Alguns exemplos de *frameworks* são (Santos, 1999): *Habanero* e *Promondia*.

- Ambientes Distribuídos para Aprendizagem Cooperativa: esta é outra modalidade de Ambientes de Ensino-Aprendizagem que tem crescido bastante. Alguns exemplos são (Santos, 1999): *QSabe* e *WebSaber*.

2.3 SISTEMAS TUTORES INTELIGENTES

Sistemas Tutores Inteligentes são sistemas voltados ao ensino que buscam 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 ou de ensino*[9]*)*. Métodos e técnicas de Inteligência Artificial podem ser utilizadas para proporcionar uma maior adaptação do sistema ao aluno.

[9] Neste trabalho, os termos estratégias de ensino, estratégias didáticas, estratégias pedagógicas ou estratégias de ensino-aprendizagem são utilizados indiferentemente.

2.3.1 ARQUITETURA BÁSICA

As arquiteturas de Sistemas Tutores Inteligentes variam de uma implementação para outra. Entretanto, existem alguns componentes ou módulos básicos que podem ser observados em diversos trabalhos (Wenger, 1987), (Yazdani, 1987), (Barr & Feigenbaum, 1982) e na maioria dos casos sob diversas denominações, mas que possuem o mesmo propósito básico. Esses componentes baseados em (Wenger, 1987), (Giraffa, 1999) são demonstrados na figura 2.1.

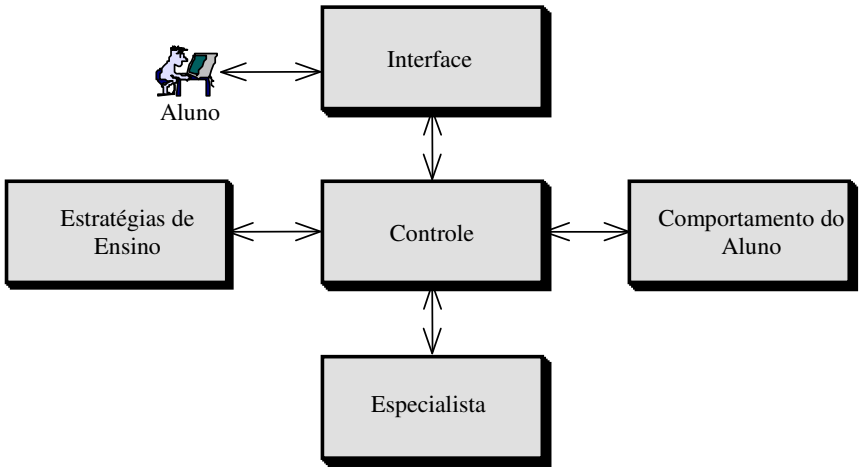


Figura 2.1: Arquitetura básica de Sistemas Tutores Inteligentes

2.3.1.1 Especialista

O módulo especialista é responsável por manipular o conteúdo que vai ser ensinado pelo STI. Nele podem ser encontrados: o material instrucional, mecanismos de geração de exemplos, formulação de diagnósticos e os processos de simulação que podem existir nos STI´s. 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 fatores que diferencia um STI de um CAI convencional é o fato do conteúdo ser 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.

2.3.1.2 Comportamento do Aluno

O módulo do comportamento do aluno deve captar o estado do entendimento do aluno a respeito do assunto que está sendo apresentado (Wenger, 1987). Os dados que fazem parte deste módulo são de fundamental importância para que o STI 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.

2.3.1.3 Estratégias de Ensino

O módulo de Estratégias de Ensino, também chamado de *módulo tutor*, *módulo instrutor, módulo pedagógico* ou ainda *módulo de estratégias didáticas*, deve ser capaz de tomar as decisões sobre as estratégias de ensino a serem utilizadas e determinar as informações que serão apresentadas a um aluno, de forma semelhante a um professor. Pode-se dizer que o comportamento de um STI é determinado diretamente por este módulo.

Segundo (Ohlsson, 1987), o STI deve ter dois níveis de planejamento: estratégias pedagógicas 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. Desta forma, pode-se concluir que*: uma estratégia didática particular é composta de um conjunto de táticas (ou ações) para realizá-la*.

Existem diversos métodos de estratégias que podem ser utilizados em STI´s. Alguns deles são os seguintes (Giraffa, 1999):

- Método Socrático: esse método permite que o STI ensine através de perguntas e diálogos, levando o aluno a tirar suas próprias conclusões.

- Método Colaborativo (Assistente): esse método faz com que o STI comporte-se como um colaborador na interação com o aluno, ajudando-o a esclarecer suas idéias.

- Método *Coaching* (Treinamento): esse método faz com que o STI observe o desempenho do aluno, a fim de aconselhá-lo nas realizações de suas atividades.

Em outro nível, as táticas que podem ser utilizadas em estratégias de ensino são as seguintes (Pereira *et al*., 1998): mostrar um exemplo usando uma situação similar; mostrar uma mensagem com a melhor opção; mostrar exemplos relacionados sem nenhuma explicação; mostrar temas que são importantes prestar mais atenção; mostrar o conteúdo de cada tópico; mostrar uma mensagem explicando as conseqüências de suas ações; mostrar ao aluno sucessi-

vas questões para que ele possa analisar hipóteses, descobrir contradições e realizar inferências corretas.

O método utilizado no STI é que deve determinar quais táticas devem ser utilizadas em uma estratégia didática. Entretanto, caso o projetista do STI não esteja adotando nenhum método específico, ele deverá escolher o conjunto de táticas para compor uma estratégia que melhor satisfazem os objetivos do STI.

É importante ressaltar que muitos dos STI´s citados na literatura têm mais que uma estratégia didática e 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.

2.3.1.4 Interface

O módulo de Interface é responsável pela comunicação entre o aluno e o STI. Através da Interface, o STI pode apresentar o seu material instrucional e monitorar o progresso do aluno pela recepção de suas respostas.

Alguns autores consideram que uma boa interface é vital para o sucesso de qualquer sistema interativo. Em STI´s, pode-se dizer que a questão da interação cresce de importância, já que se deve proporcionar ao aluno uma interação o mais amigável possível. Por isso, muitos desenvolvedores de STI´s optam pelo desenvolvimento de Interfaces Adaptativas.

2.3.1.5 Controle

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

2.4 ASPECTOS QUE DEVEM SER CONSIDERADOS EM SISTEMAS TUTORES INTELIGENTES

Sistemas Tutores Inteligentes que incorporam técnicas variadas de IA podem ser considerados tecnicamente bem projetados e implementados. Entretanto, nos últimos tempos, uma preocupação que tem sido considerada constante está relacionada a adequação pedagógica dos mesmos. É importante ressaltar que este fator estende-se a qualquer *software* com propósitos educacionais e não apenas aos STI´s.

É interessante considerar que para que um *software* educacional seja efetivo e realize os seus objetivos pedagógicos é necessário que o seu desenvolvimento reúna especialistas das áreas de Educação e Informática. Desta forma, para que possa existir este trabalho interdisciplinar, os profissionais que se dedicam ao desenvolvimento desse tipo de *software*, devem buscar alguns referenciais teóricos que fundamentem o seu trabalho. Esses referenciais teóricos têm a sua origem nas áreas de Educação e de Psicologia e estão relacionados às teorias de aprendizagem. A importância dessas teorias é que os procedimentos e princípios de ensino que estão embutidos no conjunto de táticas e estratégias de ensino de um *software* educacional, estão diretamente relacionados a elas.

Não é objetivo deste trabalho explorar as teorias psicológicas e pedagógicas a partir de seus autores e sim compreender algumas tendências relacionadas ao desenvolvimento de STI´s. De modo geral, as teorias de ensino-aprendizagem envolvidas no projeto de STI´s, estão relacionadas a determinadas abordagens pedagógicas. Algumas das abordagens que podem ser encontradas em (Giraffa, 1995) e consideradas neste trabalho são as seguintes: comportamentalista *(behaviorista)*, baseada principalmente nos trabalhos de *Burrhus Frederic Skinner*; humanista, baseada nos trabalhos de *Carls Ranson Rogers*; cognitivista, baseada nos trabalhos de *Jean Piaget* e *Jerome Seymour Brunner* e sócio-cultural, baseada principalmente nos trabalhos de *Paulo Freire* (no âmbito do Brasil).

Assim, se o *software* educacional desenvolvido segue uma das diferentes abordagens pedagógicas, qualquer que seja ela, é necessário que se tenha o conhecimento necessário desta, a fim de compreender melhor a abrangência, as limitações e as aplicações que o *software* pode atender. Conforme (Giraffa, 1995), alguns quadros comparativos (Quadros 2.1 e 2.2) podem ser utilizados para uma melhor compreensão destas abordagens:

Abordagem	Conhecimento	Ensino-aprendizagem
Comportamentalista	Consiste na forma de se ordenar as experiências e os eventos do universo, colocando-os em códigos simbólicos.	Uma mudança relativamente permanente em uma tendência comportamental e/ou na vida mental do indivíduo, resultantes de uma prática reforçada.
Humanista	A experiência pessoal e subjetiva é o fundamento sobre o qual o conhecimento é construído, no decorrer da vida da pessoa.	Ensino centrado na pessoa (primado no sujeito, no aluno).
Cognitivista	É uma construção contínua caracterizada por formação de novas estruturas que não existiam anteriormente no indivíduo.	Assimilar o objeto e associá-lo à esquemas mentais. Baseado no ensaio-erro, na pesquisa, na investigação e na solução de problemas.
Sócio-cultural	Criado e elaborado a partir do mútuo condicionamento, pensamento e prática.	Educação problematizadora ou conscientizadora

Quadro 2.1: Abordagem x Conhecimento x Ensino-aprendizagem

Abordagem	Professor-aluno	Metodologia
Comportamentalista	Relação vertical, centrada no professor.	Aula expositiva. Demonstrações do professor à classe. Aplicação da tecnologia educacional, especialmente os módulos instrucionais.
Humanista	O professor assume papel de facilitador da aprendizagem.	As estratégias e técnicas de ensino assumem importância secundária. Relação pedagógica com clima favorável ao desenvolvimento da pessoa e com liberdade para aprender.
Cognitivista	Cabe ao professor evitar rotina, fixação de respostas, hábitos. Deve propor problemas aos alunos, sem ensinar-lhes as repostas. Sua função consiste em provocar desequilíbrios, fazer desafios.	Não existe modelo pedagógico *piagetiano*. As atividades principais seriam: jogos de pensamento para o corpo e sentidos, jogos de pensamento lógico, atividades sociais para o pensamento (teatro, excursões), ler e escrever, aritmética, ciência, arte e ofícios, música e educação física.
Sócio-cultural	A relação professor-aluno é horizontal e não imposta.	Método de alfabetização que permite que alunos e professores utilizem elementos que realizam uma distanciamento do objeto cognoscível.

Quadro 2.2: Abordagem x Professor-aluno x Metodologia

2.5 TENDÊNCIAS DE PESQUISA EM SISTEMAS TUTORES INTELIGENTES

A área de pesquisa relacionada aos Sistemas Tutores Inteligentes está ainda em fase de estruturação. Desta forma, ainda existem muitas tendências dentro da área que prometem ser promissoras pelos próximos anos, bem como outras que já se estruturaram, como por exemplo: a utilização de STI´s na área de treinamento, uma vez que sistemas de treinamento tornaram-se cada vez mais necessários, por permitirem aos usuários a aquisição de perícias em determinados tópicos de um domínio (geralmente complexos e dinâmicos). Dentre algumas linhas de pesquisa que ainda se configuram como tendências de pesquisa para os próximos anos, pode-se citar as seguintes [10] (Marietto *et al*., 1997):

2.5.1 SISTEMAS TUTORES INTELIGENTES EM AMBIENTES DE REDE

Conforme (Silveira, 1996 *apud* Giraffa, 1999), um importante passo para a disseminação dos ambientes educacionais foi representado pelo desenvolvimento da tecnologia de redes, pelo aperfeiçoamento dos meios de comunicação, protocolos e técnicas de processamento distribuído.

Neste contexto, a Internet e seus diversos serviços representam uma facilitadora para o processo de ensino-aprendizagem por computador e uma das vantagens de que STI´s sejam utilizados na Internet é que eles podem ser acessados a partir de diferentes pontos geográficos, em diferentes tempos. Essas novas tecnologias também permitiram que a pesquisa na área de Ambientes de Ensino Distribuídos (do inglês: *Distributed Learning Enviroment* - ***DLE***) evoluísse rapidamente.

Alguns exemplos de STI´s utilizados na Internet (pela *world wide web* - *www*), são apresentados a seguir (Giraffa, 1997):

- *LEAP (Learn Explore And Practice):* O projeto *LEAP* é resultante do esforço cooperativo entre a *US WEST Advanced Technologies, Learning Systems* e *Mass Markets* para desenvolver uma plataforma de STI 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.

- *ELM-ART*: O projeto *ELM-ART* consiste na disponibilização de um STI na *www* para suportar o ensino da linguagem de programação LISP. O *ELM-ART* é considerado um

[10] Nesta seção estão incluídas apenas algumas tendências, mas existem muitas outras nesta área, como por exemplo: a utilização de multiagentes no desenvolvimento de STI´s, que merecerá especial atenção posteriormente, devido o contexto desta dissertação.

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.

2.5.2 SISTEMAS TUTORES INTELIGENTES E *CSCW*

Segundo (Sommervile & Rodden, 1993), a área de CSCW une pesquisadores da Ciência da Computação, Ciências Cognitivas e Ciências Sociais, que estão interessados em como um grupo de pessoas trabalhando de forma cooperativa podem ser auxiliadas por um suporte computacional.

As tecnologias de redes, tanto locais, quanto de longo alcance, incluindo a Internet, têm proporcionado que o ensino por computador torne-se mais cooperativo. Assim, muitos pesquisadores têm se dedicado à finalidade de promover ensino cooperativo em STI´s, desenvolvendo novas estratégias de ensino-aprendizagem, que consideram a cooperação como um forte aliado, conforme demonstrado no trabalho de Aïmeur (Aïmeur, 1995).

2.5.3 DESENVOLVIMENTO DE FERRAMENTAS DE AUTORIA PARA SISTEMAS TUTORES INTELIGENTES

Segundo (Murray, 1996), um dos problemas associados à construção de STI´s está relacionado ao alto custo para desenvolver um único produto. Assim, uma forma de solução para esta questão pode ser através de ferramentas de autoria. Ainda, conforme (Murray, 1996), sistemas de autoria permitem que uma pessoa sem conhecimento de como programar STI´s possa formalizar seu conhecimento e construí-los.

Neste contexto, as ferramentas de autoria constituem um importante instrumento para o desenvolvimento de STI´s em um certo domínio, permitindo ainda a sua manutenção, reduzindo o esforço computacional dos desenvolvedores e possibilitando uma diminuição no custo de produção. Entre alguns ambientes de autoria de STI´s, estão por exemplo o *Courseware Development Templates* **(CDT)** (O'Shea, 1984) e o *Cooperative Classroom Assistant* **(COCA)** (Major, 1992).

2.6 DIFICULDADES NO DESENVOLVIMENTO DE SISTEMAS TUTORES INTELIGENTES

Sistema Tutores Inteligentes permitindo uma maior adaptação possível aos seus alunos, ainda representam uma tarefa árdua para os seus projetistas. Algumas razões freqüentemente encontradas na literatura para a baixa disseminação desses sistemas são: o alto custo e o longo tempo necessário para o seu desenvolvimento. Segundo (Costa, R. M. *et al*, 1997), algumas questões importantes que dificultam a construção e utilização de STI´s, são expostas a seguir:

2.6.1 A FALTA DE MÉTODOS DE DESENVOLVIMENTO PARA STI´S

Sistemas Tutores Inteligentes possuem particularidades próprias. Entretanto, as técnicas de desenvolvimento que estão sendo utilizadas são técnicas usuais de Engenharia de *Software* aplicadas a qualquer tipo de sistema. Assim, constata-se a necessidade de modelos de ciclo de vida, métodos de desenvolvimento, técnicas para estimativas de custos, controle de qualidade, documentação e integração de equipes multidisciplinares próprios para STI´s. Certamente, a inclusão de tais aspectos padronizaria a criação de STI´s, diminuindo os seus custos e a complexidade das tarefas.

2.6.2 A FALTA DE MÉTODOS DE AVALIAÇÃO DA QUALIDADE DOS STI´S

Segundo (Leite *et al*., 1996), a avaliação de produtos com finalidades educacionais envolve não só questões técnicas, mas também conceitos de aprendizagem, ligados à psicologia, didática, pedagogia e ao domínio da aplicação.

Por envolver diversas áreas de conhecimento, existem muitas dificuldades na avaliação da qualidade dos STI´s, constatando-se a necessidade de métodos para esse propósito. Apesar de tal constatação, observa-se na literatura relacionada à área, que as tentativas de métodos de avaliação ainda são muito deficientes quanto à integração de todos os fatores necessários.

2.6.3 A INTEGRAÇÃO DOS STI´S À *WORLD WIDE WEB*

A principal vantagem de que STI´s sejam utilizados na *www* é que eles podem ser acessados a partir de diferentes pontos geográficos em diferentes tempos. Entretanto, alguns dos problemas ainda encontrados são os seguintes:

- Os altos custos envolvidos no desenvolvimento de STI´s para *www*;

- O tempo de transmissão de recursos multimídia, utilizado por algumas aplicações de STI´s ainda é muito alto e isso ocasiona que o aluno espere bastante por uma resposta.

É válido ressaltar que tais questões, bem como outras relacionadas às dificuldades de utilização de STI´s na *www*, são relacionadas a limitações de cunho tecnológico, podendo ser resolvidas em um reduzido espaço de tempo devido aos constantes avanços nessa área.

2.7 CONCLUSÕES

Neste capítulo, apresentou-se os STI´s: seus componentes, aspectos que devem ser considerados, algumas de suas tendências de pesquisa e dificuldades. Sendo que inicialmente, foi feita uma visão geral dos principais tipos de Ambientes de Ensino-Aprendizagem, finalizando com uma importante tipo: os Ambientes de Ensino-Aprendizagem a Distância via Internet. Tais ambientes são uma realidade que têm se tornado cada vez mais crescente, entretanto um fator preocupante é que a maioria deles não permite uma grande flexibilidade ao professor e nem ao aluno, funcionando quase sempre da mesma forma, independentemente do desempenho do aluno. Assim, a partir do estudo realizado, buscou-se utilizar neste trabalho conceitos associados aos STI´s, permitindo que o professor possa definir diferentes estratégias de ensino que atenda o aluno de diversas formas.

Capítulo 3
Sistemas Tutores Inteligentes Baseados em Agentes

Este capítulo está organizado em duas partes: na primeira são apresentados alguns aspectos relacionados a Agentes e Inteligência Artificial Distribuída. Na segunda parte é apresentado o tema central do capítulo: Sistemas Tutores Inteligentes Baseados em Agentes, juntamente com algumas aplicações consideradas relevantes no contexto deste trabalho.

3.1 AGENTES

Agentes representam um novo paradigma para desenvolvimento de aplicações de *software* (Wooldridge & Jennings, 1998), de tal forma que têm se tornado foco de intenso interesse de muitas sub-áreas da Ciência da Computação. Entretanto, a definição do termo *agente*, tanto por pesquisadores da área específica de Computação Baseada em Agentes, como por pesquisadores da área de IA de uma forma geral, não é universal. Isso ocorre devido a multiplicidade de papéis que um agente pode desempenhar. Assim, cada pesquisador deve buscar refletir o seu contexto em relação ao uso de agentes.

Destacam-se aqui algumas das definições de agentes, freqüentemente citadas na literatura:

*"**Um agente** pode ser visto como qualquer entidade que percebe seu ambiente através de sensores (câmeras, microfones, teclado, etc.) e age sobre ele através de efetuadores (vídeo, alto-falante, impressoras, braços, ftp, etc.)" (Russell & Norvig, 1995).*

*"**Agentes** são entidades baseadas em hardware ou software (mais freqüentemente), com as seguintes propriedades (Wooldridg &Jennings, 1995b):*

- *Autonomia: os agentes operam sem a intervenção direta de operadores (humanos ou outros agentes) e possuem algum tipo de controle sobre suas próprias ações e estados internos;*
- *Habilidade Social: os agentes interagem com outros agentes (possivelmente humanos) através de algum tipo de linguagem de comunicação de agentes;*
- *Reatividade: os agentes percebem seu ambiente (que pode ser o mundo físico, um usuário através de uma interface gráfica, uma coleção de outros agentes, a Internet, ou todas estas combinadas) e respondem aos estímulos deles recebido;*

- *Pró-Atividade: os agentes não somente reagem ao seu ambiente, mas possuem habilidades para exibir comportamento direcionado a objetivos, tomando iniciativas".*

*"**Um Agente Inteligente** é um sistema de computação capaz de ações autônomas flexíveis, a fim de alcançar seus objetivos de projeto. Por flexível, entende-se que o sistema deve ser: reativo, pró-ativo e social" (Wooldridge & Jennings, 1998).*

3.1.1 DESENVOLVIMENTO DE AGENTES

A área de pesquisa relacionada a agentes é ainda muito recente, de tal forma que vários aspectos relacionados ao seu desenvolvimento estão ainda em aberto, tais como a sua modelagem e implementação, por exemplo. Assim, cada desenvolvedor deve relacionar e escolher as técnicas que melhor satisfaçam os objetivos específicos da sua aplicação.

Várias técnicas avançadas de IA podem ser utilizadas no desenvolvimento de agentes, entre elas: capacidades de raciocínio lógico, técnicas *fuzzy*, redes neurais e algoritmos genéticos (Knapik & Johnson, 1998). Além de tais técnicas, é possível também utilizar a tecnologia de Orientação a Objetos (OO) para representá-los. É interessante utilizar OO para o desenvolvimento de agentes, pois é possível usufruir de suas vantagens, usando várias ensinamentos da Engenharia de *Software.*

3.1.1.1 Desenvolvimento de Agentes usando OO

Utilizar OO no desenvolvimento de agentes é um processo bastante natural. Entretanto, é preciso preocupar-se com detalhes de projeto e implementação para que estes realmente possam encapsular não apenas o estado que os próprios objetos já encapsulam, como também autonomia, objetivos, tomadas de decisão, mobilidade (se necessário), capacidades de raciocínio, entre outros.

Assim, é necessário analisar quais aspectos os agentes devem possuir, para então tentar representá-los segundo a OO. Para realçar a importância de que estes aspectos precisam ser meticulosamente estudados antes de representados, apresenta-se a seguir uma possível analogia entre a noção de autonomia associada aos agentes e o encapsulamento em orientação a objetos:

> *Um objeto encapsula algum estado e tem algum controle sobre este estado no qual ele pode somente ser acessado ou modificado via os métodos que o objeto fornece. Agentes encapsulam o estado da mesma forma. Entretanto, agentes também encapsulam o comportamento, em adição ao estado. Um objeto não encap-*

> *sula comportamento: ele não tem nenhum controle sobre a execução dos métodos - se um objeto x invocar um método m em um objeto y, então y não tem nenhum controle sobre se m é executado ou não - ele simplesmente é executado. Nesse sentido, o objeto y não é autônomo, pois não tem nenhum controle sobre suas próprias ações. Ao contrário, pode-se pensar em agentes como tendo exatamente este tipo de controle sobre as ações que executa. Por causa desta diferença, não se pensa em agentes que invocam métodos (ações) sobre agentes e sim requisitando ações para serem executadas. A decisão de agir ou não sobre a requisição se encontra com o receptor (Wooldridge & Jennings 1998).*

3.1.2 CLASSIFICAÇÕES DE AGENTES

Existem várias formas de classificar agentes, uma vez que é possível utilizar diferentes critérios para isso. Os seguintes critérios podem ser utilizados (Franklin & Graesser, 1996):

- O subconjunto das propriedades que eles apresentam;
- O limite e sensibilidade de seus sensores;
- O limite e eficiência de suas ações;
- As tarefas que realizam;
- Através dos seus tipos de mecanismos de controle: algorítmicos, baseado em regras de produção, lógica difusa, redes neurais, entre outros;
- O ambiente (bancos de dados, sistemas de arquivos, redes, internet);
- As linguagens em que são escritos;
- Suas aplicações.

Nas seções a seguir, serão apresentadas duas classificações entre as inúmeras que existem: a primeira trata-se de uma classificação bastante difundida (Russell & Norvig, 1995) e a outra é uma classificação de acordo com o subconjunto das propriedades que os agentes podem apresentar (Franklin & Graesser, 1996).

3.1.2.1 Classificação de agentes AIMA (Artificial Intelligence: A Modern Approach)

A classificação de agentes proposta por (Russell & Norvig, 1995) classifica-os em: reativos, armazenadores de estado do mundo, orientado a objetivos e baseados em utilidades.

- **Agentes Reativos**: realizam ações de acordo com suas percepções. Para isso, pesquisam a regra cuja condição casa-se com a situação corrente e então executam a ação associada com essa regra.
  Exemplo: um agente de *software* (um carro) pode agir de acordo com a seguinte regra de produção, sem levar em consideração nenhum estado interno, nenhum objetivo ou utilidade específica: **Se** *encontrar placa de trânsito "virar à direita"* **Então** *ligar o pisca-pisca de direção e dobrar à direita.*

- **Agentes que armazenam estados do mundo** (ou agentes reativos com estado interno): realizam ações de acordo com suas percepções, combinadas com o conhecimento do estado atual do ambiente.
  Exemplo: um agente de software (um carro) pode ter a mesma regra de produção do agente reativo citado acima. Porém, a rua na qual o carro precisa virar à direita encontra-se interditada por ter acontecido um acidente há tempos atrás. Um agente desse tipo não tomará suas decisões baseado apenas em suas regras de produção mas também no conhecimento do ambiente.

- **Agentes orientado a objetivos**: possuem o conhecimento do estado interno que os cerca, acrescidos de informações de objetivos, que descrevem situações que são desejáveis.

  Exemplo: Ao encontrar uma estrada de junção, um agente de software (carro) pode seguir em frente, virar à direita ou à esquerda. Entretanto, precisa saber qual é o desejo (objetivo) do passageiro para saber que ação escolher (a que atingirá o objetivo do passageiro).

- **Agentes baseados em utilidades**: além de basearem-se nos objetivos para realizarem suas ações, realizam um estudo para saber que forma de atingir os objetivos conduzem a uma alta utilidade para os agentes. Estes agentes possuem uma função que mapeia um estado em um número real, que descreve o grau de utilidade.

  Exemplo: O agente de software (carro) pode possuir uma seqüência de ações possíveis para realizar o seu objetivo de chegar a um determinado lugar, onde cada seqüência pode está mapeada para uma melhor característica, ou seja, caso seja necessário *maior segurança* para se chegar a este lugar, uma determinada seqüência de ações pode ser utilizada e caso seja necessário o aspecto "maior velocidade" para alcançar tal objetivo, outra seqüência de ações poderá ser usada.

3.1.2.2 Classificação conforme as propriedades do Agente

Uma forma possível de classificar agentes consiste em classificá-los de acordo com o seu subconjunto de propriedades. O Quadro 3.1 ilustra algumas destas propriedades (Franklin & Graesser, 1996).

Propriedades	Outros nomes	Significado
Reatividade	Percepção e ação	Respondem aos estímulos do ambiente.
Autonomia		Exercem controle sobre suas próprias ações.
Orientado a objetivos	Movido a iniciativas	Não agem simplesmente em resposta ao ambiente.
Continuidade temporal		É um processo em execução contínua.
Comunicação	Habilidade Social	Comunicação com outros agentes, algumas vezes incluindo pessoas.
Aprendizagem	Adaptabilidade	Alteram seu comportamento baseado em experiências anteriores.
Mobilidade		Habilidade de se transportar de uma máquina para outra.
Flexibilidade		Não possuem ações fixas.

Quadro 3.1: Classificação conforme as propriedades dos agentes

3.1.3 ARQUITETURAS DE AGENTES

Segundo (Wooldridge & Jennings, 1995a), as questões relacionadas ao projeto e construção de agentes inteligentes podem ser divididas em três áreas principais: teorias de agentes, arquiteturas de agentes e linguagens de agentes.

Neste trabalho, será abordada apenas a questão relacionada à arquitetura dos agentes, por se tratar de um assunto relacionado às suas ações. Assim, segundo (Wooldridge & Jennings, 1995a), arquiteturas de agentes são consideradas "modelos para a construção de sistemas de agentes" e podem ser classificadas em: deliberativas, não deliberativas e híbridas.

3.1.3.1 Arquiteturas Deliberativas (abordagem clássica)

Nas **arquiteturas deliberativas**, os agentes, também conhecidos como agentes cognitivos, possuem uma representação simbólica do mundo ou do problema a resolver, representada explicitamente ou não. Esta representação é denominada conhecimento e pode ser adquirido por observação ou comunicação com outros agentes. A partir de tal conhecimento e de seus objetivos, o agente deve traçar uma série de possibilidades de ações para atingi-los, e a partir destas possibilidades, deverá possuir capacidade de decisão para que possa ser capaz de optar por um dos planos possíveis. Assim, baseado no que foi apresentado em (Demazeau & Muller, 1990), a arquitetura (mínima) de um agente cognitivo está ilustrada na figura 3.1.

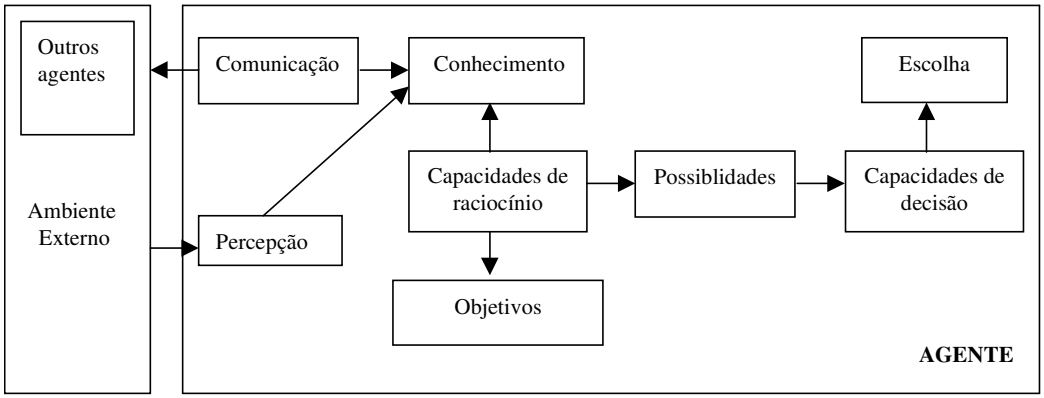


Figura 3.1: O Modelo de um Agente Cognitivo

3.1.3.2 Arquiteturas não-deliberativas (reativas)

As **arquiteturas não-deliberativas**, denominadas na literatura também como "reativas", "situadas", "baseadas em comportamento" e "tropistas" (Corrêa, 1994), baseiam-se no "paradigma reativo". O princípio básico do comportamento dos agentes reativos resume-se ao processo de estímulo-resposta, ou seja, a escolha de uma ação é definida pela situação em que o agente se encontra: o agente executa uma determinada ação conforme a ocorrência de uma condição. Nesta arquitetura, os agentes não utilizam representações simbólicas do mundo e nem trabalham com o planejamento de ações.

3.1.3.3 Arquiteturas Híbridas

Finalmente, as **arquiteturas híbridas** surgiram como forma de fornecer a um mesmo agente as duas formas de arquitetura citadas acima. Assim, a idéia é construir um agente atuante nos dois subsistemas (Wooldridge & Jennings, 1995a):

- Deliberativo: contém um modelo simbólico do mundo, utilizando o planejamento e tomada de decisões;
- Reativo: possui a capacidade de reagir aos eventos que ocorrem no ambiente, sem possuir capacidade de raciocínio.

3.2 INTELIGÊNCIA ARTIFICIAL DISTRIBUÍDA

Os sistemas desenvolvidos no campo da IA consideravam apenas a ação inteligente proveniente de um único agente. Entretanto, como existem problemas que são naturalmente distribuídos e usam conhecimento distribuído para a sua resolução, passou-se a utilizar um conjunto de agentes para resolvê-los. A IAD fornece estratégias para a resolução de problemas desse

tipo, uma vez que o modelo de inteligência utilizado é *baseado no comportamento social,* sendo a ênfase colocada em ações e interações entre agentes (Sichman, *et al*., 1992).

A maior parte da literatura na área da IAD enfatiza duas sub-áreas de trabalho que são: a Resolução Distribuída de Problemas (RDP) e Sistemas Multiagentes (SMA). Em ambas, os agentes são as principais entidades das atividades de resolução de problemas. A seguir, serão detalhadas as principais motivações científicas e características de cada uma delas.

3.2.1 RESOLUÇÃO DISTRIBUÍDA DE PROBLEMAS

A motivação inicial de uma resolução distribuída de problemas é um *problema inicial preciso* que deve ser solucionado. A abordagem RDP encontra-se representada na figura 3.2 (Sichman, 1995 *apud* Álvares & Sichman, 1997). Do ponto de vista de concepção do sistema, os agentes não existem a princípio: sua concepção, bem como a sua organização e interações, são realizadas face à existência de um problema que o sistema deve solucionar. Não existe preocupação quanto à reutilização dos agentes num outro contexto.

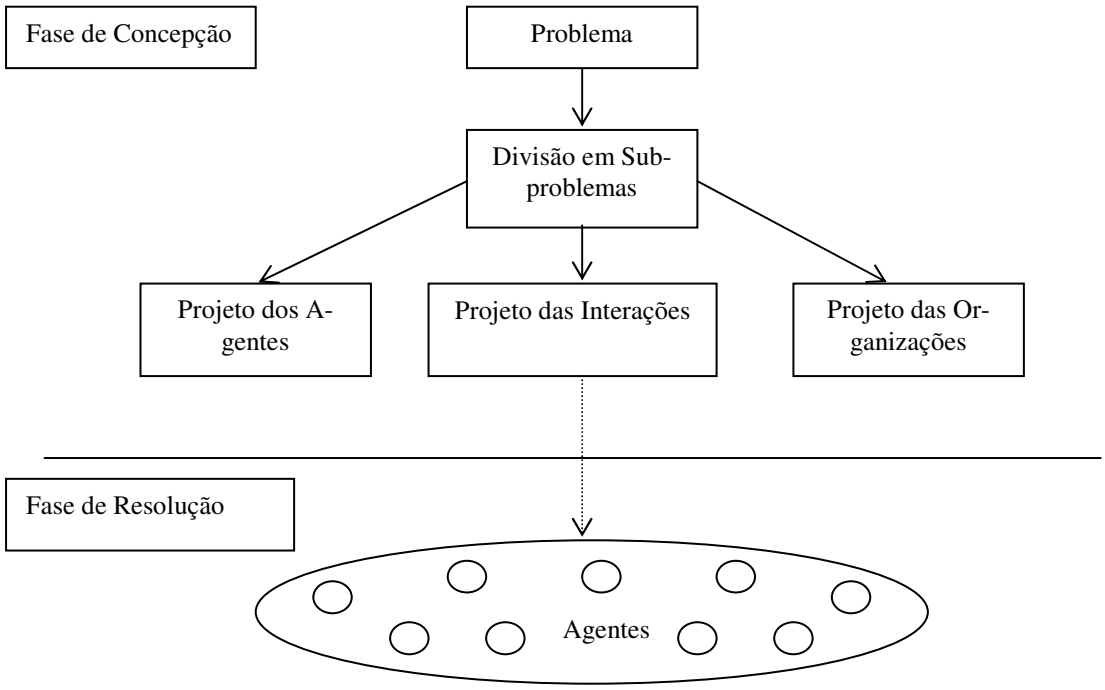


Figura 3.2: Abordagem RDP

3.2.2 SISTEMAS MULTIAGENTES

O principal foco da área de SMA é o estudo de agentes *autônomos* em um universo Multiagente (Demazeau & Muller, 1990). O termo autônomo significa que os agentes têm uma existência própria, independente da existência de outros agentes.

A abordagem SMA encontra-se representada na figura 3.3 (Sichman, 1995 *apud* Álvares & Sichman, 1997). Do ponto de vista de concepção do sistema, os agentes, suas interações e sua organização na sociedade são concebidas independentemente de um problema particular a ser resolvido. Portanto, torna-se possível a reutilização de tais componentes ao projetar-se aplicações similares.

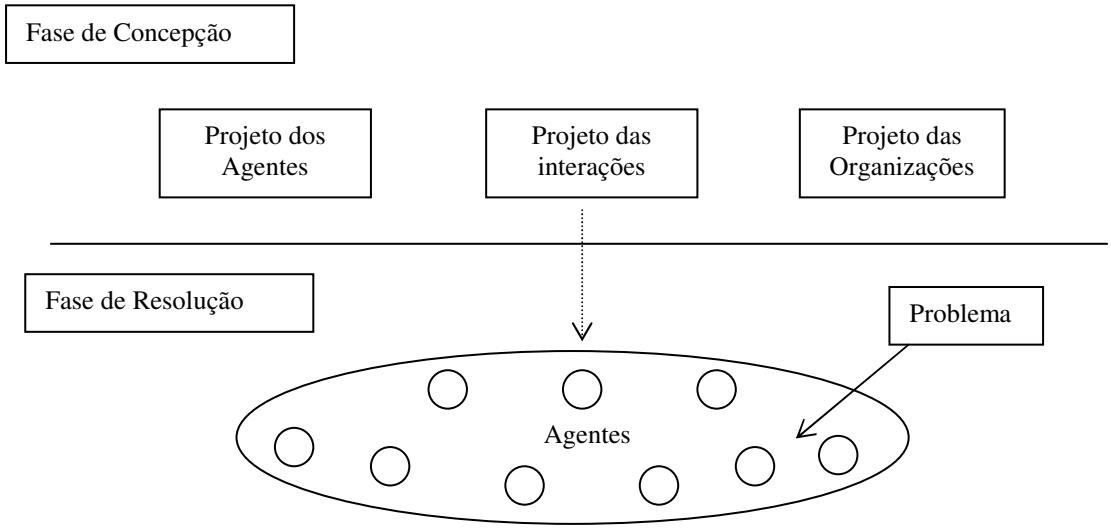


Figura 3.3: Abordagem SMA

3.3 SISTEMA TUTORES BASEADOS EM AGENTES

Conforme mencionado anteriormente[1], um dos objetivos dos Sistemas Educacionais é facilitar a aprendizagem através da adaptabilidade do sistema aos alunos. Assim, sistemas como os STI´s têm perseguido esse objetivo, sendo que agora possuem um forte aliado: o paradigma orientado a agentes.

Desta forma, vários aspectos relacionados a agentes podem ser vantajosos no desenvolvimento de STI´s, como por exemplo: a cooperação que os agentes podem embutir nos módulo do STI (sendo que este é um dos fatores pelos quais tais sistemas não possuem um bom desempenho); a facilidade de comunicação entre os seus módulos e a facilidade de aprendizagem que pode ser embutida em cada módulo separadamente.

Existem diferentes abordagens relacionadas ao uso de agentes na literatura da área. Entretanto, em (Costa, E. B., 1997) é feita uma tentativa de agrupá-las, com a finalidade de facilitar a discussão sobre os tutores baseados em agentes. As categorias ressaltadas são as seguintes:

[1] Cf. seção 1.1 (Capítulo 1).

- **Canônicas:** A maior parte dos STI´s construídos utilizam-se desta abordagem. Tal abordagem, consiste em organizar a arquitetura de um STI em agentes, de tal forma que cada agente possa ficar responsável pelo suporte a uma das especialidades presentes na maioria das arquiteturas clássicas de tais sistemas, ou seja, existe um agente especializado para domínio de aplicação, um agente encarregado de capturar o estado cognitivo do aluno, um agente pedagógico e um agente de interface. Assim, a utilização de agentes em sistemas que seguem esta abordagem, têm a vantagem de oferecer ganhos em relação a alguns aspectos de Engenharia de *Software*, como: extensibilidade, modularidade e portabilidade.
- **Mentalistas:** Esta abordagem surgiu da idéia de utilizar-se estados mentais, como: crenças, desejos e intenções nos componentes de um STI (representados através de agentes).
- **Especializadas:** Esta abordagem está relacionada aos STI´s que têm como objetivo alcançar o estado da arte em alguns dos seus componentes[2], ou seja, quando se deseja embutir o maior número de funcionalidades possíveis em um componente específico do STI, representado por um ou mais agentes.

Nas seções seguintes, são apresentados alguns trabalhos na área de Sistemas Tutores Inteligentes que utilizam-se da abordagem baseada em agentes, ressaltando que esta abordagem ainda é muito incipiente, porém bastante promissora. Ao final de cada descrição são apresentadas as características e contribuições para esta dissertação.

3.3.1 SEI: SISTEMA DE ENSINO INTELIGENTE

O sistema SEI - Sistema de Ensino Inteligente, proposto por (Tedesco, 1997), reflete a abordagem canônica[3]. O SEI é composto por uma sociedade composta por cinco agentes: Agente Domínio, Agente Tutor, Agente Estudante, Agente Controlador e Agente Comunicador (*Interface*). Além dos agentes, o SEI conta com duas Bases de Conhecimento: o Modelo do Estudante e o Modelo do Domínio. Desenvolveu-se com esse sistema, um protótipo implementado na linguagem JAVA e voltado para o ensino de Introdução à Computação. Pode-se perceber, que esse sistema trabalha sobre as mesmas funcionalidades requeridas nas arquiteturas clássicas de sistema tutores inteligentes, só que utilizando o enfoque de agentes. A seguir, uma breve descrição dos componentes do SEI.

[2] Cf. figura 2.1 (Capítulo 2), ilustrando os componentes básicos dos Sistemas Tutores Inteligentes.
[3] Cf. seção 3.3 (Capítulo 3).

Bases de Conhecimento:

- ***Modelo do Estudante***: armazena informações relativas ao aluno corrente, na forma de *frames*, que são inicializados de acordo com um questionário inicial e são atualizados a cada passo da interação. São armazenadas: *características cognitivas* (por exemplo: capacidade de memorização, nível de conhecimento do domínio e gosto por desafios), *histórico da interação*, *caminhos percorridos no currículo* e *histórico do progresso*. Para acomodar as diferenças de conhecimento prévio, o *Modelo do Estudante* dispõe de três estereótipos de aluno, usados para classificá-lo conforme o seu conhecimento prévio de Computação: *(1) Usuário Especialista*, *(2) Usuário Casual* e *(3) Usuário Leigo*. Esta atribuição inicial é feita com base na primeira interação do aluno com o sistema, sendo que posteriormente esta classificação pode ser mudada, pois tal modelo é atualizado a cada passo da interação.

- ***Modelo do Domínio:*** armazena o conteúdo do curso. Por exemplo, o domínio escolhido para teste foi a parte conceitual de *Introdução à Computação*. Este modelo está organizado como uma rede de *frames*. O sistema apresenta as informações em quatro níveis de abstração: Curso (armazena o curso completo, por exemplo: Introdução à Computação); Lições (são as grandes divisões temáticas dos cursos); Tópicos (são as unidades conceituais das lições); e apresentações (guardam o conteúdo que será apresentado para o aluno).

A Sociedade de Agentes:

- ***Agente Controlador:*** gerencia a troca de informações entre os vários agentes do SEI. O *Controlador* recebe solicitações dos outros agentes do sistema e envia para aqueles que fornecem os serviços solicitados. Quando recebe as respostas, o *Controlador* as direciona para os solicitantes. Para isso, o *Controlador* mantém um registro de todos os agentes do sistema, sua localização e serviços disponíveis. Os agentes se comunicam através da troca de mensagens *KQML*.[4]

- ***Agente Estudante:*** manipula informações relacionadas aos alunos e determina o perfil do aluno corrente, baseado nas informações contidas no *Modelo do Estudante*. Quando a sessão tutorial começa, o *Agente Estudante* recebe do *Controlador* a identificação do usuário, para que possa recuperar o *Modelo do Estudante* correspondente. Quando o aluno utiliza o sistema pela primeira vez, um arquivo de *Modelo do Estudante* é criado e inicializado de acordo com os resultados da interação inicial.

[4] KQML (do inglês: *Knowledge Query Manipulation Language*) representa uma linguagem padrão para proporcionar a interação entre agentes e originou-se dos projetos de um grupo de trabalho do DARPA (Finin, *et al*., 1992).

- ***Agente Domínio:*** recupera informações do *Modelo do Domínio*, conforme solicitado pelo *Controlador* e avalia as respostas do aluno aos exercícios propostos, auxiliando assim a determinar o desempenho e o perfil do aluno corrente.

- ***Agente Tutor:*** toma as decisões pedagógicas do SEI, determinando o *que* o SEI vai ensinar, *como* e *quando* o fará. Outra tarefa importantíssima deste agente é determinar as estratégias de resposta ao aluno. Por exemplo, se o aluno corrente está apresentando um desempenho fraco, o *Tutor* adota uma estratégia "encorajadora", fazendo comentários a cada passo da interação, explicando suas decisões e motivando o aluno a superar suas dificuldades. Quando o aluno melhora seu desempenho, o *Tutor* modifica sua estratégia de resposta. O *Tutor* raciocina com base em informações obtidas dos modelos do *Estudante* e do *Domínio*, para determinar o que ensinar, quando e como fazê-lo. O conhecimento do agente *Tutor* é modelado através de regras de produção.

Este trabalho foi importante pelo fato de ter utilizado a abordagem baseada em agentes, com o objetivo de obter vantagens em relação a aspectos da Engenharia de *Software* e este foi um dos aspectos que também foi procurado atingir neste trabalho.

3.3.2 AME-A: AMBIENTE MULTIAGENTE DE ENSINO-APRENDIZAGEM

D´Amico (D´amico, 1999) apresenta uma proposta de uma sociedade de agentes, em que todos são autônomos e se comunicam através de mensagens. D´Amico utiliza uma abordagem considerada psico-pedagógica no seu trabalho. Segundo D´amico, o modelo do aluno é definido por um *Agente Modela_Aprendiz*, que identifica o nível de conhecimento do aprendiz, seus objetivos de aprendizagem, motivações e as suas características psico-pedagógicas.

Os Agentes da sociedade envolvem dois processos básicos: ensino e aprendizagem. Em tal sociedade, é permitida a coexistência de vários *Agentes_Aprendizes*, ou seja, permite que vários alunos trabalhem ao mesmo tempo.

O processo de ensino envolve os seguintes agentes:

- ***Agente Seleção_Objetivos:*** seleciona o próximo objetivo e envia o endereço do material para ser enviado ao Agente *Material_Atual*.

- ***Agente Material_Atual:*** atende o aluno durante o processo de aprendizagem através da apresentação de material instrucional selecionado pelo agente *seleciona_objetivo* de diversas maneiras. A seleção apresentada aqui justifica-se pelas preferências do *Agente_Aprendiz*.

- ***Inic_Agente:*** define a área de dados para o *Agente_Aprendiz*, mostra os objetivos que podem ser selecionados e define o melhor caminho no grafo dos objetivos.

- ***Agente Suprimento_Prática:*** dispõem de diferentes tipos de exercícios, exemplos e aspectos de modelo para o *Agente_Aprendiz* e interfere na escolha feita pelo aluno.

- ***Agente Controle_de_Resultado:*** supervisiona a performance do *Agente_Aprendiz* durante os testes e execução dos exercícios, a fim de fornecer um retorno dos resultados obtidos pelo aluno. Ele também distingue os "erros de deslize" (erros que ocorrem quando alguém comete um engano por não entender o problema). Se o agente detecta um possível desapontamento ou entusiasmo, ele passa esta informação ao *Agente Dar_Ajuda*.

- ***Agente Dar_Ajuda:*** auxilia o *Agente_Aprendiz*, caso sua performance não seja muita boa ou se ele estiver desconcentrado. O *Agente Dar_Ajuda* oferece novos exemplos, voltando-se ao *Agente Material_Atual* e indica ainda o uso do navegador (procedimento que orienta o aluno no seu trabalho).

O processo de aprendizagem envolve os seguintes agentes:

- ***Agente_Aprendiz:*** é o próprio aluno que utiliza o sistema.

- ***Agente_Navegação:*** é o agente que acessa o material armazenado em um hiperdocumento para esclarecer dúvidas, oferecer exemplos, anexar lembretes e apresentar conceitos básicos quando o aluno requisitá-los.

- ***Agente Seleção_Estratégia:*** é o agente que seleciona a estratégia de ensino-aprendizagem mais adequada ao *Agente_Aprendiz*. As estratégias são em número fixo e ativadas em função da quantidade de parâmetros válidos entre o agente e a estrutura da estratégia.

- ***Agente_Performance:*** é o agente que analisa a história do *Agente_Aprendiz*, o seu conhecimento, o seu caminho e a sua preferência, para tentar ensinar algo que ele ainda não está apto a aprender ou não necessita.

O ambiente proposto por D'Amico envolve também as múltiplas estratégias, porém elas são selecionadas em função dos parâmetros que o ***Agente Seleção_Estratégia*** recebe dos outros agentes. Esse trabalho foi importante por considerar mais de uma estratégia em função de um modelo diferenciado de aluno, reforçando a importância do tema para a área de pesquisa.

3.3.3 A ARQUITETURA *MUTANTIS*

A arquitetura *MutAntIS,* proposta por (Azevedo, 1999), especifica uma arquitetura multi-agentes para a construção de sistemas tutores inteligentes, descrevendo as características específicas de cada um dos agentes que fazem parte da arquitetura e o esquema de comunicação entre eles. A idéia inicial desta arquitetura veio da arquitetura GIA (Cheikes, 1995). Tal arquitetura representa uma proposta para utilização de agentes inteligentes em sistemas tutores inteligentes. Azevedo (Azevedo, 1999) ressalta que a diferença entre a arquitetura *MutAntIS* e a arquitetura GIA é que a GIA apresenta de uma forma geral a arquitetura multi-agente, isto é, ela descreve as características gerais dos agentes e o esquema de comunicação entre eles. Entretanto, ela não entra em detalhes sobre quais os agentes devem fazer parte da arquitetura, e as características específicas de cada um desses agentes. A arquitetura *MutAntIS*, seguindo o modelo da arquitetura GIA (com os tipos de agentes propostos e o esquema de comunicação), especifica quantos e quais os agentes devem fazer parte da arquitetura, além das características específicas de cada um. A arquitetura *MutAntIS* é composta por seis agentes inteligentes, representados na figura 3.4.

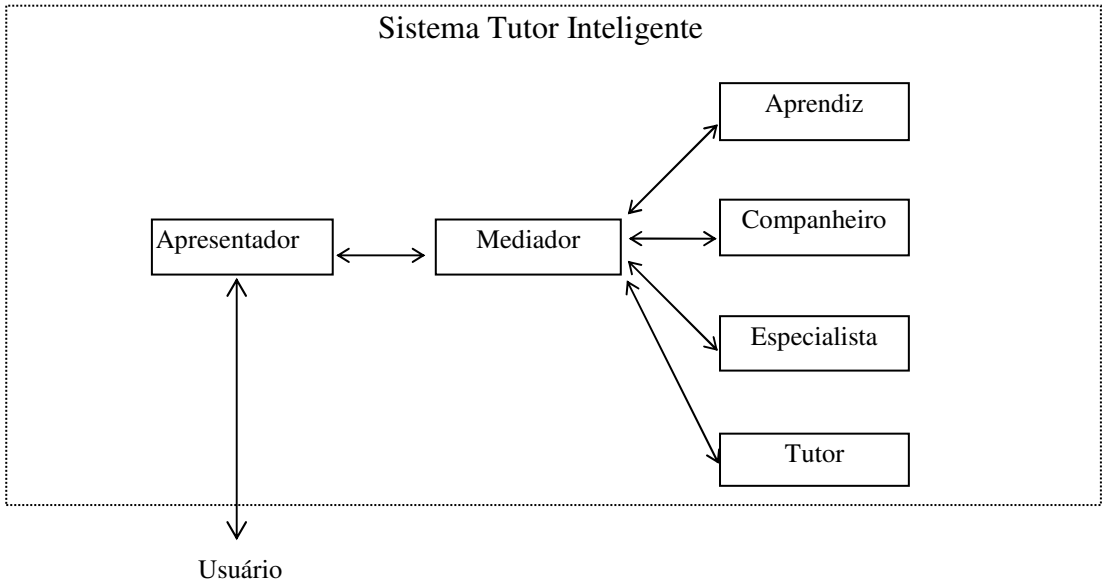


Figura 3.4: Arquitetura *MutAntIS.*

A seguir, uma breve descrição de cada um dos agentes inteligentes da arquitetura *MutAntIS*:

- ***Agente Apresentador:*** tem a função de realizar a interface do STI com o usuário, oferecendo todos os serviços relativos à apresentação de informações ao usuário e todos os serviços relativos à obtenção de informações fornecidas pelo mesmo.

- ***Agente Mediador**:* a função deste agente é atuar como um elo de ligação entre todos os agentes, fazendo um roteamento de mensagens entre eles. Quando o STI for inicializado, todos os agentes devem enviar ao *Mediador* os serviços que eles oferecem (as tarefas que eles podem executar) e as necessidades de cada um (as informações de que precisam). Com isso, o *Mediador* será capaz de "saber" quais os serviços oferecidos por cada agente e quais as necessidades de cada um.

- ***Agente Aprendiz**:* este agente é responsável por criar e manter modelos dos aprendizes (usuários) que utilizam o STI. O *Agente Aprendiz* deve também fornecer informações ao *Agente Tutor*, para identificação de deficiências no modelo do aprendiz e para controle do progresso do usuário. O *Agente Aprendiz* pode também fornecer informações para ajudar a geração de problemas e a adaptação de explicações. O modelo do aprendiz é construído dinamicamente, à medida que o usuário utiliza o STI.

- ***Agente Companheiro**:* o papel deste agente é atuar como um companheiro de estudo do usuário, podendo atuar de diversas formas (fornecendo dicas para auxiliar o usuário na resolução de problemas, na realização de tarefas, nas questões elaboradas pelo STI, por exemplo).

- ***Agente Especialista**:* contém uma representação do conhecimento a ser comunicado ao usuário, agindo como a fonte do conhecimento a ser apresentado. Ele deve ser capaz de gerar explicações, respostas, tarefas e questões que devem ser apresentadas ao usuário.

- ***Agente Tutor:*** projeta e regula as interações instrucionais com o usuário. É ele quem decide as atividades pedagógicas que serão utilizadas: avisos, suporte, explicações, tarefas diferentes, entre outras. O *Agente Tutor* determina o que o STI vai apresentar ao usuário, como isso deve ser realizado e quando deve ser feito. Para isso, ele age de acordo com informações obtidas do *Agente Aprendiz* e do *Agente Especialista*.

Este trabalho também utiliza a abordagem baseada em agentes com o objetivo de obter vantagens em relação a aspectos da Engenharia de *Software*. Como já foi comentado, um dos aspectos que também foi buscado neste trabalho.

3.3.4 O AMBIENTE MCOE *(MULTIAGENT COOPERATIVE ENVIRONMENT)*

O ambiente MCOE, proposto por (Giraffa, 1999), é um jogo educacional multimídia. Esse jogo é composto por um lago onde existe um ecossistema formado por peixes, plantas, água, microrganismos que possuem um sistema de reprodução em equilíbrio, até que a intervenção de poluentes provoquem alterações no seu estado normal. Estes poluentes aparecem de forma

aleatória ao longo do jogo e são combatidos através de ferramentas do personagem escolhido por cada aluno. As possíveis ações que o aluno pode obter mudam de acordo com o personagem que ele seleciona. Alguns dos personagens são mais poderosos que outros e possuem ferramentas mais fortes no combate à poluição e seus efeitos simulam o que acontece no mundo real. Se o aluno seleciona o personagem "Prefeito", ele terá ferramentas mais poderosas para resolver os problemas, de uma maneira mais rápida, ou mais eficaz do que um aluno que seleciona o personagem "Mãe natureza", por exemplo. O aluno tem a liberdade de escolher a ferramentas que quiser e vai interagir com um colega para construir uma estratégia comum para resolver o problema da poluição do lago. Ele pode ou não cooperar com seu parceiro e isto ocorre externamente ao sistema. O que o sistema percebe são as suas ações e estados mentais associados. Baseado nestas informações, o tutor vai decidir a forma de se comportar com o aluno.

O MCOE foi concebido numa abordagem construtivista e os seus agentes modelados através de seus estados mentais (crenças, desejos, intenções e expectativas). Considera-se dois tipos de agentes: Reativos e Cognitivos. Os Agentes Reativos são todos aqueles que sofrem as ações dos elementos poluidores e reagem conforme foram modelados. Os Agentes Cognitivos são aqueles que se utilizam de estratégias para agirem e possuem um conhecimento do ambiente. Os Agentes Cognitivos são baseados em objetivos, sendo que seu objetivo primordial é a manutenção do equilíbrio ecológico. A comunicação entre os Agentes Cognitivos e Reativos é baseada na troca de mensagens, sendo executada através de um protocolo simples, onde há um remetente, um receptor (endereço) e uma mensagem (ações que serão executadas).

O MCOE prevê diferentes formas de se trabalhar com um determinado conhecimento (estratégias de ensino e táticas associadas), levando em consideração o tipo de usuário que está interagindo com o sistema. No MCOE são utilizadas três estratégias para guiar o comportamento do tutor: guia, reativo e assistente. Segundo a autora (Giraffa, 1999), cada estratégia traz implicitamente um determinado grau de interferência no trabalho do aluno. Quando o tutor é um guia, ele sugere de forma mais direta o que o aluno deve fazer e sugere ações específicas. O tutor se comportará assim caso ele perceba (através do conjunto de estados mentais e ações) que o aluno não possui uma linha de trabalho ou não possui um entendimento do que fazer dentro do jogo, ou então caso diagnostique que o sistema está em colapso eminente (limitação de tempo de jogo). Na estratégia reativa, o tutor se comporta em função dos pontos críticos (problemas) que vão surgindo. O aluno é induzido a pensar em possibilidades de soluções através do trabalho cooperativo com seu colega. Na estratégia assistente, o tutor se comporta como um parceiro que sugere ações de forma menos invasiva e procura incentivar o aluno a descobrir caminhos e refletir sobre o que está acontecendo no sistema. O tutor não joga e sua interferência se dá através de mensagens e avisos diferenciados para os alunos.

O trabalho da autora (Giraffa, 1999) foi importante pelo fato de que a seleção e adoção de uma nova estratégia, por parte do tutor, acontece durante a mesma sessão de trabalho com o aluno. Geralmente, nos demais ambientes isso ocorre somente ao final da interação. Além disso, utiliza-se os estados mentais contidos no modelo de aluno para dirigir o comportamento do tutor no que diz respeito à seleção e adoção de estratégias de ensino.

3.4 CONCLUSÕES

O desenvolvimento de Sistemas Tutores Inteligentes através da utilização da tecnologia de agentes vem sendo bastante pesquisado, conforme demonstrado em alguns exemplos da seção anterior. Esta nova visão de STI´s traz algumas vantagens evidentes sobre a abordagem tradicional (cf. seção 3.3). As considerações sobre o uso da tecnologia de agentes nos STI's, sob a ótica do aluno, valorizam as aplicações incorporando características que permitem personalizar o atendimento, aprender sobre as suas preferências e habilidades, tornando assim a aplicação mais flexível e adaptável ao modelo do aluno.

Assim, no desenvolvimento deste trabalho, considerou-se que os módulos do TUTA são representados por Agentes que podem ser classificados segundo as características enumeradas por (Wooldridge & Jennings, 1995b) da seguinte forma:

- *Autonomia:* todos os agentes do TUTA operam autonomamente, já que possuem controle sobre as ações que executam;
- *Reatividade:* os agentes percebem seu ambiente (no caso, uma coleção de outros agentes) e respondem aos estímulos deles recebidos;
- *Habilidade Social:* um dos agentes (o Agente de Comunicação[5]) interage com todos os outros agentes do TUTA, os outros, por sua vez, interagem apenas com o Agente de Comunicação.

[5] O Agente de Comunicação será apresentado no Capítulo 5 (cf. seção 5.6).

Capítulo 4
O TUTA

Este capítulo apresenta o TUTA. Inicialmente, descreve-se a arquitetura ACVA e o TUTA no contexto dessa arquitetura. Posteriormente, apresentam-se a definição dos requisitos do TUTA, sua arquitetura, funcionamento geral e possíveis aplicações.

4.1 ACVA (ARQUITETURA DE UMA CLASSE VIRTUAL ADAPTATIVA)

A ACVA tem como principal função fornecer um serviço *telemático* do tipo "*classe virtual adaptativa"* e o seu princípio fundamental é evitar a "*homogeneidade superficial do nível de conhecimento dos alunos existente nas aulas tradicionais"* (Hernández-Domínguez, 1995), (Hernández-Domínguez, 1997). Desta forma, a ACVA considera que uma classe virtual é composta de diversos grupos heterogêneos, onde cada grupo permite que um conjunto de alunos de diferentes níveis de conhecimento participem de uma *sessão de treinamento*[16].

Dentro de cada grupo, os alunos com níveis de conhecimento similares são agrupados, e tais níveis são tratados como zonas de comportamento dentro do grupo. As zonas de comportamento estabelecidas pela ACVA (em ordem crescente) são: inferior crítica, inferior intermediária, normal, superior intermediária e superior crítica.

Tal arquitetura é dita adaptativa porque permite e controla a mobilidade dos estudantes entre os diversos grupos heterogêneos, ou mesmo dentro desses grupos (*inter-grupo* ou *intra-grupo*). Essa mobilidade ocorre conforme as necessidades educacionais dos alunos, detectadas a partir dos seus comportamentos durante uma sessão de treinamento.

[16] No contexto desta arquitetura, um curso é composto por um conjunto de sessões de treinamento. Entretanto, outros sinônimos também podem ser utilizados neste texto, como: sessões de formação, sessões de ensino-aprendizagem ou apenas sessões de ensino.

Na Classe Virtual Adaptativa, cada grupo é associado à diferentes *níveis de conhecimento* e dentro de cada grupo são associadas diferentes *zonas de comportamento*, como ilustra (por exemplo) a Figura 4.1, composta de dois grupos: *grupo i* e *grupo j*.

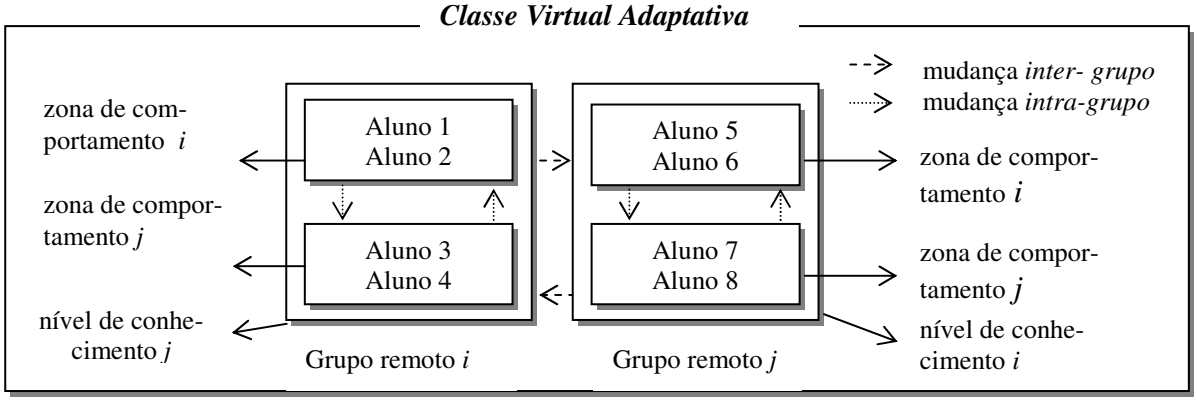


Figura 4.1: Distribuição dos Alunos na Classe Virtual Adaptativa

Uma aplicação desta arquitetura consiste em um ambiente *multi-site* para educação a distância, em que são consideradas duas classes de *site* (Hernández-Domínguez, 1998) (Figura 4.2):

- O *site*[17] central, representando um servidor composto de um conjunto de entidades a serem reutilizadas denominadas ***"SITB"*** (do inglês: *Service Independent Training Building Block*). É importante ressaltar que, neste trabalho, as entidades são apenas de natureza didática (definições, exemplos, questões de escolha múltipla, questionários, entre outros).

- Os *sites* descentralizados que permitem que os alunos participem de uma aula adaptativa. Neste caso, o computador é utilizado no controle das tarefas e das sessões de ensino.

[17] Neste trabalho, o termo *site* fica restrito a um sistema computacional mas ele pode ser muito mais abrangente, por exemplo um *site* poderia ser equivalente a um centro de recursos pedagógicos envolvendo diferentes recursos como: sala multimídia, videoconferência, produção de vídeo, a utilização do computador nas tarefas de ensino (apresentações, simulações, tutores, entre outros).

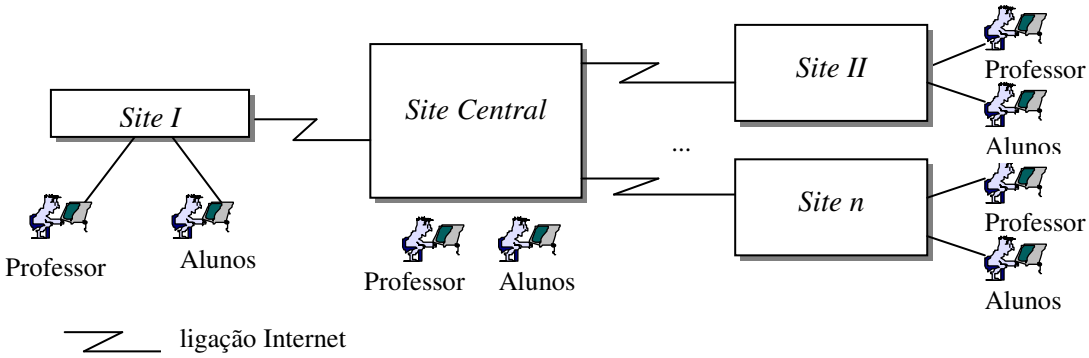


Figura 4.2: Um ambiente multi-site para educação a distância na ACVA

4.1.1. ARQUITETURA GERAL

No contexto da ACVA, os elementos armazenados no *site* principal devem ser representados, armazenados e reutilizados, usando uma representação única e conhecida pelos *sites* descentralizados. A arquitetura de cada *site* descentralizado deve obedecer o padrão proposto pela ACVA. Tal arquitetura é representada por 4 (quatro) níveis de abstração ou camadas (Hernández-Domínguez 1995) (Figura 4.3): *Camada de Serviços de Formação da Classe Virtual* (Camada de ***SF_CV***); *Camada de Serviços de Formação de Grupo* (Camada de ***SF_Grupo***); *Camada de Serviços de Formação Básicos* (Camada de ***SF_Básicos***); e *Camada de Suporte*.

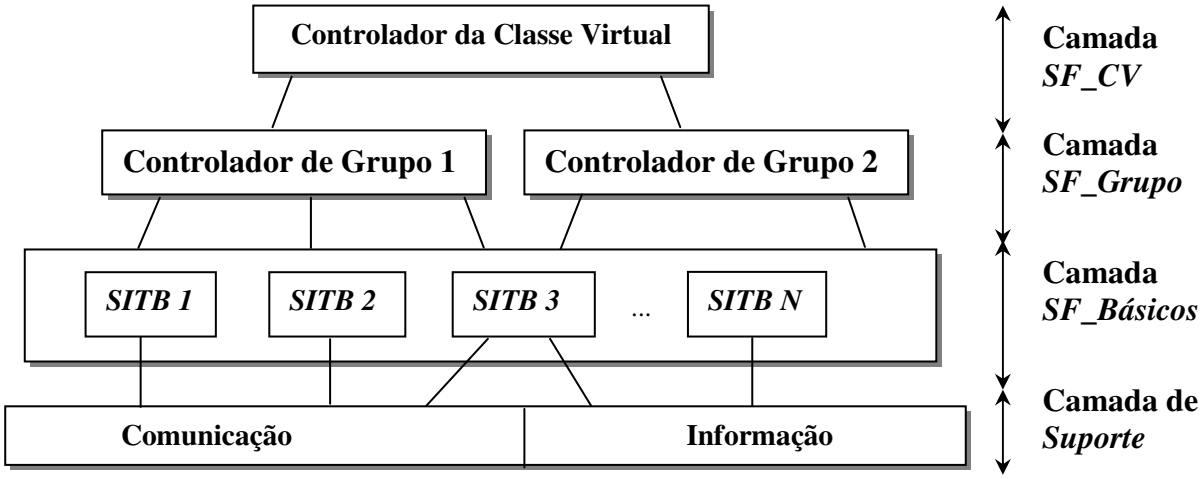


Figura 4.3: Arquitetura da Classe Virtual Adaptativa (ACVA)

4.1.1.1 Camada de Suporte

Essa é a camada mais baixa da arquitetura e permite o armazenamento da informação e o controle da comunicação entre alunos (*inter-grupo* ou *intra-grupo*) e/ou com o professor (comunicação individual ou em grupo).

4.1.1.2 Camada de Serviços de Formação Básicos

A *Camada de Serviços de Formação Básicos* contém um conjunto de componentes funcionais, independentes e reutilizáveis, permitindo que cada controlador de grupo (que é o tutor, representado pela *Camada de Serviços de Formação de Grupo*) utilize recursos comuns. Existem duas classes de entidades ou recursos nessa camada: entidades didáticas (atividades de ensino) e recursos de informação (informações complementares, tais como: sala multimídia e videoconferência). Entretanto, como já foi citado anteriormente, as entidades representadas neste trabalho são apenas de natureza didática e tal camada será referenciada apenas como **Servidor de Entidades Didáticas**.

4.1.1.3 Camada de Serviços de Formação de Grupo

A *Camada de Serviços de Formação de Grupo* é manuseada por um controlador de grupo, onde cada grupo é associado a um nível de conhecimento. Essa camada representa um tutor de um grupo de alunos e o nível real captado de conhecimento para cada aluno deve pertencer ao nível do grupo, caso contrário o aluno é candidato a mudar de grupo. Essas mudanças no interior do grupo ou *intra-grupo* são realizadas por esta camada via o controle de zonas de comportamento dentro do grupo (normal, intermediária e crítica) (Figura 4.4). Este é o *primeiro* nível de adaptação considerado na ACVA.

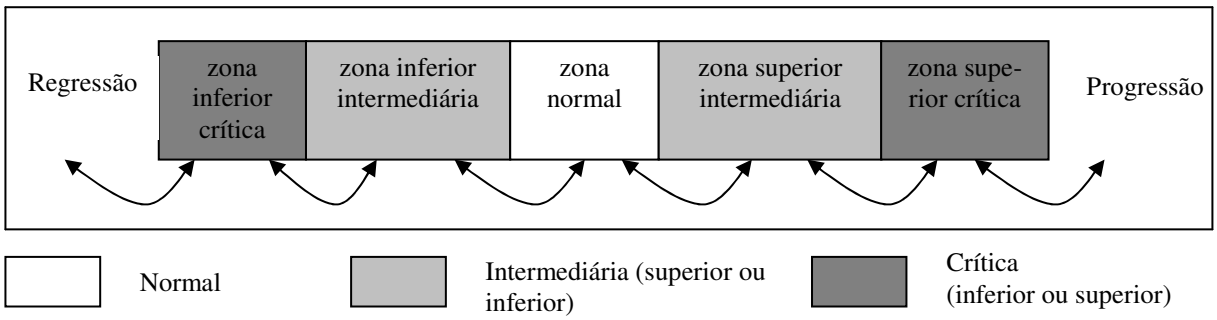


Figura 4.4: Zonas de comportamento de um grupo na ACVA

4.1.1.4 Camada de Serviços de Formação da Classe Virtual

A *Camada de Serviços de Formação da Classe Virtual* deve gerenciar uma sessão de ensino-aprendizagem de uma classe virtual, composta de um conjunto de grupos remotos. Essa camada é manuseada pelo controlador de classe, que detém o controle pedagógico dos grupos

de aprendizagem da classe virtual, em que cada grupo é classificado por um nível de conhecimento. As mudanças de grupo (*inter-grupo*) pelos alunos são controladas por esse controlador, que é composto dos seguintes módulos: módulo gerenciador de comunicação, módulo pedagógico e módulo gererenciador do comportamento dos grupos. As mudanças de grupo representam progressões ou regressões significativas do aluno. Desta forma, a ACVA tem um *segundo* nível de adaptação, considerando as mudanças *inter-grupo*.

4.2 O TUTOR (TUTA) NO CONTEXTO DA ARQUITETURA DE UMA CLASSE VIRTUAL ADAPTATIVA (ACVA)

Cada uma das camadas na ACVA pode ser considerada um sub-sistema particular, representada por um conjunto de elementos (Hernández-Domínguez, 1995). O acoplamento das camadas de *SF_Grupo* e *SF_Básicos* permitem que se tenha um sistema tutor dedicado ao ensino-aprendizagem de um domínio em particular, associado a um determinado nível de conhecimento.

Assim, o acoplamento de tais camadas permitiu a formação do TUTA: Um tutor para um grupo de alunos no contexto da ACVA. A Figura 4.5 ilustra que a associação das duas camadas permite que se tenha um grupo de alunos e professor interagindo com o sistema tutor, onde o TUTA é representado pela camada de SF_Grupo. Entretanto, para o que o sistema tutor possa funcionar corretamente é necessário a associação desta camada à camada de SF_Básicos (ou apenas Servidor de Entidades Didáticas).

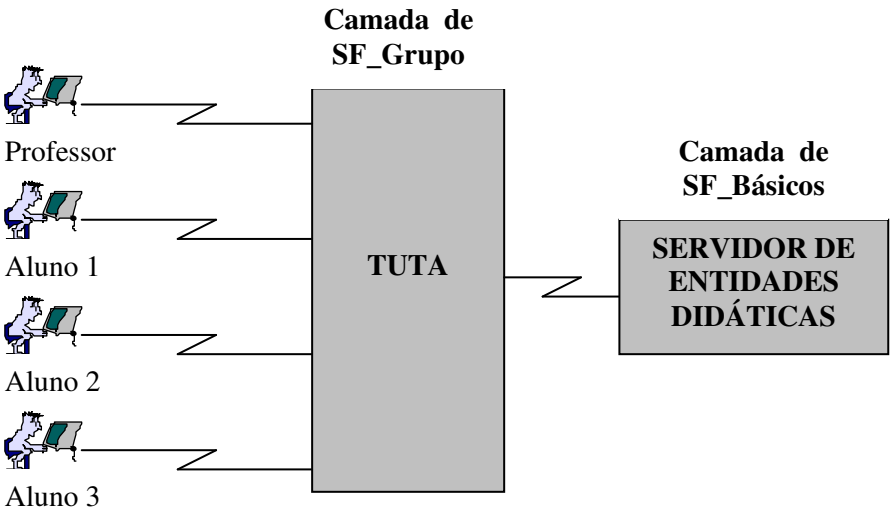


Figura 4.5: Arquitetura do TUTA baseada nas camadas da ACVA

Os elementos que devem fazer parte das camadas *SF_Básicos* e *SF_Grupo* são (Hernández-Domínguez, 1995) (Figura 4.6): o *servidor de entidades didáticas* associado a um domínio (pertencente à camada de *SF_Básicos*); o *comportamento dos alunos* durante a sessão, através dos seus *perfis individuais* e de *grupo* (pertencentes à camada de *SF_Grupo*); o funcionamento do tutor do ponto de vista didático, o qual depende da recuperação e da execução *de estratégias didáticas* (pertencente à camada de *SF_Grupo*); e a interação amigável dos alunos com o sistema tutor, que é controlada pela *interface* (pertencente à camada *de SF_Grupo*).

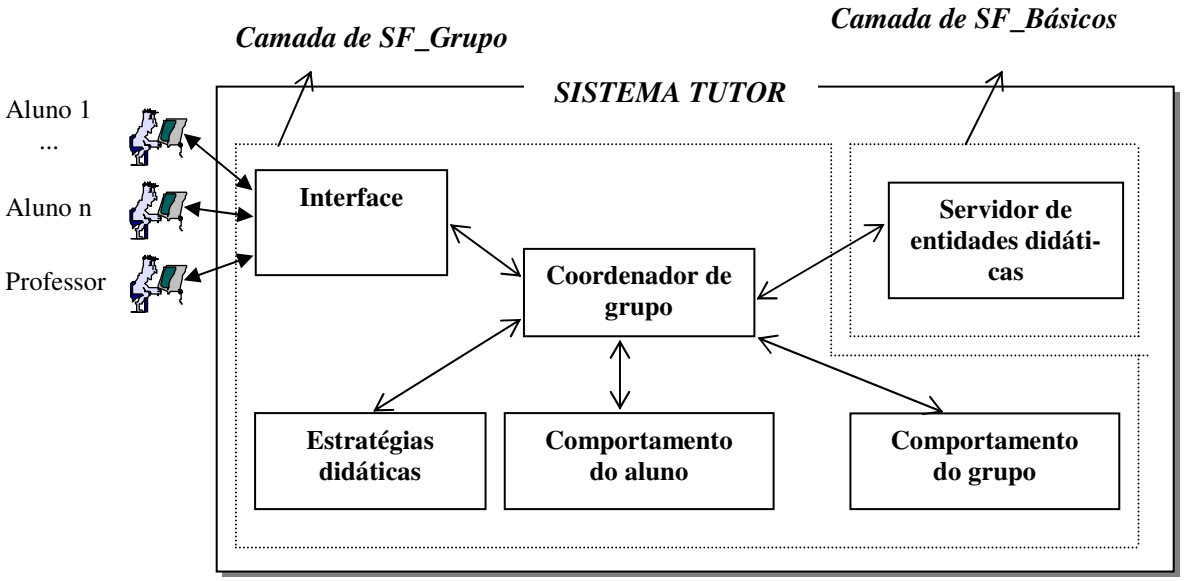


Figura 4.6: Elementos do TUTA no Contexto da ACVA

4.3 DEFINIÇÃO DOS REQUISITOS FUNCIONAIS DO TUTA

O TUTA auxilia o professor nas tarefas de *ensino* no contexto do ensino a distancia, permitindo o *aprendizado* colaborativo de um grupo de pessoas geograficamente distantes através de uma série de sessões de treinamento (o estudo de caso considerado neste trabalho foi o domínio da orientação a objetos).

4.3.1 OBJETIVOS GERAIS DO TUTA

Os objetivos gerais do TUTA são:

- Auxiliar o professor em suas tarefas de ensino.
- Permitir a aquisição de informações pelos diversos alunos participantes de um curso.
- Permitir a interação síncrona entre os diversos alunos e o professor na forma de debates.
- Permitir a interação assíncrona entre os diversos alunos e o professor.

4.3.2 FUNÇÕES DO TUTA

As funções do TUTA estão especificadas através de dois quadros[18]: o primeiro representa as *"funções de apoio ao TUTA"* (Quadro 4.1) e o segundo representa as *"funções de execução de sessão do TUTA"* (Quadro 4.2). Estas últimas funções são consideradas essenciais para a realização de uma sessão no sistema.

[18] Segundo (Larman, 1998), as funções possuem uma referência (número que a identifica) e uma categoria que pode assumir os seguintes estados: enfeite (quando é apenas um função acessória no sistema, sem caráter obrigatório); oculta (quando ocorre de forma implícita para o usuário) e visível (quando é evidente para o usuário).

Ref.	Função	Categoria
R1.1	Permitir o cadastro de professores que podem utilizar o sistema.	Visível
R1.2	Permitir o cadastro de um curso por um professor cadastrado no sistema.	Visível
R1.3	Permitir o cadastro das sessões de um curso por um professor cadastrado no sistema.	Visível
R1.4	Permitir o cadastro de alunos que podem participar de cursos.	Visível
R1.5	Permitir a criação de um grupo.	
R1.6	Permitir que o professor especifique as estratégias didáticas para o curso ministrado por ele.	Visível
R1.7	Permitir que o professor especifique as entidades didáticas a serem utilizadas pelo curso ministrado por ele.	Visível
R1.8	Notificar antecipadamente os alunos e professor participantes de um curso sobre ocorrência de sessão.	Visível
R1.9	Notificar os alunos sobre alterações no curso em que são participantes.	Visível
R1.10	Notificar os alunos sobre alterações nas sessões do curso que participam.	Visível
R1.11	Notificar os alunos sobre possível exclusão do curso em que participam.	Visível
R1.12	Notificar os alunos sobre possíveis exclusões das sessões do curso em que participam.	Visível
R1.13	Permitir que o professor possa acompanhar o rendimento do grupo no qual ele é responsável.	Visível
R1.14	Permitir que os alunos possam acompanhar o seu próprio rendimento, bem como o rendimento do seu grupo.	Visível

Quadro 4.1: Funções de apoio ao TUTA

Ref.	Função	Categoria
R2.1	Recuperar data e hora de ocorrência de sessão.	Oculta
R2.2	Permitir *login* no sistema para alunos cadastrados (na data de sessão) antes da hora prevista para o início de sessão.	Visível
R2.3	Permitir *logout* no sistema em qualquer momento da sessão.	Visível
R2.4	Validar dados de entrada do aluno (*login, senha*).	Visível
R2.5	Fornecer uma interface amigável para permitir que alunos e professores interajam entre si.	Visível
R2.6	Recuperar a(s) estratégia(s) didática(s) necessárias para o acontecimento de uma sessão.	Oculta
R2.7	Executar todos os passos de uma estratégia (anteriormente especificadas pelo professor).	Oculta
R2.8	Recuperar os objetos didáticos necessários para a execução de uma estratégia.	Oculta
R2.9	Enviar informações para os alunos, conforme execução da estratégia didática.	Oculta
R2.10	Recuperar informações dos alunos.	Oculta
R2.11	Permitir que em qualquer momento o aluno possa interromper a sessão e expor suas dúvidas para o professor e demais alunos.	Visível
R2.12	Avaliar respostas dos alunos.	Oculta
R2.13	Atualizar o perfil do aluno (através das notas).	Oculta
R2.14	Atualizar o perfil do grupo (através das notas de todos os alunos).	Oculta
R2.15	Classificar os alunos através de zonas de comportamento (através da suas notas)	Oculta
R2.16	Realizar a mobilidade lógica dos alunos através das zonas de comportamento (através da notas).	Oculta
R2.17	Permitir interações (através de debates síncronos) entre os alunos de um grupo e o professor.	Visível

Quadro 4.2: Funções de execução de sessão do TUTA

4.3.3 CASOS DE USO DO TUTA

Segundo Jacobson (Jacobson *et al*., 1992), casos de uso (do inglês: *use cases*) fornecem uma descrição sobre como o sistema será usado, ou seja, os casos de uso especificam as funcionalidades do sistema. Desta forma, utilizou-se tal técnica[19], a fim de permitir um melhor entendimento dos requisitos do TUTA. Desta forma, o diagrama de Casos de Uso geral do TUTA, está ilustrado na Figura 4.7.

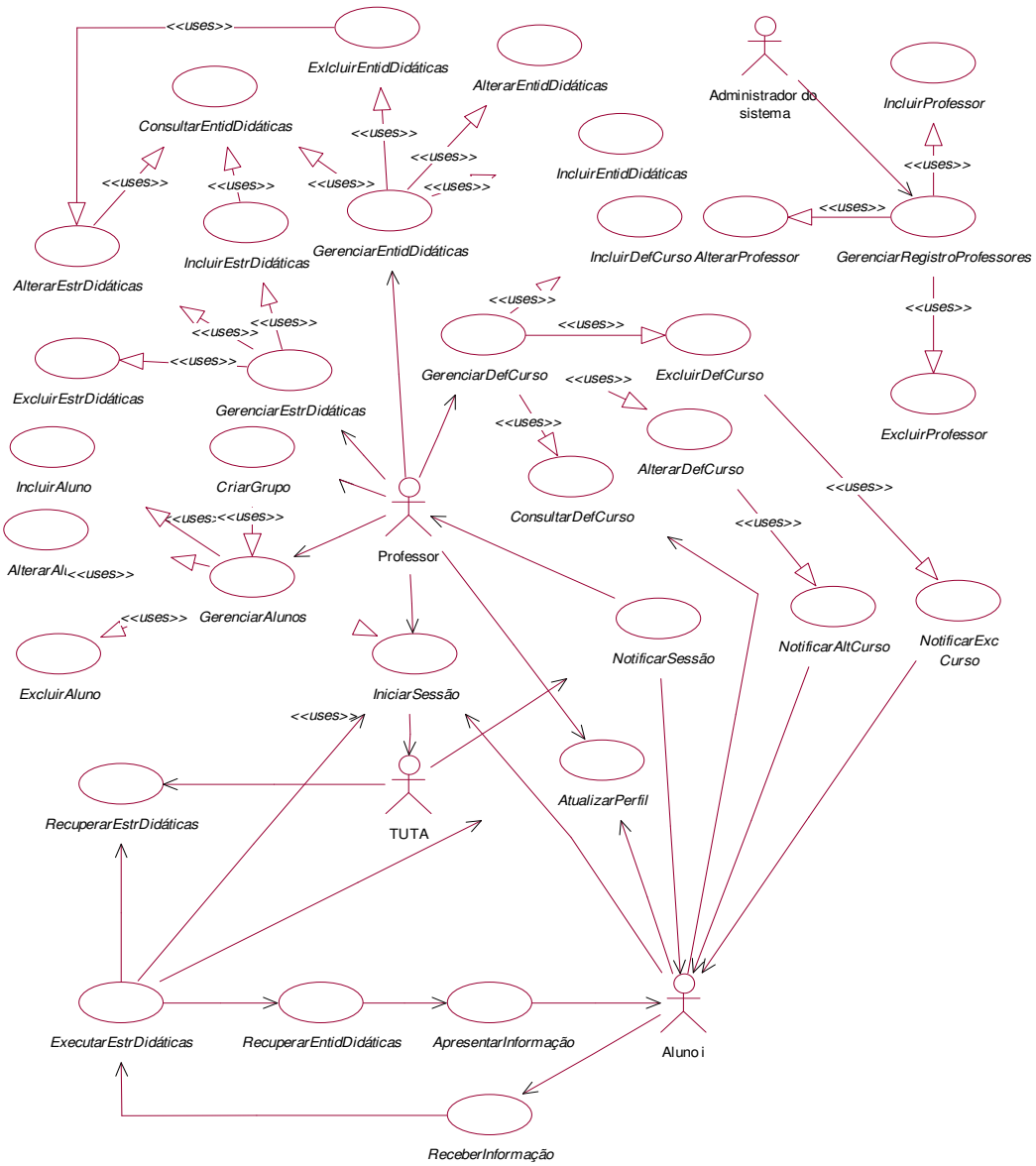


Figura 4.7: Diagrama de Casos de Uso do TUTA

[19] A terminologia utilizada em Casos de Uso encontra-se no Apêndice A deste trabalho.

As seções *4.3.3.1* à *4.3.3.16*, apresentam a descrição de cada um desses casos de uso:

4.3.3.1 Caso de Uso "GerenciarRegistroProfessores"

Através do "*GerenciarRegistroProfessores"*, o Administrador do Sistema pode gerenciar o cadastro de professores no TUTA (através de inclusões, alterações e exclusões). Esse Caso de Uso atende a seguinte funcionalidade[20]:

- Permitir o cadastro de Professores que podem utilizar o sistema (R1.1).

4.3.3.2 Caso de Uso "GerenciarDefCurso"

Através do "*GerenciarDefCurso"*, o professor pode incluir as definições do curso que será ministrado por ele (nome e descrição do curso, período em que será realizado, entre outros), bem como as definições das sessões desse curso (objetivo e conteúdo de cada sessão, entre outros). Esse Caso de Uso atende as seguintes funcionalidades:

- Permitir o cadastro de um curso por um professor cadastrado no sistema (R1.2).
- Permitir o cadastro das sessões de um curso por um professor cadastrado no sistema (R1.3).

4.3.3.3 Caso de Uso "GerenciarAlunos"

Através do *GerenciarAlunos*, o professor responsável por um curso pode gerenciar o cadastro de alunos no TUTA (através de inclusões, alterações e exclusões). Esse Caso de Uso atende a seguinte funcionalidade:

- Permitir o cadastro de alunos que podem participar de cursos (R1.4).

4.3.3.4 Caso de Uso "CriarGrupo"

Através do *CriarGrupo*, o professor responsável por um curso pode criar um grupo de alunos que estejam previamente cadastrados no TUTA. Esse Caso de Uso atende a seguinte funcionalidade:

- Permitir a criação de um grupo (R1.5).

[20] As funcionalidades do TUTA encontram-se nos quadros 4.1 e 4.2. Cf. seção 4.3.2.

4.3.3.5 Caso de Uso "GerenciarEstrDidáticas"

Através do *GerenciarEstrDidáticas*, o professor responsável por um curso pode definir (através de inclusões, alterações e exclusões) as estratégias didáticas necessárias para o curso ministrado por ele. Esse Caso de Uso atende a seguinte funcionalidade:

- Permitir que o professor especifique as estratégias didáticas para o curso ministrado por ele (R1.6).

4.3.3.6 Caso de Uso "GerenciarEntidDidáticas"

Através do *GerenciarEntidDidáticas*, o professor responsável por um curso pode definir (através de inclusões, alterações e exclusões) as entidades didáticas necessárias para o curso ministrado por ele. Esse Caso de Uso atende a seguinte funcionalidade:

- Permitir que o professor especifique as entidades didáticas a serem utilizadas pelo curso ministrado por ele (R1.7).

4.3.3.7 Caso de Uso "NotificarSessão"

Esse Caso de Uso é automaticamente disparado pelo TUTA para notificar alunos e professor que participam de um curso sobre a ocorrência de sessão e atende as seguintes funcionalidades:

- Recuperar data e hora de ocorrência de sessão (R2.1).
- Notificar antecipadamente os alunos e professor participantes de um curso sobre ocorrência de sessão (R1.8).

4.3.3.8 Caso de Uso "NotificarAltCurso"

Esse Caso de Uso é automaticamente disparado pelo TUTA para notificar alunos sobre as alterações relacionadas a um curso ou as suas sessões e atende as seguintes funcionalidades:

- Notificar os alunos sobre alterações no curso em que são participantes (R1.9).
- Notificar os alunos sobre alterações nas sessões do curso que participam (R1.10).

4.3.3.9 Caso de Uso "NotificarExcCurso"

Esse Caso de Uso é automaticamente disparado pelo TUTA para notificar alunos sobre as exclusões relacionadas a um curso ou as suas sessões e atende as seguintes funcionalidades:

- Notificar os alunos sobre possível exclusão do curso em que participam (R1.11).
- Notificar os alunos sobre possíveis exclusões das sessões do curso em que participam (R1.12).

4.3.3.10 Caso de Uso "IniciarSessão"

Através do *IniciarSessão* é possível que o alunos e professor participem efetivamente das sessões de um curso. Esse Caso de Uso atende as seguintes funcionalidades:

- Permitir *login* no sistema para alunos cadastrados (na data de sessão) antes da hora prevista para o início de sessão (R2.2).
- Permitir *logout* no sistema em qualquer momento da sessão (R2.3).
- Validar dados de entrada do aluno (*login, senha*) (R2.4).

4.3.3.11 Caso de Uso "RecuperarEstrDidáticas "

Esse Caso de Uso atende a seguinte funcionalidade:

- Recuperar a(s) estratégia(s) didática(s) necessárias para o acontecimento de uma sessão (R2.6).

4.3.3.12 Caso de Uso "ExecutarEstrDidáticas "

Esse Caso de Uso atende as seguintes funcionalidades:

- Executar todos os passos de uma estratégia (anteriormente especificadas pelo professor) (R2.7).
- Avaliar respostas dos alunos (R2.12).

4.3.3.13 Caso de Uso "RecuperarEntidDidáticas "

Esse Caso de Uso atende a seguinte funcionalidade:

- Recuperar os objetos didáticos necessários para a execução de uma estratégia (R2.8).

4.3.3.14 Caso de Uso "ApresentarInformação"

Esse Caso de Uso atende as seguintes funcionalidades:

- Enviar informações para os alunos, conforme execução da estratégia didática (R2.9).
- Fornecer uma interface amigável para permitir que alunos e professores interajam entre si (R2.5).
- Permitir que em qualquer momento o aluno possa interromper a sessão e expor suas dúvidas para o professor e demais alunos (R2.11).

4.3.3.15 Caso de Uso "EnviarInformação"

Esse Caso de Uso atende as seguintes funcionalidades:

- Recuperar informações dos alunos (R2.10).
- Fornecer uma interface amigável para permitir que alunos e professores interajam entre si (R2.5).
- Permitir interações (através de debates síncronos) entre os alunos de um grupo e o professor (R2.17).

4.3.3.16 Caso de Uso "AtualizarPerfil"

Esse Caso de Uso atende as seguintes funcionalidades:

- Atualizar o perfil do aluno (através das notas) (R2.13).
- Atualizar o perfil do grupo (através das notas de todos os alunos) (R2.14).
- Classificar os alunos através de zonas de comportamento (através da suas notas) (R2.15).
- Realizar a mobilidade lógica dos alunos através das zonas de comportamento (através da notas) (R2.16).

- Permitir que o professor possa acompanhar o rendimento do grupo no qual ele é responsável (R1.13).

- Permitir que os alunos possam acompanhar o seu próprio rendimento, bem como o rendimento do seu grupo (R1.14).

4.4 ARQUITETURA GERAL DO TUTA

A arquitetura do TUTA, ilustrada na Figura 4.8, apresenta os seguintes componentes básicos:

- *Aluno:* qualquer pessoa interessada em um dado domínio, previamente trabalhado no TUTA. O aluno desempenhará atividades diversas, conforme o funcionamento das estratégias didáticas de uma sessão, tais como: receber informações, realizar avaliações, questionar o professor e interagir com outros alunos.

- *Professor:* qualquer professor interessado em utilizar o TUTA para auxiliá-lo em suas tarefas. O professor desempenhará atividades de duas categorias, onde a primeira consiste em atividades de *definição* do curso: especificação de dados do curso e das sessões que compõem o curso; definição das estratégias didáticas e das entidades didáticas que serão utilizadas (domínio); e registro de alunos que podem participar de tal curso. A segunda categoria consiste em atividades relacionadas à execução de uma sessão, tais como: tirar dúvidas dos alunos, participar de debates síncronos e avaliar questões.

- *Administrador do Sistema:* pessoa(s) responsável (eis) pela autorização e inclusão de professores no sistema.

- *Grupo:* conjunto de pessoas (alunos e professor) que se reúnem para participar das sessões de treinamento de um curso.

- *Agente Interface Aluno:* representa o elo de ligação entre o aluno e o TUTA, encarregando-se de realizar tudo que for necessário para a execução de uma sessão de treinamento, inclusive permitindo a interação entre os alunos. Cada aluno participante de um curso possui seu próprio *Agente Interface Aluno*.

- *Agente Interface Professor:* representa um dos elos de ligação entre o professor e o TUTA, encarregando-se de permitir a interação entre o professor e os alunos durante uma sessão de treinamento

- *Agentes Interfaces de Sessão:* conjunto de interfaces (*Agentes Interface Aluno e Agente Interface Professor*) a serem utilizados em uma sessão de treinamento.

- *Agente Interface Especificação Curso* (ou *Agente de Autoria*): representa outro elo de ligação entre o professor e o TUTA, encarregando-se de permitir que um professor responsável por um curso especifique dados sobre o curso que pretende ministrar, as sessões, as estratégias didáticas e os objetos didáticos que serão utilizados (domínio), bem como registrar os alunos que podem participar de tal curso.

- *Domínio:* armazena o conteúdo a ser ensinado ao aluno, ou seja, as entidades de natureza didática (exemplo: definições, exemplos, questões de escolha múltipla, questionários).

- *Aluno:* armazena os dados gerais do aluno e o seu perfil.

- *Grupo:* armazena os dados gerais do grupo e o seu perfil.

- *Estratégias Didáticas:* armazena as estratégias didáticas a serem utilizadas nas sessões de treinamento de um curso.

- *Conjunto de Agentes do TUTA:* coleção de agentes que podem cooperar entre si, a fim de permitir a realização de uma sessão de treinamento e assim promover a aquisição de perícias em um dado domínio pelo aluno. Esses agentes foram concebidos conforme a abordagem reativa, introduzida por Brooks (Brooks, 1986). Conforme esta abordagem, a inteligência emerge da combinação de entidades simples.

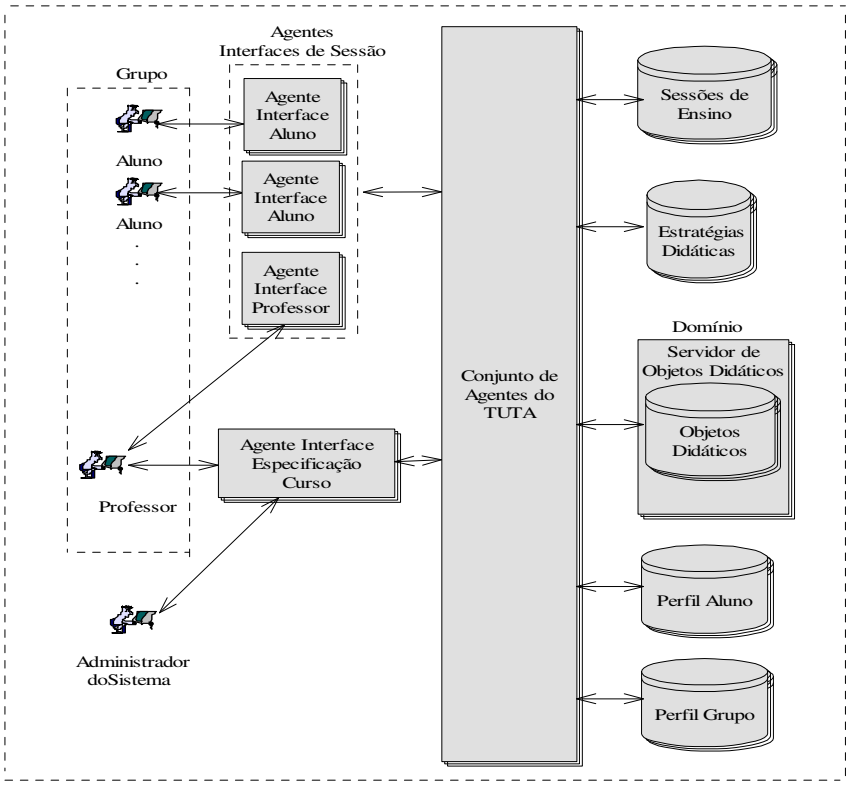


Figura 4.8: Arquitetura Geral do TUTA

4.5 FUNCIONAMENTO GERAL

Na Figura 4.8 são representadas duas unidades básicas: os componentes do TUTA e o relacionamento de interação entre eles (representado pelas setas). Desta forma, baseado nesses relacionamentos, será apresentado a seguir uma visão geral envolvendo um cenário de funcionamento das interações (visão de alto-nível) que podem ocorrer no TUTA.

Presuma que um *Professor* possua interesse em utilizar o TUTA para auxiliá-lo e então notifica o *Administrador do Sistema* a respeito do seu interesse. O *Administrador do Sistema*, após analisar a solicitação do *Professor*, autoriza-o a utilizar o sistema. A partir de então, o *Professor* poderá a qualquer momento, realizar a especificação de um curso; sessões; estratégias e entidades didáticas a serem utilizadas; e registrar os alunos e o grupo de alunos que podem participar do curso. Essa especificação deve ocorrer através do *Agente Interface Especificação Curso.* O *Agente Interface Especificação Curso,* por sua vez, solicitará que um dos agentes pertencente ao *Conjunto de Agentes* armazene os dados em suas respectivas bases de dados (*Sessões de Ensino*, *Estratégias Didáticas*, *Domínio*, *Perfil Aluno* e *Perfil Grupo*). A partir de então, torna-se possível a execução das sessões de um curso. Cada *Aluno*, após receber uma notificação de que haverá sessão, pode utilizar o sistema para participar de tal sessão, juntamente com os demais membros do *Grupo* e o *Professor*. A participação dos Alunos em uma sessão de treinamento do TUTA acontece através dos *Agentes Interfaces de Sessão*. Cada aluno possui o seu *Agente Interface Aluno* e o *Professor* também possui o seu *Agente Interface Professor.* A seguir, o *Conjunto de Agentes do TUTA* começa então a agir, conforme uma *Estratégia Didática* especificada pelo *Professor*.

4.6 CONCLUSÕES E APLICAÇÕES DO TUTA

O sistema tutor proposto não representa uma proposta de substituir o professor e sim de auxiliá-lo. O TUTA representa uma ferramenta de ensino de auxílio ao professor, podendo ser utilizado como parte de um curso completo (com partes teóricas e práticas), em que o professor pode necessitar que o aluno seja treinado em um determinado domínio da parte prática, por exemplo. Para isso, o TUTA poderá oferecer ferramentas de apoio ao professor (ferramentas de autoria), permitindo que ele especifique o conteúdo desejado, assim como escolher a forma com que o material será apresentado através das estratégias didáticas. O professor poderá também monitorar o andamento dos alunos no curso (por exemplo, através da análise das zonas de comportamento ocupadas pelos alunos).

O sistema tutor proposto poderá executar diferentes estratégias didáticas para uma mesma sessão, conforme as especificações realizadas por um professor. Por exemplo, o professor pode especificar que após uma avaliação, a nota global do grupo será analisada para

que uma decisão possa ser tomada: ou continuar executando a mesma estratégia ou trocar para uma outra estratégia, caso a nota global do grupo não esteja atingindo um certo limite aceitável pelo professor.

O TUTA executa diferentes estratégias didáticas, sem necessidade de alteração no seu código de implementação, uma vez que possui a potencialidade de interpretar diferentes estratégias didáticas especificadas por um professor. O comportamento do TUTA é dinâmico (conforme a Estratégia Didática que está sendo executada) e adaptativo (em função do ritmo de progressão dos alunos).

No contexto da ACVA, o TUTA representa um Controlador de apenas um grupo de alunos, inserido na Camada de *SF_Grupo*. Esse controlador comunica-se com a Camada de *SF_Básicos*, a fim de recuperar os objetos didáticos necessários à execução de uma sessão de treinamento.

Capítulo 5
Modelagem do TUTA

Neste capítulo, apresenta-se o processo de desenvolvimento utilizado neste trabalho, discute-se sucintamente a necessidade de uma metodologia orientada a agentes, apresenta-se então a modelagem do TUTA orientada a agentes e a arquitetura detalhada do TUTA e finalmente realiza-se outra etapa de modelagem: a modelagem orientada a objetos .

5.1 INTRODUÇÃO

Modelos servem para capturar os aspectos importantes do que está sendo modelado. Diversas áreas como Engenharia, Arquitetura e Computação, utilizam-se de modelos para expressarem o que pretendem desenvolver. Assim, neste capítulo pretende-se realizar a modelagem de um sistema de *software*. Para isso, é necessário seguir um método de desenvolvimento[20]. Um método de desenvolvimento é composto (no mínimo) de uma linguagem de modelagem e de um processo de desenvolvimento (Fowler & Scott, 1998). A linguagem de modelagem oferece um conjunto de convenções notacionais que podem ser utilizadas para descrever ou modelar uma aplicação e o processo de desenvolvimento descreve os passos que devem ser seguidos para desenvolver uma aplicação.

5.2 PROCESSO DE DESENVOLVIMENTO UTILIZADO NO TUTA

Neste trabalho, segue-se um processo de desenvolvimento proposto por (Larman, 1998), que, por sua vez, segue os seguintes princípios (Booch, 1996):

- iteratividade;
- incrementabilidade;
- direcionamento por casos de uso;
- ênfase na arquitetura (o desenvolvimento da arquitetura do sistema, deve acontecer o mais cedo possível).

Tal processo, realça que o desenvolvimento deve ocorrer através de ciclos de desenvolvimento iterativos, orientado por casos de uso, ou seja, em cada ciclo, o desenvolvedor deve escolher quais os casos de uso do *software* que serão realizados, conforme ilustrado na Figura 5.1.

[20] Neste trabalho utilizou-se mais de um método de desenvolvimento, como será visto no restante do capítulo.

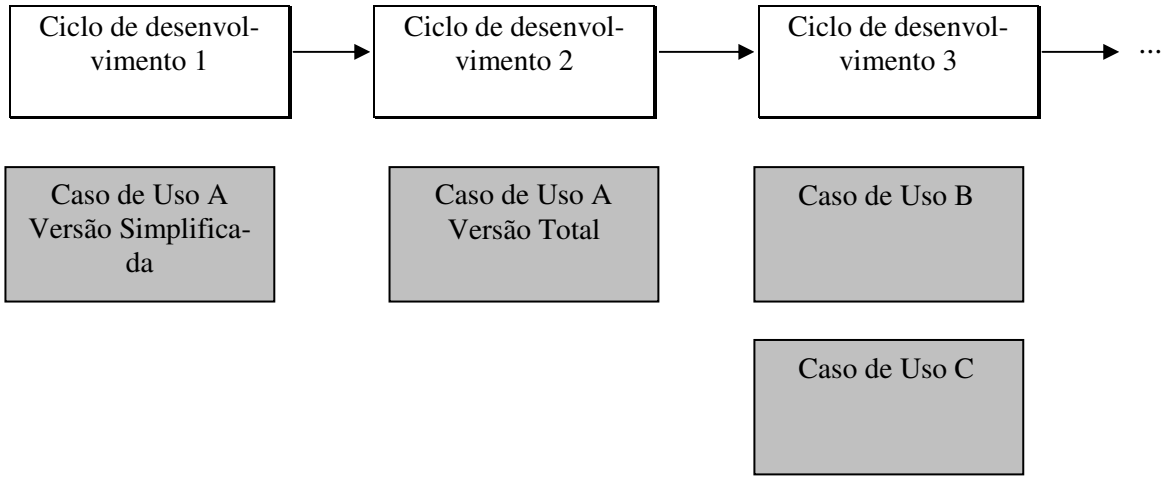


Figura 5.1: Ciclos de desenvolvimento orientados por casos de uso

Neste processo, cada ciclo de desenvolvimento deve ser composto das seguintes fases: análise, projeto e implementação, conforme ilustrado na Figura 5.2.

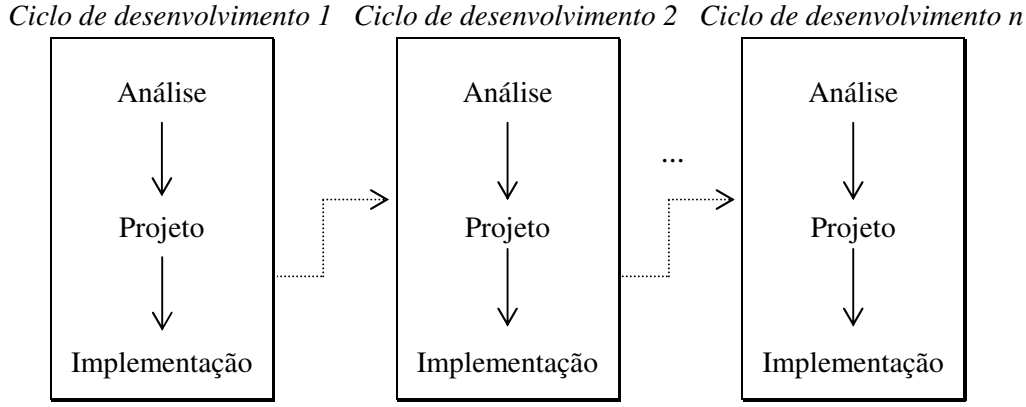


Figura 5.2: Ciclos de desenvolvimento iterativos

O processo proposto por (Larman, 1998) propõe também as atividades que devem compor as fases de análise, projeto e implementação dos ciclos de desenvolvimento. Entretanto, com a utilização da tecnologia de agentes, constatou-se a necessidade de modelos e processos de desenvolvimento apropriados para o seu desenvolvimento. Assim, o processo de desenvolvimento seguido neste trabalho originou dois tipos de modelagem: Modelagem Orientada a Agentes (MOA) e Modelagem Orientada a Objetos (MOO). A MOO tem como entrada os modelos resultantes da MOA (Silva & Hernández-Domínguez, 1999). Desta forma, os ciclos de desenvolvimento seguidos neste trabalho são melhor ilustrados pela Figura 5.3.

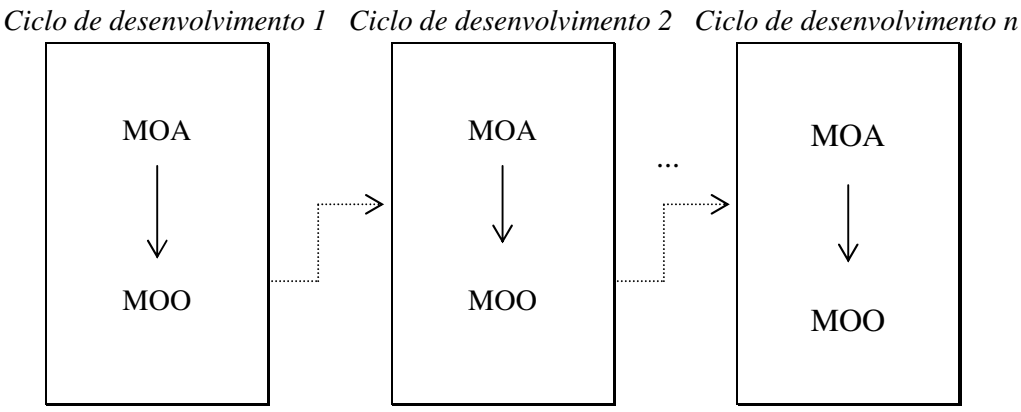


Figura 5.3: Ciclos de desenvolvimento seguidos no desenvolvimento do TUTA

5.2.1 CICLOS DE DESENVOLVIMENTO DO TUTA

O conjunto de casos de uso[21] utilizados nos ciclos de desenvolvimento realizados neste trabalho são mostrados na Figura 5.4. Tais casos de uso são responsáveis pelas funções de execução de sessão do TUTA[22], que permitem que alunos e professor sejam notificados da ocorrência de sessão, para que então de fato ela possa ser executada e possibilite a participação de seus integrantes (alunos e professor).

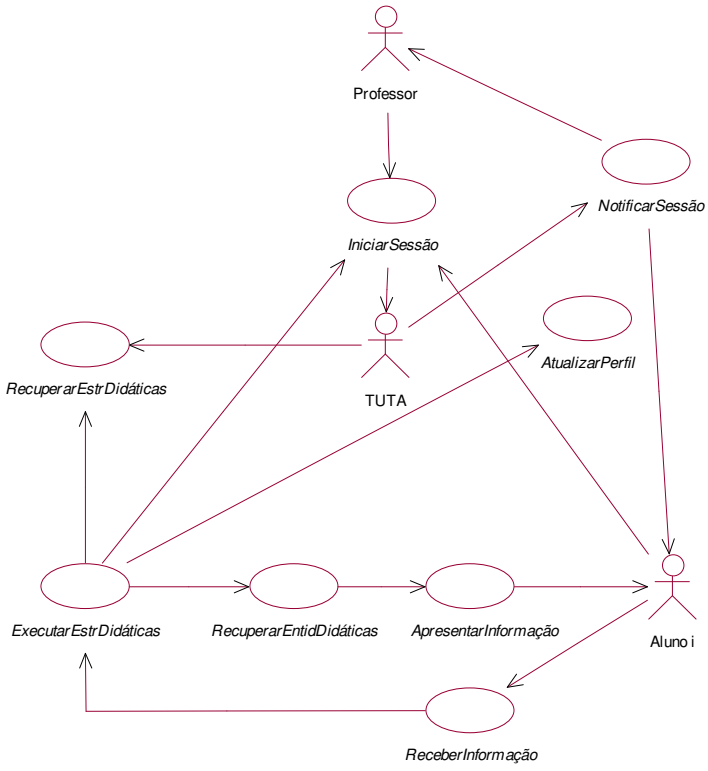


Figura 5.4: Casos de Uso utilizados no desenvolvimento do TUTA

[21] Tais casos de uso estão inseridos no diagrama de casos de uso do TUTA (cf. seção 4.3.3).
[22] Cf. seção 4.3.1.

Desta forma, a alocação dos casos de uso da figura 5.5 aos ciclos de desenvolvimento seguidos no TUTA, estão ilustrados na Figura 5.5.

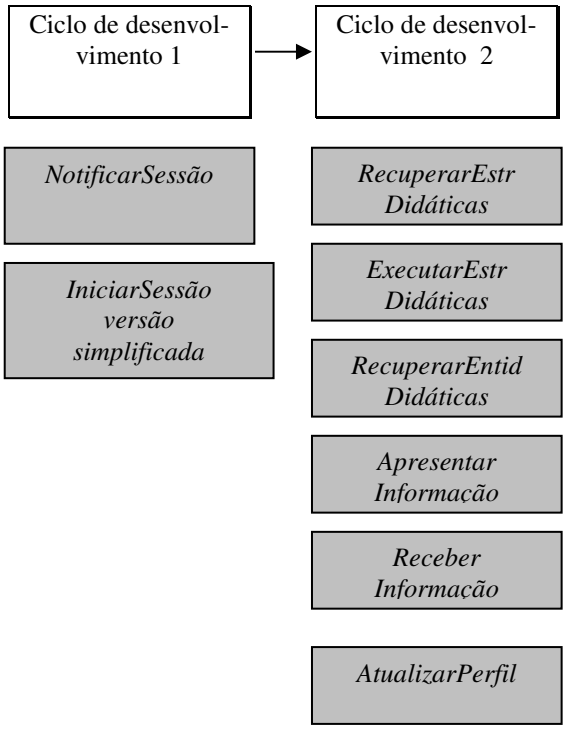


Figura 5.5: Alocação dos casos de uso aos ciclos de desenvolvimento do TUTA

A seguir (nas seções 5.4 à 5.6), serão apresentados os modelos realizados para alcançar o desenvolvimento dos casos de uso acima.

5.3 METODOLOGIA PARA ANÁLISE E PROJETO ORIENTADA A AGENTES

O paradigma orientado a agentes é uma abordagem promissora para desenvolver aplicações complexas, pois representam um importante avanço em matéria de abstração, permitindo uma compreensão, modelagem e desenvolvimento mais natural. Entretanto, é interessante seguir uma metodologia para o desenvolvimento de agentes. Segundo (Wooldridge *et al*., 1999) é necessário que uma metodologia capture aspectos tais como: a flexibilidade do agente, o seu comportamento autônomo, as interações e a complexidade das estruturas organizacionais nas quais os agentes estão inseridos. Assim, neste trabalho utiliza-se inicialmente a metodologia proposta por (Wooldrige *et al.*, 1999), especificamente desenvolvida para a análise e projeto de sistemas orientada a agentes. Essa metodologia é composta de duas fases: análise e projeto. Maiores detalhes sobre a metodologia e os seus modelos podem ser encontrados no Apêndice B.

5.4 MODELAGEM DO TUTA ORIENTADA A AGENTES

A modelagem do TUTA, baseada na metodologia citada na seção 5.3, consiste em considerá-lo como uma *"sociedade"* ou *"organização artificial"* composta por um conjunto de papéis, tal qual uma organização humana. Assim, partiu-se da análise das funções do sistema, bem como do seu funcionamento geral, a fim de realizar o mapeamento entre os elementos e os papéis, uma vez que partir da descrição comportamental das operações (funcionamento do sistema) para uma visão organizacional acontece diretamente.

Assim, nesta modelagem, os elementos do TUTA descritos no Capítulo 4 (Interface, Servidor de Entidades Didáticas, Estratégias Didáticas, Comportamento do Aluno, Comportamento do Grupo e Coordenador de Grupo), são comparados a departamentos, responsáveis pela execução das atividades de treinamento do TUTA. A identificação dos papéis aos elementos do sistema está ilustrada na Figura 5.6.

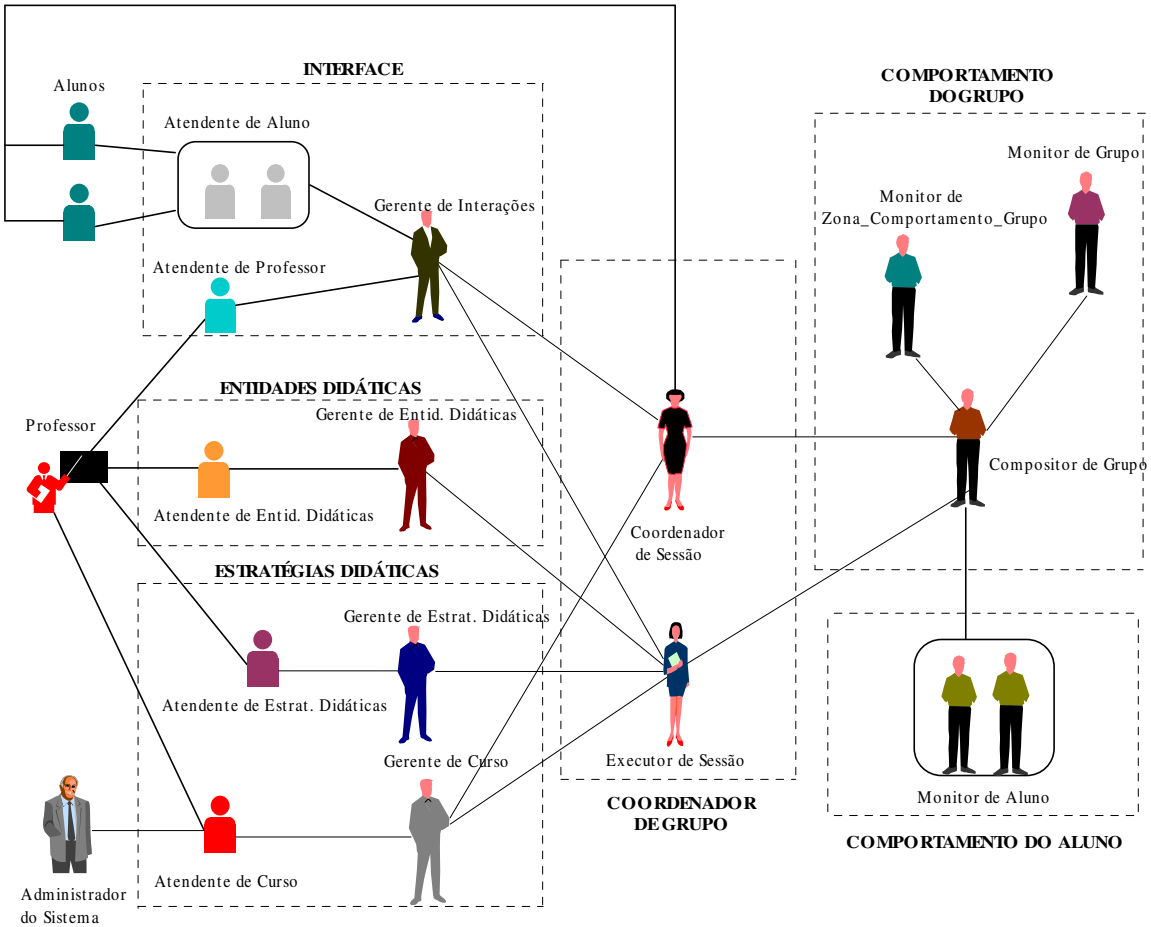


Figura 5.6: Identificação dos Elementos x Papéis do TUTA

5.4.1 MODELOS DE ANÁLISE

5.4.1.1 Modelo de Papéis

O modelo de papéis identifica os papéis fundamentais no sistema. Os papéis são caracterizados por dois tipos de atributos:

- As permissões/direitos associados com o papel: um papel terá associado a ele certas permissões, relativas ao tipo e a quantidade de recursos que podem ser explorados ao desempenhar o papel;

- As responsabilidades do papel: um papel é criado para fazer algo, ou seja, um papel tem uma certa funcionalidade. Esta funcionalidade é representada por um atributo conhecido como *responsabilidades do papel.* Existem dois tipos de responsabilidades:

  – Responsabilidades *Liveness (ou vitais):* são aquelas que dizem que "algo será feito" , e, conseqüentemente que o agente o qual desempenha o papel ainda está vivo;
  – Responsabilidades de segurança estão relacionadas aos requisitos de segurança.

A partir da identificação dos papéis realizada na seção anterior, foi elaborado o modelo de papéis do TUTA composto por um conjunto de esquemas de papéis, um para cada papel no sistema, representados nos Quadros 5.1 a 5.18.

- **PAPEL:** *ALUNO*

ESQUEMA DO PAPEL:	*Aluno*
DESCRIÇÃO:	Pessoa interessada em participar de treinamentos sobre um dado domínio
PROTOCOLOS:	*ParticiparAtividadesAluno*
PERMISSÕES:	
RESPONSABILIDADES	
LIVENESS:	*Aluno = ParticiparAtividadesAluno*

Quadro 5.1: Esquema do Papel *Aluno*

- **PAPEL:** *PROFESSOR*

ESQUEMA DO PAPEL:	*Professor*
DESCRIÇÃO:	Responsável pelas atividades relacionadas à de definição de curso e execução de sessão
PROTOCOLOS:	*DefinirCurso, ParticiparExecSessão*
PERMISSÕES:	
RESPONSABILIDADES	
LIVENESS:	*Professor = ParticiparAtividadesProfessor* *ParticiparAtividadesProfessor = DefinirCurso|ParticiparExecSessão*

Quadro 5.2: Esquema do Papel *Professor*

- **PAPEL:** *ADMINISTRADOR DO SISTEMA*

ESQUEMA DO PAPEL:	*Administrador do Sistema*
DESCRIÇÃO:	Responsável pela autorização e manutenção de professores no sistema
PROTOCOLOS:	*RealizarAtividadesAdministrador*
PERMISSÕES:	
RESPONSABILIDADES	
LIVENESS:	*Administrador do Sistema = RealizarAtividadesAdministrador*

Quadro 5.3: Esquema do Papel *Administrador do Sistema*

- **PAPEL:** *ATENDENTE DE CURSO*

ESQUEMA DO PAPEL:	*Atendente de Curso*
DESCRIÇÃO:	Responsável por permitir que: – um administrador do sistema realize manutenções no cadastro de professores; e – um professor cadastrado no sistema realize manutenções de cursos, de sessões de um curso e de alunos que podem participar de um curso.
PROTOCOLOS:	*ObterDadosProfessor, ObterDadosCurso, ObterDadosSessõesCurso, ObterDadosAluno, GerenciarRegistroProfessores, GerenciarCurso, GerenciarSessõesCurso, GerenciarAlunoGrupo*
PERMISSÕES:	
RESPONSABILIDADES:	
LIVENESS:	*Atendente de Curso=Tarefa_Curso* *Tarefa_Curso = (ObterDadosProfessor||GerenciarRegistroProfessores)|* *(ObterDadosCurso||GerenciarCurso)|* *(ObterDadosSessõesCurso||GerenciarSessõesCurso)|* *(ObterDadosAluno||GerenciarAlunoGrupo)*
SEGURANÇA:	Assegurar que os dados sejam preenchidos corretamente

Quadro 5.4: Esquema do Papel *Atendente de Curso*

- **PAPEL:** *ATENDENTE DE ESTRATÉGIAS DIDÁTICAS*

ESQUEMA DO PAPEL:	*Atendente de Estratégias Didáticas*
DESCRIÇÃO:	Responsável por permitir que um professor cadastrado no TUTA solicite a manutenção da(s) estratégia(s) didática(s) que regem o funcionamento de uma sessão
PROTOCOLOS:	*ObterEstratégiasDidáticas, GerenciarEstratégiasDidáticas*
PERMISSÕES:	
RESPONSABILIDADES	
LIVENESS:	*Atendente de Estratégias Didáticas = (ObterEstratégiasDidáticas||GerenciarEstratégiasDidáticas)*
SEGURANÇA:	Assegurar que os dados sejam preenchidos corretamente

Quadro 5.5: Esquema do Papel *Atendente Estratégias Didáticas*

- **PAPEL:** *ATENDENTE DE ENTIDADES DIDÁTICAS*

ESQUEMA DO PAPEL:	*Atendente de Entidades Didáticas*
DESCRIÇÃO:	Responsável por permitir que um professor cadastrado no TUTA solicite a manutenção das entidades didáticas necessárias para a execução das estratégias didáticas de uma sessão de treinamento.
PROTOCOLOS:	*ObterEntidadesDidáticas, GerenciarEntidadesDidáticas*
PERMISSÕES:	
RESPONSABILIDADES	
LIVENESS:	*Atendente de Entidades Didáticas = (ObterEntidadesDidáticas||GerenciarEntidadesDidáticas)*
SEGURANÇA:	Assegurar que os dados sejam preenchidos corretamente

Quadro 5.6: Esquema do Papel *Atendente de Entidades Didáticas*

- **PAPEL:** *ATENDENTE DE ALUNO*

ESQUEMA DO PAPEL:	*Atendente de Aluno*
DESCRIÇÃO:	Responsável por permitir a participação dos alunos nas sessões de treinamento de um curso.
PROTOCOLOS:	*NotificarSessão, IniciarSessão, ApresentarInformação, ReceberInformação*
PERMISSÕES:	
RESPONSABILIDADES	
LIVENESS:	*Atendente de aluno = NotificarSessão| IniciarSessão | ApresentarInformação | ReceberInformação*
SEGURANÇA:	Assegurar que os dados sejam preenchidos corretamente.

Quadro 5.7: Esquema do Papel *Atendente de Aluno*

- **PAPEL:** *ATENDENTE DE PROFESSOR*

ESQUEMA DO PAPEL:	*Atendente de Professor*
DESCRIÇÃO:	Responsável por permitir a participação dos alunos nas sessões de treinamento de um curso.
PROTOCOLOS:	*NotificarSessão, IniciarSessão*
PERMISSÕES:	
RESPONSABILIDADES	
LIVENESS:	*Atendente de Professor = NotificarSessão| IniciarSessão*
SEGURANÇA:	Assegurar que os dados sejam preenchidos corretamente

Quadro 5.8: Esquema do Papel *Atendente de Professor*

- **PAPEL:** *GERENTE DE CURSO*

ESQUEMA DO PAPEL:	*Gerente de Curso*
DESCRIÇÃO:	Responsável pelo gerenciamento dos dados e das sessões de um curso; dos professores que podem utilizar o TUTA; e ocorrência de sessões.
PROTOCOLOS:	*GerenciaRegistroProfessores, GerenciarCurso, GerenciarSessõesCurso, NotificarExistênciaSessão*
PERMISSÕES:	Ler dados do professor // ou dados do curso // ou dados das sessões de um curso // ou Alterar dados do professor // ou dados do curso // ou dados das sessões de um curso // ou
RESPONSABILIDADES:	
LIVENESS:	*Gerente de Curso = GerenciarRegistroProfessores|GerenciarCurso| GerenciarSessõesCurso|NotificarExistênciaSessão*
SEGURANÇA:	Assegurar a consistência e integridade dos dados

Quadro 5.9: Esquema do Papel *Gerente de Curso*

- **PAPEL:** *GERENTE DE ENTIDADES DIDÁTICAS*

ESQUEMA DO PAPEL:	*Gerente de Entidades Didáticas*
DESCRIÇÃO:	Responsável pelo gerenciamento das entidades didáticas necessárias para a execução das estratégias didáticas em uma sessão de treinamento.
PROTOCOLOS:	*GerenciarEntidadesDidáticas*
PERMISSÕES:	Ler Entidades Didáticas Alterar Entidades Didáticas
RESPONSABILIDADES	
LIVENESS:	*Gerente de Entidades Didáticas = RecuperarEntidadesDidáticas*
SEGURANÇA:	Assegurar a consistência e integridade dos dados.

Quadro 5.10: Esquema do Papel *Gerente de Entidades Didáticas*

- **PAPEL:** *GERENTE DE ESTRATÉGIAS DIDÁTICAS*

ESQUEMA DO PAPEL:	*Gerente de Estratégias Didáticas*
DESCRIÇÃO:	Responsável pelo gerenciamento das Estratégias Didáticas necessárias para a execução de uma sessão de treinamento.
PROTOCOLOS:	*GerenciarEstratégiasDidáticas*
PERMISSÕES:	Ler Estratégias Didáticas Alterar Estratégias Didáticas
RESPONSABILIDADES	
LIVENESS:	Gerente de Estratégias Didáticas = *RecuperarEstratégiasDidáticas*
SEGURANÇA:	Assegurar a consistência e integridade dos dados

Quadro 5.11: Esquema do Papel *Gerente de Estratégias Didáticas*

- **PAPEL:** *GERENTE DE INTERAÇÕES*

ESQUEMA DO PAPEL:	*Gerente de Interações*
DESCRIÇÃO:	Responsável pelo gerenciamento das informações enviadas/recebidas por/para um *Atendente de Professor* ou por/para um *Atendente de Aluno*
PROTOCOLOS:	*GerenciarInformaçõesEnviadasParaAluno, GerenciarInformaçõesRecebidasDoAluno, GerenciarInformaçõesEnviadasParaProfessor, GerenciarInformaçõesRecebidasDoProfessor*
PERMISSÕES:	
RESPONSABILIDADES	
LIVENESS:	*Gerente de Interações = (GerenciarInformaçõesEnviadasParaAluno| GerenciarInformaçõesRecebidasDoAluno| GerenciarInformaçõesEnviadasParaProfessor| GerenciarInformaçõesRecebidasDoProfessor)*

Quadro 5.12: Esquema do Papel *Gerente de Interações*

- **PAPEL:** *COORDENADOR DE SESSÃO*

ESQUEMA DO PAPEL:	*Coordenador de Sessão*
DESCRIÇÃO:	Responsável pela recepção dos alunos em uma sessão de treinamento.
PROTOCOLOS:	*InicializarAlunosSessão*, *FinalizarAlunosSessão*
PERMISSÕES:	
RESPONSABILIDADES	
LIVENESS:	*Coordenador de Sessão = RecepcionarAlunosSessão* *RecepcionarAlunosSessão = InicializarAlunosSessão| FinalizarAlunosSessão*

Quadro 5.13: Esquema do Papel *Coordenador de Sessão*

- **PAPEL:** *EXECUTOR DE SESSÃO*

ESQUEMA DO PAPEL:	*Executor de Sessão*
DESCRIÇÃO:	Responsável pelas atividades necessárias para a execução de uma sessão de treinamento
PROTOCOLOS:	*AguardarHoraSessão, RecuperarEstratégiasDidáticas, ExecutarEstratégiasDidáticas, RecuperarEntidadesDidáticas, EnviarInformação, ReceberInformação*
PERMISSÕES:	Ler Estratégia Didática
RESPONSABILIDADES LIVENESS:	*Executor de Sessão = AguardarHoraSessão. RecuperarEstratégiasDidáticas.(ExecutarEstratégiasDidáticas. RecuperarEntidadesDidáticas.EnviarInformação.ReceberInformação)* ∞

Quadro 5.14: Esquema do Papel *Executor de Sessão*

- **PAPEL:** *MONITOR DE ALUNO*

ESQUEMA DO PAPEL:	*Monitor de Aluno*
DESCRIÇÃO:	Responsável pelo gerenciamento do perfil do aluno durante um curso
PROTOCOLOS:	*AtualizarPerfilAluno*
PERMISSÕES:	Ler dados do aluno Alterar dados do aluno
RESPONSABILIDADES LIVENESS:	*Monitor de Aluno = AtualizarPerfilAluno*
SEGURANÇA:	Assegurar a consistência e integridade dos dados

Quadro 5.15: Esquema do Papel *Monitor de Aluno*

- **PAPEL:** *MONITOR DE GRUPO*

ESQUEMA DO PAPEL:	*Monitor de Grupo*
DESCRIÇÃO:	Responsável por monitorar o perfil do grupo durante um curso
PROTOCOLOS:	*AtualizarPerfilGrupo*
PERMISSÕES:	Ler dados do grupo Alterar dados do grupo
RESPONSABILIDADES LIVENESS:	*Monitor de Grupo = AtualizarPerfilGrupo*
SEGURANÇA:	Assegurar a consistência e integridade dos dados

Quadro 5.16: Esquema do Papel *Monitor de Grupo*

- **PAPEL:** *MONITOR DE ZONA DE COMPORTAMENTO DE GRUPO*

ESQUEMA DO PAPEL:	*Monitor de Zona de Comportamento de Grupo*
DESCRIÇÃO:	Responsável pelo gerenciamento da zona de comportamento dos alunos de um grupo durante um curso
PROTOCOLOS:	*Atualizar Z_CompAlunoGrupo*
PERMISSÕES:	Ler *Z_CompAlunoGrupo* Alterar *Z_CompAlunoGrupo*
RESPONSABILIDADES	
LIVENESS:	*Monitor de Zona de Comportamento Grupo = AtualizarZ_CompAlunoGrupo*
SEGURANÇA:	Assegurar a consistência e integridade dos dados

Quadro 5.17: Esquema do Papel *Monitor de Zona de Comportamento de Grupo*

- **PAPEL:** *COMPOSITOR DE GRUPO*

ESQUEMA DO PAPEL:	*Compositor de Grupo*
DESCRIÇÃO:	Responsável pelo gerenciamento de dados dos alunos de um curso.
PROTOCOLOS:	*GerenciarAlunoGrupo*
PERMISSÕES:	Ler dados do aluno Alterar dados do aluno
RESPONSABILIDADES	
LIVENESS:	*Compositor de Grupo = GerenciarAlunoGrupo*
SEGURANÇA:	Assegurar a consistência e integridade dos dados

Quadro 5.18: Esquema do Papel *Compositor de Grupo*

5.4.1.2 Modelo de Interações

As interações que existem entre os diversos papéis em uma organização multiagente são capturadas nesta fase de análise. O modelo de interações consiste de um conjunto de definições de protocolo, uma para cada tipo de interação *Inter-papel*. Segundo (Wooldridge *et al*., 1999), um protocolo pode ser visto como um padrão de interação institucionalizado, ou seja, um padrão de interação que foi formalmente definido e abstraído independentemente de qualquer sucessão particular de passos de execução. Vendo as Interações deste modo, significa que a atenção é focalizada na natureza essencial e no propósito da interação, e não na ordem precisa de troca de mensagem particular (como os diagramas de interação usualmente utilizados nos diagramas orientados a objetos).

Nas figuras 5.7 a 5.11 são representadas algumas das interações entre os papéis do TUTA.

- **PROTOCOLO:** *PARTICIPARATIVIDADESALUNO*

Participação em atividades de Aluno	
Aluno	Atendente de Aluno
Participar de atividades relacionadas à execução de uma sessão de treinamento	

Participação efetivada

Figura 5.7: Definição do Protocolo *ParticiparAtividadesAluno*

- **PROTOCOLO:** *GERENCIARESTRATÉGIASDIDÁTICAS*

Gerenciamento de Estratégias Didáticas	
Gerente de Estratégias Didáticas	*Atendente de Estratégias Didáticas*
Realiza a validação dos dados	

Estratégias Didáticas de uma sessão de treinamento

Manutenção em uma Estratégia Didática

Figura 5.8: Definição do Protocolo *GerenciarEstratégiasDidáticas*

- **PROTOCOLO:** *NOTIFICAREXISTÊNCIASESSÃO*

Notificação de existência de sessão	
Gerente de Curso	*Executor de Sessão, Coordenador de Sessão Compositor de Grupo Gerente de Interações*
Enviar Notificação sobre a existência de sessão	

Dia de sessão

Dados gerais da sessões

Figura 5.9: Definição do Protocolo *NotificarExistênciaSessão*

- **PROTOCOLO:** *GERENCIARINFORMAÇÕESENVIADASPARAALUNO*

Gerenciamento das Informações Enviadas para o Aluno	
Gerente de Interações	*Atendente de Aluno*
Enviar informação para os Atendentes de Alunos especificados	

Identificação do aluno
Informação para o aluno

Informação para o aluno

Figura 5.10: Definição do Protocolo *GerenciarInformaçõesEnviadasParaAluno*

- **PROTOCOLO:** *EXECUTARESTRATÉGIASDIDÁTICAS*

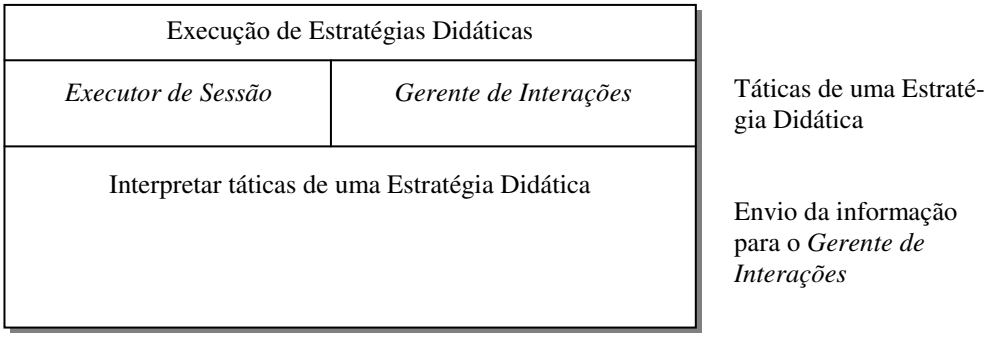


Figura 5.11: Definição do Protocolo *ExecutarEstratégiasDidáticas*

5.4.2 MODELOS DE PROJETO

5.4.2.1 Modelo de Agentes

O propósito do modelo de agentes é documentar os vários tipos de agentes (um tipo de agente representa um conjunto de papéis de agentes) que serão usados no sistema em desenvolvimento e as instâncias que vão representar esses tipos de agentes em tempo-de-execução.

A seguinte cardinalidade pode ser utilizada para representar o número de instâncias dos agentes em tempo-de-execução:

- *n*: significa que existirão não menos que *n* instâncias deste tipo no sistema em tempo-de-execução;
- *m..n*: significa que existirão não menos que *m* e não mais que *n* instâncias deste tipo no sistema em tempo-de-execução;
- *: significa que poderão existir zero ou mais instâncias em tempo-de-execução;
- +: significa que poderão existir uma ou mais instâncias em tempo-de-execução.

O modelo de agentes do TUTA está ilustrado na Figura 5.12. A decisão do empacotamento de papéis em um determinado tipo de agente é um critério do projetista. No TUTA, tal decisão foi tomada tentando absorver papéis que tivessem objetivos comuns em um determinado tipo de agente, por exemplo: os papéis de Atendente de Estratégias Didáticas, Atendente de Entidades Didáticas e Atendente de Curso têm como característica comum auxiliar o professor nas suas tarefas de especificação relacionadas a um curso, por isso decidiu-se encapsular tais papéis em um único Agente: no Agente Interface de Especificação de Curso.

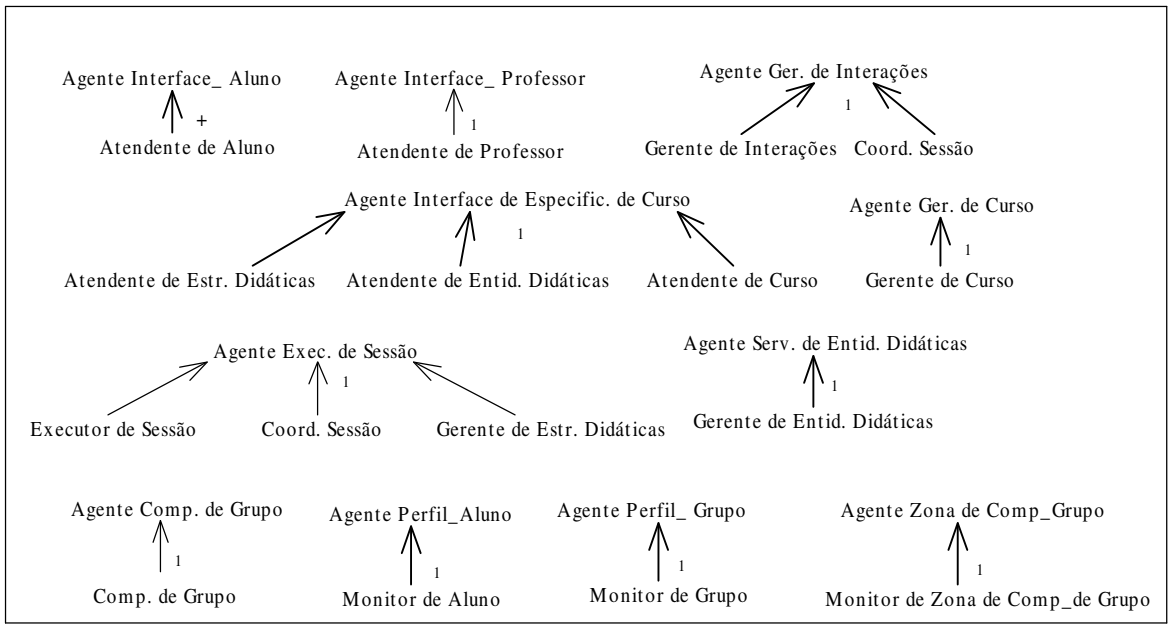


Figura 5.12: Modelo de Agentes do TUTA

5.4.2.2 Modelo de Comunicação

O Modelo de Comunicação define as ligações de comunicação que existem entre os *tipos de agentes* do sistema e não definem que mensagens são enviadas ou quando são enviadas, e indicam apenas os possíveis caminhos de comunicação existentes entre *os tipos de agentes*. O Modelo de Comunicação pode ser derivado diretamente dos papéis, protocolos e do modelo de agentes.

O Modelo de Comunicação do TUTA está ilustrado na Figura 5.13 e foi construído a partir da análise das interações entre os papéis do sistema (Modelo de Interações), mas como esse modelo representa-se apenas as ligações entre tipos de agentes, foi necessário verificar qual o tipo de agente que um determinado papel corresponde (através do Modelo de Agentes). Por exemplo, na Figura 5.8, existe uma interação entre os papéis: *Gerente de Estratégias Didáticas* e *Atendente de Estratégias Didáticas*, mas como o papel *Gerente de Estratégias Didáticas* foi absorvido pelo *Agente Executor de Sessão* e o papel *Atendente de Estratégias Didáticas* foi absorvido pelo *Agente Interface de Especificação de Curso*, é feita uma ligação de comunicação entre esses dois tipos de Agentes.

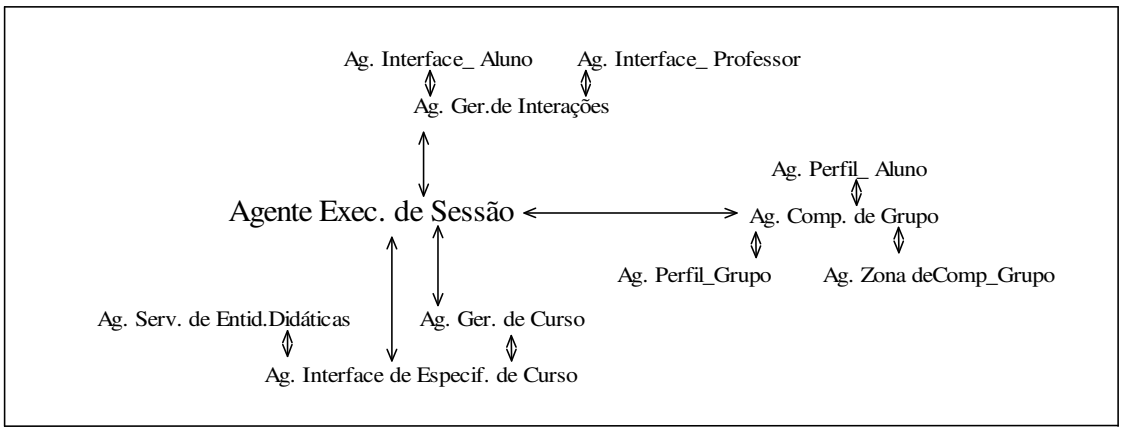


Figura 5.13: Modelo de Comunicação do TUTA

5.5 ARQUITETURA DO TUTA BASEADA EM AGENTES

Após a identificação dos agentes (Modelo de Agentes) e da comunicação entre eles (Modelo de Comunicação), atingiu-se então a arquitetura do TUTA, ilustrada na Figura 5.14. Tal arquitetura possui todos os componentes previstos em um STI[23], mas como pode se observar o módulo de controle está distribuído em alguns módulos, além de algumas funções de controle estarem também embutidas no módulo de estratégias didáticas, representada nessa arquitetura pelo *Agente Executor de Sessão*.

[23] Cf. seção 2.3.1, Figura 2.1.

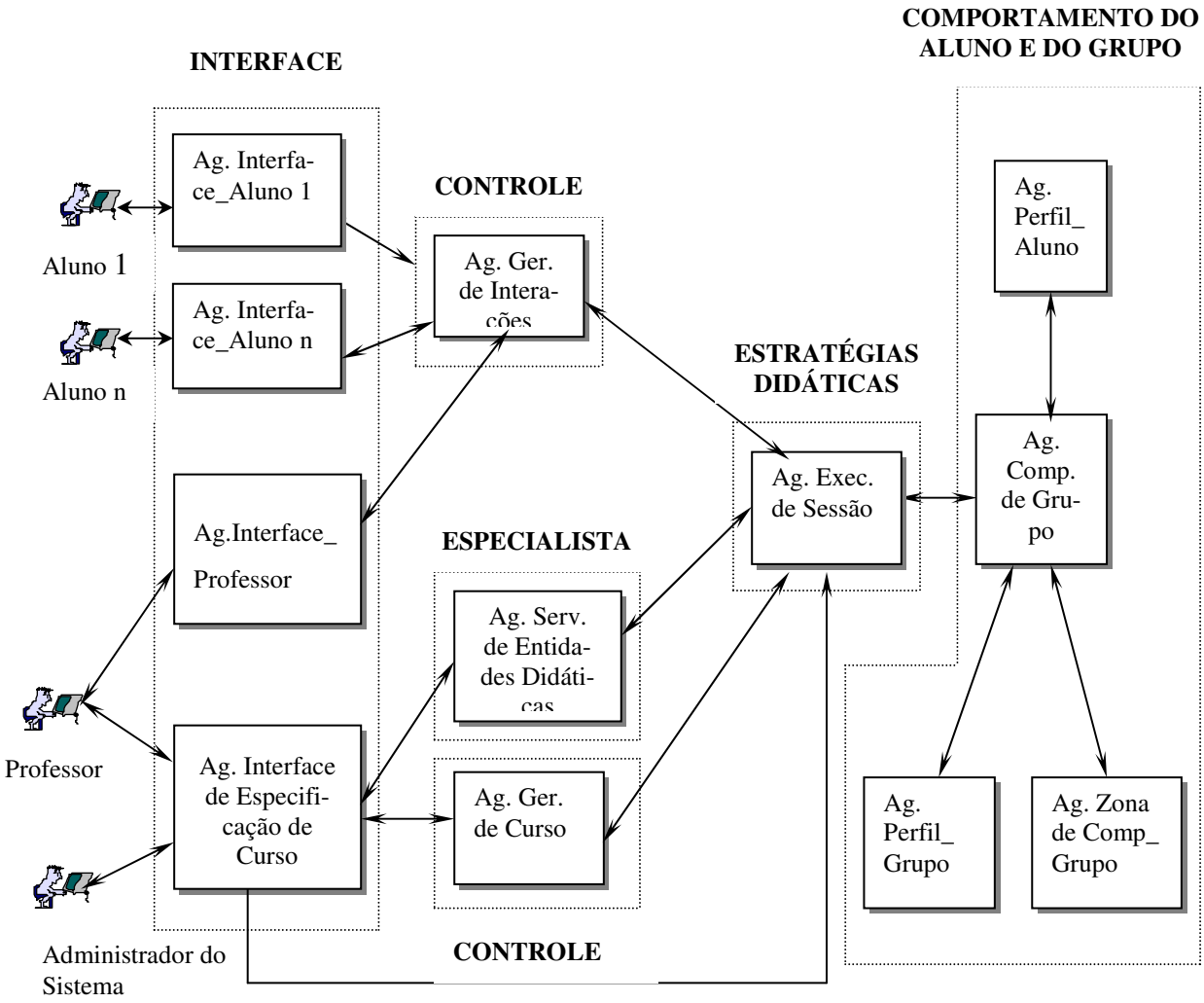


Figura 5.14: Arquitetura detalhada do TUTA

5.6 MODELAGEM ORIENTADA A OBJETOS

Na seção 5.4 realizou-se a modelagem do TUTA orientada a agentes, partindo-se da análise das funções do sistema e do seu funcionamento geral. Entretanto, após a realização dessa modelagem, ainda não é possível partir diretamente para a implementação, pois após a construção destes modelos, é necessário que sejam utilizadas técnicas de projeto tradicionais (neste caso, técnicas orientada a objetos), a fim de que ocorra a implementação desejada (Wooldridge *et al*., 1999).

Assim, esta seção visa ilustrar alguns diagramas desenvolvidos nesta etapa de modelagem orientada a objetos, tais como: definição de casos de uso reais, arquitetura do sistema, diagramas de interações (representados por diagramas de colaboração) e diagramas

de classes (Larman, 1998). Nessa etapa, a arquitetura do TUTA passou por uma ligeira modificação em termos de interações, pois surgiu a necessidade da criação de um novo agente responsável pelo envio e recebimento de mensagens: o Agente de Comunicação. Assim, as interações previstas anteriormente entre quaisquer agentes continua a existir. Entretanto, um Agente não envia mensagens diretamente para outro Agente qualquer do sistema, e sim para o Agente de Comunicação, que trata de repassá-la.

5.6.1 DEFINIÇÃO DE CASOS DE USO REAIS DO TUTA

Casos de Uso Reais representam uma importante ferramenta da fase de projeto e estão bem próximo de "como" as tarefas serão realizadas. Serão ilustrados a seguir, dois casos de uso reais: *IniciarSessão* e *Apresentar informação.* Maiores detalhes sobre casos de uso encontram-se no Apêndice A.

5.6.1.1 Caso de Uso: IniciarSessão

Atores: *TUTA, Estudante i, Professor.*

Descrição: Professor ou alunos apresentam-se no sistema para participar de uma sessão de treinamento.

O Quadro 5.19 ilustra a seqüência típica de eventos do Caso de Uso *IniciarSessão.*

Ações dos Atores	Respostas do Sistema
1. Este caso de uso inicia quando o TUTA ativa a interface do sistema para que alunos e professor apresentem-se no sistema para participar de uma sessão.	
2. Alunos e professores cadastrados no TUTA entram com os seus dados: usuário (área A) e senha (área B).	3. TUTA realiza a validação dos dados.

Quadro 5.19: Seqüência típica de eventos do caso de uso *IniciarSessão*

A figura 5.15 ilustra a Interface referente ao caso de uso *IniciarSessão.*

Figura 5.15: Interface referente ao caso de uso *IniciarSessão*

5.6.1.2 Caso de Uso: "ApresentarInformação"

Atores: TUTA, *Estudante i*

Descrição: Este caso de uso permite a participação do aluno em uma sessão de treinamento.

O Quadro 5.20 ilustra a seqüência típica de eventos do Caso de Uso "*ApresentarInformação" e "ReceberInformação".*

Ações dos Atores	Respostas do Sistema
1. Este caso de uso inicia após a validação da identificação do aluno no sistema	2. O sistema informa o nome do curso e da sessão no campo *A* da interface.
	3. O sistema após recuperar a estratégia didática, executa suas ações, recuperando as entidades didáticas, caso seja necessário. As informações são apresentadas no campo *D* da Interface e caso a ação seja de bate-papo entre alunos, a área *C* é reservada pra isso. O nome dos participantes é exibido na área *B*.
4. O aluno recebe as informações do sistema através do campo *D* da interface.	

Quadro 5.20: Seqüência típica de eventos do caso de uso *ApresentarInformação*

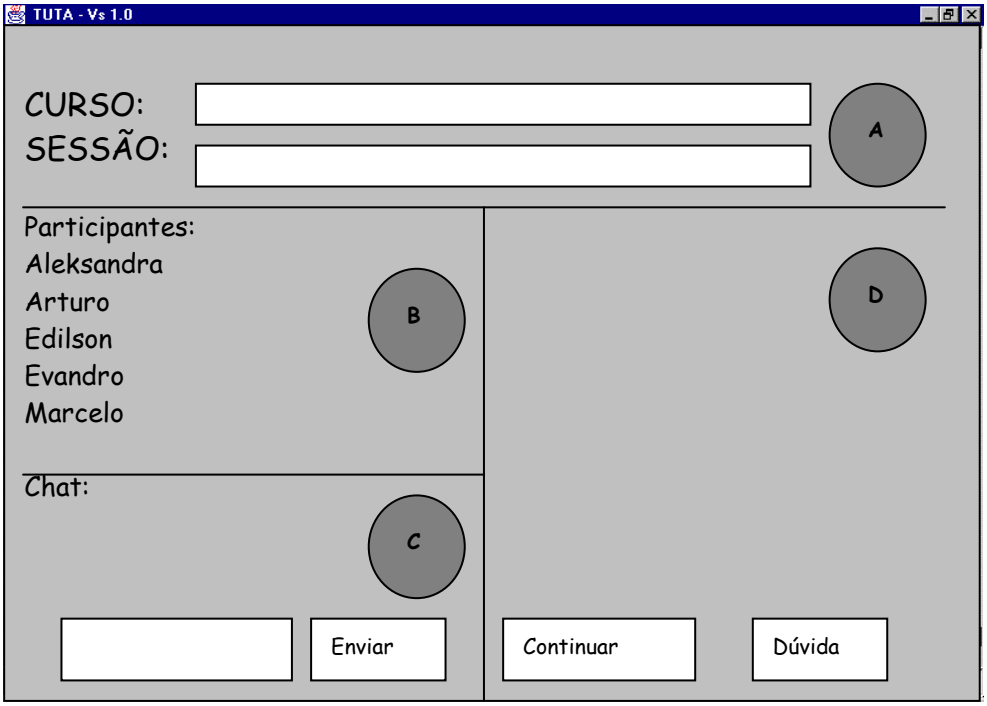


Figura 5.16: Interface Referente ao caso de uso *ApresentarInformação*

5.6.2 ARQUITETURA DO SISTEMA

A Arquitetura descreve a organização e estrutura de um sistema (Larman, 1998). Normalmente, muitos níveis diferentes de arquiteturas estão envolvidos no desenvolvimento de sistemas de *software*, da arquitetura física do hardware à arquitetura lógica de um *framework* de aplicações. Em UML é possível utilizar pacotes para ilustrar cada camada. Mais detalhes sobre o desenvolvimento de Arquitetura em Camadas, encontra-se no Apêndice A.

A figura 5.17 ilustra a Arquitetura do TUTA composta de três camadas: Camada de Apresentação; Camada Lógica da Aplicação, composta da Camada de Domínio e Camada de Serviços; e a Camada de Armazenamento. A composição de cada camada será apresentada na seção 5.6.4.

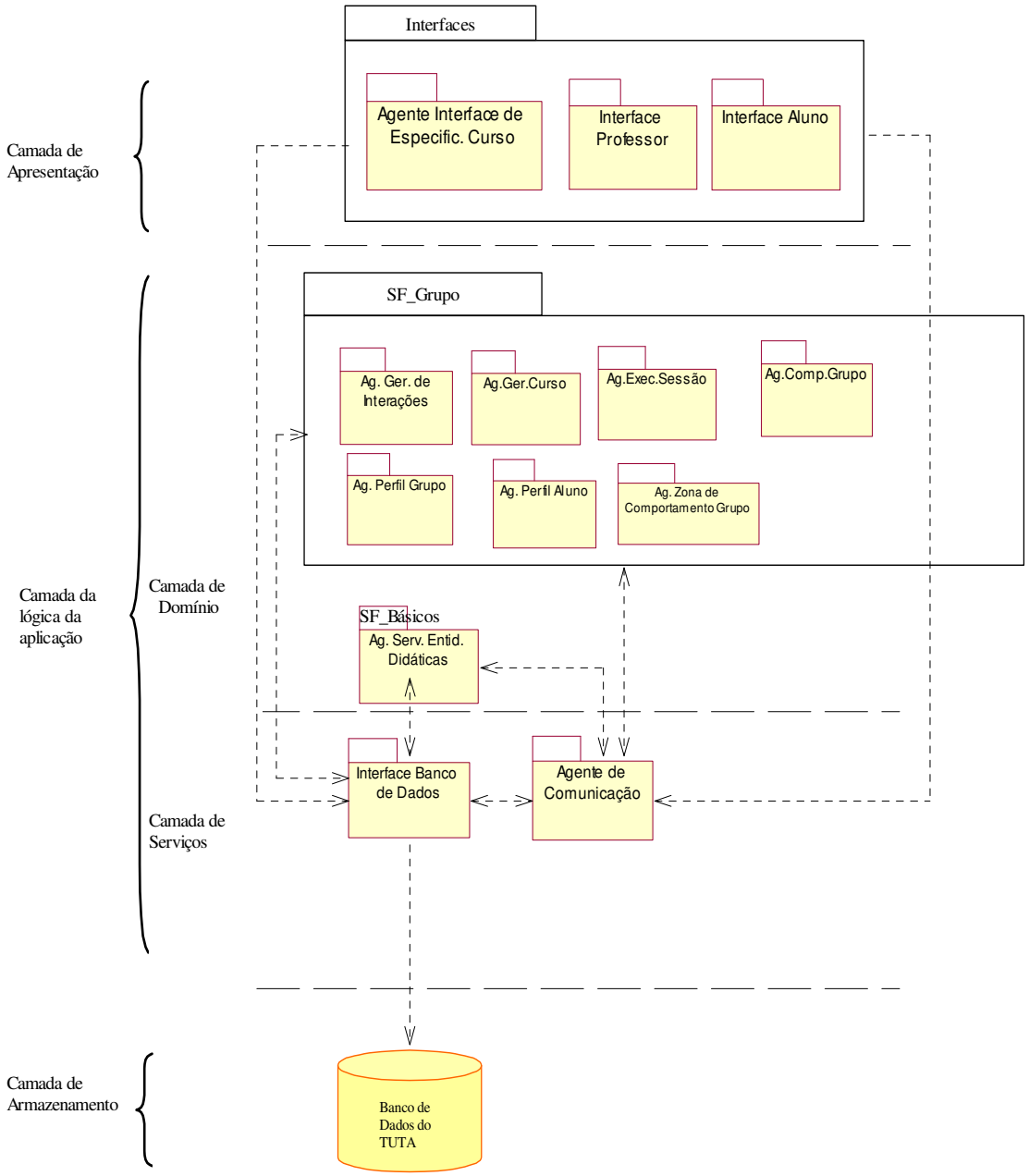


Figura 5.17: Arquitetura do TUTA em Camadas

5.6.3 DIAGRAMAS DE COLABORAÇÃO

Diagramas de Colaboração mostram o fluxo de mensagens entre os objeto de *software* (Larman, 1998). Assim, tornou-se necessário a construção de tais diagramas para uma melhor visualização das mensagens entre os Agentes. As figuras 5.18 à 5.24 ilustram diagramas de colaboração para alguns agentes do TUTA.

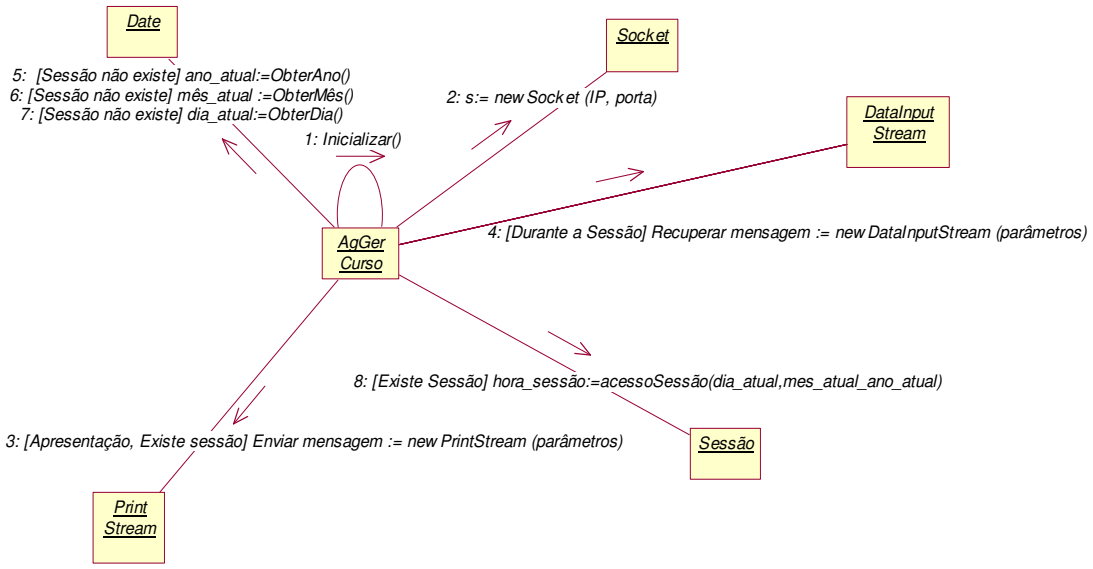


Figura 5.18: Diagrama de Colaboração do Agente Gerenciador de Curso

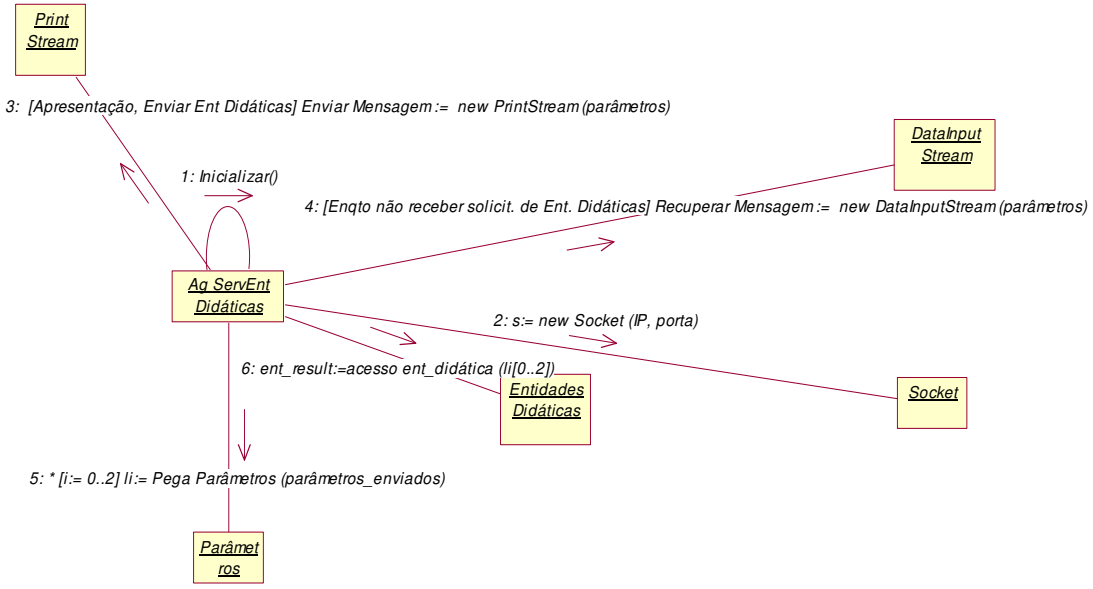


Figura 5.19: Diagrama de Colaboração do Agente Servidor de Entidades Didáticas

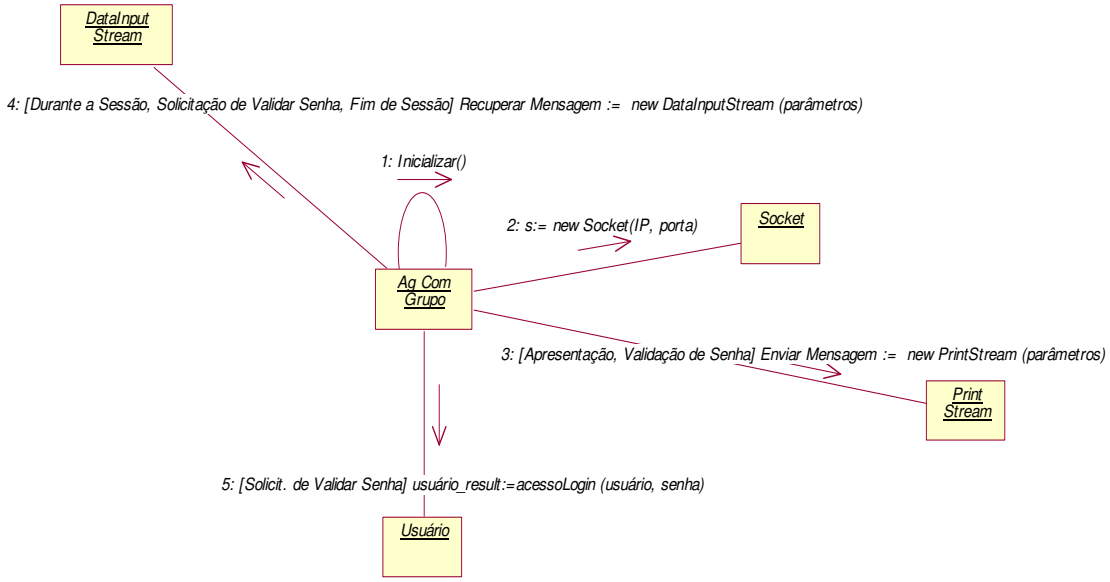


Figura 5.20: Diagrama de Colaboração do Agente Comp. de Grupo

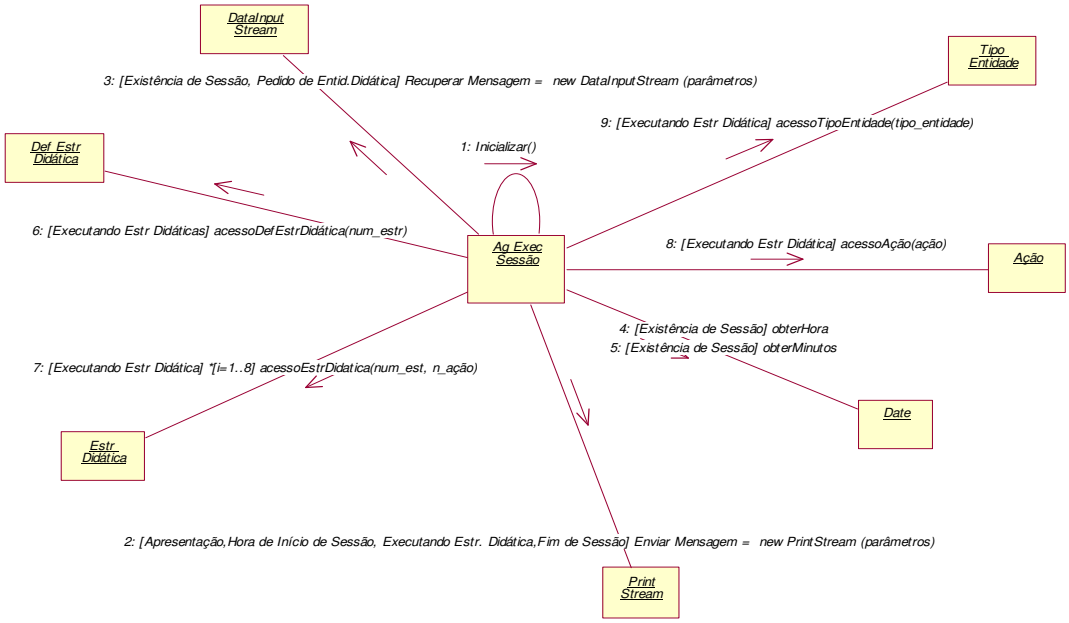


Figura 5.21: Diagrama de Colaboração do Agente Executor de Sessão

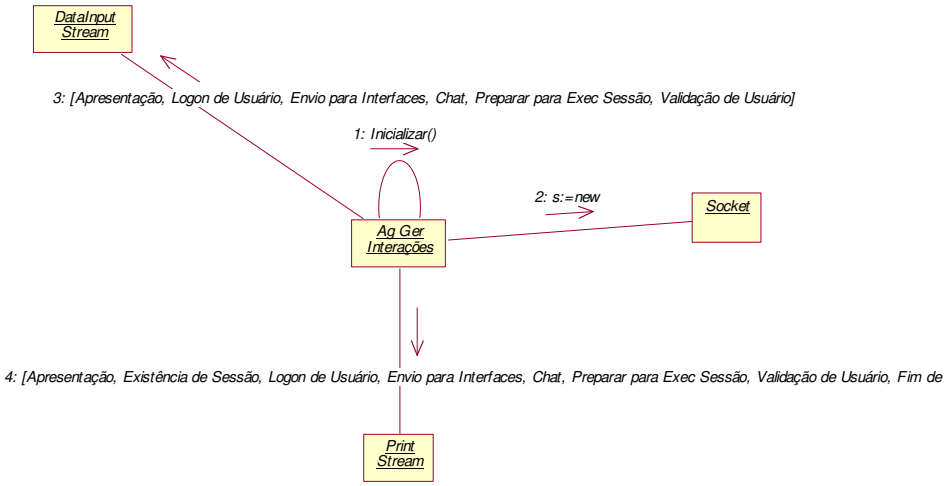


Figura 5.22: Diagrama de Colaboração do Agente Gerenciador de Interações

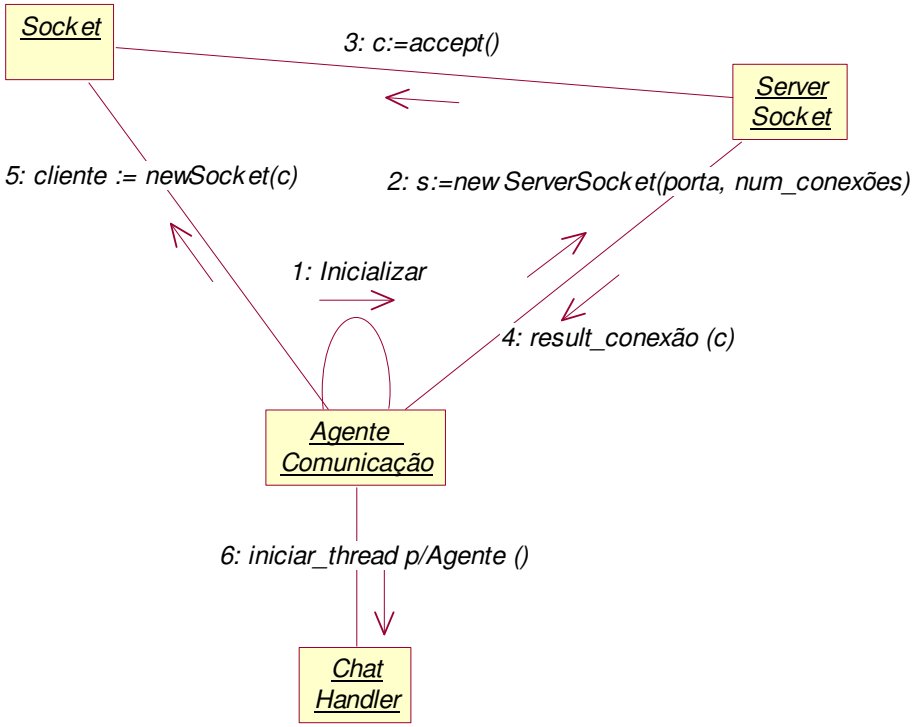


Figura 5.23: Diagrama de Colaboração do Agente de Comunicação

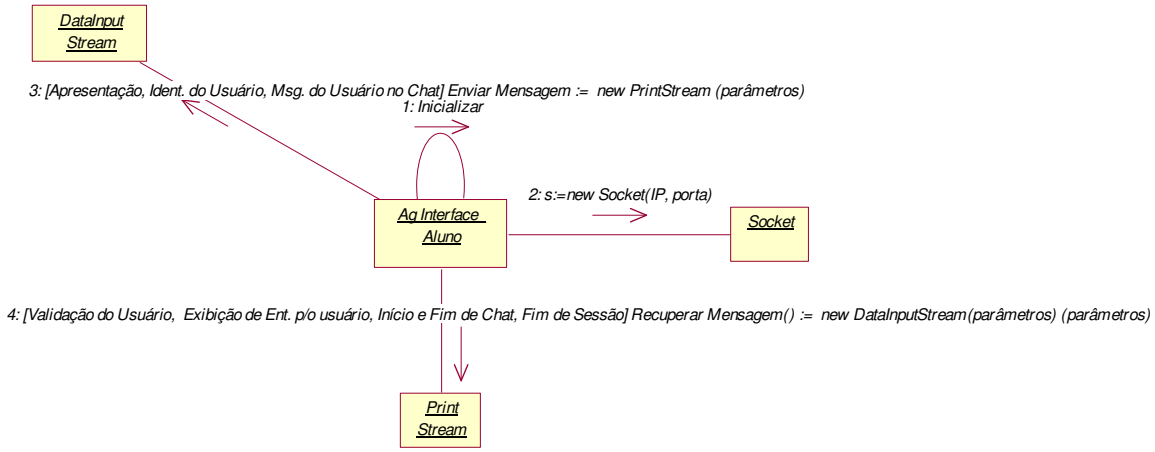


Figura 5.24: Diagrama de Colaboração do Agente Interface Aluno

5.6.4 CAMADAS DA ARQUITETURA DO TUTA E SEUS DIAGRAMAS DE CLASSES

- **Camada de Armazenamento:** Essa camada do TUTA é representada por um Banco de Dados Relacional, utilizando o Sistema Gerenciador de Banco de Dados ACCESS.

- **Camada de Serviços:** A camada de Serviços é representada pelo pacote de Interface com o Banco de Dados e pelo pacote contendo o Agente de Comunicação, responsável pela comunicação entre os demais Agentes da Arquitetura. Para realizar a Interface com o Banco de Dados, está sendo utilizado o pacote JDBC (do inglês: *Java Database Conectivity*) que se conecta com o OBDC (do inglês: *Open Database Connectivity*) para permitir o acesso ao banco de dados. O Diagrama de Classes do Agente de Comunicação está representado na Figura 5.25. Maiores detalhes sobre Diagrama de Classes encontram-se no Apêndice A.

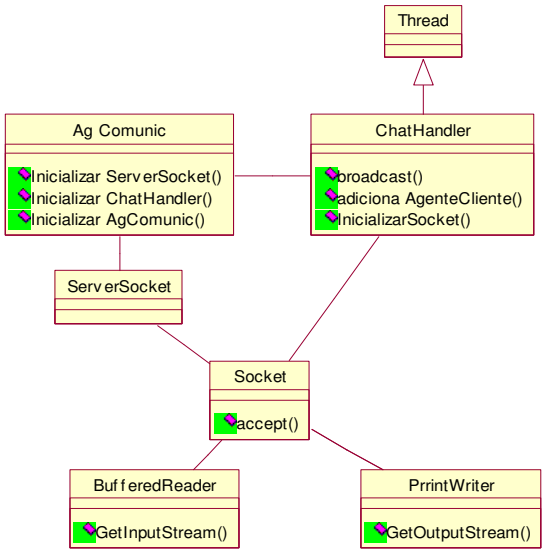


Figura 5.25: Diagrama de Classes do Agente de Comunicação

- **Camada do Domínio:** Essa Camada é composta pelas classes que representam os conceitos do domínio da aplicação. Dessa forma, agruparam-se dois pacotes, onde o primeiro é representado pelos Agentes presentes na camada *SF_Grupo* da ACVA e o segundo pelo Agente Servidor de Entidades Didáticas que faz parte da Camada *SF_Básicos* da ACVA. O Diagrama de Classes completo da Camada do Domínio ou *SF_Grupo* está ilustrado na Figura 5.26.

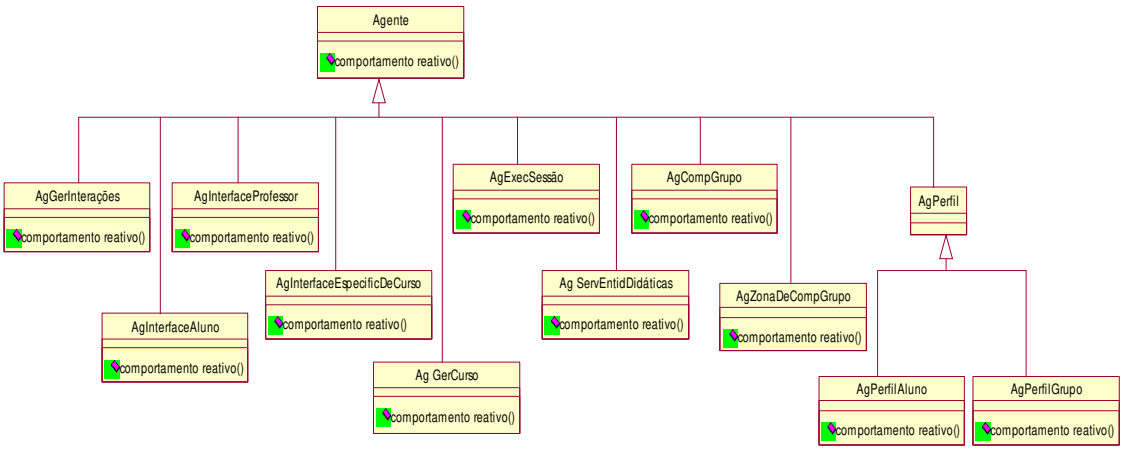


Figura 5.26: Diagrama de Classes da Camada de Domínio ou *SF_Grupo*

As Classes (Agentes) da Camada do Domínio precisam se relacionar com outras classes para realizarem suas tarefas. Assim as figuras 5.27 à 5.31, ilustram o Diagrama de Classes de alguns Agentes.

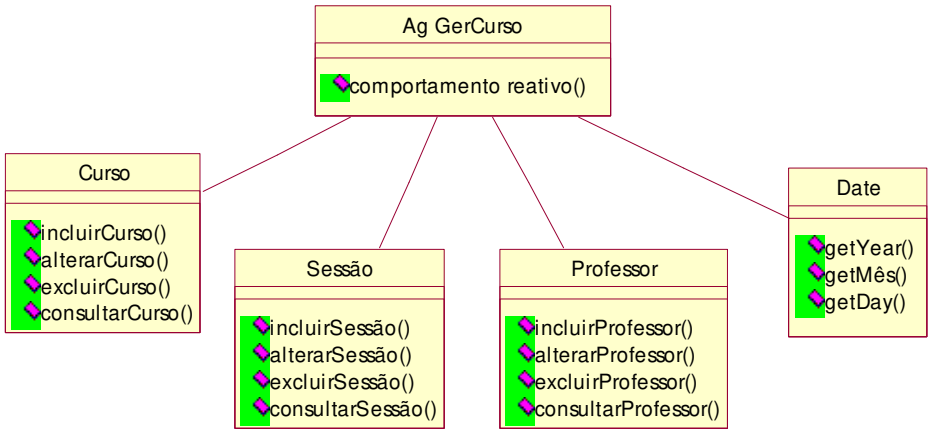


Figura 5.27: Diagrama de Classes do Agente Gerenciador de Curso

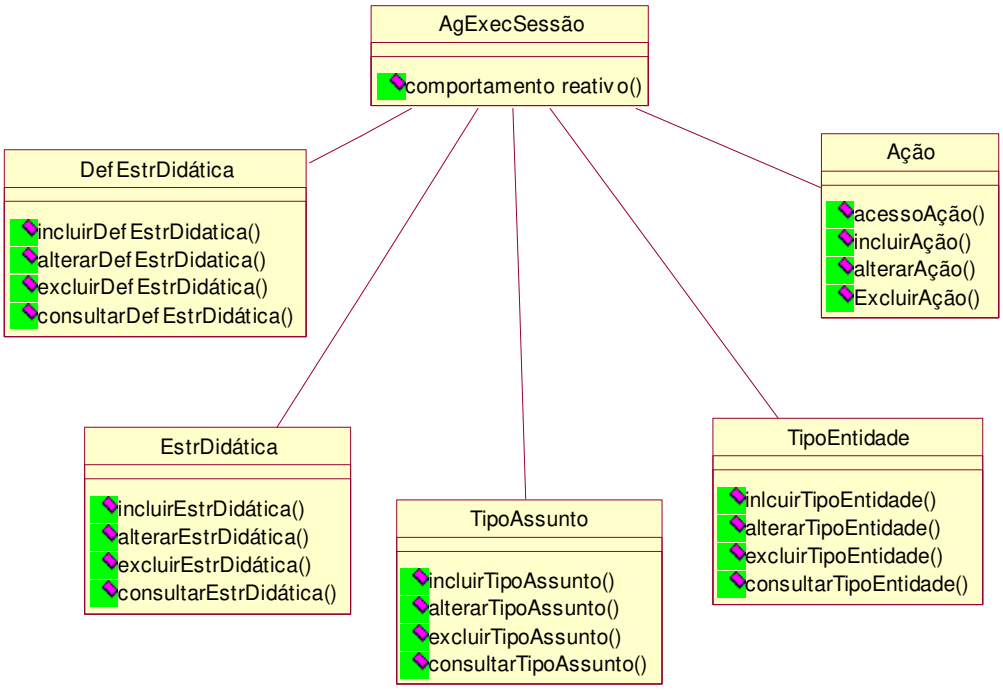


Figura 5.28: Diagrama de Classes do Agente Executor de Sessão

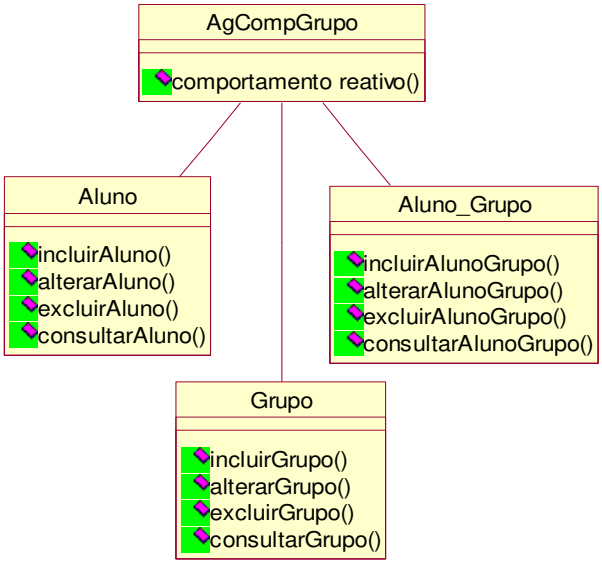


Figura 5.29: Diagrama de Classes do Agente Compositor de Grupo

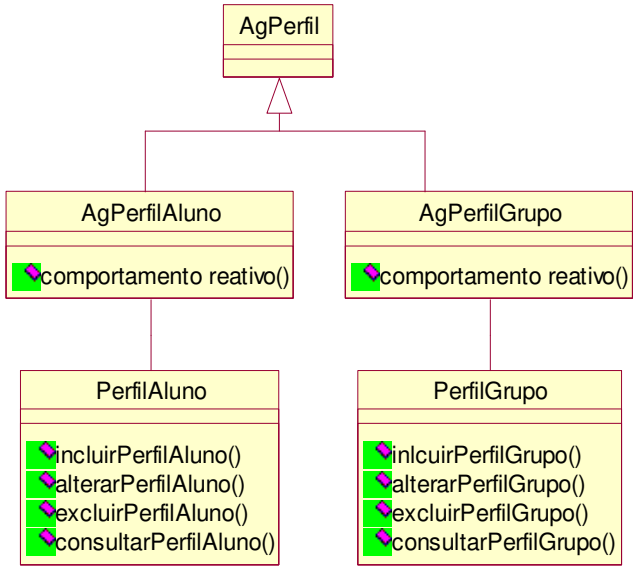


Figura 5.30: Diagrama de Classes dos Agentes Perfil Aluno e Perfil Grupo

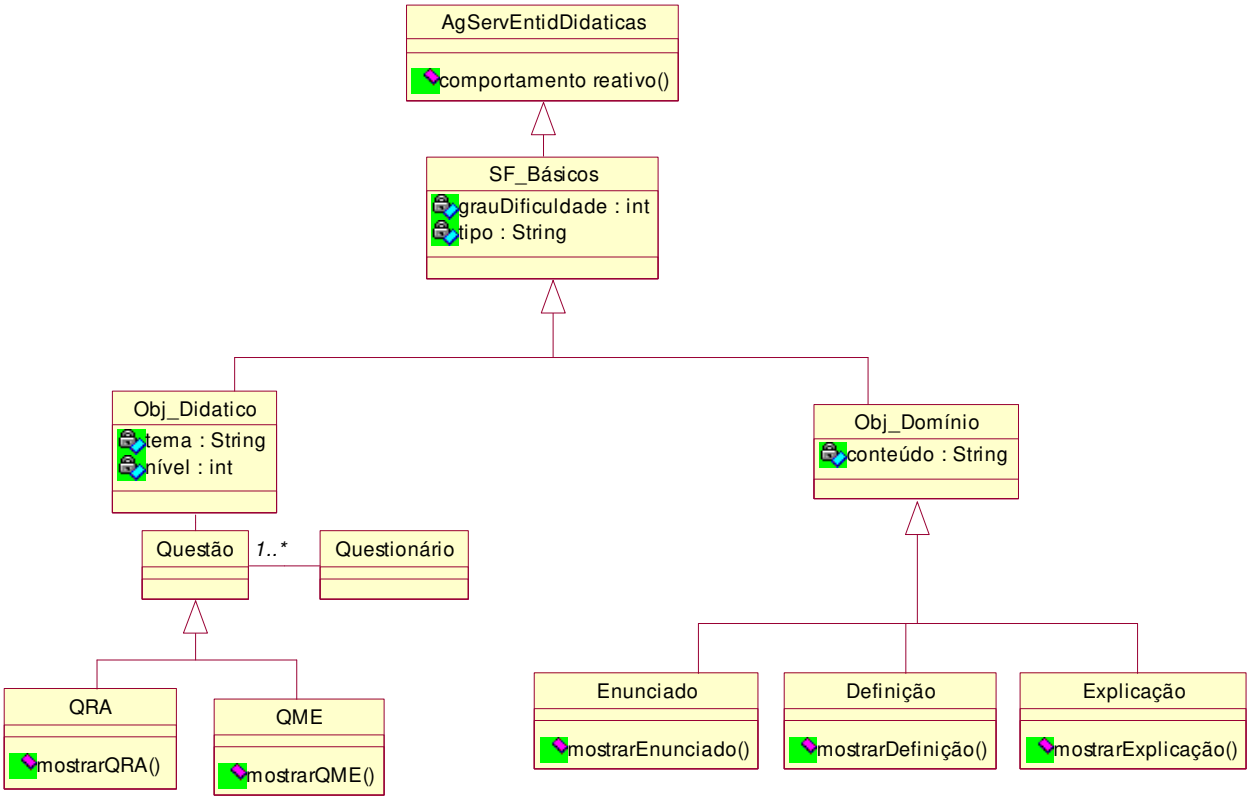


Figura 5.31: Diagrama de Classes do Agente Serv. de Entid. Didáticas

- **Camada de apresentação:** Essa camada é composta pelo Pacote de Interface (Agente Interface de Especificação de Curso, Agente Interface_Professor e Agente Interface_Aluno).

5.7 CONCLUSÕES

A adoção da tecnologia de agentes e a utilização de uma metodologia de análise e projeto orientada a agentes na concepção de uma solução baseada em agentes traz diversos benefícios que podem ser analisados sob diferentes óticas. Esses benefícios podem ser mensurados tanto do ponto de vista da qualidade da solução proposta, ou seja, do produto, cujas características tornam a aplicação mais interessante e atraente para o usuário, quanto do processo de desenvolvimento do software, na forma de uma metodologia mais adequada garantindo maior segurança e confiabilidade ao desenvolvedor.

Do ponto de vista da Engenharia de Software, a utilização de métodos e técnicas que valorizem o conceito de agente como uma poderosa abstração para compreensão e modelagem do mundo real, reduz a complexidade da fase de análise, remetendo o esforço do processo de desenvolvimento de software para as etapas de projeto e implementação. Nesse aspecto, o ganho da seleção da tecnologia de agentes é representado pela redução do *gap* semântico existente entre o mundo real e o modelo do mundo real. Finalmente, com o esforço da comunidade de pesquisa em tecnologia de agentes, espera-se uma rápida evolução dos *frameworks* e linguagens para construção de agentes. Essa evolução representa, para os desenvolvedores, maior agilidade na concepção de sistemas baseados em agentes. Características como reusabilidade e a utilização de padrões de projeto para sistemas baseados em agentes favorecem um processo de desenvolvimento com qualidade e redução de riscos. Além disso, essas características possibilitam a redução de custos, que freqüentemente tem sido citado como um dos entraves para a construção de STI's.

Capítulo 6
Implementação

Neste capítulo, apresentam-se os principais aspectos de implementação do TUTA. Inicialmente, são mostrados os aspectos relevantes em relação a linguagem utilizada, bem como as formas de distribuição e comunicação dos agentes do sistema. Posteriormente, apresenta-se o funcionamento dos agentes no TUTA e os resultados da implementação de uma aplicação.

6.1 A LINGUAGEM UTILIZADA

Conforme mencionado anteriormente[24], Linguagens Orientada a Objetos (LOO) têm sido amplamente difundidas e utilizadas no contexto do desenvolvimento de agentes computacionais. Assim, no contexto deste trabalho, utilizou-se a linguagem *JAVA* para o desenvolvimento dos agentes.

JAVA é uma linguagem orientada a objetos desenvolvida pela *Sun Microsystems*. Como a maioria dos produtos modernos, tal linguagem pode ser bem definida por um conjunto de jargões. Segundo (Flanagan, 1996), Java é uma linguagem simples, orientada a objetos, distribuída, interpretada, potente, segura, de arquitetura neutra, portável, de alto desempenho, multiencadeada e dinâmica.

6.1.1 ASPECTOS GERAIS DA LINGUAGEM

JAVA é uma linguagem com muitas potencialidades. Entretanto, nesta seção, apenas alguns aspectos serão considerados, como[25]:

- *Classe:* um conjunto de variáveis e métodos que um objeto pode possuir, ou um "modelo" para construir objetos.
- *Objetos:* o instanciamento de uma classe.
- *Thread x Runnable: java* é uma linguagem multiencadeada porque permite que um programa possa lidar com múltiplas ações simultaneamente. O *multithreading* é o

[24] Cf. Capítulo 3.
[25] Mais detalhes sobre a linguagem Java encontram-se no Apêndice C.

modo de alcançar isto. Existem duas formas de executar aplicativos e classes em *threads*: estendendo a classe *thread* ou implementando a interface *runnable.*

- *Synchronized:* palavra-chave do *JAVA* que evita a execução simultânea de mais de um *thread* dentro de um método.

- *Socket:* é uma abstração de *software*, uma espécie de *"bocal virtual",* onde pode se conectar para introduzir e retirar bits. Um *socket* existe nos dois pontos finais de uma conexão. O uso de *sockets* neste trabalho será discutido posteriormente.

6.1.2 ASPECTOS DE JAVA QUE SUPORTAM O DESENVOLVIMENTO DE AGENTES

Existem alguns aspectos em Java que facilitam bastante o desenvolvimento de agentes inteligentes, são eles (Bigus & Bigus, 1998):

- Autonomia: para que um programa possa ser considerado autônomo, ele deve ser um processo separado ou *thread.* Aplicações JAVA são processos separados e como tal podem ser considerados autônomos. Java suporta também aplicações *threads.* Assim, Java provê suporte para autonomia, usando ambas as técnicas.

- Inteligência: a inteligência dos agentes inteligentes pode originar-se de simples códigos procedurais ou lógica orientada a objetos, estendendo-se até a utilização de sofisticadas capacidades de raciocínio e aprendizado. De forma geral, Java provê todas as funções básicas necessárias para suportar esses comportamentos. Representação de Conhecimento, como *frames*, redes semânticas e *regras se-então* podem ser facilmente implementadas em Java.

Segundo (Bigus & Bigus, 1998), agentes inteligentes são definidos como programas autônomos ou processos. Desta forma, eles estão sempre esperando, prontos para responder uma solicitação do usuário ou um evento. Um agente sabe quando alguma coisa muda através do envio de um evento. De uma perspectiva de projeto OO, um evento significa uma chamada a um método ou mensagem. As informações necessárias são passadas juntas com o método. Tais informações definem o que aconteceu ou qual ação é necessária que o agente realize, assim como os dados necessários para processar o evento.

Em Java, existem mecanismos de processamento de evento que são usados para enviar os eventos gerados pelo usuário, como o movimentos de mouse, entre outros.

6.2 DISTRIBUIÇÃO DOS AGENTES DO TUTA

Conforme visto na seção 3.4., os agentes do TUTA podem ser classificados segundo as características enumeradas por Wooldridge (Wooldridge & Jennings, 1995b): Autonomia, Reatividade e Habilidade Social. Tais agentes podem ser executados utilizando as seguintes configurações:

- Localizados em uma mesma máquina.
- Localizados em máquinas diferentes no contexto da Internet.

6.3 COMUNICAÇÃO ENTRE OS AGENTES DO TUTA

No TUTA, as mensagem não são enviadas diretamente de um agente para outro, elas são antes encaminhadas para um Agente de Comunicação (conhecido por todos os outros agentes). O Agente de Comunicação envia então cada mensagem que recebe para todos os agentes (ou seja, todos os agentes da sociedade escutam a mensagem e apenas o agente que possui a habilidade para tratar a mensagem é que responde). Esse modo de comunicação é implementado através de *broadcast*[26]. O modo de comunicação utilizado é do tipo um-para-muitos (um transmissor, muitos receptores) (Tanenbaum, 1995). Assim, o Agente de Comunicação assume o papel de transmissor e os demais agentes de receptores. Uma das vantagens de enviar mensagens utilizando *broadcast* é sua simplicidade, uma vez que a mesma mensagem é enviada apenas uma vez, não havendo necessidade de enviá-la várias vezes. A Figura 6.1 ilustra como é realizado esse modo de comunicação.

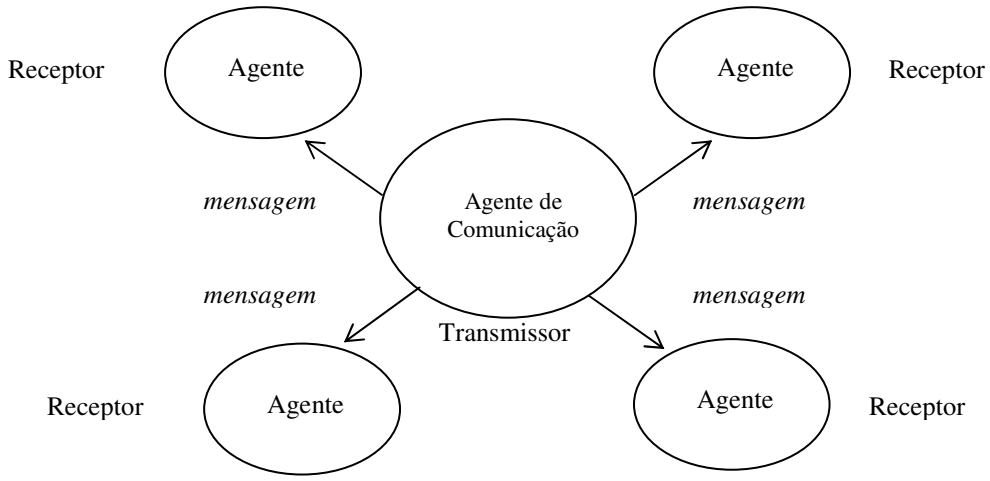


Figura 6.1: Transmissão de uma mensagem no TUTA

[26] Ao utilizar-se *broadcast*, as mensagens são enviadas para todos os agentes, sem distinção, e cada um analisará se sabe tratar a mensagem ou não.

6.3.1 PARALELISMO E CONTROLE DE CONCORRÊNCIA NO AGENTE DE COMUNICAÇÃO

O Agente de Comunicação (ou Agente Servidor[27]) é multiencadeado, ou seja, cada agente (cliente) que se conecta a ele obtém seu próprio *thread*. Entretanto, uma situação que decorre do paralelismo é o acesso concorrente. Para isso, foi utilizada a palavra-chave *"synchronized"* da linguagem JAVA, a fim de evitar que os agentes (representados por *threads* no Agente de Comunicação) tentem acessar métodos ou variáveis compartilhadas.

6.3.2 PRINCIPAIS CLASSES E CONCEITOS UTILIZADOS NA COMUNICAÇÃO

- *Socket*: classe que oferece suporte à criação de conexões em rede. *Sockets* se baseiam em um modelo cliente/servidor. Funciona da seguinte forma: um programa (o servidor) fornece o serviço em uma endereço IP e portas específicos. Desta forma, um programa qualquer que solicite um serviço (um cliente), precisa conhecer o endereço IP e a porta de comunicação do servidor. Assim, o servidor recebe as solicitações dos serviços e atende os pedidos.

- *ServerSocket*: classe JAVA usada para que os programas de servidores (no nosso caso, o Agente de Comunicação) na Internet atendam às solicitações do cliente. A *ServerSocket* não executa de fato o serviço, em vez disso, ela cria um objeto *Socket* representando o cliente, através do qual realiza-se a comunicação.

- *ChatHandler:* a *thread ChatHandler* é onde a comunicação realmente ocorre. *ChatHandler* cria um objeto *DataInputStream* (*in*) que recupera a entrada do cliente, usando o método *GetInputStream()* e um objeto *PrintStream* (*out*) que permite ao servidor escrever a saída para os clientes, usando o método *GetOutputStream().* Assim, a comunicação ocorre nas duas direções.

- *Portas e Serviços:* cada serviço está associado a uma porta. Uma porta é um endereço numérico por meio do qual são processadas as solicitações de serviços. Neste trabalho, está se utilizando a porta 23.

6.4 FUNCIONAMENTO DOS AGENTES DO TUTA

A implementação dos agentes foi feita utilizando-se a definição de eventos. Assim, baseado em (Azevedo, 1999), definiu-se que cada um dos agentes possui três eventos e ao ocorrer um deles, o agente realiza alguma(s) tarefa(s). Tais eventos estão representados no Quadro 6.1.

[27] A partir daqui o Agente de Comunicação poderá ser referenciado também como Agente Servidor e os demais agentes da arquitetura como Agentes Clientes.

Evento	Descrição	Tarefas que devem ser realizadas
AoInicializar	Ocorre quando o agente é inicializado	Tarefas do Agente de Comunicação: • Habilitar as conexões com os outros agentes Tarefas gerais de todos os agentes: • Estabelecer a conexão com o Agente de Comunicação Tarefas específicas de cada agente (Após este quadro, são explicitadas as tarefas que são específicas de cada agente)
AoReceberMensagem	Ocorre quando o agente recebe uma mensagem.	• Interpretar a mensagem recebida
AoFinalizar	Ocorre após a solicitação de término de execução do agente.	Tarefas do Agente de Comunicação: • Desabilitar as conexões com os outros agentes Tarefas dos outros agentes: • Terminar a conexão com o Agente de Comunicação

Quadro 6.1: Eventos x Tarefas dos agentes

A seguir, apresenta-se o funcionamento dos agentes do TUTA[28], através de uma pseudo-linguagem:

Agente Gerenciador de Curso

AoIniclializar:

Conectar com o Agente de Comunicação;
Enviar mensagem de apresentação;
Monitora a data atual, verificando se é dia de sessão (através do acesso ao banco de dados, contendo: dia, data, hora, identificador de sessão, de curso e descrição da sessão);
Se a data atual = dia de sessão
então enviar mensagem ao Agente de Comunicação;
fim-se

AoReceberMensagem:

Se a mensagem = "Fim
então enviar mensagem = "Fim de Sessão" ao Agente de Comunicação;
Senão
Aguardar próxima mensagem
fim-se

AoFinalizar:

[28] O funcionamento descrito eqüivale a versão implementada (versão 1.0).

Desconectar-se do ao Agente de Comunicação.

Agente Executor de Sessão

AoIniclializar:

Conectar com o Agente de Comunicação;
Enviar mensagem de apresentação;
Aguarda mensagem = "Inicio de Sessão" + parâmetros (como hora da sessão);

AoReceberMensagem:

Se a mensagem = "Início de Sessão"
então enviar mensagem = "Iniciar as Interfaces" ao Agente de Comunicação;
executar estratégia(s) didática(s)
Senão
Aguardar próxima mensagem
fim-se

AoFinalizar:

Desconectar-se do ao Agente de Comunicação.

Agente Gerenciador de Interações

AoIniclializar:

Conectar com o Agente de Comunicação;
Enviar mensagem de apresentação;
Aguarda mensagem = "Inicio de Sessão" ;

AoReceberMensagem:

Se a mensagem é do tipo "Validar Senha"
então enviar mensagem do tipo "Validar Senha" ao Agente de Comunicação;
fim-se
Se a mensagem é do tipo "Resposta de Validação"
então enviar mensagem do tipo Resposta de Validação ao Agente de Comunicação;
fim-se
Se a mensagem é do tipo "Exibir Objetos Didáticos"
então enviar mensagem do tipo Exibir Objetos Didáticos ao Agente de Comunicação;
fim-se

Se a mensagem é do tipo "Iniciar Bate-papo"
então enviar mensagem do tipo Iniciar Bate Papo ao Agente de Comunicação;
fim-se
Se a mensagem é do tipo "Encerrar Bate-papo"
então enviar mensagem do tipo Encerrar Bate Papo ao Agente de Comunicação;
fim-se
Se a mensagem é do tipo "Iniciar as Interfaces"
então enviar mensagem do tipo Iniciar as Interfaces ao Agente de Comunicação;
Senão
Aguardar próxima mensagem
fim-se

AoFinalizar:
Desconectar-se do ao Agente de Comunicação.

Agente de Interface Aluno ou Professor

AoIniclializar:
Conectar com o Agente de Comunicação;
Enviar mensagem de apresentação;

AoReceberMensagem:
Se a mensagem é do tipo "Resposta de Validação" para este usuário
então exibir resposta da validação;
fim-se
Se a mensagem é do tipo "Iniciar Bate papo"
então habilitar Bate-papo;
fim-se
Se a mensagem é do tipo "Encerrar Bate papo"
então desabilitar Bate-papo;
fim-se
Se a mensagem é do tipo "Interface" + "Exibir"
então exibir objetos didáticos;
Senão
Aguardar próxima mensagem
fim-se

AoFinalizar:

Desconectar-se do ao Agente de Comunicação.

Agente Compositor de Grupo

AoIniclializar:

Conectar com o Agente de Comunicação;
Enviar mensagem de apresentação;

AoReceberMensagem:

Se a mensagem é do tipo "Iniciar as Interfaces" (indica que a sessão já começou)
então enviar mensagem "login desabilitado";
fim-se
Se a mensagem é do tipo "Compositor"
então acessar o banco de dados e validar os dados;
enviar mensagem do tipo "Resposta de Validação", contendo a validação do usuário ao Agente de Comunicação;
Senão
Aguardar próxima mensagem
fim-se

AoFinalizar:

Desconectar-se do ao Agente de Comunicação.

Agente Servidor de Entidades Didáticas

AoIniclializar:

Conectar com o Agente de Comunicação;
Enviar mensagem de apresentação;

AoReceberMensagem:

Se a mensagem é do tipo "Pedido de entidades didáticas"
então acessar o banco de dados e recuperar as entidades didáticas;
enviar mensagem do tipo "Pedido de entidades didáticas", contendo as entidades didáticas solicitadas;
Senão
Aguardar próxima mensagem
fim-se

AoFinalizar:

Desconectar-se do ao Agente de Comunicação.

6.5 IMPLEMENTAÇÃO DE APLICAÇÕES NO TUTA

O sistema TUTA, permite a especificação de diferentes aplicações por um professor. Assim, nas próximas seções, pretende-se apresentar como isto é possível.

6.5.1 FUNCIONAMENTO DE UMA SESSÃO DE TREINAMENTO NO TUTA

No Capítulo 4, foram especificadas as funcionalidades gerais do TUTA. Entretanto, o funcionamento foi descrito até o momento da execução de uma sessão de treinamento, conforme estratégia didática especificada por um professor. Nesta seção, objetiva-se mostrar, como, de fato, acontecem as sessões de treinamento no TUTA.

6.5.1.1 Táticas de Ensino do TUTA

A seguir, serão mostradas os tipos de táticas[29] permitidas pelo TUTA, com alguns exemplos em pseudocódigo.

- Táticas de *reutilização*: esse tipo de ação indica que será apresentada alguma entidade básica de ensino para o grupo. Exemplos: *Mostrar exemplo (1) de classes para o grupo 1*, *Mostrar definição (1) de classe para o grupo 1*, entre outros.

- Táticas de *debates síncronos*: esse tipo de ação é uma espécie de *"chat"* ou *"bate-papo"*, onde os alunos do grupo podem interagir com o professor ou com outros alunos.

- Táticas de *desvios condicionais*: esse tipo de ação consiste em uma condição que será verificada pelo sistema. Um exemplo disso poderia ser o seguinte:

    Mostrar exemplo (1) de classe para o grupo 1

    Mostrar questionário (1) de classe para o grupo 1

    ***Se** média questionário (1) grupo < 5 então*

        Mostrar exemplo (2) para o grupo 1

        Mostrar questionário (2) para o grupo 1

    Fim-se

[29] As táticas que compõem uma estratégia estão codificadas em um formato padrão e definidas no banco de dados do TUTA. O Agente Executor de Estratégias é responsável por decodificá-las. Mais detalhes sobre definições de estratégias didáticas são encontrados no Apêndice D.

- Táticas de *desvios incondicionais*: esse tipo de ação consiste em um desvio para qualquer ponto da estratégia . Um exemplo disso, está representado abaixo:

  (linha 1) Mostrar exemplo (1) de classe para o grupo 1

  (linha 2) Mostrar questionário (1) de classe para o grupo 1

  (linha 3) Se nota_questionário (1) grupo 1 < 7 então

  (linha 4) ***salta*** *para linha 1*

  (linha 5) Fim-se

- Táticas de *mudanças de estratégia:* esse tipo de ação consiste em uma mudança de estratégia (baseado no enfoque *multi-estratégias*), considerando o comportamento do grupo de alunos. Um exemplo desse tipo de ação está exemplificado no pseudocódigo abaixo:

  Mostrar exemplo (1) de classe para o grupo 1

  Mostrar questionário (1) de classe para o grupo 1

  Se média questionário (1) grupo 1 < 7 então

  Mudar para Estratégia Alternativa

  Fim-se

  Definição da Estratégia Alternativa:

  Debate síncrono (Professor, Alunos do Grupo 1)

- Táticas de *Estabelecimento de tempos*: após a execução de cada tática, é permitido ao professor estabelecer um tempo para que seja realizada alguma atividade. Um exemplo desse tipo de ação está exemplificado no pseudocódigo abaixo:

  Mostrar exemplo (1) de classe para o grupo 1

  Mostrar questionário (1) de classe para o grupo 1

  Tempo = 5 minutos;

6.5.2 FUNCIONAMENTO DE UMA SESSÃO DE TREINAMENTO UTILIZANDO PADRÕES PEDAGÓGICOS

A noção formal de Padrões (do inglês: *Patterns*), originou-se com os padrões arquitetônicos de Christopher Alexander (Alexander *et al*., 1977). A partir de então, as comunidades das mais diversas áreas passaram a utilizar esta facilidade. Um padrão descreve um problema que ocorre freqüentemente e então descreve o cerne da solução ao problema de forma a poder usar a solução milhões de vezes em situações semelhantes. Desta forma, surgiram vários padrões nas mais diversas áreas e inclusive no campo pedagógico, ou seja, formas de ensinar. Assim, neste trabalho foram considerados os padrões pedagógicos relacionados à Orientação a Objetos[30].

Nesta seção, será utilizado um padrão pedagógico, a fim de explicar o funcionamento do TUTA: o padrão pedagógico ***"Laboratório-Discussão-Conferência-Laboratório"*** *(LDLL) (do inglês: Lab-Discussion-Lecture-Lab),* cuja principal utilização é apresentar uma ferramenta de *software* e/ou conceitos de uma linguagem de programação. Sua estrutura consiste no seguinte:

- **Objetivos***:* Introduzir conceitos de *software*.
- **Motivação:** Parte-se do princípio de que os alunos muitas vezes têm dificuldade em entender conceitos de *software* que são introduzidos na sala-de-aula. Isso acontece porque a técnica pedagógica *Conferência-Laboratório* utiliza "o que acontece quando você faz isto", e esta abordagem torna-se muito abstrata para prover fundamentos para o aprendizado de *software.* O método pedagógico *LDLL* oferece uma abordagem alternativa: inicia-se com um laboratório introdutório passo a passo, depois segue com uma discussão e uma explanação do que os alunos fizeram e então reforça-se os conceitos com um laboratório mais avançado.
- **Aplicabilidade:** Este padrão é utilizado para introduzir uma ferramenta de *software* e/ou conceitos em uma linguagem de programação.
- **Estrutura:**
  - **Laboratório *(Lab*)**: Alunos completam um laboratório passo a passo, preparado pelo instrutor para introduzir conceitos de *software*. Este laboratório deve ter instruções escritas detalhadas e incluir questões que encorajem o aluno a registrar e analisar o que eles veêm na tela e como eles devem proceder através do exercício.

[30] A fonte utilizada foi http://www-lifia.info.unlp.edu.ar/ppp. A maioria dos padrões pedagógicos relacionados no *site*, foram colecionados de três *workshops*: ECOOP´96, TOOLS USA´96 e OOPSLA´96.

- **Discussão (*Discussion*):** Depois do laboratório ser realizado por cada aluno, o instrutor conduz a uma discussão com os alunos a respeito do seguinte: os conceitos que não foram familiares ao aluno e como ele completaram o exercício e quais problemas que o aluno encontrou (a fim de controlar a qualidade dos futuros exercícios).

- **Conferências (*Lecture*):** O instrutor conferencia sobre novos conceitos que foram introduzidos no laboratório, fazendo contínuas referências às experiências que os alunos realizaram enquanto estavam fazendo o laboratório.

- **Laboratório (*Lab*):** Alunos realizam um exercício de laboratório mais complexo que reforça a compreensão de novos conceitos.

- **Algumas considerações:**

  - Os exercícios de laboratório podem ser curtos (cobrindo um ou dois conceitos) ou longos (cobrindo muitos conceitos). Exercícios de laboratório curtos podem ser realizados durante o período destinado ao laboratório; para exercícios longos, é recomendado que eles sejam realizados fora das horas regulares.

  - As questões que aparecem durante o primeiro laboratório devem permitir que os alunos analisem o que eles realizam e o que lhes é mostrado. Estas questões servirão então para estimular e conduzir a sessão de discussão

  - As sessões de conferência requerem apenas um período curto de tempo, já que ocorrem após a fase de laboratório e discussões, diferentemente do que seria se as conferências ocorressem no formato *lecture-lab.*

- **Exemplos de aplicações do padrão:** Este *pattern* tem sido utilizado para ensinar: o ambiente *Smaltalk*, conceitos de programação *Smaltalk,* ou **C,** ou *Basic,* ou *Fortran*.

6.5.3 EXEMPLO DE UMA SESSÃO DE TREINAMENTO UTILIZANDO UM PADRÃO PEDAGÓGICO

Partindo do padrão pedagógico especificado na seção anterior, foi definida a *Estratégia-1* que implementa o padrão pedagógico *Laboratório-Discussão-Conferência-Laboratório.* A definição completa da estratégia encontra-se nos Apêndice D.

Em primeiro lugar, é necessário que os alunos e o professor identifiquem-se, para que o TUTA realize a validação dos dados de entrada (usuário e senha). A Figura 6.1 demonstra como é feita tal identificação.

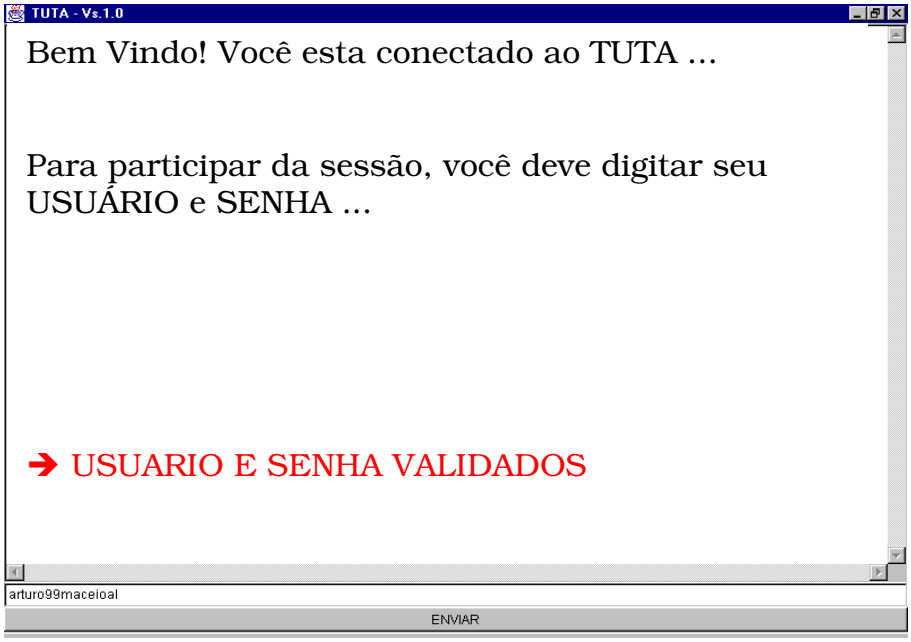


Figura 6.2: Identificação dos alunos ou do professor no TUTA

O primeiro passo da execução do padrão pedagógico está relacionado às atividades de *laboratório*. A figura 6.3 ilustra algumas atividades realizadas pelo professor nesse passo.

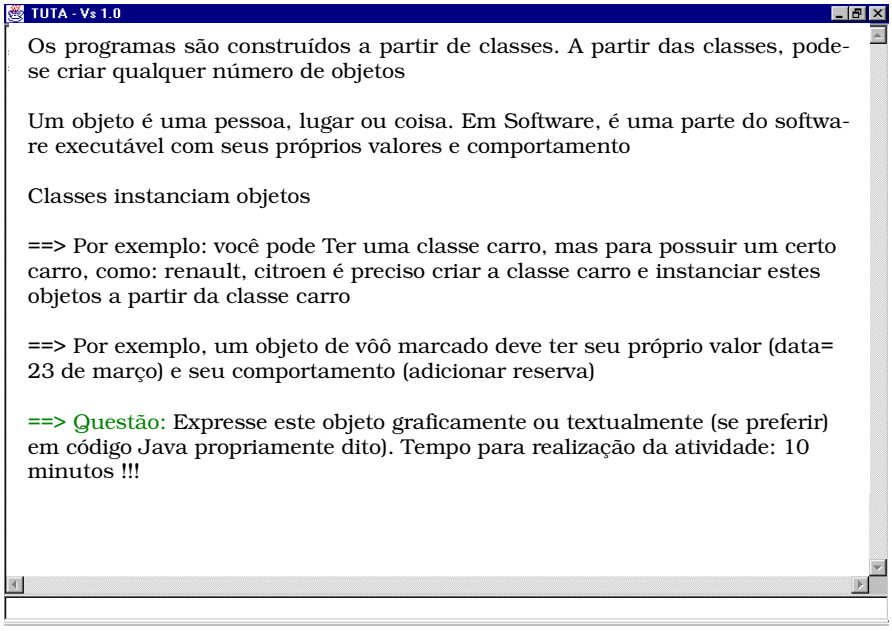


Figura 6.3: Execução das primeiras atividades de *laboratório* do padrão LDLL

O passo seguinte é composto pelas atividades de *discussão* entre o professor e os alunos. A figura 6.4 ilustra essas atividades.

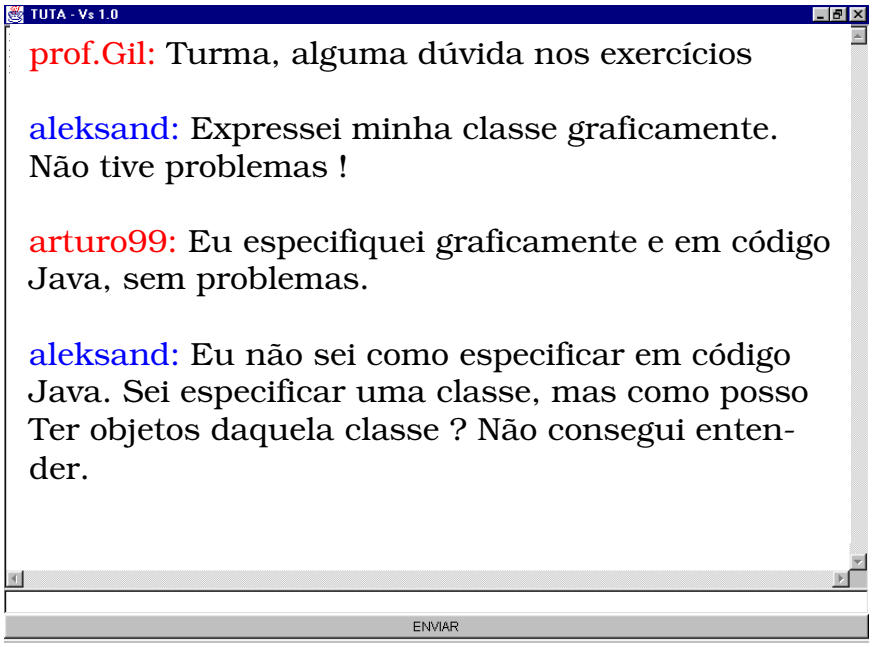


Figura 6.4: Execução da atividade *discussão* do padrão LDLL

Após esses dois passos, é realizado então uma nova etapa, mais formal em relação aos conceitos apresentados e discutidos. A Figura 6.5 ilustra as atividades de *conferência* realizadas pelo professor.

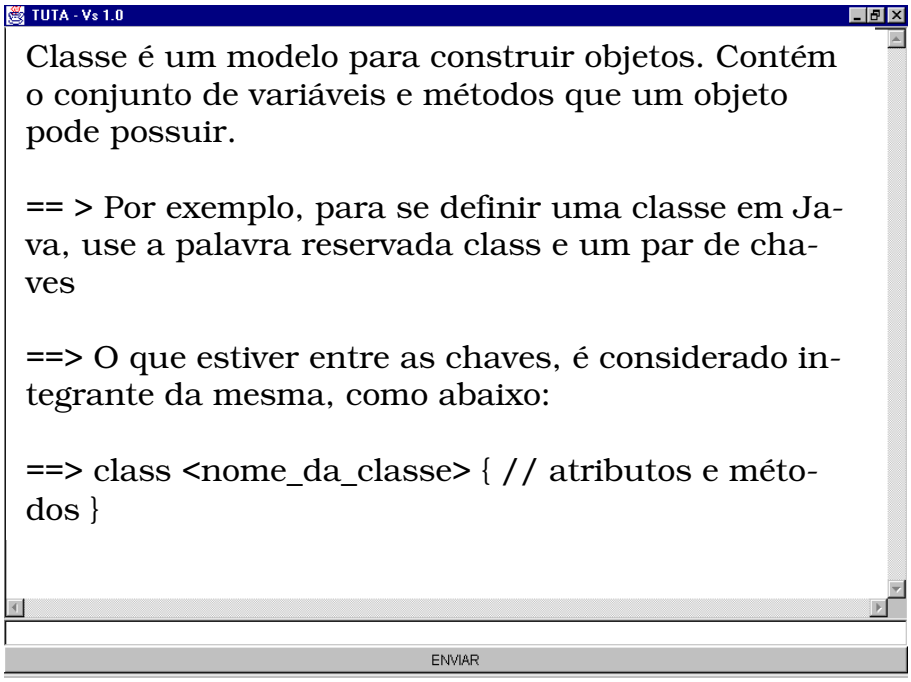


Figura 6.5: Execução da atividade *conferência* do padrão LDLL

Finalmente, realiza-se novas atividades de *laboratório*, com exercícios mais complexos. Note, que durante esta atividade, o *"chat"* está habilitado.

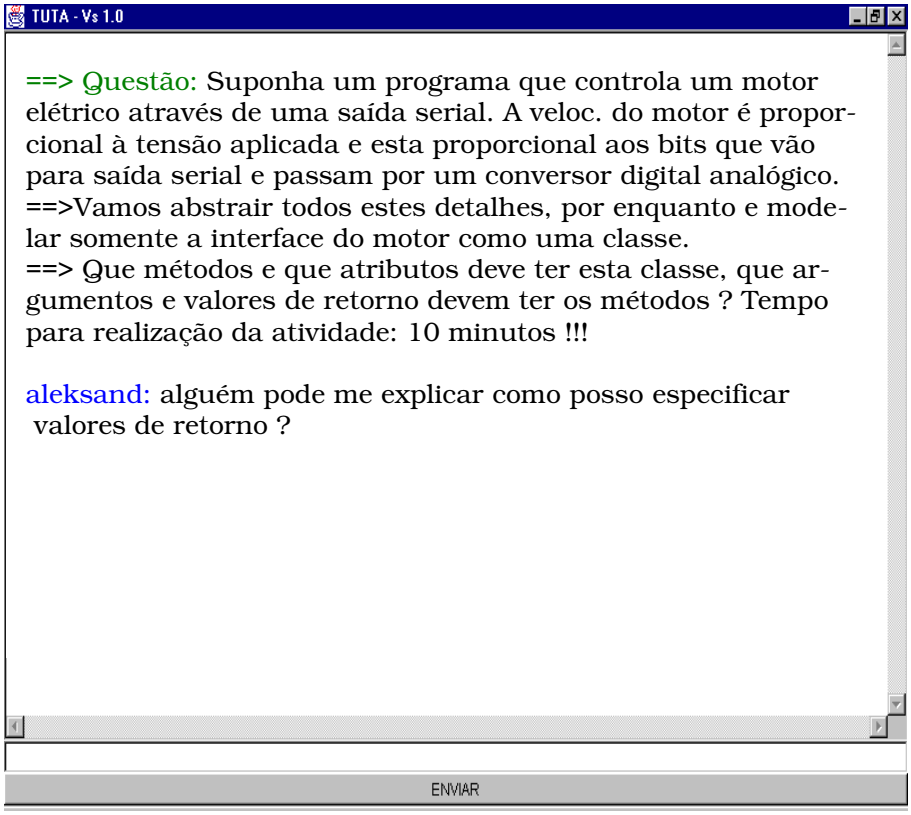


Figura 6.6: Execução da atividade *laboratório* do padrão LDLL

6.6 CONCLUSÕES

Neste Capítulo, apresentou-se os aspectos de implementação relacionados ao TUTA, bem como a demonstração do protótipo implementado, utilizando o Padrão Pedagógico LDLL. É importante ressaltar que o TUTA pode funcionar conforme diferentes estratégias didáticas definidas pelo professor, levando-se em conta apenas as Táticas de Ensino previamente implementadas no sistema, conforme visto na seção 6.5.1.1.

Capítulo 7
Conclusões

Neste capítulo, apresentam-se os resultados obtidos por este trabalho, bem como algumas perspectivas futuras relacionadas ao TUTA no contexto da ACVA.

7.1 RESULTADOS OBTIDOS

A área de informática na Educação é fascinante, entretanto uma área difícil por envolver diferentes competências. Assim, desenvolver um sistema desta natureza, com certeza não é uma tarefa fácil. Algumas contribuições que podem ser citadas após o término do trabalho, são:

- O sistema é capaz de permitir que um grupo de alunos geograficamente distantes acompanhe *on-line* suas sessões de ensino.
- Representa uma importante ferramenta de auxílio ao professor, permitindo a especificação de um curso, composto por sessões de ensino.
- Permite o acompanhamento dos alunos presente no curso.
- O TUTA permite que um professor especifique diferentes estratégias didáticas, sendo que para isso não é necessário alterar o código dos programas implementados no TUTA.
- O TUTA permite que o professor possa definir a troca de uma determinada estratégia para outra no meio de uma sessão.
- O uso de agentes no desenvolvimento deste *software,* permitiu um sistema, com uma camada a mais de abstração (os agentes), além de vantagens como: autonomia, reatividade e habilidade social. Adicionalmente, a utilização de uma metodologia apropriada, permitiu uma melhor compreensão do sistema.

7.2 PERSPECTIVAS FUTURAS

Baseado nos resultados obtidos com o TUTA, sugere-se alguns possíveis trabalhos que podem dar continuidade a esse trabalho:

- Desenvolvimento de uma interface gráfica, ou seja, um módulo de autoria para facilitar a especificação das estratégias didáticas por um professor.

- Desenvolvimento do ambiente TUTA, que deverá ser composto do módulo de execução e do módulo de autoria.

- Desenvolvimento de Interfaces gráficas para que o aluno e professor sintam-se mais motivados a participar de uma sessão.

- Permitir que o professor desenvolva estratégias didáticas que considerem a zona de comportamento individual do aluno, bem como a zona de comportamento do grupo.

- No contexto da ACVA, o TUTA representa apenas **um** controlador de grupo, interagindo com uma camada composta das entidades básicas necessárias à execução de uma sessão. Assim, a camada *SF_Grupo* completa poderá ser implementada (através de **vários** controladores de grupo).

- Ainda no contexto da ACVA, é possível implementar também o Controlador da Classe Virtual, permitindo assim a mobilidade dos alunos entre diferentes grupos.

- Incrementar mais funcionalidades aos agentes da arquitetura do TUTA.

Referências Bibliográficas

AÏMEUR, E., FRASSON, C., ALEXE, C. (1995) Towards New Learning Strategics in Intelligent Tutoring Systems. In: BRASILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 12., 1995, Campinas. *Proceedings*... Campinas: Springer-Verlag, 1995.

ALEXANDRE, C., ISHIKAWA, S., SILVERSTEIN, M. (1977) A Pattern Language-Towns-Building. Oxford University Press, 1977.

ALVARES, L.O., SICHMAN, J. S. (1997) Introdução aos Sistemas Multiagentes. In: MEDEIROS, C. M. B.(Ed.). XVI JAI – Jornada de Atualização em Informática, 1997, Brasília. *Anais*... Brasília: [s.n.], 1997.

AZEVEDO, B. F. T. (1999) *MutAntIS - Uma Arquitetura Multi-Agente para a Autoria de Tutores Inteligentes*. 1999. 115 p. Dissertação (Mestrado em Informática) – CT/UFES.

BARR, A., FEIGENBAUM, E. A. (1982) *The handbook of Artificial Inteligence*. Los Altos: Califórnia: Kaufmann, 1982.

BIGUS, J. P., BIGUS, J. (1998) Constructing Intelligent Agents With Java. A Programmer´s Guide to Smarter Applications. New York: Wiley Computer Publishing, 1998.

BOOCH, G. (1996) Object Oriented Analysis and Design. RedwwoodCity, CA: Benjamin/Cummings, 1996.

BROOKS, R. A. A. (1986) A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation, v. 2, n. 1, p. 14-23, 1986.

BROWN, J. S., BURTON, R. B. (1982) Pedagogical natural language and knowledge engineering techniques. In: SLEEMAN, D., BROWN, J. S. (Eds.). *Intelligent Tutoring Systems*. New York: Academic Press, 1982.

CARBONELL, J. R. (1970) AI in CAI: an artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, New York, v. 11, n. 2, p. 190-202, 1970.

CHEIKES, B. A. (1995) GIA: An Agent Based Architecture for Intelligent Tutoring Systems, In: CIKM WORKSHOP ON INTELLIGENT INFORMATION AGENTS, 1995, [S.l.]. *Proceedings*... [S.l.]: [s.n.], 1995.

CORRÊA, M. F. (1994) *A Arquitetura de Diálogos entre agentes Cognitivos Distribuídos*. 1994. Tese (Doutorado) – COPEE/UFRJ.

COSTA, R. M. E. M., WERNECK, V. B., MOREIRA, M., DIAS, R. (1995) *Treinar: Um Tutor Inteligente para Treinamento em Sistemas Baseados em Conhecimento*. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE´95, 1995, Florianópolis. *Anais*... Florianópolis: [s.n.], 1995.

COSTA, R. M. E. M., XEXÉO G. B. (1996) A Internet nas Escolas. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE´96, 7.,1996, Belo Horizonte. *Anais*... Belo Horizonte: DCC/UFMG, 1996.

COSTA, E. B. (1997) *Um Modelo de Ambiente Interativo de Aprendizagem Baseado numa Arquitetura Multiagente*. 1997. 143p. Tese (Doutorado em Processamento da Informação) – CPGEE/UFPB.

COSTA, R. M., ROCHA, A. R. C., SANTOS, N. (1997) Desenvolvimento de Sistemas Tutores Inteligentes: Questões e Perspectivas. In: WORKSHOP DE SISTEMAS DE TUTORIA INTELIGENTE APLICADOS À EDUCAÇÃO E TREINAMENTO – STIET´97, 1997, São José dos Campos. *Anais*... São José dos Campos: [s.n.], 1997.

CRISTÓVÃO, H. M. (1997) *Ambientes Computacionais para Apoio à Aprendizagem: Um Experimento com Frações*. 1999. 252 p. Dissertação (Mestrado em Informática) – CT/UFES.

D´AMICO, O. C. (1995) *Modelo de usuário para Sistemas Tutores Inteligentes*. 1995. Dissertação (Mestrado em Ciência da Computação) – CPGCC/UFRGS.

D´AMICO, C. B. (1999) *Aprendizagem estática e dinâmica em sistemas multiagentes de ensino-aprendizagem*. 1999. Tese (Doutorado) – CPGCC/UFRGS.

D'ABREU, J.V.V. (1991) Construção e Interfaceamento de Dispositivos com Computadores para Fins Educacionais, In: SEMINÁRIO NACIONAL DE INFORMÁTICA EDUCATIVA, 1991, Maceió – AL. *Anais*... Maceió: [s.n.], 1991.

DEMAZEAU, Y., MULLER, J. P. (1990) Descentralized Artificial Intelligence. DEMAZEAU, Y., MULLER, J. P. (Eds.). *Descentralized Artificial Intelligence*. Amsterdam: Elsevier. Science Publishers, 1990.

DIRENE, A. I. (1993) *Methodology and tools for designing concept tutoring systems*. 1993. Tese (Doutorado) – Scool of Cognitive and Computing Sciences – University of Sussex.

FININ, T., MCKAY, D., FRITZSON, R. An Overview of KQML: A Knowledge Query and Manipulation Language. [online]. Disponível: http://www.cs.umbc.edu/~finin/papers/ [capturado em 20 fev.2000]

FLANAGAN, D. (1996) Java in a Nutshell. Sebastopol, CA: O´Reilly & Associates, 1996.

FOWLER, M., SCOTT, K. (1997) UML Distilled : Applying the Standard Object Modeling Language. EUA: Addison-Wesley, 1997.

FRANKLIN, S., GRAESSER, A. (1996) Is It an Agent or just a Program ?: A Taxonomy for Autonomous Agent. In: THIRD INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITETURES, AND LANGUAGES, 1996. *Proceedings*... [S.l.]: Springer-Verlag, 1996.

GAVA. T. B. S. (1999) *Monitoria On-Line: Um Ambiente na Internet para apoio ao atendimento Extra Classe*. 1999. 222 p. Dissertação (Mestrado em Informática) – CT/UFES.

GIRAFFA, L. M. M. (1995) *Teorias de Ensino-aprendizagem e sua aplicação em Sistemas Tutores Inteligentes*. 1995. 117 p. Trabalho Individual (487) – Programa de Pós-Graduação em Ciência da Computação, UFRGS.

GIRAFFA. L. M. M. (1997) *Seleção e adoção de Estratégias em Sistemas Tutores Inteligentes*. 1997. 131 p. Exame de Qualificação – Programa de Pós-Graduação em Ciência da Computação, UFRGS.

GIRAFFA, L. M. M., VICCARI, R. M. (1998) Estratégias de Ensino em Sistemas Tutores Inteligentes modelados através da tecnologia de agentes. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE´98, 19.,1998, Fortaleza. *Anais*... Fortaleza: UFCE, 1998.

GIRAFFA, L. M. M. (1999) *Uma arquitetura de tutor utilizando estados mentais*. 1999. 177p. Tese (Doutorado em Ciência da Computação) – Programa de Pós-Graduação em Ciência da Computação, UFRGS.

GOLDSTEIN, I. (1982) *The Genetic graph*: *a representation for the evolution of procedural knowledge*, In: SLEEMAN, D., BROWN, J. S. (Eds.). *Intelligent Tutoring Systems*. Great Britain: Academic Press, 1982.

HERNÁNDEZ-DOMÍNGUEZ. A. (1995) *Systemes "Tuteur intelligent" et "Collecticiel": L'A.R.E.S.F.E.D. Architecture pour la reutilisation et l'exploitation des services de formation dans le contexte de l'Education à Distancia*. 1995. Tese (Doutorado) - Doctorat de l'Université Paul Sabatier-Toulouse, França.

HERNÁNDEZ-DOMINGUEZ, A. (1996) Specification of an Adapted Training Service in a Virtual Class Architecture. In: SEMANA DE INFORMÁTICA DA UFBA, 1996, Salvador. *Anais*... Salvador: [s.n.], 1996.

HERNÁNDEZ-DOMÍNGUEZ. A. (1997) Specification of an adaptable virtual class. In: ED-MEDIA/ED-TELECOM 97: WORLD CONFERENCE ON EDUCATIONAL MULTIMEDIA AND HYPERMEDIA AND ON EDUCATIONAL TELECOMMUNICATIONS, 1997, Calgary-Alberta, Canada. *Proceedings*... [S.l.]: [s.n.], 1997.

HERNÁNDEZ-DOMÍNGUEZ. A. (1998) Object Modelling of an Adaptable Virtual Class. using OMT methdology. In: ED-MEDIA 98: WORLD CONFERENCE ON EDUCATIONAL MULTIMEDIA AND HYPERMEDIA AND ON EDUCATIONAL TELECOMMUNICATIONS, 1998, Alemanha. *Proceedings*... [S.l.]: [s.n.], 1998.

JACOBSON, I., CHRISTERSON, M., JONSSON, P., OVERGAARD, G. (1992) Object-Oriented Software Engineering : A Use Case Drivem Approach. Reading MA: Addison-Wesley, 1992.

KLING, R. (1991). Cooperation, coordination and control in computer-supported work. Communications of the ACM, v. 34, n. 11, p. 83-88, 1991.

KNAKIP, M., JOHNSON, J. (1998) Developing Intelligent Agents for Distibuted Systems. USA: McGraw Hill, 1998.

LARMAN, C. (1998) *Applying UML and Patterns* : An Introduction to Object-Oriented Analysis and Design. New Jersey: Prentice Hall, 1998. 507p.

LEITE, A., FERNANDES, C., OMAR, N. (1996) Avaliação de Sistemas de Tutoria Inteligente. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE´96, 7.,1996, Belo Horizonte. *Anais*... Belo Horizonte: DCC/UFMG, 1996.

MAJOR, N., REICHGELT, H. (1992) COCA: A Shell for Inteligent Tutoring Systems. In: FRASSON, C., GAUTHIER, G., MC CALLA G. I. (Eds.). International Conference on Intelligent Tutoring Systems – ITS'92, 1992, Montreal. *Proceedings*... [S.l.]: Springer-Verlag, 1996.

MAJOR, N. (1995) Modelling Teaching Strategies. *Jl. of Artificial Intelligence in Education*, v. 6, n. 2/3, p. 117-152, 1995.

MARIETTO, M. G., OMAR, N., FERNANDES, C. T. (1997) Tendências nas Áreas de Sistemas de Tutoria Inteligente e Modelagem do Aprendiz. In: WORKSHOP DE SISTEMAS DE TUTORIA INTELIGENTE APLICADOS À EDUCAÇÃO E TREINAMENTO – STIET´97, 1997, São José dos Campos. *Anais*... São José dos Campos: [s.n.], 1997.

MURRAY, T. (1996) Having it all, Maybe: Design Tradeoffs in ITS Authoring Tools. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TUTORING SYSTEMS-ITS'96, 3.,1996, Montreal. *Proceedings*... Berlin: Springer-Verlag, 1996.

NWANA, H.S., NDUMU, D.T. (1998) A Brief Introduction to Software Agent Technology. In: JENNINGS, N.R., WOOLDRIDGE, M.J. (Eds.). *Agent Technology: Foundations, Applications and Markets*. Berlim: Springer, 1998.

O'SHEA, T., BORNAT, R., DU BOULAY, B., EISENSTADT, M., PAGE, I. *Tools for Creating Intelligent Computer Tutors*. In: ELITHORN, A., BARNEJI, R. (Eds.). *Human and Artificial Intelligence*. North-Holland, London: [s.n.], 1984.

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

PAPERT, S. (1985) *LOGO*: *Computadores e Educação*. São Paulo: Ed. Brasiliense, 1985.

PAPERT, S. (1987) Microworlds: Transforming education. In: LAWER, R., W., YAZDANI, M. (Eds.). Artificial Intelligence and Education – Learning Enviroments and Tutoring Systems. [S.l]: [s.n], 1987.

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

RUSSELL, S., NORVIG, P. (1995) *Artificial Intelligence* : A Modern Approach. New Jersey: Prentice-Hall, 1995.

SANTOS, N. (1999) Estado da Arte em Espaços Virtuais de Ensino e Aprendizagem. Revista Brasileira de Informática na Educação, Florianópolis, n. 1, 1999.

SCHREIBER, G., WIELINGA, B., BREUKER, J., (1993) KADS : A Principled Approach to Knowledge-Based System Development. [S.l.]: Academic Press, 1993.

SHARPLES, M. (1989*)* The Radiology Tutor: Computer-Based Teaching of Visual Categorization. In: BIERMAN, D., BREUKER, J., SANDBERG, J. (Eds.). International Conference on AI Education, 1989. *Artificial Intelligence and Education*. [S.l.]: [s.n.], 1989.

SHNEIDERMANN, B. (1992) *Designing the User Interface* : Strategies for Effective Human-Computer Interaction. 2. ed. [S.l.]: Addison-Wesley Publishing Company, 1992.

SICHMAN, J. S., DEMAZEAU, Y., BOISSIER, O. (1992) When Can Knowledge-Based Systems Be Called Agents ?. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 1992, Rio de Janeiro. *Anais*... Rio de Janeiro: [s.n.], 1992.

SILVA, A., HERNÁNDEZ-DOMÍNGUEZ, A. (1999) Uma modelagem baseada em agentes de um Tutor no contexto de uma Classe Virtual Adaptativa. In SBIE´99 (Workshop de Ambientes de Aprendizagem baseado em Agentes, 1999, Curitiba. Anais... Curitiba: UFPR, 1999.

SILVEIRA, R. A. (1992) *Inteligência Artificial em Educação: um modelo de sistema tutorial inteligente para microcomputadores*. 1992. FACED/PUCRS.

SKINNER, B. F. (1958) *Teaching Machines*. Science, v. 128. 1958.

SLEEMAN, D., BROWN, J. S. (1982) *Intelligent Tutoring Systems*. New York: Academic Press, 1982.

SOLOWAY, E. M., JOHNSON, W. L. (1984) Remembrance of blunders past: a retrospective on the development of PROUST. In: COGNITIVE SCIENCE SOCIETY CONFERENCE, 6., 1984. *Proceedings…* New Jersey: Lawrence Erlbaum, 1984.

SOMMERVILLE, I., RODDEN, T. Environments for Cooperative Systems Development. *IEEE Computer*, [S.l.], 1993.

STEVENS, A. L., COLLINS, A. (1977) The goal structure of a Socratic tutor. In: NATIONAL ACM CONFERENCE, 1., 1977. *Proceedings…* New York: ACM, 1977.

TANEMBAUN, A. S (1995) Sistemas Operacionais Modernos. Rio de Janeiro: Prentice Hall do Brasil, 1995.

TAYLOR, R. P. (1980) *The computer in school* : tutor, tools, tutee. New York: Teachers College Press, 1980.

TEDESCO, P. C. A. R. (1997) *SEI – Sistema de Ensino Inteligente*. 1997. Dissertação (Mestrado em Ciência da Computação) – DI/UFPE.

VICCARI, R. (1990) *Um Tutor Inteligente para a programação em Lógica* : Idealização, Projeto e Desenvolvimento. 1990. Tese (Doutorado) – Universidade de Coimbra.

VICCARI. R. M., GIRAFFA L. M. M. (1996) Sistemas Tutores Inteligentes: abordagem tradicional x abordagem de agentes. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL – SBIA´96, 1996, Curitiba. *Anais*... Curitiba: [s.n.], 1996.

WENGER, E. (1987) *Artificial Intelligence and Tutoring Systems* : Computational and Cognitive Approaches to the Communication of Knowledge. California, USA: Morgan Kaufmann Publishers, Inc., 1987.

WOOLDRIDGE, M., JENNINGS, N. R. (1995a) *Intelligent Agents* : Theory and Practice. *The Knowledge* Engineering Review, v.10, n.2 , 1995.

WOOLDRIDGE, M., JENNINGS, N. R. (1995b) *Agent Theories, Architetures e Languages* : a Survey Intelligent Agents. Berlin: Springer - Verlag, 1995.

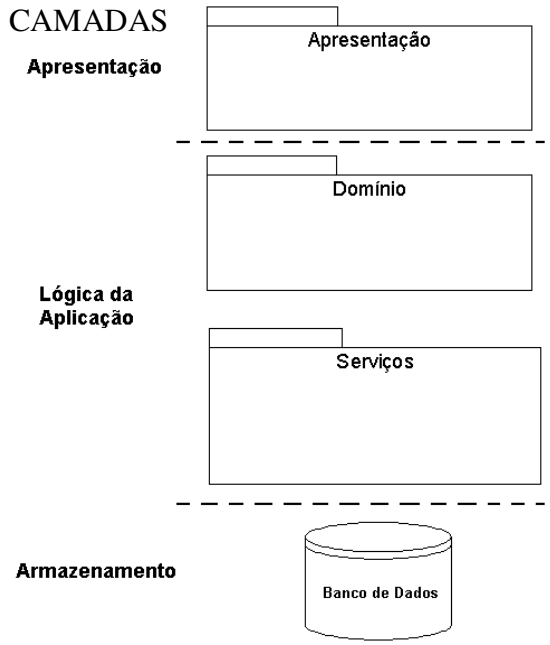
WOOLDRIDGE, M., JENNINGS, N. R. (1998) Applications of Intelligent Agents. In: JENNINGS, N.R., WOOLDRIDGE, M. (Eds.) *Agent Technology: foundations, applications, and markets*. New York: Springer, 1998.

WOOLDRIDGE, M., JENNINGS N. R., KINNY, D. (1999) A Methodology for Agent-Oriented Analysis and Design. In: THIRD ANNUAL CONFERENCE ON ON AUTONOMOUS AGENTS, 1999, Seattle – USA. *Proceedings*... [S.l.]: [s.n.], 1999.

YAZDANI, M. (1987) Intelligent Tutoring Systems: an overview. *Artificial Intelligence and Education*. [S.l.]: Ablex ,v.1, p. 183-201, 1987.

Apêndice A

UML



Este apêndice descreve a notação de alguns diagramas da linguagem de modelagem UML (do inglês: Unified Modeling Language), utilizados neste trabalho.

A.1 INTRODUÇÃO

"UML" é uma linguagem dirigida à modelagem de sistemas. Para isso, possui vários diagramas para realiza tal tarefa. As seções A.2 à A.6 descrevem os diagramas utilizados neste trabalho, com suas respectivas notações.

A.2 DIAGRAMA DE CASOS DE USO (DO INGLÊS: *USE CASE*)

Casos de uso representam um documento narrativo que descreve a seqüência de eventos de um ator (agente externo) que usa um sistema para completar um processo (Jacobson, 1992). Ou seja, são estórias ou casos de utilização de um sistema. O diagrama de casos de uso, por sua vez, ilustra um conjunto de casos de uso para um sistema, os atores e relação entre os atores e os casos de uso. A Figura A.1 ilustra a notação utilizada para o diagrama de casos de uso.

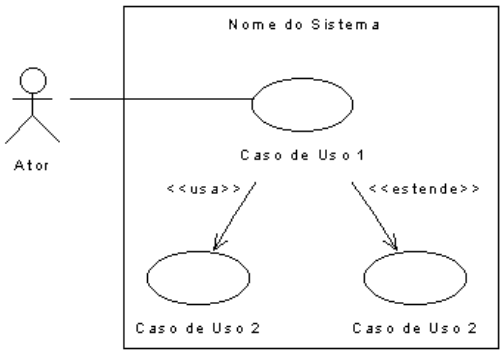


Figura A.1: Diagrama de Casos de Uso

As seções A.2.1 à A.2.4 descrevem os elementos básicos nos diagramas de caso de uso, que são: Atores, Casos de Uso, Fronteira e Relacionamentos Múltiplos.

A.2.1 ATOR

Um Ator é uma entidade externa ao sistema, que participa da estória do caso de uso. Um ator, tipicamente, estimula o sistema com eventos de entrada ou recebe algo do mesmo. Os atores são representados pelo papel que desempenham no caso de uso, tais como: Aluno, Professor, entre outros. Os Atores estimulam ou são estimulados por casos de uso e isso é indicado em UML através de linhas de comunicação representadas por flechas. A notação em UML para representar um ator de um caso de uso está ilustrada na Figura A.2.

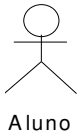


Figura A.2: Ator

A.2.2 CASOS DE USO

Cada caso de uso descreve um processo, como por exemplo, uma seqüência de eventos, ações e transações requeridas para produzir ou completar algo de valor para uma organização ou ator. A Figura A.3 ilustra a notação em UML para caso de uso.

Figura A.3: Caso de uso

A.2.3 FRONTEIRA

Um caso de uso descreve a interação com um "sistema". As fronteiras típicas de sistemas incluem:

- A fronteira hardware/software de um dispositivo ou sistema de computador.
- Departamento de um organização.
- Um organização inteira.

Definir a fronteira de um caso de uso é importante para identificar o que é externo *versus* o que é interno e quais são as responsabilidades do sistema. O ambiente externo é representado somente por atores. A Figura A.4 ilustra a notação de uma fronteira indicada pela seta.

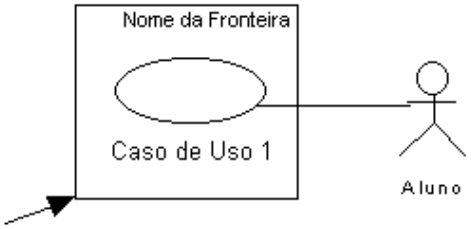


Figura A.4: Fronteira do Sistema

A.2.4 RELACIONAMENTO MÚLTIPLOS

Dependendo do que está sendo modelado, poderá surgir a necessidade de relacionar casos de uso. O objetivo é criar um diagrama de casos de uso atualizado que mostra esses casos de uso adicionais e seus relacionamentos. UML tem uma notação especial para ilustrar isso. Basicamente, existem dois tipos de relacionamento: o *usa* representado por (<<*uses*>>) e o *estende* representado por (<<*extends*>>).

O *usa* é utilizado quando um caso de uso inicia ou inclui o comportamento de um outro. Dessa forma, diz-se que ele usa o segundo caso de uso e que ambos têm um relacionamento do tipo *usa.* O relacionamento estende, por sua vez, é usado quando informalmente, uma segunda estória de caso de uso *segue* a estória de outro caso de uso. Por exemplo, muitos casos de uso "estendem" um caso de uso "*Login*".

A.3 DIAGRAMA DE CASOS DE USO REAIS

Um caso de uso real descreve o projeto real ou atual do caso de uso em termos da tecnologia concreta de entrada e saída. Nesse tipo de diagrama, o caso de uso incluirá os diagramas das janelas envolvidas, bem como a discussão da interação de nível baixo, representada no Quadro A.1, por exemplo. Eles são úteis se os desenvolvedores ou o cliente necessitarem de um nível extremamente detalhado para a descrição da interface antes da implementação. A Figura A.5 mostra a tela principal do caso de uso "*Comprar Itens*" (Larman, 1998).

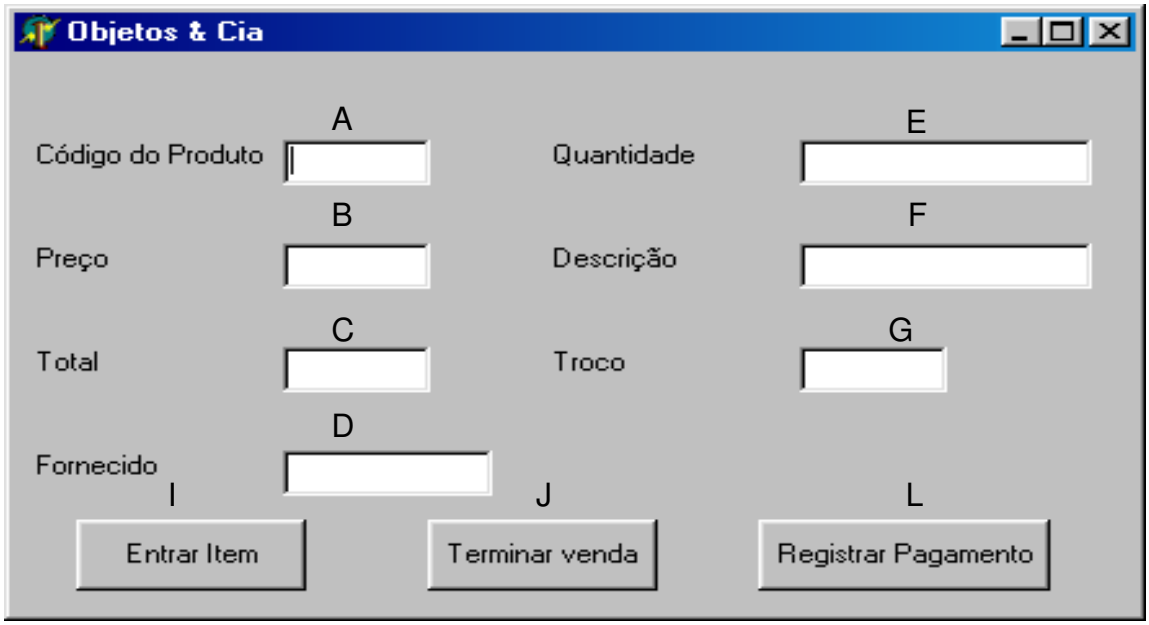


Figura A.5: Caso de uso *"Comprar Itens"*

Seqüência Típica dos Eventos	
Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando um cliente chega a um ponto-de-vendas equipado com um POST, trazendo vários itens que deseja comprar.	
2. Para cada item, o caixa registra o código Universal de produto (UPC) em A da Janela1. Se houver mais de um exemplar do item, a quantidade pode ser entrada opcionalmente em E. Ele pressiona H após cada entrada de item.	3. Acrescenta informações sobre o item à transação de vendas em andamento. A descrição e o preço do item corrente são apresentados em B e F da Janela1.
4. No término da entrada de itens, o caixa indica para o POST que a entrada de itens está completa, apertando o botão Terminar venda	5.Calcula e exibe o total da venda em C.
	6. ...

Quadro A.1: Seqüência típica de eventos do caso de uso Compras Itens

A.4 ARQUITETURA DO SISTEMA

Para ilustrar a arquitetura do sistema com a notação UML, deve-se usar pacotes. Um pacote é um conjunto de elementos do modelo de qualquer tipo, tal como classes, casos de uso, diagramas de colaboração ou mesmo outros pacotes. Dessa forma, um diagrama de pacotes da UML para representar agrupamentos lógicos é ilustrado na Figura A.6.

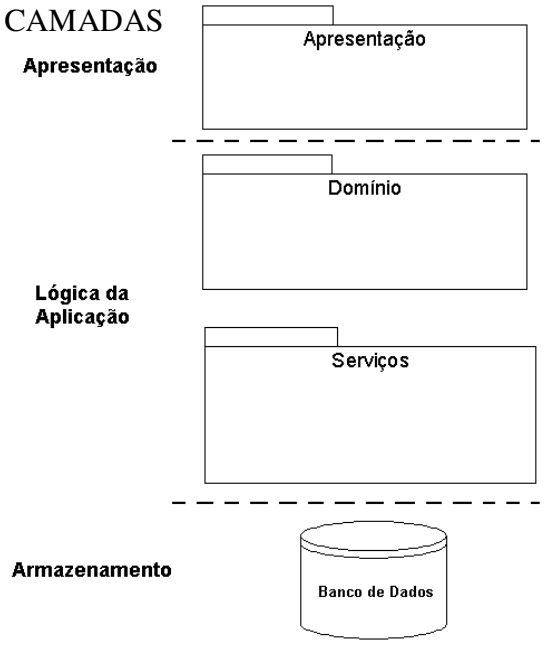


Figura A.6: Arquitetura do sistema, usando diagramas de pacotes

Uma descrição clássica de cada camada é a seguinte (Larman, 1998):

- **Apresentação**: Representada pelas interfaces do usuário.
- **Lógica da Aplicação**: Representada pelas tarefas ou regras que governam o processo. Essa camada pode ser decomposta em:
  - Objetos do Domínio: Representada pelas classes que representam conceitos do domínio.
  - Serviços: Representada pelos objetos de serviço para funções, tais como interação com o banco de dados, geração de relatórios, comunicações, segurança, entre outros.
- **Armazenamento:** Representada pelos mecanismos de armazenamento persistente.

A.5 DIAGRAMA DE CLASSES

O diagrama de classes ilustra as especificações para as classes de *software* e de interfaces de uma aplicação. As informações típicas são as seguintes:

- Classes, associações e atributos.
- Métodos.
- Informações do tipo do atributo.
- Navegabilidade.

O diagrama de classes da Figura A.7 (Larman, 1998) ilustra uma definição de *software* parcial para as classes *POST* e *Venda*. Além das associações básicas e dos atributos, o diagrama é estendido para mostrar, por exemplo, os métodos de cada classe, as informações sobre os tipos dos atributos, a visibilidade de atributo e a navegação entre os objetos.

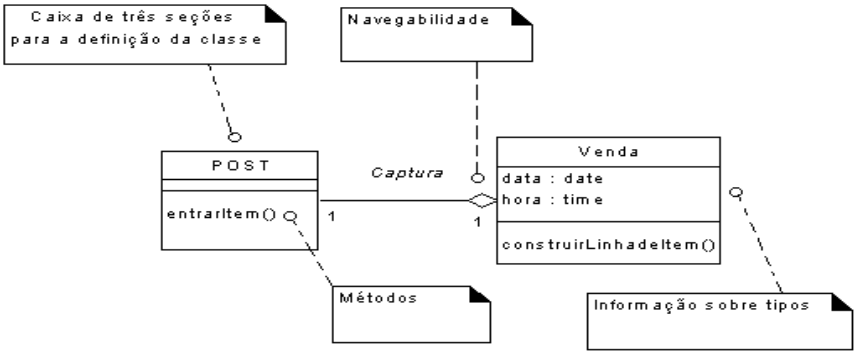


Figura A.7: diagrama de classes em UML

A.6 DIAGRAMA DE COLABORAÇÃO

Diagramas de Colaboração mostram a definição de classes e o fluxo de mensagens entre os objetos de *software*. Por exemplo, o diagrama da figura A.9 (Larman, 1998), ilustra o passo essencial do jogo, através do envio de mensagens à instâncias das classes *Jogador* e *Dado*.

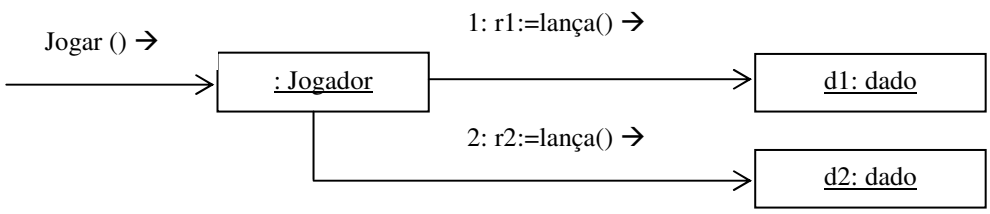


Figura A.8: Diagrama de Colaboração ilustrando mensagens entre objetos de *software*

Apêndice B
Metodologia para Análise e Projeto Orientada a Agentes

Este apêndice descreve a Metodologia para Análise e Projeto Orientada a Agentes proposta por (Wooldridge *et al*., 1999)*, utilizada na Modelagem do TUTA.*

B.1 INTRODUÇÃO

A idéia chave da Metodologia para Análise e Projeto Orientada a Agentes proposta por (Wooldridge *et al*., 1999), consiste em definir um sistema baseado em agentes como uma sociedade ou organização artificial. Isto significa, que se deve pensar no sistema como tendo um comportamento similar a uma organização humana ou uma típica companhia, composta de departamentos e indivíduos, ocupando os mais diversos cargos. Assim, o primeiro passo no desenvolvimento de um sistema baseado em agentes, consiste em definir o conjunto de papéis que fazem parte deste, e para isso deve-se fazer uma comparação aos cargos de uma organização humana. Os papéis especificados devem ser definidos através de três atributos, que são: permissões, reponsabilidades e protocolos. Tais conceitos são abstratos e servem para conceitualizar o sistema.

A metodologia é composta das fases de análise e projeto. Durante a análise, a ênfase está nos conceitos abstratos, e no projeto, a ênfase recai nas entidades concretas. Estas, tipicamente, têm contrapartida direta no sistema em tempo de execução. Os principais modelos usados na metodologia estão ilustrados na Figura B.1.

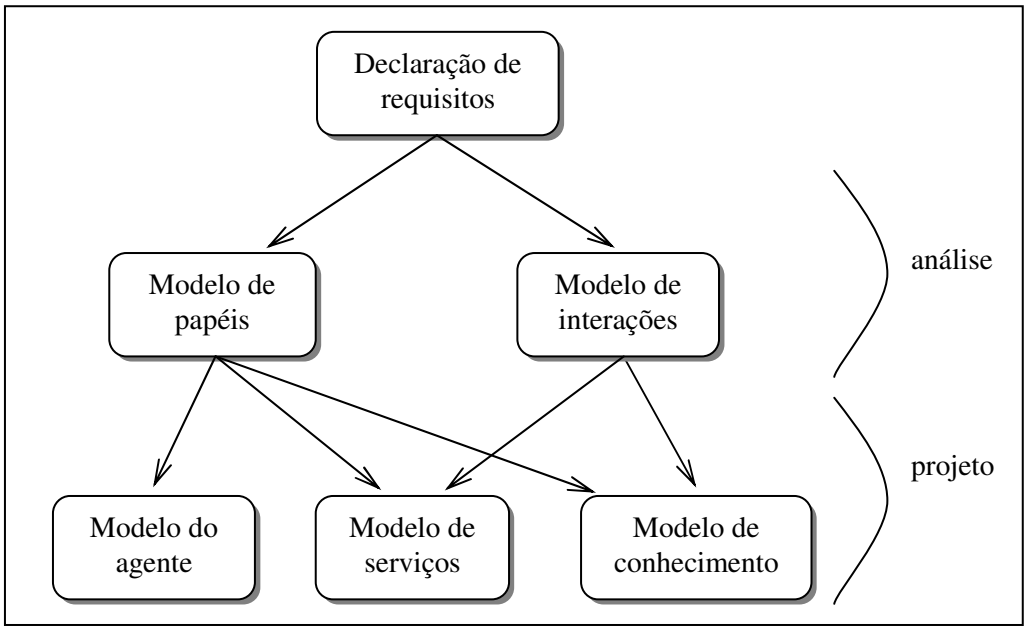


Figura B.1: Modelos da Metodologia e os seus relacionamentos

B.2 ANÁLISE

A fase de análise tem como objetivo capturar a compreensão do sistema. Nesta metodologia, esse objetivo é alcançado através do estudo da organização do sistema. Para definir uma organização, é necessário definir os papéis da organização, como estes se relacionam uns com os outros e como um papel pode interagir com outros papéis. Para isso, o modelo da organização é compreendido de dois modelos: modelo de papéis e modelo de interações.

Os passos que devem ser considerados para o desenvolvimento da fase de análise são os seguintes:

1. A partir da descrição comportamental das operações do sistema, deve-se definir os papéis existentes na organização. Em muitos casos, existe um mapeamento um-para um entre departamentos e papéis. Entretanto, existem casos em que o comportamento de um departamento pertence a dois papéis distintos. A saída deste passo é uma lista dos papéis fundamentais no sistema e uma descrição informal de cada um deles.

2. Para cada papel do sistema, devem ser identificados e documentados os protocolos associados. Nesta metodologia, protocolos são os padrões de interações que acontecem no sistema entre os vários papéis. A saída deste passo é um modelo de interação que captura as interações entre os papéis.

3. Usando os protocolo definidos no passo anterior, elabora-se então um modelo de papéis. A saída deste passo é um modelo de papéis que documenta os papéis fundamentais que acontecem no sistema, suas permissões, responsabilidades e os protocolos que eles fazem parte.

B.2.1 MODELO DE PAPÉIS

O Modelo de Papéis identifica os papéis fundamentais no sistema. Esse modelo é composto de um conjunto de esquemas de papéis (um para cada papel do sistema). O esquema de um papel é composto pelas permissões/direitos associados a ele, suas responsabilidades e os protocolos nos quais participa. O modelo de definição do esquema de um papel está ilustrado na Figura B.2.

ESQUEMA DO PAPEL:	*Nome do papel*
DESCRIÇÃO:	*Pequena descrição do papel*
PROTOCOLOS:	*Protocolos nos quais o papel participa*
PERMISSÕES:	*"Direitos" associados ao papel*
RESPONSABILIDADES	
LIVENES:	*Responsabilidades Liveness*
SEGURANÇA:	*Responsabilidades de segurança*

Figura B.2: Modelo para o esquema de um papel

O exemplo de um papel denominado *Enchedor de Café* (Wooldridge et al., 1999) está ilustrado na Figura B.3

ESQUEMA DO PAPEL:	*Enchedor de Café*
DESCRIÇÃO:	Este papel envolve assegurar que o café seja mantido cheio, e informa os trabalhadores quando o café acabou de ser feito
PROTOCOLOS:	*Encher, InformarTrabalhadores, ChecarEstoque* e *EsperarEsvaziar*.
PERMISSÕES:	Ler cafeteira a ser abastecida // nome da cafeteira Ler estado do Café // cheio ou vazio Alterar estoque do Café // nível de estoque do café
RESPONSABILIDADES	
LIVENES:	*Enchedor de Café* = (*Encher.InformarTrabalhadores.ChecarEstoque.EsperarEsvaziar*)∞
SEGURANÇA:	*Estoque de Café > = 0*

Figura B.3: Esquema do Papel *Enchedor de Café*

A seguir, descrevem-se os atributos básicos de um papel (Permissões, Responsabilidades e Protocolos).

- **Permissões:** representam os recursos de informação que podem ser utilizados por um papel para desempenhar suas funcionalidades. Como exemplo, um papel pode ter a habilidade de ler, modificar ou gerar um item particular de informação. A forma de definir permissões está baseada na notação *FUSION*, especificada por (Coleman *et al*., 1994, apud Wooldridge *et al*., 1999).

- **Responsabilidades:** representam as funcionalidades de um papel e podem ser divididas em responsabilidades vitais e responsabilidades de segurança.

  - **Responsabilidades Vitais:** São aquelas que declaram o que será feito por um papel. Tais responsabilidade são expressas na metodologia através de *expressões vitais. Expressões vitais* definem o ciclo de vida de um papel e são semelhantes às

expressões *ciclo de vida* do método *FUSION,* especificadas por (Coleman *et al*., 1994). A forma geral de uma *expressão vital* é especificada da seguinte forma: *Nome do papel = Expressão vital*

Os operadores permitidos para a especificação das expressões vitais, estão representados na Figura B.4.

x.y	x seguido de y	x|y	x ou y ocorrem
x*	x ocorre 0 ou mais vezes	x +	x ocorre 1 ou mais vezes
x∞	x ocorre infinitamente freqüentemente	[x]	x é opcional
x ‖ y	x e y acontecem intercaladamente		

Figura B.4: Operadores permitidos para expressões vitais

Para ilustrar como devem ser definidas tais expressões, será utilizado o exemplo do papel *Enchedor de Café*. Esse exemplo, especifica que o propósito do papel *Enchedor de Café* é assegurar que uma panela de café seja mantida cheia para um grupo de trabalhadores. Assim, as responsabilidades *vitais* para esse papel podem ser as seguintes:

- Sempre que o café terminar, encha.

- Sempre que terminar de preparar o café, avise os trabalhadores.

Assim, o papel *Enchedor de Café*, pode ser expresso da seguinte forma:

Enchedor de Café = (*Encher.InformarTrabalhadores.ChecarEstoque.EsperarEsvaziar*)∞

Essa expressão significa que o papel *Enchedor de Café* executa o protocolo *Encher*, seguido pelos protocolos: *InformarTrabalhadores, ChecarEstoque* e *EsperarEsvaziar*.

- **Responsabilidades de Segurança:** São aquelas que um agente deve manter ao desempenhar um papel. No exemplo anterior, voltando ao papel *Enchedor de Café*, pode ser necessário que um agente o qual desempenha esse papel, assegure que o café nunca fique vazio. Assim, é possível especificar a seguinte expressão de segurança: *Estoque de Café > = 0*

- **Protocolos:** Os componentes atômicos de uma expressão *vital* são os *protocolos*. Protocolos serão largamente utilizados no modelo de interações, pois representam uma interação entre papéis.

B.2.2 MODELO DE INTERAÇÕES

O modelo de interações serve para capturar o conjunto das interações que existem entre os diversos papéis especificados no sistema. Esse modelo é composto de um conjunto de definições de protocolo, um para cada tipo de interação *entre papéis,* onde o foco principal está na natureza e no propósito das interações, diferentemente dos típicos diagramas de interação que utilizam OO e têm como foco a ordem precisa de troca das mensagens. A Figura B.5 ilustra o modelo utilizado para definição de protocolos entre papéis.

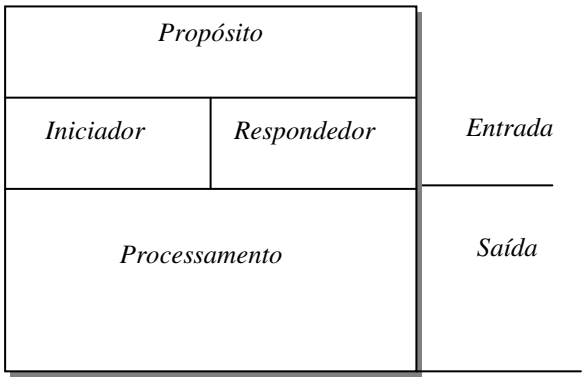


Figura B.5: Modelo para definição de um protocolo

Os seguintes atributos devem ser considerados nas definições de protocolo:

- Propósito: breve descrição da natureza da interação (por exemplo “solicitação de informações”, “assinalamento de tarefas”, entre outros.
- Iniciador: o(s) papel (éis) responsável (eis) por iniciar a interação.
- Respondedor: o(s) papel (éis) com que o iniciador interage.
- Entradas: informações usadas pelo papel iniciador enquanto está realizando o protocolo.
- Saídas: informações fornecidas pelo/para o respondedor durante o acontecimento da interação.
- Processamento: breve descrição de qualquer processamento que o iniciador do protocolo executa durante o acontecimento da interação.

Na Figura B.6 é definido o protocolo *Encher* que faz parte do papel *EnchedorDeCafé,* definido na seção B.3.1. O protocolo *Encher* é iniciado por esse papel e envolve o papel *MáquinaDeCafé*. O protocolo especifica que o papel *EnchedorDeCafé* enche o café na máquina nomeada *Cafeteira* e o resultado é que *MáquinaDeCafé* fica informada sobre o valor do *EstoqueCafé*.

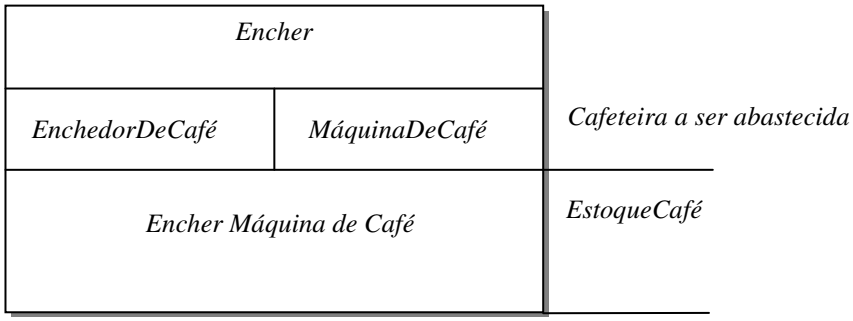


Figura B.6: Definição do protocolo *Encher*

B.3 PROJETO

O objetivo da fase de Projeto é transformar os modelos abstratos derivados durante a fase de análise em um nível suficientemente baixo de abstração, onde técnicas de projeto tradicionais (incluindo técnicas OO) possam ser aplicadas. Ou seja, após a realização das fases de análise e projeto, é necessário passar por uma nova fase de projeto. Essa nova fase (que pode ser projeto OO) deve tornar possível a realização da implementação do sistema.

A fase de projeto desta metodologia permite a geração dos Modelo de Agentes, Modelo de Serviços e Modelo de Comunicação, que devem ser desenvolvidos através dos seguintes passos:

1. Elaboração de um Modelo de Agentes.

2. Elaboração de um Modelo de Serviços, após a análise dos protocolos e das propriedades de segurança e *vitais* dos papéis.

3. Elaboração de um Modelo de Comunicação a partir do Modelo de Interações e do Modelo de Agentes.

B.3.1 MODELO DE AGENTES

O Modelo de Agentes é composto pelos tipos de agentes que farão parte do sistema. Esse modelo é construído a partir da análise dos papéis do sistema. O desenvolvedor pode estabelecer que um papel eqüivale a apenas um tipo-de-agente ou escolher empacotar vários papéis em

um mesmo tipo de agente, por questão de conveniência ou eficiência, por exemplo. As questões relacionadas à eficiência podem estar ligadas à fatores como: poder do processador, espaço de memória, entre outras. Entretanto, é necessário que o desenvolvedor se preocupe também com a coerência, ou seja, que empacote papéis em apenas um tipo de agente, se estes possuírem objetivos comuns.

A notação utilizada para especificar tipos-de-agentes está representada na Figura B.7. A seguinte cardinalidade pode ser utilizada para representar o número de instâncias dos agentes em tempo-de-execução:

- *n*: significa que existirão não menos que *n* instâncias deste tipo no sistema em tempo-de-execução;
- *m..n*: significa que existirão não menos que *m* e não mais que *n* instâncias deste tipo no sistema em tempo-de-execução;
- *: significa que poderão existir zero ou mais instâncias em tempo-de-execução;
- +: significa que poderão existir uma ou mais instâncias em tempo-de-execução.

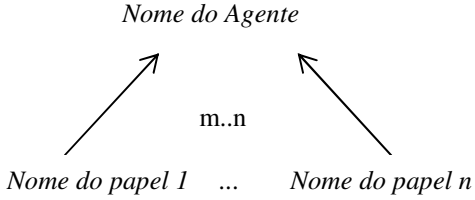


Figura B.7: Modelo para especificação de um tipo-de-agente

Exemplos de especificações de tipos de agentes estão ilustrados na Figura B.8 (Wooldridge *et al*., 1999).

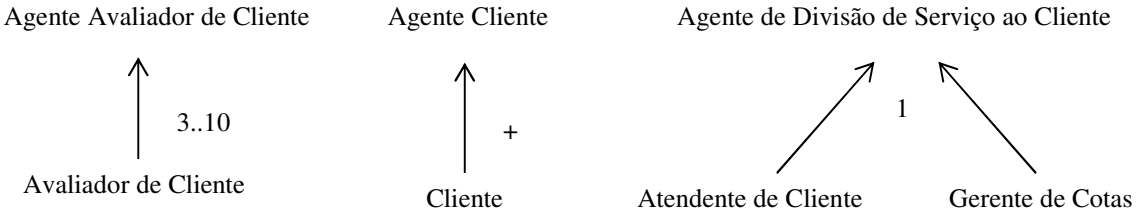


Figura B.8: Especificações de tipos-de-agentes

B.3.2 MODELO DE SERVIÇOS

O objetivo do Modelo de Serviços é identificar os serviços e as principais propriedades associadas a cada papel de agente. Serviços são as funções dos agentes. Em OO, um serviço cor-

responde a um método. No entanto, os métodos de um objeto estão disponíveis para outros objetos invocarem e este não é o objetivo da abordagem Orientada a Agentes (OA). Ou seja, os serviços de um agentes não devem estar disponíveis para outros agentes.

As seguintes propriedades precisam ser especificadas para cada serviço:

- entrada;
- saída;
- pós-condições;
- pré-condições.

Os serviços são derivados a partir da lista de protocolos e responsabilidades associadas a um papel, e em particular, a partir das *responsabilidades* vitais de um papel. As Entradas e Saídas podem ser derivadas de um modo óbvio a partir da análise dos protocolos no modelo de interações, enquanto que as pré-condições e pós-condições representam restrições nos serviços.

No exemplo do sistema de *café*, há quatro protocolos associados com o papel *Enchedor de Café,* que são: *Encher, InformarTrabalhadores, ChecarEstoque* e *EsperarEsvaziar*. Existe então, pelo menos um serviço associado a cada protocolo. No caso de *ChecarEstoque*, o serviço (que pode ter o mesmo nome), terá como entrada o nível de estoque e algum valor limiar e simplesmente comparará os dois. As pós-condições e pré-condições declararão que o nível de estoque do café é maior que 0 (zero) e esta é uma das condições de segurança do papel.

Após especificar os serviços de um sistema, o desenvolvedor pode realizar os serviços em qualquer estrutura de implementação que julgar apropriada. Poderá, por exemplo, implementar os serviços através de métodos em um linguagem OO.

B.3.3 MODELO DE COMUNICAÇÃO

O Modelo de Comunicação define as ligações de comunicação que existem entre os diversos tipos de agentes do sistema. Em particular, o propósito de um modelo de comunicação é identificar qualquer potencial gargalo de comunicação que pode causar problemas em tempo de execução. Esse modelo pode ajudar a analisar o grau de acoplamento entre os diversos agentes do sistema, a fim de diminuí-lo.

Um modelo de comunicação de agente é representado por um grafo, onde os nós do grafo correspondem a tipos de agentes e os arcos correspondem aos caminhos de comunicação que existem entre os agentes. Esse Modelo é especificamente um grafo direcionado, ou seja, um arco $a \rightarrow b$ indica que a enviará mensagens para *b*, mas não necessariamente que *b* enviará mensagens para *a*.

O modelo de comunicação pode ser derivado diretamente dos papéis, protocolos, e do modelos de agentes. Um exemplo de modelo comunicação está ilustrado na Figura B.9 (Wooldridge *et al*., 1999).

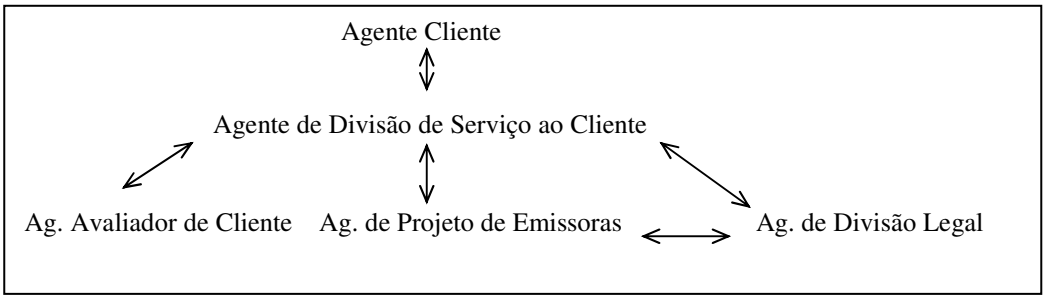


Figura B.9: Modelo de Comunicação para Gerência de Negócios Empresariais

Apêndice C
Java

Este apêndice descreve alguns aspectos da linguagem de Programação Java, utilizada nesta dissertação para implementar o TUTA.

C.1 INTRODUÇÃO

Java é uma linguagem orientada a objetos, que destaca-se por possuir algumas características, como (Lernay & Perkins, 1996): Simplicidade, Independente de Plataforma, Robusta, Segura, Multi-linha (do inglês: Multithreading), entre outros.

C.2 DEFINIÇÕES BÁSICAS EM JAVA

- **Classes:** são representações gerais para um objeto, ou seja, uma espécie de modelo para um objeto que contém variáveis e métodos representando atributos e comportamentos.

- **Atributos:** são definidos através de variáveis e são análogos à variáveis globais.

- **Comportamento:** o comportamento de uma classe em JAVA é definido através de métodos.

- **Definição de uma Classe:**

  class nome_da_classe extends nome_da_superclasse {
  Variáveis
  Métodos
  }

  Onde:

  - *class:* é a palavra reservada que marca o início da declaração de uma classe.

  - *nome_da_classe:* é o nome dado à classe.

- **Definição de variáveis:** a definição de variáveis em *Java* é feita da seguinte forma: *tipo_var nome_var*

Onde:

- *tipo_var* é o tipo de dado permitido para uma variável. Em Java, uma variável pode assumir os seguintes tipos:
  - Um dos 8 tipos primitivos: *integer* (byte: 8 bits (-128 a 127); *short*: 16 bits (-32768 a 32767); *int*: 32 bits (2 147 483 648 a 2 147 483 647); *long*: 64 bits (...).
  - Nome de uma classe/interface.
  - Um array.
- *nome_var:* é o nome dado à variável.

- **Definição de Métodos:** *returntype nome_do_método (tipo 1, tipo 2, tipo 3, tipo 4...)*
  { *corpo_do_método*
  }

Onde:

- *nome_do_método*: nome do método.
- *returntype:* tipo de objeto que ele retorna (que pode ser: um tipo primitivo, ou nome de uma classe, ou a palavra *void,* senão retorna nada).
- *tipo 1, tipo 2, tipo 3, tipo 4:* é a possível lista de parâmetros que o método utiliza.
- *corpo_do_método*: é o código existente no Método.

C.3 PRINCIPAIS CLASSES E MÉTODOS UTILIZADOS NA COMUNICAÇÃO DO TUTA

C.3.1 CLASSES UTILIZADAS NA COMUNICAÇÃO

Uma aplicação pode aguardar uma conexão numa porta pública não utilizada de uma máquina e quando um outro processo realizar uma conexão para essa máquina na mesma porta, estará estabelecida uma comunicação entre eles, um *Socket*. Uma aplicação pode esperar utilizando um objeto da classe *ServerSocket*. A seguir, uma explanação dessas Classes e de alguns conceitos utilizados por elas:

- **Socket:** classe que representa um cliente *Socket*.

  Exemplo: public Socket (String Host, int Porta)

  Onde:

  - **Host:** string que contém o endereço IP que foi definida a conexão.
  - **Porta:** é um endereço numérico por meio do qual são processadas as solicitações de novos serviços.

- **ServerSocket:** classe que representa um servidor de socket´s, aguardando requisições de conexões pela Internet numa determinada porta da máquina.

  Exemplo: public ServerSocket (int Porta, int Quantos)

  Onde:

  - **Porta:** é um endereço numérico por meio do qual são processadas as solicitações de novos serviços.
  - **Quantos:** é o número máximo de conexões que o servidor pode receber nessa porta.

C.3.2 MÉTODOS USADOS NA COMUNICAÇÃO

Dois métodos são essenciais na Comunicação entre clientes e servidores: getInputStream e getOutputStream, que permitem escrever ou ler informações.

Exemplo: public InputStream getInputStream ()

public OutPutStream getOutputStream ()

Apêndice D

Definição de uma Estratégia Didática no TUTA

Este apêndice apresenta como uma estratégia didática deve ser definida no TUTA, a fim de que possa ser acessada e executada. Assim, será apresentado o formato ao qual esta deve obedecer e um exemplo completo da definição de uma Estratégia Didática.

D.1 FORMATO

O TUTA executa conforme o que foi definido pelo professor na tabela *estr_didática*. Mas alguns valores precisam estar definidos em outras tabelas pra que a tabela *estr_didática* possa ser utilizada corretamente pelo Agente responsável pela Execução da Estratégia Didática. Essas tabelas são as seguintes:

- ***def_estr_didática:*** contém a identificação das estratégias didáticas cadastradas no TUTA.
- ***ação_estr_didática:*** contém as táticas de ensino cadastrados no TUTA.
- ***tipo_entidade_didática:*** contém os tipos de entidades didáticas cadastradas no TUTA.
- ***entidade_didática:*** contém os conteúdos de cada tipo de entidades didáticas cadastradas no TUTA.
- ***tipo_assunto:*** contém os tipos de assunto cadastrados no TUTA.

Assim, os campos da tabela *estr_didática* devem obedecer as seguintes regras:

- **Campo:** ***seq_estr_didatica***

  Descrição: identificação de qual estratégia didática está sendo executada

  Valores Permitidos: qualquer número que esteja previamente armazenado na tabela *def_estr_didatica*.

- **Campo:** ***seq_ação_est***

  Descrição: identificação da ordenação de uma ação em uma estratégia

  Valores Permitidos: números seqüenciais. Exemplo: 1, 2, 3, e assim por diante.

- **Campo: *seq_ação***

  Descrição: identificação de qual o tipo de ação será utilizada na estratégia

  Valores Permitidos: qualquer número que esteja previamente armazenado na tabela *ação_estr_didática*. Atualmente os números permitidos são:

  1: reutilização (mostrar)

  2: condição (se)

  3: *salta*

  4: *chat* ou *bate-papo*

  5: tempo

  6: trocar estratégia

  7: fim da sessão

- **Campo: *seq_tipo_ent***

  Descrição: identificação do tipo de entidade utilizado em uma ação da estratégia didática.

  Valores Permitidos: qualquer número que esteja previamente armazenado nas tabelas: *tipo_ entidade_didática* e *entidade_didática*. Atualmente os números existentes na tabela *tipo_ entidade_didática* são:

  1: definição

  2: explicação

  3: exemplo

  4: questão aberta

  5: questão de múltipla escolha

  6: questionário

  7: ini

  8: fim

- **Campo: *seq_assunto***

  Descrição: identificação do assunto utilizado em uma ação da estratégia didática

  Valores Permitidos: qualquer número que esteja previamente armazenado nas tabelas: *tipo_assunto* e *entidade didática*. Atualmente os números existentes na tabela *tipo_assunto* são:

  1: classes

  2: atributos

  3: métodos

  4: mensagem

  5: construtor

  6: destrutor

  7: herança

  8: generalização/especialização

  9: classes abstratas

  10: composição

- **Campo: *seq_obj_ent***

  Descrição: identificação de uma certa entidade didática em particular. Por exemplo, *primeira* definição de objeto, *segunda* definição de objeto, entre outros.

  Valores Permitidos: qualquer número que esteja previamente armazenado na tabela *entidade_didática.*

- **Campo: *num_muda_est***

  Descrição: identificação de uma estratégia didática

  Valores Permitidos: Qualquer número que esteja previamente armazenado na tabela *def_estr_didatica*.

- **Campo: *num_tempo***

  Descrição: identificação de um tempo permitido entre uma e outra ação da estratégia didática. Também utilizado quando se pretende definir um certo tempo para um *bate-papo.* Nesse caso, o tempo deve ser estabelecido antes da definição de início do bate-papo

  Valores Permitidos: qualquer número (unidade de tempo utilizada: minutos).

- **Campo: *num_linha_salta***

  Descrição: identificação de uma linha da estratégia didática

  Valores Permitidos: qualquer número que tenha sido definido nesta mesma tabela, no campo *seq_ação_est*. Atenção: só é permitido saltar para outra ação de uma mesma estratégia didática.

D.2 EXEMPLO DA DEFINIÇÃO DE UMA ESTRATÉGIA DIDÁTICA

Nesta seção, mostra-se passo-a-passo, como é realizada a inclusão de uma estratégia didática na *tabela estr_didática*.

- *1º passo*: verificar se existe na tabela *def_estr_didática*, a estratégia que se pretende utilizar. Por exemplo, a tabela *def_estr_didática* pode estar organizada da seguinte forma:

seq_estr_didática	*desc_estr_didática*
1	*Estratégia 1*
2	*Estratégia Pattern LDLL*

Tabela C.1: Tabela *def_estr_didática*

- *2º passo*: incluir o número seqüencial que indica a ação daquela estratégia didática. Exemplo: 1, 2, 3, e assim por diante.

- 3º passo: verificar na tabela *ação_estr_didática* a ação que se deseja incluir.

seq_ação	*desc_ação*
1	*Mostrar*
2	*Condição (se)*
3	*Salta*
4	*Bate-papo*
5	*Tempo*
6	*Trocar estratégia*
7	*sessão*

Tabela C.2: Tabela *ação_estr_didática*

- 4º passo: verificar na tabela *entidade_didática,* as entidades didáticas que se deseja incluir na ação da estratégia didática.

seq_ tipo_ent	*seq_ assunto*	*seq_ obj_ent*	*desc_entidade*
1	1	1	Classe é um modelo para construir objetos. Contém o conjunto de variáveis e métodos que um objeto pode possuir
1	1	2	Classe é um modelo para criar objetos, bastante similar a um tipo de variável.
2	1	2	Charter´s é uma empresa de transporte regional, com serviços de aeronave pequenas para destinos próximos.
2	1	3	Charter´s precisa de um aplicativo para planejar a escala de vôos e fazer reservas.
2	1	4	Um objeto é uma pessoa, lugar ou coisa. Em Software, é uma parte do software executável com seus próprios valores e comportamento.
3	1	3	==> Por exemplo, um objeto de voô marcado deve ter seu próprio valor (data= 29 de marco) e seu comportamento (adicionar reserva)
4	1	1	==> Questão: Expresse este objeto graficamente ou textualmente (em código Java propriamente dito). Tempo: 10 minutos !!!
4	1	2	==> Questão: Suponha um programa que controla um motor elétrico através de uma saída serial. A veloc. do motor é proporcional
4	1	3	==> a tensão aplicada e esta proporcional aos bits que vão para saída serial e passam por um conversor digital analógico.
4	1	4	==> Vamos abstrair toods estes detalhes, por enqto e modelar somente a interface do motor como uma classe.
4	1	5	==> Que métodos e que atributos deve ter esta classe, que argumentos e valores de retorno devem ter os métodos ? => Tempo: 10 m.

Tabela C.3: Tabela *entidade_didática*

- 5º passo: verificar se após a execução de uma ação, haverá troca de estratégia. Caso a resposta seja verdadeira, incluir o número da estratégia didática.

- 6º passo: caso a ação incluída seja uma ação de tempo, especificar o tempo em minutos.

- 7º passo: verificar se após a execução de uma ação, haverá troca de ação, para uma ação que não seja seqüencial àquela que acabou de ser executada. Caso a resposta seja verdadeira, incluir o número da seqüência da ação.

O exemplo completo do preenchimento da tabela *estr_didática* está representado na Tabela C.4, com a estratégia 1 definida.

seq_estr_didática	*seq_ação_est*	*seq_ação*	*seq_tipo_ent*	*seq_assunto*	*seq_obj_ent*	*num_muda_est*	*num_tempo*	*num_linha_salta*
1	1	1	1	1	1	0	0	0
1	2	5	7	0	0	0	1	0
1	3	4	7	0	0	0	0	0
1	4	5	7	0	0	0	1	0
1	5	1	1	1	2	0	0	0
1	6	5	7	0	0	0	1	0
1	7	4	8	0	0	0	0	0
1	8	1	1	1	1	0	0	0
1	9	7	8	0	0	0	0	0

Tabela C.4: Tabela *estr_didática*

Apêndice E
Implementação dos Agentes do TUTA

Este apêndice apresenta o código de implementação dos agentes do TUTA (referentes à versão 1.0).

E.1 CÓDIGO IMPLEMENTADO

Código do Agente de Comunicação

```
// AGENTE SERVIDOR DE COMUNICACAO
//
import java.net.*;
import java.io.*;
import java.util.*; /* necessita-se de um vetor para manipular um numero desconhecido de conexoes*/

public class AgComunic
{
  public AgComunic() throws IOException

  {
    // inicializar novo ServerSocket
    ServerSocket s = new ServerSocket(23, 10);

    // loop para as conexoes - instanciar um handler para cada conexao
    while(true)
    {
      Socket client = s.accept();
      System.out.println("Conexao aceita para o endereco: " + client.getInetAddress()+":"+ client.getPort());
      chatHandler c = new chatHandler(client);
      c.start(); // chatHandler e' derivado da Thread
    }
  }

  public static void main(String[] args) throws IOException
  {
    AgComunic cs = new AgComunic();
  }

} // fim da definicao da classe

// definir uma classe chatHandler como uma derivacao da Thread
class chatHandler extends Thread
{
  // declaracao de variaveis
  protected Socket s;
  protected BufferedReader in;
  protected PrintWriter out;
  protected static Vector handlers = new Vector();
```

```
public chatHandler(Socket s) throws IOException
{
  // designar um socket para o objeto da variavel de classe s
  this.s = s;
  // inicializar as cadeias
   in = new BufferedReader(new InputStreamReader(s.getInputStream()));
   out = new PrintWriter(s.getOutputStream(), true);
   System.out.println("Agente de Comunicacao inicializado");
}

public void run()
{
  System.out.println("chamando o metodo run...");
  try
  {
    // adicionar uma thread a variavel vetor
    handlers.addElement(this);
          System.out.println("Elemento adicionado ao Vetor");

    // loop que le a entrada e passa para o metodo broadcast
    while(true)
    {
      String message = in.readLine();
           System.out.println("Lendo a cadeia - input");
           System.out.println(message);
           broadcast(message);
           System.out.println("Mensagem passada para o metodo broadcast...");
    }
  }
  catch(IOException e)
  {
    System.err.println(e);
  }
  // Se esta thread is cancelada, entao...
  finally
  {
    // remover do vetor
    handlers.removeElement(this);
    try
    {
      // arrumar os restantes...
      s.close();
    }
    catch(IOException e)
    {
      System.err.println(e);
    }
  }
}

// este metodo e' usado para todas as threads
protected static void broadcast(String message)
{
  /* para prevenir que duas threads tentem
                acessar o vetor simultaneamente
          usa-se um bloco sincronizado*/

     System.out.println("Chamando o Broadcast...");
  synchronized(handlers)
```

```
    {
      /* A enumeracao das interfaces eh uma forma rapida de
                    procurar no vetor  */

      Enumeration e = handlers.elements();

      // retorna verdadeiro se existe outros sockets executando
      while (e.hasMoreElements())
      {
        // loop para o proximo chatHandler no vetor
        chatHandler c = (chatHandler) e.nextElement();

        // escreve msg para para ele
        try
        {
          /* prevenir que duas threads tentem escrever
             no stream c.out */
          synchronized(c.out)
          {
                                    if (!message.equals(null))
                                    {c.out.println(message);}
                                    else {System.out.println("tentativa de enviar null");}
          }  // fim do bloco de sincronizacao
          //c.out.flush();
        }
        catch(Exception ex)
        {
          System.err.println(ex);
          c = null;
        }

      } // fim do while
    } // fim do bloco de sincronizacao
  } // fim do broadcast

} // fim da classe
```

Código do Agente Gerenciador de Curso

```
// Agente Gerenciador de Curso
//
// este agente acessa uma base de fatos contendo as identificacoes de
// sessao
// a base deve estar no MS-ACCESS
// OBS: data no formato:  1/1/2000
//          hora no formato: 02:00
//
import java.net.Socket;
import java.io.IOException;
import java.io.DataInputStream;
import java.io.PrintStream;
import java.util.Date;
import java.util.Calendar;
import java.lang.*;
import java.net.URL;
import java.io.*;

class Data
{
```

```
 int yyyy, mm, dd;
}

public class AgGerCurso {

 public static void main(String[] args) {
  Socket s = null;
  Date d;
  Sessao db;

  Data data_hoje, data_p;
  int flag, flag_rec, dataok, inicio;
  String mensagem;
  data_p = new Data();
//
  try {
   s = new Socket("127.0.0.1",23);
    DataInputStream server_in = new DataInputStream(s.getInputStream());
    PrintStream server_out = new PrintStream(s.getOutputStream());
    DataInputStream input = new DataInputStream(System.in);

    String line, line_rec, hora_sessao;

            line = "Agente Gerenciador de Curso";
            flag = inicio = flag_rec = 0;
            line_rec = " ";
            dataok = 0;

            server_out.println (line);
            System.out.println("mensagem de apresentacao:" +line);
// Inicio
            while (true) {

                    inicio = 0;

// quando for fim de sessao, zera a variavel
// inicio - Isto porque ele continua testando ate'
// acontecer a proxima sessao
//  Se nao esta no meio de uma sessao... entao..
                    System.out.println("Monitorando os dias de sessao...");
              while (inicio == 0)
              {
                    // pega a data atual e pesquisa  no banco

              // recupera a data atual
              d = new Date();
              data_p.yyyy = (1900 + d.getYear());
              data_p.mm= (d.getMonth() + 1);
              data_p.dd = (d.getDate());

              // pesquisa no banco

              db = new Sessao();

              // recebendo a hora da sessao
              // --> alterarar para receber 3 parametros: hora, descricao, estrategia

              hora_sessao = db.acesso_sessao(data_p.dd, data_p.mm, data_p.yyyy);
             // mostra a data da sessao - HOJE!!!
             // System.out.println("---->" + Data_Banco.data_sessao_db);
```

```
            //testa se hoje tem sessao

             if (hora_sessao.equals(null)) {flag = 1;}
                    else {
                            System.out.println("Hoje tem Sessao as: " + hora_sessao);
                            // alterar para enviar hora da sessao, descricao e estrategia inicial
                            line = "Inicio de Sessao" + hora_sessao; inicio=1;
                                    server_out.println(line);}
                    // incluir para enviar o numero da estrategia associada a estrategia
                    // inicial

             }  // fim do while

             while (inicio == 1) {
                      line_rec=server_in.readLine();
                      if (line_rec.length() >= 3) {mensagem=line_rec.substring(0,3);}
                              else {mensagem=line_rec.substring(0,line_rec.length());}
                      if (mensagem.equals("Fim")) {
                                line="Fim de Sessao - Gerenciador de curso";
                                System.out.println("Fim de sessao");inicio=0; flag=0;}
             } // fim do while

           } // fim do while
  } // fim-try
  // Se ocorrer erros:
  catch (IOException e) { System.out.println(e);}
  finally {
             try { if (s != null) s.close() ; }
             catch (IOException e2) { System.out.println(e2);}
  }
 }

}
```

Código do Agente compositor de Grupo

```
// Agente Compositor de Grupo
//
// Servicos:
// 1. Acessa a base e valida os dados dos usuarios da sessao
//
// OBSERVACOES:
// a base deve estar no MS-ACCESS
// os campos USUARIO e SENHA devem conter 8 caracteres
//
import java.net.Socket;
import java.io.IOException;
import java.io.DataInputStream;
import java.io.PrintStream;
import java.util.Date;
import java.util.Calendar;
import java.lang.*;
import java.net.URL;
import java.sql.*;
import java.io.*;


public class AgCompGrupo {

 public static void main(String[] args) {
```

```
  Socket s = null;
  Usuario db;
  int flag, flag_rec, sessao_comecou;
  String mensagem, usuario_rec, usuario_result, senha_rec;
//
  try {
   s = new Socket("127.0.0.1",23);
    DataInputStream server_in = new DataInputStream(s.getInputStream());
    PrintStream server_out = new PrintStream(s.getOutputStream());
    DataInputStream input = new DataInputStream(System.in);

    String line, line_rec;

            line = "Agente Compositor de Grupo";
            flag = flag_rec = sessao_comecou= 0;
            line_rec = " ";

// Inicio
            while (true) {
              if (flag == 0) {
                        server_out.println (line);
                        System.out.println("mensagem enviada:" +line);
                        flag=1;}

                        line_rec=server_in.readLine();

// fica atento para saber se a sessao
// ja comecou... caso tenha comecado, nao deixa mais ninguem
// logar no sistema
              if (line_rec.length() >= 10)
                      {mensagem=line_rec.substring(0,10);}
                      else {mensagem=line_rec.substring(0,line_rec.length());}

                      // quando a sessao comecar, desabilitar o login
                      // ninguem pode mais entrar na sessao


                              if (mensagem.equals("Iniciar as"))
                               {sessao_comecou = 1;
                              System.out.println("A sessao comecou. Login desabilitado");}

                              // -- ficar atento para o fim de sessao

                        if (line_rec.length() >= 3) {mensagem=line_rec.substring(0,3);}
                              else {mensagem=line_rec.substring(0,line_rec.length());}
                        if (mensagem.equals("Fim")) {
                                  line="Sessao terminada";
                                  System.out.println("Fim de sessao");
                                  sessao_comecou=0;}

                              if (line_rec.length() >= 10)
                                        {mensagem=line_rec.substring(0,10);}
                                        else {mensagem=line_rec.substring(0,line_rec.length());}

                                        if (mensagem.equals("Compositor"))
                                        {
                              if (line_rec.length() >= 19)
                                        {usuario_rec=line_rec.substring(11,19);}
                                        else {usuario_rec="invalido";}
                              System.out.println("usuario: " + usuario_rec);
```

```
                        if (line_rec.length() >= 27)
                                {senha_rec=line_rec.substring(19,27);}
                                else {senha_rec="invalido";}
                        System.out.println("senha: " + senha_rec);
                        // pesquisa no banco
                        db = new Usuario();
                        usuario_result = db.acesso_login(usuario_rec, senha_rec);
                        // se usuario_rec receber null eh pq nao achou nada na base
                        System.out.println("usuario: " + usuario_result);
                        if (usuario_result.equals(null))  {
                                line="RESAI" + usuario_rec + "Invalido ...";
                                flag = 0;}
                        else {
                        if (sessao_comecou == 0) // ainda nao iniciou
                        {line="RESAI"+ usuario_rec +
                        senha_rec + "OK";
                        flag=0;   }
                        else {line="RESAIACESSO NEGADO  -"; flag=0;}
                        }
                                } //fim-se

        } // fim while
 }  // fim do try
 // Se ocorrer erros:
 catch (IOException e) { System.out.println(e);}
 finally {
            try { if (s != null) s.close() ; }
            catch (IOException e2) { System.out.println(e2);}
                        } // fim do try
 }  // fim do metodo
} // fim da classe
```

Código do Agente executor de Estratégias Didáticas

```
// Agente Executor de Estratégias Didáticas
//
// fique atento na hora
// hora no formato: 02:30, 02:15....
// cuidado com 02:00 (02:0)
// mude o fuso horario. o java pega corretamente se estiver
// setado para canada(leste) -5 horas.
//

import java.net.Socket;
import java.io.IOException;
import java.io.DataInputStream;
import java.io.PrintStream;
import java.util.Date;
import java.util.Calendar;
import java.*;

class Hora {
        int hh, mm, ss;
}

public class AgExecEstrDidaticas {

 public static void main(String[] args) {
```

```
 Socket s = null;
 int flag;
 int horaok;
 int hora_sessao, minuto_sessao;
 String line, line_rec, mensagem, hora_string, minuto_string;
 Date h;
 Hora hora;


// variaveis para a execucao das estrategias
        int num_est, n_acao, fim_estr;
        String nm_estrat;
        DefEstrDidatica def;


//
 try {
  s = new Socket("127.0.0.1",23);
 //
   DataInputStream server_in = new DataInputStream(s.getInputStream());
   PrintStream server_out = new PrintStream(s.getOutputStream());
   DataInputStream input = new DataInputStream(System.in);


   hora = new Hora();
        line = "Agente Executor de Estr.Didaticas";
        line_rec = hora_string = minuto_string = "";
        flag = horaok = hora_sessao = minuto_sessao = 0;


//
  while (true) {
          if (flag == 0) {flag=1;
                  server_out.println (line);
                  System.out.println("mensagem enviada: " +line);}

          line_rec=server_in.readLine();

          System.out.println("mensagem recebida: " +line_rec);
          if (line_rec.length() >= 10)
                  {mensagem=line_rec.substring(0,10);}
                  else {mensagem=line_rec.substring(0,line_rec.length());}

//
    while ((!mensagem.equals("Inicio de ")) & (horaok == 0)) {
                   line_rec=server_in.readLine();
          if (line_rec.length() >= 10)
                  {mensagem=line_rec.substring(0,10);}
                  else {mensagem=line_rec.substring(0,line_rec.length());}

          } //fim do while
//
     if (mensagem.equals("Inicio de ")) {
                                        System.out.println("esperando a hora...");
                                        h = new Date();
          hora.hh=(h.getHours());
          hora.mm=(h.getMinutes());
                  System.out.println("A hora do sistema e': " + hora.hh + ":" +  hora.mm);
          hora_sessao = Integer.parseInt(line_rec.substring(16,18));
          minuto_sessao = Integer.parseInt(line_rec.substring(19,21));
```

```
                System.out.println("Oba! Hoje vai ter sessao as " + hora_sessao + minuto_sessao );
//
    while (horaok == 0) {
    h = new Date();
    hora.hh=(h.getHours());
    hora.mm=(h.getMinutes());
        if (hora.hh == hora_sessao & hora.mm == minuto_sessao) {
                System.out.println("E' AGORA!!!");
                horaok=1;
                line = "Iniciar as Interfaces" + hora_sessao+minuto_sessao;
                System.out.println("A sessao comecou...");
                server_out.println(line);

        }  // fim do if
        }  // fim do while
        }  // fim do if

    // inicar a execucao das estrategias didaticas

    fim_estr = 0;

    num_est = 1;   // numero da estrategia. Alterar para receber msg
                                // do professor com este parametro

    n_acao = 1;  // a acao sempre comeca da primeira para cada estrategia

            // mostra qual estrategia sera executada

            def = new DefEstrDidatica();
            System.out.println("A estrategia executada sera : "
                    + def.acesso_def_estr_didatica(num_est));

    while (fim_estr == 0)
    {
            // variaveis para carregar a estrategia
            EstrDidatica c;
            int a [] = new int [9];
            // inicializando variaveis para carregar as acoes
            // da estrategia
            a[0]=0;a[1]=0;a[2]=0;a[3]=0;a[4]=0;
            a[5]=0;a[6]=0;a[7]=0; a[8]=0;

            c = new EstrDidatica();

            // chama o metodo para carregar as acoes
            for (int i = 1; i<=8; i++)
            {a[i]= c.acesso_estr_didatica(num_est,n_acao)[i];}

            // atualiza a acao para a proxima

            n_acao = n_acao + 1;

            // declara variaveis para carregar a acao e o tipo de entidade
                        TipoEntidade ac;
                        ac = new TipoEntidade();
                        Acao tp_e;
                        tp_e = new Acao();
                        String a2 = tp_e.acesso_acao(a[2]);
                        String a3 = ac.acesso_tipo_entidade(a[3]);
```

```
                System.out.println ("acao: " + a2  + " - e tipo de entidade: " + a3);

// testa se for fim de sessao

if ( (a2.equals("sessao")) & (a3.equals("fim")) )
{server_out.println ("Fim de sessao");
        fim_estr=1; horaok=0;}
else
{

        if(a2.equals("bate-papo"))
        {
                if (a3.equals("ini"))
                {server_out.println("BATEPAPOINICIO");}
                if (a3.equals("fim"))
                {server_out.println("BATEPAPOFIM");}
        } // fim do teste de bate-papo

        // Se for uma acao de tempo, chama o temporizador
        if(a2.equals("tempo"))
        {
                Temporizador tempo;
                String parametro, retorno;
                String t = "" + a[7];
                tempo = new Temporizador();
                retorno= tempo.chama(t);}

        // se for uma troca de estrategia, atualiza para a nova
        if (a2.equals("troca"))
        { System.out.println("Estrategia"+ num_est + " alterada para " + a[6]);
        num_est = a[6];
        n_acao=1;
        } // fim do teste de troca de estrategia

        // se for um salto na estrategia, atualiza o n_acao
        if (a2.equals("salto"))
        { System.out.println("Salto da acao "+ n_acao + " para a acao " + a[8]);
        n_acao= a[8];
        } // fim do teste de salto  de acao

        // se for uma acao de mostrar, entao...
        if (a2.equals("mostrar"))
        {server_out.println("SERDID" + a[3] + "," + a[4] + "," + a[5] + "*");
        line_rec=server_in.readLine();
        System.out.println("mensagem recebida: " +line_rec);
        if (line_rec.length() >= 6)
                {mensagem=line_rec.substring(0,6);}
                else {mensagem=line_rec.substring(0,line_rec.length());}

                // fica recebendo msgs ate que o servidor de
                // entidades didaticas devolva a entidade solicitada
                while (!mensagem.equals("OBJDID")) {
                        line_rec=server_in.readLine();
                        if (line_rec.length() >= 6)
                                {mensagem=line_rec.substring(0,6);}
                        else {mensagem=line_rec.substring(0,line_rec.length());}
                        } // fim do while
                server_out.println("EXIBIR"+ line_rec.substring(6,line_rec.length()));
                System.out.println ("objeto didatico enviado: " +
                line_rec.substring(6,line_rec.length()));
```

```
                        } // fim do teste da acao mostrar

                    } // fim do teste se a sessao - else

            } // fim do while


   }  // fim do while
  } // fim do try

  //
  // Se ocorrer erros:
  catch (IOException e) { System.out.println(e);}
  finally {
              try { if (s != null) s.close() ; }
              catch (IOException e2) { System.out.println(e2);}
  }
 }
}
```

Código do Agente Gerenciador de Interações

```
// Ag. Ger.Interações
//
import java.net.Socket;
import java.io.IOException;
import java.io.DataInputStream;
import java.io.PrintStream;
import java.util.Date;
import java.util.Calendar;

public class AgGerInteracoes {

public static void main(String[] args) {

  Socket s = null;
  int flag, interf_ok, inicio_sessao;

//

  try {
   s = new Socket("127.0.0.1",23);
  //
    DataInputStream server_in = new DataInputStream(s.getInputStream());
    PrintStream server_out = new PrintStream(s.getOutputStream());
    DataInputStream input = new DataInputStream(System.in);

    String line, line_rec, mensagem, usuario, senha;
            line = "Agente Gerenciador de Interacoes";
            line_rec = "";
            flag = interf_ok= inicio_sessao = 0;

            //----
            // Aguarda ate' receber uma mensagem avisando ser um dia de sessao
   while (true) {
              if (flag == 0) {
                        flag = 1;
                        server_out.println (line);
                        System.out.println("Mensagem enviada: " + line);}
                        line_rec=server_in.readLine();
```

```
//----
            if (line_rec.length() >= 10)
                    {mensagem=line_rec.substring(0,10);}
                    else {mensagem=line_rec.substring(0,line_rec.length());}
// Recebe mensagens ate q seja inicio de sessao
    while ((!mensagem.equals("Inicio de ")) &
                     (inicio_sessao == 0))  {
                     line_rec=server_in.readLine();
                      if (line_rec.length() >= 10)
                              {mensagem=line_rec.substring(0,10);}
                              else {mensagem=line_rec.substring(0,line_rec.length());}
                    }
           if (mensagem.equals("Inicio de ")) {inicio_sessao = 1;}
// -- ficar atento para o fim de sessao
                      if (line_rec.length() >= 3) {mensagem=line_rec.substring(0,3);}
                              else {mensagem=line_rec.substring(0,line_rec.length());}
                      if (mensagem.equals("Fim")) {
                                line="Sessao terminada";
                                System.out.println("Fim de sessao");
                                inicio_sessao=0;}
//------
// se a primeira mensagem que recebe vem de uma interface,
// entao envia para o compositor o USUARIO
// e a SENHA para serem validados
            if (line_rec.length() >= 9)
                    {mensagem=line_rec.substring(0,9);}
                    else {mensagem=line_rec.substring(0,line_rec.length());}
//          mensagem=line_rec.substring(0,9);
            if (mensagem.equals("Inter-VAL"))
            {
                     if (line_rec.length() >= 19)
                    {usuario=line_rec.substring(11,19);}
                    else {usuario="invalido";}
//
            if (line_rec.length() >= 27)
                    {senha=line_rec.substring(19,27);}
                    else {senha="invalido";}
            line = "Compositor:" + usuario + senha;
            flag=0;
            }
            System.out.println("msg recebida: " + line_rec);
//-------
// verifica se o compositor de grupo enviou alguma
// mensagem referente a validacao do usuario
// se sim, entao enviar para as interfaces...
           if (line_rec.length() >= 5)
           {
                    mensagem=line_rec.substring(0,5);

           if (mensagem.equals("RESAI")) // mensagem do compositor para avaliacao
            {
                     line="RESOK" + line_rec.substring(5,21);
                    System.out.println("Encaminhando msg para interface: " + line);

                    flag=0;
           }
           }

// recebe objetos didaticos para serem mostrados
           if (line_rec.length() >= 6)
```

```
            {
                        mensagem=line_rec.substring(0,6);

            if (mensagem.equals("EXIBIR")) // mensagem do compositor para avaliacao
             {
                        line="INTEREXIB" + line_rec.substring(6,line_rec.length());
                       System.out.println("Encaminhando msg para interface: " + line);

                       flag=0;
            }
            }

// recebe msg do executor de estrategia didatica
            // e trata o bate-papo
            if (line_rec.length() >= 9)
            {
                        mensagem=line_rec.substring(0,9);

            if (mensagem.equals("BATEPAPOI")) // mensagem do executor de estr
             {line="STARTBATP";
                       System.out.println("Comando para habilitar o bate-papo");     flag=0;}
            else { if(mensagem.equals("BATEPAPOF")) // mensagem do executor de estr
             {line="STOPBATEP";
                       System.out.println("Comando para desabilitar o bate-papo");  flag=0;}
            }
            }
// quando receber a mensagem de Iniciar as interfaces
// e' porque a sessao ja vai comecar,
// entao envia a mensagem para as interfaces
              if (line_rec.length() >= 10)
                        {mensagem=line_rec.substring(0,10);}
      if (mensagem.equals ("Iniciar as") &
                          (interf_ok == 0)) {
        line = "todos=A sessao comeca agora";
                        interf_ok = 1;
                        flag = 0;}
            }
  }
  //
  // Se ocorrer erros:
  //
  catch (IOException e) { System.out.println(e);}
  finally {
              try { if (s != null) s.close() ; }
              catch (IOException e2) { System.out.println(e2);}
  }
 }
}
```