Dissertação de Mestrado

FA_PorT: Um Framework para sistemas Portfólio-Tutor baseado em Agentes

Fábio Nicácio de Medeiros
fabionicacio@gmail.com

Orientador:
Arturo Hernández-Domínguez

Maceió, Dezembro de 2006

Fábio Nicácio de Medeiros

FA_PorT: Um Framework para sistemas Portfólio-Tutor baseado em Agentes

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

Orientador:

Arturo Hernández-Domínguez

Maceió, Dezembro de 2006

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

M488f Medeiros, Fábio Nicácio de.
FA_PorT : um framework para sistemas portfólio-tutor baseado em agentes / Fábio Nicácio de Medeiros. – Maceió, 2006.
xi, 147f. : il.

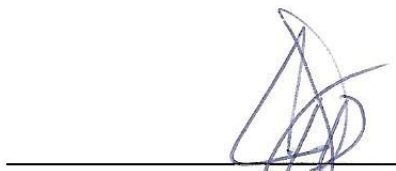
Orientador: Arturo Hernández-Domínguez.
Dissertação (mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2006.

Bibliografia: f. 143-147.

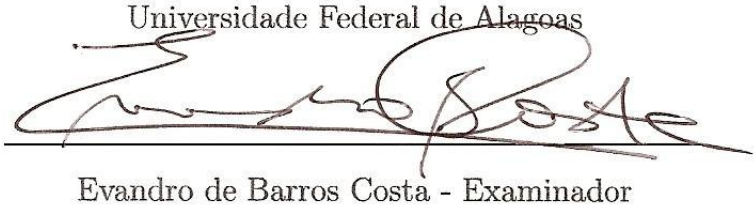
1. Framework (Arquitetura de Software). 2. Portfólio eletrônico. 3. Sistemas tutoriais inteligentes. 4. Agentes dos serviços de inteligência. 5. Ensino a distância. I. Título.

CDU: 004.78:37.018.43

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Curso de Mestrado em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.



Arturo Hernández-Domínguez - Orientador
Instituto de Computação
Universidade Federal de Alagoas



Evandro de Barros Costa - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Fábio Paraguaçu - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Edilson Ferneda - Examinador
Departamento de Sistemas e Computação
Universidade Católica de Brasília

Maceió, Dezembro de 2006

Agradecimentos

Este trabalho não teria sido concluído sem a ajuda de diversas pessoas a quem expresso aqui meus sinceros agradecimentos.

Em primeiro lugar, agradeço a Deus por tudo que ele me ensinou, através das dificuldades e alegrias que passei.

Aos meus pais, Socorro e Nicácio, e minha irmã, Natália, pelo incentivo em todos os momentos da minha vida principalmente nessa fase difícil. Vocês são a melhor família que eu poderia ter.

À minha namorada linda, Roberta, por me apoiar sempre, me amar muito e permanecer ao meu lado em todos os momentos da minha vida.

Agradeço ao professor Arturo Hernández Domínguez pela paciência e por confiar em mim nesse longo caminho.

Aos professores do mestrado pela dedicação e empenho em todas as disciplinas que participei.

Aos colegas do mestrado, em especial aos amigos Felipe, Allan e Flávio, por me ajudarem em Maceió e no mestrado.

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

E, a todos aqueles que não citei, mas que com certeza me ajudaram a chegar até aqui.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

A Educação a Distância (EAD) amplia o alcance da modalidade de ensino presencial, já que fornece aos indivíduos, independentemente do local onde moram ou tempo disponível, a oportunidade de iniciar ou complementar seus estudos. Em virtude da facilidade de acesso, disponibilidade e recursos de interação existentes, a Internet é uma mídia bastante utilizada na EAD.

O objetivo deste trabalho é projetar e implementar um framework, FA_PorT, para sistemas Portfólio-Tutor acoplado a um sistema Tutor Inteligente (STI) utilizando Agentes e apresentar duas aplicações Portfólio-Tutor, criadas a partir do framework.

O FA_PorT visa propor um modelo de ambiente interativo de ensino e aprendizagem baseado numa arquitetura para sistemas Portfólio-Tutor. Cada sistema Portfólio-Tutor construído a partir do framework pode ser utilizado no contexto de ambientes de EAD na web.

O framework proposto objetiva o reaproveitamento da estrutura interna (núcleo comum) de um sistema Portfólio-Tutor baseado em STI e Agentes, fazendo com que o tempo de desenvolvimento de um novo sistema Portfólio-Tutor seja reduzido.

Palavras-Chave: Educação a Distância, Framework, Portfólio Eletrônico, Sistema Tutor Inteligente e Agentes.

Abstract

Distance Education expands the scope of the inside classroom teaching model since anybody, independent of where he lives or how much available time he has, has the opportunity to initiate or complement his studies. Considering access facility, availability and interaction resources provided, the Internet is a midia sufficiently used for Distance Education.

This work has as objective the design and implementation of a framework, FA_PorT, for Portfolio-Tutoring systems coupled a Intelligent Tutoring Systems (STI) and using agents and present two applications Portfolio-Tutoring, created from the framework

The FA_PorT mainly aiming the design of a model computer-based interactive learning environment for Portfolio-Tutoring systems. Each system Portfolio-Tutoring will be able to be used in the context of environment for EAD in the web.

The framework proposed, aims to reuse of internal architecture (or skeleton) of a portfolio-tutoring system based on ITS and Agents, doing with that development time of a new portfolio-tutoring system will be reduced.

Keywords: Distance Education, Framework, Electronic Portfolio, Intelligent Tutoring Systems and Agents.

Capítulo 1

Introdução

1.1 Motivações

No ensino presencial[1], temos a Educação à Distância, que possibilita aos alunos e professores estudar independentemente do local onde moram ou do tempo disponível de cada um, tendo, dessa forma, a possibilidade de começar seus estudos e concluí-los.

A Educação à Distância, corroborando a mudança do paradigma educacional no ensino, isto é um novo modelo focado na aprendizagem (Valente 1998), exige uma mudança no perfil dos alunos e dos professores. Os alunos devem ter um comportamento ativo, participativo, devendo assumir uma responsabilidade maior do que em um ensino tradicional. E os professores passam a ser orientadores, devendo incentivar os alunos a pesquisarem e participarem efetivamente do novo sistema de educação.

O estudante quando está distante precisa de motivação para o acompanhamento do curso. Devido a isso, quanto mais adaptado for o local de estudo às suas preferências em relação a sua aprendizagem, maior será seu sucesso. No projeto de Frozza (Frozza 2000), a adaptação entre os alunos e os professores acontece através da geração de páginas dinâmicas e monitoração de suas atividades. Em outro projeto, (Souto 2000), a adaptação ocorre de acordo com o estilo cognitivo de aprendizagem.

1.2 Contexto da Dissertação

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

- Descoberta de Conhecimento e Otimização de Decisões;

[1]Forma de ensino que o professor e os alunos se encontram periodicamente no mesmo local e horário pré-determinado para a realização de uma aula.

- Modelagem Computacional em Educação;
- Modelos Quantitativos e de Simulação.

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

Neste trabalho, apresenta-se um framework para sistemas portfólio-tutor utilizando agentes, o FA_PorT, que busca apoiar as atividades do professor, auxiliando-o no processo de ensino, fornecendo mecanismos para que o mesmo possa acompanhar a aprendizagem dos alunos e a eficiência das estratégias didáticas aplicadas.

As aplicações que irão se derivar do framework representam um Portfólio Eletrônico[2], acoplado a um Sistema Tutor[3].

A funcionalidade do portfólio eletrônico está em registrar os trabalhos desenvolvidos pelos alunos, registrar suas participações e os resultados obtidos através das interações com o sistema. E a partir da análise destes dados, o professor poderá avaliar os alunos de forma individual e/ou em grupo, avaliar também as estratégias didáticas utilizadas nas sessões de ensino. Uma outra funcionalidade do portfólio bastante útil é um comportamento pró-ativo, que permite estabelecer prazos para realizações de atividades, gera-se avisos sobre atividades a expirar ou atrasos aos professores e alunos e também avisa-se quando uma sessão de ensino será realizada.

O tutor tem como objetivo personalizar o ensino segundo o ritmo de aprendizagem dos alunos através das sessões de ensino utilizando estratégias didáticas.

1.3 Objetivos da Dissertação

O objetivo geral desta dissertação é a utilização de reuso de software no desenvolvimento de sistemas Portfólio-Tutor e apresentar um Framework para sistemas Portfólio-Tutor baseado em Agentes denominado FA_PorT. Alguns objetivos específicos que podem ser destacados para alcançar esta tarefa são os seguintes:

- Permitir ao desenvolvedor de aplicações criar aplicações Portfólio-Tutor a partir do uso de componentes no framework;
- Disponibilizar uma interface personalizada para cada categoria de usuários (aluno, professor e administrador) na internet;
- Auxiliar o professor em suas tarefas de ensino e poder interagir com o aluno de forma síncrona e assíncrona;

[2]Registra e acompanha as atividades dos alunos computacionalmente.

[3]Controla as atividades, a realização de sessões de ensino on-line e artefatos associados ao Portfólio-Tutor.

- Possibilitar aos usuários (professor e administrador) fazerem consultas e atualizações no sistema (atualizar estratégias didáticas, cadastros de sessões de ensino e cadastros de usuários);
- Enviar automaticamente mensagens aos professores e alunos no que diz respeito ao início de sessões de ensino, atividades com prazo a vencer, dentre outras, através do comportamento pró-ativo.

1.4 Organização da Dissertação

Esta dissertação está organizada em 8 capítulos sendo este, o primeiro. Os demais estão organizados como segue abaixo:

- No **Capítulo 2 (Contextualização do Trabalho)** apresentam-se conceitos sobre Educação a Distância, os trabalhos correlatos e a necessidade do reuso de software;
- No **Capítulo 3 (Reuso)** apresenta-se componentes de software e também são mostrados alguns padrões de projeto. E, por fim, apresenta-se frameworks;
- No **Capítulo 4 (Portfólio Eletrônico)** discute-se inicialmente sobre a definição e, logo depois, sobre sua utilização. Fala-se ainda, de alguns sistemas portfólio eletrônicos existentes, tipo de portfólio eletrônico e de um framework para sistemas portfólio eletrônico;
- No **Capítulo 5 (Sistema Tutor Inteligente)** apresenta-se a definição e a arquitetura básica (Domínio, Comportamento, Estratégias, Interface e Controle) de um Sistema Tutor Inteligente;
- No **Capítulo 6 (Agentes)** mostra-se inicialmente a definição, depois temos as propriedades e classificações. Por fim, fala-se sobre a modelagem orientada a agentes com a metodologia GAIA;
- No **Capítulo 7 (Especificação do FA_PorT)** mostra-se o processo de desenvolvimento do framework proposto, o detalhamento de sua arquitetura e o controle. Discute-se ainda, sobre a construção de aplicações a partir do uso do FA_PorT;
- No **Capítulo 8 (Implementação do FA_PorT)** tem-se os principais aspectos da implementação do FA_PorT, a tecnologia empregada, as ferramentas utilizadas, a distribuição física dos módulos do framework e a interface disponibilizada. Posteriormente, temos dois estudos de caso construídos a partir do uso do FA_PorT;
- No **Capítulo 9 (Considerações Finais)** apresentam-se os resultados obtidos nesta dissertação, as contribuições em relação aos sistemas portfólios eletrônicos existentes

e as perspectivas para futuras pesquisas que podem ser realizadas a partir dos resultados obtidos.

Capítulo 2

Contextualização do Trabalho

No capítulo 2, é descrito a contextualização do trabalho. Na seção 2.1, fala-se sobre a educação a distância, na seção 2.2, mostra-se os trabalhos correlatos que foram utilizados, a seção 2.3 apresenta os sistemas Portfólio-Tutor no FA_PorT. E, na seção 2.4, descreve-se a necessidade do reuso no desenvolvimento de aplicações Portfólio-Tutor.

2.1 Educação a Distância

2.1.1 Conceitos e Fundamentos da EAD

"Educação a Distância é uma forma sistematicamente organizada de auto-estudo, onde o aluno se instrui a partir do material de estudo que lhe é apresentado, onde o acompanhamento e a supervisão do sucesso do estudante são levados a cabo por um grupo de professores. Isto é possível de ser feito a distância através da aplicação de meios de comunicação capazes de vencer longas distâncias. O oposto de "educação a distância" é a "educação direta" ou "educação face-a-face": um tipo de educação que tem lugar com o contato direto entre professores e estudantes." (Nunes 2000).

"Educação a Distância é um sistema tecnológico de comunicação bidirecional (e massivo), utilizado como estratégia preferencial de ensino, substituindo a interação professor-aluno em sala de aula, pela ação sistemática e conjunta de recursos didáticos e apoio de uma organização tutorial, propiciando a aprendizagem autônoma do estudante." (Neto 1998).

"Educação a Distância é uma forma de organização de ensino aprendizagem, na qual alunos de diversas idades e antecedentes estudam, quer em grupos, quer individualmente em seus lares, locais de trabalho ou outros lugares, com materiais auto-instrutivos distribuídos por meios de comunicação, garantida a possibilidade de comunicação com docentes, orientadores/tutores ou monitores." (Neto 1998).

Segundo as definições de Ivônio (Nunes 2000) e Neto (Neto 1998) podemos caracterizar a Educação à Distância por:

- A ausência ou diminuição da presença física entre os alunos e os professores favorece uma dinâmica auto-didata por parte do aluno;
- Não podemos considerar como educação à distância materiais de estudo de forma isolada, portanto um texto impresso ou programa educativo vinculado por rádio ou televisão não a caracterizam isoladamente. O material de estudo deve ser disponibilizado sistematicamente através de um meio de comunicação específico e preparado por uma equipe multidisciplinar (Campos 2000);
- A EAD possibilita o atendimento de alunos geograficamente diferentes, através de meios de interação entre eles e professores, promovendo assim o acompanhamento e a monitoração dos alunos.

A EAD se apresenta como outra opção metodológica para a educação. Portanto, as instituições educacionais que decidirem realizá-la terão que reorganizar seus procedimentos desde a matrícula até as forma de avaliação que serão utilizadas. Para tornar isso possível, é preciso um compromisso pedagógico vinculado ao contexto histórico, político e cultural tanto pela EAD quanto pela instituição que está propondo essa forma de ensino (Nascimento 2002).

A Educação à Distância colabora para a educação formal e informal utilizando diversos meios de comunicação como rádio, televisão, fita cassete, vídeo cassete, CD-ROM, além das mídias disponíveis para interação como correspondência, telefone e fax. A inclusão da EAD na capacitação profissional favorece a educação continuada, enriquecendo os seus conhecimentos e as suas competências (Nascimento 2002).

Entretanto, o uso da internet alastrou-se pela comunidade, atingindo os mais variados níveis sócio-econômicos da sociedade. Devido a sua facilidade de acesso e utilização, liberdade de expressão, diversidade de recursos e serviços oferecidos, também exerce grande influência na área educacional (Nascimento 2002).

2.1.2 Educação a Distância no Contexto Internet

A web permite a disponibilização de informações com a utilização de vários recursos que auxiliam a capacidade cognitiva dos alunos como textos, imagens, áudios, animações, vídeos, além do acesso a banco de dados interativos. A navegação e o acesso a estes recursos ocorrem intuitivamente, de forma não linear, por links de conexão, já que o conteúdo é disponibilizado no formato de hiperdocumentos (Nascimento 2002).

A internet possibilita as comunicações síncrona (os chats e as vídeo conferências) e assíncrona (o correio eletrônico e os grupos de discussões). A empregabilidade do chat na EAD promove a participação informal, a motivação e a expressão de idéias ou opiniões dos indivíduos, já que permite a comunicação instantânea e textual com uma ou um conjunto de pessoas. A videoconferência aumenta o grau de contato e, portanto, o censo de comunidade, já que as pessoas podem falar e se ver ao mesmo tempo. É a forma de comunicação que mais se aproxima da presencial, contudo ainda tem restrições na relação custo/benefício relacionadas às velocidades dos meios de comunicação. O whiteboard ainda é um outro recurso síncrono, pois se trata de uma janela cujo conteúdo é visível e editado coletivamente. Pode contribuir significativamente para o ensino na medida em que pode ser utilizada em paralelo ao chat ou à videoconferência, sendo possível a visualização de uma seqüência de slides ou a resolução de exercícios (Nascimento 2002).

Em relação aos recursos assíncronos, apresentam a vantagem de poderem contribuir ou questionar a respeito de um assunto independentemente de outros participantes estarem conectados. O correio eletrônico propicia respostas bem mais rápidas que uma correspondência comum, estimulando assim consultas extra sala de aula ou horário de trabalho, permitindo que cada um mantenha o seu ritmo de aprendizado. As conferências textuais mais formais e direcionadas a um tema específico podem ocorrer nos grupos de discussões. Nestes, os alunos podem expor suas idéias e contestá-las, tomando um posicionamento ativo, o que torna interessante a presença de um moderador. A troca de experiências também pode dar-se através da troca de arquivos, sejam eles anexados aos correios eletrônicos, disponibilizados para download em páginas Web ou pelo uso do protocolo de FTP (File Transfer Protocol) (Nascimento 2002).

Desafios a serem enfrentados

Para que os objetivos educacionais da EAD sejam alcançados, é necessário que seja imposta uma mudança cultural (Nascimento 2002). Novos papéis, assim, seriam assumidos. Os alunos, nesse contexto, são ativos, participativos e responsáveis pela construção do seu conhecimento, transformando as informações, analisando-as e pesquisando-as. Os professores seriam facilitadores, orientadores e incentivadores do processo de aprendizagem.

O envolvimento de uma equipe multidisciplinar se faz necessário para a elaboração de um curso, pois é interessante servir-se de todos os recursos disponibilizados, e não simplesmente transferir uma aula do método tradicional de ensino para a EAD. O tempo de dedicação do professor será empregado no planejamento, elaboração e atualização dos materiais a serem disponibilizados, no atendimento ao aluno, no seu acompanhamento e avaliação, como nos trabalhos produzidos pelo mesmo (Nascimento 2002).

As maiores dificuldades encontradas pelo ensino a Distância são a ausência da per-

cepção pelo professor em relação à compreensão do aluno sobre determinado assunto e a falta de conhecimento prático do contexto social, na qual este tipo de estudante está inserido (Nascimento 2002).

2.2 Histórico dos Trabalhos Correlatos em termos de Pesquisa

2.2.1 POETA

O POETA (Portfólio Eletrônico Temporal e Ativo) (Sistêlos 1999) é um ambiente que favorece a dinâmica de uma avaliação autêntica e faz uso do conceito de Portfólio Eletrônico, utiliza uma Metodologia para Projetos de Banco de Dados Temporais Orientados a Objetos, o FADO (Ferramenta de Análise e Desenvolvimento Orientados a Objetos) (Furtado 1993).

Os elementos envolvidos no aprendizado do aluno excedem a sala de aula (aluno/professor) e miscigenam-se com as outras partes integrantes de seu desenvolvimento (família e instituição). A interação aluno e professor se dará através do seu contínuo relacionamento com o sistema, onde estes trabalharam em conjunto: os professores elaboram tarefas, os alunos as executam, os professores as julgam, os alunos acessam estes julgamentos e observações e procedem as devidas correções quando permitido. Ambos, professor e aluno, têm a opção de selecionar este documento para o Portfolio de Apresentação do aluno.

A família do aluno terá a possibilidade de, a qualquer tempo, visualizar o progresso do aluno e seu desempenho em sala de aula através dos trabalhos por ele realizados e armazenados pelo sistema. Além disso, a Instituição poderá perceber quando for implantada alguma alteração em sua estrutura, seja esta em um conteúdo programático ou em uma metodologia de ensino.

Estes indicadores de desempenho e de perfis serão fornecidos automaticamente pelo sistema e comunicados às entidades aluno, professor, família e instituição através de mensagens.

2.2.2 TUTA

O TUTA (Tutor baseado em Agentes no contexto do Ensino a Distância) (Silva 2000), que utiliza os princípios e módulos de um Sistema Tutor Inteligente (Módulo Especialista, Módulo do Comportamento do Aluno, Módulo de Estratégias de Ensino, Módulo de Interface e Módulo de Controle), 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 para um grupo de alunos geograficamente

distantes. Esse tutor está inserido no contexto da ACVA (Arquitetura de uma Classe Virtual Adaptativa) (Hernández-Domínguez 1997). 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. Estes são tratados como zonas de conhecimento. Assim, o objetivo da 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 sessões de ensino adaptáveis.

O principal propósito da ACVA é a construção de um ambiente de ensino/aprendizagem, com enfoque na criação de um ambiente colaborativo de aprendizagem e no ensino personalizado.

2.2.3 Portfólio-Tutor

O Portfólio-Tutor (Nascimento 2002) é um sistema tutor acoplado a um portfólio eletrônico no contexto da Educação a Distância, que visa apoiar as atividades do professor, auxiliando-o no processo de ensino/aprendizagem e fornecendo mecanismos para que o mesmo possa acompanhar a aprendizagem dos alunos e a eficiência das estratégias didáticas aplicadas.

O Portfólio-Tutor também faz parte da ACVA. Com o objetivo de personalizar o ensino, a funcionalidade do Tutor (Tutor-ACVA), deverá capturar o grupo e zona de comportamento do aluno e, assim, aplicar a estratégia didática correspondente. Dada a característica adaptativa desta arquitetura. o Tutor-ACVA também deverá controlar a mobilidade dos alunos dentro dos grupos e zonas de comportamento.

A funcionalidade do portfólio será baseada no POETA (Sistêlos 1999), mas o ambiente de aplicação é a internet. Desta forma, o portfólio eletrônico servirá para arquivar os trabalhos desenvolvidos pelos alunos, registrar suas participações e os resultados obtidos através de suas interações com o tutor. Estes registros servirão para análise do progresso do aluno, em relação à assimilação do conteúdo e outras habilidades adquiridas.

Outra funcionalidade do portfólio é a definição de atividades e, conseqüentemente, prazos para a realização das mesmas, onde avisos automáticos serão gerados sobre prazos a expirar ou atrasos, a professores e alunos.

2.2.4 Análise dos Trabalhos Correlatos

Após análise dos trabalhos correlatos (POETA, TUTA e Portfólio-Tutor), chegou-se a conclusão que suas principais contribuições para este trabalho podem ser vistas na Tabela 2.1.

Sistema	**Contribuição Principal**
POETA	Representação de um sistema Portfólio Eletrônico na camada Portfólio.
TUTA	Representação de um sistema Tutor baseado em Agentes através da camada Tutor.
Portfólio-Tutor	Acoplamento da camada Portfólio Eletrônico a camada Tutor em um sistema Portfólio-Tutor.

Tabela 2.1: Análise dos Trabalhos Correlatos

2.3 Sistemas Portfólio-Tutor no FA_PorT

O sistema Portfólio-Tutor (Nascimento 2002), como foi visto na seção 2.2.3, possui duas camadas, a camada Portfólio Eletrônico e a camada Tutor.

O sistema Portfólio-Tutor criado a partir do framework proposto (FA_PorT), possui, além das camadas propostas por Nascimento (Nascimento 2002), e, de acordo com o TUTA (Silva 2000), uma camada baseada em agentes. Estes possui o objetivo de dar maior autonomia e flexibilidade ao sistema Portfólio-tutor.

Sendo assim, a arquitetura[1] de um sistema portfólio-tutor, que será gerado a partir do uso do framework FA_PorT, será formada por cinco camadas (Figura 2.1), a camada de persistência (Sistema de Gerenciador de Banco de Dados), três camadas de negócio (Agentes, Tutor e Portfólio Eletrônico) e a camada de apresentação (Interface). A camada agentes será responsável pela flexibilidade e autonomia; a camada portfólio eletrônico tratará do registro e acompanhamento das atividades; enquanto que a camada Tutor, trabalhará com a sessão de ensino propriamente dita.

[1]Esta arquitetura encontra-se especificada no capítulo 7.

Figura 2.1: Camadas de um sistema Portfólio-Tutor

2.4 Necessidade do Reuso no Desenvolvimento de Aplicações Portfólio-Tutor

Percebemos a necessidade do reuso no desenvolvimento de aplicações Portfólio-Tutor devido que a simples adoção da tecnologia de objetos, sem a existência de um padrão de reuso explícito e de um processo de desenvolvimento de software orientado ao reuso não fornece o sucesso esperado para produção de software em larga escala (Jacobson et al. 1997). Uma das técnicas de reuso de software para um determinado domínio de problema é a de implementar as funcionalidades comuns em frameworks (Johnson & Foote 2001).

O FA_PorT apresenta uma arquitetura de software que permite acoplar classes ou componentes a um conjunto de componentes existentes. E, com este acoplamento, tem-se a construção de uma aplicação Portfólio-Tutor.

Para o desenvolvimento de aplicações Portfólio-Tutor é utilizado componentes de software como objetos de reuso.

2.5 Conclusões

Neste capítulo, apresentou-se a contextualização do trabalho, bem como, conceitos e fundamentos da educação a distância, histórico dos trabalhos correlatos em termos de pesquisa, a utilização de sistemas Portfólio-Tutor no framework FA_PorT e, por fim, a necessidade do reuso no desenvolvimento de aplicações Portfólio-Tutor.

É iminente a intensa utilização da internet para fins de educação a distância, através da gama de novas possibilidades que a rede traz para a EAD.

Este trabalho se propõe apresentar o desenvolvimento de um framework e aplicações no contexto de EAD através da internet, resultando no desenvolvimento de dois protótipos.

Para a criação do framework foram analisados vários trabalhos correlatos e esse framework foi usado para o desenvolvimento das aplicações derivadas do framework.

Os próximos capítulos apresentam a fundamentação teórica deste trabalho.

Parte I

Fundamentação Teórica

Capítulo 3

Reuso

Neste capítulo serão descritos os principais conceitos sobre reuso de software utilizados no desenvolvimento do framework proposto nesta dissertação. Na seção 3.1 é descrito a definição de componentes de software e de interface. Na seção 3.2 descrevemos o conceito de padrão de projeto. Definições, classificações, funcionamento e tipos de framework existentes observaremos na seção 3.3.

3.1 Desenvolvimento de Software Baseado em Componentes

O DBC (Desenvolvimento de Software Baseado em Componentes) é definido como o desenvolvimento de sistemas de software através da integração planejada de componentes de software reutilizáveis (Brown & Wallnau 1998). Embora o paradigma DBC só tenha se tornado realidade recentemente, a idéia de componentes de software surgiu ainda no final da década de 60. Em outubro de 1968, foi proposto a criação de uma indústria de componentes de software. Há uma grande semelhança com a indústria de componentes de hardware, que era para servir de base para a produção de sistemas de software em grande escala.

3.1.1 Definição de Componentes de Software

De acordo com Szyperski (Szyperski 1998), um componente de software é uma unidade de composição com interfaces especificadas contratualmente e somente dependências explícitas de contexto. Uma interface identifica um ponto de interação entre um componente e o seu ambiente. Esse ambiente pode ser constituído por outros componentes de software e partes deste não baseadas em componentes. As interfaces de um componente podem ser de dois tipos: (a) uma interface provida de um componente, que identifica um ponto de acesso a serviços onde o componente é capaz de prover para o ambiente, e, (b) uma

interface requerida de um componente, que identifica um ponto de acesso a serviços que o componente requer do ambiente. Na Figura 3.1 é representado, usando a notação UML, um componente de software com três interfaces (I1, I2 e I3), uma fornecida (I1) e duas requeridas (I2 e I3).

Figura 3.1: Representação UML de um componente de software

As dependências de um componente são especificadas através de um modelo de componentes. Um modelo de componentes especifica os padrões e convenções impostas aos desenvolvedores dos componentes e os pressupostos básicos que podem ser assumidos a respeito do ambiente. A forma de comunicação entre componentes de software e os protocolos correspondentes são exemplos de padrões normalmente especificados no modelo de componentes.

Um modelo de componentes pode especificar também uma infra-estrutura de componentes a ser integrada ao ambiente dos sistemas. Esta infra-estrutura oferece serviços básicos para todos os componentes de software, tais como serviços de persistência e segurança de acesso.

Um aspecto fundamental do paradigma DBC é a separação entre especificação de um componente de software e sua implementação. A especificação define o comportamento observável externamente ao componente, com a abrangência e precisão necessárias para sua integração em diferentes sistemas, porém abstraindo detalhes de qualquer implementação específica. A implementação fornece um modelo concreto para instanciação em diferentes ambientes de componentes que realizam uma dada especificação.

A implementação de um componente de software requer um projeto interno que pode, recursivamente, ser também baseado em componentes. A implementação de componente pode, ao mesmo tempo, ser composta por subcomponentes menores e ser parte de um componente maior. Em seu nível mais concreto requer a utilização de uma linguagem de programação. A escolha de uma linguagem específica está sujeita apenas às restrições que, porventura, seriam impostas pela infra-estrutura de componentes adotada. Quando utilizada uma linguagem orientada a objetos, as operações especificadas nas interfaces providas pelo componente são implementadas através de uma ou mais classes de implementação.

Uma implementação de componente é, portanto, um componente de software concreto

composto por vários artefatos de software, tais como: (i) o projeto interno do componente; (ii) o código fonte da implementação, em uma linguagem de programação específica e (iii) o módulo executável correspondente no código binário especificado pela infra-estrutura de componentes adotada. Esses artefatos compõem o pacote de implementação do componente.

A instanciação do componente ocorre apenas no momento da execução de um sistema onde o componente está integrado. Quando a implementação do componente utiliza uma linguagem orientada a objetos, essa instanciação resulta na criação de um ou mais objetos das suas classes de implementação.

3.1.2 Especificação de Interface

As linguagens de definição de interfaces existentes permitem expressarmos apenas a sintaxe e, em alguns casos, a semântica das interfaces sob forma de assinaturas de métodos, pré-condições, pós-condições e invariantes. A especificação dos aspectos de sincronismo e qualidade de serviço, por sua vez, ainda é objeto de pesquisa (Bachmann 2000).

Essa lacuna na definição das interfaces é de vital importância para o desenvolvimento de sistemas confiáveis, pois os atributos de confiabilidade de um sistema serão sempre função da qualidade dos componentes utilizados. Sem informações sobre o comportamento de um componente em situações de falha e sobre a qualidade do seu projeto, é impossível oferecer qualquer garantia quanto ao comportamento de um sistema que dependa daquele componente.

Implementação das Interfaces

A implementação de uma interface possui propriedades que nem sempre fazem parte da sua definição. A memória utilizada e o tempo de resposta são exemplos de propriedades inerentes a qualquer implementação e que, muitas vezes, não estão definidas nas interfaces. Uma vez que uma implementação é integrada a um sistema, essas propriedades podem criar dependências não previstas pelas interfaces dos componentes, caracterizando um fenômeno chamado vazamento de propriedade. A simples substituição de uma implementação de uma interface por outra em perfeita conformidade com a mesma, pode, nessas condições, resultar em uma condição de falha para o sistema (Bachmann 2000).

3.2 Padrões de Projeto

O desenvolvimento dos padrões de software teve como inspiração os trabalhos do arquiteto Christopher Alexander, daí o significado atual de "*pattern*" foi utilizado por ele para o planejamento urbano e os projetos de arquitetura em vários livros publicados nas décadas de 60 e 70 (Appleton 2006).

No contexto de desenvolvimento de software orientado a objetos, foi proposto um catálogo de padrões de projeto que foi publicado no livro "*Design Patterns: Elements of Reusable Object-Oriented Software*" (Gamma et al. 1995).

3.2.1 Definição

De acordo com Appleton (Appleton 2006), os padrões de projeto descrevem soluções para problemas que ocorrem repetidamente em diversos sistemas. Estas soluções são descritas de forma que possam ser reutilizadas.

Um padrão, além de apresentar uma solução para um determinado problema, indica o contexto necessário para sua aplicação e explica porque a solução é bem sucedida. Também pode ser visto como uma formalidade utilizada para registrar o conhecimento e uma forma de documentação.

Gamma (Gamma et al. 1995) define padrões de projeto da seguinte maneira:

"Um padrão de projeto abstrai e identifica os aspectos essenciais de uma estrutura de modo que esta possa ser usada para criar um projeto orientado a objetos reutilizável. O padrão de projeto identifica as classes participantes, seus papéis, colaborações e a distribuição de responsabilidades. Cada padrão de projeto está relacionado com um determinado tipo de problema de projeto orientado a objetos. O padrão de projeto descreve quando é possível aplicá-lo, se é possível ou não aplicá-lo em virtude de outras restrições de projeto e os custos e consequências resultantes de sua aplicação."

Um padrão de projeto é descrito pelos seguintes elementos:

- Nome: um identificador que mostra a idéia geral do padrão, o problema a ser resolvido, a solução e a conseqüência em uma única expressão. O uso de nomes bem definidos também ajuda no desenvolvimento sobre padrões.

- Problema: descreve os aspectos essenciais que caracterizam o problema a ser resolvido. Mostra também o contexto em que o padrão pode ser aplicado, incluindo as pré-condições e restrições (como exemplo podemos citar a interação entre dois componentes do sistema, que deve ser implementada utilizando um determinado protocolo de comunicação). O contexto também explica o tempo, o modo e as razões da ocorrência do problema.

- Solução: demonstra como resolver o problema ou como equilibrar todas as restrições e pré-condições especificadas da melhor maneira possível. A solução pode ser composta por duas partes: uma parte descreve uma estrutura estática da solução, que define os componentes participantes (objetos, classes ou outras entidades), suas responsabilidades e relacionamentos, a outra parte descreve uma estrutura dinâmica que consiste na especificação de colaborações e interações entre os participantes da solução.

- Conseqüências: descrevem os resultados e custos da aplicação do padrão. Também podem indicar o impacto do uso do padrão, por exemplo, na estabilidade e extensabilidade do sistema.

Segundo Appleton (Appleton 2006), um modelo para ser considerado um padrão deve ser um fenômeno corrente. Existe uma convenção definida como *"rule of three"*, indicando que um determinado modelo deve ocorrer em pelo menos três sistemas diferentes para ser considerado um padrão de projeto. Outros critérios para se definir um padrão são: um padrão resolve problemas, ou seja, descrevem soluções e não estratégias ou princípios abstratos. Representa um conceito provado na prática, descreve uma solução que não é aparentemente óbvia e explica porque a solução apresentada é útil e funcional.

Outros autores (Yacoub & Ammar 2003) definem um ciclo de vida para um padrão de projeto (Figura 3.2), da seguinte maneira:

- Fase 1: Mineração

  Nesta fase é feita a documentação inicial do padrão. Autores que estão trabalhando em projetos práticos podem vir a descobrir e documentar novos padrões. No caso, o autor deve decidir o que faz parte do padrão e em qual domínio é aplicável. A chamada "*rule of three*" pode ser usada para decidir sobre a validação do padrão de projeto.

- Fase 2: Refinamento

  Aqui o padrão é avaliado por especialistas com o objetivo de ajudar o autor na criação do padrão ou abandoná-lo caso não estejam de acordo. O resultado desta fase é um padrão bem documentado e pronto pra reuso.

- Fase 3: Reuso

  Nesta fase ocorre o reuso propriamente dito. Usuários procuram por padrões em catálogos e livros e eles são utilizados em projetos reais. Os usuários podem enviar sugestões ou reclamações ao autor a respeito de sua experiência naquele padrão, o que contribui para o seu aperfeiçoamento.

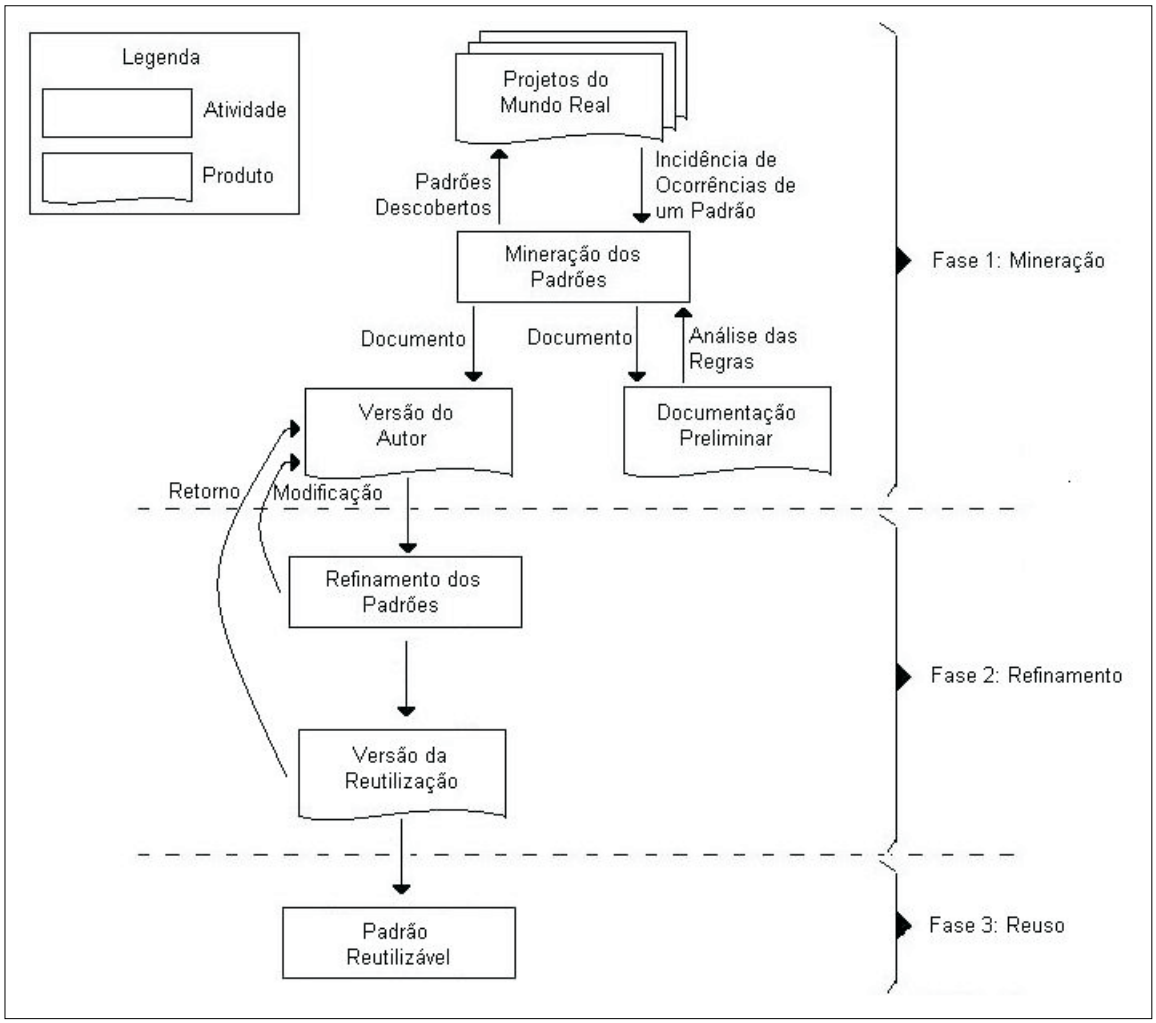


Figura 3.2: Ciclo de vida de um padrão

3.2.2 Padrões de Projeto

A classificação proposta por Gamma (Gamma et al. 1995) organiza seus elementos segundo dois critérios:

- Escopo: especifica se um padrão é aplicável a classes ou instâncias. Um padrão aplicável a classe determina relações entre classes e subclasses, ou seja, é definido através de herança e os padrões aplicáveis a instâncias de classes definem relações entre objetos que podem ser alterados dinamicamente;
- Propósito (função): define as categorias padrões de criação, padrões estruturais e padrões comportamentais, este sendo o critério mais utilizado.

A Tabela 3.1 possui a classificação de padrões proposto por Gamma (Gamma et al. 1995).

		Propósito		
		Criação	**Estruturais**	**Comportamentais**
Escopo	**Classe**	*Factory Method*	*Adapter*	*Interpreter* *Template Method*
	Objeto	*Abstract Factory* *Builder* *Prototype* *Singleton*	*Adapter* *Bridge* *Composite* *Decorator* *Facade* *Proxy*	*Chain of Responsability* *Command* *Iterator* *Mediator* *Memento* *Flyweight* *Observer* *State* *Strategy* *Visitor*

Tabela 3.1: Classificação de Padrões segundo Gamma

No contexto do desenvolvimento do framework FA_PorT foram utilizados os padrões de projeto *Singleton*, *Facade* e *Template Method*. A seguir, estes padrões são descritos.

Padrão Singleton

- Objetivo

Garantir que uma classe tenha somente uma única instância e fornecer um ponto global de acesso para tal instância.

- Contexto

Algumas classes devem possuir exatamente uma instância. Tais classes geralmente estão envolvidas no gerenciamento de algum recurso ou controlando alguma atividade (*controller*). O recurso pode ser externo, como no caso em que um objeto gerencia a reutilização de uma conexão com um gerenciador de banco de dados. O recurso também pode ser interno, como no caso em que um objeto mantém estatísticas de erro para um compilador.

- Estrutura

O padrão Singleton é relativamente simples, uma vez que ele envolve uma única classe como mostra a figura 3.3.

A classe unitária possui uma variável estática que mantém uma referência para a única instância que se deseja manipular. Esta é criada quando a classe é carregada na memória ou quando ocorre a primeira tentativa de acesso à instanciação. Devemos implementar a classe unitária de tal modo, que não seja possível criar instâncias adicionais à única

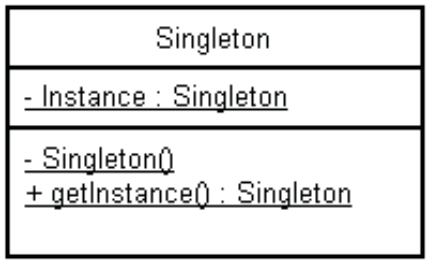


Figura 3.3: Estrutura do Padrão Singleton

instância permitida. Isto significa que devemos assegurar que todos os construtores da classe unitária sejam declarados como privados.

- Aplicabilidade

Devemos utilizar este padrão quando houver uma única instância de uma classe. Essa instância deve poder ser acessada pelos clientes a partir de um ponto já conhecido.

Padrão Facade

- Objetivo

O padrão Facade simplifica o acesso a um conjunto de objetos relacionados utilizando um outro objeto que é responsável pela comunicação entre os objetos externos ao conjunto e o conjunto de objetos em questão.

- Contexto

Organizar um sistema em subsistemas, ajudando-o a reduzir a sua complexidade. Um objetivo comum a todos os projetos é diminuir a comunicação e as dependências entre os subsistemas que fazem parte de uma aplicação. Uma forma de alcançar este objetivo é inserindo um objeto facade (fachada), pois ele nos dá uma interface única para os recursos de um subsistema.

- Estrutura

A Figura 3.4 mostra um diagrama de classes que ilustra a estrutura geral do padrão Facade. O objeto Cliente interage com o objeto Facade que fornece a funcionalidade necessária para a interação com o restante dos objetos. Se existir alguma funcionalidade adicional, necessária apenas para alguns clientes, será melhor então que o objeto Facade forneça um método em separado, para que seja possível acessar diretamente o objeto responsável por tal funcionalidade, em vez de incluí-la na interface principal.

De acordo com a Figura 3.4 temos que a classe Cliente precisa de uma funcionalidade de um dado subsistema, mas não precisa estar ciente da complexidade que envolve a sua

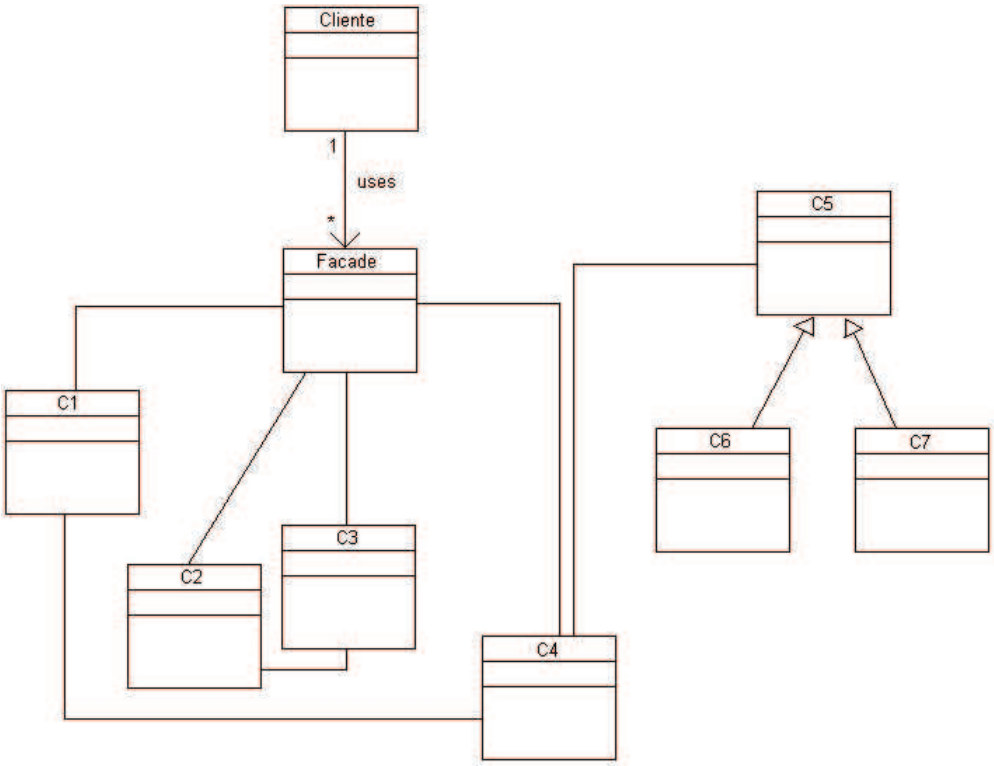


Figura 3.4: Estrutura do Padrão Facade

execução. Facade conhece as classes do subsistema que são responsáveis pelo atendimento de uma solicitação. Em função disso, delega as solicitações dos clientes aos objetos apropriados do subsistema. Este encarrega-se do trabalho que lhes foi atribuído pelo objeto Facade. Como regra geral, as classes do subsistema não devem manter referências para o objeto Facade.

- Aplicabilidade

É preciso utilizar o padrão Facade quando precisa-se utilizar uma interface simples para um subsistema muito grande, quando existirem muitas dependências entre os clientes e as classes que implementam um subsistema. Também quando se deseja estruturar os subsistemas de um sistema em camadas, para cada subsistema usa-se uma fachada para definir o ponto de acesso para cada subsistema.

Padrão Template Method

- Objetivo

O padrão Template Method (Método Template) possui o objetivo de definir o esqueleto de um algoritmo em uma operação, postergando alguns passos para subclasses. O Método Template permite que subclasses criem novos passos em um algoritmo, sem que mude a estrutura do mesmo.

- Contexto

Pode ser utilizado para a construção de frameworks, considerando um framework de aplicações com a definição de alguns passos de um algoritmo usando operações abstratas, o método template define a ordem, mas permite a subclasses variarem alguns passos do algoritmo.

- Estrutura

A Figura 3.5 mostra um diagrama de classes que ilustra a estrutura geral do padrão Método Template.

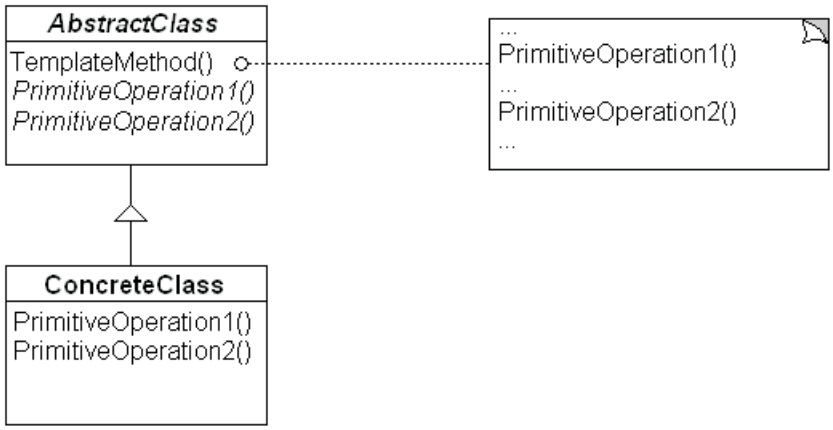


Figura 3.5: Estrutura do Padrão Método Template

Na figura 3.5 temos a AbstractClass que é uma classe abstrata e a ConcreteClass que é uma classe concreta. Também é mostrado um esqueleto de aplicações e o método template que chama as operações primitivas. A classe concreta implementa todas as operações primitivas.

- Aplicabilidade

O Método Template pode ser usado para implementar as partes invariantes de um algoritmo pré-determinado uma única vez e, deixar as subclasses com a implementação do comportamento que pode variar. Controla também as extensões de subclasses através de pontos de adaptação de código em locais específicos, permitindo assim, extensões somente nesses pontos de adaptação.

3.3 Framework

Os frameworks começaram a ser construídos no início da década de 80. Além de minimizar os esforços de desenvolvimento das aplicações, eles possibilitam a reutilização

não só do código, mas também de toda a análise e projeto de um framework. Utiliza-se a definição da arquitetura das aplicações geradas a partir dele, por ter pré-definido (embutido) o fluxo de controle destas aplicações.

3.3.1 Definição

Um framework se define como um comportamento de uma coleção de objetos fornecendo um meio inovador para reutilizar a fase de projeto e codificação. Programadores podem adequar às necessidades do cliente para soluções específicas (Fayad & Schmidt 1997).

Framework é um projeto reutilizável de todo ou parte de um sistema que podem ser representados por um conjunto de classes abstratas e concretas, isto é, descreve a estrutura de um framework. Este, pode ser entendido como um esqueleto de aplicações que pode ser customizado por um desenvolvedor de aplicações. Detalhes específicos de uma aplicação são implementados com o acréscimo de componentes e o fornecimento da implementação de métodos abstratos das classes abstratas (Fayad & Schmidt 1997).

O framework descreve componentes e objetos e como eles interagem, além de mostrar como as responsabilidades dos sistemas são mapeados sobre esses objetos.

Tipicamente, um framework é implementado com uma linguagem orientado a objetos.

3.3.2 Benefícios

Segundo Fayad (Fayad et al. 1999), os benefícios primários da aplicação de frameworks em orientação a objetos são:

- **Modularidade**: os frameworks aumentam a modularidade por encapsular em detalhes a implementação por trás das interfaces. A modularidade dos frameworks facilita a localização de mudanças no projeto e impactos na implementação. Esta localização reduz os esforços requeridos para o entendimento e manutenção do software;

- **Reusabilidade**: com a reusabilidade é possível criar novas aplicações através de componentes genéricos, garantindo maior produtividade na programação, garantia de qualidade, performance e confiabilidade do software;

- **Extensibilidade**: um framework aumenta a extensibilidade, pelos explícitos pontos de adaptação de código, permitindo assim que aplicações possam ser customizadas (personalizadas) pelo desenvolvedor de aplicações;

3.3.3 Classificação

Os frameworks podem ser classificados de acordo com seu escopo, ou seja, a abrangência de sua aplicação em um determinado domínio e com a forma de reuso do framework

para gerar novas aplicações.

Fayad, Schmidt (Fayad & Schmidt 1997) Somerville (Somerville 2003) identificaram três classes de frameworks: framework de infraestrutura de sistema, framework de integração de middleware e framework de aplicação corporativos.

1. Framework de infra-estrutura de sistema: são compatíveis com o desenvolvimento das infra-estruturas de sistemas, como interface com o usuário e compiladores;

2. Framework de integração de mmiddleware: é um conjunto de classes de objetos-padrão e associadas, que permitem a comunicação de componentes e a troca de informações;

3. Framework de aplicações corporativos: se ocupam de domínios específicos de aplicações, como sistemas financeiros ou telecomunicações. Possuem o conhecimento de domínio de aplicações e são compatíveis com o desenvolvimento de aplicações.

Os frameworks de aplicação orientado a objetos é uma tecnologia promissora devido à redução de custos no desenvolvimento de aplicações e ao aumento na qualidade do sistema.

3.3.4 Funcionamento

Inversão de Controle

Uma das principais características do framework é a inversão de controle. Os métodos definidos pelo usuário para especializá-lo são chamados dentro do próprio framework, em vez de serem chamados a partir do código de aplicação do usuário (Johnson & Foote 2001). Em um framework, o programa principal é reutilizável e os métodos fornecidos pelo usuário, desenvolvedores de aplicações geradas a partir do framework, especializam os algoritmos genéricos definidos no framework para uma nova aplicação.

O framework (através do controle) geralmente faz o papel do programa principal, coordenando e sequenciando as atividades da aplicação.

Especificação dos Pontos de Adaptação de Código (*Hot spots*)

Um aspecto variável de uma aplicação é chamado de "ponto de especialização" (em inglês, "*Hot spot*"). Estes são os pontos onde o framework irá ser estendido para o desenvolvimento das aplicações. Diferentes aplicações dentro de um mesmo domínio são diferenciadas por um ou mais *hot spots*. Eles representam as partes de um framework que são específicas de cada sistema. Os *hot spots* são projetados para serem genéricos, podendo assim, serem adaptados às necessidades de cada aplicação.

Hot Spot faz com que o framework seja flexível.

Frameworks Caixa Branca e Caixa Preta

Existem dois tipos de frameworks: caixa branca e caixa preta (Johnson & Foote 2001). No framework caixa branca, o reuso é provido por herança, ou seja, o usuário deve criar subclasses das classes abstratas contidas no framework para criar aplicações específicas. Para tal, ele deve entender em detalhes como o framework funciona para poder usá-lo. Já no framework caixa preta o reuso é por composição, isto é, o usuário combina diversas classes concretas existentes no framework para obter a aplicação desejada. Assim, ele deve entender apenas a interface para poder usá-lo.

A Figura 3.6 ilustra um framework caixa branca com um único *hot spot* R. Para utilização do framework é necessário fornecer a implementação referente a este hot spot, que no caso da Figura 3.6 é R3.

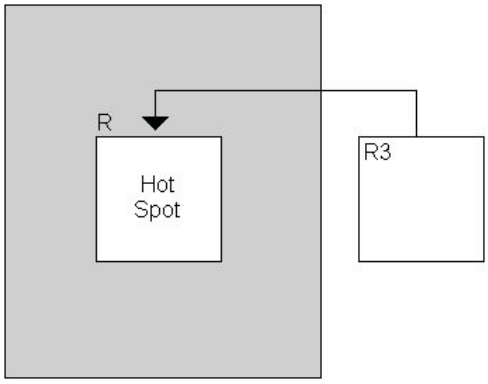


Figura 3.6: Framework Caixa Branca ("*White-Box*")

A Figura 3.7 ilustra um framework caixa-preta, também com um único *hot spot* R. Neste framework existem três alternativas (R1, R2 e R3) para implementação da responsabilidade R. O usuário deve escolher uma delas para obter sua aplicação específica. Note que R1, R2 e R3 fazem parte do framework e são as únicas alternativas possíveis de implementação do *hot spot*. Já no caso da Figura 3.6, R3 não fazia parte do framework, mas em compensação, qualquer alternativa de implementação do *hot spot* seria possível.

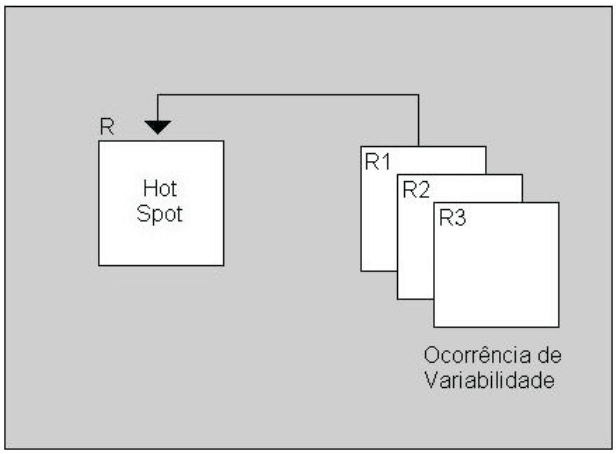


Figura 3.7: Framework Caixa Preta ("*Black-Box*")

Resumindo, um framework caixa branca é mais fácil de projetar, pois não há necessidade de prever todas as alternativas de implementação possíveis. Já o framework caixa preta é mais difícil de ser projetado, por haver a necessidade de fazer essa previsão. Por outro lado, o framework caixa preta é mais fácil de usar, porque basta escolher a implementação desejada; enquanto no caixa branca é necessário fazer a implementação completa.

Os frameworks caixa branca podem evoluir para se tornar cada vez mais frameworks de caixa preta (Johnson & Foote 2001). Isto pode ser conseguido de forma gradativa, implementando-se várias alternativas que depois são aproveitadas na instanciação do framework. Ao mesmo tempo, não se fecha totalmente o framework, permitindo ao usuário continuar usando-o como caixa branca. Após um certo tempo, estarão disponíveis diversas alternativas e então pode-se decidir tornar o framework caixa preta. À medida em que o framework vai se tornando mais caixa preta, diminui o número de objetos criados, embora aumente a complexidade das suas interconexões.

3.4 Conclusões

Neste capítulo, apresentou-se o reuso de software: o desenvolvimento de software baseado em componentes e aspectos que devem ser considerados, como, definição e especificação; padrões de projeto, também com suas definições e alguns exemplos; e, finalizando, frameworks, mostrando sua definição, classificação e funcionamento.

No contexto da Engenharia de Software, diferentes abordagens buscam melhorar a qualidade dos artefatos de software, bem como diminuir o tempo e o esforço necessários para produzi-los. Frameworks são estruturas de classes que constituem implementações incompletas que, estendidas, permitem produzir diferentes artefatos de software. A grande vantagem desta abordagem é a promoção de reuso de código e projeto, que pode diminuir

o tempo e o esforço exigidos na produção de software. Em contrapartida, é complexo desenvolver frameworks, bem como aprender a usá-los.

A abordagem de frameworks pode se valer de padrões para a obtenção de estruturas de classes bem organizadas e mais aptas a modificações e extensões.

O desenvolvimento orientado a componentes pretende organizar a estrutura de um software como uma interligação de artefatos de software independentes, os componentes.

O reuso de componentes previamente desenvolvidos, no contexto de frameworks, permite a redução de tempo e esforço para a obtenção de um software.

O presente capítulo se preocupa em tratar as abordagens, que têm em comum a reutilização, seja de projeto ou de código, isto é, o aproveitamento de experiência de desenvolvimento de software.

No próximo capítulo, apresenta-se o portfólio eletrônico.

Capítulo 4

Portfólio Eletrônico

Neste capítulo, serão descritos os conceitos de portfólio eletrônico. Este conhecimento é útil no entendimento da camada de portfólio que será utilizada nas aplicações Portfólio-Tutor. A seção 4.1 apresenta a definição de portfólio eletrônico. Na seção 4.2, é descrita a utilização de portfólios eletrônicos no contexto da avaliação da aprendizagem. A seção 4.3 descreve os tipos de portfólio eletrônico. E, na seção 4.4, descreve-se os sistemas portfólios eletrônicos existentes.

O primeiro país que começou a usar Portfólio foi os EUA na década de 80 como meio para avaliar alunos e programas pedagógicos de várias escolas públicas (Mokhtari & Yellin 1996).

Existem várias razões para se utilizar portfólios, algumas são:

- Pode ser utilizada como uma ferramenta de ensino, para que se possa fornecer aos estudantes segurança, motivação, participação e envolvimento dos alunos no processo de auto-avaliação, além de ajudar professores e alunos a estabelecerem objetivos, dentre outros;
- Ajudar no desenvolvimento profissional do professor, a partir do estudo do currículo e das práticas de ensino;
- Permitir boa comunicação entre grupos e identificar pontos fortes e fracos nas escolas;
- Para o método de avaliação, utilizando testes padronizados;
- Professores podem pesquisar sobre o progresso de um aluno ou de um grupo de alunos e, desta maneira, realizar um acompanhamento mais preciso.

4.1 Definição

O portfólio utiliza o conceito de pasta de documentos e arquivos. Na Figura 4.1, temos um portfólio tradicional, onde, os documentos que estão nas pastas com identificação, são chamados de artefatos[1].

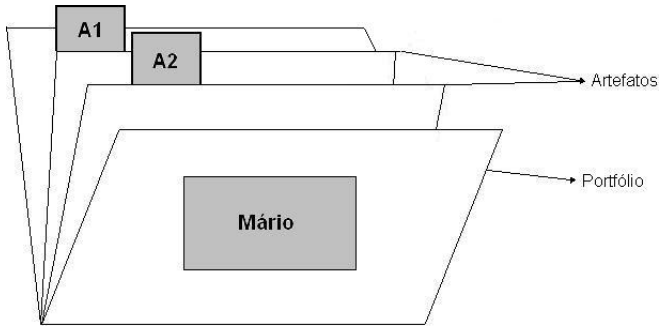


Figura 4.1: Um portfólio tradicional

Dentre os vários conceitos de Portfólio, temos:

"*Um Portfólio é uma pasta que contém evidências de habilidades individuais, interesses e acompanhamento de um aluno durante um certo período de tempo. Um portfólio tem um corpo do trabalho de um aluno, ou seja, ele é mais do que um mera coleção de documentos*" (Hart 1994).

"*Um portfólio não é um colecionador de atos de um aluno nem um arquivo de exercícios, projetos de curso e listas. É uma documentação organizada e direcionada ao crescimento e competência no ato de ensinar e através desta documentação, um portfólio representa um registro de informações, disposições e habilidades que são úteis no contexto profissional e educacional*" (Campbell 1997).

Armazenar e gerenciar o material existente nos portfólios tem sido um problema para muitos professores interessados em utilizá-los em sala de aula. Para manter portfólios que podem incluir papéis, projetos, vídeos e fitas de audio para alunos de um curso, uma escola teria que manter diversas salas adicionais a fim de armazenar esta abundância de informação. Em virtude disso, muitos educadores se tornam resistentes em adotar programas de avaliação de aprendizagem que incluam portfólios. Uma solução para isto é a criação e armazenamento de portfólios computadorizados "portfólios eletrônicos", ou seja, com as informações armazenadas em formato eletrônico acessíveis através do computador. Estes portfólios podem conter os mesmos tipos de informações, com a diferença de que estes são coletados, armazenados e gerenciados eletronicamente (Lankes 1995).

[1]Um artefato é uma evidência de um conhecimento que é adquirido, habilidades que são trabalhadas e valores que são mostrados ou disposições e atitudes que são características de cada pessoa.

Como as tecnologias existentes atualmente permitem a captura e armazenamento de informações na forma de textos, gráficos, sons e vídeo, os alunos podem gravar seus trabalhos em documentos coerentemente organizados. Com a disponibilização cada vez maior de acesso à internet, é possível manter os portfólios em um servidor, ao qual todos os alunos tenham acesso. Neste servidor, pode-se também implantar um sistema para armazenamento e gerenciamento de portfólio, que poderia auxiliar remotamente os alunos na criação ou na publicação dos mesmos. Desta forma, os professores também podem acompanhar praticamente, em tempo real, o desenvolvimento do trabalho dos alunos, podendo identificar problemas e orientá-los caso seja necessário.

A disponibilização de sistemas portfólios eletrônicos pode oferecer novas possibilidades de realização de avaliações mais profundas na aprendizagem dos alunos, porque facilitam e aumentam as possibilidades de análise dos dados existentes.

4.2 Utilização de Portfólios eletrônicos no contexto da avaliação

Mais de 500 faculdades e universidades nos Estados Unidos começaram, a partir da década de 90, a utilizar portfólios de ensino para a avaliação (Souza 2000). Na Europa, algumas universidades estão usando como instrumento de acompanhamento e melhoria do ensino (Souza 2000). No Brasil, não se tem muita tradição no uso de portfólios como instrumento para avaliação na Educação Superior (Souza 2000). Alguns educadores de cursos de formação de professores utilizam-no para registro de ações e reflexões especialmente no Estágio Supervisionado. Há escolas de Educação Básica utilizando esse tipo de instrumento para acompanhamento do desenvolvimento da aprendizagem dos alunos, especialmente para a Educação Infantil. No entanto, a maioria não faz o acompanhamento do desempenho do docente (Souza 2000).

Ao avaliarem os portfólios, os professores consideram o trabalho não de forma pontual (como a prova e testes), mas no contexto do ensino e como uma atividade complexa baseada em elementos de aprendizagem que se encontram relacionados.

O portfólio eletrônico na educação superior deve ser entendido como um facilitador da reconstrução e re-elaboração do processo de ensino-aprendizagem ao longo de um curso ou de um período de ensino (Winograd et al. 1991). Permite, conforme o autor, que os professores considerem o trabalho no contexto global do ensino e como uma atividade complexa baseada em elementos de aprendizagem que se encontram relacionados.

O portfólio eletrônico apresenta-se como uma estratégia flexível que oferece inúmeros benefícios a alunos e professores, principalmente por permitir documentar a "evolução" dos alunos, o processo e o produto da aprendizagem e a capacidade de resolver problemas, além de possibilitar a análise das vivências diárias dos alunos, podendo adaptar-se a qualquer

área disciplinar (desde o ensino das línguas até à matemática e ciências naturais) e nível de ensino (desde o pré-escolar ao ensino superior) (Nunes 1999).

4.3 Tipos de Portfólio Eletrônico

A partir do público que irá usufruir do portfólio é que são definidos tipos de conteúdo a serem armazenados como, dados pessoais do aluno (biografia, objetivos, trabalhos desenvolvidos, avaliação dos professores, atividades extracurriculares, comentários e sugestões dos professores ou pais, etc).

Um portfólio eletrônico é organizado em Portfólio do Aluno e Portfólio do Professor (Nascimento et al. 2002).

4.3.1 Portfólio do Aluno

Os portfólios podem ter diversas formas e tamanhos (Sistêlos 1999). Existem dois tipos de portfólios de alunos que podem ser estudados: portfólio do trabalho e o portfólio de apresentação.

Os portfólios dos trabalhos armazenam todos aqueles trabalhos (pesquisas, provas, trabalhos, etc) realizado pelos alunos, que podem servir como elementos de avaliação e de análise para redefinição de objetivos.

Os melhores trabalhos dos alunos, selecionados por alunos e professores, serão mostrados no portfólio de apresentação, para que qualquer pessoa tenha acesso aos trabalhos que mostram o conhecimento e as habilidades adquiridas. Isso traz motivação no processo de aprendizagem.

4.3.2 Portfólio do Professor

O portfólio do professor tem a intenção de servir como ajuda ao professor nas suas práticas de ensino, registrará as disciplinas para as quais estão qualificados, as turmas lecionadas e as estratégias didáticas utilizadas (Nascimento et al. 2002). As principais funcionalidades são:

- Identificar e registrar as diversas atividades (trabalhos, exercícios, listas, provas), realizados por cada aluno;
- Promover o acompanhamento do desempenho do aluno, turma, instituição, etc;
- Gerar chamadas a partir dos prazos colocados pelo professor nas atividades;
- Possibilitar a análise do aluno durante um período de tempo;
- Permitir a avaliação de estratégias de ensino e recursos utilizados.

Através da construção do portfólio do professor, eles também podem ser utilizados para mostrar aos próprios professores um crescimento profissional enquanto avaliam o desenvolvimento dos alunos.

4.4 Sistemas Portfólios Eletrônicos existentes

Existem vários tipos de portfólios que foram desenvolvidos, como *Connecticut College e-Portfolio* (Eportconsortium 2006), o *School of Education Webfolio* (Eportconsortium 2006), *Student Portfolio System* (Eportconsortium 2006). O Portfólio Eletrônico para Web (Silva 2002), o POETA (Sistêlos 1999) e o Portfólio-Tutor (Nascimento 2002).

4.4.1 *Connecticut College e-Portfolio*

O programa *Connecticut College e-Portfolio*, desenvolvido por *Career Enhancing Life Skills* (CELS), é uma ferramenta on-line que permite aos estudantes documentar seus processos de planejamento, estudos acadêmicos, acompanhamento extra-curriculares e toda a sua experiência de desenvolvimento durante seus anos de universidade.

Este recurso on-line faz com que estudantes documentem e apresentem informações aos seus professores do CELS. O centro e o programa de certificados de professores prepara para possuir experiência, participando dos estudos através de processos de desenvolvimento de aprendizagem on-line. O sistema fica apresentando evidências de seu desenvolvimento através de habilidades acadêmicas e experiências extra-curriculares, que podem ser incluídas em múltiplas pastas de Apresentação.

O *ePortfolio* fornece funcionalidades que aumentam a interação do estudante com seus professores. Pastas dos estudantes ou pastas de apresentação podem ser vistas via permissão especial pelo estudante em um ambiente seguro. A funcionalidade geral do *ePortfolio* inclui um "Upload de Arquivos", característica que os estudantes podem usar para documentar informação nos seus arquivos acadêmicos, internship e experiências extra-curriculares em vários formatos multimídias. Uma "Pasta de Apresentação" tem a função de permitir que estudantes selecionem e apresentem informações sobre si e suas experiências em vários formatos multimídias. Um "Programa de Resumo" cria um resumo do estudante em um documento. Um "Notas" tem uma função que permite aos professores registrar e compartilhar informação de seus alunos para outros professores e outros alunos. A versão 3.0 do ePortfolio está atualmente sob desenvolvimento.

A Faculdade Connecticut está atualmente empenhada em um Projeto de Colaboração de *ePortfolio* com a Faculdade de Dartmouth, a Faculdade de Holyoke de Monte e a Faculdade de União.

4.4.2 *School of Education Webfolio*

O portfólio é usado no programa de educação dos professores na Universidade de Lutheran na Califórnia (CLU) nos níveis elementar, secundário e educação especial para professores candidatos. Os professores submetem evidências de prática com a finalidade de fornecer um *feedback* aos candidatos. Esta conversação é continuada dentro do portfólio que fornece uma avaliação formativa de trabalho para os estudantes. O portfólio é organizado em cinco níveis de A à E. CLU usa os seis padrões da Califórnia para os professores ou padrões especiais de programa de educação e itens adicionais selecionados de acordo com a finalidade. Uma revisão da pasta de trabalho do estudante em uma conferência ocorre no fim de cada semestre e no fim do programa. O currículo e o programa do mestre da instrução em CLU também usa o Webfolio para endereçar sete metas do programa e é usado como um crescimento e progresso.

A flexibilidade do sistema *Webfolio* de CLU permite a seleção dos padrões/competências do programa da faculdade. O trabalho do estudante é considerado uma entrada permanente no sistema, onde cada semestre novos padrões e cursos podem ser incluídos.

O portfólio também é usado pelos instrutores no programa de suas aulas e pelos alunos para submeter seus trabalhos para avaliação e realimentação via *webfolio*.

O portfólio fornece uma base de dados valiosa com os trabalhos dos estudantes e avaliações no sentido de fornecer dados para pesquisa de professores e desenvolvimento de programas.

4.4.3 *Student Portfolio System*

Os estudantes da Universidade do estado de Illinois, Faculdade de Administração, trabalharam na criação de portfólios com multimídia baseado na internet desde 1995. Os portfólios são utilizados para demonstrar a estudantes, a partir de padrões curriculares à exposição, como trabalhar formalmente no curso, planejando carreira e atividades de vida de estudantes em suas experiências educacionais.

Na faculdade de Administração, os estudantes começam seus portfólios como um requisito em um curso de criação e edição de artefatos usando o Sistema *Profport Webfolio*. Os estudantes produzem os artefatos iniciais (exemplos de trabalhos) em resposta às atribuições neste curso. Trabalhando em equipe, a faculdade cria atribuições endereçando os padrões subjacentes para o curso. As "Melhores práticas" são apoiadas pelo software portfólio, que permite facilmente compartilhar recursos de planos de ensino entre faculdades.

Os estudantes podem adicionar a seus portfólios qualquer informação como um requisito ou uma atividade opcional em cursos subseqüentes e incluir planejamento de carreira e pesquisa de trabalho relacionados. Cada estudante pode controlar quais informações podem ser visualizadas em cada seção de seu portfólio. Em outras palavras, o estudante

pode criar representações alternativas de seu portfólio seja para professores, outros estudantes, recrutadores e conselheiros. Recentemente, estudantes têm a capacidade de exportar os dados de seu portfólio para CDs.

O software organiza automaticamente os padrões do programa às competências do curso e diretrizes para aumentar as atividades extra curriculares como recursos para estudantes que apresentem artefatos para o professor. A faculdade pode colocar padrões/ competências profissionais no sistema com artefatos associados produzindo atividades complementares para estudantes e então, pode fornecer comentários (professores podem comentar) para o estudante. E um comentário é mantido para cada artefato que um estudante produz.

4.4.4 POETA

O POETA (Portfólio Eletrônico Temporal e Ativo) é um ambiente que favorece a dinâmica de uma avaliação autêntica e faz uso do conceito de Portfólio Eletrônico (Sistêlos 1999). Este sistema propõe um Portfólio Eletrônico, que atenda os seguintes requisitos:

1. Como uma ferramenta do ensino:
    - Prover aos estudantes segurança, motivação, um senso de acompanhamento e participação;
    - Envolver estudantes no processo de auto-avaliação;
    - Ajudar professores e estudantes a estabelecerem objetivos;
    - Construir um tempo de reflexão sobre o acompanhamento do estudante;
2. Para o desenvolvimento profissional dos professores:
    - Estudar o currículo e a efetividade das práticas de ensino;
    - Identificar áreas fortes e fracas da escola e necessidades de melhoras;
    - Construir uma seqüência do planejamento de instrução.
3. Para avaliação:
    - Complementar ou substituir exames de competências;
    - Servir como registro de atividades realizadas durante o período letivo;
4. Para pesquisa:
    - Examinar o progresso do estudante;
    - Como instrumento em um processo de revisão.

Arquitetura

O POETA (Sistêlos 1999) foi projetado para funcionar em uma rede local como mostra a Figura 4.2.

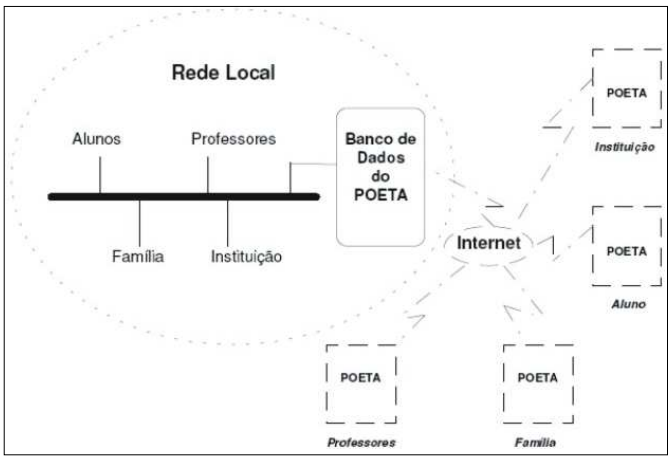


Figura 4.2: Arquitetura do POETA - Portfólio Eletrônico Temporal e Ativo

Os elementos envolvidos no aprendizado do aluno são alunos, professores e partes integrantes de seu desenvolvimento como família e instituição. A interação aluno/professor se dará através do seu contínuo relacionamento com o sistema, em que trabalharam em conjunto. Os professores elaboram tarefas e os alunos as executam, os professores as julgam e os alunos acessam estes julgamentos e observações e procedem as devidas alterações quando permitidas.

A família do aluno terá a possibilidade de, a qualquer momento, visualizar o progresso do aluno e seu desempenho em sala de aula através dos trabalhos por ele executados e armazenados pelo sistema. Além disso, a Instituição poderá perceber quando for implantada alguma alteração em sua estrutura, seja esta em um conteúdo programático ou em uma metodologia de ensino.

Indicadores de desempenho e de perfis serão fornecidos automaticamente pelo sistema e comunicados às entidades aluno/professor/família/instituição através de mensagens.

4.4.5 Portfólio Eletrônico para Web

Este sistema também utiliza o conceito de Portfólio Eletrônico utilizando técnicas de Banco de Dados Ativo e Temporal (Silva 2002), possibilitando a seus usuários um acompanhamento contínuo e dinâmico da evolução do aluno. Este Portfólio promove a adoção da Internet para o ambiente.

Arquitetura

O Portfólio Eletrônico para Web (Silva 2002) mostra na sua arquitetura (Figura 4.3) os componentes organizados em camadas, representados pelas seguintes camadas: a Camada de Apresentação (que possui o pacote de Interface), a Camada de Aplicação (com os pacotes Turma, Disciplina, Usuário, Atividade, Equipe, Perfil, Análise de Dados e Fórmula no pacote de Domínio e no pacote de Serviços temos Acesso ao Banco de Dados e Segurança) e a Camada de Armazenamento.

Existem atualmente dois tipos de Portfólios que podem ser desenvolvidos, como foi descrito na seção 4.3. O Portfólio de Trabalho, caracterizado pelo registro sistemático da documentação de trabalhos em curso. Este registro representa um meio para auto-avaliação e estabelecimento de objetivos e é sempre maior e mais completo que um Portfólio de apresentação. O Portfólio de Apresentação é fruto da seleção do trabalho que melhor reflete as competências registradas no Portfólio de Trabalho. Nele é guardado o melhor do trabalho do aluno.

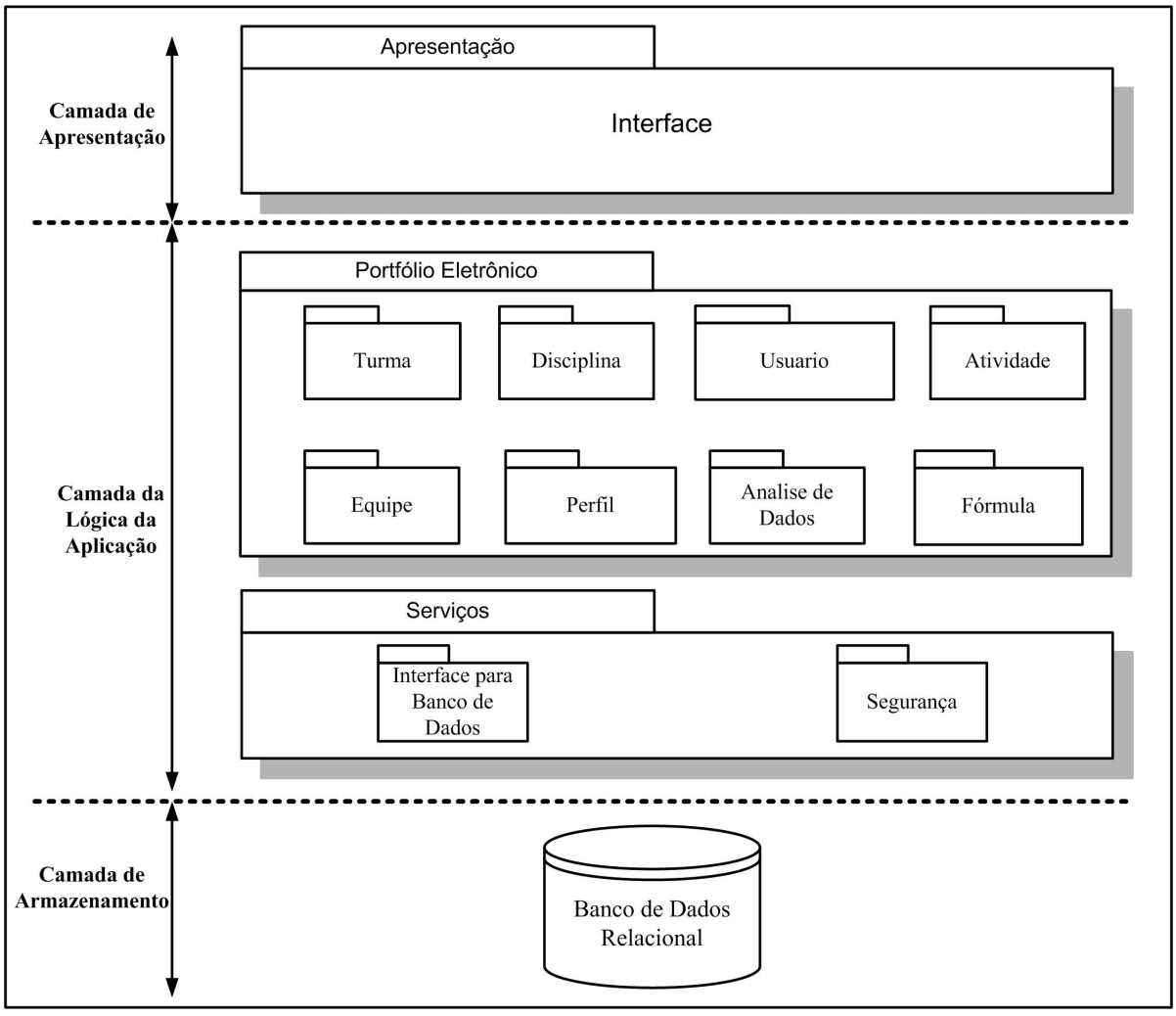


Figura 4.3: Arquitetura do Portfólio Eletrônico para Web

4.4.6 Framework para sistemas Portfólios Eletrônicos

Poucas iniciativas têm sido propostas objetivando o reaproveitamento da estrutura (núcleo ou esqueleto baseado em componentes) e do controle (predefinido) para um novo sistema portfólio eletrônico, por exemplo o framework Sakai (Ellis & Coppola 2006). Em relação aos sistemas portfólios eletrônicos existentes, o framework proposto nesta dissertação, para sistemas portfólio eletrônicos, tem como principal contribuição o reuso da arquitetura de um portfólio eletrônico e a diminuição do tempo de desenvolvimento de um novo sistema portfólio eletrônico.

4.5 Conclusões

Neste capítulo, apresentou-se o portfólio eletrônico, bem como, sua definição, utilização de portfólios eletrônicos no contexto da avaliação e tipos de portfólio eletrônico. É importante ressaltar que seus tipos são portfólio do aluno e portfólio do professor. Foi mostrado também alguns sistemas portfólios eletrônicos e um framework para sistemas portfólios eletrônico existentes.

É importante ressaltar a possibilidade do uso do computador em sala de aula, como instrumento de apoio às atividades de professores e alunos.

A necessidade de acompanhar o desenvolvimento de outras habilidades, além da apropriação de conteúdo, tem levado à adoção de portfólios. Em sua proposta pedagógica, o portfólio provê o registro sistemático dos trabalhos desenvolvidos por cada aluno, fornecendo ao professor e instituição de ensino melhores informações de como ajudar individualmente cada aluno e a instituição como um todo. Paralelamente, o aluno tendo consciência de como seus trabalhos são avaliados, juntamente com a possibilidade de disponibilização dos mesmos ao público em geral, sente-se mais motivado e passa a refletir sobre a produção de trabalhos de qualidade. Porém, para se obter eficácia esperada na aplicação desta estratégia, é indispensável o emprego de portfólios eletrônicos, arquivando, recuperando os documentos e fornecendo subsídios à tomada de decisão.

No próximo capítulo são apresentados os Sistemas Tutores Inteligentes.

Capítulo 5

Sistemas Tutores Inteligentes

Neste capítulo, será descrito o conceito de sistema tutor inteligente (STI), o que é útil para o entendimento da camada tutor que será utilizado nas aplicações Portfólio-Tutor. A seção 5.1 apresenta os ambientes de ensino-aprendizagem por computador. Na seção 5.2, é descrito a definição de sistema tutor inteligente. E na seção 5.3, descrevemos a arquitetura básica de um STI com seus módulos (Domínio, Comportamento do Aluno, Estratégias Pedagógicas, Controle e Interface).

5.1 Ambientes de Ensino-Aprendizagem por Computador

De acordo com Viccari (Giraffa & Viccari 2003), existem várias taxonomias para os programas educacionais. As mais tradicionais não contemplam as modalidades que utilizam técnicas de IA e os ambientes cooperativos. Inclusive, muitas delas não levam em consideração o tipo de aprendizagem proporcionada pelo ambiente. Devido a isto, originou-se a Tabela 5.1 que apresenta a proposta de Viccari para uma taxonomia.

Programas Educacionais	
1. Aprendizagem de Habilidades Específicas	**2. Aprendizagem de Habilidades Cognitivas Amplas**
CAI	MICROMUNDOS
Tutoriais	SISTEMA DE AUTORIA
Exercício-prática	JOGOS EDUCACIONAIS
Demonstração	ILE
Jogos e Simulação	
ICAI	

Tabela 5.1: Taxonomia para Programas Educacionais

De acordo com a Tabela 5.1, na categoria 1 estão os programas onde a aprendizagem

proporcionada pelo computador está centrada na aquisição de habilidades específicas.

Segundo Viccari (Giraffa & Viccari 2003), os programas educacionais dessa modalidade se dividem em dois grandes grupos:

- Os CAI (*Computer Aided Instruction*) surgiram no início de 1960, a partir de projetos na área de educação, por isso que nem sempre satisfaz os requisitos de uma teoria formal na área de Ciência da Computação.

  Os CAI foram criados para oferecer uma ajuda ao ensino em diversas áreas sem a utilização do modelo do aluno para orientar a forma de interação.

  Os CAI possuem as seguintes modalidades:

  - *Tutoriais:* o aluno estuda como se estivesse na sala de aula, porque o assunto é organizado pelo professor e o aluno escolhe qual tutorial estudar. Exemplos de tutoriais utilizam hipermídia e com frequência são disponibilizadas na internet;
  - *Exercício-prática:* o aluno pratica e testa conhecimentos de forma dirigida e procedural. As versões mais atuais desses programas utilizam recursos hipermídia mantendo essas características;
  - *Jogos e as simulações:* esta modalidade vem crescendo muito devido ao desenvolvimento dos recursos de *hardware* e *software*. A diferença entre os jogos e as simulações encontra-s no fato de que os jogos utilizam processos competitivos enquanto que as simulações utilizam um modelo pré-definido que o aluno utiliza de forma individual;

- Os ICAI (*Intelligent Computer Aided Instruction*) surgiram na década de 70 com a tentativa de aumentar as limitações impostas pelos sistemas CAI's. Para isso, passaram a incorporar recursos da Inteligência Artificial (IA) e da Psicologia Cognitiva (Costa 1997). Sistemas ICAI passaram a ser conhecidos por muitos autores como Sistemas Tutores Inteligentes (Wenger 1987).

Na categoria 2 referente a Tabela 5.1, estão os programas onde a aprendizagem proporcionada pelo computador está centrada na aquisição de habilidades cognitivas amplas.

Os programas educacionais dessa modalidade está dividida da seguinte forma:

- *Micromundos:* surgidos na década de 60, se caracterizaram por ser uma proposta contrária aos CAI. Nesse sentido, o ambiente permite que o aluno trabalhe de forma diversificada, segundo seu próprio ritmo, utilizando recursos de programação inerentes ao ambiente. Os micromundos se baseiam em que a ênfase da aprendizagem está na construção do conhecimento por parte do aluno e não na trasmissão de conhecimentos (como ocorre na categoria CAI). Os micromundos foram criados para desenvolver habilidades cognitivas no aluno e estimular o pensamento reflexivo;

- *Sistemas de Autoria:* ferramenta de criação que possibilita ao aluno viabilizar seu projeto de trabalho. Um sistema de autoria oferece ao aluno a possibilidade de explorar um grande conjunto de habilidades cognitivas, exercendo sua criatividade;

- *Jogos Educacionais:* apresentam uma concepção diferenciada daqueles apresentados na categoria 1, porque, nesses ambientes, existe um modelo de simulação em que seu tipo de ação executada pelo aluno fará diferença no resultado do jogo. São ambientes sofisticados que utilizam uma complexidade alta, tanto no seu projeto como na sua implementação;

- *ILE* (*Intelligent Learning Enviroment*): também conhecido como Sistema Tutor Cooperativo ou Sistema de Aprendizagem Social, utilizam técnicas de IA em seu projeto e em seu desenvolvimento.

  De acordo com Costa (Costa 1997), um ILE caracteriza-se como uma categoria de software educacional na qual o aluno é inserido em uma situação de descoberta, englobando atividades de resolução de problemas. Durante esse processo, o tutor assiste o aluno em sua atividade e monitora o seu aprendizado. O foco de atenção de problema por parte do aluno passa a considerar mais o processo de solução, verificando passo-a-passo o seu conteúdo.

A Tabela 5.2 mostra as grandes diferenças entre o grupo CAI e os STI.

	CAI	**STI**
Origem	Educação	Ciência da Computação
Bases Teóricas	Behaviorista	Psicologia Cognitiva
Estruturação e Funções	Uma única estrutura pré-definida, em que o aluno não influi na sequência	Estrutura subdividida em módulos, e sua sequência se dá nas respostas do aluno
Estruturação do Conhecimento	Algorítmica	Heurística
Modelagem do Aluno	Avaliam a última resposta do Aluno	Tentam obter um histórico do aluno durante toda a interação
Modalidades	Tutorial, exercício e prática, simulação e jogos educativos	Socrático, ambiente interativo, diálogo bidirecional e guia

Tabela 5.2: Diferenças entre o grupo CAI e os STI

Os STI utilizam técnicas de Inteligência Artificial. Por volta da segunda metade da década de 80, as pesquisas de STI tiveram concentração para envolver questões pedagógicas e, no terceiro estágio na década de 90, a concentração foi também na área pedagógica, sendo que foi uma concentração específica em que existiam equipes interdis-

ciplinares (Costa & Werneck 1996). Os sistemas tutores inteligentes foram criados e tem potencial para serem utilizados pelos professores e alunos em diferentes domínios.

5.2 Definição

Sistema 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) (Silva 2000) (Wenger 1987).

O Sistema Tutor Inteligente é um programa que possui uma base de domínio (Viccari 1990). Esses sistemas possuem uma capacidade de adaptação em função das estratégias didáticas que são criadas pelos professores.

Cada aluno possui uma forma de aprendizado diferente, por isso não deve-se ensinar de maneira igual a todos os alunos, ou seja, deve-se possuir vários tipos de atividades no decorrer da aprendizagem levando em conta o ritmo dos alunos e, para isso, deve-se utilizar vários tipos e formas de recursos para os alunos (Giraffa & Viccari 2003).

Os Sistemas Tutores Inteligentes caracterizam-se pelo contexto interdisciplinar, ou seja, deve existir uma equipe de especialistas em diversas áreas como instrutores, pedagogos, psicólogos e especialistas em programação para que se possa analisar, projetar e implementar um STI (Giraffa & Viccari 2003).

Em um STI o sistema interage com o aluno, com funcionalidades parecidas as de um professor em que, a modelagem cognitiva é constante e progressiva (Silveira 1995). Para que um tutor possa aumentar sua base de conhecimento, ele deve ser flexível, ou seja, poder aprender com os alunos através das interações.

Os STI possuem um marco histórico em 1970 que chama-se *Scholar*, considerado o clássico dos STI. Foi projetado e construído pela equipe do prof. Jaime Carbonell, no BBN-Bolt Beranek and Newman Inc., laboratório de pesquisa em Cambridge, Massachussets, EUA. É um sistema tutorial que conduz a um diálogo com o aluno e seu objetivo é o ensino de geografia da América do Sul.

O tutor ensina através de uma abordagem de uma exposição indireta. Os alunos devem avaliar hipóteses, descobrir contradições e fazer inferências corretas. Um exemplo de uso do *Scholar*, é mostrado na Figura 5.1.

A base de domínio construída nesse tutor foi desenvolvida através de regras de produção. Ele utiliza módulos separados para analisar as respostas e discutir erros.

Scholar: O planalto central e leste da Argentina é...
Aluno: Chaco
Scholar: Não. O planalto nordeste da Argentina é o Chaco.O planalto central e leste da Argentina são os Pampas.
Aluno: Chaco é no Chile?
Scholar: Não. O Chaco é na Argentina, Paraguai e Bolívia, não no Chile.
Aluno: Os Andes é uma região do Chile?
Scholar: Bem, você está quase certo. Mas especificamente, a região que envolve o Chile são os Andes do Sul.

Figura 5.1: Diálogo entre *Scholar* e o aluno

5.3 Arquitetura Básica

As arquiteturas de Sistemas Tutores Inteligentes variam de acordo com o sistema implementado. Entretanto, existem alguns componentes ou módulos básicos que podem ser observados em diversos trabalhos (Wenger 1987), (Barr & Feigenbaum 1982), (Yazdani 1987), (Self 1988), (Viccari 1990) e (Oliveira 1994). A arquitetura de um STI possuem *o modelo do aluno, as estratégias de ensino, o modelo de domínio e uma interface.* A figura 5.2 mostra a arquitetura com suas inter-relações.

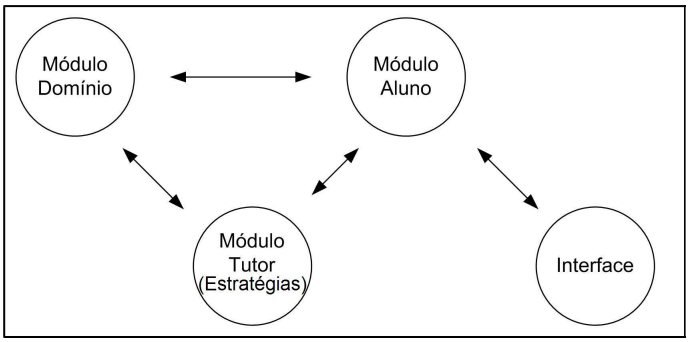


Figura 5.2: Arquitetura Clássica de um STI

Essa arquitetura clássica foi um avanço à modelagem de ambientes, porque separou o domínio da sua forma de manipulação, levando em conta, estratégias de ensino que consideram o comportamento do aluno.

A arquitetura clássica foi ampliada para se tornar uma arquitetura associada ao modelo de interação, que ocorre em uma interação entre alunos e o ambiente.

Os elementos (módulos) da arquitetura de um Sistema Tutor Inteligente é mostrada na Figura 5.3.

O *Especialista* é responsável por manipular o conteúdo que vai ser ensinado pelo STI. Esse componente do sistema é muitas vezes referenciado na literatura como a denominação de "Base de Conhecimento do Domínio" (Silva 2000). Vários modelos podem ser utilizados na implementação desse componente, como exemplo redes semânticas, *frames*, regras de

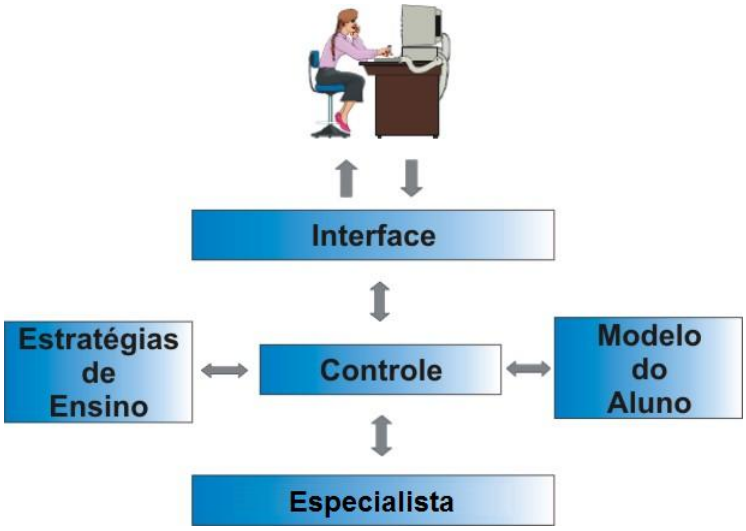


Figura 5.3: Arquitetura de um Sistema Tutor Inteligente

produção, dentre outras.

O *modelo do aluno* deve captar o estado do entendimento do aluno a respeito do assunto que está sendo apresentado (Wenger 1987). O modelo do aluno relaciona-se com as informações obtidas sobre um domínio. Representa o conhecimento e as habilidades do aluno (Viccari 1990) e esse conhecimento é de grande importância para o tutor, pois ele pode comprovar hipóteses a respeito do conhecimento e comportamento do aluno. A partir deste modelo, o sistema pode ser capaz de escolher a melhor estratégia de ensino para cada aluno.

As *estratégias de ensino* representam as possíveis ações a serem tomadas, a fim de que o objetivo da aprendizagem seja atingido. A decisão sobre qual estratégia será aplicada depende do modelo do aluno e de como o conteúdo instrucional está estruturado na base de domínio. Portanto, este módulo conterá o conhecimento de como executar as ações capazes de apresentar um assunto a determinado aluno, a partir do diagnóstico e monitoração do mesmo. Ressalta-se que as estratégias de ensino estão intimamente ligadas ao projeto instrucional que está sendo utilizado.

O *módulo de controle* é responsável pelas interações dos módulos, ou seja, coordena o funcionamento do sistema tutor, faz a integração dos módulos e permite a ativação e o encerramento de sessões de ensino.

Uma boa *interface* é muito importante para qualquer sistema, principalmente nos Sistemas Tutores Inteligentes que através da interface, o STI apresenta o material instrucional para o aluno e monitora seu progresso, e, através dessas funções existem objetivos a serem cumpridos. São eles:

1. É preciso que os recursos de apresentação sejam bem interpretados pelos alunos;

2. Intervenção na conversação: o aluno pode intervir na conversa com o tutor e vice-

versa;

3. A resposta tanto do aluno como do tutor não devem ser muito demoradas, porque existem limites;

4. A monitoração do aluno deve ser em estilo background.

5.4 Conclusões

Neste capítulo, apresentou-se os STI's, através dos ambientes de ensino-aprendizagem por computador, sua definição, os módulos da sua arquitetura, aspectos que devem ser considerados e algumas de suas tendências de pesquisa.

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 ao ritmo de aprendizado do aluno.

No capítulo seguinte será apresentado Agentes.

Capítulo 6

Agentes

Neste capítulo, serão descritos os principais conceitos sobre agentes, o que é útil para o entendimento da camada agentes nas aplicações Portfólio-Tutor. A seção 6.1 apresenta a definição de agentes. Na seção 6.2, são descritas as propriedades dos agentes. A seção 6.3 descreve os tipos de classificação de um agente, bem como sua taxonomia. E na seção 6.4, fala-se da modelagem orientada a agentes.

6.1 Definição

Agentes representam um novo paradigma de desenvolvimento de aplicações (Wooldrige & Jennings 1998) e a cada dia que passa, aumenta o interesse na área. A definição de agentes não é universal, cada pesquisador (em áreas diferentes) busca procurar o termo em seu contexto.

Dentre as várias definições de agentes, destacam-se as seguintes:

"Agente é uma entidade autônoma que percebe seu ambiente através de sensores e age sobre o mesmo utilizando-se de atuadores" (Russell & Norving 2004). Como mostra a Figura 6.1.

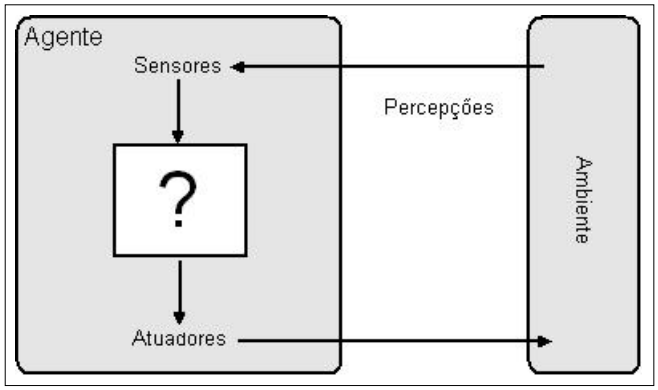


Figura 6.1: Ambiente de um Agente

O termo percepção é utilizado para atuar como entradas (exemplos de dispositivos de entradas são câmeras e detectores de infra-vermelho funcionando como sensores) do agente em qualquer momento, ou seja, um agente possui como entrada qualquer sequência de percepções e, a partir disto, é gerado uma ação em resposta a esta sequência.

"Agentes são entidades baseadas em hardware ou software, com as propriedades de Autonomia, Habilidade Social, Reatividade, Pró-Atividade e Mobilidade" (Wooldrige & Jennings 1995).

6.2 Propriedades

Entender as propriedades de agentes auxilia o processo de classificação conforme será visto na seção 6.3 (Franklin & Graesser 1996) e (Tecuci 1998). Abaixo estão relacionadas as principais propriedades dos agentes.

- **Reatividade:** o agente deve ser capaz de reagir apropriadamente perante as informações sensoriais provenientes do ambiente. Para que ocorra este processo de percepção, os agentes são dotados de um conjunto de sensores;
- **Autonomia:** a maior diferença entre agentes e softwares tradicionais é a capacidade de alcançar suas metas sem interações ou comandos do ambiente. De uma forma simplificada, podemos dizer que o agente não precisa ter todas as suas decisões aprovadas por um usuário;
- **Habilidade Social:** os agentes interagem com outros agentes através de algum tipo de linguagem de comunicação de agentes;
- **Pró-Atividade:** um agente não reage somente aos impulsos do ambiente, mas também toma iniciativas sobre circunstâncias específicas. Esta é a capacidade de agir conforme metas. O agente não reage simplesmente ao ambiente, as ações também são determinadas por metas;
- **Contínuo Temporalmente:** o agente é um processo que está continuamente em execução no tempo;
- **Comunicação:** é a capacidade de trocar informações com outros agentes computacionais e com usuários humanos. Para que este processo ocorra é necessário a existência de um protocolo padrão de comunicação;
- **Adaptação:** é a habilidade de extrair conhecimento a partir de experiências anteriores e adaptar, sucessivamente, seu comportamento perante o ambiente;

- **Mobilidade:** é a capacidade dos agentes navegarem através de redes de computadores. Os agentes limitados a uma máquina específica são chamados de estacionários;

6.3 Classificação

Existem diversas classificações possíveis para o universo dos agentes considerando os diferentes aspectos que caracterizam um sistema de agentes. Esta seção apresenta uma classificação baseada na proposta de Muller (Muller 1998) e Wooldrige (Wooldrige & Jennings 1998).

Muller (Muller 1998) considera que o ponto central dos agentes autônomos é a sua arquitetura de controle, isto é, a descrição de seus módulos e de como eles trabalham juntos. Considera que todas as inúmeras arquiteturas propostas possuem a influência de três correntes agentes reativos, agentes deliberativos e agentes interativos. Deste modo, existem três formas de classificar a arquitetura de software onde um determinado agente será executado:

6.3.1 Agentes Reativos

A idéia principal deste tipo de arquitetura é que o comportamento inteligente do agente surge a partir da interação de vários comportamentos simples. A propriedade da pró-atividade é implementada de forma implícita (Brenner et al. 1998). Uma arquitetura está baseada em tarefas relacionadas a comportamentos (Wooldrige 1999). Cada um destes representam uma tarefa desenvolvendo um tipo de ação, como mostra a Figura 6.2.

Cada comportamento não utiliza nenhuma representação simbólica, no geral usam regras do tipo "*se determinada condição então execute a ação x*". Um ponto importante desta arquitetura é que mais de um comportamento pode ser ativado, portanto existe a necessidade de um mecanismo para escolher a melhor ação. Então, estes comportamentos devem ser organizados em camadas, sendo as camadas inferiores as que possuem uma prioridade mais alta.

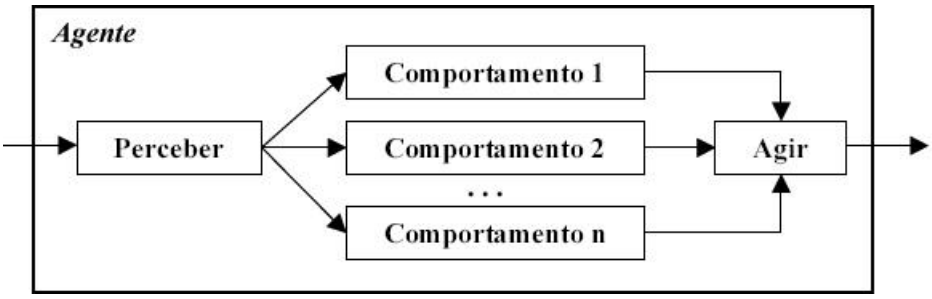


Figura 6.2: Arquitetura dos Agentes Reativos

Esta é uma abordagem robusta contra falhas. Caso ocorra algum problema em um comportamento, o agente não deixa de agir, porque os outros comportamentos estão ativos (Brenner et al. 1998).

6.3.2 Agentes Deliberativos

Uma arquitetura que utiliza um estado interno cujo processo decisório é fundamentado na dedução lógica. Os agentes que são baseados nesta arquitetura são chamados de deliberativos ou também conhecidos como agentes cognitivos. O estado interno do agente é representado por um conjunto de informações, um modelo explícito e um ambiente (Brenner et al. 1998). Em geral utiliza técnicas de Inteligência Artificial.

A arquitetura deliberativa é uma abordagem simples e com uma semântica clara, mas em compensação possui algumas desvantagens. A principal destas é que agentes baseados em lógica lidam com variáveis simbólicas e o mapeamento de algumas informações sensoriais são difíceis de serem representadas simbolicamente. Além disso, esta arquitetura não é muito efetiva em ambientes dinâmicos porque o processo de dedução em geral é complexo e, por conseqüência, é um pouco lento (Brenner et al. 1998). Desta forma, a ação resultante da inferência pode não ser a mais adequada pelo fato do ambiente ter sido modificado durante o processo decisório.

6.3.3 Agentes Interativos

São capazes de coordenar suas atividades com as de outros agentes através de comunicação. Agentes interativos têm sido principalmente investigados na área de Inteligência Artificial Distribuída (DAI), isto é, na construção de sistemas multiagentes. Este tipo de agente possui representação explícita dos outros agentes e pode ser capaz de raciocinar sobre eles. A propriedade no qual ele tem foco é a habilidade de comportamento social cooperativo. A construção de agentes interativos deve tratar de aspectos de coordenação e cooperação entre os agentes inteligentes distribuídos.

6.3.4 Agentes Híbridos

Possui componentes das arquiteturas reativas e deliberativas. As decisões são tomadas via várias camadas de software (Wooldrige & Jennings 1995).

6.4 Modelagem Orientada a Agentes

O progresso da engenharia de software nas duas décadas passadas teve um desenvolvimento devido as abstrações de alto nível. Como exemplo de abstrações temos processos, tipos abstratos de dados e na maioria das vezes, classes / objetos e componentes. É nossa crença que agentes representem um avanço semelhante na abstração. Podem ser usados por engenheiros de software, mais naturalmente, para entender, modelar e desenvolver um sistema distribuído complexo. Será necessário desenvolver técnicas de engenharia de software adequadas a agentes. As técnicas existentes de desenvolvimento de software (as fases de análise e projeto na orientação a objetos) são inadequadas para esta tarefa. Por estas razões existem várias metodologias para a modelagem orientada a agentes, entre elas, GAIA (Wooldrige et al. 2000) e AUML (Odell et al. 2000), que foram criados para análise e projeto de sistemas baseados em orientação à agentes. No apêndice A é descrita a metodologia GAIA em detalhes.

6.5 Ferramentas de Implementação de sistemas baseados em Agentes

Várias ferramentas estão disponíveis para o desenvolvimento de sistemas utilizando agentes, tais como: AgentBuilder (AgentBuilder 2006), Zeus (Zeus 2006) e JADE (JADE 2006).

AgentBuilder (AgentBuilder 2006) fornece ferramentas gráficas para suportar todas as fases do processo da construção de um sistema baseado em agentes. A ferramenta ZEUS (Zeus 2006) consiste de um conjunto de componentes, escritos na linguagem de programação Java, que podem ser categorizados em três grupos funcionais (ou bibliotecas): uma biblioteca de componentes agentes, uma ferramenta para a construção de agentes e um grupo de agentes de utilidade que incluem os agentes nameserver, facilitator e visualiser. A ferramenta JADE (Java Agent Development framework) (JADE 2006) é um ambiente para desenvolvimento de aplicações baseadas em agentes conforme as especificações da FIPA (Foundation for Intelligent Physical Agents) (FIPA 2006) para interoperabilidade entre sistemas multiagentes.

No apêndice B são descritas estas ferramentas em detalhes.

6.6 Conclusões

Neste capítulo, apresentou-se os agentes: com suas definições, propriedades, classificação (agentes reativos, agentes deliberativos e agentes interativos), mostrou-se a modelagem orientada a agentes e as ferramentas utilizadas atualmente para implementação de sistemas baseados em agentes.

A utilização da tecnologia de agentes vem sendo bastante pesquisada.

Assim, no desenvolvimento desta trabalho, considerou-se que uma das camadas de um sistema Portólio-Tutor, criado a partir do framework FA_PorT, são representados por agentes quem podem ser classificados segundo a característica de reatividade - os agentes percebem seu ambiente (no caso, uma coleção de outros agentes) e respondem aos estímulos deles recebidos.

Nos próximos capítulos serão especificados e implementados o frameworks FA_PorT e dois estudos de caso de aplicações Portfólio-Tutor construídas a partir do framework.

Parte II

Especificação do FA_PorT

Capítulo 7

Especificação do FA_PorT

Neste capítulo, será mostrada a especificação do framework. A seção 7.1 apresentará os requisitos para desenvolvimento do framework. Na seção 7.2, descrevemos a identificação e caracterização do domínio que o framework aborda. Na seção 7.3, mostra-se a arquitetura do FA_PorT, na seção 7.4, os diagramas de componentes, na seção 7.5, a especificação de cada componente e sua respectiva interface são apresentadas. A seção 7.6 apresenta o controle do framework, na seção 7.7, o funcionamento de sistemas Portfólio-Tutor é descrito e, por fim, na seção 7.8, a construção de aplicações a partir do uso do framework FA_PorT é mostrada.

7.1 Requisitos

Os principais requisitos do framework FA_PorT, são:

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

7.2 Identificação e Caracterização do Domínio

No contexto do framework FA_PorT, foram analisados dois sistemas tutores, o Portfólio-Tutor – Um Sistema Tutor Acoplado a um Portfólio Eletrônico (Nascimento 2002) e o TUTA – Um Tutor Baseado em Agentes (Silva 2000). E, foram analisados os sistemas de tipo portfólio eletrônico, apresentados no capítulo 4.

7.2.1 Análise de Sistemas Tutores Inteligentes

Portfólio-Tutor

O Portfólio-Tutor representa um sistema tutor acoplado a um portfólio eletrônico. O Portfólio-Tutor (Nascimento 2002) foi desenvolvido no contexto do projeto ACVA (Arquitetura de uma Classe Virtual Adaptativa) (Hernández-Domínguez 1997) com o propósito de construir um ambiente de ensino/aprendizagem com enfoque na criação de um ambiente colaborativo de aprendizagem e no ensino personalizado. A camada tutor-ACVA do Portfólio-Tutor tem como principal objetivo gerenciar o grupo e a zona de comportamento do aluno, aplicando uma estratégia didática, ele também controla a mobilidade dos alunos nos grupos e zonas de comportamento.

A partir do conteúdo obtido através do portfólio eletrônico, o professor, poderá avaliar os trabalhos, as participações dos alunos, de forma grupal e individualmente.

O Portfólio-Tutor é formado por quatro camadas, como mostra a Figura 7.1: a camada banco de dados, duas camadas de negócio (a camada Portfólio que é responsável pelas funcionalidades administrativas, registro e acompanhamento e a camada Tutor-ACVA que será responsável pelo ensino dos alunos) e a interface.

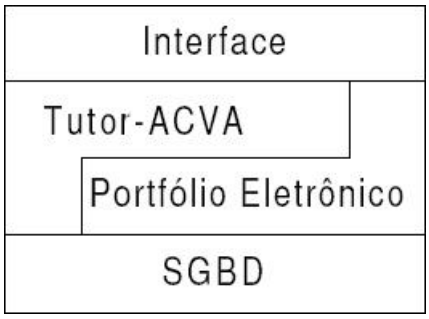


Figura 7.1: Camadas do Portfólio-Tutor

- Arquitetura

A arquitetura do Portfólio-Tutor é representada na Figura 7.2.

A camada lógica da aplicação foi dividida em Camada de Domínio, que foi sub-dividida em Portfólio (onde possue os pacotes elementos administrativos e registros), Tutor-Acva

(que contém os pacotes estratégia didática, perfil do aluno e base de domínio) e camada de Serviço (formada pelos pacotes de acesso a banco de dados, segurança e comunicação), mostrados na Figura 7.2.

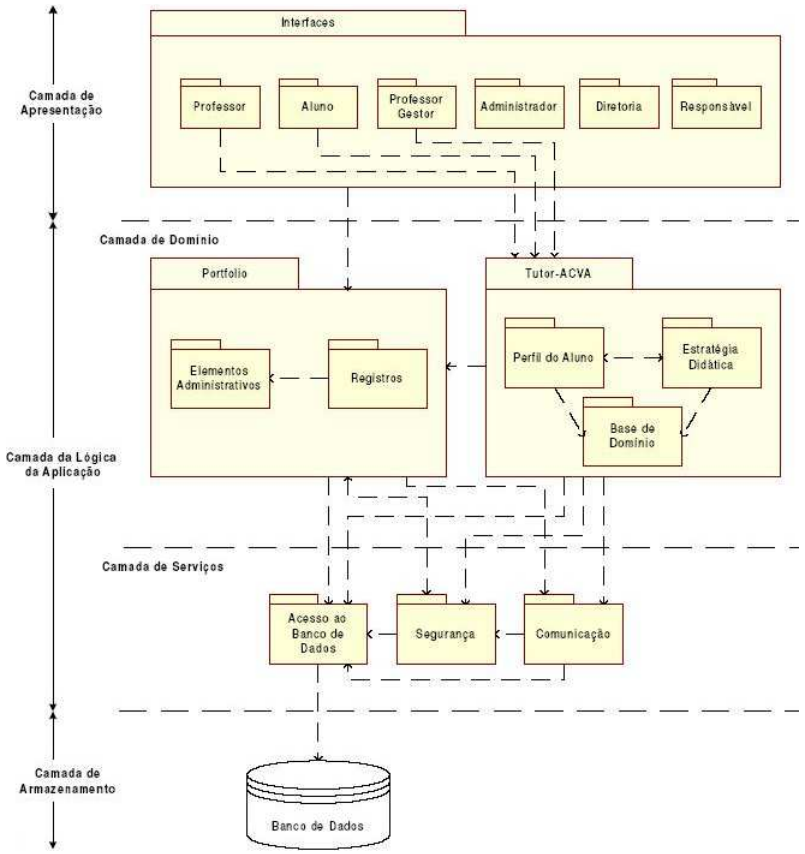


Figura 7.2: Arquitetura do Portfólio-Tutor

A camada de armazenamento é composta por um banco de dados relacional. A camada de serviços possui os pacotes de acesso ao banco de dados e segurança, que corresponde às classes responsáveis pelo controle de acesso às funcionalidades da aplicação. O pacote de comunicação possui os elementos de envio de emails, transferências de arquivos e comunicação síncrona. A camada de domínio é formada pelas classes que representam os conceitos de domínio da aplicação. A camada de apresentação representada pelas interfaces que constituem o sistema.

TUTA

O TUTA (Um Tutor Baseado em Agentes) (Silva 2000) representa um sistema tutor que gerencia um grupo de alunos, auxiliando o professor nas tarefas de ensino.

O TUTA possui as camadas Tutor e Servidor de Recursos Didáticos, sua junção permite que se tenha um sistema tutor dedicado ao ensino-aprendizagem de um domínio em particular associado a um determinado nível de conhecimento. A Figura 7.3 mostra essas camadas.

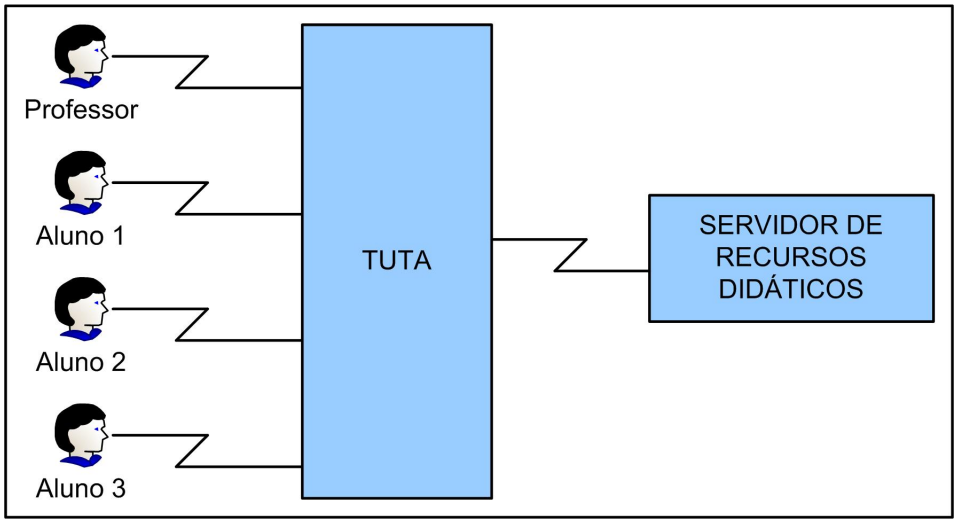


Figura 7.3: Camadas do TUTA

- Arquitetura

No TUTA o professor pode auxiliar ao aluno ou a um grupo de alunos geograficamente distribuídos através da internet. Ele também segue o contexto da arquitetura ACVA como mostra a Figura 7.4.

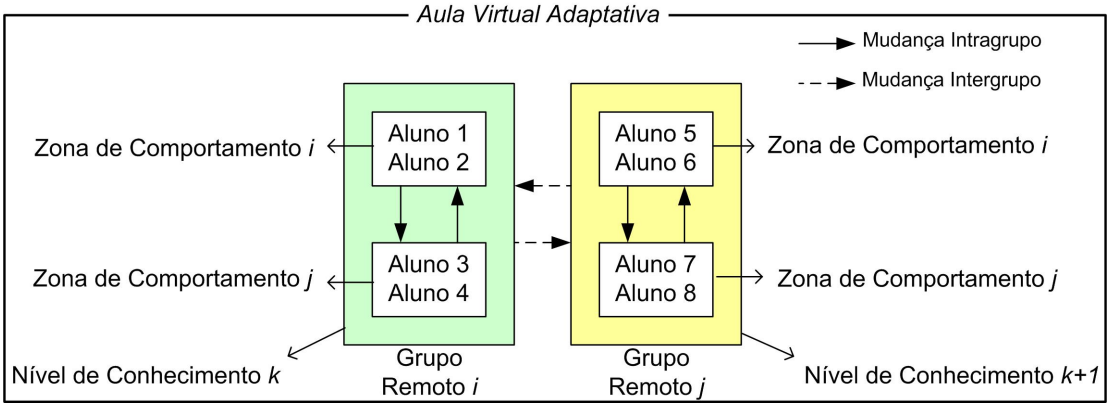


Figura 7.4: Distribuição dos Alunos na ACVA

Na ACVA, uma aula virtual adaptativa é formada por vários grupos heterogêneos de alunos com níveis de conhecimento diferentes, e em cada nível os diferentes níveis de progressão dos alunos são representados através das zonas de comportamento. Então, a adaptabilidade dos alunos na ACVA é representada através do gerenciamento dos diferentes níveis de conhecimento e zonas de comportamento.

A Figura 7.5 mostra a arquitetura geral da ACVA, que possui quatro níveis de abstração ou camadas, *Camada de Serviços de Formação da Aula Virtual SF-CV* que é manuseada pelo controlador de aula que detém o controle pedagógico dos grupos de aprendizagem da aula virtual, *Camada de Serviços de Formação de Grupo (SF-Grupo)* representa um

tutor de um grupo de alunos associados a um nível de conhecimento, *Camada de Serviços de Formação Básicos (SF-Básicos)* permite que cada controlador do grupo utilize recursos didáticos comuns e a *Camada de Suporte* permite o armazenamento de informações e comunicação entre os alunos e/ou professor.

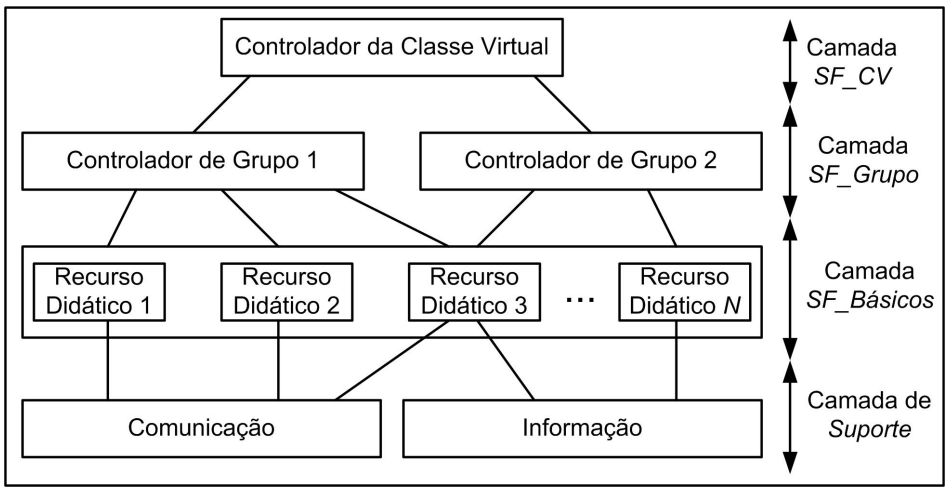


Figura 7.5: Arquitetura da ACVA

A arquitetura geral do TUTA, mostrada na Figura 7.6, apresenta os seguintes elementos básicos:

- *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*: desempenhará atividades de duas categorias. A primeira consiste em atividades de definição do curso, são elas: 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;
- *Grupo*: conjunto de pessoas (alunos e professores) que se reúnem para participar das sessões de ensino de um curso;
- *Administrador do Sistema*: responsável pela autorização e inclusão de professores no sistema;
- *Agente Interface Aluno*: encarrega-se de realizar tudo que for necessário para a execução de uma sessão de ensino, inclusive permitindo a interação entre os alunos. Cada aluno participante de um curso possui seu próprio Agente Interface Aluno;

- *Agente Interface Professor*: encarrega-se de permitir a interação entre o professor e os alunos durante uma sessão de ensino;
- *Agentes Interfaces de Sessão*: conjunto de interfaces (Agentes Interface Aluno e Agente Interface Professor) a serem utilizados em uma sessão de ensino;
- *Agente Interface Especificação Curso (ou Agente de Autoria)*: encarrega-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;
- *Conjunto de Agentes do TUTA*: coleção de agentes que podem cooperar entre si, a fim de permitir a realização de uma sessão de ensino e assim, promover a aquisição de conhecimento em um dado domínio pelo aluno.
- *Sessões de Ensino*: armazenam as sessões de ensino definidas e suas estratégias didáticas associadas;
- *Estratégias Didáticas*: armazena as estratégias didáticas a serem utilizadas nas sessões de ensino de um curso;
- *Domínio*: armazena o conteúdo a ser ensinado ao aluno através de recursos de natureza didática;
- *Perfil do Aluno*: armazena os dados gerais do aluno e o seu perfil;
- *Perfil do Grupo*: armazena os dados gerais do grupo e o seu perfil;

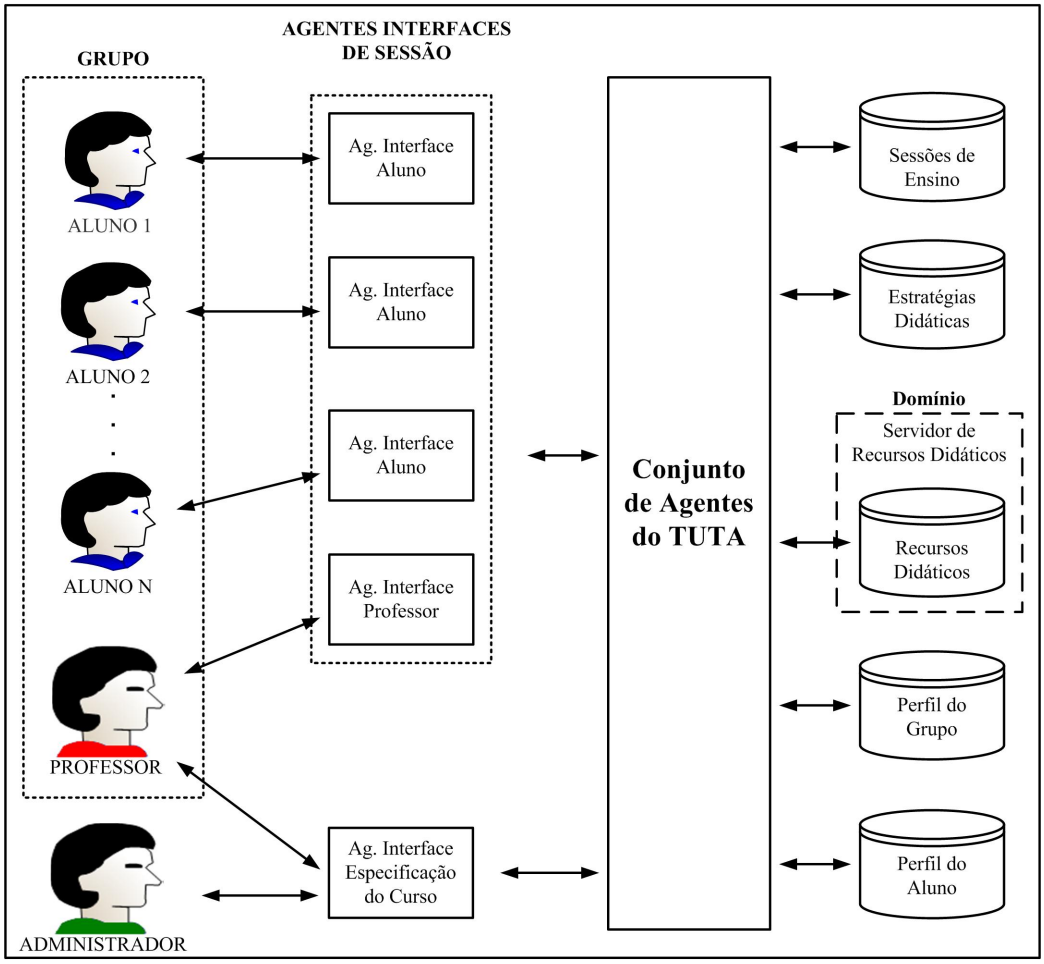


Figura 7.6: Arquitetura geral do TUTA

- Modelagem dos Agentes

Segundo Wooldrige (Wooldrige et al. 1999), uma metodologia orientado a agentes deve possuir os aspectos, a flexibilidade do agente, seu comportamento autônomo, as interações e a complexidade das estruturas organizacionais onde os agentes estão trabalhando. Para a especificação dos agentes do TUTA foi utilizada a metodologia GAIA. A modelagem consiste em uma sociedade ou uma organização artificial que possuem um conjunto de papéis. A especificação dos agentes do TUTA (Silva 2000) é descrita no Apêndice E

7.2.2 Identificação das Entidades em comum

Depois de analisarmos a camada Portfólio do Portfólio-Tutor (Nascimento 2002), o POETA (Sistêlos 1999), o Portfólio Eletrônico para Web (Silva 2002), a camada Tutor do Portfólio-Tutor (Nascimento 2002) e o TUTA (Silva 2000), identificamos entidades em comum que podem ser representadas em um modelo conceitual (Figura 7.7).

O modelo conceitual, mostrado na Figura 7.7, representa os elementos em comum da camada portfólio eletrônico, dividido em elementos administrativos (armazenamento

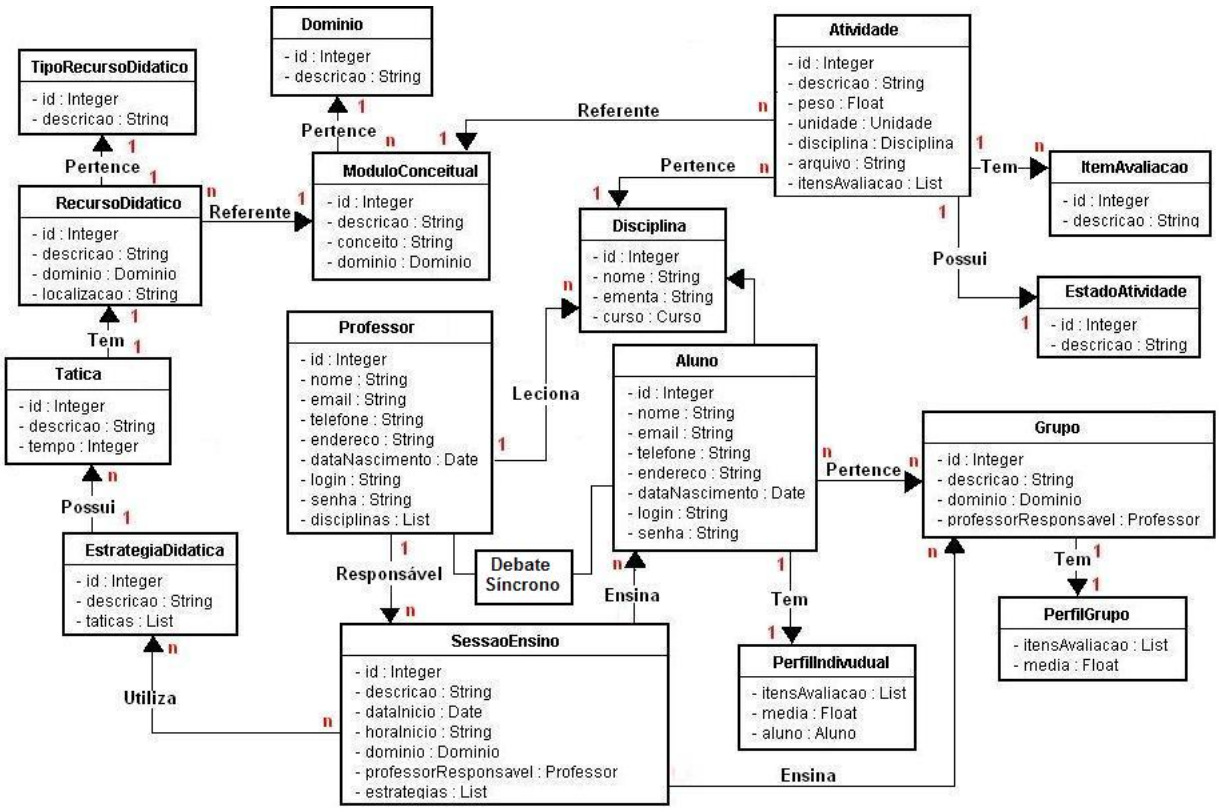


Figura 7.7: Modelo Conceitual das Entidades em comum

das informações relevantes a respeito dos alunos, professores e disciplinas) e registros de atividades (armazenamento de informações relacionadas com o acompanhamento dos alunos) e a camada tutor, com os elementos base de domínio (conteúdo a ser apresentado), estratégias didáticas (como aqueles conteúdos vão ser apresentados, utilizando táticas de ensino), perfil do aluno (para quem irá ser apresentado) e táticas de ensino (reuso de recursos, debate síncrono e envio de informações).

O Diagrama de Componentes da Figura 7.8 mostra a arquitetura de um Sistema Tutor Inteligente, utilizando os componentes: Estratégias Didáticas (o componente em que os professores montam suas estratégias de ensino para determinadas sessões de ensino); Base de Domínio (representa o conteúdo que será apresentado em uma sessão de ensino); Perfil do Aluno e Perfil do Grupo (dados sobre os alunos e grupos de determinadas sessões de ensino); e, o Controle (através do agente executor de sessão).

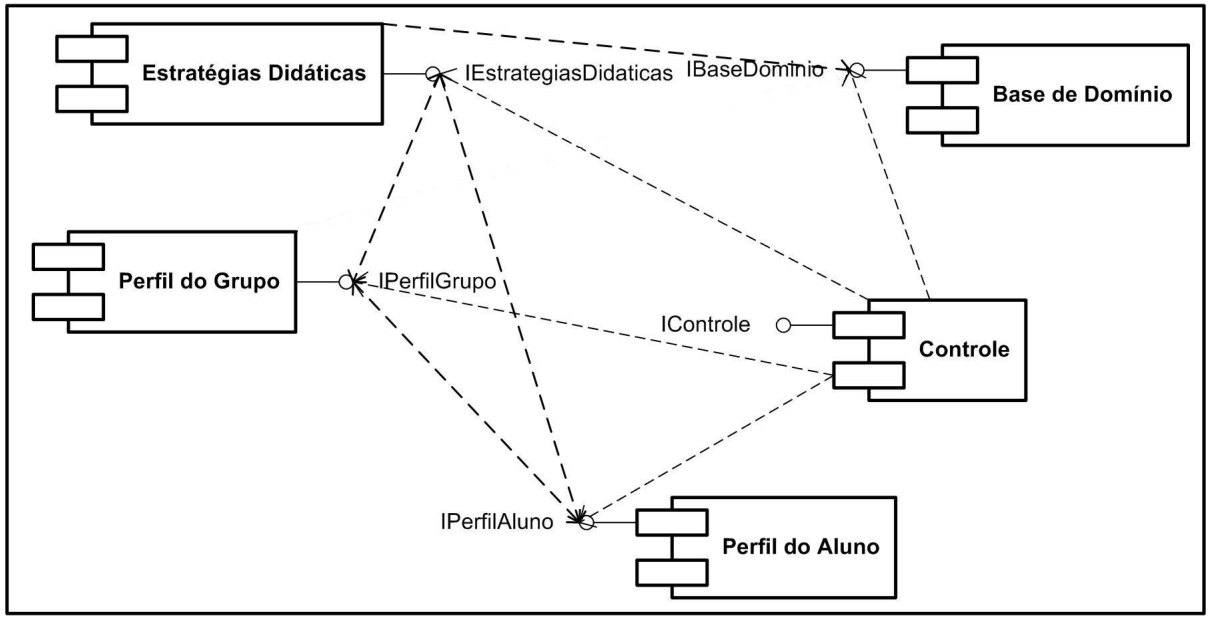


Figura 7.8: Diagrama de Componentes de um Sistema Tutor Inteligente

Depois de ter analisado os sistemas portfólio eletrônico, Portfólio-Tutor e o sistema tutor baseado em agentes, as camadas em comum para um sistema Portfólio-Tutor são: a camada de Apresentação, a camada de Agentes (a camada de agentes justifica-se porque o TUTA utiliza agentes reativos e principalmente devido a flexibilidade e a autonomia que os agentes possuem), a camada Tutor, a camada Portfólio Eletrônico e a camada de Serviços (Figura 7.9).

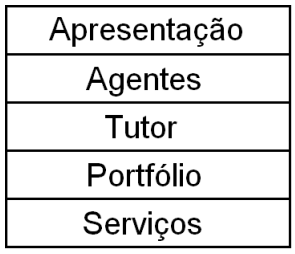


Figura 7.9: As Camadas de um Portfólio-Tutor

A Figura 7.10 mostra a identificação dos componentes nas seguintes camadas: Camada Interface, Camada Lógica da Aplicação (com as Camadas Domínio e Serviços) e a Camada Armazenamento.

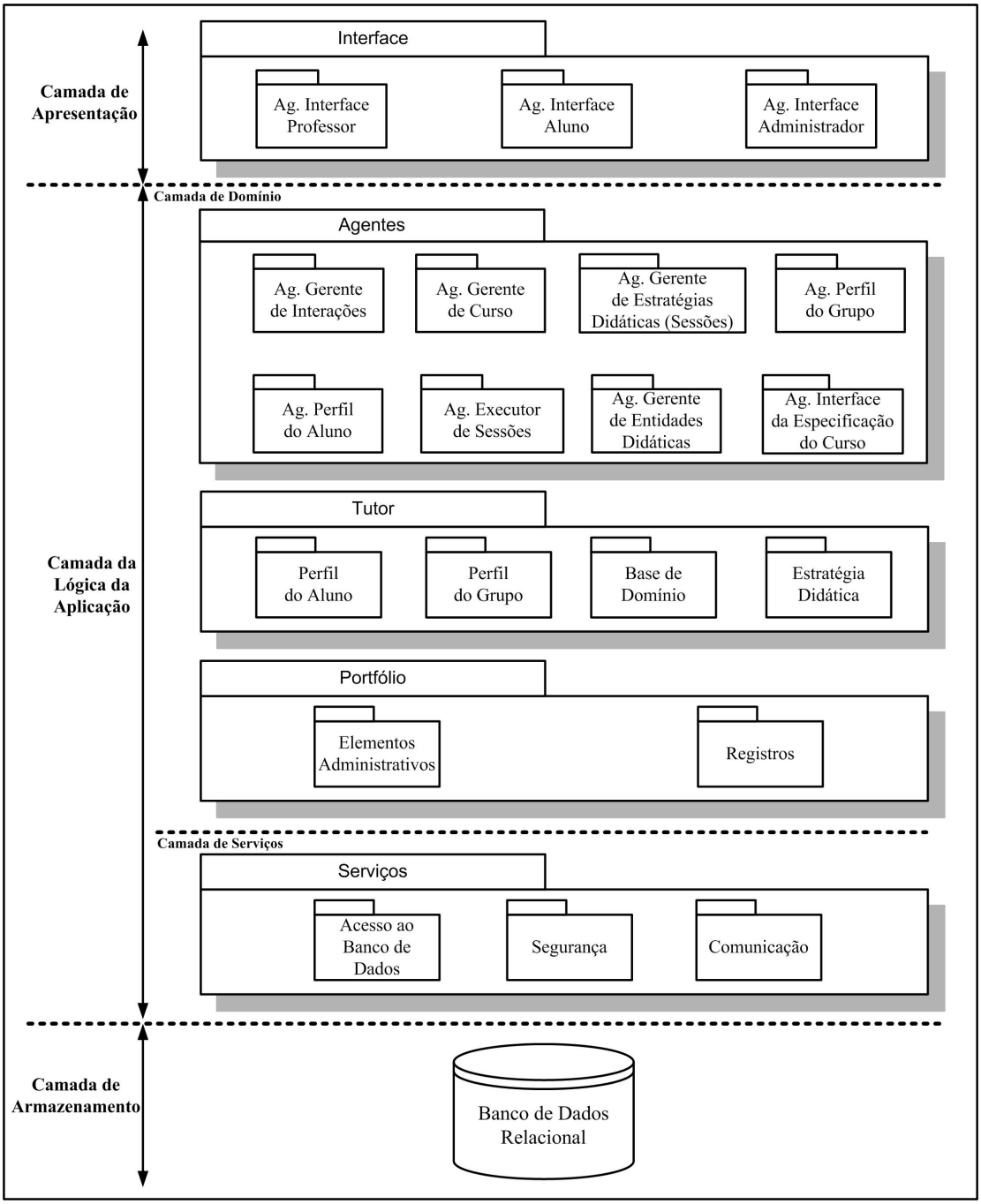


Figura 7.10: Componentes de um Portfólio-Tutor

Arquitetura Básica de uma Aplicação Portfólio-Tutor

A arquitetura de uma aplicação gerada a partir do framework FA_PorT é mostrada na Figura 7.11.

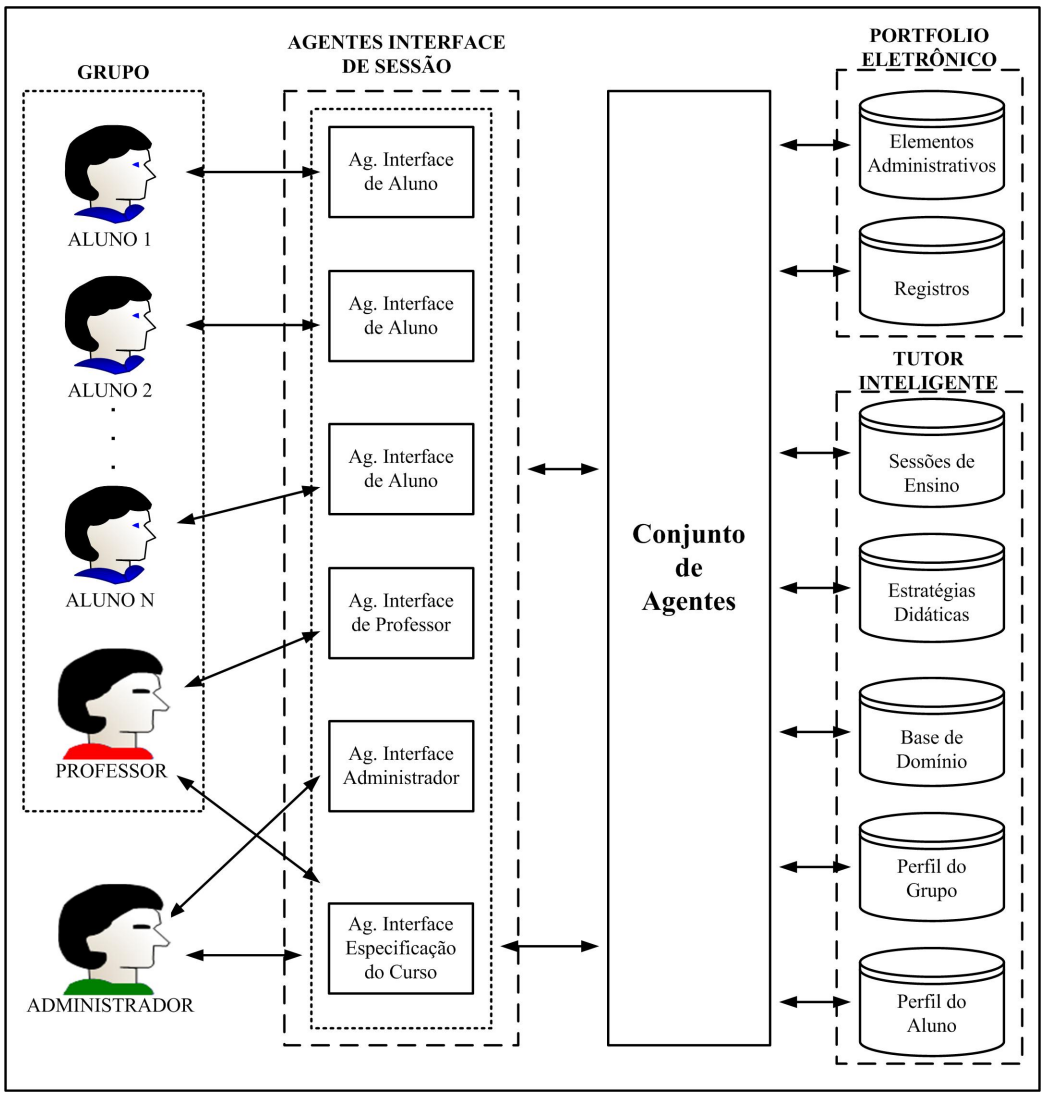


Figura 7.11: Arquitetura Básica de uma Aplicação Portfólio-Tutor gerada a partir do Framework FA_PorT

A Figura 7.11 apresenta os seguintes elementos:

- Aluno: pessoa que se interessa pelo conteúdo associado a um domínio;
- Professor: pessoa interessada em utilizar o sistema para auxiliar os alunos nas suas tarefas de aprendizagem;
- Administrador do Sistema: pessoa responsável pelo cadastro de professores no sistema;
- Grupo: conjunto de pessoas (alunos e professor) que utilizam o sistema em uma sessão de ensino;
- Agente Interface Aluno: representa a comunicação entre o aluno e o sistema proposto;

- Agente Interface Professor: representa a comunicação entre o professor e a aplicação;
- Agente Interface Especificação do Curso: representa a comunicação entre o professor e o sistema para especificar: sessões de ensino, estratégias didáticas e recursos didáticos que serão utilizados e também registrar os alunos que irão compor o curso;
- Agentes Interfaces Sessão: grupo dos agentes de interfaces que irão utilizar uma sessão;
- Conjunto de Agentes da Aplicação: representa a camada de agentes através de uma coleção de agentes que irão se comunicar entre si, com o objetivo de realizar uma sessão de ensino para os alunos.
- Elementos Administrativos: faz parte da camada Portfólio que possui os elementos administrativos do sistema;
- Registros: faz parte da camada Portfólio que armazena os registros do sistema;
- Sessões de Ensino: faz parte da camada Tutor que armazena as sessões de ensino de um curso;
- Estratégias Didáticas: faz parte da camada Tutor que armazena as estratégias didáticas a serem utilizadas nas sessões de ensino do curso;
- Base de Domínio: faz parte da camada Tutor que armazena e gerencia os recursos didáticos associados ao o conteúdo do curso;
- Perfil do Grupo: faz parte da camada Tutor que armazena os dados dos grupos e seus perfis;
- Perfil do Aluno: faz parte da camada Tutor que armazena os dados do aluno e seus perfis.

Funcionamento Geral de uma Aplicação Portfólio-Tutor

O Funcionamento Geral de uma Aplicação Portfólio-Tutor procede da seguinte forma, quando um professor quiser utilizar o sistema, ele deve contatar o Administrador para poder cadastrá-lo. O administrador do sistema depois de analisar sua solicitação, autoriza o professor a utilizá-lo, então o professor realiza a especificação do curso, sessões de ensino e estratégias didáticas a serem utilizadas e registra também os alunos que irão participar do curso, quem fica responsável por isso é o Agente Interface Especificação Curso. Então, o Agente Interface Especificação Curso, pedirá ao Agente Gerente de Entidade Didática que armazene os dados nas suas bases de dados (Elementos Administrativos, Registros, Perfil do Aluno, Estratégias Didáticas, Base de Domínio e Sessões de Ensino). Cada aluno

irá receber uma notificação por e-mail que haverá uma sessão com todos os participantes (Alunos e Professor). Os alunos utilizam os Agentes Interfaces de Sessão para participar da sessão de ensino.

O aluno poderá participar de sessões de ensino síncronas e sessões assíncronas, dependendo da sessão de ensino que foi atribuída em seu grupo. Nas sessões síncronas é o sistema que controla a sequência de conceitos e recursos didáticos a serem utilizadas, enquanto que nas sessões assíncronas, o aluno pode escolher qual conceito estudar e qual recurso didático utilizar. Depois da apresentação de todos os recursos didáticos relacionados a um conceito (sessão síncrona) ou depois de toda a navegação de todos os recursos obrigatórios, o aluno irá passar por uma avaliação de aprendizagem.

O professor pode em qualquer hora atribuir atividades aos alunos de acordo com os itens de avaliação determinados. O sistema acompanhará automaticamente os prazos das atividades a vencer, vencidas e a avaliar. Quando uma atividade é realizada, o aluno deve informar da sua execução ao sistema e disponibilizar o documento feito pelo por ele. Desta forma, o professor será informado automaticamente que existe uma atividade a ser avaliada. Todas as atividades que os alunos executam são acrescentadas ao seu portfólio de trabalho.

7.3 Arquitetura do Framework FA_PorT

Apresenta-se na Figura 7.12, a arquitetura do Framework FA_PorT e as camadas associadas a cada aplicação construída a partir do mesmo, em que os elementos das camadas (Interface, Agentes, Tutor, Portfólio Eletrônico e Serviços) de uma nova aplicação portfólio-tutor são representadas através de um conjunto de componentes organizados da seguinte maneira:

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

Cada componente possui sua interface. A classe *FrameworkPortfolioTutor* possui atributos representando as interfaces dos componentes do framework e os métodos, o método template que representa o controle e os pontos de adaptação de código. As aplicações são construídas considerando as camadas de um sistema portfólio-tutor (Interface, Agentes, Tutor, Portfólio e Serviços).

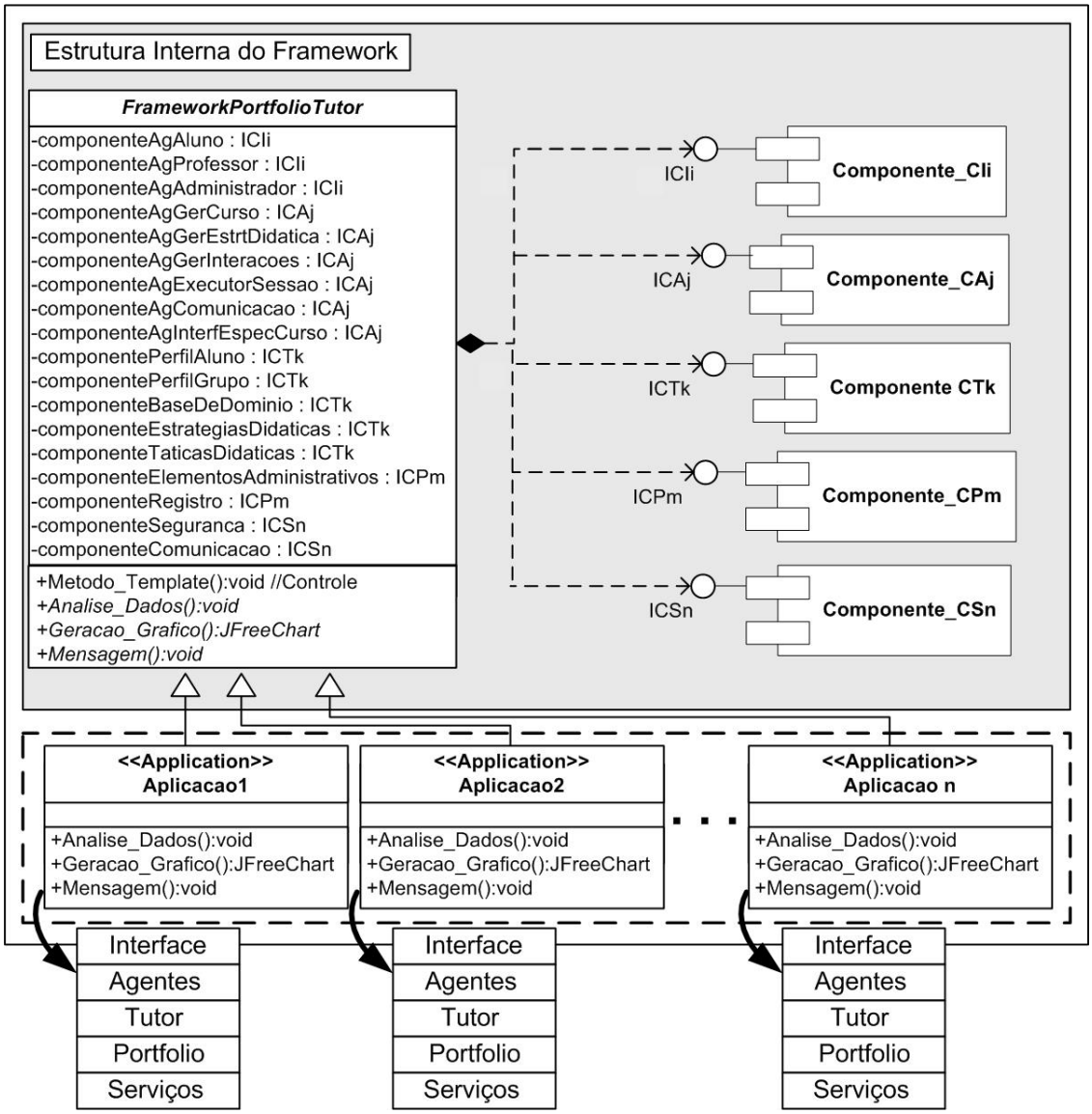


Figura 7.12: Arquitetura do framework FA_PorT para sistemas Portfólio-Tutor

7.4 Diagrama de Componentes

No diagrama de componentes, pode-se representar a estrutura do framework através de componentes e a interação entre os mesmos. Os componentes foram definidos levando em consideração as entidades em comum identificadas na caracterização do domínio (Figura 7.13).

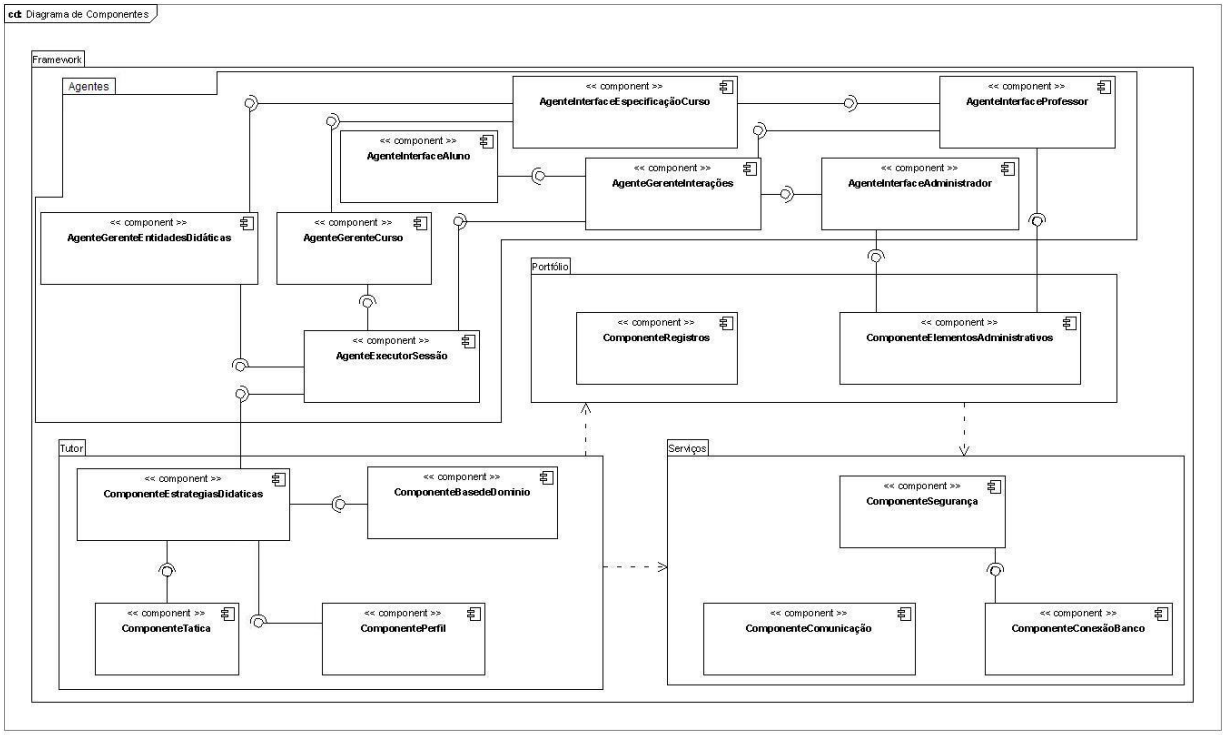


Figura 7.13: Diagrama de Componentes do Framework FA_PorT

7.5 Especificação dos Componentes e suas Interfaces

Cada componente possui classes e interface associadas, referentes ao projeto de cada componente, são eles:

- **Camada Agentes**: os diagramas de classes dos Agentes, que representam os componentes Agente Gerente de Curso, Agente Gerente de Estratégias Didáticas, Agente Perfil de Grupo e Agente Perfil de Aluno, são representados respectivamente nas Figuras 7.14, 7.15 e 7.16.

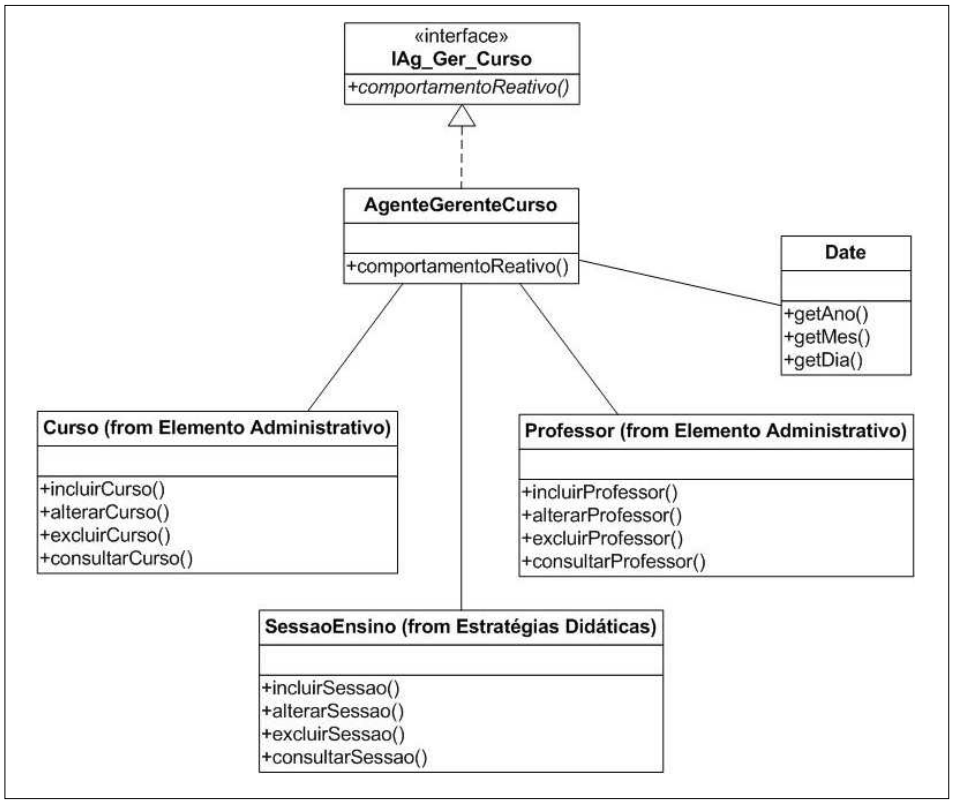


Figura 7.14: Diagrama de Classes do Componente de Agente Gerente de Curso

Na Figura 7.14 é mostrado o diagrama de classes do componente agente gerente de curso, onde possui a classe *AgenteGerenteCurso* que implementa o serviço *comportamentoReativo* oferecido pela interface *IAg_Ger_Curso*. A classe *AgenteGerenteCurso* também utiliza outras classes, como: *Date*, *Professor* e *Curso* (do componente Elemento Administrativo) e *SessaoEnsino* (do componente Estratégia Didática).

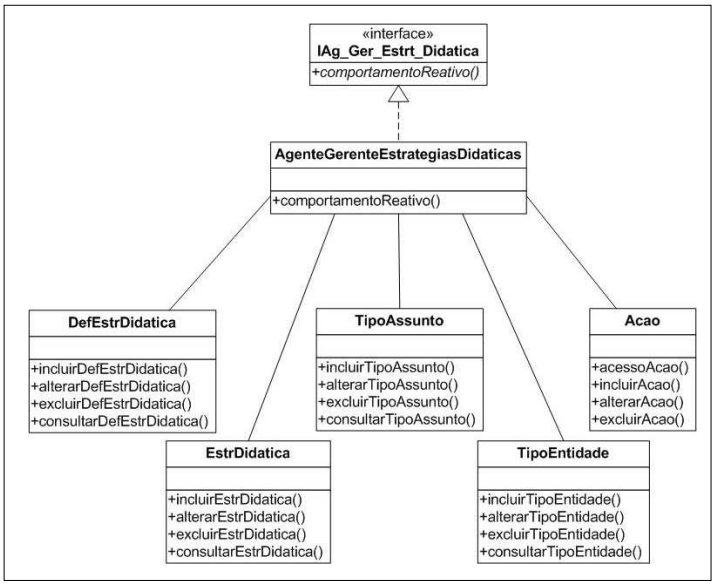


Figura 7.15: Diagrama de Classes do Componente de Agente Gerente de Estratégias Didáticas

Na Figura 7.15 mostra-se o diagrama de classes do componente agente gerente de estratégias didáticas, responsável por gerenciar todas as estratégias didáticas do sistema, possui como classe principal a *AgenteGerenteEstrategiaDidatica* que implementa o serviço *comportamentoReativo* oferecido pela interface *IAg_ Ger_ Estrt_ Didatica*. A classe principal também utiliza as classes *EstrDidatica*, *TipoAssunto*, dentre outras.

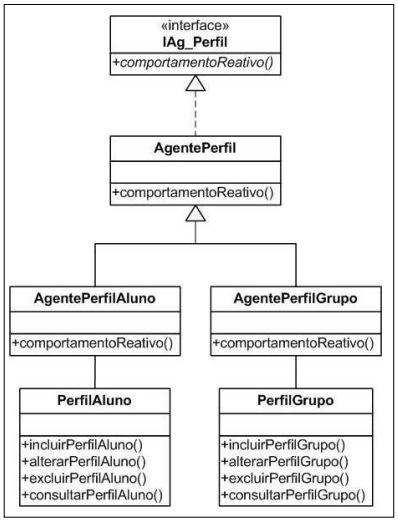


Figura 7.16: Diagrama de Classes do Componente de Agente Perfil do Grupo e do Aluno

A Figura 7.16 possui o diagrama de classes do componente agente perfil do grupo e do aluno, responsável por gerenciar todos os perfis de um sistema Portfólio-Tutor. A classe principal é a *AgentePerfil* que implementa o serviço *comportamentoReativo* oferecido pela interface *IAg_ Perfil*. O diagrama também possui as classes *AgentePerfilAluno* e *AgentePerfilGrupo*.

- **Camada Tutor**: possui os componentes Perfil do Aluno e do Grupo (Figura 7.17), o componente Base de Domínio (Figura 7.18), o componente Estratégia Didática (Figura 7.19) e o componente Táticas de Ensino (Figura 7.20).

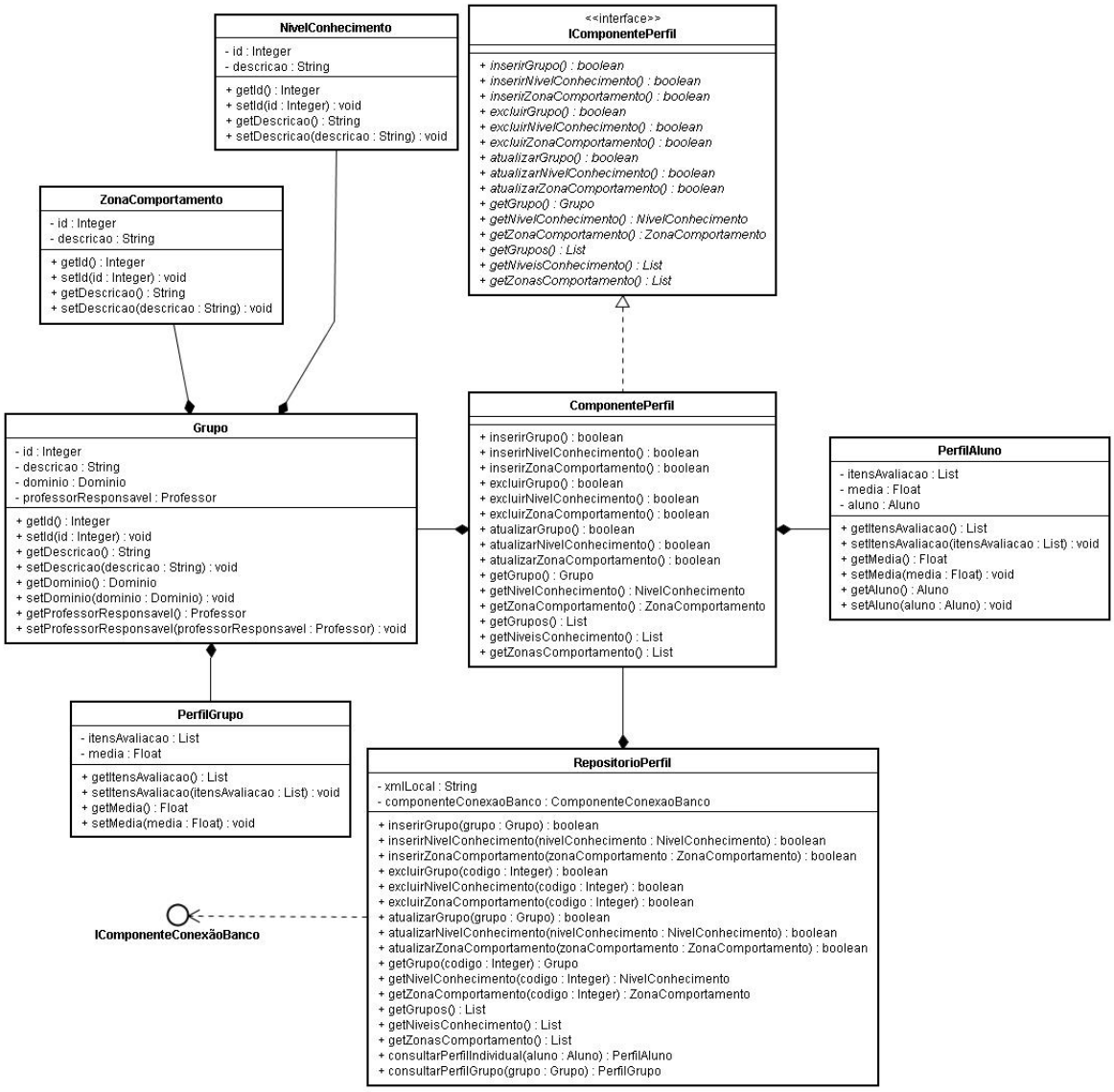


Figura 7.17: Diagrama de Classes do Componente Perfil do Aluno e do Grupo

Na Figura 7.17 temos o diagrama de classes do componente perfil do aluno e do grupo, a classe principal é *ComponentePerfil*, que relaciona-se com as classes *PerfilAluno* e *Grupo*, implementa os serviços da interface *IComponentePerfil*. A classe Grupo relaciona-se com *PerfilGrupo*, *NivelConhecimento* e *ZonaComportamento*.

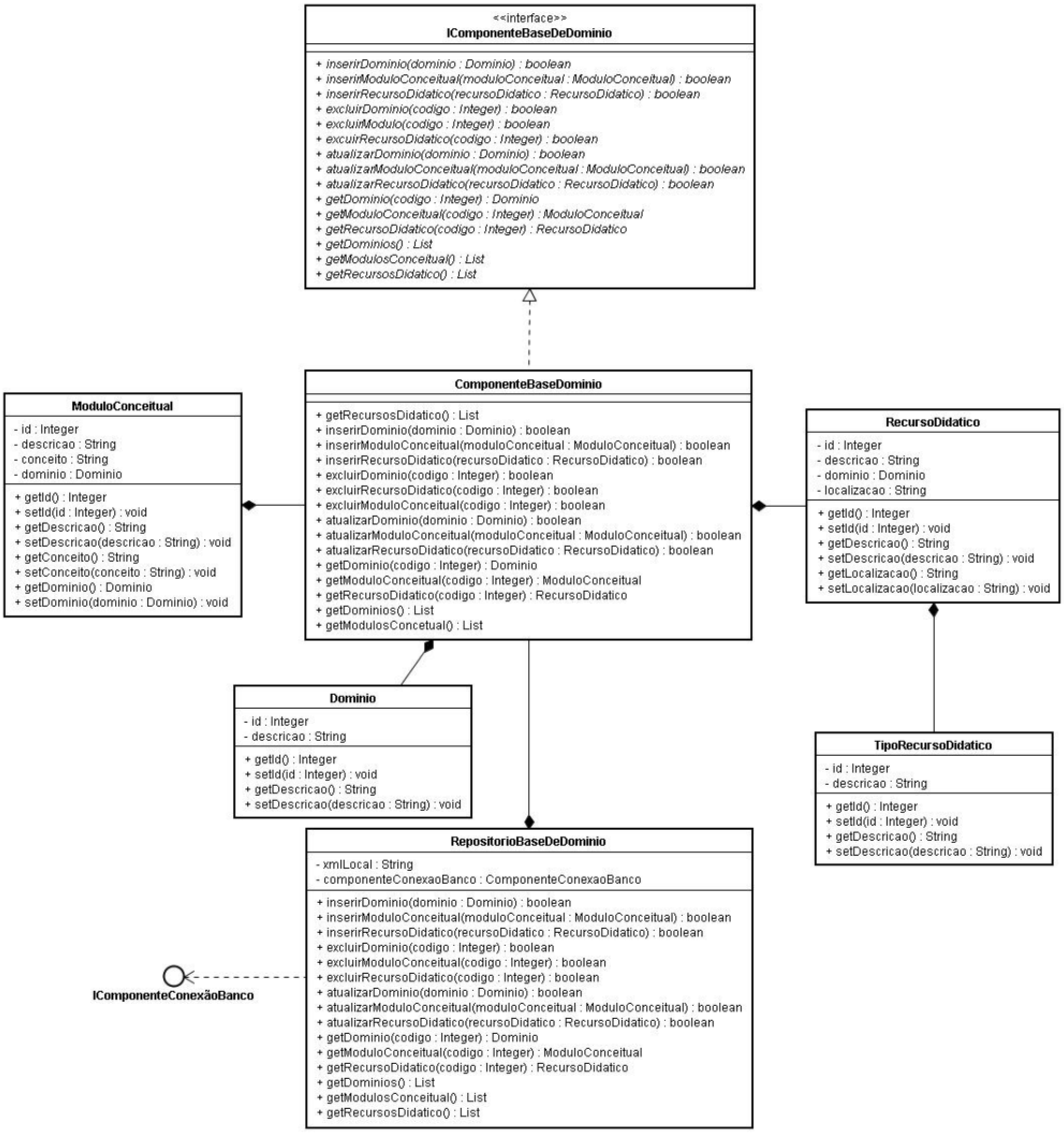


Figura 7.18: Diagrama de Classes do Componente Base de Domínio

A Figura 7.18 mostra o diagrama de classes do componente base de domínio, a classe principal é *ComponenteBaseDominio*, que relaciona-se com as classes *RecursoDidatico* (que possui a classe *TipoRecursoDidatico* como relacionamento) e *Dominio* e implementa os serviços da interface *IComponenteBasedeDominio*.

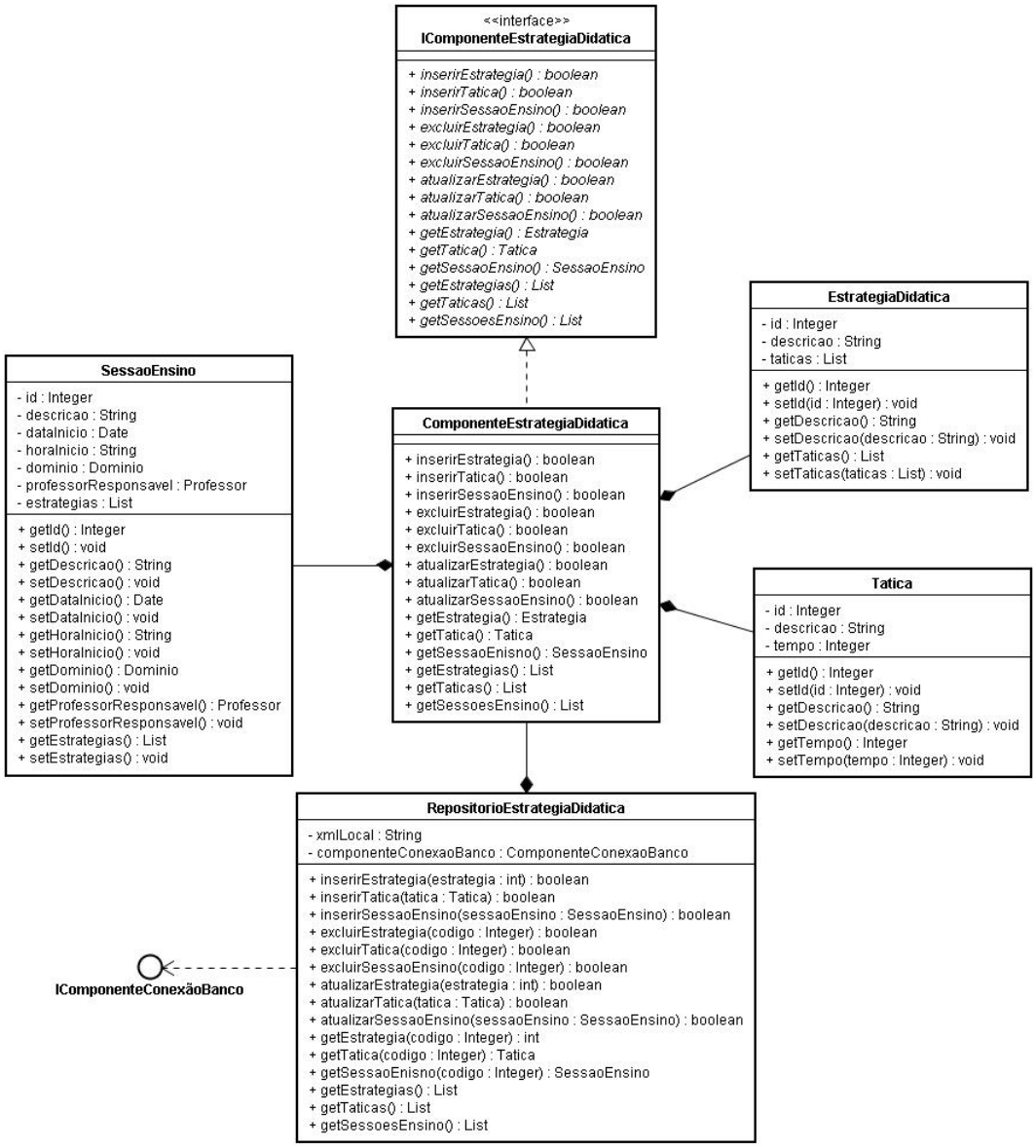


Figura 7.19: Diagrama de Classes do Componente Estratégia Didática

Na Figura 7.19 temos o diagrama de classes do componente estratégia didática, a principal classe deste componente é *ComponenteEstrategiaDidatica*, que possui relacionamento com as classes *SessaoEnsino*, *EstrategiaDidatica* e *Tatica* e implementa os serviços da interface *IComponenteEstrategiaDidatica*.

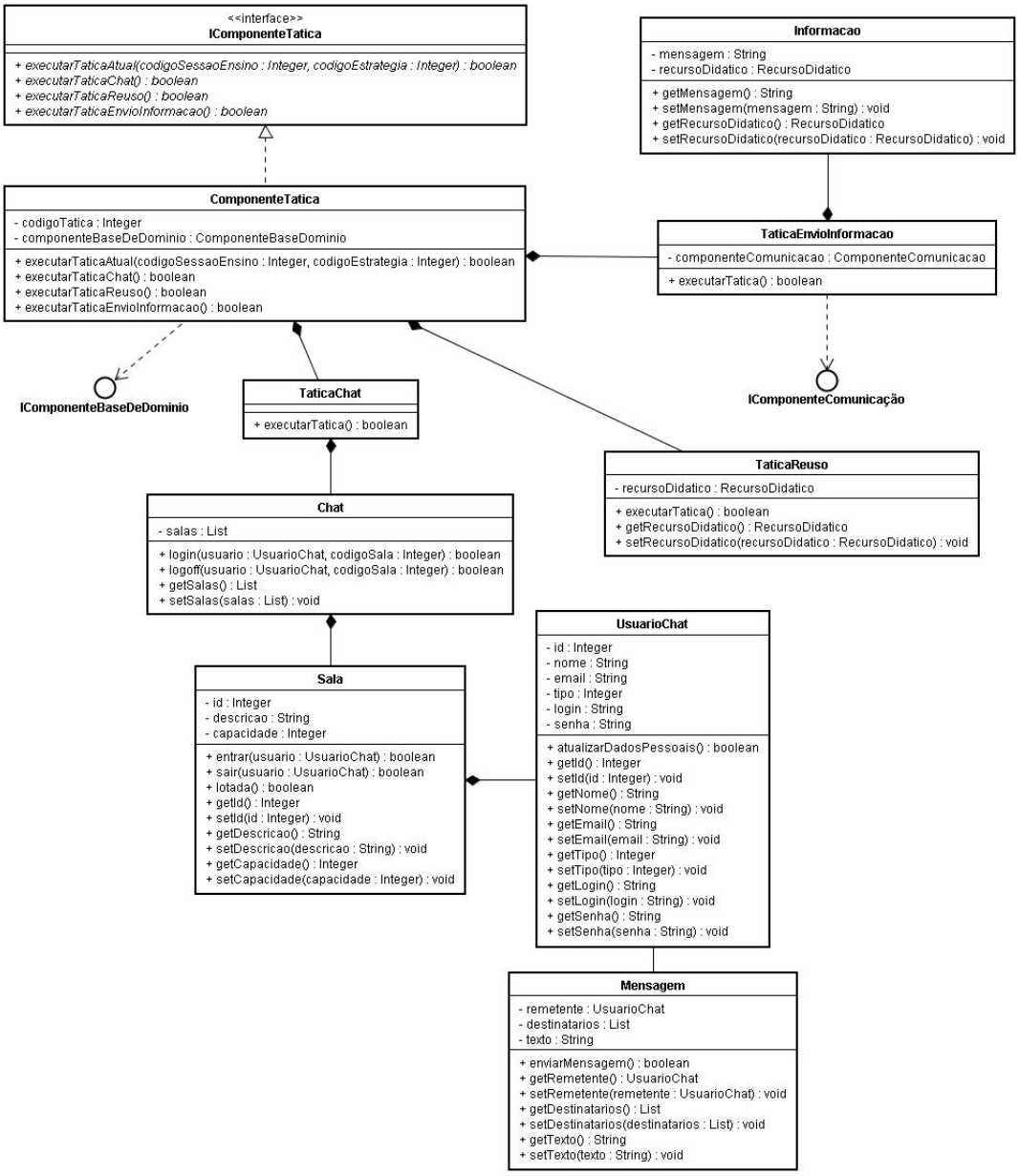


Figura 7.20: Diagrama de Classes do Componente Táticas de Ensino

No diagrama de classes do componente táticas de ensino (Figura 7.20), temos como classe principal *ComponenteTatica* que relaciona-se com as classes *TaticaReuso*, *TaticaChat* e *TaticaEnvioInformacoes*, implementa os serviços da interface *IComponenteTatica*.

- **Camada Portfólio Eletrônico**: possui os componentes Elementos Administrativos mostrado na Figura 7.21 e o Registros, na Figura 7.22.

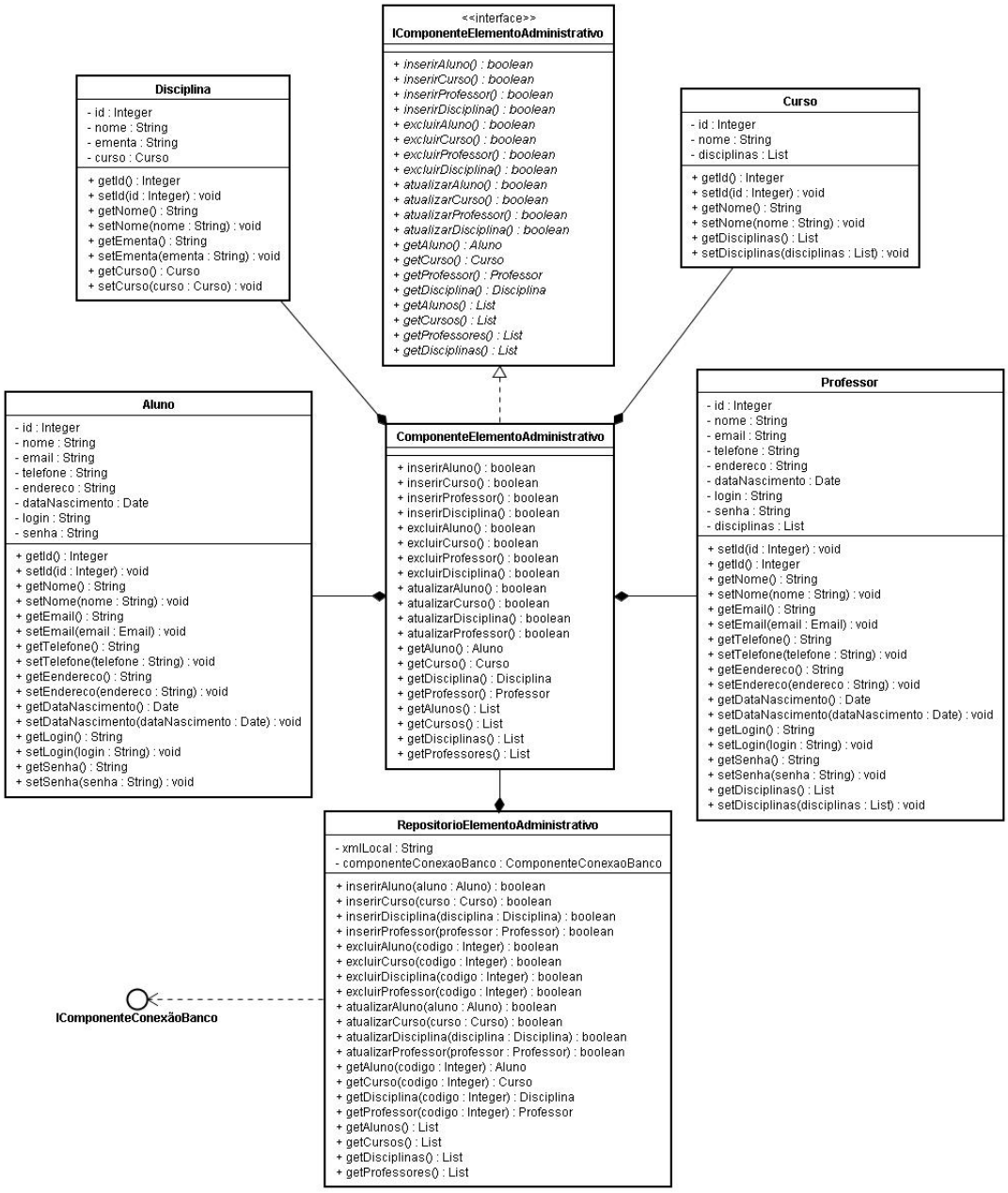


Figura 7.21: Diagrama de Classes do Componente Elementos Administrativos

A Figura 7.21 mostra o diagrama de classes do componente Elementos Administrativos, possui como principal classe *ComponenteElementoAdministrativo* que implementa a interface *IComponenteElementoAdministrativo* e relaciona-se com as classes *Aluno*, *Professor*, *Disciplina* e *Curso*. A classe *RepositorioElementoAdministrativo* possui relacionamento com a classe principal.

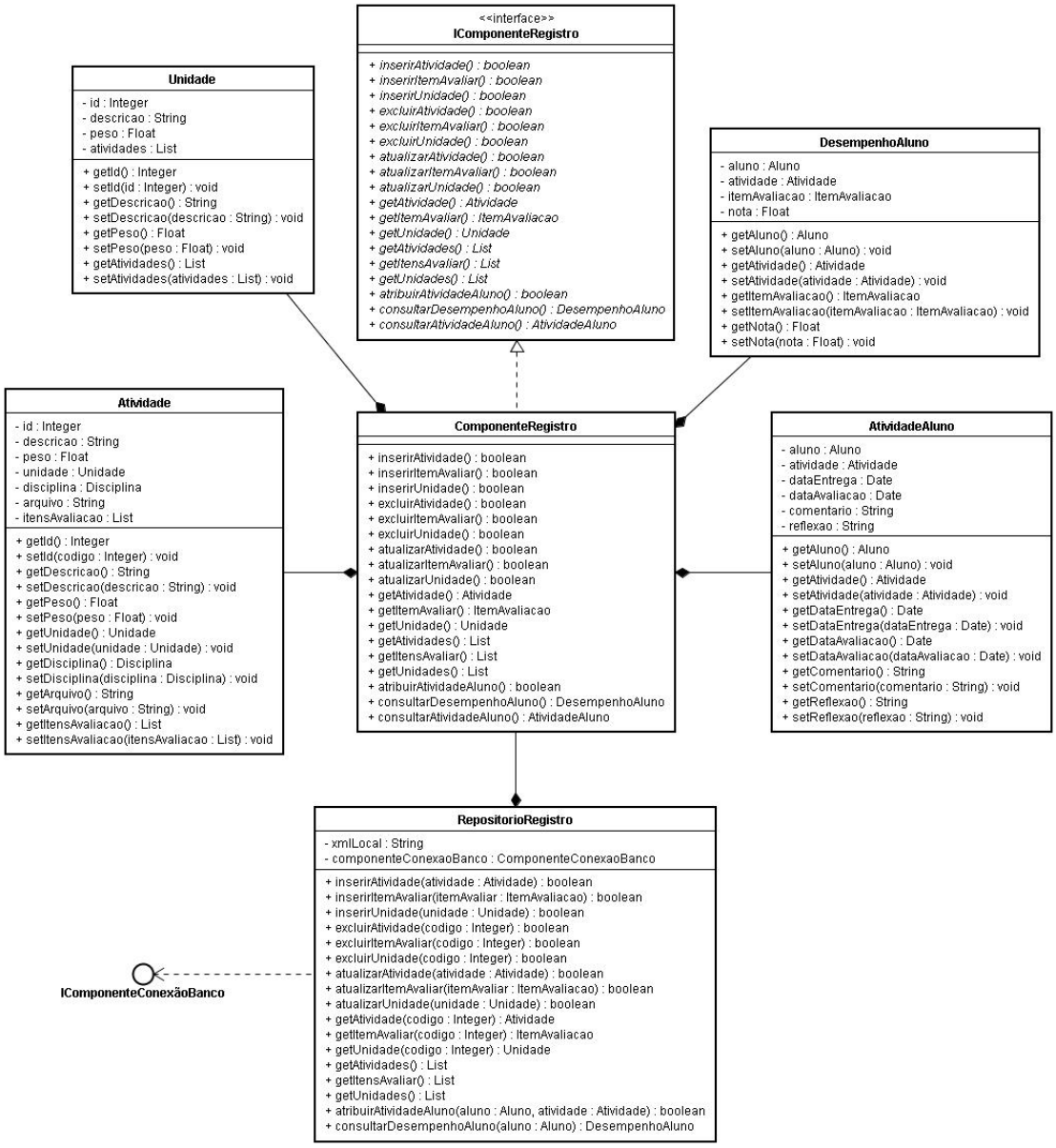


Figura 7.22: Diagrama de Classes do Componente Registros

Na Figura 7.22 temos o diagrama de classes do componente registros, a principal classe deste componente é *ComponenteRegistro*, que implementa a interface *IComponenteRegistro*, e possui relacionamento com as classes *Atividade*, *AtividadeAluno*, *DesempenhoAluno* e *Unidade*. A classe *RepositorioRegistro* possui relacionamento com a classe principal.

- **Camada Serviços**: a camada de serviços possui os componentes Acesso ao Banco de Dados (Figura 7.23), responsável pela conexão com o banco de dados e o componente Comunicação (Figura 7.24), que possuem os elementos que tratam do envio eletrônico de mensagens, transferência de arquivos e comunicação.

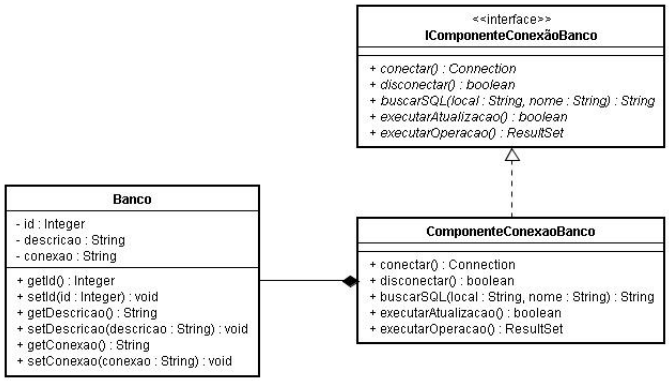


Figura 7.23: Diagrama de Classes do Componente Acesso ao Banco de Dados

Na Figura 7.23 temos o diagrama de classes do componente acesso ao banco de dados, a classe *ComponenteConexaoBanco* é a principal classe que possui relacionamento com a classe *Banco* e implementa os serviços da interface *IComponenteConexaoBanco*.

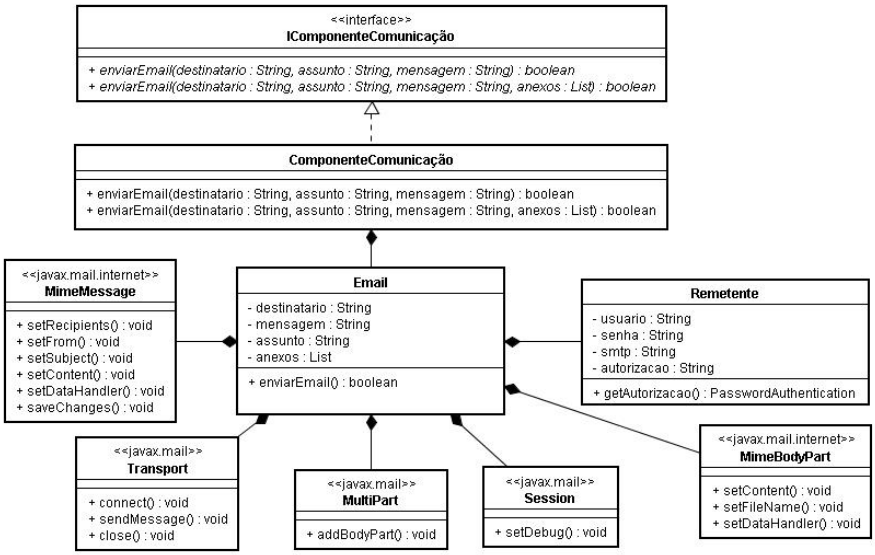


Figura 7.24: Diagrama de Classes do Componente Comunicação

A Figura 7.24 mostra o diagrama de classes do componente comunicação, a classe principal deste componente é *ComponenteConexao* que possui relacionamento com a classe *Email* (possui como relacionamento as classes *Transport*, *MimeMessage*, *MultiPart*, *Session*, *MimeBodyPart* e *Remetente*) e implementa os serviços da interface *IComponenteComunicacao*.

7.6 O Controle

O controle se dá através do padrão de projeto *Template Method*, que define o esqueleto de um algoritmo em uma operação, postergando alguns passos para subclasses. O *Template Method* permite que subclasses redefinam certos passos de um algoritmo sem mudar a estrutura do mesmo.

No controle associado a cada sistema portfólio-tutor, deve-se levar em conta o comportamento pró-ativo da camada portfólio e a realização de sessões de ensino gerenciadas pela camada tutor.

A seguir no código 7.1, é definido o controle através da utilização do padrão *Template Method* para o framework que está sendo desenvolvido. O código java associado ao controle do framework é apresentado no Apêndice C

Código 7.1: Controle do Framework

```
1  //Classe Framework que deve ser reutilizada.
2  public abstract class Framework {
3
4      // Ponto de Adaptacao de Codigo Cuztomize Grafico (Geracao de
5      // Graficos em Barra e em Pizza).
6      public abstract JFreeChart customizeGrafico(int aprovados,
7          int reprovados, List desempenhos);
8
9      // Ponto de Adaptacao de Codigo Cuztomize Comunicacao de
10     // Alunos (Avisos por Email ou no Sistema).
11     public abstract void comunicacaoAluno(Integer codigoAluno,
12         Integer codigoAtividade, String mensagem);
13
14     // Codigo associado ao metodo template ou fluxo de controle
15     // do framework FA_PorT.
16     public void templateMethod(){
17         // Consultar o codigo detalhado no Apendice B.
18         . . .
19     }
20 }
```

7.6.1 Pontos de Adaptação

Um framework é uma arquitetura de software reutilizável em termos de contratos de colaboração entre classes abstratas e um conjunto de pontos adaptáveis ou *hot spots*.

Uma aplicação criada a partir do framework pode ser "customizada" através de hotspots. *Hot Spots* consistem em pontos adaptáveis que precisam ser especializados em uma aplicação (Fayad et al. 1999).

A especificação dos pontos de adaptação de código utilizando a notação UML-F[1] do framework FA_PorT, são mostrados na Figura 7.25. Os pontos de adaptação considerados no framework FA_PorT são: a análise dos dados[2], a geração de gráficos e a mensagem. A definição desses pontos de adaptação na classe *FrameworkPortfolioTutor* podem ser consultados no código do controle (Código 7.1) e a implementação fornecida pelas aplicações podem ser consultadas no Apêndice C.

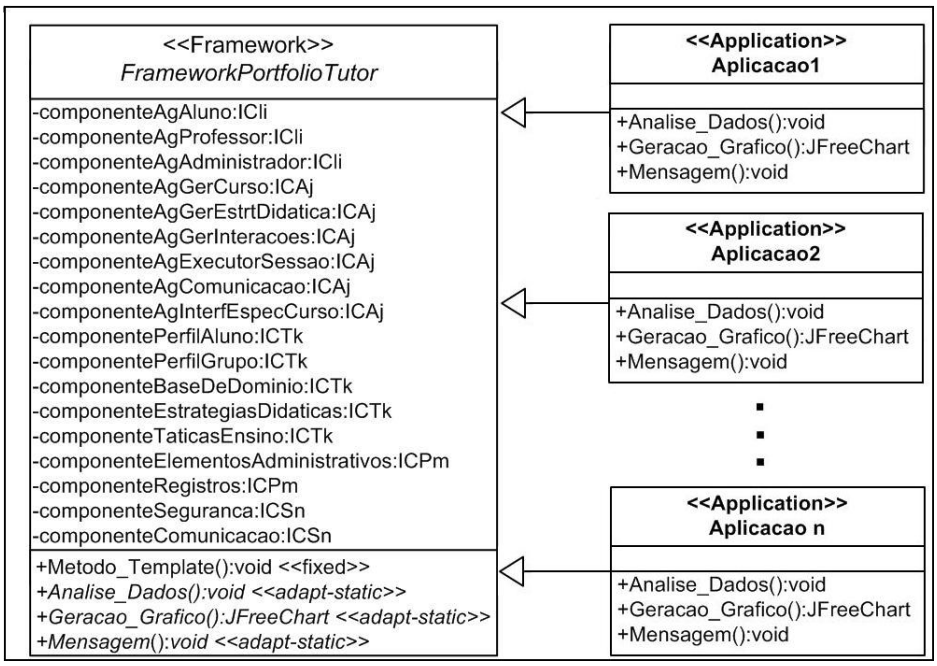


Figura 7.25: Especificação do framework FA_PorT utilizando UML-F

De acordo com a Figura 7.25, na aplicação 1 temos a classe Aplicacao1 que implementa os pontos de adaptação de código e na aplicação 2, o nome da classe é Aplicacao2. Abaixo temos a construção das duas aplicações portfólio-tutor geradas a partir do framework FA_PorT.

- **Aplicação 1**

Código 7.2: Controle da Aplicação 1

```
1 // Construindo a aplicacao 1 derivada do framework FA_PorT
2 public class Aplicacao1 extends Framework{
3
```

[1]Uma extensão da UML (*Unified Modelling Language*) específica para a modelagem de frameworks orientados a objetos (Fontoura et al. 2000).

[2]A análise dos dados ainda não foi implementado na versão atual do framework.

```
4       // Metodo customize - mostra mensagens aos alunos a respeito
5       // de suas atividades, que sao enviadas por email.
6       // Ver Apendice B.
7       public void comunicacaoAluno(integer codigoAluno,
8           Integer codigoAtividade, String mensagem){
9           ...
10      }
11
12      // Metodo customize - grafico de barras para representar os
13      // desempenhos dos alunos. Ver Apendice B.
14      public JFreeChart customizeGrafico (int aprovados,
15          int reprovados, List desempenhos) {
16          ...
17      }
18  }
```

- **Aplicação 2**

Código 7.3: Controle da Aplicação 2

```
1   // Construindo a aplicacao 2 derivada do framework FA_PorT
2   public class Aplicacao2 extends Framework{
3
4       // Metodo customize - mostra mensagens aos alunos a respeito
5       // de suas atividades, que sao enviadas por email.
6       // Ver Apendice B.
7       public void comunicacaoAluno(integer codigoAluno,
8           Integer codigoAtividade, String mensagem){
9           ...
10      }
11
12      // Metodo customize - grafico de barras para representar os
13      // desempenhos dos alunos. Ver Apendice B.
14      public JFreeChart customizeGrafico (int aprovados,
15          int reprovados, List desempenhos) {
16          ...
17      }
18  }
```

7.7 Funcionamento de um novo Sistema Portfólio-Tutor

Um novo sistema portfólio-tutor é construído a partir do uso do framework FA_PorT. Cada novo sistema portfólio-tutor (ou aplicação) terá um funcionamento pré-definido. Este é especificado no framework através do controle ou funcionalidades de cada sistema portfólio-tutor.

7.7.1 Camada Tutor

Uma sessão de ensino on-line associada a um novo sistema portfólio-tutor é especificada para a aprendizagem de um grupo virtual específico de alunos e é iniciada quando a camada tutor inicia uma estratégia (Silva 2000). Uma estratégia é representada por um conjunto de táticas. As táticas de ensino consideradas são:

- **Tática de Reutilização**: indica que será apresentado algum recurso didático de ensino (definições, exemplos, exercícios, estudos de caso, etc);
  - Reuso (recurso, tempo): indica que deve-se mostrar o conteúdo de um recurso por um período de tempo (em minutos).
- **Tática de Debate Síncrono**: representa um "chat" ou "bate-papo", onde os alunos do grupo podem interagir com o professor ou com outros alunos;
  - DebateSincrono (alunos, professor, tempo): este elemento é do tipo chat e será utilizado por um grupo de alunos e um professor em um período de tempo (em minutos).
- **Tática de Envio de Informação**: permite o envio de informações por e-mail.
  - EnvioInformacao (lista de informações, lista de email dos destinatários): este elemento representa o envio de informação, indicando que deve-se enviar uma lista de informações por email para um grupo de alunos.
- **Tática de Mudança de Estratégia**: permite mudança de estratégia didática.
  - MudarEstrategia (nova estratégia): este elemento permite a mudança para uma nova estratégia didática.
- **Tática de Regras**: Define uma seleção, onde as ações serão executadas dependendo da condição. As regras são criadas da seguinte forma:

  Se (condição) então
  Ação (ões)

Fim_Se

A execução de uma estratégia é mostrada na Figura 7.26.

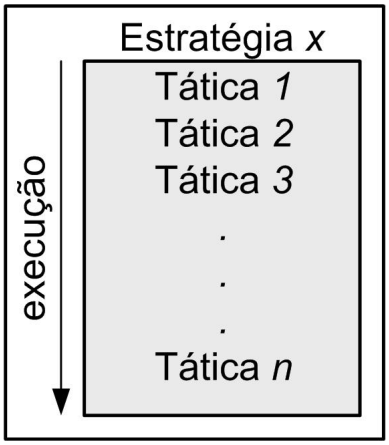


Figura 7.26: Execução de uma Estratégia

7.7.2 Camada Portfólio Eletrônico

As principais funcionalidades da camada portfólio eletrônico no sistema portfólio-tutor são (Nascimento 2002) analisar o progresso do aluno; identificar e armazenar os trabalhos elaborados por cada aluno; registrar os resultados (avaliação e comentários do professor); fornecer suporte para o monitoramento da performance dos alunos e dos grupos; através da pró-atividade no portfólio eletrônico, o sistema poderá enviar mensagens para os alunos e professores através do email, informando, por exemplo, sobre datas de atividades que serão realizadas, prazos que se vencerão, dentre outros; gerar, de forma pró-ativa, gráficos e relatórios associados a performance e ao progresso do grupo dos alunos (durante as atividades).

7.7.3 Comportamento do Aluno

O "modelo do aluno" proposto no sistema portfólio-tutor é representado pelo perfil do grupo, zona de comportamento, perfil individual e histórico de navegação. O perfil do grupo corresponde ao nível de conhecimento do aluno, como exemplo básico, intermediário ou avançado, de acordo com os conceitos que serão aprendidos, enquanto que em uma sessão de ensino, dependendo de seu desempenho, os alunos irão fazer parte das zonas de comportamento. Estas são consideradas como zona crítica inferior, zona intermediária inferior, zona normal, zona intermediária superior e zona crítica superior. O critério para fazer parte de uma zona de comportamento é o desempenho registrado do aluno durante as sessões de ensino (Nascimento 2002).

O perfil individual do aluno permite acompanhar a aprendizagem do aluno e registrar todos os conceitos aprendidos. Assim como o histórico de navegação que registra todos os recursos didáticos utilizados pelo aluno.

7.8 Construção de aplicações a partir do uso do framework FA_PorT

A geração de uma aplicação ocorre a partir da produção de novas classes e da utilização de classes presentes no framework, que podem ser vistas a partir da arquitetura do FA_PorT.

Para se construir uma aplicação a partir do uso do Framework, o desenvolvedor de aplicações deve construir as classes da parte variante de uma aplicação.

Estas classes herdam da classe abstrata *FrameworkPortfolioTutor*, que está contida no pacote FA_PorT (Figura 7.25). A classe *FrameworkPortfolioTutor* contém os Hot Spots (pontos adaptáveis) do sistema. Através deles que o framework FA_PorT poderá ser customizado para criar novas aplicações.

Assim, para que o desenvolvedor de aplicações consiga implementar a aplicação que deseja, deverá implementar os métodos abstratos de acordo com a funcionalidade que o seu sistema necessita, e a aplicação criada terá a arquitetura de um Portfólio-Tutor.

7.9 Conclusões

Os modelos elaborados neste capítulo, permitem visualizar o framework FA_PorT sob duas perspectivas, a estrutural e a comportamental.

Para a estrutural, foram empregados os diagramas de classes que ilustram a visão estática do framework. Sendo que, de acordo com as responsabilidades assumidas, as classes foram agrupadas em componentes (diagrama de componentes) através de camadas e partições, gerando uma arquitetura em múltiplas camadas.

Já os aspectos dinâmicos do framework, isto é, a visão comportamental, foram representados através do detalhamento do controle do framework e do funcionamento de um novo sistema Portfólio-Tutor.

Estas visões em conjunto capturam as decisões mais importantes sobre o framework e suas aplicações e serviram de base para a construção do framework e de dois protótipos de aplicações Portfólio-Tutor, detalhados no próximo capítulo.

O próximo capítulo apresenta a Implementação do framework e as duas aplicações construídas a partir do FA_PorT.

Parte III

Implementação do FA_PorT

Capítulo 8

Implementação do FA_PorT

No capítulo 8, é descrita a implementação do framework FA_PorT. A seção 8.1 apresenta a passagem da fase de projeto orientado a agentes para uma implementação baseada no reuso de software. Na seção 8.2, fala-se sobre a linguagem e as ferramentas utilizadas e na seção 8.3, mostra-se as duas aplicações criadas a partir do framework.

8.1 Passagem da Fase de Projeto Orientado a Agentes para uma Implementação baseada no Reuso

Para o projeto Orientado a Agentes, foi utilizado a metodologia GAIA, e, como esta metodologia não é completa, só possui duas fases, a fase de análise e a fase de projeto, esta passagem se deu da seguinte forma: depois de concluir a modelagem orientada a agentes, abstraímos os agentes em forma de componentes para depois podermos implementá-los. Essa transição podemos ver com detalhes na Figura 8.1.

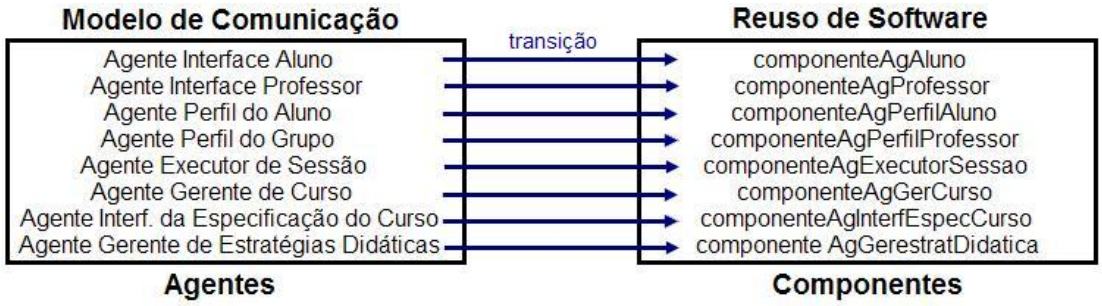


Figura 8.1: Transição da fase de projeto para a implementação dos agentes

A Figura 8.1 mostra os agentes criados na metodologia GAIA e a transição destes agentes para a sua implementação através de componentes de software.

A principal dificuldade encontrada para efetuar essa passagem foi representar o comportamento de um agente nos componentes.

8.2 Linguagem e Ferramentas Utilizadas

O framework FA_PorT foi desenvolvido para a criação de aplicações Web, onde o acesso às informações se dá via um navegador (*browser*).

Java foi a linguagem utilizada para implementação[1] do framework FA_PorT e das duas aplicações geradas a partir dele. Java é uma linguagem de programação orientada a objetos bastante poderosa, devido as suas principais características:

- **Simples**: não contém redundâncias e é fácil de entender, implementar e usar;
- **Orientada a objetos**: suporta os principais conceitos de orientação a objetos, tais como objeto, classe, polimorfismo, herança, encapsulamento, entre outros. Favorece extensabilidade e reusabilidade;
- **Robusta**: fortemente tipada. Programas são confiáveis, reduz imprevistos em tempo de execução: variáveis são automaticamente inicializadas, uso disciplinado de ponteiros, rotinas devem ser chamadas corretamente, etc;
- **Portável**: aplicações funcionam da mesma forma em qualquer ambiente. Não contém aspectos dependentes da implementação: o tamanho dos tipos é fixo para qualquer implementação, etc;
- **Segura**: restrições de acesso a arquivos (applets), manipulação de ponteiros, etc. Implica que não é útil para desenvolver certas aplicações como device drivers, etc;
- **Concorrente**: suporta aplicações concorrentes como multithreads e monitores.

Foi empregada a tecnologia, fundamentada na linguagem Java, chamada *Java Server Pages* (JSP). Esta simplifica o processo de desenvolvimento de aplicações Web, devido a separação entre a camada lógica da aplicação e a camada de apresentação (HTML[2]) (Fields & Kolb 2000). Estes arquivos de texto contém HTML com código Java embutido através da utilização de marcações JSP (Figura 8.2). As características gerais do HTML são documentos que escritos em *American Standard Code for Information Interchange* (ASCII) que podem ser criados em qualquer editor de texto. Existem editores específicos para cada plataforma e conversores para cada formato, por exemplo, doc para html, etc.

[1] O ambiente utilizado para a implementação foi o eclipse.

[2] *HyperText Markup Language* (linguagem de marcação de hipertexto). É a linguagem de formatação mais utilizada nos documentos publicados na Internet, sendo capaz de dar formato a textos, imagens, sons e vídeos e, principalmente, de vincular diferentes tipos de arquivos, por meio de links.

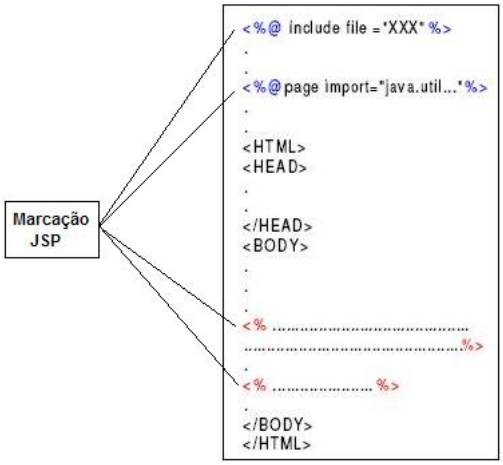


Figura 8.2: Arquivo JSP

O arquivo JSP é executado em um container[3], que é instalado em um servidor Web ou servidor de aplicações. O container envia a requisição do cliente ao JSP e a resposta do JSP ao cliente.

O pré-processador do container cria uma classe Java contendo toda a funcionalidade do JSP. A partir desse momento, o JSP passa a ser uma extensão do container, sendo executado no mesmo ambiente e compartilhando recursos com ele e com outros JSPs e servlets que estiverem no mesmo container (Fields & Kolb 2000). A Figura 8.3 ilustra o funcionamento de uma página JSP.

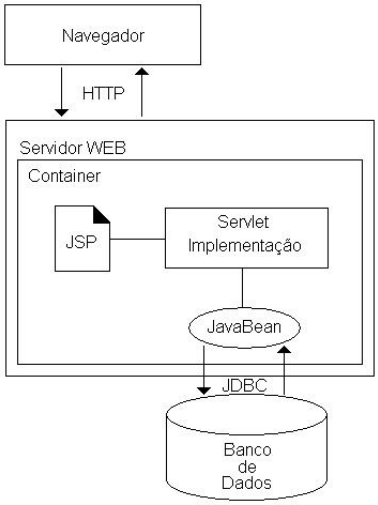


Figura 8.3: Funcionamento de Páginas JSP

Para poder organizar, facilitar e possuir maior facilidade de implementação em sistemas para Web, foi utilizado o framework webwork, que usa o padrão *Model-View-*

[3]É um componente, que corresponde a implementação de uma funcionalidade específica, podendo ser independentemente desenvolvido e distribuído como uma unidade, como também pode possuir uma interface bem definida para os serviços disponibilizados, que pode conter outros componentes.

Controler (MVC[4]), fazendo com que o desenvolvimento e a manutenção do sistema se tornem muito mais fáceis. O framework webwork faz o papel do controlador, deixando para o programador as tarefas relativas as regras de negócio e a camada de visualização (Lightbody & Carreira 2005). Alguns recursos que estão disponíveis no webwork são:

- **Ações**: o gerenciamento de ações ou de comandos é a característica mais básica do WebWork. Descrevendo uma ação no arquivo de configuração e criando uma classe Java para esta ação é o suficiente para ver o WebWork em funcionamento;
- **Redirecionamento**: como resultado de uma ação temos um redirecionamento. Ao ser chamada uma ação, esta retorna para onde deseja redirecionar a aplicação na camada de visualização;
- **Validação de formulários**: o WebWork pode automatizar a validação de formulários com arquivos simples em XML, separando e facilitando sem a necessidade de programação em Java;
- **Componentes**: com a característica de Inversão de Controle do WebWork[5], é possível ter objetos que ficam disponíveis para as ações em um determinado escopo;
- **Interceptadores (Interceptors)**: com os interceptors, uma ação pode ser interceptada antes e depois da sua execução podendo ter seu fluxo desviado. Eles são modulares e múltiplos interceptors podem ser utilizados para uma ação;
- **Internacionalização**: o WebWork possui arquivos de recursos separados por ações e por línguas. Isso permite a fácil criação de recursos internacionalizados nos aplicativos web.

Utilizando o WebWork, os JSP´s terão apenas a tarefa de controlar a visualização do sistema, a camada View e os JSP's é quem irão fazer a requisição das ações ao WebWork.

Para o armazenamento das informações foi utilizado o MySQL (Axmark et al. 2006), que é um banco de dados objeto-relacional e foi criado na Suécia desde a década de 80. A partir do modelo conceitual apresentado na seção 7.2.2, chegamos ao código SQL gerando as tabelas. A criação do banco de dados envolve várias tarefas como a configuração da base de dados, a criação de usuários, etc.

As principais características do MySQL são portabilidade (suporta praticamente qualquer plataforma atual), compatibilidade (existem drivers ODBC, JDBC e .NET e módulos

[4]É um padrão de desenvolvimento que preconiza a separação da camada de visualização das regras de negócios através de um controlador.

[5]É uma implementação genérica do Command Pattern e é totalmente desacoplada do ambiente Web, o que facilita o testes de suas ações. Ela provê ao processamento das ações uma poderosa linguagem de expressões conhecida como OGNL, um container de IoC, interceptadores, conversão de tipos e um framework de validação (Lightbody & Carreira 2005).

de interface para diversas linguagens de programação), excelente desempenho e estabilidade, pouco exigente quanto aos recursos de hardware, facilidade de uso e o fato de ser um software livre.

Para o desenvolvimento dos agentes reativos, foi utilizado o framework JADE conforme as especificações FIPA (ver Apêndice B), que objetiva simplificar e facilitar o desenvolvimento de sistemas multiagentes garantindo um padrão de interoperabilidade entre sistemas multiagentes.

8.2.1 Disposição Física dos Módulos do Sistema

As tecnologias empregadas para o desenvolvimento do framework FA_Port e das duas aplicações criadas a partir dele possibilita a disposição física dos módulos em três camadas (Figura 8.4). A primeira camada é a interface da aplicação, que é feita através de um navegador (*browser*) via web, a camada central corresponde a aplicação, é executada por um servidor web (estendido como um container JSP) e a terceira camada, o armazenamento, realizado pelo servidor de banco de dados.

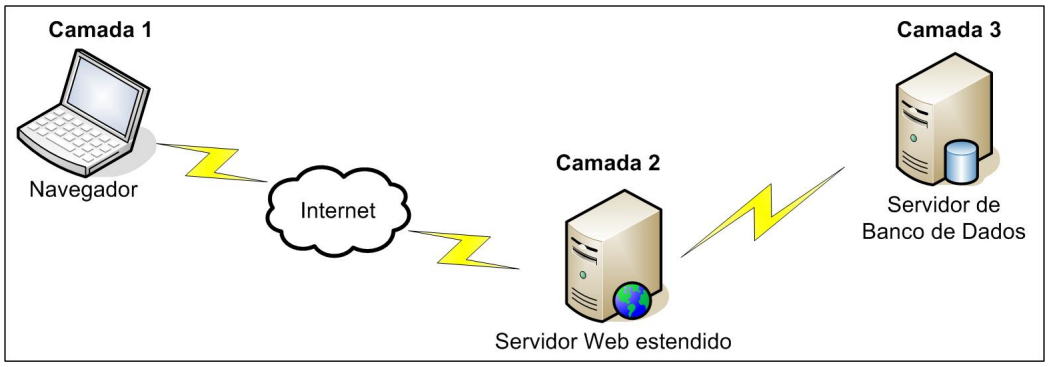


Figura 8.4: Disposição Física em 3 Camadas

8.2.2 O Componente de Comunicação

O componente de comunicação é constituído pelo elemento responsável pela execução da funcionalidade envio de mensagens de correio eletrônico gerado na utilização das aplicações. Para o envio de mensagens eletrônicas foi utilizada a API JavaMail[6]. O controle do framework verifica se é necessária a geração e envio de uma mensagem, no caso afirmativo, deve-se enviar essa mensagem ao usuário (Comportamento Pró-Ativo). As classes utilizadas são as seguintes:

- Email: classe responsável por enviar o email ao destinatário correto;

[6]É um conjunto de classes abstratas que modelam um sistema de correio eletrônico. Esta API fornece uma plataforma e protocolos independentes para se trabalhar com IMAP, POP, SMTP, MIME e todos os outros protocolos de envio de mensagens pela internet. Assim, esta oferece facilidades para envio e leitura de mensagens eletrônicas.

- Session: classe do pacote "javax.mail[7]", que representa uma sessão de mail, contendo o conjunto das propriedades usadas pela API JavaMail;
- MimeMessage: classe do pacote "javax.mail.internet[8]", que representa uma mensagem de e-mail no estilo MIME[9] (Multipurpose Internet Mail Extensions). Os métodos setFrom, addRecipient, setSubject e setText especificam o remetente, o destinatário, o assunto e o conteúdo para a mensagem, respectivamente.
- Transport: classe abstrata do pacote "javax.mail", que modela o transporte da mensagem. O método sendMessage envia a mensagem para todos os destinatários especificados na mesma.

8.3 Estudos de Caso

A partir do uso do framework FA_PorT foram construídas duas aplicações, que foram criadas devido à necessidade de um sistema computacional que facilite o melhor controle e o acompanhamento do desempenho dos alunos nas diversas atividades de aprendizagem, já que sem estas aplicações esse trabalho seria realizado de forma manual e lenta.

Estas aplicações foram desenvolvidas baseadas na tecnologia web. A interface é basicamente a do navegador (*browser*), com as informações ocupando a área existente para a navegação.

A sessão de ensino que faz parte da camada Tutor é especificada pelo professor, ou seja, o professor cria uma sessão de ensino, e o sistema, através do comportamento próativo, informa a criação desta sessão de ensino para os participantes (professores e alunos). Estas sessões de ensino podem conter uma ou várias estratégias didáticas (que também é criada pelo professor, ou pode ser utilizada alguma estratégia didática existente), que são compostas por táticas de ensino, como visto na seção 7.7.1.

A seguir é apresentada com detalhes as aplicações Portfólio-Tutor desenvolvidas.

8.3.1 Aplicação 1

A aplicação 1 objetiva auxiliar o professor da disciplina de Engenharia de Software. Esta trabalha com a implementação default do framework FA_PorT, ou seja, é um sistema portfólio-tutor que possui funcionalidades específicas predefinidas sem precisar da redefinição dos pontos de adaptação de código.

A tela padrão da aplicação 1 é apresentado na Figura 8.5.

[7]Faz parte da API JavaMail.
[8]Faz parte da API JavaMail.
[9]é uma norma da Internet para o formato das mensagens de correio eletrônico. A grande maioria das mensagens de correio eletrônico são trocadas usando o protocolo SMTP e usam o formato MIME.

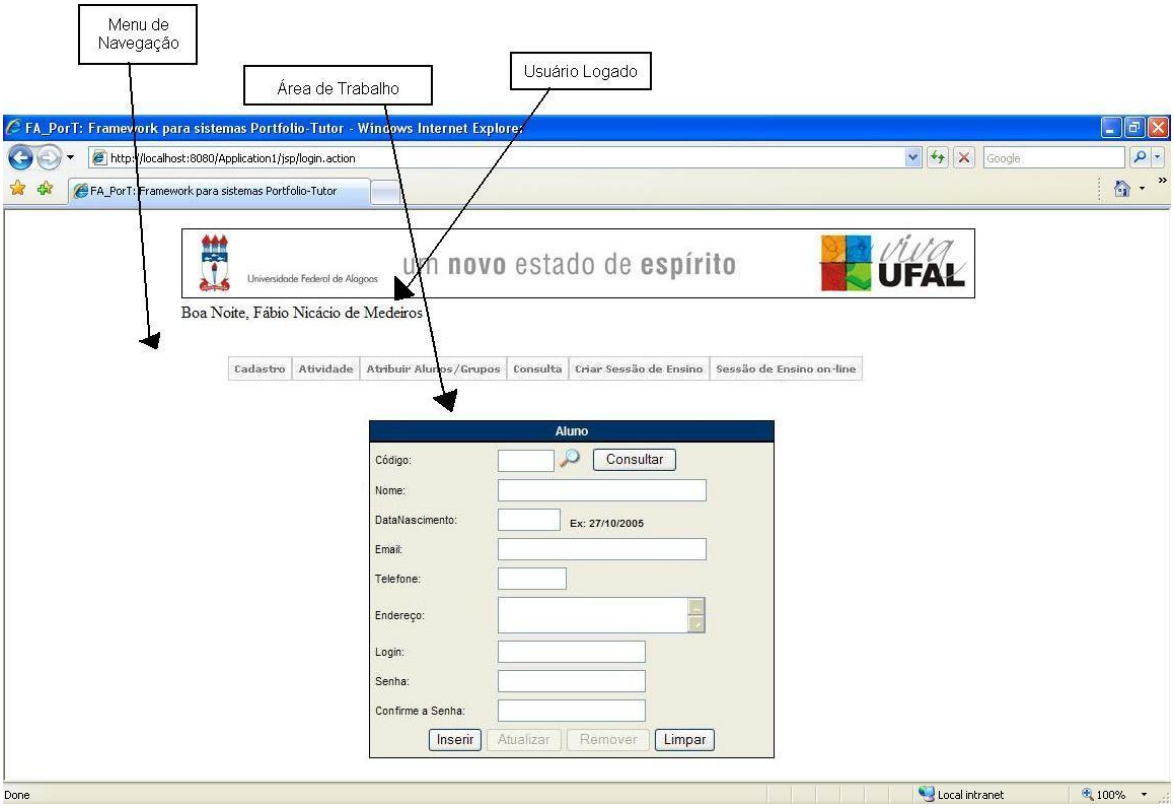


Figura 8.5: Tela Padrão da Aplicação 1

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

O usuário ao inicializar a utilização do sistema, precisará identificar-se através de um login e uma senha (Figura 8.6). O sistema quando reconhecer o perfil do usuário, direcionará para sua respectiva interface.

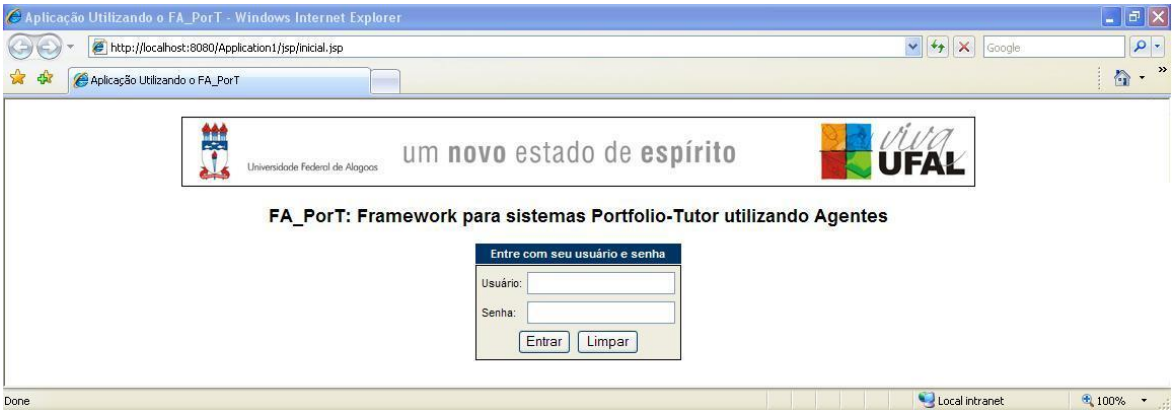


Figura 8.6: Tela Inicial da Aplicação 1

A seguir serão mostradas algumas funcionalidades da Aplicação 1.

A Figura 8.7 representa a primeira tela de navegação apresentada na aplicação 1.

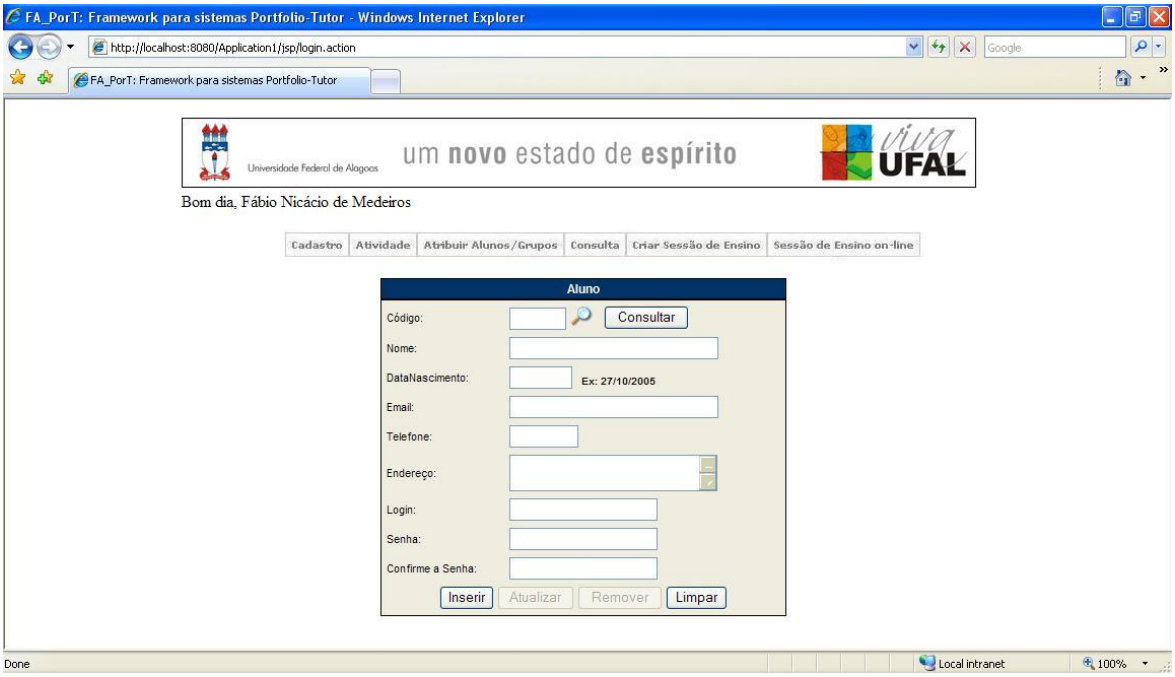


Figura 8.7: Tela de Menu da Aplicação 1

As seguintes opções são disponibilizadas: "Cadastro", "atividades", "Atribuir Alunos/ Grupos", "Consulta", "Criar Sessão de Ensino", "Sessão de Ensino On-Line" e "Quadro de Avisos".

Ao selecionar a opção de "Cadastro", as opções que aparecem são cadastrar: aluno, grupo, curso, disciplina, professor, domínio, conceito, unidade, item de avaliação, atividade (Figura 8.8), recurso didático e estratégia didática. Através destas opções o professor pode efetuar qualquer cadastro.

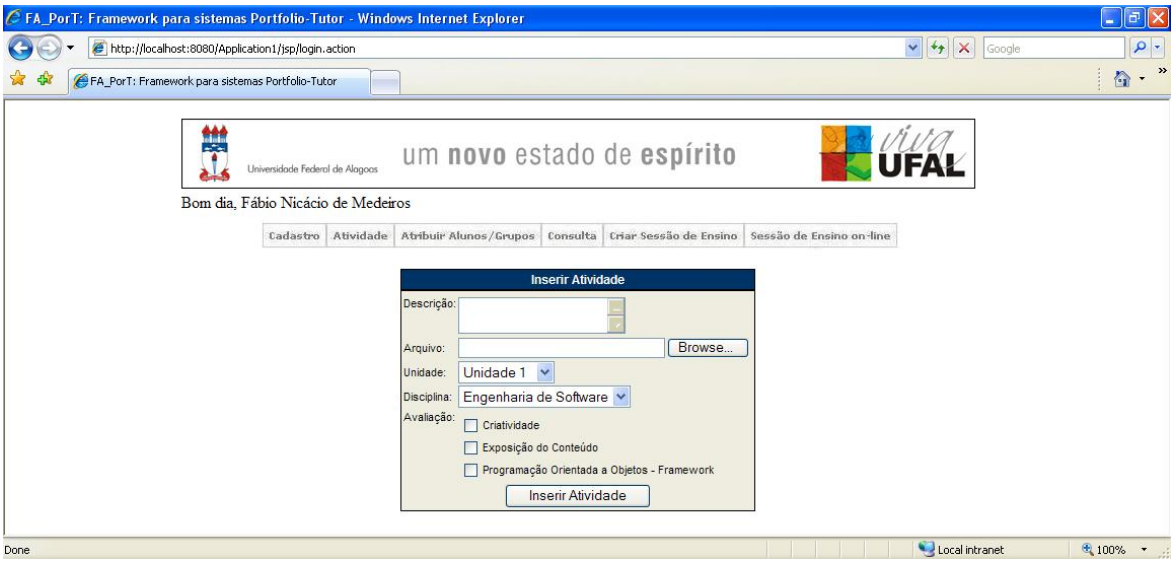


Figura 8.8: Tela "Cadastrar Atividade" da Aplicação 1

Na opção de "Atividade", as seguintes opções são previstas:

- Atualizar Atividade: o professor pode, em qualquer momento, atualizar alguma atividade criada por ele;
- Atribuir Atividade: através desta opção, o professor pode atribuir uma nova atividade para os alunos (Figura 8.9), informando a data final de entrega por parte dos alunos, bem como a data final para avaliação da mesma;
- Encerrar Atividade: o aluno, através desta opção, pode encerrar suas atividades e entregar ao professor (Figura 8.10);
- Avaliar Atividade: selecionando esta opção, o professor poderá avaliar as atividades de seus alunos, atribuindo uma nota para cada item de avaliação da atividade (Figura 8.11);

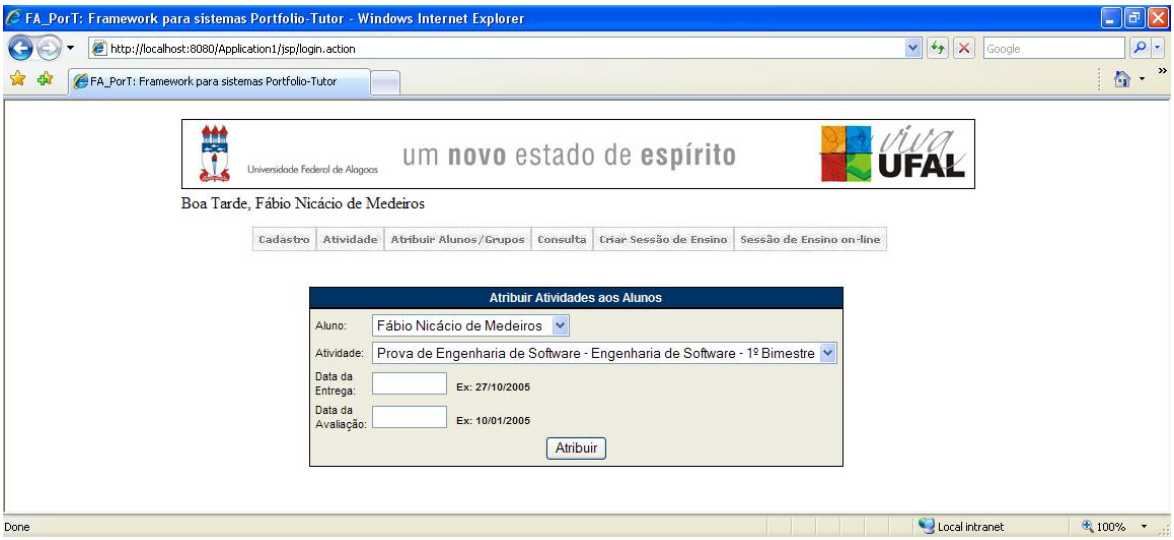


Figura 8.9: Tela "Atribuir Atividade" da Aplicação 1

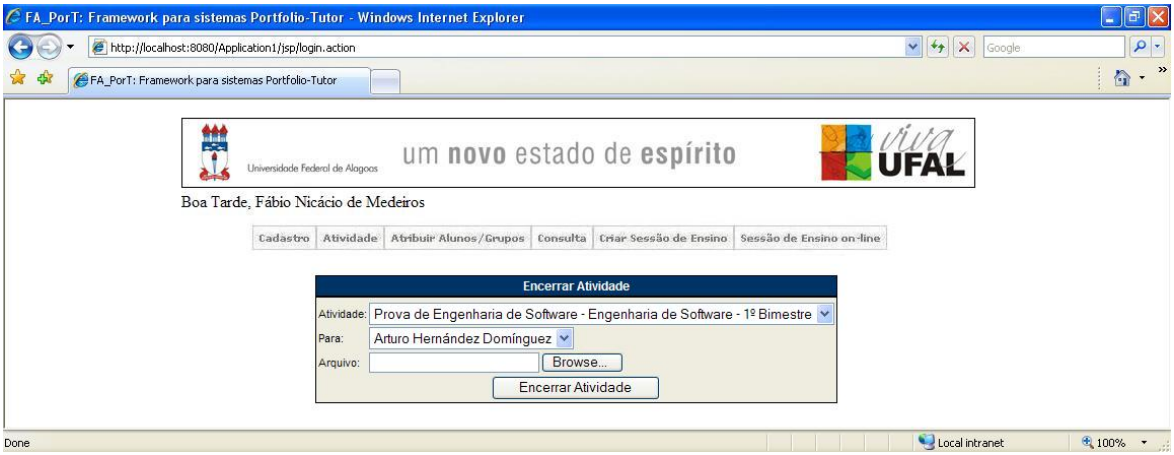


Figura 8.10: Tela "Encerrar Atividade" da Aplicação 1

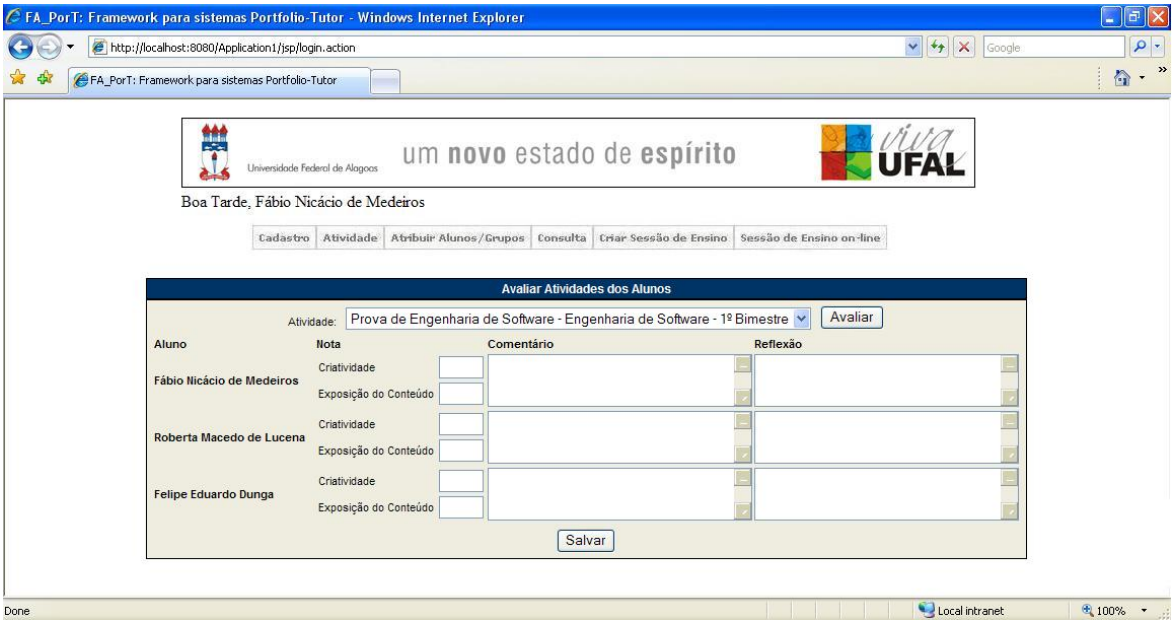


Figura 8.11: Tela "Avaliar Atividade" da Aplicação 1

Na opção de "Atribuir Alunos/Grupos", temos que, um professor pode definir grupos de alunos (Figura 8.12).

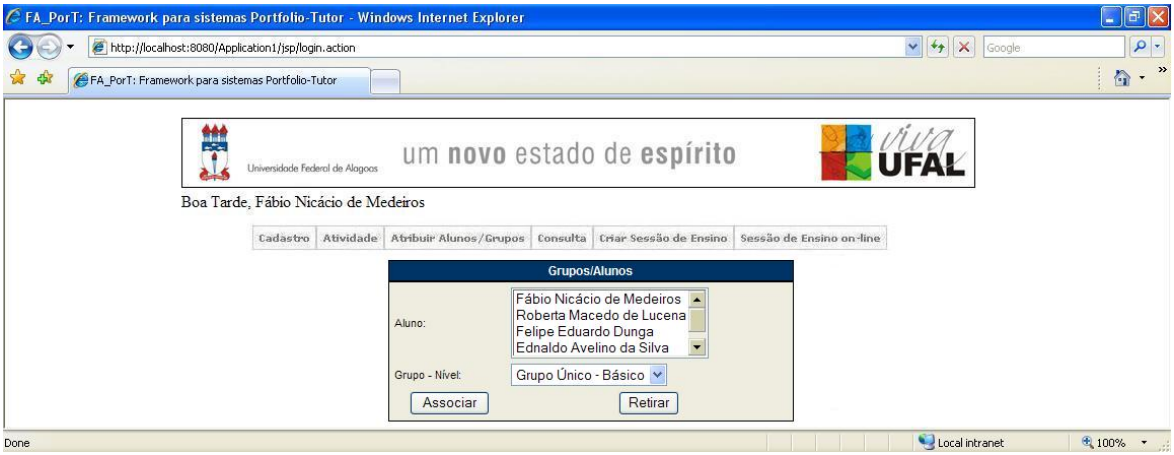


Figura 8.12: Tela "Atribuir Alunos/Grupos" da Aplicação 1

Na opção de "Consulta", as seguintes opções são previstas:

- Desempenho: esta opção serve para verificar o desempenho dos alunos em uma determinada atividade de ensino. Será melhor vista na Seção pontos de adaptação de código desta aplicação;
- Recursos Didáticos: o professor pode consultar todos os recursos didáticos cadastrados por ele (Figura 8.13);
- Perfil do Aluno: o professor ou o aluno pode verificar o perfil de cada aluno de acordo com o item de avaliação e, para cada item, pode-se ver a atividade que foi

avaliada, sua nota e data de entrega. Será melhor vista na Seção Perfil Individual e do Grupo desta aplicação;

- Perfil do Grupo: o professor ou o aluno pode verificar o perfil de grupo de acordo com o item de avaliação e, para cada item, pode-se ver os alunos participantes do grupo, a atividade que foi avaliada, a nota e data de entrega. Será melhor vista na Seção Perfil Individual e do Grupo desta aplicação;

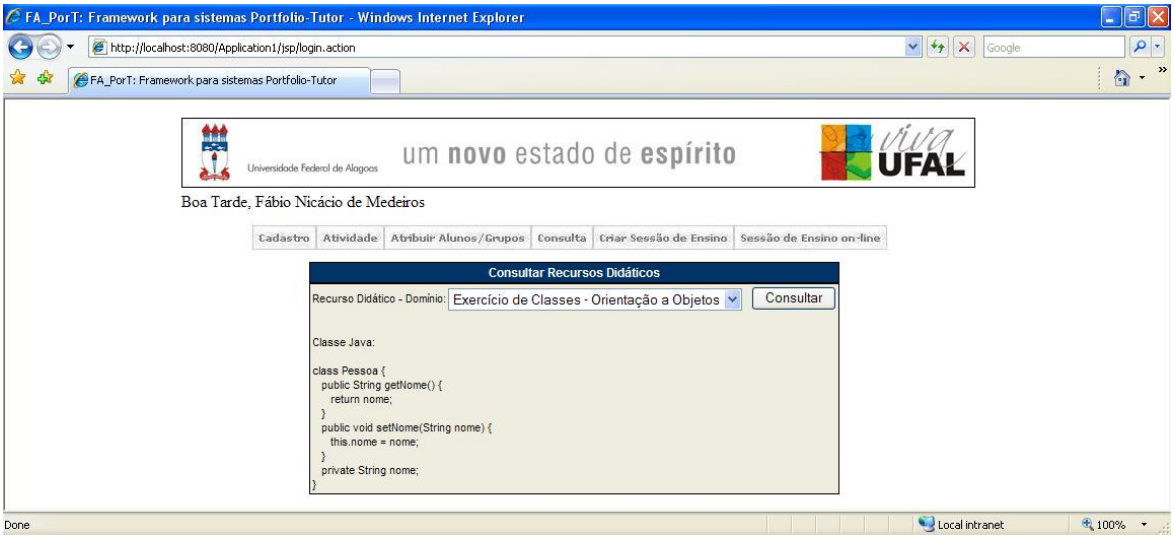


Figura 8.13: Tela "Consultar Recursos Didáticos" da Aplicação 1

E, por fim, temos as funcionalidades "Sessão de Ensino On-Line", "Criar Sessão de Ensino" e "Quadro de Avisos", que serão melhores descritos nas Seções Exemplo de uma Sessão de Ensino On-line e Pontos de Adaptação de Código desta aplicação.

Exemplo de uma Sessão de Ensino On line

A seguir é apresentado uma sessão de ensino sobre o conteúdo Processos de Desenvolvimento de Software (Figura 8.14).

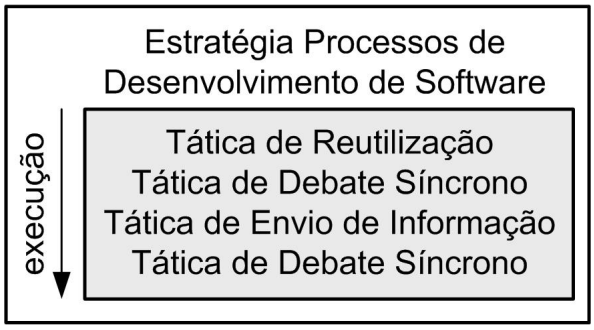


Figura 8.14: Execução da Estratégia Didática na Aplicação 1

A definição das táticas de ensino referente a estratégia didática Processos de Desenvolvimento de Software são:

1. (Revisão) Tática: Reuso (Definição de Processo de Desenvolvimento de Software, 10min);
2. (Fixação) Tática: DebateSincrono (Grupo Intermediário, Professor, 15min);
3. (Aprofundamento) Tática: EnvioInformacao (Arquivos sobre Processo de Desenvolvimento de Software);
4. (Aprofundamento e fixação dos conceitos) Tática: DebateSincrono (Grupo Intermediário, Professor, 20min).

A Figura 8.15, mostra a execução da primeira tática de ensino utilizando a tática de reuso do recurso para mostrar a definição do processo de desenvolvimento de software em um período de dez minutos. A aplicação 1 utiliza o programa Acrobat Reader, para visualizar os arquivos pdf. A partir da Figura 8.15, será mostrada apenas a área de trabalho.

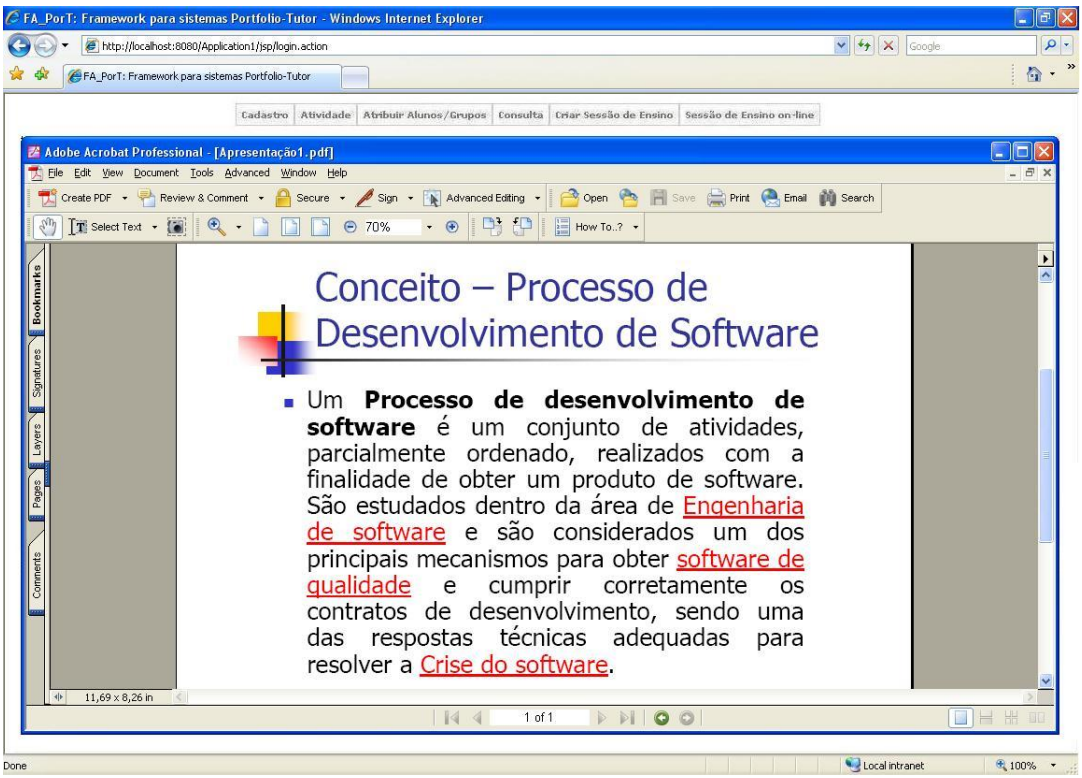


Figura 8.15: Primeira Tática de Reuso de Recurso para definição do Processo de Desenvolvimento de Software para a Aplicação 1

A segunda tática de ensino é utilizada uma tática de debate síncrono através de um chat para um grupo (intermediário) de alunos com o objetivo de fixação do assunto e utiliza um tempo de quinze minutos, como mostra a Figura 8.16.

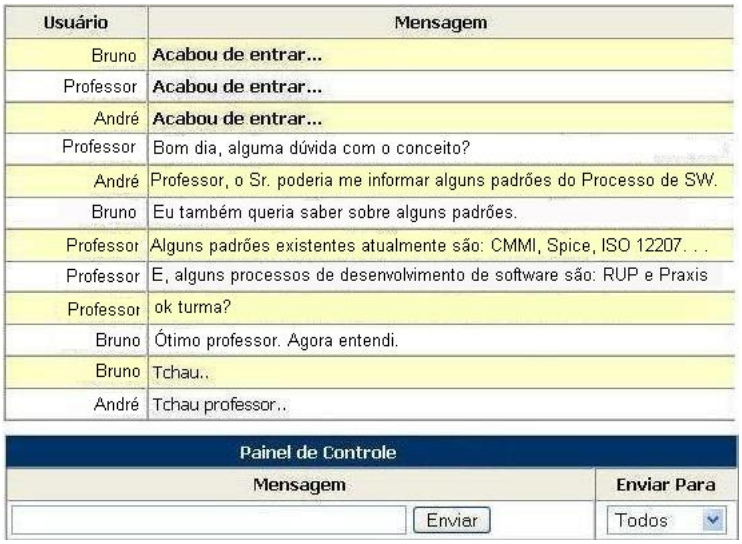


Figura 8.16: Segunda Tática de Ensino utilizando Debate Síncrono para a Aplicação 1

Para a terceira tática de ensino é utilizada a tática de envio de informação (Figura 8.17) com o objetivo de aprofundar o conteúdo. Neste exemplo foi enviado por email um material sobre Processo de Desenvolvimento de Software.

Figura 8.17: Terceira Tática de Envio de Informações para a Aplicação 1

E, para a quarta e última tática de ensino nesta sessão foi utilizada a tática de debate síncrono com o objetivo de aprofundar e fixar os conceitos apresentados durante vinte minutos (Figura 8.18).

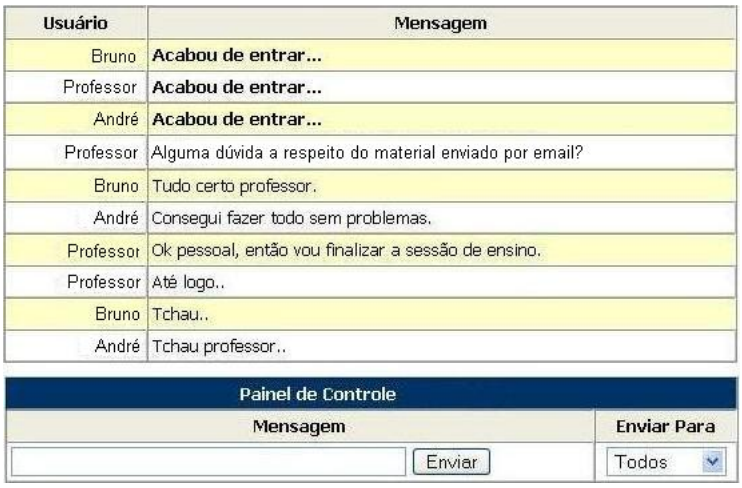


Figura 8.18: Quarta Tática de Ensino utilizando Debate Síncrono para a Aplicação 1

Pontos de Adaptação de Código

Esta aplicação trabalha com a implementação default do framework FA_PorT. Um sistema Portfólio-Tutor possui funcionalidades específicas predefinidas sem precisar da redefinição dos pontos de adaptação de código[10], que são a geração de gráficos e a mensagem.

A interface gráfica da geração default de gráficos (gráfico de pizza) associada a essa aplicação representa o desempenho dos alunos em uma atividade. A interface gráfica do ponto de adaptação do quadro de avisos é mostrado na Figura 8.19.

Figura 8.19: Ponto de Adaptação do Gráfico na Aplicação 1, tela do desempenho dos alunos

A interface gráfica da geração default de mensagens (na tela do sistema) associada a essa aplicação representa informações sobre atividades a serem concluídas e sessões de ensino criadas (Figura 8.20).

[10] O ponto de adaptação geração de gráfico se diferencia de uma aplicação para a outra, devido ao tipo de gráfico, podendo ser os gráficos de barras e de pizza, e a mensagem pode ser através da interface gráfica da aplicação ou por email.

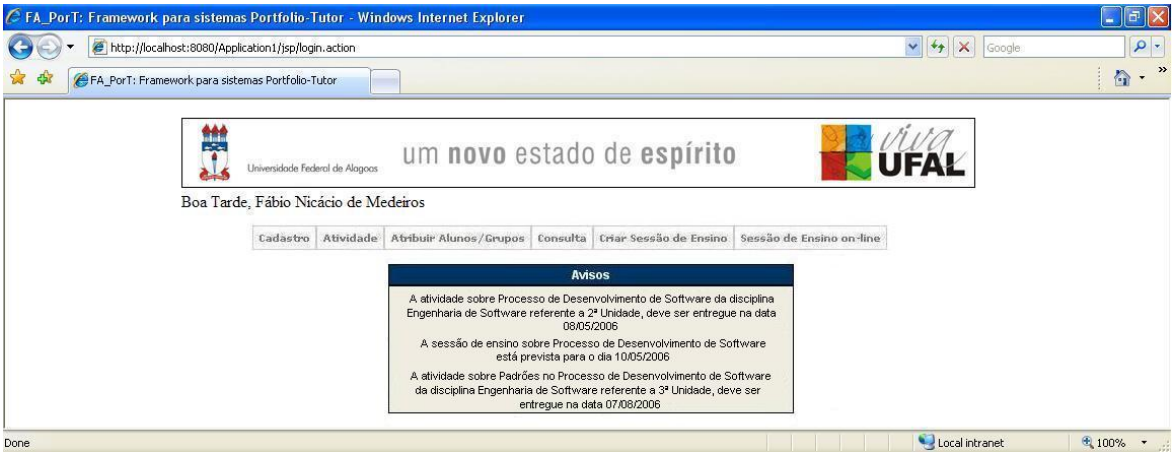


Figura 8.20: Ponto de Adaptação das Mensagens na Aplicação 1, tela do Quadro de Avisos

Perfil Individual e do Grupo

Na Figura 8.21, temos o perfil individual de acordo com os itens de avaliação e seu comportamento (Ruim, Regular ou Bom[11]) da primeira aplicação.

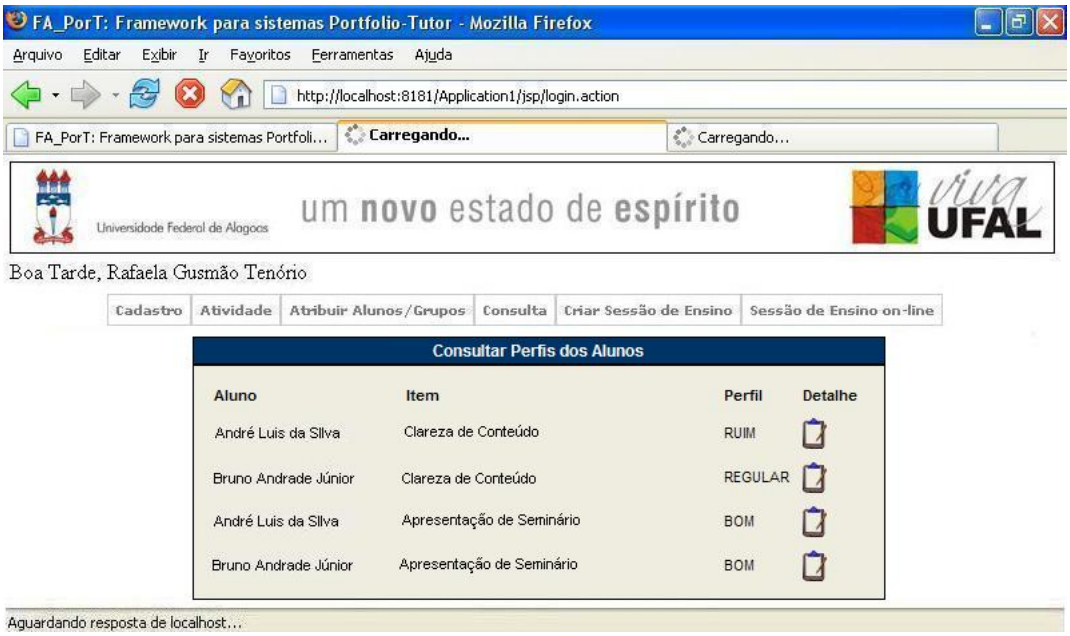


Figura 8.21: Tela do Perfil Individual na Aplicação 1

A Figura 8.22 mostra os detalhes (as notas e a data de entrega) da aluna Roberta Macedo Severo de Lucena no item de avaliação Especificação de um Framework.

[11]Esta classificação é gerada a partir da média das notas referente a um determinado item de avaliação, se a média for menor que 5.0 é classificado como ruim, se for entre 5.0 e 7.0 é regular e se for maior que 7.0 é classificado como bom.

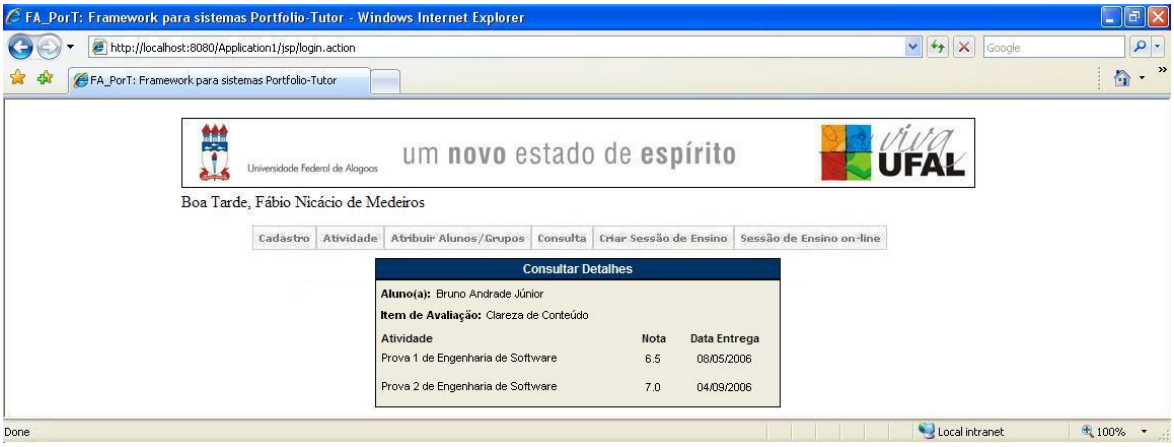


Figura 8.22: Tela do Detalhe do Perfil Individual na Aplicação 1

Na Figura 8.23, temos o comportamento de um determinado grupo também de acordo com os itens de avaliação. E, para detalhar cada item de avaliação, temos a Figura 8.24, que mostra os alunos que participam daquele grupo com as suas atividades, as notas e a data de entrega de cada atividade.

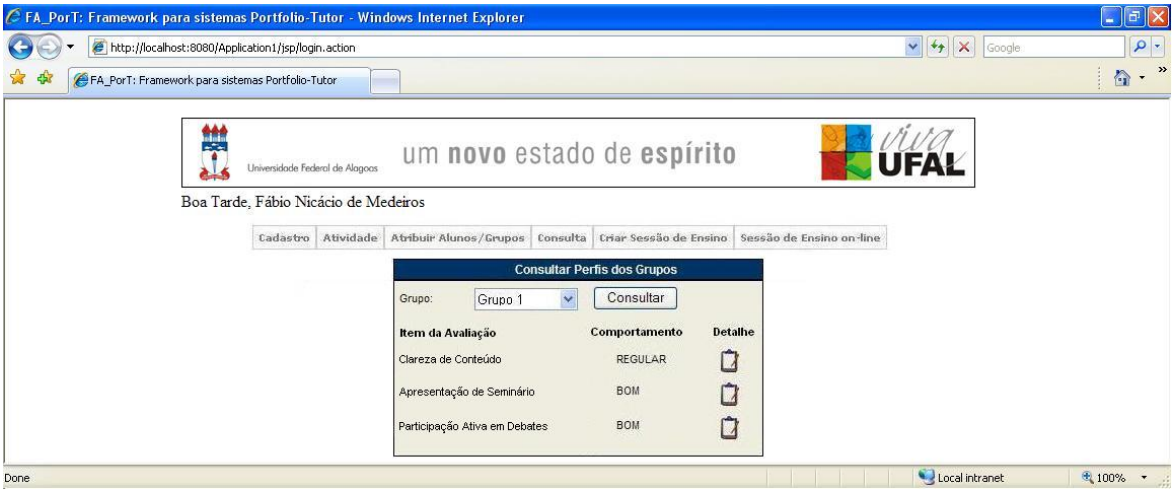


Figura 8.23: Tela do Perfil do Grupo na Aplicação 1

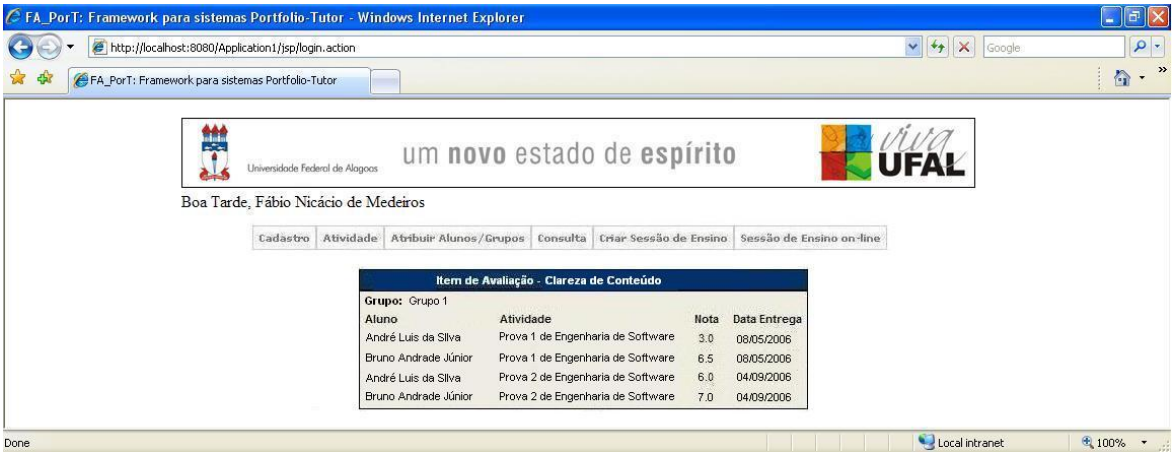


Figura 8.24: Tela do Detalhe do Perfil do Grupo na Aplicação 1

8.3.2 Aplicação 2

Esta segunda aplicação objetiva auxiliar o professor da disciplina Análise de Sistemas II. Ela representa um novo sistema portfólio-tutor que possui uma funcionalidade redefinida, que é o método geração de gráfico.

A tela padrão da aplicação 2 é apresentado na Figura 8.25.

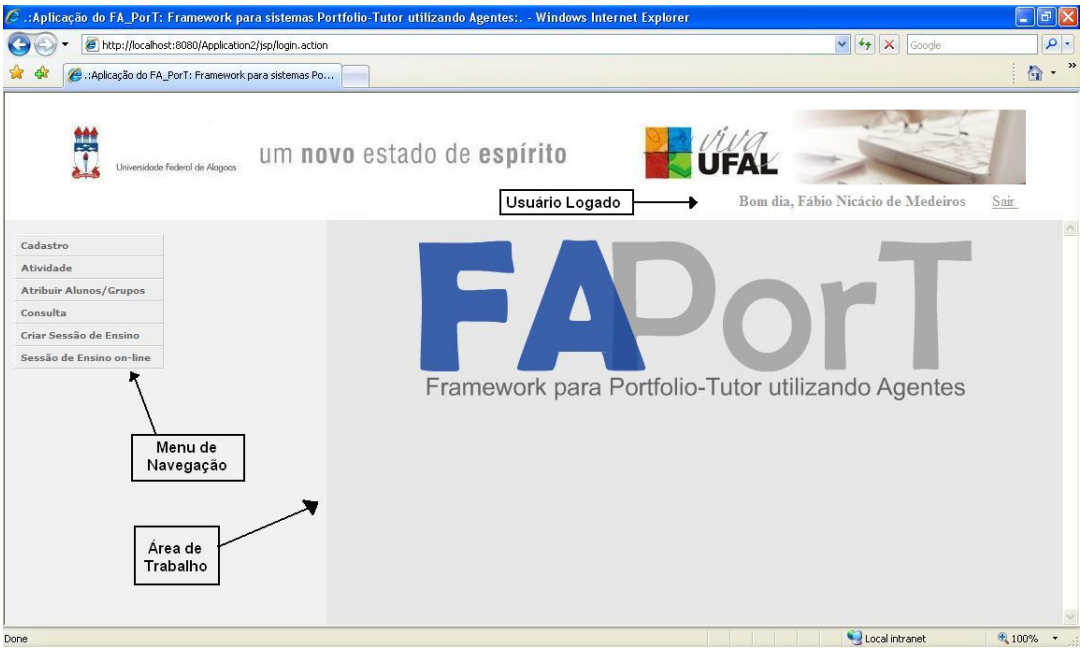


Figura 8.25: Tela Padrão da Aplicação 2

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

O usuário ao inicializar a utilização do sistema, também precisará identificar-se através de um login e uma senha (Figura 8.26). O sistema quando reconhecer o perfil do usuário,

direcionará para sua respectiva interface.

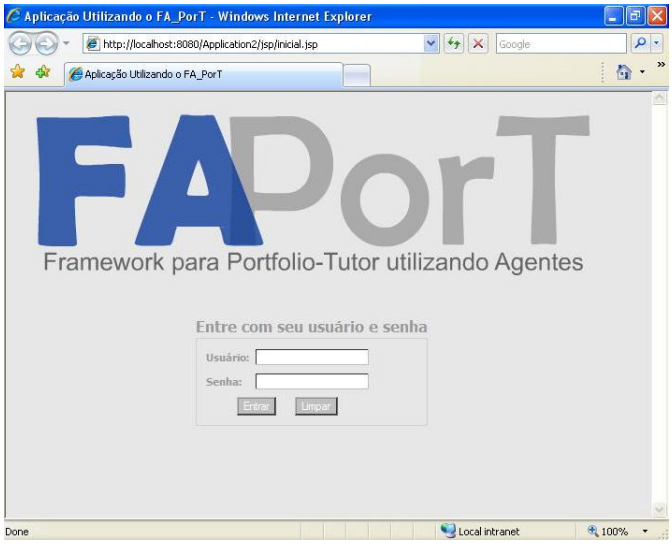


Figura 8.26: Tela Inicial da Aplicação 2

A aplicação 2 diferencia-se da aplicação 1 devido aos pontos de adaptação de código, por isso, serão mostradas apenas as telas destes pontos logo após o exemplo da sessão de ensino on-line.

Exemplo de uma Sessão de Ensino On-line

A seguir é apresentada uma sessão de ensino sobre o conteúdo Framework (Figura 8.27).

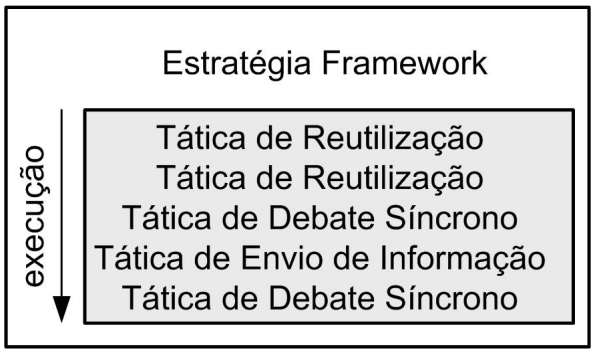


Figura 8.27: Execução da Estratégia Didática na Aplicação 2

A definição das táticas de ensino referente a estratégia didática Framework são:

1. (Revisão) Tática: Reuso (Definição de Framework, 10min);
2. (Revisão) Tática: Reuso (Exemplos de Framework, 10min);
3. (Fixação) Tática: DebateSincrono (Grupo Avançado, Professor, 15min);

4. (Aprofundamento) Tática: EnvioInformacao (Arquivos sobre Framework, artigo introdutório sobre framework, exercício e avaliação);

5. (Aprofundamento e fixação dos conceitos) Tática: DebateSincrono (Grupo Avançado, Professor, 20min).

Primeiramente, devemos criar a estratégia didática referente a esta sessão de ensino, a Figura 8.28 mostra esta criação.

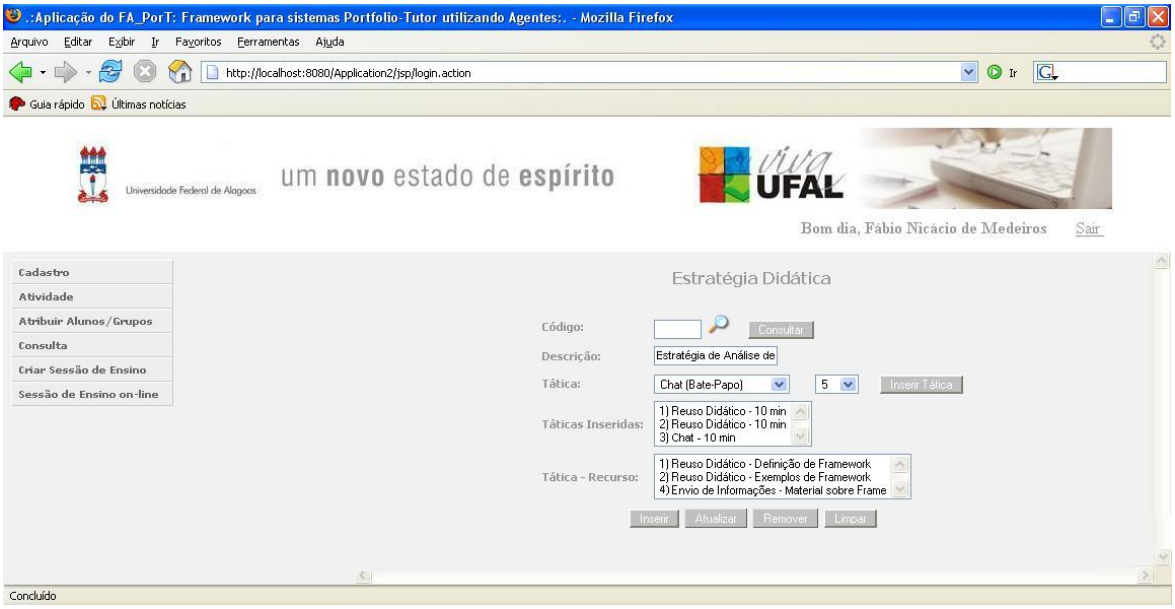


Figura 8.28: Tela "Cadastrar Estratégia Didática" da Aplicação 2

O segundo passo, é que o professor deve criar a sessão de ensino com os alunos participantes, a estratégia didática que irá utilizar, a data e a hora de início (Figura 8.29).

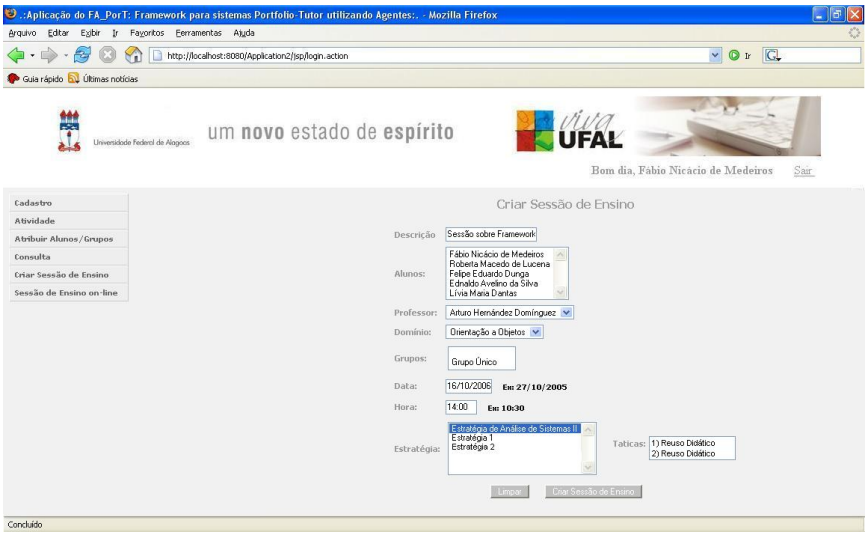


Figura 8.29: Tela "Cadastrar Sessão de Ensino" da Aplicação 2

O primeiro passo da execução da sessão de ensino está relacionado a Definição de Framework, para o mesmo utilizamos a Tática de Reuso com um recurso didático em uma duração de 10 minutos, podemos ver na Figura 8.30 e utiliza também o programa Acrobat Reader, para visualizar os arquivos pdf. A partir da Figura 8.30, será mostrada nas figuras posteriores apenas a área de trabalho.

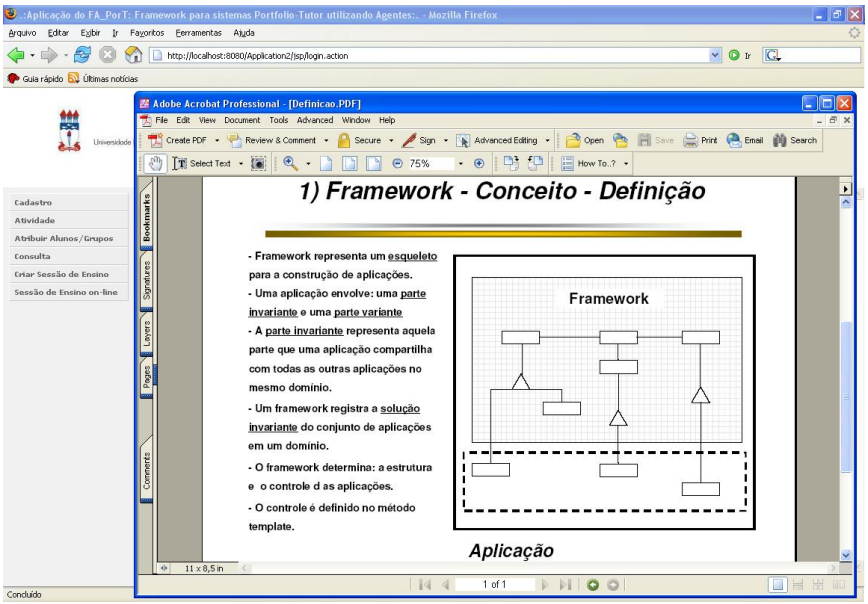


Figura 8.30: Primeira Tática de Reuso de Recurso para definição de framework para a Aplicação 2

Na segunda tática de ensino é utilizada também uma tática de reuso do recurso com um exemplo do framework apresentado durante dez minutos (Figura 8.31). Na Figura 8.31.a temos a especificação do exemplo e na Figura 8.31.b é apresentado a implementação do framework exemplo.

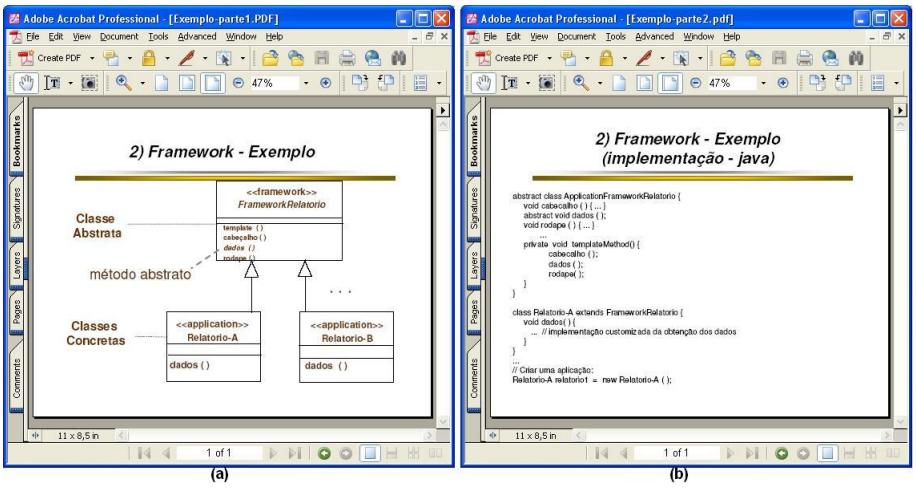


Figura 8.31: Segunda Tática de Ensino utilizando Reuso do Recurso para o exemplo do framework na Aplicação 2: (a) Especificação e (b) Código

Na terceira tática de ensino é utilizada uma tática de debate síncrono através de um chat com um tempo de quinze minutos como mostra a Figura 8.32.

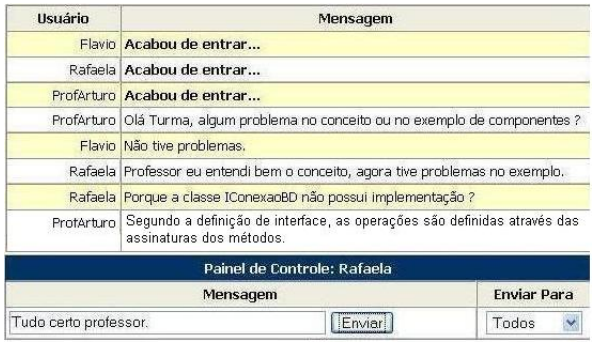


Figura 8.32: Terceira Tática de Ensino utilizando Debate Síncrono para a Aplicação 2

Na quarta tática, (envio de informações), é enviado a todos os alunos o material sobre framework, uma lista de exercício e uma avaliação por email. Na Figura 8.33, temos o exercício (Figura 8.33.a) e a avaliação (Figura 8.33.b).

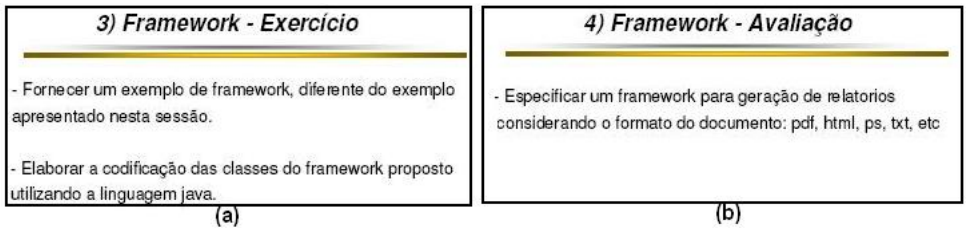


Figura 8.33: Quarta Tática de Envio de Informações para a Aplicação 2: (a)Lista de Exercícios e (b) Avaliação

Para o aprofundamento com o professor, revisão do exercício elaborado e fixação dos conceitos envolvidos, temos a quinta e última tática, que é uma tática de debate síncrono com a duração de vinte minutos (Figura 8.34).

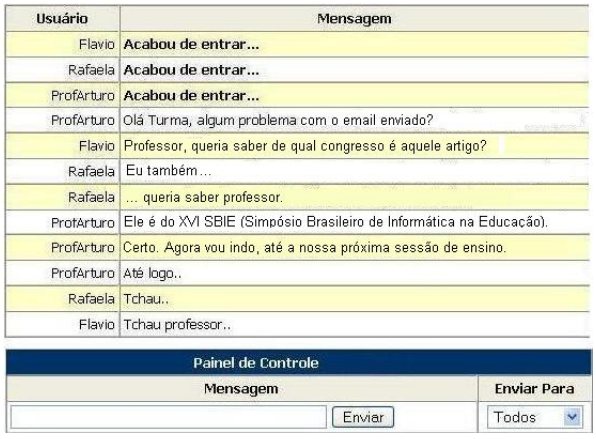


Figura 8.34: Quinta Tática de Ensino utilizando Debate Síncrono para a Aplicação 2

Pontos de Adaptação de Código

Esta segunda aplicação representa um novo sistema portfólio-tutor que possui funcionalidades redefinidas (os pontos de adaptação de código), como o método geração de gráfico (construída com gráfico de barras, que pode ser visto na Figura 8.35) e mensagens (através de email).

Figura 8.35: Ponto de Adaptação do Gráfico na Aplicação 2, tela do desempenho dos alunos

A Figura 8.35 mostra o desempenho de um grupo de alunos referente a uma avaliação (com o assunto Programação Orientado a Objetos - Framework) na disciplina de Análise de Sistemas II.

Perfil Individual e do Grupo

Na Figura 8.36, temos o perfil individual de acordo com os itens de avaliação e seu comportamento da segunda aplicação.

Figura 8.36: Tela do Perfil Individual na Aplicação 2

A Figura 8.37 mostra com detalhes o perfil individual da aluna Roberta Macedo Severo de Lucena no item de avaliação Especificação de um Framework.

Figura 8.37: Tela do Detalhe do Perfil Individual na Aplicação 2

E, por fim, na Figura 8.38 temos o comportamento de um determinado grupo também de acordo com os itens de avaliação. E, para detalhar cada item de avaliação, temos a Figura 8.39, que mostra os alunos que participam daquele grupo com as suas atividades, as notas e a data de entrega de cada atividade.

Figura 8.38: Tela do Perfil do Grupo na Aplicação 2

Figura 8.39: Tela do Detalhe do Perfil do Grupo na Aplicação 2

8.4 Conclusões

Neste capítulo, apresentou-se os aspectos de implementação relacionados ao framework FA_PorT, bem como a demonstração dos protótipos das aplicações implementadas a partir do framework.

A tecnologia empregada no desenvolvimento do framework possibilita a sua utilização no contexto de ensino a distância, uma vez que se utiliza da infraestrutura da internet. Uma aplicação Portfólio-Tutor além de acompanhar o andamento das atividades, fornece

ao professor informações sobre o rendimento de um grupo e de cada aluno individualmente, referente as atividades de ensino.

Com o portfólio, tem-se o acompanhamento do rendimento do aluno e evolução ao longo do tempo, no contexto de uma disciplina, unidades e atividades desenvolvidas.

É importante ressaltar que uma sessão on-line de uma aplicação Portfólio-Tutor pode funcionar conforme diferentes estratégias didáticas definidas pelo professor, levando-se em conta as táticas de ensino previamente implementadas.

No próximo capítulo apresenta-se as considerações finais deste trabalho.

Capítulo 9

Considerações Finais

Neste capítulo, apresenta-se as considerações finais, os resultados obtidos e as perspectivas para trabalhos futuros.

O trabalho apresentado nesta dissertação teve por objetivo a especificação e implementação parcial de um framework para sistemas Portfólio-Tutor baseado em Agentes e a construção de duas aplicações geradas a partir dele. Suas contribuições são contextualizadas a partir de trabalhos de pesquisa que envolveu os sistemas portfólios eletrônicos, o POETA (Sistêlos 1999), o Portfólio na Web (Silva 2002), a camada portfólio do sistema Portfólio-Tutor (Nascimento 2002) e o sistema tutor inteligente do TUTA (Silva 2000).

A utilização de sistemas do tipo portfólio eletrônico é crescente. Estes tem sido desenvolvidos, porém poucos foram implementados focalizando o reaproveitamento da estrutura de sistemas de tipo portfólio eletrônico acoplado a um Sistema Tutor Inteligente. O FA_PorT favorece o desenvolvimento de novos portfólios eletrônicos (Madeiro et al. 2005) baseados em componentes e agentes (Medeiros et al. 2006), reduzindo de forma considerável o tempo de desenvolvimento de um novo sistema portfólio-tutor.

O framework FA_PorT (Madeiro et al. 2005) (Medeiros et al. 2006) permite o reuso da estrutura interna (esqueletos de aplicações) para criação de aplicações Portfólio-Tutor, que podem ser personalizados, levando em consideração as necessidades específicas de uma aplicação.

9.1 Resultados Obtidos

A informática na educação tem muitas aplicações, entretanto é uma área difícil por envolver diferentes competências. Assim, desenvolver um framework e aplicações desta natureza, com certeza não é uma tarefa fácil.

9.1.1 Contribuições

- Em relação ao sistema Portfólio-Tutor (Nascimento 2002), o FA_PorT tem como contribuição, além do reuso da arquitetura e componentes, o acoplamento de uma camada tutor baseada em agentes para um sistema Portfólio-Tutor, a abordagem de agentes permite que os sistemas Portfólio-Tutor sejam flexíveis, tendo como principal característica a reatividade;

- Em relação ao sistema Tutor TUTA (Silva 2000), o FA_PorT apresenta um arcabouço que permite o desenvolvimento de um sistema Tutor on-line (via web), isto é, a camada tutor de um sistema Portfólio-Tutor. Tradicionalmente um sistema tutor é utilizado por um aluno durante uma sessão, no caso das aplicações criadas a partir do FA_PorT, uma sessão de ensino on-line (via web) permite a participação de um grupo de alunos e o sistema leva em consideração o comportamento do aluno, assim como o comportamento do grupo;

- Em relação a Frameworks, o FA_PorT representa uma arquitetura reutilizável no domínio específico de ambiente interativos de aprendizagem, particularmente, sistemas Tutor e Portfólio-Tutor. As aplicações Portfólio-Tutor são aplicações web.

A Tabela 9.1 mostra claramente a contribuição do trabalho desenvolvido em relação aos trabalhos correlatos.

Sistema	**Principal Contribuição do Trabalho**
POETA	O Reuso da Arquitetura de um sistema Portfólio Eletrônico e o Acoplamento da camada Tutor com Agentes.
TUTA	O Reuso da Arquitetura de um sistema Tutor e o Acoplamento da camada Portfólio.
Portfólio-Tutor	O Reuso da Arquitetura de um sistema Portfólio-Tutor e o Acoplamento de Agentes na camada Tutor.

Tabela 9.1: Contribuição do Trabalho Desenvolvido

9.1.2 Situação atual do desenvolvimento do Framework

- Permite que o professor especifique uma sessão de ensino através de uma estratégia didática, onde cada estratégia didática é representada através das táticas de ensino: Reutilização, Debate Síncrono, Envio de Informação, Mudança de Estratégia e Regra.

- O framework é capaz de gerar várias aplicações Portfólio-Tutor, facilitando e diminuindo o tempo para o desenvolvimento de novas aplicações;

- Fornece o registro histórico do progresso dos alunos;
- Possui um comportamento pró-ativo;
- Gerencia automaticamente, através do comportamento pró-ativo, o envio de avisos sobre a realização de atividades para professores e alunos através do controle de datas;

9.1.3 Utilização do Framework

- O Portfólio-Tutor representa uma importante ferramenta de auxílio ao professor, permitindo a especificação de sessões de ensino on-line;
- Ajuda o professor no processo de ensino / aprendizagem, proporcionando um mecanismo de acompanhamento e de um ensino adaptado ao nível de conhecimento do aluno;

9.1.4 Discussão de Resultados

A versão atual do framework FA_PorT atende aos objetivos desta dissertação e para se tornar um framework completamente implementado, necessita ainda de algumas funcionalidades, como é descrito na seção 9.2.

Conforme visto anteriormente, o framework FA_PorT utiliza uma arquitetura em camadas (Figura 2.1), e, essas camadas foram implementadas da seguinte forma:

- Para a camada Portfólio foi feita uma análise dos sistemas: POETA (Portfólio Eletrônico Temporal e Ativo) (Sistêlos 1999), a camada Portfólio do sistema Portfólio-Tutor (Nascimento 2002) e Portfolio Eletrônico para Web (Silva 2002);
- Para a camada Tutor foi feita uma análise dos sistemas: TUTA (Tutor baseado em Agentes no contexto do Ensino a Distância) (Silva 2000) e a camada Tutor do sistema Portfólio-Tutor (Nascimento 2002);
- Para a camada dos agentes foi feita uma análise do sistema TUTA (Tutor baseado em Agentes no contexto do Ensino a Distância) (Silva 2000) no contexto de agentes reativos;

9.2 Perspectivas

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

- Implementação do restante das funcionalidades e agentes;

- Implementação de mais pontos de adaptação de código;
- Implementação de agentes com base de conhecimento e uma máquina de inferência;

Apêndice A

Metodologia para Análise e Projeto Orientada a Agentes - GAIA

Este apêndice descreve a Metodologia para Análise e Projeto Orientada a Agentes - GAIA, proposta por Wooldrige (Wooldrige et al. 1999) utilizada na Modelagem dos Agentes do framework FA_PorT.

A.1 Introdução

A Metodologia para Análise e Projeto Orientada a Agentes (Wooldrige et al. 1999), consiste em definir um sistema baseado em agentes como uma sociedade ou organização artificial. O sistema pode ser pensado como tendo um comportamento igual a uma organização humana ou uma simples companhia, composta de departamentos e pessoas, ocupando vários cargos. O primeiro passo para desenvolver um sistema baseado em agentes é definir o conjunto de papéis que irão fazer parte do sistema, e para isso deve-se fazer uma analogia a uma organização humana. Esses papéis devem ser definidos através de três atributos, permissões, responsabilidades e protocolos. Estes conceitos são abstratos e servem para conceitualizar o sistema.

GAIA é composta por duas fases, fase de análise e fase de projeto. Durante a fase de análise, a ênfase está nos conceitos abstratos, e no projeto, está nas entidades concretas. Os principais modelos utilizado na metodologia pode ser visto na Figura A.1.

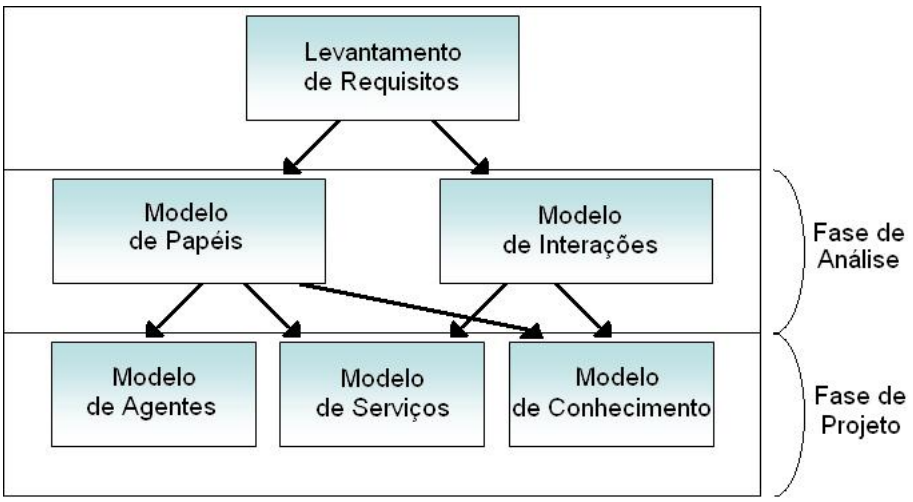


Figura A.1: Modelos da Metodologia GAIA e seus relacionamentos

GAIA utiliza algumas terminologias e notações da análise e projeto da orientação a objetos. Entretanto, não é simplesmente uma tentativa de aplicar tais métodos ao desenvolvimento orientado a agentes.

A.2 Análise

O objetivo da etapa de análise é desenvolver um entendimento do sistema e sua estrutura sem referenciar qualquer detalhe de implementação, ou seja, este objetivo é alcançado através do estudo da organização do sistema. Para definir uma organização é necessário definir os papéis da organização, como estes papéis se relacionam e como um papel pode interagir com outros papéis. Para isso, o modelo de organização pode ser visto como dois modelos: modelo de papéis e modelo de interações.

Devem ser considerados os seguintes passos para o desenvolvimento da fase de análise:

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

2. Para cada papel do sistema, devem ser identificados e documentados os protocolos associados. Em GAIA, protocolos são os padrões de interações que acontecem no sistema entre vários papéis. Para isto deve-se ter um modelo de interação que capture as interações entre os papéis;

3. Através dos protocolos definidos no passo anterior, cria-se um modelo de papéis

para poder documentar os papéis fundamentais que acontecem no sistema, suas permissões, responsabilidades e os protocolos que eles fazem parte.

A.2.1 Modelo de Papéis

O modelo de papéis identificam os papéis chaves no sistema. Este modelo é composto de um conjunto de esquemas de papéis (um para cada papel do sistema). Tais papéis são caracterizados pelos tipos de atributos: os direitos/permissões associados a ele, as suas responsabilidades e os protocolos nos quais participa. Na Figura A.2 temos o modelo de definição do esquema de um papel.

Esquema do papel	Nome do papel
Descrição	*Uma descrição do papel*
Protocolos e Atividades	*Uma parte do protocolo e atividade que o papel utiliza*
Permissões	*"Direitos" de associação com o papel*
Responsabilidades	
Vivacidade	*Responsabilidades de Vivacidade*
Segurança	*Responsabilidades de Segurança*

Figura A.2: Modelo para o Esquema de um Papel

O exemplo de um papel denominado *Enchedor de Café* (Wooldrige et al. 1999) pode ser visto na Figura A.3.

Nome do papel: COFFEEFILLER
Descrição: Este papel envolve assegurar que o pote de café esteja mantido cheio e informar aos trabalhadores que o café fresco está pronto.
Protocolos e Atividades: Fill. InformWorkers. CheckStock. AwaitEmpty
Permissões **reads** supplied *coffeeMaker* // Nome da Maq. de Café *coffeeStatus* // Cheia ou Vazia **changes** *coffeeStock* // Nível do Café
Responsabilidades *Vivacidade:* *COFFEEFILLER* = (Fill. InformWorkers. CheckStock. AwaitEmpty)$^{\omega}$ *Segurança:* *coffeeStock > 0*

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

Os atributos básicos de um papel são:

- **Permissões:** representam os recursos de informação que podem ser utilizados por

um papel para desempenhar suas funcionalidades. Cada papel pode ter a habilidade de ler, modificar ou gerar um item particular de informação;

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

  - **Responsabilidades Vitais:** declara o que será feito por um papel. São expressas através de expressões vitais, e essas expressões definem o ciclo de vida de um papel. A forma geral de uma expressão vital pode ser especificada de acordo com a Figura A.4.

RoleName = *expression*

Figura A.4: Forma Geral de uma Expressão Vital

Os operadores permitidos para a especificação das expressões vitais, podem ser vistos na Figura A.5.

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 A.5: Operadores permitidos para expressões vitais

Será utilizado o exemplo do papel *Enchedor de Café* para poder ilustrar como devem ser definidas tais expressões. Este 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, as responsabilidades vitais para esse papel são: sempre que o café terminar, encha; e, sempre que terminar de preparar o café, avise aos trabalhadores.

O papel *Enchedor de Café*, pode ser expresso como mostra a Figura A.6.

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

Figura A.6: Expressão vital do papel *Enchedor de Café*

Essa expressão (Figura A.6) significa que o papel *Enchedor de Café* executa o protocolo *Encher*, seguido pelos: *InformarTrabalhadores*, *ChecarEstoque* e *EsperarEsvaziar*.

  - **Responsabilidades de Segurança:** São aquelas que um agente deve manter ao desempenhar um papel. No exemplo *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, que serão bastante utilizados no modelo de interações, devido que representam uma interação entre papéis.

A.2.2 Modelo de Interação

O modelo de interações serve para capturar o conjunto das interações que existem entre os diversos papéis especificados no sistema. Este é composto de um conjunto de definições de protocolo, um para cada tipo de interação entre papéis. A Figura A.7 representa o modelo utilizado para definição de protocolos entre papéis.

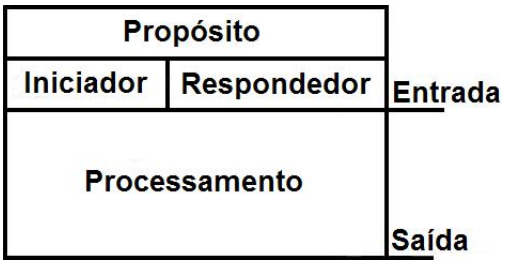


Figura A.7: Modelo para definição de um protocolo

Na definição de protocolo os seguintes atributos devem ser considerados:

- Propósito: descrição breve da natureza da interação (ex.: solicitação de informação, designação de tarefas, dentre outras);
- Iniciador: o(s) papel(éis) responsável(eis) para iniciar a interação;
- Respondedor: o(s) papel(éis) com que o iniciador interage;
- Entradas: informação usada pelo iniciador, enquanto o protocolo é realizado;
- Saídas: informação fornecida 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.

A figura A.8 demonstra que o protocolo *Encher* faz parte do papel *Enchedor Café*. Este protocolo é iniciado por este papel e envolve o papel *MáquinaDeCafé*. O protocolo especifica que o papel *Enchedor Café* enche o café na máquina chamada *Cafeteira* e o resultado é que *MáquinaDeCafé* fica informada sobre o valor do *EstoqueCafé*.

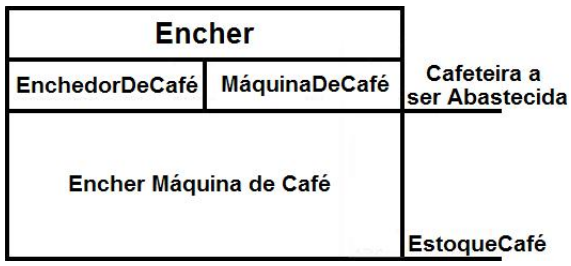


Figura A.8: Definição do protocolo *Encher*

A.3 Projeto

O objetivo de um processo de projeto é transformar os modelos abstratos derivados durante a etapa de análise em modelos em um nível suficientemente baixo de abstração, de tal modo que técnicas de projeto tradicionais possam ser aplicados. Ou seja, após a realização das fases de análise e projeto, é necessário passar por uma nova fase de projeto, onde deve tornar possível a realização da implementação do sistema.

O processo de projeto em GAIA envolve três modelos: o modelo de agente, o modelo de serviço e o modelo de conhecimento.

A.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 equivale a apenas um tipo de agente ou escolher empacotar vários papéis em um mesmo tipo de agente.

A notação utilizada para especificar tipos de agentes pode ser vista na Figura A.9.

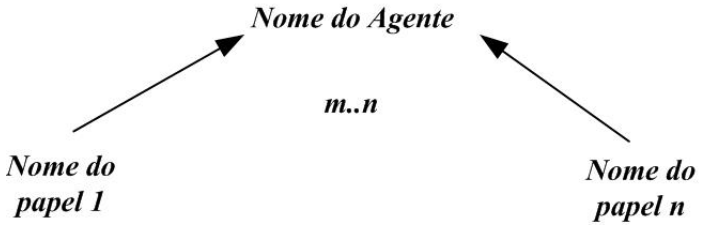


Figura A.9: Modelo para especificação de um tipo de agente

As seguintes cardinalidades podem ser utilizadas para representar o número de instâncias dos agentes em tempo de execução:

- n: significa que existirão n instâncias deste tipo no sistema em tempo de execução;
- $m..n$: significa que existirão de m a n instâncias deste tipo no sistema em tempo de execução;

- *: significa que poderão existir nenhum ou mais instâncias em tempo de execução;
- +: significa que poderão existir uma ou mais instâncias em tempo de execução.

Exemplos de especificações de tipos de agentes são ilustradas na Figura A.10.

Figura A.10: Exemplos de Especificações de tipos de agentes

A.3.2 Modelo de Serviços

Como seu nome sugere, o objetivo do modelo de serviços é identificar os serviços associados a cada papel e especificar as principais propriedades destes serviços (associado a cada papel de agente). Em termos de Orientação a Objetos, um serviço corresponderia a um método, entretanto, os métodos de um objeto estão disponíveis para outros objetos invocarem e este não é o objetivo da Orientação a Agentes (OA). Isto é, os serviços de um agente não devem estar disponíveis para outros agentes.

Para cada serviço que um agente possa executar, é necessário documentar suas propriedades. Especificamente devemos identificar as entradas, saídas, pré-condições e pós-condições de cada serviço.

Os serviços que um agente executará são derivados da lista de protocolos e responsabilidades associadas a um papel, a partir das responsabilidades vitais de um papel. As entradas e saídas podem ser derivadas de um meio óbvio a partir da análise dos protocolos no modelo de interações, enquanto que as pré-condições e pós-condições representam restrições nos serviços.

Retornando ao exemplo do café, há quatro protocolos associados com o papel *Enchedor de Café*, são eles: *Encher*, *InformarTrabalhadores*, *ChecarEstoque* e *EsperarVaziar*. Em geral, há ao menos um serviço associado a cada protocolo. No caso de *ChecarEstoque*, o serviço (que pode ter o mesmo nome) tomará como entrada o nível de estoque e algum valor similar e simplesmente comparará os dois. As pré-condição e pós-condição declararão que o nível de estoque de café será maior que zero. Isto é uma das propriedades de segurança do papel.

Após especificar os serviços de um sistema, o desenvolvedor é livre para compreender os serviços em qualquer estrutura de implementação que julgar apropriado. Poderá, por exemplo, implementar os serviços através de métodos em uma linguagem orientados a objetos.

A.3.3 Modelo de Comunicação

O último modelo da fase de projeto em GAIA é o modelo de conhecimento. Este simplesmente define os elos de comunicação que existem entre os diversos tipos de agente. Em particular, o propósito de um modelo de conhecimento é identificar quaisquer gargalos de comunicação que possam causar problemas em tempo de execução.

O modelo de conhecimento é representado por um grafo, onde os nós correspondem ao tipos de agentes e os arcos correspondem aos caminhos de comunicação que existem entre os agentes. Estes modelos são gráficos direcionados, então um arco $a \longrightarrow b$ indica que a enviará mensagens para b, mas não necessariamente que b enviará mensagens para a.

Um modelo de conhecimento pode ser derivado diretamente dos papéis, protocolos e modelo de agentes. Um exemplo de modelo de comunicação pode ser visto na Figura A.11.

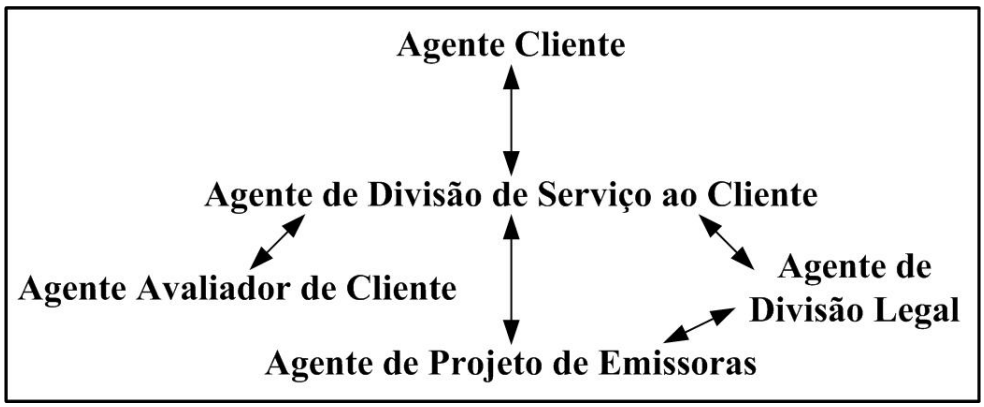


Figura A.11: Exemplo de Modelo de Comunicação para Gerência de Negócios Empresariais

Apêndice B

Ferramentas de Implementação de Agentes

Este apêndice descreve algumas ferramentas para implementação de sistemas baseados em agentes.

B.1 Framework JADE

JADE (*Java Agent DEvelopment framework*) (JADE 2006) é um ambiente para desenvolvimento de aplicações baseado em agentes de acordo com as especificações da FIPA (FIPA 2006)(*Foundation for Intelligent Physical Agents*).

As especificações da FIPA servem para facilitar e possibilitar a comunicação entre os agentes. Seus serviços são o serviço de nomes (*naming service*), as páginas amarelas (*yellow-page service*), o transporte de mensagens, os serviços de codificação, a decodificação de mensagens e uma biblioteca de interação (padrão FIPA). Toda a comunicação entre os agentes é feita via troca de mensagens.

B.1.1 Características de JADE

De Acordo com Silva (Silva 2003), as principais características que JADE fornece para a programação de sistemas multiagentes são:

- Plataforma distribuída de agentes: JADE pode ser dividido em várias máquinas (desde que eles possam ser conectados via RMI[1](*Remote Method Invocation*)). Os agentes são implementados em java através de *threads* e colocados em um repositório de agentes (*Agent Containers*) que provêm todo o suporte para a sua execução;

[1]É um conjunto de classes e interface em Java que encapsulam vários mecanismos de troca de dados, com o objetivo de simplificar a execução de chamadas de métodos remotamente localizados em espaços de endereçamento diferente.

- GUI ou *Graphical User Interface*: interface gráfica que gerencia os agentes e containers de agentes;
- Ferramentas de *Debugging*: ferramentas que auxiliam no desenvolvimento e depuração de aplicações multiagentes;
- Suporte a execução das atividades dos agentes: através dos modelos de comportamentos;
- Ambiente de agentes: adota o sistema gerenciados de agentes (AMS - *Agent Management System*), o diretório facilitador (DF - Directory Facilitator) e o canal de comunicação de agentes (ACC - Agent Communication Channel), que são automaticamente carregados quando o JADE é iniciado;
- Transporte de mensagens: o transporte das mensagens se dá no formato FIPA-ACL dentro da mesma plataforma de agentes;
- Biblioteca de protocolos FIPA: possui uma biblioteca de protocolos para a interação entre os agentes JADE;
- Automação de registros: registro e cancelamento automático de agentes com o AMS;
- Serviços de nome (*Naming Service*) em conformidade aos padrões FIPA: os agentes quando inicializados obtêm seus GUID (*Globally Unique Identifier*) da plataforma, que são identificadores únicos em todo o ambiente;
- Integração: mecanismo que permite que outros sistemas carreguem agentes autônomos em JADE.

Além dessas características, JADE possui também algumas ferramentas com o objetivo de simplificar a administração da plataforma de agentes.

B.1.2 JADE e FIPA

O modelo de plataforma padrão especificado pela FIPA (FIPA 2006), é mostrado na Figura B.1.

A Figura B.1 é composta pelas seguintes estruturas:

- Software: sistema multiagentes;
- O Agente (*Agent*): é o agente que possuirá tarefas de acordo com a aplicação. Encontra-se dentro da plataforma de agentes (*Agent Plataform*) e realiza toda sua comunicação com agentes através das trocas de mensagens;

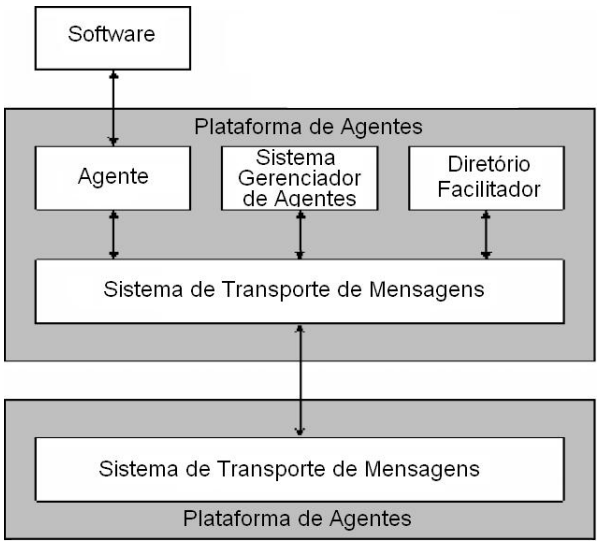


Figura B.1: Modelo padrão da plataforma dos agentes definido pela FIPA

- O Sistema Gerenciador de Agentes (AMS - *Agent Management System*): é o agente que supervisiona o acesso e o uso da plataforma de agentes. Só existe um AMS em cada aplicação. Ele fornece guia de endereços e controle de ciclo de vida, mantendo um pasta de identificadores de agentes (AID[2] - *Agent Identifier*) e estados de agentes, que é responsável por autenticar os agentes e controlar os registros. Cada agente deve se registrar no AMS para obter um AID válido;

- O Diretório Facilitador (DF - *Directory Facilitator*): é o agente que fornece o serviço de páginas amarelas (*yellow pages*) dentro da plataforma;

- O Sistema de Transporte de Mensagens (*Message Transport System*): também é conhecido como canal de comunicação dos agentes (ACC - *Agent Communication Channel*), é o agente responsável por fornecer toda a comunicação entre os agentes dentro e fora da plataforma.

B.1.3 Arquitetura Interna do JADE

A arquitetura de um sistema multiagente em JADE (Silva 2003), pode ser visto na Figura B.2.

A arquitetura do JADE é baseada na existência simultânea de máquinas virtuais Java (JVM - *java Virtual Machine*). Na figura B.2 temos uma presença distribuída da plataforma de agentes JADE distribuída em 3 máquinas ou *hosts*. Em cada *host* existe uma JVM para ressaltar o conceito de independência de plataforma. Em cada JVM existe um container de agentes que fornece um ambiente completo para sua execução, além de admitir que muitos agentes possam ser executados concorrentemente no mesmo *host*. E, a

[2]É uma classe de JADE qua atribui identificadores único aos agentes.

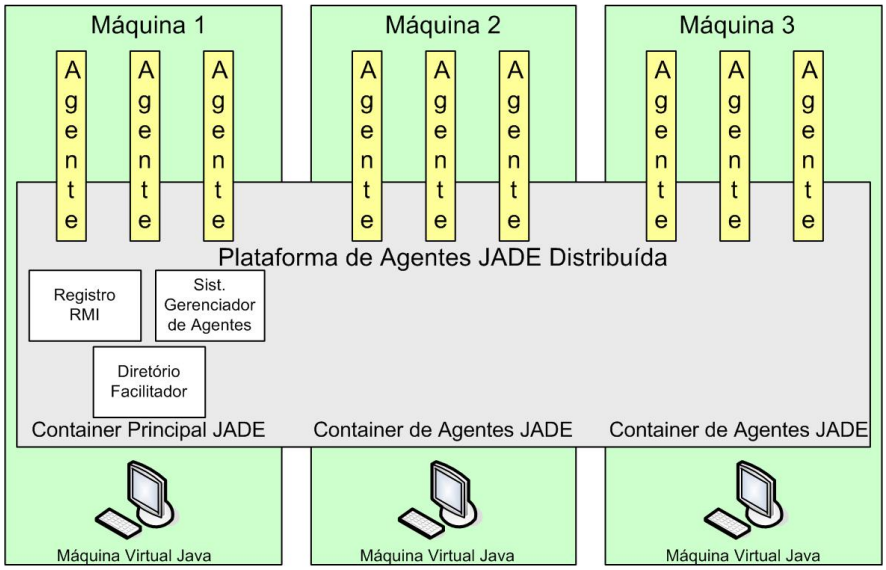


Figura B.2: Arquitetura JADE

comunicação entre JVMs é feita através de invocação remota de métodos (registro RMI) de Java.

B.1.4 Troca de Mensagens

A troca de mensagens realizadas no JADE é feita através de métodos próprios e pela instância da classe *ACLMessage*. Esta possui um conjunto de atributos que estão de acordo com as especificações da FIPA, implementando a linguagem FIPA-ACL, que consiste de um conjunto de tipos de mensagens. Além da ACL, a FIPA também especifica uma notação para descrição de protocolos de comunicação. Esta notação é exemplificada com protocolos para requisição de serviços, rede contratual e outros (FIPA 2006).

Podemos ver um exemplo de comunicação entre os agentes no Apêndice D.

B.2 AgentBuilder

AgentBuilder (AgentBuilder 2006) fornece ferramentas gráficas para suportar todas as fases do processo da construção de um SMA. Essas ferramentas dão suporte para: organizar e controlar o projeto do desenvolvimento; analisar o domínio do problema; especificar protocolos de interação; definir a arquitetura do sistema multiagente; examinar o funcionamento da sociedade; especificar o comportamento do agente; criar agentes executáveis; e depurar erros. O agente genérico da ferramenta AgentBuilder, basei-se nas arquiteturas BDI descritas previamente. Em AgentBuilder, um agente é caracterizado pelos seguintes conceitos que conformam o chamado modelo mental do agente:

- **Estado**: representa o estado atual do ambiente do agente. Esse estado é atualizado

conforme as percepções do agente;

- **Habilidades**: descreve as ações que o agente pode executar quando condições pré-definidas são satisfeitas;
- **Negociação**: é um acordo feito com outro agente para que uma ação seja executada em um determinado instante;
- **Regras de comportamento**: podem ser vistas como sentenças do tipo: quando se então. Ou quando da regra se aplica a novos eventos ocorrendo no ambiente de um agente, como novas mensagens recebidas de outros agentes. O se compara o estado atual com condições que são requeridas para uma regra ser aplicável. O então define as ações do agente executadas em resposta a um evento;
- **Intenções**: são similares a negociações, em que um agente executa ações em nome de outro agente. No entanto, uma negociação é um acordo para executar uma ação simples, já uma intenção é um acordo para executar quaisquer ações que são necessárias para atingir um estado desejado do ambiente.

O conjunto dessas características forma a Arquitetura do Agente Inteligente Reticular, a arquitetura do agente genérico da ferramenta AgentBuilder, ilustrada na Figura B.3. Um interpretador monitora continuamente as mensagens que chegam, atualiza o modelo mental do agente e executa ações. No começo do processo, o agente é inicializado com crenças, intenções, habilidades e regras de comportamento iniciais. Um agente não trivial requer pelo menos uma regra de comportamento, os outros elementos são opcionais. O modelo mental contém as crenças, intenções, habilidades e regras atuais do agente.

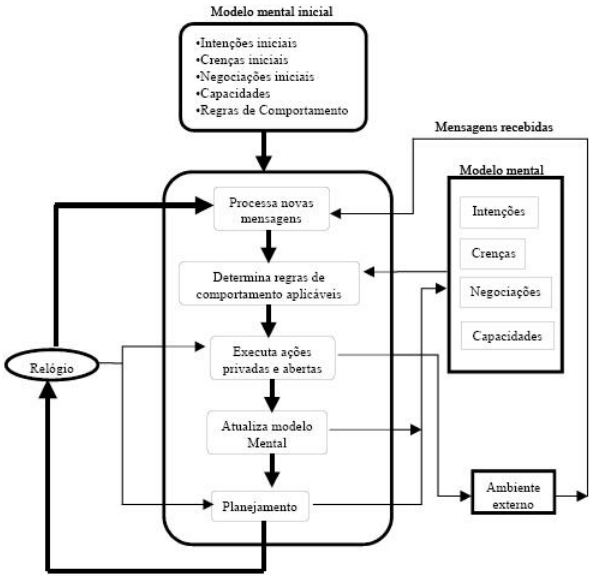


Figura B.3: Arquitetura do agente genérico da ferramenta AgentBuilder

B.3 Zeus

A ferramenta ZEUS (Zeus 2006) consiste de um conjunto de componentes, escritos na linguagem de programação Java, que podem ser categorizados em três grupos funcionais (ou bibliotecas): uma biblioteca de componentes agentes, uma ferramenta para a construção de agentes e um grupo de agentes de utilidade que incluem os agentes name-server, facilitator e visualiser.

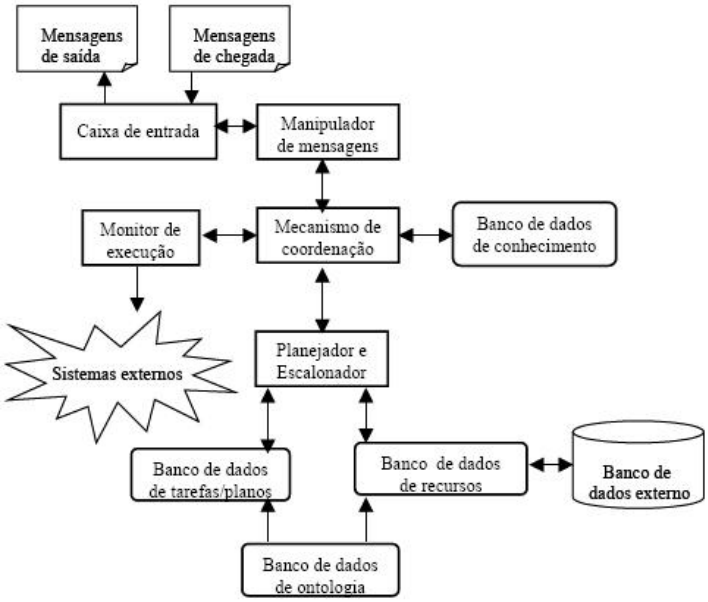


Figura B.4: Arquitetura do agente genérico Zeus

De acordo com a Figura B.4, o agente genérico Zeus inclui os seguintes componentes:

- Uma caixa de correio que manipula as comunicações entre o agente genérico e outros agentes;
- Um manipulador de mensagens que processa as mensagens que chegam na caixa de correio, despachando-as para os componentes relevantes do agente genérico;
- Um mecanismo de coordenação que toma decisões a respeito das tarefas dos agentes e também é responsável pela coordenação das interações com outros agentes;
- Um banco de dados de conhecimento que descreve as relações do agente genérico com outros agentes na sociedade e suas crenças sobre as habilidades desses agentes;
- Um planejador e escalonador que planeja as tarefas dos agentes, baseadas nas decisões tomadas pelo mecanismo de coordenação;
- Um banco de dados de recursos que mantém uma lista de recursos que pertencem e estão disponíveis ao agente genérico;

- Um banco de dados de ontologias que armazena a definição lógica de cada tipo de atributo;
- Um banco de dados de tarefas/planos que fornece uma descrição lógica das tarefas disponíveis nos agentes;
- Um monitor de execução que é responsável pelo relógio interno do agente genérico. Ele inicia, suspende e monitora as atividades programadas pelo Planejador / Escalonador. Também informa ao Planejador sobre as condições de término satisfatórias e excepcionais das tarefas que estão sendo monitoradas.

Apêndice C

Implementação do Controle do Framework e de suas Aplicações

Este apêndice descreve a implementação do controle do framework FA_PorT e as aplicações.

C.1 Classe *Framework*

O Código C.1 é referente ao controle do framework FA_PorT.

Código C.1: Implementação do Controle do Framework

```
1  package framework;
2
3  public abstract class Framework {
4
5    //Componentes do Framework
6    private InterfaceElementosAdministrativos componenteElementosAdministrativos;
7    private InterfaceBaseDeDominio componenteBaseDeDominio;
8    private InterfaceSeguranca componenteSeguranca;
9    private InterfaceRegistros componenteRegistro;
10   private InterfaceComunicacao componenteComunicacao;
11   private InterfacePerfil componentePerfil;
12   private InterfaceEstrategiasDidaticas componenteEstrategiasDidaticas;
13   private InterfaceTaticaEnvio componenteTaticaEnvio;
14
15   //Tempo para envio de avisos sobre vencimento de atividades em dias
16   private int tempoEnvio;
17
18   //Construtor
19   public Framework(int tempo) {
20     componenteElementosAdministrativos = new ComponenteElementosAdministrativos();
21     componenteBaseDeDominio = new ComponenteBaseDeDominio();
22     componenteSeguranca = new ComponenteSeguranca();
23     componenteRegistro = new ComponenteRegistros();
24     componenteComunicacao = new ComponenteComunicacao();
25     componentePerfil = new ComponentePerfil();
26     componenteEstrategiasDidaticas = new ComponenteEstrategiasDidaticas();
27     componenteTaticaEnvio = new ComponenteTaticaEnvio();
28     this.tempoEnvio = tempo;
29     this.templateMethod();
30   }
31
32   //Metodo customize para representar o desempenhos dos alunos em graficos
33   public abstract JFreeChart customizeGrafico(int aprovados, int reprovados, List desempenhos);
34
35   //Metodo customize para especificar a maneira como as mensagens sao mostradas aos alunos
36   public abstract void comunicacaoAluno(Integer codigoAluno, Integer codigoAtividade, String mensagem);
37
```

```
38    public void templateMethod(){
39      //Controle da camada Portfolio
40      //Envio de informacoes sobre as atividades dos alunos
41      List atividadesAlunos = null;
42      Aluno aluno = null;
43      Atividade atividade = null;
44      try{
45        atividadesAlunos = componenteRegistro.getAtividadesAluno();
46      }catch (Exception e) { System.out.println(e.getMessage()); }
47      if(atividadesAlunos != null){
48        for(int i = 0; i < atividadesAlunos.size(); i++){
49          AtividadeAluno atividadeAluno = (AtividadeAluno)atividadesAlunos.get(i);
50          if(atividadeAluno.getEnviou().equals(new Integer(0))){
51            try{
52              aluno = componenteElementosAdministrativos.getAluno(atividadeAluno.getAluno().getCodigo());
53              atividade = componenteRegistro.getAtividade(atividadeAluno.getAtividade().getCodigo());
54            }catch (Exception e) { System.out.println(e.getMessage()); }
55            String data = atividadeAluno.getDataEntrega();
56            Date a = new Date();
57            String formato = "dd/MM/yyyy";
58            SimpleDateFormat dataFormatada = new SimpleDateFormat(formato);
59            a.setTime(a.getTime() + (24 * this.tempoEnvio) * 60 * 60 * 1000);
60            if(data.equals(dataFormatada.format(a))){
61              try{
62                String aviso =
63                  "A atividade " + atividade.getDescricao() + " da disciplina (" + atividade.
64                  getDisciplina().getNome() + ") referente a unidade (" + atividade.getUnidade().
65                  getDescricao() + ") deve ser entregue na data " + data + ".";
66
67                // Aqui e chamado o metodo customize  - Comunicacao com o Aluno.
68                this.comunicacaoAluno(aluno.getCodigo(), atividade.getCodigo(), aviso);
69              }catch(Exception e){ System.out.println(e.getMessage()); }
70            }
71          }
72        }
73      }
74
75      //Controle da camada Tutor
76      //Perfil Individual e Perfil do grupo
77      try{
78        List alunos = componenteElementosAdministrativos.getAlunos();
79        List itens = componenteRegistro.getItensAvaliar();
80        componentePerfil.excluirTodosPerfil();
81        for(int i = 0; i < alunos.size(); i++){
82          aluno = (Aluno)alunos.get(i);
83          for(int j = 0; j < itens.size(); j++){
84            ItemAvaliar itemAvaliar = (ItemAvaliar)itens.get(j);
85            List desempenhos = componenteRegistro.getDesempenhosAlunoItem(aluno.getCodigo(),
86              itemAvaliar.getCodigo());
87            if(desempenhos != null && desempenhos.size() != 0){
88              float soma = 0;
89              float count = 0;
90              for(int k = 0; k < desempenhos.size(); k++){
91                DesempenhoAluno desempenho = (DesempenhoAluno)desempenhos.get(k);
92                soma = soma + desempenho.getNota();
93                count++;
94              }
95              PerfilIndividual perfil = new PerfilIndividual();
96              perfil.setAluno(aluno);
97              perfil.setItemAvaliar(itemAvaliar);
98              perfil.setMedia(soma/count);
99              componentePerfil.inserirPerfil(perfil);
100           }   }   }
101     }catch (Exception e) { System.out.println(e.getMessage()); }   }
102
103   //Controle das taticas de ensino
104   public RecursoDidatico controleEstrategias(Integer codigoEstrategia) throws Exception{
105     InterfaceEstrategiasDidaticas componentesEstrategias = new ComponenteEstrategiasDidaticas();
106     InterfaceTaticaReuso componenteTaticaReuso = new ComponenteTaticaReuso();
107     Integer codigoTatica = componentesEstrategias.getCodigoProximaTatica(codigoEstrategia);
108     List taticas = componentesEstrategias.getTaticas(codigoEstrategia);
109     Tatica taticaAtual = null;
110     for(int j = 0; j < taticas.size(); j++){
111       Tatica taticaAux = (Tatica)taticas.get(j);
112       if(taticaAux.getCodigo().equals(codigoTatica) && taticaAux.getFinalizada().equals(new Integer(0)))
113       { taticaAtual = taticaAux;
114         break;
115       }
116     }
```

```
117     if(taticaAtual != null){
118       if(taticaAtual.getCodigo().equals(new Integer(1))&&taticaAtual.getFinalizada().
119           equals(new Integer(0))){
120         return null;
121       }
122       if(taticaAtual.getCodigo().equals(new Integer(2))){
123         RecursoDidatico recurso = componenteTaticaReuso.executarTatica(codigoEstrategia,
124           taticaAtual.getCodigo(), taticaAtual.getOrdem().intValue());
125        return recurso;
126       }
127     }
128     return null;
129   }
130 }
```

C.2 Classe *Aplicacao1*

O Código C.2 é referente a primeira aplicação.

Código C.2: Implementação da Primeira Aplicação

```
1  package framework;
2
3  package principal;
4
5  import java.util.*;
6  import org.jfree.chart.*;
7  import org.jfree.data.category.DefaultCategoryDataset;
8  import portfolio.elementosAdministrativos.*;
9  import portfolio.registros.*;
10 import com.opensymphony.webwork.ServletActionContext;
11 import framework.Framework;
12
13 public class Aplicacao1 extends Framework{
14
15   // Padrao de projeto singleton para deixar apenas um objeto desta classe instanciado
16   private static Aplicacao1 instance;
17
18   public static Aplicacao1 getInstance(){
19     if(instance == null){
20       instance = new Aplicacao1();
21     }
22     return instance;
23   }
24
25   // O inteiro passado no argumento e a quantidade de dias que antecede o envio do lembrete ao aluno a
26   // repeito do vencimento de suas atividades
27   private Aplicacao1() {
28     super(2);
29   }
30
31   // Metodo customize 1 - Mostra mensagens aos alunos a repeito de suas atividades no quadro de avisos
32   public void comunicacaoAluno(Integer codigoAluno, Integer codigoAtividade, String msg) {
33     try{
34       Aluno aluno = this.getAluno(codigoAluno);
35       this.inserirAviso(aluno.getCodigo(), msg);
36       this.atualizarEnvio(codigoAluno, codigoAtividade);
37     }catch (Exception e) {
38       System.out.println(e.getMessage());
39     }
40   }
41
42   // Metodo customize 2 - Grafico de pizza para representar o desempenhos dos alunos
43   public JFreeChart customizeGrafico(int aprovados, int reprovados, List desempenhos) {
44     DefaultPieDataset pieDataset = new DefaultPieDataset();
45
46     pieDataset.setValue("Reprovados (%)", new Integer(reprovados));
47     pieDataset.setValue("Aprovados (%)", new Integer(aprovados));
48
49     JFreeChart chart = ChartFactory.createPieChart("Desempenho dos Alunos", pieDataset, true, true, true);
50
51     chart.setBackgroundPaint(Color.WHITE);
52
```

```
53      ChartPanel chartPanel = new ChartPanel(chart);
54      chartPanel.setPreferredSize(new java.awt.Dimension(500, 270));
55
56      return chart;
57    }
58
59    public Usuario getUsuario(){
60      return (Usuario)ServletActionContext.getContext().getSession().get("usuario");
61    }
62
63    public void setUsuario(Usuario usuario) {
64      ServletActionContext.getContext().getSession(). put("usuario", usuario);
65    }
66  }
```

C.3 Classe *Aplicacao2*

O Código C.3 é referente a segunda aplicação.

Código C.3: Implementação da Segunda Aplicação

```
1   package principal;
2
3   import java.util.*;
4   import org.jfree.chart.*;
5   import org.jfree.data.category.DefaultCategoryDataset;
6   import portfolio.elementosAdministrativos.*;
7   import portfolio.registros.*;
8   import com.opensymphony.webwork.ServletActionContext;
9   import framework.Framework;
10
11  public class Aplicacao2 extends Framework{
12
13    // Padrao de projeto singleton para deixar apenas um objeto desta classe instanciado
14    private static Aplicacao2 instance;
15    public static Aplicacao2 getInstance(){
16      if(instance == null){
17        instance = new Aplicacao2();
18      }
19      return instance;
20    }
21
22    // O inteiro passado no argumento e a quantidade de dias que antecede o envio do lembrete ao aluno a
23    // respeito do vencimento de suas atividades
24    private Aplicacao2() {  super(2);  }
25
26    // Metodo Customize 1 - Mostra mensagens aos alunos a respeito de suas atividades por Email
27    public void comunicacaoAluno(Integer codigoAluno, Integer codigoAtividade, String msg) {
28      try{
29        Aluno aluno = this.getAluno(codigoAluno);
30        this.enviarEmail(aluno.getEmail(), "Lembrete de Atividade", msg);
31        this.atualizarEnvio(codigoAluno, codigoAtividade);
32      }catch (Exception e) { System.out.println(e.getMessage()); }
33    }
34
35    // Metodo Customize 2 - Grafico de barras para representar o desempenhos dos alunos
36    public JFreeChart customizeGrafico(int aprovados, int reprovados, List desempenhos) {
37      Aplicacao2 application = Aplicacao2.getInstance();
38      List alunos = null;
39      try{
40        alunos = application.getAlunos();
41      }catch (Exception e) { System.out.println(e.getMessage()); }
42      DefaultCategoryDataset dataset = new DefaultCategoryDataset();
43      for(Iterator j = alunos.iterator(); j.hasNext();){
44        Aluno aluno = (Aluno)j.next();
45        float totalNotas = 0;
46        int total = 0;
47        for(Iterator i = desempenhos.iterator(); i.hasNext();){
48          DesempenhoAluno desempenho = (DesempenhoAluno)i.next();
49          if(desempenho.getAluno().getCodigo().equals(aluno.getCodigo())){
50            ItemAvaliar item = null;
51            try{
52              item = application.getItemAvaliar(desempenho.getItemAvaliar().getCodigo());
```

```
53              }catch (Exception e) {     System.out.println(e.getMessage()); }
54              total = total + item.getPeso().intValue();
55              totalNotas = totalNotas + desempenho.getNota() * item.getPeso().intValue();
56            }
57          }
58          if(total != 0 && totalNotas != 0){
59            dataset.setValue(totalNotas/total, aluno.getNome(), "");
60          }
61        }
62        JFreeChart chart = ChartFactory.createBarChart("Desempenho dos Alunos","", "Nota", dataset,
63          PlotOrientation.VERTICAL, true, true, true);
64        return chart;
65      }
66
67      public Usuario getUsuario(){
68        return (Usuario)ServletActionContext.getContext().getSession().get("usuario");}
69      public void setUsuario(Usuario usuario) {
70        ServletActionContext.getContext().getSession().put("usuario", usuario);}
71  }
```

Apêndice D

Implementação dos Agentes do Framework

Este apêndice descreve a implementação dos agentes que fazem parte do framework FA_PorT.

D.1 Agente Executor de Sessão

O Código D.1 possui o objetivo de controlar uma sessão de ensino, inicializando-a, controlando as táticas de ensino (de acordo com o tempo) e finalizando-a.

Código D.1: Implementação do Agente Executor de Sessão

```
1  package agentes;
2
3  import jade.Boot;
4  import jade.core.*;
5  import jade.domain.*;
6  import jade.domain.FIPAAgentManagement.*;
7  import jade.lang.acl.ACLMessage;
8  import java.sql.*;
9  import java.text.SimpleDateFormat;
10 import java.util.*;
11 import tutor.estrategiasDidaticas.SessaoEnsino;
12
13 public class AgenteExecutorSessao extends Agent{
14
15   private Connection connection;
16   private ResultSet resultSet;
17   private PreparedStatement statement;
18
19   // Metodo de inicializacao do agente
20   public void setup(){
21     this.cadastroNasPaginasAmarelas("executorSessao");
22     AgenteExecutorSessaoRecebedorMsg recebedorDeMensagensRecebedor =
23       new AgenteExecutorSessaoRecebedorMsg(this);
24     this.addBehaviour(recebedorDeMensagensRecebedor);
25   }
26
27   //Metodo para executar comandos sql no banco
28   public ResultSet executeQuery(String sql) throws Exception{
29     resultSet = null;
30     statement = null;
31     connection = null;
32     Class.forName("com.mysql.jdbc.Driver");
33     connection = DriverManager.getConnection("jdbc:mysql://localhost/faport", "root", "12345");
34     statement = connection.prepareStatement(sql);
35     resultSet = statement.executeQuery();
```

```
36      return resultSet;
37    }
38
39    //Metodo para executar atualizacoes no banco
40    public int executeUpdate(String sql) throws Exception{
41      resultSet = null;
42      statement = null;
43      connection = null;
44      Class.forName("com.mysql.jdbc.Driver");
45      connection = DriverManager.getConnection("jdbc:mysql://localhost/faport", "root", "12345");
46      statement = connection.prepareStatement(sql);
47      return statement.executeUpdate();
48    }
49
50    //Metodo para fechar conexao com o banco
51    public void close() throws Exception{
52      if (resultSet != null)       resultSet.close();
53      if (statement != null)     statement.close();
54      if (connection != null)  connection.close();
55    }
56
57    //Metodo que trata a ordem enviada pelo agenteComunicacao
58    public void tratarMensagem(ACLMessage message){
59      try{
60        //Seleciona todas as sessoes nao inicializadas e nao terminadas
61        List sessoesEnsino = new ArrayList();
62        resultSet = this.executeQuery("SELECT * FROM sessao_ensino WHERE terminada = 0 AND iniciada = 0;");
63        while(resultSet.next()){
64          SessaoEnsino sessaoEnsino = new SessaoEnsino();
65          sessaoEnsino.setCodigo(new Integer(resultSet.getInt("codigo")));
66          sessaoEnsino.setDataInicio(resultSet.getString("data"));
67          sessaoEnsino.setHora(resultSet.getString("hora"));
68          sessaoEnsino.setDescricao(resultSet.getString("descricao"));
69          sessaoEnsino.setCodigoDominio(new Integer(resultSet.getInt("codigoDominio")));
70          sessoesEnsino.add(sessaoEnsino);
71        }
72        this.close();
73        Date data = new Date();
74        SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
75        for(int i = 0; i < sessoesEnsino.size(); i++){
76          SessaoEnsino sessao = (SessaoEnsino)sessoesEnsino.get(i);
77          String dataInicial = sessao.getDataInicio();
78          String dataAtual = formato.format(data);
79          String[] dataInicialAux = dataInicial.split("/");
80          String[] dataAtualAux = dataAtual.split("/");
81          String dataAtualDia = dataAtualAux[0];
82          String dataInicialDia = dataInicialAux[0];
83          String dataAtualMes = dataAtualAux[1];
84          String dataInicialMes = dataInicialAux[1];
85          String dataAtualAno = dataAtualAux[2];
86          String dataInicialAno = dataInicialAux[2];
87
88          //Verifica as datas para inicializar as sessoes de ensino
89          if(Integer.parseInt(dataInicialAno) == Integer.parseInt(dataAtualAno)){
90            if(Integer.parseInt(dataInicialMes) == Integer.parseInt(dataAtualMes)){
91              if(Integer.parseInt(dataInicialDia) == Integer.parseInt(dataAtualDia)){
92                String horaInicialCompleta = sessao.getHora();
93                String[] auxInicial = horaInicialCompleta.split(":");
94                String horaInicial = auxInicial[0];
95                String minutoInicial = auxInicial[1];
96
97                Calendar calendar = new GregorianCalendar();
98                int horaAtual = calendar.get(Calendar.HOUR_OF_DAY);
99                int minutoAtual = calendar.get(Calendar.MINUTE);
100
101               //Inicializa as sessoes de ensino na hora de inicio
102               if(horaAtual == Integer.parseInt(horaInicial) &&
103                   minutoAtual == Integer.parseInt(minutoInicial)){
104                 try{
105                   this.executeUpdate("UPDATE sessao_ensino SET iniciada = 1 WHERE terminada = 0 AND"
106                   + " codigo = " + sessao.getCodigo() + ";");
107                   this.close();
108                 }catch (Exception e) {  System.out.println(e.getMessage());  }
109               }
110             }
111           }
112         }
113       }
114
```

```
115         //Finalizar taticas da sessao de ensino
116         sessoesEnsino = new ArrayList();
117 
118         //Seleciona todas as sessoes de ensino nno finalizadas
119         resultSet = this.executeQuery("SELECT * FROM sessao_ensino WHERE terminada = 0 AND iniciada = 1;");
120         while(resultSet.next()){
121           SessaoEnsino sessaoEnsino = new SessaoEnsino();
122           sessaoEnsino.setCodigo(new Integer(resultSet.getInt("codigo")));
123           sessaoEnsino.setDataInicio(resultSet.getString("data"));
124           sessaoEnsino.setHora(resultSet.getString("hora"));
125           sessaoEnsino.setDescricao(resultSet.getString("descricao"));
126           sessaoEnsino.setCodigoDominio(new Integer(resultSet.getInt("codigoDominio")));
127           sessoesEnsino.add(sessaoEnsino);
128         }
129         this.close();
130 
131         for(int i = 0; i < sessoesEnsino.size(); i++){
132           SessaoEnsino sessao = (SessaoEnsino)sessoesEnsino.get(i);
133           int codigoEstrategia = 0;
134 
135           //Seleciona o codigo de estrategia didatica
136           resultSet = this.executeQuery("select codigoEstrategia from sessao_ensino_estrategia where "
137             + "codigoSessao = " + sessao.getCodigo() + ";");
138           if(resultSet.next()){
139              codigoEstrategia = resultSet.getInt("codigoEstrategia");
140           }
141           this.close();
142 
143           String dataInicial = sessao.getDataInicio();
144           String dataAtual = formato.format(data);
145           String[] dataInicialAux = dataInicial.split("/");
146           String[] dataAtualAux = dataAtual.split("/");
147           String dataAtualDia = dataAtualAux[0];
148           String dataInicialDia = dataInicialAux[0];
149           String dataAtualMes = dataAtualAux[1];
150           String dataInicialMes = dataInicialAux[1];
151           String dataAtualAno = dataAtualAux[2];
152           String dataInicialAno = dataInicialAux[2];
153           if(Integer.parseInt(dataInicialAno) == Integer.parseInt(dataAtualAno)){
154             if(Integer.parseInt(dataInicialMes) == Integer.parseInt(dataAtualMes)){
155               if(Integer.parseInt(dataInicialDia) == Integer.parseInt(dataAtualDia)){
156                 String horaInicialCompleta = sessao.getHora();
157                 String[] auxInicial = horaInicialCompleta.split(":");
158                 String horaInicial = auxInicial[0];
159                 String minutoInicial = auxInicial[1];
160 
161                 Calendar calendar = new GregorianCalendar();
162                 int horaAtual = calendar.get(Calendar.HOUR_OF_DAY);
163                 int minutoAtual = calendar.get(Calendar.MINUTE);
164 
165                 //Seleciona a tatica atual de uma determinda sessao de ensino
166                 resultSet = this.executeQuery("select min(ordem) from estrategias_taticas where "
167                   + "finalizada = 0 and codigoEstrategia = " + codigoEstrategia + ";");
168                 int ordemTaticaAtual = 0;
169                 if(resultSet.next()){
170                   ordemTaticaAtual = resultSet.getInt(1);
171                 }
172                 this.close();
173 
174                 //Seleciona o tempo total da sessao de ensino ate a tatica atual
175                 resultSet = this.executeQuery("select sum(tempo) from estrategias_taticas where "
176                   + "ordem <= " + ordemTaticaAtual + " and codigoEstrategia = " + codigoEstrategia + ";");
177                 int tempoAtual = 0;
178                 if(resultSet.next()){
179                   tempoAtual = resultSet.getInt(1);
180                 }
181                 this.close();
182 
183                 int acrescimoHora = tempoAtual + Integer.parseInt(minutoInicial);
184                 int s = acrescimoHora / 60;
185 
186                 //Verifica o tempo para encerrar as taticas da sessao de ensino
187                 if(Integer.parseInt(horaInicial) == horaAtual){
188                   if(tempoAtual + Integer.parseInt(minutoInicial) == minutoAtual){
189                     this.executeUpdate("UPDATE estrategias_taticas SET finalizada = 1 WHERE "
190                     + "codigoEstrategia=" + codigoEstrategia + " AND ordem = " + ordemTaticaAtual + ";");
191                   }
192                   this.close();
193                 }else{
```

```
194                 //Encerra tatica anterior e atualiza o banco para a tatica atual
195                 if(Integer.parseInt(horaInicial) + (1 * s) == horaAtual){
196                   int acrescimo = tempoAtual + Integer.parseInt(minutoInicial);
197                   acrescimo = acrescimo - 60 * s;
198                   if(acrescimo == minutoAtual){
199                     this.executeUpdate("UPDATE estrategias_taticas SET finalizada = 1 WHERE "
200                       + "codigoEstrategia = " + codigoEstrategia + " AND ordem = " + ordemTaticaAtual
201                       + ";");
202                   }
203                   this.close();
204                 }
205               }
206
207               //Finaliza a sessao de ensino se todas as suas taticas estiverem finalizadas
208               resultSet = this.executeQuery("SELECT estrategias_taticas.finalizada , "
209                 + "sessao_ensino_estrategia.codigoEstrategia FROM estrategias_taticas , "
210                 + "sessao_ensino_estrategia WHERE estrategias_taticas.codigoEstrategia = "
211                 + "sessao_ensino_estrategia.codigoEstrategia and "
212                 + "sessao_ensino_estrategia.codigoSessao = " + sessao.getCodigo() + ";");
213               int teste = 0;
214               while(resultSet.next()){
215                 int finalizada = resultSet.getInt("finalizada");
216                 if(finalizada == 0){
217                   teste = 1;
218                   break;
219                 }
220               }
221               this.close();
222               if(teste == 0){
223                 this.executeUpdate("UPDATE sessao_ensino SET terminada = 1 WHERE codigo = "
224                   + sessao.getCodigo());
225               }
226             }
227           }
228         }
229       }
230     }catch (Exception e) {
231       System.out.println(e.getMessage());
232     }
233   }
234
235   //Funcao que encapsula os procedimentos para o agente se cadastrar nas paginas amarelas
236   public boolean cadastroNasPaginasAmarelas(String nome){
237     DFAgentDescription descritorDeServicosDeUmAgente = new DFAgentDescription();
238     ServiceDescription descricaoDeUmServico = new ServiceDescription();
239     descricaoDeUmServico.setName(nome);
240     descricaoDeUmServico.setType(nome);
241     descritorDeServicosDeUmAgente.addServices(descricaoDeUmServico);
242     try{
243       DFService.register(this,descritorDeServicosDeUmAgente);
244       return true;
245     }catch(FIPAException e){
246       return false;
247     }
248   }
249
250   //Funcao que resgata enderecos das paginas amarelas
251   protected AID[] getAIDs(String nome){
252     DFAgentDescription descritorDeServicosDeUmAgente = new DFAgentDescription();
253     ServiceDescription descricaoDeUmServico = new ServiceDescription();
254     descricaoDeUmServico.setName(nome);
255     descricaoDeUmServico.setType(nome);
256     descritorDeServicosDeUmAgente.addServices(descricaoDeUmServico);
257     try{
258       DFAgentDescription results[] = DFService.search(this,descritorDeServicosDeUmAgente);
259       AID[] aid = new AID[results.length];
260       for (int i = 0; i < results.length; i++) {
261         aid[i] = results[i].getName();
262       }
263       return aid;
264     }catch(FIPAException e){
265       return null;
266     }
267   }
268 }
```

D.2 Agente Executor de Sessão Mensagem

O agente executor de sessão mensagem (Código D.2) tem a função de monitoração do agente executor de sessão.

Código D.2: Implementação do Agente Executor de Sessão Mensagem

```
1  package agentes;
2
3  import jade.core.behaviours.TickerBehaviour;
4  import jade.lang.acl.ACLMessage;
5
6  // Classe de comportamento que estende o comportamento TickerBehaviour
7  public class AgenteExecutorSessaoRecebedorMsg extends
8  TickerBehaviour{
9
10   // Seta o tempo de comunicacao com os outros agentes (em segundos)
11   private static int TEMPO_ENTRE_INVESTIDAS = 1000 * 10;
12   private AgenteExecutorSessao agenteExecutorSessao;
13
14   // Metodo construtor que incializa um agente com seu tempo de çãcomunicao com os outros agentes
15   public AgenteExecutorSessaoRecebedorMsg(AgenteExecutorSessao agenteExecutorSessao){
16     super(agenteExecutorSessao, TEMPO_ENTRE_INVESTIDAS);
17     this.agenteExecutorSessao = agenteExecutorSessao;
18   }
19
20   // Chama o metodo para tratar a mensagem
21   public void onTick(){
22     ACLMessage message = this.agenteExecutorSessao.receive();
23     if (message != null){
24       this.agenteExecutorSessao.tratarMensagem(message);
25     }
26   }
27 }
```

Apêndice E

Especificação dos Agentes do TUTA

A especificação Orientada a Agentes do TUTA (Silva 2000) é apresentada nas seções E.1 e E.2.

E.1 Fase de Análise

E.1.1 Modelo de Papéis

Identifica os principais papéis do sistema que possuem dois tipos de atributos: As permissões/direitos associados ao papel e suas responsabilidades, que podem ser vitais e de segurança.

O modelo de papéis é composto por um conjunto de esquemas de papéis, um para cada papel, representados nas Tabelas E.1 a E.8, mostradas abaixo.

PAPEL: Gerente de Curso

Esquema do Papel: Gerente de Curso
Descrição: Pessoa responsável pela gerência dos dados e das sessões de um curso; dos professores que podem utilizar o framework e ocorrência de sessões **Protocolos:** *GerenciarRegistroProfessores, GerenciarCurso, GerenciarSessõesCurso, NotificarExistênciaSessão* **Permissões:** Ler dados do professor // ou dados do curso // ou dados das sessões de um curso Alterar dados do professor // ou dados do curso // ou dados das sessões de um curso **Responsabilidades** **Liveness:** *Gerente de Curso = GerenciarRegistroProfessor | GerenciarCurso | GerenciarSessõesCurso | NotificarExistênciaSessão* **Segurança:** Assegurar a consistência e integridade dos dados

Tabela E.1: Esquema do Papel Gerente do Curso

PAPEL: Aluno

Esquema do Papel: Aluno
Descrição: Participa de cursos sobre um determinado domínio **Protocolos:** *ParticiparAtividadesAluno* **Permissões:** **Responsabilidades** **Liveness:** *Aluno = ParticiparAtividadesAluno*

Tabela E.2: Esquema do Papel Aluno

PAPEL: Gerente de Entidades Didáticas

Esquema do Papel: Gerente de Entidades Didáticas
Descrição: Responsável pelo gerenciamento das entidades didáticas para a execução das estratégias didáticas em uma sessão de um determinado curso **Protocolos:** *GerenciarEntidadesDidáticas* **Permissões:** Ler Entidades Didáticas e Alterar Entidades Didáticas **Responsabilidades** **Liveness:** *Gerente de Entidades Didáticas = RecuperarEntidadesDidáticas* **Segurança:** Assegurar a consistência e integridade dos dados

Tabela E.3: Esquema do Papel Gerente de Entidades Didáticas

PAPEL: Gerente de Interface da Especificação do Curso

Esquema do Papel: Gerente de Interface da Especificação do Curso
Descrição: Pessoa responsável pelo gerenciamento dos alunos e do professor necessários para a execução de um curso **Protocolos:** *GerenciarCurso* **Permissões:** **Responsabilidades** **Liveness:** *Gerente de Interface da Especificação do Curso = GerenciarCurso*

Tabela E.4: Esquema do Papel Gerente de Interface da Especificação do Curso

PAPEL: Professor

Esquema do Papel: Professor
Descrição: Pessoa responsável pelas atividades dos cursos e execução de sessões **Protocolos:** *DefinirCurso, ParticiparExecSessão* **Permissões:** **Responsabilidades** **Liveness:** *Professor = ParticiparAtividadesProfessor* ParticiparAtividadesProfessor = DefinirCurso | ParticiparExecSessão

Tabela E.5: Esquema do Papel Professor

PAPEL: Gerente de Informação

Esquema do Papel: Gerente de Informação
Descrição: Pessoa responsável pelo gerenciamento das informações enviadas e recebidas por um aluno ou por um professor **Protocolos:** *GerenciarInformaçõesEnviadasParaAluno, GerenciarInformaçõesRecebidasPeloAluno, GerenciarInformaçõesEnviadasParaProfessor, GerenciarInformaçõesRecebidasPeloProfessor* **Permissões:** **Responsabilidades** **Liveness:** *Gerente de Informação = GerenciarInformaçõesEnviadasParaAluno| GerenciarInformaçõesRecebidasPeloAluno| GerenciarInformaçõesEnviadasParaProfessor| GerenciarInformaçõesRecebidasPeloProfessor*

Tabela E.6: Esquema do Papel Gerente de Informação

PAPEL: Administrador do Sistema

Esquema do Papel: Administrador do Sistema
Descrição: Pessoa responsável pela autorização e manutenção dos alunos e professores no sistema **Protocolos:** *RealizarAtividadesAdministrador* **Permissões:** **Responsabilidades** **Liveness:** *Administrador do Sistema = RealizarAtividadesAdministrador*

Tabela E.7: Esquema do Papel Administrador do Sistema

PAPEL: Gerente de Estratégias Didáticas

Esquema do Papel: Gerente de Estratégias Didáticas
Descrição: Pessoa responsável pelo gerenciamento das estratégias didáticas necessárias para a execução de uma sessão de um determinado curso **Protocolos:** *GerenciarEstratégiasDidáticas* **Permissões:** Ler Estratégias Didáticas Alterar Estratégias Didáticas **Responsabilidades** **Liveness:** *Gerente de Estratégias Didáticas = RecuperarEstratégiasDidáticas* **Segurança:** Assegurar a consistência e integridade dos dados

Tabela E.8: Esquema do Papel Gerente de Estratégias Didáticas

E.2 Fase de Projeto

E.2.1 Modelo de Agentes

O modelo de agentes consiste em documentar os vários tipos de agentes utilizados no TUTA e suas instâncias em tempo de execução.

O modelo de agentes é mostrado na Figura E.1.

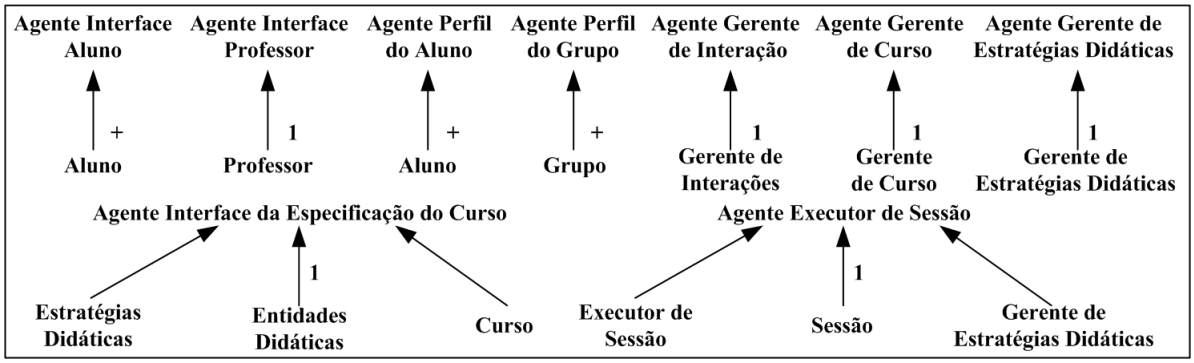


Figura E.1: Modelo de Agentes do TUTA

E.2.2 Modelo de Comunicação

O modelo de comunicação define as ligações de comunicação existentes entre os tipos de agentes do sistema, mas eles não definem as mensagens que são enviadas ou recebidas, indicando apenas os possíveis caminhos de comunicação existentes entre os tipos de agentes. O modelo de comunicação do framework proposto é mostrado na Figura E.2.

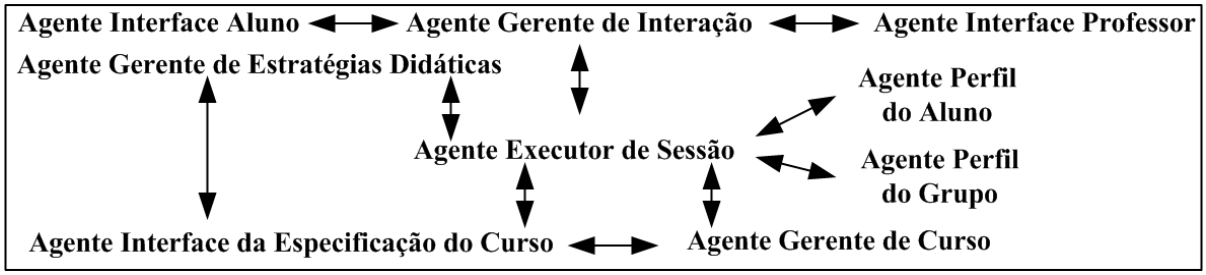


Figura E.2: Modelo de Comunicação do TUTA

Referências Bibliográficas

AgentBuilder (2006), 'Agentbuilder documentation', *Reticular Systems, external documentation, Disponível em: http://www.agentbuilder.com/* .

Appleton, B. (2006), 'Patterns of software: Essential concepts and terminology', *http://www.bradapp.net/docs/patterns-intro.html* .

Axmark, D., Larsson, A. & Widenius, M. (2006), 'Mysql', *http://mysql.com/* .

Bachmann, F. (2000), 'Technical concepts of component-based software engineering', *Relatório Técnico CMU/SEI-2000-TR-008. Software Engineering Institute, Carnegie Mellon* **2**.

Barr, A. & Feigenbaum, E. A. (1982), 'The handbook of artificial inteligence', *Los Altos, Califórnia: Kaufmann* .

Brenner, W., Zarnekov, R. & Witting, H. (1998), 'Intelligent software agents: Foundations and applications', *Berlim: Springer* .

Brown, A. W. & Wallnau, K. C. (1998), 'The current state of cbse', *IEEE Software, Setembro* **15**(5), 37–46.

Campbell, C. (1997), 'How to develop a professional portfolio - a manual for teachers', *Allyn & Bacon, USA* .

Campos, G. (2000), 'Pensando a educação a distância. timaster. rio de janeiro. seção escola internet - formação e treinamento on line', *http://www.timaster.com.br* .

CCEAD (2006), 'Coordenação central de educação a distância.', *PUC, Rio de Janeiro. Disponível em: http://www.ccead.puc-rio.br/* .

Costa, E. B. (1997), 'Um modelo de ambiente interativo de aprendizagem baseado numa arquitetura multiagente', *Tese de Doutorado em Processamento da Informação, CPGEE/UFPB* .

Costa, R. M. E. M. & Werneck, V. M. B. (1996), *Tutores Inteligentes*, COPPE/UFRJ, Rio de Janeiro.

Ellis, J. & Coppola, C. (2006), 'Sakai framework', *http://www.sakaiproject.org/* .

Eportconsortium (2006), 'Electronic portfolio white paper', *http://eportconsortium.org* .

Fayad, M. E. & Schmidt, D. C. (1997), *Object-Oriented Application Frameworks*, Communications of the ACM.

Fayad, M. E., Schmidt, D. C. & Johnson, R. E. (1999), *Building Application Frameworks: Object-Oriented Foundations of Framework Design*, John Wiley & Sons.

Fields, D. K. & Kolb, M. A. (2000), 'Desenvolvendo na web com javaserver pages', *Rio de Janeiro: Ciência Moderna* .

FIPA (2006), 'The foundation for intelligent physical agents', *Gênova, Suíça. Disponível em: http://www.fipa.org/* .

Fontoura, M., Pree, W. & Rumpe, B. (2000), 'Uml-f: A modeling language for object-oriented frameworks', *14th European Conference on Object Oriented Programming (ECOOP 2000), Springer, 63-82, Cannes, France* .

Franklin, S. & Graesser, A. (1996), 'Is it an agent, or just a program?: A taxonomy for autonomous agents', *In Springer-Verlag, ECAI 96 Third International Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III* pp. 21–36.

Frozza, R. (2000), 'Uma arquitetura para acompanhamento pedagógico em ensino a distância', *XI Simpósio Brasileiro de Informática na Educação - SBIE – Maceió, Alagoas* .

Furtado, M. E. S. (1993), 'Uma metodologia para projeto de banco de dados temporal orientado a objetos', *Dissertação de Mestrado, Universidade Federal da Paraíba – COPIN/DSC, Campina Grande, Paraíba, Brasil* .

Gamma, E., Helm, R., Jonhson, R. & Vlissides, J. (1995), 'Design patterns: elements of reusable object-oriented software', *Addison-Wesley* .

Giraffa, L. M. M. & Viccari, R. M. (2003), *Fundamentos de Sistemas Tutores Inteligentes. Sociedades artificiais: a nova fronteira da inteligência nas máquinas*, Porto Alegre: Bookman, Artmed Editora.

Hart, D. (1994), *Authentic Assessment - A Handbook for Educations*, Addison Wesley, USA.

Hernández-Domínguez, A. (1997), 'Specification and implementation of an adaptable virtual class', *In: ED-MEDIA 97, World Conferences on Educational Multimedia and Hypermedia on Educational Telecommunications, Calgary, Canada* .

Jacobson, I., Griss, M. & Johnson, P. (1997), *Software Reuse-architecture process and organization for business success*, Addison Wesley, New York: Springer.

JADE (2006), 'Java agent development framework', *http://jade.tilab.com/* .

Johnson, R. E. & Foote, B. (2001), 'Designing reusable classes', *Journal of Object Oriented Programming, JOOP* .

Lankes, A. M. D. (1995), 'Electronic portfolios: A new idea in assessment', *Clearinghouse on Information & Technology, Syracuse University, Center for Science and Technology, Syracuse, New York* .

Lightbody, P. & Carreira, J. (2005), 'Webwork in action', *Manning Publications Co., Greenwich* .

Madeiro, J. M., Medeiros, F. N. & Hernández-Domínguez, A. (2005), 'Um framework para portfólios eletrônicos', *XVI Simpósio Brasileiro de Informática na Educação, Workshop de Arquiteturas Pedagógicas para Suporte à Educação a Distância mediada pela Internet, Juiz de Fora/MG, Brasil* .

Medeiros, F. N., Medeiros, F. M. & Hernández-Domínguez, A. (2006), 'Fa_port: Um framework para sistemas portfólio-tutor utilizando agentes', *XVII Simpósio Brasileiro de Informática na Educação, Brasília/DF, Brasil* .

Mokhtari, K. & Yellin, D. (1996), 'Portfolio assessment in teacher education: impact on preservice teachers' knowledge and attitudes', *Journal of Teacher Education* .

Muller, J. P. (1998), *Architectures and applications of intelligent agents. In Knowledge Engineering Review*, Vol. 13, A survey.

Nascimento, D. M. C. (2002), 'Um sistema tutor acoplado a um portfolio eletrônico no contexto da educação a distância - portfólio-tutor', *Dissertação de Mestrado, Universidade Federal da Paraíba – COPIN/DSC, Campina Grande, Paraíba, Brasil* .

Nascimento, D. M. C., Hernández-Domínguez, A. & Schiel, U. (2002), 'Portfolio-tutor: Um tutor acoplado a um portfolio eletrônico no contexto da educação a distância', *XIII Simpósio Brasileiro de Informática na Educação, São Leopoldo, Rio Grande do Sul* .

Neto, F. J. S. L. (1998), 'Educação a distância: Regulamentação, condições de Êxito e perspectivas', *http://www.intelecto.net/* .

Nunes, I. B. (2000), 'Noções de educação a distância', *http://www.intelecto.net/* .

Nunes, J. (1999), 'Portfólio: Uma nova forma de encarar a avaliação', *Revista Noesis* .

Odell, J., Parunak, H., & Bauer, B. (2000), 'Extending uml for agents', *In Proc. of the 2nd Int. Bi-Conference Workshop on Agent-Oriented Information Systems, AOIS 00, Austin, USA* .

Oliveira, F. M. (1994), 'Critérios de equilibração para sistemas tutores inteligentes', *Tese de Doutorado - Porto Alegre: CPGCC/UFRGS* .

Russell, S. & Norving, P. (2004), *Inteligência Artificial*, Elsevier.

Santos, E. C. O. (2003), 'Sistemas java de educação a distância na internet', *Dissertação de Mestrado, Universidade Federal da Pernambuco – Pós-Graduação em Ciências da Computação, Recife, Pernambuco, Brasil* .

Self, J. (1988), 'Artificial intelligence and human learning', *London: Chapman Hall* .

Silva, A. S. (2000), 'Tuta - um tutor baseado em agentes no contexto do ensino à distância', *Dissertação de Mestrado, Universidade Federal da Paraíba – COPIN/DSC, Campina Grande, Paraíba, Brasil* .

Silva, E. P. (2002), 'Desenvolvimento de um portfolio eletrônico para web baseado nos princípios de banco de dados ativo', *Trabalho de Conclusão de Curso, Universidade Federal de Alagoas - UFAL – Departamento de Tecnologia da Informação - TCI, Maceió, Alagoas, Brasil* .

Silva, L. A. M. (2003), 'Estudo e desenvolvimento de sistemas multiagentes usando jade: Java agent development framework', *Trabalho de Conclusão de Curso, Universidade de Fortaleza - UNIFOR, Fortaleza, Ceará* .

Silveira, R. A. (1995), 'Inteligência artificial em educação: um modelo em sistema tutorial inteligente para microcomputadores', *Dissertação de Mestrado - Porto Alegre - FACED/PUCRS* .

Sistêlos, A. J. C. M. (1999), 'Um ambiente computacional de apoio ao método de avaliação autêntica: Projeto poeta (portfólio eletrônico temporal e ativo)', *Dissertação de Mestrado, Universidade Federal da Paraíba – COPIN/DSC, Campina Grande, Paraíba, Brasil* .

Somerville, I. (2003), 'Engenharia de software', *Addison-Wesley, 6 ed* .

Souto, M. A. (2000), 'Modelo de ensino adaptativo na internet basedo em estilos cognitivos de aprendizagem', *XI Simpósio Brasileiro de Informática na Educação, Maceió, Alagoas* .

Souza, E. C. B. M. (2000), 'Avaliação instrucional: Uma abordagem prática', *Instituto de Educação Superior de Brasília, Universidade de Brasília. Brasília* **2**.

Szyperski, C. (1998), 'Component software beyond object-oriented programming', *Addison-Wesley* .

Tecuci, G. (1998), 'Building intelligent agents, an apprenticeship multistrategy learning theory methodology tool and case studies', *Academic Press: San Diego* p. 315.

Valente, J. A. (1998), 'Formação de profissionais na área de informática em educação. in: Valente (org.) computadores e conhecimento: repensando a educação', *Campinas: UNICAMP/NIED* pp. 139–164.

Viccari, R. M. (1990), 'Um tutor inteligente para a programação em lógica-idealização, projeto e desenvolvimento', *Tese de Doutorado - Universidade de Coimbra* .

Wenger, E. (1987), 'Artificial intelligence and tutoring systems: Computacional and cognitive approaches to the communication of knowledge', *Morgan Kaufmann Publishers, Inc. Califórnia, USA* .

Winograd, P., Paris, S. & Bridge, C. (1991), 'Improving the assessment of literacy', *The reading Teacher* pp. 108–116.

Wooldrige, M. (1999), 'Intelligent agents. in: Multiagent systems', *G. Weiss, The MIT Press* .

Wooldrige, M. & Jennings, N. R. (1995), 'Agent theories, architetures e languages: a survey intelligent agents', *Berlin: Springer - Verlag* .

Wooldrige, M. & Jennings, N. R. (1998), *Applications of Intelligent Agents. In: JENNINGS, N. R., WOOLDRIGE, M. (Eds.) Agent Technology: foundations, aplications and markets*, Addison Wesley, New York: Springer.

Wooldrige, M., Jennings, N. R. & Kinny, D. (1999), *A Methodology for Agent-Oriented Analysis and Design. In: THIRD ANNUAL CONFERENCE ON AUTONOMOUS AGENTS*, Proceedings, Seattle - USA.

Wooldrige, M., Jennings, N. R. & Kinny, D. (2000), 'The gaia methodology for agent-oriented analysis and design', *Kluwer Academic Publishers, Boston* .

Yacoub, S. & Ammar, H. (2003), 'Pattern oriented analysis and design', *AddisonWesley* .

Yazdani, M. (1987), 'Intelligent tutoring systems: An overview. artificial intelligence and education', *Ablex* .

Zeus (2006), 'Zeus documentation', *http://more.btexact.com/projects/agents.htm* .